Applying SaaS Architecture to Large Enterprises for Enterprise Application Integration

Zhiliang Zhu, Long Chen, Jie Song, Guoqi Liu
College of Software, Northeastern University, Shenyang, P.R. China
E-mail: zzl@mail.neu.edu.cn, 1988clong@sina.com, songjie@mail.neu.edu.cn, giantliu@126.com

Abstract—SaaS (Software-as-a-Service) has gained an increasing momentum, and it is being adopted at a very fast pace. But SaaS is widely considered that it's not suitable for Large and Medium-size Enterprises (LME), meanwhile Enterprise Application Integration (EAI) is the key issue in LME, traditional SOA based EAI approaches have some shortcomings. In this paper, we propose that SaaS can be well used in LME for EAI. Based on this, we propose *SaaSified* approach which re-architects one analog of legacy applications to SaaS architecture, makes the rest applications can be configured, further solves the EAI problem in LME. We update the SaaS maturity model to evaluate the *SaaSified* approach.

Keywords; SaaS; Large and Medium-size Enterprises; Enterprise application integration; SaaS maturity model

I. INTRODUCTION

It was only in 2005-2006 that the SaaS (Software-as-a-Service) wave gained momentum[1], because not only the Internet had become high-speed and affordable, but also the customers had started to be comfortable with doing business on the web[2]. Today SaaS has become an acceptable mainstream business model for not only the end users but also enterprises[3].

People believe that SaaS is not suitable for Large and Medium-size Enterprises (LME) in many aspects such as data security, configurability, SLA (Service Level Agreement) and sustainability.

Enterprise Application Integration (EAI) can realize effective combination of various independent systems and data sharing between all processes of an enterprise. With the development of a LME, there must be more and more *analogs* (term for the applications are similar in function) for branches. As a result, it may be difficult in communicating between applications, because some of the applications may be developed in different languages or based on heterogeneous data. In this case, EAI is urgently on demand.

Traditionally, there are two patterns that EAI systems implement:

- **Mediation**: the EAI system acts as the go-between or broker between multiple applications.
- **Federation**: the EAI system acts as the overarching facade across multiple applications.

Most of EAI related previous works[4][5][6] are based on mediation or federation, in which SOA architecture are adopted as the implementation of interoperability. But there exists some shortcomings in the two patterns. **Firstly**, wide knowledge is required when integrating the heterogeneous applications; **Secondly**, the wrappers for interoperability will cause high complexity in both interface and logic; **Thirdly**, the cost of maintaining multiple applications is still high even if there are integrated. For example, most EAI projects usually start off as point-to-point efforts, it will create lots of connections as the number of applications increases so that the exposed interfaces are unmanageable.

Above all EAI is a key issue of LME, its implementation has some drawbacks. In this paper, we propose an approach that can partly solve the EAI problem by SaaS architecture, thus change the misconception about SaaS used in LME. The EAI problems can be solved by customizing a single instance for multiple tenants, instead of multiple applications. We *SaaSify* (terms for converting an application into SaaS architecture) one of analogs in LME's legacy applications, making it can be customized to the other applications by clients, and then we update the SaaS maturity model for such approach.

The rest of this paper is organized as follows. In the section II, we introduce the related works of SaaS. And following section III explains how SaaS can be used in LME and further updates the SaaS maturity model to evaluate the proposed approach. Section IV introduces the steps of the *SaaSified* approach, including implementing the configuration of user interface, business logic and data. Section Ⅴ concludes the paper.

II. RELATED WORKS

Recently, SaaS has been researched in many fields, such as creating a fine-grained operating cost model for SaaS in order to provide a secure and trustworthy service consumption environment[7], creating methods and communication infrastructures that can enable distributed SaaS applications over the data network[8], enabling progressive migration of multi-version applications in SaaS via evolving schema [9] and establishing a fitness evaluation model to estimate whether SaaS is suitable for the information system evaluated[10]. These are initial works of SaaS but none of them is used for EAI in LME.

Simultaneously, SaaS providers have made a great deal of efforts on providing all kinds of services to customers. Most of the SaaS companies such as DataMentor, LiveRoute, Alisoft are dedicated to make SaaS services for the small enterprises. They have built several services such as ERP (Enterprise

Supported by Shenyang Financial Foundation Project (1091176-1-00) and Liaoning Province Natural Science Foundation (20092006)

978-1-4244- 7618-3 /10/$26.00 ©2010 IEEE

Resource Planning), financial software and CRM (Customer Relationship Management) for the small enterprises.

Both in theory and in practice, SaaS for small enterprises is the mainstream by now. But SaaS can be used in LME too. For example, the SaaS services such as net meeting or online training can be used in LME because this kind of services doesn't have the data security problem and legacy system problem. Requirements can be satisfied meanwhile the cost can be cut down. The practices of WebEx and G-net can prove SaaS can be used in LME.

Recently, research of EAI focuses on flexibility, easy expansion and loose coupling, such as proposing a SOA-based solution to the integration of enterprise application realized via J2EE[4], and proposing an EAI framework based on Web services[5]. Some other works such as [6][11][12] also adopt the SOA and Web services, there have same shortcomings such as wide knowledge required, high complexity of interface and high cost for support comparing with the proposed *SaaSified* approach.

The above cases show that SaaS services can be used in both small enterprises and LME. But none of them shows that the SaaS service is used for EAI in LME. So the proposed idea is innovative to the best of our knowledge.

III. SAAS FOR EAI

In this section, we will explain why SaaS architecture is suitable for EAI in LME. And we will update the SaaS maturity model to meet the practical requirements of LME.

A. SaaSify Model for EAI

Currently, the SOA based EAI in LME is popular. The SOA method focuses on interoperability. Original applications are encapsulated as services according to the same standard so that they can interoperate each other. The SOA solution for EAI in LME is shown as Figure 1.

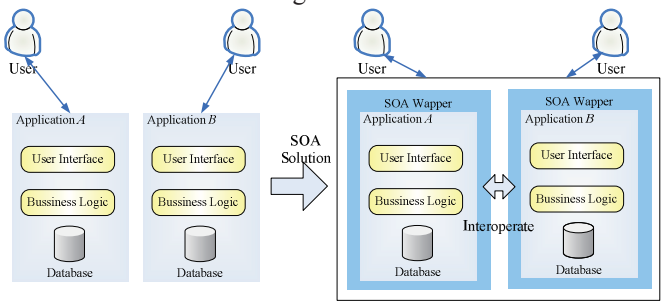

Figure 1. The SOA solution for EAI

After the target analog is selected, it is *SaaSified* to reach high configurability. The rest of analogs which are removed from context can be configured from the *SaaSified* analog by customers (See Figure 2). The SaaS solution for EAI in LME is shown in Figure 2:

In Figure 2, appliaction *A* and appliaction *B* are similar. *A* is chosen to be re-written as SaaS. The users of *B* can configure the new SaaS appliaction as the "original" *B* which is shown as the dashed part. When they use the SaaS appliaction, there will be no differences from using the original *B*.

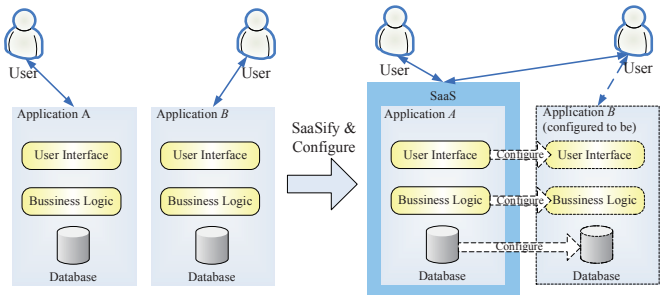

Figure 2. The SaaS solution for EAI

By the proposed approach, some of the problems existed in EAI such as data format will be solved. Compared with the interoperation-ability of SOA, what SaaS approach emphasizes is configurability. There have same shortcomings such as wide knowledge required, high complexity of interface and high cost for support comparing with the proposed *SaaSified* approach.

Due to the aspects why SaaS is not suitable for LME (see section I), if the SaaS architecture is adopted by LME, there must be some changes in SaaS business concept. The SaaS application will be deployed in LME's context instead of in vendor's context. In this situation, there is no need for LME to worry about data security, SLA and sustainability of SaaS. At the same time, SaaS applications will cut down the cost of maintaining multiple applications in each branch.

As for the configurability of SaaS, it depends on the maturity level of the SaaS application. So an updated SaaS maturity model is proposed in next section to evaluate the configurability of SaaS.

B. Updating SaaS maturity model

Due to the SaaS solution for EAI, the configurability of the SaaS application should be paid close attention to. So the SaaS maturity should be updated for proposed approach. In *Level 2*, *Level 3* and *Level 4*, the SaaS application can be configured. So we divide them into three sub-levels individually to measure the configurability. The configurability levels which contains *UI level*, *UI&Data level* and *UI&Data&Logic level*, are added to original *Level 2*, *3* and *4*. Then the updated maturity model which is shown as Figure 4 contains two dimensions, the one is original level 1 to 4, the other is added levels for configurability.

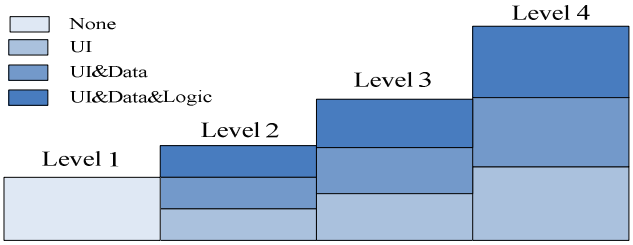

Figure 3. Updated SaaS maturity model

In the configurability levels, the reasons why UI configurability is chosen as the first level are: (1) User interfaces are the entrances of the applications, configurability of UI is the basic requirements of in a SaaS applications; (2)

configurability of UI is the easiest to be *SaaSified* comparing with that of data and logic; (3) UI is important to the users to interact with the application, the configuration of UI is visual and appreciable. As for the configuration of data and logic, the implement of data configuration will cause the changes in logic layer such as data logic. It's inevitable that the configuration of data is more basic than the configuration of logic. So the priority of the configurability levels should be UI, data and logic.

By now the updated SaaS maturity model, it is more clear and detailed to evaluate EAI oriented SaaS application in LME.

IV. *SAASIFIED* APPROACHES

In this section, we will explain how to make data, user interfaces and business logic configurable so that the SaaS application can be configured as the original ones.

A. Configurability of User Interface

UI's configurability can be implemented by two approaches: system menu configuration and page content configuration.

1) System menu configuration

Every tenant has his own requirement for the application. System menu is very important in navigation, so it should be able to be customized for each of the tenants in menu hierarchy, menu name, etc. When making the menu configurable, some issues should be considered such as: (1) each tenant should have his own menu (2) one menu item links to an atom function (3) the menu items should be organized as tree (4) the menu items in the same level should have order to be displayed. The ideal design is presented as Figure 4:

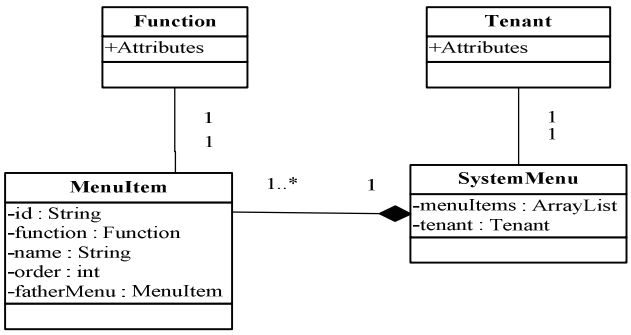

Figure 4. Class diagram of system menu configuration

In Figure 4, *menuItems* is a container of *MenuItem* and it can build a hierarchical system menu with the help of *fatherMenu* in *MenuItem.* If the application is designed like what's shown above, it'll be flexible in system menu configuration and helpful for function atomization.

2) Page content configuration

Similar with system menu, the page content is also important to the tenant to interact with the system. Tenants may have some requirements for the number, position and order of the elements in pages. So it's important to make page content configurable.

B. Configurability of Data

1) Customized field

Customized field is the most common solution for configurability. In this solution, customized fields will be added to tables depending on tenants' need. For example tenant *A* want field *sex*, *job* to store his extra data beside the basic data, meanwhile tenant *B* want field *birthday* to store his extra data. Based on this situation, the table may be designed as TABLE I:

TABLE I. CUSTOMIZED FIELD

ID	*Name*	*Age*	*Sex*	*Job*	*Birthday*
1	Mary	21	Female.	Singer.	
2	Chirs	22			07-07.

This is the easiest way to meet all the tenants' need. But this method causes chaos in tables. There are lots of fields in a table and most of them are meaningless to some tenants. So this isn't an ideal way to make data configurable.

2) Pre-assigned field

Pre-assigned field means some fields are kept in a table for tenants to extend their need. The table may be designed as TABLE II:

TABLE II. PRE-ASSIGNED FIELD(TYPE LIMITED)

ID	*Name*	*Age*	*ExtInt*	*ExtStr*	*ExtDate*
1	Mary	21	3	Google	
2	Chirs	22		Singer	2010-6

This solution is the same as "customized field" using extra fields in a table to extend data. But the pre-assigned fields are without specific meaning. Such as the *ExtStr* field, tenant *A* can use it to store *company name* and tenant *B* can use it to stand for *job*. In the TABLE II, the data type is associated with the field. For more flexibility, data type can be removed as shown in TABLE III:

TABLE III. PRE-ASSIGNED FIELD(COMMON)

ID	*Name*	*Age*	*Ext1*	*Ext2*	*Ext3*
1	Mary	21	3	Google	
2	Chirs	22		Singer	2010-6

In this table, all the stored data are converted into String type in advance. It can satisfy all of the tenants when the number of pre-assigned fields equals to the maximum number required by tenants. Though this method saves the storage space, it will be a problem when the data are reverted into the original type. So another table contains the information of the data is necessary. It is designed as TABLE IV:

TABLE IV. ASSISTED TABLE OF TABLE III

ID	*UserID*	*Table*	*Column*	*Content*	*Type*
11	1	user	Ext1	Grade	int
22	2	user	Ext2	Job	string

With this table, the real meaning and type of the data can be understandable.But it still needs to figure out how many extra fields should be kept in tables at first..

3) Name-Value pair

Name-Value pair changes the "extending table horizontally by adding fields in table" into the "extending data vertically by adding data table". In this solution a former extra field is changed into rows of a new table. The relationship of tables is shown as following Figure 5:

It's more flexible to extend data by adding a new table than adding some extra fields in tables. In this way the extending data can be added unlimitedly, without idle and unnecessary fields wasting resource and space. However, this solution makes operation on the data more complex.

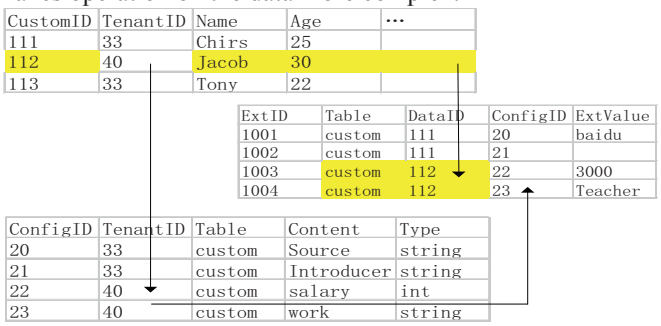

Figure 5. Name-Value pair

In the design phase of *SaaSifying*, factors like development ability and requirements of extension should be considered to choose the best solution.

C. Configurability of Logic

Undoubtedly, there must be some functional differences between analogs of legacy software. If branches want to keep former business operations in the new SaaS application, the business operations existed in the legacies must be merged.

Logic division is to divide the whole application into the basic and independent atomic functions. When an application is divided, there are some basic rules should be followed. (1) Every function is valuable. (2) Every function can't be divided into smaller functions. (3) Functions shouldn't be overlapping. (4) Functions shouldn't depend on each other in a circle. (5) The application can be reverted by the functions.

But not all the atomic functions can be used individually. Some of them depend on others. It is the so-called functional dependency that some functions can not be used without other function. To make the division clear, the definition of the atomic function is necessary. It should be designed as Figure 6.

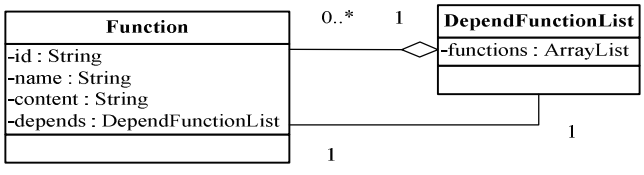

Figure 6. Class diagram of function dependency

Id, *name* and *content* are used to describe the basic information of the function. And *depends* is used to show the dependency relationship of the function.

Firstly, logics existed in legacies can be divided atomically. Secondly the atomic functions can be merged into the new SaaS application. Lastly with the help of system menu configuration, tenants can get the same function and experience in the new SaaS application.

V. CONCLUSIONS

In this paper, A *SaaSified* approach for Enterprise Application Integration (EAI) in the Large and Medium-size Enterprises (LME) is proposed. It re-architects one selected analogs of legacy applications in different branches, making the rest applications can be configured from the *SaaSified* one by customs, further solved the EAI problems, and proves then SaaS architecture can be well applied in LME. The primary works of this research are following five aspects:

- Analyze the misunderstanding of SaaS, the drawbacks of traditional EAI approaches.
- Propose a *SaaSified* approach for EAI in LME.
- Update the SaaS maturity model for proposed approach.
- Explain how to select and *SaaSify* the target application step by step to implement the EAI.

The proposed approach reduces the resistance of applying SaaS architecture to LME, and avoids the shortcomings of traditional EAI approaches. The future works include introducing SOA features to the *SaaSified* applications, and applying the other XaaS, such as PaaS and IaaS to the Large and Medium-size Enterprises

REFERENCES

[1] "New MidMarket | Enterprise SaaS Adoption Research", June 2008, http://conformity.wordpress.com/2008/06/05/new-midmarketenterprise-saas-adoption-research/

[2] "Special report: SaaS set for major boom, says Gartner", Aug 2007, http://www.mycustomer.com/cgibin/item.cgi?id=133195

[3] Saugatuck Research, "Enterprises Ready or Not: SaaS Enters Mainstream", July 2008

[4] Xu He, Hongqi Li, Qiaoyan Ding: The SOA-Based Solution for Distributed Enterprise Application Integration. IFCSTA '09: 330 - 336

[5] Xiaogang Ji: A Web-based Enterprise Application Integration solution. ICCSIT'09 : 135 - 138

[6] Mengjian Chen: Research and Implementation on Enterprise Application Integration Platform. IFITA '09 : 93-96

[7] Yücel Karabulut, Ike Nassi: Secure Enterprise Services Consumption for SaaS Technology Platforms. ICDE 2009:1749-1756

[8] Feng Liu, Li Li, Wu Chou: Communications Enablement of Software-as-a-Service (SaaS) Applications. GLOBECOM 2009:1-8

[9] Jianfeng Yan, Bo Zhang: Support Multi-version Applications in SaaS via Progressive Schema Evolution. ICDE 2009:1717-1724

[10] Yonghe Lu, Bing Sun: The Fitness Evaluation Model of SAAS for Enterprise Information System. ICEBE 2009:507-511

[11] Hanwei Chen, Jianwei Yin, Lu Jin: JTang Synergy: A Service Oriented Architecture for Enterprise Application Integration. CSCWD.2007: 502 – 507

[12] Scheibler, T., Mietzner, R, Leymann, F. : EAI as a Service - Combining the Power of Executable EAI Patterns and SaaS, EDOC '08: 107-116