Proceedings of 1st Shri Chhatrapati Shivaji Maharaj QIP Conference on Engineering Innovations
Organized by Shri. Chhatrapati Shivaji Maharaj College of Engineering, Nepti, Ahmednagar
In Association with JournalNX - A Multidisciplinary Peer Reviewed Journal, ISSN No: 2581-4230
21st - 22nd February, 2018

RUNTIME SERVICE DISCOVERY

Miss.Rohini Jadhav,

Prof. Dr. S.D.Joshi
BVCOE Computer Department
Email:Rohini.jadhav@outlook.com,sdj@live.in

Abstract - During the execution of service based application there is an important issue for replacing the service in them which fails to satisfy the current requirement or is no longer present. Services based can be identified on various service parameters such as quality,behavioural,contextual andstructural leads to effective runtime discovery of service. So, we are introducing a framework that support runtime services discovery.The framework supports two modes of execution of query for service discovery namely push and pull mode. In pull mode (reactive way), queries are executed oncerequirement for searching a replacement service rises (for example: unavailability or malfunctioning of service, emergence of new service, changes in structure functionality etc.).In push mode (proactive way), execution of query is carried out parallely to the servicebased system execution. Hence, if there is a requirement to substitute or changea service in service-based application, push(proactive) mode of execution of query makes it possible to escape interruptions in the working of application. Both types (i.e., pull and push) of queries are indicated in a XML based query language known as SerDQueL. SerDQueL is the language that allows the representation of behavioral (functionality), structural(interface), contextual and quality conditions of services need to be found. So in this paper, we are going to discuss the runtime service discovery framework.

Keywords—Web Services, XML, Service Discovery.

I. INTRODUCTION

Services are nothing but entities that are owned by a third party in a service based application. Such services are helpful in creating the dynamic business processes. Due to various changing requirements in market, it is important to identify the services that will directly fulfill the need of user by providing the appropriate quality and functional features of service based application .The process of identifying such services is called as Service DiscoverySeveral researches and approaches has been created to support service discovery, categorized as dynamic[1], [2], [3], [4] and static [5], [6], [7], [8] approaches. The services which are identified during the application development and prior to execution comes under static approach while the services that are identified throughout the execution of the application comes under dynamic approach. There are various scenarios where we need to substitute the service throughout the execution of application.

Following are the scenarios due to which we need to replace the services:
1)Variations in the interface, behavioral, quality, service context in the application due to the service which no longer fulfill its task.
2) Due to service unavailability , service malfunctioning.
3) The occurrence of new services which satisfy the task in much improved way than the existing service.
4) Changes in application context due to services which is used no longer fulfill its task.

The above scenarios give rise to a question, that how to support the application when the service which is being used is not working properly or stop functioning as desired, as well as changing the application and their services continuously at runtime .To address such questions we require a flexible and dynamic service identification at the runtime of service based application.
Currently most of the approaches uses pull mode of query execution for dynamic service discovery. This approach of query execution is often not efficientas the discovery process starts only if the requirement for a new service rises (as inabove case 1) and it may also consume more time to complete, which will affect the application performance and its capability to prepare appropriate “real-time” answer to the user and it should also be taken into consideration that pull mode discovery approach cannot recognize better or appropriate services (as in case 3 above). Similarly, for other cases 2 & 4 above, pull mode discovery would required to wait unless the changes that made used services become inadequaterises at runtime, similar to

case 1. On the other hand, the pull mode of query execution required to be improvised for polling service registries on a regular basis and/or context data resources to discover changes that can produce another issues. Such polling would consume resources as even if there is no need to do so it would required to be executed at regular intervals (i.e., in case of emergence of new services ,application environment, in the absence of service context changes or context changes).

In addition, existing methods to service discovery do not consider various characteristics of the application simultaneously when attempting to discover services such as, functional (i.ebehavioral), quality, interface (i.estructural) and contextual aspects .

To address the existing methods lacunas, we provide a web service discovery framework that provides runtime service discovery based on complex queries that can articulate flexible groupingsof quality, structural, contextual and behavioral parameters. These complex queries are described in query language based on XML, known as SerDQueL. The framework considers services that have unusual descriptions with service functional, , quality, interface and context descriptions.

To support above cases 1 to 4 and avoid the lacunas of traditional mechanisms for polling, our framework permits service discovery based on both push and pull mode of query execution . Pull mode query execution is started inabove cases 1 . In push mode, query execution is carried out in parallely to the application execution using queries. These queries maintain current sets of candidate replacement services for these services & are related with particular services in an application . In both pull and push modes, execution of query is based on similar& the computation of service specifications and distances among queries.

II. OVERVIEW

In this section, we will be presenting some situations for runtime web service discovery and provide an in general description of how our framework will be helpful to deal with these scenarios.

A) Scenarios of Runtime Web Service Discovery

Considering to a mobile service based application called On-the-go-News many scenarios or situationsrelated to runtime service discovery can be found [10] .

On-the-go-News is the application which permits its users to request news from their mobile phone from different sites as well as get the response accordingly.

To do so, the application provides services letting users to:

1. searchingnews topics and selecting the source from where the user desired to receive the news,
2. display news from different sources about a topic,
3. create tailored on-the-fly “magazines” or with datafrom variousunlike news sites,
4. in a tailored magazine flip through articles from numerous sources,
5. find and pay for the paid available data, by applying the amount in the user’s invoice at mnth end, and
6. check new balance of user’sinvoice after using the application for 5.

On-the-go-News uses$S_{Service}$ an peripheral service, which findsvarious news sites to locate news about particular topics, and$S_{CustMag}$another service, which enables the combination of news and their look in an tailored on-the-fly magazine.

After getting a request for news on a particular topic one runtime service discovery scenario may occur, on-the-go-News is unable to reach or contact $S_{Service}$ due to which thenext service is not available (Case 1). In this case, the application will requireto find a new service to substitute or change$S_{Service}$. Once the new service is found and bound to on-the-go-News,the user who requested service will start getting the requested data from varioussites.

A second situation may rise if a user who wants an on-the-fly magazine regardingchange in weather on her/his mobile phone &developed such a magazine using $S_{CustMag}$ starts receiving response slowly from $S_{CustMag}$ as the service is used by many different users at the same time.(Case 2). In such scenerios, a substitute service for $S_{CustMag}$ with suitable response time required to be foundand bound to on-the-go-News.

A third scenario rises while a user of on-the-go-News is traveling by train, he loses access to the service that displays and supports flipping through news (i.e., a service
Known as$S_{DisFlip}$) since $S_{DisFlip}$is not able to begain access at his/her current location. This alter in the location of on-the-go-News
(Case 3) needs finding for ansubstitute service that could be used in the current location of user.

A fourth situation arises once a new service that permits payments by debiting the user’s account and payments by card rather than applying charges in the user’s invoice(phone bill) becomes accessible. In on-the go news,if flexibility in payment is, the new service should be bound to the application.

B) Framework Support

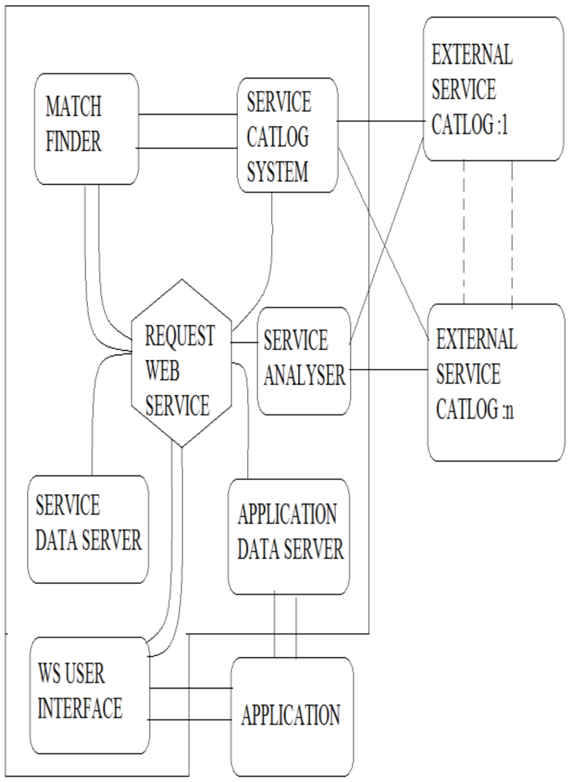

Fig.1 Architecture of various components in runtime service discovery framework

Runtime service discovery consists of web service user interface, application data server, service data server, service analyzer, request web service, match finder, service catalog system, external service catalogs.

The web service user interface is used by the user to query for a web service which provides access to framework. It represent request entry point and response exit point.

The request web service block offers various functionalities. It instantiates the service queries to be executed by the matchfinder based on the various situations, receives responses from matchfinder, collect, organize the result and send it to the web service interface etc.

The matchfinderis responsible to analyze the constraints of query and assessconstraints in contradiction of service specifications in the different service registries. The matchfinder consists of two stages filtering and ranking.

The service catalog system provides the use of different service registries.Service catalog system's main aim is to provide an interface, which permits the matchfinder to access services from different type registries.

The service data server and application data server enable context data is to be subscribed. They also allow to spread contextdata of the application environment and services.

The service analyzer provides the data about the new service available or data about changes in interface, quality ,functionalfeatures of existing services of an application.

III. WEB SERVICE DISCOVERY QUERY LANGUAGE

A runtime service discovery query may havevarious criteria, namely:1) behavioral criteria i.eit describes the functionality of the service needed 2) structural criteriai.eit describes the interface of the service needed; 3) constraints i.eit specifies additional criterias for the service to be discovered; 4) The last criteria may refer tointerface characteristics of servicesor quality aspects of the desired service which is not possibleto represent by the standardized forms of structural descriptions used in the framework. Examples of constraints referring to quality features of services may concern to cost or the maximum response time to execute a certain operation in a service.

The constraints in a query can be either non-contextual or contextual. A contextual constraint is concerned with dynamic informationi.e which chages during the service based application operation or the services deployed by the application,. The constraints are classified as soft or hard. Soft constraints do not required to be fulfilled by all services that are dicovered, but are used to rank candidate services.

A hard constraint must be fullfiled or satisfied by all services that are discovered for a query and is used to filter services that do not accomplish with them.

To specify runtime service discovery queries, we have created a language based on XML, known asSerDQueL. SerDQueLallows the specification of all the contextual ,structural,quality,and behavioral characteristics needed from the services which will be discovered. Pervious version of SerDQueLwas presented in [9]. The new version of thelanguage that we represent in this article enables the behavior specification of a service needed using behavioral criteriainstead of a full BPEL model of service behavior as in the original version.

A) UML Diagrams

In this section we are providing the use case and class diagram for the runtime service discovery.

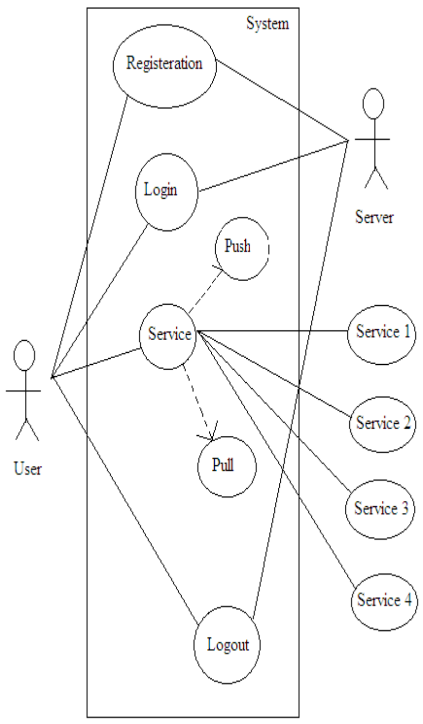

Fig. 2 Class Diagram for Runtime Service Discovery

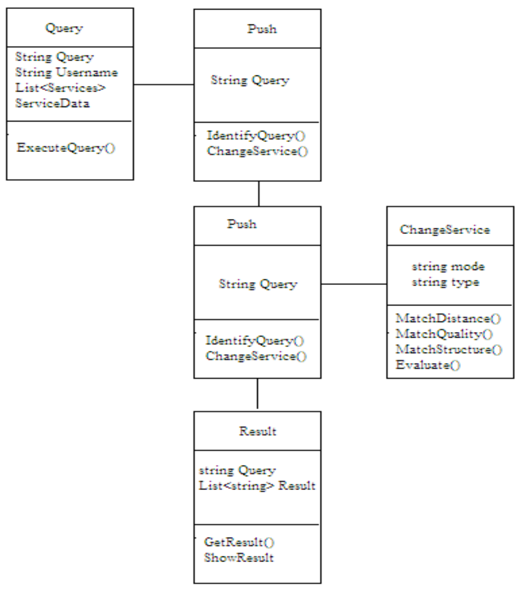

Fig. 3 Use Case Diagram for Runtime Service Discovery

IV. CONCLUSION

In this paper, we have seen a framework for dynamic service discovery in which candidate services are used to reinstate the existing services of service based application. This framework also overcome the problem of various scenarios for example. Malfunctioning or unavailability of services, etc. In push and pullmode of query execution, a service is coordinatedin contradiction of a query based on service specifications and computation of distances between query. The framework uses complex queries articulated in query language based on XML namely SerDQueL. This language allows the representation of behavioral, structural, quality and contextual featuresof applications and services. Now we are planning to check the correctness of SerDQueL as well as reduce the need of polling.

REFERENCES

[1] ,J.C. Corrales, D. Grirori and M. Bouzeghoub, "Behavioral Matching for Service Retrieval," Proc. Int'l Conf. Web Services, 2006.

[2] R. Lara, U. Keller, A. Polleres, H. Lausen, and D. Fensel, "Automatic Location of Services," Proc. European Semantic Web Conf., 2005.

[3] L. Li and I. Horrock, "A Software Framework for Matchmaking Based on Semantic Web Technology," Proc. Int'l Conf. World Wide Web, 2003.

[4] J. Su and Z. Shen , "Web Service Discovery Based on Behavior Signatures," Proc. Third Int'l Conf. Service Computing, 2005.

[5] H. Lutfiyya ,M. Katchabaw ,S. Cuddy, "Context-Aware Service Selection Based on Dynamic and Static Service Attributes," Proc. IEEE Int'l Conf. Wireless and Mobile Computing, Networking, and Comm., 2005.

[6] Vazirgiannis N. Loutas,andC. Doulkeridis, "A System Architecture for Context-Aware Service Discovery," Electronic Notes of Theoretical Computer Science, vol. 146, no. 1, pp. 101-116, 2006

[7] H. Niu and Y. Park, "An Execution-Based Retrieval of Object-Oriented Components," Proc. 37th ACM Southeast Regional Conf.,1999

[8] J. Grundy, S. Singh, ,J. Sun and J. Hosking, "An Architecture for Developing Aspect-Oriented Web Services," Proc. Third European Conf. Web Services, 2005.

[9] A. Zisman, K. Mahbub, and G. Spanoudakis "A Platform for Context-Aware Run-Time Service Discovery," Proc. IEEE Int'l Conf. Web Services, 2007.

[10] G. Spanoudakis, A. Zisman, James Dooley, IgorSiveroni"Proactive and Reactive Runtime Service