Design of Quality-Based Composite Web Services

F. De Paoli, G. Lulli, and A. Maurino

Università degli Studi di Milano Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
{depaoli, lulli, maurino}@disco.unimib.it

Abstract. One of the key factors for successful SOC-based systems is the ability to assure the achievement of Quality of Services. The knowledge and the enforcement of the Quality of Services allows for the definition of agreements that are the basis for any business process. In this paper we discuss a method for the evaluation of qualities associated with services. This method is based on a set of quality evaluation rules that state the relations between Web services quality dimensions and process structure. The method is part of a design methodology that addresses quality issues along the service life-cycle. A case study in the e-placement field is presented to illustrate a practical use of the approach.

1 Introduction

Service Oriented Architecture (SOA) is rapidly evolving to become a business integration architecture that supports the dynamic composition of Web services to enact business processes. For the enactment of business processes the conditions under which certain features are provided are as important as the features themselves. Therefore, models and tools to address the non-functional aspects of a service, such as privacy, security, exception handling, performance, and so on, should be defined for effective service composition.

Traditional approaches to service composition focus on service operations, while there is little attention dedicated to the possibility of composing services with, for instance, different security models. A service is usually implemented by a Web service described by means of a WSDL (Web Service Description Language) interface that includes information on operation signatures and access points. Currently a major effort is devoted to the definition of descriptions that deal with issues beyond WSDL. SLA (Service Level Agreement) [1] and WS-Policy [2] are examples of descriptions of the mutual understandings and expectations of both the service provider and the service requester. These contracts regulate and define matters such as contents, price, delivery process, acceptance and quality criteria, penalties and so on. Semantic descriptions are related to the definition of what a service provides in terms of functionality and how the provided services are supplied in terms of behavior and non-functional properties. Emerging proposals are OWL-S [3] and WSMO [4]. Their aim is to define architectural abstractions and description languages that supply machine

A. Dan and W. Lamersdorf (Eds.): ICSOC 2006, LNCS 4294, pp. 153–164, 2006.

interpretable semantics to facilitate dynamic interoperability, composability, and substitutability.

In this paper, we propose a method for Web service composition to address quality evaluation in composed services. The work is part of an effort in the development of a quality-based design methodology that addresses the non-functional requirements of (composite) services [5]. According to the model-driven development approach [17], the goal is to deliver an enhanced platform independent model (PIM, in Model Driven Architecture terminology) that includes non-functional descriptions.

The methodology includes a customization phase that takes into account the actual profiles of target end-users and service providers. Technical characteristics and user profiles are evaluated with manifold objectives: select a set of candidate services that fulfill the requirements, tune-up the service specification and/or identify the target users for the service. The designer, with the support of the rules discussed in this paper, composes and evaluates the qualities of candidate services that satisfy classes of requests so to be able to support future agreements on the base of specific user requirements. The result is a set of (composite) services that differs on quality attributes, each one depending on the specific set of requirements and composing services.

A composite Web service is composed of a number of Web services that are invoked according to a given execution flow described in a language such as WsBPEL. Starting from the distinction proposed in WSDL 1.1 between abstract and concrete service, we introduce the term *abstract service* to represent a service which is only the description of its interface, and the term *concrete service* to refer to fully implemented and described Web services. It is important to underline that only concrete services can be invoked. According with this taxonomy, the proposed quality-based composition model refers to abstract processes that is composed of abstract services only. In our vision each abstract service represents a prototype of a set of concrete services that share the same functional description, but provide different implementations and QoSs. The substitution of each abstract service with a concrete Web service, provides a concrete composite Web service that can be invoked. We identify this activity with the term *concretization*. It is worth noting that successful concretization activities are influenced, at the same time, by the desired quality dimensions and by the process execution flows. To address the issue, we propose a set of metrics (called *quality evaluation rules*) that can be used to evaluate specific QoS dimensions according to specific process language constructs (such as sequence, parallel, switch and loop).

The paper is organized as follow: in Section 2 we present the mathematical formulation of the concretization activity, then Section 3 presents a set of quality evaluation rules that are used according with specific quality dimensions and process language structures. Section 4 discusses the method in a case study that is under development as part of the IST project SEEMP. Section 5 is dedicated to related works and, finally, Section 6 draws some conclusions and outlines future works.

2 Mathematical Model

The structure of a composite abstract Web service can be described as a direct graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$; where $\mathcal{N}$ is the set of nodes representing abstract services, and $\mathcal{E}$ is the set of arcs that define the execution flow. The abstract process is composed of smaller and homogeneous parts according to their structure. Each part can be classified as a simple service or a group of services, which are structured to form a sequence, a parallel, a switch or a loop. According to [6], a process graph can be collapsed into a single node (i.e., single composite service) by substituting each process-language structure with a Web service that exposes the same functional and QoS description of the original set.

Illustrating with an example, let us consider a generic quality dimension q_i and the Ws-BPEL process in Fig.1(a). The first transformation considers the parallel execution of OperationB, OperationC, and OperationD. The three Web services can be substituted by a new one called OperationBCD (Fig.1(b)) whose quality value is the one obtained by applying the quality evaluation rule (see next) to the three Web services. Then the transformation of the switch structure generates another Web service, OperationFG (Fig.1(c)). Finally, by means of another quality evaluation rule, we obtain the quality dimension q_i of the composite service (Fig.1(d)).

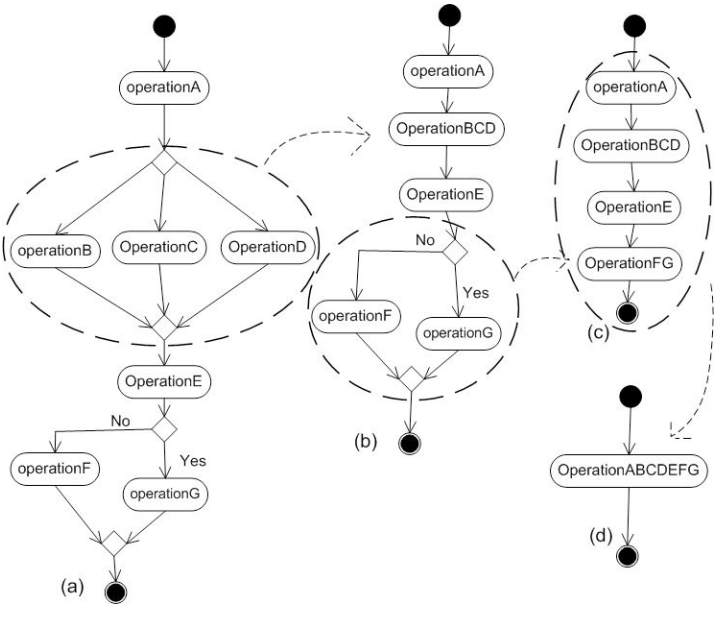

Fig. 1. A stepwise graph transformation

The notation of the model adopted in the sequel is the following:

$\mathcal{S} \equiv$ the set of concrete services;
$\mathcal{A} \equiv$ the set of abstract services;
$\mathcal{C} \equiv$ the set of clients;
$\mathcal{S}_a \subseteq \mathcal{S} \equiv$ the set of concrete services which may implement the abstract one $a \in A$;
$w_c \equiv$ weights assigned by customer c to qualities (vector);
$f_s \equiv$ qualities of the concrete service $s \in \mathcal{S}$ (vector);
$r_c \equiv$ qualities of client $c \in \mathcal{C}$ (vector).

The decision variables are:

$$x_{ij}^c = \begin{cases} 1 & \text{if the abstract service } j \text{ for customer } c \text{ is concretized by the concrete service } i; \\ 0 & \text{otherwise.} \end{cases}$$

The mathematical formulation of the concretization activity is the following:

$$Min \sum_{c \in \mathcal{C}} w_c \cdot (r_c - \theta_c)^+ \tag{1}$$

where θ_c are auxiliary variables representing the vector of the value of the features guaranteed to the client c. $(\cdot)^+$ denotes the positive part of the argument. With respect to formula (1), a concrete service must be selected for each abstract one; the mathematical representation of this constraint is:

$$\sum_{i \in \mathcal{S}_j} x_{ij}^c \geq 1 \qquad \forall c \in \mathcal{C}, \ \ \forall j \in \mathcal{A}. \tag{2}$$

3 Quality Evaluation Rules

In order to use the formula (1) the right quality metric for each quality dimension need to be defined. The evaluation of QoS in composite Web service is not only related to the specific QoS selected, since the QoS value offered to end users refers also to the composition process. Three features have to be considered in order to evaluate a quality dimension in composite Web services: (i) a quality metric for the quality dimension measured for each simple concrete Web services, (ii) the process-language structure in which Web services are inserted, and (iii) the identification of which Web services have to provide the requested quality. Concerning the first feature, the method assumes that each concrete Web service exposes a value for the considered QoS. To address the second feature, we propose three different types of quality evaluation rules to represent the most typical ways in which QoSs can be evaluated according with specific process language primitives. Finally, according with the specific application domain, quality dimensions can be classified as *local*, if the QoS is provided by a subset of concrete Web services or *global*, if all Web services contribute in the definition of the quality dimension.

The three quality evaluation rules are explained in the next sub sections.

Additive Quality. This quality evaluation rule can be used by designers to evaluate qualities exposing an additive behavior, in a composite Web service. An example of additive QoS is the completion time of a Web service. For a concrete composite service to compute the value of an additive quality, the QoS values of all the service's components are added (in formula, $\sum_{k \in \mathcal{K}} t_c^k$). According to the composition structure, for each component k, the additive quality is computed by one of the following sets of constraints:

$$\text{Sequence } (\mathcal{S}eq_k) \qquad t_c^k = \sum_{j \in \mathcal{S}eq_k, i \in \mathcal{S}_j} t_{ij} \cdot x_{ij}^c \tag{3}$$

$$\text{Parallel } (\mathcal{P}_k) \qquad t_c^k \geq \sum_{i \in \mathcal{S}_j} t_{ij} \cdot x_{ij}^c \qquad \forall j \in \mathcal{P}_k. \tag{4}$$

$$\text{Switch } (\mathcal{S}w_k) \qquad t_c^k \geq t_{ij} \cdot x_{ij}^c \qquad \forall j \in \mathcal{S}w_k, \ \ \forall i \in \mathcal{S}_j. \tag{5}$$

$$\text{Loop } (\mathcal{L}_k) \qquad t_c^k = l_k \cdot \sum_{j \in \mathcal{L}_k, i \in \mathcal{S}_j} t_{ij} \cdot x_{ij}^c \tag{6}$$

where t_{ij} is the completion time of the concrete service i in executing the abstract service j, and l_k is the expected number of loops executed.

For instance, referring to the example depicted in Figure 1, the service completion time for customer c is given by

$$t_c^A + t_c^{BCD} + t_c^E + t_c^{FG}$$

where t_c^{BCD} is the completion time of the Web services *OperationBCD*, which is the result of a parallel structure. In this case, the completion time corresponds to the longest completion time of *OperationB, OperationC* and *OperationD*. In fact, according with the Ws-BPEL specification, it is not possible, to proceed in the process execution before *OperationsB, OperationsC* and *OperationsD* have been completed. In the formula, t_c^{BCD} is given by the following constraints:

$$t_c^{BCD} \geq \sum_{i \in \mathcal{S}_B} t_{iB} \cdot x_{iB}^c$$

$$t_c^{BCD} \geq \sum_{i \in \mathcal{S}_C} t_{iC} \cdot x_{iC}^c$$

$$t_c^{BCD} \geq \sum_{i \in \mathcal{S}_D} t_{iD} \cdot x_{iD}^c$$

Full Additive Quality. This quality evaluation rule is similar to the previous one. It can be applied to quality dimensions, such as the cost, that involve all concrete Web services. To compute the value of a full additive quality, the constraints given above are still valid, but the one for parallel composition that it needs to be re-formulated as follows:

$$\text{Parallel } (\mathcal{P}_k) \qquad C_c^k = \sum_{j \in \mathcal{P}_k} C_{ij} \cdot x_{ij}^c \tag{7}$$

In this case, all QoS values exposed by concrete services involved in the parallel execution are considered.

Non-additive Quality. It represents quality dimensions that involve a subset of Web services that affect the whole composite service. For example, if a composite Web service is requested to support multichannel provisioning, it is mandatory that at least one of the Web services is able to interact with the users in a multichannel way. Another example is security. In this case, every component has to contribute to the global security, which is represented here as a boolean value (i.e., $0, 1$). The following set of constraints captures this case:

$$\theta_c^{sec} \geq f_s^{sec} \cdot x_{si}^c \qquad \forall s \in \mathcal{S}^i, \;\; \forall c \in \mathcal{C}, \;\; , \forall i \in \mathcal{N} \setminus \{\mathcal{S}w\}. \tag{8}$$

The sets of constraints presented so far refer to global features of the Web service. On the other hand, some of the requirements can be specific of a service, i.e., local requirements. For instance, by fixing the value of a decision variable $x_{ij} = 1$, an abstract service j is executed by the concrete service i. The following constraint declares that specific features are met by the concrete service s concretizing the abstract service j:

$$\sum_{s \in \mathcal{S}_j} x_{sj}^c \cdot f_s \geq \bar{f}_j \qquad \forall j \in \mathcal{I}, \;\; \forall c \in \mathcal{C}. \tag{9}$$

4 The Case Study

In this section, an example of Web service composition is discussed. The case study is derived from a larger case study developed within the IST SEEMP project [7], whose goal is to develop a European ePlacement market place.

Public Employment Services (PESes) are becoming more and more important for Public administrations since social implications on sustainability, workforce mobility and equal opportunities are of strategic importance for any central or local government. In our case, we are interested in the development of a service to search for jobs in European countries (e.g., Italy, Ireland and Great Britain). Such a service, *FindJob*, has the following functionalities:

- search for job vacancies in a specified country;
- glue together and rank retrieved job vacancies;

– translate vacancies in different languages; and
– notify end-users of results via different communication channels.

Note that these are *added-value* features that go beyond the simple functionalities to enrich the service in order to fulfill advanced customer requirements better. Users are requested to specify the kind of job he/she is seeking, along with languages spoken and personal contact information (email addresses, mobile phone numbers ...). Language and the job being searched are parameters of the searching job functionality. Contact information is used to identify and use the proper notification channel.

Besides the above functionalities, *FindJob* is requested to meet a set of quality requirements. In particular, we are interested in:

– the duration time to perform the process;
– the cost of the composite service invocation;
– the number of distribution channels on which the answer can be delivered.

Figure 2 shows a possible UML activity diagram representing the Ws-BPEL process including six different abstract Web services. Three kinds of information are supplied: the native language, the job description (MyRequest in the figure) and the communication preferences (MyData).

The service starts with the orchestration engine that registers the user by sending his/her contact information to a *Notification* service that will be in charge

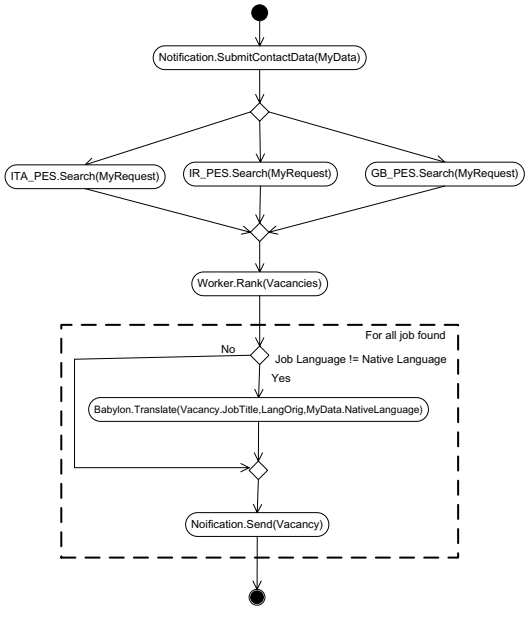

Fig. 2. The case study

of delivering the results. Such a registration might end-up with a failure, which is not discussed here to keep the example simple. Then the orchestration engine invokes the *search* operation on three different PESes, one for each country. These services discover job vacancies according to job descriptions. Discovered job vacancies are inputs for a Web service that sorts the results according to certain criteria (the discussion of which is not of interest in this context). Then for each job vacancy, the process checks the language in which it is written, and, when necessary, invokes a translation Web service. At the end of this operation, the engine invokes the *send* operation of the *Notification* Web service by passing the job vacancy.

Table 1. List of concrete services

ID	Virtual Service	Type	Cost	Answers	Time	Accuracy
1	Job Search	Regular	33	25	100	-
2	〃	Premium	100	60	20	-
3	Translation	High Accuracy	100	-	15	1.0
4	〃	Low Accuracy	50	-	20	0.0
5	Notification	2 channels	20	-	1	-
6	〃	3 channels	50	-	1	-

In Table 1, the available concrete services that implement the abstract services and their qualities are listed. For each abstract service, we consider two possible concrete ones, a cheaper one with a lower level of qualities and a more expensive one with a higher level of qualities. The evaluation of each quality is handled using sets of constraints presented in Section 3.

For instance, denoting *A, B* and *C* the Job Search services in Italy, Ireland and Great Britain respectively, we use the following set of constraints to evaluate the service completion time:

$$t^{JS} \geq t_{A,1} \cdot x_{A,1} + t_{A,2} \cdot x_{A,2}$$
$$t^{JS} \geq t_{B,1} \cdot x_{B,1} + t_{B,2} \cdot x_{B,2}$$
$$t^{JS} \geq t_{C,1} \cdot x_{C,1} + t_{C,2} \cdot x_{C,2}$$

$$t^{T} \geq t_{T,3} \cdot x_{T,3} + t_{T,4} \cdot x_{T,4}$$

$$t^{\mathcal{L}} = l \cdot (t_T + t^N)$$

$$t = t^{\mathcal{L}} + t^{JS} + t^N + t^R$$

where t^{JS}, t^T, t^R, t^N are the execution times of search, translation, rank and notification activities, respectively. $t^{\mathcal{L}}$ is the execution time of the loop which

depends on the number l of iterations. Finally, t is the completion time of the composite Web service. Similar constraints (see Section 3) should be added to the mathematical definition in order to evaluate the full set of qualities.

In this case study, we identified 8 possible processes. For instance, a process can be composed of the following concrete services: the 2-channel service for the Notification, the High-Accuracy service for the Translation and the Premium service for the Job Search (service pattern P-HA-2ch). In this case, by applying the computational formulas presented in Section 2, the cost and the time of the process are 420 and 2409 respectively. In Table 2, the global qualities for each process are reported.

Table 2. Computed qualities of the alternative processes

Service Pattern	Time	Cost	Channel	Accuracy
R-HA-2ch	1381	220	2	1
R-HA-3ch	1381	250	3	1
R-LA-2ch	1781	170	2	0
R-LA-3ch	1781	200	3	0
P-HA-2ch	2409	420	2	1
P-HA-3ch	2409	450	3	1
P-LA-2ch	3149	370	2	0
P-LA-3ch	3149	400	3	0

According to the weights assigned by each customer or class of customers to the qualities, the customer can be served by a specific process. For instance, a customer who is interested only in minimizing the cost will be served with the services R-LA-2ch (Regular Job Search service, Low-Accuracy Translation service and 2-channel Notification service), which deliver the cheapest process.

5 Related Works

The issue of quality of services is getting increasing consideration in the service-oriented literature (for example [8], and [9]); the focus, however, is often in modeling issues rather than in the design process. The problem of evaluating QoS dimensions in composite Web services is faced in [10], [11], and [12] where a set of composition rules is defined to evaluate the global value of a QoS dimension according to specific workflow patterns. However, the problem of joint design of services and their qualities is not addressed.

Concerning papers addressing QoS issues in design processes, [13] presents the ADD (Attibute-Drived Design) method, which is based on understanding the relationship between software qualities and the architectural mechanisms used to achieve these qualities. The lack of a quality model that can help the designer to identify and relate qualities is a drawback. Moreover, ADD is tailored for generic software development, without specific focus on SOA. Dealing

with web services, [14] presents a QoS model, addressing time, cost, and reliability dimensions. The model computes the quality of service for workflows automatically based on QoS attributes of an atomic task. This QoS model is then implemented on the top of the METEOR workflow system. [15] presents a fixed QoS model for the Self-serv model-driven design and an approach to select the optimal set of Web services according to QoS. The limited and fixed number of QoS dimensions considered reduces the possibility of adopting this approach in different application domains. Finally, QoS is considered in [16] to improve the outcome of web services discovery.

6 Conclusions

The non-functional properties, often referred to Quality of Services, are one of the most relevant issues in service-oriented architectures. In fact, the capability of describing, composing, evaluating and monitoring qualities associated with services is critical to the effective enactment of business processes.

This paper proposed and discussed a method to support the design processes to deliver (composite) services augmented with quality descriptions which can be exploited for service and agreement descriptions. The method helps in describing and evaluating generic qualities.

The scope of the mathematical model presented can be twofold. First, the model can be used to define a set of processes according to the desired qualities. The mathematical model suggests which concrete services have to be selected to implement the abstract services. The selection of concrete services depends on the classes of customers that will use the composite Web service and other requirements established by the Web service designer, such us budget constraints, service-level agreements, etc. Once the concrete services have been selected, many alternative processes can be identified.

Second, the model addresses the issue related to process selection. It supports the selection of the process to be supplied to each class of customers according to their quality requirements and the weights assigned to each quality by the customer.

The work presented here needs to be further developed to reach maturation, but we believe that the path is promising. We are studying the integration of the quality model presented in this paper with WSMoD [5] a methodology for the design of QoS-aware Web service. In particular we want to enrich the ontology approach adopted in WSMoD to facilitate both the understanding and the sharing of concepts. The aim is to let independent organizations, developers, and providers rely on common understandings, which is the basis for every future development. This understanding should go beyond the syntax barriers of XML/RDF dialects in order to enable semantic, and possibly automatic enactment of business processes. The association of evaluation rules with concepts (i.e. qualities) enhances the computing capabilities of the approach. Moreover, we are currently developing a visual tool which is implemented as part of the Eclipse development environment to support WSMoD. The capabilities of expressing and evaluating qualities will

allow the different players -business clients, developers and providers- to understand each others and build tools to support aspects other than design, such as discovery and selection of services, contracts definition and monitoring.

Acknowledgements

The work presented in this paper has been partially supported by the European IST project n. 27347 SEEMP - Single European Employment Market-Place and the Italian FIRB project RBNE05XYPW NeP4B - Networked Peers for Business.

References

1. Dan, A., Davis, D., Kearney, R., Keller, A., King, R.P., Kuebler, D., Ludwig, H., Polan, M., Spreitzer, M., Youssef, A.: Web services on demand: Wsla-driven automated management. IBM Systems Journal **43**(1) (2004) 136–158
2. VA: Web services policy framework (ws-policy). Technical report, BEA Systems Inc., International Business Machines Corporation, Microsoft Corporation, Inc., SAP AG, Sonic Software, and VeriSign Inc (2006)
3. Martin, D.L., Paolucci, M., McIlraith, S.A., Burstein, M.H., McDermott, D.V., McGuinness, D.L., Parsia, B., Payne, T.R., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K.P.: Bringing semantics to web services: The owl-s approach. In Cardoso, J., Sheth, A.P., eds.: SWSWPC. Volume 3387 of Lecture Notes in Computer Science., Springer (2004) 26–42
4. Lausen, H., Roman, D., Keller, U.: Web service modeling ontology (wsmo). Technical report, DERI (2004)
5. Comerio, M., De Paoli, F., Grega, S., Maurino, A., Batini, C.: Wsmod: a methodology for qos-based web services design. International Journal of Web Services Research (2007)
6. Jaeger, M.C., Rojec-Goldmann, G., Mühl, G.: Qos aggregation for web service composition using workflow patterns. [17] 149–159
7. VA: Single European Employment Market-Place. http://www.seemp.org/, (IST SEEMP project n. 27347)
8. Menascé, D.A.: Composing web services: A qos view. IEEE Internet Computing **8**(6) (2004) 88–90
9. Patil, A.A., Oundhakar, S.A., Sheth, A.P., Verma, K.: Meteor-s web service annotation framework. In Feldman, S.I., Uretsky, M., Najork, M., Wills, C.E., eds.: WWW, ACM (2004) 553–562
10. Raje, R.R., Bryant, B.R., Olson, A.M., Auguston, M., Burt, C.C.: A quality-of-service-based framework for creating distributed heterogeneous software components. Concurrency and Computation: Practice and Experience **14**(12) (2002) 1009–1034
11. Ulbrich, A., Weis, T., Geihs, K.: Qos mechanism composition at design-time and runtime. In: ICDCS Workshops, IEEE Computer Society (2003) 118–
12. Weis, T., Ulbrich, A., Geihs, K., Becker, C.: Quality of service in middleware and applications: A model-driven approach. [17] 160–171
13. Bachmann, F., Bass, L.J.: Introduction to the attribute driven design method. In: ICSE, IEEE Computer Society (2001) 745–746

14. Cardoso, J., Sheth, A.P., Miller, J.A., Arnold, J., Kochut, K.: Modeling quality of service for workflows and web service processes. J. Web Sem. **1**(3) (2004) 281–308
15. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. Software Eng. **30**(5) (2004) 311–327
16. Ran, S.: A framework for discovering web services with desired quality of services attributes. In Zhang, L.J., ed.: ICWS, CSREA Press (2003) 208–213
17. 8th International Enterprise Distributed Object Computing Conference (EDOC 2004), 20-24 September 2004, Monterey, California, USA, Proceedings. In: EDOC, IEEE Computer Society (2004)