Dynamic Service Choreography using Context Aware Enterprise Service Bus

Abstract—Enterprise Service Bus (ESB) is responsible for publishing and discovery of services in a global distributed delivery system. Context-aware systems offer entirely new opportunities for application developers and for end users by gathering context data and adapting systems’ behavior accordingly. In this paper, we propose a Context Aware ESB (CA-ESB) that will publish and discover services based on location context. The main modules of the framework consist of Context Provider (senses location context), Context Aware Logic Module (decides which regional service to be selected based on location context) and Service Choreographer (choreographs selected services). We propose a graphical model named Context Aware Graph (CA-Graph) that will help us to dynamically choreograph the services. These modules along with other modules of SOA reference architecture will help the ESB to sense the location of users, to select the required services and dynamically choreograph those services. We define a set of metrics based on CA-graph and analyze the performance of CA-ESB. An algorithm is proposed that will dynamically choreograph the selected services based on location context. The results of the case study of an Insurance System are used to illustrate our approach.

Keywords— Cloud Computing, Context-aware, Enterprise Service Bus, SOA based global delivery model, dynamic service choreography, CA-Graph

I. INTRODUCTION

Distributed Delivery Model[11] is a bi-directional sequence of activities consisting of requirements specification, analysis, design, development, integration, testing, and maintenance. These activities may not always be performed in a linear fashion, as there may be some overlap between and across certain processes. The distributed delivery model has gained immense importance as these days software is developed in a distributed manner with a common core component and various regional components interfacing with it.

Service-orientation requires loose coupling of services with operating systems, and other technologies that underlie applications. SOA separates functions into distinct units known as services, which developers make accessible over a network in order to allow users to combine and reuse them in the production of applications. These services and their corresponding consumers communicate with each other by passing data in a well-defined, shared format, or by coordinating an activity between two or more services.

In the distributed delivery model the services can be divided into core services and regional services. The core services will remain same for all applications and regional services will vary from one application to another application. The regional services interface with the global services to serve the overall business function requirements at each location. In this paper, we propose a Context Aware ESB (CA-ESB) that will publish and discover services based on location context. This type of ESB framework is very relevant for region specific global development scenario. A new graphical model named Context Aware Graph (Ca-Graph) is proposed that models the services/processes and their interconnections for all the locations. Discovery of services becomes much simpler using CA-ESB as the CA-Graph proposed here unveils the service choreography for each location dynamically. Analysis of CA-Graph using a metrics proposed in this paper clearly illustrates the performance of CA-ESB vis-à-vis a traditional ESB.

II. RELATED WORK

Areas related to SOA have become prominent research domains. Here we discuss about some of the relevant research works in the field of Enterprise service bus (ESB).

A dependable ESB framework that enables automated recovery from component failures is proposed in [1]. The authors in [2] propose an ESB framework to enable the content-based intelligent routing path construction and message routing. [3] presents computation-independent models (CIMs) and platform independent models (PIMs) for service oriented architectures.

Existing ESB-based system have difficulties to manage complex events of real-world applications very well. A complex event-processing model based on the relational algebra is proposed in [4], and then it proposes a complex event processing oriented enterprise service bus. [5] presents Omnipresent, which is a service-oriented architecture for
context-aware applications that may be accessed from either mobile devices or Web browsers, and it is based on Web services page layout. In [8], a design of a Dynamic Composition Handler on Enterprise Service Bus (ESB) is presented which analyze different types of service compositions to clarify what dynamic composition really holds in SOC. In [9], service composition and discovery are not treated separately. Here a matching algorithm is developed that combines several services which are not known and need to be discovered. In [10], the flexible architecture of the discovery engine Glue2 is proposed which comes with a powerful set of discovery components (for functional matching, non-functional matching, data fetching, etc.) that can be executed in different order as required by specific execution workflows.

In our previous work in [7] a graph named D-SG is proposed to model design models in distributed environment. The sequence diagrams are interleaved such that the business process at a location spans several sequence diagrams modeling the common and regional use cases. This is modeled graphically using Distributed Scenario Graph (D-SG). Our work in this paper closely relates to this graph model.

In the domain of context aware software architectures, the current work proposes to encompass the design of a context aware ESB that will enable publishing and discovery of services based on location context of the users. Our approach is to incorporate context awareness to ESB unlike other existing work. Incorporating context awareness to ESB will enable the developer to control the algorithms related to context awareness centrally in the ESB (which act as a middleware) where as in the existing works in Context Aware SOA the context awareness is achieved through end devices. This kind of ESB framework will be of immense importance in global software development scenario where services can be region specific of the users.

III. SCOPE OF WORK
In this work, Context Aware Graph (Ca-Graph) is proposed that models the services/processes and their interconnections for all the locations. In this framework, the traditional ESB with the help of Ca-graph and location information functions in such a manner that it gets the essence of CA-ESB that performs dynamic service choreography. An Algorithm is proposed which will perform the dynamic choreography of services based on location context. Some metrics are also proposed to analyse the performance of this CA-ESB framework.

In this work we have used some terms to describe our work. Before the description of our proposed work, we will define those terminology that are frequently used throughout this paper for better understanding of our work.

- **Service Vs Process**
  The term *Process* is used when we describe a specific task in the Analysis Phase of SOA software development. Different processes are arranged together (choreographed) to render a specific functionality. *Services* on the other hand are loosely coupled entities spread over a distributed system. Once the processes are choreographed for a particular application, they will be mapped with the loosely coupled services. So, processes are tightly coupled logical entities and services are loosely coupled physical entities. In this work, we will use *process* in parallel with *service*. Once the processes are choreographed, the same process will be mapped with the services and that service will be either constructed or discovered in ESB.

- **Regional Processes Vs Common Processes**
  In a global development scenario, the s/w development is done for requirements that cross regional boundaries. In an application, there will be multiple region specific processes for the same functionality. For example, GUI interface will be different for different location (language specific). Also, different regions have different business policies leading to multiple processes of the same function.

IV. PROPOSED CA-ESB FRAMEWORK
The CA-ESB (Context Aware Enterprise Service Bus) framework is shown in Figure 1.

The modules of CA-ESB are:

A. **Service Consumer**
They are the users of the application. In our context aware scenario, they are spread over multiple regions. Their request will be served based on the location context provided by the Context Provider.

B. **Context Provider**
Context Provider will provide location context of the users. The enterprise service bus will be provided with location context of the users. Based on this location context, the regional services will be chosen by the ESB. The location context can be the URL or IP address of the users. The output of this module (URL or IP address) will be forwarded to Context Aware Logic Module (CAL) for selecting region specific services.

C. **Context Aware Logic Module (CAL)**
Using the context data (i.e. The IP address or URL) of the user, this module will choose region specific services that are different for different users. Different regions have different business policies, different GUIs etc. So, the services will be different for different users of different location. This module will store the information regarding region specific services and select those services based on the location context of particular user. These selected
services will be used for choreography by the next module.

![Diagram of the CA-ESB Framework](image)

**Figure 1: The CA-ESB Framework**

D. **Service Choreographer**

This module choreographs the services chosen by CAL dynamically at run time. Services are regional services as well as core services. Dynamic choreography is required as the regional services are different for different applications (or users).

The fifth module is the traditional enterprise service buses that will work along with the other four modules giving the flavor of CA-ESB.

V. **DYNAMIC SERVICE CHOREOGRAPHY AND EFFICIENCY OF CA-ESB**

The Context aware logic (CAL) module of the CA-ESB framework stores the region specific process/services information and selects services accordingly when location information is acquired and sent by the context provider. To model the process/service information in CAL module, we propose a Context Aware Graph and calculate the efficiency of CA-ESB compared to traditional ESB using the proposed graph. CA-Graph is used to model the processes of an application and their interconnections for different locations of users.

A. **CA-Graph: Graphical representation of processes**

We propose a graph called Context Aware Graph (CA-Graph) that will help us to categorize the processes of the application according to the location of the users. The processes represent the business processes of the SOA reference Architecture [6].

The processes will be mapped with the services in the service choreography module. The CA-Graph = (V, E) is a graph comprising of nodes/vertices and directed edges. The vertices represent processes and edges are drawn to connect the vertices based on the interconnection of the processes. Different graph construct for CA-graph is tabulated in table I.

<table>
<thead>
<tr>
<th>Graph Const</th>
<th>Graph Construct</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.L1</td>
<td>Naming syntax for regional process means Process 1 for location 1</td>
</tr>
<tr>
<td>2</td>
<td>2.C</td>
<td>Naming syntax for common processes which means process 2 for all location</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Flow of events (or processes)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Process ‘a’</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Exclusive-OR flow which mean from process ‘a’ the flow will move to either ‘b’ or ‘c’ depending on some condition</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>Parallel flow which means from event(process) ‘b’ and ‘c’ will occur in parallel after ‘a’</td>
</tr>
</tbody>
</table>

The key points of the graph are:

- Regional processes are labelled as <process name>. < location name > Eg. 1. L1, 2. L2 etc where L1, L2 etc stands for different location.
- Common processes will have extension C.
- Solid Arrow is used for flow of events
- Circle indicates events/process.

B. **CA-ESB and CA-graph**

The process information in a tabular form will be stored in the Context Aware Logic (CAL) part of our proposed CA-ESB framework. This process represents a collective main process that consists of sub processes. This tabular information is named as Process Table. The Process Table has the following fields

- **Sub process name**: The name of sub processes that constitutes the main process.
- **Pre process**: This is the set of probable processes preceding the process.
- **Post process**: This is the set of probable processes succeeding the process.
- **Type of Process**: This field indicates whether the processes are regional (R) or common (C).
- **Process Flow Type**: This field indicates the flow of the process with its post process(es). This can be normal flow (N), parallel flow (P) or exclusive-OR flow (E).
- **Total Processes**: This field consists of list of all possible location based processes for a particular sub process.

Table III in appendix represents a process table for the case study of insurance system that is explained in Section VI.
When the context provider provides the information regarding the location, the CAL module chooses the processes for that particular location from the process table along with common processes. These processes will be mapped with the services. The next section briefly discusses the process of dynamic choreography.

C. DYNAMIC SERVICE CHOREOGRAPHY

The service choreographer does the dynamic choreography of services with the selected processes at runtime based on the process connections as modelled in CA-Graph.

As an example, if the context provider senses the location as location1 (L1), it will send the information to the CAL module. CAL module has the information regarding which sub processes are required for location L1 and Service Choreographer will do dynamic service choreography at runtime with the selected processes (or services). Once the services required for a particular location are determined, the ESB will publish only those services in the registry. This will reduce the time associated with the discovery of services.

The following algorithm will identify services based on location context from the process table and identify the scenario path composed of services in a particular order.

**Algorithm:**

a) Input: Location context as LocationName
b) Data Structure to be used:
   - Graph construct schema table (TABLE I)
     Schema: GraphID, GrConstruct
   - Process table (TABLE III of appendix)
     Schema: ProcessId, ProcesssName, PreProcess, PostProcess, ProcessType, ProcessID, ProcessFlowType
   - ProcessFlow
     Schema: Source, Destination
c) Steps to be followed:
   1. L := GetLocationInput;
   2. Set ProcessIDLoc := 1;
   3. LOOP
      T = GetTuple from Process Table where (ProcessId = ProcessIDLoc);
      If (ProcessFlowType = ‘N’) then
         ProcessFlow = (Select GrConstruct when GraphId = 3 from Graph construct table)
      Else If (ProcessFlowType = ‘E’) then
         ProcessFlow = (Select GrConstruct when GraphId = 5 from Graph construct table)
      Else If (ProcessFlowType = ‘P’) then
         ProcessFlow = (Select GrConstruct when GraphId = 6 from Graph construct table)
      8. If T.ProcessType == ‘C’
         Then ProcessIDLoc = T. PostProcess
      9. Else If T.ProcessType == ‘R’ then
         T = GetTuple from Process table where (LocationName = L AND ProcessID=ProcessIDLoc)
         Set ProcessIDLoc := ProcessID. LocationName;
      END LOOP
d) Output:
Choreographed process based on the particular location context (As Figure 3)

D. Performance Analysis

In this section, we calculate the complexity metrics and search metrics for the CA-ESB. The notations used to evaluate different performance metrics of CA-ESB is given in table II.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{mx}$</td>
<td>Complexity metrics of CA-Graph for traditional ESB</td>
</tr>
<tr>
<td>$C_{mx}(CA)$</td>
<td>Complexity metrics of CA-Graph for CA-ESB</td>
</tr>
<tr>
<td>$T_{mx}$</td>
<td>Search metrics of CA-Graph for traditional ESB</td>
</tr>
<tr>
<td>$T_{mx}(CA)$</td>
<td>Search metrics of CA-Graph for CA-ESB</td>
</tr>
</tbody>
</table>

- Complexity Metrics

Here we will calculate the complexity metrics for the CA-ESB. This metrics will give the measure of no. of processes and interconnections giving us an idea of the complexity of the application.

If we suppose a Global development scenario that consists of:
- No. of common processes = m
- No. of regional processes for a particular main process = n

No of location = L

Total no. of regional process = (No of location) x (No. of regional processes for a particular main process) = n*L

So, total no. of processes = (No. of common processes) x (No. of regional processes) = m + n*L

In a CA-Graph, the processes (common as well as regional) are represented as the vertices of a graph and the communication between processes are represented as the edges of the graph.

Total no. of vertices of the CA-graph = m + n*L

Then, the maximum complexity of the graph (when all the m+n*L processes communicate with all other processes)

$$C_{mx} = (m + n*L) (m + n*L - 1) \text{----------------------------------- (1)}$$
When CA-ESB using Context Aware Logic discovers the services, only the services pertaining to a particular location along with common services become visible in the CA-ESB for discovery. As a result the graph effectively reduces to Subgraph for a particular region having m+n processes. The maximum complexity of the graph (when all the m+n processes communicate with all other processes)
\[ C_{mx}(CA) = (m + n) \times (m + n - 1) \]  
(2)

The maximum complexity of the graph (when all the m+n processes communicate with all other processes)
\[ C_{mx}(CA) = (m + n) \times (m + n - 1) \]  
(2)

The maximum complexity of the graph (when all the m+n processes communicate with all other processes)
\[ C_{mx}(CA) = (m + n) \times (m + n - 1) \]  
(2)

As evident from the graph in figure 2 that the Cmx (CA) values are lower than the Cmx values indication that the use of CA-ESB significantly reduces complexity.

- **Search Metrics**

We use a metrics to mathematically compare the time complexity involved in searching and discovering services in a CA-ESB as against a normal ESB.

If CA-ESB is not used, all the processes/services will be visible, the total vertices (processes/services) to be discovered is
\[ T_{mx} = m + n*L \]  
(3)

When CA-ESB is used, the total vertices (processes/services) to be discovered only pertains to a particular location,
\[ T_{mx}(CA) = m + n \]  
(4)

So, Search complexity without using CA-ESB
\[ T_{mx} = O(n*L) \]

Search Complexity when CA-ESB is used
\[ T_{mx}(CA) = O(n) \]

From (3) and (4), the time complexity of searching of services in the ESB is reduced by L times if the CA-ESB is used.

**VI. CASE STUDY**

We consider an application for an Insurance company who plans to start its business operations in several countries across the globe. The application consists of different processes, which are being used by different users located in different places. Some processes are common which are being used by all users who are accessing the application irrespective of where they are located. While some processes are specific to a country. Let us consider that the application consist the following main processes

- Policy Creation
- Policy Maintenance
- Policy Claim
- Policy Termination

The flow of events (or sub processes) of the first processes “Policy Creation” is given in a tabular form as Table III of appendix. The processes are categorized as common processes (marked as ‘C’) and regional processes (marked as ‘R’). Here, the processes are considered for two locations namely L1 and L2.

The main process (“Policy Creation”) consists of 10 sub processes that are labelled according to the rule defined in section V.A. CA-Graph is generated for process “Policy Creation” as in Figure 3.

\[ C_{mx} = (m+nL) \times (m+nL - 1) \]  
\[ = 20 \times 19 = 380, \]

\[ C_{mx} = (m+nL) \times (m+nL - 1) \]  
\[ = 11 \times 10 = 110 \]  
(Using equation (1) and (2))

\[ T_{mx} = 20 \]  
and \[ T_{mx}(CA) = 11 \]  
(Using equation (3) and (4))
Figure 4 shows a performance metrics chart that depicts the fact that the performance of CA-ESB is improved both in terms of complexity metrics ($C_{mx}$) and search metrics ($T_{mx}$) as compare to traditional ESB for the process Policy Creation which is distributed in two location ($L=2$). With the increase of the value of $L$, we will get more increase in performance of CA-ESB as compare to a traditional ESB.

VII. CONCLUSION
Presently, global delivery model is gaining significance where applications are being developed in an integrated manner for different users spread over geographically different locations forming a cloud. The core processes remain same, with several region specific processes catering to different because of variations of languages, currency, business policies, etc. In SOA architecture, ESB is responsible for publishing and discovery of all services. We propose a new variation of ESB named Context Aware ESB (CA-ESB) that will be very useful for publishing and discovery of services in a global development scenario. CA-ESB is able to sense location context, selectively discover relevant services for a region and finally dynamically choreograph them with the core services such that the whole application behaves uniquely for each different location context. We demonstrate using an algorithm and a set of metrics that the efficiency of a global software development scenario improves to a significant extent by using CA-ESB. This framework attempts to provide solution for the problem of efficient dynamic coordination of geographically distributed services.

REFERENCES


Appendix

Table III: Process Table for the Policy Creation

<table>
<thead>
<tr>
<th>Sub Process No</th>
<th>Sub Process Name for Policy Creation</th>
<th>Pre-Process</th>
<th>Post Process</th>
<th>Type of Process (R/C)</th>
<th>Process Flow Type</th>
<th>Total Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Application Entry</td>
<td>-</td>
<td>2</td>
<td>R</td>
<td>N</td>
<td>1.L1 1.L2</td>
</tr>
<tr>
<td>2</td>
<td>Underwriting</td>
<td>3 or 5</td>
<td>C</td>
<td>E</td>
<td>2.C</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Underwriting 1</td>
<td>2</td>
<td>4</td>
<td>R</td>
<td>N</td>
<td>3.L1 3.L2</td>
</tr>
<tr>
<td>4</td>
<td>Process Error</td>
<td>3,6</td>
<td>7</td>
<td>R</td>
<td>N</td>
<td>4.L1 4.L24</td>
</tr>
<tr>
<td>5</td>
<td>Accept More Details</td>
<td>2</td>
<td>6</td>
<td>R</td>
<td>N</td>
<td>5.L1 5.L2</td>
</tr>
<tr>
<td>7</td>
<td>Process Error</td>
<td>3,6</td>
<td>7</td>
<td>R</td>
<td>N</td>
<td>7.L1 7.L2</td>
</tr>
<tr>
<td>8</td>
<td>Premium Calculation</td>
<td>4</td>
<td>8</td>
<td>R</td>
<td>N</td>
<td>8.L1 8.L2</td>
</tr>
<tr>
<td>9</td>
<td>Premium Payment</td>
<td>7</td>
<td>9</td>
<td>R</td>
<td>N</td>
<td>9.L1 9.L2</td>
</tr>
<tr>
<td>10</td>
<td>Policy Issuance</td>
<td>8</td>
<td>10</td>
<td>C</td>
<td>N</td>
<td>10.C</td>
</tr>
</tbody>
</table>