Modeling Cloud Software-As-A-Service: A Perspective

Ritu Sharma and Manu Sood

Abstract—**Cloud computing, an emerging computing paradigm, is still in its stage of infancy. It can be considered as an amalgam of a variety of legacy and evolving technologies and concepts such as distributed computing, grid computing, virtualization, service-oriented architecture, software-as-a-service etc to mention a few. It aims at optimally utilizing the distributed, dynamically-scalable and often virtualized resources such as processors, storage and software, which are made available to the users as services over a network, on a pay-per-use basis. These services can be easily accessed by the user through an interface, as simple as a browser. But as the technologies are in a state of flux, developing the software services in the cloud directly using existing technologies will make them obsolete as and when new technologies evolve. Model-driven Architecture (MDA), an initiative by Object Management Group (OMG), is an open, vendor-neutral approach that supports the complete software development process with the modeling activity. In this paper, the authors propose to leverage Model-Driven Architecture to specify the software services in the cloud using platform-independent model (PIM), from which one or more platform-specific models (PSM) can be derived using automated transformation tools. The software services thus yielded will be more robust, flexible and agile in the wake of changing technologies.**

Index Terms—**Cloud computing, cloud SAAS, model-driven architecture (MDA), platform independent model (PIM), platform specific model (PSM).**

I. Introduction

Cloud computing is an emerging computing model that came into existence at the onset of this century with the advent of Amazon's Web based services. Some other big vendors who gradually joined the fray include Yahoo, Google, Microsoft, IBM, Sun, Intel, Oracle and Adobe. A cloud, in the given context, refers to a complex, internet-based infrastructure of hardware and software components which include processors, storage, applications, development environments etc., and are available remotely as a service. Accordingly, the cloud services are offered at various levels – software, platform, infrastructure and hardware. The Internet serves as a medium to transmit data and/or information between the user and the cloud. The user may gain access to the service from anywhere and at anytime, using a thin client or a laptop or even a mobile phone, through an interface. (usually the browser), on an on-demand pay-per-use basis.

The resources in a cloud are dynamically allocated and reallocated on a sharing basis and hence are optimally utilized and at the same time are very economical.

With the rapid advancements in the field of Information and Communications Technology (ICT), there has been a growing significance of model-based software development solutions. Model Driven architecture (MDA), a recent model-driven approach in software development offers the advantage of developing a platform-independent model (PIM) of the system and transforming it into any number of platform-specific models (PSMs), one for each platform or technology in which the final system would be deployed, and which eventually can be transformed into code. The transformations are performed using transformation tools developed for the purpose.

The MDA approach may be leveraged in developing the software applications that would run in the cloud as software services. The PIM of the cloud application would reflect the structure, behaviour and functionality of the application irrespective of the technology used for its implementation. The PSM would be more implementation-oriented and bound to a given execution platform. The transformations from the PIM to PSM would be carried out using (semi)automated transformation tools.

This paper is an attempt to model the convergence of software development principles of MDA with implementation of cloud software services. Section II highlights the fundamentals of cloud computing and its enabling technologies. Section III briefly discusses the basic concepts of MDA. Based on MDA approach, section IV illustrates the computation-independent model, platform-independent model and platform-specific model of a cloud software application taken as example. Section V draws the conclusion of the paper and the future work being carried out by the authors.

II. Cloud Computing and Enabling Technologies

Cloud computing owes its evolution to a number of relevant legacy technologies and concepts such as – distributed computing and storage, grid computing, Service Oriented Architecture (SOA), virtualization, Software-as-a-Service (SaaS), Web services and supporting technologies mainly SOAP (Simple Object Access Protocol), WSDL (Web Service Definition Language) and UDDI (Universal Description Discovery and Integration).

The massive computing capabilities of the clouds are powered by computer grids (or grid computing environments) having coordinated resources and open standard protocols and frameworks. Cloud computing differs from grid computing in the sense that a cloud is managed by a computational grid but the reverse is not true [1]. The grids do not rely on virtualization as much as clouds do. Also, the grids are more suitable for batch computing while clouds are suitable for interactive

Manuscript received January 25, 2012; revised March 5, 2012.

R. Sharma is a Research Scholar at the Department of Computer Science, Himachal Pradesh University, Summer Hill, Shimla, India. (email: rituchetan@gmail.com)

M. Sood is an Associate Professor at the Department of Computer Science, Himachal Pradesh University, Summer Hill, Shimla, Himachal Pradesh, India. (email: soodm_67@yahoo.com)

computing [2].

Virtualization technology enables abstraction of logical resources from the underlying physical resources. In cloud computing this technology facilitates the dynamic allocation and reallocation of resources from a shared pool, to cloud service consumers. In order to meet a sudden increase in demand the existing resources are diverted from a low-priority application to a high-priority one [3]. This, in turn, helps to minimize the need for additional investment in infrastructure. Furthermore, it enables the cloud to give an illusion of the existence of infinite resources to its customers accessing the services.

Service Oriented Architecture (SOA) is an architectural paradigm wherein the services (distinct units of logic) available in a network such as the web, are used as the core components in building the logic of software applications. Each *service* is characterized by a well-defined interface which exposes the functionality provided by the service. Individually, the services may be distributed. The key principles that govern individual services in SOA are abstraction, loose-coupling, reusability, composability, statelessness, autonomy, discoverability and adherence to a service contract [4]. Cloud based systems, in order to be more effective and efficient, must be built on sound SOA principles. The cloud should not be looked at as a new architecture but instead as another option of storing and running services within SOA [5].

Software-as-a-Service (SaaS) is a software application based on multi-tenant model. It is owned and managed remotely by one or more service provider(s) and delivered to the customers over a network on a shared, pay-per-use or subscription basis. A SaaS application must be elastic and scalable in order to qualify as a true cloud computing service.

Having evolved from other similar computing paradigms such as SaaS, SOA and Web services, the cloud software services are simple, scalable, elastic, autonomous, readily available, easily accessible and service-oriented. Cloud computing aims at optimizing the usage of distributed and virtualized resources to achieve higher throughput and to tackle large scale computation problems. The services offered in the cloud are configured dynamically and the usage is metered. The service interfaces are location-independent and user-centric. The service level agreement (SLA) between the service provider and the service consumer ensures Quality of Service (QoS).

Although cloud computing has been extensively defined by various authors and researchers, a comprehensive definition is provided by US National Institute of Standards and Technology (NIST). The essential characteristics of cloud computing are – 1) On-demand self-service, 2) Broad network access, 3) Location independent resource pooling, 4) Rapid elasticity, and 5) Measured Service [6].

A cloud service may be delivered by a *private* cloud, a *public* cloud or a *hybrid* cloud depending on whether the resources are provisioned and managed internally within the organization, or externally by a third party on a shared basis, or both i.e. some resources internally while others externally.

Cloud computing is an umbrella term that encapsulates various services such as – Software-as-a Service (SaaS), Platform-as-a-Service (PaaS), Infrastructure-as-a Service (IaaS), Hardware-as-a-Service (HaaS). In the absence of a standard taxonomy, other suggested categories include Development, Database and Desktop as a Service (DaaS), Business as a Service (BaaS), Framework as a Service (FaaS), Organization as a Service (OaaS) etc [2][6][7].

Some key benefits of cloud computing are:

- The ability to access the resources in the cloud from anywhere and at any time.
- It minimizes the need for advanced hardware and software on client side thus eliminating the need for additional physical space and also minimizing the capital expense.
- Clients can achieve supercomputing powers from minimal resources.
- It eliminates the cost incurred on hiring trained professionals or upgrading the skills of manpower managing and maintaining the related infrastructure.
- It eliminates complexities of buying, configuring and managing the resources needed to build and deploy applications which in turn are delivered as a service.
-

III. Model-Driven Architecture (MDA)

MDA®, an initiative by OMG®, is an open, vendor neutral approach [8] to enterprise application development. It shifts the focus of software development from the solution domain to problem domain thereby bridging the gap which exists between domain-specific concepts and the programming technologies used to implement them; and enhances the rigor, productivity and manageability of software development process.

The software development process, in MDA, is driven by the activity of modeling the software system. The models are the prime artifacts and are used throughout the process of software development. During the development process these formal, domain-specific models are transformed into implementation code using automated transformation tools.

The models in MDA are defined at three levels of abstraction – 1) Computation Independent Model (CIM) or the domain model bridges the gap between the domain experts and system experts. It specifies the business logic of the application, which remains the same irrespective of whether the system is ICT based or otherwise. 2) Platform Independent Model (PIM) specifies the functionality of the ICT-based system independent of the platform that would be used for its implementation. 3) Platform Specific Model (PSM) describes the system with respect to the specific platform on which it would finally be implemented. Miller et al in [9] define a platform as "a set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns which any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented".

A complete MDA specification consists of a definitive platform-independent model (PIM), plus one or more platform-specific models (PSM), sets of interface definitions, each describing how the base model is implemented on a different middleware platform and sets of

transformation definitions. The PIM depicting the structure, behaviour and functionality is modeled only once. And then, the transformation definitions enable the transformation of the PIM to one or more PSMs.

The key to the success of MDA lies in automating the model-to-model and model-to-code transformations. A transformation is defined as a process of the automatic generation of a target model from a source model, according to a transformation definition. A transformation definition is a set of transformation rules that together 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 [10].

As this approach focuses primarily on the functionality and behaviour of the software system, and not the technology in which it will be implemented, it is not necessary to repeat the process of modeling system's functionality and behaviour each time a new technology comes along.

The three primary goals of MDA are interoperability, reusability and portability through architectural separation of concerns [9].

The key benefits of MDA include – improved quality of software design and implementation, reusability, enhanced productivity, portability, interoperability, improved Return on Investment (ROI), reduced cost and reduced time to market.

IV. A Cloud SaaS – CIM, PIM and PSM

Cloud SaaS are web-based software applications that run remotely on the Internet and are provided as services to the consumers on an on-demand pay-per-use basis. These applications in the cloud may be as simple as a time zone converter performing a single discrete function, or as complex as a Customer Relationship Management (CRM) system performing a set of related business functions. As is evident, the technologies are constantly evolving. Rather than directly developing these software services using available technologies, modeling them at a higher level of abstraction will decouple them from the undesired effects of technology change and enhance their longevity.

Therefore, the MDA-based development of cloud SaaS (application) is going to play a significant role in improving the quality of cloud software services, making them more robust, flexible and agile. MDA will enable defining these cloud software services in a technology-independent manner. Encapsulating business logic in a manner that is independent of the technical mechanisms will formally capture the essence of the applications; and will also make it possible to reuse them in a variety of contexts [11].

Keeping this in mind, the authors propose leveraging MDA to develop Platform-Independent Model (PIM) of a cloud SaaS (a software application in the cloud) and specify its functionality, structure and behaviour independent of the platform used for its implementation. Since the PIM would be less tied to underlying technologies, it needs to be modeled only once. From this PIM, one or more technology specific artifacts, the Platform-Specific Models (PSMs), could be derived (semi)automatically using open or proprietary model-to-model transformation tools. The PIM and PSM would represent the logic of the application, independent of the implementation technology used to expose it as a web service, such as WSDL, Hyper Text Markup Language (HTML), Wireless Access Protocol (WAP), etc. Eventually, the PSMs may be used for automated code generation for specific technology platforms.

We illustrate our approach using an example of a software application running as a service in the cloud – the Credit Card Validation System (CCVS). The CCVS may be accessed by anyone connected to the Internet, through an interface as simple as a browser. For instance, a customer may purchase products or services from an online/offline store and make payment through his credit card. In this situation, the CCVS may be accessed for validating the card.

We assume a simplified approach to credit card validation where the various steps in the business process may be listed as:

- The consumer purchases goods or services from a merchant (online/offline) through his credit card.
- In case of online shopping, the credit card details are submitted at the customer's browser, from where they are securely transmitted to the merchant site and finally to the CCVS. In case of an offline merchant store, the credit card details are submitted to CCVS through the Electronic Funds Transfer Point of Sale (EFTPOS) terminal by swiping the card.
- Once received by the CCVS, the credit card information and the bill amount is sent to the Clearing House.
- The Clearing House then submits the transaction to the bank that issued the card.
- The bank verifies the card information and transaction amount, and sends a message to the Clearing House, approving or declining the transaction.
- The Clearing House in turn sends the message (approve/decline) to the merchant.
- In case the transaction is approved, the bill amount is reimbursed to the merchant by the Clearing House.
- The bank pays the amount to the Clearing House later on.
- The bank deducts the transaction amount from the credit available to the customer.
- The bank, in turn, receives the payment from the cardholder at a later date.
- Any processing fee charged by the Clearing House or the card issuing bank is recorded on the card statement as expense on part of the customer.

Although MDA does not restrict itself to Unified Modeling Language (UML) for modeling the system, we are using UML for the purpose of illustration.

A computation-independent model (CIM) of a software system may be represented using Use Case diagrams in UML. A use case diagram capturing the functionality of the system under consideration is depicted in Fig. 1.

The characteristics of the actors in the system are:

- Customer is a person who uses the credit card to make payments for purchases.

- Merchant is a person who receives the payment from the customer for the goods or services sold.
- Clearing House is a firm that has contract with the merchant's bank to clear charges in exchange for a flat fee and a percentage of every charge processed.
- Bank is the bank that has issued the credit card to the customer.

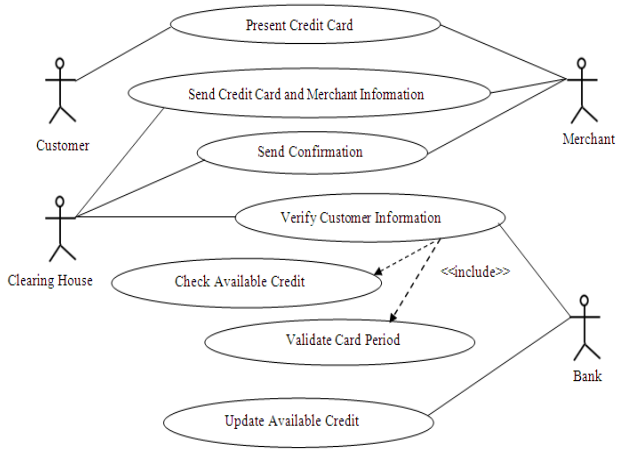

Fig. 1. Use Case diagram for CCVS.

As mentioned earlier, the cloud application is web-based; the PIM design vocabulary describes the attributes and operations in ways that are entirely independent of XML, WSDL, SOAP, UDDI, Java, and other Web Service implementation technologies. Transformation tools can then be used to generate XML, WSDL, SOAP, UDDI, and the technology-specific artifacts and finally the implementation code from the design input [12].

A PIM specifies the system at a higher level of abstraction as compared to a PSM. Since, it is independent of any technology to be used for implementing the system, it can therefore be reused across several different platforms.

The class diagram for a Platform-Independent Model (PIM) of the application under consideration is depicted in Fig.2. The PIM, here, defines the static aspects of the application through a static view. It shows the various classes (entities) and the associations among them.

A multiplicity adorns each association. The relevant attributes and the operations are also listed in the model. The *accessor* and *mutator* operations have been intentionally excluded from the diagram in order to keep it simple.

The various classes included in the PIM, for the CCVS, are – Customer, CreditCard, Merchant, Bank, ClearingHouse, and Product. The associations among the classes are defined as under:

- A bank is composite aggregation of customers. A bank has one or more customers. A customer account is unique and can belong to only one Bank.
- A customer may have more than one account in a bank.
- Also, a customer can have different accounts in different banks.
- A bank may issue one or more credit cards, but a credit card must belong to only one Bank.
- A customer may have one or more credit cards, but a credit card belongs to only one customer.
- A customer may purchase from one or more merchants. Also, a merchant may sell products to one or more customers.
- A merchant may deal with one or more clearing houses. A clearing house will deal with one or more merchants.
- A bank may deal with one or more clearing houses. A clearing house may deal with one or more banks.
- A customer may purchase one or more products. A product may be purchased by one or more customers.
- A product can be purchased on one credit card. A credit card can be used to purchase more than one product.
- A clearing house accepts more than one credit card. A credit card is presented at only one clearing house.
-

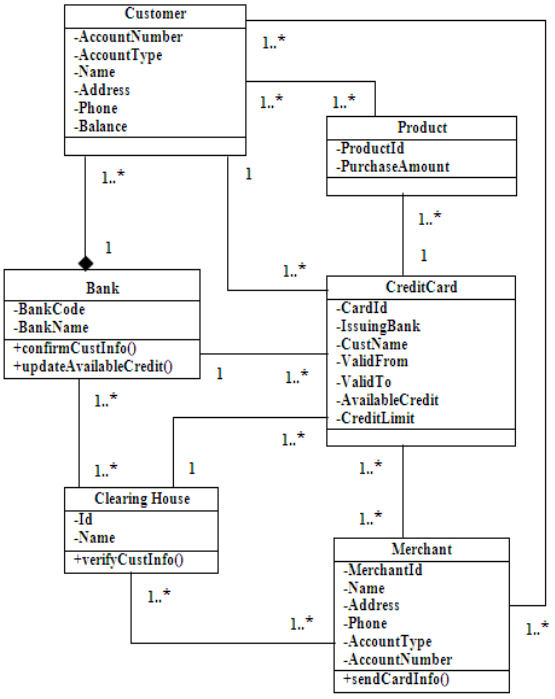

Fig. 2. PIM with associations

For the sake of simplicity, the following assumptions have been made by the authors, with regard to the system under consideration:

- An account in a bank can be owned by only one person.
- A credit card can be processed by only one clearing house.
- A customer cannot own a credit card that is not issued by a bank.

The PSM is derived from the PIM using automated transformation tools. It contains the technology specific details for the target platform. The Platform-Specific Model (PSM) for the application under consideration is depicted in Figure 3 using a class diagram. It is targeted specifically on Java platform. The PSM also depicts the classes, the associations among the classes and the multiplicities of the associations in a manner similar to PIM. The two models differ in the sense that the PIM does not mention the language-specific (technology-specific) details of the system such as the data type of the attributes, the list of

parameters with their data types to be passed to the function, or the return type of the data that would be returned by the function. In contrast to PIM, the PSM includes these details as is clear from Fig. 3.

As discussed earlier, a number of PSMs targeted on different platforms such as CORBA, .NET, J2EE, etc. can be derived from a single PIM. Since these models differ in their structure, each having its own platform-specific constructs, different sets of transformation rules is required to generate different PSMs. Once the PSM has been generated, the next step is to generate the implementation code from it. Finally, the software (code) obtained is deployed in specific environments.

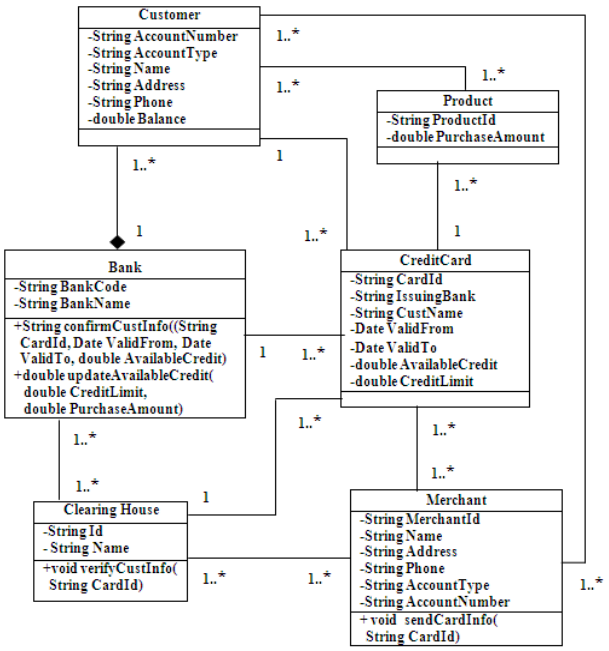

Fig. 3. PSM targeted on java.

V. Conclusion and Future Work

Cloud computing is leading into a future where the enterprise applications would run on centralized facilities managed and controlled by third-party compute and storage resources, instead of local computers. Besides, the technologies are evolving at a tremendous pace. With the emergence of newer technologies, the older ones may become obsolete and/or get replaced. Consequently, a technology-specific development of these software applications running in the cloud is not viable in the long run. The illustration of the PIM and PSM of a cloud SaaS taken as example in this paper reinforces that developing software applications in the cloud in a manner that is independent of the specific technologies, using MDA, will enable to reap the benefits of MDA based software development. Above all, it will enhance the rigor, longevity and reusability of the cloud service developed thus.

The authors are presently developing a transformation tool that would transform the CCVS cloud SaaS PIM into its PSM targeted on Java platform. A set of transformation rules has been defined by the authors to transform the PIM constructs into their respective PSM constructs. In addition, an SOA-based approach for the development of the illustrated cloud application is also underway.

References

[1] F. Maria Aymerich, G. Fenu, and S. Surcis. "An Approach to a Cloud Computing Network." *In First International Conference on the Applications of Digital Information and Web Technologies*, 2008, ICADIWT.2008 pages 113–118, August 2008, ISBN: 978-1-4244-2623-2, doi: 10.1109/ICADIWT.2008.4664329 .2008

[2] I. Foster, Y. Zhao, I. Raicu, and S. Lu. "Cloud Computing and Grid Computing 360-Degree Compared." In *IEEE Grid Computing Environments Workshop*, 2008. GCE '08 (Nov 2008), pages 1–10, November 2008.

[3] R. Mikkilineni and V. Sarathy, "Cloud Computing and the Lessons from the Past," in *Proc*, *18th IEEE International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises*, 2009, Pages:57-62 , ISBN ~ ISSN:1524-4547 , 978-0-7695-3683-5

[4] T. Erl, "Service Oriented Architecture: Concepts, Technology and Design," ISBN 978-81-317-1490-4, *Pearson Education*, Inc. 2005.

[5] "Cloud Computing and SOA Convergence in Your Enterprise," SearchSOA.com, 20 Nov 2009 [Online] Available: http://searchsoa.techtarget.com/generic/0,295582,sid26_gci1375000_mem1,00.html .2009.

[6] P. Mell and T. Grance, "The NIST Definition of Cloud Computing", Version 15, 10-7-09, [Online] Available: http://thecloudtutorial.com/nistcloudcomputingdefinition.html

[7] B. P. Rimal, E. Choi, and I. Lumb, "A Taxonomy and Survey of Cloud Computing systems", In *Proc. Proceedings of the 2009 Fifth International Joint Conference on INC*, IMS and IDC, 2009, Pages: 44-51.

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

[9] J. Miller and J. Mukerji, "MDA Guide Version 1.0.1" [Online] Available: http://www.omg.org/docs/omg/03-06-01.pdf

[10] A. Kleppe, J. Warmer, and W. Bast, "MDA Explained: The Model Driven Architecture: Practice and Promise," 2003, *Addison-Wesley Longman Publishing Co*. Inc

[11] D. Frankel and J. Parodi, "Using MDA to develop Web Services," IONA Technologies PLC, Second Edition, April, 2002 [Online] Available: http://www.iona.com/archwebservice/WSMDA.pdf

[12] S. David Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing"* 1st ed., Wiley Publishing Inc, 2003.