Multi-tenancy in Software as a Service Applications

Eveliina Pakarinen
University of Helsinki
Helsinki, Finland

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, who share the resources of the infrastructure. The use of multi-tenancy pattern brings multiple benefits both to the service provider and to the customers. In addition to benefits here are also some complexities that affect the use of multi-tenancy. Although multi-tenancy is a popular paradigm it 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. In this paper Software as a Service business model and multi-tenancy are introduced and the connection between multi-tenancy and SaaS is investigated. The key concepts of multi-tenancy in SaaS applications are also presented. Finally some concerns that need to be taken into account when developing a multi-tenant SaaS application are discussed.**

Keywords: multi-tenancy, Software as a Service, architectural pattern

I. Introduction

Software as a Service (SaaS) is a novel business model where software and the underlying infrastructure are offered as an on-demand service for customers through Internet (dream or nightmare). In SaaS business model the service provider maintains the application and offers the software as a service to the customers (maintenance dream or nightmare 1). By using software offered by a third party companies can use various IT services without maintaining or purchasing their own IT infrastructure (maintenance dream 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 (Defining multi-tenancy: 1). 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 (Maintenance Dream or Nightmare 1). A tenant is the organizational entity which rents a multi-tenant SaaS solution (maintenance dream 2). A tenant groups typically multiple users of the same organization and these users are the stakeholders in the organization.

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 (maintenance dream 1). In multi-tenant model 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 SaaS application (application multi-tenancy for SaaS 1). That increases and improves the utilization rate of the hardware which is the second benefit of the multi-tenant model (maintenance dream 1).

These two benefits reduce the software delivery costs for the service provider which can help improve the profit margin (a framework for native multi-tenancy application Guo 1). The reduced delivery costs enable that software provider can offer the service to the customers at lower service subscription costs (a framework Guo 1). That makes multi-tenant applications interesting for customers in the small and medium enterprise segment (maintenance dream 1).

In addition to the benefits of the multi-tenancy there are also some complexities that come with the multi-tenancy. Challenges can arise in the application development, deployment and management phases (framework for native Guo 1). Challenges that arise are for example application performance, scalability, security, zero-downtime and maintenance (maintenance dream 3).

Although multi-tenancy is a popular paradigm it is still a relatively new topic in scientific literature (defining 1). The term multi-tenancy was explicitly mentioned for the first time in a scientific paper in year 2006. Since then many definitions for multi-tenancy have been proposed (defining 5). Also many solutions related to 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 (definition 4-5). This indicates that the research domain of multi-tenancy is not yet mature and that the solutions have not yet been implemented or evaluated. 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 this domain (definition 4-5).

In this paper Software as a Service business model and multi-tenancy are introduced and the connection between multi-tenancy and SaaS is investigated. The key concepts of multi-tenancy in SaaS applications are also presented. Finally some concerns that need to be taken into account when developing a multi-tenant SaaS application are discussed.

This paper is organized as follows. In Section 2 the research methods for data collection for this paper are introduced and the research questions are presented. In section 3 an introduction to multi-tenancy and SaaS is given. In section 4 the benefits and challenges of multi-tenancy are presented. In section 5 the answers to the research questions are discussed. A conclusion is presented in section 6.

II. Research methods

This seminar work is based on academic papers published in various conferences and journals. The search for these papers was mainly done in the ACM Digital Library and in the IEEE Xplore Digital Library. Initial search criteria for papers were that the papers were published recently between years from 2014 to 2016 and that the papers had something to do with multi-tenancy and architecture or multi-tenancy and SaaS applications or generally mentioned multi-tenancy. After finding a couple of papers the references in them were used to find more papers related to the theme discussed in the paper. 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 work. Characteristics for not suitable papers were for example that the paper was published in a too small conference or 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.

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. 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 the papers that belonged to groups 2, 3 and 4 were studied further. The references that are used in this seminar work are based on the papers that belonged to groups 2, 3 and 4.

This seminar paper tries to provide a clear description of multi-tenancy and multi-tenant SaaS applications. The main research questions for this seminar work are: 1. What is multi-tenancy? 2. What is the connection between SaaS and multi-tenancy? 3. What needs to be taken into consideration in software architecture when developing multi-tenant SaaS application?

III. Introduction to Software as a Service and multi-tenancy

Software as a Service is a business model where software is offered as an on-demand service for customers through Internet (dream or nightmare). Simplistic characterization for Software as a Service is: Software deployed as a hosted service and accessed over the Internet. (Carraro Tail) From architectural perspective a mature and a well-designed SaaS application has three different attributes that separate it from a poorly designed SaaS application (Carraro tail). Those attributes are configurability, multi-tenant-efficiency and scalability (Carraro tail).

With the help of these attributes a SaaS maturity model with four distinct levels can be defined (Carraro tail). SaaS 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 (Carraro Tail). The Figure 1 presents the maturity model of a SaaS application and the attributes which are added to the model on levels one, two and three.

At the first level of the maturity model each tenant has own customized version of the application (Carraro Tail). Each customized application instance is run separately on the hosts servers. At this level hosts can reduce costs by running multiple application instances on same server hardware and by consolidating server administration (Carraro Tail).

At the second level of the maturity model each tenant has own separate instance of the application but all instances use the same identical code implementation (Carraro Tail). 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 (Carraro Tail). 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 (Carraro Tail).

At the third level of the maturity model multi-tenancy is added to the SaaS application architecture (Carraro Tail). Adding multi-tenancy to the application makes it possible to run a single instance of the application on service providers infrastructure and serve every customer using this single instance (Carraro Tail). 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 (Carraro Tail). The resources needed to host a single instance of an application on level three are lower than the resources needed for multiple instances on levels one and two which lowers the costs of hosting the application (Carraro Tail). 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 is to host it on a more powerful server (Carraro Tail).

At the fourth level of the maturity model scalability is added to the SaaS application architecture (Carraro Tail). At this level the use of tenant load balancer enables adding and removing application servers and instances according to the current demand (Carraro Tail). [1]

Multi-tenancy is a high level architectural pattern which can be used to share computing resources when offering software products as Software as a Service (defining multitenancy). In multi-tenancy a single instance of an application is hosted on service providers infrastructure and this single instance is accessed by multiple tenants and can be customized according to the requirements of different tenants (defining multi tenancy). Multi-tenancy has evolved from previous information technology paradigms like time-sharing, application service provider (ASP) model and multi-user model (defining mt). Multi-tenancy was explicitly mentioned in scientific literature

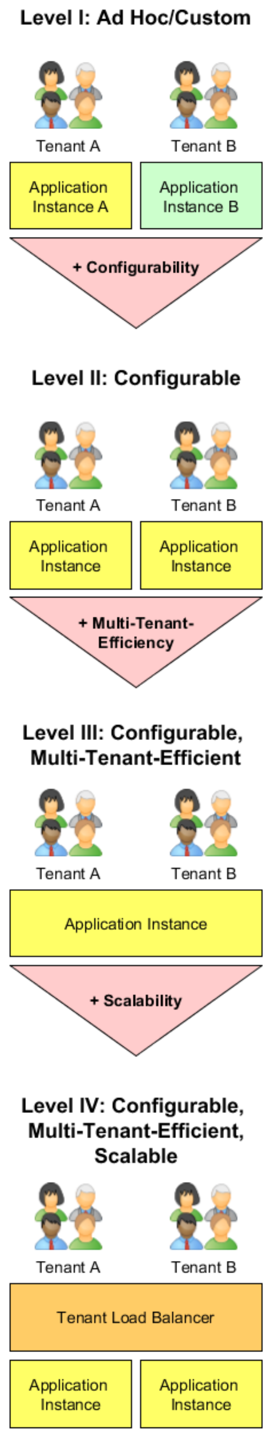

Fig. 1. The levels of SaaS maturity model.

for the first time in the domain of software and hardware systems in year 2006 (defining mt).

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 (defining mt). 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. (defining mt) performed a structural search in academic literature and blog posts. They propose a comprehensive definition for multi-tenancy which is based on the definitions of multi-tenancy presented in academic literature (defining mt). Their definition of multi-tenancy is: Multi-tenancy is a property of a system where multiple customers, so-called tenants, transparently share the systems 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 (defining mt).

In Figure 2 different system levels are illustrated in different variants of multi-tenancy and in single-tenancy for two tenants. In Figure 2 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 (defining multitenancy). 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 Figure 2.

In native or pure multi-tenancy (framework for native multitenancy, defining multitenancy) 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 (framework for native mt).

In the semi-multi-tenancy variants of multi-tenancy 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 table -variant and shared application and separate database -variant (maintenance dream). 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 (dream or nightmare). These performance problems are caused by the expensive operation of loading a database or table in memory when a tenant accesses the application. When comparing this 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 (dream or nightmare).

In multiple instances multi-tenancy 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 (framework for native mt). Two variants of multiple instances multi-tenancy are presented in the Figure 2. 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 (framework for native mt). Multiple instances multi-tenancy variants can support several up to dozens of tenants while native multi-tenancy can support hundreds or thousands of tenants (framework for native mt).

As a comparison point to different multi-tenancy variants a single-tenancy approach is also presented in Figure 2. In traditional single-tenant approach each tenant has own dedicated server and own customized application instance which they use (dream or nightmare). This means that single-tenant application may have many separate running instances which all can be different from each other (dream or nightmare). 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 (framework for native mt).

In single-tenancy server utilization can be low especially when applications are used by customers in the small and medium enterprise (SME) segment of the market (maintenance dream). In that situation server utilization can be improved by placing several tenants on the same server (maintenance dream). Different multi-tenancy variants from multiple instances multi-tenancy to native multi-tenancy can be used to place several tenants on the same server which improves the utilization of the servers. 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 costs of the application will be lower (maintenance dream).

IV. Challenges and benefits of multi-tenancy

Challenges and benefits of multi-tenancy
Overview of the research field

V. Discussion

The section 3 provides answers to research questions 1 and 2. In this paper multi-tenancy has been introduced to the reader from the architectural point of view in Section 3. Multi-tenancy can be applied to different levels of the system. In multiple-instance multi-tenancy level only the hardware or the operating system environment is shared among different tenants and each tenant has own dedicated application instance. Multi-tenancy can also be applied to the database level of the system in which case tenants share one application instance while the tenant specific data is stored using separate databases

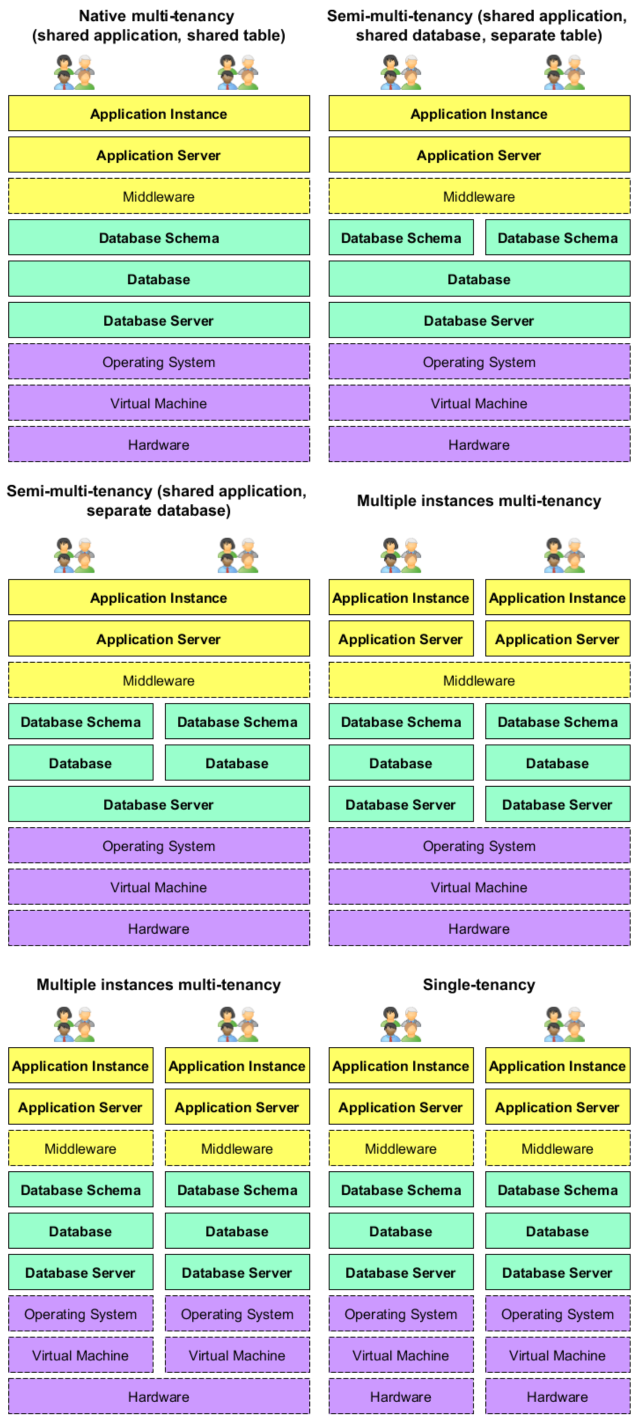
Native multi-tenancy (shared application, shared table)
Semi-multi-tenancy (shared application, shared database, separate table)
Semi-multi-tenancy (shared application, separate database)
Multiple instances multi-tenancy
Multiple instances multi-tenancy
Single-tenancy
Application Instance
Application Server
Middleware
Database Schema
Database
Database Server
Operating System
Virtual Machine
Hardware

Fig. 2. Illustration of different system levels in different variants of multi-tenancy and in single-tenancy.

or database schemas. The highest level of sharing is achieved in native multi-tenancy where all system levels are shared among all tenants.

The connection between Software as a Service business model and multi-tenancy is described in Section 3 by introducing SaaS maturity model to the reader. Multi-tenancy is presented as of the key attributes of a mature SaaS application. Multi-tenancy is part of the third and fourth levels of the SaaS maturity model. 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. The topic of SaaS is also addressed a lot in the academic research of multi-tenancy. This indicates that multi-tenancy is positioned as an architectural tactic for online software (defining mt lis t tekstiin kpl 3!!!!).

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

Some of the challenges of multi-tenancy are discussed in Section 4. One of the challenges is Another challenge is What do they cause? When developing a multi-tenant SaaS application those challenges need to be taken into consideration in order to avoid possible problems. This seminar paper discusses only some of the challenges of multi-tenancy and there can be other challenges which are not mentioned in this seminar paper.

One 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 authors interpretation of multi-tenancy and multi-tenant architecture is only based on scientific literature without real industry experience of multi-tenancy.

Another limitation is that almost only scientific papers were used as references for this seminar paper. The only exception to scientific papers used as a references is the article Architecture Strategies for Catching the Long Tail (Carraro tail) from the MSDN library of Microsoft Corporation. The consequence of this limitation of references is that the current situation of multi-tenancy in industry is not considered at all in this seminar paper

VI. Conclusion

In this paper the multi-tenancy in Software as a Service applications was discussed from the architectural point of view. Multi-tenancy is a high level architectural pattern which can be used in Software as a Service applications to ¡benefit of using multi-tenancy here¿. Multi-tenancy is one of the attributes of a mature well-defined SaaS application. This connection between multi-tenancy and a SaaS application can be seen in SaaS maturity model where multi-tenancy appears on levels three and four. When developing multi-tenant Software as a Service applications the challenges of multi-tenancy should be taken into consideration. ¡maybe something more or is this enough¿

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