Service interdependencies: insight into use cases for service composition

Witold Abramowicz, Agata Filipowska, Monika Kaczmarek, Tomasz Kaczmarek, Marek Kowalkiewicz, Wojciech Rutkowski, Karol Wieloch, Dominik Zyskowski

Department of Management Information Systems, the Poznan University of Economics, Poznan, Poland
{w.abramowicz, a.filipowska, m.kaczmarek, t.kaczmarek, m.kowalkiewicz, w.rutkowski, k.wieloch, d.zyskowski} @kie.ae.poznan.pl

Abstract. The paper analyses several most appealing use cases for Semantic Web services and their composition. They are considered from the perspective of service types, QoS parameters, semantic description and user preferences. We introduce different levels of service composition and discuss implications of the above.

Keywords: Web services, information services, service composition, use cases

1 Introduction

With the emergence of the idea of Semantic Web and Semantic Web Services a lot of different scenarios and use cases appeared, that were supposed to illustrate and justify their application in the real life. The primary purpose of introducing semantics into the world of Web services (WS) was to enable semantic-based discovery and composition. WS composition comes into play when a client request cannot be satisfied by one available service, but by an adequate combination of existing services. The autonomous agents should be able to discover services that would suit their goals, invoke them and execute series of tasks that required to this day human involvement. However, most of the scenarios defined to illustrate this vision fail a very basic test, namely applicability. This is not only a result of problems with semantics, interoperability or lack of features that numerous WS-standards try to address, but also many use cases for Semantic Web Services fail to take into account some general properties of services. This is particularly visible in scenarios where automatic service composition is considered.

Our motivation in this article is to clarify limitations of service composition, taking into account use cases already discussed in the literature. We rely on our experiences in several research projects that deal with semantic-enabled Web services. We also propose meta-requirements for use cases that include service composition at various levels. Additionally, we show interdependencies between different service types. The main focus of the paper is to present distinctions between Web services and real-world services, and show how these two worlds mix and interplay.

The paper is structured as follows. First the related work is briefly discussed. In the following section a number of use cases used in the research projects and some practical examples of service exploitation are presented. Next, the service types that influence service compositions are given. Finally, the conclusions and future directions of research are discussed.

2 Related work

Service composition may be defined as building more powerful and feature-rich functionality from simpler elements. Each simple element (a service) must have discrete, independent capabilities of its own, but can also be added to (composed with) other elements to create more complex solutions [9]. Service composition has a lot to offer. It has triggered a considerable number of research efforts and is currently one of the most hyped and addressed issues in the Service Oriented Computing and Service Oriented Architecture as the Web services standards have largely been designed to be composable. The interoperability was the key requirement, and interoperability triggers composability. Moreover, the web is a particularly interesting domain for service composition for several reasons. Firstly, increasing numbers of interesting services are moving online and the web is fast transforming from a collection of static pages to a provider of numerous useful services. Another reason is that Web services conform to the standard HTTP protocol which makes it (relatively) easier to integrate them into a common framework. Third, because the web has many independent service providers providing related services, there is an inherent need for composing complementary services provided by independent providers to achieve the end-user's needs. [12]

Of course the ultimate challenge is not only service composition but automatic service composition performed by autonomous agents. Additionally, automatic composition has a large potential in the workflow area (see [5]) as it enables dynamic reconfiguration of workflow, which is hardly achievable if workflows are defined and managed manually. Milanovic and Malek [4] propose four requirements that must be satisfied by service composition mechanism: connectivity, nonfunctional quality-of-service (QoS) properties, correctness, and scalability. Connectivity guarantees that WS can be composed in terms of input and output messages. Nonfunctional QoS properties denote parameters like timeliness, security and dependability. Correctness assures that composed service's properties are verified and the composition framework must scale appropriately.

In order to illustrate the applicability of the Web services technology and Semantic Web services composition a lot of different use cases were invented. They are analyzed at different angels. There are many works and initiatives that look at the use cases from the perspective of what requirements to Web services, Web services' architecture and languages used to describe them, may be identified (see e.g. http://www.daml.org/services/use-cases.html for a choice of use cases). There is a tremendous ongoing research effort in several EU-funded projects that deal with semantics for Web services (e.g. DIP, Knowledge Web, InfraWebs, SEKT, SWWS, ASG and Esperonto). Some of these projects aim at applying Semantic Web Services

to real-world scenarios. They deal with subjects such as semantic description of service capability, choreography, quality parameters, service discovery, invocation and composition. The main problem of such projects is to apply the results of scientific efforts in the real life. To handle this issue properly, special actions are undertaken. For example, in the ASG project [2] one work component is responsible for the preparation and further exploitation of use-cases. From the other side, there is another work component focused entirely on service composition matters. Their task is to define requirements to used services, prepare mechanisms capable of composing and creating all needed ontologies. However, to our best knowledge, there are no research initiatives looking at the use cases from the side as we present in this article.

3 Use cases discussion

In this section several of the scenarios that are considered when applying Semantic Web Services are discussed. The aim of this survey is to prepare ground for further analysis of common features of these use cases and comparison of different service types that take part in the scenarios. The scenarios were split into two groups. The first one includes scenarios that are tailored for using semantic descriptions or composing services and are the basis for various Web Services development projects (subsection A-F), like e.g. WSMO [14], ASG [2], USE-ME.GOV [16]. The second group consists of existing and running applications of WS in particular business areas.

A. Travel booking

The most popular use case for Web services deals with traveling [7]. In this scenario travel agent offers to book complete vacation packages. Airlines, hotels are providing Web services to query their offerings and perform reservations. This use case assumes that consistent ontology is used within the entire process. The other issue is services reliability and trustworthiness of parties engaged. Another travel based scenarios comes from WSMO group [14]. From the end-user point of view both scenarios give interfaces to external information systems. Both scenarios are based on a service (called Virtual Travel Agency and travel-agent service respectively) that behaves like some kind of integration platform with a fixed workflow.

B. Tourist attraction booking

The scenario presents a location based mobile service - the Attraction Booking Service (end service). It is part of a larger service platform in the ASG project, providing all kinds of tourist services for a mobile end user. In this scenario an end service customer (e.g. a tourist visiting a foreign city) wants to book a cultural event. Hence, the goal of the Attraction Booking Service is to provide the customer with information about attractions in the nearest surroundings of the current location. Additionally, the customer is able to perform certain actions based on this information (retrieve details, book, pay, get route description). First, the distributed platform that manages the services, and on the basis of the user request composes a flow of service specifications that can fulfill the goal. However, there is not only one service that implements the service specification – e.g. implementation of AttractionInformation service may be offered by three different service providers. So, the most suitable

service implementation to be contracted by negotiation to each service specification is selected. At the very end the composed service is executed [2].

C. Buddy scenario

Buddy scenario assumes that the user owns a mobile device. The scenario is based on keeping contacts with your buddies. It allows to get the availability status and location of the buddies, and to set up a group communication, based on instant messenger, SMS or voice. The scenario typically consists of service components, which are configured dynamically (e.g. configuration of your buddy list, getting information about activities, identification of friends in the vicinity and group communication). The development of the service has to be dynamic, based on service components from different providers. For example, we want to check who of our friends is in the vicinity. In order to fulfill our need, the platform integrating the services must take advantage of three services: finding phone number, finding localization and drawing a map on the screen of mobile device. There is a set of services in each category and the platform must dynamically select the best services to be executed in the workflow taking into account QoS criteria and other user criteria [2].

D. Healthcare services

This is a scenario taken from a medical domain. Imagine that we would like to visit dermatologist. There are several health services provided by medical centers that offer us as a result the appointment at a dermatologist chosen. What differentiates them is a price of a service, quality of specialist and medical center, date of possible appointment. The service that will be chosen depends on user preferences (i.e. urgency, price, distance to medical center, etc.) [16].

E. Dynamic supply chain

The dynamic supply chain scenario [10] assumes the existence of QoS ontologies and of an engine capable of dynamic orchestration. The motivation of this use case is simple – the retailer wants to choose the best suppliers based on its business logic and the suppliers want to maximize their profit as well as increase their business with the retailer. A key feature of the scenario is the use of quality of service (QoS) as an essential criterion for dynamic selection of services that are later dynamically composed. The scenario assumes multiple interactions between the platform (as a proxy) and the user (retailer). The result of the interactions is a placed order.

F. Complaints handling by local authority

This is an example of e- or m-government solution. Most of the administrative processes are predefined by law or internal regulations of the institution e.g. when authority receives an inquiry or a request from citizen it is obliged to send response within a predefined time. Moreover also the processes in institutions are defined, so when citizen sends a complaint, it is redirected to a certain department, dealt with and then sender is notified about the action that was undertaken. It is envisioned that citizens could send complaints to the local authority with their mobile devices, and be notified of the status of the complaint and solution for it [16].

G. Bookstore on the Web

The bookstore scenario [3] motivates the need for semantic annotation in protocol definition languages. It is shown that WSDL does not describe some important aspects of web services such as implications and effects of an operation. Interested parties are the bookseller, the customer and Semantic Web services middleware vendors. On the bookstore side, Semantic Web services allow for the definition of

more sophisticated behavior. On the customer side, the interest lies in easier understanding of the service's behavior and of how to leverage them. Similar order-purchase-like scenario is presented in [13]. For a B2B domain there is a scenario [11] whose actors are companies and organizations that would like to exchange messages that will trigger their business processes.

H. Amazon

The Amazon.com web application is compound in nature. It consists of several services: www service (which in turn comprises of search service, product catalog online etc.), transaction handling service, package delivery service. Some of the complexity of this service is exposed through Web service interface. This includes information services e.g. getting detailed information about particular item or list of items as a response to a query and shopping cart interface that allows for placing orders through Amazon. WSMO group (www.wsmo.org) has developed Amazon E-commerce Service use case which describes the service in the WSMO language. The functionality includes various kinds of searches and setting up shopping cart. Purchasing is not possible in this scenario, as well as through native service [15].

I. eBay

eBay's Web services allow for executing almost every operation accessible on the website (listing items, searching, receiving notifications), although they do not support bidding yet (as opposed to other regional auction services, e.g. www.allegro.pl). In the year 2004 the eBay Platform handled over 1 billion Web service requests per month [8].

J. Google

Last but not least is the example of simple Web service which is Google API. It allows querying the Google retrieval engine, and obtaining results in XML format. This is another example of successful application of Web services as an interface to the already existing information services.

The common point of all the services described in this section is the added value to the end user – the ability to query offerings, perform reservations, transmit documents and made payments etc. Moreover, some of the scenarios consider services that are performed in the real world, like health services, supply of materials, traveling or even book delivery. Finally, as one may see not only the semantics are important for successful composition of Web services but the other crucial criterion is the quality of service aspect. In the consequence, taking these three aspects into account, we may distinguish two types of Web services presented in the next section that have a crucial impact on the composition process.

4 Service types

The use cases presented above can be analyzed from the perspective of the type of services' results. The services in all the scenarios follow at least a simple interaction pattern: query – response. The queries are used to retrieve an offer of a particular kind (books, transportation routes, tickets, healthcare, industrial suppliers). The expected result is a list of interesting items (information result). The other type of interaction is to send a message with the goal to trigger some further workflow. The results in such

a case are twofold. The first is an immediate confirmation sent to the user (order confirmation). The second is an actual, real-world (possibly postponed) service, e.g. the book delivered to the user, the reserved place on the plane the user is just entering, etc (real-world result). As there are two kinds of result of services we distinguish two kinds of services – real-word and information services. Real-world service is a service (properly defined piece of work) offered by some provider (through traditional channels like face-to-face communication and hand-to-hand delivery) in the real world. Information service is such a service that its sole relevant result is a piece of information and the result can be transmitted to the customer through any possible communication channel.

Note that in order to utilize a service in some information system the service must be wrapped-up with a machine-processable interface. Such an interface is a Web service according to the W3C, which states that it is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface that is described in a machine-processable format such as WSDL.

The above division (to real-world and information services) together with the notion of Web service as an interface allows us to describe services in terms of tangibility of their results by any information system. In case of information system the user is only interested in the result that can be captured by some Internet communication protocol. In case of a real-world service the user besides the feedback information is interested in results that cannot be controlled. This intangibility is an effect of the lack of full integration between service provider's system and systems of its environment as well as machines' inability to control physical results of provided services.

The use cases presented in the third section mix different notions of service. On one hand they are all information services. Actually some of the use cases utilize WWW browsers, some provides API, and some provides dedicated solutions. On the other hand the core of some of the use cases is to book or order something. Wrapping information service is quite simple, because information service may be seen as a computational function with simple input and output. Wrapping real-world service is more complex. We think that the goals for designing scenarios discussed above miss this distinction, which is the reason for the fact, that only simple, well known services like Amazon, Google, auctions actually works. Moreover only few of their capabilities are accessible through API (not GUI). Not taking into account differences in meaning of Web service is the reason, why it is hard to find good use cases that deal with real-world services, and yet requires Web services, and reach semantics and semantic-based composition.

5 The impact of service types on composition

Taking into account the types of service distinguished in the previous section, the conclusion appears that the service composition is not as monolithic as it would seem. If we take into account different service types and their semantic description accordingly, we obtain different levels of composition. The following levels of composition may be distinguished:

- Full composition – it appears in the scenarios that consider large set of substitutive services. When the user goal is formulated, then the process specification, taking into account desired inputs, outputs, preconditions and effects, needs to be defined. Then adequate services implementations are bound to the process specification.
- Limited composition – in this case the process definition is fixed in terms of types of services (service specifications), however different services (implementations from different providers) may be attached / negotiated to become a part of the composition. Those negotiated services may be already contracted or internal to the company, or they may be external. In the latter case, the composition usually requires SLA's and negotiation of contract.
- No composition – the process is fixed, the providers of services on each step of process are known.

Each of the composition levels mentioned above is available to both real-world services and information services (possibly wrapped into Web services). Yet due to characteristics of real-world services it is very hard to mix them with information services within one process. If we try to do so we would have to consider such effects of real-world services that are not accessible to information services. Moreover, composition is also dubious due to the approach to semantic service description. Real-world services may have semantic description which is just symbolic representation of what they perform. However information services (like ticket booking service) must be described as information delivering or changing service (in this example – enabling some other service), having only minor impact on the real world.

Mixing information and real-world services in one composed process is also challenging for its execution. It is only possible to monitor execution of real-world services (the platform that executed the whole process gets notifications when the service is finished), while for information services wrapped in WS it is actually possible to execute them remotely. Moreover, it is rather easy to handle the composition of information services, because an output of one service is an input of the next service. Real-world, complex services that really effect in the creation of real objects do not provide simple data types that can be processed electronically.

A few more limitations that may be applied to the service composition will be discussed in the following subsections.

5.1 The role of Quality of Service in service composition

Service composition is definitely limited by the quality of service considerations. It is important to note, that service composition is usually quality-driven, while its goal is to achieve certain functionality. However, when talking about quality of service issues two aspects need to be taken into account. On one hand the quality of Web service implementation (the interface) and on the other hand the quality of the resulting service, made available through the use of Web service. In the consequence, the quality concept should be divided into two groups – Quality of Execution (QoE) and Quality of Result (QoR) [1].

QoE parameters characterize services in general. They usually include execution time, execution cost (not to mistake with different prices appearing around the

service), latency, response time etc. QoR in turn describes the output of the service – it is hard (or even impossible) to enumerate all the parameters for measuring quality of arbitrary output, however it always includes price. The example for this distinction may be ticket booking service (wrapped with WS). Its QoE parameters would measure how long did it take to book a ticket and what was the cost of using this particular service. However the price of the ticket is QoR for this service, together with place for which the ticket is booked, time, etc. Quality of Execution is rather domain-independent whereas QoR concept is domain specific. However, in some cases the differentiation between QoE and QoR is rather subjective. It depends strongly on the users' point of view, their previous experiences as well as goals and expectations.

Taking this all into account, the quality of service is not easy to be defined and measured. For those scenarios that include binding concrete services to their specifications (limited composition or second step of full composition), services are only distinguishable via their QoS. As QoS is only partially measurable it leads to problems in satisfying user preferences. Additionally, from the user perspective, problems with QoR have deep impact on possible queries that can be asked against service repositories. For example users can ask for cheapest services but not for services that would get them cheapest tickets because that would require executing the services even before they are chosen.

5.2 Requirements for service composition from the perspective of semantic service description and goals formulation

The indepth analysis of the use cases implies that different types of services (real-world, information) will have different semantic description. Does this influence composition capability (in terms of mixing both types or even within single type)?

It seems that the deepest challenges lie in the case of full composition. There the services are not known in advance because the user does not specify which service he or she would like to use. He or she only specifies the desired goal, giving its semantic description. This may be viewed as specifying service capabilities in terms of preconditions and effects. Four illustrative examples of such goals (corresponding to the scenarios discussed) would be:

- I want to have a certain book,
- I have a free evening and I want to get out spending it on something interesting,
- I'm in New York today (15th of January) and on March 12th I need to be in Hong Kong on the conference,
- I have ill sister and I want her to get better.

It seems like these questions would fit for bookstore, attraction booking, travel or healthcare use cases. However those scenarios require specifying the service composition on some general level first - preferably from certain available service categories. Those categories should include services that deliver books, make arrangements for evenings, transport or heal. While all what is described in the scenarios are information services: that inform about books (and allow ordering them), find attractions or book tickets. It is hard to match user goals expressed this way with such service descriptions.

Furthermore, it seems that the composition may be achieved only after knowing the outcomes of certain services. It is possible to bind some services to the process only if the results of previous services are known. For example: if my process includes paying for a ticket, the payment methods available for ticket booking service determine which transaction handling service I should use.

Users of platforms that act as middleware between service providers and customers need to query for service to be discovered and executed. This is often referred to as specifying user's goal. Several types of user queries are conceivable:

- Give me all available services that can do... – this is a general question about certain category of services. If they are not directly discoverable through semantic descriptions then full composition is required.
- I do not know what to do with my free evening... - another question that requires full composition.
- I want to go to the cinema on the movie titled... Which one do you propose? – here the user specifies some of the results of services that he or she would like to obtain. This is the case for limited composition - only some categories of services need to be considered, the process is fixed and all is necessary is finding the service that books tickets and then plays the right movie.
- I want to book a ticket for cinema... – in this case there is no need for composition. The process is fixed and the user specifies (indirectly) which service (from which provider) he would like to use.

The level of the above questions determines how the services should be described. For the first one any description is sufficient (even textual), however it is tremendously challenging even with fullest semantic description to conduct automatic service composition against such queries. The sheer number of variations of processes may be prohibitive, not to mention that the user will have to choose which one he or she prefers. With regard to the QoS and other parameters – user does not provide any values of parameters (possibly only some QoS parameters – like execution time of the whole composed service or its price; for example: evening for 100 USD)

For the second – the description should include information which cinemas a given service is able to book (+ values of other runtime parameters) – here the characteristic thing is that the user provides some of the input values with his or her goal description (here – the name of the movie). The problem here is how to check which of the service is able to handle the request prior to invoking it!

For the third, a concrete service may be successfully discovered and invoked. If there are many services available the user may choose one of them based on QoS parameters, his or her preferences or any external information.

6. Conclusions

Introducing several levels of service composition we discussed how the type of a service, its QoS parameters and semantic description influence the feasibility of composition. From the discussion of several use cases we clearly see that currently only simple information services are applicable, possibly with fixed workflows and limited composition. We observe that Web service composition use cases often miss

the distinction between real-world services and information service. The reason for this is the misconception that Web services that serve as front-end for real-world services and may be described as if they would be the real services. Google and eBay examples show that Web services make sense in business environment where they really make business processes more efficient or simplify them.

Moreover, it is inherent for (real-world) services that they are partially undefined and become defined on the time of their execution (which also determines the price). Therefore for complex services it is very hard to automate the process of composition and even to deliver proper description. It is something that should be remembered.

References

1. W. Abramowicz et al., "A Survey of QoS Computation for Web Service Profiling , ISCA 18th International Conference on Computer Applications in Industry and Engineering, 2005
2. J. Noll et al., "ASG based scenarios in Telecommunications, Telematics and enhanced Enterprise IT", Deliverable on http://asg-platform.org , 2004
3. B. Benatallah, F. Casati, F. Toumani, "Conversation Protocol of the book purchase service", 2003
4. N. Milanovic, M. Malek, "Current Solutions for Web Service Composition", Internet Computing Vol. 8, No. 6(6), pp. 51-59, 2004
5. M. Gajewski, M. Momotko, "Dynamic Failure Recovery of Generated Workflows", 16th International Workshop on Database and Expert Systems Applications (DEXA'05) pp. 982-986, 2005
6. G. Piccinelli, Service Provision and Composition in Virtual Business Communities (HPL-1999-84), Technical report, Hewlett-Packard, 1999
7. H. Haas, Web service use case: Travel reservation, 2002
8. W. Iverson, "Web Services in Action: Integrating with the eBay Marketplace", retrieved 10th January 2005, http://developer.ebay.com/join/whitepapers/webservicesinaction, 2004
9. P. Lipton, "Composition and Management of Web Services", SOA Web Services Journal., 2004
10. K. Verma, A. Sheth, J. Miller, R. Aggarwal, "Dynamic QoS based Supply Chain", retrieved 10th January 2005, http://www.daml.org/services/use-cases, 2004
11. C. Bussler, M. Zaremba, "Semantic Web-enabled Business Protocol Standards", retrieved 10th January 2005, http://www.daml.org/services/use-cases, 2003
12. S. R. Ponnekanti, A. Fox, "SWORD: A Developer Toolkit for Web Service Composition", in the Eleventh World Wide Web Conference (WWW2002), 2002
13. H. He, H. Haas, D. Orchard, "Web Services Architecture Usage Scenarios", retrieved 10th January 2005, http://www.daml.org/services/use-cases, 2004
14. M. Stollberg, R. Lara, "WSMO Use Case <<Virtual Travel Agency>>", retrieved 10th January 2005, http://www.wsmo.org, 2004
15. J. Kopecky, D. Roman, J. Scicluna, "WSMO Use Case: Amazon E-commerce Service", retrieved 10th January 2005, http://www.wsmo.org, 2005
16. D. Tilsner, W. Abramowicz, M. Wiśniewski, P. Moore,G. Peinel, USE-ME.GOV (USability-drivEn open platform for MobilE GOVernment), The Proceedings of the First European Conference on Mobile Government, 2005, University Sussex, Brighton UK, ISBN 9763341-0-0Baldonado, M., Chang, C.-C.K., Gravano, L., Paepcke, A.: The Stanford Digital Library Metadata Architecture. Int. J. Digit. Libr. 1 (1997) 108–121