INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 5, Issue 3)
Available online at: www.ijariit.com

On the fly integration of applications using context aware approach

Anumita Sinha
anumita.s@somaiya.edu
K. J. Somaiya College of Engineering, Mumbai, Maharashtra

Nishi Mehta
nishi.mehta@somaiya.edu
K. J. Somaiya College of Engineering, Mumbai, Maharashtra

Pramod Bhakta
pramod.bhakta@somaiya.edu
K. J. Somaiya College of Engineering, Mumbai, Maharashtra

Era Johri
erajohri@somaiya.edu
K. J. Somaiya College of Engineering, Mumbai, Maharashtra

ABSTRACT

IT Applications have dominated every sector making the services online, there is a heartfelt need to standardize these applications so as when these applications when work in conjugation give better results real time for any kind of queries of customers maintaining levels of abstraction and information hiding. This paper aims at proposing a model for the on fly integration of the applications using context-aware self-adaptation approach for application service to cater to the need to integrate the data and services provided by the various applications. Our research within this direction has depicted the characteristics of the existing information and application systems primarily of the QoS Broker Architecture and Service Oriented Architecture. The latter part of this paper proposes an architectural model to describe the fly culmination of services by standardizing it and providing the result of a query using the context-aware engine. The aim of this paper is to study the existing systems, identify drawbacks and improve systems integrations to achieve an optimized and dynamic platform for real-time query processing.

Keywords— *Service integration, Integrating and standardizing applications, Normalize data, Context-aware, Web service, UDDI, REST API, WS-QoS, tModel*

I. INTRODUCTION

With the advent of an application for each purpose, it certainly has allowed the user to lead an easier lifestyle. Moreover, as the information has become ubiquitous, it should allow the user to access a single platform for all of its queries to promote better communication and collaboration. It has become a prime concern now to integrate many legitimate independent systems into one platform to provide platform integration.

The most usual method of the systems integration requirement is data integration which is considered to be the most required parameter for today's technological approach. Traditionally, organizations implement information and dynamic integration of the systems to solve internal business issues. Consequently, heaps of information systems were common in organizations. The factors affecting the use of such a common platform may be attributed to constantly reviewing and managing products with new technologies, security, isolation, timing, relevancy and many more. Applications are becoming popular as a basic technology in the internet field. They are revolutionizing the cutting edge of communication amongst the various applications there exist. Using the need for pervasive computing and the increased manifold use of mobile devices, the on-fly context-aware application service becomes a necessity. This takes into consideration to adapt the queries to the user's context such as his type of Internet connection, language, specific work environment, devices and preferences. Here, we introduce an on-fly context-aware approach which culminates the dynamic adaptations of application services. The proposed approach takes into account the user's context along with the other requirements. Thus, the literature suggests that this rises from users having to access many databases containing similar data, wherein the transfer and use of information amongst the databases were fairly restricted.

The implementation is therefore divided into two parts, first stage where we standardize the data obtained by the various applications and the second stage wherein the user or client can run queries to derive inferences using the context-aware engine for processing.

2. MOTIVATING SCENARIO

Considering various applications that exist in today's market, redundancy is observed when the user tries to access a similar set of information across multiple platforms to congregate and deduce inferences with their individual efforts. To avoid this situation it is essential to increase the focus on data integration and platform integration in a pervasive environment [2]. Along with the context of the services, there exists a need for adapting dynamically in response to those context changes which can be maintained by contextual contracts. Context information required by a service is termed as a contextual contract. Therefore to create an on-fly context-aware service for the queries the user has, two types of processing needs to be elicited: standardization of services and querying using on-fly context-aware system. The former stage concerns the extraction of relevant data and services from various applications and ensuring they adhere to the QoS Models. In the latter stage, the ability to adapt dynamically to the behaviors in response to changing environments with the additional support of Natural Language Processing (NLP) without explicit user intervention is catered to. In Section IV, we explain using an example of how the above proposition could be implemented with our architecture.

3. EXISTING MODEL

The existing architecture of web services with an SOA Framework is depicted along with the QoS Broker Model. In the following section, the challenges imposed the existing model are elaborated and the need for a new architecture is stated.

3.1 Web services architecture

A service-oriented architecture implements the design, application and on-fly integration of user-requested services in an extensible manner allowing expeditious, uniform responses to satisfy the ever-changing customer requirements.

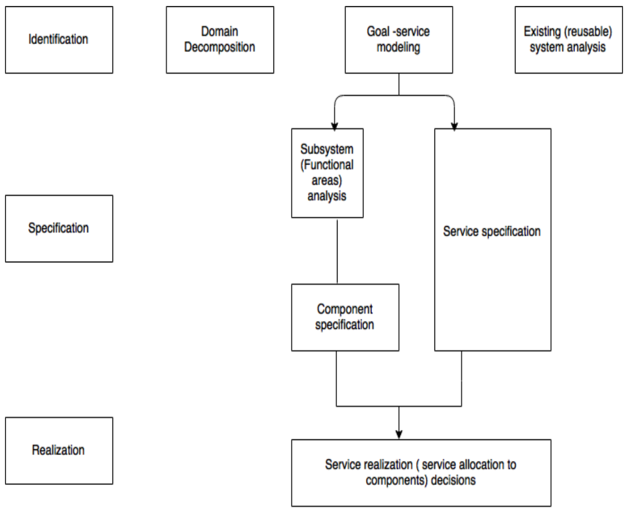
Identification
Domain Decomposition
Goal -service modeling
Existing (reusable) system analysis
Subsystem (Functional areas) analysis
Service specification
Specification
Component specification
Realization
Service realization (service allocation to components) decisions

Fig. 1: The service-oriented modelling and architecture method

It allows the simultaneous collaboration implemented through open standards, thus increasing reuse and maintainability. Such an architecture also segregates software applications into smaller manageable, self-contained units of code referred to as the services, to deliver relevant information across various channels. Services are often composed to create a business task. Eventually, business tasks are combined to execute a business method. And business processes functionally gather to form enterprise applications.

The subsequent sections in this paper elaborate on how the data across applications could be integrated as services to bring closer the different functional contexts by identifying an on-fly context-aware approach and presenting the user who does not need to recognize the underlying implementation technology and is just concerned with the inference posited to a particular query. The design plan for an SOA has a number of important activities and decisions with respect to the consumer and provider. These activities influence the integration, enterprise and application architectures. The provider's activities consist of all the activities of the consumer.

The service-oriented architecture and modeling as shown above consists of three major activities: identification, specification and realization of the services, the components and the flows.

Consequently, the service mechanisms facilitating QoS play a vital aspect in the existing architecture, as many business and other applications would want to avail of the services which accurately meet the requirements inclusive of the lag time experienced.

Challenges which cater the need of a more featured model are:

(a) A heterogeneous computing environment with many software solutions running on multiple platforms which are not intended to work together.
(b) Absence of real-time on-fly integration causing delays in delivering information to each function across channels, performance slowdown owing to complex integration issues.
(c) Discrete and independent autonomous business entities - which lack good connectivity.

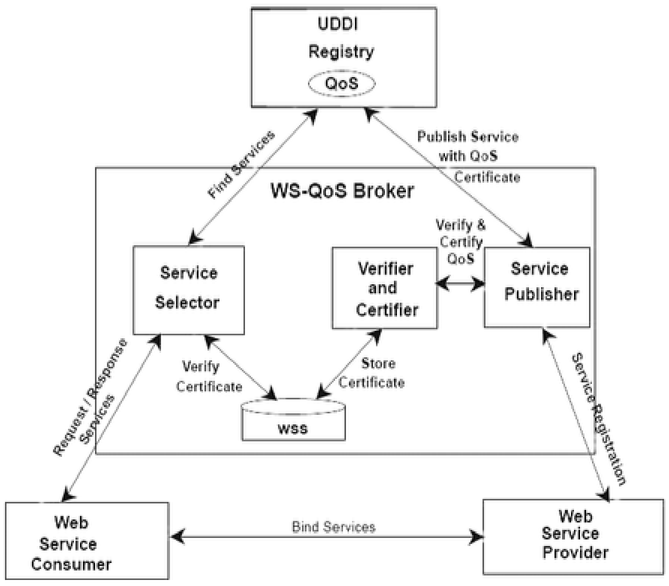

Fig. 2: QoS-Broker Model

The QoS characteristics are represented in UDDI registry by a tModel, which allows reuse, standardization and specification, of QoS related concepts. It enables the use of brokers to facilitate service selection according to functional and non-functional requirements, further which monitors to scrutinize QoS attributes. [10]

A tModel typically consists of a key, name, description and a Uniform Resource Locator (URL) that points to the location where details about the actual concept could be identified. When a service is published in the UDDI registry, a tModel representing the service QoS information is created. It is then eventually registered with the registry and referenced in the binding template that portrays the deployment information of the web service. In the tModel, every QoS metric is illustrated as a Keyed Reference containing the name of a QoS attribute as keyName and key Value containing the value. To update the QoS information, the web service provider searches the UDDI registry through the service publisher to locate the analogous tModel. Consequent updation of QoS information takes place in the tModel and also saves it back using the same key that was assigned to it on creation. Service is called a 'match' if it fulfils the customer's functional requirements and QoS limitations. If no matched service is found by the matching process, the service selector returns an empty result to the customer. If multiple services match the functional and QoS requirements, the service selector calculates a QoS score for each matched service based on the dominant QoS attribute specified by the customer, or on the default dominant attribute, average response time. The best service is assigned a score of 1, and the other services are assigned scores based on the value of the dominant QoS attribute. The top M services (M is the maximum number of services to be returned as specified by the customer) with the highest QoS scores are returned to the customer. If M is not specified, one service is randomly selected from those services whose QoS score is greater than Low Limit. [11]

4. PROPOSED MODEL

4.1 Standardization of Services

REST API could be used as in alternative and integrated as it uses XML/JSON to send or receive data in a human-readable format which makes the on fly integration easier to incorporate while requesting for parameters. It is assumed that the data access of the applications from wherein the parameters would be extracted would be rightly available to the user.

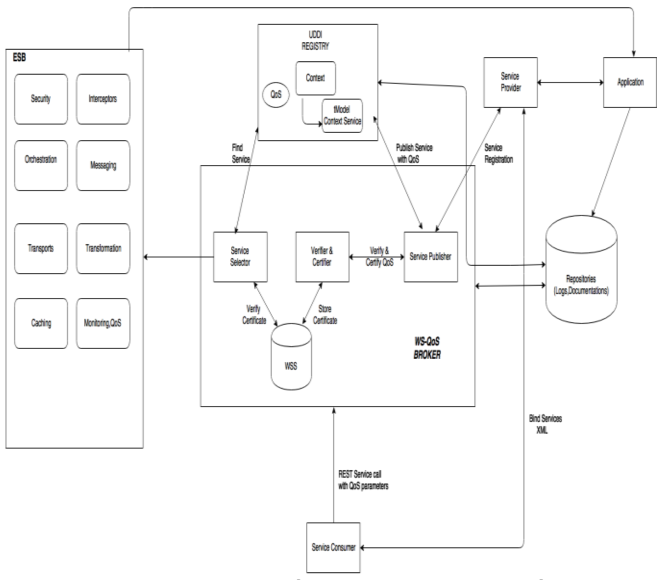

Fig. 3: Proposed Model for Standardisation of Services

The proposed model approaches following a data architecture instead of application architecture. Ensuring the SOA framework for Enterprise Application Integration (EAI), the functionality of ESB has proved to be the framework architectural pattern of how the integration is achieved. The core idea was to create services that are small, distinct units of software providing specific functionality and are reusable in every application. But only a top-down approach could not achieve the main purpose of SOA. And hence an ESB (Enterprise System Bus) is introduced in place of a full propriety stack. ESBs can power the creation and orchestration of services without requiring an application server or another infrastructure component, eliminating the high upfront costs of implementing SOA. They are developed according to open standards which facilitates flexibility to the business applications. Further, the WS-QoS Broker consisting of the service selector, verifier and certifier and the service publisher functions are posited in the existing model. In Web Service architecture, the exchange and flow of services involve the use of SOAP, WSDL and UDDI, but we introduce another service call known as REST API which accepts requests from the service consumer and directs them to the WS-QoS Broker. REST analyses operations pertaining to the data and inherits GET, POST, PUT, DELETE from HTTP which makes it easier to document and write services against. Unlike SOAP, REST is designed to be stateless, and its reads can be cached for better performance and scalability. It also allows for easy, quick calls to a URL for fast return responses. A request call with QoS parameters like price, time for achieving the response, jitter, bandwidth, latency, throughput and availability is given to the above specified broker. Then the service selector finds the services in the UDDI Registry to which it responds and verifies the certificate provided. The Verifier and Certifier verifies and certifies QoS with the service publisher and stores the certificate in WSS. Further, the service selector in the WS-QoS broker delegates the service to the ESB. ESB renders to provide security, caching, transformation and orchestration of the services. Alongside, it caters to secure messaging, transportation, monitoring of the services, QoS and also works as an interceptor and so on. It then devolves the service back to the application through the mode of mediation. The application then passes it to the service provider. Now the service consumer binds these services from the provider and receives an output in XML format. Repositories are maintained which consist of logs, documentation and references of the services registered in the UDDI Registry. It can be accessed by the application, UDDI Registry and the WS-QoS Broker as well.

4.2 Context-Aware Approach for Querying

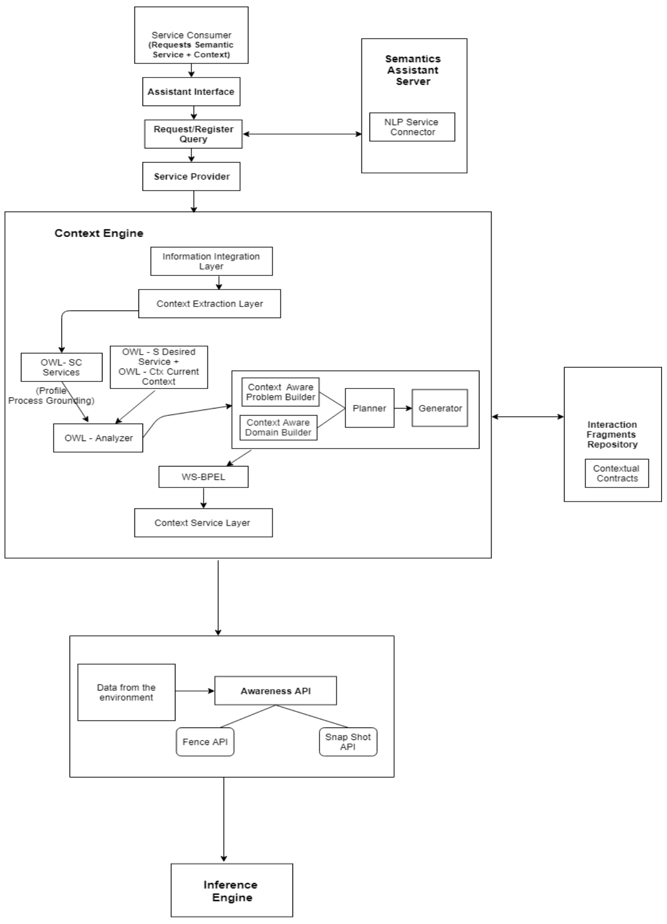
Service Consumer (Requests Semantic Service + Context)
Assistant Interface
Request/Register Query
Service Provider
Semantics Assistant Server
NLP Service Connector
Context Engine
Information Integration Layer
Context Extraction Layer
OWL- SC Services
(Profile Process Grounding)
OWL - S Desired Service + OWL - Ctx Current Context
OWL - Analyzer
Context Aware Problem Builder
Context Aware Domain Builder
Planner
Generator
WS-BPEL
Context Service Layer
Interaction Fragments Repository
Contextual Contracts
Data from the environment
Awareness API
Fence API
Snap Shot API
Inference Engine

Fig. 4: Proposed Model for Context-Aware Approach

In this approach, we present a model for the retrieval of the query posed by the user based in a conducive context environment. The user or any application requests a semantic query along with the context to the assistant interface. The interface then registers the query with the service provider who interacts with various other applications to obtain the required data. It first connects to the Semantic Assistant Server to extract NLP equivalents. The service provider is also aware of the context and the state of the application's request and passes it on to the context engine for further processing. [4] The context engine being an abstract term comprises of the information integration layer, the context extraction layer, context service layer and an Awareness API for extraction of context. The Information Integration Layer process the information obtained from the provider. It then delegates it to the next layer. The Context Extraction Layer processes the different contexts provided to it by the various content providers.

It presents the output in the form of ontology. The OWL-SC services include service description as three core concepts- Profile, Process and Grounding and is associated with a ContextAdaptationRule. [2] The OWL Analyser analyses the OWL files. The obtained output is presented as an input to the system along with the semantic description of the initial state of the environment as well as the final goal state. The Context-Aware Domain Builder evaluates the context correlating to the current context and generates an instance. This instance needs to be converted into PDDL (Planning Domain Definition Language). The Context-aware Problem Builder then augments the depiction by injecting into it relevant information from the context. Planner and Generator are in charge of generating a concrete representation of binding, process and grounding phases. Next, the Context Service Layer procures this representation and communicates to the Interactions Fragment Repository which consists of the contextual contracts. Data from the environment is processed by the Awareness API which is of two types-The Fence API and the SnapShot API. Finally, it then collects it from the above engine and sends it to the user as the WS-BPEL generator produces the resulting concrete process using the available context knowledge and grounding information.

5. RESULTS

The proposed system is on-fly context-aware if it takes into consideration the input contextual contract characteristics for the queries to produce the related data to the user. It is applicable in the case of a stock market application which needs to be real time by constantly updating data from the news from various, often distributed, sources [1].

Envisaging that a user wishes to run some queries concerning the stock market regarding which stocks to buy, the first stage of processing would encapsulate all the relevant and required data from the various applications ensuring it meets all the QoS parameters. A tModel entry is maintained for each service in the UDDI Registry which primarily deals with the management and communication between the Service Provider, WS QoS-Broker and ESB Bus using REST API Protocols.[3] The standardised output obtained from the various stock market applications such as extracting all the information regarding a company, their past trends, the number of shares bought, web crawling for updated news or even analysing the social media posts are parsed and stored in contextual contracts which does primarily two functions, provide the functional services with the context information and accordingly also update the other context contracts. A number of inferences in the form of APIs can be derived after processing it through the Context Engine. [5] Moreover, the NLP Service Connector allows the user to raise queries supporting multiple languages which adds to the efficiency. The Google Awareness API is also implemented which unifies signals such as time, location, places, beacons, headphones, activity and weather in a single API, enabling us to create powerful on-fly context-based features.

6. CONCLUSION

In this paper, we acknowledged the challenges that existed for the data integration platforms and suggested an architecture of hybrid technologies to overcome the barriers for successful application integration. Describing QoS architecture and using REST API for protocols with an underlying SOA framework for the different services of the application to interact with each other, provide a sustainable architecture to enable rapid extraction of heterogeneous data from various applications. With the standardised output, proper querying can be achieved by using the on-fly context-aware approach to obtain a set of inferences. Thus the data extracted by each application is encapsulated as inference and thereby combining the various technologies, the combined and optimised characteristics of the proposed architecture procure the aim of on the fly integration of various user requests and cater services.

7. REFERENCES

[1] Modeling and adapting to Context changes: The case of stock market decisions making Jacques Ajenstat1, Amir Padovitz, 2 Arkady Zaslavsky2 and Seng W. Loke 2.

[2] Scenario-Driven Development of Context-Aware Adaptive Web Services, Mahmoud Hussein, Jian Yu, Jun Han, Alan Colman

[3] Towards Context-Aware Adaptable Web Services Markus Keidl, Alfons Kemper

[4] An Integrated context-aware Planning Approach to Self-Adaptation Web Service Composition, Sihem Cherif, Raoudha Ben Djemaa, Ikram Amous

[5] Keidl M., Kemper A. (2004) A Framework for Context-Aware Adaptable Web Services. In: Bertino E. et al. (eds) Advances in Database Technology - EDBT 2004. EDBT 2004. Lecture Notes in Computer Science, vol 2992. Springer, Berlin, Heidelberg

[6] Challenges in Ubiquitous Data Management Michael J. Franklin EECS Computer Science Division University of California, Berkeley, CA 94720, USA

[7] Network Access and Security Issues in Ubiquitous Computing Upkar Varshney CIS Department, Georgia State University, Atlanta

[8] Enhancing Web Process Self-awareness with Context-aware Service Composition Angelo Furno

[9] Integrating Context-aware Pervasive Environments Muhammad Taimoor Khan, Kashif Zia, Dr Nadeem Daudpota, Dr S A Hussain1, Najma Taimoor .

[10] WS-QoSM: A Broker-based Architecture for Web Services QoS Management Elarbi Badidi1, Larbi Esmahi2, M. Adel Serhani3, Mohamed Elkoutbi4 1,3College of Information Technology, United Arab Emirates University, PO.Box. 17555, Al-Ain, United Arab Emirates.

[11] An Efficient WS-QoS Broker-Based Architecture for Web Services Selection T.Rajendran AP cum Research Scholar Department of CSE SNS College of Technology Dr.P.Balasubramanie Professor Department of CSE Kongu Engineering College Resmi Cherian Final Year ME (CSE) Department of CSE SNS College of Technology

[12] Challenges and Future of Enterprise Application Integration Tariq Rahim Soomro College of Engineering and IT Al Ain University of Science and Technology, Al Ain, UAE Abrar Hasnain Awan Department of Information Technology, SZABIST Dubai Campus, UAE.