A Systematic Mapping Study on the Multi-tenant Architecture of SaaS Systems

Victor Hugo S. C. Pinto, Helder J. F. Luz, Ricardo R. Oliveira, Paulo S. L. Souza and Simone R. S. Souza
Institute of Mathematical and Computer Sciences, University of São Paulo (ICMC-USP)
São Carlos-SP, Brazil
victor.santiago@usp.br, helderfl, ricardoramos, pssouza, srocio{@icmc.usp.br}

***Abstract*—Background: SaaS (Software as a Service) is a services delivery model in Cloud Computing whose applications are remotely hosted by the service provider and available to customers on demand over the Internet. Multi-tenant Architecture (MTA) is an organizational pattern for SaaS that enables a single instance of an application to be hosted on the same hardware and accessed by multiple customers, so-called tenants, with the aim of lowering costs. Tenants are able to configure the system according to their particular needs. Objective: This research aims at the obtaining an overview of the challenges and research opportunities in MTA context for SaaS through a Systematic Mapping Study. Results: Eighty nine primary studies were selected for discussions on advances and opportunities for further investigations. The results showed the relevancy of MTA and pointed out the main research trends for next years in this topic.**

Cloud Computing; software as a service; multi-tenant architecture; systematic mapping study.

I. Introduction

Cloud Computing has emerged from the contribution of techniques from parallel computing, distributed computing and platform virtualization technologies [1]. It provides dynamic resource allocation and has become one of the main research fields in Software Engineering. Furthermore, this technology enables cost reduction, optimization and opportunity for the creation of new business models [2]. A set of resources can be efficiently accessed on demand from anywhere and managed with a minimum possible interaction [3]. Cloud can be understood as a repository of virtualized resources (hardware, development platforms/or services) easily accessible [4]. These resources can be dynamically reconfigured to be adjusted to diversified loads, which optimizes their usage. This wide range of resources has directly contributed to the emerging of different services delivery models, as SaaS (Software as a Service), which is a software deployment model of applications remotely hosted by a service provider and available to customers on demand. It offers benefits, as improved operational efficiency and reduced costs. As an instance, Salesforce.com[1] provides an SaaS for Customer Relationship Management. Salesforce uses a subscription revenue model and charges clients per user on a monthly basis.

The cloud computing environment is different from a traditional environment in terms of hosted deployment, configuration, execution and management of applications. The main difference is related to type of users, security and sharing of resources such as databases, virtual machines or network connections among customers [5]. The sharing of resources among customers through logical separation is one of the main characteristics of multi-tenant architecture (MTA) for SaaS systems.

Multi-tenancy can be referred to an organizational pattern in which a single instance of an application is hosted on the service provider, and multiple companies, so-called tenants, access the same instance [6]. MTA enables a high degree of customization of software according to the requirements of many tenants and resources required for its execution are shared and provided on demand. For the end users, the application is executed in a dedicated environment, i.e., a fault of software in use by another tenant should not affect them. Furthermore, they are able to exclusively configure the system to their specific needs. MTA provides benefits, such as (i) optimization of the use of hardware resources, (ii) costs reduction by the maintenance of applications and (iii) new opportunities for data aggregation. However, challenges as those related to security, data sharing, database, customization, validation and testing, performance and migration from conventional web applications [7][8] must be overcome.

A Systematic Mapping Study (SMS) is a proper method to map a certain topic when few evidence exists or the research topic is wide or scattered. Therefore, we have carried out an SMS on the multi-tenancy of SaaS systems following the guidelines proposed by Kitchenham [9]. Eighty nine primary studies were selected to answer two research questions from the academic perspective. The analysis of the results focuses on presenting the frequencies of publications for different research categories. As main contribution, we have provided a definition of main challenges to guide future research on the multi-tenant architecture domain.

The paper is organized as follows: Section 2 discusses the phases of the SMS; Section 3 addresses the threats to validity and Section 4 reports the conclusions and future work.

II. The Systematic Mapping Study

The SMS was conducted considering three main phases: (i) planning, (ii) conducting and (iii) reporting. The next sections address these phases and the obtained results.

[1] www.salesforce.com/sales-cloud/overview/

DOI reference number: 10.18293/SEKE2016-068

A. Planning

In this phase, the review protocol containing (i) research questions, (ii) search strategy, (iii) inclusion and exclusion criteria and (iv) data extraction process and methodology for the synthesis of the data was defined.

The main goal was the achievement of a background of difficulties related to MTA, alternatives proposed in the literature and research opportunities. Therefore, two research questions (RQ) were defined:

RQ_1: What research topics related to MTA can be found on the current literature?

RQ_2: What are the main research challenges and opportunities related to the development, testing and evolving of multi-tenant SaaS applications?

A search string and the electronic databases were also defined. The search string was elaborated and refined according to an initial set of key papers selected and based on citations of these papers. During the string validation these papers must always be retrieved from electronic databases (Table I). Although subjective, this control enabled the string calibration and identification of possibly relevant studies.

TABLE I. List of key papers used to calibrate the search string

Authors	Ref.
Seungseok et al.	[10]
Sengupta and Roychoudhury	[7]
Tsai et al.	[8]
Ru et al.	[11]

We defined the search string considering the following keywords: cloud, SaaS and multi-tenancy, their frequent variations and boolean operations. Figure 1 shows the search string elaborated. The following databases were considered: *ACM*, *IEEE*, *Scopus* and *Wiley Online*. Such databases cover the main conferences and journals on cloud computing.

(cloud ***and*** *(SaaS* ***or*** *"Software as a Service")* ***and*** *(multi-tenancy* ***or*** *multi-tenant* ***or*** *tenancy* ***or*** *tenant* ***or*** *tenants))*

Fig. 1: Search String.

Relevant primary studies were selected based on the following inclusion (IC) and exclusion criteria (EC). Not all inclusion criteria should be satisfied for each primary study; IC_a is the only mandatory criterion for the inclusion of papers.

IC_a: The primary study presents at least one challenge or research opportunity in the context of MTA;

IC_b: The primary study presents at least one tool, framework, process or APIs for MTA context;

IC_c: The primary study addresses at least one difficulty involving the MTA in usage and migration terms;

IC_d: The primary study presents at least one property, classification or evaluation of a solution considering the MTA;

EC_a: The study presents a challenge or a research opportunity in the MTA context. However, it is a short paper;

EC_b: The study is a Systematic Literature Review;

EC_c: The whole study is unavailable.

We have used a data extraction form to answer the review questions, presented in Table II. We have included some categories (Item 5) in order to classify the main domain of each primary study, for instance, "Customization" and "Database" are defined categories.

TABLE II. Contents of data extraction form

	Attributes
Metadata	ID, reviewer and date.
Content	Title, year, source (i.e. conference or journal) and search database.
Data extracted	1) Challenge/opportunities; 2) Tools, frameworks and APIs; 3) Difficulties in the MTA usage and adoption/migration; 4) Specifications, classification and evaluation of solutions for MTA and 5) Category of paper.

During the data extraction process, the data from primary studies were collected by three reviewers, PhD candidates in Computer Science. They were extracted by one researcher and checked by another. This SMS was performed between May and August, 2015 and the data have been documented and are available[2].

B. Conducting

In this phase, the primary studies were identified in the aforementioned search databases. *Scopus* returned a larger set of studies (638). *IEEE*, *ACM*, *Wiley* returned 168, 594 and 25, respectively. Figure 2 shows the distribution of papers retrieved in each search database and after applying the inclusion and exclusion criteria in the reading process. From this initial set, 135 duplicated studies were identified and removed. In the selection phase, based on the partial reading (titles and abstracts), a set of 149 papers was selected according to the inclusion and exclusion criteria; after the full reading, only 89 papers were selected. We wanted to be conservative as possible, therefore the search string has become generic to retrieve many studies from electronic databases, even if it would give us more effort in the selection process. Many papers were introduced as primary studies, but only few of them had more contributions or larger impacts.

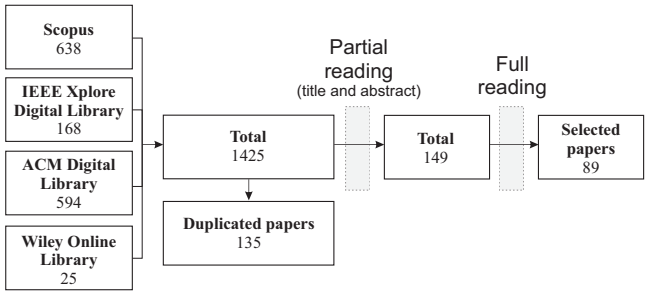

Fig. 2: Distribution of papers (conduction phase).

In order to validate the inclusion and exclusion criteria application, each primary study was scored by reviewers in

[2]https://goo.gl/0K68jp

both partial (title and abstract) and full readings. A score "0" means rejected and "1" accepted. In cases of doubt, the reviewer scored the paper with "0.5" and the other reviewers were asked about its relevance, so that a consensus could be reached through the adoption of score "0" or "1".

The scoring process was conducted in a sequential and independent way, i.e., one reviewer read and scored each paper without interference from others. In partial reading all papers were scored and posteriorly, in full reading they were classified again by reviewers until reaching a set of relevant studies to answer the research questions.

C. Reporting

This section discusses an overview of MTA based on selected primary studies.

1) RQ_1: **What research topics related to MTA can be found on the current literature?**

In order to clarify the focus of the selected studies in quantitative terms, we have defined some categories according to the paper domain, as aforementioned. Figure 3 shows a mapping containing number of primary studies distributed according to publication year and category, which one paper can be classified and more than one category.

Security, testing activity and experiments may be considered important issues to quality assurance and, therefore, these issues are into quality assurance category.

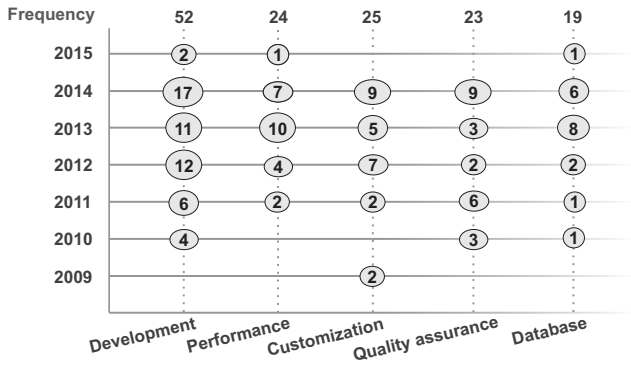

Fig. 3: Distribution of primary studies according to categories.

Figure 4 presents the disposal of the papers selected from workshops, journals and conferences. Conferences have a dominant position with 63 papers (70.8%), followed by journals with 18 papers (20.2%) and workshops, 8 papers (9%). It can indicate that workshops still have to be formed and researchers submit their results to conferences and journals with a larger scope.

The next sections discuss the main idea of the selected studies organized by the categories shown in Figure 3.

a) Development: Architectures, frameworks, requirements and variability management, and migration from web conventional applications to multi-tenant SaaS applications are mentioned in the current multi-tenancy literature as important issues to address in future research.

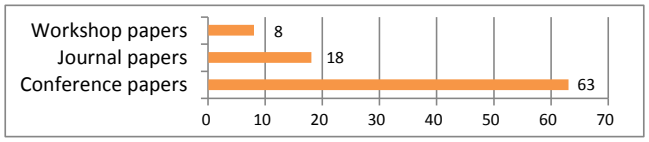

Fig. 4: Quantity of selected papers from workshops, journals and conferences.

Due to the complexity of the management and maintenance costs, SaaS providers commonly develop a single version of the application for all tenants. Truyen et al.[12] evaluated the context-oriented programming (COP) for such applications aiming to improve their development process. The main idea was to enable the customization of specific requirements for each tenant, providing positive results in terms of development effort.

Requirements management is one of the difficulties related to the supply of satisfactory applications for certain tenants. Walraven et al. [13] presented an alternative based on product lines and the co-existence of specific settings for tenants to facilitate the requirements management. They used the approach in some multi-tenant SaaS applications and as a result, the efforts to configure and compose variant applications were reduced.

Multi-tenant SaaS applications aim at providing different settings to address the demands of their tenants in functional requirement terms. However, it is natural that different requirements from the previously provided be needed by new tenants. On the one hand, new features can be developed and included in an application, so that the existing instances continue running without failure. On the other hand, costs related to efforts in the regression testing must be considered during the development of new features [7].

Kale and Borhade [14] provided a framework that covers features as portability, customization, security and scalability aiming to reduce the time spent on the software development. In the same vein, Nam and Yeom [15] proposed a framework to support different services for a large set of tenants. According to the authors, despite the benefits of MTA, few processes can support the development of multi-tenant SaaS applications.

Manduca et al.[16] described an approach for the development of multi-tenant SaaS applications with a single database from conventional web applications. It employs architectural components and design patterns and keeps the functionalities in the current programming language with no need for substantial changes. As a difficulty, the authors highlighted the platform documentation, which is often insufficient to develop this type of applications.

Table III presents the issues related to development of multi-tenant SaaS applications investigated. Due to space limitation we are showing only some papers for each issue.

b) Performance: MTA has introduced new challenges related to load balancing and resource allocation, including the requests on the tenant level, service level agreement, performance objectives and quality of services. Sun et al. [18]

TABLE III. Main issues about development of multi-tenant SaaS applications

	Description	Ref.
Requirements	Composition of variant multi-tenant applications following the product lines engineering	[13]
Implementing	Context-oriented programming	[12]
	Guide to implementing new tenants without impacting those already deployed	[7]
Frameworks	Process to support the portability, customization, security and scalability	[14]
	Framework to address availability, extensibility and scalability in a multi-tenant application	[17]
Migrating	Process to support the migrating of conventional web applications towards multi-tenant SaaS application with relatively less effort	[16]

proposed a suitable load balancing policy for a multi-tenant environment to provide satisfactory quality of services. On a database level, Moon et al. [19] presented a load balancer for multi-tenant databases to increase the performance and sharing of resources among tenants. Patikirikorala et al. [20] developed an approach that uses a nonlinear replenished control to keep the performance in distinct usage levels for different tenants, depending on their priorities. It enables the detection of overload, therefore the control of tenants operations can be dynamically changed.

Krebs et al. [21] extended a web benchmark called TPC-W to include multi-tenancy and compared the cloud usage under two perspectives: (i) multi-tenancy and (ii) virtualization. Multi-tenancy shown higher efficiency than virtualization considering the throughput, number of tenants and when memory was a bottleneck.

c) Customization: A considerable number of papers have addressed applications customization. A multi-tenant SaaS application which address a large set of tenants should make possible a large number of customizations [22]. The customization of a complex application is an error-prone task, it requires high manual efforts and the users may not know the best choice in terms of customization. Thus, the authors performed a study of the possibilities available for the customization of an SaaS application, and a semi-automatic customization process was created to reduce efforts.

Ramachandran et al. [23] observe that the customization may result in high cost of readjustments. For multi-tenant systems, it involves the configuration of specific instances and management of allocated resources for the tenants. For Walraven et al. [24] customization involving variations in the core of the application is expensive for SaaS providers, introducing an additional complexity.

d) Quality assurance: Quality assurance is a promising research topic in MTA [25] that includes testing strategies, metrics and quality criteria, and alternatives related to security. For Tsai et al. [8], one of the main challenges in the testing activity of multi-tenant SaaS application is to deal with the large set of composition possibilities and interactions among components. The authors provided a combinatorial testing approach to generate dynamic test sequences and achieve a high structural coverage. The main idea was to identify the compositions likely to result in failures by an algorithm. When a new component is composed in a certain application instance becomes available, the algorithm reveals defects in the interactions among components.

The complexity of the cloud computing model and lack of standardization become the security a critical issue for cloud providers and customers. According to Wood et al. [26], multi-tenancy directly impacts on the applications development and the way they are provided. Almorsy et al. [27] created a framework to improve collaboration between service providers and consumers and manage the security of cloud platform and its hosted services.

Table IV presents the main issues investigated in relation to quality assurance in the MTA context. Due to space limitation we are showing only some papers for each issue.

TABLE IV. Main issues about quality assurance of multi-tenant SaaS applications

	Description	Ref.
System Testing	Combinatorial testing: dynamic test sequences were used to achieve high architectural coverage	[8]
Regression Testing	Continuous testing with partitioning of data from tenants and generation of test case based on meta-data	[28]
Security	Framework for security management	[27]
	SecPlac: resource allocation model to support the security in the sharing of infrastructure among tenants	[29]
	TOSSMA (Tenant Oriented SaaS Security Management Architecture): an architecture to isolate resources for tenants through the injection of authorization controls	[30]
	Data combination privacy	[31]
QoS	MSSOptimiser (Multi-tenant SaaS Optimizer): an approach to select services addressing quality requirements	[32]

e) Database: Nineteen papers in SMS have cited database-related issues as a promising research direction. Saraswathi and Bhuvaneswari [33] presented two alternatives for multi-tenant data architecture: i) one related to authentication and authorization and ii) a non-intrusive approach for large-scale applications. The authors described a process to apply them and guide engineers in the development of databases.

Maenhaut et al. [34] developed an approach for data management in a hierarchical way and taking into account some performance metrics. The main question addressed concerned the distribution of users and data into multiple instances of database. Yaish et al. [35] discussed an access control model based on a database schema. They also proposed an access control algorithm that enables users to access the data granted based on users groups or assigned roles.

2) RQ_2: **What are the main research challenges and opportunities related to the development, testing and evolving of multi-tenant SaaS applications?**

Although most studies have addressed the development of multi-tenant SaaS applications, standards are scattered, and do not often follow a methodical approach. Furthermore, the solutions are proprietary and rarely interoperable [25]. Traditional software testing cannot be applied to test applications in a Cloud environment due to it is designed for on-premise single-tenant applications [36].

Several issues should be considered during the testing of multi-tenant SaaS application: (i) resources are shared among tenants and their end-users, (ii) each variant application addresses a specific requirements set for a tenant, it is executed as if it was in a dedicated environment and can be composed of several components and (iii) a variant application is delivered to the customers through a run-time engine from cloud provider that weaves the tenant customization data and specific metadata to kernel code. Thus, each application provides different screens and logic.

According to Alkhatib et al. [25], the community still does not have effective quality metrics for the SaaS and new testing strategies are required to meet the challenges imposed by the cloud computing model. Software integration testing issues, validation methods and quality assurance standards addressing the interaction interfaces must be established. Since high system availability is essential to SaaS, the re-testing techniques considering the multi-tenancy feature are mandatory whenever software is changed for improvements or bug-fixing.

Recent studies on tests in cloud computing have addressed the verification of nonfunctional requirements as performance and security. Considering the selected studies, we have identified that research fields as (i) adjustments of conventional test criteria, (ii) test strategies for customization components, (iii) alternatives to verify the composition interfaces and impact new components, and (iv) approaches to regression test require more cooperation between industry and academia.

Regarding the evolution of multi-tenant SaaS applications, the community still has not provided well-defined approaches. The evolving activity of distributed systems may indicate guidelines for dealing with the isolated execution of instances of these applications.

3) Research agenda: In order to guide future research in the area of multi-tenancy, this section presents the major trends identified in this study, as follows:

Development process. Most of selected studies about development process proposed a solution or a process without a practical evaluation. Methodical approaches to guide the development of cloud-based applications require more research effort, especially taking into account the multi-tenancy.

Quality metrics. Metrics are used to guide managers during the software quality evaluation. Despite many studies mentioning their importance, there is still a lack of quality metrics for this context.

Testing activity. Few organizations and academy provide security testing, recovery testing, fault-tolerance testing or some alternative to cover the complexity of multi-tenancy for SaaS. In addition, there is a lack of standards to driven the development of interoperable test tools. From our point of view, an ideal testing environment for multi-tenant SaaS systems needs to support the testing of a tenant specific application in runtime, without impacting others. For this, there are two main issues that should be carefully considered: (i) a variant application is generated through a dynamic compiler that combines the tenant specific metadata and customization data to kernel code and (ii) tenants can apply changes according to their concerns. Thus, it is important to ensure that the testing will not result in side effects to other tenants. A possible testing strategy is to generate the variant application that we want to test, perform its isolating and conduct the testing activity without making the other applications unavailable.

Continuous validation. Despite the on-demand software validation in the cloud environment includes the regression testing and frequent changes in the application, most of the regression test studies have focused on retest a version in a previously configured test environment. However, validation methods must be dynamic to deal with multi-tenancy.

Empirical studies to evaluate testing techniques and criteria. There is a lack for guidelines about which testing techniques and criteria to use considering the testing objectives in context of SaaS systems. It is necessary to know the usability, effectiveness and cost of these techniques and criteria. We have observed a lack of empirical studies to evaluate the use of techniques and testing criteria in cloud computing domain. In addition, many researchers argue that the traditional software testing cannot be satisfactorily applied to test cloud computing applications. This happens due to implicit characteristics of these applications, such as the high customization capabilities, dynamic environment and multi-tenancy.

Migrating process. Consolidated processes to guide the migration from web conventional to multi-tenant SaaS model can contribute with the adoption of this model for applications where only the multi-user model is not enough.

III. THREATS TO VALIDITY

Selection of primary studies. In order to ensure an unbiased selection process, research questions were defined and exclusion and inclusion criteria were specified for the obtaining of relevant studies. However, threats cannot be ruled from a quality evaluation perspective, although the studies were selected by a score assignment based on relevance.

Relevant primary studies not selected. Although many sources were used for the selection of primary studies, some might have been neglected. We tried to reduce this threat by selecting sources that index studies from the main scientific sources in cloud computing and covering most of the relevant papers.

Reviewers reliability. All reviewers that participated in this study work on cloud computing. The protocol was assessed by specialists, so that deviations during the analysis could be avoided.

Data extraction. It is worth mentioning that not all information was obvious to answer the research questions. Several sources, as external papers and technical reports were consulted, so that the validity of the process could be ensured.

In case of disagreement among the reviewers, a specialist was called to guarantee the correct decision.

IV. CONCLUDING REMARKS

Software as a service is a way of delivering applications over the Internet as a service for multiple customers. From the point of view of service providers, the computing resources to be offered must be broadly shared. For the users, it is important to customize the application according to their specific requirements. In this scenario, an architectural pattern called multi-tenancy is gaining more ground in the application space on cloud.

In order to provide a mapping of research topics on the multi-tenancy of SaaS systems and identifying new research opportunities, we have conducted an SMS in which 89 primary studies were selected for discussions. We have defined two research questions that reflect the scope of the study and five categories to map the contributions and challenges.

This mapping study also points out the need for experimental studies evaluating the proposed approaches and a systematic test strategy extending different techniques to increase the quality of these applications. In our future research, we intend to compare the evidence identified in this work with evidence from industrial cloud projects in order to define new hypotheses, which will guide the definition of approaches for testing of multi-tenant SaaS applications.

REFERENCES

[1] J. Ru and J. Keung, "An empirical investigation on the simulation of priority and shortest-job-first scheduling for cloud-based software systems," in *Software Engineering Conference*. IEEE, 2013, pp. 78–87.

[2] S. Tai, J. Nimis, A. Lenk, and M. Klems, "Cloud service engineering," in *32nd Int. Conf. on Software Engineering*, 2010, pp. 475–476.

[3] P. Mell and T. Grance, "The nist definition of cloud computing," NIS, Tech. Rep., 2010.

[4] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds," *ACM Computer Communication Review*, vol. 39, no. 1, p. 50, 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1496091.1496100

[5] I. Chana and P. Chawla, "Testing perspectives for cloud-based applications," in *Software Engineering Frameworks for the Cloud Computing Paradigm*. Springer, 2013, pp. 145–164.

[6] C.-P. Bezemer and A. Zaidman, "Multi-tenant saas applications: Maintenance dream or nightmare?" in *Work. on Software Evolution and Int. Work. on Principles of Software Evolution*, 2010, pp. 88–92.

[7] B. Sengupta and A. Roychoudhury, "Engineering multi-tenant software-as-a-service systems," in *3rd Int. Work. on Principles of Engineering Service-Oriented Systems*, 2011, pp. 15–21.

[8] W.-T. Tsai, Q. Li, C. J. Colbourn, and X. Bai, "Adaptive fault detection for testing tenant applications in multi-tenancy saas systems," in *IEEE Int. Conf. on Cloud Engineering*, 2013, pp. 183–192.

[9] B. Kitchenham, "Procedures for performing systematic reviews," Keele University, Tech. Rep., 2004.

[10] S. Kang, J. Myung, J. Yeon, S.-w. Ha, T. Cho, J.-m. Chung, and S.-g. Lee, "A general maturity model and reference architecture for saas service," in *Database Systems for Advanced Apps.*, 2010, pp. 337–346.

[11] J. Ru, J. Grundy, and J. Keung, "Software engineering for multi-tenancy computing challenges and implications," in *Int. Work. on Innovative Soft. Dev. Methodologies and Practices*, 2014, pp. 1–10.

[12] E. Truyen, N. Cardozo, S. Walraven, J. Vallejos, E. Bainomugisha, S. Günther, T. D'Hondt, and W. Joosen, "Context-oriented programming for customizable SaaS applications," in *27th ACM symposium on applied computing*, 2012, pp. 418–425.

[13] S. Walraven, D. Van Landuyt, E. Truyen, K. Handekyn, and W. Joosen, "Efficient customization of multi-tenant software-as-a-service applications with service lines," *JSS*, vol. 91, pp. 48–62, 2014.

[14] S. S. Kale and R. H. Borhade, "Development of multitenant saas framework at single instance and with zero effort multitenancy," in *Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2013, pp. 834–839.

[15] T. Nam and K. Yeom, "Ontology model to support multi-tenancy in software as a service environment," in *Int. Conf. on Future Internet of Things and Cloud*, 2014, pp. 146–151.

[16] A. M. Manduca, E. V. Munson, R. P. Fortes, and M. G. C. Pimentel, "A nonintrusive approach for implementing single database, multitenant services from web applications," in *29th ACM Symposium on Applied Computing*, 2014, pp. 751–756.

[17] P. Morakos and A. Meliones, "Design and implementation of a cloud saas framework for multi-tenant applications," in *5th Int. Conf. on Information, Intelligence, Systems and Applications*, 2014, pp. 273–278.

[18] H. Sun, T. Zhao, Y. Tang, and X. Liu, "A qos-aware load balancing policy in multi-tenancy environment," in *8th Int. Symposium on Service Oriented System Engineering*, 2014, pp. 140–147.

[19] H. J. Moon, H. Hacigumus, Y. Chi, and W.-P. Hsiung, "Swat: A lightweight load balancing method for multitenant databases," in *16th Int. Conf. on Extending Database Technology*, 2013, pp. 65–76.

[20] T. Patikirikorala, I. Kumara, A. Colman, J. Han, L. Wang, D. Weerasiri, and W. Ranasinghe, "Dynamic performance management in multi-tenanted business process servers using nonlinear control," in *Int. Conf. on Service-Oriented Computing*, 2012, pp. 206–221.

[21] R. Krebs, A. Wert, and S. Kounev, "Multi-tenancy performance benchmark for web application platforms," in *Int. Conf. on Web Eng.*, 2013, pp. 424–438.

[22] W.-T. Tsai and X. Sun, "SaaS multi-tenant application customization," in *IEEE 7th Int. Symposium on Service Oriented System Engineering*, 2013, pp. 1–12.

[23] L. Ramachandran, N. C. Narendra, and K. Ponnalagu, "Dynamic provisioning in multi-tenant service clouds," *Service Oriented Computing and Applications*, vol. 6, no. 4, pp. 283–302, 2012.

[24] S. Walraven, E. Truyen, and W. Joosen, "A middleware layer for flexible and cost-efficient multi-tenant applications," in *Int. Conf. on Middleware*. Springer, 2011, pp. 370–389.

[25] H. Alkhatib, P. Faraboschi, E. Frachtenberg, H. Kasahara, D. Lange, P. Laplante, A. Merchant, D. Milojicic, and K. Schwan, ""IEEE CS 2022 report"," IEEE Computer Society, Tech. Rep., 2014.

[26] K. Wood and M. Anderson, "Understanding the complexity surrounding multitenancy in cloud computing," in *IEEE 8th Int. Conf. on e-Business Engineering*, 2011, pp. 119–124.

[27] M. Almorsy, J. Grundy, and A. S. Ibrahim, "Collaboration-based cloud computing security management framework," in *IEEE Int. Conf. on Cloud Computing*, 2011, pp. 364–371.

[28] W.-T. Tsai, Q. Shao, Y. Huang, and X. Bai, "Towards a scalable and robust multi-tenancy SaaS," in *Asia-Pacific Symp. on Internetware*, 2010, p. 8.

[29] E. Saleh, J. Sianipar, I. Takouna, and C. Meinel, "Secplace: A security-aware placement model for multi-tenant SaaS environments," in *Int. Conf. on Ubiquitous Intell. and Comp.*, 2014, pp. 596–602.

[30] M. Almorsy, J. Grundy, and A. S. Ibrahim, "Tossma: A tenant-oriented saas security management architecture," in *IEEE 5th Int. Conf. on Cloud Computing*, 2012, pp. 981–988.

[31] K. Zhang, Q. Li, and Y. Shi, "Data privacy preservation during schema evolution for multi-tenancy applications in cloud computing," in *Web Information Systems and Mining*. Springer, 2011, pp. 376–383.

[32] Q. He, J. Han, Y. Yang, J. Grundy, and H. Jin, "Qos-driven service selection for multi-tenant saas," in *Int. Conf. on Cloud Computing*, 2012, pp. 566–573.

[33] M. Saraswathi and T. Bhuvaneswari, "Multitenant SaaS model of cloud computing: Issues and solutions," in *Communication and Network Technologies*. IEEE, 2014, pp. 27–32.

[34] P.-J. Maenhaut, H. Moens, M. Decat, J. Bogaerts, B. Lagaisse, W. Joosen, V. Ongenae, and F. De Turck, "Characterizing the performance of tenant data management in multi-tenant cloud authorization systems," in *Network Operations and Management Symposium (NOMS), IEEE*, 2014, pp. 1–8.

[35] H. Yaish and M. Goyal, "A multi-tenant database architecture design for software applications," in *ICCSE*, 2013, pp. 933–940.

[36] Z. Mahmood and S. Saeed, *Software engineering frameworks for the cloud computing paradigm*. Springer, 2013.