Towards Implementation and Design of Multi-tenant SaaS Based on Variabiliy Management Mechanisms

Houda Kriouile, Bouchra El Asri, M'barek El Haloui and Asmae Benali
IMS Team, SIME Laboratory
ENSIAS, Mohammed V University
Rabat, Morocco

Email: [houda.kriouile, mbarek.haloui, asmae.benali]@um5s.net.ma,
elasri@ensias.ma

Abstract—**Software as a Service (SaaS) is a form of Cloud Computing that involves offering software services on-line and on-demand via Internet deemed a main delivery support. Multi-tenancy is a tool to exploit economies of scale widely promoted by SaaS model. Even so, the ability of a SaaS application to be adapted to individual tenant's needs seem to be a major requirement. Thus, in this paper we introduce an approach proposing a more flexible and reusable SaaS system for Multi-tenant SaaS application. The approach introduced is based on integrating a deployment variability that enables the customers to choose with which others tenants they want or do not want to share instances with a functional variability using Rich-Variant Components.**

Keywords-SaaS; Rich-Variant Component; Functional Variability; Deployment Variability; Multi-tenancy.

I. INTRODUCTION

With the age of Cloud Computing, several forms of Cloud services have emerged thanks to the Internet services development and the customers' needs evolution, in particularly the Software as a Service (SaaS) form. The latter refers to software distribution model in which applications are hosted by a service provider and made availability to customers over a network, typically the Internet. A key enabler in Cloud Computing to exploit economies of scale is the multi-tenancy, a notion of sharing resources among a large group of customer organizations, called tenants. But, the multi-tenant application responds only to needs that are common to all tenants. So, a plethora of work research has been performed to facilitate SaaS applications customization according to the tenant-specific requirements by exploiting benefits of both variability management and multi-tenancy [1][2][3]. In the same vein, we aim to create a flexible and reusable environment enabling greater flexibility and suppleness for customers while leveraging the economies of scale. For this purpose, we propose a solution integrating a functional variability at application components level and a deployment variability at multi-tenants level as well.

This paper is divided into five main sections along with this introduction. Section II provides an overview on variability management mechanisms, Cloud Computing and Multi-tenancy as a background concepts for our work research, then deals with the incentives and motivations for the proposed approach. Section III presents several approaches studied as related work and positions our contribution. Section IV initiates our contribution which consists on integrating functional and deployment variabilities for SaaS applications. Section V provides some outstanding of our approach and future works. Finally, Section VI is a conclusion of the paper.

II. BACKGOUND AND MOTIVATION

In the following subsection, some variability management mechanisms are presented, followed by a short introduction to the Cloud Computing and the Multi-tenancy notions as a background for our work. Finally, the motivation of our contribution consisting on the need of managing variability for Cloud environment is discussed.

A. Variability managment mechanisms

Variability is the ability of a software artifact to be adapted for a specific context [4]. For example, it could be the ability to be extended, configured, customized or modified. A request for change requires the evolution of systems. Therefore, the variability of the system or the software must be managed during all lifecycle's phases (e.g., the specification phase, the conception phase, the testing phase, the implementation phase, etc.).

A variety of mechanisms and approaches are proposed for the management of system's variability. These mechanisms intervene at the level of the different lifecycle's phases. Examples are cited bellow:

- Specification phase: Iqbal, Zaidi and Murtaza propose a model for requirement prioritization using Analytical Hierarchical Process (AHP) [5].
- Conception phase: Several approaches were proposed in this phase to model software product lines by using feature models as the Feature Oriented Domain Analysis (FODA) approach [6], which aims to capture the commonalities and the points of difference at requirement level. Many other approaches provided extensions to the FODA approach. Such as the Feature-Oriented Reuse Method (FORM) [7], whose main contribution is the decomposition of the feature model layers to describe different perspectives (capacity, environment, technology).
- Testing phase: Erwing and Walkingshaw propose organizing the space of all variations by dimensions, which provides scoping and structuring choices [8]. They consider the concept of "variation programming" for a flexible construction and transformation of all kinds of variation structures [8].

 ISBN: 978-1-61208-367-4

- Implementation phase: Trummer proposed a corresponding data model [9] that is based upon the Composite Application Framework (Cafe) model [1]. Applications are composed out of components that may be provisioned separately.

From the cited works above, the interest of variability management mechanisms is evident. Particularly, these mechanisms are useful for managing the functional variability and the deployment variability into Cloud environment, specially for Multi-tenant SaaS applications.

B. The Cloud Computing and Multi-tenancy

Cloud Computing as defined by the National Institute of Standards and Technology (NIST) is the access via a telecommunications network, on demand and self-service, to a shared pool of configurable computing resources [10]. Cloud Computing is the use of computing resources - hardware and software - that are provided as a service over a network, usually the Internet. Cloud Computing entrusts remote services with a user's data, software and computation [10].

Our work focuses on Cloud Computing Services from the kind of Software as a Service (SaaS). In this type of service, applications are made available to consumers. Applications can be manipulated using a web browser. As a tool to exploit economies of scale, SaaS promotes Multi-tenancy [3].

Multi-tenancy is the notion of sharing resources among a large group of customer organizations, called tenants. That is, a single application instance serves multiple customers. But, even though multiple customers use the same instance that each of which has the impression that the instance is designated only to them. This is achieved by isolating the tenants' data from each other. Compared to single-tenancy, Multi-tenancy has the advantage that infrastructure may be used most efficiently as it is feasible to host as many tenants as possible on the same instance. Thus, maintenance and operational cost of the application decreases [3]. In Multi-tenant SaaS applications, the variability may have fundamentally different sources (evolution, maintenance, tenant's requirements, etc.), but is naturally present [2].

C. Motivation: On the need of managing variability for the Cloud Environment

The emergence of Cloud Computing has necessitated more and more variability in the form of types of services, types of deployment, and the different roles of Cloud participant. Thus, variability modeling is required to manage the inherent complexity of Cloud systems.

SaaS application are consumed by many customers that have different requirements. Thus, customers that consume the same application usually exhibit varying requirements needs. Varying requirements usually necessitate varying software architectures. In other words, when applications' requirements are changed, the software architectures of these applications are modified to satisfy the changed requirements. Therefore, both requirements and architectures have intrinsic variability characteristics.

Moreover, other concerns are raised by Multi-tenancy. For example the need to ensure the correctness of all possible configuration of the application. It is not sufficient to guarantee the correctness of a single application's configuration.

On the other hand, in Multi-tenant SaaS application, the consumer does not have to worry about making updates and upgrades, adding security and system patches and ensuring service availability and performance. In addition to that, the rapid elasticity and the resource pooling are essential Cloud characteristics [10], which promote variability for Cloud Computing environment and particularly for Multi-tenant contexts.

III. RELATED WORK

Several research works have been performed in the context of architectural patterns for developing and deploying customizable Multi-tenant applications for Cloud environment. Fehling and Mietzner propose the Composite-as-a-Service (CaaS) model [11]. They show how applications built of components, using different Cloud service models, can be composed to form new applications that can be offered as a new service. These applications have been designed in the spirit of customization, thus their variability was modeled using the application model and variability model from the Cafe Framework [1], which allows exploiting economies of scale by the use of highly flexible templates enabling increasing customers base. Our work aims to exploit economies of scale from two sides by the use of Multi-tenancy and the introduction of the new concept of Multiview, that has not been used in any of the related work studied.

In the context of the Late Binding Service - which enables service loose coupling by allowing service consumers to dynamically identify services at runtime - Zaremba, Vitvar, Bhiri, Derguech and Gao present models of Expressive Search Requests and Service Offer Descriptions allowing matchmaking of highly configurable services that are dynamic and depend on request [12]. This approach can be applied to several types of services. In the remainder of their work, Zaremba, Bhiri, Vitvar and Hauswirth apply their approach [12] on the domain of Cloud Computing, more exactly on the IaaS services that are highly configurable, change dynamically and depend on requests [13]. This approach deals with a different area of cloud application which is IaaS services. Moreover, this approach does not propose a solution to exploit economies of scale and only deals with one type of variability, which is the deployment variability.

Ruehl, Wache and Verclas address the deployment variability based on the SaaS tenants requirements about sharing infrastructure, application codes or data with other tenants [3]. They propose a hybrid solution between Multi-tenancy and simple tenancy, called the mixed tenancy. The purpose of this approach is to allow the exploitation of economies of scale while avoiding the problem of customers hesitation to share with other tenants [3]. Authors focus on the deployment variability and neglect the functional variability management.

In [2], an integrated service engineering method, called service line engineering, is presented. This method supports co-existing tenant-specific configurations and that facilitates the development and management of customizable, Multi-tenant SaaS applications without compromising scalability [2]. In contrast to our approach, this method - as well as the other approaches cited - does not address to the accessibility by roles, which is allowed in our work by the use of Multiview concept.

 ISBN: 978-1-61208-367-4

The Multiview notion allows applications to dynamically change the behavior according to the enabled user's role or viewpoint.

The next section deals with the initiation and the explanation of the approach subject of our contribution, consists of integrating the functional variability with the deployment variability for Multi-tenant SaaS applications.

IV. TOWARDS THE IMPLEMENTATION AND THE DESIGN OF MULTI-TENANCY SAAS

One of the main focuses of our team research is to work on complex systems variability according to functional areas that have initiated the concept of Multiview Component [14]. The Multiview Component concept is a component model for viewpoint in the perspective of View-based Unified Modeling Language (VUML) approach [15].

Our work aims to define a way to design and descript capabilities and variability of Rich-Variant services. In this intention, our contribution is to establish a flexible and reusable SaaS environment while exploiting economies of scale. That is, our work in progress consists of providing Multi-tenant applications based on Rich-Variant Components (RVC), which allow customers to choose among other tenants who they want or do not want to share the deployment with. This implies that these composite SaaS applications are made up of a number of RVC components; each one of which provides an atomic functionality and modifies their behavior dynamically depending on the Multi-tenant variability.

A glimpse of our architectural vision for the approach is provided in Figure 1. All tenants use the same execution engine that executes tenants' specific configurations by communicating with a Web server. A tenant is a customer who pays to use the application. It could be an enterprise, a company or any kind of organization wishing to rent the application.

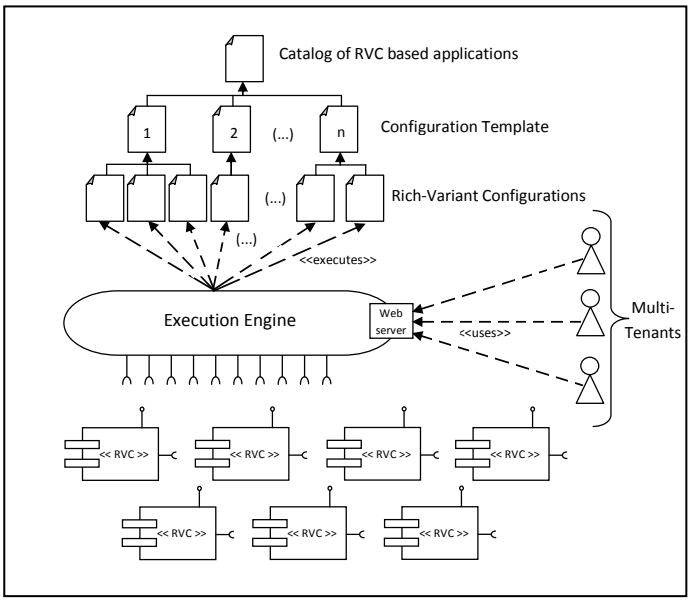

Figure 1. The architectural vision

Each tenant has several users who are actually their employees. These end-users will be categorized according to their business and needs to form different roles or viewpoints. So, applications which are based on RVC components behave differently and this is according to the enabled role or viewpoint thereof, as it was mentioned earlier.

The *catalog* is a formal description of all the applications offered by a provider. It describes the functional variability of each application and specifies the variability points of an application to show how it could be customized.

For each application, the *configuration template* describes the different RVC components that must be linked to create the specific application. The configuration template contains instantiations of the catalog related to the application. Thus, the variability points of each RVC component that require specific tenants information are not configured yet at this level.

Based on a particular configuration template, the *Rich-Variant configuration* describes a specific application tailored for a specific tenant with a dynamic behavior changing during the execution according to the end-users' enabled role or viewpoint. In addition, the parameters or variability points provided by each RVC component are defined at the Rich-Variant configuration level.

From the catalog, the Multi-tenants choose the features and functionalities they need and specify the necessary parameters to obtain their specific Rich-Variant configuration. This fact enables the functional variability. Besides, the use of RVC components allows more flexibility according to end-users business.

In a further work, we plan to define a number of deployment models. Namely a Public deployment model to enable sharing deployment with all other tenants. A Private deployment model to not enable sharing deployment with any other tenant. And a Hybrid deployment model to enable specifying with who share and who do not share the component's instance. For each RVC component, the tenant chooses one deployment model. Thus, we intend to allow the variability of deployment by permitting customers to choose with whom they want to share an instance of a particular RVC component based on the aforementioned deployment models.

Our work has been implemented on a case study to demonstrate the value and feasibility of the approach. The case study consists of a SaaS application for managing private schools, accessible from a web browser. Schools which are tenants of the application benefit, undoubtedly, from deployment variability and functional variability in a flexible, reusable and dynamic environment according to the different needs of the end-users (e.g., administrator, professor, student, etc.).

V. OUTSTANDING AND FUTURE WORKS

In our research work, we seek to integrate both functional and deployment variability. Also, we look to improve reusability by the use of RVC components. In addition, our approach enables flexibility according to the tenants' requirements and the viewpoint or role activated, too. In our approach, we aim to exploit economies of scale by the use of Multi-tenancy concept as the most of approaches cited as related work do. But, we also rely on the use of Multiview

 ISBN: 978-1-61208-367-4

notion predicating on the RVC components to exploit more and more economies of scale.

As future works, we will define an new artifact based on Rich-Variant Component enabling customers to choose to share or not to share with other customers. The next step will be devoted to the implementation of our approach by applying it to the case study consisting of SaaS application for managing private school so as to show its interest and improve it by tests evaluation.

VI. CONCLUSION

The variability management, the RVC component notion and the Multi-tenancy rationalization are key enablers for the accomplishment of flexibility, reusability and exploiting economies of scale in customizable SaaS applications. For this objective, we have initiated in this paper our approach which is primarily based on integrating two types of variability to create a more flexible and reusable SaaS environment while exploiting economies of scale. For this purpose, we introduced the background knowledge of our work: variability management mechanisms, Cloud Computing and Multi-tenancy. Then, we showed the need of managing variability for Cloud environments. Finally, we presented the Multiview component concept to introduce our contribution. Our present work is devoted to the implementation of our approach by applying it to a case study showing its interest.

REFERENCES

[1] R. Mietzner, "A method and implementation to Define and Provision Variable Composite Applications, and its Usage in Cloud Computing," Dissertation, University of Stuttgart, August 2010.

[2] S. Walraven, D. V. Landuyt, E. Truyen, K. Handekyn, and W. Joosen, "Efficient customization of multi-tenant Software-as-a-Service applications with service lines," Journal of Systems and Software vol. 91, Jan. 2014, pp. 48-62.

[3] S. T. Ruehl, H. Wache, and S. A. W. Verclas, "Capturing Customers' Requirements towards Mixed-Tenancy Deployments of SaaS-Applications," IEEE CLOUD, 2013, pp. 462-469.

[4] M. Aiello, P. Bulanov, and H. Groefsema, "Requirements and tools for variability management," Proc. the 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops (COMPSACW '10), Washington, DC, USA, 2010, pp. 245-250, doi:10.1109/COMPSACW.2010.50.

[5] M. A. Iqbal, A. M. Zaidi, and S. Murtaza, "A new requirement prioritization model for market driven products using analytical hierarchical process," Proc. DSDE'10, IEEE, Feb. 2010, pp. 142-149.

[6] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," Technical report, CMU/SEI TR-21, USA, Nov. 1990.

[7] K. C. Kang, S. Kim, J. Lee, K. Kim, G. J. Kim, and E. Shin, "FORM: A feature-oriented reuse method with domain-specific reference architectures," Annals of Software Engineering, vol. 5, 1998, pp. 143–168.

[8] M. Erwig and E. Walkingshaw, "Variation programming with the choice calculus," Generative and Transformational Techniques in Software Engineering, Springer-Verlag Berlin Heidelberg, 2012, pp. 55-100, doi:10.1007/978-3-642-35992-7_2.

[9] I. Trummer, "Cost-Optimal Provisioning of Cloud Applications," Diploma thesis, University of Stuttgart, Faculty of computer sciene, Feb. 2010.

[10] NIST. Definitoon of Cloud Computing - National Institute of Standards and Technology, Gaithersburg, MD, 2009.

[11] C. Fehling and R. Mietzner, "Composite as a Service: Cloud Application Structures, Provisioning, and Management," it - Information Technology Special Issue: Cloud Computing, April 2011, pp. 188-194.

[12] M. Zaremba, T. Vitvar, S. Bhiri, W. Derguech, and F. Gao, "Service Offer Descriptions and Expressive Search Requests - Key Enablers of Late Service Binding," Proc. 13th International Conference on E-Commerce and Web Technologies (EC-Web), Vienna, Austria, Sept. 2012, pp. 50-62, doi: 10.1007/978-3-642-32273-0_5.

[13] M. Zaremba, S. Bhiri, T. Vitvar, and M. Hauswirth, "Matchmaking of IaaS cloud computing offers leveraging linked data," Proc. *28th Annual ACM Symposium on Applied Computing* (SAC) , New York, NY, USA, 2013, pp. 383-388, doi:10.1145/2480362.2480440.

[14] B. El Asri, "A model of multiview components for VUML," National dissertation, Engineering School of Information Technology and System Analysis (ENSIAS), Rabat, Oct. 2005.

[15] M. Nassar, " VUML: a viewpoint oriented UML extension," Proc. 18th IEEE International Conference on Automated Software Engineering, Oct. 2003, pp. 373-376, doi: 10.1109/ASE.2003.1240341.

 ISBN: 978-1-61208-367-4