Where to Adapt Dynamic Service Compositions*

Bo Jiang
The University of Hong Kong
Pokfulam, Hong Kong
bjiang@cs.hku.hk

W. K. Chan
City University of Hong Kong
Tat Chee Avenue, Hong Kong
wkchan@cs.cityu.edu.hk

Zhenyu Zhang, T. H. Tse
The University of Hong Kong
Pokfulam, Hong Kong
{zyzhang,thtse}@cs.hku.hk

ABSTRACT
Peer services depend on one another to accomplish their tasks, and their structures may evolve. A service composition may be designed to replace its member services whenever the quality of the composite service fails to meet certain quality-of-service (QoS) requirements. Finding services and service invocation endpoints having the greatest impact on the quality are important to guide subsequent service adaptations. This paper proposes a technique that samples the QoS of composite services and continually analyzes them to identify artifacts for service adaptation. The preliminary results show that our technique has the potential to effectively find such artifacts in services.

Categories and Subject Descriptors
D.2.5 [**SOFTWARE ENGINEERING**]: Testing and Debugging – *Monitors*.

D.2.8 [**SOFTWARE ENGINEERING**]: Metrics – *Product metrics*.

General Terms
Measurement, Reliability, Verification

Keywords
Service composition, service adaptation

1. INTRODUCTION
Web services, a kind of web-based service-oriented technology, encompass a set of application functions for service consumers to accomplish their business processes and for composite services to invoke one another's functions [1]. A business process may evolve, the functions of supportive services may change, and a service composition may shift from a set of services to another set. Providing alerts on the weak parts of business processes or service compositions that potentially require service adaptation are fundamental to effective service adaptation. We propose such a technique in this paper.

Let us motivate our work via a mortgage loan scenario. Suppose Ricky applies for a mortgage loan via InstantLoan, which is a bank service that appoints AskGrace, a risk assessor service, to estimate Ricky's chance of breaching the loan contract. Suppose further that AskGrace rates the loan higher in credit risk than it should be. InstantLoan accepts this assessment result and sends it to RecommendMe, a service of loan interest recommendation, which in turn recommends an approval of the loan despite a high interest rate. Ricky judges the interest rate to be too high and turns down the loan offer. Thus, the bank loses business, an innocent service (RecommendMe) suffers, and the *root cause* of the problem (AskGrace) remains concealed. If InstantLoan had identified the inadequacy of AskGrace early, it might have adapted its service composition by, say, using another risk assessor to avert the problem. On the other hand, condemning AskGrace to be guilty is only a straightforward solution in a simple illustrative example. In real life, a service composition may be large in scale, deep in service invocation chains, data-dependent on the dynamic behaviors of member services, and complex to analyze in detail.

Other risk assessors selectable by InstantLoan may also incur similar problems. If this is the case, from the perspective of InstantLoan, the current risk assessment workflow does not fully support the needs of the banking business. One design solution is to adapt InstantLoan to include a quick scan of a loan application and to invoke multiple risk assessors to obtain a consensus. For an effective adaptation, InstantLoan must identify its workflow inadequacy first. We observe, however, that existing research has not adequately studied the mechanism to recommend fragments of a service composition to ease dynamic service adaptation.

Engineers may specify the business process in, say, WS-BPEL (the de facto language). To support service adaptation, Moser et al. [3] extend a WS-BPEL engine by an interception and adaptation layer to enable dynamic service selection. This layer monitors WS-BPEL programs over a set of specified Quality-of-Service (QoS) constraints and dynamically invokes services that satisfy the constraints. Nevertheless, invocating a replaced service at an arbitrary invocation endpoint may not be effective in alleviating the QoS problem. The Pareto Principle (also known as the 80-20 rule) suggests that most of such invocations can be in the "long tail". Finding the right services and invocation endpoints (which we call ***cracks***) for effective adaptation helps reduce the problem of high cost in maintaining dynamic service compositions.

We outline our technique in Section 2 and evaluate it in Section 3. Section 4 concludes the paper.

2. OUR TECHNIQUE
This section describes how our technique samples QoS values at service invocation endpoints, ranks services repeatedly, analyzes the rank history of the services, and suggests the cracks.

2.1 Service Monitor and QoS Collection
To be scalable, our technique samples service endpoints (such as an <invoke> statement of a WS-BPEL program [2][3]) in a composite service. To be non-intrusive, similarly to [3], it intercepts and records the invocation history at the middleware tier.

During the execution of a service composition, whenever a sampled service endpoint invokes another service, we record the invocation. At the same time, we also collect the QoS values of the invoker. As such, our technique records a sequence of invocations among collaborating services and models the sequence as a list of invocation history, denoted by H. For each invocation, we collect a value vector $v = \langle v_1, v_2, \ldots, v_m \rangle$ representing the set of given QoS metrics values at the endpoint. We further denote the pool of services invoked by the endpoints in H and the set of QoS values associated with H by Φ and V, respectively. For ease of presentation, we assume that all the collected QoS values have been normalized between 0 (worst) and 1 (best).

* This work is partially supported by GRF grants of the Research Grants Council of Hong Kong (project nos. 111107, 123207, and 717308).

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

2.2 Detection of Cracks

During the collection of the execution data and QoS data of the sample service invocations, our technique conducts an online statistical analysis of the dataset and computes the relative ranks of the services in Φ according to their chances of being a crack.

For each service s in Φ, we compute the sensitivity of being a crack (dubbed *crack-sensitivity*) by the equation

$$M(s) = \frac{\sum_{v \in V} \left[\frac{(1 - \bar{v}) \cdot N(v,s)}{N(v)} \right]}{\sum_{v \in V} \left[\frac{N(v,s)}{N(v)} \right]}$$

Here, $N(v)$ is the total number of service invocations in the current H. Every such service is associated with a QoS value vector $x = \langle x_1, x_2, ..., x_m \rangle$ such that $x_i \geq v_i$ for all $i = 1, 2, ..., m$. $N(v, s)$ is the number of innovations for service s. Further, $\bar{v}$ is the mean value of all the elements v_i in v.

For such an evaluation step, the technique sorts the services (in Φ) in descending order of metrics values into a list. We deem that those services having *consistently* received higher metrics values than other services in the generated lists are more crack-sensitive. To categorize a service to have consistently received higher ranks than others, the technique first conducts analysis of variance (ANOVA) on the above lists of ranked services to check whether the means of the crack-sensitivity values are significantly different from one another. If so, the technique further performs a multiple statistical comparison procedure to find out whether the topmost services significantly differs from the rest. If it is also the case, the technique reports that crack-related services are found.

Having identified such a service, the technique searches its corresponding endpoints from the latest H. It reports these endpoints as cracks (for subsequent service adaptations) if such endpoints are still on the latest dynamic service composition.

3. EVALUATION

We have conducted a simulation experiment in a case study of a real-life insurance application for damage claim service [4][5].

Case study: A motor vehicle insurance policyholder first calls the Europe Assist service (EA for short) to initiate a damage claim procedure. EA then submits a claim form to AGFIL and assigns a garage service (G for short) for repair. AGFIL forwards a copy to Lee Consulting Services (Lee for short). Based on the claim form, Lee appoints a loss adjustor (LA for short) to inspect the damage of the claimed vehicle and to negotiate the repair price with G. Once Lee approves the quotation of the repair price with G, the garage commences the repair work. Lee then sends AGFIL an adjustor's report. Finally, AGFIL pays G and LA.

Experimental Setup: The experiment sets up 10 instances of each kind of service for selection. We randomly assigned one garage (referred to as G1) to exhibit 1% chance of producing results with abnormal size under uniform distribution. The experiment composes services according to [5], and invokes the service of damage claim application 100,000 times. The service G1 may reply to the invoker with abnormally huge results, which deplete the processing resources of the invoker and lead to the abortion of the composite service. We consider such an abortion as an occurrence of QoS violation (that is, binary metrics). We evaluated the crack-sensitivity at each invocation of EA.

Experimental Result: Figure 1 shows the crack-sensitivity distribution of each service. An ANOVA test (which checks whether the mean crack-sensitivity of services differ significantly from one another) further confirms that their means differ at the 0.1% significance level. Following the ANOVA result, we further perform a multiple comparison procedure to check which pairs of such means are significantly different from each other, which further strengthens the observation from Figure 1 that G1 is more crack-sensitive than all other services. Figure 2 depicts the result of the multiple comparison procedure, which confirms that G1 differs significantly from all other 49 services in terms of mean crack-sensitivity. The p-values of the multiple comparison procedure are small enough to reject the null hypotheses. For example, the p-value of the comparison between G1 and G2 is 7.9578×10^{-6}. Based on G1, the service endpoint "Assign garage" in EA (in the case study [5]) is reported as a crack.

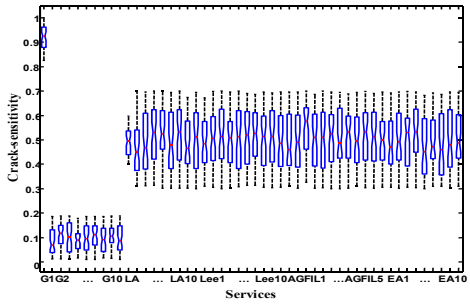

Figure 1: ANOVA of Service Crack-Sensitivity

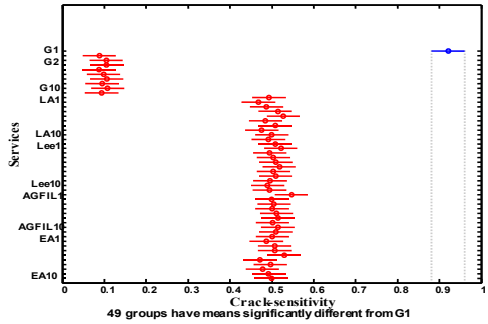

Figure 2: Finding Cracks (G1) with Multiple Comparisons

4. CONCLUSION

In this paper, we have proposed a statistical framework to assess member services and to identify vulnerable areas called cracks to support service adaptation. We will improve our algorithm and investigate an effective and lightweight online metrics evaluation strategy (including better evaluation metrics) to support continuous assessment of potential cracks and reduce false positive reports. To cater for long-running services, we will also explore different sampling strategies to make use of subsets of available historical data for analysis. Ontology-based service composition (via OWL-S, for instance) and the testing of adaptation patterns will also be interesting areas for future study.

5. REFERENCES

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer, Berlin, 2004.

[2] L. Mei, W. K. Chan, and T. H. Tse. Data flow testing of service-oriented workflow applications. In *Proceedings of ICSE 2008*, pages 371–380. ACM Press, New York, NY, 2008.

[3] O. Moser, F. Rosenberg, and S. Dustdar. Non-intrusive monitoring and service adaptation for WS-BPEL. In *Proceedings of WWW 2008*, pages 815–824. ACM, New York, NY, 2008.

[4] CrossFlow Consortium/AGFIL. Insurance requirements. *CrossFlow deliverable D1.b*. La Gaude, March 1999.

[5] C. Ye, S.C. Cheung, W.K. Chan, and C. Xu. Atomicity analysis of service composition across organizations. *IEEE Transactions on Software Engineering* 35(1):2–28, 2009.