2-Tier Cloud Architecture and Application in Electronic Health Record

Wenjun Zhang
Research Institute of Applied Computing Technology, China Women's University, Beijing, China
Email: voicefromzhwj@yahoo.com.cn

***Abstract*—We find that there are serious problems in Cloud application development such as complex architecture and WS API, highly running cost, poor UI and interaction and so on, and that these problems stem from not fully utilized RIA's (Rich Internet Application) advantages and inappropriate functionality segmentation between client-side and server-side. That is, Web services, application logic and transaction logic are overly concentrated on the server side; while on the client-side computing power has not been fully utilized. We propose a novel 2-Tier Cloud-ARchitecture (2TCAR), which contains RIA-based rich client tier and SimpleDB-based server-side Cloud tier. The rich client tier is maximized to implement most of functionalities of Cloud application and transaction logic; in contrast, the functionality in server-side Cloud tier is minimized to only implement data storage and query. The communication between these two tiers is also simplified via REST. In the article we researched corresponding technologies such as Cloud computing, Web services, REST, Flex in RIA and SimpleDB storage Cloud. In addition, we proposed how to use Flex to implement UI presentation & interaction, transaction logic, REST requests & responses in rich client tier; and we described how to design SimpleDB Cloud in server-side Cloud tier and communication between two tiers via REST. Last, a Cloud application system, which is called Cloud System of Electronic Health Records (EHR) for Orthodontics, is developed based on the research findings illustrated by the author in this paper. It has been shown that the 2TCAR is effective and valuable for Cloud application development.**

***Index Terms*—2TCAR, Cloud Computing, EHR, Flex, EMR, RIA, SimpleDB**

I. INTRODUCTION

Cloud computing is an area that is experiencing a rapid advancement both in academia and industry. This technology, which aims at offering distributed, virtualized, and elastic resources as utilities to end users, has the potential to support full realization of "computing as a utility" in the near future [1]-[5].

Along with the advancements of the Cloud technology, new possibilities for Cloud-based applications development are emerging. These new application models are mainly based on HTML on the client side, complex Web Services on the server side and their APIs based on SOAP and so on in Figure 1. And almost all components such as Web services, application logic, transaction logic and data storage, are deployed and executed in the back-end Cloud servers. This traditional Cloud application architecture results in many problems [5]: 1) Web services APIs in Cloud system is too complex for developing Cloud application because all services need to be defined; 2) Running cost is too high because all transaction logic is executed in back-end Cloud server and much servers' CPU computation is consumed; 3) Interaction between client and Cloud server is frequent and of low efficiency because HTML Web pages on the client side do not contain the script which can run and interact with end-users independently, but only display content of Web pages; 4) The client-side requires data only through request and response session because all datum and states host on the Cloud server-side. The content from Cloud-servers contains not only data but also a lot of redundant format markup tags resulting in much Internet bandwidth to be consumed and wasted. 5) Unlike desktop applications, HTML Web pages are not equipped with multifunctional controls such as DataGrid, Tree, PieChart and so on. And the codes in presentation, transaction logic and data persistence layers on the Cloud server-side, are tightly coupled together so as to result in low reuse, high couple and difficult maintainability.

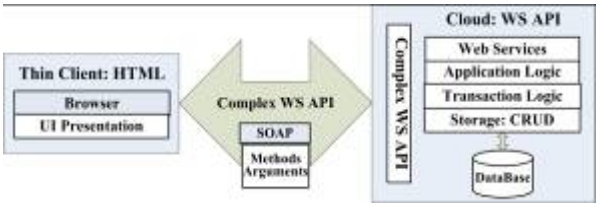


Figure 1. Traditional Cloud implementation architecture.

The above problems stem from not fully utilizing Rich Internet Application (RIA) and inappropriate functionality segmentation between client-side and server-side. That is, Web services, application logic and transaction logic are overly concentrated in the back-end Cloud; in contrast, PCs' computing power on the client-side is not fully utilized. In this article we propose a novel 2-Tier Cloud ARchitecture (2TCAR), which by maximizing the client-side functionality via RIA on the client-side. RIA with script codes implements original UI

The work was sponsored by the Natural Science Foundation of China Women's University under grants No. KG0903019 and Peking University School of Stomatology.

doi:10.4304/jsw.7.4.765-772

presentation, Web services, application logic and transaction logic; on the Cloud server-side functionality is simplified and minimized into only storing and querying data via Amazon's SimpleDB storage Cloud; the APIs interface between the RIA client-side and the Cloud-side is simplified from original complex APIs based on SOAP to minimized REST (REpresentational State Transfer).

Currently many countries, healthcare providers, medical practitioners are adopting Electronic Health Record (EHR) systems, and are building health record clouds underway to modernize health records systems for greater efficiency, improve patient care, patient safety, and costs savings [6]-[8]. The potential benefits from EHR, which include lab tests, images, diagnoses, prescriptions and medical histories, are without precedent. Providers can instantly access patient histories that are relevant to future care and patients can take ownership of their health records. Cloud computing provides an attractive IT platform to cut down the cost of EHR systems in terms of both ownership and IT maintenance burdens for many medical practices.

It is widely recognized that cloud computing and open standards are important cornerstones [9] to streamline healthcare whether it is for maintaining health records, monitoring of patients, managing diseases and cares more efficiently and effectively, or collaboration with peers and analysis of data. Many predict that managing healthcare applications with clouds will make revolutionary change in the way we do healthcare today. Enabling the access to healthcare ubiquitous not only will help us improve healthcare as our data will always be accessible from anywhere at any time, but also it helps cutting down the costs drastically.

Now for orthodontists there are no commercial orthodontic EHR Cloud systems suitable for their clinical needs in China, so we firstly study orthodontist's daily workflow and analyze the requirements. And according to orthodontist's workflow we apply the research findings about 2-Tier Cloud architecture to develop the orthodontic EHR Cloud for Peking University School of Stomatology, which contains Flex-based rich client tier and server-side Cloud tier based on SimpleDB.

II. Cloud Computing and Implementation Technologies

Implementing the 2-Tier Cloud architecture needs integrating many new Web technologies such as Cloud computing, RIA, Web services, REST, and SimpleDB Cloud. In the section we present them.

A. Cloud Computing [1]-[5]

Very simply stated, Cloud computing is the delivery of a service or capability over the network. Cloud computing is often segmented into three areas:

1) Software as a service (SaaS): Applications services delivered over the network.

2) Platform as a service (PaaS): A software development framework and components all delivered on the network. Offered as on-demand, pay for usage model.

3) Infrastructure as a service (IaaS): An integrated environment of computing resources, storage, and network fabric delivered over the network. Offered as an on-demand, pay for usage model.

The advantages of SaaS to both end users and service providers are well understood. Service providers enjoy greatly simplified software installation and maintenance and centralized control over versioning; end users can access the service "anytime, anywhere", share data and collaborate more easily, and keep their data stored safely in the infrastructure.

B. Service Oriented Architectures

A service provides business functions to its consumer and it is defined as "Distinct part of the functionality that is provided by an entity through interfaces". In particular, in computing terms, a service is an application that provides information and/or functionality to other applications. Services are typically non-human-interactive applications that run on servers and interact with applications via an interface.

Traditional SOA [10] is based on a protocol known as SOAP, whose strengths and weaknesses have been well-understood for a while now. SOAP is not Web-oriented for reasons too detailed and technical to go into here, but competing protocols such as REST are very clearly Web-oriented. SOAP is generally not a great protocol for Web-facing systems. This is one reason that Google got rid of its SOAP search API late last year and it's why Amazon offered it's very commercially successful APIs in both flavors (SOAP and REST) and the market place chose REST.

So we replaced SOAP-based Web services with REST to design 2-Tier Cloud architecture.

C. REST

Representational State Transfer (REST) [11] is a software architectural style for distributed hypermedia systems like the World Wide Web. REST has gained widespread acceptance across the Web as a simpler alternative to SOAP- and Web Services Description Language (WSDL)-based Web services. Key evidence of this shift in interface design is the adoption of REST by mainstream RIA or Web 2.0 service providers—including Yahoo, Google, and Facebook—who have deprecated or passed on SOAP and WSDL-based interfaces in favor of an easier-to-use, resource-oriented model to expose their services.

REST strictly refers to a collection of architectural principles: Principled Design of the Modern Web Architecture. The term is also often used to describe any simple interface that uses XML (or JSON, plain text) over HTTP without an additional messaging layer such as SOAP.

In the REST architectural style, data and functionality are considered resources, and these resources are accessed using Uniform Resource Identifiers (URIs). The resources are acted upon by using a set of simple, well-defined operations. The REST architectural style constrains architecture to a client-server architecture, and is designed to use a stateless communication protocol,

typically HTTP. In the REST architecture style, clients and servers exchange representations of resources using a standardized interface and protocol. These principles encourage RESTful applications to be simple, lightweight, and have high performance. RESTful Web services typically map the four main HTTP methods to the operations they perform: create, retrieve, update, and delete to implement storage and retrieval of data resource.

As we have seen, REST uses HTTP to support Web services in a simple fashion without the need to add new protocols, as long as the Web service is structured as a set of resources. So we use REST to design the communication interface between rich client tier and SimpleDB Cloud tier on the Cloud server side.

D. Rich Internet Application

Most of the current Cloud systems are B/S thin client application mode based on HTML pages. With increasing complexity this application mode is not longer able to meet the requirements of providing interactive and rich user experience.

In this section we'll be focusing on RIA structure. Examples will be drawn from Flex, but our conclusions should apply to any stateful RIA whether Silverlight or Javascript-based Ajax. Flex [12] is one kind of RIA technology, a framework for creating RIA application based on Flash Player. Its core is MXML, a markup language based on Extensible Markup Language (XML) that makes it really easy and efficient to create Cloud applications.

Flex offers highly visual, fluid, and rich experience and user interface components. When Cloud computing and Flex are integrated together in a Cloud application, the best of both worlds are combined. In Cloud SimpleDB provides the data storage via REST on the Cloud server-side, while Flex and Adobe Flash Player make the rich, dynamic user interfaces, application logic and transaction logic possible on the client-side.

1) Flex framework technology and development process: The Flex framework [13] shown in Figure 2 is synonymous with the Flex class library and is a collection of ActionScript classes used by Flex applications. The Flex framework defines controls, containers, and managers in order to simplify the process of building RIA. The four main parts in the Flex framework are represented in the following.

a) MXML. MXML is an XML-based markup language that primarily describes screen layout. Using MXML tags, you can add components such as form controls and media playback to layout containers such as panel. In addition to screen layout, MXML could be used to describe effects, transitions, data models, and data binding. Flash Builder, a Flex integrated developent tool, enables developers to construct MXML with a What-You-See-Is-What-You-Get approach—build basic Flex applications without writing any code.

b) ActionScript. ActionScript is the programming language understood by Flash Player and is the fundamental engine of all Flex applications. MXML is best suited for screen layout and basic data features, while ActionScript is best suited for user interaction, complex application and transaction logic.

c) Flex class library. Flex framework defines the Flex class library. It consists of predefined components such as controls, containers, data components, and Flex Data Services.

d) Flex data services. They provide the remoting and messaging foundation for connecting a Flex-based front-end to back-end services in SimpleDB Cloud, and transport data between the client and Cloud server. HTTPService and WebService components, a kind of Flex Data Services, will be represented in the following section.

Figure 3. Flex framework.

The development process [14] of Flex applications in Cloud is shown in Figure 3.

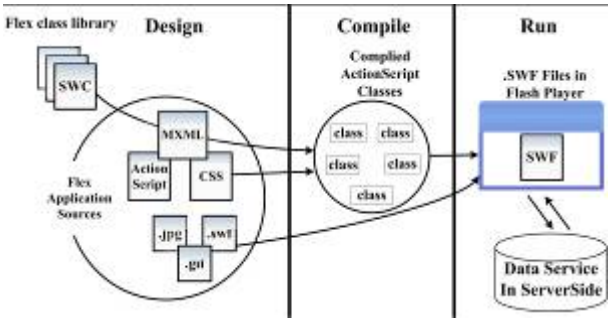


Figure 2. Development process of Flex applications in Cloud.

2) Strategies of Flex access to Cloud server-side: In Cloud application main Flex component [14] to access SimpleDB Cloud via REST is HTTPService. The HTTPService component sends HTTP REST requests to SimpleDB Cloud on the server side, and consumes the response from SimpleDB. Although HTTPService is typically used to consume XML, it can also be used to consume other types of responses such as JSON.

E. SimpleDB Cloud

Amazon SimpleDB [15] is a Web service for running queries on structured data. This service works in close conjunction with RIA, collectively providing the ability to store, process and query data sets in the cloud. These services are designed to make Web-scale computing

easier and more cost-effective for developers. SimpleDB is a database that is accessed via REST.

Amazon SimpleDB is easy to use and provides the core functionality of a database – real-time lookup and simple querying of structured data – without the operational complexity. SimpleDB requires no schema, automatically indexes our data and provides a simple API for storage and access. This eliminates the administrative burden of data modeling, index maintenance, and performance tuning. Developers gain access to this functionality, are able to scale instantly, and pay only for what they use.

1) The concepts: SimpleDB removes some of the constraints associated with relational database systems. For instance, SimpleDB-based data is organized into domains. All data contained in the domain may be queried. Domains are comprised of items, and items are described by attribute-value pairs.

Amazon uses the analogy of comparing SimpleDB-based data to a spreadsheet. The following list defines the various concepts associated with SimpleDB.

a) Customer Account: A user's SimpleDB account may be thought of as a spreadsheet; it may contain multiple individual sheets.

b) Domains: The individual sheets within the spreadsheet container are called domains. Domains are analogous to tables in the relational database world. Queries and all data operations use domains as their data source.

c) Items: The individual rows within domains (individual sheets) are called items. These items may contain one or more attribute name-value pairs.

d) Attributes: The individual columns within a domain or sheet are called attributes; they represent categories of data that can be assigned to items.

e) Values: The individual cells within a domain or sheet are called values. Values are instances of attributes or the actual values.

These concepts offer a simple approach to working with structured data.

2) SimpleDB APIs: SimpleDB has a small set of APIs that provide everything you need to work with SimpleDB-hosted data. These APIs allow you to create and work with domains, as well as the individual items, attributes, and values contained in individual domains. The following components are involved in SimpleDB transactions: 1) Subscriber: Any calls to the SimpleDB Web service use a unique access key for identification and billing. 2) Request: A call and its data to the SimpleDB Web service. 3) Response: The response and its data from the SimpleDB Web service. SimpleDB API contains:

a) CreateDomain creates a new named domain within the scope of your AWS account.

b) DeleteDomain deletes a domain and all of the items within it.

c) ListDomains returns a list of all of our domains.

d) PutAttributes creates a new item (if necessary) and adds or replaces attributes.

e) DeleteAttributes deletes one or more attributes from an item.

f) GetAttributes returns all or specified attributes of an item.

g) Query retrieves a set of items which match a query expression. Large result sets can be retrieved in chunks of up to 250 items.

SimpleDB makes it easy and straightforward to store and retrieve structured data in Cloud.

III. 2-TIER CLOUD ARCHITECTURE IMPLEMENTATION

Traditional development methods of Cloud application system are very complex. The main reasons are: 1) the architecture of Cloud implement is inappropriate; 2) the RIA's advantages can not be utilized fully; 3) almost all functionalities of application logic and transaction logic are overly concentrated on the server side. That is, inappropriate functionality segmentation between client-side and server-side.

To address the above problems, we proposed a novel Cloud architecture: 2-Tier Cloud Architecture, shown in Figure 4.

The main idea of 2-Tier Cloud Architecture is that by using RIA in the rich client-side tier is maximized, implements almost all functionalities such as UI presentation and human-machine interaction, application logic and transaction logic except of data storage. Rich client tier implements a complete application; by using SimpleDB Cloud in the server-side Cloud tier is greatly simplified, and implements storage, update and query of data sets in the cloud; the API interface between the RIA client-side tier and server-side Cloud tier is also simplified via minimized REST.

A. Rich Client Tier Implementation Based on Flex

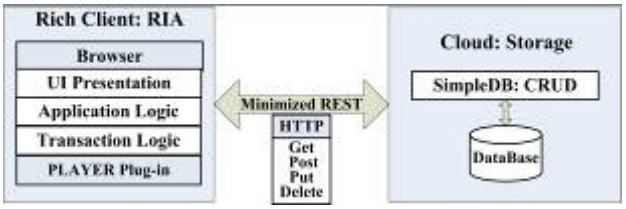


Figure 4. 2-Tier Cloud architecture.

Almost all 2TCAR's functionalities are implemented by MXML and ActionScript.

1) UI presentation and human-machine interaction: Using MXML, we can add components such as different controls and data charting into containers such as panel. Effects, transitions, data models, and data binding can be introduced into UI design. Human-machine interaction can be implemented by ActionScript and event-driven.

2) Application logic and transaction logic: All logic functionalities are programmed by ActionScript's Classes, Interfaces and predifined Class library.

3) Communication between rich client tier and server-tier in SimpleDB: SimpleDB REST calls are made

using HTTP GET requests. The Action query parameter provides the method called and the URI specifies the target of the call. Additional call parameters are specified as HTTP query parameters. The response is an XML document that conforms to a schema. By using Flex's HTTPService component, a request from rich client tier to server-tier in SimpleDB Cloud is constructed, but the format of the request has to comply with SimpleDB's REST API standard. And the receivement of the reponse from SimpleDB is implemented by HTTPService's callback function, which is parsed to acquire the data and refresh user interface.

The following shows a REST request that puts three attributes and values for an item named Item123 into the domain named MyDomain.

```
https://sdb.amazonaws.com/?Action=PutAttributes
  &DomainName=MyDomain
  &ItemName=Item123
  &Attribute.1.Name=Color&Attribute.1.Value=Blue
  &Attribute.2.Name=Size&Attribute.2.Value=Med
  &Attribute.3.Name=Price&Attribute.3.Value=14.99
  &AWSAccessKeyId=<valid_access_key>
  &Version=2010-01-08
  &Signature=Dqlp3Sd6ljTUA9Uf6SGtEExwUQE
  &SignatureVersion=2
  &SignatureMethod=HmacSHA256
  &Timestamp=2010-01-01T15%3A01%3A28-07%3A00
```

The following is the sample response from SimpleDB:

```
<PutAttributesResponse
    xmlns="http://sdb.amazonaws.com/doc/2010-01-
    08">
  <ResponseMetadata>
    <StatusCode>Success</StatusCode>
    <RequestId>
        f6820318-9658-4a9d-89f8-b067c90904fc
    </RequestId>
    <BoxUsage>0.0000219907</BoxUsage>
  </ResponseMetadata>
</PutAttributesResponse>
```

B. *Cloud Server-Side Tier Design Based on SimpleDB Cloud*

When accessing Amazon SimpleDB using REST, we must provide the following items so that the request can be authenticated:

1) Authentication:

a) AWSAccessKeyId—Our AWS account is identified by your Access Key ID, which AWS uses to look up our Secret Access Key.

b) Signature—Each request must contain a valid HMAC-SHA signature, or the request is rejected. A request signature is calculated using your Secret Access Key, which is a shared secret known only to you and AWS.

c) Date—Each request must contain the time stamp of the request.

Depending on the API we're using, we can provide an expiration date and time for the request instead of or in addition to the time stamp. For details of what is required and allowed for each API, see the authentication topic for the particular API.

2) Authentication process [16]: The following is the series of tasks required to authenticate requests to AWS using an HMAC-SHA request signature. It is assumed we have already created an AWS account and received an Access Key ID and Secret Access Key. We perform the first three tasks for creating authentication requests shown in Figure 5. AWS performs the next three tasks in Figure 6.

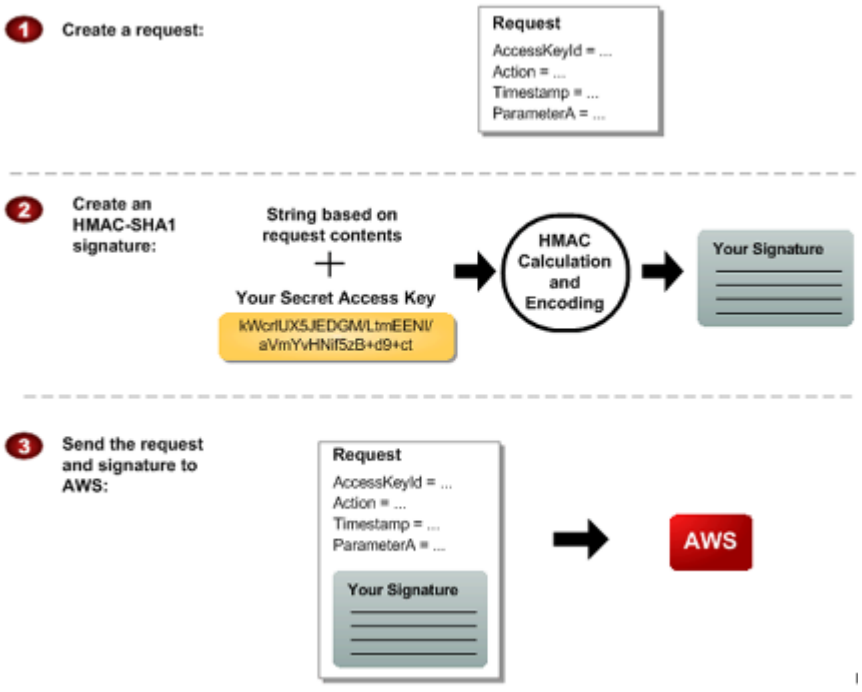


Figure 5. Process of creating authentication requests.

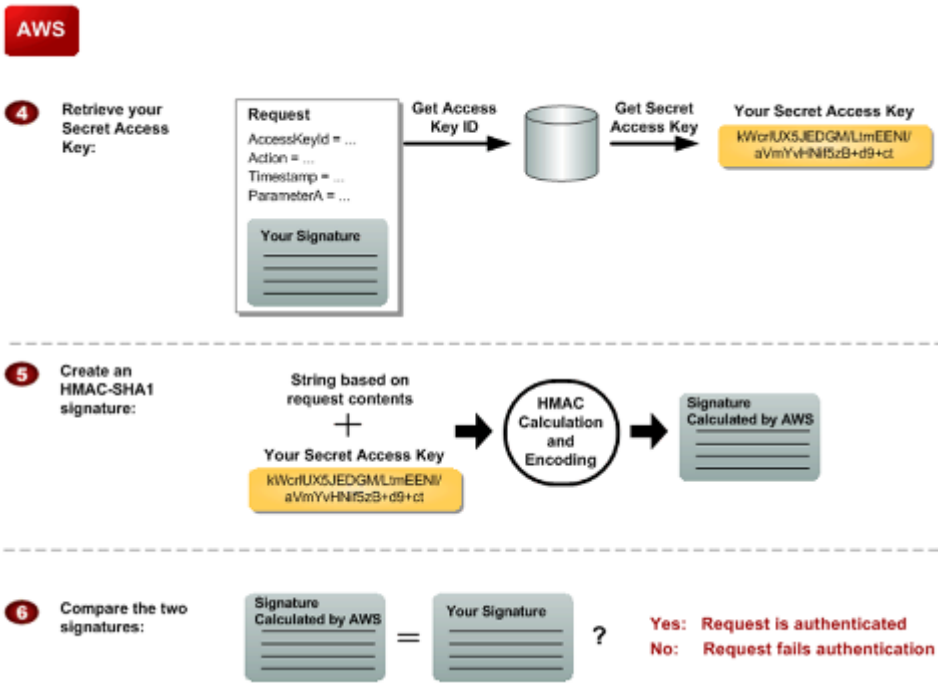


Figure 7. Process of confirming authentication by Amazon SimpleDB Cloud.

3) Authenticating REST requests and REST responses: We can send REST requests over either HTTP or HTTPS. Regardless of which protocol we use, we must include a signature in every REST request. If a REST request authenticated are confirmed SimpleDB will be responding a corresponding response. The samples of REST request and response are shown the above.

IV. Design of EHR Cloud System Based on 2TCAR

A. Design of the Orthodontic EHR Cloud Architecture

In the EHR Cloud for orthodontics we use the proposed 2TCAR Cloud architecture. That is that by using Flex on the client side, the rich client tier is maximized, implements almost all orthodontic workflow as shown in Figure 7 and functionalities such as UI presentation, human-machine interaction, and application logic and transaction logic except of data storage. Rich client tier developed by Flex implements a complete application; by using SimpleDB Cloud, the Cloud tier is greatly simplified, and implements only storage, update, query and deletion of data sets in the cloud; the API interfaces between the RIA client tier and Cloud tier on the server side are also simplified via minimized REST.

B. Workflow of the Orthodontic EHR Cloud

Firstly focusing on the contents of orthodontics and orthodontist' clinical practices, we analyze requirements and features of the EHR Cloud for orthodontics.

The health records for orthodontics are cumbersome, including three main examinations: general, special orthodontics, and temporomandibular joint, each of which contains lots of items. But according to the requirement analysis and orthodontists' daily workflow, it can be mainly divided into: patient's complaint, the relationship between front and back molars, degree of tooth-mouth-opening, and x-ray film of tooth positions, and so on.

Therefore, the operation of running the EHR Cloud must be ensured to completely match the above orthodontist's daily workflow. And after finishing the three examinations, all related data and images such as x-ray films and 3D models provided by the decision-support system, will be integrated and embedded into the EHR Cloud UI. At the same time the patient's problem list will be generated automatically and intelligently so as to facilitate the orthodontist to understand the patient's problems and determine the final treatment planning. Additionally, the EHR Cloud can also use a large number of patients' data for clinical research, analysis and statistics.

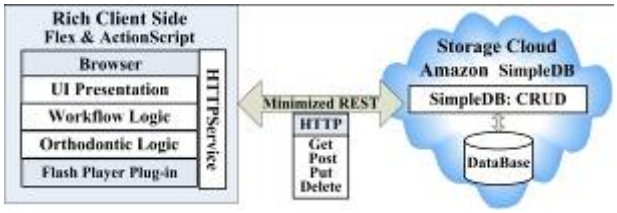


Figure 6. Orthodontic EHR Cloud architecture based on 2TCAR.

The workflow and software modules of the EHR Cloud mainly includes: Patient's Basic Information, Teeth States, Initial Examination, Orthodontic Examination, Temporomandibular Joint, Problem List, Initial Diagnosis, Initial Disposition, Treatment Planning, and Treatment Results as shown in Figure 8. To store data from different sub-workflows, we create the corresponding tables in the SimpleDB Cloud and the associations between the tables via the primary key case-id as shown in Figure 9.

To store data from different sub-workflows, we create the corresponding tables in the database and associations between the tables via the primary key case-id as shown in Figure 9.

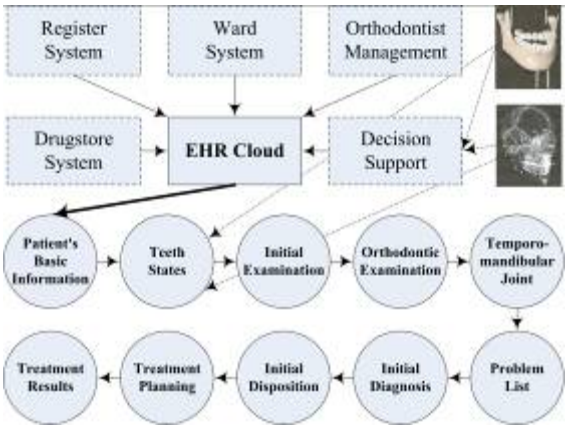


Figure 8. Workflow of the EHR Cloud.

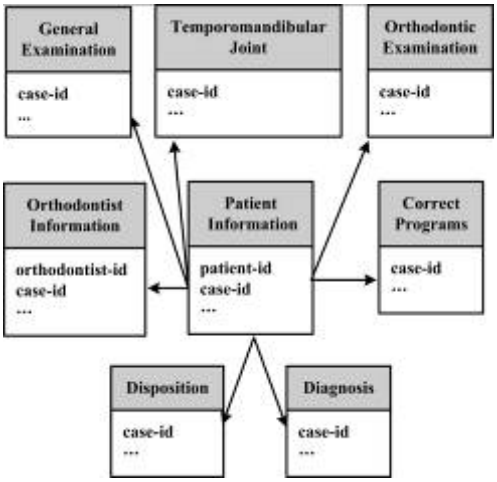


Figure 9. Workflow of the EHR Cloud.

C. *Design and Implement of the Orthodontic EHR Cloud Based on 2TCAR*

According to the above workflow and 2TCAR architecture of the EHR Cloud, we use Flex to implement almost all functionalities in the EHR Cloud exception of storing and querying data in the SimpleDB Cloud: 1) a special user interface (UI) for each sub-workflow detailed in Figure 10 and 11; 2) Application logic and transaction logic; 3) Communication between rich client tier and server-tier in SimpleDB. And different sub-workflow's UI can be specified and switched through clicking the corresponding button in the left-side main menu.

The data in different sub-workflow's UI is stored into and retrieved from the corresponding table in SimpleDB Cloud using Flex communication components such as HTTPService via REST. The resulting data and images from the Decision-Support systems can be also embedded in the EHR as shown in Figure 10.

The intelligent templates for orthodontists detailed in Figure 8 are also developed and embedded in different sub-workflows in the EHR Cloud. At the same time the patient's problem list can be generated automatically and intelligently from the data in the sub-workflows.

Additionally we also developed the system of query, analysis & statistics for research and education. The EHR Cloud can also connect with existing Resister, Ward and Drugstore information system so as to share all data.

In short working processes of the EHR Cloud completely follows orthodontist's daily workflow to become orthodontist's intelligent assistant. In the EHR Cloud Flex greatly enhances the interactive user experience and implements the workflow on the rich client-tier in the Cloud; SimpleDB Cloud also simply implements data storage and query on the server-tier in the Cloud; the API interface between the rich client-tier

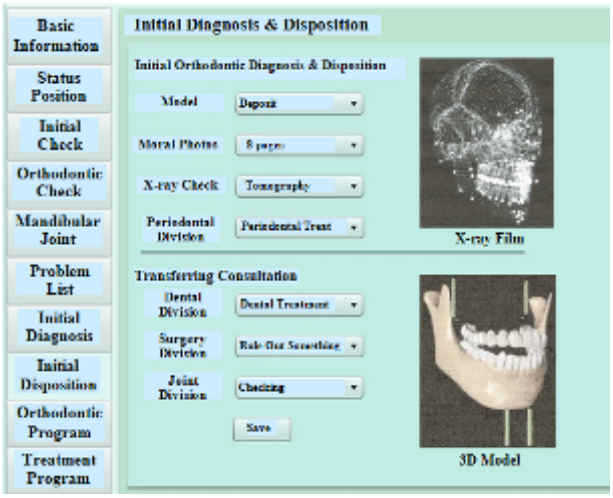


Figure 10. User interface of initial disposition.

and server-tier in the Cloud is also simplified via minimized REST.

V. CONCLUSIONS

By analyzing the serious problems and drawbacks in Cloud application development, we find that the problems stem from not fully utilized RIA's advantages and inappropriate functionality segmentation between client-side and server-side. That is, most of functionalities are overly concentrated in the back-end Cloud servers. We propose 2TCAR Cloud architecture containing RIA-based rich client tier and SimpleDB-based server-side Cloud tier. The Rich client tier is maximized to implement most of functionalities of Cloud application and transaction logic; in contrast functionality in server-side Cloud tier is minimized to only implement data storage and query. The communication between these two tiers is also simplified via REST.

The research results are applied to develop a Cloud application called EHR Cloud Based on 2TCAR for Orthodontics. According to orthodontist's daily workflow and 2TCAR, the EHR Cloud is successfully implemented, and is put into operation in Peking University School of Stomatology. It can fit seamlessly into orthodontist's daily workflow and effectively replace current paper medical records used in orthodontics.

It is shown that the 2TCAR is simple, effective and valuable for agile Cloud design. The 2TCAR Cloud

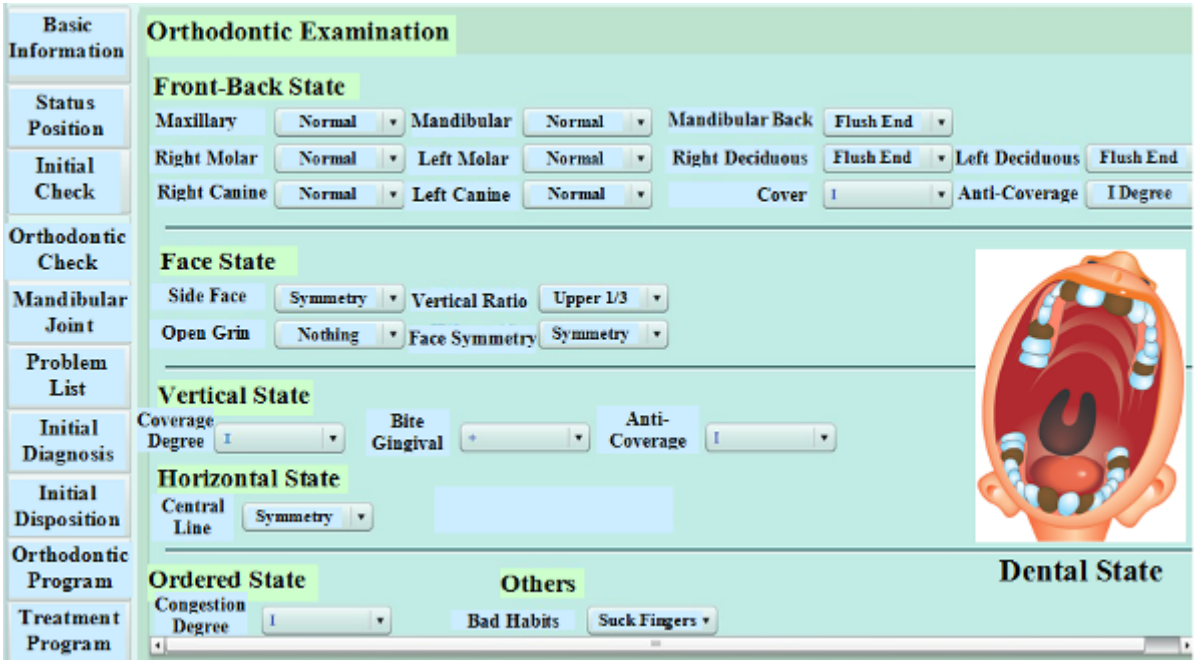


Figure 11. User interface of orthodontic examination.

architecture can be widely applied into Cloud application development.

ACKNOWLEDGMENT

This research project has been sponsored by Department of Orthodontics under Peking University School of Stomatology and China Women's University. We would like to express our heartfelt thanks to the sponsors.

REFERENCES

[1] W. J. Zhang, "From E-government to C-government via cloud computing," *in Proceedings of IEEE International Conference on E-Business and E-Government*, Guangzhou, China, pp. 679-682, May 2010.

[2] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," *In Grid Computing Environments Workshop*, pp. 1-10, April 2008.

[3] P. Gaw, "What's the difference between cloud computing and SaaS?," *Tech. Rep.*, 2008, http://blog.fortiva.com/fortivablog/2008/05/what-is-the-dif.html.

[4] M. Armbrust, A. Fox, and R. Griffith, "Above the clouds: a Berkeley view of cloud computing," *Technical Report*, UC Berkeley, 2009.

[5] J. X. An, "The Demonstration of Cloud Retrieval System Model," *Joural of Software*, Vol. 6, pp. 249-256, February 2011.

[6] C. F. Buck, "Designing a consumer-centered personal health record," *Tech. Rep.*, *California Health Foundation*, March 2007.

[7] G. R. Kim and C. U. Lehmann, "Pediatric aspects of inpatient health information technology systems," *In Pediatrics*, Vol. 122, pp. 1287-1296, December 2008.

[8] U.S. Department of Health and Human Services, "The nationwide privacy and security framework for electronic exchange of individually identifiable health information," *Office of the National Coordinator for Health Information Technology*, December 2008.

[9] R. Zhang and L. Liu, "Security models and requirements for healthcare application clouds," *in Proceedings of IEEE 3rd International Conference on Cloud Computing*, Miami, Florida, USA, pp. 268-275, July, 2010.

[10] E. Jamil, "What really is SOA. A comparison with Cloud Computing, Web 2.0, SaaS, WOA, Web Services, PaaS and others," *White Paper, Soalib Incorporated*, 2009.

[11] R. Lucchi and M. Millot, "Resource oriented architecture and REST," *Office for Official Publications of the European Communities*, 2008.

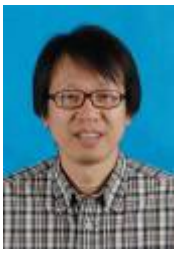
[12] Adobe Systems Incorporated, "Flex 3 help," 2009. http://livedocs.adobe.com/flex/3/html/help.html?Content=profiler3.html.

[13] Adobe Systems Incorporated, "Flex 3 language reference," 2009. http://livedocs.adobe.com/flex/3/langref/.

[14] Adobe Systems Incorporated, "Flex 3 developer's guide," 2009. http://livedocs.adobe.com/flex/3/html/help.html?Content=Part2_DevApps_1.html.

[15] W. J. Zhang, "Research of RIA design pattern based on Flex, Spring and Hibernate," *in Proceedings of the 5th China Conference on Software Engineering*, Beijing, China, vol. 1, pp. 126-128, November 2008.

[16] Amazon Web Services, "Amazon SimpleDB developer guide," 2009. http://simplerdb.rubyforge.org/.



Wenjun Zhang received the PhD from the China University of Mining and Technology in 1994 on Engineering and Computer Science.

He is an associate professor at the China Women's University currently. His research interests lie in the fields of Cloud Computing, 3D Visualization in Medical Images and Web Architecture.

Dr. Zhang is a member of China Computer Association and IEEE.