A Classification of Issues and Approaches in Automatic Service Composition

Ulrich Küster,[1] Mirco Stern,[2] Birgitta König-Ries[1]

[1] Institute of Computer Science, Friedrich-Schiller-Universität Jena, D-07743 Jena, Germany, ukuester|koenig@informatik.uni-jena.de
[2] Institute for Program Structures and Data Organization, Universität Karlsruhe, D-76128 Karlsruhe, Germany, mirco.stern@stud.uni-karlsruhe.de

Abstract. Resulting from the constantly increasing usage of service oriented computing, the problem of automatic composition of services is rapidly gaining attention. Many solutions have been proposed, yet too little attention has been paid to thoroughly analyzing the different needs that require the ability of (automatic) service composition. This paper therefore aims at identifying the different cases in which service composition is needed and at classifying existing approaches accordingly. Three distinguishable types of service composition applications are described along with their basic problems and various approaches addressing the identified classes are presented.

1 Introduction

Automatic service composition remains one of the key challenges of service oriented computing today. In general, "service composition" can be defined as creating a composite service, obtained by combining available component services. It is used in situations where a client request cannot be satisfied by any single available service, but by a combination thereof [1]. In terms of software engineering automatic service composition will significantly enhance the power of service oriented architectures (SOAs). SOAs combine available base services in order to build higher level services or distributed applications. Specifying manually which base services to use and how to combine them is a cumbersome task. Furthermore the resulting composite process is vulnerable to change as it must be corrected manually when base services become unavailable or new better services appear. Automatic service composition has the potential to dramatically change the way how SOAs are engineered: Given a rich set of base services and efficient, reliable automatic service composition methods the vision of programming as specifying *what* a program is supposed to do and not *how* it is supposed to do it can become reality. That way software engineers could create flexible programs, able to adopt to changes in the environment completely autonomously.

Service composition in general can be differentiated into *synthesis* and *orchestration*. *Synthesis* refers to *generating a plan* how to achieve a desired behavior by combining the abilities of multiple services. In contrast, *orchestration* refers to *coordinating the control and data flow among the various components* when

executing that plan. While our focus in this paper is on automatic synthesis, orchestration is an important problem that is complementary to synthesis. Examples of "service composition" approaches referring to orchestration include [2, 3]. Concerning synthesis the definition of service composition given above covers a number of different problems with distinct characteristics, each of which has been addressed by different approaches. This work's goal is to (a) identify the distinct classes of composition applications, (b) point out specific problems of each of those classes and (c) give an overview of existing approaches and their suitability for the individual classes.

A number of surveys on service compostion exist, however, none of these meet the goals listed above: Rao et al. [4] provide a quite comprehensive overview categorized according to the technique used, i.e. workflow techniques and various AI planning methods. Another survey solely centered around AI planning is [5]. In a more general overview Benatallah et al. [6] focus on workflow-based approaches for web service integration. Some of the considered approaches are abstracted in the form of software design patterns. Another survey on web service composition platforms is [7]. Milanovic et al. [8] provide an overview of different approaches for modelling composite services that are evaluated against a number of requirements to service composition modelling.

In contrast to all these surveys, this paper is structured around a classification of different *applications* of service composition and furthermore specifically focuses on *automatic* service composition. Given a service request that can't be met right away those existing automatic composition approaches are introduced that seem best suited to overcome the specific type of problem at hand. We believe that a categorization along this domain is the most helpful one from an engineering point of view. In detail we identify the following three classes of problems:

- **Fulfilling preconditions**: A service that can provide the desired effects exists, however, not all of this service's preconditions are met from the outset.
- **Generating multiple effects**: The request encodes multiple effects that are related, yet can be generated by different services.
- **Overcoming a lack of knowledge**: Additional knowledge gathering is necessary in order to correctly fulfill a request.

In the following Sections 2, 3 and 4 we elaborate on these different classes of service composition needs. In Section 5 we give a summary and conclude.

2 Fulfilling Preconditions

In an attempt to find an appropriate service providing a requested functionality it is a common situation to find a service that is able to generate the requested effects but some of its preconditions are not fulfilled. In these cases, "chaining" services can be used to overcome that problem. There might, e.g., be a request for a service that translates a given document from Chinese to German. If no such service exists a service that translates English documents to German can be used

in conjunction with another service that translates from Chinese to English. Such chaining can be understood as a way to compose services by recursively regarding the effects of one service as the preconditions of a following one until a desired effect is reached. The most common approaches to service chaining include graph search, forward chaining, backward chaining and estimated-regression planning. Each of them will be considered in turn.

Graph Search approaches rely on building a graph representation of all services available. In [9], Zhang et al. build a graph whose nodes represent the available services and whose edges encode whether one of a service's outputs may serve another service as one of its inputs. Edges are weighted according to the level of correspondence of the associated input and output. An adapted Bellman-Ford algorithm is used to find the shortest paths from the user's inputs to the expected outputs which represent the best available composition. The same idea has been used in an earlier work [10] with the restriction that services are only allowed to have a single input and a single output which simplifies the graph and the corresponding searching algorithm. A problem of this class of approaches is that they do not scale well with the number of offered services. For instance, the algorithm by Zhang et al. works in $O(n^3)$.

Forward Chaining is typically used by approaches rooted in logic-based planning systems. Starting with the available knowledge about the world and the preconditions and inputs from the service request the planning system uses applicable services, i.e. services whose preconditions can be met, in order to infer additional knowledge until the requested effects are fulfilled. An example of this approach is SWORD [11]. Another example is [12] which also exhibits the fundamental problem that forward chaining has: its undirected search. While the requested effects always have to be generated completely, in many cases it is not necessary to use all knowledge about the world in order to accomplish this. Thus, forward chaining tends to search in directions that are unnecessary for the requested effects[1].

Backward Chaining overcomes this problem. Backward chaining works similar to forward chaining with the difference that the composition starts with those services generating the requested effects instead of those whose preconditions are grantable. Any algorithm then recursively tries to create the necessary preconditions using other services until all preconditions are met. In [13], Sirin et al. use a semi-automatic approach to establish the composite service. Here, the user decides which service to add to the chain whenever more than one possibility exists. An example of a fully automatic algorithm using backward chaining is [14] where Sheshagiri uses heuristics[2] to pick the next service in the chain whenever there is a choice. While backward chaining generally implements a better directed search than forward chaining the problem space typically is still large.

[1] [12] tries to overcome this problem by including search control in the deduction rules of their planner.

[2] To give an example, one of the proposed heuristics is to use the service with the smallest number of preconditions.

Estimated-Regression Planning is an attempt to improve the performance of the search by implementing a forward chaining guided by a heuristic based on backward chaining. McDermott [15] applies this idea to the domain of web services. While he doesn't actually compose services but rather plans the interaction with a single web service (i.e., he plans an execution sequence of atomic interactions leading to the requested result) his results are applicable to the domain of service composition as well.

Concluding remarks. Generally, when implementing automated chaining of services, it should be noted that chaining may insert a degree of uncertainty regarding the semantic correctness of the generated composite service. In theory, a service that translates Chinese to German is equivalent to a composite service that translates Chinese to Japanese to English to German. In practice, each translation involves a loss of accuracy and, thus, the two services are far from equivalent. Although these conversion tasks are a natural application of chaining, to the best of our knowledge this problem hasn't been addressed so far.

Another problem of service chaining are cases where a user does not want a precondition of a service to be fulfilled due to side effects of that action. Take for instance a service that requires the user to have a valid credit card. Assuming one doesn't have a credit card a chaining algorithm might try to create a credit card contract in order to overcome this problem. Obviously that might be considered unacceptable. Consequently, a user's preferences need to be regarded when designing automated service chaining algorithms.

As a final remark it is noted that a large number of approaches to chaining is based on AI planning systems which generally leads to scalability problems in case of an increasing number of available services.

3 Generating Multiple Effects

Service requests like "I want to travel to Italy and need a hotel and a flight to a nearby airport" are characterized by the fact that multiple effects (flight and hotel booking) are requested. These effects are related (e.g. by the travel dates and by a shared location) but typically can be fulfilled by different services.

Chaining is used to provide a requested functionality that typically could be provided by a single service which simply is currently not available. In contrast, when dealing with multiple effects, service composition is somewhat more inherent to the request. Furthermore, when multiple effects are specified, the problem of composition synthesis is really a problem of decomposing the request into a collection of component requests each of which can be satisfied by a single service. Basically such service composition approaches can be divided into two categories. These categories differ in the granularity at which services are regarded. The first category, *behavior-based service composition*, relies on services being described by the their messages and the possible sequences thereof, i.e., the interaction with a service. In contrast, in the second category, *component-based service composition*, services are regarded as atomic, that is they can only be executed as a whole. Thus, in the first case, it is possible to combine parts of

services to new services, whereas in the second approach, only complete services can be combined to new ones. In addition to exploring approaches in these two categories, a closer look at at special subclass of requests with multiple effects, namely quantified requests, is worthwhile and will be provided later on in this section.

Behavior-Based Service Composition. Most approaches for modelling behavioral descriptions, especially those targeted towards composition, are based on finite state machines [16]. As an example, in [1], Berardi et al. describe all available component services as well as the request as finite state machines. Their goal is to find a composition of offered services that permits the interaction sequences given by the request. By using DPDL (Deterministic Propositional Dynamic Logic) the problem of service composition is reduced to the problem of satisfiability of a constructed formula. One of this approach's problems is that it doesn't scale well (needs $EXPTIME$). A related approach is described by Bultan et al. [17].

Component-Based Service Composition. The second category can be characterized by the use of decomposition frameworks that directly contain the component services to be used. A large body of research in this field uses some sort of rules to manage the decomposition of requests in a more or less automated fashion. As an example, Wu et al. [18] have implemented a hierarchical task network planner whose knowledge base contains methods representing the decomposition of higher level tasks. Similarly, McIlraith et al. [19] encode the decomposition using constructs of the logic programming language Golog, which is used for creating a sequence of services providing the requested functionality[3].

An approach based on data integration techniques is proposed by Thakkar et al. [20]. Here, the decomposition is implied by regarding the available services as views over a global schema. As a consequence, only information retrieval requests can be supported.

The work by Medjahed et al. [21] is an example of an approach in which the composition is contained in the client's request, given in a proprietary specification language CSSL (Composite Service Specification Language). Here, the exact sequence of desired operations (or to be more precise, of *descriptions* of the desired components) is specified in the request. The main contribution of this work is a composability model which captures information about whether available services that match the components' descriptions can be combined as specified in the request.

Quantified Requests. A special case of requests asking for multiple effects are quantified requests like "I want to download *all* publications of a certain author" where the same effect (download a publication) is requested multiple times. Such a request might be answered by a single service if that source has all publications available (e.g. a publication database) but often it will be neces-

[3] An interesting point about this work is that additionally chaining can sometimes be used to fulfill the preconditions of a component service.

sary to query multiple services (e.g. all common publishers) and to combine the results.

This kind of requests should not be confused with a request for "all services" providing a certain effect, like all services offering publications by the author. From a procedural point of view the difference is that in the case of a request for all services, the problem is in locating services, whereas in our case the problem is in composing services. From a semantics point of view, the result of the first request could contain duplicates of the same publication (provided by different service providers), which is not possible in the second case.

Here, a request's effect (or an effect's attribute) is quantified and the problem is to combine various services in order to cover the requested publications. The challenge is first to get to know which publications to look for and second to combine the available sources in an optimized way. To the best of our knowledge, this problem hasn't been addresses so far regarding service composition, even though it contains a large number of quite natural requests. There are approaches available that probably could be extended to cover this case, e.g. those rooting in first order logic planning systems like [12] but this remains as future work.

Concluding Remarks. One problem that needs to be dealt with when using composition in order to generate multiple effects is how to enforce transactional properties when invoking multiple services. In the context of the travel example one probably doesn't want to book a hotel anymore, if the booking of the flight fails. This problem has been addressed, e.g., by Casati et al. [22] or by the Web Service Transaction Specification [23]. It should also be noted that especially some of the earlier approaches listed above are based on semantically weak service descriptions (the same is true for approaches addressing service chaining). More recent approaches have broadened their focus to address this problem.

4 Dealing with Missing Knowledge

When searching for an appropriate service it may happen that knowledge necessary to choose the correct service is missing. Taking up the traveling example once more, assume the requested service is supposed to book a flight given a maximum price. Assume further that there is a service available booking flights with a certain airline by their flight number. This offer could be used – provided one knows how much each flight costs. Moreover, even if one has this knowledge, one still needs to know which flights are available.

But since assuming complete knowledge is obviously unsuitable for the Semantic Web [24], it is a basic requirement of any approach that aims at finding a (maybe composed) service matching a certain request that this approach is able to deal with incomplete knowledge. That is, the algorithm must be able to detect such a situation and implement procedures to acquire the missing knowledge.

One can distinguish two approaches addressing this problem: (a) one can make use of knowledge services **while searching** for a suitable service or (b) this search can be deferred until the time of execution. In this case a **conditional plan** has to be established. Note that in the first case, even if the planner returns

a single service sufficient for providing the requested effects, we consider this to be a form of service composition as the cooperation of multiple services is necessary and is only brought forward to planning time.

Creating Conditional Plans. Here, the knowledge gathering services are executed in conjunction with the remaining services and are used to navigate through the plan. In [15] McDermott proposes an extension of DPDL to model gaining information (and thereby to overcome the restriction of relying on perfect knowledge). Whenever the planner encounters a situation in which it lacks knowledge that can be learned it inserts an information gathering action into the plan. Branches are created for different outcomes of the information retrieval. The approach for building conditional plans taken by Martinez et al. [12] is based on a similar idea. One advantage of using conditional plans is that the time consumption at planning time of the alternative approach is dominated by the time spent waiting for knowledge gathering services to complete their execution which can be arbitrarily long. A further advantage is that knowledge services that have world altering effects like causing costs might not be tolerated at planning time. Finally, if one gathers information at planning time, this information may have become outdated at execution time [19][4]. The disadvantages of using conditional plans include a sometimes unrealistically high number of alternative paths. In the flight booking example for instance, one would have to create a path for each available service offering flights to the target location in case that all other services are booked up. In this case the number of paths is determined by the number of available services. This doesn't scale well. Additionally, having to consider a large number of alternative paths leads to an increased time consumption when creating the plan. These disadvantages lead to approaches that retrieve information at planning time.

Searching While Planning. SHOP2 [18] is an example of this approach. Wu et al. designed their planner's decomposition rules such that the planner is able to distinguish between component services gathering information and others, so that the former ones can be executed at planning time. While in this work the mechanism to determine *when* to ask for *what* information can be regarded as "hard wired" in the rules, in a follow up work the authors remedied this leading to automatic recognition that information is missing [25]. An approach to gathering information similar to that of SHOP2 is taken by McIlraith et al. [19].

Concluding Remarks. A fundamental problem of approaches aiming at *automatically* recognizing that information is missing: there needs to be a mechanism to determine whether the available knowledge is complete or not. In the flight booking example it would be necessary to know when the found set of flights is complete to be able to pick the cheapest flight with certainty. The problem here is to know when to stop searching for more information. This problem has been, for instance, addressed by [24].

[4] McIlraith et al. address this problem using a monitor that repeats the knowledge gathering at execution time.

5 Summary and Conclusion

In this paper we have analyzed the different needs that require the ability to (automatically) compose services and have identified three distinct classes:

First of all, service chaining addresses the problem of *fulfilling a service's preconditions* by using another service. Solutions to this problem have to take into consideration that the chaining of services may insert a degree of uncertainty and that automated chaining is sometimes unacceptable for the user.

A second class of service composition approaches aims at *generating multiple effects* that are somehow related. As a special case we identified requests containing a quantification of some attributes. Problems that have to be addressed by approaches of this class as well as by approaches of the former one include ensuring scalability in the number of offered services and establishing transactional properties of the service composition.

Finally, the problem of *incomplete knowledge* can be solved using service composition by adding knowledge gathering services to acquire it. The main difficulty here is how to decide when to stop searching for additional knowledge.

Our overview has shown that while quite a number of approaches to service composition exist, it is necessary to take a close look at the requirements of ones particular situation to find a suitable approach. Also, while many problems concerning service composition have been solved, there remain important open issues that need to be addressed in order for service composition to become a feasible tool. Among the most important of these issues are the degrading quality in long service chains, the lack of transactional properties of composed services, scalability with an increasing number of offered services, the foundation of most approaches on semantically weak service descriptions, the lack of approaches to allow for quantified requests and an appropriate treatment of missing knowledge.

References

1. Berardi, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Mecella, M.: Automatic composition of e-services that export their behavior. In: Proc. of 1st Int. Conf. on Service Oriented Computing (ICSOC-03), Trento, Italy. (2003)
2. Benatallah, B., Sheng, Q.Z., Ngu, A.H.H., Dumas, M.: Declarative composition and peer-to-peer provisioning of dynamic web services. In: Proc. of the 18th Int. Conf. on Data Engineering (ICDE'02), San Jose, CA, USA. (2002)
3. Pistore, M., Bertoli, P., Barbon, F., Shaparau, D., Traverso, P.: Planning and monitoring web service composition. In: Proc. of the 14th Int. Conf. on Automated Planning and Scheduling (ICAPS 2004), Whistler, BC, Canada. (2004)
4. Rao, J., Su, X.: A survey of automated web service composition methods. In: Proc. of the 1st Int. Workshop on Semantic Web Services and Web Process Composition, SWSWPC2004, LNCS, San Diego, USA. (2004)
5. Peer, J.: Web service composition as AI planning - a survey. Technical report, Univ. of St. Gallen, Switzerland (2005)
6. Benatallah, B., Dumas, M., Fauvet, M.C., Rabhi, F.A.: Towards patterns of web services composition. In: Patterns and skeletons for parallel and distributed computing, London, UK, Springer-Verlag (2003) 265–296

7. Dustdar, S., Schreiner, W.: A survey on web services composition. Int. J. Web and Grid Services **1** (2005) 1–30
8. Milanovic, N., Malek, M.: Current solutions for web service composition. Internet Computing, IEEE **8** (2004)
9. Zhang, R., Arpinar, I.B., Aleman-Meza, B.: Automatic composition of semantic web services. In: Proc. of the 2003 Int. Conf. on Web Services (ICWS'03), Las Vegas, NV, USA. (2003)
10. Mao, Z.M., Brewer, E.A., Katz, R.H.: Fault-tolerant, scalable, wide-area internet service composition. Technical Report UCB//CSD-01-1129, University of California, Berkeley, USA (2001)
11. Ponnekanti, S.R., Fox, A.: SWORD: A developer toolkit for web service composition. In: Proc. of the 11th Int. WWW Conf. (WWW2002), Honolulu, HI, USA. (2002)
12. Martínez, E., Lespérance, Y.: Web service composition as a planning task: Experiments using knowledge-based planning. In: Proc. of the 14th Int. Conf. on Automated Planning and Scheduling (ICAPS 2004), Whistler, BC, Canada. (2004)
13. Sirin, E., Hendler, J.A., Parsia, B.: Semi-automatic composition of web services using semantic descriptions. In: Proc. of the 1st Workshop on Web Services: Modeling, Architecture and Infrastructure (WSMAI'03), Angers, France. (2003)
14. Sheshagiri, M.: Automatic composition and invocation of semantic web services. Master's thesis, University of Maryland, Baltimore County, USA (2004)
15. McDermott, D.V.: Estimated-regression planning for interactions with web services. In: Proc. of the 6th Int. Conf. on Artificial Intelligence Planning Systems (AIPS'02), Toulouse, France. (2002)
16. Hull, R., Benedikt, M., Christophides, V., Su, J.: E-services: a look behind the curtain. In: Proc. of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'03), San Diego, CA, USA. (2003)
17. Bultan, T., Fu, X., Hull, R., Su, J.: Conversation specification: a new approach to design and analysis of e-service composition. In: Proc. of the 12th Int. Conf. on World Wide Web (WWW'03), Budapest, Hungary. (2003)
18. Wu, D., Parsia, B., Sirin, E., Hendler, J.A., Nau, D.S.: Automating DAML-S web services composition using SHOP2. In: Proc.of the 2nd Int. Semantic Web Conf.(ISWC2003), Sanibel Island, FL, USA. (2003)
19. McIlraith, S.A., Son, T.C.: Adapting golog for composition of semantic web services. In: Proc. of the 8th Int. Conf. on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France. (2002)
20. Thakkar, S., Knoblock, C.A., Ambite, J.L.: A view integration approach to dynamic composition of web services. In: Proc. of the 13th Int. Conf. on Automated Planning and Scheduling (ICAPS'03), Trento, Italy. (2003)
21. Medjahed, B., Bouguettaya, A., Elmagarmid, A.K.: Composing web services on the semantic web. The VLDB Journal **12** (2003) 333–351
22. Casati, F., Ilnicki, S., jie Jin, L., Krishnamoorthy, V., Shan, M.C.: Adaptive and dynamic service composition in *eflow*. In: Proc. of the 12th Conf. on Advanced Information Systems Engineering (CAiSE*00), Stockholm, Sweden. (2000)
23. IBM, BEA Systems, Microsoft, Arjuna, Hitachi, IONA: Web service transaction specification (2005)
24. Heflin, J., Munoz-Avila, H.: Lcw-based agent planning for the semantic web. In: Proc. of the 18th Nat. Conf. on Artificial Intelligence (AAAI2002), Edmonton, AB, Canada. (2002)
25. Kuter, U., Sirin, E., Nau, D.S., Parsia, B., Hendler, J.A.: Information gathering during planning for web service composition. In: Proc. of the 3rd Int. Semantic Web Conf. (ISWC2004), Hiroshima, Japan. (2004)