Multi-tenancy in scientific literature from Software as a Service perspective

Eveliina Pakarinen
Department of Computer Science
University of Helsinki
Email: eveliina.pakarinen@helsinki.fi

***Abstract*—Multi-tenancy is a high level architectural pattern which can be used in cloud computing environment when offering applications using Software as a Service business model. In multi-tenancy pattern service provider hosts a single instance of an application on his or her infrastructure. This application is accessed by multiple customers, so-called tenants. Tenants share the resources of the infrastructure. Multi-tenancy is still a relatively new topic in scientific literature. The research domain of multi-tenancy is not yet mature which can be seen from the lack of industrial experience reports on multi-tenancy. The use of multi-tenancy pattern brings multiple benefits both to the service provider and to the customers. In addition to the benefits there are also some challenges that affect the use of multi-tenancy. In this paper the different variants of multi-tenancy in scientific literature are investigated from the perspective of Software as a Service (SaaS) business model. The connection between SaaS and multi-tenancy is also investigated. Finally some challenges caused by the characteristics of multi-tenancy are presented.**

Keywords: multi-tenancy, Software as a Service, definition of multi-tenancy

I. Introduction

Software as a Service (SaaS) is a novel business model where software and the underlying ICT infrastructure are offered as an on-demand service for customers through Internet [1]. In Software as a Service business model the service provider maintains the application and offers the software as a service to the customers [1]. By using software offered by a third party service provider companies can use various ICT services without maintaining or purchasing their own ICT infrastructure [1].

Multi-tenancy is a high level architectural pattern which can be used when offering applications as Software as a Service in cloud computing environment [2]. In multi-tenancy pattern the service provider hosts a single instance of the software product on his or her infrastructure and multiple customers, so-called tenants, access the same instance of the software [1]. The definition for a tenant is: *"A tenant is the organizational entity which rents a multi-tenant SaaS solution"* [1]. A tenant groups typically multiple users of the same organization and these users are the stakeholders in the organization [1].

There are multiple benefits for the service provider when using multi-tenant architecture pattern when implementing SaaS applications. The first benefit is that the application deployment becomes easier because only one application instance has to be deployed [1]. In multi-tenancy multiple customers access the same software instance and they do not need own dedicated instance of the software. This means that the customers share the same hardware resources when using multi-tenant Software as a Service application [3]. That increases and improves the utilization rate of the hardware which is the second benefit of the multi-tenancy [1].

These two benefits reduce service provider's overall delivery costs of the application [1]. The reduced delivery costs of a multi-tenant application can help improve service provider's profit margin [3]. The reduced delivery costs make it possible for the service provider to offer the service to the customers at lower service subscription costs [3]. The reduced overall costs of a multi-tenant application make multi-tenant applications interesting for customers in the small and medium enterprise (SME) segment of the market [1].

When describing multi-tenancy in scientific literature the three often mentioned characteristics of multi-tenancy are sharing of hardware resources, possibility to configure the shared application instance in runtime and shared application and database instance. Those characteristics are also part of the definition of multi-tenancy which is used in this seminar paper.

In addition to the benefits of multi-tenancy there are also some challenges that come with the use of multi-tenancy. Challenges can arise in the application development, deployment and management phases [3]. The three characteristics of multi-tenancy can cause some of the challenges of multi-tenancy. Areas where challenges can arise are for example in application performance and maintenance and in security [1].

Multi-tenancy is still a relatively new topic in scientific literature in the domain of software and hardware systems [2]. The term *'multi-tenancy'* was explicitly mentioned for the first time in a scientific literature in year 2006 [2] by Chong and Carraro in their paper *"Architecture strategies for catching the long tail"* [4]. Since then many definitions for multi-tenancy have been proposed [2]. Also many solutions related to implementing multi-tenancy have been proposed in the multi-tenancy research over the years but there has been very few industrial reports about experiences on multi-tenancy [2].

The low amount of industrial reports on multi-tenancy indicates that the research domain of multi-tenancy is not yet mature and that the proposed solutions have not yet been implemented or evaluated [2]. The high amount of proposals and the low amount of industrial reports can also indicate that

there is a lack of cooperation between industry and academia in multi-tenancy research domain [2].

In this paper the different variants of multi-tenancy in scientific literature are investigated from the perspective of Software as a Service business model. The connection between SaaS business model and multi-tenancy is also investigated. Finally some possible challenges caused by the characteristics of multi-tenancy are presented.

This paper is organized as follows. In Section 2 the research questions and the research methods for data collection for this paper are presented. In section 3 the characteristics of multi-tenancy and some challenges caused by those characteristics are presented. In section 4 the different variants of multi-tenancy are presented from the viewpoint of a SaaS business model. In section 5 the answers to the research questions are discussed. The paper is concluded section 6.

II. Research questions and methods for data collection

This seminar paper tries to provide a description of the different variants of multi-tenancy in scientific literature from the perspective of Software as a Service business model. The main research questions for this seminar paper are:

1) How multi-tenancy is defined in scientific literature?
2) Is there a connection between multi-tenancy and SaaS business model?
3) What challenges the characteristics of multi-tenancy can cause?

The perspective of SaaS was chosen for this seminar paper because SaaS and architecture play a big role in the academic research of multi-tenancy [2]. As a comparison point to the academic research the perspective of multi-tenancy used by practitioners is mostly on infrastructural or platform level application of multi-tenancy [2].

The work in this seminar paper is based on academic papers and technical articles. The topic for this seminar paper was found during a search for papers for a different topic. The initial article which inspired the topic of this seminar paper was *"Multi-Tenant Web Application Framework Architecture Pattern"* by Bien and Thu [5]. After learning about multi-tenancy for the first time from this article the search for papers about multi-tenancy continued in the ACM Digital Library and in the IEEE Xplore Digital Library.

The search was done with quick search using combinations and different variations of words *'multi-tenancy'* or *'multi-tenant'* with or without '-'-character and *'SaaS'* or *'Software as a Service'* and *'architecture'*. After getting the results from the search the results were refined by setting the range of publication year from 2014 to 2016. After finding a couple of papers that discussed multi-tenancy the references in those papers were used to find more papers related to the topic. Web of Science was also used to find papers that had cited the papers found during the search in digital libraries.

The initial goal for data collection was to find as many papers as possible. After that the first thing to do was to filter out papers that were not suitable as sources for this seminar paper. Characteristics for not suitable papers were for example that the paper was published in a too small conference or in an unknown journal or that it did not discuss multi-tenancy in application level in SaaS applications or that multi-tenancy was not discussed from architectural perspective.

In order to find out if the paper discussed multi-tenancy in application level in SaaS applications or from architectural perspective the titles and abstracts of the papers were first briefly read through. After that the decision was made whether to filter out the paper or to include it for further analysis. If the decision to filter out a specific paper could not be made based on the title and abstract parts of the paper also the introduction and conclusion parts were analysed.

After that the remaining papers were organized to four groups based on the abstract, introduction and conclusion parts of the papers. The groups in which the papers were organized were:

1) not so relevant papers for this seminar work
2) probably important papers for this seminar work
3) papers that propose some multi-tenant pattern or architecture
4) supporting papers with relevant background information.

The papers that belonged to group 1 were left out and not read after the screening of the papers. The papers that belonged to groups 2, 3 and 4 were studied further. Some but not all of the papers that belonged to groups 2, 3 and 4 were read through as a whole in order to find out if the paper had potential to be used as a reference in this seminar paper.

The references that are used in this seminar paper are based on the papers that belonged to groups 2, 3 and 4 and were read through as a whole during the paper screening. The papers *"Architecture strategies for catching the long tail"* by Chong and Carraro [4] and *"Multi-tenant data architecture"* by Chong, Carraro and Wolter [6] were found during the writing process of this seminar paper by using the references in other papers.

III. Characteristics of multi-tenancy

There are three different characteristics of multi-tenancy that are often mentioned when describing multi-tenancy in scientific literature. Those characteristics that are used to describe multi-tenancy are sharing of hardware resources, possibility to configure the shared application instance in runtime and shared application and database instances [1].

When applications are served to customers as single-tenant applications server utilization can be low especially when applications are used by customers in the small and medium enterprise (SME) segment of the market [1]. In that situation server utilization can be improved by placing several tenants on the same server [1]. Different multi-tenancy variants from multiple instances multi-tenancy to native multi-tenancy can be used to place several tenants on the same server. These variants of multi-tenancy are presented in the Figure 1 and described in detail in section 4.

Through higher utilization of the servers the total amount of hardware required to serve the tenants is lower than when

using single-tenancy where each tenant has its own dedicated server. The result of requiring lower amount of hardware because of higher server utilization in multi-tenancy is that the overall delivery costs of the application will be lower [1]. The reduced delivery costs make it possible for the service provider to improve the profit margin and to offer the service to the customers at lower service subscription costs [3].

Sharing the same hardware resources can also cause problems in addition to the benefits of sharing the same resources. In a situation where the hardware resources are shared a problem caused by one tenant can affect all the other tenants that use the same shared resources [7]. In such a situation when one tenant blocks the resources the performance of all other tenants can be compromised [1].

In multi-tenancy tenants should be able to make customizations to their own application in runtime without impacting other tenants [3]. In addition to multi-tenant and multiple instances multi-tenancy concepts there is also a third software system concept namely multi-user concept [1]. One difference between multi-tenant and multi-user concepts is that in multi-tenancy each tenant can configure for example the appearance or the workflow of the application heavily [1]. In multi-user applications all customers use the same application and have only limited configuration options [1]. This means that in multi-user concept the application functionality is the same for all customers but the application can be still partly configurable [2].

Introducing configurability in a multi-tenant application can result in more complex application code [7]. In single-tenancy application configuration can be done by creating a branch in the development tree and deploying a separate instance of the application [1]. In multi-tenancy the possibility to configure the application is integrated in the application architecture which increases the code complexity [1]. Increased code complexity can cause problems in the maintenance phase of the application [1].

The third characteristic of multi-tenancy is the ability to share one application and database instance. In native and semi-multi-tenancy variants of multi-tenancy the application instance is shared among different tenants which can be seen from the Figure 1. In native multi-tenancy also the database and schema are shared among tenants. When the data of all of the tenants is on the same server it can cause risks in terms of security [7]. A security breach can cause the exposure of one tenant's data to other tenants [1]. Therefore the protection of each tenant's critical information is an important topic in multi-tenancy [3].

IV. Variants of multi-tenancy in scientific literature

A. The definition of multi-tenancy

Multi-tenancy is a high level architectural pattern which can be used to share computing resources when offering software products using Software as a Service business model [2]. In multi-tenancy a single instance of an application is hosted on service provider's infrastructure [2]. This single instance is accessed by multiple tenants and can be customized according to the requirements of different tenants [2]. Multi-tenancy has evolved from previous information technology paradigms like time-sharing, application service provider (ASP) model and multi-user model [2]. Multi-tenancy was explicitly mentioned in scientific literature for the first time in the domain of software and hardware systems in year 2006 [2] by Chong and Carraro in their paper *"Architecture strategies for catching the long tail"* [4].

In academic literature the definition of multi-tenancy has been varying and there has been differences in the interpretation of multi-tenancy among academics in academia [2]. In order to chart and bridge the varying definitions of multi-tenancy and to provide an overview of the multi-tenancy domain Kabbedijk et al. [2] performed a structural search in academic literature and in public blog posts.

As a result of their structural search Kabbedijk et al. proposed a comprehensive definition for multi-tenancy which is based on 43 different definitions of multi-tenancy presented in academic literature [2]. The definition which Kabbedijk et al. proposed for multi-tenancy is: *"Multi-tenancy is a property of a system where multiple customers, so-called tenants, transparently share the system's resources, such as services, applications, databases, or hardware, with the aim of lowering costs, while still being able to exclusively configure the system to the needs of the tenant"* [2]. In their paper Kabbedijk et al. call for this definition to be used in future research on multi-tenancy [2]. This definition of multi-tenancy also mentions all of the characteristics of multi-tenancy which are mentioned in section 4.

In the Figure 1 different system levels are illustrated in different variants of multi-tenancy and in single-tenancy. Tenants are marked with groups of person figures in the Figure 1. The system levels marked with yellow background color represent application and software levels of the system. The infrastructure levels of the system are marked with purple background color and the database levels are marked with green background color. The levels of the system that are influenced by software level multi-tenancy are marked with solid borders and bold font style. Those levels are application instance, application server, database schema, database and database server [2]. The levels that are not influenced by software level multi-tenancy are middleware, operating system, virtual machine and hardware and those levels are marked with dashed line borders and plain font style in the Figure 1.

The system levels illustrated in the Figure 1 are based on the software stack presented in the paper of Kabbedijk et al. [2]. The different variants of multi-tenancy illustrated in the Figure 1 represent the different perspectives from which multi-tenancy can be seen. The different variants illustrated in the Figure 1 are based on variants of multi-tenancy presented by Guo et al. [3] and Chong et al. [6]. As a comparison to multi-tenancy variants a single-tenancy variant is also presented in the Figure 1.

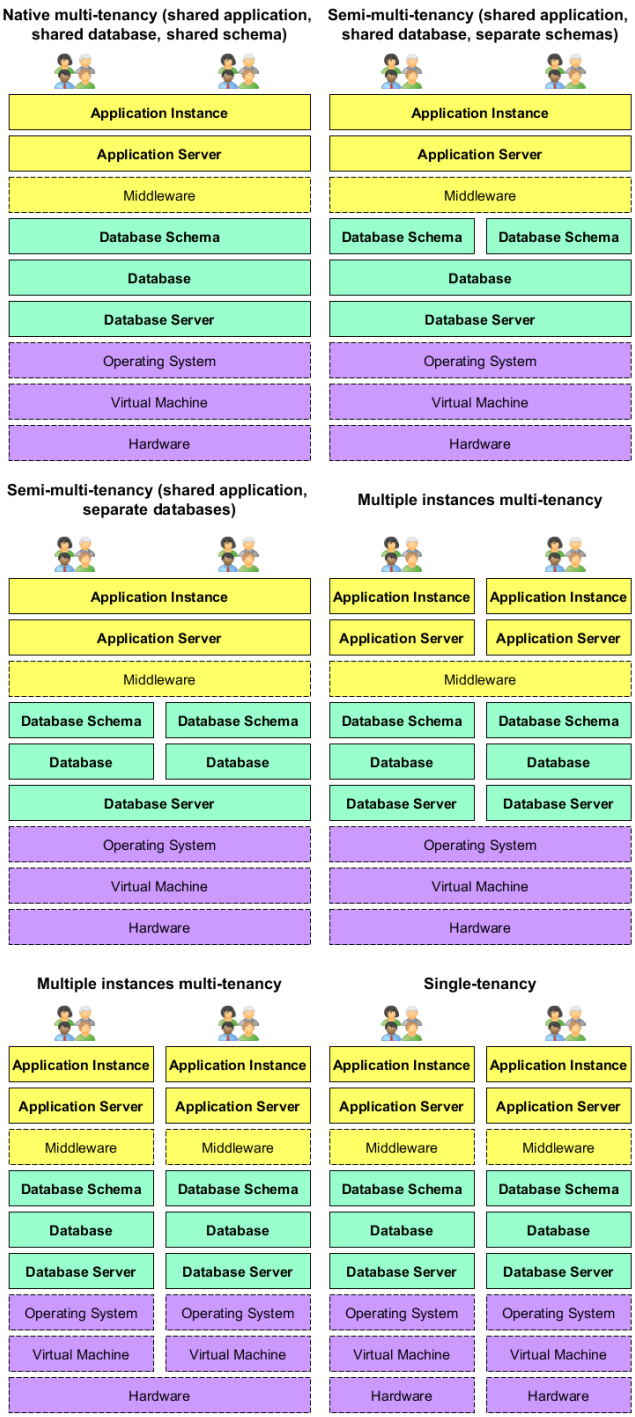

Fig. 1. Illustration of different system levels in different variants of multi-tenancy and in single-tenancy.

B. The description of the multi-tenancy variants

In native or pure multi-tenancy [2], [3] the tenants share the same application instance and database tables which are hosted on a shared infrastructure. Native multi-tenancy variant is used to support a large number of tenants and the number is usually in the hundreds or thousands of tenants [3].

In the semi-multi-tenancy variants of multi-tenancy [6], [1] the tenants share the same application instance but the level of sharing on the database levels of the system vary. The two variants of semi-multi-tenancy are *"shared application, shared database and separate schemas"*-variant and *"shared application and separate databases"*-variant [6].

When a high number of tenants are placed on the same server and one of the semi-multi-tenancy variants of multi-tenancy is used it can cause performance problems on the server [1]. These performance problems are caused by the expensive operation of loading a database or a table in memory when a tenant accesses the application. When comparing database or table loading requirements in semi-multi-tenancy variants to native multi-tenancy native multi-tenancy enables placing more tenants on the same the server because the shared database table must be loaded only once to memory [1].

In multiple instances multi-tenancy [3] tenants no longer share the same application instance. Instead of sharing an application instance each tenant has its own dedicated application instance over a shared middleware server, operating system or hardware [3]. Two variants of multiple instances multi-tenancy are presented in the Figure 1. Multiple instances multi-tenancy variants scale differently than native multi-tenancy when comparing the number of tenants which multiple instances variants or native multi-tenancy can support [3]. Multiple instances multi-tenancy variants can support from several up to dozens of tenants on the same server while native multi-tenancy can support hundreds or thousands of tenants [3].

As a comparison point to different multi-tenancy variants a single-tenancy variant is also presented in the Figure 1. In traditional single-tenant approach each tenant has its own dedicated server and own customized application instance which they use [1]. This means that a single-tenant application may have many separate running instances which all can be different from each other [1]. A traditional single-tenancy scenario from the early 90s was that companies moved their hardware and applications from their premises to data centers where the applications were hosted without any hardware or software sharing [3].

C. Multi-tenancy in Software as a Service business model

In the academic research of multi-tenancy Software as a Service and architecture are important research topics and both of the topics are addressed a lot in multi-tenancy research [2]. The frequent occurrence of those topics indicates that multi-tenancy is positioned as an architectural tactic for online software [2].

Software as a Service is a business model where software is offered as an on-demand service for customers through Inter-

net [1]. Simplistic characterization for Software as a Service is: *"Software deployed as a hosted service and accessed over the Internet"* [4]. From architectural perspective a mature and a well-designed SaaS application has three different attributes that separate it from a poorly designed SaaS application [4]. Those attributes are configurability, multi-tenant efficiency and scalability [4].

With the help of those attributes a Software as a Service maturity model with four distinct levels can be defined [4]. Software as a Service maturity model expresses the maturity of a SaaS application. At each level of the maturity model one of the attributes of a mature SaaS application is added to the model [4]. The Figure 2 presents the maturity model of a Software as a Service application. The attributes of a mature SaaS application can also be seen from the Figure 2.

At the first level of the maturity model each tenant has own customized version of the application [4]. Each customized application instance is run separately on service provider's servers. At this level service providers can reduce costs by running multiple application instances on the same server hardware and by consolidating server administration [4]. Running multiple instances of the same application on the same server hardware corresponds to the multiple instances multi-tenancy variant presented in the Figure 1.

At the second level of the maturity model each tenant has an own separate instance of the application but all instances use the same identical code implementation [4]. At this level configurability is added to the SaaS application architecture so that the separate instances of the application can be configured to meet the needs of the customers [4]. Changes made to the code implementation can be delivered to all tenants. This reduces service requirements of a SaaS application because there is no need to upgrade individual customized instances anymore [4]. At this level the application is still served using multiple instances multi-tenancy variant.

At the third level of the maturity model multi-tenancy is added to the SaaS application architecture [4]. Adding multi-tenancy to the application makes it possible to run a single instance of the application on service provider's infrastructure and serve every customer using this single instance [4]. At this level the tenants are not aware of sharing the application instance with other tenants and the application instance can be customized to meet the needs of the tenants [4]. The resources needed to host a single instance of an application on the level three are lower than the resources needed for multiple instances on levels one and two which lowers the costs of hosting the application [4]. The disadvantage of the single instance multi-tenant approach at this level is that when the amount of tenants increases the only way to scale the application up is to host it on a more powerful server [4].

At the fourth level of the maturity model scalability is added to the SaaS application architecture [4]. At this level the use of a tenant load balancer enables adding and removing application servers and instances according to the current demand [4].

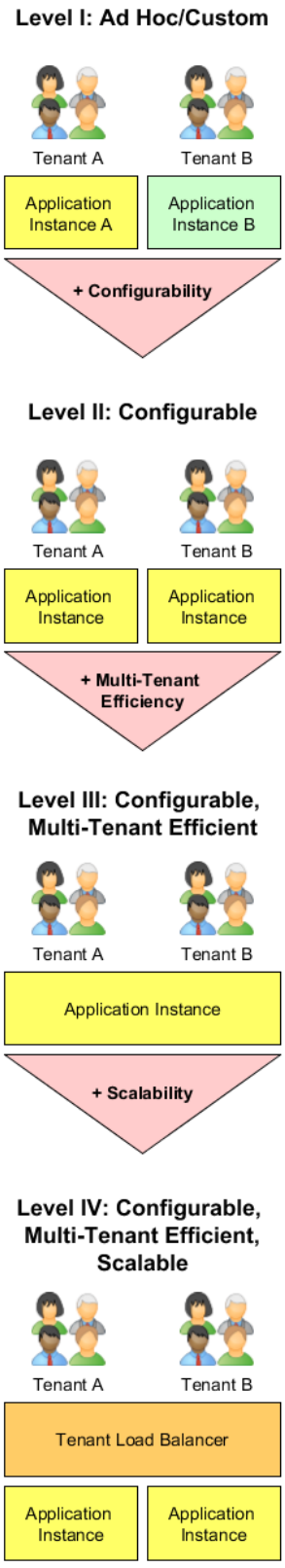

Fig. 2. The levels of the Software as a Service maturity model.

V. Discussion

The definition of multi-tenancy used in this seminar paper is the definition which Kabbedijk et al. present in their paper *"Defining multi-tenancy: A systematic mapping study on the academic and the industrial perspective"* [2]. This definition was chosen for this seminar paper because it is clear and comprehensive and is based on scientific research of several different multi-tenancy definitions. Answers to the research questions one and two can be found in the Section 4 where the definition of multi-tenancy and the connection between multi-tenancy and Software as a Service business model are presented.

In the different variants of multi-tenancy presented in the Section 4 multi-tenancy is applied to different levels of the system. In the multiple instances multi-tenancy variant only the hardware or the operating system levels are shared among different tenants while each tenant has its own dedicated application instance. When using semi-multi-tenancy the amount of sharing on the database level of the system vary from shared database and separate schemas to separate databases. The highest level of sharing is achieved in native multi-tenancy where all system levels are shared among all tenants.

This seminar paper does not provide a concrete example of a multi-tenant Software as a Service application. In order to provide a deeper description of the different variants of multi-tenancy some concrete examples of multi-tenant applications and their differences could have been useful.

Since Software as a Service is a frequent topic in the academic research of multi-tenancy and multi-tenancy is part of the Software as a Service maturity model the connection between multi-tenancy and Software as a Service business model is strong. With the help of multi-tenancy service provider can achieve savings in application maintenance and server infrastructure costs by maintaining only a single code base for the application and hosting only a single instance of an application on his or her server infrastructure.

Some of the challenges caused by the characteristics of multi-tenancy are presented in the Section 3. The characteristics of multi-tenancy can cause challenges in the performance, maintenance and security areas of the application. When developing multi-tenant applications it would be good to take those challenges into consideration in order to mitigate the problems that the challenges can cause.

This seminar paper discusses only some of the challenges of multi-tenancy and those challenges are selected based on the three characteristics of multi-tenancy. One limitation of this seminar paper is that in addition to the challenges discussed in this seminar paper there can be other challenges which are not mentioned in this paper.

Other limitation that affects the validity of this seminar paper is that the author has never, at least not knowingly, used a real multi-tenant application or researched the architecture of a real multi-tenant application. The consequence of this limitation is that the author's interpretations of multi-tenancy and the variants of multi-tenancy are only based on scientific literature without real industrial experience of multi-tenancy.

VI. Conclusion

In this paper the different variants of multi-tenancy in scientific literature were illustrated from the perspective of Software as a Service business model. Multi-tenancy is a high level architectural pattern which can be used to reduce service provider's overall delivery costs of the application in order to improve the profit margin. The reduced delivery costs make it also possible for the service provider to offer the service to the customers at lower service subscription costs. Multi-tenancy is one of the attributes of a mature well-defined Software as a Service application. The topic of Software as a Service is also addressed a lot in the academic research of multi-tenancy. In addition to the benefits of multi-tenancy the characteristics of multi-tenancy can cause challenges in the performance, maintenance and security areas of the application. Those challenges need to be taken into consideration when developing multi-tenant applications in order to mitigate the problems caused by those challenges.

References

[1] C.-P. Bezemer and A. Zaidman, "Multi-tenant saas applications: Maintenance dream or nightmare?" in *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*, ser. IWPSE-EVOL '10. New York, NY, USA: ACM, 2010, pp. 88–92. [Online]. Available: http://doi.acm.org.libproxy.helsinki.fi/10.1145/1862372.1862393

[2] J. Kabbedijk, C.-P. Bezemer, S. Jansen, and A. Zaidman, "Defining multi-tenancy: A systematic mapping study on the academic and the industrial perspective," *Journal of Systems and Software*, vol. 100, pp. 139 – 148, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121214002313

[3] C. J. Guo, W. Sun, Y. Huang, Z. H. Wang, and B. Gao, "A framework for native multi-tenancy application development and management," in *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 2007)*, July 2007, pp. 551–558.

[4] F. Chong and G. Carraro, "Architecture strategies for catching the long tail," 2006, accessed: 23.03.2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa479069.aspx

[5] N. H. Bien and T. D. Thu, "Multi-tenant web application framework architecture pattern," in *2015 2nd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS)*, Sept 2015, pp. 40–48.

[6] F. Chong, G. Carraro, and R. Wolter, "Multi-tenant data architecture," 2006, accessed: 23.03.2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa479086.aspx

[7] C. P. Bezemer, A. Zaidman, B. Platzbeecker, T. Hurkmans, and A. . Hart, "Enabling multi-tenancy: An industrial experience report," in *2010 IEEE International Conference on Software Maintenance*, Sept 2010, pp. 1–8.