Model-Driven Development Patterns for Mobile Services in Cloud of Things

Hongming Cai, Senior Member, IEEE, Yizhi Gu, Athanasios V. Vasilakos, Senior Member, IEEE, Boyi Xu, Jun Zhou

Abstract—Cloud of Things (CoT) is an integration of Internet of Things (IoT) and cloud computing for intelligent and smart application especially in mobile environment. Model Driven Architecture (MDA) is used to develop Software as a Service (SaaS) so as to facilitate mobile applications development by relieving developers from technical details. However, traditional service composition or mashup are somewhat unavailable due to complex relations and heterogeneous deployed environments. For the purpose of building cloud-enabled mobile applications in a configurable and adaptive way, Model-Driven Development Patterns based on semantic reasoning mechanism are provided towards CoT application development. Firstly, a meta-model covering both multi-view business elements and service components are provided for models transformation. Then, based on formal representation of models, three patterns from different tiers of Model-View-Controller (MVC) framework are used to transform business models into service component system so as to configure cloud services rapidly. Lastly, a related software platform is also provided for verification. The result shows that the platform is applicable for rapid system development by means of various service integration patterns.

Index Terms—Software Engineering, Cloud of Things, Modeling methodologies, Mobile Applications, Relations between models, Web-based services, Software as a Service

1 INTRODUCTION

With the rapid development of Internet of Things (IoT) applications, the number of devices and mobile applications has increased rapidly. It is said that the number of devices has already exceeded the number of people on the Earth since 2011. And the number of devices is expected to grow to 24 billion by 2020. Thus the requirement of a strong and flexible environment for IoT application support has become a critical issue. Fortunately, Cloud Computing provides a strong basis for resource sharing in a flexible way. IoT and cloud computing working in integration makes a new paradigm named Cloud of Things (CoT) [1]. IoT objects especially mobile devices will be connected via cloud platforms for different business applications. Merging cloud platform and IoT, CoT will take a more and more important role in different industries and research areas [2]. In CoT, IoT objects are extended from sensors to every front-end thing on the Internet. And distributed objects are connected as a whole system for complex and intelligent applications, such as smart house, smart factory, and smart city [3]. Other than data and resources in a single view of cloud or current IoT applications, CoT will pay more attention to intelligent and mobile applications in a business insight. The issues are related to integration of IoT with cloud computing and require smart gateway to perform the complex tasks and comprehensive solution for a certain application requirement, other than the simple event acquiring and transforming as traditional sensors do.

The construction of cloud-enabled mobile applications can take advantage of existing resources. Mobile application in cloud platform could be regarded as a special IoT application in CoT. With the wide employment of IoT application, the physical devices allow users to share the mobile services conveniently. Along with the extensive use and fast improvement of web information technology, an urgent need is generated from end users to develop or configure web-based mobile applications rapidly especially in cloud environment. End users are suggested to implement their customized applications by reusing existing IT resources including system components, services and databases on a cloud platform. On the other hand, to meet the continuously changing personal requirements for business purpose, these mobile applications are required not only to be developed rapidly, but also to be adjusted easily. The later requirement is especially important because related personal or business requirements are always changing.

However, different from other distributed IoT applications, Mobile Services in cloud computing are restricted by very limited resources which bring out some important considerations as following:

(I) Data contents and related I/O operations are the main consideration for mobile services development. Complex logics such as image processing or scientific calculation are less used in such situation, or these computing-intensive tasks are always carried out in cloud platform other than front devices with limited computing
ability.

(2) Resources are limited both from available services and IT components. Therefore services are limited in a certain application domain. And the developing process of a mobile service is somewhat in a lightweight service composition or component configuration manner on a relatively simple business requirement.

(3) Cooperation between mobile devices is very important for complementing intelligent interaction. Contextual data are required to support decision making from an interaction from different devices. And rules for dynamic behaviors are important for intelligent applications.

Research challenges in mobile application development cover different levels from identification and communication to distributed system and distributed intelligence [4]. Due to the highly complex system and a huge quantity of heterogeneous data exist, related information and relations are too complicated for data representation and processing. Information integration and intelligent interaction cannot be archived effectively by means of data integration. From the view of data processing, these challenges could be divided into different stages of data abstracting, data gathering, data integration, and intelligent interaction. Therefore IoT application not only needs to realize data integration but also requires realizing intelligent interaction [5].

Aim to facilitate mobile services development so as to rapidly construct an intelligent application with adaptive features when changes occurs, a model-driven service configuration architecture is provided for web-based mobile application development in cloud computing. Starting from the point of resource configuration of cloud platform, a meta-model covering multi-view business models and system components is provided for model abstraction and management. Based on a formal representation language, automatic model transformation and service integration are realized in three patterns. Furthermore, a related development platform is also provided for verification.

The contributions can be concluded as following points:

(1) A united meta-Model which integrates CIMs and PIMs is proposed as a referenced structure to encapsulate and manage business and IT resources for application implementation in cloud platform. The meta-model acts as a bridge to connect business requirements and executable components. Thus, by modeling process, function, organization, and data, different views of business information are connected with executable services by a united information framework.

(2) Three development patterns of Role-driven, Data-driven, and Process-driven are given as service configuration. Based on common service resources in cloud platform, these patterns provide a best practice for rapid mobile service development. Since formal language is also introduced in the process, the adjustment of mobile application based on three patterns can be carried out quickly by means of relation reasoning.

(3) Dynamic behaviors between different mobile devices and users can also be configured based on the semantic relations between IT components. By means of resource-based contextual structure, intelligent interaction can be realized for complex application with adaptive actions.

This paper is structured as follows. Related works are described in Section 2. Then a framework for development mobile application in cloud platform is given in Section 3. Section 4 presents and discusses details in the information meta-model, and the method consisted of information configuration, system configuration and interaction configuration are then given. The development platform is demonstrated and discussed in Section 5. Finally, the conclusion is given in Section 6.

2 RELATED WORK

There are many ideas and technologies for end users to develop and customize mobile applications. On the view of development process, we divide related researches into three areas: service modeling, system configuration and service execution.

2.1 Service modeling with contextual information

Context is used to represent various issues in mobile environments. Contextual data are usually gathered through distributed and heterogeneous sensors of IoT applications. These data form the basis for information integration and reasoning purpose.

S. De [6] presented a semantic modeling approach for service modelling for different IoT components. Associations between physical entities and services provided through devices are also given in the frameworks. H. Zhu [7] proposed an approach to enrich the contextual information of mobile Apps, then constructed a classifier so as to realize intelligent application based on user preference understanding. Taherkordi [8] proposed a framework-based middleware to manage contextual information of distributed nodes. These nodes containing context information are processed by means of five components, which are Context Process, Context Reasoning, Context Configuration, Activity Manager and Message Manager. KASOM [9], which represents Knowledge-Aware and Service-Oriented Middleware, was proposed to offer advanced and enriched pervasive services. And user interfaces with semantic interaction descriptions [10] were proposed to generate user interface for smart devices. It is a model-based interface description scheme to describe behaviors of devices. A Service-Oriented Context-Aware Middleware architecture named SOCAM [11] was proposed to support the acquisition, discovery and interpretation of various contexts for building context-aware services. Adistributional statistical model [12] of semantics was used to present an approximate model so as to divide the semantic coupling dimension for events processing from real-world smart city.

However, most contextual model construction still focuses on data integration instead of semantic integration, which contained both complex business requirement and execution environment.

2.2 System configuration

In system configuration, MDA (Model Driven Architec-
service composition, so as to meet ever-changing environment. By mapping concepts to actual information in a distributed environment, S. Hallsteinsena [28] aimed to propose a model-driven development framework with several adaptation mechanisms, which gives a clear separation of business logic, context awareness and adaptation concerns. Kiev Gama [29] applied SOC principles to integrate heterogeneous services for leveraging the existing IoT devices, with the purpose of realizing dynamic evolution of related applications. By means of an event process language (EPL), context-aware template method [30] was proposed to enhance business performance in the domain of cold chain logistical system. By means of service registering with the semantic description and service searching the required expectations, a new service model [31] was proposed to implement a semantically enhanced cloud service discovery platform and application. SMART [32] integrated all key activities into one platform, covering development, deployment and distribution, management, and subscription and accounting for service-based mobile applications. And a semi-automatic development model and environment [33] was used to provide necessary components for device-side mobile service interface and cloud-side service content.

Thus, due to some environment constraints, the main consideration of service execution in cloud platform may vary from operational cost, reliability, energy consumption [34] or others. Semantic disposing methods provide a promising manner when the number of services and data are amassed in the cloud platform.

In short, existing approaches provided a popular supporting architecture that focus on service modeling and application integration, but not a comprehensive solution to construct and execute an intelligent application. Mobile application construction in cloud platforms requires not only data acquiring/transferring but also semantic integration of both web component resources and business elements. Therefore, a comprehensive semantic solution covering the whole process of mobile application development still needs to be provided. Our approaches focus on introduce model-driven service composition disposing method into the process of information modelling, information configuration and information interaction in semantic level but not data level so as to facilitate mobile services development.

3 FRAMEWORK

For the purpose of supporting both data integration and intelligent interaction for mobile application, some major requirements are given from information processing phases such as device identification, abstracting, data transformation and integration and interaction.

A general development for IoT application can be described as several steps. (1) In the data acquirement stage, different comprehensive identification methods should be provided to identify things in a variety of processing conditions. Not only RFID (Radio Frequency Identification), but also 2D barcode, DPM (Direct Part Mark), and other
identification methods could be taken into account. (2) Data gathering from devices are always heterogeneous and distributed. It is always necessary to build an abstract information model so as to eliminate heterogeneity. Therefore, a comprehensive information abstraction considering different devices, application requirements, and working environment is an important base. (3) Interaction or cooperation between devices and clients is needed for a certain business goal in complex environments. Thus a contextual-aware model is important in the execution period so as to achieve a dynamic intelligent interaction.

Therefore, based on the requirements covering different disposing stages, a mobile application implementation framework using Model-Driven Development Patterns is proposed, as is shown in Fig. 1. The whole framework can be divided into three phases as follows.

![Framework Diagram](image)

Fig. 1. Mobile application implementation framework

1. Information Configuration. Based on information model, information of IoT objects is attained and represented according to an abstract structure. By means of multi-view business models, a united view related to a certain domain is used to manage different models.

2. System Configuration. According to typical architecture, business models are transformed into service components so as to construct a completed system. And different patterns are carried out to realize information configuration, process integration in semantic level.

3. Interaction Configuration. According to business requirement, related application scene with contextual information is constructed based on resources, which forms a logical runtime environment. Based on state space, ECA (Event-Condition-Action) rules are used to define actions based on related resources in the interaction configuration stage. State space of related resources is applied to construct an execution control mechanism in the end of interaction configuration stage.

In the stage of system configuration, MVC (Model-View-Controller) [35] is taken as a logic reference for service component integration. Related CIMs are grouped as three types of components, including (a) UI type components such as user profiles, Widgets and Gadgets, (b) Controller type components representing the control flow of business logic such as the business process which consists of functions, rule connectors, roles and related relationship between these elements, and (c) Data type components such as O/R mapping objects and data entities which are stored in database or data sources. And PIMs in web-based systems are represented as Forms/webpages, data connections, SOAP and RESTful services so as to be connected to related components in CIMs.

Based on existing resources of a certain cloud platform, the core task of MDA focuses on model transformation from CIMs to PIMs and PSMs so as to develop a runnable system to fulfill the business requirement of enterprises. After CIMs and PIMs are grouped based on the reference MVC framework, three patterns are then proposed as driven elements related to business requirements from different levels. Based on different types of components as a main driven element, three patterns are listed as follows:

- Role-driven pattern. Role-driven pattern is presented when roles and UIs are defined first as requirements. In such situation, Use-Case Diagram that represents role and functional relations is always used in the process of development.

- Process-driven pattern. Process-driven pattern is presented when business process is defined first as requirements. In such situation, Process Diagram such as BPMN (Business Process Modeling Notation) and EPC (Event-driven Process Chains) and Activity Diagram that represent control-flows of tasks are always used in the process of development.

- Data-driven pattern. Data-driven pattern is presented when data model is defined first as requirements. In such situation, Class Diagram or Entity-Relation Diagram, which represents information entities and relations, is always used in the process of development.

Three patterns provide a reference for system configuration. They are applicable to different types of applications such as structural functions, workflow-based systems and data-centric systems.

And based on semantic integration, changing management is also constructed for system evolution so as to meet the changing requirements for enterprise to adapt to external environment.
4 MODEL-DRIVEN DEVELOPMENT PATTERNS

In the framework, CIMs and PIMs are core models for software requirement definition. Thus a meta-model combining business models with system components is given for model representation firstly. Then Information configuration is given for complex information representation. Aim to realize system configuration, three patterns are proposed by means of formal methods. Lastly, interaction configuration is given in this section.

4.1 Model and model relations

The meta-model, which covering both business description and service components, provide a fundamental representation for application development. Thus, a unified reference meta-model combining CIMs with PIMs is given, as Fig. 2 shows. A meta-model is a multi-view model consisted of four views models. In the meta-model, DataView models, UserView models, and ControlView models are CIMs designed for business description, and SystemView models are PIMs which are used for system development. Combine these models together, a meta-model combining is constructed for models integration, which integrated both CIMs with PIMs into a united view. Relations between these models could be used to carry model transformation, and semantic integration.

In the ControlView Models, Functional Model (FM) is the basic element to represent atomic logical task. It is defined as the following tuple:

\[ \text{FM} = \langle \text{IOR}, \text{PRE}, \text{POE}, \text{RSS} \rangle \]  

In it, IOR (Input and Output Resource) is a string list that contains input and output parameters of a function. The formula is IOR = (Input-Paras, Output-Paras). Both Input and Output can be a list of strings or a single string. IOR will be used to select appropriate resources from resource pools. PRE is defined as the pre-condition of a FM. POE is defined as the post-condition of a FM. RSS (Related SOAP Services) are related to one or many SOAP services.

Definition 3: RoleModel (RM)

RoleModel = <RoleID, Positions, Users, OrgUnit>  

In it, Positions belong to a certain Organizational, Users refers to one or more certain user, and OrgUnit represent Organizational Unit used as authority or management purpose, it could be used as a tenant in Cloud platforms.

Definition 4: ProcessModel (PM)

ProcessModel is core model for dynamic behavior description, which has a start and one or several end to realize a process. It is defined as the following tuple:

PM = <ProID, Status, Tasks, Event, Connector, Permission>
Definition 6: Semantic Relationship (SR)

SR = <RDU, RUC, RDC, BusinessLogic> (6)

- In it, RDU (Relations between DVMs and UVMs) represent the relations between DVMs and UVMs. It is used to assign data access authority to a certain user especially by means of RESTful API.
- RUC (Relations between UVMs and CVMs) represent the relations between UVMs and CVMs. It is used to realized system authority between role and function.
- RDC (Relations between DVMs and CVMs) represent the relations between DVMs and CVMs. It is used to connect data state with a certain function.
- BusinessLogic is a true subset of SR. It is defined as a set of relations between FM, PM, RM and ERM, which can be used to represent relations for further system configuration.

Some relations between models are given in table 1.

4.2 Information configuration

For the purpose of information processing, attributes, states, and storing information should be represented. Thus, ERM is used as an abstract structure for device representation firstly. Based on ERM model as the basic information structure, device data are abstracted and acquired form an information source for further disposing process.

Business information such as Order, Inventory, and Contract, is always with complex data structure and of complex relationship with more than one independent data sources. For this reason, information configuration should be carried out to an available business structure. Therefore, based on independent ERM, composed resources are constructed for representation of business information. As shown in Fig. 3, a loose-coupling structure is thus constructed for representation and disposing of complex business information.
Since ERMs are connected to different data sources independently, the relationship of data sources is maintained inner the model when we configure multiple ERMs so as to construct a cRM for complex business information such as a contract or an order.

Definition 7: Composed Resource Model (cRM)

A composed Resource Model, just as its name implies, is a kind of resource contains at least one other resource. A cRM not only contains URI, ASet, SSet parts as an ERM does, but also contains combination relationships to other resources. Composed Resource Model can be denoted by the following expression:

\[ \text{cRM} = \langle \text{URI}, \text{ASet}, \text{SSet}, \text{RSet} \rangle \]  

In it, URI, ASet, SSet have the same meanings with those in the meta-resource model. \( \text{RSet} = \{ \text{R}_1(r_1), \ldots, \text{R}_n(r_n) \mid \text{C}_1, \ldots, \text{C}_m, \text{R}_1(r_1), \ldots, \text{R}_n(r_n) \} \).

RSet represents combination relationship for business purpose by means of composing resources. Here, \( \text{R}_1, \ldots, \text{R}_n \) represents the three kinds of combined resources and \( \text{C}_1, \ldots, \text{C}_m \) represents the additional independent attributes as in some cases we may need to add some attributes that are not resources on the combination resource. \( r_1, \ldots, r_m \) following after \( \text{R}_1, \ldots, \text{R}_n \) means the relationship between combination resource and referred resources.

Composed resource is constructed based on related resource, which is a process of information configuration. The composed resource consists of several ERMs to form a tree structure. The relationship of Association and Composition is used to connected resource nodes.

According to the relation between the combination resource and the referred resource, the combination methods can be defined as three types of relations as ISA (is-a), Association and Composition. The combination operation can be represented with the following formula:

\[ \{ \text{R}_1(r_1), \ldots, \text{R}_n(r_n) \mid \text{C}_1, \ldots, \text{C}_m, \text{R}_1(r_1), \ldots, \text{R}_n(r_n) \} \rightarrow \text{cRM} \]

Therefore, related business information construction can be carried out with a varied-granularity implementation. All resources, whatever ERMs or cRMs, can be invoked by means of URI in RESTful services, which help to achieve STATELESS features in distributed environment.

4.3 System configuration

After the entity resource models are encapsulated from mobile devices, and complex structures for business purpose are composed. Then, we can develop a complete application referring MVC framework containing data, business logic and UI by means of different patterns driven by different models. It is a model configuration process based on existing models.

(1) Role-driven pattern

Role-driven Pattern, also named as structural function driven pattern, defines requirements from roles and functions and related relations of them. It is a common pattern for web-based application development. The core of this pattern is to implement service transformation from functional requirement.

For the purpose of implementing automatic model-driven application development, the steps of Role-driven pattern are given as follows:

- Based on roles, describe business logic based on FMs (Functional Models), construct detailed Event flow so as to decompose FMs. Use-Case diagram is used to describe requirement in this phase.
- Design UIs or webpages for each role. Based on FM, develop or encapsulate SOAP services;
- Design ERMs based on FMs, generate the database based on ERM, and construct a system from UI, SOAP services and the related database.

(a) Roles-driven Pattern.

(b) Process-driven development pattern

(c) Data-driven Pattern.
About requirements, role authority and process are involved directly in this process. ERM is related to FMs after they are decomposed.

For the purpose of model transformation, a formal representation and transformation are then implemented based on linear logic. Linear logic [36] is a sub-structural logic proposed by Girard as a refinement of the classical and intuitionistic logic. It is a consensus that linear logic lends itself to many different presentations, explanations and intuitions. In addition to various given concrete development steps, we use classical linear logic (CLL) to formally describe the specific rules.

In order to implement application based on the pattern, mapping mechanism from roles to data, and from process to function is the core of model transformation. Relations from roles to data are defined in business modeling stages. Construction of FMs is a critical step, which depends on User-case analysis. The information we can get refers to the user’s inputs, outputs and even required sequence of the operations. Design of ERM based on FM is easy but some artificially defined strategies are needed as Δ in the formula. And the remaining steps of this role-driven pattern could be expressed likewise.

The changing management mechanism in the pattern is described as follows:

- It is easy to re-allocate role authority, which is a relationship between roles and services.
- Business logic related to processes and functions is also easy to implement by means of workflow engine.
- Relations between function/SOAP services and ERM can be implemented by O/R mapping mechanism, which are easy to allocate if ERM is the same. However, it is relatively more complex than others.

In fact, this pattern fits with the structural system that can be developed according to the top-to-down strategy.

(2) Process-driven pattern

Process-driven Pattern defines requirements based on business processes. The core of this pattern is to integrate existing services so as to execute a certain process.

For the purpose of implementing automatic model-driven application development, the steps of Process-driven pattern are given as follows:

- Based on the business process PM, decomposed FMs to find tasks. BPMN diagram is used to describe requirement in this phase.
- Based on the relations between roles and functions, FMs are used to allocate tasks to roles, UI forms or web pages for each role are also developed;
- Relations between FM and ERM are used to find related data sources. SOAP services are generated based on FMs, according to data access requirement, RESTful services and data connections are implemented based on ERM. Related database is generated based on ERM lastly.

Thus, by mean of process analysis, related models are attained by model relations reasoning. After services are attained from service repository, web pages are generated, and database connection generation, a completed application consisted of UIs, functions, and data connection is thus constructed.

In order to implement rapid development based on this pattern, model transformation from PM to FMs is the core of this process, then FMs are allocated to roles and data. These two relations can also be defined in business modeling stage. In fact, this process-driven pattern is a task-based service composition that fits with service re-use.

The changing management mechanism of element in this pattern is described as follows:

- Role authority is the relationship between roles and services, which is easy to re-allocate.
- Business logic related to processes and functions is easy to implement by means of workflow engine.
- Relations between function and SOAP services are the core for service re-use.
- Relating functions to ERM is complex. Here mapping from function to ERM is used, but the data connections lack united view.
- ERM to RESTful services and data connections are easy to configure.

(3) Data-driven pattern

Data-driven Pattern defines requirements based on existing database or data. The core of this pattern is to carry out information extraction, encapsulation, configuration and publication of data-centric service based on business information requirements or existing database.

The pseudo code is given as follows:

The pseudocode of data-driven pattern

BEGIN
Input dataEntity ERM;       // Input streams of structured data
Call DeviceAbstract(devices); // To build abstract models for devices

// STEP1: Information Configuration
Call DataAnalysis(ERMs);     // Attain all dataEntity of a process
    dataEntity = ERM;       // To archive a ERM
    InformationPool.AddStates(dataEntity) // To add state of data
while (ERM ∈ ERMs) {        // Combine ERMs into an Information Pool
    InformationPool.AddStates(dataEntity) // To add state of data
}

// STEP2: Services Transformation
Call InformationConfiguration(ERMs); // To configure ERMs
Call RelationReasoning(ERMs);       // To find related models
while (ERM ∈ ERMs) {                // To find related service for each ERM
    GenerateWSDL(service); // To generate service in WSDL files
    GenerateUI(WADL); // To generate webpage based on WADL
    MappingServiceToERM (service, ERM) // Connect service to a entity
}

// STEP3: Application Configuration
Call ScenarioConstruction(ERMs); // Scenario Construction
Call GenerateDB(ERMs); // Generate database
Call MappingServiceToUI(service, targetUI); // UI Bounding service
Call MappingUIToRoleUI(UL, ERMs); // UI-based authority generation
SetBusinessLogic(roles); // Setup rules for Interaction
Call ApplicationPackage; // Package application
END
For the purpose of implementing automatic model-driven application development, the steps of Data-driven Pattern are given as follows:

- Based on device abstraction and information requirement, model ERMs. Class diagram is used to describe requirement in this phase.
- Based on each ERM, combine ERMs to construct an information pool as business logic which act as a united access module, then generate RESTful services and related database; 
- Based on FM related to information pool, generate UIs and web pages, then according to user authority of RM, allocate web pages to certain users; 

By means of automatic model transformation, data-driven pattern will help generate data-centric services or application rapidly.

According to the requirement, firstly we need to model related data entities, and then business logic is described in service composition based on data-centric services. The pattern is useful for data-centric application such as information services. A simple operation, for example attaining and updating information, is implemented through GET, PUT, POST and DELETE operations of REST.

In order to implement rapid development based on models, model transformation from processes to data, and from data to role models is the core of this pattern. Relations between data and roles are data authorities defined in the business modeling stage. Relationship between processes and roles is the core of this stage. The relations from processes to data should be defined as constraints in this pattern.

In such situation, requirement changing is mainly about ERM, role authority and business logic related to service composition. The changing management mechanism in the pattern is described as follows:

- ERM changing is easy to dispose for the reason that services are loose-coupled with data. We can use add-but-not-delete strategy for engineering applications. ERM changing is used to generate RESTful service and update database or data connections.
- User authority, in fact, is the relationship between roles and services, which is easy to re-allocate. Business logic related to service composition is relatively more complex to change than others. In most cases, service composition is required to construct new business logic. For the purpose of reducing development work, we build a big granularity service related to processes so as to implement a flexible service composition in cloud environment.

4.4 Interaction configuration

Aim to realize intelligent interaction, a contextual structure is needed to assemble and manage dynamic elements of a certain application scenario. Therefore, a model named ContextScene is proposed to represent business contextual information of a certain application environment. Fig. 5 shows the application scenario construction based on resource models.

Definition 9: ContextScene.

ContextScene is a contextual structure for the description of application scenario, which is represented as:

\[
\text{ContextScene} = (\text{Event, Time, Location, Actors, TaskSet, ResGroup}) \tag{9}
\]

In the definition above, Time and Location define when and where the scene occurs. Actors are the users who are responsible for the tasks in this scene. Tasks indicate the defined process with adaptive behaviors. ResGroup are connected with the scene.

ContextScene gives a runtime context for application execution. In this case, condition mainly depends on the value of Time, Location, and resource states. By means of event-condition-action (ECA), some rules for intelligent interaction can be designed. Generally events are generated from external environment. When an Event comes, based on ECA rule, the condition including Time, Location and other factors will decide which actions can be activated. And the order of execution can also be archived based on the defined process with adaptive behaviors.

5 Platform development and case study

Based on the patterns above, a software platform is given for mobile application development. Then the approach is compared with other approaches to show its feasibility.

5.1 Software architecture of the platform

For the purpose of developing mobile service development, related services and models are constructed as resources and then stored in database. Thus a development platform is provided based on the above patterns, as Fig. 6 shows. The platform mainly consists of three modules, which are Information Modeler, Resource Repository and Runtime Environment.

(1) Information Modeler for device encapsulation
Information Modeler supports multi-views of business modelling so as to abstract and encapsulate devices from distributing and heterogeneous information environment. Information Modeler is used to build the functional model, the resource model and the process model, as well as the united view for the purpose of managing these models in a semantic level.

(2) Resource Repository for information configuration
Resource Repository is designed to connect Information Modeler and Runtime Environment. Based on ontology, these resources are organized and managed with semantic relationships. Thus resource models from different sources are mapped and configured for further use like service transformation or information configuration.

(3) Runtime Environment for interaction configuration
Runtime Environment is used to configure application environment and rules for execution of a business application. When services are generated from Resource Repository, ECA rules are also defined before application execution. Thus based on the state space constructed by means of resource groups, the process will be executed to archive an intelligent and flexible aware environment.

Therefore, the platform can be used to develop mobile applications covering different design stages of SaaS development. The platform is used in a cloud-based platform for SaaS development, some enterprise applications such as logistical system, customer relation management system are realized based on the platform.

5.2 Case study
A business scenario from a sales management system is given to illustrate the process of mobile application development. As Fig. 7 shown, a business scenario for a mobile sale application is given.

There are two businessmen visiting one client for business purpose in this case. The mobile phone owned by the businessmen can get its position through GPS (Global Positioning System) services. When two businessmen are in a close area, the system will depart them from visiting one same client to avoid conflict. By connecting the database of historical data, then according to the certain rules, the system will find out which the businessman should leave and send him a notice automatically.

This scenario focuses on data access and decision-making, and no complex process or role structure is involved. Based on data-driven pattern, we will implement this business scene to mobile application by four steps.

Step 1: Resource abstraction
First, we build the ERM connecting each data source in the cloud platform. As Fig. 8 shows, there are two single resources such as Partner and Businessman in the case. The levels of partner and businessman, which will be considered for further disposing, are the states in this case. Each of them has a single table in the database. These resources form an information base for further disposing process.
entities in this case: Has, Based, SimilarTo. And there are four semantic reasoning rules between the relations:

Rule 1: Partner has visit history, so if a partner is similar to another partner, then their visit histories are inferred to be similar.

Rule 2: Businessman has visit history, so if a businessman is similar to another businessman, then their visit histories are inferred to be similar.

Rule 3: Visit history has task, so if one visit history is similar to another visit history, then their tasks are inferred to be similar.

Rule 4: Visit history has document, so if one visit history is similar to another visit history, then their documents are inferred to be similar.

Based on these rule, information composition are then carried out to archive CRM VisitHistory.

Step 3: System configuration

Based on the ERMs, related RESTful service files are generated in WADL format, these WADL files are then used to generate UIs under predefined UI style such as CSS automatically. And an information pool contained four resources of VisitHistory, Businessman, Partner, and Document are also generated as a RESTful API to connect the data source in the Cloud platform.

Some system component related to this scene is given, as Table 2 shows.

<table>
<thead>
<tr>
<th>TABLE 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>System component type</td>
</tr>
<tr>
<td>RESTful services</td>
</tr>
<tr>
<td>SOAP services</td>
</tr>
<tr>
<td>Roles</td>
</tr>
<tr>
<td>UIs/WebPages</td>
</tr>
<tr>
<td>Relations</td>
</tr>
</tbody>
</table>

Step 4: Interaction configuration

In this step, we define BusinessVisit as an application scenario with contextual information, related tasks, actors, and resources involved, as Table 3 shows:

<table>
<thead>
<tr>
<th>TABLE 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
</tr>
<tr>
<td>Tasks</td>
</tr>
<tr>
<td>Event</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>Location</td>
</tr>
<tr>
<td>Actors</td>
</tr>
<tr>
<td>ResGroup</td>
</tr>
</tbody>
</table>

Then, for the purpose of avoiding two businessmen visiting the same partner, we defined three rules for decision-making as below:

Rule 1: When the two businessmen have different levels but the level difference between each businessman and the partner is the same, the businessman with a higher level should go to the partner’s office while the one with a lower level should leave and visit another partner.

Rule 2: When the two businessmen have same level, the businessman who came more rarely before should go to the partner’s while the one who came more frequently should leave and visit another partner.

Rule 3: When the two businessmen have different levels, the businessman with a level closer to the partner should go to the partner’s while the one with a level farther should leave and visit another partner.

Related ECA rules are given as follows:

```
Event Position conflict occurred
Condition 1 businessman1.level ≠ businessman2.level and abs(businessman1.level - partner.level) = abs(businessman2.level - partner.level)
Action SentNotice businessman with lower businessman.level to leave
Condition 2 businessman1.level = businessman2.level
Action SentNotice businessman with bigger businessman.frequency to leave
Condition 3 businessman1.level ≠ businessman2.level
Action SentNotice businessman with bigger abs(businessman.level - partner.level) to leave
```

Using these rules, the services can cooperate with each other and find out which businessman should visit the partner. Finally, we deploy all these resources and rules to the server, and thus implement a mobile application. An interface is given, as Fig. 10 shows.

Fig. 9. Composed Resource Model of VisitHistory

Fig. 10 An Interfaces of mobile sale management system
In fact, real mobile applications are more complex than this scene. Intelligent and adaptive dynamic mechanism can be constructed by semantic reasoning based on contextual scene which contains process, tasks, actors and related resource models. We do not expand the details here. The case is an illustration of data-driven pattern. In fact, different patterns have different features. Three patterns are compared from elements and features, as shown in Table 4.

**TABLE 4**  
**COMPARISON BETWEEN DIFFERENT PATTERNS**

<table>
<thead>
<tr>
<th>Features</th>
<th>Data-driven pattern</th>
<th>Roles-driven Pattern</th>
<th>Process-driven pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driven element</td>
<td>Data</td>
<td>Role</td>
<td>Process</td>
</tr>
<tr>
<td>Purpose of Pattern</td>
<td>Application development</td>
<td>Application development</td>
<td>Service Integration</td>
</tr>
<tr>
<td>Relationships</td>
<td>Process-data</td>
<td>Role-function</td>
<td>Process-function</td>
</tr>
<tr>
<td>Service transformation</td>
<td>Data-RESTful</td>
<td>Function-SOAP</td>
<td>Function-SOAP</td>
</tr>
<tr>
<td>Supported IT component</td>
<td>RESTful</td>
<td>SOAP</td>
<td>Both SOAP and RESTful</td>
</tr>
<tr>
<td>Adaptive for Changing</td>
<td>Data</td>
<td>Role-function</td>
<td>Role-task</td>
</tr>
<tr>
<td>Role-data</td>
<td>Process-function</td>
<td>Process-function</td>
<td>Function-data</td>
</tr>
<tr>
<td>Automatic degree</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Flexible</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Development difficulty</td>
<td>Medium</td>
<td>Easy</td>
<td>High</td>
</tr>
<tr>
<td>Development period</td>
<td>Very short</td>
<td>Short</td>
<td>Long</td>
</tr>
<tr>
<td>Development cost</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>

From the table, we could conclude that different patterns are applicable to different types of applications. Roles-driven Pattern could be used in structural functional application such as enterprise management systems with a strict or clear roles and functions. Process-driven Pattern could be used in workflow-based systems such as web-based position apply application. And Data-driven pattern is powerful when we need to implement data-centric systems such as web-based content management system.

### 5.3 Comparison and Discussion

On construction of cloud-enabled applications based on existing resources, service composition and lightweight mashup are two typical methods. Therefore, we compare our approach with these two types of methods as follows.

An ontology-based approach [37] for home-based scenarios provides a typical IoT application implementation based on service composition. By integrating data and its management procedure so as to realize remote management tasks at home sites, the approach facilitates the integration of several management services at home sites which involve not only clinical measurements but also the patient’s context.

Mashup [38] provides a kind of common lightweight service composition for web-based application development. Based on existing Web API or small components, it fulfills the individual and heterogeneous integration requirements of end users and fosters End User Development in a visual interactive way. It provides a typical development for web-based applications.

From the aspects of design, execution and features, a comparison between our approaches with these two methods is given in Table 5.

**TABLE 5**  
**COMPARISON WITH OTHER SAAS METHOD**

<table>
<thead>
<tr>
<th>Features</th>
<th>Our approach</th>
<th>Ontology-based approach [37]</th>
<th>Mashup[38]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build-time structure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supported data sources</td>
<td>Distributed</td>
<td>Distributed</td>
<td>Distributed</td>
</tr>
<tr>
<td>Data abstraction</td>
<td>One-one mapping</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Data management</td>
<td>Semantic networks</td>
<td>Tree structure</td>
<td>No mention</td>
</tr>
<tr>
<td>System configuration</td>
<td>Varied-granularity model configuration</td>
<td>Information configuration</td>
<td>Functional component configuration</td>
</tr>
<tr>
<td>Ontology structure</td>
<td>OWL</td>
<td>OWL</td>
<td>none</td>
</tr>
<tr>
<td>Data integration</td>
<td>Heterogeneity data integration</td>
<td>K73standard</td>
<td>N/A</td>
</tr>
<tr>
<td>Runtime structure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Context construction</td>
<td>Ontology-based business scene</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Interactive rules</td>
<td>ECA rules</td>
<td>SPAQL</td>
<td>No</td>
</tr>
<tr>
<td>Status monitoring</td>
<td>State space</td>
<td>Single task</td>
<td></td>
</tr>
<tr>
<td>Interactive integration</td>
<td>Process</td>
<td>Process</td>
<td></td>
</tr>
<tr>
<td>Executed component</td>
<td>RESTful services</td>
<td>Functional components</td>
<td>Wiget/Caget</td>
</tr>
</tbody>
</table>

### In order to evaluate proposed approach, we will discuss the platform from the following aspects:

1. **Functionality**

The functionality involves the data, business logic, and user interfaces to realize applications in design period. From the table, we can find that business requirement is
defined based on different views of business models rather than IT components, and then services are generated and integrated based on these business models or CIMs. Compared to Mashup or service composition, our approach provide a flexible information configuration so as to fulfill the requirement of practice applications.

(2) Adaptability
The adaptability involves the component changed management to support applications execution in runtime period. From the table, we can find that business scene is used to configure dynamic rules for different conditions so as to realize intelligent interaction. Thus our approach provide a flexible control mechanism so as to adapt to the change of execution environment.

(3) Usability
The Usability involves the invoke manner when an application is configured with services. From the table, we can find that Semantic-based reasoning is used to configure and manage model relations so as to realize intelligent interaction. And our approach provide a RESTful API and related data source so as to provide an easy access manner for web-based applications.

In whole, the platform is more suitable to implement services that involve complex relations covering processes, data, and functions with a high automatic level. The platform realize high functionality by means of information configuration and contextual application scenario configuration so as to fulfill the requirement of practice applications. Also, the platform provides great convenience for modifying or creating new applications when requirement changes. Therefore, with high usability and adaptability, the platform reduces developing and adjustment cost and it is also more suitable in the situations of semantic information compared with the two approaches.

7 Conclusion
The convergence of Cloud and IoT can provide huge opportunities because the applications are location independent, and the users can access the cloud services from any location and with any mobile devices through the Internet connection. Aimed to provide a systemic integration architecture to develop applications easily and adaptively, we propose a model-driven service configuration platform which supports semantic reasoning. There are mainly three points:

- In the process of designing mobile service, three patterns are proposed to transform requirements to a mobile application. Thus a rapid development is archived.
- In the process of execution, contextual information is added by means of ECA rules. Therefore, mobile applications can be easy to cooperate with other devices so as to realize intelligent interaction.
- In the process of adjustment, applications are easy to change by means of relation reasoning of models.

The solution provides not only a semantic-driven method across different stages such as device abstraction, information configuration, contextual construction and acting rules for interaction integration, but also a semantic conception and reasoning mechanism for further application related to big data.

Further research will focus on ontology evolution so as to provide a continuously mechanism for service governance. And further works will focus on adaptive dynamic execution after application deployment, especially for cooperation among different types of devices.

Acknowledgment
This paper is supported by the National Natural Science Foundation of China under Grant No. 61373030, 71171132.

References


Hongming Cai received his BS degree, MS degree, Ph.D. degree from Northwesrern Polytechnical University, China in 1996, 1999 and 2002, respectively. He served as a postdoctoral research fellow at the Computer Science and Technology Department of the Shanghai Jiao Tong University, China during the period of 2002 to 2004. And he served as a visiting professor at the Business Information Technology Institute of University of Mannheim, Germany during the period of 2008 to 2009. The visiting scholarship was appointed and sponsored by Alfred Krupp von Bohlen und Halbach Foundation, Germany. He is rewarded as a “National outstanding scientific and technological workers” by China Association for Science and Technology (CAST) in 2012. Now he is an associate professor, School of Software, Shanghai Jiao Tong University, China. He is a standing director of China Graphics Society, a senior member of ACM/IEEE, and a senior member of China Computer Federation.

Yizhi Gu received the B.S. degree in software engineering from Shanghai Jiao Tong University, China, in 2014, now he is a postgraduate student in School of Software, Shanghai Jiao Tong University, China.

Athanasios V. Vasilakos received the B.S. degree in electrical and computer engineering from the University of Thrace, Xanthi, Greece, in 1983, the M.S. degree in computer engineering from the University of Massachusetts, Amherst, in 1986, and the Ph.D. degree in computer engineering from the University of Patras, Patras, Greece, in 1988. Athanasios V. Vasilakos is recently Professor with the Lulea University of Technology. He served or is serving as an Editor for many technical journals, such as the IEEE Transactions on Network and Service Management; IEEE Transactions on Cloud Computing, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Cybernatics; IEEE Transactions on Information Technology in Biomedicine; ACM Transactions on Autonomous and Adaptive Systems; the IEEE Journal of Computer and Engineering. He is a senior member of China Computer Federation.

Boyi Xu received the B.S. degree in industrial automation and the Ph.D. degree in management science from Tianjin University, Tianjin, China, in 1987 and 1996, respectively. Currently, he is an Associate Professor at the College of Economics and Management, Shanghai Jiao Tong University, Shanghai, China. His research interests include enterprise information systems, Internet of things, and business intelligence.

Jun Zhou received the B.S. degree in software engineering from Shanghai Jiao Tong University, China, in 2014, now she is a postgraduate student in School of Software, Shanghai Jiao Tong University, China.