Model Driven Approach For Developing Cloud Application

Roshan Kumar, Jeevith Bopaiah, Pranjal Jain, Dr. Nalini N, Dr. K. Chandra Sekaran

Abstract: In recent years, cloud computing has become increasingly popular. Many application developers are now switching to cloud platform. Thus there is a need to develop a framework that guides the software development process for cloud software. Model driven architecture is a software development process that models the system at different levels of abstraction and uses transformation tools for model-model or model-code transformations. In this paper we have presented an overall generic framework using model-driven approach which creates software applications that are robust and flexible in nature. The applications are developed at a platform independent level so that they can be easily adapted to different cloud platforms.

Keywords: Model Driven Architecture, Service Oriented Architecture, Web Services, Meta-models, Meta Object Facility.

1. INTRODUCTION

Cloud computing is an emerging technology that has evolved from other technologies like distributed computing, grid computing, virtualization, etc., and supports the theme of one application and many users. The earlier software development methodologies involves writing scripts which describe steps to deploy an application on target platform and these scripts are tightly coupled to the platform. Hence when the application is updated these scripts must also be modified to accommodate the changes. Thus these scripts are written many times during the software life cycle. These methodologies describe the system with respect to a particular development environment which is difficult to understand by the domain experts and creates a gap between domain and system experts. The various teams involved in the development process uses specialized tools based on their level of expertise and hence they cannot co-ordinate with each other in the development process and the applications developed conform to a specific platform. This results in re-designing the applications every time it has to migrate from one cloud provider to another [1]. In order to alleviate the above problems we should design applications in platform-independent manner.

- *ROSHAN KUMAR is currently pursuing bachelor degree program in computer science engineering from Nitte Meenakshi Institute of technology(VTU),BANGALORE, INDIA, PH-9535692887. E-mail: rkroshank7@gmail.com*
- *JEEVITH BOPAIAH is currently pursuing bachelor degree program in computer science engineering from Nitte Meenakshi Institute of technology (VTU), BANGALORE, INDIA, PH-9880407161. E-mail: jeevith.christite@gmail.com*
- *PRANJAL JAIN is currently pursuing bachelor degree program in computer science engineering from Nitte Meenakshi Institute of technology(VTU),BANGALORE, INDIA, PH-9886316970. E-mail: pranjal17pokharna@gmail.com*
- *Dr.Nalni.N is currently working for Professor and Head Department of Computer Science and Engineering at Nitte Meenakshi Institute of Technology,Bangalore.PH-8722455452.E-mail: nalinaniranjan@hotmail.com*
- *Dr.K Chandra sekaran is currently working as a professor in NITk, surathkal, mangalore. E-mail- kchnitk@gmail.com*

Object Management Group (OMG) introduced the Model Driven Architecture (MDA) as an approach to software development [2]. MDA describes the system at a higher level of abstraction in which models form the primary artifacts. These models are simple as they describe only the essential features of the system and helps in better understanding the system. The models are described at different levels of abstraction:- 1) CIM (Computation Independent Model) - describes the basic features of the system and produces a structured and coherent document of requirement specification. 2) PIM (Platform Independent Model) - describes the structure, behavior and functionality of the system in a generic manner. 3) PSM (Platform Specific Model) - describes the system with respect to a specific platform. MDA allows automation of various steps in the development process and it semi-automatically generates code from the models. Therefore, it produces error-free code and increases the productivity of the application. This paper is organized into five sections. In section 2 we discuss the related work. Section 3 describes the general model driven framework for developing cloud application. Section 4 presents an analysis of our approach. Section 5 draws the conclusion of our work.

2. RELATED WORK

Application Platform Based on model driven approach. A proper design of SaaS Application Platform Based on MDA [9] divides whole architecture into two layers. The first layer describes about data, business process and user interface. The second layer describes about model engines responsible for parsing the models to achieve customization. Through customization each user gets unique identity for the service. This service can be customized or altered as per the requirements of customer. Thus customer will get specified field on preference. But this paper has not mentioned about scalability, which can be monitored by including numeric values in the model, which creates multiples copies when the demand for services increases above these values and maintains the proper working of the application. Cloud SaaS and MDA. Software development process emphasis on the relation between problem domain and solution domain and thus increases the manageability, productivity and speed. Service Oriented Architecture decomposes the application into services enabling reusability, autonomy, abstraction, compensability, statelessness, discoverability and loose-coupling of services. The authors, Ritu Sharma and Manu Sood [6], [8], [11] mentioned about the modelling of the system at a higher level of abstraction to prevent the effects of

technology change on the application. But it does not notice/mark the issues of customization as describes by Xiaoyan Jiang, Yong Zhang and Shijun Liu [9]. Model Based Adaption. Christian Inzinger, Benjamin Satzger, Philipp Leitner addresses the maintenance of application by the cloud provider [10]. They use the information from the service provider to adapt the application as per requirements of the service provider. Moreover, they use the runtime information to determine the behavior of the application and provides an efficient management solution to the software. Application management includes infrastructure provisioning, application maintenance and thus adaption at runtime. Table 1 shows the comparison between the different approaches to cloud software development.

Architectures	Drawbacks
1)DSL based Approach to Software Development and Deployment on cloud [12]	It is platform dependent and it requires deploying Domain Specific Language on every cloud platform where the software is to be deployed
2) Reusability Framework for Cloud Computing [13]	It compromises on the quality of the software produced and some of the required components may not be available in the cloud repository
3) On Demand Service Oriented MDA Approach for SaaS and Enterprise Mashup Application Development [14]	It does not address customization of services and hence does not describe how multi-tenancy is achieved
4) Object Oriented Architecture [15]	It is platform dependent and the application is developed with respect to a particular development environment.
5) Cloud SaaS and Model Driven Architecture [11]	It does not mention about customization of services as per user requirements.

Table1. Comparison between different software development approaches

The earlier software development approaches listed above have a few drawbacks:- The development process are tied to a target platform. The customization of services is not addressed. Hence it does not describe how multi-tenancy is achieved. Developing applications using reusable components compromises on the quality of the software.

3. GENERIC MODEL DRIVEN FRAMEWORK FOR CLOUD

3.1 Combining Service Oriented Architecture and Model Driven Architecture:

Service Oriented Architecture is a development paradigm which involves breaking down of an application into smaller services. These services are loosely coupled and can communicate with each other. SOA enables interoperability among services and these services can be reused in different applications. Thus SOA decreases time and cost of development in software product-lines. MDA can be combined with SOA by describing each of these services as models and then integrating them to form the complete application. By combining these two technologies we incorporate platform-independence and interoperability into our application.

3.2 Our Approach:

We describe a generic framework for developing an application for cloud environment as shown in figure 1. We start the development process with requirement specification phase in which we develop a Computation Independent Model (CIM).

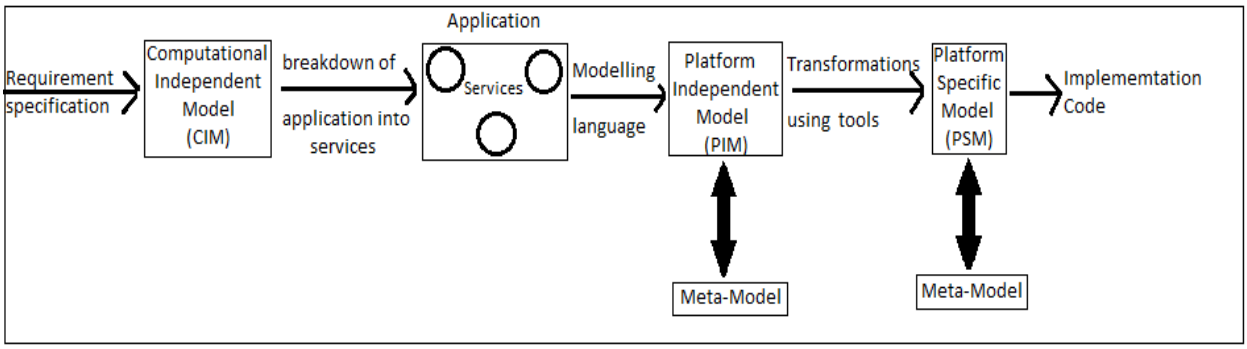

Figure 1. Model Driven Framework for Cloud Application

Then by leveraging the Service Oriented Architecture (SOA), we decompose the application into several services, each implementing a smaller function of the application. Then we create a model for each of these services using model driven approach. These represent the Platform Independent Model (PIM).

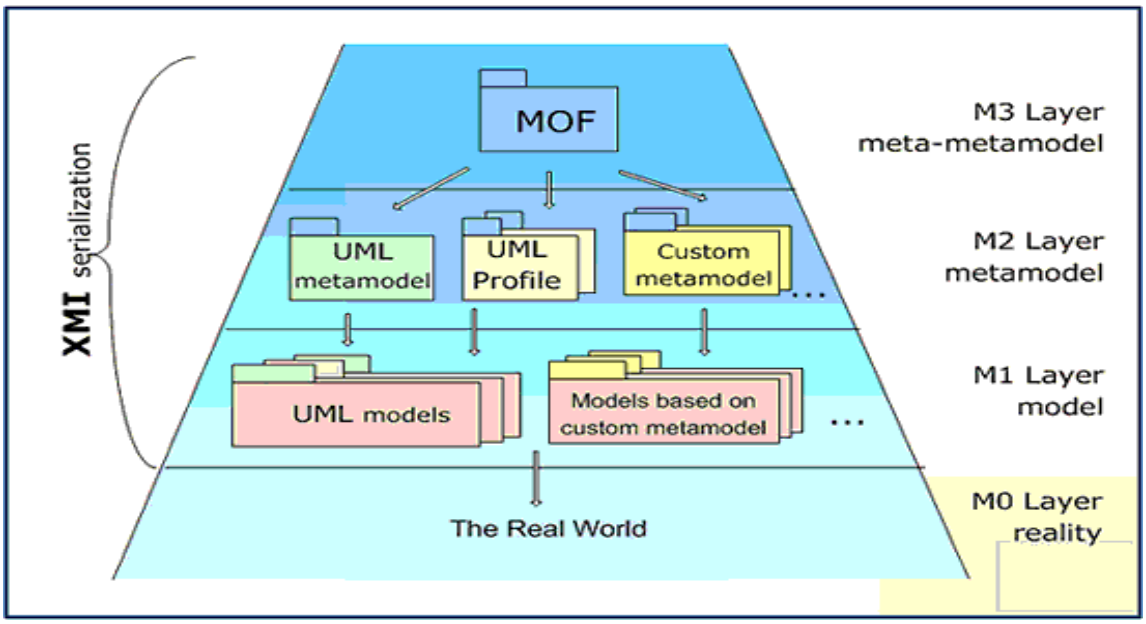

Figure: 2 Meta Model Architecture

These models should conform to their metamodels. Metamodels, in simple terms are model of a model [3]. They describe the various elements, constructs and syntax of a model. Figure 2, shows four-layered architecture of Metamodel as given by Dragan Djuric, Dragan Gaševic, Vladan Devedžic [4].

- The lowermost layer contains the real-world data to be modelled. It represents the area of concern, and depicts the behaviour and functionality the system should exhibit.
- M1 layer contains the models. Any modelling language can be used to create these models. These models describe the real world data present in M0 layer. Various elements and constructs of the modelling language are used in these models.
- M2 layer is also known as the metamodel layer. It consists of metamodels which describe the user models in M1 layer. Models in M1 layer are instances of these metamodels. Meta models are essential in model driven development because they describe the various elements and syntactic details used in the user-defined models. These metamodels are used by the transformation tools to determine how each construct in source model is mapped to the target model.
- M3 layer represents the Meta Object Facility (MOF) - a standard that provides Meta Object Language (MOL) for standardized description of the metamodels specified in different modelling languages in M2 layer. This helps to integrate the various models, specified in different modelling languages into one framework and carry out various model transformations.

This four-layered architecture supports the use of different modelling languages to create the models and their metamodels. Thus it increases flexibility in the application development process. Any modelling Language can be used to create PIM, however UML is most commonly used. Another self-expressive language is Object Process Model (OPM) [5]. It models the systems as objects and processes and establishes the relationship as processes acting on objects. They are self-expressive because they do not have separate metamodels and the elements of the models describe itself. Also they describe the system as a single model and hence relieves the effort of managing large number of models in complex application. Once the PIM is developed, the Platform Specific Model (PSM) is generated from PIM using transformation tools [6]. These tools are useful for model-model, model-code transformations. The transformation tools contain transformation definition which are specified by transformation rules. These rules are also represented as models describing the semantics of how a construct in source language is interpreted in target language. During mapping of PIM to PSM each service model in PIM is mapped individually to target environment. Finally these models are integrated to form a complete PSM. After each such transformations the Quality of Service (QoS) is measured to ensure that the models conform to the requirements. Any number of PSM can be generated from the PIM using different transformational tools. Now the implementation code is generated from PSM. This is semi-automatic step in which some user involvement is required. The tools automatically generate the code templates from the models, however additional information should be entered by the user to obtain a complete code

3.3 Deployment

Once the application is developed the next phase is deployment in the cloud [7]. The application to be deployed consists of models and application bits. The models is simply the composition of various models, and the actual function and behaviour of different components are specified in application bits. These application bits are available as a package or URL from where it can be downloaded. These models and application bits are combined into a bundle and introduced into cloud. In the cloud, the platform makes an analysis of the application bundle and develops a plan to distribute the various components among different servers so that it can achieve the required Quality of Service (QoS). This plan should be dynamic in nature such that when the demand for services increases new instances must be created and when the demand decreases these instances should be deleted. When the application is up and running we need to provide interface through which client in a remote location can access these services. This interface is provided by web services [8]. The web service has several standards- WSDL, UDDI, SOAP. Web Service Definition Language (WSDL) describes the services available within the application and also provides information about how to access these services. Simple Object Access Protocol (SOAP) specifies a messaging format for exchanging information among services via a communication protocol. Universal Description, Discovery Integration (UDDI) maintains the registry of different types of services and locates them. Each user has a unique ID through which they can access the service. Each user uses different sets of options based on their working environment. This implies that personalization of service is important factor [9]. This can be done by modifying the models. The application package consists of model templates, model description files and model engines. The model templates are the actual models in standardized form. The model description files are files obtained after modifying or customizing the model templates. They are specific to one user and stored in a separate directory of that user. Thus each user uses the services independent of the other. The model engines helps in customizing the template files and producing model description files.

3.4 Scalability

One of the characteristics of cloud applications is scalability. When the demand for service increases above a threshold level new instances of services are created. These instances must be deleted when the demand falls below a certain level. We can automatize the creation and deletion of instances by including appropriate metrics in the model which specifies when the instances are to be created and when it is to be deleted. Along with this, we can also develop models to provide information about how the application should adapt to the cloud environment [10]. Different service providers want their applications to behave in a certain way, some require that their application is always available, while others want their application to have faster response time at peak hours and others focus on lower operational costs, etc. This information helps the cloud provider to make decisions on better management of application. Since cloud providers manage different applications they use the history of similar applications to predict the workload and dynamically allocate resource and provide the required service to various clients.

3.5 Overcoming the drawbacks

In our approach we have used Model driven architecture which creates platform independent model and hence the application can be ported to different cloud platforms. We have also described customization of services which helps us to create multi-tenant aware applications. Here the users can customize the service as per their requirements and each user's data is stored in a separate directory associated with their user-id. Thus each user can access the service independent of other users and one copy of the software can provide service to multiple clients. Thus our approach takes into consideration the various issues present in earlier architecture and provides a superior framework for development of cloud software.

3.6 Exemplification: Online Bus Reservation System (OBRS)

We illustrate the basic structure of our framework using an example- Online Bus Reservation System. This system allows people to book tickets online. They can access these service by first registering themselves with the services which is made available through a remote client or web browser. The basic services provided by the system are reservation, cancellation, check fares, seat availability etc. The users of this system can be classified into two groups:- customers and system administrators. We first model the system at the highest level of abstraction known as Computation Independent Model(CIM). The use case diagram for online bus reservation system is shown in figure 3. The two roles

identified in the UML diagram are:
1. Customer: Customers are the people who use the system to book or cancel tickets online. They avail the facilities provided by the provider.
2. System administrator: System administrator is a person who represent the travel organization. He has special privileges and can reschedule buses, change routes, modify fares. The changes made by the administrator is automatically updated in the system.

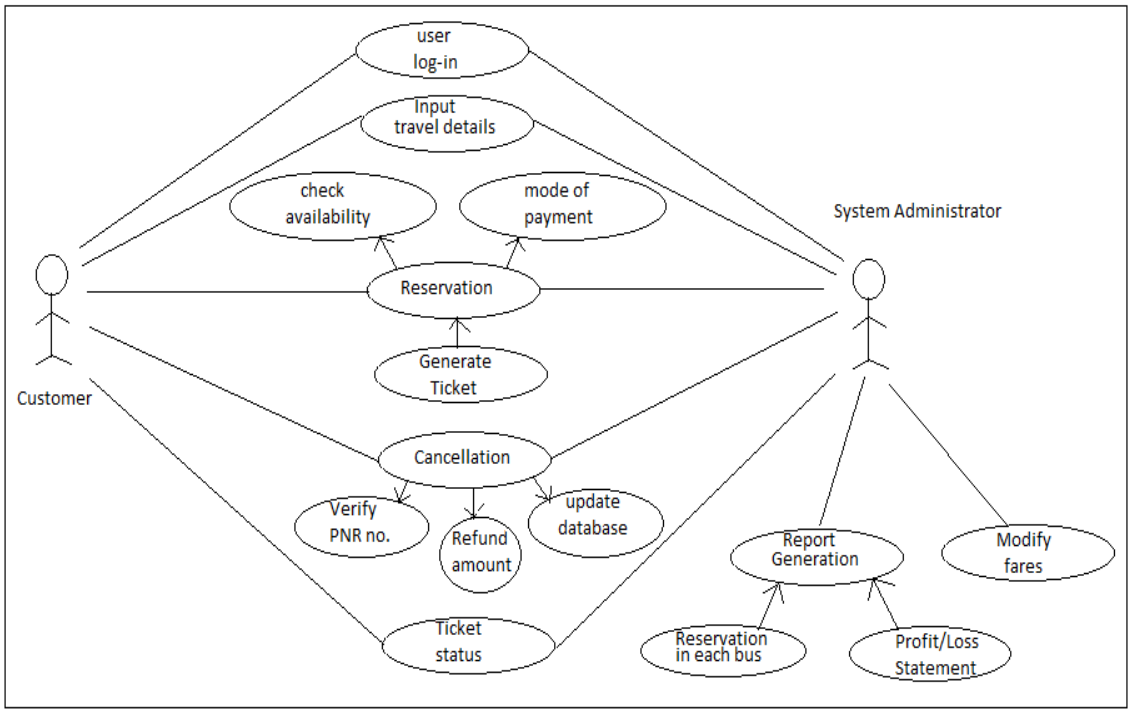

Figure 3. Use Case Diagram for OBRS

The various steps involved in booking ticket is as follows:-

- The customer logs in using his username and password.
- The user then inputs the source, destination and date of travel.
- The software then displays a list of buses along with their route-no, departure/arrivals time, no of seats and their cost.
- Then the user chooses a seat from available seat in the seat matrix.
- Next he proceeds for payment using debit/credit card or other modes of payment through a secure card verification system.
- Once the seat is booked a PNR no. is sent to the customer. The PNR no. can be used to track ticket status. The same is also used for cancellation.

The platform–independent model (PIM) is represented using UML language. It is modelled as classes and relationship between them. The relationship between classes represent the association between them i.e an object of one class can refer one or more objects of a related class. The PIM of online bus reservation system is shown in figure 4

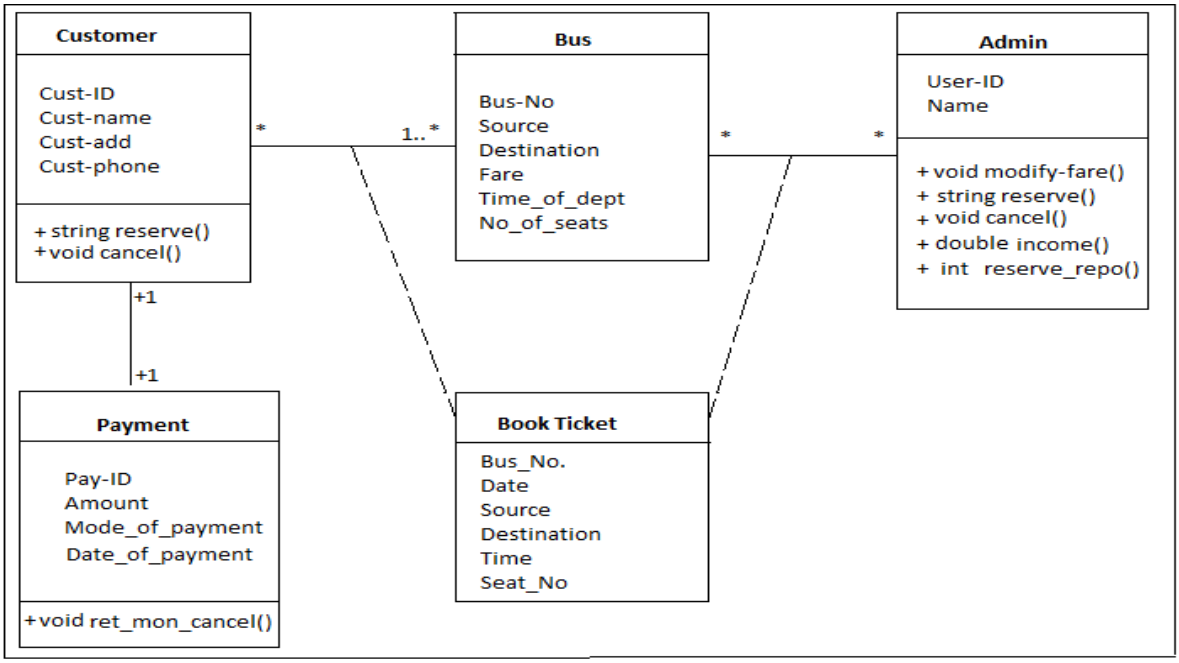

Figure 4. PIM for OBRS

The various classes are as follows

Customer: It represents the client using the service to book tickets. It has attribute like cust-id, cust-name, cust-add, cust-phone. The cust-id is the username which authenticates the user. The cust-name, cust-add, cust-no. refers to the name, address and phone number of the customer. It has many-to-many association with Bus class and one-to-one association with payment class. It also has many functions which enable the customer to book tickets, cancel tickets and make payments.

Bus: It maintains the details of all buses in a particular travels company. It has attributes like bus_no., source, destination, fare, time_of_dept (time of departure), no_of_seats (number of seats available). It is modified whenever necessary by the system administrator. It has many-to-many association with customer class and admin class. Number of seats available has to be updated whenever a ticket is booked to ensure that there is no overlapping of transaction.

Admin: It is a class with special privileges. It represents the organization providing the travel facility. It has a unique user-id which has privileges to modify fares, make reservations and cancellations on behalf of the customer. Admin also generates reports to calculate net income, view the reservation report for each bus. Based on these reports admin reschedules the buses appropriately.

Payment: This class has attributes Pay_ID, amount, mode_of_payment, date_of_payment. It helps in transaction of money from customer's account to travels company. It has a function ret_mon_cancel which is used to refund the money to customer in case of cancellation.

Book_Tickets: This is an association class. An association class is one which is related to classes which already have a relationship with each other. It has the following attributes bus_no, date, source, destination, seat_no, etc which provides details of seats booked.

The Platform Specific Model(PSM) is generated from PIM by using transformation tools. Here in this example we have used Java as the target language as shown in figure 5. The only difference between PIM and PSM is that in PSM the data types of attributes are specified. Also functions are specified along with their data types. These language specific details are not present in the PIM model. Various other PSM can also be generated based on different target platform such as .NET, CORBA etc. Once the PSM is generated the next step is to generate implementation code from the models by using transformation tools which maps each construct in model to target language.

3.6.1 Transformation Definitions: PIM to PSM

Platform Independent Model is transformed into Platform Specific Model by using transformation tools. These tools contain transformation rules that map the constructs in the source model to target model. Here we discuss the transformation rules that map PIM for Online Bus Reservation System to PSM targeted on java platform. The rules are as follows [16]:-

- Each class in the PIM is mapped to its corresponding class in PSM.
- For each public attribute of a class present in the PIM –
  - a private attribute exists in the PSM.
  - the type of the attribute is specified in java language.
  - there exists two public operations getAttributeName() and setAttributeName().

- Each operation of a class present in the PIM is mapped to a private member function with its return type specified in java language.

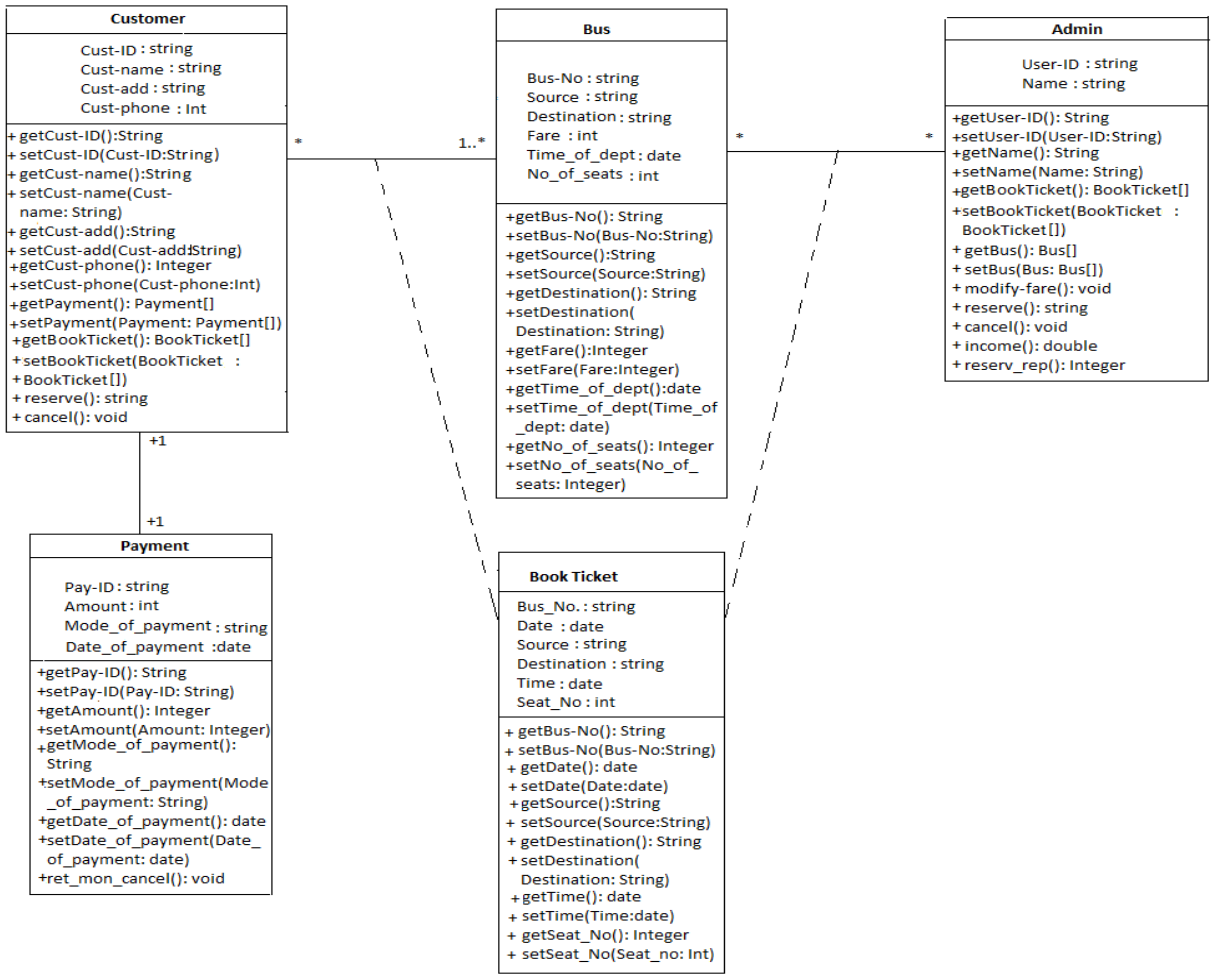

Figure 5. PSM for OBRS

To provide complete services to the customer this Online Bus Reservation system needs to interact with another service called Banking service. This interaction is required because the customer pays to the travel organization through smart cards or other forms of electronic money. This banking service provides secure transaction between customer and agency. This takes place as follows. The customer uses Online Bus reservation system to book tickets. Once the seat is selected they proceed for payment. The web service client for Online Bus Reservation System requests the banking service to provide service for secure payment. Then the banking service displays forms to be filled by the customer and finally enabling the transfer of money from customer's account to agency's account. The same sequence of events takes place with respect to cancellation, however the transfer of money is from agency's account to customer's account. Once these transactions are completed successfully the seat is booked and ticket is generated for the customer.

4. ANALYSIS

Model driven approach for software development creates models that are platform independent, thus making the application portable and robust. Only one platform independent model is created and transformed into several different platform specific model. This prevents re-designing the application for different platforms and increases the lifetime of the application. The system is described as models and transformation between them which provide better understanding of the system to the stakeholders. Service Oriented Architecture decomposes the application into services. These services are loosely coupled to each other which enables these services to be reused in different applications. This reduces the time and effort invested in the development process. Multi-tenancy is achieved by customization of the services in which each client can customize some parts of the service such as user interface. Each client's data is stored in a separate directory. This ensures privacy of client's data. Some of the phases in this approach can be automatized. Hence the application need not be subjected to extensive testing, which reduces the cost involved in the testing process. This approach also has

a few drawbacks: - Modelling a large application may result in the creation of several models and managing these models is complex. These models developed cannot be executed directly, instead it should be transformed into implementation code. Generation of implementation code from the models is semi-automatic and requires additional information to be provided by the system developer. The applications developed using model driven approach cannot directly interact with legacy applications and it requires re-engineering techniques to extract models from legacy applications so that it can communicate with model driven applications. Extracting the models from legacy systems incurs additional overhead.

5. CONCLUSION

This paper presents a model driven paradigm for developing cloud enabled applications. It provides a complete description of different phases of model driven development. This approach enables automatization of the development thereby reducing time, effort and money invested in the development process. Applications developed using this approach are robust, flexible, agile and can dynamically scale as per user requests. In this paper we have considered only Unified Modelling Language (UML) to model the system. We are currently working on other modelling languages which can be used to model the system.

6. REFERENCES

[1] Danilo Ardagna, Elisabetta Di Nitto Politecnico di Milano. “MODACLOUDS: A Model-Driven Approach for the Design and Execution of Applications on Multiple Clouds”. (Position paper)

[2] OMG Model Driven Architecture. [Online]. Available: http://www.omg.org/mda

[3] Ritu Sharma and Manu Sood 2011. “A Model-Driven Approach to Cloud SaaS Interoperability”. International Journal of Computer Applications (0975 – 8887)

[4] Dragan Djuric, Dragan Gaševic, Vladan Devedžic. “The Tao of Modelling Spaces”, in Journal of Object Technology, vol. 5. no. 8, Novmeber-December 2006, pp. 125-147.

[5] Nikola Tankovi´c . Model Driven Development

[6] Approaches:Comparison and Opportunities [online]. Available:

[7] www.fer.unizg.hr/_download/repository/N.Tankovic,_rad_za_KDI.pdf

[8] Ritu Sharma and Manu Sood . “Enhancing Cloud Saas Development with Model Driven Architecture”. International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol.1, No.3, November 2011, DOI: 10.5121/ijccsa.2011.1307.

[9] Javier Esparza-Peidro, Francesc D. Mu˜noz-Escoi. “Towards the Next Generation of Model-Driven Cloud Platforms”. Institut Universitari Mixt Tecnologic` d’Informatica`Universitat Politecnica` (SPAIN), Technical Report TR-ITI-SIDI-2011/001

[10] Divya Sharma, Ritu Sharma and Manu Sood . “Modeling Cloud SaaS with SOA and MDA”.

[11] Xiaoyan Jiang, Yong Zhang and Shijun Liu. “A Well-designed SaaS Application Platform Based on Model-driven Approach”.Ninth International Conference on Grid and Cloud Computing .978-0-7695-4313-0/10 © 2010 IEEE DOI 10.1109/GCC.2010.62

[12] Christian Inzinger, Benjamin Satzger, Philipp Leitner, Waldemar Hummer and Schahram Dustdar. “Model-based Adaption of Cloud Computing Applications”. Modelsward 2013- Internatioal Conference on Model-Driven Engineering and Software Development.

[13] Ritu Sharma and Manu Sood. “Cloud SaaS and Model Driven Architecture”. Proc. of the International Conference on Advanced Computing and Communication Technologies (ACCT2011)

[14] Krzysztof Sledziewski, Behzad Bordbar and Rachid Anane. “A DSL-based Approach to Software Development and Deployment on Cloud”. 24th IEEE International Conference on Advanced Information Networking and Applications 2010.

[15] Sukhpal Singh1, Rishideep Singh. “Reusability Framework for Cloud Computing.” International Journal Of Computational Engineering Research Vol. 2, 2012.

[16] Xiuwei Zhang, Keqing He, Jian Wang, Jianxiao Liu, Chong Wang, Heng Lu. ”On-Demand Service-Oriented MDA Approach for SaaS and Enterprise Mashup Application Development”. International Conference on Cloud Computing and Service Computing 2012 IEEE.

[17] Nabil Mohammed Ali Munassar, Dr. A. Govardhan. “Comparison between Traditional Approach and Object-Oriented Approach in Software Engineering Development. International Journal of Advanced Computer Science and Applications, Vol. 2, No. 6, 2011.

[18] Erik Perjons, DSV. “The Introduction to MDA, the Core of MDA“ [online]. Available: ftp://ftp.dsv.su.se/users/maria/MDA1100.pdf.