

A Multi-tenant Software as a Service Model for Large Organization

Wen Huang, Xiaoyi Wei, Yixia Zhao, Ziming Wang, Yun Xiao
Computer Network Information Center
Chinese Academy of Sciences
Beijing, China
{hw, wxy,zyx,wzm,xy}@cnic.cn

Abstract—software as a service (SaaS) has been adopted by more and more small and medium-sized businesses with the spread of relevant ideas in recent years. But its application is still limited in large organizations. Sophisticated organizational structure, various customized sub-organization services and complex cross-organizational interaction are un-ignored challenges for its success. Based on the analysis of organizational structure and business characters for large organization, the paper presents a novel multi-tenant SaaS model. The model fully absorbs strongpoints from traditional one-tenant software architecture, common SaaS data model and SOA based distributed architecture, and owns stronger strength in solving challenges occurred in SaaS project for large organization. A case is given to clearly show the model's advantages beyond the other three. A software implementation for the case and related discussion is presented later.

Keywords—software as a service; multi-tenant; model; large organization

I. INTRODUCTION

Software as a service (SaaS), is a software delivery model in which software and associated data are centrally hosted on the cloud [1] and users can enjoy software service with only a web browser. Multi-tenancy is a key technology in this field. With this architecture, a single service environment can be used for all customers ("tenants") [2] [3], which has a high cost performance, and gains much attention from both academia and industry. Various multi-tenant SaaS software has emerged, and a large number of small and medium-sized businesses (SMBs) have been the customers of SaaS [4] [5]. But because of a number of restrictions, compared with the success in SMB market, SaaS is still used very limited in large organizations, especially Group Corporations. And exploration and research in this aspect is also not much.

Based on years of research and practice experiences in SaaS field, we find that an obstacle not to be neglected for the application of SaaS in large organization is: commonly, there are many hierarchical sub organizations which not only have the requirement in customized management and data isolation but also need resource sharing and data interaction among them in a large organization. The majority of SaaS software in the market belongs to one-level architecture and is difficult to implement sophisticated interaction among tenants. Industry always seeks help from Service Oriented Architecture (SOA) [6] to solve this issue. One piece of software is deployed as a

service for each sub organization and Enterprise Service Bus (ESB) [7], web service are used to support data interaction among those services. This architecture always is too complex and needs to possess too many hardware and software resources, which causes the difficulty of maintenance and low utilization. The purpose of this paper is to explore a system level SaaS model that not only conforms to the requirement of tenant in customized configuration and data isolation but also effectively implements inter-tenant collaboration and resource sharing for large organization.

The contributions of this paper can be summarized as follows:

- 1) An identification of the characters and requirements of a large organization in organizational structure and business aspects.
- 2) A novel SaaS model that fulfills those characters and requirements is proposed.
- 3) A case and its implementation of the SaaS model are presented.

The rest of the paper is organized as follows: Section 2 introduces this novel model from both model factor and data architecture aspects with the analysis of common characters and requirements of a large organization. Section 3 displays a real case to further indicate the model's advantages beyond others; an implementation for the case is also presented. A discussion of related papers is made in Section 4. Section 5 summarizes the paper and points out our future work.

II. MODEL DESCRIPTION

A. Common organizational structure

Organizational structure is the formal arrangement of jobs within an organization. It is important to help accomplish organizational goals. Many types of organizational structure have been designed to satisfy the practical requirements of organizational management work. Following are the most popular ones [8].

Simple Structure: An organizational design with low departmentalization, wide spans of control, centralized authority, and little formalization. It is popular with small and start-up organizations.

Functional Structure: An organizational design that groups together similar or related occupational specialties. Cost-saving as it is specialization (economies of scale, minimal duplication of people and equipment).

Divisional structure: An organizational structure made up of separate, semiautonomous units or divisions. It focuses on results, managers are responsible for what happens to their products and services.

Matrix structure: An organizational structure that assigns specialists from different functional departments to work on one or more projects. The project manager has authority over team members in project, but all members in the team are still subject to their functional departments.

Contemporary large organization always uses functional-and-divisional-mixed structure, assisted with matrix structure to manage the whole organization. Its organizational structure often can be divided into several levels: department of management in headquarters, region, sub organization, department, and team, which is like Figure 1.

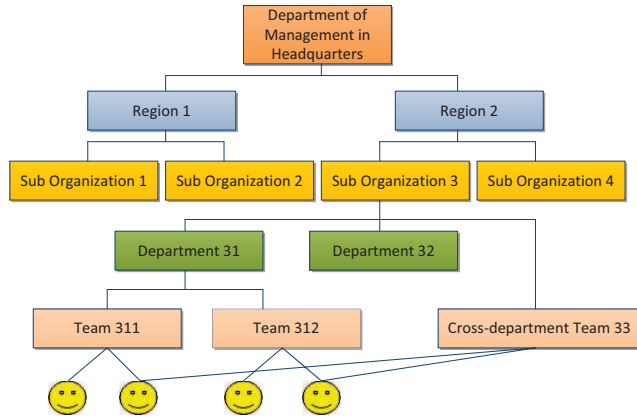


Fig. 1. Sample of common structure of large organization

In this figure, *Cross-department Team 33* represents a business team whose members come from different teams under *Department 31*.

Different level subordinate organization has different business characters and management requirements. It is necessary to provide customized service to different subordinate organizations. But on the contrary, they are all under the same large organization, intercross of employees and data sharing are also very universal. The emergence of a suitable SaaS model for large organization is of great value.



B. Organization and Employee

The basic framework of our model is based on the mixed type of organizational structure introduced. Organization and employee are two important factors in the model:

Organization: the basic management unit of data and employee in an organizational structure, no matter it represents the department of management in headquarters, a region, a sub organization, a department or a team. According

to the existence type, organizations can be divided into *real organization* (RO) and *virtual organization* (VO). RO exists in organizational design of functional and divisional structure. It can have employees and subordinate organizations. VO services for matrix structure, and is mainly used to represent an organization that crosses several ROs. It can have sub VOs, but not sub ROs. It does not have employees, but is connected with employees under ROs.

One employee is only affiliated with one RO, but can be connected with several VOs at the same time.

Figure 2 is a transformed model of Figure 1 based on above definitions. In this figure, “” and “” are used to represent RO and VO separately. ‘—’ is used to represent both the superior-subordinate relationship between organizations and the owner-member relationship between an organization and an employee. “-----” is used to represent the incidence relationship between a VO and a RO or between a VO and an employee.

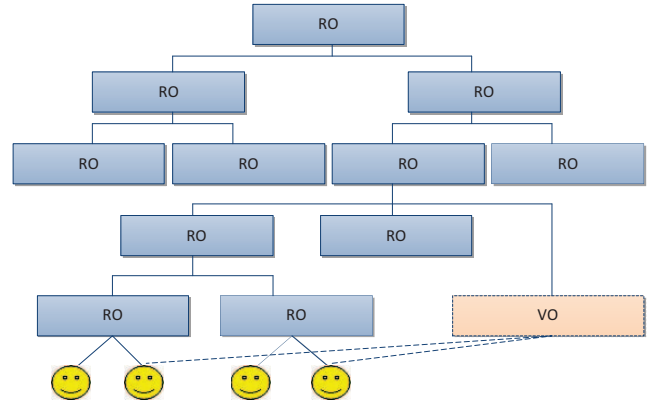


Fig. 2. Transformed organizational structure of Figure 1

Compared with Figure 1, *Cross-department team 33* is set as a VO, and linked with related employees under ROs. In actual business, those links represent the target employees are joining in projects under *Cross-department team 33*.

C. Tenant

Tenant represents the independent consumer in SaaS [2]. It can be a person, an organization or an application [9] [10] [11]. In this SaaS model for large organization, tenant is defined as a specific organization or organization group which shares one separated service environment in SaaS. The scope of a tenant is determined according to its business requirement. It is the basic unit for obtaining customized configuration and services in a large organization. A tenant includes one organization at least with full management authority.

Relationship between tenants can be categorized into separate, inclusion, incidence and lean-on.

Separate Relationship: there is no direct relationship between two tenants. Only the public resource can be shared and business data are communicated through workflow

between each other. Since any inter-tenant business management can be implemented by workflow, two tenants which have two organizations with superior-subordinate relationship in an organization tree are also categorized into separate relationship,. Figure 3 is a sample to show the relationship. As a superior organization, *RO 1* gathers the business data of *RO 2* through the workflow between *RO 1* and *RO 2*, any other *RO 2*'s private data can not be accessed by *RO 1*.

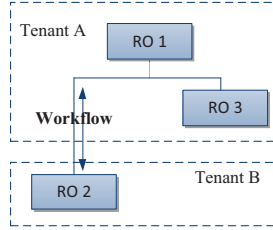


Fig. 3. superior-subordinate relationship in an organization tree

Inclusion Relationship: one tenant includes another, like the relationship between *Tenant A* and *Tenant B* in Figure 4. According to the agreement between the two tenants, all or part of *Tenant B*'s data can be seen by *Tenant A* and gathered as part of *Tenant A*'s statistics. Employees in *Tenant B*, like *Employee 1* and *Employee 2*, are affiliated to both *Tenant A* and *Tenant B*. This relationship often exists in the scenario that an organization has a strong requirement in business customization, but the business is more or less monitored by its parent organization.

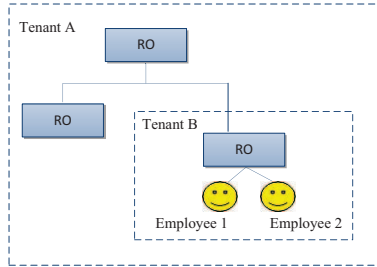


Fig. 4. Tenants with inclusion relationship

Incidence Relationship: If one or more VOs in a tenant are connected with employees under the RO in another tenant, incidence relationship is between the two tenants. Like *Tenant A* and *Tenant B* in Figure 5. *Employee 2* and *Employee 3* under *RO* in *Tenant A* have also joined *VO* in *Tenant B*. This relationship often exists in organizational architecture of matrix structure. *VO* is a temporary organization or project, and its members are from *ROs*.

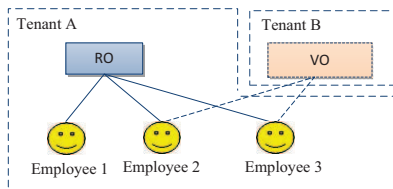


Fig. 5. Tenants with incidence relationship

Lean-on Relationship: If one or more VOs in a tenant lean on the ROs in another tenant, lean-on relationship is between the two tenants. Like *Tenant A* and *Tenant B* in Figure 6. *VO 2* in *Tenant B* leans on *RO 2* in *Tenant A*. This relationship often exists in the business scenario that a department in a tenant is one organization with two bands, and needs to deal with business from both of the two tenants.

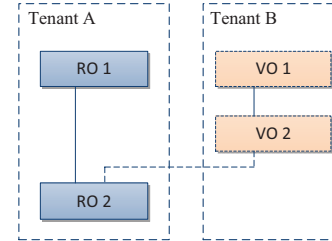


Fig. 6. Tenants with lean-on relationship

D. Data

Data is the most valuable asset for an organization. Its management work is indispensable in organizational daily activities. There are two types of data designed in our model: *business data* and *resource data*.

Business data is the data related to daily business management work. In major situation, its access authority is restricted to related employees. **Organization** is the basic unit in data management. **Business data** can be sent among organizations through workflow.

Resource data refers to various resources uploaded or online produced by tenants. It always includes a big amount of non-structural ones, such as videos, documents and pictures. Employee can administer his own resources, and set their open scope to private, self-organization open, self-tenant open or public level.

Private: resource can be online used only by its owner, and accessed by users which owner specifies.

Self-organization Open: resource can be accessed and online used in activities by users in the organization it belongs to.

Self-tenant Open: resource can be accessed and online used by employees in the tenant it belongs to.

Public: the resource can be accessed and online used by all employees in the platform.

E. Architecture of the model

Common data architecture design for SaaS [3] uses separate-schema or shared schema approach to do tenant data isolation. It can have a nice behavior in multi-tenancy and tenant customization aspects. But this approach is lack of solution in how to deal with interaction and data sharing among tenants.

Traditional data architecture for one-tenant software is based on an organization tree. All organizations are under the tree with hierarchical levels. Higher level organization

administrator can monitor lower level organization's business. Data sharing and interaction among sub organizations is easy to be implemented. But on the contrary, it is lack of configurability and data isolation ability for subordinate organization.

SOA based distributed architecture design follows the approach that one application represents one service for one subordinate organization, cross-sub-organization's data sharing and interaction is implemented through workflow with the help of ESB and Business Process Management (BPM) [12] service. It provides a possibility for sub organization's customized configuration, public data sharing and private data isolation. But this approach always is complex and costs a lot of manpower and material resources in practice.

The data architecture for the model proposed in the paper is a hybrid one. It supports both unified interaction management and multi-tenant data isolation. It owns a unified organization tree and user information storage, which means only one piece of information stored for an organization or an employee, no matter how many tenants the organization or employee may exist in. It supports data isolation for tenants with both shared schema and separate-schema approaches. It lowers the whole system's complexity and saves resources compared with SOA based distributed architecture.

In the model, root of the unified organization tree for a large organization can be represented as a tuple $R=(T, O_{out})$, where: T is a finite set of tenants under R , $T=\{t_1, t_2, \dots, t_n\}$; O_{out} is a finite set of organizations which are under R , but not affiliated to T . Tenant t also can be presented as a tuple $t=(O, custom)$. O is a finite set of organizations under T , $O=\{o_1, o_2, \dots, o_n\}$ $o_i \in t$. $Custom$ represents the customized configuration of t . Interaction between tenants is implemented by organizations and theirs employees. Organization can be represented as a tuple $o=(E_c, O_b, D_b, D_s)$, where: E_c is a finite set of (*employee, type*) pairs, *type* includes directly under and connect, which correspond to the relationship of RO and

employee and the relationship of VO and employee separately. O_t is a finite set of (*o_b, relationship*) pairs, o_t is target organization, *relationship* includes two types: superior-subordinate and lean-on, superior-subordinate means there exists direct superior-subordinate relationship from o to o_t in the organizational tree, lean-on means o is a VO and leans on RO o_t ; D_b is a finite set of business data of this organization; and D_c is a finite set of resource data of this organization. Figure 7 is an implement of the data architecture for this model.

Organization is the core in the model. An organization is directly linked with its parent by attribute *parentid*. Attribute *type* is used to identify the type of an organization, which may be RO or VO. *Tenant* is used to identify the highest layer organization in a tenant. It is helpful to model the *Inclusion Relationship* between two tenants, which is introduced before. That it does not set an organization into a tenant makes it is possible that parent tenant can access son tenant data, and an employee under the son tenant is affiliated to the two tenants at the same time. VO can lean on a RO with a *leanon_org* association. When the VO and the target RO are in different tenants, the two tenants match *Lean-on Relationship*. *Employee* is directly linked to RO with *orgid* attribute. It also can be connected by VO with *employee_vorg* association. When the connected employee and VO are in different tenants, the two tenants match *Incidence Relationship*. *Resourcedata* refers to *resource data* defined before. They are separated with attribute *tenantid* by shared schema approach. Since all data are stored in one schema, it is easy to implement data access authority control and data sharing with attribute *openscope*. *Businessdata* corresponds to *business data*. Since the intersection of this part data among tenants is small, both separate-schema and shared schema approaches are suitable for data isolation among tenants. But when the amount of *business data* stored is big, separate-schema approach is a better choice, for it can separate data into different schema and disperse external access pressure. *Workflow* is of great benefit

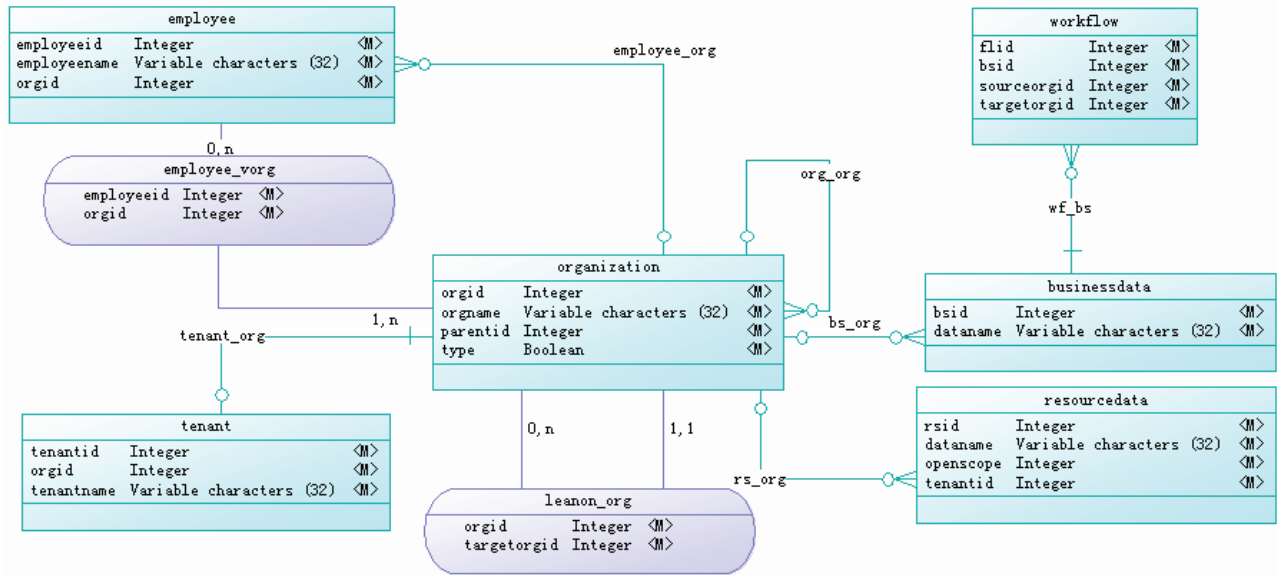


Fig.7. Data Structure of the SaaS Model

to *business data* interaction between organizations. When a piece of *business data* is sent from one organization to another, a copy of the data is saved into the schema which target organization belongs to with the change of attribute values of *sourceorgid* and *targetorgid*. Because of the existence of records in *Workflow*, a complete flow sequence can be found from any organization node in the flow, and it is easy to implement data synchronization for all copies in the workflow sequence.

III. CASE STUDY: CONTINUING EDUCATION BASE IN CHINESE ACADEMY OF SCIENCES

Chinese Academy of Sciences (CAS) is a large organization with more than 100 institutes and numerous business organizations in China. Business interaction and resource sharing is very universal among those subordinate organizations. In training management field, *Department of Training Management of CAS* is the highest training management organization in CAS. National Professional and Technical Personnel Continuing Education Base in Beijing branch of CAS (*Continuing Education Base*) is a national professional and technical personnel continuing education base in CAS. It takes on the responsibility of organizing institutes under Beijing branch to hold various continuing education training activities for the society every year. Its office staffs come from *Headquarters* of CAS and major work includes monitoring training implementation, gathering training statistics and online sending e-training certificates. Continuing education training is part of training courses in an institute. Sharing training program and courseware resource among institutes is encouraged in CAS.

If a platform based on traditional one-tenant application architecture is set up for the training management business of CAS, especially *Continuing Education Base*. It can easily implement interaction and data sharing among organizations. Through a mark, a training program can be catalogued into continuing education training. Employees under *Headquarters* or *Institutes* can deal with both the work of *Continuing Education Base* and their affiliated organizations with one account, like *Employee 1*, *Employee 2* and *Employee 3* in Figure 8. This model's shortage is difficult to implement service customization for each organization, such as each organization's logo, portal and configurable data fields.

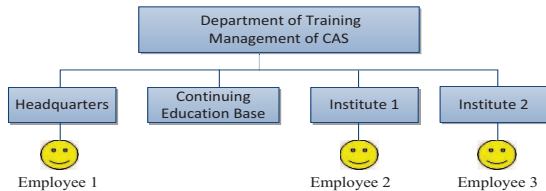


Fig. 8. One-tenant architecture model for Continuing Education Base case

Both common SaaS architecture and SOA based distributed design can solve this issue in customization aspect. But they have their own shortages. Figure 9 is a reference model for the case based on common SaaS architecture. In this figure, both organizations and employees are separated into different tenants. It is difficult to implement training program

and courseware resource information sharing, interaction among *Continuing Education Base* and each *Institutes* (gather training statistics and send e-training certificates), and switch between *Headquarters* and *Continuing Education Base* with one account for an employee under *Headquarters*, like *Employee 1* in Figure 9.

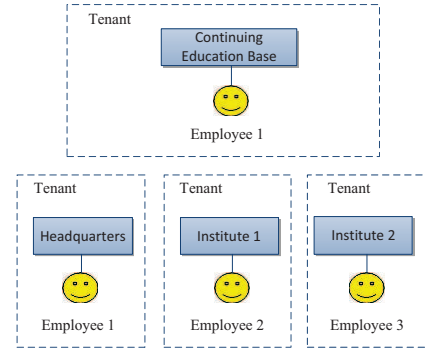


Fig. 9. Common SaaS model for Continuing Education Base case

SOA based distributed design can solve issues in cross-organization aspect with the help of other software tools. Figure 10 is a reference implement for the case based on this design. In this architecture, each organization (*Continuing Education Base*, *Headquarters* and *Institutes*) owns an independent system. Employees under *Headquarters* have two accounts, one in *Headquarters* and the other in *Continuing Education Base*. With the help of authentication system, they can easily switch business between *Headquarters* and *Continuing Education Base*. Sending training statistics from *Institute* to *Continuing Education Base* and sending e-training certificates from *Continuing Education Base* to *Institute* are predefined as workflows and stored in a BPM system. An ESB system is used to support workflows and distribution of training program and courseware resource information among *Institutes*. This architecture can satisfy all the multi-tenancy requirements of a large organization in functional level, but new tools' usage makes the whole architecture complex and difficult to be maintained.

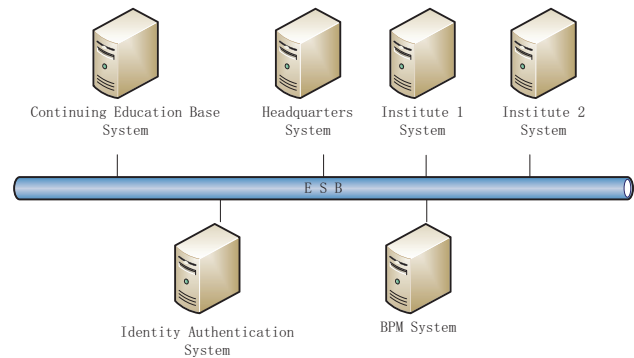


Fig. 10. SOA based distributed design for Continuing Education Base case

The new SaaS model proposed in this paper is especially useful in this scenario. It can overcome all those issues mentioned. Following is a simplified model set up for CAS, especially focusing on *Continue Education Base*, with this approach.

As Figure 11 shows, tenant is registered institute or cross-institute business organization (like *Continuing Education Base*) in this case. Since all data object except *Employee* and *Organization* in the model have been separated by tenant, customized configuration, no matter data field or platform style, can be separated well for each tenant. *Continuing Education Base* is a tenant with cross-institute business characters. For the convenience of management, *Office* and *Training Centers* are set as two organizations under *Continuing Education Base* though they don't exist in fact. *Office* is connected with related employees under *Headquarters*. Since only one piece of employee information exists in the model, it is convenient for employee's work switch between *Headquarters* and *Continuing Education Base* with no need to consider of how to identify two accounts that belong to the same employee. Training program and courseware resource information are catalogued into *resource data*, and the sharing scope is determined by their owners. Training statistics and e-training certificate are *business data*. *Lean-on relationship* exists between *Training Center* and *Institute*. Each *Institute* can use the workflow between *Training Center* and *Continuing Education Base* to send training statistics to *Continuing Education Base* and receive e-training certificates from *Continuing Education Base*. The whole model needs only one database to support with no SOA tools help and more simple than SOA based distributed architecture.

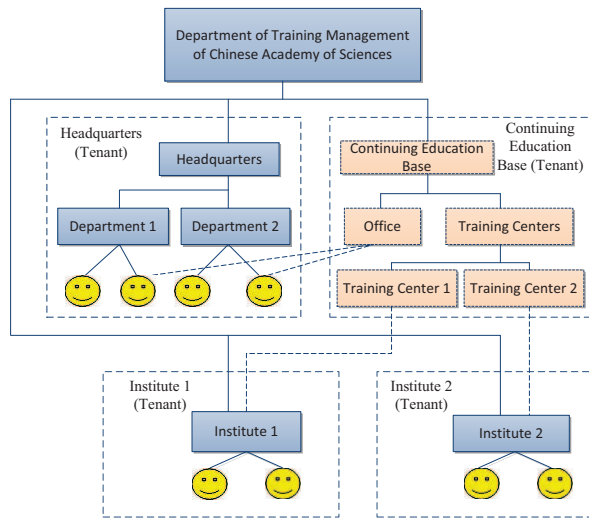


Fig. 11. The new SaaS model proposed for Continuing Education Base case

Training Cloud Platform for Science and Research is a software implement of the model case and it provides training management and self-help online course learning services for organizations in CAS. It mainly includes *portal system*, *training & e-learning environment*, *cloud management system*, *courseware access router* module and *distributed courseware*

storage environment. The whole platform is hosted by Infrastructure as a Service (IaaS).

Portal system is used for tenants' registration and information's publication. *Cloud management system* is used for courseware delivery, platform level's data statistics and cloud environment's monitor. *Training & e-learning environment* is a platform hosting web applications to provide SaaS service for tenants. It is composed by numbers of independent virtual machines. One web application is deployed at one machine and services one tenant. A unified database is shared by those web applications. Storage environments locate at several different cloud centers in China, and it is convenient for tenants in different places to store and access courseware in nearby environment. Unlike common video website can simply depend on peer-to-peer (P2P) technology [13] to broadcast programs, courseware types for an e-learning platform are various and most of them (such as Shareable Content Object Reference Model (SCORM) standard [14] courseware, three-part-separated screen courseware and flash courseware) do not support P2P and need unbroken storage. In this platform, the rule for courseware storage and distribution is designed as follows. Each center has one space for shared resource storage and many size-specified separated spaces for each tenant's private resource storage. Private resource cannot be accessed by other tenants. Public resource can be distributed from the storage space of its owner tenant to any shared one and accessed by other tenants. Figure 12 is a sample of resource storage and distribution, where arrow indicates resource distributed route. For example, public resource of *Tenant C* in *Storage 1* can be distributed to *Share 2* in *Storage 2* and *Share 3* in *Storage 3*, its copies in *Share 2* and *Share 3* can be accessed by other tenants. The best route for accessing a package of courseware can be given by *Courseware Access Router* module through the performance data of shared space in each storage environment are analyzed.

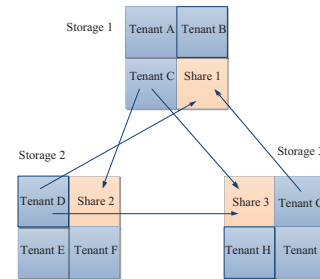


Fig. 12. Sample of resource data storage and distribution

Figure 13 shows the whole platform's architecture.

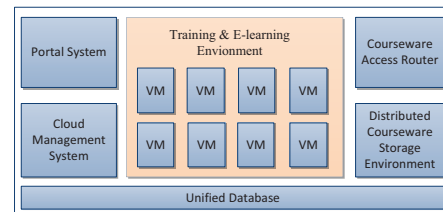


Fig. 13. Architecture of Training Cloud Platform for Science and Research

If an organization's registration is approved, the cloud platform will create a new tenant account, add a virtual machine with web application into *Training & E-learning Environment* and create a separated space in *Distributed Courseware Storage Environment* for it. When the organization administrator logs on with the tenant account, he can customize the service and begin to use. The platform supports customized configuration of user interface style, menu, table field and statistics and has been steadily running since Jun.1st, 2013. More than 70 institutes and business organizations have registered as tenants by Oct. 20th, 2013 with more than 8000 employees. And there are still many organizations that wish to use.

IV. RELATED WORK

Three models (separate databases; shared database, separate schema; shared database, shared schema) of managing multi-tenant data for SaaS are proposed in [3]. Those models have been adopted as main references to implement multi-tenancy for designing SaaS application by both academia and industry. Based on the three models, [15] introduces two mixed data model: Mixed Database data Model (MDM) and Mixed Schema data Model (MSM). MSM is a mix of shared-database-separate-schema model and shared-database-shared-schema model. It is the same as the approach in the model we proposed for *resource data* and *business data* storage. Compared with other models, its challenge is the difficulty in implementation, but advantage is it can satisfy different data isolation level requirement for different business with a relative low cost in an application. A Multi-tenancy data migration approach with minor modification of the codes is also given. Reference [16] introduces a framework called Codename^{MT} which can be used to transform single-tenant application into multi-tenant one. It uses a global session ticket to identify user and his access authentication across organizations. This approach solves the challenge of one user existing in more than one tenant in a SaaS platform, but how to gather data from different accounts for the user is not solved. The purpose of [17] is partly similar with us. It proposes a SaaS solution for integration of similar applications without SOA, which avoids the shortages of SOA based Enterprise Application Integration (EAI) [18], such as high complexity, and high cost of maintaining multiple applications in a large organization. Reference [19] presents an experimental comparison of Private Tables, Extension Tables, Sparse Columns, XML in IBM DB, Pivot Tables in HBase for implementing flexible schemas for SaaS, Extension Tables is the most popular one among those five techniques, and has already won success in Salesforce.com [20]. In this architecture, every tenant-configurable aspects data are stored in metadata tables. Runtime engine generates customized application for individual tenant with tenant-specific metadata at runtime [21]. Our model focuses on the management and interaction among tenants, and the configurability of tenant data is not our emphasis. But this technology has been accepted by *Training Cloud Platform for Science and Research* to customize services for tenants.

Reference [22] introduces an educational video publishing and sharing system based on internet. It maintains a course

directory tree in each video course server, and a kind of asynchronous message communication mechanism is applied among servers to keep a consistent view of the course directory tree. This system had a nice support history for large volume of courseware storage and access as author said. This approach is a valuable reference for the further research of courseware storage, distribution and access for *Training Cloud Platform for Science and Research*.

V. CONCLUSION AND FUTURE WORK

This paper has presented a novel multi-tenant SaaS model for large organization. It absorbs advantages from traditional one-tenant software architecture, common SaaS model and SOA based distributed design. Its strength includes matching a large organization's sophisticated structure, providing customized space for sub organizations, supporting various data interaction and sharing among them, unified account for a cross-tenant user. And at the same time, it lowers system's complexity and saves resources. It especially meets the trend of data sharing and socialization, and has broad development space and potential application in enterprise SaaS application domain. There are still many places to be improved for its engineering application. Keeping employee information unique among tenants is not easy in practice at present. This sometimes causes statistics is not precise and cross-tenant logon is difficult. Workflow between tenants depends on the predefined ones or the setting by platform level administrator and is lack of ability for tenant to define by themselves. This sometimes makes trouble for tenant's self-usage. Storage of all resource data in the same schema may bring performance problem in the future when the amount of data stored is bigger [23]. It needs both the improvement of model itself and the support of new style application architecture. In the future, on one hand we will promote the flexibility and usability of the model and explore how to satisfy requirements of more business scenarios with more abundant real cases; on the other hand research on high availability SaaS architecture, which brings better engineering of the model to support high concurrency and big data in a large organization.

ACKNOWLEDGMENT

We thank Beijing Branch of CAS for their help in collecting and sorting the business requirements in the case of continuing education base in Chinese Academy of Sciences.

This paper is supported by the Educational Informationization Special Fund of Chinese Academy of Sciences under the name "Continuing Educational Informationization and Science - Educational Service Project", CAS-Guangdong Strategic Cooperation Project under the name "Scientific Resource Sharing and Communication Platform for Middle and Primary School".

REFERENCES

- [1] http://en.wikipedia.org/wiki/Software_as_a_service
- [2] G. Carraro, "SaaS simple maturity model," 6 Mar 2006. [Online]. <http://blogs.msdn.com/b/gianpaolo/archive/2006/03/06/544354.aspx>
- [3] F. Chong, G. Carraro, and R. Wolter, "Multi-tenant data architecture," June 2006. [Online]. Available: <http://msdn.microsoft.com/en us/library/aa479086.aspx>

- [4] Z. Diamadi, A. Dubey, D. Pleasance, and A. Vora, "Winning in the SMB cloud," July 2011. [Online]. Available: http://www.mckinsey.com/client_service/high_tech/latest_thinking/winning_in_the_smb_cloud
- [5] S. Taylor, A. Young, J. Macaulay, "Small businesses ride the cloud: SMB cloud watch—U.S. survey results," February 2010. [Online]. Available: http://www.cisco.com/web/about/ac79/docs/pov/SMB_Cloud_Watch_FINAL_FINAL_021810.pdf
- [6] N. Josuttis, *SOA in Practice*. O'Reilly Media, Inc., 2007.
- [7] D. A. Chappell, *Enterprise Service Bus*. O'Reilly Media, Inc., 2004.
- [8] Stephen P. Robbins, Mary Coulter, "Common structure designs," in *Management* (10th Edition), Prentice Hall, pp. 193-195, Nov. 2011
- [9] R. Krebs, C. Momm, and S. Konev, "Architectural Concerns in multi-tenant SaaS applications," in *Proceedings of the 2nd International Conference on Cloud Computing and Service Science (CLOSER'12)*. SciTePress, April 2012.
- [10] R. Mietzner, T. Unger, R. Titze, and F. Leymann, "Combining different multi-tenancy patterns in service-oriented applications," in *Proceedings of the 13th IEEE Enterprise Distributed Object Conference (EDOC 2009)*. IEEE, September 2009.
- [11] C. Guo, W. Sun, Y. Huang, Z. Wang, and B. Gao, "A framework for native multi-tenancy application development and management," in *Proceedings of the 9th IEEE International Conference on E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'07)*. IEEE, 2007
- [12] S. Howard, and F. Peter, *Business Process Management*. Meghan Kiffer Pr, Oct. 2006.
- [13] <http://en.wikipedia.org/wiki/Peer-to-peer>
- [14] <http://www.adlnet.gov/scorm/>
- [15] L. Jiang, J. Cao, P. Li, and Q. Zhu, "A mixed multi-tenancy data model and its migration approach for the SaaS application," in *Proceedings of 2012 IEEE Asia-Pacific Services Computing Conference (APSCC 2012)*. IEEE, Dec. 2012
- [16] C. Bezemer and A. Zaidman, "Enabling multi-tenancy: an industrial experience report," in *Proceedings of 26th IEEE International Conference on Software Maintenance*. IEEE, 2010
- [17] Z. Zhu, L. Chen, J. Song, and G. Liu, "Applying SaaS architecture to large enterprises for enterprise application integration," in *Proceedings of 2nd International Conference on Information Science and Engineering (ICISE 2010)*. IEEE, 2010.
- [18] D. S. Linthicum, *Enterprise Application Integration*, Addison-Wesley Pub Co, Nov. 12th, 1999.
- [19] S. Aulbach, D. Jacobs, A. Kemper, and M. Seibold, "A comparison of flexible schemas for software as a service," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 881-888
- [20] C. D. Weissman and S. Bobrowski, "The design of the force.com multitenant internet application development platform," in *Proceedings of the SIGMOD*. IEEE, 2009, pp. 889-896 .
- [21] S. Kang, S. Kang, and S. Hur, "A design of the conceptual architecture for a multitenant SaaS application platform," in *Proceedings of First ACIS/JNU International Conference on Computers, Networks, Systems, and Industrial Engineering*. IEEE 2011, pp 462 – 467
- [22] L. Cheng, and X. Li, "Realcourse: A distributed course video sharing system," in *e-Science Technology & Application*. Beijing, vol.2 No.2, pp.34-41, March 2011.
- [23] T. Kwok, T. Nguyen and Li. Lam, "A software as a service with multi-tenancy support for an electronic contract management application," in *Proceedings of the International Conference on Services Computing (SCC)*. IEEE CS, 2008, pp.179-186.