A Backwards Composition Context Based Service Selection Approach for Service Composition

Hong Qing Yu and Stephan Reiff-Marganiec
Department of Computer Science
University of Leicester
Leicester, United Kingdom
Email: {hqy1,srm13}@le.ac.uk

***Abstract*—In SOA applications are built from individual services offered by different providers. Typically an application comprises of several such services usually stemming from different providers leading to the question of which services to select and compose. We present the new concept of composition context together with a novel service selection algorithm. The approach has been evaluated in our test bed and shows good scalability.**

I. Introduction

In SOA (Service-Oriented Architecture), services are defined as "self-contained, self-describing, modular applications that can be published, located and invoke across the Web" [TRPA06]. These characteristics allow services to be selected and composed easily at software design time. However, one ultimate goal of SOA is to go beyond design time to dynamically compose service at runtime. With the Semantic Web [FLP$^+$06] and the Business Process Specification Language (e.g. BPEL [Org07]) developing, achieving this goal is becoming more and more realistic. Meanwhile, many challenges are still existing on the road. One of the most important question is what kind of information should be considered for service selection and composition at runtime. Currently, researchers suggest three aspects of information: Service IOPE [CV04], Service QoS (e.g. [JWJY08], [WU$^+$08]) and Business rules/policies (e.g. [OYP03], [SGS09]). In this paper, we introduce a new aspect, *composition context* information, to drive runtime service selection and composition. The composition context includes service non-Functional properties, the collaboration and communication history between services and business collaboration constraints between different organizations. Since the composition context has its own features, which are different from other information aspects, it introduces some unique research issues. The main contributions of this paper are:

- We introduce a new concept of composition context and its classifications.
- We develop a backwards composition context based service selection approach (BCCbSS) for service composition.
- We adopt the Type-based Logic Scoring Preference Extension (TLE) service selection method [YRM08] inside the service composition approach.

The rest of the paper is organized as follows. We firstly study two service composition scenarios in detail to discuss what kind of information affects the service composition and then present the composition context in section II. Then the research problem and challenges are clearly defined in section III. The composition context based service selection mechanism is illustrated in section IV. The evaluation results are discussed in section V. Finally, related work, conclusion and future work are drawn in section VI and section VII.

II. Defining Composition Context

Let us study the real world service composition scenarios of organizing a meeting and planning a trip. The main purpose of studying these scenarios is to understand in more detail about what information influences the selection of services in conjunction with each other. Clearly, there is a need to organize the results: we will call the resulting structure *composition context*. We consider two scenarios as the introduce different aspects relevant to composition. We could of course create an artificial scenario with all aspects, but we feel it better to use real examples from our industry partners in the inContext project.

A. Organizing a meeting

A meeting is required to be held for discussing a plan to deal with an emergency [TRMY07]. Organizing a meeting involves a series of tasks. The tasks include searching suitable participants inside the organization, finding a suitable date, booking a meeting room and sending invitation notifications to the participants. The meeting organizer integrates these tasks as a workflow template (see Figure 1). Each task can be performed by a service.

1) The participant search task that requests finding suitable participants who are in the organization can be performed by a people-search service. There are two available people-search services offered by different providers. Both services have the same function of taking people requirement attributes, such as skills, experiences and positions to produce a list of people as output. However, these two services have different NFPs. One service can find the people who are in the organizer's organization and is more accurate by having access to more information about people. The other can search people who are both inside and outside

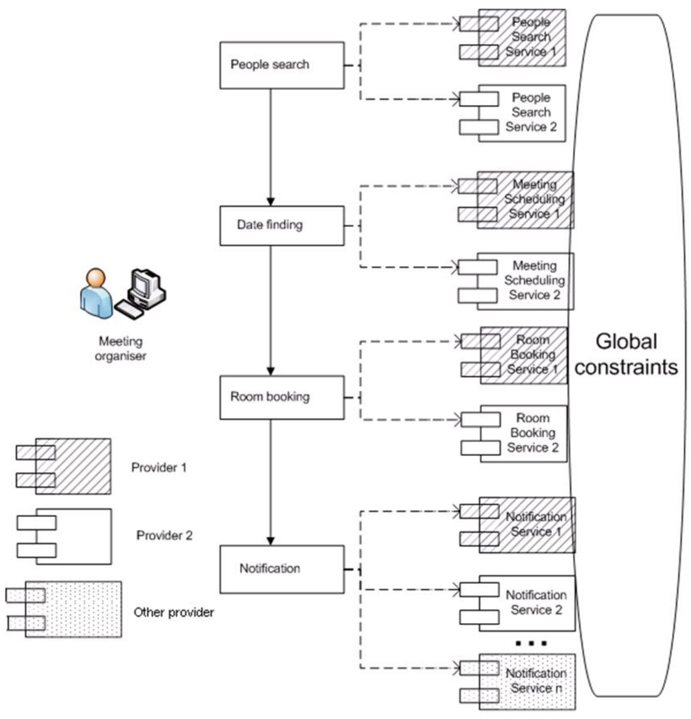


Fig. 1. Workflow of organizing a meeting

the organization, but it is less accurate. Also, the first service's response speed is slower than the second one.

2) The date finding task can be completed by the meeting scheduling service. Again, there are two scheduling services available offered by different providers. They both use people's calendars' URL addresses as input and return the most suitable date for all involved people as output. One scheduling service only has ability to check Google and MSN online calendar systems and supports around 90% optimal dates (e.g. 9 people out of 10 are available on the scheduled date). The other service has ability to check all kinds of current existing online calendar systems and supports around 70% optimal date.
3) The room booking task can be executed by room booking services. The booking service takes the date and facility requirements as input and produces the place address and room information as output. There are two booking services available. One service supports booking rooms with normal meeting facilities. The other service supports booking rooms with both normal facilities and advanced equipment.
4) The notification task can be performed by the notification services. There are many services available. We already discussed the notification services as the first case study of the single service selection scenario.

1. Local constraints: The meeting organizer invokes the workflow template. For task 1, a people search service is required and two services are discovered. Local constraints are a set of requirements for the service's NFPs. "Local" means the requirements are individual considerations for each type of the services. These requirements could either be hard criteria or soft criteria. Each requirement has a weight for prioritizing. For instance, the organizer has preferences stating that accuracy of the search is more important than speed, only participants within the organization are acceptable. Therefore, the first service (from provider 1) can be found better than second one (from provider 2).

2. Invocation error context: Supposing a service from provider 1 has been selected based on the local constraints (e.g. accuracy and speed). Invoking the selected service produces an invocation error, and hence, the other service from provider 2 has to be used instead. If this error can be saved as context information and retrieved for future service compositions, it can reduce the composition time and increase the composition reliability. Thus, invocation error history is related to service composition at runtime.

3. Coordination context: We assume the "people search service" from provider 2 has been selected for Task 1. For the current Task 2 of date finding, all target participants use online Google calendars. The only local preference is the optimal rate. When only considering the local constraints on their own, it is easy to see that the service from provider 1 is the better one because its optimal rate is better. However, it known that the selected people search service has more coordination failures with the scheduling service of provider 1 than that offered by provider 2, a fact is learned from historical composition records available in the context. Taking this into account it is more difficult to decide which service is better.

4. provider distance: We suppose the date finding service from provider 2 has been selected, thus the previous two tasks are performed by the same provider. For Task 3, the aim is to book a room with some equipments. Since both services are qualified for the local constraints, A service can be selected at random as there is no other user's preference. However, the history of coordination activities shows that services from the same provider have a more efficient coordination rate, then service from provider 2 may be better because it has provided the previous two services.

B. Planning a trip

Let us consider another typical workflow example: planning a trip. Generally, the planning activity requires three tasks of booking transports, purchasing travel insurance and booking hotels as shown in Figure 2. Moreover, purchasing insurance and booking hotel are two independent tasks but both rely on the transport date and time.

Because many travel related services are available, the competition is tight.

1) There are many different transport services available, the local constraints are faster speed and cheaper price (price refers to the service fee, not tickets price or other buying price through out this dissertation).
2) There are many insurance service as well, the local constraints are cheaper service fee and better service reputation.

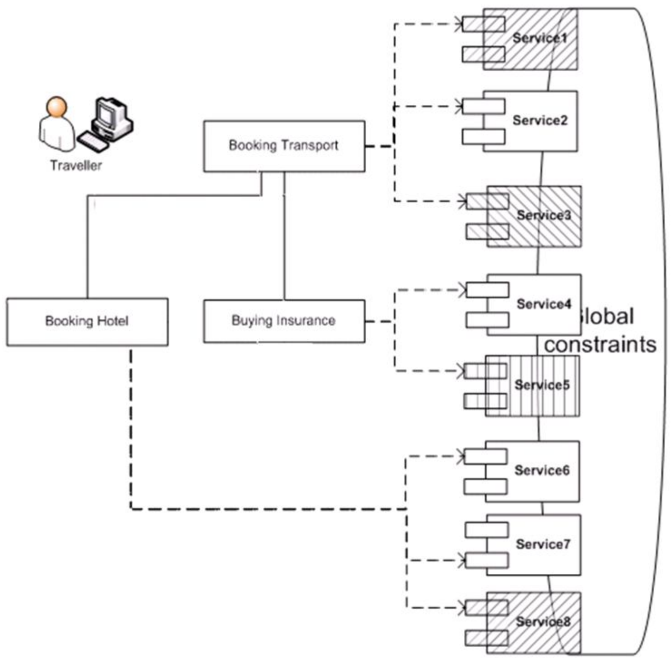


Fig. 2. Workflow of planning a travel

3) There are also many hotel booking services, the local constraints are the place is covered by the service, between 3 stars and 4 stars hotel, economic and good reputation.

5: Allowance policy

We assume that airline service A has been selected for the booking transport task based on the user local context constraints of covered locations, faster service response and cheaper service fee. For the hotel booking task, some business corporation policies are also applicable. It is useful to make selections depending on the previous selected services' corporation policy. The following two are examples.

- It is not allowed to continue invoking the service more than 10 times in one workflow.
- Provider A does not allow its services to be used by services owned by provider B.

6. Cost policy: In the commercial market, cost is an important factor to be considered from the global point of view. The aim of using the cost policy context is to find the cheapest composite service, which does not necessarily mean every single service is the cheapest one because the coordination among different providers has different prices. This is the main difference between local cost constraint and global cost constraint. For example, one insurance service A takes $\pounds 10$ for traveling with airline service but the airline gives $\pounds 8$ discount for working with insurance service A. The other insurance service takes $\pounds 5$ for traveling with the same airline service A. Therefore, the first service is likely selected from global point of view.

7. Composition time: Time is also a crucial factor for the business. The time can be considered global and local. The local view focuses on the individual service response time. In contrast, the global view concentrates on the overall composite service response time. For example, one insurance service takes more time to complete with airline A. However, the other one will take less time to complete with the same airline. As results, it is more reasonable to select the second one, if we considering the composition time constraint.

TABLE I
COMPOSITION CONTEXT CLASSIFICATIONS

Context categories	Context elements	Type
Execution context	Execution error rate	Numerical
	Connection error rate	Numerical
Coordination context	Provider distance	Boolean
	Coordination time	Numerical
	Physical distance	Numerical
Composition policy context	Special cost	Numerical
	Allowance	Boolean
	Times of uses	Numerical

C. Composition Context

Based on the presented case studies, we define eight composition context constraints in three categories: execution context, coordination context and composition policy context (see Table I).

Remember that composition context contains only the data related to explore how services behave with each other and hence how desirable their composition might be; non-functional aspects and matching user requirements for each specific service is conducted in the local evaluation of each individual service selection.

The composition context focuses on the context information which will affect service composition. We believe that the three top-level categories are complete, as information either is related to events occurring during service execution, is given by the static relation between two services or can be influenced by business decisions.

We do not claim the elements defined inside the categories are complete, but they have shown sufficient for the case studies encountered in our work. Further elements can be added if needed and they should not affect the feasibility of our selection mechanism.

Analyzing the 3 groups of context, we find that composition context can also be separated in dynamic context and static context. The dynamic context (e.g. coordination time, execution context of execution error rate and connecting error rate) means the context changes very frequently. Thus, dynamic context needs to be detected, calculated and stored at runtime. Static context refers to the composition information which does not change frequently (e.g. provider distance and special cost).

III. RESEARCH PROBLEM AND CHALLENGES

Definition 1: Given a set of subtasks, defined in a service composition template, the composition context aware service selection problem is to dynamically and efficiently find the most suitable set of executed services for completing each

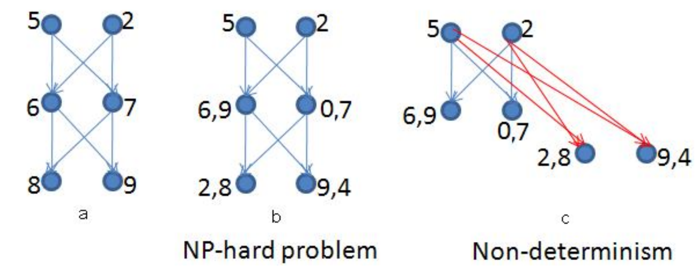


Fig. 3. Composition complexity analysis

subtask.

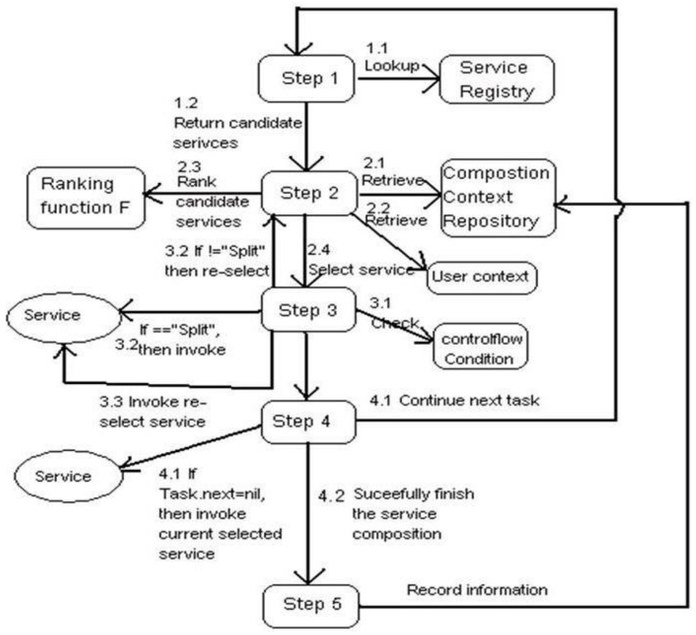
What exactly is *most suitable* depends on three external factors:

- the composition context constraints which defined in the Table I,
- the user context constraints for selecting individual service for a sub-task, and
- the services' runtime context information.

More detailed information about user context constraints can be found in [YRM09]. The composition context aware service selection problem raises some important issues:

(1) **The balance between globally optimal and locally optimal solutions** is an important issue to achieve. Unlike general global optimization problems, composition context aware service selection requires to not only consider global composition context constraints but also user's local context constraints. For example, in order to send a suitable message to a meeting participant (the last sub-task for organizing a meeting service composition), based on the current context of a user (the user currently has a mobile phone but is without Internet connection), an SMS message sending service is required. If we only consider composition context from global optimization point of view we may choose an unusable service to the user right now. Therefore, we need to select the service which should satisfy both sides of the optimization. Otherwise, the selection result may not correct.

(2) **The balance between complexity/efficiency and correctness.** Composition context information is a dynamic and multiple value constraint that is a difficult global optimization problem. Normally, global optimization problems have a fixed value. For example, if each node has a fixed value in a graph when computing a cheapest path problem (see Figure 3.a), then we can use a greedy algorithm to find the cheapest solution. However, in our case, the service composition context data is different for each service. For example, the composition price is £6 between Service1 and Service2, but is free between Service1 and Service3 (see Figure 3.b). Additionally we have to consider the multiple constraint dimensions, and hence the global optimization becomes infeasible. Therefore, existing global optimization solutions are unsuitable for the dynamic composition context-aware service selection. Finding a service composition solution with low complexity and efficiency is required.


Step 1
1.1 Lookup
Service Registry
1.2 Return candidate serivces
Ranking function F
2.3 Rank candidate services
Step 2
2.1 Retrieve
2.2 Retrieve
Compostion Context Repository
3.2 If !="Split" then re-select
2.4 Select service
User context
Service
Step 3
If =="Split", 3.2 then invoke
3.1 Check
controlflow Condition
3.3 Invoke re-select service
Step 4
4.1 Continue next task
Service
4.1 If Task.next=nil, then invoke current selected service
4.2 Suceefully finish the service composition
Step 5
Record information
Fig. 4. The BCCbSS approach

(3) **Control flow structure affects the global optimal strategy**. The other unusual global optimization issue is that the composition specification is a workflow with control flow structures such as sequence, parallel (and) and split (or). Especially, the split control flow decides the possible choice of workflow paths, which means that only when a service is selected and invoked, the next workflow path can be determined based on an assessment of the output data. However, the runtime service output cannot be predicted. As result, an upfront selection (rather than during execution) has a chance to be completely wrong and would need to be recomputed after each split (see also Figure 3.c).

IV. Our Solution: The BCCbSS approach

A. The approach

We developed a backwards composition context based service selection approach (BCCbSS). The basic idea is to always go back one step to check if the currently selected services are the best composition in the light of current existing composition knowledge and invoke the selected service as soon as possible. The whole process is shown in Figure 4).

Step 1: *Searching and returning all candidate services from registry for the current request task in the composition workflow.* This step will ensure that service provide the right functionality (we have not considered interface mediation in our work, but existing work in this area can be used). For simplifying the explanation, we only use *Special Cost* as example composition context to demonstrate the selection and composition process which shows in Figure 5.

Step 2: *Invoking the ranking function F (the function will be discuss in next subsection) to give a fixed evaluation value*

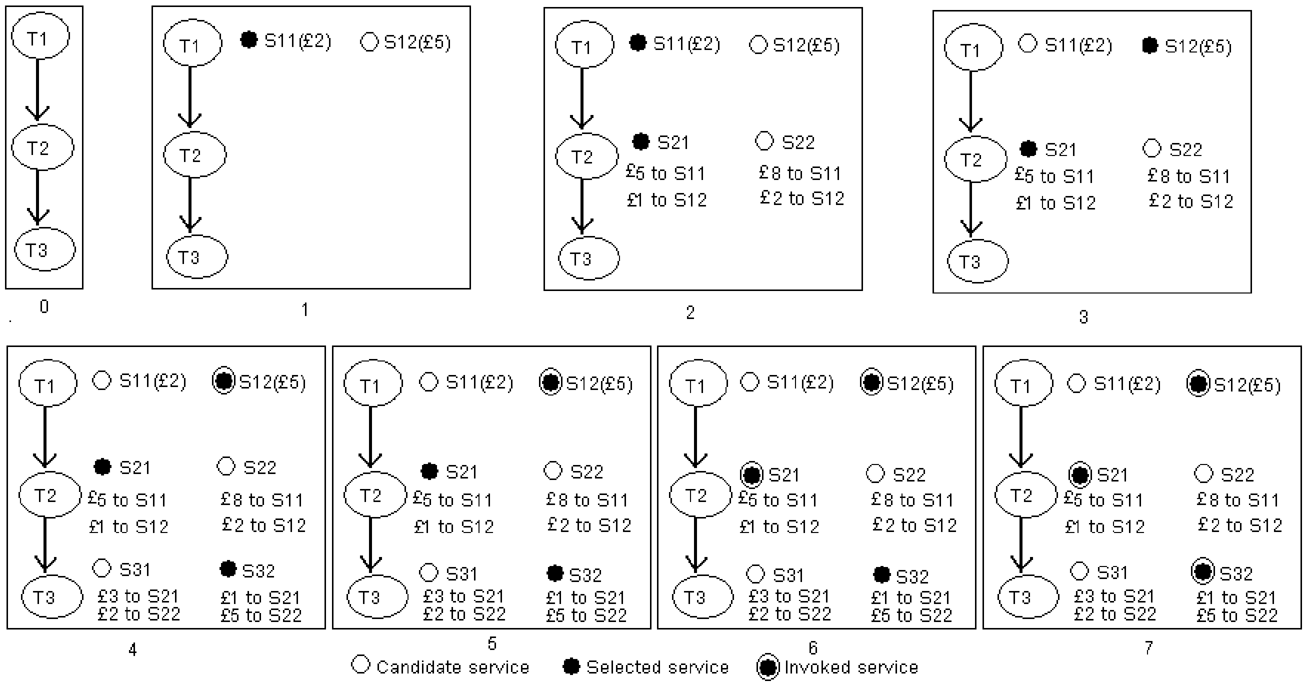


Fig. 5. Demonstrated example for composing a cheapest composite service

to each candidate service by considering the user constraints and current context information, composition context criteria of selected service for previous task and next task. If there is no previous selected service and no next selected service, the ranking function only bases on the current user's requirements. As Figure 5.1 shows that 2 candidate services (S11 and S12) are discovered for T1, where S11 is $\pounds 2$ cost and S22 is $\pounds 5$. On this step, there is no composition context is available, so S11 is selected.

Step 3: *If the next control workflow is "split", then invoke the current selected service from previous step. Otherwise re-select the previous service if it exist (otherwise it continues to select next task from Step 1) and has not been invoked, then invoke the re-selected service. If an error occurs when the service is invoked, then record this error information into the composition context store. Restart Step 3 to select the next best service.* Because T1 is the first task, then we go back to Step 1, 2. 2 candidate services (S21 and S22) are discovered for T2, where S21 requires $\pounds 5$ to be composed with S11 and $\pounds 1$ with S12; S22 requires $\pounds 8$ to be composed with S11 and $\pounds 2$ with S12. Based on the first selected service S11, the S21 is selected (see Figure 5.2). Now, we find that there is a previous selected service existing, then we re-select the service for T1 based on its next selected service S21 to make sure the best possible combination between T1 and T2. As result, S12 is calculated as the better service than S11 in this process and S12 is invoked (see Figure 5.3). The same process happens to T3 as well as shows in Figure 5.4, 5.5 and 5.6.

Step 4: *If current task is the last task in the composition workflow, then invoke the current selected service and finish the whole thing. Otherwise, it goes back to step 1 with next required task.* (See Figure 5.7)

Step 5: *If the invocation finishes successfully, log the execution details to the context store. Move to next activity and return to Step 1.*

B. BCCbSS optimization using TLE service selection method

In [YRM08], we introduced the Type-based LSP Extension (TLE) service selection method to solve the multiple criteria based service evaluation problem. In order to addressing the composition context based global optimization issue, we define a global ranking function F by adopting the TLE method and using following definitions:

$E_{1.1}$ = Soft local optimization criteria considering user requirements and derived from user context;

$E_{1.2}$ = Soft global optimization criteria amended and derived from composition context related to the previous selected service (if the previous selected service does not exist, then $E_{1.2} = 0$) for the task in the workflow;

$E_{1.3}$ = Soft global optimization criteria amended and derived from composition context related to the next selected service (if the next selected service does not exist, then $E_{1.3} = 0$)for the task in the workflow;

$$E_1 = (\mid W_1 \mid E_{1.1}^r + \mid W_2 \mid E_{1.2}^r + \mid W_3 \mid E_{1.3}^r)^{1/r}, \qquad (1)$$

where E_1 aggregates all the soft optimization criteria, r=1, $\mid W_1 \mid = \mid W_2 \mid = \mid W_3 \mid = 1/3$.

E_2 = Hard optimization criteria (including both global and local context) represent all mandatory requirements which must be satisfied. Any hard criterion evaluating to 0 will lead to an aggregation result of 0. The soft criteria handle all other preferences. Finally,

$$F = (\mid W_1 \mid E_1^r + \mid W_2 \mid E_2^r)^{1/r}, \tag{2}$$

where r=-0.72, $\mid W_1 \mid = \mid W_2 \mid = 0.5$.

$E_{1.2}$ is used to evaluate the composition context; the result of the evaluation will then be merged with the local score (the suitability of the service for the user's needs) when computing the overall score.

C. Contributions of BCCbSS

The overall services composition approach can be considered as a sequence of service selections and the length of that sequence depends on the number of tasks in the workflow template. When the composition process starts, there are only local constraints (user context) available as the composition context is empty because we do not know which set of services will be relevant to the current candidate services. After the first service is selected, the composition constraints will be considered based on the first selected service's composition context. With more and more services being selected, the composition context will become richer in information. The data of the composition context are gained through addition of facts from observation and the history. The BCCbSS approach has following advanced characteristics:

1) The approach performs the selection and invocation step by step. Some research work [CPEV05], [ZBN+05], [YL05] suggests completing a service composition by selecting all the services for the whole workflow template. However, this is not an efficient way, if there are many tasks involved in the workflow and many candidate services are available for each of them. Taking the organizing meeting scenario as an example, if there are 4 services for each of the 4 tasks, then the selection method has to compare the totally 256 different composition solutions for identifying the correct service composition choice. With more service available, the state explosion problem will affect the approach efficiency and scalability. The step by step strategy can essentially avoid such a problem because each step has only to consider a small number of the services, which is the number of the available services for the task, previous selected or invoked service and next selected service (only in re-selection process). Take the same example, the step by step strategy only needs to consider $4 + (5 + 5 + 5) + 5 + (5 + 5 + 5) + 5 = 44$ different selection solutions.
2) The approach can guide the selection method to make a choice based on existing knowledge of the composition context. In the organizing meeting scenario, when the people search service from Provider 1 has been selected for task 1, the rest of the selection tasks should only consider the composition context related to the selected service because other people search services' composition context is no longer useful.
3) The approach considers not only local constraints (user context) but also the composition context as inputs for the selection method. The local constraints specify the user's preferences for the individual service, e.g. quality, execution duration and prices. Since it is difficult for user to judge the global view of the composition service, the composition context should be automatically applied. The composition context or global constraints consider the interaction and composition properties among the selected services and available services for the current task.
4) The approach is a run-time approach. The user's preferences may frequently change according to his/her current status, as will the service's NFPs and composition context. These dynamic features require a run-time composition approach rather than a design-time composition approach. The run-time composition approach can make sure that the composite solution is the most suitable one for the current user's status and composition context.
5) The approach is fault tolerant. When a selected service can not perform correctly in a certain step, the approach allows for the next best service to substitute the current selected one in order to complete the composition task. Fault tolerance is very important for a run-time approach and real service composition scenarios. It increases the likelihood of successful completion of the workflow. In the example of planning a trip, when the air ticket has been booked, the composition requires the insurance purchase to be successful, because the previous step is costly and irreversible.

V. Evaluation

We evaluated the process by analyzing a number of test scenarios including the following (note that we have added values to all 8 aspects of the composition composition context):

1) Increasing the number of services available for each step of a three step workflow.
2) Increasing the length of the workflow template keeping the number of services available for each step static.

In the first scenario we combine the 8 composition context criteria and increasing numbers of services from 2^1 to 2^8 for each of the 3 steps. Figure 6 represents that the method is quite efficient to deal with up to 250 services in this situation. The results suggest that method is linear with respect to available services.

The second scenario is designed to test the scalability with regard to composition steps. There are again 8 composition context criteria and there are 4 services for each step. The test results in Figure 7 show that the method works efficiently with 40 steps. Again, we have a linear increase in run-time when the workflow length increases.

There are three observations that should be made here: (1) it is unusual that workflows are much longer than 40 steps,

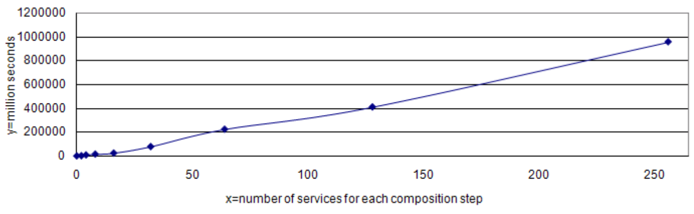


Fig. 6. Evaluation results for composition selection test case 2

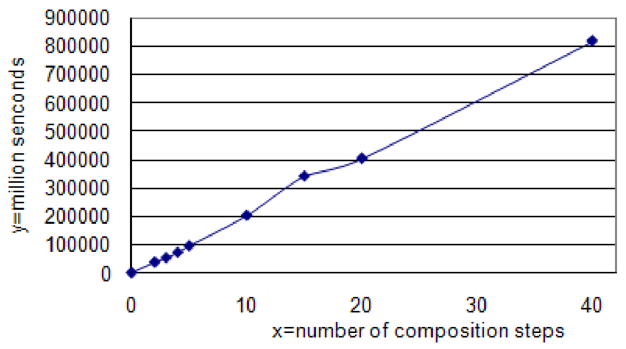


Fig. 7. Evaluation results for composition selection test case 3

(2) the execution of the actual services will also be time-consuming and some might be long-running services where it is more crucial that the right service is selected than that it is selected more quickly and (3) the method does interleave execution and selection, so a user does not have to wait until the selection mechanism has completed before the execution of the workflow can start.

From the conducted experimentation, we can conclude that the method is quite efficient as run-time increases are linear with respect to the increase in workflow length as well as an increase in the number of services.

VI. Related Work

Two kinds of service selection approaches are developed for Web service composition problem, which are local optimal selection and global optimal selection.

Local optimization based service selection refers to selection methods which only take certain selection constrains related to the current activity in the workflow without specifying and considering the constraints implied by the workflow context and the consequences that the choice will have on later activities. For example, a policy based BPEL workflow Web service selection method is presented in [KHC+05]. It extends BPEL for run-time adaptation of service by adding the policy reference to each node. The policy documents provide the local optimization rules which are independent from each other. The service selection process is applied at each node separately. A similar approach was also presented in the earlier e-Flow project [CIJ+00]. The biggest advantage of the local optimization methods is efficiency in selection time - the worst case can be solved in polynomial time. However, it does not necessarily select the optimal or even close to optimal service in the global composition context.

Global optimization based service selection, on the other hand, takes the global selection constraints to select a group of services rather than one service for a node in the composition workflow. The key assumption of this strategy is that all suitable services for each node have already been discovered and are inside the global optimization search space. [CPEV05], [ZBN+05] are two example approaches. By studying these approaches, we find they surely narrow the disadvantages pointed out for local optimization. However, they introduce their own problems.

- Low scalability: In general, multi-QoS constrained service selection with optimization is an NP-complete problem [YL05], which reduces scalability of the methods.
- Lack of fault tolerance: Global optimization methods return a set of combined services as the final solution package. However, if one service is not available or throws an exception at run-time, then the whole solution package fails.
- Low flexibility: Global optimization methods need to know all constraints at design time. However, some selection constraints are only known when certain data is produced at run-time. For example, considering a conditional choice in a composition, the complete global constraints are available only after the condition is evaluated.
- Lack of reflection to local constraints that are important to reflect user context.

In contrast, the BCCbSS approach does not need to predict all the global constraints in advance. It makes the selection decisions activity by activity based on the currently existing local and global composition context. The composition context is growing as we proceed through the activities. Based on these context constraints, we may select the best service according to real-time knowledge for the next activity. As we continue to select services, the composition context grows allowing for more fine-grained selection.

Some people may argue that the knowledge for selecting the first service probably is empty and hence we will not select the best one without knowing the forward selection context when the next control flow is “Split” . While this is true, in practice it is impossible to predict the execution path as this is going to be influenced by runtime data. Furthermore, we should not make a decision relying on predicted knowledge which has large chance to be wrong. For example, when selecting the best service for the first task, one does not know all the currently available services for the later stages. Therefore, we have to make the service choices only based on certain knowledge that are the user’s context constraints.

VII. Conclusion

Selecting the most suitable services to complete a complex composite service is an important research topic. Industrial scenarios requests to consider composition context which introduces some new challenges for service selection and composition.

By studying the real world scenarios, we introduced the concept of the composition context which is divided into 3 classes and 8 specific elements. Based on the composition context, we presented a novel Backward Composition Context based Service Selection (BCCbSS) approach to meet the composition context aware challenges. The BCCbSS composition process fully considers composition context factor by adopting the TLE service selection method.

Comparing the approach to the context-aware service composition requirements and other composition approaches, our approach has several advantages:

The approach is a fault tolerant step by step process. The method scales well for large workflow as well as large numbers of services, as the ranking considers only services for the current task and has access for the wider workflow condition through the composition context. The selected services are dynamic bound to and invoked at run-time rather than statically bound at design time.

For the future work, we are going to investigate more service composition scenarios in order to obtain a better understanding about the completeness of the composition context. Moreover, we are going to compare the BCCbSS approach to other planning approaches on complexity and adaptability to see whether our approach can also be used for other selection or optimization problems.

ACKNOWLEDGMENT

This work is partially supported by the EU projects inContext IST-2006-034718 and SENSORIA IST-2005-16004.

REFERENCES

[CIJ+00] F. Casati, S. Ilnicki, L. Jin, V. krishnamoorthy, and M. C. Shan. *Adptive and Dynamic Service Compostion in eFlow*. HP Laboratories Technique report, 2000.

[CPEV05] G. Canfora, M. D. PentaRaffaele, R. Esposito, and M. L Villani. *An approach for QoS-aware service composition based on genetic algorithms*. Proceedings of the 2005 conference on Genetic and evolutionary computation, SESSION: Search-based software engineering table of contents, pp. 1069-1075, 2005.

[CV04] I. Congiu and G. Valetto. Using owl-s iopes to drive services selection and composition according to user preference. In *Proceeding of Semantic Web Services Workshops at ISWC*, 2004.

[FLP+06] D. Fensel, H. Lausen, A. Polleres, J. De Bruijin, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services*. Springer, 2006.

[JWJY08] C. Jin, M. Wu, T. Jiang, and J. Ying. Combine automatic and manual process on web service selection and composition to support qos. In *12th International Conference on Computer Supported Cooperative Work in Design CSCWD*, pages 459–464. IEEE, 2008.

[KHC+05] D. Karastoyanova, A. Houspanossian, M. Cilia, F. Leymann, and A. Buchmann. *Extending BPEL for Run Time Adaptability*. Proceedings of the 9th IEEE International EDOC Enterprise Computing Conference, pp. 15-26, 2005.

[Org07] Oasis Organization. *Web Services Business Process Execution Language Version 2.0 - Primer*. http://docs.oasis-open.org/wsbpel/2.0/Primer/wsbpel-v2.0-Primer.pdf, 2007.

[OYP03] B. Orriens, J. Yang, and M.P. Papazoglou. A framework for business rule driven web service composition. Springer-Verlag Berlin Heidelberg, 2003.

[SGS09] S Reiff-Marganiec S Gorton, C Montangero and L Semini. Stpowla: Soa, policies and workflows. In *In E. Di Nitto and M. Ripeanu (eds.): ICSOC'07 Workshops*, pages 351–362. LNCS 4907, 2009.

[TRMY07] M. Tilly, S. Reiff-Marganiec, and H.Q. Yu. *Design and Implemetationn of monitoring and aggregation mechanisms for context-based services - Version 1*. inContext project deviverables, D3.2 V1, 2007, http://www.in-context.eu/page.asp?PageRef=10, 2007.

[TRPA06] D.T. Tsesmetzis, I.G. Roussaki, I.V. Papaioannou, and M.E. Anagnostou. Qos awareness support in web-service semantics. In *International Conference on Internet and Web Applications and Services/Advanced International Conference on Telecommunications AICT-ICIW '06*, pages 128–128. IEEE Computer Society, 2006.

[WU+08] C. Wan, , C. Ullrich, L. Chen, R. Huang, J. Luo, and Z. Shi. On solving qos-aware service selection problem with service composition. In *Proceedings of Seventh International Conference on Grid and Cooperative Computing, 2008. GCC '08.*, pages 467–474. IEEE Computer Society, 2008.

[YL05] T. Yu and K. Lin. *Service Selection Algorithms for Composing Complex Services with Multiple QoS Constrains*. ICSOC2005, LNCS, vol: 3826, pp. 130-143, 2005.

[YRM08] H. Q. Yu and S. Reiff-Marganiec. A method for automated web service selection. In *2008 IEEE International Conference on Services Computing*, volume 0, pages 513–520, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[YRM09] H.Q. Yu and S. Reiff-Marganiec. Automated context-aware service selection for collaborative systems. In *Proceedings of The 21st International Conference on Advanced Information Systems*, pages 193–200. Springer Lecture Notes in Computer Science, 2009.

[ZBN+05] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. *QoS-aware middleware for web services composition*. IEEE Transactions on Software Engineering, pp. 311-327, 2005.