Information and Software Technology 56 (2014) 352–373

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Semantic-based automatic service composition with functional and non-functional requirements in design time: A genetic algorithm approach

Yong-Yi FanJiang [a,*], Yang Syu [b]

[a] Department of Computer Science and Information Engineering, Fu Jen Catholic University, No. 510, Zhongzheng Rd., Xinzhuang Dist., New Taipei City 24205, Taiwan, ROC
[b] Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei City, Taiwan, ROC

ARTICLE INFO

Article history:
Received 5 March 2013
Received in revised form 2 December 2013
Accepted 2 December 2013
Available online 14 December 2013

Keywords:
Service composition
Semantic web
Quality of service
Transaction
Genetic algorithm

ABSTRACT

Context: In recent years, the composition of ready-made and loosely coupled services into desired systems is a common industrial approach and a widely followed research topic in academia. In the field, the current research trend is to automate this composition; however, each of the existing efforts automates only a component of the entire problem. Therefore, a real automation process that addresses all composition concerns is lacking.
Objective: The objective is to first identify the present composition concerns and subsequently to devise a compositional approach that covers all concerns. Ultimately, we conduct a number of experiments to investigate the proposed approach.
Method: We identify the current composition concerns by surveying and briefly describing the existing approaches. To include all of the identified concerns, the solution space that must be searched is highly dimensioned. Thus, we adopt a genetic algorithm (GA) due to its ability to solve problems with such characteristics. Proposed GA-based approach is designed with four unusual independent fitness functions. Additionally, experiments are carried out and discussions are presented for verification of the design, including the necessity for and correctness of the independence and priority of the four fitness functions.
Results: The case studies demonstrate that our approach can automatically generate the required composite services and considers all identified concerns simultaneously. The results confirm the need for the independence of the fitness function and also identify a more efficient priority for these functions.
Conclusions: In this study, we present an all-inclusive automatic composer that does not require human intervention and effort during the composition process and is designed for users who must address multiple composition concerns simultaneously, including requirements for overall functionality, internally workable dataflow, and non-functional transaction and quality-of-service considerations. Such multiple and complex composition requirements cannot be satisfied by any of the previous single-concern composition approaches.

1. Introduction

Currently, one of the most promising, popular, and widely followed software development paradigms is service-oriented architecture (SOA), which advocates reusing the existing loosely coupled and network-accessible services provided by professional organizations to quickly compose on-demand service-based systems rather than programming new systems from scratch to satisfy customers [1,2]. Several approaches to implementing SOA exist, but the most common and famous approach is delivered via web services (WSs) technology [3], which depends on the World Wide Web's infrastructure and uses open XML standards (e.g., SOAP, WSDL). Another benefit of WSs is the interoperability that can cross boundaries between different organizations or heterogeneous units [4].

Under SOA, a service that exactly meets the requirement must be found (Service Discovery) to satisfy the requirements of a service with certain functional and non-functional properties. If such a single service does not exist, the service integrator (in this study, a more intuitive term is *service composer*, as used in [5]) must choose the appropriate services and compose them into a value-added composite web service (CWS) to fulfill the requirements. In general, a situation in which a requirement for service can be directly satisfied by a single service is quite rare because the functionality of a service is always limited (fine-grained) [6] for better reusability. Hence, in most cases, the composition of services is inevitable.

* Corresponding author. Tel.: +886 2 29052444; fax: +886 2 29023550.
E-mail address: yyfanj@csie.fju.edu.tw (Y.-Y. FanJiang).

0950-5849/$ - see front matter
http://dx.doi.org/10.1016/j.infsof.2013.12.001

Many services are currently available on the web or in certain repositories, and the total number of these services is continually increasing [7], which raises the complexity of composition [8]. The cost and effort involved in manual composition of an intended CWS is sufficiently higher than that of automation [9]. In addition, the difference in reasonable cost and effort between manual handcraft and automation is particularly magnified in cases in which the requirement for service (composition) is frequently requested or changed. Thus, an urgent need exists for techniques that can compose services automatically, thus reducing the cost and effort.

According to our investigation, the topic that addresses this technique is classified as "automatic service composition". This topic focuses on how to use the existing and available services offered by various organizations with different features, including functional (i.e., booking, payment) and non-functional (i.e., quality-of-service (QoS) requirements, such as response time and availability) as well as behavioral transaction property (TP, i.e., retry-able or compensatable) perspectives, to quickly compose workable service-based applications or software (CWSs) that satisfies requirements that cannot be fulfilled by any single service. In this manner, the cost and effort needed to develop software or applications will be significantly reduced, thus enhancing the competition for the advocators of SOA.

Generally, the life cycle of a CWS includes three stages: (1) design, (2) runtime and monitoring, and (3) re-engineering [10]. The first stage of design is used to identify the concrete specifications of the CWSs. The runtime and monitoring stage addresses the execution of the CWSs and also detects violations and errors that emerge during this stage. Finally, the re-engineering stage revises the specifications of the CWSs depending on information gained from stage 2. This process is rather static, and more dynamic approaches (e.g., [1]) are able to revise the CWSs immediately at stage 2 (dynamic binding) using certain automations or mechanisms.

In this paper, we address the design stage. The entire design stage can be further divided into three steps [9]: (1) construction of the CWS workflows (known as planning in [9]) in which each workflow consists of non-executable abstract activities; (2) discovery of proper candidate services for each abstract activity; and (3) selection of the fittest service for each abstract activity from all of the candidate services discovered in the previous step. Initially, the requirement for service includes a description of the desired functionality and non-functional features and the impossibility of satisfying the requirement with a single service. Based on the requirement, a service composer (integrator) must manually (or semi-automatically) generate a workflow, which is a logically ordered set of activities, to provide an abstract plan that exactly meets the requirement. Thus far, each activity in the generated plan is still an abstract functional unit (abstract service [11]) instead of a truly executable service (concrete service [11]). Next, the composition process enters step 2, or service discovery, which involves discovering the appropriate and concrete candidate services for each abstract activity in the generated plan [9]. In this step, the discovery primarily involves functional matchmaking between concrete services and abstract activities (e.g., to identify, classify, or cluster qualified concrete services with the input–output (IO) specification required by an activity). According to our observations, most automatic service selection processes operate in a manner similar to those presented in the next section and do not mention or address this step. An implicit assumption of the previous works is that all qualified services are already known and aligned for the activity and are waiting to be selected in the next step. In other words, the previous investigators did not address how the qualified candidate concrete services for an activity are classified and distinguished (discovered). Consequently, after step 2, each activity could be driven by many candidate services. In step 3, the task is to pick the fittest service for each activity from all candidates according to the specified criteria (e.g., QoS). The goal is to obtain a CWS with optimal or acceptable specified non-functional features.

As described in the previous paragraph, the entire design-stage service composition process is time consuming and complex, and if it is carried out by automation, the cost and effort will certainly be lower than if this process is carried out manually [9]. Thus, many research efforts are currently underway to attempt to automate service composition. However, after a comprehensive investigation of the existing works, the issue that emerges is that a truly automatic composition method is necessary because all of the existing works contain certain shortcomings.

Overall, a subset of the works surveyed aims to find a executable workflow that will connect services in a CWS to fulfill a functionality demand [7,12–18]. The others concentrate on how to select the fittest service for each abstract activity contained in a predefined workflow to meet such non-functional demands as QoS [1,2,4,6,11,19–24] or a behavioral TP [25–29]. In contrast with QoS, the number of publications on composition research in the TPs of WSs is small; for additional information and background on WS transactions, refer to [30]. Otherwise, there are a few remarkable special cases, such as that presented in [3], which simultaneously consider both QoS and the TP, but in terms of automation, a primary drawback of this work is that this approach requires executable workflow templates as an input. Furthermore, the approaches in [7,13,22–24,18] consider the concepts of semantic reasoning instead of employing only poorly constructed syntactic keyword comparisons, thus making connections (dataflow) between services that are closer to reality. The reasons for and advantages to considering semantic reasoning rather than more traditional methods, such as keyword comparison, can be found in Section 3 of [31], which provides a clear introduction and analysis of the advantages and disadvantages of such approaches.

In the previous paper [32] of this study, we presented a theoretical keyword-based composition approach, which was designed to compose a workflow automatically with QoS awareness and thus closed the existing gap in the current automatic composition works. In that preliminary paper, without any implementation and experiments, we briefly described the three defined fitness functions as well as the operators and chromosome pattern of the Genetic Algorithm (GA) to describe an approach that addresses the two composition concerns (automatic composition workflow with QoS awareness) that we found at that time. The current paper is a complement to an extension of the previous paper. First, we conduct a more thorough survey of this field to identify and include another two common composition concerns in the field (awareness of the transactional property and semantic matchmaking). Next, we specify the entire approach in detail, including precise mathematic definitions of the proposed four fitness functions. Finally, this paper presents implementation, experiments, and discussions of the modeled problem and devised approach.

In summary, the designed approach, with its plural and ordered fitness function design as well as a special selection mechanism, will allow users of the approach to request services with the intended functionality, TP, and QoS level simultaneously. Through a series of experiments, we discover that the independence of and the priority order among the fitness functions are necessary, and an effective and efficient priority order is obtained in the experiments. In the following, we itemize the contributions of this paper.

(1) First, we survey and analyze the current automatic composition efforts and identify several composition concerns as well as the existing gap in the reviewed efforts.
(2) To fill the gap, this paper proposes a fully automatic, one-step, design-stage service composition approach based on the GA that covers all of the identified composition concerns.