Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 11, November 2014, pg.407 – 413

RESEARCH ARTICLE

IDENTIFYING AND REPLACING WEB SERVICE USING SERVICE COMPOSITION AND GENETIC TECHNIQUE

K.MALINI[1], M.AKILA RANI[2]
PG STUDENT, ASSISTANT PROFESSOR
NPR COLLEGE OF ENGINEERING AND TECHNOLOGY, TAMIL NADU, INDIA
EMAIL: malinimoorthy007@gmail.com , akilakamalam@gmail.com

Abstract: Web service is a middleware service between business transactions. If the service causes some errors like deadlock, network failure in current system the business transaction totally fails. The problem is easily identified and tolerate through fast bully algorithm. The algorithm identifies the failed service and replace the new service with help of election process conducted by Fast Bully, which minimizes the response time. The master service will allocate a composite of slave service with availability ,best band width and best fit. When a master service sends IAMUP message to slave service, it will replace its position with master service. The Fast Bully algorithm used for the purpose of services are identified and replaced with the business transaction by using Genetic Techniques. Avoids the violation of QoS constraints after replanning by defining and evaluating a replace-ability property. A proactive approach that searches for an optimal plan with a lower risk of violating the constraints in the event that re-composition is needed.

1. INTRODUCTION

Web service is a method of communication between two electronic devices over a network. Web services are open standard (XML, SOAP, HTTP etc.) based

Web applications that interact with other web applications for the purpose of exchanging data.

Service provider

It needs to describe the web service in a standard format, which in turn is XML (extendable mark-up language) and publish it in a central Service Registry (UDDI-Universal Discovery Description Integration).

Service registry

Web services can be developed and deployed by many vendors there are often multiple web services that can perform similar tasks with varying Quality of Service (QoS) attributes. When composing services into a workflow or plan to jointly accomplish processing toward a final result, the services must be compatible with respect to their input, output, and functionality for a temporally ordered interaction that can successfully complete the required task or query. The QoS of the plan depends on individual service selections for these designated interactions.

Service consumer

It retrieves the information from the registry and uses the service description obtained to bind to and invoke the web service.

Benefits of using Web Services

- Exposing the existing function on to network
- Connecting Different Applications is Interoperability
- Standardized Protocol
- Low Cost of communication
- Loosely Coupled

- Ease of Integration
- Service Reuse

A major problem with web service composition is that QoS values can also change at execution time from original estimations. The service may become unavailable, unreliable, or no longer provide the best solution fit. Other services must be dynamically evaluated to complete the plan. These services are chosen from the same abstract type, a group of services with functionalities that can substitute or replace any service in their type.

Changing QoS values can disrupt the expected compliance of the plan to maintain certain thresholds, such as costs and response time. The impact is even more dramatic if the service lies within a loop of large number of iterations in the composition. These non-periodic changes require a dynamic planning environment in which certain events force reselection from the physical services of the same abstract type in which the service change occurred to form a new, yet compliant plan.

QoS attributes are increasing-dimension or decreasing-dimension. Availability and reliability are increasing-dimension attributes because the resulting plan should incorporate the highest values associated with them. Cost and response time are decreasing-dimension attributes because the plan should incorporate the lowest values associated with them. Techniques for web services composition based on QoS optimization aim to maximize increasing-dimension attributes and minimize decreasing-dimension attributes, while at the same time maintaining any quality constraints imposed on the plan itself. These characteristics make the composition fall into the domain of multi-objective optimization. However, none of those different techniques explored in this domain takes into account redundancy as an inherent property of composition.

A proactive approach that searches for an optimal plan with a lower risk of violating the constraints in the event that re-composition is needed. Our approach introduces replaceability as a metric applied to plan composition. We define replaceability as the degree to which a plan or a service is exchangeable with one that accomplishes the same goal or processing, respectively. By including a replaceability metric in the selection process, we significantly reduce the potential violation of constraints during plan execution that can result from QoS changes requiring service reselection. A major challenge is the impact of service reselection on the plan, since substituting a service for one whose QoS values have caused the plan to violate at least one constraint consumes time. Pre-knowledge of alternatives based on replaceability values counteracts this added factor and reduces the time spent in the process.

2. LITERATURE SURVEY

2.1On the Evolution of Services

To manage and control the evolution of services is an important goal for the Service-Oriented paradigm. Unifying **Theoretical Framework** for controlling the evolution of services that deals with structural, behavioral, and QoS level-induced service changes in a type-safe manner, ensuring correct versioning transitions so that previous clients can use a versioned service in a consistent manner.

2.2 A fast and elitist multiobjective genetic algorithm: NSGA-II

Web services are grouped as a community to facilitate and speed up the process of Web services discovery. The Web Service Community can continue providing services even when master Web service fails operationally. Solution customizes a distributed election algorithm called **Fast Bully Algorithm** to identify a temporary master Web service when there is any operational failure in

existing master Web service of Community. Permanent master Web service takes back the mastering responsibilities from temporary master Web service when it resumes.

2.3 Understanding Approaches for Web Service Composition and Execution

Web services have received much interest due to their potential in facilitating business-to-business or enterprise application integration. The creation of a work owns that realizes the functionality of a new service and its subsequent deployment and execution on a runtime environment. A significant number of solutions have analyzed for **Composition and Execution of Web Services** and the strengths and weaknesses of the solutions. Based on multiple metrics that critical for a WSCE system, e.g. composition effort, composition control, and ability to handle failures.

2.4 QoS Aware Middleware for Web Services Composition

The paradigmatic shift from a Web of manual interactions to a Web of programmatic interactions driven by Web services is creating opportunities for the formation of online Business-to-Business collaborations. Many available Web services provide identical functionality, though with different Quality of Service. **Ag Flow's algorithm** Select Web services for the purpose of their composition in a way that maximizes user satisfaction expressed as utility functions over QoS attributes. Two selection approaches are described and compared: one based on local (task-level) selection of services, and the other based on global allocation of tasks to services using integer programming.

2.5 Quality of Service and Semantic Composition of Workflows

For the composition of Web services non-functional characteristics are commonly considered criteria for finding and selecting available services **Work**

Flow Pattern for Aggregation of QOS. Work focuses on a mechanism that determines the overall Quality-of-Service (QoS) of a composition by aggregating the QoS of the individual services. With aggregated QoS it can be verified whether a set of services satisfies the QoS requirements for the whole composition or not. The aggregation performed builds upon abstract composition patterns, which model basic structural elements of a composition like parallel paths, a sequence, or a looped execution.

3. CONCLUSION

The most crucial stage in achieving a new successful system and in giving confidence on the new system for the users that will work efficiently and effectively. The system can be implemented only after thorough testing in done and if found to work according to the specification.

If involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, an evaluation of changeover methods apart from planning. Two major tasks of preparing the implementation are education, training of the users and testing the systems. System analysis and design efforts will be more for complex systems beings implemented. Based on policies of individuals organization an implementation coordinating committee has been appointed.

The implementation process begins with preparing a plan for the implementation system. According to this plan, the other activities are to be carried out. In this plan, discussion has been made regarding the equipment, resources and how to test the activities. Thus a clear plan is preparing for the activities.

REFERENCES

1. V. Andrikopoulos, S. Benbernou, and M. Papazoglou, "On the Evolution of Services", IEEE Trans. on Software Engineering, 2011.

2. K Deb, A Pratap, S Agarwal, and T Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," in IEEE Trans. on Evolutionary Computation, 2002.

3. V. Agarwal, G. Chafle, S. Mittal, and B. Srivastava, "Understanding Approaches for Web Service Composition and Execution," Proc. 1st Bangalore Computing Conference, 2008.

4. G. Canfora, M. Penta, R. Esposito, and M. L. Villani, "An Approach for QoS-aware Service Composition based on Genetic Algorithms," Proc. of GECCO, 2005.

5. L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoSAware Middleware for Web Services Composition," in IEEE Trans. on Software Engineering, 30(5), 2004.

