Orchestrating SOA Using Requirement Specifications and Domain Ontologies

Manoj Bhat, Chunyang Ye, and Hans-Arno Jacobsen

Application and Middleware Systems Research Group
Technische Universität München, Germany
{manoj.mahabaleshwar,yec}@tum.de

Abstract. The composition of web services requires process designers to capture the goals of the service composition in a partial process model. Manually deriving the partial process model from the requirement specifications is not trivial. A clear understanding of the requirements, interaction among services, their inputs and outputs are precursors for developing the partial process models. To reduce the complexity, we propose an approach to guide process designers in deriving the partial process models by reusing the knowledge captured in requirement specifications and domain ontologies. The results of the evaluation shows that our approach is promising in terms of correctness and completeness.

Keywords: Web service composition, domain ontology, requirements engineering, knowledge reuse.

1 Introduction

Services Computing is an interdisciplinary field that covers the science and technology of using computing and information technology (IT) to model, create, operate, and manage services that bridges the gap between business and IT [1]. Increase in the creation and consumption of web services has made the analysis and generation of composition plan challenging [2]. Approaches that tackle the issue of service composition require users to capture the service composition requirements in the form of service templates, service query profiles, or partial process models [8–10]. The requirements include: list of sub-services, inputs, outputs, preconditions and effects (IOPEs) of the sub-services, and the execution order of these sub-services. Henceforth, we refer to the templates that capture these requirements as *partial process models*. The existing approaches assume that the partial process models are readily available to initiate the service composition engine. However, this assumption does not always hold in practice [3].

In this paper, we address the issue of automatically deriving the partial process model for service composition. The goal is to reduce the burden of process designers to a great extent, especially for non-domain experts. Our experiment discussed in Section 3 indicates that the main challenges for process designers include: understanding the composition requirements of complex services, correctly correlating the inputs and outputs of sub-services, and designing the business

X. Franch et al. (Eds.): ICSOC 2014, LNCS 8831, pp. 403–410, 2014.

logic of the process model. To address these issues, we propose to use domain ontologies, user stories in the requirement specification documents (RSDs), and user queries to recommend a set of services along with their inputs, outputs, and execution order to describe the partial process model.

Ontologies are extensively used in different phases of software engineering [4]. In recent years, organizations are putting in the extra effort for manually creating domain ontologies due to their significant advantages as a means of knowledge sharing and reuse. There also exist automatic and semi-automatic approaches for ontology creation. For instance, LexOnto [5] uses the web services in the Programmable Web directory [6] to create classification ontologies. As ontologies are generally available or they can be generated using existing tools, we consider using ontologies to facilitate the automatic generation of partial process models.

On the other hand, the popularity of agile methodologies has made the use of user stories to capture requirements a common practice. The user stories are expressed in a standard format such as "As a *role*, I want *goal* so that *benefit*." To simplify the algorithm, we focus only on the user stories. However, our approach can be generalized to the individual statements in the RSDs.

Since a service composition may involve services from different domains, one of the main challenges of our approach is to link domain ontologies to handle requirements from multiple domains. To address this challenge, we extended the existing approach by Ajmeri et al. [7] that helps requirement analysts visualize how requirements span across multiple domains. The ontologies are linked using the semantic similarity of concepts. The linking of ontologies is used to derive a conceptual model of the requirements to help requirement analysts improve the completeness of requirements. Furthermore, we use natural language processing (NLP) to link the concepts in the ontologies with the terms in the user stories and to classify the concepts as either services or input-output (i/o) of services. Once the atomic services and their inputs and outputs are identified for a queried service, the data dependency constraints determine the execution order.

The main contributions of this paper are two-fold. First we propose an approach to automatically generate the partial process model for service composition. Our approach complements the ideas behind the existing ontology-based and NLP-based service composition approaches. Second we realize our approach as a recommender system that can integrate ontologies and user stories to substantially reduce the time and effort involved in service composition.

2 Related Work

Artificial Intelligence (AI) Planning and Ontology Based Solutions: To address the problem of service composition, different planning techniques have been proposed. These approaches require capturing the requirements as a partial process model which is given as input to the composition engine. The engine generates a plan by comparing the requirements against the services in the repository. For instance, in [8] the goals are captured in a goal language and the planner generates a plan that satisfies these goals. Similarly, in [9] the similarity

between the user requests and the services is computed based on the syntactic, semantic and operational details. Furthermore, Grigori et al. [10] propose a method to transform the behavior matching of services and user requests to a graph matching problem. Capturing the requirements in a partial process model is a non-trivial task. We address this issue in our approach by automatically deriving the partial process model from the RSDs and the domain ontologies.

Tag-Based Service Composition: The tag-based approaches for service composition are becoming popular [11]. They are easy to implement and to use. However, they have their own shortcomings, for instance, tags are usually too short to carry enough semantic information and most services have only a few tags. These tags can belong to different types such as content-based, context-based, attribute, and subjective. This results in a large tag-space and low efficiency and effectiveness in semantic matching. Therefore, these approaches are oriented towards mashup. They do not address how to generate traditional workflows which involve sequentially and parallel executing tasks.

Publish/Subscribe-Based Distributed Algorithm: Hu et al. [12] propose a distributed algorithm to enable service composition via a content-based pub/sub infrastructure. Even though the distributed approach seems promising, it considers matching of services at a syntactic level, whereas our solution concerns both syntactic and semantic levels.

Service Composition Using NLP: The service composition system proposed in [13] addresses the shortcomings of the existing NLP-based solutions [14]. The solution proposed in [13] comprises of an integrated natural language parser, a semantic matcher, and an AI planner. The parser extracts the grammatical relations from the requirements and generates the service prototypes comprising of process names and input values. The service prototypes are further used by the semantic matcher to identify services from the repository and the AI planner generates a composed service. In this approach, a clear set of patterns used to identify the process names and their input values from the requirements is not captured. Furthermore, a detailed evaluation of the system with respect to correctness and completeness of the generated composed services is missing.

Although, there exist a large body of knowledge addressing service discovery and composition, understanding the technical and domain specific requirements for process designers to use these approaches still remains a challenge.

3 Case Study of Service Composition

As part of the course curriculum [15], participants are required to complete a project comprising three milestones (M). In M1, each group is required to develop three web services. In M2, participants develop a search engine to look up services from the server. Finally, in M3, each group is required to develop a Travel Agency (TA) service by composing the existing services.

Out of 76 deployed services in M1, 47 services were correctly implemented. However, in M3, 8 groups deployed the TA service and only 1 group correctly implemented the service. The drop in the performance is overwhelming but understandable. One of the main challenges for the participants was to address

the problem at a conceptual level. The requirements of M1, to develop atomic services were straightforward and easy to comprehend. The participants successfully developed a conceptual model which helped them implement the services. On the other hand in M3, the TA service consisted of several sub-services which were from different domains and implemented by different groups. The participants were unsuccessful in creating a conceptual model due to the lack of a clear understanding of the complex relationships among the services.

This example shows that manually creating a correct partial process model is challenging. Our investigation indicates that the main difficulties include how to design the business logic of the process correctly and how to correlate their inputs and outputs correctly. If the participants were presented with a process model as shown in Figure 1, understanding the composition requirements would be easier and the time and the effort involved would be substantially reduced.

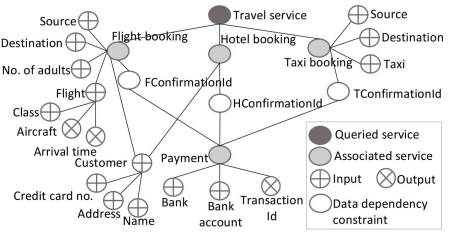

Fig. 1. Partial process model for TA service

4 Approach

The challenges that we address in our approach include: mapping of domain ontologies using semantic similarity mapping approach, identifying if a concept is a service or an i/o of a service, capturing the data dependency of services, and integrating the recommendation with user preferences.

Overview: In this section, we briefly introduce how the partial process model is generated. As shown in Figure 2, the process designer inputs the keywords corresponding to the service he plans to develop. He also selects the relevant domain ontologies and provides the list of user stories to the system. The system maps the concepts in the domain ontologies to the terms in the user stories and identifies the candidate sub-services. The concepts in the domain ontologies associated with the candidate services are identified as i/o concepts of sub-services. Users' past preferences filter the candidate services and tag the i/o concept as either input to a service or output of a service. Tagging of concepts as inputs and outputs of sub-services establishes the data-dependency constraints to determine the execution order of the sub-services. The process designers' selection iteratively updates the user preference repository to improve the recommendations. Process designers can also suggest the missing concepts to evolve the ontologies, which are validated by a domain expert before updating the ontologies. The partial process model created based on the recommendations is further given as input to the composition engine that retrieves services from the service repositories.

Cross-Domain Ontology Mapping: Ajmeri et al. [7] propose an approach to identify the concepts that link different ontologies using semantic similarity mapping. The semantic similarity between concepts is calculated based on

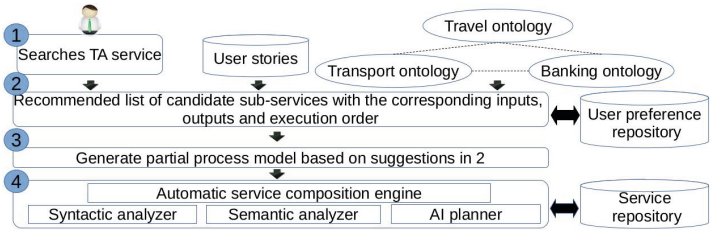

Fig. 2. Service composition using user stories and domain ontologies

syntax, *sense*, and *context* similarity. The *syntactic similarity* matches concepts based on the string equivalence. The *sense similarity* matches concepts based on the similar usage sense determined using a set of synonyms called synset [16]. The *context similarity* matches concepts based on the similarity of their neighborhood. The neighborhood of a concept is determined based on the inheritance relationships from the parent concept to the child concept. The *semantic similarity* between the concepts is computed as a weighted mean of syntactic, sense, and context similarity. The semantically similar concepts identified using the above approach are the concepts which map two different domain ontologies (cf. [7] for a detailed description of the cross-domain ontology mapping algorithm).

Identification of Candidate Services: Once the domain ontologies are linked using the semantic similarity mapping approach, we identify if the concepts in the ontology that are associated with the user query are services. We use the following criteria in [7] to identify if the concept represents a service: the concept or its equivalent concept in the ontologies must be present in the user stories, and the concept or its substring should be part of a verb phrase (VP). However, if it is part of a noun phrase (NP), it should be prefixed by a VP.

For example, in the user story "As a customer, I want *flight booking* functionality, to book flight from source to destination city" the substring *booking* of the concept *flight booking* is part of a VP and hence the concept *flight booking* is suggested as a candidate service. To parse a sentence and to create a constituent tree of objects, we use the open-source Link Grammar library [17].

service(c_1) = US_i.contains(c_1) and (c_1.parseType = "VP" or (c_1.parseType = "NP" and c_1.parent.parseType = "VP")) ? true : false;
where US_i is the i^{th} user story; c_1.parseType is the tag associated with the concept c_1 w.r.t. parse tree of US_i; c_1.parent is the parent token in the parse tree.

Constituent parse tree: (S (PP As (NP a customer)), (S (NP I) (VP want (NP (NP flight) (*VP booking* (NP functionality, (PP to (NP book flight) from source city to destination city.)))))))

Also, if the concept is part of a NP, it should be prefixed by a VP. In the user story "As a customer, I want to *make payment* for flight, taxi, and hotel booking using credit card", *payment* is in NP but is prefixed by verb *make*.

Identification of Input and Output Parameters: To suggest the i/o of the identified services, we consider the concepts associated with each of these services in the ontologies. The associated concepts are referred to as complementary concepts (CC). We consider 1st and 2nd degree CC. The 1st degree CC is a concept associated with the service concept directly through an object property relationship and the 2nd degree CC is a concept associated with the service concept via an intermediary concept. The open-source OWL API [18] provides interfaces to parse the ontologies and to identify the object and data properties.

User Preference Integration: Based on the users' past preferences of services and their i/o, the identified services are filtered and the complementary concepts are tagged as either inputs or outputs. When the user selects services from the recommended list, the confidence of these services is incremented. However, if a service is considered irrelevant for a specific query, the confidence score of that service is decremented. For a specific query, the recommended services are sorted based on the confidence score. Similarly, as the user tags the concepts as either inputs or outputs of a service, the corresponding confidence-input or confidence-output is incremented. If the user deletes the complementary concept, the corresponding confidence-input and confidence-output is decremented. For the subsequent queries, the complementary concept is tagged as input or output by comparing the score of confidence-input and confidence-output.

Data-flow Analysis: The tagging of complementary concepts as inputs and outputs of a service establishes the *data dependency constraint* which helps to compose the services to meet the requirements of the queried service. For instance in a *Travel Agency* service, the output of *Flight*, *Hotel*, and *Taxi Booking* service are the inputs to the *Payment* service, indicating that *Flight*, *Hotel*, and *Taxi Booking* service should be executed before the *Payment* service.

5 Evaluation

To evaluate the system developed based on the approach discussed in Section 4, we have considered two practical use-case scenarios which are commonly used as the benchmark examples for service composition[1]. We also conducted an empirical study to evaluate the quality of the recommended partial process model.

Empirical Study: We introduced our system in the course assignment which had the same requirements as the assignment in Section 3. The participants were assigned groups and each group was required to submit three milestones. M3, involved developing the TA service by composing other services. The participants were given a brief introduction on how to use our recommender system.

27 groups deployed their services for M3 and 33% of the groups correctly implemented the service. Services implemented by 44% of the groups failed to pass the test cases due to syntactical errors such as *incorrect variable initialization* and *incorrect namespaces*. However, these services were complete with respect to the requirements; in the sense that, the TA services included invocation of all the

[1] Due to space limitations, the case studies are available online via `https://sites.google.com/site/wsccs2013`

necessary services, assignment of input-output variables and also maintained a correct execution order. The performance of the groups in this assignment (33% correctly implemented and 44% partially completed) is significantly higher than the groups in the assignment discussed in Section 3 (12%).

On completion of the project, 27 groups provided their feedback regarding the recommender system. A five-point Likert scale [19] is used to capture the responses. In the five-point scale, 1 indicates strongly disagree, 2 indicates disagree, 3 indicates neutral, 4 represents agree and 5 indicates strongly agree. We refer to this scale in the following observations *(O)*:

Understanding Service Composition Requirements: *O1*: 37% of the groups strongly agree and 25% of the groups agree that the recommended process model helps in understanding the requirements of service composition.

Quality: *O2*: 40.7% of the groups strongly agree and 14.8% of the groups agree that the recommended services fulfill the service composition requirements.

O3: 51.8% of the groups agree and 14.8% of the groups strongly agree that the data dependency constraints help to define the execution order of services.

O4: The average rating for the satisfaction of recommended inputs and outputs for each of the suggested service is 3.89. 46.8% of the groups strongly agree that the recommended inputs and outputs for all the suggested services are correct and complete with respect to the requirements. Moreover, 92% of the groups rate between 3 to 5 on the scale and only 8% of the groups, rate 1 and 2 on the scale (strongly disagree and disagree) in O4.

O5: 59% of the groups indicate that the recommended process model is the same as the manually created service composition.

Time: *O6*: The groups that used the recommender system for developing the TA service spent on average 9.68 hours less than those groups that did not use our recommender system.

Difficulty: *O7*: The feedback also indicates that the challenges in service composition are: "understanding the BPEL syntax (48%)", "handling i/o parameters in BPEL (30%)", and "invoking the external services (18%)", apart from "constructing the control-flow of the process (26%)" and "understanding the requirements (11%)". These results show that although our approach has addressed the issues related to conceptual modeling of services, some technological issues need further effort (e.g., high percentage of failure of services due to syntactical errors and difficulties in adapting to service specification languages).

6 Conclusions

Our approach, realized as a recommender system derives the partial process model using domain ontologies and user stories. The observations based on the evaluation indicate that our approach not only helps in understanding the requirements of service composition but also reduces the time and effort involved in the development of service composition. Our evaluation is based only on the case study and the empirical study. In our future work, we plan to extend the recommender system by integrating a composition engine so as to retrieve desired service compositions in a specific service specification language. Also, in

our approach, we have only considered data dependency constraint to identify the execution order of the composed services. Inferring similar constraints and non-functional requirements (e.g., QoS) on the services would further improve the recommendations for service composition.

References

1. Yan, Y., Bode, J., McIver, W.: Between service science and service-oriented software systems. In: Congress on Services Part II. SERVICES-2 (2008)
2. Xiao, H., Zou, Y., Ng, J., Nigul, L.: An approach for context-aware service discovery and recommendation. In: ICWS (2010)
3. Srivastava, B., Koehler, J.: Web service composition - current solutions and open problems. In: ICAPS Workshop on Planning for Web Services (2003)
4. Happel, H.J., Seedorf, S.: Applications of ontologies in software engineering. In: Proc. of Workshop on SWESE on the ISWC (2006)
5. Arabshian, K., Danielsen, P., Afroz, S.: Lexont: A semi-automatic ontology creation tool for programmable web. In: AAAI Spring Symposium Series (2012)
6. The Programmable Web, `http://www.programmableweb.com`
7. Ajmeri, N., Vidhani, K., Bhat, M., Ghaisas, S.: An ontology-based method and tool for cross-domain requirements visualization. In: Fourth Intl. Workshop on MARK (2011)
8. Traverso, P., Pistore, M.: Automated composition of semantic web services into executable processes. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 380–394. Springer, Heidelberg (2004)
9. Cardoso, J., Sheth, A.: Semantic e-workflow composition. Intell. Inf. Syst. (2003)
10. Grigori, D., Corrales, J.C., Bouzeghoub, M., Gater, A.: Ranking BPEL processes for service discovery. IEEE Trans. on Services Comput. (2010)
11. Liu, X., Zhao, Q., Huang, G., Mei, H., Teng, T.: Composing data-driven service mashups with tag-based semantic annotations. In: ICWS (2011)
12. Hu, S., Muthusamy, V., Li, G., Jacobsen, H.-A.: Distributed automatic service composition in large-scale systems. In: Second Intl. Conf. on Distributed Event-Based Syst. (2008)
13. Pop, F.C., Cremene, M., Vaida, M., Riveill, M.: Natural language service composition with request disambiguation. In: Service-Oriented Comput. (2010)
14. Lim, J., Lee, K.H.: Constructing composite web services from natural language requests. In: Web Semantics: Science, Services and Agents on the WWW (2010)
15. Introduction to Service Comput., `https://sites.google.com/site/sc2012winter`
16. Miller, G.A.: WordNet: A lexical database for English. Commun. of the ACM (1995)
17. Sleator, D.D., Temperley, D.: Parsing English with a link grammar. arXiv preprint cmp-lg/9508004 (1995)
18. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web (2011)
19. Allen, I.E., Seaman, C.A.: Likert scales and data analyses. Quality Progress (2007)