

SOA-A GENERIC APPROACH FOR INTEGRATING LOOSELY COUPLED SYSTEMS WITH OTHER SYSTEMS

D.R. Ingle¹ and Dr. B.B. Meshram²

¹Department of Computer Engineering, Bharati Vidyapeeth College of Engineering, Navi
Mumbai, India

dringleus@yahoo.com

²Department of computer Technology, VJTI, Mumbai, India

bbmeshram@vjti.org.in

ABSTRACT

Various organizations generate data in various domains which is queried and analyzed by users. This limits the possibility of database integration with other systems. We describes a generalized approach comprising a loosely couples system and integrate with other system. It deals with the setting up the environment for implementing the system. User Interface screen shows how the user will interact with the system and the data entry forms required to gather data for the system.

KEYWORDS

SOA, CORBA, DCOM, BPEL, XML, SOAP.

1. INTRODUCTION

The fundamental concept in SOA is a service [1]. We define a service as a discrete unit of business functionality that is made available through a service contract [2]. The service contract specifies all interactions between the service consumer and service provider [3]. This includes: Service interface, Interface documents, Service policies, Quality of service (QoS) and Performance [4]. One of the main differences between a service and other software constructs is that a service is explicitly managed. The QoS and performance are managed through a service level agreement (SLA). In addition, the entire service life cycle is managed—from design, to deployment, to enhancements, to maintenance [5]. There are two main aspects to the service itself [6]. The Web Services architecture describes the principles behind the next generation of e-business architectures, presenting a logical evolution from object-oriented systems to systems of services [7]. Web Services systems promote significant decoupling and dynamic binding of components [8]. All components in a system are services, in that they encapsulate behavior and publish a messaging API to other collaborating components on the network [9]. Services are marshaled by applications using service discovery for dynamic binding of collaborations [10]. With the Web Services approach, application design becomes the act of describing the capabilities of network services to perform a function and describing the orchestration of these collaborators [11]. At runtime, application execution is a matter of translating the collaborator requirements into input for a discovery mechanism, locating a collaborator capable of providing the right service and orchestrating message sends to collaborators to invoke their services [12].

What is important here is that consumers of the service should. Fusion Middleware face the challenges like communicating entities may take on different roles such as [13] client server or peer-to-peer. Based on the interaction mode like synchronous invocations, asynchronous message passing, coordination through shared objects performance affects [14]. Fusion middleware also face problem of the scalability, availability, reliability, concurrency, security widespread computing [15].

1.1 Background and Motivation

SOA is a long-term transition that will reshape the business, and IT -- all parts of IT -- need to be full partners with the business side in this journey [16]. Service-oriented architecture differs from most other integration projects in that it's an effort that reaches outside the IT silo, requiring participation from all ends of the enterprise [17].

1.1.1 Objective

A. Logical view:

1. Business level operation for this model include checking the status of the student in the merit list based on the certain condition
2. It is also based on the suggestion of the educational loan from the number of the banks
3. Orchestration of the process is formed to get the final result as the list of the admitted student taken the advantage of the educational loan.

B. Message Orientation

1. From the service provider to the consumer how the values of the variables are moving across is considered using message orientation.
2. Depending on the synchronous and the asynchronous type of messaging the variables are defined and the values are used.
3. Choreography of the business rules are sequenced to check the working.

C. Description orientation

1. A service is described by machine-process able metadata.
2. Merit list generation is specified with the single service which further can be fragmented using various conditions.

1.3 Problem Statement

Student admission is the process for admitting the student in the organization. This process includes list of student's personnel information with the academic information. List of the selected students are displayed according to their educational information. Selected students can take admission with the payment of the fees. Some time due to financial problem and the unaware of the educational loan student may lose admission in the good organization. For the above process different application may be constructed. Specifically application form, merit list module, admission form, payment modules. If status of the student's admission is to check whether admitted or not, two different application modules need to be checked. Retention of both modules may be problematic if the both the packages are different. Writing the combined interface is the costly and time consuming phase if the existing modules are working fine. Integration can be a problematic it affects a lot on the business logic of one on another. Basic problem is to make integration of the modules without changing the business logic of the existing

one to form the legacy module. In the institute Student's personnel information is an application storing the student's personnel information. Similarly Student's academic information is another application for storing the student's academic performance. Services provided by these two are different. Similarly banking application is for the educational loan having various services like interest rate, documents required, repayment etc. Accessing all these application under one roof is difficult. So there should be a combine system which will provide all the information at the same time. Reconstructing the whole system will be time consuming. Also it will be standalone application which will be platform dependent.

1.3.1 Detailed Statement of the Problem

Problem of combining the interfaces from the different application with business logic can be done with Service Oriented Architecture. SOA, based on Web Services, promises to simplify integration by providing universal connectivity to existing systems and data. SOA is an architectural approach to building systems based on the architecture and a strategic and business vision, with engineering discipline and governance, and a supporting organizational structure. Using SOA strategic reuse of modules across multiple departments' applications can be done. SOA provide more agile support to business processes, with the change management impacts more efficiently and effectively.

Modules used in the Proposed System

a. Student application form module

1. Online application form for student resulted as application number.

b. Merit list generation module

1. generating the merit list based on the condition like gate

2. generating the merit list based on the condition like gate with category

3. sponsor student s merit list generation based on the interview and written test performance

c. Admission module by Administrator

1. Admitting student listed in the merit list

2. Payment option selected

d. Banking educational loan module

1. List of banks educational loan schemes

e. SOA integrated module

1. Check the status of the student based on the application form no.

2. Check the status of the student present in the merit list as per the condition for gate/sponsor

3. Check the financial status of the student

4. If it is not satisfactory then suggest for the educational loan

5. Educational loan from certain bank is the output of the number of comparison done among various banking educational loan scheme.

6. If the student sanctioned the loan this information is saved into the admission form as a separate record

7. This database is utilized as the future reference for the next student for better selection of the bank.

The rest of the paper is organized as follows. Section 2 deals with Case study of M.Tech admission, system analysis, design and proposed system deals with Section 3. Section 4 deals with Algorithms for admission using SOA and section 5 gives the conclusion.

2. CASE STUDY M.TECH ADMISSION MODEL USING SOA

2.1 Introduction

The data integration model to overcome the data integration issues related to the SOA. Each layer in this model performed the task related to data integration. Uniform data accessor is the interface to various heterogeneous data sources. XML view and SOA mapper encapsulates the uniform data accessor to the standard service of Web Service and then starts data integration through it. Data integration using processing engine uses metadata of XML Schema which is the basis of XML view. XML Schema is like the table in relational database while XML view is just like the view corresponding to the table in relational database. The XML data integration view represents the constitution of practical data as where the data is from, which process is needed for table data, how to combine the data from different tables and so on.

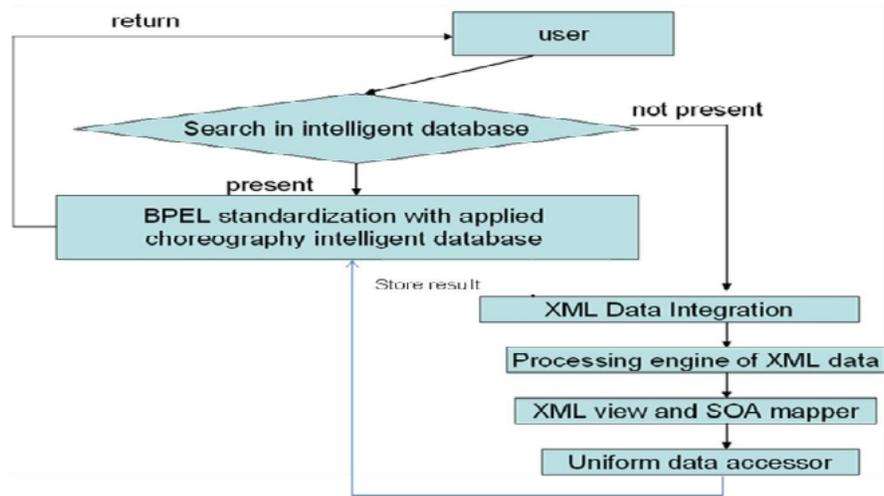


Figure 1 Proposed System For Dynamic Data Integration Model Using Intelligent Database

2.2 Intelligent Database

With this model data sources from the various application can be integrated to respond to the end user. For this response model uses XML schema to represent the result. Problem arises if the similar type of the request raised from the user then again the same integration technique is repeated. This causes the efficiency problem. Solution for the above problem can be solved using the intelligent database after the generation of the result. This result is in the form of XML format. So the task is to convert the XML representation in the database format.

2.3 XML into database storage

XML is designed to transport and store data. XML is important to know, and very easy to learn.

XML is used as storage or interchange format for data that is structured, appears in a regular order and is most likely to be machine processed instead of read by a human. Problem can be solved by mapping from xml message inputs to stored procedure input parameters using business objects

XML stands for eXtensible Markup Language.

This include the advantages as

- Contract between xml and table
- Guarantees the matching of elements and columns

This technology will reuse the existing application of student's personnel information and student's academic information with banking information application to form integrated new application. Student applying for the educational loan with all the information from personnel and academic information with selected banking information will be saved in the database. Security of the transaction is build using choreography. Storing and retrieving the transactions intelligently into db.

3 . SYSTEM ANALYSIS

This section explains how the system is analyzed to carry out the work for proposed system. First we will perform requirement analysis is done for the system. Secondly the UML Diagrams and workflow Diagrams, along with the ER Diagram and various Data Flow Diagrams for the system are explained.

3.1 Introduction

Analyzing requirements involves studying the current system and finding out how it works. Requirement is gathered by studying basic business process, collecting input data and data which is produced. ER diagram is basically used to identify the data for the system to be developed. A data flow diagram is graphic description of a system or a portion of the system.

3.2 Static Modeling of the system

This system contains the database of student applied along with banks. This system also stores information of the student who has taken the admission with the educational loan facility. To analyze the system we have used UML strategy.

3.2.1 Use Case Diagram

The systems use case diagram is shown in figure 3. The system use case diagram contains student and admin as the main actor.

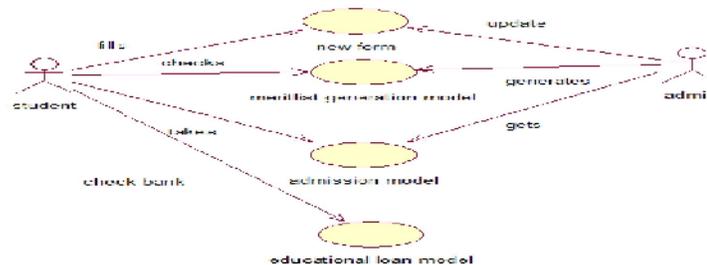


Figure 3. M.Tech. Admission System Using SOA Use Case Diagram.

Following figure 5 shows the merit list generation operation for the admin. Using this he can generate different merit list based on the type or the category.

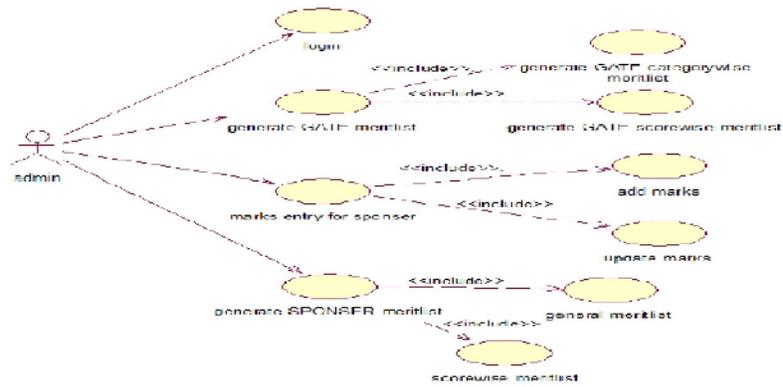


Figure 5. Merit list generation use case diagram

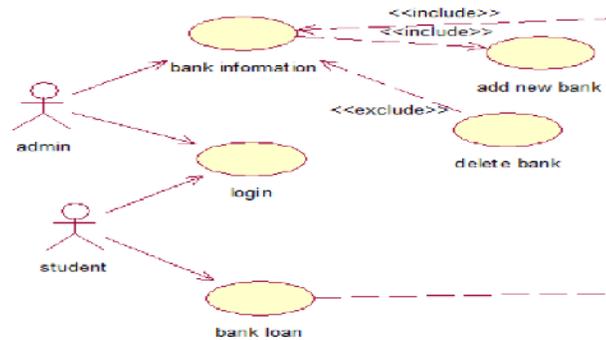


Figure 6. Bank information use case diagram

Figure 6 explains the graphical representation of the admission model which is handled with admin. He checks for the form no of the student and then performs the admission procedure.

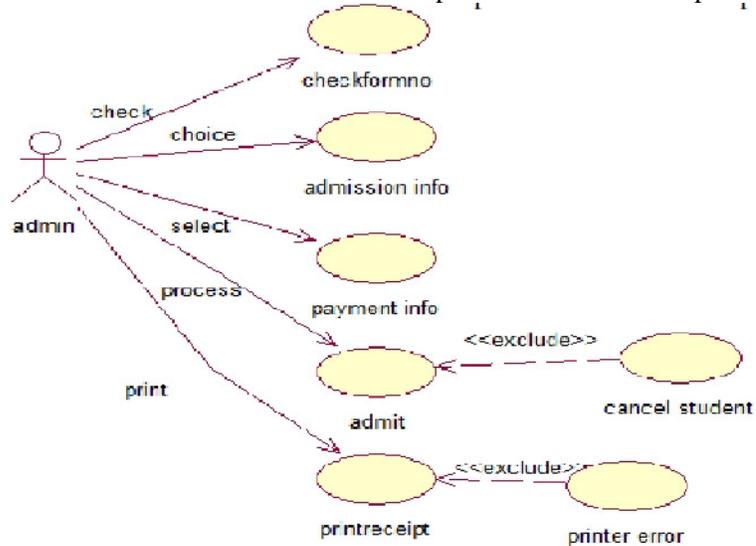


Figure 7. Admission procedure use case diagram

Figure 7 shows the bank information model. Figure 8 shows the educational loan model of the system using this student can see the bank information for educational loan

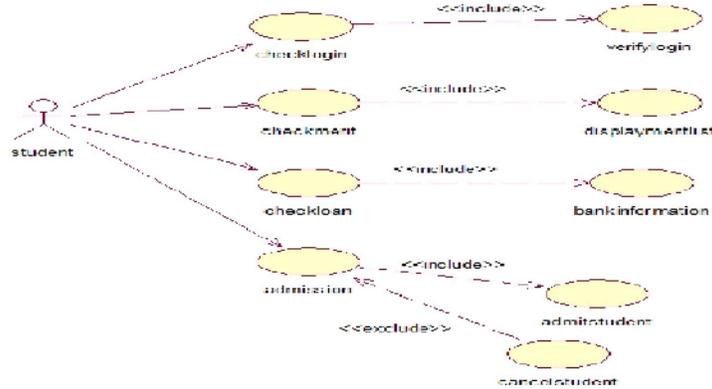


Figure 8. Educational loan use case diagram

3.2.2. Class Diagram

The class diagram is a static diagram. The purpose of the class diagram can be summarized as:

- Forward and reverse engineering
- Describe responsibilities of a system.
- Analysis and design of the static view of an application.
- Base for component and deployment diagrams.

Figure 9 below shows class diagram for the online application system. It includes following classes:

1. Validatelogin: this class uses the students information to check the validity of the student. this is used if student wants to update his information.
2. Editrecords: this class allows student's information to be updated.
3. Serverenquiryform: this class works as the servlet to allow students information to save in the database.

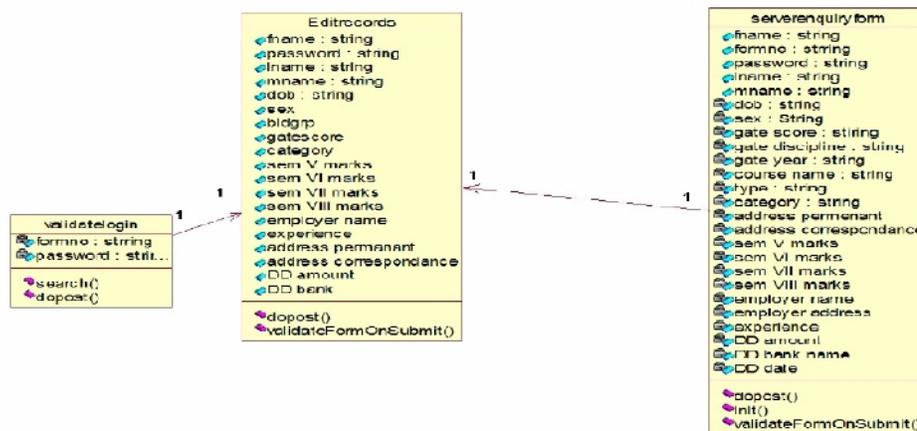


Figure 9. Online application class diagram

Figure 10 below shows class diagram for the meritlist generation system. It includes:

1. Meritgeneration: this class uses the gate score information to create the merit list for gate student.
2. Meritgeneratelist: this class is used to access the database information from the merit student information to display the information.
3. Sponser: saves the info of the student in the sponser's examination performance into the database.

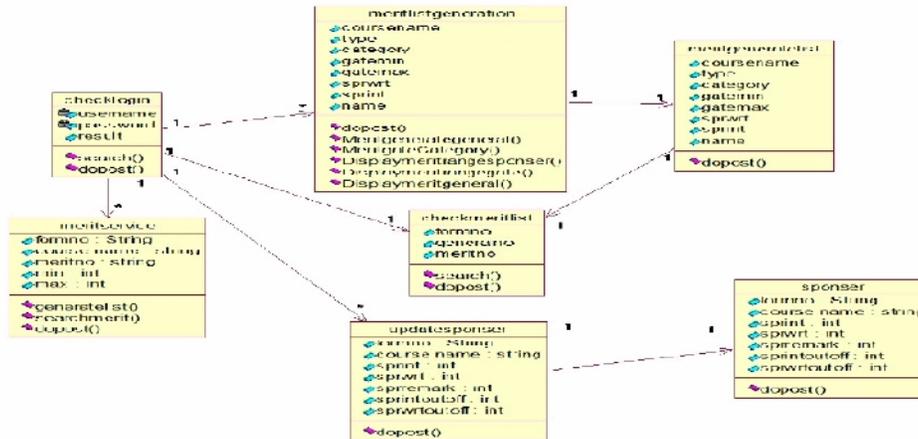


Figure 10. Merit list generation class diagram

Figure 11 below shows class diagram for the bank information system. It includes following classes:

1. Login : this class use for the admin as well as for student to check the login validity.
2. Bankinfo: this class use to add the banking information into the database.
3. Education bank loan: this class is use to save the selected bank information for the particular student

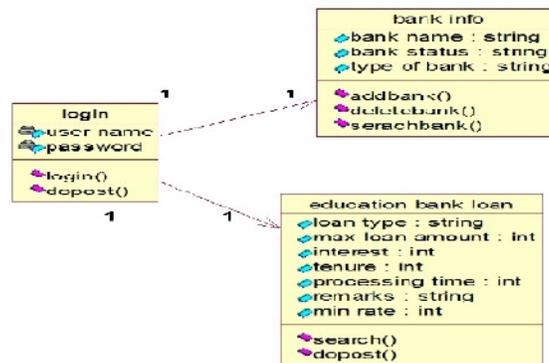


Figure 11 Bank information class diagram

Figure 12 below shows class diagram for the educational loan for the integrated system. It includes following classes:

1. Validatelogin: this class used to check validity of the student.

2. Checkmeritlist: this class is used to check presence of the student in the merit list. This shows the general and category wise merit list number of the student.

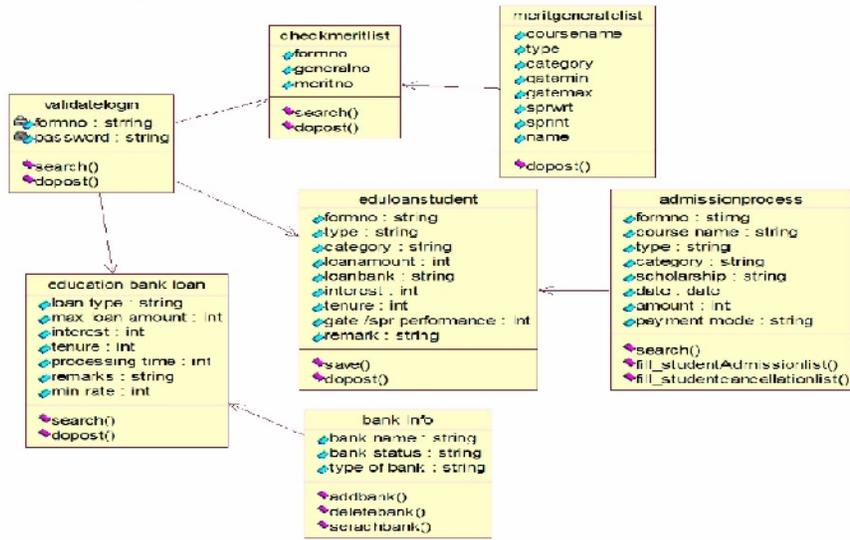


Figure 12. Educational loan class diagrams

3.2.3. Dynamic UML Diagram

UML is a general purpose modelling language. It was initially started to capture the behavior of complex software and non software system and now it has become an OMG standard. UML provides elements and components to support the requirement of complex systems. UML follows the object oriented concepts and methodology. So object oriented systems are generally modeled using the pictorial language. UML diagrams are drawn from different perspectives like design, implementation, deployment etc.

- UML Interaction diagram: describes the dynamic interaction of different elements of the system.
- Activity diagram: describes the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

3.2.3.1. State chart Diagram

State Chart Diagram describes different states of a component in a system. State chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML). In this context, a state defines a stage in the evolution or behavior of an object, which is a specific entity in a program or the unit of code representing that entity. State diagrams are useful in all forms of object-oriented programming (OOP). The concept is more than a decade old but has been refined as OOP modeling paradigms have evolved. It is used to model dynamic aspect of a system.

Fig 14 shows State chart diagram for online application model. model goes through different states as connected, logging, adding new, updating information.

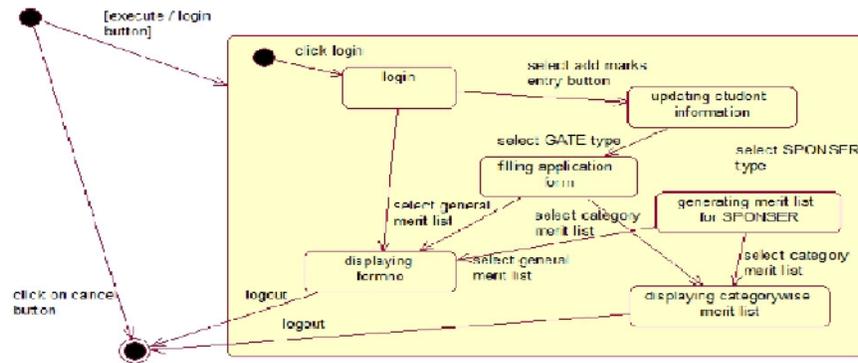


Figure 14. Online application state chart diagram

Fig 15 shows Statechart diagram for merit list generation application model. model goes through different states as connected, logging, creating and displaying merit list

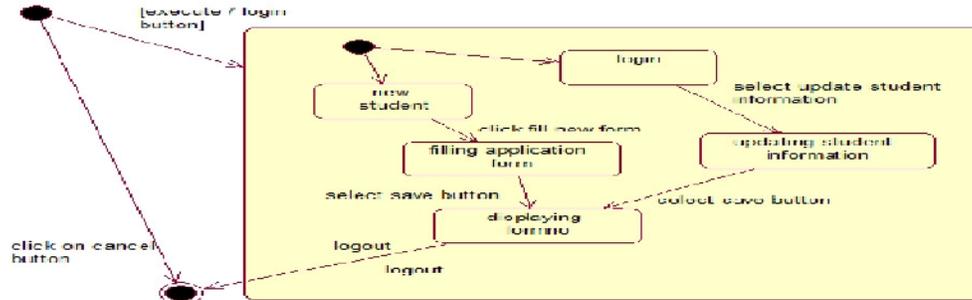


Figure 15. Merit list generation state chart diagram

Fig 16 shows Statechart diagram for educational loan for the integrated system model. model goes through different states as connected, logging, selecting the bank, saving the records in the database.

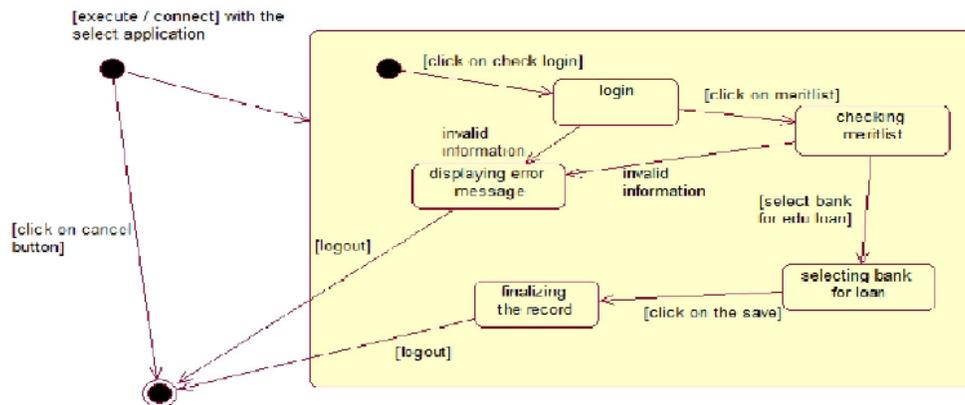


Figure 16 Educational Loan Using Intergrated Model State Chart Diagram

3.2.3.2. Interaction diagrams

Interaction diagrams are models that describe how a group of objects collaborate in some behavior - typically a single use-case. The diagrams show a number of example objects and the messages that are passed between these objects within the use-case. So the purposes of interaction diagram can be describes as:

- To capture dynamic behavior of a system.
- To describe structural organization of the objects.
- To describe interaction among objects.

Figure 17 shows sequence diagram for merit list generation model. This shows the way admin interact with the merit list generation model.

Figure 18. shows sequence diagram for online application model.

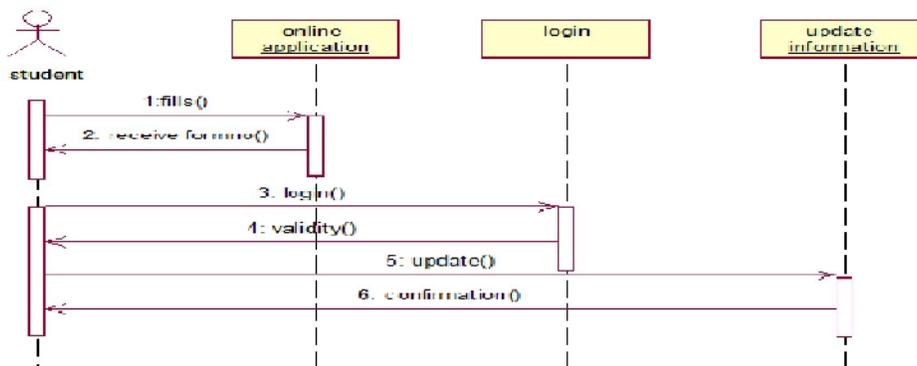


Figure 17. online application model sequence diagram

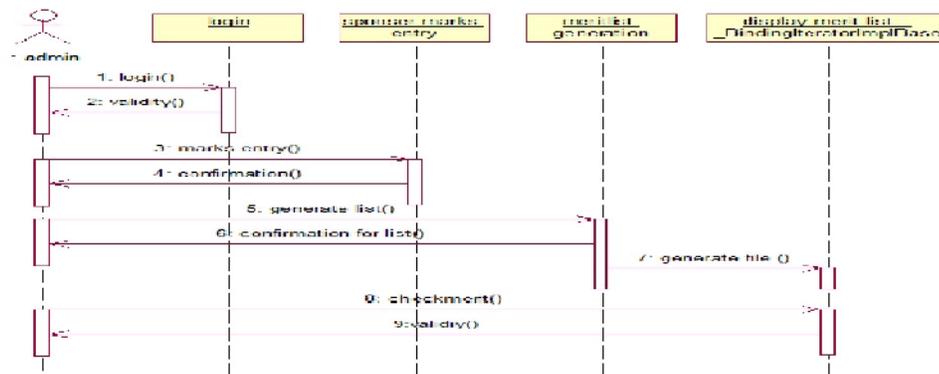


Figure 18. merit list generation sequence diagram

3.2.3.3 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions^[1] with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organisational processes (i.e. workflows). Activity diagrams show the overall flow of control

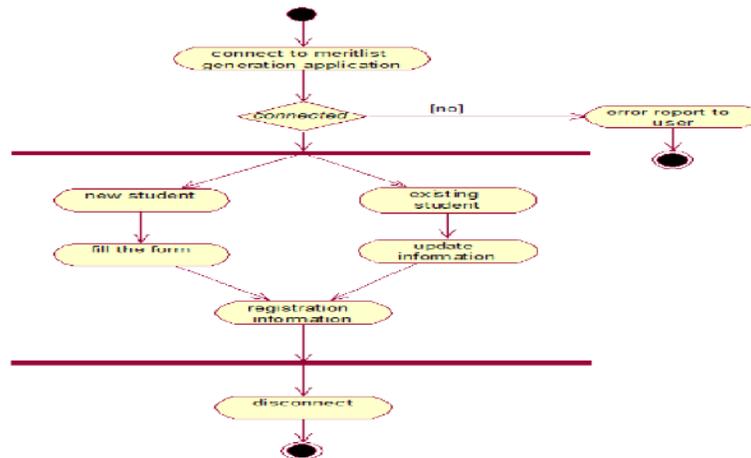


Figure 19. online application activity diagrams

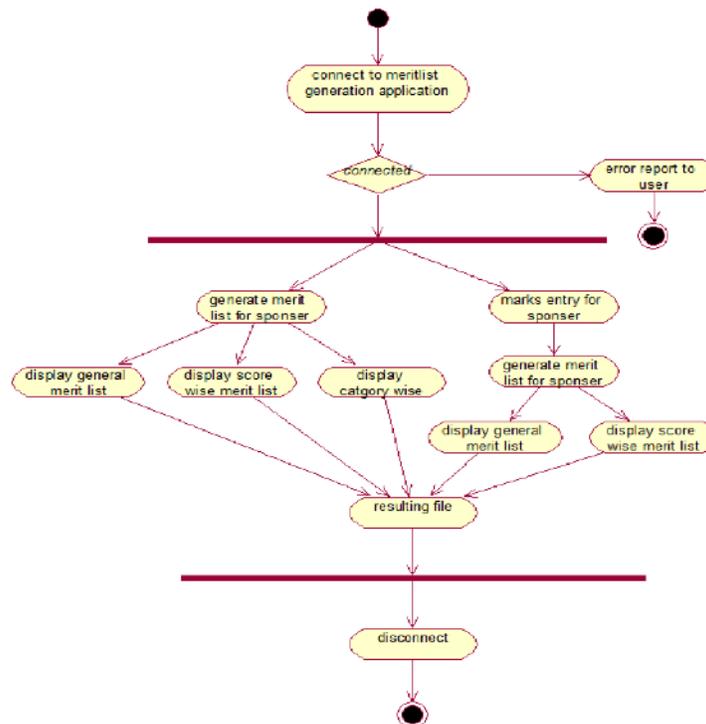


Figure 20. Merit list generation activity model

3.2.3.4 Workflow of System

A **workflow** consists of a sequence of connected steps where each step follows without delay or gap and ends just before the subsequent step may begin. It is a depiction of a sequence of operations, declared as work of a person or group, an organization of staff, or one or more simple or complex mechanisms. Workflow may be seen as any abstraction of real work. For control purposes, workflow may be a view on real work under a chosen aspect, thus serving as a virtual representation of actual work. The flow being described may refer to a document or product that is being transferred from one step to another.

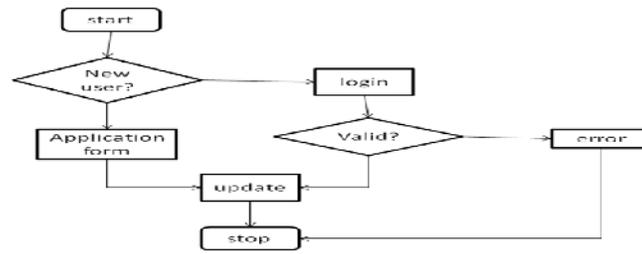


Figure 21. Workflow of the student online application model

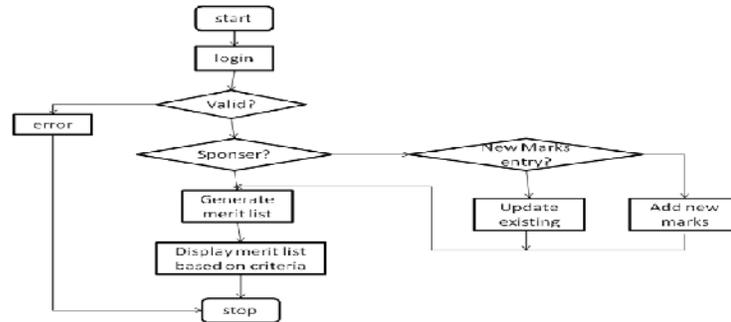


Figure 22. Workflow diagram of the merit list generation model

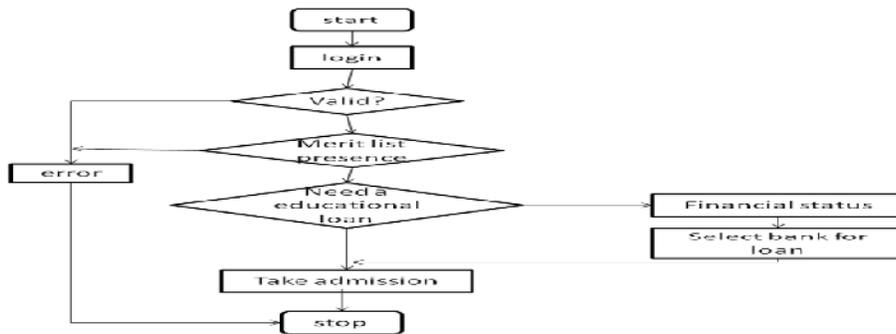


Figure 23. Workflow diagram of the educational loan for the student integrated model

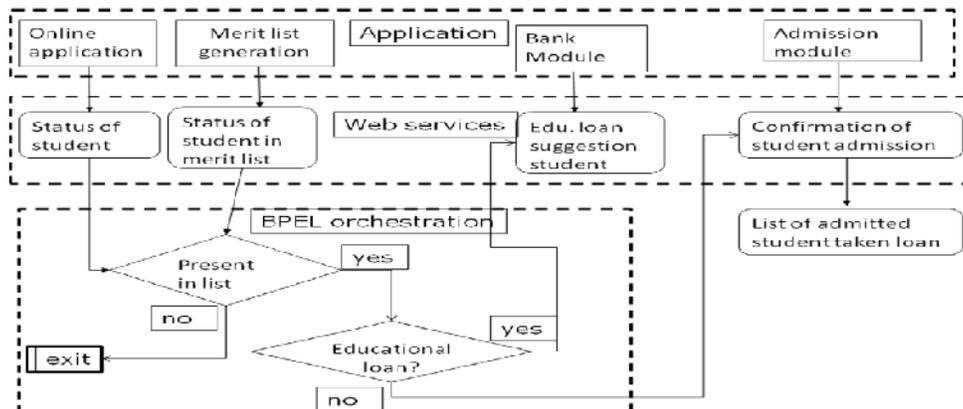


Figure 24 Workflow of Educational loan for SOA module

3.3. Design M.Tech admission model using SOA System

3.3.1 Database System Analysis and Design

Database system contains the data to load in the database. For analyzing the system we have used structured analysis. As a part of this process, we have generated ER diagram dataflow diagram and component diagram for different modules of the system.

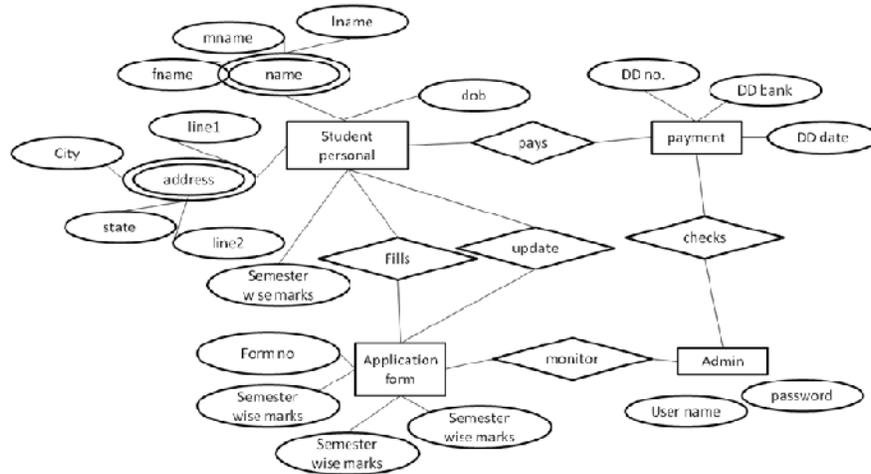


Figure 25 E-R diagram of the online application model

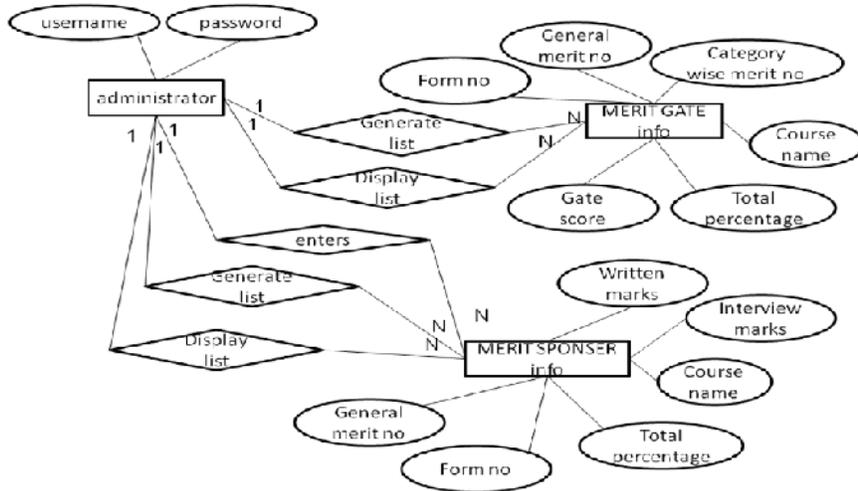


Figure 26 E-R diagram of the educational loan for the student using integrated model

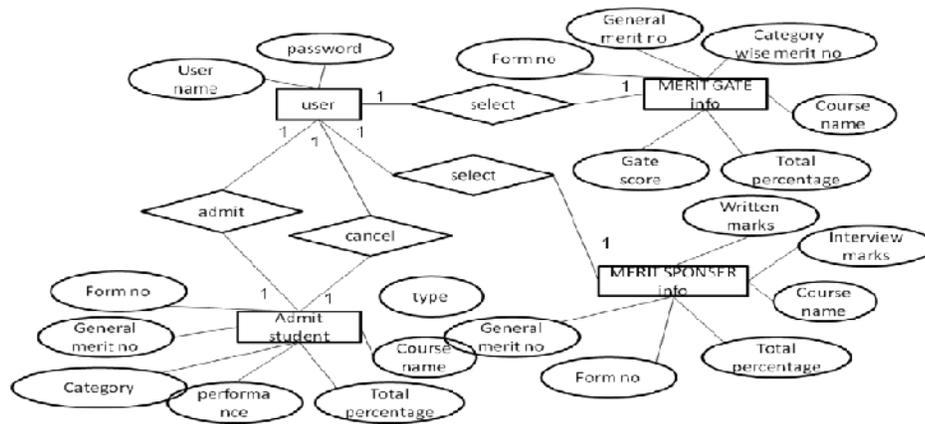


Figure 27 E-R diagram of the admission process model

3.3.2 Data Flow Diagram

Data flow diagram shows the transformation of data from input to output, through processes. We have analyzed the system up to the second level of data flow diagram.

3.3.2.1 Context Level DFD

The student will receive the data or requested resources at the client side in terms of response from the server. Figure 28 shows the context level DFD of the system.



Figure 28 Context diagram of the educational loan for student using integrated model

3.3.2.2 Level-1 DFD

Figure 29 shows the level-1 DFD for the system. The proposed system allows student to check the presence in the merit list. Also check for the educational loan form the existing bank information. Admin will monitor the application model and admit the student those are present in the merit list.

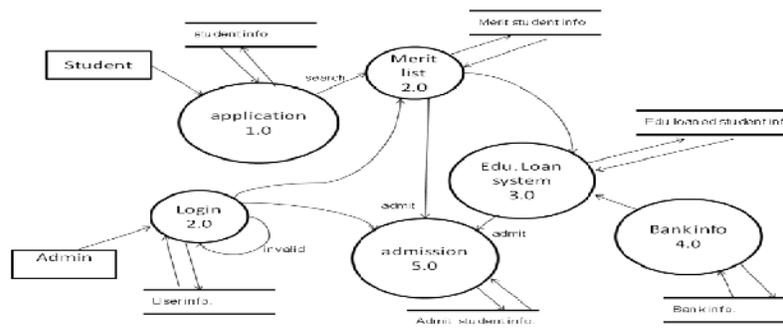


Figure 29: Level DFD diagram of the educational loan for student using integrated model.

3.4 Component Diagram

The Component Diagram helps to model the physical aspect of an Object-Oriented software system. It illustrates the architectures of the software components and the dependencies between them. Those software components including run-time components, executable components also the source code components.

So the purpose of the component diagram can be summarized as:

- Visualize the components of a system.
- Describe the organization and relationships of the components.

Fig shows component diagram.

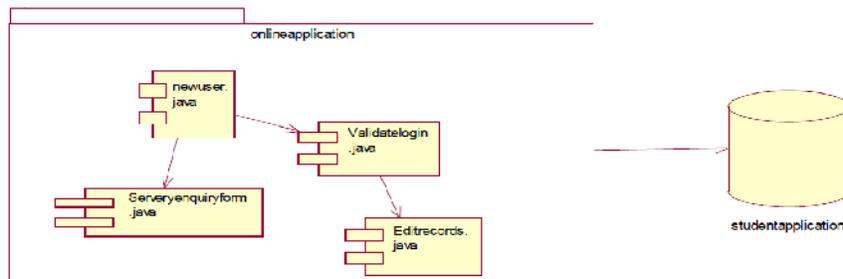


Figure 30 Online application component diagrams

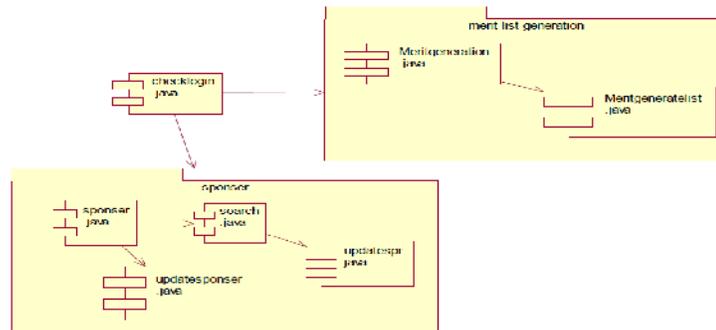


Figure 31 Merit list generation component diagram

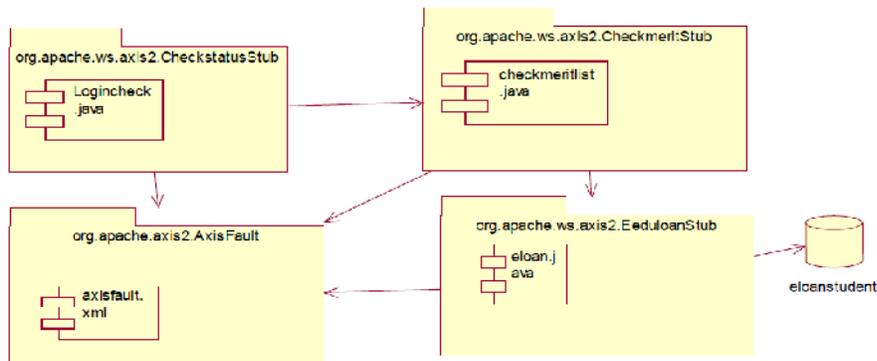


Figure 32 Educational loan for student using integrates system's component diagram

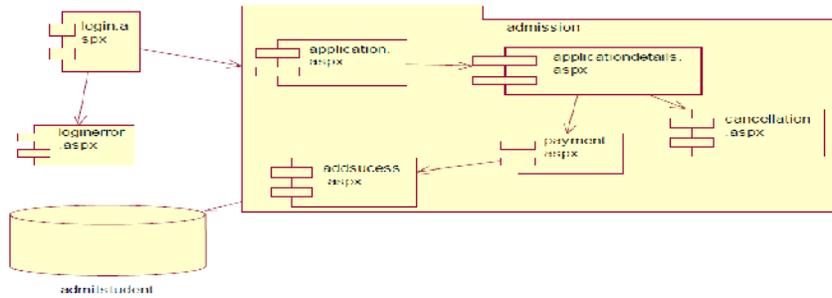


Figure 33 Admission procedure model component diagram

3.5 Package Diagram

Package diagrams organize the elements of a system into related groups to minimize dependencies among them. Fig shows package diagram.

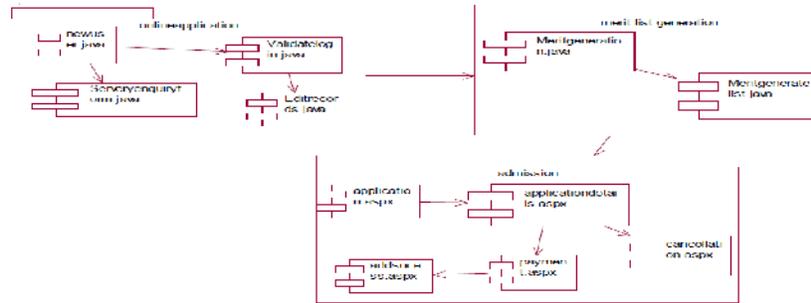


Figure 34 package diagram of the system

4. ALGORITHMS

4.1 Pseudocode for M.Tech admission model using SOA

4.1.1 Publishing Web services to a UDDI registry

1. Creating services: create the services / reusable methods at the producer side. Service description of the method should be made available at the producer side.

4.1.2 Pseudo code for producer

Input: publishURL, UDDIuserid, UDDIpassword

Output: provider in the list

1. Class name for producer
2. {
3. Declare UDDIProxy proxy;
4. Assign publishURL=path; uddiuserid and uddipassword; trustStoreFilename = path; trustStorePassword = value;
5. classname {
6. call setEnvironment();
7. call setupProxy();
8. }
9. }

Pseudo code for setEnvironment()

Input : username, password

Output: updated provider list

1. {
2. Set appserver username and password
3. Add SSL as a supported protocol.
4. Add provider to the provider list
5. }

Pseudo code for creating UDDIProxy

Creating the UDDIProxy

Input : inquiryURL;

Output: Proxy UDDI setup

1. setupProxy()
2. {
3. Create object proxy for UDDIProxy();
4. proxy.setInquiryURL as inquiryURL;
5. if MalformedURLException occurred
6. display error Can't create proxy - exit ...
7. }
8. }

2. Setting up the registry and Publishing your services: defined services at the register with it type of service i.e. business.

Pseudo code for producer

Input: modelkey

Output: published business service businesskey

1. Producer
2. {
3. Create Classname classobject;
4. Classobject.publishBusinessEntity();
5. Classobject.publishBusinessService(businesskey, modelkey);
6. ...//repeat for the all the class to be register at the registry;
7. }

4.1.3 Discovering Web Services from a UDDI registry

1. Running Web services: for accessing the web services that are published at the register side first develop the client and then access the web service The client: first assign the location of the register to the client using following locator.

- Locator -- sets up the connection to the UDDI registry for queries.
- ModelLocator -- locates the technical model based on a model name.
- ServiceLocator -- locates all the services implementing a given technical model and finds the URLs of the actual services.
- BusinessLocator -- retrieves and displays information about a business. Connecting to the registry -- the Locator class Input: inquiryurl path Output: connection to the registry\

1. Class locator

2. {
3. Declare Uddi proxy object representing the UDDI registry;
4. Assign inquiryurl=path;
5. Locator();
6. Call setupproxy();
7. }

Locating the technical model -- the ModelLocator class Call the locator class

The find_tModel method

find_tModel() method to find tModels by name, categories, identifiers, or any combination of them. The last parameter, (int maxRows)

TModelList find_tModel()

input: tModelName, categorybag, identifierbag, findqualifier, maxrows

output: tModelKey

```
1. {
2. Assign tModelInfoVector as Null;
3. Define findQualifier, findQualifier findqualobject;
4. Settext for the object using sortbydatedesc;
5. Create findQualifiers findqualifierobject;
6. find model (Name, findQualifiers);
7. assign values to vector = tModelList. getTModelInfos().getTModelInfoVector();
8. catch(TransportException)
9. catch(UDDIException)
10. if(Vector.size())>0)
11. {
12. Assign tModelInfo = 0th vector; tModelKey = tModelInfo.getTModelKey();
13. }Else
14. {
15. Display no tmodel
16. name;
17. }
18. return tModelKey;
19. }
```

Locating the services -- the ServiceLocator class

Locating BusinessService using UDDI by using the find_service() method of the UDDIProxy class. find_service() method of the UDDIProxy class to find services in the registry based on the following:

- The UUID of a BusinessEntity
- The name of the service
- The category information of a service
- The tModel information of this service
- Any combination of these parameters.

ServiceList find_service

Input: businessKey, Vector names, CategoryBag, TModelBag , FindQualifiers , maxRows

Output: serviceKeys

findServices()

```
1. {
2. Create mpdellocator modelobject;
3. Assign tModelKey = findTModel(tModelName);
4. Create tmodelbag modelbagobject;
5. Add tmodelkey to tModelBag;
6. create ServiceList myServiceList = proxy.findservice (null, null, null, tModelBag,
null, 0);
7. serviceInfoVector = myServiceList.getServiceInfos().getServiceInfoVector();
8. }catch(TransportException )
9. catch(UDDIException)
10. if(serviceInfoVector.size() == 0)
display no service found;
11. Create Map serviceKeysobject;
12. repeat till serviceInfoVector.size reached
13. {
14. serviceInfo = serviceInfoVector.get(i); serviceKey = serviceInfo.getServiceKey();
15. businessKey = serviceInfo.getBusinessKey(); serviceKeys.put(serviceKey, businessKey);
```

```

16. }
17. return serviceKeys;
18. } 19. }
    
```

4.1.4 The find_business()

BusinessEntity can be located using UDDI by using the find_business() method of the UDDIProxy class.

```

find_business(
input: Vector names, DiscoveryURLs, IdentifierBag, CategoryBag, TModelBag,
FindQualifier, int maxRows
output: BusinessList
getserviceURL()
1. {
2. Create urlmap object;
3. Assign servicedetails as proxy.get_servicedetail(service keys);
4. Assign businessServices = serviceDetail.getBusinessServiceVector();
5. }catch(TransportException t)
6. catch(UDDIException)
7. Check for (businessServices.size() == 0){
8. Display failed to find;
9. }
10. repeat till businessServices.size reached
11. {
12. assign serviceKey = businessService.getServiceKey(); bindingVector= 13.
businessService.getBindingTemplates(). getBindingTemplateVector();
14. bindingTemplate = null;
15. Get bindingTemplate = bindingVector.elementAt(); Get AccessPoint accessPoint
16. = bindingTemplate.getAccessPoint();
17. if(accessPoint.getURLType().equals("http"))
18. {
a. urlMap.put(accessPoint.getText(),serviceKey);
19. } 20. } 21. }
22. return urlMap;
23. }
Function bindservice()
Input : business key
Output: business vector
1. class BusinessLocator extends showBusinessDetail(businessKey){
2. usinessDetail = proxy.get_businessDetail(businessKey);
3. }catch(TransportException ).
4. }catch(UDDIException)
5. Get businessEntityVector =businessDetail.getBusinessEntityVector();
6. Check if(businessEntityVector.size() == 0)
    
```

5. CONCLUSION

Proposed system is constructed to integrate number of the application with the help of web services. Web services produced by the service producer are stored at the central registry to be available to use for the service consumer. Business process Execution language is used to composite the web services. Composition of the web service will give the result as per the business logic implemented for the requirement.

REFERENCES

- [1] Michael Rosen, Boris Lublinsky, Kevin Smith, "Applied SOA: Service-Oriented Architecture and Design Strategies", Wiley Publishing, Inc. 10475 Crosspoint Boulevard Indianapolis, IN 46256 2008 by Wiley Publishing, Inc., Indianapolis, Indiana Published simultaneously in Canada ISBN: 978-0-470- 22365-9.
- [2] A Dynamic Data Integration Model Based on SOA Jun Wang 2009 ISECS International Colloquium on Computing, Communication, Control, and Management
- [3] DCIO, DOD OASD NII, Net-Centric Checklist , version 2.1.2 March 31th, 2004.
- [4] W. Kernochan, Mainframe Security Changes as Web Services Arrive .
- [5] M. P. Singh and M. N. Huhns, Service-Oriented Computing, John Wiley & Sons, 2005 .
- [6] Business Process Intelligence System: Architecture and Data Models Liping An1, Jianyuan Yan1, Lingyun Tong2
- [7] Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Commun.* 7, 39–59 (1994)
- [8] Lingjuan Li, Wenyu Tang. Research of the Applications of CBR in Business. *Journal of Nanjing University of Posts and Telecommunications*, No.10, 2006 pp:17-21 Based on SOA and CBR
- [9] Yan Qin, Zhang Guoliang, Wang Keyi Knowledge-enabled Human Resource Development Based on e-HR. *Computer Applications and Software*, No.11, 2008
- [10] I. Matsumura, T. Ishida, Y. Murakami, and Y. Fujishiro. Situated web service: Context-aware approach to high speed web service communication. In *IEEE International Conference on Web Services (ICWS-06)*, pages 673–680, 2006.
- [11] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Golan, Satish Thatte, "Business Process Execution Language for Web Services Version 1.1", 5 May 2003
- [12] Service oriented architectures: approaches, technologies and research issues Mike P. Papazoglou • Willem-Jan van den Heuvel
- [13] Service-Oriented Architecture Concepts, Technology, and Design Thomas Erl
- [14] ESB Infrastructure's Autonomous Mechanism of SOA Rui Wen1, 2, Yaping Ma1 and Xiaoqing Chen1 2009 International Symposium on Intelligent Ubiquitous Computing and Education
- [15] Luis Garcías "Building an Enterprise Service Bus for Real-Time SOA: A Messaging Middleware Stack" -Ericse 2009 33rd Annual IEEE International Computer Software and Applications Conference
- [16] a b c Bell, Michael (2008). "Introduction to Service-Oriented Modeling". *Service-Oriented Modeling: Service Analysis, Design, and Architecture*. Wiley & Sons. pp. 3. ISBN 978-0-470-14111-3.
- [17] Bell, Michael (2010). *SOA Modeling Patterns for Service-Oriented Discovery and Analysis*. Wiley & Sons. pp. 390. ISBN 978-0470481974.

Authors Short Biography

Mr. D.R. Ingle (ISTE LM'2004) is Professor of Computer Engineering Department at Bharati Vidyapeeth College of Engineering, Navi Mumbai, Maharashtra state, India received bachelor degree, and Master degree in computer engineering. He has participated in more than 10 refresher courses to meet the needs of current technology. He has contributed more than 25 research papers at national, International Journals. He is life member of Indian Society of Technical Education. His area of interest are in Databases, intelligent Systems, and Web Engineering.



Dr. B.B. Meshram (CSI LM'95, IE '95) is Professor and head of Computer Technology Department at VJTI, Matunga, Mumbai, Maharashtra state, India. He received bachelor degree, Master degree and doctoral degree in computer engineering. He has participated in more than 30 refresher courses to meet the needs of current technology. He has chaired more than 15 AICTE STTP Programs and conferences. He has received the appreciation for lecture at Manchester and Cardiff University, UK. He has contributed more than 200 research papers at national, International Journals. He is life member of computer society of India and Institute of Engineers. His current research interests are in Databases, data warehousing, data mining, intelligent Systems, Web Engineering and network security.

