Cloud SaaS and Model Driven Architecture

Ritu Sharma\(^1\) and Manu Sood\(^2\)
Department of Computer Science
Himachal Pradesh University
Shimla, India
\(^1\)rituchetan@gmail.com, \(^2\)soodm_67@yahoo.com

Abstract—Cloud computing is an emerging computing paradigm in which the resources such as processors, storage and software applications are provided as services, remotely over the Internet. The software applications in the cloud may be targeted on different platforms. As is evident, the software technologies are evolving constantly. But, as newer software technologies emerge the existing ones become obsolete and need to be replaced. Therefore, it becomes obligatory to adopt a software development methodology where the impact of technological advancements on software applications is minimal. Model Driven Architecture (MDA) is a software development approach where the models are the prime development artifacts. These models abstract the business functionality of the software system from its implementation on specific platforms. Automated tools are used for model-to-model and model-to-code transformations.

Keywords—Cloud computing, Cloud SaaS, Model-Driven Architecture (MDA), Platform-independent Model (PIM), Platform-specific Model (PSM)

I. INTRODUCTION

Cloud computing is the most recent computing model in the field of Information and Communication Technology, where the computational resources – hardware, software, development environment and other infrastructure – are made available as services, remotely over a network (intranet or Internet). These services can be easily accessed using a thin client or even a mobile phone through an interface, as simple as a browser, on an on-demand pay-per-use basis. The resources in the cloud are dynamically scalable and often virtualized.

The rapid advancements witnessed in the field of ICT require development of software solutions in a manner that is independent of technology change. Model Driven Architecture (MDA) is a model-driven approach to software development where the models are used as the prime development artifacts. These models are formal in nature and can be machine processed. The transformation of models from one level of abstraction to another, or the transformation of models to executable code are performed by using (semi)automated transformation tools.

The MDA approach may be leveraged to develop the software applications that would be deployed in the cloud, the cloud Software-as-a-Service (SaaS). A platform-independent model (PIM) of the cloud application would reflect its structure, behavior and functionality irrespective of the technology used for its implementation. The platform-specific model (PSM) of the application, on the other hand, would be more implementation-oriented and bound to a given execution platform. The transformations from the PIM to PSM would be carried out using transformation tools developed for the purpose.

This paper attempts to incorporate the model-driven software development methodology in the development of cloud SaaS. Section II discusses the cloud computing and its evolution from relevant contemporary technologies. Section III briefly discusses the concept of MDA. Section IV illustrates the platform-independent and platform-specific models of a cloud software application with the help of an example. Section V draws a conclusion of the paper and the future work undertaken by the authors.

II. CLOUD COMPUTING AND ITS EVOLUTION

The term cloud was first used in early ‘90s to refer to large ATM networks. Cloud computing began in earnest at the beginning of this century with the advent of Amazon’s Web Services [1]. The other big vendors who gradually joined the fray include Yahoo, Google, Microsoft, IBM, Sun, Intel, Oracle and Adobe. The first to give prominence to the term ‘cloud computing’ was Google’s CEO Eric Schmidt, in late 2006 [2].

The cloud computing has evolved from a range of relevant legacy technologies and concepts such as grid computing, virtualization, Web Services and 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).

The grid computing technology matured into a well-established technology in the early 2000s. Grid computing enables efficient management of geographically distributed computational resources that support different hardware and software configurations. The large computing and storage capacity offered by grid technology led to the development of another category of services that later came to be identified as cloud services [2]. Virtualization technology allows for dynamic reallocation of resources. A sudden increase in demand of resources is met by diverting existing resources from a low-priority to a high-priority application [3]; thus, giving an illusion of the existence of infinite resources to the cloud customers. A web service is any service that is available over a network, uses a standardized XML messaging system, is...
not tied to any operating system or programming language, is self-describing and discoverable [4]. Service Oriented Architecture is an architectural style in which services are utilized. Service Oriented Architecture is an architectural style in which services can be distributed and based on the key principles of loose-coupling, autonomy, abstraction, reusability, composability, statelessness, discoverability and adherence to a service contract [5]. Cloud based systems built on sound SOA principles are more effective and efficient. The cloud should not be looked at as a new architecture but instead as another option of storing and running services within SOA [6].

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.

Cloud computing is an umbrella term that covers 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. A cloud SaaS is a multi-tenant platform that uses common resources and a single instance of both – the object code of an application as well as the underlying database – to support multiple customers simultaneously. It is based on the Application Service Provider (ASP) model and heralds a new wave in application software distribution. The key providers of this service are SalesForce.com, NetSuite, Oracle, IBM, Microsoft etc [7, 8].

Cloud computing has changed the way the enterprises – especially the small and medium enterprises – look at their business solutions. Instead of investing their capital in purchasing the traditional, stand alone on-premises hardware, software and other infrastructure and hiring trained professionals for the job, the cloud services available over the Internet, are utilized.

III. MODEL-DRIVEN ARCHITECTURE (MDA)

MDA®, an initiative by OMG®, is an open, vendor neutral approach to enterprise application development. It shifts the focus of software development from the solution domain to the 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 [9]. The entire software development process is model-driven with models as the primary artifacts for understanding, design, construction, deployment, operation, maintenance and modification of a system [10].

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 [10] define a platform as “a set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented”.

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 [11].

IV. MDA BASED DEVELOPMENT OF A CLOUD SAAS

A Cloud Software-as-a-Service (SaaS) is a capability provided to the consumer to use the provider’s applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings [12]. The applications in the cloud may be as simple as a time zone converter performing a single discrete function, or as complex as a holiday packaging system performing a set of related business functions.

As is evident, the technologies are constantly evolving. Rather then directly developing these cloud 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. 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. 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 [13].

We illustrate our approach using an example of a software application running as a service in the cloud – the Credit Card Verification System (CCVS). The CCVS may be accessed by anyone connected to the Internet, through a web browser interface. 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 would be accessed from the merchant site for validating the card.
We assume a simplified approach to credit card verification 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 the information, approving or declining the transaction, to the Clearing House.
- The Clearing House in turn forwards this information 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, finally, 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.

The CCVS application thus needs to interact with other entities which are external to the system – the merchant, the bank (the bank that issues the card) and the Clearing House, in order to accomplish the task.

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.

In general, the payment through Credit Cards involves two main processes – 1) Credit Card validation, and 2) Settlement of payment. For the sake of simplicity, we are only focusing on the first process, the validation of the credit card.

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 use case diagram capturing the functionality of the system is depicted in Figure 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.

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

A platform-independent model (PIM) defining the static aspects of the CCVS application through a static view, is depicted in Figure 2. The class diagram shows the various classes (entities) in the system and the associations among them. A multiplicity adorns each association. The relevant attributes and the operations are also listed in the model. The get and set operations have been intentionally excluded from the diagram in order to keep it simple.

The various classes in the class diagram used for specifying the PIM for CCVS are – Customer, CreditCard, Merchant, Bank, and ClearingHouse. 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 clearing house accepts more than one credit card. A credit card is presented at only one clearing house.

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 [14].

The PSM can be 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 Figure 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.

V. CONCLUSION AND FUTURE WORK

Cloud computing hints at a future where the computations would be performed on centralized facilities operated by third-party compute and storage resources, instead of local computers. A technology-specific software development is not viable in the long run, since with the emergence of newer technologies, the older ones may become obsolete and/or may get replaced. The illustration of the PIM and PSM 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.

At present the authors are working on an SOA-based approach for the development of the illustrated cloud application. The efforts are being made to ensure interoperability among the PIMs and the PSMs of the services identified in the SOA for the application.
REFERENCES


