A Utility-Aware Runtime Conflict Resolver for Composite Web services

Xiao Ning, Jiuyun Xu, Nan Xu, Di Li
School of Computer and Communication Engineering
China University of Petroleum(China East)
Qingdao, China
xiaon2013@gmail.com, jiuyun.xu@ieee.org,
xuxinanzi@gmail.com, zazadeanlee@gmail.com

Stephan Reiff-Marganiec
Department of Computer Science
University of Leicester
Leicester, UK
srm13@le.ac.uk

***Abstract*—Web services are developed independently and deployed in a distributed environment, new service can be obtained by composing existing ones. The rapid introduction of new services also results in undesirable interactions between services. These conflicts are not mismatches of interfaces, but are usually based on the data in the executing instance and therefore runtime management of conflicts in Web services should be considered. We study the problem from the perspective of user's revenue, and propose an online approach to resolve conflicts is proposed[1].**

***Keywords*-Composite Web service; Conflict Resolution; Bi-Objective Optimization; Runtime Solution**

I. Introduction

Services assumption, which are theoretically met at design time but are violated at runtime may lead to unexpected and often undesired interactions between services negatively affecting user satisfaction. It is important to note that not all interactions are necessarily bad; some might be positive for example such where services have special preferential relations with one another. [1] describes these conflicts as Web services Feature Interaction (WSFI) problem.

Currently, existing methods for conflict management focus on offline methods using formal verification and online methods based on behaviour grammar descriptions. Challenges lie in the facts that Web service are development independently from each other with internal logics unavailable for business reasons. That means there is a main obstacle to describe each service accurately for formal method approaches. Second, conflicts caused by Web services are often dependent on data (either data from user profiles for personalizable services or runtime data in the process). Consequently, there will be conflicts appearing at run time which cannot be detected and resolved at service and process design time. So it is meaningful and urgent to design runtime conflict management methods.

In our previous work [2][3], we presented run-time conflict detection methods. Meanwhile, **resolving conflicts** is the ultimate aim of conflict management and this paper presents a novel online approach to conflict resolution. Prototype system experiments illustrate that our approach is efficient for resolving conflicts at runtime taking into account the constraints and preferences of the users.

[1]Acknowledgement: The work is fully supported by grants (13CX06009A and 14CX06007A)from the Fundamental Research Funds for the Central Universities. Part of this work was conducted while Reiff-Marganiec was on study leave from the University of Leicester.

II. The Proposed Approach

We use three fundamental concepts as follows: (1) a *service plan* p is a composite service which satisfies a user's demand, (2) a *service model* M is a set of available service plans for achieving a user's requirements and (3) the *plan utility* u_p which is the utility value of a service plan. The plan $p \in M$ with maximal utility will be the preferred execution plan; all others plans are candidate plans.

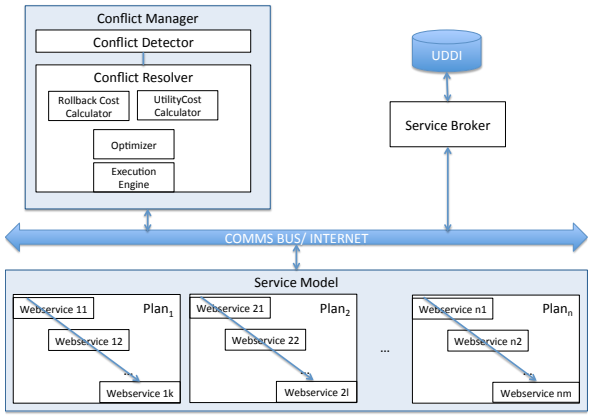

Figure 1: System Architecture

The architecture of the conflict resolution system is shown in Figure 1. The *service broker* allows providers to register services in an UDDI registry (registering both capability and QoS of the service). The *conflict manager* consists of the *conflict detector*(described in Ref.[2]) and the *conflict resolver* (the key contribution of this work). Once a conflict between services is detected the conflict instance is passed to the *conflict resolver*. The conflict resolver will:

1) Calculate the roll-back service set and respective cost;
2) determine Pareto solutions for the service plan (II-A);
3) rank strategies and propose the best resolution (II-B);
4) and the process execution engine applies the conflict resolution strategy.

A. The bi-objective model based-on users revenue

The conflict problem is expressed as bi-objective optimization problem with constraints. The corresponding bi-objective optimization model is established and a set of Pareto solutions is obtained. The solutions are ranked to identify the optimal roll-back strategy and the new plan.

The two objective functions used express the profit gained from the new execution plan and the compensation profit of roll-back, that is the opposite of the compensation cost. The strategy space is $C = \{hold, drop\}$, that is we can decide to retain or to drop a specific service. In order to get the optimal resolution strategy for conflicts, we aim to maximise the value of the two objective functions. The model considers users' QoS requirements as differentiating factor between services. Compensation transaction make it possible to undo committed transactions. However, compensation is potentially expensive [4], so the calculation of compensation cost is necessary. We consider the fee for the transaction and the time needed to compensate.

B. Solving the Bi-Objective Model

The fundamental character of a bi-objective optimization problem is the conflict between the two objectives. The result of the bi-objective problem is not a unique solution, but a set of Pareto solutions under constraints. In this paper, the solution of the bi-objective model is a trade-off between user expected utility of new plan and profit of roll-back.

Resolutions are ranked. Larger values represent closer proximity to the ideal point; the highest value identifies the optimal solution. If no solution satisfies the user constraints for a given task, an execution exception will be raised and the system will ask the user to relax their constraints.

III. Experimentation and Implementation

To study the performance of the proposed solution approach, we conducted experiments using a prototype system. The service model for an e-shopping example is shown in Figure 2, which also shows the utility values (in the top left). p_4 has the maximum expected utility value and is chosen to execute. However, during execution a conflict between service *Shop_3* and service *ICBC* is detected.

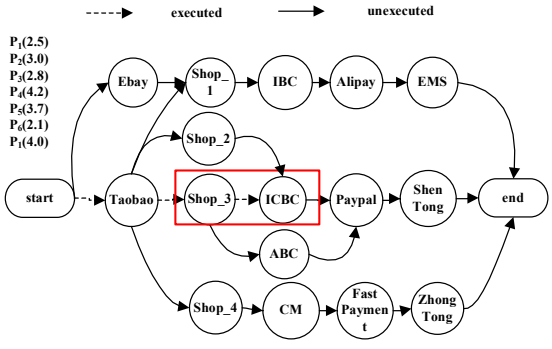

Figure 2: Composition services model of E-commerce

Assuming values for the backtracking cost of each service an optimal solution can be calculated, which will show which services to roll back and which plan to execute. Experiments for different sizes of the service model (*6, 9, 12, 15, 18 and 21*) show a linear increase in run time. In practice, the expected number of alternative service plans is not very large. That is to say the conflict resolution approach for composite Web service is efficient and feasible.

IV. Related Work and Conclusions

The feature interaction problem in the domain of Web services has attracted a number of studies including [5], [6], [1], which focus on the requirements analysis phase. But in dynamic Web services composition, conflicts often occur during the execution of Web services, which cannot be completely avoided in the design phase. Therefore, in order to improve the efficiency of Web services composition, on-line management conflicts requires solving. Work in this area concentrates on conflict detection [7] presents an immune-inspired on-line detection system for WSFI problem. [3], [2] adopt the method of situation calculus to achieve a dynamic detection of feature interaction.

The presented work established a user-centred bi-objective optimization model to obtain a strategy to dynamically resolve conflicts in service compositions. Experimental results show that under the premise of fully considering the user's revenue we obtained optimal strategies for resolving conflicts. Future work will consider how to improve the efficiency and robustness of the resolution mechanism as well as combining the detection and resolution mechanisms.

References

[1] M. Weiss and B. Esfandiari, "On feature interactions among web services," in *Web Services, 2004. Proceedings. IEEE International Conference on*. IEEE, 2004, pp. 88–95.

[2] J. Xu, W. Yu, K. Chen, and S. Reiff-Marganiec, "Web services feature interaction detection based on situation calculus," in *Services (SERVICES-1), 2010 6th World Congress on*. IEEE, 2010, pp. 213–220.

[3] J. Xu, K. Chen, Y. Duan, and S. Reiff-Marganiec, "Modeling business process of web services with an extended strips operations to detection feature interaction problems runtime," in *Web Services (ICWS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 516–523.

[4] B. Limthanmaphon and Y. Zhang, "Web service composition transaction management," in *Proceedings of the 15th Australasian database conference-Volume 27*. Australian Computer Society, Inc., 2004, pp. 171–179.

[5] G. Huang, X. Liu, and H. Mei, "Online approach to feature interaction problems in middleware based system," *Science in China Series F: Information Sciences*, vol. 51, no. 3, pp. 225–239, 2008.

[6] D. Amyot and L. Logrippo, "Feature interactions in web services," *Feature Interactions in Telecommunications and Software Systems VII*, p. 149, 2003.

[7] J. Zhang, F. Yang, K. Shuang, and S. Su, "Immune-inspired online method for service interactions detection," in *SOFSEM 2007: Theory and Practice of Computer Science*. Springer, 2007, pp. 808–818.