Analysis and Design of Cloud Publishing Platform and Its Social Network

Xiuhong Chen, Rong Peng, Han Lai, Keqing He
State Key Lab. of Software Engineering, School of Computer, Wuhan University, Wuhan, China
E-mail: {chenxiuhong, rongpeng, laihan, hekeqing}@whu.edu.cn

***Abstract*—SaaS (Software as a Service) has been attracted significant attentions from both industry and academia. Owing to serving multiple clients in the Long Tail, many notable SaaS applications have accomplished big successes in many traditional domains, such as CRM (Customer Relationship Management) and HRM (Human Resource Management).**

To promote the effect and efficiency of the traditional publishing industry, a solution of CPP (Cloud Publishing Platform) based on SaaS is proposed. Compared to the traditional e-publishing systems used by the publishers, it has the following distinguished features: 1) it is developed and operated by independent SaaS provider who not only provides valuable publishing services to enterprise tenants, such as periodical presses, but also provides significant services to individual tenants, such as scholars and researchers; 2) any individual tenant is not just a dedicator to a specific publisher, but a dedicator to all enterprise tenants, and his academic reputation can be accumulated by all the activities carried out in the platform; 3) a publishing cycle will be established based on CPP and benefit all the stakeholders.

At the end, five key points of developing and operating a successful SaaS application are concluded, which can be used as a guideline to general SaaS application development.

***Index Terms*—SaaS; multi-tenancy; manuscript submission and review; sustainable ecological environment**

I. INTRODUCTION

Recently, SaaS (Software as a Service) has been attracted significant attentions from both industry and academia [1]. With PaaS and IaaS, SaaS becomes one of the most distinguished representative of Cloud Computing Model which is designed to overcome the limitations and inefficiency of traditional software solutions.

From Wikipedia, SaaS is a computing paradigm which can provides software as a service continually through networks on an on-demand basis, by which the tenants are exempt from the exorbitant expense of software upgrade, functionality extensions, software and hardware management, and its operation and maintenance, et al [2]. Therefore, it has been successfully adopted in various fields, such as client relationship management, storage management and public administration [3-7]. Many newly emergent SaaS providers gained significant successes, such as SalesForce.com [8], Workday [9] and RightNow Technologies [10], and many traditional IT industry giants, such as IBM, Oracle and SAP, have purchased leading SaaS companies to gain market positions [11].

In traditional publishing industry, many puzzles hinder its development. For example, famous periodical presses are flooded with manuscripts and it's difficult for them to find enough qualified reviewers to response quickly; while the newly emerged or unknown periodical presses distress with lack of qualified manuscripts [12]. As to authors, they always contribute their manuscripts to seldom familiar journals and suffer from their late responses, and it is also difficult for them to identify the reputation of newly emergent journals. As to reviewers, the peer review process is energy consuming but anonymous, which means they must pay a lot but gain little, so the enthusiasm of reviews is relatively low. Obviously, the above problems are beyond the capability of any independent online manuscript submission and peer-review system [13-15]. Establishing a well-formed publishing ecological environment with affordable cost maybe a feasible way [16]. In this paper, how to establish a Cloud Publishing Platform (CPP) based on SaaS is elaborated. Based on the CPP, publishers registered as enterprise tenants can maximize the benefits by not only various publishing services and brand promotion services, but also the large amount of crowds including readers, authors and reviewers; individuals registered as individual tenants can gain benefits from not only the services rented by the registered publishers and the services provided by the platform, but also the social society established in CPP. Owing to SaaS rental mechanism, the operation cost is rather low. Therefore, within the CPP, all the stakeholders can achieve the win-win.

The rest of this paper is organized as follows: Section II introduces the related work. Section III analyzes the similarities and differences between traditional mode and SaaS mode, and elaborates the rationality of establishing CPP based on SaaS. Section IV describes the platform envisioning, and the focus lies on elaborating the information and events flows in the publishing cycle based on CPP. Section V analyzes the requirements of CPP which put emphases on how to embody SaaS features. Section VI elaborates the platform design from the perspectives of architecture design, database design and service design. Section VII concludes the analysis and design experiences which can be generalized to guide the development of SaaS applications.

II. RELATED WORK

There are many existing electronic publishing systems, such as Elsevier Editorial System, ACS Paragon Plus

doi:10.4304/jsw.9.9.2251-2259

Environment, and Open Journal Systems based on ASP (Application Service Provider) mode [17-19]. The ASP mode uses a single commitment model by allocating a specific service instance for each clients. Thus, if the clients increase, the corresponding service instances increase proportionally. Therefore, it is difficult for ASP providers to maintain and upgrade the lending system collectively, as every instance has its specific customization. At the same time, there is no aggregation effect regardless how many clients it has served.

To promote the effects and efficiencies of the periodical presses, Tananbaum and Holmes [13] focused on the evolution of Web-based peer-review systems; [20] designed an e-Journal Management System; Zhong [21] discussed the feasibility of making use of cloud computing in periodical presses, and presented an operational model.

In this paper, we focused on how to plan, analyze and design a SaaS based CPP to establish a virtuous publishing cycle for publishing industry.

III. SAAS MODEL

Owning and maintaining software remotely and providing customizable services to multiple customers are typical SaaS operating mode. The advantages include [22]:

- Software is provided on an on-demand basis and the one-to-many service model based on common codes and data definition reduces the cost of update and maintenance.
- Real-time access and update services are provided through the network.
- The billing system is based on usage or measures and clients do not need to pay for the software development, update and management.

Because of these advantages, SaaS has been adopted as successful solutions in many fields, such as CRM [8] and HRM [9].

TABLE I.
COMPARISON BETWEEN TRADITIONAL MODE AND SAAS MODE

Mode	Traditional mode	SaaS mode
Development cycle	• Long	• Short
Maintenance	• Difficult	• Convenient and flexible
Scalability	• Weak	• Strong
TCO	• High	• Low
Accumulation effect	• No	• Yes
Risks	• High	• Low

From the viewpoints of consumers, SaaS mode has many advantages compared with traditional mode (as shown in Table I):

1) The traditional software development cycle is longer than SaaS mode. In traditional mode, building a system on demand is time-consuming as the developing process always complies with one of the traditional models, such as waterfall model, adaptive software development and scrum. But in SaaS mode, SaaS application must provide customizable, composable and scalable components to tenants to construct their applications rapidly. Thus, from the viewpoints of clients, the time spent to built their on-demand application is rather short and it is just the time for customization.

2) To traditional software, the maintenance is always driven by customer. Thus, the tradeoff between the responsibilities and benefits always makes the maintenance difficult to fulfill. But the maintenance decisions of SaaS applications are made by their owners: SaaS providers. SaaS providers always utilize one instance to serve multiple tenants, which make the maintenance convenient and flexible.

3) The scalability of traditional software is weak, for it is designed according to the initial requirements specification. Once it have been delivered, any additional features will cost large amount of money. But in SaaS mode, scalability is the basic feature, as SaaS provider must support the fluctuation of the workload in the long term.

4) The Total Cost of Ownership (TCO) in traditional mode is higher than SaaS mode. In traditional mode, the client should pay for the development, infrastructure and maintenance; while in SaaS mode, the TCO of tenants are only rental fee measured by usage.

5) The application developed in traditional mode is designed for a single client. Its development, operation and maintenance are all irrelevant with other similar clients. In SaaS mode, the target clients of SaaS application are clients with similar requirements. This kind of similarity makes the accumulation effect possible by accumulating similar clients and similar end users.

6) The risks in traditional mode are rather higher than in SaaS mode. In traditional mode, the success of the project relies on not only the talent and ability of the development team, but also the maintenance team. Its expenditure is also high. In SaaS mode, most SaaS applications provide free trial before the tenant makes any decision. If you are not satisfied with the services, you can abandon it without a penny lost.

ASP is another typical development mode which has many similarities with SaaS, such as:

- Both of them provide software services through the Internet;
- Customers only need to pay for what they actually use; and
- Service providers are responsible for software management, upgrade and maintenance.

But there are still many differences between them, as Table II shows.

As to operation, compared with ASP who provides a unique instance to each client, SaaS only provides a single version to all tenants via Internet. Thus, it is easy for SaaS to maintain and upgrade in a stable way.

As to scalability, the most common solution of ASP is scale up or scale down, namely increase or decrease the resources to meet the demands of clients. But the scalability solution of SaaS can be either scale up/down or scale in/out which use the specific design of database to satisfy the demands on scalability.

As to cost, the tenants of SaaS only need to pay for the usage of selected services [23, 24]. But in ASP mode, the ustomer needs to pay for the levy license fee, customization fee and usage fee.

Last but not least, the ASP platform can't provide any aggregation effects regardless how many clients the ASP provider served, for it deploys a unique instance for each customer. But in SaaS mode, all tenants share the same platform, which is convenient for SaaS provider to develop value-added service to promote the aggregation effects.

TABLE II.
COMPARISON BETWEEN ASP MODE AND SAAS MODE

Mode	ASP mode	SaaS mode
Operation	Allocate a dedicated instance for each user, and each user will have its own application	All tenants share a unique instance in most cases, and each tenant will have its own metadata
Scalability	Scale up/down	Scale up/down and Scale in/out
Cost	Levy license fee and customization fee as the initial expense; Pay by use	No license fee and customization fee needed; Pay by use
Aggregation Effect	No	Yes

Although there are several issues needed to be considered in SaaS mode, such as security and data integrity, SaaS has still shown its great potential as a best solution in terms of operational effectiveness, economy efficiency and customer satisfaction. Therefore, we are concentrating on design a CPP based on SaaS.

IV. PLATFORM ENVISIONING

The enthusiasm of CPP provider is not enough to well serve the overall publishing industry and their customers due to the lack of enough energy and resources. Establishing a well-organized publishing economical environment based on CPP is the most important. Therefore, the platform envisioning should be the first step of the whole development process.

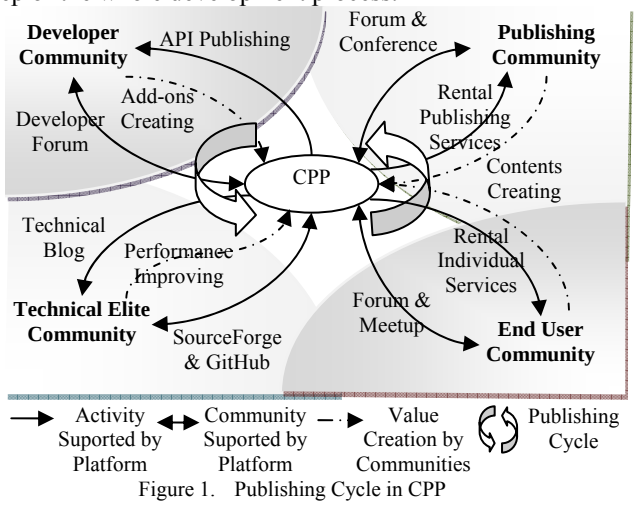


Figure 1. Publishing Cycle in CPP

The envisioned Publishing Cycle in CPP, as shown in Figure 1, is the first important guarantee for the success of CPP. According to the publishing cycle, the CPP provider must establish and make full use of the following four social networks for four kinds of users:

For **publishing enterprises**, except providing customizable rental publishing services to publishing industry, especially those publishing organization in the Long Tail, CPP provider will advocate the establishment of a healthy publishing community. It can establish an online enterprise forum which supports the publishers to communicate the insights of publishing industry and present their feedbacks to the CPP provider. It can also use its portal to call for the annual on-site conference to enhance the communication among publishers and encourage the publishers to organize seminars, symposiums and conferences by themselves. By these online and offline activities, the publishers can share their experiences and lessons on using the platform and developing their business, which will help the platform to evolve and improve the quality of the publications.

For **end users**, except providing valuable individual services to authors, reviewers and readers, CPP provider will establish an online discussion forum to support the individuals to communicate any specific topics related with the published contents and present their feedbacks to the CPP provider. CPP will use its portal and other forms, such as "Meetup" [25], to organize offline communication among the individuals. The face-to-face communication will help the individuals share their viewpoints and make friends, which will enhance the attractiveness of the platform.

To create a publishing environment with diversity, the participation of large third party developers is vital too.

For **developers**, CPP will open its API and establish a Developer Forum. With the open API, the developers can develop various add-ons to enhance the system and associate it with true life. The developers can use various developer forums to discuss and communicate their experiences and puzzles. And it can also be used as a communication channel between CPP engineers and developers. By monitoring and analyzing the forums, CPP engineers will learn the developers' demands and concerns, and sometimes they may even discover the vulnerabilities of the platform.

Encountering difficulties is inevitably as the knowledge of any team is limit. Finding the right person and asking for help may be the most economical and convenient method to overcome difficulties. Thus, establishing and maintaining a technical elite community is vital.

For **technical elites**, CPP can use the existing forums, such as GitHub [26] and Hacker News [27], to establish a collaboration network with the experts from Open Source Community to share and discuss various issues encountered together. At the same time, CPP can release Technical Blogs to share the solution they found and ask for improvement.

In summary, using CPP, on-line forums, on-site conferences and offsite activities, it can encourage publishers and end users to create diverse contents, share their experiences and accumulate their reputations. The huge user group will ignite the passion of developers. Developers then create various valuable Add-ons to enrich the CPP and support the users to create better contents. The challenges aroused by the platform, such as huge increase of access and storage, can be addressed by a technical elite community. Therefore, by establishing the

cycle, the publishing environment established on CPP will be well organized.

V. PLATFORM REQUIREMENTS

According to the platform envisioning, the strategic goal of CPP is to build a sustainable ecological environment for stakeholders. The detailed requirements analysis process of Online Submission and Peer Review Service Package (OSPRSP), consisted of online submission and peer review service and its auxiliary services, are selected to illustrate the requirements analysis process of CPP.

The objective of OSPRSP is to achieve a win-win among the main four roles in publishing process: periodical presses, readers, authors and reviewers. Since the OSPRSP is a SaaS based service package, its requirements should be elicited not only from system level and tenant level, but also from SaaS level.

A. System Level Requirements

In CPP, there are two major tenants, enterprise tenants and individual tenants. Enterprise tenants, such as periodical presses, rent the OSPRSP to carry out their business and promote their reputations. Individual tenants, such as scholars and students, can use the OSPRSP to manage their academic activities, such as manuscript submission, review and periodical retrieval. Thus, the platform should provide valuable functions for them to fulfill their tasks. The platform level requirements are described hierarchically according to the demands of the roles.

The roles include **System Administrator** and **Tenant**. The role **Tenant** is refined as **Enterprise Tenant** and **Individual Tenant**.

System Administrator is responsible to satisfy the demands from tenants, manage and maintain the platform. His main requirements include:

1) Support an on-line Service Level Agreement(SLA) negotiation mechanisam to negociate SLA with tenants' representatives;

2) Provide a friendly dashboard to monitor the status of the platform and tenant applications;

3) Support system management and maintenance, such as data replication and recovery, and usage metering.

For **Enterprise Tenant**, taking periodical press as an example, the main requirements are as following:

4) Support multiple negotiation mechanisms to communicate with CPP provider, such as telephone, fax, email and on-line negotiation, etc.

5) Support its main business processes: online submission and peer review;

6) Support efficiency improvements, such as manuscript-oriented reviewer recommendation and journal recommendation;

7) Support its brand broadcasting, such as electornic journal subscription and preference-oriented (PO) journal recommendation;

8) Support its application management, such as service configuration, authority management and data management;

9) Support multiple billing modes.

For **Individual Tenant**, the main requirements are as following:

10) Support individual academic activity management, such as preference-oriented (PO) or manuscript-oriented (MO) journal recommendation, unified submitted manuscripts tracing and remainder, to-be-reviewed manuscripts tracing and reminder, et al;

11) Support the multi-role based personal information management for any individal can be both manuscript contributor (author) and reviewer at the same time. The multi-role based personal information management can provide services such as personal preference, reputation and academic social network management, etc;

12) Support the subscription of various electroic services provided by enterprise' tenants;

13) Support service configuration.

After synthesize the requirements from the three roles, the UseCase diagram can be drawn as Figure 2.

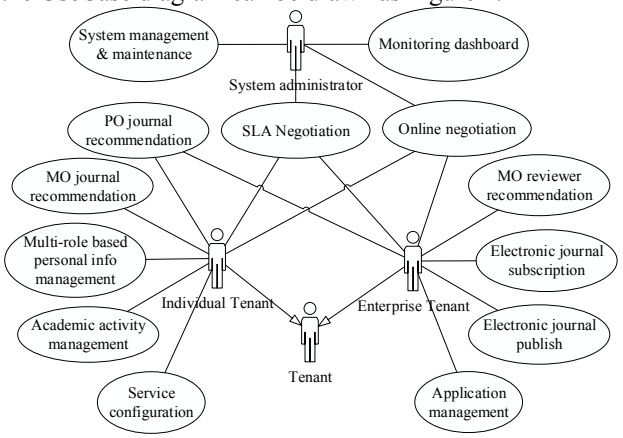


Figure 2. The UseCase Diagram of OSPRSP

B. Tenent Level Requirements

From tenant application level, the requirements of OSPRSP must be elicited and analyzed further. For example, the four roles of enterprise tenant, **Enterprise Administrator**, **Enterprise Tenant Editor**, **Author** and **Reviewer** must be identified to fulfill the task of online submission and peer review (OSPR).

Enterprise Tenant Administrator: The responsibility of the role is the same as the system administrator of traditional OSPR systems. The requirements of the role include:

1) Tenant application customization, such as OSPRSP subscription, and UI and workflow customization;

2) Tenant application management, such as role-based member management, authority assignment and management, and information publication.

Enterprise Tenant Editor: The main responsibilities of the role are to manage manuscripts submitted, assign the manuscripts to peer reviewers, and communicate with authors and reviewers until the acceptation or rejection of

the manuscript. Thus, the requirements of the role include:

3) Reviewer and author management, such as their contact information management;

4) Review process management, such as reviewer selection, manuscript assignment, and time-based review reminder, results synthesis,and results feedback, etc.;

5) Manuscript management, such as journal compilation.

Author: The author who has already registered as an individual tenant can use his/her profile to submit and manage manuscripts to any publication published by the enterprise tenants in CPP; and then, they become the authors of the press automatically. For unregistered authors, they must register as an individual tenant first, and then submit their manuscripts.

Reviewer: Similar as the role of author, there are two kinds of reviewers in the system, too. One is the individual tenant and the other is not. The tenant reviewers can use their unique profile to review papers for all the presses in the platform and accumulate their reputations. While non-tenant reviewers will be asked to register first.

The detailed use cases can then be identified according to the above elicited requirements. Based on the analysis, all of the use case diagrams can be got and they are omitted due to the page limitation.

After the requirements being elicited, the commonality and variability analysis should be performed accordingly [28].

C. SaaS Level Requirements

Since CPP is a SaaS application which is designed for serving multiple tenants, the distinguished features, such as multi-tenancy architecture (MTA), customization, scalability, replication and recovery, and security, must be taken into consideration.

1) MTA

CPP must provide qualified services to multiple enterprise tenants, namely publishing enterprises, to share the platform. Thus, its MTA must satisfy the following demands:

- Load balance: The load balancing and resource reallocation should be supported to balance the requests;
- Data Isolation: The tenant-specific data should be isolated from other tenants' and kept secure;
- Dynamic operation: A specific tenant application can be customized, added or removed from CPP dynamically without affecting the functionality and availability of other tenants applications;
- Independent deployment and maintenance: The code base of CPP can be patched and updated independently and unaware to tenants.

2) Customization

The 4-level customization should be supported: service customization, workflow customization, data customization and UI customization, which can support tenants to customize its applications according to their preferences.

- Service customization: CPP should allow each tenant to subscribe their preferred services. The tenant application should be able to be generated automatically based on the selection.
- Workflow customization: The workflow of a specific service should be customizable to comply with its business workflow.
- UI customization: The UIs, such as the logo, layout, style, and label description, should be customizable according to the tenant's preferences.
- Data customization: CPP should allow tenant to extend the existing data set and allow the optional data fields selection.

3) Scalability

The CPP should support hundreds enterprise tenants concurrently. Each enterprise tenants can publish weekly, biweekly, monthly and bimonthly journals according to its routine. The behaviors of authors, readers, reviews and editors are asynchronous. The obvious peeks are related to the working hours of stakeholders from an overall point of view. Thus, the scalability requirements can be categorized as following:

- User scalability: The concurrent number of users during peak hours that CPP can support should not degrade its performance.
- System scalability: When the number of access exceeds the threshold of its capability, CPP should support horizontal extension to meet the demands automatically.
- Database scalability: the database architecture needs to be scalable to serve the anticipated increase requests from online users, with no noticeable performance decrease and without any errors, holds and locks.

4) Replication & Recovery

The CPP should support triple-replication to provide high reliable services to tenants. And the replication should be dispersed physically. The failure of any server and database should be recovered without the notice of tenants.

At the same time, the CPP should support the tenant to customize its own replication & recovery policy.

5) Security

The 4-level security must be considered: physical security, network security, application security and data security.

- Physical security: the regulations of creating, access, and modification must be issued, monitored and recorded; the replication and recovery mechanism must be constructed.
- Network security: the proactive security protections such as perimeter defense and network intrusion prevention systems must be deployed; the security in transit must be ensured;
- Application security: the role-based access security and group policy-based security should be provided; and all tenants in CPP and all users in tenants' applications must have unique identifications;

- Data security: the data segregation must be guaranteed. When a user or a tenant requests data access, the system should validate his identification to ensure that it retrieves only the information corresponding to his authority; and the data in the system must be complete and credible.

D. Other Important Issues

To speed up its development and minimize the cost, locating and using appropriate existing software should be given high priority. Meanwhile, various monitors should be deployed in CPP to monitor the status of the whole system, give alerts whenever exception happens and provide the firsthand data to the decision of platform evolution.

VI. Platform Design

Based on the above requirements analysis, the design of CPP is illustrated from the following three perspectives: architecture design, database design and service design.

A. Architecture Design

To meet the requirements of dynamic customization, generation, deployment and operation, the architecture of CPP must be dynamic in nature to satisfy the distinguished expectations of tenants and their users. Thus, a metadata driven architecture is adopted by CPP.

The architecture of CPP is a 5-layer architecture, which includes Presentation Layer, Logic Layer, Application Layer, Web Service Layer and Data Layer, as shown in Figure 3. Functions of each layer are described as follows:

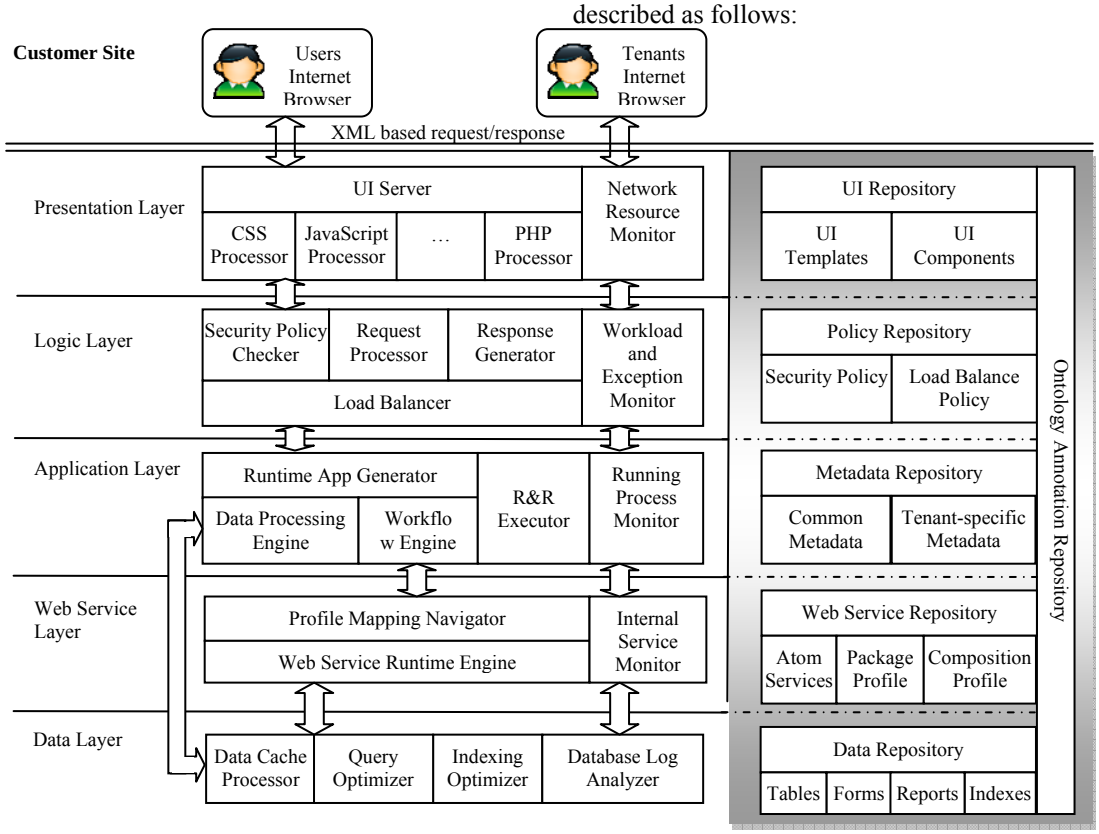


Figure 3. Architecture design of CPP

- **Presentation Layer**: This layer offers the interactive interfaces to users and tenants. It can support multiple interface definition languages, such as CSS, JavaScript and PHP, to describe UIs. For flexibility, the UI interface is designed completely separate from the business logic. Thus, updating or even completely re-architecture the interface layer will not touch the underlying data and business logic. The compositional UI Components are stored in UI Repository together with UI Templates. With the support of UI repository, a tenant can select preferred UI templates and UI components to configure his own interface.

- **Logic Layer**: This layer is responsible to analyze requests from the presentation layer, transfer it to executable tasks and assign them to different application server according to the load balance policy. The **Security Policy Executor** is responsible to execute necessary security checks according to the security policies and the parameters in the requests. The **Request Processor** is responsible to interpret the requests to executable tasks, such as application customization request, service access request and data retrieval request. The **Response Generator** is responsible to generate responses according to the feedbacks from the application layer and data layer. The **Load Balancer** is responsible to assign

tasks to appropriate servers according to load balance policies. All policies needed are stored in the **Policy Repository**.

- **Application Layer**: This layer is supported by a serious of Application Servers. Each **Application Server** adopts Multi-Tenancy Architecture. Its responsibility is to perform the tasks assigned by Load Balancer. The tasks include generating tenant's application, calling specific services to fulfill the task; metering the usage and executing R&R policies, etc. For example, when **Runtime App Generator** receives the tenant application generation request, it will search the **Metadata Repository** first to find out the corresponding common metadata and tenant-specific metadata; with these metadata, it will call the **Workflow Engine** to compose the tenant application and call the **Data Processing Engine** to access the **Data Repository** to get the necessary data, and then a tenant application instance will be returned. At the same time, the **R&R Executor** will execute the replication task according to the tenant-specific replication policy. And all the processes running in the Application Server will be monitored by Running Process Monitor.
- **Web Service Layer**: In this layer, **Profile Mapping Navigator** is responsible to interpret the relationship within the service package according to the package profile and composition profile in Service Repository; after interpretation, the Workflow Engine calls the Web Service Runtime Engine to execute the command. In CPP, atom web services are the minimum functional units. All functions are completed by web services.
- **Data layer**: This layer consists of many Persistent Data Servers physically. **Data Servers** store the **Metadata Repository, UI Repository**, **Policy Repository**, **Web Service Repository** and **Data Repository**. The data inside these repositories can be divided into three categories: Common System Data, Tenant Specific Data and User Specific Data. The Common System Data includes various templates, policies, components, services and data needed to construct CPP and the base functionality of tenant applications. The Tenant Specific Data includes various tenant-specific metadata, policies and data are used to construct customized tenant application. The User Specific Data includes the user profiles, privileges and user data.

Therefore, the Runtime App Generator is the kernel of the platform. It is responsible for composing customized tenant application according to tenant specific metadata and data. The load balancer, security policy checker and R&R executer are responsible for the scalability, security and R&R respectively. And the monitors in each layer are responsible for monitoring the status of the system and reporting exceptions and alerts.

B. Database Design

Database design is vital to the performance, scalability and availability of the CPP.

To support tenants to create, use and update their customized applications without affecting other tenants' usage, CPP adopts shared database policy.

The database is designed as Figure 4 shown.

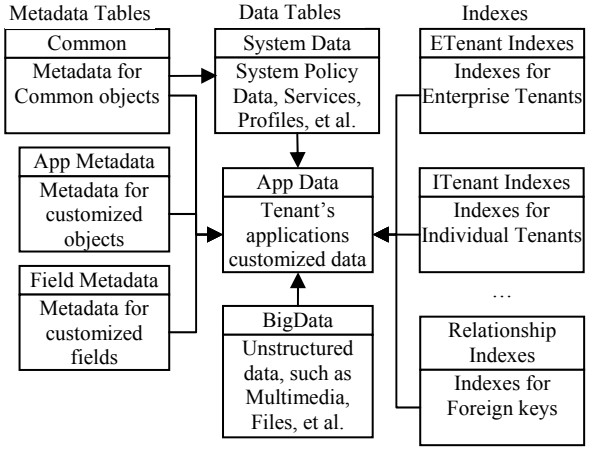


Figure 4. Database design of CPP

- **Metadata Tables**: The Common Metadata Table, App Metadata Table and Field Metadata Table are three major metadata tables. The Common Metadata Table stores the metadata of the common objects in CPP and its tenant applications. These objects cannot be customized by tenants. And some of them are necessary to construct tenant applications while others are optional. The App Metadata Table stores the metadata of the customized objects of tenant application. Using the TenantID and ObjID in each row can locate the unique object. The Field Metadata Table stores the metadata of customized fields that is defined for customized objects. Using TenantID, ObjID and FieldID in each row can locate the unique field. And the definition of each field includes FieldName (the name of the field), FieldType (the type of the field) and FieldOrder (the order of the field).
- **Data Tables**: The System Data Table, App Data Table and BigData Table are three major data tables. The System Data Table stores the information needed to construct CPP. The App Data Table stores the tenant-specific data. With the metadata defined in App Metadata Table and Field Metadata Table, the App Data Table can store different types of tenants' data in a same table by transform them into unique type "Variable-length String". And when the application want to read data from the table, it can use the internal functions provided by database to transform the string to its corresponding data type, such as Number, Date, and so on. The BigData Table stores the unstructured data of tenants, such as multimedia data and files. The information in it can be searched by professional search engine for better user experiences.

- **Indexes**: To speed up the high-frequency search functions, various indexes will be built in the database. The ETenant Indexes Table will construct indexes according to the usage frequency of the service by enterprise tenants; and the same will be done in ITenant Indexes Table. To speed up the Join functions among different tables, the Relationship Table will be established to record the combination relationships among tables.

By the above metadata driven multi-tenant architecture design, CPP can support tenants to customize their applications respectively while the others are using their applications concurrently. By the separation of UIs, workflows, services and data, CPP could support multi-level and multi-granularity customization; by the information got from various Monitors and Load Balancer in Logic Layer, CPP could scale accordingly; by runtime R&R executor and customized R&R policies, CPP can duplicate the data of tenant application in real time according to the customized R&R policies; by deploying peripheral firewall and the Security Policy Checker in Logic Layer, CPP can guarantee the authorized access; and by the TenantID based logic partition, the data of different tenant is isolated.

C. Service Design

To develop the CPP rapidly and economically, reusing existing software is a good choice. Based on the above design, the social community and OSS search engine, such as SourceForge, can be used to find reusable software. The PaaS selection should not only pay much attention to its functions and price, but also to its availability. And the existing software selection must concern on the functionality, maintenance status of the software and the constraints of its license.

By wrapping them as web services, they can be used as the basic functional units of CPP. According to the commonality and variability analysis performed in the phase of Requirements Analysis, the available services and service packages in Web Service Layer can be constructed.

After the requirements analysis and design of CPP have been done, the agile development process can be carried on.

VII. Conclusion

The emergence of SaaS computing paradigm brings new opportunities to many traditional industries, especially for those enterprises in the Long Tail. CPP based on SaaS can help academic society to establish a more energetic ecological environment. Publishers can customize their own applications with lower cost. By using the services provided, publishers can promote their reputations, attract origin manuscripts and expand their influences. Scholars can use the services provided to manage their academic activities and promote their academic experiences and prestige. And the most important value of this paper is that it elaborates the requirements analysis and design processes of CPP to show how to develop a SaaS application successfully. From the illustration, the following lessons can be drawn:

1) How to Establish a Well-organized SaaS Environment Should be Taken into Consideration from Beginning

SaaS application is designed to serve multiple tenants. Due to the diversity of tenants, it is not enough to rely solely on SaaS provider. SaaS provider must establish a good operation environment to arouse the participation enthusiasm of all stakeholders. And the plan must be economical and energy-saving to carry out. The publishing cycle proposed in Section IV is a good case to exemplify how to establish a well-organized SaaS environment.

2) The Requirements must be Elicited and Analyzed from both System Level and Tenant Level

The users of a SaaS application may come from multiple levels, such as tenants, subtenants, and the customers of tenants and subtenants, according to its business model. Thus, eliciting and analyzing its requirements should take all these factors into consideration. The requirements from system level emphasizes its overall functionality and quality features, while the requirements from tenant level concentrates on demands of its users and customers. The requirements elicitation and analysis in Section V is a good example.

3) The Requirements from Distinguished SaaS Features must be Considered Carefully

Customizability, MTA, scalability, replication and recovery, and security are the distinguished common features of SaaS applications, which are proven by both academia and industries [1, 8-10, 29]. Eliciting and analyzing the impacts from these features, the earlier the better, which have been shown in Section V.

4) The MTA and Database Design is Vital to the Success of a SaaS Application Development

SaaS applications must afford the capability of tenant application customization and operation concurrently, securely and rapidly. Thus, MTA design and database design are important. The platform design in Section VI gives a good illustration of how to build a scalable MTA and database. And the design can be extended to the design of other SaaS applications by topic replacement.

5) Wise External Resource Utilization can Significantly Reduce the Cost and the Time Needed by Developement

Whether developing the SaaS application from scratch or reusing some existing software, and whether tackling all the difficulties entirely by the development team or resorting to the technical elites outside are both important decisions that should be made. As shown in Section VI, selected existing software can be wrapped into services and integrated into a SaaS application.

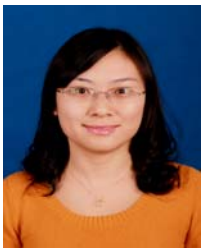
In future, CPP will be implemented accordingly and the design method of CPP will be extended to general SaaS application development.

ACKNOWLEDGMENTS

The research was supported by the National Natural Science Foundation of China under Grant No. 61170026, the National High Technology Research and Development Program of China (863 Program) under Grant No. 2013AA12A206, the Key Technologies R&D Program of Wuhan under Grant No. 201212521826.

REFERENCES

[1] W. Tsai, X. Bai and Y. Huang, "Software-as-a-service (SaaS): perspectives and challenges," Science China Information Sciences, pp. 1-15, 2014.
[2] Software as a Service, Wikipedia, http://en.wikipedia.org/wiki/SaaS.
[3] H. Wang, "Information Services Paradigm for Small and Medium Enterprises Based on Cloud Computing," *Journal of Computers,* vol. 8, pp. 1240-1246, 2013.
[4] J. Cho, "Study on a SaaS-based library management system for the Korean library network," *The Electronic Library,* vol. 29, pp. 379-393, 2011.
[5] X. Shi, Y. Sui and Y. Zhao, "Hybrid Cloud Computing Platform for Hazardous Chemicals Releases Monitoring and Forecasting.," *Journal of Computers,* vol. 7, pp. 2306-2311, 2012.
[6] C. D. Weissman and S. Bobrowski, "The design of the force. com multitenant internet application development platform.," in *2009 ACM SIGMOD International Conference on Management of data*, 2009, pp. 889-896.
[7] C. J. Guo, W. Sun, Z. B. Jiang, Y. Huang, B. Gao, and Z. H. Wang, "Study of Software as a Service Support Platform for Small and Medium Businesses," *New Frontiers in Information and Software as Services*, pp. 1-30, 2011.
[8] SalesForce.com, http://www.salesforce.com/company/.
[9] Workday, Workday's Co-CEO Looks Ahead, Entrepreneurs, http://www.forbes.com/sites/tomtaulli/2013/01/09/workdays-co-ceo-looks-ahead-to-2013/, 2013.1.
[10] Oracle, Oracle RightNow Cloud Service, http://www.oracle.com/us/products/applications/rightnow/overview/index.html.
[11] Newsfactor, Analysis: SaaS Companies Are Hot Acquisitions, http://www.newsfactor.com/story.xhtml?story_id=0010000MO8Y0, 2012.4
[12] S. K. S. Esseh, "*Strengthening Scholarly Publishing in Africa: Assessing the Potential of Online Systems*," Ph.D. dissertation, University of British Columbia, Vancouver, Canada, 2011.
[13] G. Tananbaum and L. Holmes, "The evolution of Web-based peer-review systems," *Learned Publishing,* vol. 21, pp. 300-306, 2008.
[14] R. Clarke and D. Kingsley, "e-Publishing's impacts on journals and journal articles," *Journal of Internet Commerce,* vol. 7, pp. 120-151, 2008.
[15] M. Ware, "Online submission and peer-review systems," *Learned publishing,* vol. 18, pp. 245-250, 2005.
[16] H. Lai, R. Peng, J. Cui, Y. Ni, and Y. Huang, "Design and implementation of journal manuscript submission and review system based on SaaS," in *2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems*, 2013, pp. 1-6.
[17] Elsevier Editorial System, http://www.elsevier.com/editors/elsevier-editorial-system-ees.
[18] ACS Paragon Plus Environment, https://acs.manuscriptcentral.com/acs.
[19] M. Cyzyk and S. Choudhury, "A survey and evaluation of open-source electronic publishing systems," *unpublished paper, Sheridan Libraries staff research,* 2008.
[20] S. Bhattacharyya, K. Mondal, S. Agarwal, and A. Nath, "Design and Analysis of e-Journal Management Systems: SXC International Journal of Advanced Computing Sciences (SXC-IJACS)," in *2012 International Conference on Communication Systems and Network Technologies*, 2012, pp. 913-918.
[21] X. J. Zhong and Y. Li, "Cloud computing and journals cloud - science and technology journal of a new platform for the future," *Publish Research,* pp. 77-81, 2011.
[22] N. I. S. Agency, "IT environmental change and tasks according to appearance of SaaS," Issue Analysis of Information Society 2007.
[23] M. Xin and N. Levina, "Software-as-a service model: Elaborating client-side adoption factors," in *2008 International Conference on Information Systems*, 2008.
[24] H. C. Liao and C. Q. Tao, "An anatomy to SaaS business mode based on Internet," in *2008 International Conference on Management of e-Commerce and e-Government*, 2008, pp. 215-220.
[25] L. F. Sessions, "How Offline Gatherings Affect Online Communities: When virtual community members 'meetup'." *Information, Communication & Society,* vol. 13, pp. 375--395, 2010.
[26] GitHub, https://github.com.
[27] Hacker News, https://news.ycombinator.com/news.
[28] B. Sengupta and A. Roychoudhury, "Engineering multi-tenant software-as-a-service systems," *3rd International Workshop on Principles of Engineering Service-Oriented Systems,* 2011, pp. 15-21.
[29] G. Lin, Y. Bie and M. Lei, "Trust Based Access Control Policy in Multi-domain of Cloud Computing," *Journal of Computers,* vol. 8, pp. 1357-1365, 2013.



Xiuhong Chen, Hubei, China, 10/13/1981. Master in Computer Software and Theory, Wuhan University, Wuhan, Hubei, China, 2006.

She is a PH.D. candidate of of State Key Lab of Software Engineering in School of Computer at Wuhan University. Her research interests include component repository modeling and management, software engineering and cloud computing.

Rong Peng is a Professor of State Key Lab of Software Engineering in School of Computer at Wuhan University. She has a Ph.D. in Computer Software and Theory from Wuhan University in China. Her research interests include requirements engineering, software engineering, and mobile computing.

Han Lai is a Ph.D. candidate of State Key Lab of Software Engineering in School of Computer at Wuhan University. His research interests include requirements engineering and service computing.

Keqing He is a professor of State Key Lab of Software Engineering in School of Computer at Wuhan University. His research interests include service computing, cloud computing, information system interoperability.