Feature Interaction Problems in Web-based Service Composition

(position paper)

Qi Zhao, Jiyu Huang, Xiangping Chen, Gang Huang*

Key Laboratory of High Confidence Software Technologies, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Beijing, 100871, China
zhaoqi06@sei.pku.edu.cn, huangjy07@sei.pku.edu.cn, chenxp04@sei.pku.edu.cn,
* huanggang@sei.pku.edu.cn (corresponding author)

Abstract. The web-based service (WBS) composition, e.g. mashup, is becoming a popular style to assemble web services. Numerous web-based services are developed by independent organizations and individuals due to the open nature of Web and SOA. As a result, when these components are assembled, the undesired interactions will take place and bring negative impacts. From the perspective of the feature interaction problem (FIP), we study such undesired interaction problems in WBS composition. We illustrate three WBS FIP cases and explain how on-the-fly characteristic, the significant feature of WBS composition, brings new challenges and opportunities to FIP detection and resolution.

Keywords: feature interaction problem; service composition; mashup

1 Introduction

Nowadays, there are many services published via Internet with open APIs, such as Amazon S3 and Google Map. Many existing work provided web-based service (WBS) composition environments [6] [14], which are hosted in web browsers. In these environments, the service business logic and user interface (UI) are encapsulated into a single component, called WBS components. Developers are able to create applications, called mashups, by assembling these components, in a rich user experience manner.

Because of the open nature of Web and Service Oriented Architecture (SOA), every organizations and individuals can make their own WBS components and publish them via Web. Hence, WBS components are always developed in a parallel, independent and incremental manner. When these components are assembled, some interactions among them, such as potential resources competition or implicit data dependency, are hard to be predicted and controlled beforehand. Even worse WBS composition always lacks explicit specifications and requirements which explain interactions among components clearly, since WBS composition prefers lightweight

and agile service composition approaches. As a result, the undesired interactions possibly take place and bring some negative impacts, such as demotion of quality of service (QoS) or even crash of the whole system. The problem of undesired interactions in WBS composition is quite similar with the feature interaction problem (FIP) in telecom [7], which has been well studied from 1989. We argue that the plentiful achievements and experience of FIP can help to investigate and solve the interaction problem in WBS composition.

Moreover, comparing with FIP in telecom, the interaction problem in WBS composition has its own characteristics. The significant difference between WBS composition environments and traditional software systems is that WBS components are actively running entities without design-time and runtime separation. These components can be used without compilation and deployment. Such difference makes WBS composition environments keep all components online and assemble them on-the-fly. Therefore, online techniques should play more important role in FIP detection and resolution in WBS composition.

In this paper, we study FIP in WBS composition. The rest of this paper is organized as follows. Section 2 illustrates three cases of FIP in WBS compositions which we meet in practice. In section 3, we explain the on-the-fly characteristic of WBS composition and how this characteristic affects FIP detection and resolution. Finally, section 4 makes conclusion and discusses the future work.

2 Feature Interaction Problem Cases in Web-based Service Composition

2.1 Case 1: FIP of Ajax Invocations

Ajax (Asynchronous JavaScript and XML), is a group of interrelated web development techniques used to create interactive web applications or rich Internet applications. With Ajax, web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page [2]. In WBS composition, Ajax is widely used for invoking remote services. The responses of invocations are used to modify the status of components which are assembled with these services.

Considering a mashup application assembles three WBS components. The first two components both change the status of the third one with Ajax invocations. However, the order of Ajax invocation responses may be different with the order of requests due to the un-deterministic network delay and server response time. Therefore, FIP will occur due to the undesired order of responses. The first invocation may be responded at last, and the final status of the third component may seem strange, since in common requirement, the final status is expected to be modified by the response of the last invocation but not the first one’s. The undesired sequence diagram of these invocations is shown as Fig. 1.

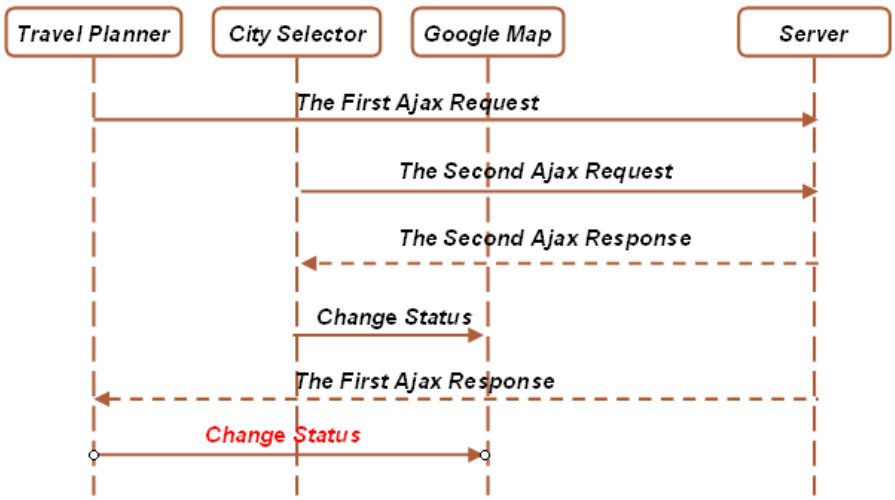

Fig. 1. The Sequence Diagram of Ajax Invocations FIP

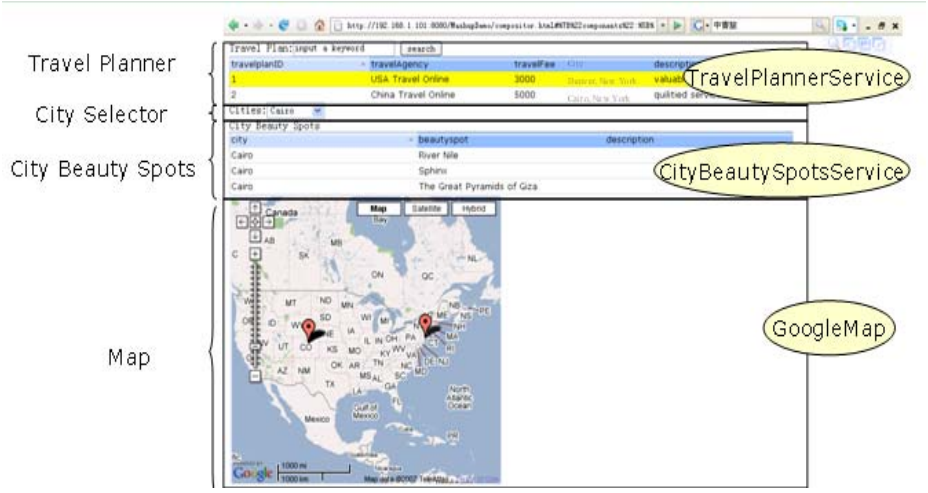

Fig. 2. A Mashup Sample which may Indicates Ajax Invocations FIP

[3] gives a sample of Ajax Invocations FIP. In this sample, a mashup application that provides travel information to users. This mashup has a travel planner, a city beauty spots explorer and a google map, as shown in Fig. 2. The first two components will both clear the map and place location markers on it. Suppose a user first chooses a “New York” travel plan, and then he/she thinks it is better to look through all the attractions of Cairo before making a decision. So he/she selects Cairo’s spots very quickly after the selection of the travel plan. The “city beauty spots explorer” will

show all the beauty spots of Cairo, and the “google map” is expected to display the locations of these beauty spots. But in some cases, FIP occurs. The “google map” will display the locations of New York, which is the result of the previous action, choosing the travel plan.

2.2 Case 2: FIP of HTTP Connections

Major WBS composition environments are hosted in web browsers, which typically limit the number of concurrent HTTP connections per server to be two [4]. If a browser session has two HTTP connections already, any further connection requests will have to wait until one of these two HTTP connections is released. On the other hand, in WBS composition environments, most remote service invocations are delegated by a proxy because of the cross-domain limitation of web browsers. The remote service invocation architecture of WBS composition environments is shown as Fig. 3. The “cross-domain proxy” is some kind of “bottle-neck”.

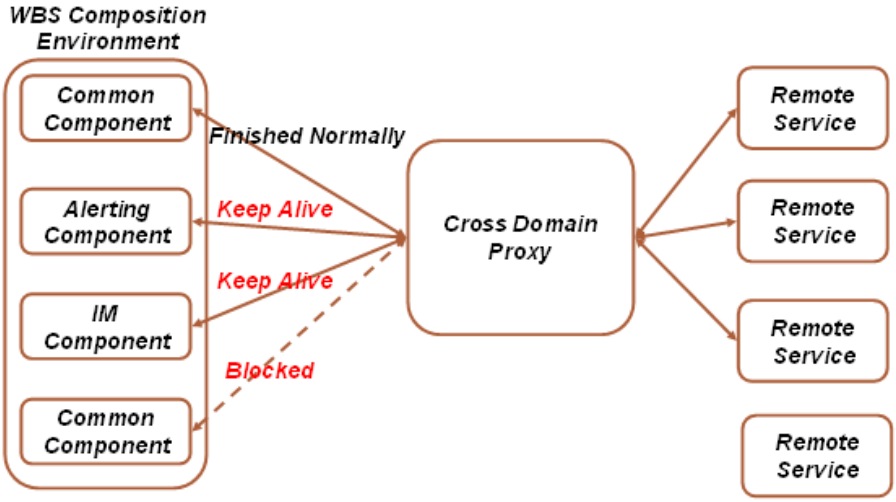

Fig. 3. The Service Invocation Architecture of WBS Composition Environments

Currently, many web applications require server-push style communications, which allows web servers pushing real-time dynamic web data to browsers, such as web instant message chat, real-time calendar alerter and stock tickers [11]. More and more server-push style WBS components are also implemented. HTTP server-push implementations use some form of "long-lived connection" for delivering server-push messages [5]. This "long-lived HTTP connection" is a normal HTTP connection but the server keeps it open. Whenever a message occurs, the server uses this connection to deliver the message to the client. However, this long-lived connection consumes one of the two connections limit. Hence, when two components which use server-push technology are instantiated in the WBS composition environment, the normal

HTTP requests initiated by the other components will be blocked since both connections are consumed. From the perspective of FIP, undesired interactions take place due to resource, HTTP connections, competition.

2.3 Case 3: FIP of Data Dependency

WBS components can retrieve server data by invoking remote services. They may store the data into their own data model and use it to keep their status. If one server data item is associated with more than one component and these components are developed independently, FIP will occur due to the implicit data dependency.

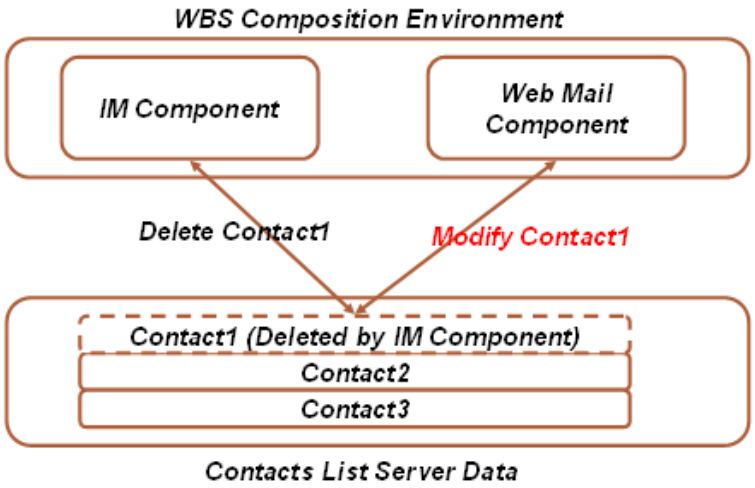

Fig. 4. The Data Dependency between WBS Components

FIP of data dependency is similar with FIP of entity components in middleware based systems, which occurs since multiple entity components associate with one database item [1]. For example, in many personal information services, such as Windows Live Services, live messenger and web mail share one contacts list. A personal information mashup application may include live messenger and web mail WBS components which connect with one contacts list at the server. Therefore the deletion of one contact, which is executed by live messenger component, may cause the contact modification behavior of web mail component undesirably, since the two components are designed and implemented independently.

3 On-the-fly Characteristic of Web-based Service Composition

Unlike traditional components that need compilation and deployment, services on the Web are actively running entities. Such a significant difference implies that the

separation between design-time and run-time is not an inherent nature of web-based service composition. In other words, it is possible to assemble services in an on-the-fly manner, that is, 1) when a component is loaded into the composition environment, it becomes available with real appearance and functionality immediately; 2) when two components are assembled, they can interact with each other actually. This characteristic takes a lot of advantages in service composition, such as qualification [6]. Given a set of components, and a schema for assembling them, qualification is to check the proposed composition satisfies a given set of requirements [9]. With an on-the-fly composition approach, the services and composition can be qualified in a "what you see is what you get" way, since developers can qualify a service by invoking it directly and checking the real result immediately. They can also qualify a service composition by activating the service flows in the composition and checking the real changes of the services. All of these provide a quick and effective feedback way for qualification.

With on-the-fly composition approach, FIP will occur as soon as the components are assembled. The characteristic brings both new opportunities and challenges to FIP detection and resolution. Firstly, developers can detect some FIP at once if the FIP makes the user experience undesired. In that sense, the negative impact will be only perceivable to the developers instead of customers. Therefore, On-the-fly characteristic provides an effective manner for FIP detection. On the other hand, FIP approaches for WBS composition should be online techniques due to the on-the-fly characteristic and need a combination of detection and resolution mechanisms at runtime. Moreover, automated resolution is important, since feature interactions detected at runtime should be resolved just-in-time. Otherwise systems may have negative impacts or even crash down.

The existing work on FIP always treats runtime stage as the final defense [7][8][10]. Major types of FIP approaches, such as software engineering and formal methods, try to detect and resolve FIP as early as possible. These approaches always focus on requirement and design stages in software life-cycle, which do not support on-the-fly characteristic of WBS composition. For example, [12][13] address a method for detecting feature interactions related to the functionality of a composite web service. The method focuses on web service composition in enterprise environment which is based on WS-* and BPEL specifications. However, it is an offline detection approach and hard to be applied to WBS composition as we mentioned above. Though these approaches still have some valuable technical points, they have to be evolved and even re-invented hence.

Current online approaches are intended to be applied at runtime, which are suited for WBS composition. For instance, [3] proposes a model-checking based approach which can detect FIP of Ajax invocations as mentioned above. This approach first specifies the application behavior at runtime by analyzing the javascript source codes of WBS components. The source codes are downloaded into web browser before components instantiation since javascript is an interpreted language. And then the pre-defined environment behavior will combine with the application behavior retrieved at the first step. As a result, a precise and complete behavior model of the application is generated automatically. FIP, which is represented as model flaws, can be detected by model checker. However, these online approaches still have some limitation and are not being used widely in practice. For example, for these online approaches, runtime

resolution is still a difficult problem due to the limited amount of information available [8]. Therefore, an online approach which aims at the characteristics of FIP in WBS composition is still required.

4 Conclusion

Feature interaction problem is a serious obstacle to increase new service and improve QoS in telecom. In web-based service composition, WBS components are functional units like the features in telecom. Therefore when they are assembled, FIP will occur due to incomplete or inconsistent requirements, conflicting assumptions, bad designs and so on. In this paper, we illustrate three FIP cases in WBS composition which come from our practice. Moreover we also explain how the on-the-fly manner, which is the significant characteristic of WBS composition, affects FIP detection and resolution.

In our existing work [1], we propose an online approach to FIP in middleware based systems. It is an online approach to the detection and resolution of FIP based on runtime software architecture (RSA), which facilitates the analysis and decision making by expressive software architectures and supports the monitoring and adaptation by reflective middleware. In our future work, we will try to apply and improve this approach to detect and resolve FIP in WBS composition, since it focuses on detecting and resolving FIP at runtime. In more specific terms, we have already implemented a WBS composition environment which supports on-the-fly composition in browser [6]. We plan to develop a runtime monitoring module and deploy it into this environment. The module collects information from the runtime system. And then with the above runtime information, an intelligent program, which is based on our approach, can measure and analyze the runtime system for detecting FIP or threats to FIP. Once FIP or threats are detected, the program would try its best to resolve or prevent the problems.

Acknowledgements

This work is partly sponsored by the National Key Basic Research and Development Program of China (973) under Grant No. 2009CB320703; the Science Fund for Creative Research Groups of China under Grant No. 60821003; the National Natural Science Foundation of China under Grant No. 60873060; the High-Tech Research and Development Program of China under Grant No. 2009AA01Z16; and the EU Seventh Framework Programme under Grant No. 231167.

References

1. HUANG Gang, LIU Xuanzhe, MEI Hong. An Online Approach to Feature Interaction Problems in Middleware based Systems. Science in China, series F Vol 51, No. 3, 2008, pp. 225-239.
2. Ajax (programming), http://en.wikipedia.org/wiki/AJAX
3. Xiangping Chen, Gang Huang, Hong Mei. Towards Automatic Verification of Web-based SOA Applications. 10th Asia Pacific Web Conference (APWeb), 2008, pp. 528-536.
4. The Two HTTP Connection Limit Issue , http://www.openajax.org/runtime/wiki/The_Two_HTTP_Connection_Limit_Issue
5. Cometd Bayeux Ajax Push, http://cometdproject.dojotoolkit.org/
6. Qi Zhao, Gang Huang,Jiyu Huang, Xuanzhe Liu, Hong Mei, Ying Li, Ying Chen. An On-the-fly Approach to Web-based Service Composition. Proceedings of 2008 IEEE Service Congress and International Conference on Web Services (ICWS 2008), September, 2008, pp 208-209.
7. Chi, C.X., R.B. Hao, G. Huang, Y.H. Jin, D. Lee, Y.J. Lin, C. Liu, H. Mei, F. Peng, D. Wang, Y. Xing, W. Zhang. Feature Interaction: A Survey, Bell Labs Technical Report, March 17, 2003.
8. Calder, M., E.Magill, M.Kolberg, and S.Reiff- Marganiec, Feature Interaction: A Critical Review and Considered Forecast. Computer Networks, 2003, 41(1): 115-141.
9. Hafedh Mili, Fatma Mili, and Ali Mili. Reusing Software: Issues and Research Directions. IEEE Transactions on Software Engineering, Vol 21, no. 6, June 1995.
10. Kech, D., P.J. Kuehn, The Feature and Service Interaction Problems in Telecommunications Systems: A Survey. IEEE Transactions on Software Engineering, 1998, 24(10): 779-796.
11. Engin Bozdag, Ali Mesbah, and Arie van Deursen. Performance Testing of Data Delivery Techniques, for AJAX Applications. Journal ofWeb Engineering (JWE), 2008.
12. Michael Weiss, Alexander Oreshkin, and Babak Esfandiari, Offline Detection of Functional Feature Interactions of Web Services, International MCETECH Conference on eTechnologies, 2006.
13. Weiss, M., Oreshkin, A., and Esfandiari, B., Method for Detecting Functional Feature Interactions of Web Services, Special issue on Web Service Engineering, Journal of Computer Systems Science and Engineering, 21(4), 273-284, 2006
14. Yahoo! Pipes, http://pipes.yahoo.com