Towards a High Configurable SaaS
To Deploy and Bind Auser-aware Tenancy of the SaaS

Houda Kriouile, Zineb Mcharfi and Bouchra El Asri
IMS Team, SIME Laboratory, ENSIAS, Mohammed V University, Rabat, Morocco

Keywords: SaaS Application, Rich-variant Component, Algorithm, Graph Theory.

Abstract: User-aware tenancy approach integrates the flexibility of the Rich-Variant Component with the high configurability of multi-tenant applications. Multi-tenancy is the notion of sharing instances among a large group of customers, called tenants. Multi-tenancy is a key enabler to exploit economies of scale for Software as a Service (SaaS) approaches. However, the ability of a SaaS application to be adapted to individual tenant's needs seem to be a major requirement. Thus, our approach proposes a more flexible and reusable SaaS system for Multi-tenant SaaS application using Rich-Variant Components. The approach consists in a user-aware tenancy for SaaS environments. In this paper, an algorithm is established to derive the necessary instances of Rich-Variant Components building the application and access to them in a scalable and performing manner. The algorithm is based on fundamental concepts from the graph theory.

1 INTRODUCTION

Software as a Service (SaaS) is a form of Cloud computing that 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. The user-aware concept consists in considering the end-user connected while deciding of systems behavior. As a key enabler to exploit the economies of scale, SaaS promotes the multi-tenancy, the notion of sharing resources among a large group of customer organizations, called tenants. An advantage of multi-tenancy is that the infrastructure may be used most efficiently as it is feasible to host as many tenants as possible on the same instance. However, multi-tenants application only satisfies the requirements that are common to all tenants.

To tackle this problem, a plethora of research work has been performed to facilitate SaaS applications customization according to the tenant-specific requirements by exploiting the benefits of multi-tenancy, variability management and improving tenants' isolation on a single instance (Mietzner, 2010; Walraven et al., 2014; Zaremba et al., 2012). In the same direction, our approach aims to create a flexible and reusable environment enabling greater flexibility and suppleness for customers while leveraging the economies of scale. The approach is a solution integrating a functional variability in application components level using Rich-Variant Component (RVC), with the high configurability benefit of multi-tenancy. The RVCs are multiview components that allow applications, dynamically, to change the behaviour according to the enabled user's role or viewpoint.

This paper presents our contribution, the user-aware tenancy, and addresses the algorithmic problem of deriving an optimal distribution of RVC instances. The remainder of this paper is structured as follows. Section 2 introduces the approach by presenting the multi-functional and the multi-tenancy notions and treats some challenges of the user-aware tenancy, essentially the high configurability and the performance. Section 3 provides some fundamental concepts from the graph theory, which are relevant to the work presented in this paper as well as it presents the main contribution of the paper consisting in the algorithm which derives the necessary instances of a RVC. Section 4 presents several approaches studied as related work and positions our approach. Finally, Section 5 is a conclusion of the paper.

Kriouile H., Mcharfi Z. and El Asri B..
Towards a High Configurable SaaS - To Deploy and Bind Auser-aware Tenancy of the SaaS.
DOI: 10.5220/0005468206740679
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 674-679
ISBN: 978-989-758-097-0
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

2 USER-AWARE SaaS: TOWARDS HIGHT CONFIGURABILITY OF MULTI-TENANCY FOR SaaS

2.1 On the Multi-functionality

The main activities and goals of a system are described by functional areas. System decomposition into a set of functional areas already existed in the field of database resulting the concept of view. Multifunctional systems have been introduced to overcome problems of inconsistency and overlap between different system perspectives. The multi-functionality notion was introduced under closely related terms such as role, subject, aspect, and view (Kriouile, El Asri and El Haloui, 2014). Our work is rather interested in the notion of view as a mechanism of functional separation. It uses the view concept to take into account the variability of service customers' needs. Moreover, our contribution takes into account the end-user based on the Multiview component concept. Functional concern separation is an important concept for our work, but we also have to be able to dynamically and efficiently configure systems, too.

2.2 On the Multi-tenancy

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 that multiple customers use the same instance, 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. Contrary to the single-tenancy where customization is often done by creating branches in the development tree, in the multi-tenancy configuration options must be integrated in the product design as in software product line engineering. However, 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 (Bezemer and Zaidman, 2010).

2.3 On the Challenges of User-aware Tenancy: High Configurability and Performance

Between the multi-functionality and the multi-tenancy, our contribution aims to benefit from the multi-functional notion of multiview as well as the high configurability characteristic of multi-tenancy. The user-aware tenancy consists in a multi-tenancy SaaS approach using Rich-Variant Components (RVC). The RVCs are multiview components that allow applications to dynamically change the behavior according to the enabled user's role or viewpoint (Kriouile et al., 2014). The goals of our work is to ensure the relevance of information for end-users as well as the control of access rights. When thinking how user-aware tenancy affects an application, we came up with an architectural overview as showed in Figure 1. It comes to a layers architecture. After an authentication, tenants may express their requests. The configuration module does the features matchmaking based on tenants requirements as well as access permissions. Then, the instantiation module instantiates a high configurable instance, which changes the behavior dynamically. It is decompose into two parts: a shared part which is common and a dynamic part consisting of features which could be bind or not according to the point of view enabled.

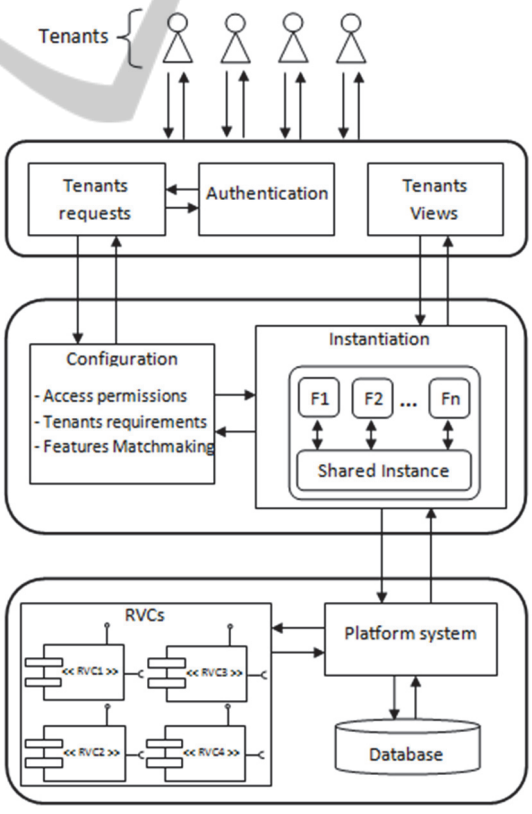

Figure 1: Architectural overview of user-aware tenancy.

Unfortunately, user-aware tenancy also has its

challenges. Some challenges come from the multi-tenancy characteristics such as the high configurability. Other challenges are general as performance, security and maintenance. In this paper, we focus on two essential challenges for the user-aware tenancy which are the performance and the high configurability.

In multi-tenancy, tenants share the same application instance, while it must appear to them as if they are using a specific instance dedicated to. Because of this, a key requirement of user-aware tenancy approach is the possibility to configure and customize the application to a tenant's specific need. Moreover, because of the high degree of configurability of user-aware tenant software systems, it may be necessary to run multiple versions of an application next to each other.

While multiple tenants share the same resources and hardware utilization is higher on average, we must make sure that all tenants can consume these resources as required. If one tenant obstructs resources, the performance of all other tenants may be compromised.

For an example, we consider a SaaS application for a private school management. Some features of this application have been presented in later work (Kriouile, El Asri & El Haloui, 2014).

Each feature is realized over a number of components' views. Sharing a feature means sharing RVC's views used for the realization of the feature. So, for remainder of the paper, we will work on the sharing of RVCs' views and we will work on a RVC at a time. For example, we consider the component "Schedules". This component has four views as it is shown by Figure 2. Also, we consider six private schools tenants of the application T1,T2,T3,T4,T5 and T6.

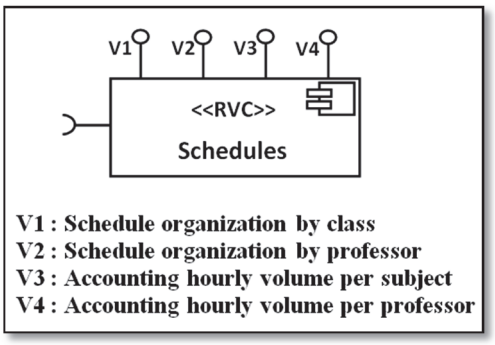

Figure 2: The RVC "Schedules".

3 TO DEPLOY AND BIND A USER-AWARE TENANCY OF THE SaaS

3.1 Fundamentals from Graph Theory

To formulate our problem we had to think about a suitable deployment information representation. Since a graph in mathematics and computer science is an abstract representation of a set of objects where some pairs of these objects may be connected, we have chosen to use graphs to analyze our problem as it has serve to many concrete problems in the real-world. The objects in graphs are represented by abstractions called vertices, and connections between vertices are called edges. A graph is defined as a pair G = (V,E), with a finite set of vertices V and a set of edges E □ V × V (Diestel, 2012).

Our work will be based on Undirected Edge Labelled graph. A Edge labelled graph is a graph where the edges are associated with labels. Besides, an undirected graph is a graph with undirected edges, edges which have no direction. Also, we will use, in this work, terms from graph theory such as the inverse graph and the clique. An inverse graph $\overline{G}$ of graph G contains all vertices of G but those vertices adjacent -that are connected by an edge- in G will not be adjacent in $\overline{G}$ and those vertices not adjacent in G will be adjacent in $\overline{G}$. Further, a clique is a complete sub graph of a given graph, while a complete graph is a graph in which every vertex is adjacent to every other vertex. Every clique in a given graph G is an independent set in the inverse graph $\overline{G}$ and every independent set in a given graph G is a clique in the inverse graph $\overline{G}$, such that an independent set is a set of vertices belonging to the same graph where no two vertices are adjacent (Diestel, 2012).

In graph theory, there is a plethora of problems that are well-known in literature, two of them are essentially related to the work of this paper:

- Vertex Coloring Problem: Is the problem of segmenting a graph into a number of independent sets of vertices.
- Clique Cover Problem: Is the problem of determining a minimum number of sets of vertices of a graph, so that all sets are disjoint cliques.

These two problems are the same since a clique cover of a graph is the same as finding a minimal colouring of the inverse graph (Karp, 1972). Indeed, vertices that belong to a clique are an independent set in the inverse graph. As a result, solving the

problem of splitting a graph into a disjoint set of cliques may be solved by determining the chromatic number of the inverse graph. The first step is to inverse the graph. The second one is the Vertex colouring. And the last one that is optional is to inverse the graph coloured to get the initial graph.

3.2 Some Results from the Mixed-tenancy Approach

As in the Mixed-Tenancy approach (Ruehl, 2014), we will use the graph theory, in particular, the clique cover problems to made up our algorithm to derive the necessary instances and to access to them in a scalable and efficient way. However, we are working on Undirected Edge Labelled graph which made the definition of the clique cover and the colouring problems different for our work.

In the Mixed-Tenancy approach (Ruehl, 2014), the Deployment Information is represented by an undirected graph. For this graph the vertices represent tenants and the edges represent whether two tenants may share infrastructure or not. In case where two tenants may share resources, they would be connected by an edge. This is only the case if neither one of them has Deployment Constraints that express that they shall not share resources. Such a graph is needed for each application component.

Ruehl (2014) analyses the Elementary mixed-tenancy Deployment Problem (a simplified version of the mixed-tenancy Deployment Problem where there is only one Application Component), and has defined a solution of the problem as a Set of Clique Covers. Actually, from a theoretical point of view, every instance in a given Deployment corresponds to a clique in the Deployment Information graph. Thus, a solution is, in fact, a collection of clique covers, one per Deployment Level. What is more, Ruehl (2014) argued that for only one Deployment level, which is our case, a solution for this problem is optimal if and only if the clique cover is minimal. Thus, the problem of finding a Valid and Optimal Solution for the given Elementary mixed-tenancy Deployment Problem is equivalent to finding a minimal clique cover.

3.3 Our Approach: A User-aware Tenancy

The information about relations of features sharing between tenants are translated technically in relations of RVC's views sharing. In our work, we represent this information by an Undirected Edge Labelled Graph. While the vertices represent Tenants, the edges represent whether two Tenants may share views or not. Also, the labels on the edges specify the views concerned with the sharing association represented by the edge. If an edge doesn't have a label, it means that the sharing association concerns the RVC with all its views. An example of these deployment information represented by a graph is graphically shown in Figure 3.

Inspired from the Mixed-Tenancy approach, we deduce that an optimal solution for our problem may be found as a minimal clique cover for each feature of the RVC and this is based on our deployment information graph. However, we have to define our own inversing graph along with our own coloring function, too.

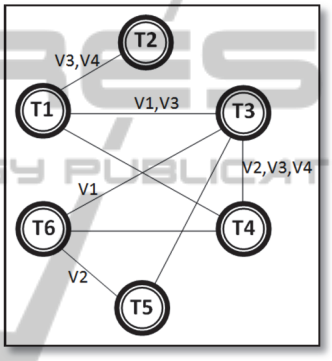

Figure 3: Example of deployment information graph.

The steps of our algorithm deriving the necessary instances are the steps to find a minimal clique cover for our case and they are as follows:

3.3.1 Step 1: Inversing the Undirected Edge Labelled Graph

The first step consists in Inversing the Undirected Edge Labelled Graph, and this by:

- Keeping the same vertices.
- Making each two non-adjacent vertices become adjacent by an unlabelled edge.
- Making each two vertices that are adjacent without label become non adjacent.
- Making each two adjacent vertices with a label become adjacent with a label containing the complement of views in initial label.

For example, for a RVC which have 5 views V1, V2, V3, V4, and V5; if the initial label is "V2,V3,V5" so the label on the inverse graph will be "V1,V4".

3.3.2 Step 2: Divide the Vertices by the Number of the RVC Views.

The second step is to divide the vertices by the number of the RVC views, if the number of views is n views, there will be n parts on each vertex as represented in Figure 4. Each part from the vertex refers to a view of the RVC.

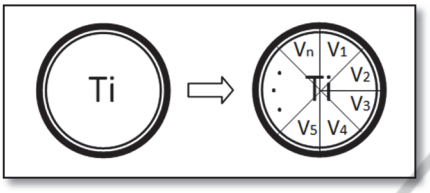

Figure 4: Illustration of the second step.

3.3.3 Step 3: Colouring the Inverse Graph

The third step consists in colouring the inverse graph. Our colouring function gives for each part of each vertex a colour such that two adjacent vertices according to a view have a different colour for the part referring to that view. This is formalized in the Algorithm 1 below:

Algorithm 1: The Coloring Algorithm

Input : T_1, ...,T_m, and V_1, ..., V_n
Output : $\mathcal{C}$ ={C_1, ..., C_d}

1:Give a colour C_1 to T_1 (for all features) and put d=1
2:For i from i=2 to i=m do
3: For j from j=1 to j=n do
4: For k from k=1 to k=d
5: if T_i isn't adjacent to any T from C_k according V_j
6: then give the colour C_k to $T_i.V_j$ and put j=j+1
7: else if k=d
8: then put d=d+1
9: and give the new colour C_d to $T_i.V_j$
10: and put j=j+1
11: else put k= k+1
12: end if
13: end if
14: end For
15: end For
16: end For
17: return $\mathcal{C}$ ={C_1, ..., C_d}

The colouring algorithm returns a set of colours used. Each colour is a set of parts of vertices coloured by this colour.

Lemma 1: When instantiating a RVC for a view, we can use the same instance for the other views.

Taking the Lemma 1 into account, we obviously deduce that the number of instances needed to realize the deployment is the number of colour used, it means that it is the cardinality of the set C. Moreover, we can also deduce the optimal distribution of these instances on the different tenant, and this from the same output of Algorithm1. Indeed, each colour Ck refers to a specific instance of the RVC and the elements of that colour Ck refer to the tenants that will use that instance and for which view they will use it.

To conclude, our algorithm, which aims to derive the necessary instances of a RVC, can be simplified and formalized in Algorithm 2. This algorithm takes as Input the Undirected Edge Labelled Graph representing the deployment information concerning the RVC, and returns as output the set of colour used.

Algorithm 2: Compute Deployment Algorithm

Input : G an Undirected Edge Labelled Graph,
and n the number of views
Output : $\mathcal{C}$ ={C_1, ..., C_d}

1: Inverse the graph G to G'
2: Divide the vertices of G' by n part
3: Colour the graph G' using Algorithm 1
4: return $\mathcal{C}$ ={C_1, ..., C_d}

The next section presents several approaches studied as related work and positions our approach in comparison with those approaches.

4 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 (2011) propose the Composite-as-a-Service (CaaS) model. They show how applications which are 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 (Mietzner, 2010), 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 et al. (2012) present models of Expressive Search Requests and Service Offer Descriptions allowing matchmaking of highly configurable services that are dynamic and depend on request. This approach can be applied to several types of services. This approach does not propose a solution to exploit economies of scale and only deals with one variability type, the deployment variability.

In (Walraven et al., 2014), 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 application without compromising scalability. In contrast to our approach, this method - as well as the other approaches mentioned - does not address to the accessibility by roles, which is allowed in our work by the use of Multiview concept. The Multiview notion allows applications to dynamically change the behaviour according to the enabled user's role or viewpoint.

Ruehl (2014) addresses the deployment variability based on the SaaS tenants requirements about sharing infrastructure, application codes or data with other tenants. Ruehl (2014) proposes 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. The author focuses on the deployment variability and neglects the functional variability management.

5 CONCLUSIONS

Flexibility and reusability are challenging issues for multi-tenancy SaaS applications. In this regard, our approach consists in integrating two types of concepts, the multi-functionality and the multi-tenancy, to create a more flexible and reusable SaaS environment while exploiting economies of scale. It comes to the user-aware tenancy approach. Moreover, this paper addresses the algorithmic part of the work, which aims to derive an optimal distribution of instances for a RVC. For this purpose, we first introduced in this paper the user-aware tenancy approach. Then, we presented some challenges for this approach. Also, we introduced some background knowledge of our work from the graph theory concepts. Also, we presented our algorithm deriving an optimal distribution of RVC instances over tenants. And finally, we compared our approach to similar approaches studied as related work to make clear the benefits brought by our approach. Our following step will be dedicated to the implementation of our approach by applying it to a case study showing its interest and improving it by tests, as a major instrument of measurement.

REFERENCES

Bezemer, C. P., Zaidman, A., 2010. 'Multi-tenant SaaS applications: maintenance dream or nightmare? ', in *IWPSE-EVOL'10*, Antwerp, Belgium, 20-21 September, pp. 88-92.

Diestel, R., 2012. *Graph Theory*, Graduate texts in mathematics 173, Springer, 4th Edition. ISBN 978-3-642-14278-9, pp. I-XVIII, 1-436.

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

Karp, R. M., 1972. *Reducibility among combinatorial problems*. Tech. rep. Springer, pp. 85-103.

Kriouile, H., El Asri, B., El Haloui, M., Benali, A., 2014. 'Towards Implementation and Design of Multi-tenant SaaS Based on Variabiliy Management Mechanisms', in *ICSEA'14, the Ninth International Conference on Software Engineering Advances*, Nice, France, 12-16 October, pp. 468-471.

Kriouile, H., El Asri, B., EL Haloui, M., 2014. 'Towards Flexible and Reusable SaaS for Multi-tenancy to design, implement and bind multi-functional variability for Rich-Variant services', in *WCCS'14, the Second World Conference on Complex Systems*, Agadir, Morocco, 10-12 November.

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

Ruehl, S. T., 2014. 'Mixed-Tenancy Systems A hybrid Approach between Single and Multi-Tenancy', Doctoral Thesis, Department of Informatics, Clausthal University of Technology, 16 June.

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

Zaremba, M., Vitvar, T., Bhiri, S., Derguech, W., Gao, F., 2012. 'Service Offer Descriptions and Expressive Search Requests - Key Enablers of Late Service Binding', in *Proceeding of the 13th International Conference on E-Commerce and Web Technologies (EC-Web)*, Vienna, Austria, Sept. 2012, pp. 50-62.