Applied Mathematics & Information Sciences
An International Journal

Application of Services Relation Tracing to Automated Web Service Composition

Qixuan Liang, Shizhan Chen and Zhiyong Feng*

School of Computer Science and Technology, Tianjin University, Tianjin 300072, China

Received: 15 Oct. 2012, Revised: 18 Dec. 2012, Accepted: 30 Dec. 2012
Published online: 1 Feb. 2013

Abstract: Web service composition has received much interest due to its potential to achieve the visionary promise of Service-Oriented Computing paradigm in which services can be utilized as the basic constructs to create flexible dynamic business processes and agile applications that may span organizations and computing platforms with little effort even automatically. In this paper, we proposed a snowball method for automated Web service composition which is based on service relation tracing which starts off rather like a standard ego-centered network study. It discoveries one or several available Web services which can accept the input parameters in user requirement through sematic reasoning and determines whether its output parameters meet the user requirements. And then it fetches other candidate services following the Sequential-total and Sequential-partial relations, investigates their outputs, also asks further sequential relations, and so forth through a succession of "waves" of spreading. So the process is repeated and the composite service is constructed incrementally tracing sequential relations in Service Network just like a snowball. The experiment shows that the approach proposed in this paper can greatly improve the accuracy and efficiency of Web service composition.

Keywords: Web service composition, service relation tracing, Services-Oriented Computing, Web service

1. Introduction

Recently, Service-Oriented Computing (SOC)[1,2] has received much interest due to its potential to tackle many adaptive system architecture issues that were previously hard to overcome by other computing paradigms. It utilizes functionalities provided by business applications and encapsulated within services, to build low-cost and flexible dynamic business processes or service-aware application[3] within and across organizational boundaries and computing platforms, even in highly distributed, open and heterogeneous Internet environments.

With the characteristics of loose coupling, cross platform, self-describing and modularization and constructed on the basis of publishing, discovery and interaction, Web service[4] has become the best practice of realizing SOC, turning Software as a Service (SaaS)[5], and developing business process across organization boundary[6], and has drawn much attention. When no single Web service can satisfy the users request in functionality, there should be a possibility to combine existing services together to satisfy the request. That is Web Service Composition (WSC). Logically, composite service (result of WSC) can be considered as s services chain consisting of series of relevant Web services. Each component service on the services chain, controlled by control flow, receives output parameters of its immediate previous service (except initial service, in which, users request is directly input), and then output its result to its immediate following service. Thus repeatedly, till the component service at end of services chain outputs users expected messages.

Compared with static composition method provided by WS-BPEL and WS-CDL , automated Web service composition is the key of realizing the visionary promise of Service-Oriented Computing. The reason is that Web services can be created and updated on the fly and it may be beyond humans capability to analyze the required services and compose them manually. [7] improves BPEL to make automatically composition. Therefore, on the road of realizing Service-Oriented Computing, in addition to the challenge of composition automation level, other challenges are the efficiency of composition method and the flexibility of composition proposal.

Tianjin Key Laboratory of Cognitive Computing and Application "to Author Affiliation in a new line before"

* Corresponding author e-mail: shizhan@tju.edu.cn

School of Computer Science and Technology. Web service composition based on ontology and Semantic Web [8] technology is a very active hotspot of research in SOC. There are many activities and much effect mainly concentrated on the semantic model for services discovery and dynamic service composition methods applying ontology and AI planning techniques. In this field, there are various solutions, from recapitulative abstract method to framework that has potential of becoming industrial standard. And major research efforts in this area have summarized, classified and evaluated in [9–14] from several perspective. However, there are still many open research questions and challenges in how to accurately identify the services on demand and aggregate them into a complex business process (or composite services) automatically [15–17].

Some research thought that Web service composition is similar to work flow, thus directed graph[18] or Petri net [19] representing various basic service and execution sequence in the flow can be used to construct model of composite service; or Web service can be abstracted into an automaton that receives events via inputting arrays and decides how to respond (in form of outputting events) [20]. Other research considered the automated composition of Web services as planning issue of artificial intelligence (AI): given clients request and a set of component services (atomic services or composite service), find out a composition that can satisfy the target, and apply AI based planning, reasoning and other techniques to automatically Web service composition [21–24], validation [22,25] and monitoring in run-time [26]. Aiming at the situation that Web service based WSDL is lacking of semantic information impedes the dynamic and adaptive interaction among services, the research focus of Semantic Web Services (SWS) [27,28] under the background of Semantic Web is to depict services using Ontology technique (OWL-S, WSMO (Web Service Modeling Ontology), WSMO-Lite, SAWSDL METEOR-S etc.), or to expand semantics of current solutions, thus web services can be understood consistently by different machines, thus promote automated service discovering [29–32], composition and interaction [33,34].

Relevance among Web services and its positive effect on service computing process has also been attracting some attention. An early contribution to the Web services relation was that by Zhang et al [35,36], in which the authors developed a business relationship description language, WSRL, to facilitate services discovery and integration. However, they only focused on static relevance information among services, and paid insufficient attention to dynamic relation and Quality of service (QoS) information. Barros et al [37] presented a collection of service interaction patterns, which allows emerging Web services functionality, especially that pertaining to choreography and orchestration, to be benchmarked against abstracted forms of representative scenarios from a business perspective. Combining social network and Semantic Web technique, Chen et al [38,39] focused on the dependence, relevance, interaction and competition among Web services caused by attribute and capability, etc., and defined explicitly service relation of the three levels, parameters, operations and services, designed and constructed a service computing infrastructure, named Service Network (SN), to introduce service relation into service discovery, composition and management. On the basis of SN, Guo et al [40] realized a service composition method based on service relation and shortest path, thus Web services composition is remarkably improved in accuracy and efficiency. However, the composition method only considers single chain in composition; it cannot solve the complex circumstances such as branches and multiple merge, etc., in composite structure.

In this paper, we describe a snowball composition method for automated Web service composition which is based on SN and service relation tracing and reasoning. The composition process in our case is divided into two phases. First, a description of a possible solution (namely abstract service composition) fulfilled the user requirements, is constructed incrementally just like a snowball. Second, the non-functionality attributes are used to determine which concrete services would be bounding to the abstract business process to generate the composite service. This paper is focusing on the first step of the composition process. We assume that WSDL presentation of Web services has been fabricated SN by semantic annotation and service relation mining [41].

The paper made the following contribution technically.

a) On the basis of previous work[38–40], complete and partial subsequent relation is distinguished to realize automatically the modes of parallel split and synchronization in composition.
b) Through service relation tracing and reasoning, snowball composition method is designed to improve composite efficiency.
c) The process of Web service composition is divided into two stages, abstract service composition based on relatively stable function construction and dynamic service binding considering non-functional factor, thus flexibility of composition is improved.
d) Test on prototype system shows, for rather complex test scene, the response time of snowball composition based service relation tracing is controlled within seconds, and meanwhile it has better response capability.

The rest of this paper is organized as follows: Section 2 presents a motivating example and the reason behind it, Section 3 describes the details of snowball composition algorithm based on service relation tracing, Section 4 is involved in experiential study and discuss, and with a conclusion in Section 5 we will finish this paper.

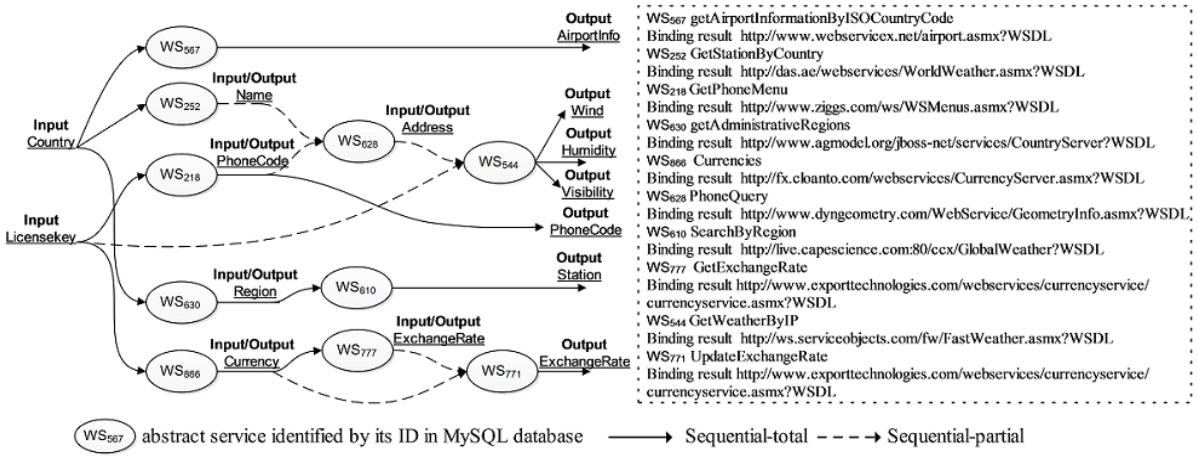

Figure 2.1: A motivating example of Web services composition

2. Motivating Example

The complete automation of WSC is still a hard problem and requires further research. One reason is that composite structure is complex net/graph instead of simple chain. For example, a tourist plans to enjoy vocation at some tourism site. After successful booking of air ticket, he wants some information of destination, like weather, exchange, airport or railway station, etc. A solution and the required services is represented in Figure. 2.1. It is important to mention that the flow mode is more than a simple chain of services. In particular, it is a complex graph which consists of ten standalone Web services, which can be developed by different organizations and described in different semantic models, and two workflow patterns, which are Parallel Split pattern and Synchronization pattern in BPEL4WS[42]. The interaction among Web services originates from the fact that in a composition, services can interact with each other to exchange parameters, for example a service's output parameter could be another service's input parameter. Service relation is constructed on the basis of parameter semantics and represents the semantic relevance among parameters of Web services. In WSDL archive, a complete parameter may be a simple type or a composite type with many simple types or composed by complex types. Naturally, there is difference in semantics matching level among parameters. For example, subsequent relation can be further divided into two types, total subsequent relation and partial subsequent relation. Sequential-total$< WS_1, WS_2 >$ refers to WS1 output parameter completely includes WS_2 input parameter, e.g., WS_1 may drive WS_2 independently, and no service is required between the two services to realize functional interaction, as the relation between WS_{630} and WS_{610} shown in Figure 2.1. Sequential-partial$< WS_1, WS_2 >$ refers that WS_1 output parameter is included in WS_2 input parameter, e.g., WS_1 can only provide partial parameters WS_2 input requires, and cooperation with other service output parameter is relied on to drive WS_2 to realize interaction. For example, WS_{252} and WS_{218} together can drive WS_{628}. Figure. 2.2 shows the four forms of

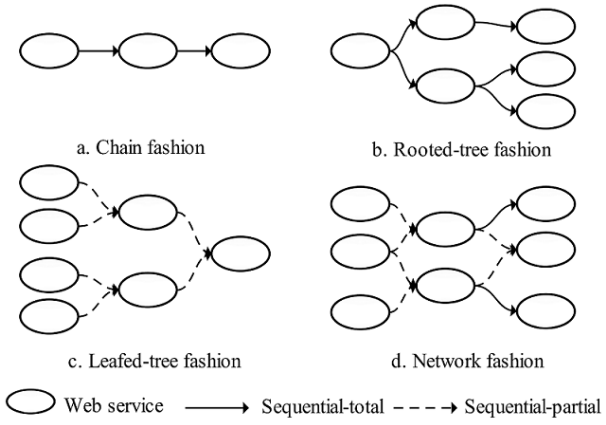

Figure 2.2: Four fashion in sequential relationship

composition flow constructed by total subsequent relation and partial subsequent relation. Service chain (Figure. 2.2-a) depicted the simplest services composition form. That is, multiple Web services connected between previous and subsequent services form single chain structure. Service tree depicted a combination form of tree type. If each service in service tree only has one previous service, named positive service tree (Figure. 2.2-b). If each service in service tree has only one subsequent service, named reverse service tree (Figure. 2.2-c). Service graph (Figure. 2.2-d) describes the most

complex combination forms among services composition, and each service may satisfy many subsequent services, and meanwhile, there are many previous services satisfying such service at the same time. In particular, simple service chain is rare, and more composition is a rooted-tree, a leafed-tree, or a complex graph.

3. Two-phase Service Composition Framework

With the primary purpose of snowball composition based on service relation tracing is to construct a possible solution to meet the given goal, our method contains two stages, abstract service composition and specific service binding.

The former stage only considers the semantic relevance between clients function demand and services, and it's also the stage to specially embodying the characteristics of snowball composition based on service relation tracing. The result of abstract services composition is an abstract business process that can satisfy clients demand functionally, in which, each abstract service represents a function, and all specific services relevant with the abstract service can realize the function Abstract business process is a key factor affecting Web service composition.

The responsibility of the later stage is selecting a specific service for each abstract service in the abstract business process according to QoS-aware selection as executor of the function.

3.1. Snowball composition based on service relation tracing

Based on semantic relevance among Web services, we define a service collaboration graph (SCG). As a connection sub-graph of SN, each node in SCG expresses an abstract service, and the directed edge between nodes means a service can be totally or partially driven by another service, e.g., there is possibility of combination. Figure. 3.1 gives an example of SCG. Such abstract service composition thus is developed into the problem of how to efficiently, accurately and incrementally construct SCG corresponding to clients demand from SN utilizing clients initial input, as well as search one or many path(s) connecting client's input parameters and output parameters from it.

The typical snowball composition based on service relation tracing starts off rather like a standard ego-centered network study. First, finding one or several available Web services through the input parameters in user requirement to initialize the SCG and determining whether its output parameters meet the user requirements. Then, upon the Sequential-total and Sequential-partial relationships, we find other candidates of the target with

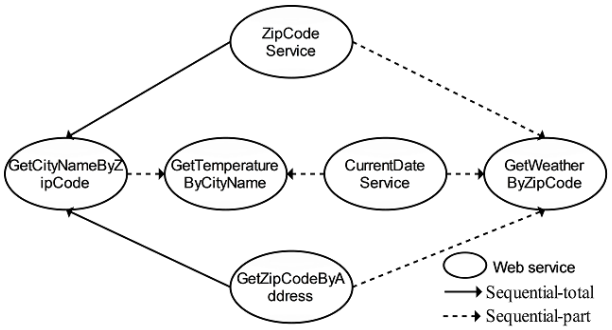

Figure 3.1: A sample of SCG

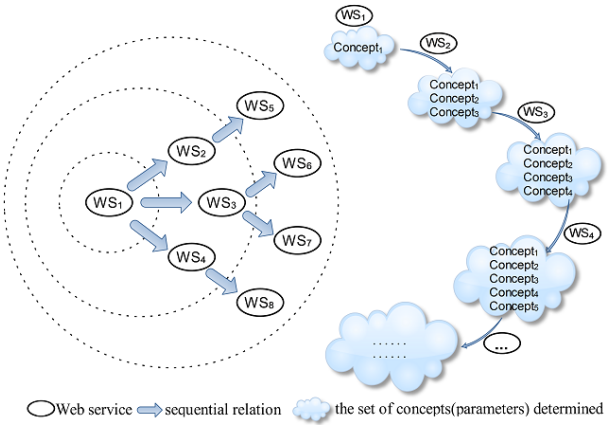

Figure 3.2: schematic diagram of snowball composition process

whom they are sequenced. Then we update the SCG using these sequences and investigate their output parameters, and also ask further sequential relations, and so forth through a succession of "waves" of spreading. The process is repeated with any whose outputs mismatch, tracing their sequential relations as well, and so forth, until the requirements are matched or all leads have been exhausted. Its schematic diagram is as shown in Figure. 3.2. In the whole searching and composition process, similar with LevelOrderTraverse, subsequent service of current service is stepwise added into SCG. Meanwhile, with level by level searching, SCG and parameters reviewed are become more gradually, just like snowballing. Detailed Algorithm is as follows:

Detailed Algorithm is as follows:

Algorithm 1 Snowball composition based on SR tracing.
Input: the array of request input items by user (requestIn- putList),the array of request output items (requestOutputList)
Output: roadList
BEGIN: //Initializing module. parameter_as is a hash table storing parameter and all abstract services

whose output semantically match the parameter in <Paramet- er,List of AbstractService> form. part_meet_list, a list of abstract services storing those services whose input cannot be provided fully. The road and roadList are used to store solution. All of them are initialed as NULL.

```
Pamameter_AS_Map < Parameter,List < AbstractService > parameter_as;
Part_Meet_AS_List part_meet_list;
Road road;
List<Road> roadList;
Queue<AbstractService> queue ; // SCG in queue fro- m Creating virtual start node on the basis of requestInp- utList which has been extended through semantic reasoning, then it includes all parameters in SN whose semantics is less than the user input parameters.
AbstractService start;
Start.outputList = SemanticReasoning ( requestInputList );
queue.push(start); //Initializing SCG with the virtual start node. Loop until queue is empty or parameter_as includes all the user request outputs (the given goal is satisfied!)
Snowball: While(queue is not empty and not parameter_as.containAllKey(requestOuptList)){
currentService = queue.pop();
// dealing with the totally sequential service
List <AbstractService> totalSequentialServiceList = currentService.totalSequentialServiceList
for AbstractService si in totalSequentialServiceList do
    for Parameter pi in si.OutputList do
        parameter_as.put(<pi, si >)
    end for
    Si.FatherList = currentService
    queue.push(Si)
end for
// dealing with the partally sequential service
List< AbstractService > partlSequentialServiceList = currentService. getPartSequentialServiceListByHighMatchDegree();
part_meet_list.addAll(partlSequentialServiceList);
for AbstractService si in part_meet_list do
    if parameter_as.containAllKey(Si.inputList) then
        for Parameter pi in si.outputList do
            parameter_as <pi, si >
        end for
        si.FatherList = parameter_as.get(pi)
        queue.push(si)
    end if
end for//Building a solution through traversing the output parameters required with the FatherList pointer. When there are several possible solutions, our method only chooses the one who is composed of the fewest nodes (The complexity will be relatively small.).
for Parameter pi in requestOutputList do
    AbstractService si = getNodeByOutputParameter (parameter_as,pi)
    while si is not NULL do
        road.add(si)
        si = si.FatherList.get(0)
    end while
    roadList.add(road)
end for
END
```

When there is possibility of various abstract business processes, only solution with the least nodes (whose complex degree will also be less) is selected and returned to client. Pretty soon the process ”snowball” and we have a possible solution to user requirement. Clearly this is a better way of finding a composition based on sequential service relation tracing than random-walk search in service repository, since each WS we had inspected must has potentially collaboration contacts to the former. Essentially, the snowball composition based on service relation tracing is a typical form of snowball sampling[43] applied to SOC.

3.2. Dynamic service binding

On the basis of abstract business process generated at the first stage, the second stage is to dynamically bind specific WS as the executor of the functional unit in abstract business process.

As the selection of the WS with multiple QoS attributes has been proved NP-hard[44]. Considering that in Internet environment the consumers are unlikely to care about all aspects of the QoS when they use Web services, we only choose three QoS properties (response time, successability and reputation) specified by the Web Services Quality Modeling to made simplicity of QoS-aware selection. Based on the QoS metrics preprocessing by normalization method, our method simply select the most adequate specific services from the set of concrete services who attach to an abstract one, which can be proved the algorithm complexity is O(n).

3.3. Optimization

In snowball composition based on service relation tracing the sample size grows exponentially with the number of relationship tracing waves while in random walk discovering the sample size grows only linearly. Measures taken in optimizing performance include:

a) When adding subsequent service nodes incrementally, total subsequent relation is considered first, and then followed by partial subsequent relation. Meanwhile, in partial subsequent relation, partial subsequent service with high parameters is considered first. That is, those with less missing parameters (can be driven easily) are considered with priority.

b) Hash table is used to store parameters. All abstract services that can output the parameter are guaranteed to quickly search and output all the abstract services of the parameter within O(1) time complexity.

It can be proved the snowball algorithm complexity is $O(n^2)$

4. Experiment

4.1. Experimental design

SN system prototype is a typical B/S structure. All aggregating and reasoning work are carried out at server, including:

a) Server, the main component of the system, including MySQL 5 database storing actual Web services collected from Internet, service relations and QoS information, and several background processes such as, service annotation, service relation mining and QoS inspector, etc., as well as automatic service compositor based on service relation tracing. The server is a PC with Windows 7, equipped with Intel Q8200 CPU, 4GB RAM and 500GB SATA hard disk.
b) PC with Windows 7 as client, equipped with Intel Q8200 CPU, 4GB RMB and 500GB SATA hard disk, IE8, connected with server via 100 Mbps Ethernet. In addition to receiving users input and displaying composition result via IE, client does not need to undertake any composition or reasoning.

Experimental scene setting simulated information inquiry in Section 2, in order to validate that snowball composition based on service relation tracing is rational and efficient.

Test dataset: 300 actual Web services related with tourism are downloaded from Seekda ,and added to SN after being pretreated by WSDL archives validation, service annotation and service relation mining, etc., including 1,026 abstract services and 49,321 service relationships.Response time is simulated with the response time of host that each Web service resides; success-ability and reputation are set as default value 1, to construct test dataset.

Test process: client submits initial input (including destination country and license key that drives the inquiry) and expected output information (like weather, airport, railway station and exchange rate of the destination) from clients machine; abstract business process and binding results satisfying clients demand are output and average time of five operation times is recorded.

4.2. Experimental result

In the above experimental scene, the abstract business process constructed with snowball composition based on service relation tracing and binding result, say, solution A, as shown in Figure. 2.1 in Section 2, takes 3,959 ms. Considering that the performance of the server we used in the experiment is rather low, the efficiency of our method is completely within users tolerance.

To further measure the effectiveness of the composition method, we get new result after adding one specific Web services, Weather , and corresponding three abstract services, GetWeatherInformation(1027), GetCityForecastByZIP(1028), and GetCityWeatherByZIP(1029) into test dataset on purpose (solution B) as shown in the Figure. 4.1. It takes 4,001ms.

By comparison, solution A and solution B are consistent in the service quantity they involve, say, both are 10 services. And there is not much difference in time cost. However, solution A has more complex composite structure. Due to the abstract service, GetCityWeatherByZIP, to join, the service 628 in solution A was replaced by the service 1013, 544 by 1029, which makes two Synchronization pattern resulted by partial sequential relation disappear in solution B that can be considered as consisting of two standalone subprocesses. Comparatively speaking, each subprocess is simpler too.

5. Conclusion

The most important contribution of this paper is to propose a snowball method for parallel split and synchronization pattern in automated WSC issue to make the aggregation of relevant Web services efficient and on-demand. Efficient means it bases on service relation tracing mechanism and reduces search space by coordination relation mined off-line. On-demand means the composition is distinguished abstract business process based on functionality, which is stable relatively and can be reused in future, and dynamic specific service binding based on non-functionality which is variable in the reality.

Future work will focus on two aspects. One is to design proposal selection policy of more sophisticated and ingenious abstract composition, to select strong composite proposal with simple structure instead of only considering service quantity involved. The other is to analyze service web structure based on complex web theory and method, to guide the improvement of composite computing and further increase automation level, efficiency and users experience of services composite.

Acknowledgement

The authors wish to thank Guangquan Xu for his valuable advice and the members of service computing group from Institute of Knowledge Science and Engineering, Tianjin University. This work was supported by the NSF of China under Grant No. 61173155, the 863 Program of China

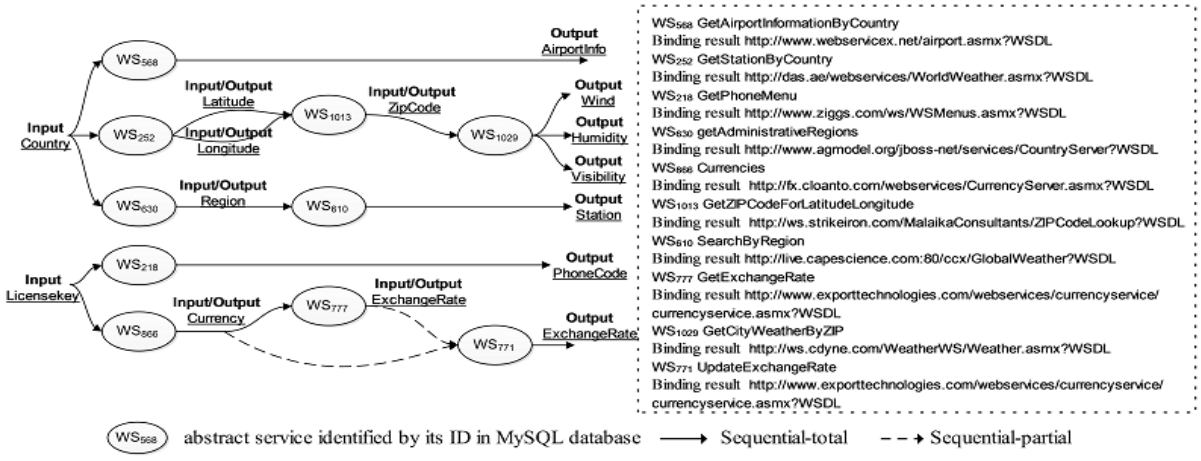

Figure 4.1: A candidate solution for the example in Sec.2

under Grant No. 2007AA01Z130, the 985 Project of Tianjin University under Grant No. 06050110000, and the Innovation Foundation of Tianjin University under Grant No. 2010XG-0009.

The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

References

[1] Papazoglou, M.P. and D. Georgakopoulos, *Service-oriented computing.* Communications of the ACM, 2003. **46(10)**: 24-28.

[2] Papazoglou, M.P., et al., *Service-Oriented Computing:* A research roadmap. International Journal of Cooperative Information Systems, 2008. **17(2)**: 223-255.

[3] Zhang . DG and Zhang . XD, *A New Service-Aware Computing Approach for Mobile Application with Uncertainty.* APPLIED MATHEMATICS & INFORMATION SCIENCES . 2012. **6(1)**: 9-21.

[4] Ferris, C. and J. Farrell, *What are Web services?* Communications of the ACM 2003. **46(6)**: 31-31.

[5] Turner, M., D. Budgen, and P. Brereton, *Turning software into a service.* Computer, 2003. **36(10)**: 38-44.

[6] Newcomer, E. and G. Lomow, *Understanding SOA with Web Services (Independent Technology Guides)*2004: Addison-Wesley Professional.

[7] Wu. J and Jin. L, *A Linear Logic Representation for BPEL Process Protocol.* APPLIED MATHEMATICS & INFORMATION SCIENCES. 2011. **5(2)**: 25-31.

[8] Berners-Lee, T., J. Hendler, and O. Lassila, *The Semantic Web.* Scientific American, 2001. 284(5): 34-43.

[9] Milanovic, N. and M. Malek, *Current solutions for Web service composition.* IEEE Internet Computing, 2004. **8(6)**: 51-59.

[10] Rao, J. and X. Su, *A Survey of Automated Web Service Composition Methods, in Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition(SWSWPC 2004)*2004, Springer: San Diego, California, USA. p. 43-54.

[11] Dustdar, S. and W. Schreiner, *A survey on web services composition.* International Journal of Web and Grid Services 2005. **1(1)**:1-30.

[12] Peer, J., *Web Service Composition as AI Planning - a Survey. University of St. Gallen, Switzerland, 63pp.* 2005.

[13] Strunk, A., *QoS-Aware Service Composition: A Survey, in IEEE 8th European Conference on Web Services (ECOWS, 2010)*2010: Ayia Napa p. 67-74.

[14] Cardinale, Y., et al., *Transactional-Aware Web Service Composition: A Survey, in Handbook of Research on Service-Oriented Systems and Non-Functional Properties: Future Directions,* S. Reiff-Marganiec and M. Tilly, Editors. 2011. p. 116-141.

[15] Srivastava, B. and J. Koehler, W*eb Service Composition: Current Solutions and Open Problems, in 13Th International Conference on Automated Planning&Scheduling(ICAPS2003)*2003: Trento, Italy. p. 28-35.

[16] Zhang, L.J., *Special issue: Web services discovery and composition - Editorial preface.* International Journal of Web Services Research, 2007. 4(1).

[17] Yi, W. and M.B. Blake, *Service-Oriented Computing and Cloud Computing: Challenges and Opportunities.* Internet Computing, IEEE, 2010. **14(6)**: 72-75.

[18] Casati, F., et al., *Adaptive and dynamic service composition in EFlow, in Proceedings of 12th International Conference on Advanced Information Systems Engineering (CAiSE 2000)*2000, Springer: Stockholm, Sweden. p. 13-31.

[19] Zhovtobryukh, D., *A Petri net-based approach for automated goal-driven web service composition.* Simulation: Transactions of the Society for Modeling and Simulation International, 2007. **83(1)**: 33-63.

[20] Berardi, D., et al., *Automatic Composition of E-services That Export Their Behavior, in 1st International Conference on Service Oriented Computing(ICSOC2003)*, M.E. Orlowska, Editor 2003, Springer: Trento, Italy. p. 43-58.

[21] McIlraith, S. and T.C. Son, *Adapting Golog for composition of semantic web services, in Proc. of the 8th International Conference on Knowledge Representation and Reasoning (KR2002)*2002. p. 482-493..

[22] Narayanan, S. and S.A. McIlraith, *Simulation, verification and automated composition of Web service, in Proceedings of the 11th International World Wide Web Conference(WWW 2002)*2002, New York: ACM Press: Honolulu, Hawaii, USA. p. 77-88.

[23] Sirin, E., et al., *Htn planning for web service composition using shop2.* Journal of Web Semantics, 2004. **1(4)**: 377-396.

[24] Traverso, P. and M. Pistore, *Automated composition of semantic web services into executable processes, in 3rd International Semantic Web Conference(ISWC 2004)*2004, Springer: Hiroshima, Japan. p. 380-394.

[25] Kazhamiakin, R., M. Pistore, and L. Santuari, A*nalysis of communication models in web service compositions, in Proceedings of the 15th international conference on World Wide Web*2006, ACM: Edinburgh, Scotland. p. 267-276.

[26] Barbon, F., et al., *Run-time monitoring of instances and classes of web service compositions,* in *Proceedings of the IEEE International Conference on Web Services (ICWS 2006)*2006, IEEE Computer Society: Chicago, IL, USA. p. 63-71.

[27] McIlraith, S.A., T.C. Son, and H. Zeng, *Semantic Web Services.* IEEE Intelligent Systems, 2001. **16(2)**: 46-53.

[28] Paolucci, M. and K. Sycara, *Autonomous Semantic Web services.* IEEE Internet Computing, 2003. **7(5)**: p. 34-41.

[29] Dogac, A., Y. Kabak, and G.B. Laleci, *Enriching ebXML registries with OWL ontologies for efficient service discovery,* in *Proceedings of the 14th International Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications (RIDE'04)*2004, IEEE Computer Society: Boston, USA. p. 69-76.

[30] Celik, D. and A. Elci, *Searching Semantic Web Services: An Intelligent Agent Approach Using Semantic Enhancement of Client Input Term(s) and Matchmaking Step, in International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*2005, IEEE. p. 916-922.

[31] Wang, X. and X. Yu, *A OWL-Based Semantic Web Service Discovery Framework, in International Conference on Internet and Web Applications and Services/Advanced International Conference on Telecommunications(AICT-ICIW '06)*2006. p. 125-125.

[32] Nawaz, F., et al., *Semantic web service registry for efficient discovery of OWL-S based web services, in International Symposium on High Capacity Optical Networks and Enabling Technologies(HONET 2007)*2007, IEEE: Dubai. p. 1-6.

[33] Klusch, M. and K.-U. Renner, *Fast dynamic re-planning of composite OWL-S services,* in *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology - Workshops,* C.J. Butz, et al., Editors. 2006, IEEE Computer Soc: Hong Kong, China. p. 134-137.

[34] Martin, D., et al., *Bringing semantics to web services with OWL-S.* World Wide Web, 2007. **10(3)**: 243-277.

[35] Zhang, L.-J., H. Chang, and T. Chao, *Web Services Relationships Binding for Dynamic e-Business Integration, in International Conference on Internet Computing (IC'02)*2002: Las Vegas, NV. p. 561-570.

[36] Zhang, L.-J., J. Zhang, and H. Cai, *Services Relationship Modeling*, in *Services Computing* 2007, Springer Berlin Heidelberg. p. 114-133.

[37] Barros, A., M. Dumas, and A.H.m.T. Hofstede, S*ervice Interaction Patterns: Towards a Reference Framework for Service-based Business Process Interconnection*2005.

[38] Chen, S. and Z. Feng, *Services NetworkThe New Anchor for Web Services Composition.* Journal of Application Research of Computers, 2008. **25(5)**: 1378-1382.

[39] Chen, S., Z. Feng, and H. Wang, S*ervice Relations and its Application in Services-Oriented Computing.* Chinese Journal of Computers, 2010. **33(11)**: 2068-2083.

[40] Yi, G., C. Shizhan, and F. Zhiyong, *Composition Oriented Web Service Semantic Relations Research, in 2011 International Joint Conference on Service Sciences (IJCSS)*2011: Taipei, TAIWAN. p. 69-73.

[41] Chen, S., et al., *Building the semantic relations-based web services registry through services mining,* in *8th IEEE/ACIS International Conference on Computer and Information Science(ICIS 2009)*2009, IEEE Computer Society: Shanghai, China. p. 736-743.

[42] Wohed, P., et al., *Analysis of Web Services Composition Languages: The Case of BPEL4WS,* in *Proceedings of the 22nd International Conference on Conceptual Modeling (ER)*2003, Springer: Chicago, USA. p. 200-215.

[43] Thompson, S.K. and O. Frank, *Model-based estimation with link-tracing sampling designs.* Survey Methodology, 2000. **26(1)**: 87-98.

[44] Zeng, L., et al., *QoS-aware middleware for Web services composition*. IEEE Transactions on Software Engineering, 2004. **30(5)**: 311-327.

Qixuan Liang, born in 19- 88, is a master of Software Engineering at Tianjin University, China. His current research include service computing and SOA.

Shizhan Chen, born in 19- 75, is a lecturer at Tianjin University, China. He obtained his PhD at Tianjin University in 2010 and his current research interests include service computing and SOA.

Zhiyong Feng, born in 19- 65, PhD, is a full tenured professor and head of the School of Computer Science and Technology, Tianjin University, China. His current research interests include knowledge engineering, service computing, and security software engineering.