ISSN 2320 – 2602
International Journal of Advances in Computer Science and Technology (IJACST), Vol. 3 No.2, Pages : 19 – 22 (2014)
Special Issue of ICCSIE 2014 - Held during February 16, 2014,Bangalore, India

Time Reduction Approach within Service Oriented Architecture(SOA) Framework Provide Status Report of The Service To Client

Ranaditya Haldar
CSIR-Central Mechanical Engineering
Research Institute
West Bengal,India
Durgapur-713209
+91-9051368379
sunshine.rana@gmail.com

Sandip Rakshit
Kaziranga University
NH-37,jorhat-785006
Assam,India
+91-9830781273
rakshitsandip@yahoo.com

ABSTRACT

In recent times , the use of Service Oriented Architecture (SOA) is becoming increasingly popular as the reliable architectural system, for developing a dynamic enterprise system, for ensuring the high quality of service aspect such as availability, reliability, security etc.In this paper we explore the issue of service availability for getting services within an SOA framework from the client side. We propose an algorithm within an SOA framework to ensure uninterrupted service availability, that is if any fault occurs during the period of processing of the service request. By providing a STATUS report, service requester can be automatically aware of the current status of the service. For our work, to establish the proposed algorithm, we are considering different processes as Semantic analyzer, Segregator, Send, Dynamic service composer, and one repository named Work details repository. These processes and one repository together are responsible for generating STATUS signal. And based on the status signal, service requester can control and make a decision the next step, according to his or her requirement.

KEYWORDS

Service oriented architecture (SOA),Dynamic service composer, Work details repository, Quality of service (QoS).

1. INTRODUCTION

Service-Oriented Architectures (SOA) provides a flexible and dynamic platform to implement open environments and distributed enterprise system. Its dynamism and loose coupling allow for automated service publication and discovery at run-time. In an SOA, services are self-described [1]. It means that along with each service there is a service description which defines its features. These descriptions are kept in a component called “service discovery” or “service registry”. Service requesters, depending on specific requirements, find a service in the service

Registry then binds to the service for execution. Due to the increasing need of high quality services in SOA, it is desirable to consider different Quality of Service (QoS) aspects of this architecture as security, availability, reliability, fault tolerance, etc. especially to develop critical dependable systems. In these systems, the occurrence of any fault is undesirable.

In this paper we find out a problem when a service requester initiates request within an SOA framework to get a service, it is impossible for a service requester to know which servers are responsible for providing that service and if the fault occur, during the time of processing the request, it is difficult for a service requester to know the status of the request.

2. DEFINITION

There are various definitions exists about service with different viewpoints [2, 3, 4]. We summarize characteristics of a service as follows: service is an action or operation directed by a service actor (a provider or requestor). Service is a marketing service in the sense of applying the marketing process: service advertisement, service discovery, and service engagement. Service is developed, deployed and invoked within a certain technical environment (e.g. Operating systems and component standards). Service communicates with the environment through its own interfaces, which encapsulate with clear specifications of what the service requires and provides. Service resides on IP (e.g Internet –Protocol) capable devices; it can be remotely accessed and invoked via Web-accessible terminals. A service can be used for application composition and service composition. Application composition is an abstract process of composing service descriptions. The composed application only becomes concrete at run time [5]. Composed applications do not provide interfaces for other services. In this sense, a composed application is volatile and cannot be deployed over the Internet, which means the composition information between elementary services will be lost after the application executes. In contrast, a service composition has the same process as the application composition, but the composed service provides interfaces for other services and applications, which can be deployed over the Internet. The composite service will be stored and registered with the composition information (i.e. Service dependency) between elementary services for further service composition. The person who manages the service composition is called service composer. Service dependency is beyond traditional poor service description, and directed by various sources, such as data, resource, procedure control, utilizing techniques, etc. Dependency-aware service management stresses on a complete service description. Elementary service is a service that does not have a service dependency with other services. In contrast, composite service is compossed of elementary services, which has at least a kind of service dependency with other services.

3. SOA FRAMEWORK TIME REDUCTION APPROACH

3.1 PROPOSED WORK

We are considering a SOA framework with only 3 levels shown in Figure 1. 1st level of this model consists Client, 2nd level consists L1_server and 3rd level consists multiple number of servers named L2_server #1, L2_server #2, L2_server #3.

In our work, we are trying to ensure that if fault occurs, during the time of processing service requester's request within SOA framework, then requester receives the present status of that request, by which client can make a further decision.

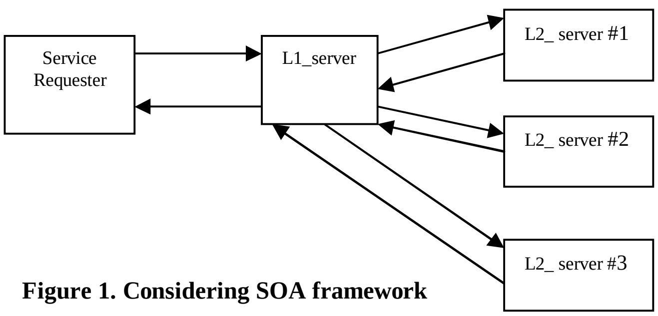

Figure 1. Considering SOA framework

3.2 JOBS OF DIFFERENT LAYERS

3.2.1 JOB OF SERVICE REQUESTER

1. Service Requester will initiate the request and wait for the reply.

3.2.2 JOBS OF L1_SERVER

1. Receives the request from Client and segregates the request.
2. Sends the segregated request to corresponding L2_servers.
3. Composes all reply, which comes from L2_servers and send as a service to client.

3.2.3 JOBS OF L2_SERVER

1. Accepts the request from L1_server.
2. Processes the request.
3. Send the reply to L1_server.

3.3 PROCESSES AND RESOURCES NEED TO PERFORM THE JOB

(a) For job no 1 of L1_server(Receive the request from client and segregates the request)

Required processes are

Semantic analyzer: This process helps to analyze the composite request semantically, comes from service requester.

Segregator:- Based on the output of semantic analyzer and with the help of "**Work_details_repository**" segregator process segregates the composite request, in the form of piece meal of request.

Required resource is

Work_details_repository:- This repository keeps record about which L2_server processes what type of request. Diagrammatic view of processes and resources for job no 1 of L1_server has shown in Figure 2.

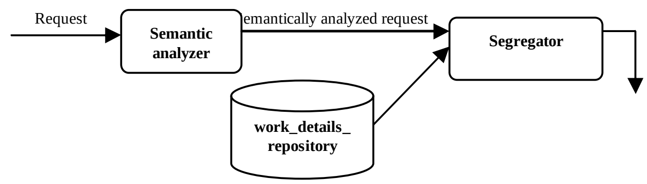

Figure 2. Diagrammatic view of processes and resource

(b) For job no ii of L1_server "(Sends the segregated request to L2_servers)"

Required Processes is

Send process: Send process will send each segregated request to the corresponding L2_server.

Required resource is

Work_details_repository:- This repository keeps record about which L2_server processes what kind of request. Diagrammatic view of processes and resources for job no 2 of L1_server has shown in Figure 3.

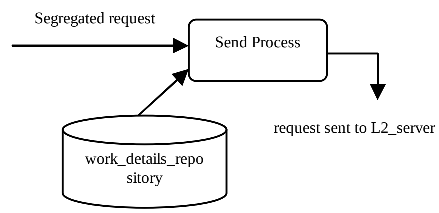

Figure 3. Diagrammatic view of process and resource

(a) For job no 3 of L1_server "(Composes all reply, which comes from L2_servers and send as a service to client)"

Required Processes are

Dynamic service composer:- The task of this process is to dynamically compose all reply, which are coming from different L2_servers.

ISSN 2320 – 2602
International Journal of Advances in Computer Science and Technology (IJACST), Vol. 3 No.2, Pages : 19 – 22 (2014)
Special Issue of ICCSIE 2014 - Held during February 16, 2014,Bangalore, India

Send Process:- Send process will send the composed reply to client as a service. Diagrammatic view of this process for job 3 of L1_server has shown in Figure 4.

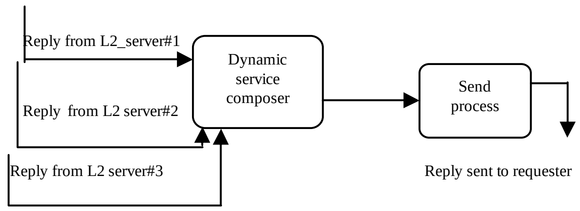

Figure 4. Diagrammatic view of processes

(d) For job no 1 of L2_server"(Accepts the request from L1_server)"

Required Process:

No process is required.

(e) For job no 2 of L2_server "(Processes the request)" No process is required.

f) For job no 3 of L2_server"(Sends the reply to L1_server)"

Required process is

Send Process:- Send process will send the reply of a request, to L1_server. Diagrammatic view of this process for job 3 of L2_server has shown in Figure 5.

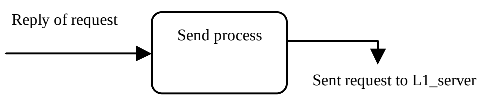

Figure 5. Diagrammatic view of processes

4. PROPOSED ALGORITHM

Begin

1. Client sends the request to L1_server.
2. L1_server segregates the request.
3. L1_server searches "work_details_repository" to find corresponding L2_server for processing segregated requests.
4. L1_server searches "time_repository" for corresponding L2_server which has already identified by searching "work_details_repository" to find out the time requires to process the request.
5. If L1_server finds out the time duration for L2_server likes T1 for L2_server#1, T2 for L2_server#2,T3 for L2_server#3 and these hold T1>T2>T3 then L1_server assigns T3 time duration with each segregated request and send them to corresponding 3rd level L2_server.
6. Monitoring mechanism of L2_server will accept the request, and read corresponding attached time duration of each request,
7. Request will enter into the job queue of the system, for allocation to the processor of L2_server.
8. Reply of the corresponding request goes from 3rd level servers(L2_server) to L1_server through monitoring mechanism.
9. **IF** (monitoring mechanism finds out that reply of a particular request is not sent to L1_server within the time duration of that request)

   9.1. Monitoring mechanism of that particular 3rd level server(L2_server) will create a high priority process which initiates a STATUS signal consists current state of that process

   9.2. STATUS signal will sent to L1_server and L1_server will further forward this signal to Client.

   9.3. Client will make the decision whether to wait for the service or reject the service.

10. **END IF**
11. **ELSE**
12. After receiving all reply from all L2_servers, L1_server will compose the service and send to Client.

End

5. LIMITATIONS OF OUR PROPOSED APPROACH

i). Network Failure:

If any network failure is occurred in any Levels of server, according to our proposed method client does not get any information of initiated service.

ii). Monitoring mechanism fails to work:

If monitoring mechanism in any 3rd level servers(L2_server) does not work according to our proposed work, then service requester does not receive any message from that corresponding L2_server.

iii). Server crashes at 3rd level(L2_server):

If single server or multiple servers crash at 3rd level(L2_server), according to our proposed approach service requester will be unaware about the current state of the service.

6. CONCLUSION

Service-oriented architectures can support applications with sensitive, personally identifiable information. In such cases a new challenge is to enhance application functionality and flexibility with business-process management. Business processes in heterogeneous, open environments raise new requirements for proper service availability. In this work, we suggested a quality enhancement method as a time reduction approach to be encapsulated inside web service running for SOA framework. This proposed algorithm is totally based on our proposed SOA framework and to be implemented by web service related technologies like Web Service Description Language, Simple Object Access Protocol etc.

Future work will be based on robustness of the this proposed algorithm. Our approach is likely to be a solution for service availability that fits well into existing solutions of business-process-management systems.

REFERENCES

[1] Baresi L., Heckel R., Thöne S. and Varró D.: Style- Based Modeling and Refinement of Service-Oriented Architectures: A Graph Transformation-Based Approach, Journal of Software and System Modeling, vol. 5, 187-207, (2006)

[2] "OSGi Service Platform Specifications Release 4," http : //osgi.org/osgi_technology/download_specs.as p? section =2#Release4, 4 May, 2007.

[3] J. Zhou and E. Niemela, "Beyond developmentoriented software engineering: Service-OrientedSoftware Engineering (SOSE)," in Service- Oriented Software System Engineering: Challenges and Practices, Z. Stojanovic and A. Dahanayake, Eds. Hershey, USA: IDEA Group Publishing,2005, pp. 27-47.

[4] M. Fowler, "Inversion of Control Containers and the Dependency Injection pattern," http://martinfowler.com/articles /injection.html#ConstructorVersusSetterInjection, 4 May, 2007.

[5] Dependency Management in a Service-Oriented Component Model," Proceeding of the 6th International Workshop on Component-Based Software Engineering - (CBSE), Portland, USA, 2003.