A Model-Driven Approach to Cloud SaaS Interoperability

Ritu Sharma
Himachal Pradesh University
Summer Hill, Shimla, India

Manu Sood
Himachal Pradesh University
Summer Hill, Shimla, India

ABSTRACT
Cloud computing is a promising computing paradigm wherein the resources are made available to the clients as services, over high bandwidth networks. Cloud SaaS refers to a cloud computing service model in which the software applications are offered as services. These cloud software applications may require interacting with each other in order to accomplish a task. Thus, interoperability among services is an important issue for consideration in cloud computing. With the hardware and software technologies constantly evolving at a tremendous pace, the IT industry is persistently faced with the challenges of technology obsolescence. These changing technologies have more serious consequences in B2B context. Therefore, it becomes essential to promote a technology-agnostic software development approach that could alleviate the undesirable effects of technology change. In this perspective, Model-driven Architecture (MDA) becomes a preferred methodology for developing cloud software services. This paper presents an MDA-based model-driven approach to develop cloud software services and exhibit interoperability between them.

General Terms
Cloud computing, Cloud SaaS, Model Driven Architecture, Cloud Service, WSDL Metamodel, Interoperability

Keywords
Platform Independent Model, Platform Specific Model, Meta model, Web Service, WSDL, SOA

1. INTRODUCTION
Cloud computing, a leading edge technology, is reshaping the world of Information and Communication Technology (ICT). In the ambit of cloud computing, a variety of computing resources such as processors, storage, software applications, databases, development environments etc are made available to the clients as services, over a high speed network. These resources in the cloud may be accessed as per demand across a simple interface such as a browser running on devices that range from desktop, laptop, iPhone or even a PDA. The usage is metered and the client may choose to either pay for every single use of the service or subscribe to it for a definite period [1, 2]. A software application running in the cloud can be implemented on a variety of technology platforms. Besides, the applications in the cloud may require interacting with each other to achieve a business goal. In spite of the business functionality remaining relatively constant, as the technologies evolve these legacy software applications tends to become obsolete and need to be replaced. This incurs additional expenses on part of the service providers. A software development approach that can cope with multiple implementation technologies and extend the lifetime of the cloud software application is, therefore, desirable and required.

Model-Driven Architecture (MDA), an Object Management Group (OMG) initiative, is a model-driven software development approach which mitigates the undesirable effects of technology change to a considerable extent. This approach does not eclipse the various technologies; rather, it works with them synergistically and enhances their efficiency. The MDA approach uses formal models to describe various aspects of the software system. The semantically rich, programmable models are the prime artifacts in the entire software development process directing the course of software application development from understanding, design, construction, deployment, operation, maintenance and modification. The models themselves are specified in well-defined modeling languages. Programming with semantically rich modeling languages enhances the productivity, quality and longevity of the software application.

In this paper, the authors attempt to present a model-driven approach for ensuring interoperability among the software services in the cloud. Section 2 introduces the fundamental concepts of cloud computing. Section 3 focuses on the significance of SOA and Web services in cloud SaaS. Section 4 highlights the concept of models and transformation in MDA. Section 5 illustrates the development of a cloud SaaS based on MDA. The WSDL and its metamodel are discussed in Section 6. Section 7 briefly lists the rules for transforming the PIM of a cloud SaaS into its WSDL PSM. Section 8 addresses the interoperability issue in cloud SaaS. Section 9 draws the conclusion of the work attempted by the authors in this paper.

2. CLOUD COMPUTING AND CLOUD SERVICES
Cloud computing is the technology underpinning the cloud services. Cloud services are analogous to utility services, and offer computing resources as services to the clients. The resources are pooled and shared among the clients, thereby ensuring their optimal utilization. These resources are dynamically scalable and can be acquired from the pool or released to the pool with great elasticity, in order to meet the varying demands of the clients. The virtualization technology makes this allocation and reallocation of resources transparent to the client, giving him an illusion of the existence of infinite resources in the cloud. Based on the type of resource offered as a service, the cloud service models are broadly categorized as – (i) Cloud Software-as-a-service (SaaS) where software applications running on cloud infrastructure are provided as services to the consumers, (ii) Cloud Platform-as-a-Service (PaaS) where programming languages, development tools, utilities etc comprising the development platforms are provided

as service, and (iii) Cloud Infrastructure-as-a-Service (IaaS) where the basic computing resources such as processors, storage, networks etc are provided as service to the consumers. A cloud itself may be deployed as – a public cloud, a private cloud, a hybrid cloud or a community cloud [3, 4, 5].

The key attributes that characterize this technology include – 1) ubiquitous access of resources, 2) resource pooling and sharing, 3) dynamic scalability of resources, 4) on-demand availability of resources, and 5) metered usage of resources.

The advantages of this technology are that the resources are owned, managed and controlled by third-party service providers and the client is only required to perform certain configuration settings on the accessing device, in order to use the services. Also, the ability to use the service requires no or minimal IT skills on part of the user. The use of cloud services prevents the users from incurring any upfront infrastructure costs for their business solutions.

3. SIGNIFICANCE OF SOA AND WEB SERVICES IN CLOUD SAAS

Cloud computing technology has evolved from a variety relevant legacy technologies and concepts such as distributed computing, grid computing, virtualization, Web Service with its supporting technologies such as Web service Definition Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description, Discovery and Integration (UDDI), Service Oriented Architecture (SOA), Software-as-a-Service (SaaS) to mention a few.

Service Oriented Architecture (SOA) represents an architectural style in which the automation logic of an enterprise application is decomposed into smaller, distinct units of logic called services. These services are then configured to provide business solutions. These units of logic in SOA must conform to the technology-independent principles of loose-coupling, autonomy, abstraction, reusability, discoverability, statelessness, adherence to a service contract and compos ability [6]. SOA is an implementation-agnostic paradigm that can be realized with any suitable technology platform. Currently, the Web Services platform with its supporting standards is the choice for realization of SOA.

A web service is any service that is available over the Internet or an intranet, uses standardized XML messaging system and is self-describing, discoverable and not tied to any operating system or programming language [7]. It is characterized by open standards, cross-platform capabilities and human-readable messages that can be sent across firewalls. The standards comprising the Web Services framework are – Web Service Description Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description, Discovery, and Integration (UDDI). WSDL is an XML-based document that describes a web service. It specifies the location of the service and the functionalities exposed by the service. SOAP is an XML-compliant communication protocol that specifies the message format for exchanging information among the applications in a distributed environment. The service description registry and discovery is realized through UDDI. Thus, while WSDL specification provides a means to expose the details of how to interact with the service and its methods, SOAP provides a protocol for calling remote methods and passing complex data types.

While SOA focuses on decomposing the business logic into distinct services, which are then orchestrated or choreographed to provide business solutions, cloud computing technology focuses on provision of computing resources as services to the consumers. The cloud should not be looked at as a new architecture, but instead as another option of storing and running services within SOA [8]. As the software applications in the cloud may be implemented on varied platforms and technologies, an SOA-based cloud will naturally abstract and hide the vendor-specific and platform-specific aspects of the various software applications by leveraging the open Web services communications framework and establishing a predictable communications medium for all applications exposed via Web service. A quality 'Cloud' therefore requires the individual cloud services to conform to common design principles of SOA in order to fully realize the benefits of reusability, interoperability, federation, and others.

4. MDA AND MODEL TRANSFORMATION

Model-Driven Architecture (MDA) is an open, vendor-neutral approach [9] to enterprise application development where modeling activity drives the process of software development. The models in MDA are defined at different levels of abstraction based on separation of concerns – the Computation Independent Model (CIM), the Platform Independent Model (PIM) and the Platform Specific Model (PSM). Each level may be comprised of one or models to specify the structural, functional and behavioral aspects of the system. The CIM is at the highest level of system abstraction and describes the business functionality of the system in a software independent manner using a vocabulary familiar to domain experts. The PIM is at the next lower level of abstraction and specifies the structure and function of the system in a manner that is independent of the platform technology that would be used for its implementation. The PSM is at the lowest level of abstraction and describes the system in terms of constructs that are specific to a particular implementation technology. A 'platform' is a set of subsystems and technologies that provide a coherent set of functionality through interfaces and specified usage patterns, which any application supported by that platform can use without concern for the details of how the functionality provided by the platform is implemented [10].

The key to success of the MDA approach lies in the automation of model-to-model or model-to-code transformation. A transformation is the process of automatic generation of a target model from a source model, according to a transformation definition. A transformation definition comprises of a set of transformation rules that describe how a model in the source language can be transformed into a model in the target language. A transformation rule is a description of how one or more constructs in the source language can be transformed into one or more constructs in the target language [11]. An essential characteristic of transformation is that it should be able to preserve the semantics of the source and target models.

The Model Driven Architecture systematically builds upon other OMG standards such as the Unified Modeling Language™ (UML), the Meta Object Facility (MOF), XML Metadata Interchange (XMI), Software Process Engineering Metamodel (SPEM), the Common Warehouse Metamodel (CWM) and Business Process Modeling Notation (BPMN).

The models in MDA must conform to their respective metamodels. The OMG's Meta Object Facility (MOF) standard facilitates this conformance. Meta Object Facility (MOF) is the language definition mechanism of MDA [11]. It defines an abstract syntax of modeling constructs for specifying, constructing, and managing technology-neutral metamodels; the technique is referred to as metamodeling. A four-layered MOF architecture [12], starting from bottom to the top, comprises of – the data/information layer (M0), the model layer (M1), the metamodel layer (M2), and the meta- metamodel layer (M3). Each i^{th} layer in this architecture consists of a set of constructs which are used to define the elements of $(i-1)^{th}$ layer. While, a model at M1 layer is an abstraction of a real world phenomenon described at M0 layer, a metamodel at M2 layer is an abstraction of the model at M1 layer. The M3 layer essentially models the technology neutral metamodels at M2 layer. The MOF M3 layer is self-describing i.e. the M3 elements are instances of M3 elements. The MOF is, therefore, a *closed* metamodeling architecture.

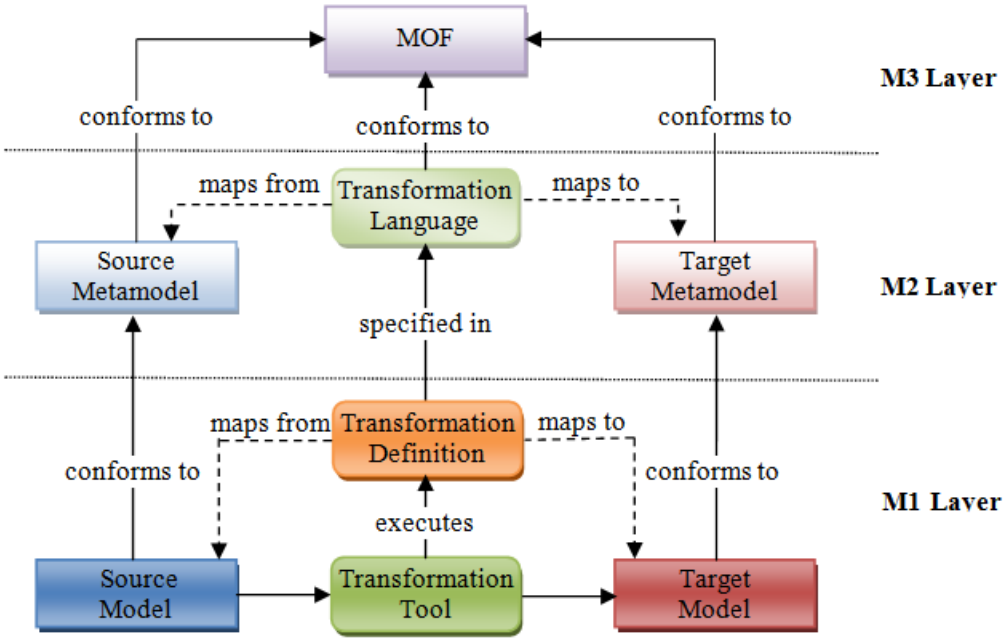

Figure 1 Model Transformation in MDA

Figures 1 depicts the model transformation in MDA based on MOF standard. A transformation tool transforms a source model into a target model by executing a transformation definition specified for the purpose. The source and target models must conform to their respective metamodels. The transformation definition is described using a transformation language (metamodel) which is chosen such that it maps the constructs of source metamodel into constructs of target metamodel. In a PIM to PSM transformation, the PIM is the source model and the PSM is the target model. The PIM and PSM are specified at the M1 level of the MOF architecture, and must conform to their respective metamodels at M2 level.

The primary goals of MDA are – interoperability, reusability and portability through architectural separation of concerns [10]. The MDA approach to software development benefits the stakeholders by enhancing productivity, improving software quality, preserving Return on Investment, reducing development cost and reducing time to market.

5. INCORPORATING MDA APPROACH IN CLOUD SAAS DEVELOPMENT

US NIST (National Institute of Standards and Technology) [5] defines a Cloud SaaS as a service model in which software applications running on a cloud infrastructure and managed by the service provider are offered as services to the clients. The application may be a simple one performing a single function or a complex one performing a set of related functions. As discussed earlier, with the technologies evolving constantly, these software applications in the cloud are vulnerable to technology obsolescence. Also, the changing technologies have more serious ramifications in the B2B context as it becomes increasingly difficult to control the impact of change when external partners are involved. Hence, rather than developing these cloud software services directly using available technologies, modeling them at a higher level of abstraction will decouple them from the undesired effects of technology change and will also enhance their longevity. An MDA based development of cloud SaaS (application) will enable defining these services in a technology-independent manner and will play a significant role in improving the quality of cloud software services, making them more robust, flexible and agile [13, 14, 15, 16]. Encapsulating the business logic in a manner that is independent of the technical details will formally capture the essence of the applications; and will also make it possible to reuse them in a variety of contexts [17, 18].

Based on MDA approach, a model-driven development of a cloud SaaS – an Online Hotel Reservation System (OHRS) – is illustrated. OHRS is a cloud software application which may be availed as a service – OHRS_Service – by any small or medium Hotel enterprise. It provides a variety of online business

functionalities to the hotel enterprise and hotel customers such as book an accommodation, cancel a booking, check availability of accommodation, and generate reports for the management etc.

Although MDA does not restrict itself to the use of UML for modeling, we are using UML to specify the PIM and PSM of the cloud SaaS. The PIM of the OHRS cloud service provides a formal definition of the operations offered by the service that can be accessed through the OHRS_Service interface. Although, the service offers a range of operations through its interface, for the sake of simplicity, only a couple of operations associated with the service are depicted in Figure 2. The business rules for the operations are specified by declaring the constraints as pre-conditions, post-conditions and invariants in Object Constraint Language (OCL) and are depicted in the lower part of the Figure 2. Combining UML with OCL results in a PIM specification that is more precise, rigorous and semantically rich. The PIM is then transformed into a PSM targeted on WSDL. Figure 3 depicts the mappings of the source model (PIM) constructs to the target model (PSM) constructs. Again, to keep the diagram simple, mapping for only one operation is exhibited. The 'checkAvailability' operation in Figure 3 takes the type of unit, the number of units and the dates specifying the booking period as input, and provides the availability status as output.

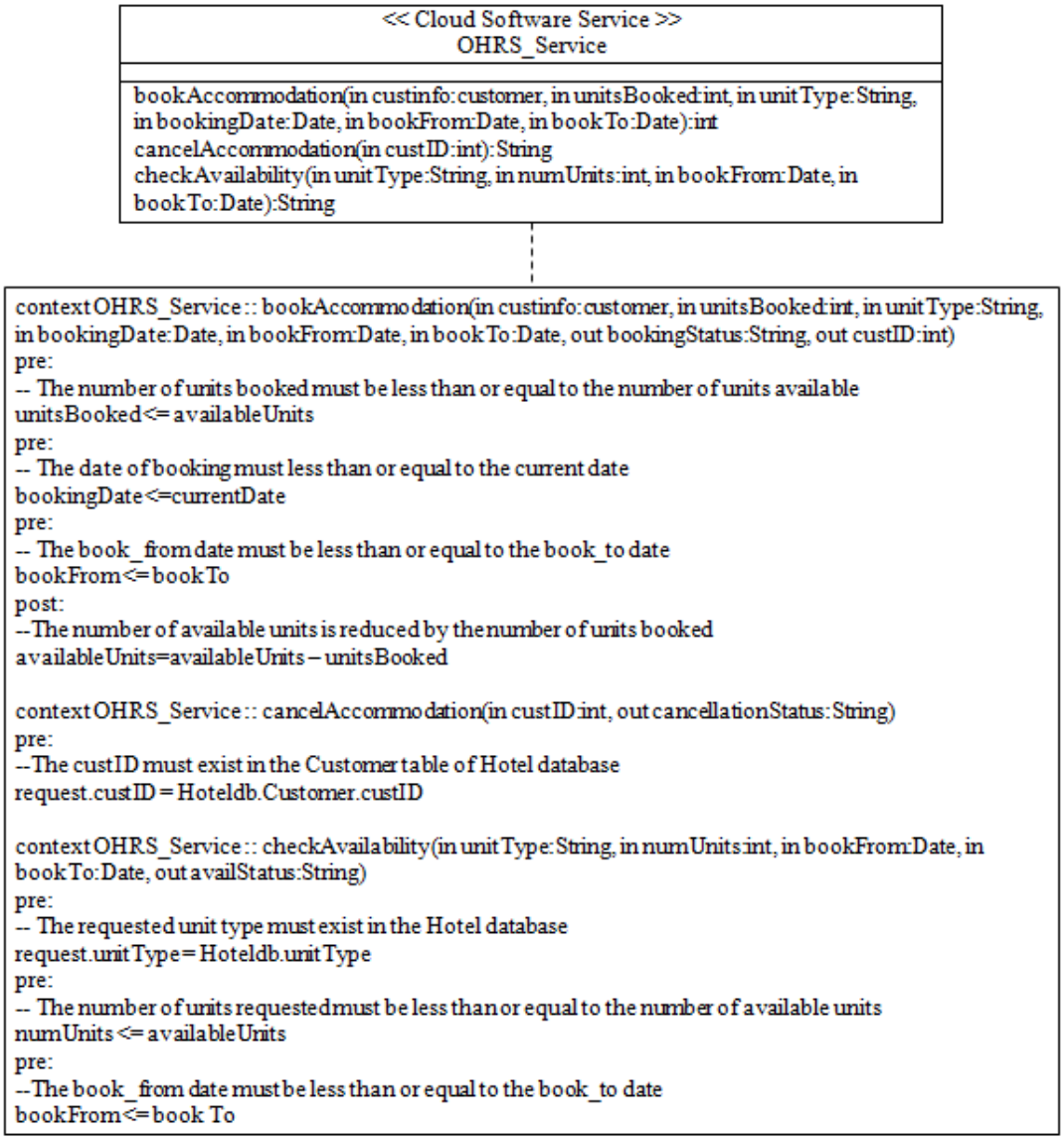

Figure 2 A formal model (PIM) of a business service

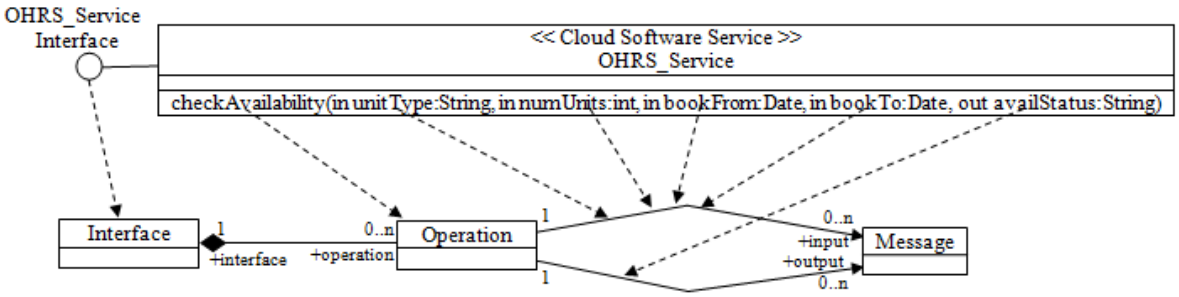

Figure 3 A Cloud Service PIM mapped to a WSDL PSM

As discussed earlier, the MDA approach requires that the models defined at different levels of abstraction during the process of development of the software system must conform to their respective metamodels. The platform-specific WSDL model must therefore be in conformance with the WSDL metamodel. A metamodel for WSDL2.0 is presented in the section 6.

6. WSDL AND WSDL METAMODEL

Web Service Description Language (WSDL), an XML-based specification, is the cornerstone of the Web service architecture, providing a common language for describing the services and a platform for integrating those services. These service descriptions enable communication between the loosely-coupled services in an SOA-based cloud. A WSDL specification describes four critical pieces of information: 1) interface information describing all the publicly available functions; 2) data type information for all the message requests and responses; 3) binding information about the messaging and transport protocol to be used; and 4) address information for locating the specified service [7]. A WSDL specification consists of distinct abstract and concrete definitions, which separates the abstract details of 'what' functionality is offered by the service from the concrete details of 'how' and 'where' this functionality is offered. Each of these definitions in turn uses a number of constructs to promote reusability of the description and to separate independent design concerns. This separation of definitions helps to preserve the integrity of the service description regardless of the changes in the underlying technology platform.

A metamodel in simple words is a 'model of a model'. It describes the abstract syntax of the set of elements of a model. A WSDL metamodel, thus, is an explicit description of the elements (constructs and rules) of a WSDL model.

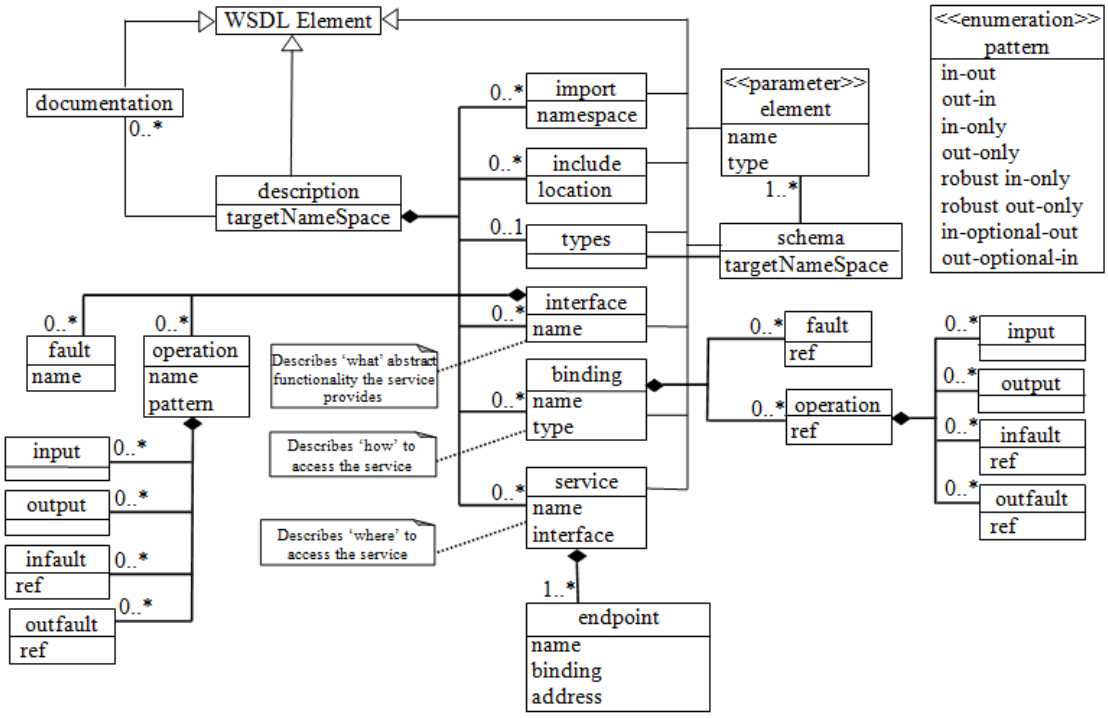

Figure 4 WSDL Metamodel

The `description` is the root element in a WSDL2.0 model and serves as a container for zero or more child elements in the following order [19, 20]:

1. zero or more `documentation` element
2. zero or more elements from among the following, in any order:
    i. zero or more `include` element
    ii. zero or more `import` element
3. an optional `types` element
4. zero or more elements from the following, in any order:
    i. `interface` element
    ii. `binding` element
    iii. `service` element

These six elements – `import`, `include`, `types`, `interface`, `binding` and `service` – are the top-level elements directly contained in the `description` root element. The top-level elements in turn may contain other elements called nested elements, for example the `operation` elements nested in the `interface` element or the `endpoint` element nested in the `service` element. Figure 4 depicts the WSDL metamodel specified using a UML class diagram.

The technology-independent abstract description includes the `types` and `interface` elements while the technology-specific concrete description includes the `binding` and `service` elements.

The `description` element holds the namespace declarations that would be used throughout the WSDL document. The `import` element is used to refer to the elements belonging to a different WSDL namespace while `include` element is used to refer to the elements in the same WSDL namespace. The `types` element uses the schema definition language such as XSD to describe the message types – `input`, `output` or `fault` – that the service would send and receive. The `interface` element contains `operation` elements to specify a set of operations offered by the service, each operation representing a single set of interaction between the client and the service. The `operation` element describes the `input`, `output` and `fault` messages, and also the `pattern` (message exchange pattern) between the communicating partners. The message exchange pattern may be one of the eight, enumerated in Figure 4. The `binding` element specifies the concrete details regarding message format and the transmission protocol for each `operation` and `fault` in the `interface`. The `service` element defines one or more `endpoint` elements to specify the locations where it can be accessed and a single `interface` which it supports. Each `endpoint` must reference a previously defined binding to indicate the protocols and transmission formats to be used at that endpoint.

7. MAPPING THE PIM (UML) TO PSM (WSDL)

Once the PIM of the cloud software service is specified, it can be transformed into a WSDL PSM based on the transformation definition specified for the purpose. An automated or a semi-automated transformation tool may be used to execute the transformation definition.

The complete transformation definition for transforming the cloud service PIM to the WSDL PSM cannot be described here due to spatial constraints. Therefore, only the key mappings of the transformation definition are listed below:

i. The interface of the service in the PIM maps to the interface element of WSDL PSM.
ii. The operations in the PIM map to the operation element in the WSDL PSM.
iii. The data types in the PIM map to the types element of WSDL PSM.
iv. Each input message in the PIM maps to the input element in the operation element of the WSDL PSM.
v. Each output message in the PIM maps to the output element in the operation element of the WSDL PSM.

8. CLOUD SAAS INTEROPERABILITY

Cloud computing architectures are a heterogeneous blend of technologies and platforms. The various software applications residing in the cloud do not exist in isolation. They must be able to communicate and exchange information transparently, irrespective of the technologies used to implement them. Thus, interoperability among the cloud SaaS is a relevant and significant issue in cloud computing.

The IEEE Glossary [21] defines interoperability as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged".

In early- and mid-90s, the sharing of data among the applications on same computer or different computers was facilitated through the then prevalent technologies such as COM/DCOM (Microsoft) and CORBA/IIOP (OMG), which allowed the units of functionality to be reused as binary objects. Sun created its own native ORB, called RMI (Remote Invocation Method). But these technologies were platform-specific. For example, COM/DCOM was tied to Windows platform and though CORBA/IIOP did not have this limitation, it was used with non-Java languages. Also, in the absence of a universal standard for data representation at that time, the communication among the applications was carried out in binary form; but this binary data was restricted by firewalls.

The year 1999-2001 witnessed the advent of Web services. Web-services are characterized by non-proprietary standards, cross-platform capabilities and human-readable messages that can be sent across firewalls. Since the web service message is in human-readable form, it requires more bytes to transmit the same amount of information as compared to that of its predecessors.

Web Service offers a suitable technology platform for realizing SOA. The services in SOA are inherently interoperable. This intrinsic interoperability builds on the principle of loose-coupling among the services which is achieved by virtue of vendor-neutral communications framework that enables implementation of highly standardized service descriptions and message structures. Loose-coupling results in the independence of service logic. Services only require being aware of each other; they can evolve independently.

Thus, the standard-based service descriptions are the key ingredients in establishing a consistently loosely coupled form of communication among the services implemented in the cloud. A service description document which describes the functionality offered by a service via its interface, accompanies each service. This document is specified in WSDL. WSDL is

not tied to any specific XML messaging system such as XML-RPC or SOAP, but it does include built-in extensions for describing SOAP services. Once a WSDL document (.wsdl file) is specified for a service, a SOAP client can be manually created to invoke any of the service's publicly available functions as defined in the WSDL document. Alternatively, the service can be automatically invoked using a WSDL invocation tool available for the purpose such as Web Service Invocation Framework (WSIF) from IBM, SOAP::Lite, GLUE from The Mind Electric etc. These tools may be command-line based or may provide a web-based interface.

In the context of MDA, the WSDL represents the PSM of the cloud SaaS. The WSDL in a cloud SaaS is responsible for message exchange and communication between the interacting services, thus ensuring interoperability at the PSM level in the model-driven approach.

In order to address the interoperability issue we extend the example further to include another cloud SaaS – the COHS (Cab_on_Hire System) Service. The service may be availed by any Cab Enterprise. A variety of business functionalities are offered by this cloud software service such as online bookings and/or cancellations of cabs by the customers, generating reports for the management etc. To illustrate, we consider two fictitious enterprises – ABCHotels and XYZCabs – that subscribe to these cloud services [15] and provide services to their customers through their respective websites – ABCHotels.com and XYZCabs.com. Also, the two enterprises complement each other in terms of the services they provide, i.e. the ABCHotels hires cabs from XYZCabs for its customers. Similarly, XYZCabs hires rooms for its customers from ABCHotels. This requires a B2B communication between the two enterprises. This interaction is facilitated by the cloud services through simple interfaces – an OHRS_Service interface for OHRS and a COHS_Service interface for COHS. Although, a service may maintain different interfaces for B2B and B2C interactions, for simplicity we are assuming a single interface to enable both. Let us, for example, consider that ABCHotels wishes to enquire about the total number of their customers who have hired cab(s) from XYZCabs on a particular day. This requires the ABCHotels to invoke a specific method through the Cab_Service interface. In an SOA-based cloud this interaction is enabled through an XML-based communication framework that uses SOAP messages. The HTTP request is submitted through the browser, and is routed via a controlling servlet on the OHRS web server (in the cloud). The web service client provided by the OHRS Service uses the web service interface published by COHS Service to invoke the method on its server that returns the required information. The method invocation is performed by creating an XML message that contains the method name and any required parameters and then sending it to COHS Service using the SOAP protocol. The value(s) returned by the method call are then wrapped in another XML message and sent back to the OHRS web client, which extracts the information that it needs and uses a server-side script engine to render it as HTML. The HTML is then returned to the client's browser. The advantage of using XML instead of HTML is that only raw data is required to be transferred which does not include presentation markups, thereby reducing network traffic. Also, the code required to make a request is much simpler than that required to extract data from an HTML page.

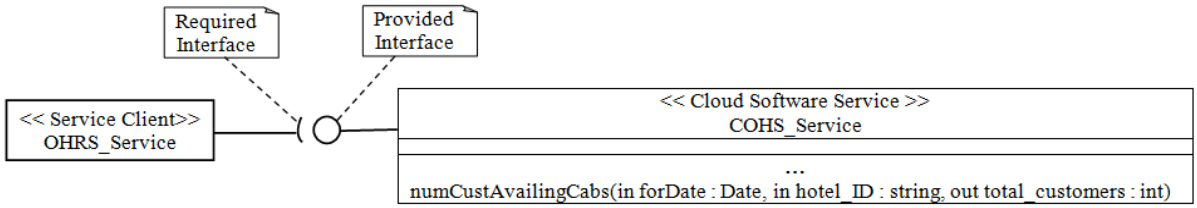

Figure 5 Interoperability between two services

9. CONCLUSION

The hardware and software technologies are evolving at a tremendous pace. The cloud computing is the latest technology in this timeline of evolution, where computing resources are made available as services over high bandwidth linkages. This technology is gradually changing the way the small and medium enterprises would look at their business solutions. The enterprises may opt for subscribing to these services, thereby preventing the upfront cost on the required hardware and software infrastructure. In addition, it will reduce the need to train, or hire skilled professionals for the job. But, the technology evolution has a negative aspect too. The rapid transitions may lead to the obsolescence of several legacy enterprise applications. The consequences are more serious when the application involves business-to-business interaction among enterprises.

In order to overcome this drawback, the authors in this paper present a model-driven approach to develop cloud software services, and exhibit interoperability among them. Based on MDA, the PIM and PSM of a cloud SaaS, taken as an example, are specified in UML. A transformation definition to transform the PIM of the cloud SaaS into its PSM is discussed briefly. The PSM is targeted on WSDL, which is an essential component of Web Services communications framework, in addition to SOAP and UDDI. It is the WSDL document of the cloud service which defines the interface of a cloud SaaS, and which is invoked by other software applications in the cloud. SOAP, a messaging and communication protocol, is used for the exchange of messages between the services. WSDL itself has the ability to bind to SOAP, simple HTTP or MIME. WSDL, which represents the PSM level of the MDA-based cloud SaaS, thus ascertains interoperability among the cloud SaaS. At present, the authors are working towards the completion of a transformation tool to transform the cloud SaaS PIM into its WSDL PSM.

10. REFERENCES

[1] Rimal, B.P., Choi, E., Lumb, I. 2009. A Taxonomy and Survey of Cloud Computing systems. In Proceedings of the Fifth International Joint Conference on INC, IMS and IDC, 44–51.

[2] Foster, I., Zhao, Y., Raicu, I., Lu, S. 2008. Cloud Computing and Grid Computing 360-Degree Compared. In IEEE Grid Computing Environments Workshop, 1–10.

[3] Youseff, L., Butrico, M., Da Silva, D. 2008. Toward a Unified Ontology of Cloud Computing. In Grid Computing Environments Workshop, 1–10. ISBN: 978-1-4244-2860-1, doi: 10.1109/GCE.2008.4738443

[4] Maggiani, R. 2009. Cloud computing is changing how we communicate. In IEEE International Professional Communication Conference, 1–4.

[5] Mell, P., Grance, T. 2009. The NIST Definition of Cloud Computing. Version 15 (July 10, 2009), http://thecloudtutorial.com/nistcloudcomputingdefinition.html

[6] Erl, T. 2005. Service Oriented Architecture: Concepts, Technology and Design. Pearson Education, Inc.

[7] Cerami, E. 2007 Web Services Essentials. Third Indian Reprint. O'Reilly Media, Inc. ISBN 10:81-7366-339-4

[8] Cloud Computing and SOA Convergence in Your Enterprise, http://searchsoa.techtarget.com/generic/0,295582,sid26_gci1375000_mem1,00.html

[9] OMG Model Driven Architecture. http://www.omg.org/mda/

[10] Miller, J., Mukerji, J.: MDA Guide Version 1.0.1, http://www.omg.org/docs/omg/03-06-01.pdf

[11] Kleppe, A., Warmer, J., Bast, W. 2003 MDA Explained: The Model Driven Architecture: Practice and Promise. Pearson Education, Inc., London.

[12] OMG's Meta Object Facility http://www.omg.org/mof/

[13] Ritu Sharma and Manu Sood, 2011. Cloud SaaS and Model Driven Architecture. In Proceedings of the International Conference on Advanced Computing and Communication Technologies (ACCT11), 18-22. ISBN: 978-981-08-7932-7

[14] Sharma, R., Sood, M. 2011. Modeling Cloud Software-as-a-Service: A Perspective. In Proceedings of the International Conference on Network Communication and Computer (ICNCC 2011), 170–174. ISBN: 978-1-4244-9550-4

[15] Ritu Sharma, Manu Sood and Divya Sharma, 2011. Modeling Cloud SaaS with SOA and MDA. In Proceedings of the International Conference on Advances in Computing and Communications (ACC 2011). Communications in Computer and Information Science, 2011, Volume 190, Part 5, 511-518, DOI: 10.1007/978-3-642-22709-7_50

[16] Ritu Sharma and Manu Sood, 2011. Cloud SaaS: Models and Transformation. In Proceedings of the First International Conference on Computer Science, Engineering and Information Technology (CCSEIT-2011). Communications in Computer and Information Science, 2011, Volume 205, 305-314.

[17] Frankel, D.S. 2003. Model Driven Architecture: Applying MDA to Enterprise Computing. Wiley Publishing Inc., Chichester.

[18] Frankel, D., Parodi, J. 2002. Using MDA to develop Web Services, 2nd edn. IONA Technologies PLC

[19] Web Services Description Language (WSDL) Version 2.0 Part 0: Primer W3C Recommendation 26 June 2007. http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626

[20] Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language W3C Recommendation 26 June 2007. http://www.w3.org/TR/2007/REC-wsdl20-20070626

[21] Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.