Journal of Computer Science & Computational Mathematics, Volume 8, Issue 3, September 2018
DOI: 10.20967/jcscm.2018.03.001

Customization of Software as a Service Application: Problems and Objectives

Abdulrazzaq Qasem Ali*, Abu Bakar Md Sultan, Abdul Azim Abd Ghani, Hazura Zulzalil

Dept. of Software Engineering and Information System,
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia
*Corresponding author email: s100032@yahoo.com

***Abstract*:** The key feature of SaaS application is that it addresses the needs of many different customers using only one software, rather than multiple developments and versions, a concept known as multi-tenancy. However, it typically results in a one-size-fits-all approach and the application only fulfills the requirements that are generic to all tenants. Therefore, a multi-tenancy SaaS application is reliant upon an ability to be customized in order to be successful. Responding to distinctive needs of each customer and maintaining the key feature of developing SaaS application creates complexity in SaaS customization. This paper delves into some challenges related to SaaS customization, and then maps among these challenges in order to structure the problem and objective trees of SaaS customization. This paper may serve as a step towards reducing complexity in SaaS customization.

***Keywords*:** software as a service, customization, multitenant, problem tree, objective tree.

1. Introduction

Software as a service has become a commonplace model for many business applications. More and more companies have followed this model in delivering their software to their customers. SaaS delivery model is a multi-tenant model which is different from multi-instance in which all tenants share one instance and it reduces cost by serving more users [1, 2]. In SaaS delivery model, SaaS providers are not allowed to develop a version of application code for each individual tenant, but they have to allow each tenant to fulfil their unique requirements by enabling them to customize the application [2, 3].

The data relating to a multi-tenancy application must be held separately in a database, and since all such applications will use the same software, it is unwise to make changes to the software itself. Also, it must be possible to access this information when required to ensure an efficiently customized application. The addition of such options does make the overall procedure slightly slower due to the extra computer input. It also makes the process more difficult, with more stores of data [4].

This paper delves into some challenges related to SaaS customization, and then maps among these challenges in order to structure the problem and objective trees of SaaS customization. This paper may serve as a step towards low complexity in SaaS customization.

This paper is organized as the following parts: Section 2 outlines the method of this study, Sections 3, 4, 5, 6, and 7 illustrate some challenges related to SaaS customization, Sections 8 and 9 present the problem and objective trees of SaaS customization, and Section 10 concludes and gives suggestion for future investigations.

2. The Study Method

Two phases were considered for this study, as shown in Figure 1. Each of the phases leads to the formulation of an outcome which is connected to the next phase.

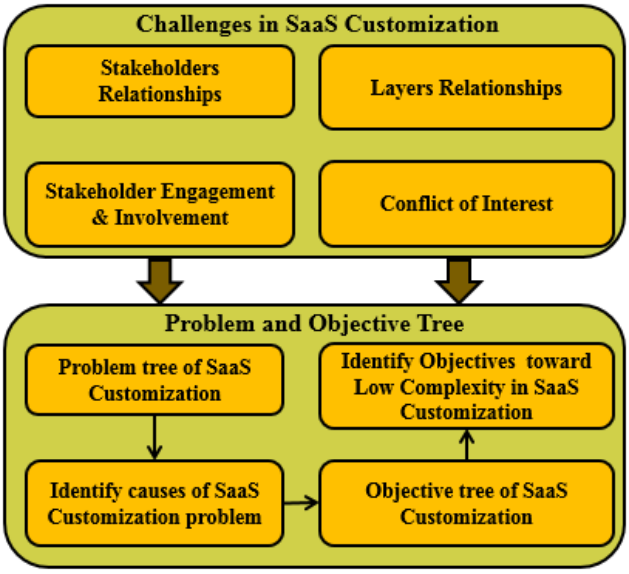

Figure 1. The Study Method

The first phase explores some challenges related to SaaS customization, illustrated by using previous research about Software and SaaS customization specifically those related to SaaS stakeholders and architecture layers.

The second phase of this study uses both problem and objective trees to identify the hurdles encountered by SaaS customization. Whilst the former is concerned with the reasons for the problem existing, the latter is a more comprehensive breakdown of targets that contains the routes to meet the main objective.

3. Stakeholders Relationships in SaaS Customization

Relationships between stakeholders characterize how the behaviors of different stakeholders overlap or interact. Figure

2 captures the overview of the stakeholder relationships involved in SaaS customization.

As observed in Figure 2, both SaaS providers and SaaS tenants intersect at the configuration management facility which provides them with APIs to manage the variability of the application as well as to manage and retrieve tenant-specific configurations [5-7]. However, both SaaS providers and SaaS tenants customize the SaaS application separately where the configuration management facility offers separate interfaces with different capabilities for each of them.

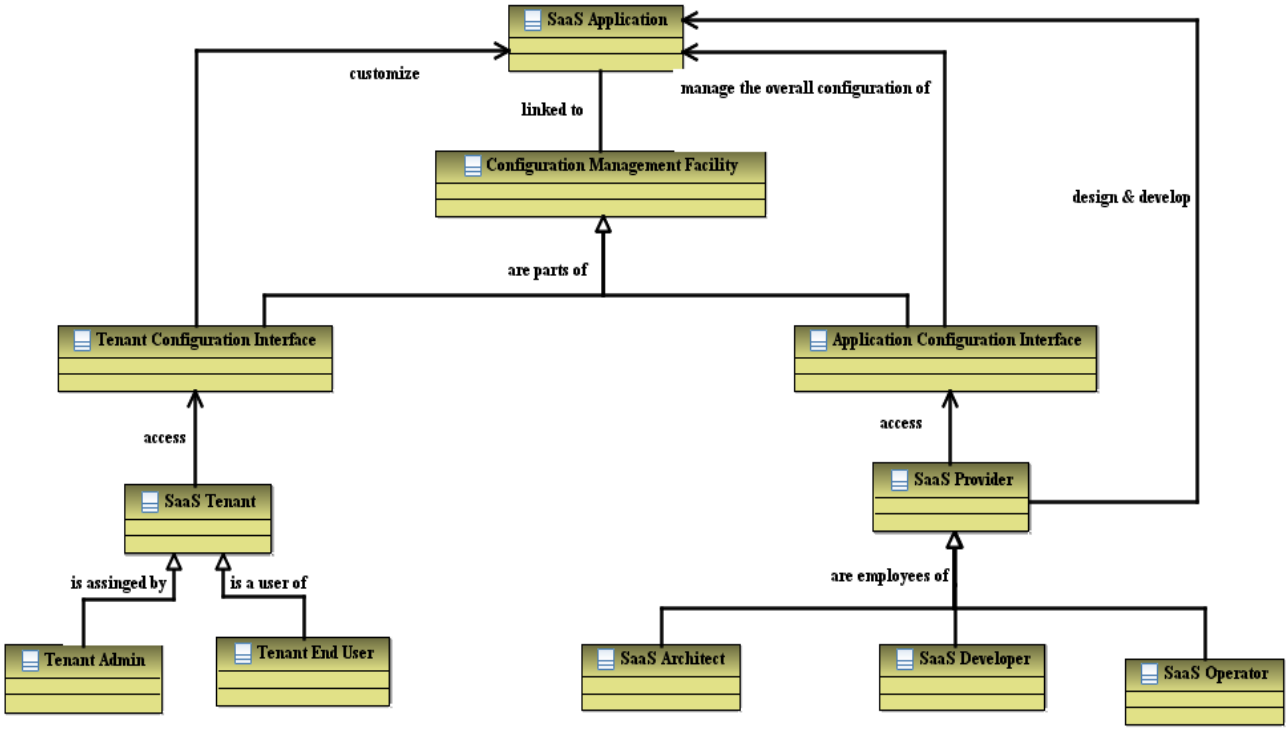

Figure 2. Overview of Stakeholder Relationships involving in SaaS Customization

4. Layers Relationships in SaaS Customization

All elements of SaaS application are impacted by tenant-specific customization, including both the functional and the design aspects of the GUI, business processes and databases [8]. Customizations which involve multiple levels are more complicated, since they may have an unwanted impact on various relationships within the structure to be customized [5]. Figure 3 shows the multilayered relationships involved in SaaS customization.

5. Stakeholder Engagement in SaaS Customization

The interests of the various stakeholders are not of equal importance, and so each has a different impact on the targets of the SaaS application. Stakeholders' impact on SaaS customization can be categorized into power or influence, and accessibility scale with regards to SaaS application resources. Furthermore, many other factors have to be considered in order to measure the stakeholders' impact, such as involvement in SaaS DLC and stakeholders' interests, which are discussed in the next section.

SaaS tenants are empowered to customize their SaaS application but their customizations are limited by the customizations done by the SaaS provider and tenant administrator. Conversely, the SaaS provider's customization has no such prerequisite but has the power to access all SaaS application resources.

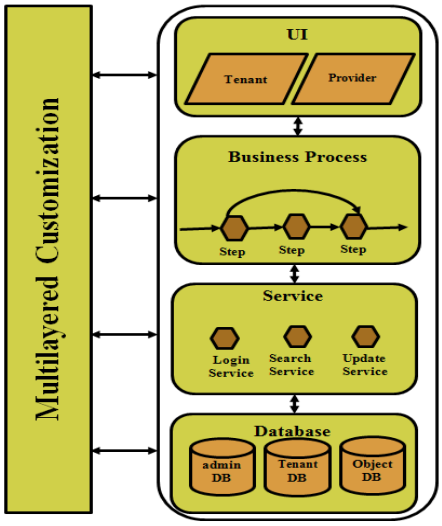

Figure 3. Customization Crosscut SaaS Architecture

Table 1 shows the degree of stakeholders' impact associated with their power and accessibility scale.

Table 1 indicates that an increase of resource accessibility by the stakeholder means an increase in their power in

customizing SaaS application, and it increases their impact and involvement in SaaS customization. In addition, it shows that stakeholders from the tenant's side have a lower impact on SaaS customization, and this shows the need to improve the influence of the SaaS tenants on SaaS customization, whilst taking into account the other key features of SaaS.

SaaS DLC highlights that customization must be at the forefront of development, even at the start of the process. The lack of consideration of customization at the requirements stage leads SaaS providers to design and implement a poor SaaS application. Therefore, consideration of SaaS customization is at its highest degree at the first stage of SaaS DLC, and this degree decreases gradually in the next stages of SaaS DLC. Conversely, the degree of capturing tenant customizations is the highest at the last stage of SaaS DLC where the new tenant and market requirements must be collected using the provided customization and configuration facility, and this degree decreases in deployment and test stages.

Table 1. The Degree of Stakeholder's Impact in SaaS Customization

Resources Accessibility	GUI	High	Slightly High	Slightly High
	Service	High	Low	Medium
	Workflow	High	Low	Medium
	Database	High	Low	Medium
	All	High	Low	Medium
	Stakeholder	**SaaS Provider**	**Tenant End User**	**Tenant Admin**
Power	GUI	High	Low	Medium
	Service	High	Low	Medium
	Workflow	High	Low	Medium
	Database	High	Low	Medium
	All	High	Low	Medium
Overall Impact		High	Low	Medium

6. Stakeholder Involvement in SaaS Customization

Furthermore, the more stakeholders involved in SaaS DLC, the more they have an impact on any SaaS application, and the more influence they have on the goal of the customized SaaS application. Table 2 shows the degree of stakeholders' involvement in SaaS DLC.

7. Conflict of Interest among SaaS Stakeholders

The main reason for SaaS providers to develop SaaS application is to exploit the economies of scale [9]. To achieve that, the SaaS application has to be offered with the following features [9, 10]:

- Multitenancy: an application which addresses the needs of many different customers using only one software, rather than multiple developments and versions.
- Scalability: the capability of an application to handle a growing number of tenants by dynamically creating multiple copies of the same SaaS software, each of which can be called upon to provide services to be enlarged in order to accommodate that growth.

Conversely, the main interest of SaaS tenant is to gain a competitive advantage, and this is in relation to low costs and differentiation advantage. A low-costs advantage is gained by subscription and pay-per-use pricing in SaaS. On the other hand, a differentiation advantage is achieved by having an application that satisfies the common requirements of their domain and their own unique requirements. SaaS application typically achieves these commonalities by a one-size-fits-all approach [6]. Thus, customization avoids the problem of individual requirements being overlooked. It is not possible, however, for the delivery model to enable the SaaS application provider to create individual application codes for each customer.

Table 2. The degree of Stakeholder involvement in SaaS DLC

SaaS DLC / Stakeholder	**Specification**	**Design**	**Implementation**	**Deployment & Testing**	**Maintenance & Support**	**Overall**
SaaS Provider	High	High	High	High	High	High
SaaS Tenant	Medium	NA	NA	Low	Low	Low

Figure 4 shows the main interests of both the SaaS provider and the SaaS tenant and indicates where the conflicts occur

8. Problem Tree of SaaS Customization

The purpose of the problem tree is to define and analyze the causes and effects of SaaS customization problems as the first step towards an optimal solution for SaaS customization. The problem tree is comprised of the roots, the trunk and the branches. The roots show the reasons for the primary problem, represented by the trunk, and the branches show the impacts. Indeed, the main elements of the problem tree of SaaS customization have been identified from previous sections of this work. Figure 5 shows the problem tree related to SaaS customization.

The main problems of SaaS customization are as follows:

8.1 One-size-fits-all Approach

The key feature of SaaS application is that it addresses the needs of many different customers using only one software, rather than multiple developments and versions. An SaaS application achieves the highest degree of resource sharing between tenants, but typically results in a one-size-fits-all approach: the multi-tenant application only satisfies the requirements that are common to all tenants. Consequently, one significant issue that should be taken into consideration when the tenant intends to subscribe to an SaaS application is the standard processes that make the tenants almost identical and reduce the possible competitive advantages. Expressly, the functionality and quality that individual tenants need in an SaaS application are typically different from each other and these unique requirements need to be implemented in the application as well.

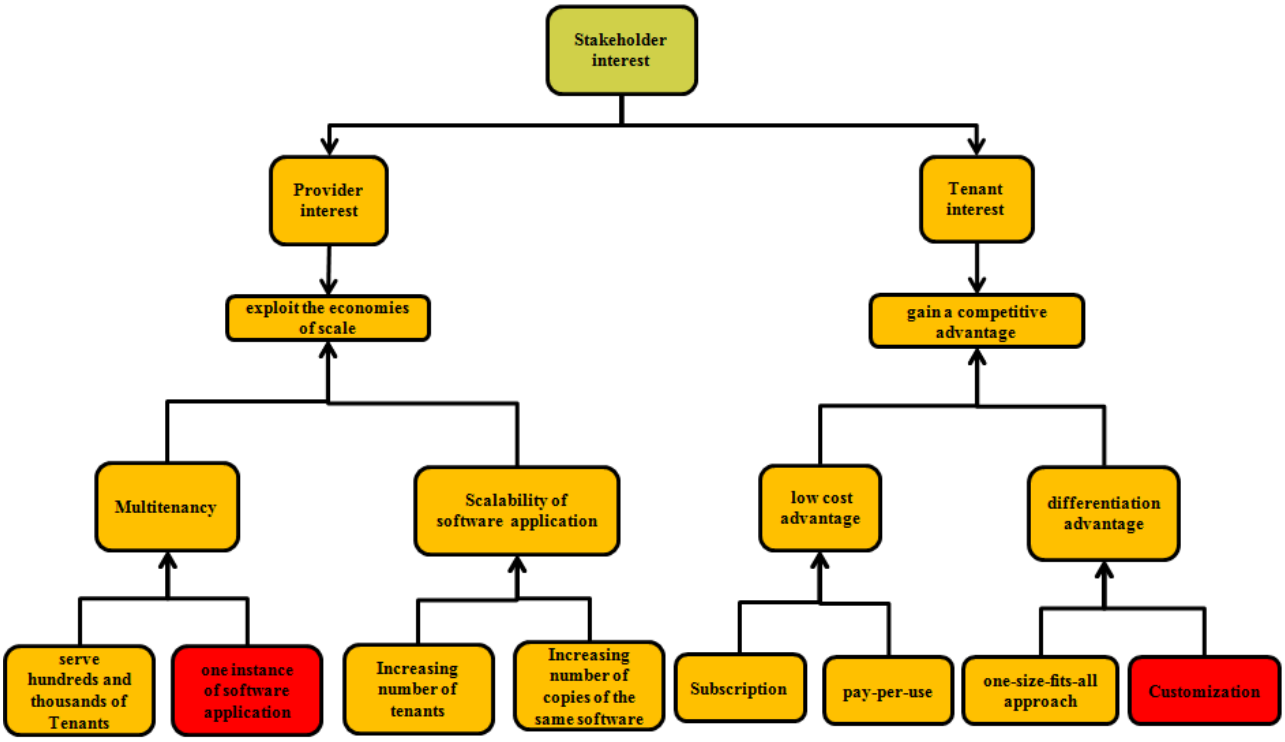

Figure 4. Stakeholder Interests Indicating the Conflict Points

8.2 Customization Implementation

In many cases, customization is defined in the process of enforcing a new requirement into the software that has not been considered during the development life cycle which requires modifications in source code[11]. This type of modification requires a deep knowledge and understanding of application domain and business domain as well [11,12]. Therefore, implementation of such a requirement is usually limited to SaaS provider.

8.3 Cross-Layer Relationships

Customizations which crosscut different layers of SaaS architecture are likely to be more complex and thus more difficult to provide. For instance, the customization will not be accomplished if the customized object has a relationship with another object (i.e. depends on another object). Unfortunately, there is lack of research in the literature on how to facilitate the complexity of customizing SaaS applications' objects that have cross-layer relationships.

8.4 Low level of Tenant Engagement in SaaS Customization

It has to be shown in section 5 that SaaS tenants are empowered to customize their SaaS application but their customizations are limited by the customizations done by the SaaS providers. In addition, it shows that the tenant has a lower impact on SaaS customization and this shows the need to improve the influence of the SaaS tenants on SaaS customization by taking into account the other key features of SaaS. Prior to that, there is a need to analyze the impact of SaaS customization on the unique features of SaaS due to the very nature of the development and deployment of this type of application.

8.5 Low Level of Tenant Involvement in SDLC

Verily, the fewer tenants involved in SaaS DLC, the smaller the impact they have on SaaS application, and the lesser the influence they have on the goal of the customized SaaS application. At the same time, heavy involvement of tenants in all SaaS DLC is not desirable. Often, problems arise with SaaS applications because customization has not been a consideration of the design at the initial requirements stage.

It is thus challenging to make alterations later on. As a result, SaaS application provider may be forced to make substandard choices due to the restrictions imposed upon the requirements.

8.6 Low Security Isolation of SaaS Resources

A high degree of resource sharing complicates isolation between tenants, as customizations have to be isolated from the application so that they only assign to a particular tenant and do not affect the service behavior of other tenants (e.g. functionality, performance, and security). Tenants may be unaware of the location of their personal data, and it is important that this data is not combined with that of other customers.

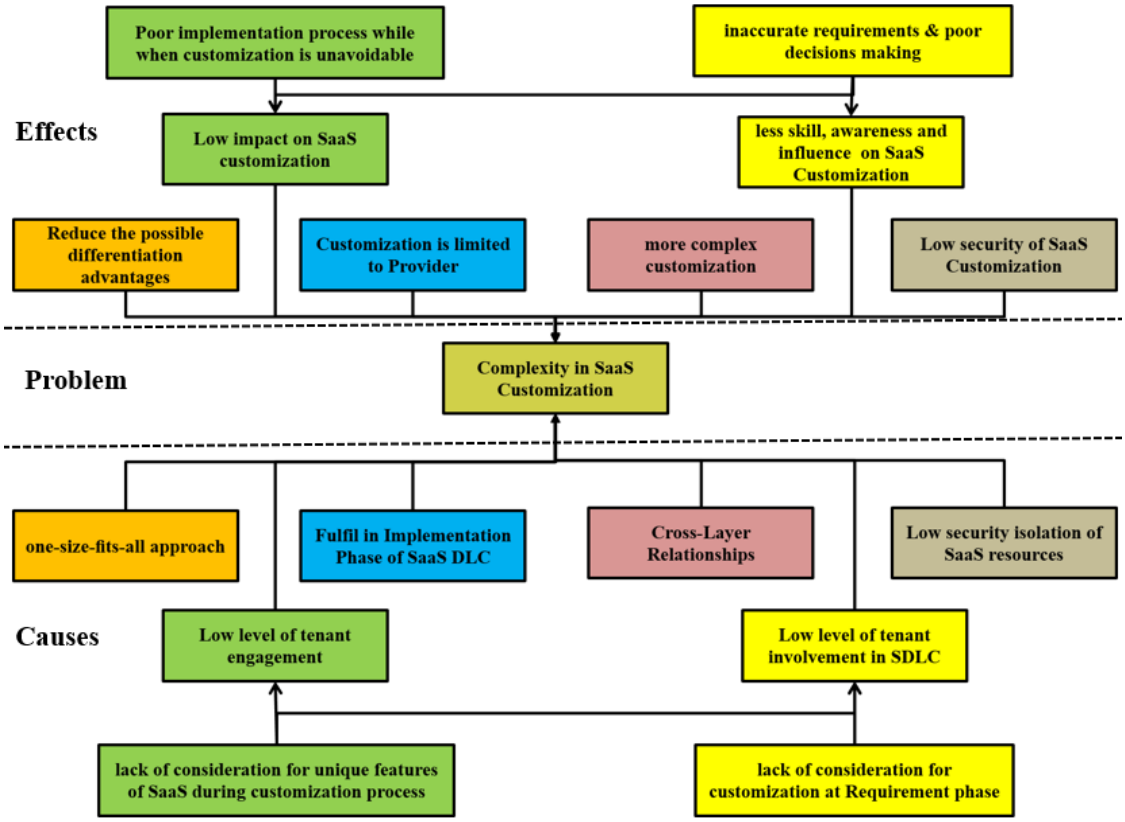

Figure 5. Problem Tree of SaaS customization

9. Objective Tree of SaaS Customization

An objective tree tries to turn the causes of a problem (the roots) into a plan of action to deal with the problem, thus removing the roots from the tree. Figure 6 shows the impact of using this method. Many problems can be turned into advantages if they are adapted wisely.

Accordingly, an initial optimal framework for SaaS customization should involve the effortless self-customization by SaaS applications' tenants, and that they apply the changes to the application automatically. This is intended to cover all possible customizations that are needed by the tenant involving the customization of an application's object that is tightly coupled with other objects. The implementation of customization has to be isolated from the application and support for concurrent.

10. Conclusion

This paper has explored some challenges related to SaaS customization, illustrated by using previous research about Software and SaaS customization, specifically those which are related to stakeholders and layers of SaaS application. Most importantly, this paper has structured the problem tree of SaaS customization which shows the main causes of SaaS customization in terms of a one-size-fits-all approach; customization implementation in SaaS DLC; cross-layer relationships; low level of tenant engagement; and low security isolation of SaaS resources. Furthermore, the objective tree method has been implemented to turn the causes of a problem (the roots) into a plan of action to deal with the problem, thus removing the roots from the tree. This is outlined in Figure 6.

A better understanding of the causes of the complexity in SaaS customization would assist practitioners and researchers with similar intentions to find a direction for future work. However, it is not the intention of this work to

claim that SaaS customization challenges, problems and objectives are limited to those mentioned in this paper.

In conclusion, we recommend a reflection on the problems and objectives that have been extracted from this study.

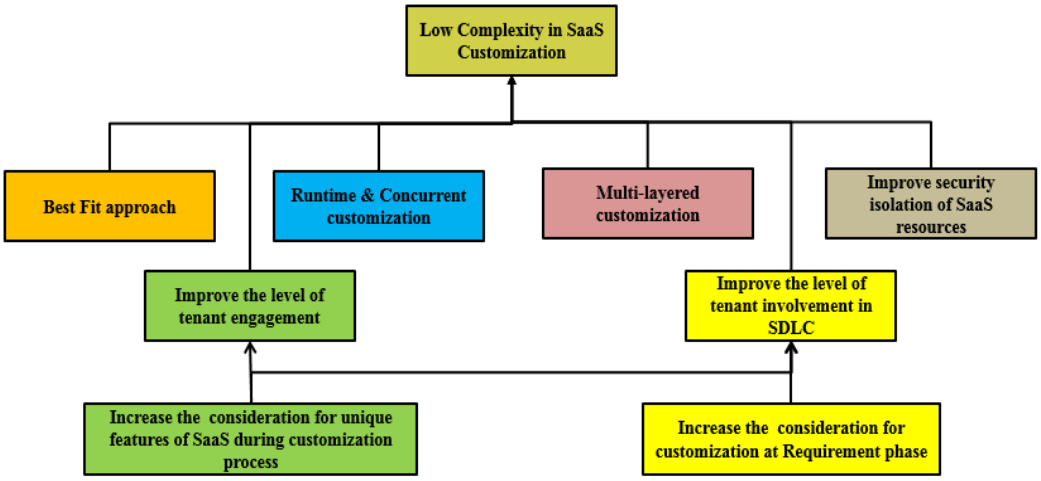

Figure 6. Objectives Tree of SaaS customization

References

[1] A. A. Shahin, A. Samir, A. Khamis, "An aspect-oriented approach for saas application customization", *arXiv preprint arXiv:1409.1656*, 2014.

[2] W.-T. Tsai, X. Sun, "Saas multi-tenant application customization", in *IEEE 7th International Symposium on Service Oriented System Engineering (SOSE)*, IEEE, , pp. 1–12, 2013.

[3] A. Q. Ali, A. B. M. Sultan, A. A. A. Ghani, H. Zulzalil, "Critical issues across SaaS development: learning from experience", *International Journal of Advances in Electronics and Computer Science*, vol. 4, no. 9, pp. 69–74, 2017.

[4] Q. Shao, "Towards effective and intelligent multi-tenancy saas", Ph.D. dissertation, Tempe, AZ, USA, 2011, aAI3434805.

[5] M. M. Al-Shardan, D. Ziani, "Configuration as a service in multi-tenant en- terprise resource planning system", *Lecture Notes on Software Engineering*, vol. 3, no. 2, p. 95, 2015.

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

[7] J. Schroeter, P. Mucha, M. Muth, K. Jugel, M. Lochau, "Dynamic configura- tion management of cloud-based applications", in *Proceedings of the 16th Interna- tional Software Product Line Conference-Volume 2*. ACM, 2012, pp. 171–178.

[8] W.-T. Tsai, Q. Shao, W. Li, "Oic: Ontology-based intelligent customization framework for saas", in *2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE, pp. 1–8, 2010.

[9] R. Mietzner, A. Metzger, F. Leymann, K. Pohl, "Variability modeling to sup- port customization and deployment of multi-tenant-aware software as a service applications", in *Proceedings of the 2009 ICSE Workshop on Principles of Engi- neering Service Oriented Systems*. IEEE Computer Society, pp. 18–25, 2009.

[10] W.-T. Tsai, Y. Huang, Q. Shao, "Easysaas: A saas development framework", in *2011 IEEE International Conference on Service-Oriented Computing and Ap- plications (SOCA)*. IEEE, pp. 1–4, 2011.

[11] M. Mijac, R. Picek, Z. Stapic, "Cloud erp system customization challenges", in *Central European Conference on Information and Intelligent Systems*. Faculty of Organization and Informatics Varazdin, p. 132, 2013.

[12] Y. Dittrich, S. Vaucouleur, S. Giff, "Erp customization as software engineering: Knowledge sharing and cooperation", *IEEE Software*, vol. 26, no. 6, pp. 41–47, Nov 2009.