International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 1, February 2020, pp. 962~977
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i1.pp962-977

A service-oriented cloud modeling method and process

Chee-Yang Song[1], Eun-Sook Cho[2]
[1]Department of Software, Kyungpook National University, Republic of Korea
[2]Department of Software Engineering, Seoil University, Republic of Korea

Article Info

Article history:

Received Aug 26, 2019
Revised Oct 4, 2019
Accepted Oct 14, 2019

Keywords:

Cloud metamodel
Cloud modeling process
Cloud service
Development method
MDA

ABSTRACT

The transition of software development from web to cloud has been accelerated. The development of cloud services requires a modeling method that reflects the characteristics of cloud including personalized service, resource sharing service, grouped and distributed services, and cross-platform operability. This study aimed to suggest a method of developing UML-based cloud services suitable for the characteristics of cloud services. A cloud service metamodel was defined using cloud applications' characteristic modeling elements, and after that, how these cloud modeling elements are expressed into UML modeling elements was defined with an integrated metamodel between cloud and UML. By applying this hierarchical cloud metamodel, an MDA and MVC-based service-oriented cloud modeling process was established. By doing so, it will be possible to easily design services (applications) and solutions that are suitable for cloud computing environments, and in particular, to create hierarchical reuse models by the level of the abstraction of model-driven development.

Corresponding Author:

Chee-Yang Song,
Department of Software,
Kyungpook National University,
2559, Kyeongsang Dae-ro, Sangju-Si, Gyeongsang Buk-Do 742-711, Republic of Korea.
Email: cysong@knu.ac.kr

1. INTRODUCTION

Computing environment has evolved from web-based software to cloud computing-based service, which is a paradigm that can provide IT resources in the form of service through a network regardless of location or equipment [1]. Cloud services can be characterized by customization (tenant, self) [2], multiple media, shared utilization of resources, distributed processing, group access management, on-demand measured service, network-based or web or program-based interface control, etc. Since various communication media can be used, cloud services can be accessed anytime and anywhere, and can be freely purchased, and the costs can be significantly reduced by sharing resources. E-mail and social networking service (SNS) are personalized cloud services, and Google's AppEngine, Amazon's Elastic Compute Cloud (EC2), IBM's Blue Cloud and Microsoft's Azure are among the most representative commercial cloud services. In addition, it is critical to maintain security in cloud services [3, 4] suggested a guide for developing cloud services based on privacy (personal or enterprise information) in order to prevent any leakage of users' sensitive information and personal information. To establish methods of developing cloud services that meet these characteristics of computing, several studies have been conducted on approaches that expand cloud services based on architecture-oriented unified modeling language (UML) or feature models. For instance, the characteristics of the distributed processing of cloud services can be expressed by expanding a UML deployment model. Other efforts have also been made to port existing applications to cloud services.

Meanwhile, as interest in software reuse technology increased in the 2000s, patterns of model driven architecture (MDA) according to the level of abstraction in development phase and model view controller (MVC) based on separation of concern was appeared. MDA signifies an architecture pattern for generating the conceptual independent model (CIM), platform independent model (PIM), and platform specific model (PSM), which separates the model depending on target domain and implementing environment. MVC is

a design pattern for relating the user interface, function process and data structure separately. Using both MDA and MVC patterns may improve reusability of model by creating a modular, independent model.

This study aimed to suggest a service-oriented cloud metamodel and modeling process for developing applications that reflect the characteristics of cloud, and support hierarchical modeling using MDA and MVC from early cloud requirements to architecture design to deployment. An MDA approach defines development phases and task activities with CIM, PIM and PSM depending on the level of the abstraction of development. First, a metamodel is defined to express the characteristics of cloud well. Based on the metamodel, a cloud development process is established [5]. This study suggests a service-oriented development methodology based on the characteristics of cloud services, and a hierarchical development process using an MDA approach based on the modeling elements of the developed metamodel.

The rest of this paper is organized as follows. Section 2 analyzes the MDA based software and cloud development methods from relevant studies. Section 3 deals with the service-oriented cloud metamodel. Section 4 describes a design method for the service-oriented modeling framework and process using these metamodels. Section 5 applies the proposed modeling process to the PDMCS system. In Section 6, a comparison with existing methods is discussed.

2. RELATED WORK

2.1. MDA web-based software

There has been a lot of research on model transformation as a development method of MDA based web application (software). Let's look at the existing methods of converting a CIM model into a PIM model to develop a web application by using MDA. In [6], based on the MDA, it addressed the model transformation of the BPMN model at CIM level to the use case model and class model (with MVC expression) at PIM level. The transformation rule between models was implemented in the atras transformation language (ATL). Rhazali et. al [7] proposed a method for converting the CIM activity model into a web model of PIM use case model and class model, and then into SoaML and interaction flow modeling language (IFML). In [8], it addressed a method for converting a CIM business model to a PIM web model. That is, the business model expressed as the activity model is transformed into the web model which is the use case model. In [9], it provided the model transformation of the CIM E3value model to the PIM IFML model. It suggested the automatic transformation between models by applying ATL transformation rules based on the metamodel. In this paper, MDA approach will be used for defining a model-based layered cloud modeling process

2.2. Cloud service development method

For the development of UML-based cloud applications, CAML (Cloud Application Modeling Language) [10] expanded from a UML deployment model is used to expresses cloud applications with component models and deployment topologies based on the patterns of the model driven architecture (MDA). This, however, does not include an early phase of defining requirements, meets some characteristics of cloud services only, and does not provide a method that considers security. For this reason, it is not easy to use CAML in reality. Kamali et al [11] develops cloud applications by identifying and designing nine requirements for designing, executing and managing cloud applications. As this method is limited to deployment modeling only, this does not support the overall modeling process for cloud applications. For the development of feature-based cloud applications, Hwang et al [12] suggested a software development methodology for multi-tenant SaaS cloud services (SCoD: SaaS Cloud-oriented Development). Since this development method considers tenant-focused cloud features only, it is difficult to use it for general purposes as a feature-based, not object-based, cloud development process. Benfenatki et al [13] Suggested, as an agile methodology based on service-oriented architecture (SOA), a cloud application development methodology, called MADONA (Methodology for Automatic Development of clOud-based busiNess Application), that encompasses several processes from specifying requirements to combining, implementing and deploying services. However, since this method is based on reuse, that is, a development method focused on identifying and combining services, this does not provide a UML-based development method for developing new cloud services. For the development of architecture-oriented cloud applications, Hamdaqa et al [14] suggested a cloud software architecture for designing cloud applications as a metamodel. Establishment of SaaS cloud services [15] is similar to the development of SOA-based applications, which has processes including analysis, design, development and transformation

Among cloud service development cases, Zhang et al [16] suggested an architecture design process and domain-specific architecture description language for developing cloud robotic systems. One of the commercial cloud service development methods is Cloud Native Application development in PaaS (CNAPS) [17], a cloud development methodology for PaaS services characterized by microservice

architecture and domain-driven designs. CNAPS is an incremental model composed of the phases of defining requirements and architecture, domain-driven design, agile model-driven development (MDD), sustainable development, integrated test and system implementation.

Meanwhile, many studies have been conducted on porting existing applications to cloud applications. Existing UML models are transformed into cloud services after being converted into SOA-based models. Sabiri and Benabbou [18] suggested a method of porting an existing application or component to a cloud application, but this is also not for developing a new cloud application. [19] Introduced an abstract model for standard application programming interface (API) to secure compatibility between different clouds. The model suggested 8 common and standard API services and developed APIs using UML, and the developed API was applied to Amazon's EC2 and Google's AppEngine, which indicates that this model is for the development of standard APIs, not for the development of cloud applications. Jagli and Yeddu [20] suggested Cloud SaaS SDLC (Software Development Life Cycle) that is suitable for the development of cloud applications, and this covered the life cycle model for the cloud development phase only. Some earlier studies were conducted on the types of cloud services to be established [21-22].

3. RESULTS SERVICE-ORIENTED CLOUD METAMODEL

To develop services that reflect the characteristics and policies of cloud, this study defined a metamodel with modeling elements required for modeling from defining requirements at an early phase to designing an architecture for grouping distributed deployments. Depending on the phase of development, a modeling process was established using these modeling elements. In this chapter, a feature-based metamodel for developing a cloud service and its elements were defined.

3.1. Cloud service metamodel

A metamodel for developing a cloud service needs to be composed of the characteristic elements of requirements that this service need to have. In addition, the features of cloud need to be considered from the perspective of design and implementation. From the perspective of design, there are features like security (privacy etc.), resource sharing (group management, distributed management, change management), tenant-base and self-service, and from the perspective of implementation, features such as dynamic environments, various media and on-demand measured service are included.

Based on these characteristic elements, a cloud service metamodel (CSM) is defined as shown in Figure 1. First, elements that are essential and common in a cloud service such as SaaS, PaaS, IaaS, DaaS (Database) and XaaS are identified as a delivery type (or model), and its characteristics such as resource_sharing, tenent_service, and self_service are extracted. Requirements that a cloud service needs to meet are composed of business-processes and components as a function, and security as a non-function. In addition, execution_environment (including media_independent) for the implementation of services, and in particular synchronization that is important for sharing resources are identified, and cloud_policy for implementing a cloud service, and cloud_model and cloud_property (function_property, security_property) for implementing based on this policy are identified.

In terms of the relations between the elements of cloud modeling, since the elements of cloud_service including service_name, service_definition, user, delivery_type, operation_type and use_type are essential components, their relations are expressed with aggregation and composition. On the other hand, the relation between cloud_service and cloud_policy is that of realization because cloud_policy is an element for realizing the service. In addition, to provide hierarchical approaches depending on the level of the abstraction of development, the modeling elements of a cloud service need to be hierarchized using the MDA method. In other words, based on the level of patterns of CIM (domain-commonality, thin solid line box), PIM (domain customizability, platform non-specific-commonality, thick dotted line box), and PSM (platform specific-variability, thick solid line box), the components of a cloud service metamodel are hierarchized and defined. For example, modeling elements including cloud_service, service_name, service_difinition, delivery_type, use_type and cloud_policy are involved in the CIM modeling level; function_property and security_property, in the PIM modeling level; and modeling elements (detailed algorithm, protocol) including cloud_model, component, cloud_mechanism, SLA_monitor (Service Level Aggreement) and synchronization, in the PSM modeling level. When modeling a cloud application, an application model can be generated in an easy and accurate manner by using the modeling elements of this cloud service metamodel and the relations between them. In addition, it is possible to create a cloud model by the level of the abstraction of development, which improves the reusability of the model.

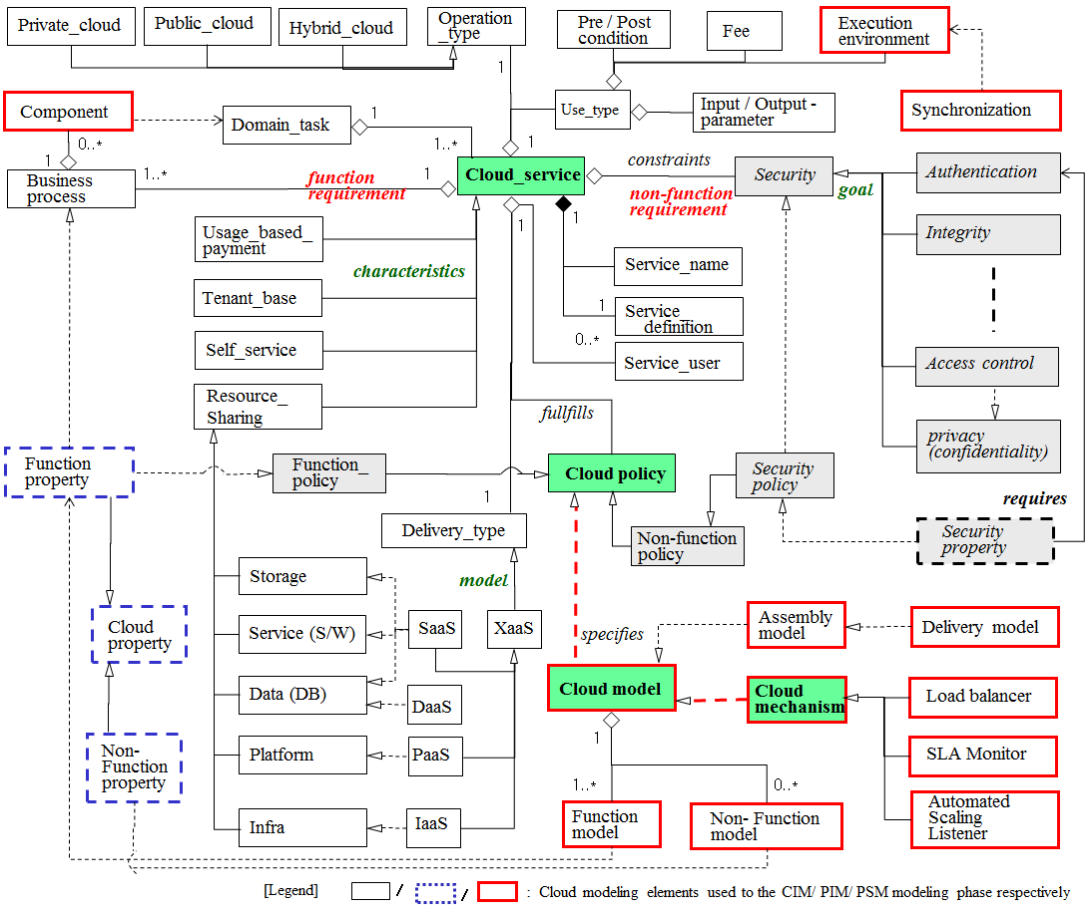

Figure 1. A cloud service metamodel for designing cloud application: CSM

For the standardized specification of modeling elements of a unit "cloud service," the elements are defined according to the method of defining data elements (classification standard and attribute specification) in the Metadata Registry (MDR, ISO/IEC 11179) standard [23-25]. Table 1 shows the definition of the cloud service specification (or attribute specification) of a unit cloud service that have key common elements that are directly involved, focusing on the elements of "Cloud_service" in the cloud service metamodel. That is, the modeling elements in the CIM level that are directly involved in "cloud_service" in the metamodel in Figure 1 are defined. Therefore, the cloud service specifies the attributes that the service needs to have according to the format in Table 1, which can be utilized to search and share the cloud service by doing so.

3.2. OCL definition of cloud modeling elements

In order to clearly express the constraints of the cloud modeling elements defined in the metamodel, OCL (Object Constraint Language) [26-27] is used, because the metamodel expressed as a class model does not fully express the meaning of modeling elements themselves and the constraints between them in a graphic way due to its non-standardized specification. The attribute profile of modeling elements in Table 1 is also stated in natural language, which contains ambiguity. For this reason, the syntactic relations of modeling elements and the profile of attributes are not clearly stated in the class model. Therefore, invariant conditions that modeling elements have, pre/post conditions and other constraints need to be added to define the modeling elements in a clear and specific manner. For example, Figure 2 shows the specification of the modeling element of "Cloud_service" in the cloud service metamodel as shown in Figure 1 written in OCL. The specification converted from the class model to OCL maps the class model to a context, attributes to types, and operation to operation. The specification written in OCL expresses operation by declaring the types of constants or variables that use inv and using pre/ post/ let. As shown in Figure 2, since there is only one name of the cloud service and one type of the model, which is unchangeable, it is expressed as "1" in the declaration of the type.

Table 1. Cloud service specification (attribute specification) of unit cloud service

Cloud service property (CIM)		Cloud service definition
Identifying Attributes	Service_name	Name of a cloud service
	Service_definition (foundation)	Specification of common properties of cloud services - Describe the functions, staus of fuctions, and semantics of functions provided by the service
	Service_user	Users who use cloud services - Related to the realization of tenant-base
	Domain_task	The name of the upper domain and task to which the cloud service belongs. - Domain, application area
	Cloud_policy	The policy on how to provide and manage Shared Services Assets, ie, guidelines and regulations (Resource policy, Service Level Agreement service area policy, legal policy, payment policy, etc.)
	- Function policy	- Functional goals that this service must meet
	- Security policy	- The security objectives that this service must meet (non-functional) Authentification, privacy(confidentiality), integrity, availability, access control, non-repudiation
	Business process	Business flow or scenario of a cloud service
Delivery Attributes	Delivery_type	Choice of cloud service delivery type: SaaS, PaaS, IaaS
	Opearation_type	Select the type of target that provides cloud service: Public_cloud, Private_cloud, Hybrid_cloud
Use Attributes	Pre_condition	Conditions that must be satisfied before the execution of the cloud service
	Post_condition	Conditions that must be satisfied after the implementation of the cloud service
	Fee	The fee for using this cloud service

```
context Cloud_service inv:
    self.nameOfCloud_service = 1
    self.modeltypeOfCloud_service = 1  // (SaaS, PaaS, IaaS)
    self.numberOfComponent  > 1
    self.oclIsTypeOf(Cloud_service)  : Boolean   -- is true
context Cloud_service::useCloud_service(m: Member)
    pre registeredMember : user.includeGroupMember(m)
    pre authentification : u.checkLogin(u: User ) : Boolean
    pre shareResource : numberOfShareResource >= 1
    Post payBills(cost: int) : self.rentFee = self.rentFee@pre -> add(cost)
```

Figure 2. An OCL specification for the "Cloud_service" modeling element

3.3. An integrated metamodel between cloud and UML

Normally, business applications are designed using unified modeling language (UML), a universal language, for developing software, and thus cloud applications are modeled using UML. Therefore, the modeling elements of the cloud service in Figure 1 need to be expressed with the elements of a UML model. The Integrated Metamodel of mapping Cloud into UML design (IMCU) in Figure 3 shows the connected modeling between cloud modeling and UML modeling. Figure 3 shows how the application model of cloud can be established into a UML model through a three-layered modeling approach. The UML model on the right side is the function-security integrated metamodel suggested by [28], and the cloud model (gray box) on the left side is the one that was newly defined based on Figure 1 in this study. Mutual mapping between these models is defined based on the realization relation between modeling elements. The cloud requirement modeling marked in gray on the left side is the one that this study suggests.

The composition of the IMCU is defined by identifying the key elements of individual metamodels associated with cloud modeling and UML modeling (for example, "class" in the class model), and connecting modeling elements between models and elements in each modeling phase in a vertical (materialization of models) and horizontal (relation between cloud and UML mapping) manner. The IMCU is hierarchized according to the MDA-based CIM, PIM and PIM modeling levels in order to establish a cloud application model by the level of the abstraction of development.

Horizontal mapping between cloud modeling and UML-based modeling is expressed with the relations of the cloud elements that give constraints (function, non-function) to UML elements. In the CIM level, use cases that express functions in an early use case model for UML modeling are the elements that compose cloud_service in cloud modeling, and thus the relation between them is the realization relation. This means that a unit cloud service is expressed as a use case of UML, and is realized later. In the PIM level, since classes in the class model need to meet the attributes of cloud, their

relation with cloud_property is the realization relation. In the PSM level, the cloud model is realized as a component model by assembling components, which is the realization relation. As a modeling element for specifying systems in the PSM level, certain component types, in this study spring (spring frameworks) and COM+, were expressed in UML modeling, and certain cloud sites, in this study Google's AppEngine and Amazon's EC2, were expressed in cloud modeling.

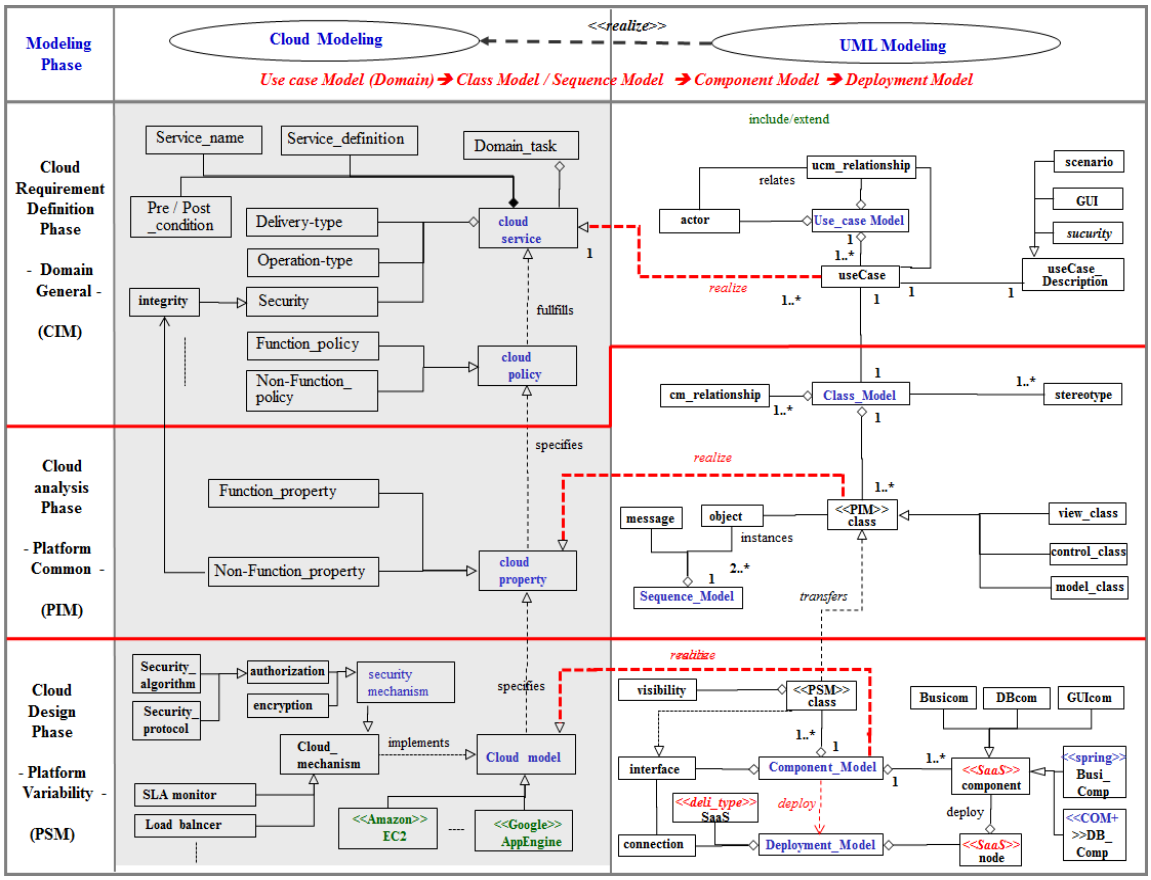

Figure 3. The integrated metamodel of harmonizing cloud into UML design: IMCU

4. SERVICE-ORIENTED CLOUD MODELING PROCESS

In this chapter, a service-oriented cloud modeling process is established using the metamodel (chapter 3) according to the life cycle of cloud development. To do so, a conceptual modeling framework is defined, and the structure and behavior models of the cloud development process are defined in detail based on the framework. For the metamodel-based definition of development process, task activities in each modeling phase are performed using the modeling elements of the hierarchical metamodel in chapter 3 according to the MDA paradigm. By doing so, the productivity of development can be improved, and the development process can be easily used. Meanwhile, the design of the suggested cloud modeling process has the following principles: multi-tenant-focused customized service (specified as service user in the cloud service specification); cloud service-oriented; architecture-focused (specifying the type of deployment in a deployment model); and reuse-oriented (MDA approach).

4.1. Cloud service modeling framework

The software development process (methodology) is defined with methods (or paradigms) and processes. This development process has incremental and repeated development paradigms, and development phases and task activities are performed in order. That is, the approaching method between development phases has an incremental and repeated life cycle, which allows to continuously improve and expand the created model. Based on this approaching method, the framework of the general cloud development process was defined as shown in Figure 4. This framework is defined as a two-dimensional structure composed of the function and non-function requirements of the cloud application service from a vertical

perspective, and modeling phase based on MDD (Model Driven Development) that has time concept from a horizontal perspective. Therefore, the cloud service modeling process is defined with three development phases: cloud requirement definition phase (CIM level), cloud analysis phase (PIM level) and cloud design phase (PSM level). Each phase can be modeled hierarchically using the cloud service metamodel CSM, Figure 1 and the integrated metamodel IMCU, Figure 3.

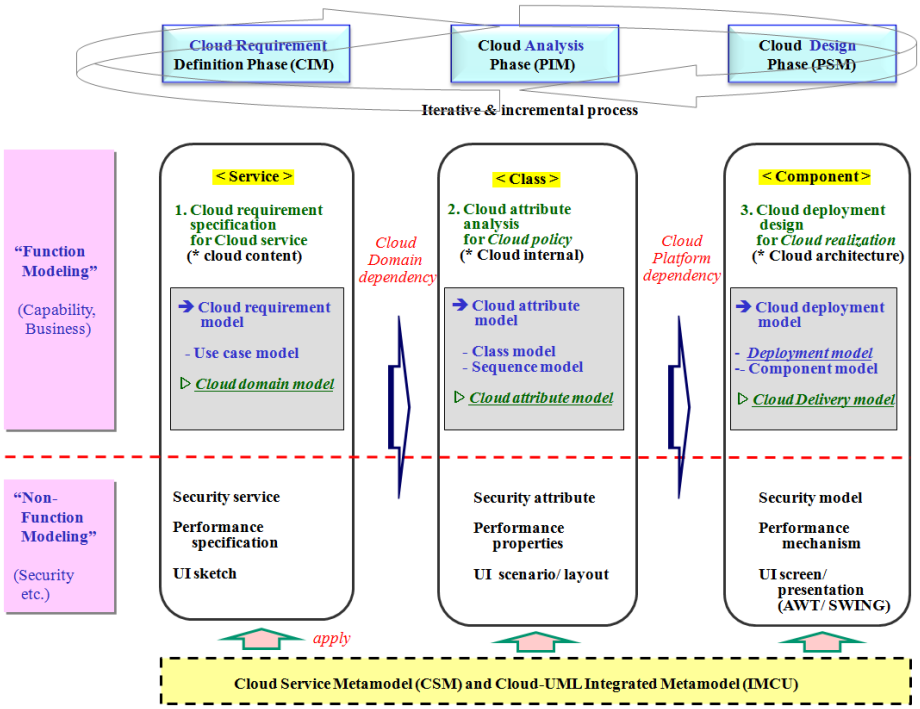

Figure 4. A modeling process framework for cloud service: MPF

4.2. Service-oriented cloud modeling process

Based on the cloud service modeling framework MPF, Figure 4, detailed development process models are obtained as follows: SM-CMP (Structural Model for Cloud Modeling Process) as shown in Figure 5, and BM-CMP (Behavioral Model for Cloud Modeling Process (skipped for lack of space). The structural model for modeling process expresses task activities required during the development process, and the behavioral model (skipped for lack of space) shows the implementation of activities in the development process.

Modeling phases are composed of multiple task activities, and outputs are generated as the outcomes of the activities. The activities are modeling tasks associated with the function and non-function (security, performance, reliability, etc.) of cloud. As an example, Definition 4-1 shows the definition of task activities in the development phase not the definition of modeling phase (set of activities). Definition 4-1 (Cloud modeling activity). A cloud modeling activity of cloud analysis phase CMA = (n, Role, Input, Task, →, WP, Criteria) consists of: (1) the modeling activity's name n (Identify cloud attribute identify, Make analysis class model, Build analysis sequence model); (2) the role and responsibility of modeling activity Role (function analyst); (3) the required input for modeling activity (use case model, cloud service specification); (4) a finite-set Task of tasks s, t, r ∈ CMA, where the task can contain options ([Task]) or can be mandatory (Task), and where a cloud model should be built as an artifact using the work product of the previous task as well as the modeling elements in the metamodel; (5) a transition relation → ⊆ (Task x Task), where → contains the relation type of previous/after and fork/join among the tasks; (6) the output or work product WP of a task; and (7) Initiation and end criteria for initiating and finishing this task.

The overall process of cloud application modeling is as follows. In the phase of defining cloud requirements, requirements are expressed with a use case model, and associated use cases are grouped to identify unit cloud services, and to write the specification of each cloud service. To specify the service model in detail, class models (function structural models), and sequence models (implementation models) for each

use case included within a unit cloud service are written in the cloud analysis phase. In the cloud design phase in which architectures and specific mechanisms are established considering implementation environments, classes are grouped to generate components, and cloud application component models are created. Unit cloud services are generated with components associated with use cases included within a unit service that is defined in the requirement definition phase. The internal design of components is performed using multiple class models.

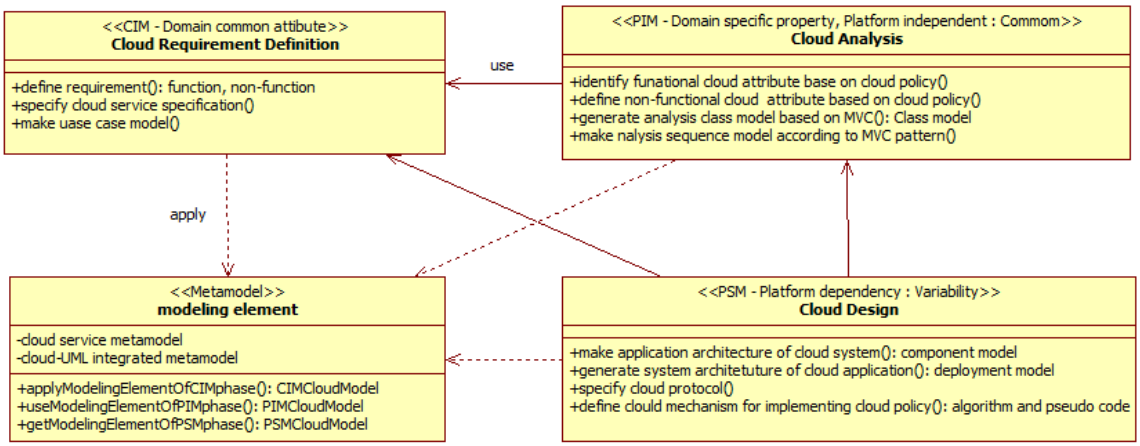

Figure 5. A structural model for cloud modeling process: SM-CMP

Here, the relations between modeling elements can differ depending on granularity, but they can be set as follows:

- Identifying unit cloud services with more than one use case
- Writing class and sequence models for unit cloud services
  (If the size of a service is big, models can be written for each use case within each service.)
- Generating components with more than one use case within a service
- Forming services with components, and components with classes

In the cloud requirement definition phase (CIM-level), for generating cloud domain models in Figure 5, requirements and policies that are necessary for independently developing cloud applications are established in target domains. Task activities in this phase include “definition of requirements” of functions and non-functions; “writing function models based on use cases” and “writing cloud service specifications.” Here, the specific task activities for “writing function models based on use cases” include “writing use case models” and “writing use case specifications.” When writing cloud service specifications, the policies of cloud, that is, the function and non-function (security, etc.) policies of cloud are specified.

For requirements for functions, use case models are written, and “use case specifications” are written for one use case. After that, based on functional similarity, associated use cases are combined to identify unit cloud services and to specify each service in detail through cloud service specifications. Here, high-level cloud policies are also defined. For tasks involved in “defining requirements,” the requirements of cloud applications are analyzed and identified by tenant based on commonality and variability [12]. In use case models, use cases that have an “extend” relation between use cases shows the requirements of variability, while the rest use cases show the requirements of commonality. Cloud service specification are written based on Table 1. Unit cloud services are identified using a use case-based method [29], but their types are selected according the granularity of services.

In the cloud analysis phase (PIM-level) for establishing cloud attribute models, cloud attribute models for unit cloud services (or the size of use cases), the outputs of the requirement definition phase, are written independently in a development platform by identifying and adding cloud attributes in class models and sequence models using the MVC method based on the use case and cloud service specifications. In other words, the function and non-function attributes of cloud are defined based on the already-defined cloud policies. The attributes are reflected and expressed in the class and sequence models in the form of stereotypes in the PIM level of the metamodel using the modeling elements. The reason why one class model and sequence model are created for a unit cloud service is to generate service-oriented analysis models, and to secure the size of models in the case when the size of use cases is small.

Lastly, in the cloud design phase (PSM-level), resource-sharing distributed or C/S (Client/Server) cloud architecture models are established and the inner structure of cloud classes are designed to define cloud deployment and cloud delivery models by adding implementation environments. First, a system architecture is expressed as a deployment model, and an application architecture is expressed as a component model. A unit cloud service defined in the CIM level is expanded and expressed using the notation of deployment models. Components included in a unit cloud service are deployed and expressed within nodes, and are grouped to express the service and components together within nodes using the stereotype of <<Unit_Cloud_Service>>. Models are designed to provide the cloud computing service by mounting the components that independently function within the architecture model, and the cloud service in which these components are combined and converted into an executable file. By doing so, service-oriented cloud applications can be established. After that, class models and sequence models based on internal cloud attributes to which implementation environments are added are defined. In addition, the protocol and algorithm of cloud are designed in class operations.

5. CASE STUDY

In order to demonstrate the effectiveness of the proposed method, practices are conducted and deliverables by activities are described based on the procedures of the SM-CMP in Figure 5 targeting the Personal Data Management Service (PDMCS) system. At this time, the conversion and mapping between the UML models are as described in [28]. That is, the transformation mapping between modeling elements over modeling phases is applied the method in [28]. Each activity is conducted to do modeling by using modeling elements of CSM metamodel as shown in Figure 1, UML metamodels [5], and IMCU metamodel as shown in Figure 3 hierarchized by development phases. The PDMCS system is a system to manage processes of supporting storage capacity support, conducting synchronization and backup, providing email and folder management services. In this case, the design on a multi-tenant service for data storage, folder management and synchronization functions is performed. Deliverables of this case are seen in use case model and cloud service specification at the CIM cloud requirement definition phase, class model and sequence model at the PIM cloud analysis phase, and the component model and deployment model at the PSM cloud design phase.

5.1. Cloud requirement definition for PDMCS system (CIM)

The modeling work in the cloud requirement definition phase as shown in Figure 5 create a domain model of the PDMCS system as a use case model, identify the unit cloud services based on the use case, and specify the cloud policy for these services. Shown in Figure 6 is a use case model that consists of the requirement functions for this system: account, folder, capacity, file, synchronization, and so on. For how the cloud's characteristics in Figure 1 are represented in the identified use cases in Figure 6, the "account" use case represents the cloud personalization service of the "Tenant_base" and "Self_service" modeling elements. The "charge" use case represents "Usage_based_payment", and the "folder_share" use case expresses the cloud characteristic of "Resource_sharing" modeling element respectively. Through this model, commonality and variability also can be identified by relationship type between use cases. That is, use cases that have an "extend" relationship between use cases are identified as requirement functions of variability. In other words, "folder_share", "synchronization (back-up)", "charge" use cases are variability functions and the remaining use cases mean common functions. Next, based on conceptual similarity of functions, use cases are grouped to identify unit cloud services. As a result, the "account_management", "folder_management", "file_management", "payment_management", and "capacity_management" are identified as unit cloud services and they are expressed in package notation. For instance, the "Account_management" unit cloud service is a group of use cases for account, login, security (SSO: Single Sign On). Next, in order to specify unit cloud services, cloud service specification is written by using the attribute specification with cloud in Table 1. As an example, the cloud service specification for unit "folder" cloud service is shown in Table 2. In table 2, the cloud policy for the "folder" cloud service may have functional-policy and non-functional-policy, as shown in the modeling elements in Figure 1. That is, these include resource policy (computer, network, storage, firewall, etc.), Service Level Agreement (SLA), security policy, and payment policy. For instance, the "Provide folder service only to account person registered in the cloud." sentence describes security policy of this cloud service. This is shown on how the "Authentication" security policy of the "account" use case is specified by cloud service specification in detail. In other words, this is shown by mapping the characteristics of the cloud in Figure 3 to UML elements. That is, "Non-Function_policy" of "cloud_policy" is expressed into UML "useCase" modeling element.

5.2. Cloud analysis for PDMCS system (PIM)

In the cloud analysis phase as shown in Figure 5, the functional models that is composed of the class model and the sequence model as analysis model of PDMCS are specified in detail based on the cloud service specification asshown in Table 2, deliverable from prior phase by the MVC pattern at the PIM level independently of the platform environment. Here, one class model and one sequence model are written for one cloud service.

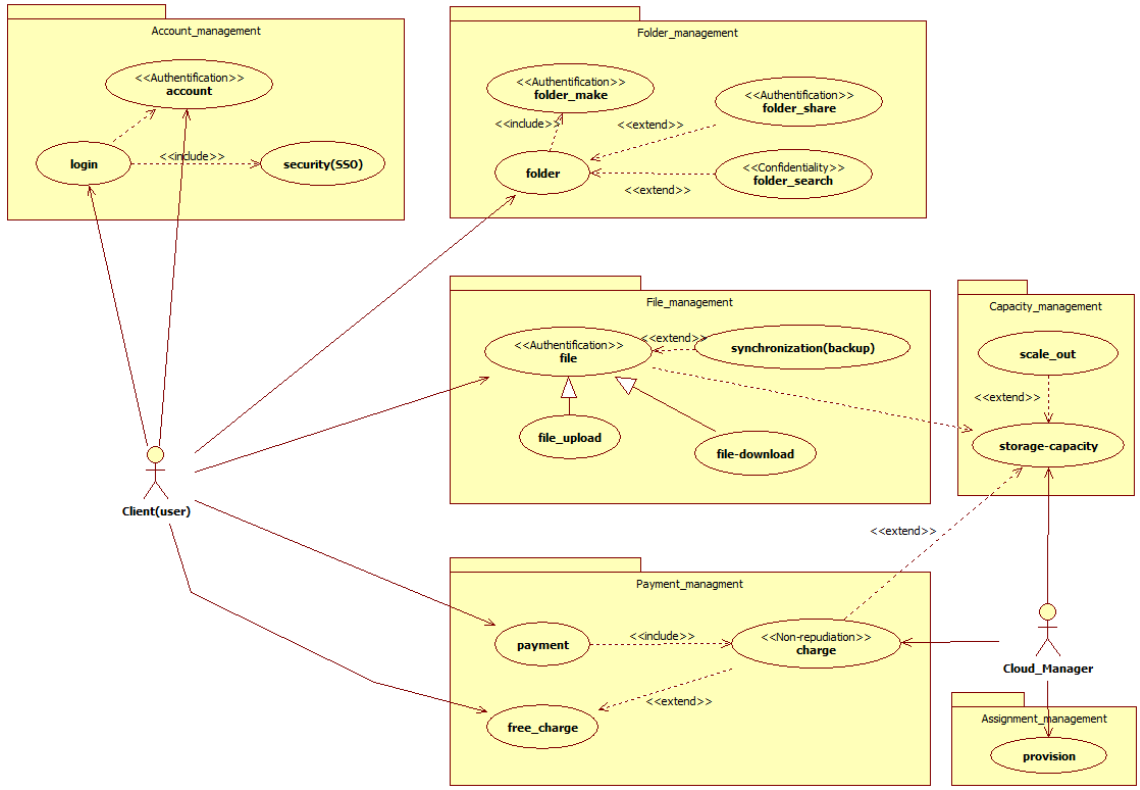

Figure 6. Use case model of PDMCS system

First, the attributes of functional and non-functional aspects related to cloud policy based on the cloud service specification are extracted and incorporate them into the analysis model. Therefore, the class model is generated as shown in Figure 7, reflecting the attributes of the cloud. Policy-based cloud attributes was described in Table 2 in the cloud policy section. These attributes can be expressed within the classes (in attributes and operations) of the class model. For instance, the cloud attribute for the policy of "① To create a folder, user must have a certain amount of personal storage." in Table 2 is set to attribute of "personal_storage_capacity" class as shown "storage_size = 25giga" that the specified capacity was declared as the initial value in Figure 7. As shown in Figure 7, the class model with the attributes of the cloud (underlined) is created by the MVC pattern for the unit "folder" cloud service. In Figure 7, the classes directly related to folder function are grouped (in bold box) and represented in the "folder_management" package that is a unit folder cloud service. The class model is created with both folder itself classes and the related classes to use folder function, such as login class, and file class that folder includes.

In this phase, personalization service and self service, which are the characteristic elements of the cloud as shown in Figure 1, are represented into respectively "personal_service_UI" class for handling individual use, "folder_controller" class for processing resource sharing service and also "account" class and "personal_storage_capacity" class for tenant service. The MVC expression for “folder” class is expressed as follows: the "folder_UI" as a View class represents screen display of folder explorer; the "folder_controller" as a Control class indicates performing actual function processing; and the "folder" as a Model class means an entity class for storing and managing related folder data. On the other hand, the security attributes of the non-functional cloud policy are expressed within the related classes. For example, in the "folder_Controller" class, a security of "non-repudiation" is expressed as a stereotype.

5.3. Cloud design for PDMCS system (PSM)

In the cloud design phase, the environments of an operation platform are reflected, and a system architecture is designed by adding cloud to it for the implementation of a PDMCS system based on the cloud analysis model as shown in Figure 7. The inner structure of the architecture is specified with the protocol and mechanism of cloud, and is written using the modeling elements applied in the PSM level as shown in Figure 1 and Figure 3. For the PDMCS system, application and system architectures are written. These models mean the "cloud model," a PSM modeling element in Figure 1. Figure 8 shows the application architecture designed using the MVC method for a unit cloud service, "Folder."

Table 2. Cloud service specification of unit "folder" cloud service

Cloud service property (CIM)		Cloud service definition
Identifying Attributes	Service_name	Folder management
	Service_definition (foundation)	The cloud service creates and shares folders, stores files in folders, and provides search.
	Service_user	Users using Google cloud service
	Domain_task	File management
	Cloud_policy - Function policy - Security policy	① To create a folder, user must have a certain amount of personal storage. ② The folder determines whether to share according to the individual security policy. ③ Provide folder services only to account users registered in the cloud service - The Google Account holder of a specific tenant that is authorized has access and management rights to the folder. - Login is required to use folder service. (Authentication) - Reading and writing from an illegal subject to a folder are prohibited. (Confidentiality, Integrity)
	Business process	(1) Make a folder. (2) Save the file. (3) Set up and provide sharing and searching of folders.
Delivery Attributes	Delivery_type	SaaS
	Opearation_type	Private_cloud (a dedicated system managed by users, defined by privacy settings and administrative responsibilities)
Use Attributes	Pre_condition	- Users should be logged in to Google's cloud system. - Storage capacity should be at least 15 gigabytes.
	Post_condition	The folder is created and the file is saved.
	Fee	Free

After that, based on this component model, a system architecture model is written as shown in Figure 9. Here, a C/S (Client/ Server)-type cloud architecture model is designed. Cloud application, file and DB servers are established. The unit cloud service, "Folder," was created by grouping three execution components including "FolderUIWeb.war." As an example of reflecting operation platform environments, among cloud storage types (object, file, block), file storage was selected, and Network Attached Storage (NAS) was expressed as its server using Amazon's Elastic File System (EFS) solution.

Meanwhile, for communications between client and server (C/S), the processing of log-in through web sites was expressed with HTTP, and P2P for file transfer. In addition, for security assurance for mutual communications, <<secrecy, encrypted, authentification>> was expressed, which expresses the requirement of security among non-function cloud policies. The deployment type of the "Folder" cloud service was expressed with <<SaaS>>. The cloud model above is specified and implemented with protocols and mechanisms. Cloud mechanisms as shown in Figure 1 include auto-expand listener, load distribution, SLA monitor and on-demand measurement monitor. For example, auto-expand listener is an agent that automatically expands the capacity when resources lack. In the PSM level, its processing mechanism is written in detail. The characteristics of cloud are reflected in this level as follows. "Customization" is reflected by expressing that each user computer (UC1) has "Folder_Service" in its server in Figure 9, and "distributed processing" is reflected by expressing that users in A and B areas use servers in each area. For example, "different platform operation" is implemented by allowing "FolderControlSearch.jar" to be operated in both the Spring Framework and COM+.

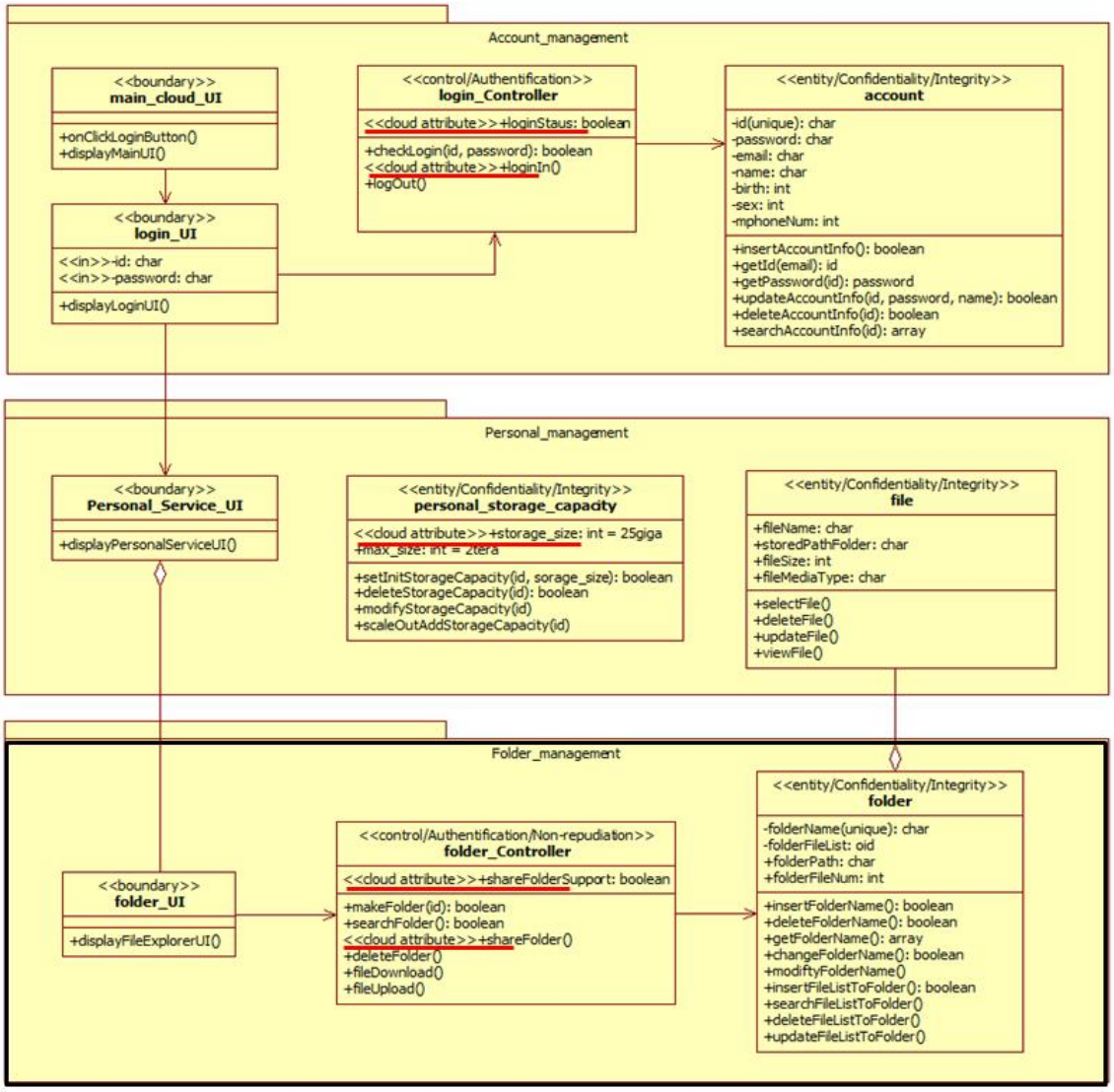

Figure 7. Class model with cloud attribute of the unit "folder" cloud service in PDMCS system

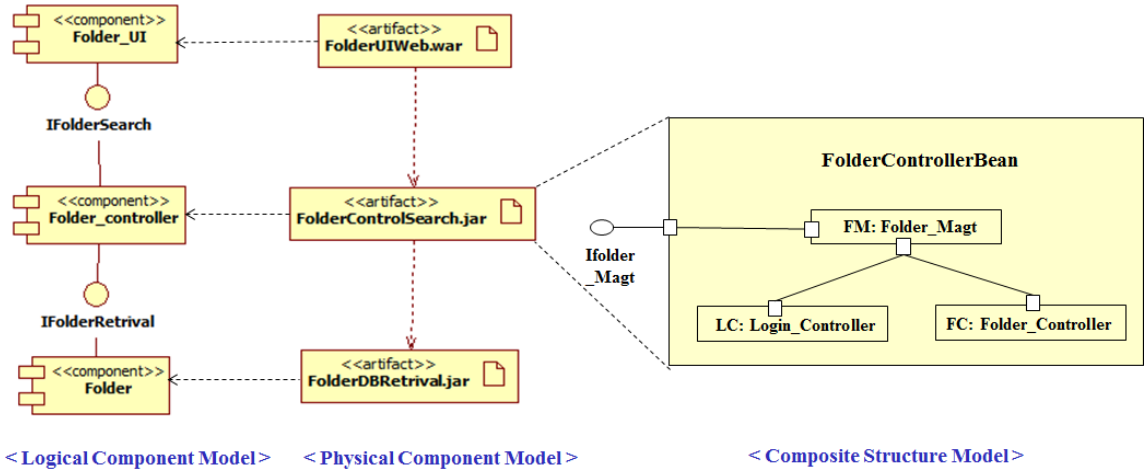

Figure 8. Component model of application architecture for unit "Folder" cloud service

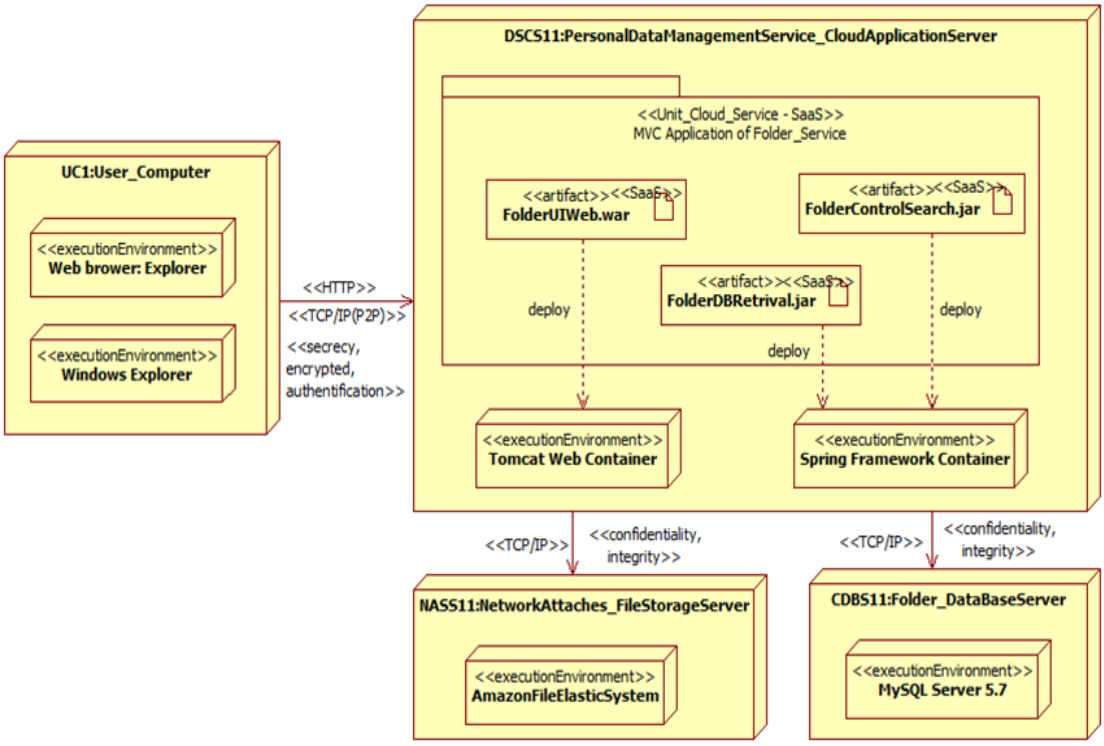

Figure 9. Component model of system architecture for unit "Folder" cloud service

6. EVALUATION

As an evaluation of the proposed method, this section describes the self-evaluation, comparison with the existing methods, and also characteristics and shortcomings of the proposed method.

6.1. Comparative evaluation with the existing method

In order to compare proposed methods in this study and existing methods together, matrices related to metamodels and modeling processes were analysed as shown in Table 3. Table 3 shows the analysis of the metamodel and the matrices related to the modeling process for comparison between the presentation method and the existing method. The score of each item is 1.0. The value of the item for comparison methods means level or degree of support.

CAML proposed a MDA approach method for creating a PIM component model to cloud applications and designing it as a PSM deployment model using CAML that extends UML. However, it did not support service oriented method and requirement definition phase, and cloud characteristics such as personalization and resource sharing were not reflected enough. UCC presented a method for designing, executing, and managing nine requirements for building cloud applications. This also defined meta-models that is consisted of modeling elements for deploying cloud applications in the cloud environment. However, it does not provide a service-oriented and the modeling process covering overall development pahses. SCoD suggested a feature-based cloud development method that supports multi-tenants. However, this was not a general UML-based object development method and did not support service-oriented development. MADONA proposed a reuse-based service-oriented cloud development method, so was not a method for developing new cloud services. Conventional research did not support fully based on service-oriented design, hierarchical modeling by MDA and MVC pattern, and metamodel-based approach. Therefore, it was not easy to use and did not maximize the number of reuse-available model of created application model.

Table 3. Comparison of metamodel and process with legacy method

Assessed items (point)	CAML [10]	UCC [11]	SCoD [12]	MADONA [13]	Proposed method
Mapping metamodel from Cloud to UML modelling (1.0)	0.50	0.50	0.25	0.25	0.75
Layered cloud Metamodel (1.0)	0.25	0.25	0.25	0.25	1.00
Cloud modelling process Covering whole development life cycle (1.0)	0.50	0.75	1.00	1.00 (Reuse)	0.50 (Design)
Hierarchical designing by layered metamodel or process (1.0)	0.75	0.50	0.50	0.25	0.75
Service-oriented modelling (1.0)	0.25	0.25	0.75	1.00	0.50
Security modelling method (1.0)	0.25	0.25	0.25	0.25	0.75
MDA-based modelling (1.0)	0.75	0.50	0.50	0.50	0.75
MVC-based modelling (1.0)	0.25	0.25	0.25	0.25	0.75
Conversion profile between Models(1.0)	0.25	0.25	0.50	0.25	0.50
Weight total 100% (9.0)	42%(3.75)	39%(3.50)	47%(4.25)	44%(4.00)	69%(6.25)

[Legend] 1.00: Very high (support), 0.75: Good, 0.5

6.2. Characteristics and limitations

The characteristics and expected effects of the proposed methods are as follows:

a. Cloud property-driven
   - Cloud service metamodels are proposed based on cloud properties such as personalization (tenant, self-service), resource sharing, distributed processing, etc.
   - Cloud chracteristics are modeled and detailed by top-down approach according to service, policy, attribute, mechanism, etc.

b. Cloud-UML Mapping-driven in Figure 3
   - The integrated metamodel (IMCU) and modeling process is provided on how the characteristic elements of the cloud are mapped and represented into UML modeling elements.

c. Service-driven
   - This method provides that is designed the architecture and interior focusing to unit cloud service.

d. Model pattern-driven
   - The number of reusable models can be maximized by using approaches such as the CIM/PIM/PSM of the MDA, and Model/View/Control methods. CIM model + {(number of models by PIM/PSM) x (number of models by MVC)} = 1 + (2 x 3) = 7
   - Independent modularity between models is more increased.

e. Metamodel-based driven
   - Modeling practices become easy by applying the modeling elements and relationships of the formalized individual as shown in Figure 1 and integration Figure 3 metamodels.

The limitations of the proposed methods are as follows:

a. Proposed cloud metamodels and modeling processes cannot provide a perfectly integrated modeling between function and security.
   - To use proposed modeling processes to real practices as commercialized methodologies, task, step, guideline and artifact should be specified more and connecting practices should be clarified more.

b. It does not provide more concreted methods of establishing various cloud policies, attributes, protocols, and techniques that implement and realizes this policy.
   - The development methods for SLA policy, resource policy, service area policy, legal policy, payment policy, etc. are required.

c. It is necessary to support successful practice deliverables by continuously applying them to cases

7. CONCLUSION

Conventional methods of developing cloud applications create services and expand deployment models based on porting or reuse, but UML-based unified development methods that consider the overall characteristics of cloud have not been researched much. This study suggested metamodels, transformation profiles, frameworks and processes based on MDA and MVC patterns for developing service-oriented applications that reflect the characteristics of cloud. The cloud service metamodel was defined with cloud requirement features and elements for implementation including policies, attributes and mechanisms. How these cloud elements are transformed and expressed into UML modeling elements was established in an MDA-based integrated metamodel. The framework of three development phases to which MVC patterns were applied was defined, and the cloud modeling process composed of development phases and task activities was defined. Task activities were ensured to be performed using the modeling elements and

relations of standardized individual and integrated metamodels. In terms of their expected effects. For example, it will be possible to design unified service-oriented cloud applications suitable for cloud computing environments, and to design clearer cloud service models using standardized cloud metamodels and processes. As a follow-up study, it will be necessary to develop specific guidelines for task activities within the cloud modeling process that can accept various policies, and to research the automated establishment of cloud applications by developing case tools with the improved methods.

REFERENCES

[1] E. y. Jang, et al., "A study of Modeling and Simulation for the Availability Optimization of Cloud Computing Service," *Journal of The Korea Society for Simulation*, vol. 20, pp. 1-8, Mar 2011.

[2] R. Gour, "9 Major Characteristics of Cloud Computing," *DZone*, Jan 2019. [Online], Available: https://dzone.com/articles/9-major-characteristics-of-cloud-computing

[3] Samjong KPMG, "Analyzing domestic cloud adoption issues: Focusing on the policy of major countries," Economic Research Institute, May 2016. [Online], Available: https://home.kpmg.com/kr/ko/home/insights/2016/05/issue-monitor-_52----------.html

[4] S. Pearson, "Taking Account of Privacy when Designing Cloud Computing Services," HP Laboratories, 2009. [Online], Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.306.5035&rep=rep1&type=pdf.

[5] C. Y. Song, et al., "An Integrated Design Method for SOA-Based Business Modeling and Software Modeling, *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, pp. 347-377, 2016.

[6] Y. Rhazalia, Y. Hadia, and A. Mouloudia, "Model Transformation with ATL into MDA from CIM to PIM Structured through MVC," *Procedia Computer Science*, vol. 83, pp. 1096-1101, 2016.

[7] Y. Rhazali, Y. Hadi, and A. Mouloudi, "A model transformation in MDA from CIM to PIM represented by web models through SoaML and IFML," *4th IEEE International Colloquium on Information Science and Technology*, pp. 116-121, Jan 2017.

[8] Y. Rhazali, Y. Hadi, I. Chana, M. Lahmer, and A. Rhattoy, "A Model Transformation in Model Driven Architecture from Business Model to Web Model," *IAENG International Journal of Computer Science*, vol. 45, no. 1, pp. 104-117, Feb 2018.

[9] N. Kharmoum, S. Ziti, Y. Rhazali, and F. Omary, "An automatic transformation method from the E3value model to IFML model: An MDA approach," *Journal of Computer Science*, vol. 15, no. 6, pp. 800-813. 2019.

[10] A. Bergmayr, et al., "UML-based Cloud Application Modeling with Libraries, Profiles, and Templates," 2014. [CAML], [Online], Available:http://ceur-ws.org/Vol-1242/paper7.pdf

[11] A. Kamali, et al., "UCC: UML Profile to Cloud Computing Modeling Using stereotypes and tag values," *2014 7th International Symposium on Telecommunications (IST'2014)*, 2014. [UCC]

[12] M. S. Hwang, et al., "Software Development Methodology for SaaS Cloud Service," The Journal of the Institute of Internet, Broadcasting and Communication, vol. 14, pp. 61-67, Feb 2014. [SCoD], [Online], Available: http://www.earticle.net/article.aspx?sn=215116

[13] H. Benfenatki, et al., "Cloud Application Development Methodology," 2014 *IEEE/WIC/ACM International Joint Conferences*, pp. 13-20, Aug 2014. [MADONA]

[14] M. Hamdaqa, et al., "A Reference Model for Developing Cloud Applications," *In Proceedings of the 1st International Conference on Cloud Computing and Services Science*, pp. 98-103, 2011. [Online], Available: https://www.researchgate.net/publication/220865572_A_Reference_Model_for_Developing_Cloud_Applications

[15] J. S. Park, "Cloud adoption strategy and process," Cloud Computing Support Center, vol.4, Oct 2014. [Online], Available: http://www.kosta.or.kr/mail/2015/download/1-KOSTA%202015_150226_KAIST_JSPark.pdf

[16] L. Zhang, et al., "Towards An Architecture-Centric Approach to Manage Variability of Cloud Robotics," Cornell University, Jan 2017. [Online], Available: https://hal.archives-ouvertes.fr/hal-01376287/document

[17] SK, "Cloud Native Application development in PaaS (CNAPS)," *SK C&C*. [Online], Available: https://www.cloudz.co.kr/service/cnaps

[18] K. Sabiri and F. Benabbou, "Methods Migration from On-premise to Cloud," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 17, pp. 58-65, 2015.

[19] M. F. P. Escalera and M. A. L. Chávez, "UML Model of a Standard API for Cloud Computing Application Development," *2012 9th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pp. 26-28, Sep 2012.

[20] D. Jagli, and S. Yeddu, "CloudSDLC: Cloud Software Development Life Cycle," *International Journal of Computer Applications (0975 – 8887)*, vol. 168, pp. 6-10, Jun 2017.

[21] D. H. Sharmaa et al., "Identity and Access Management as Security-as-a-Service from Clouds," *Procedia Computer Science*, vol. 79, pp. 170 – 174, 2016.

[22] CSA, "SecaaS Implementation Guide Category 1 // Identity and Access Management," *Cloud Security Alliance*, Sep 2012. [Online], Available: https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_1_IAM_Implementation_Guidance.pdf

[23] D. W. Gillman, "ISO/IEC JTC1 SC32, ISO/IEC 11179: Specification and Standardization of data elements, Part 1-6," Oct 1997. [Online], Available: http://jtc1sc32.org/doc/N0001-0050/32N0017T.pdf

[24] Australian Institute of Health and Welfare, "Metadata Online Registry (METeOR)," *Australian government*, [Online], Available: https://meteor.aihw.gov.au/content/index.phtml/itemId/181162
[25] M.N. Sylvie, et al., "The ISO/IEC 11179 norm for metadata registries: Does it cover healthcare standards in empirical research?," *Journal of Biomedical Informatics*, vol. 46, pp. 318–327, 2013.
[26] J. Cabot and M. Gogolla, "Object Constraint Language (OCL): a Definitive Guide," 2012. [Online], Available: https://modeling-languages.com/wp-content/uploads/2012/03/OCLChapter.pdf
[27] OMG, "Object Constraint Language," *Object Management Group*, ver. 2.4, Feb 2014, [Online], Available: https://www.omg.org/spec/OCL/2.4/PDF
[28] C. Y. Song and Y. H. Kim, "A Software Modeling Method for Integrating Functional and Security Design," *Journal of Knowledge Information Technology and Systems*, vol. 12, no. 1, pp. 131-155, Feb 2017.
[29] H. R. Yoon, et al., "*The identification method of Web Service based on UseCase*," Proceedings of the Korean Information Science Society, pp. 352-354, Jul 2005.

BIOGRAPHIES OF AUTHORS

Chee-Yang Song received the bachelor and master's degrees in Computer Science from the Hannam university in 1985 and the Chungang University in 1987, respectively. He received the Ph.D. degree in Computer Science from Korea University in 2003. From 1990 to 2005, he was a researcher at Research center of Korea Telecom (KT). He has been a Professor in the Department of Software at Kyungpook National University since 2008. His research interests include service oriented modeling, business-software integrated design process, security design, and cloud computing

Eun-Sook Cho received the B.S. degree in Computer Science from DongEui University, Korea in 1993. He received the M.S and Ph.D degree in Computer Science from SoongSil University, Korea, in 1996 and 2000, respectively. Dr. Cho joined the faculty of the Department of Software Engineering at Seoil University, Seoul, Korea, in 2005. He is currently a Professor in the Department of Software Engineering, Seoil University. He is interested in component-based development, cloud computing, and IoT applications