CA-ESB: Context Aware Enterprise Service Bus

Jayeeta Chanda
B.P.Poddar Institute of Management &Technology
Kolkata, India

Sabnam Sengupta
B.P.Poddar Institute of Management &Technology
Kolkata, India

Ananya Kanjilal
B.P.Poddar Institute of Management &Technology
Kolkata, India

Swapan Bhattacharya
Jadavpur University
Kolkata ,India

ABSTRACT

Enterprise Service Bus (ESB) is responsible for publishing and discovery of services in such environments. 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 analyse of performance 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.

General Terms

Service Oriented Architecture

Keywords

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

1. 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.

Using the SOA approach, 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. Enterprise Service Bus (ESB) is responsible for publication, registration and discovery of services in distributed heterogeneous environments. 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. The common/core services and the location specific services are represented as nodes and the edges are formed based on the interconnection of these services to implement a requirement scenario. Discovery of services becomes much simpler using CA-ESB as the CA-Graph 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.

2. REVIEW OF 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.

3. SCOPE OF WORK

In this paper, we propose a model to publish and discover services in a distributed delivery model. In a geographically distributed development scenario, there are some services that are region specific. These services may include language specific user interfaces, region specific business policies, different currencies etc. The region specific services will have multiple location dependent copies for a single service. The no. of copies will be equal to the no. of location. If these services are maintained in the registry of a traditional ESB, the discovery of services will be very cumbersome because in an application for specific services of a particular location, the discovery of services will be very complex. The CA-ESB framework is proposed such that it can sense the location context from the URL or IP address of the service consumer (Context Provider) chose the region specific services along with core services using context aware logic and perform service choreography dynamically with the selected regional services as well as core services by traditional ESB..

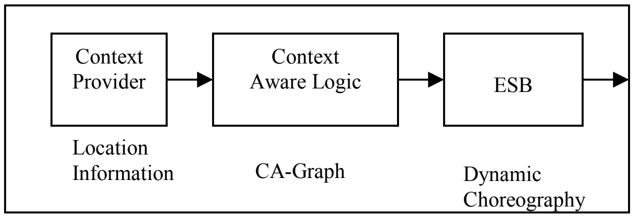


Figure 1: Roadmap to our approach

Dynamic Choreography, Context Aware Logic and Context Provider add the essence of context awareness to the Enterprise Service Bus While developing the application, the publishing of services can also be location context aware. The service registry will display services that are specific for that particular region (for which the application is developed).

The scope of this paper is depicted in Figure 1. Here, 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.

4. SOME COMMON TERMINOLOGY USED

In this work we have used some terms to describe our work. Before the description of our proposed work , in this section we will define those jargons/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. *Service*s 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.

5. PROPOSED CA-ESB FRAMEWORK

The CA-ESB (Context Aware Enterprise Service Bus) framework is shown in Figure 2.
The modules of CA-ESB are:

5.1 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.

5.2 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.

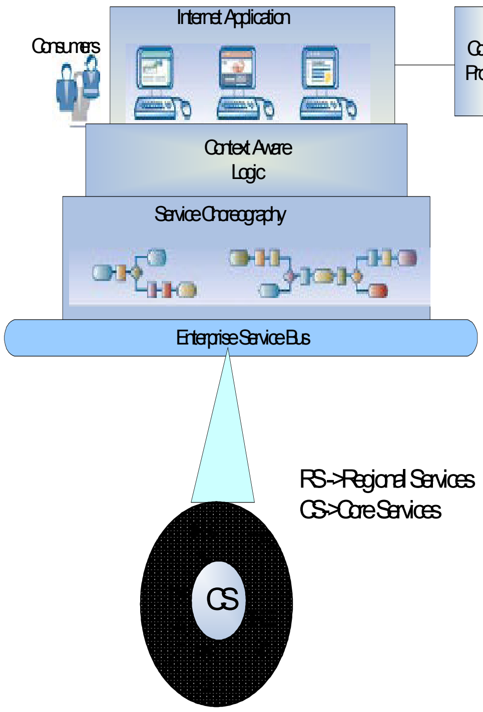


Figure 2: The CA-ESB Framework

5.3 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.

5.4 Service Choreographer

This module choreographs the services chosen by CAL dynamically in 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 give the flavor of CA-ESB.

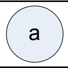
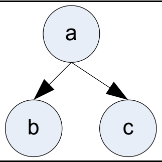
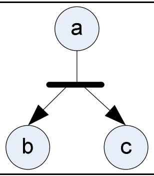
6. DYNAMIC SERVICE CHREOGRAPHY AND EFFICIENCY OF CA-ESB

The Context aware logic (CAL) module of the CA-ESB framework stores the region specific process/ services information and select 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.

6.1 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].

TABLE 1: CA-GRAPH CONSTRUCTS

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

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 1.
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.

6.2 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 II 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.

6.2 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 II of appendix)
      Schema: ProcessId, ProcesssName, PreProcess,
      PostProcess, ProcessType, ProcessID,
      ProcessFlowType
    ❑ ProcessFlow
      Schema:  Source, Destination

c) Steps to be followed:
  1 .L: = GetLocationInput;
 # ProcessIDLoc is used to store instantaneous ProcessId locally
  2. Set ProcessIDLoc: = 1;
  3.  LOOP
        T = GetTuple from Process Table
        where (ProcessId = ProcessIDLoc);
  4.  If (ProcessFlowType = 'N') then
      ProcessFlow = (Select GrConstruct when GraphId = 3
      from Graph construct table)
  5.  Else If (ProcessFlowType = 'E') then
      ProcessFlow = (Select GrConstruct when GraphId = 5
      from Graph construct table)
  6. Else If (ProcessFlowType = 'P') then
      ProcessFlow = (Select GrConstruct when GraphId = 6
      from Graph construct table)
  7. Set ProcessFlow. Source: = T. ProcessId and
        ProcessFlow. Destination = T. PostProcess
  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;
      Else continue;
END LOOP
d) Output:
Choreographed process based on the particular location context (As Figure 5 and Figure 6 of the Appendix)
```

6.3 Notation Used for Performance Analysis

The notations used to evaluate different performance metrics of CA-ESB is given in table II.

TABLE II: Notations

Notation	Meaning
C_{mx}	Complexity metrics of CA-Graph for traditional ESB
$C_{mx(}(CA)$	Complexity metrics of CA-Graph for CA-ESB
T_{mx}	Search metrics of CA-Graph for traditional ESB
$T_{mx(}(CA)$	Search metrics of CA-Graph for CA-ESB

6.4 Complexity Metrics

In this section, we 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)

$$C_{mx}(CA) = (m + n)(m + n - 1) \text{------------------------(2)}$$

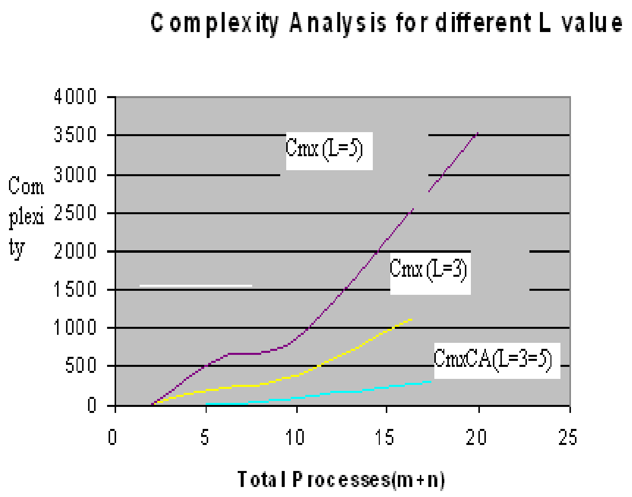


Figure 3: Comparison of CA-ESB vis-à-vis traditional ESB

Based on equations (1) and (2) we chose different values of L=1, 3 and 5 and plotted the C_{mx} and C_{mx} (CA) values corresponding to normal ESB and CA-ESB respectively. As evident from the graph in figure 3 that the C_{mx} (CA) values are lower than the C_{mx} values indication that the use of CA-ESB significantly reduces complexity.

6.5 Search Metrics

This section uses 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

$$T_{mx} = m + n * L \text{-------------------------------------- (3)}$$

When CA-ESB is used, the total vertices (processes/ services) to be discovered only pertains to a particular location,

$$T_{mx}(CA) = m + n \text{ -------------------------------------- (4)}$$

So, Search complexity without using CA-ESB

$$T_{mx} = O(n * L)$$

Search Complexity when CA-ESB is used

$$Tmx(CA) = O(n)$$

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.

7. 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 5 of appendix. Details are given below

- It consists of 2 common processes i.e. m=2
- It consists of 9 regional processes i.e. n = 9
- The common processes are 2.C and 10.C labelled in the graph as vertices.
- The regional processes are (1.L1, 1.L2), (3.L1, 3.L2)………….(9.L1, 9.L2) and (11.L1, 11.L2)
- CA-graph is generated for location L1 and L2, where common processes are shared by both the locations.
- There is an Exclusive-OR flow from process 2 to the process 3 or 5
- The complexity metrics for the graph
  (m = 2, n=9 and L =2)
- Total processes = m + nL = 2 + 9x2 = 20
  C_{mx} = (m+nL) x (m+nL -1)
  = 20x19 = 380,
- Using CA-ESB,
  Total processes = m + n = 2 + 9 = 11
  C_{mx} (CA) = (m+n) x (m+n -1)
  = 11x10
  = 110 (Using equation (1) and (2))
- The search metrics for the graph
  (m=2, n =9 and L =2)
  T_{mx} = 20 and T_{mx} (CA) = 11
  (Using equation (3) and (4))

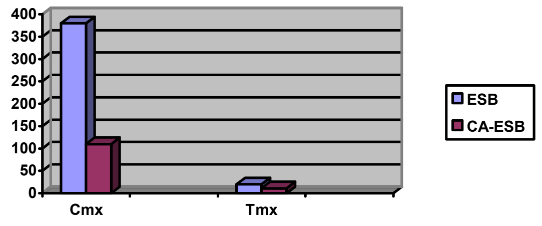


Figure 4: Performance metrics comparison for Policy Creation

Process table for "Policy Maintenance" process is given in table IV of appendix. CA-Graph is generated for process "Policy maintenance" as in Figure 6 of appendix.
Details are given below

- It consists of 2 common processes i.e. m=2
- It consists of 9 regional processes i.e. n = 9
- The common processes are 14.C and 21.C labelled in the graph as vertices.
- The regional processes are (12.L1, 12.L2), (13.L1, 13.L2), (15.L1, 15.L2)………….(20.L1, 20.L2) and (22.L1, 22.L2)
- CA-graph is generated for location L1 and L2 where common processes are shared by both the location.
- There is a parallel-flow from process 12 to the process 13 ,16 and 20
- The complexity metrics for the graph (m = 2 , n=9 and L =2)
  $C_{mx} = 380$
  C_{mx} (CA) = 110 (Using equation (1) and (2))
- The search metrics for the graph
  (m=2, n =9 and L =2)
  $T_{mx} = 20$
  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. Performance metrics chart for the process *Policy Maintenance* will be same as that of *Policy Creation* because the no. of regional processes (n) and no. of common processes (m) are same in both the cases.

8. CONCLUSION

In the present day scenario 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.

9. REFERENCES

[1] Jianwei Yin, Hanwei Chen,Shuiguang Deng,and Zhaohui Wu ,A Dependable ESB Framework for Service Integration, IEEE transaction, March/April 2009 (vol. 13 no. 2), pp. 26-34, http://www2.computer.org/portal/web/csdl/doi/10.1109/MIC.2009.26

[2] Gulnoza Ziyaeva, Eunmi Choi and Dugki Min , Content Based Intelligent Routing and Message Processing in Enterprise Service Bus , International Conference on Convergence and Hybrid Information Technology 2008(ICCIT08) , Nov 11-13 ,2008 ,Busan , Korea ,

[3] Gerald Weber, Technology-Independent Modeling of Service Interaction, IEEE Proceedings of the 2008 12th Enterprise Distributed Object Computing Conference Workshops (EDOC08) , Pages 35-42 , 15-19 September 2008, Munchen, Germany

[4] Deng Bo Ding Kun Zhang Xiaoyi , A High Performance Enterprise Service Bus Platform for Complex Event Processing , IEEE 2008 Seventh International Conference on Grid and Cooperative Computing(GCC 2008),October 24–26,2008. Shenzhen, China

[5] de Almeida , D.R. de Souza Baptista , C. da Silva, E.R. Campelo , C.E.C. de Figueiredo , H.F. Lacerda ,. A context-aware system based on service-oriented architecture, 20th International Conference on Advanced Information Networking and Applications(AINA 2006), April 18 - 20 , 2006 ,Vienna, Austria.

[6] Design a SOA solution using Reference Architecture www.ibm.com/developerworks/library/ar-archtemp/ -

[7] Ananya Kanjilal, Goutam Kanjilal, Swapan Bhattacharya, "Integration of Design in Distributed Development using D-Scenario Graph", Proceedings of IEEE International Conference on Global Software Engineering, page 141-150, Bangalore, India, Aug 17-20, 2008.

[8] S.H. Chang, H. J. La, J. S. Bae, W. Y. Jeon, S. D. Kim, "Design of a Dynamic Composition Handler for ESB-based Services" Proceedings of ICEBE 2007, page 287-294 ,Hong Kong , 24-26 Oct ,2007

[9] Vaculin, R.; Neruda, R.; Sycara, K., "Towards Extending Service Discovery with Automated Composition Capabilities" Proceedings of ECOWS 2008 , page 3-12 ,Dublin, December 12 ,2008

[10] Carenini, A.; Cerizza, D.; Comerio, M.; Della Valle, E.; De Paoli, F.; Maurino, A.; Palmonari, M.; Turati, A.; "GLUE2: A Web Service Discovery Engine with Non-Functional Properties" Proceedings of ECOWS 2008 , page 21-30 ,Dublin, December 12 ,2008

Appendix

TABLE III: PROCESS TABLE FOR THE *POLICY CREATION*

Sub Process No	Sub Process Name for Policy Creation	Pre-Process	Post Process	Type of Process (R/C)	Process Flow Type	Total Process
1.	Application Entry	-	2	R	N	1.L1 1.L2
2.	Underwriting	1	3 or 5	C	E	2.C
3.	Underwriting 1	2	4	R	N	3.L1, 3.L2
4.	Process Error	3,6	7	R	N	4.L1 , 4.L24
5.	Accept More Details	2	6	R	N	5.L1 5.L2
6.	Underwriting 2	5	4	R	N	6.L1 6.L2
7	Process Error	3,6	7	R	N	7.L1 7.L2
8	Premium Calculation	4	8	R	N	8.L1 8.L2
9	Premium Payment	7	9	R	N	9.L1 9.L2
10	Policy Issuance	8	10	C	N	10.C
11	Certificate Generation	9	-	R	N	11.L1 11.L2

TABLE IV: PROCESS TABLE FOR THE *POLICY MAINTENANCE*

Sub Process No	Sub Process Name for Policy Creation	Pre-Process	Post Process	Type of Process (R/C)	Process Flow Type	Total Process
12	Policy Maintenance	---------	13and16and 20	R	P	12.L1 , 12.L2
13	Policy Upgradation	12	14 or 16	R	E	13. L1 13.L2
14	Non-Financial update	13	16	R	N	14.L1 14.L2
15	Financial update	13	16	C	N	15.C
16	Policy Value Calculation	12	17	R	N	16. L1 16.L2
17	Add new premium	16	18	R	N	17. L1 17.L2
18	Calculate interest	17	19	R	N	18.L1 18.L2
19	Deductions	18	----	R	N	19.L1 19.L2
20	Loan Processing	12	21	R	N	20. L1 20.L2
21	Accept Loan request	20	22	C	N	21.C
22	Verify/recalculate Loan amount	21	-----	R	N	22. L1 22.L2

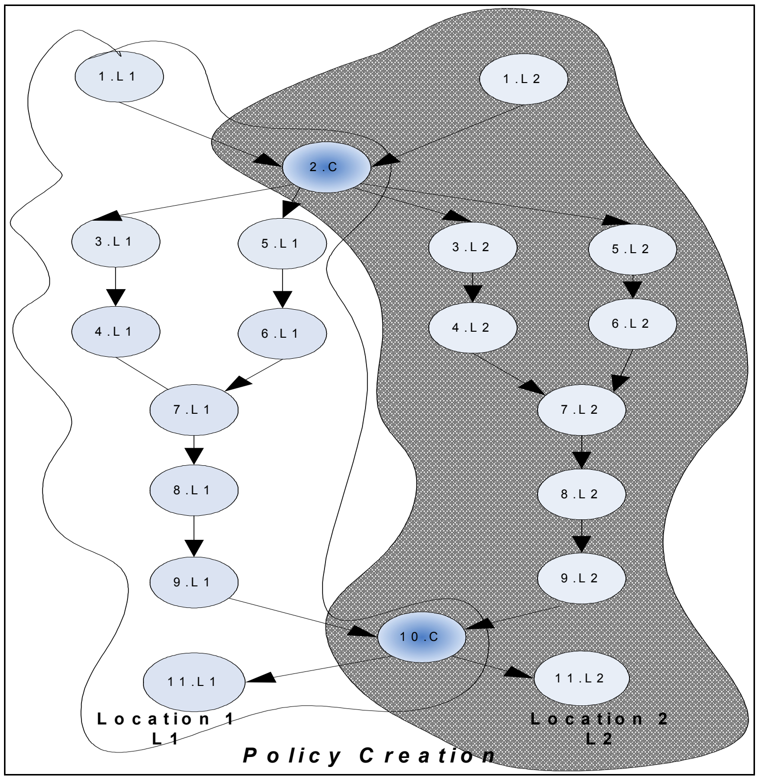


Figure 5: The CA-Graph for "Policy creation"

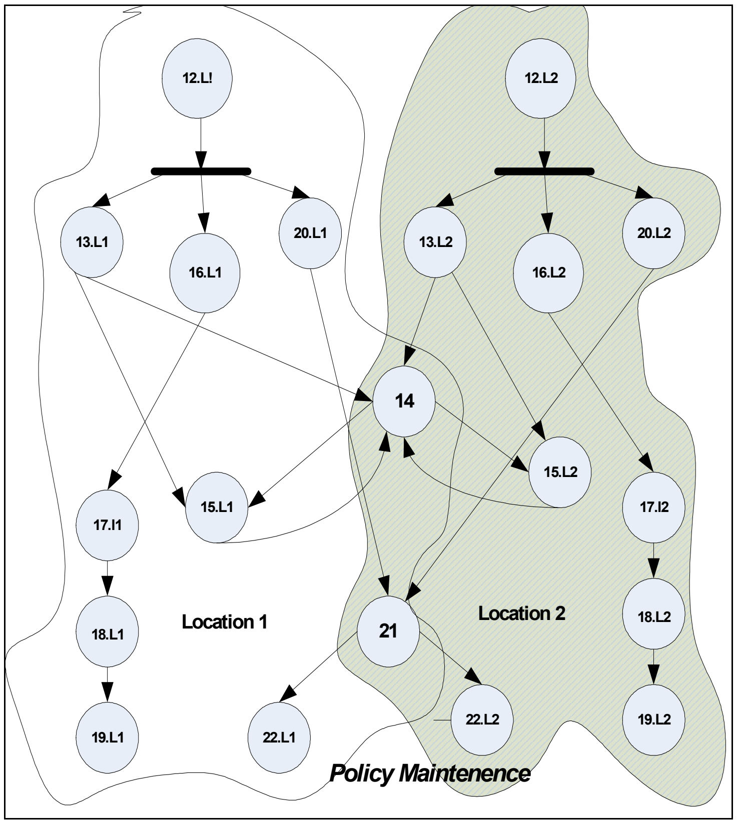


Figure 6: The CA-Graph for "Policy maintenance"