Article

# An Empirical Investigation of Software Customization and Its Impact on the Quality of Software as a Service: Perspectives from Software Professionals

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

Department of Software Engineering and Information System, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang 43400, Selangor, Malaysia; abakar@upm.edu.my (A.B.M.S.); azim@upm.edu.my (A.A.A.G.); hazura@upm.edu.my (H.Z.)
* Correspondence: abdulrazzaq.alyhari@gmail.com

**Abstract:** Although customization plays a significant role in the provision of software as a service (SaaS), delivering a customizable SaaS application that reflects the tenant's specific requirements with acceptable level of quality is a challenge. Drawing on a pr-developed software customization model for SaaS quality, two fundamental objectives of this study were to determine whether different software customization approaches have direct impacts on SaaS quality, and also to assess the construct reliability and construct validity of the model. A questionnaire-based survey was used to collect data from 244 software professionals with experience in SaaS development. Structural equation modeling was employed to test the construct reliability, construct validity, and research hypotheses. The measurement model assessment suggested that the six-construct model with 39 items exhibited good construct reliability and construct validity. The findings of the structural model assessment show that all customization approaches other than the integration approach significantly influence the quality of SaaS applications. The findings also indicate that both configuration and composition approaches have positive impacts on SaaS quality, while the impacts of the other approaches are negative. The empirical assessment and evaluation of this model, which features a rich set of information, provides considerable benefits to both researchers and practitioners.

**Keywords:** customization approaches; empirical investigation; quality attributes; software as a service; structural equation modeling

check for updates

**Citation:** Qasem Ali, A.; Md Sultan, A.B.; Abd Ghani, A.A.; Zulzalil, H. An Empirical Investigation of Software Customization and Its Impact on the Quality of Software as a Service: Perspectives from Software Professionals. *Appl. Sci.* **2021**, *11*, 1677. https://doi.org/10.3390/app11041677

Received: 24 January 2021
Accepted: 18 February 2021
Published: 26 February 2021

## 1. Introduction

Customization plays a crucial part in the rendering of software as a service (SaaS) to various tenants [1–3], as the interface, business logic, and data are highly likely to vary for every tenant [4]. Thus, customization could engender performance threats which should be taken into account by the SaaS hosts [5]. Furthermore, all elements of a SaaS application (such as the design elements of the GUI, business practices, and databases) are impacted by tenant-oriented customization [6]. As such, a customizable SaaS application should consider the customization of all components of the SaaS application, including, for example, those with cross-layer relationships [5]. Additional consideration should be given to the likelihood that the software source code modifications necessary for customization will be rendered more and more complicated [1,7,8] by the necessity to separately preserve the customization code of every tenant [8].

Moreover, a tenant's requirement changes usually surface after the development of applications and services; hence, the run-time specific customization pertinent to a tenant should be provided during the same application instance [1,7,9,10], and it should not impact tenant isolation and application availability [7,10]. Usually, SaaS applications do not possess extensibility comparable to that of licensed software products [11].

Finer control towards customization would decrease application maintenance efforts [12,13], where certain maintenance jobs can be offloaded to the tenant-side (client-side).

Nevertheless, extensive customization increases the regular maintenance requirements of any SaaS application, and this may increase scalability-related and cost-efficiency-related risks [1,10]. Generally, the small investment made at the beginning and the monthly usage charges obtained from the tenants are insufficient to account for the total expenditure accrued by complex customization. Hence, SaaS vendors must remain vigilant to ensure they understand their customization expertise [12,14] and gauge any customization's effect on the essential features of SaaS [1,7,15,16]. These aspects must be accounted for before introducing customization requirements within a SaaS application.

Previously-conducted research offers little information regarding the impact of software customization on SaaS quality [17]. Before investigating this impact, it is crucial to take note of the category of customization. This enables determination of the consequences and risks associated with a specific change [18], where, regardless of the type of customization, there is a chance that software product quality could be affected [19,20]. For example, providing support for new requirements that are not yet covered by the SaaS application can have an effect on the software architecture [1], resulting in decreased quality of the application [21]. In the case that these new requirements are identical to requirements already provided for other tenants, it is important to update the predefined configuration options of the tenant every time a tenant's requirements change [1]. Such changes lead to ever-growing complexity, and under these circumstances, quality is likely to decline as the application grows [21,22].

The specified obstacles to and shortfalls concerning SaaS customization have already provided the necessary motivation for researchers to formulate an initial software customization model for SaaS quality based on academic-related literature [23] and opinions of academic-related experts [24]. The model comprises (1) generic approaches used for software customization along with a set of common practices pertaining to every customization approach in the context of multi-tenant SaaS, and (2) essential quality attributes concerning SaaS customization. However, the empirical evidence on the association between software customization and SaaS quality has not been reported in the model. Accordingly, the twofold intention of this study is to assess the reliability and validity of developed model in conjunction with industry experts and related professionals, and to employ empirical analysis to assess how different software customization approaches influence the quality of SaaS application.

The output of this study makes a contribution to the body of knowledge for software engineering and information systems in two ways. First, It provides a better understanding of the potential impact of different customization types on SaaS quality. Prior to any decisions about customizing a SaaS application, understanding customization's impact on the quality of SaaS will mitigate the risk of reduced quality. Second, the empirical assessment and evaluation of this model from the perspectives of software professionals , which features a rich set of information (e.g., customization types, quality attributes, and potential impacts), can be used as useful guidelines or references for both researchers and practitioners working on SaaS customization.

## 2. Related Literature

A brief review of the literature pertinent to software and SaaS customization, followed by that pertinent to software and SaaS quality, is presented in this section. It then presents empirical studies related to this study.

### *2.1. Software Customizations*

In contemporary software engineering, the field of customization is the focus of significant research interest, growing every time new software types emerge. Software customization is defined as the process of tailoring and delivering software corresponding to customers' needs and companies' processes [20,25,26]. Though several proposals address the issue of classifying distinct software customization approaches, only some are suitably generic that they can be applied to any software system. For example, Gilmore and Pine [27] have defined four approaches to software customization, namely: collaborative, adaptive, cosmetic, and transparent.

In the domain of Enterprise Resource Planning (ERP) systems, some researchers have proposed generic approaches for software customization [28–34]. Davenport [28] postulates that customization is just a choice, the configuration of modules and tables. Brehm et al. [29] present a rough classification of software customization approaches, specifically: configuration, bolt-ons, screen masks, extended reporting, workflow programming, user exits, ERP programming, interface development, and code modification. The Brehm classification is further modified by Rothenberger and Srite [30], who group customization options into three areas: configuration/selection, bolt-ons, and system change. Luo and Strong [31] summarize customization into three approaches: selection of modules, table configuration, and code modification. Haines [32] suggests three customization options, specifically configuration, extension (user exit), and modification.

The software customization approaches specified above were not discussed in the SaaS or cloud context. Only a few studies have attempted to devise classes for generic software customization pertaining to the SaaS multi-tenant scenario. For instance, Tsai and Sun [35] described three customization aspects: source code, configuration, and composition. Depending on the location of the execution or hosting of the customizations, Muller et al. [36], in their study, describe three varieties of customization pertaining to multi-tenant SaaS software: desktop integration, customization of the user-interface, and the backend. Additionally, Kabbedijk and Jansen [37] specified the varieties of customization in the tenant base. Customization was described using segment variability and tenant-oriented variability.

Although these are generic customization approaches, they are somewhat problematic due to conflicts; either the existence of approaches with the same name and different meaning(s), or approaches with different names but the same meaning(s). Accordingly, this study used the categorization of software customization approaches presented in Ali et al. [23], which appears to provide the most holistic categorization of customization options, resolving conflicts and redundancies in terms and meanings, and then been validated by Ali et al. [24]. Table 1 gives a brief overview of these approaches used in this study.

**Table 1.** Overview of customization approaches used in this study (based on [23,24]).

| Approach | Description |
|---|---|
| Configuration | Techniques and solutions that offer a predefined setting for the alteration of application functions within the pre-defined scope. |
| Composition | Techniques and solutions that bring together a distinct collection of pre-defined application components that jointly amount to a custom solution. |
| Extension | Techniques and solutions that expand the functionality of the application by inserting the custom code at pre-defined places in the application's code. |
| Integration | Techniques and solutions that implement third-party components designed to work with the application. |
| Modification | Techniques and solutions that alter the application design and other functional requirements of the application by means of alterations implemented to the source code. |

### 2.2. Software Quality

According to the ISO/IEC 9126 standard [38], software product quality can be defined as, "The totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs." The assessment of software quality (SQ) is intended to ascertain whether the required level of quality has been reached [39], and this evaluation plays a crucial role in the discipline of software engineering. Numerous SQ models and standards have been proposed for the improvement of SQ, including those proposed by McCall [40], Boehm [41], Dromey [42], and the ISO/IEC 25010 standard [38].

Although these quality models considered the most used model in software engineering, none of them account for certain quality attributes used specifically in SaaS contexts like multitenancy [43,44]. This gap provided the impetus for the transformation of SaaS quality models. Hence, several studies have paid due attention to recognizing and defining the quality models necessary for a SaaS application. For example, Khanjani et al. [44] suggest a set of 33 quality attributes and provided definitions for those quality attributes from a SaaS perspective. La and Kim [43] propose a detailed quality model to evaluate SaaS quality. Using ISO/IEC 9126, the authors determined characteristics, quality attributes, and established metrics required to gauge SaaS quality level. A structured process is suggested by [45] to produce high-quality SaaS software by taking into consideration the primary SaaS design criteria.

Albeit such studies, along with those of others (e.g., [46–48]), centered their attention on SaaS software quality models, and specified customizability as only one of the quality aspects relevant to SaaS application, they focused on SaaS quality from customization perspectives. An investigation on which quality attributes of SaaS applications are associated with software customization showed that previous studies on customization solutions for SaaS application, focused mainly on the following quality attributes: multi-tenancy, security, functionality, scalability, availability, efficiency, commonality, accessibility, usability, maintainability, interoperability, reliability, and response time [23]. In fact, according to further investigation [24], it was revealed that these attributes best represent the key quality attributes of SaaS applications that might be impacted by software customization.

### 2.3. Related Empirical Studies

There are many empirical studies that provided empirical evidence on the impact of customization over SaaS adoption in the context of SMEs [49–52]. For example, a cross-sectional field study conducted by [49] revealed that there are convincing grounds for software vendors to perform customizations to SaaS ERP systems. However, these studies did not provide any empirical evidence on the impact of software customization on software quality. Therefore, an extensive search in SpringerLink, IEEE Xplore, ScienceDirect, ACM, and Google Scholar digital libraries was conducted [17]. Though the primary objective of extensive search was not restricted to any type of software product, the overwhelming majority of the related-empirical studies were executed on Enterprise Resource Planning (ERP) software (e.g., [19,20,53,54]).

Some statistical techniques were used to empirically investigate the relationship between software customization and software quality. For example, the results of the ordinary least square regression analysis showed that database and source code customization significantly influence the ERP quality, whereas the module customization does not [19]. A correlation analysis found that there was a strong negative relationship between ERP efficiency and customization [20]. Conversely, software customization did not a significant impact on ERP usability [54]. It should be noted that the results given by some these studies referred to the customization effect without specifying the type of customization [20,54].

Nevertheless, such studies provide empirical insight into the effect of software customization on software quality, there has been no evidence specifically addressing the effect of customization on SaaS quality, where more attention is being given to customization. Furthermore, all of these studies had subjectively assessed the effect of software customization on the quality attributes which supports the fact that most software quality attributes are conceptually subjective and are experienced by users when the system is in operation [55–57].

## 3. Research Model and Hypotheses

The initial development of a software customization model for SaaS quality was performed using systematic mapping [23]. Then, the model was refined iteratively based on the results of two rounds of content validity determination using SaaS expertise derived from academia, and internal consistency reliability testing conducted with software engineering researchers [24]. Drawing on Ali et al. [24], the model consists of 6 constructs and 44 items. These 6 constructs and their associated items are: (1) configuration (8 items), (2) composition (4 items), (3) extension (6 items), (4) integration (8 items), (5) modification (5 items), and (6) SaaS quality (13 Items). A detailed description of model used in this study can be found in the Supplementary Materials.

At the same time, there is a dearth of empirical evidence about the impact of software customization specifically on SaaS quality; thus, the research hypotheses were formulated based on (1) the literature review presented in this study, (2) the results of pre-conducted systematic mapping study [23], which showed the importance of considering SaaS quality when proposing customization solutions, and (3) reviews and studies on other software-related areas such as software changes. In this study, customization is conceptualized as changes made to SaaS application in order to address specific tenant requirements. Thus, customization could be considered by selection within predefined settings (configuration), combining application components (composition), inserting custom code (extension), implementing third-party components (integration), or alteration of the source code of the application (modification).

Taking into account the fact that tenant requirements evolve, often after the development of applications and services, run-time customization capability must be provided for the tenant [1,9,10]. Moreover, customization capability must address the challenge of changing user requirements by facilitating ongoing updates of software to meet and satisfy user requirements over a system's lifetime [21,22]. Software customization caters explicitly to the propensity of the tenants' base to evolve to higher sophistication levels which demand powerful features to fulfil their requirements.

Conversely, it is found that most changes are requested to enhance the quality and functionality of systems [19,21,58]. For instance, the database customization has a positive impact on the software functionality, whereas the source code customization has a positive impact on the usability and maintainability [19]. Generally, any customization is expected to affect the quality of any software product [19]. Accordingly, two-tailed hypotheses have been formulated as follows:

**Hypothesis 1 (H1).** *Software configuration significantly influences the quality of SaaS application.*

**Hypothesis 2 (H2).** *Software composition significantly influences the quality of SaaS application.*

**Hypothesis 3 (H3).** *Software extension significantly influences the quality of SaaS application.*

**Hypothesis 4 (H4).** *Software integration significantly influences the quality of SaaS application.*

**Hypothesis 5 (H5).** *Software modification significantly influences the quality of SaaS application.*

Based on the literature review and the facts discussed above, and the findings of our pr-conducted studies [23,24], a research model was developed, depicted in Figure 1.

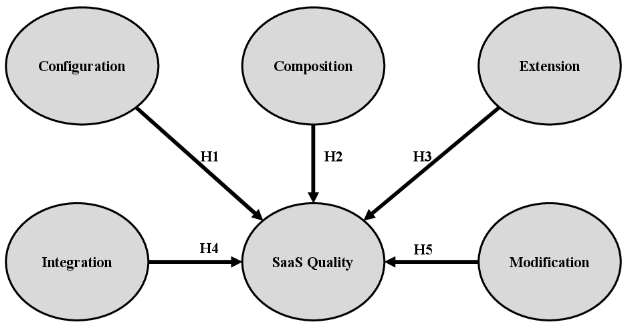

**Figure 1.** Research model.

## 4. Methodology

Our methodology includes sampling and data collection, Instrument development, and data analysis. This section details the methods and analysis used to achieve the objectives of this study.

### *4.1. Sample and Data Collection*

As no data was available for the population of SaaS-specialized software engineers within the cost and time parameters of the study, a population of other SaaS implementation team members (SaaS providers, developers, and operators) was chosen using the purposive method for sampling. In this method, researchers focus on respondents who are interested in the topic under investigation [59]. Therefore, in this research, software and cloud professionals who have been involved in any step of the SaaS development life cycle (e.g., requirements analysis, design, development, testing, maintenance, and support) were selected as respondents.

An online questionnaire was designed to gather survey data, using a Google form for ease of response. The professional network LinkedIn was selected as the best medium to recruit software engineering and cloud computing professionals to participate in the survey [60]. Recruitment was achieved in two ways, a general announcement calling for participation, and direct messaging of selected individuals. Specifically, the survey was announced in many related professional groups, and more than 2000 selected individuals were contacted directly via LinkedIn Messaging. In addition to the online questionnaire, manual data collection was also conducted by distributing questionnaires to potential respondents at software companies in Malaysia. Specifically, the survey was distributed at Technology Park Malaysia, CyberJaya, and the Malaysian Technology Development Corporation (UPM-MTDC) where most software companies are located. Moreover, the survey was manually distributed in some events, to individuals with similar interests, organized by meetup groups (meetup.com (accessed on 21 February 2021) is a platform used to connect people with similar interests) in Malaysia (e.g., Google Cloud, DevOps Malaysia, and the Elastic Kuala Lumpur User Group).

A total of 251 questionnaires were returned (161 were from the online survey and the remaining from the portion distributed by hand), of which 7 were spoilt (respondents did not complete the survey), leaving 244 for the analysis. A summary of the demographic characteristics of the 244 respondents is provided in Table 2. As the present study made use of the Structural Equation Modeling (SEM) with analysis of Moment Structure (AMOS V. 24) technique, the collected data considered an appropriate sample size for SEM in order to achieve reliable and valid analysis results [61–63].

**Table 2.** Demographic profiles of the respondents (n = 244).

| Demographic Variable | Category | Frequency (n) | Percentage (%) |
|---|---|---|---|
| Gender | Male | 197 | 80.7 |
| | Female | 47 | 19.3 |
| Age | 21–30 | 113 | 46.3 |
| | 31–40 | 97 | 39.8 |
| | Over 40 | 34 | 13.9 |
| Job Title | Software engineer | 92 | 37.7 |
| | Software developer/programmer | 62 | 25.4 |
| | Software quality engineer/Software Tester | 22 | 9 |
| | Software Consultant | 26 | 10.7 |
| | Others | 42 | 17.2 |
| Years of experience | 1–2 | 44 | 18 |
| | 3–4 | 45 | 18.4 |
| | >4 | 155 | 63.5 |
| Company Level | Multinational company | 159 | 65.2 |
| | National company | 79 | 32.4 |
| | Don't know | 6 | 2.5 |
| Involvement in SaaS development | Yes | 217 | 88.9 |
| | No | 0 | 0 |
| | Somewhat | 27 | 11.1 |
| SaaS application | Customer-Relationship Management (CRM) | 77 | 31.6 |
| | Enterprise Resource Planning (ERP) | 47 | 19.3 |
| | Document Management System (DMS) | 34 | 13.9 |
| | Other | 27 | 11.1 |
| | Many | 59 | 24.2 |

### *4.2. Instrument Development*

This study adopted instruments that were developed based on the results of a systematic mapping study [23] and iterative validity study [24]. All reflective constructs in this study use multi-items measured using 5-point Likert scales anchored at 1 for “strongly disagree” to 5 for “strongly agree”. The instrument was pilot tested by submitting it to an internal consistency reliability test conducted by software-engineer researchers from 4 Malaysian universities [24]. A summary of survey measurement items used in this study are given in the Supplementary Materials.

### *4.3. Measurement Model Assessment*

Confirmatory Factor Analysis (CFA) was used to examine the reliability and validity of the measurement model, however, determining outliers, checking for normality, and testing model fit should be considered before.

#### 4.3.1. Outliers

The Mahalanobis distance is a widely known measure for pinpointing outliers in a multivariate data set [64]. Potential outliers can be detected by checking for high Mahalanobis distance, where both p1 and p2 are zero. Although outlier removal was performed accordingly for 14 cases, the omission of these outliers had non-significant influence on the SEM results. On this basis, further analyses in the study were conducted with the outliers included.

#### 4.3.2. Normality

Data normality testing was used to examine the sample distribution shape. According to [65], skewness should be within the range −2 to +2, and kurtosis should range from −7 to +7 [66]. The value of Skewness and Kurtosis higher than 3 and 10 respectively indicating a problem [67]. Following the establishment of normal data distribution, where all items fell below the threshold point, inferential statistical tests were conducted.

### 4.3.3. Model Fit Criteria

In accordance with Hair et al. [63], at least one fitness index from each category (absolute, incremental, and parsimonious) of model fit indices has to be met. As literature stresses the importance of selecting fit indices unaffected by sample size [68,69], three such indices were selected for use in this study [69]: (1) Chi-Square/df (CMIN/DF) $< 3.0$, (2) comparative fit index (CFI) $> 0.90$, and (3) root mean square error of approximation (RMSEA) $< 0.08$.

In addition to the fitness indices, the threshold for item factor loading must be satisfied. In this study, items with factor loadings below 0.50 or greater than 1.0, and those with a negative value, were deleted. If all items meet these criteria but the fitness index is not satisfied, items associated with highest modification index were sought [70,71]. It was possible to either delete one of the factor loadings, or constrain both factor loadings [72,73]. The deletion process was preferred to improve construct definition with respect to the high factor loadings and uncorrelated items [73]. If the items in one of the construct are consistently below the three items, the researcher can apply an alternative process, the constraint procedure [73,74]. Figure 2 summarizes the steps for evaluating the model fit in this study.

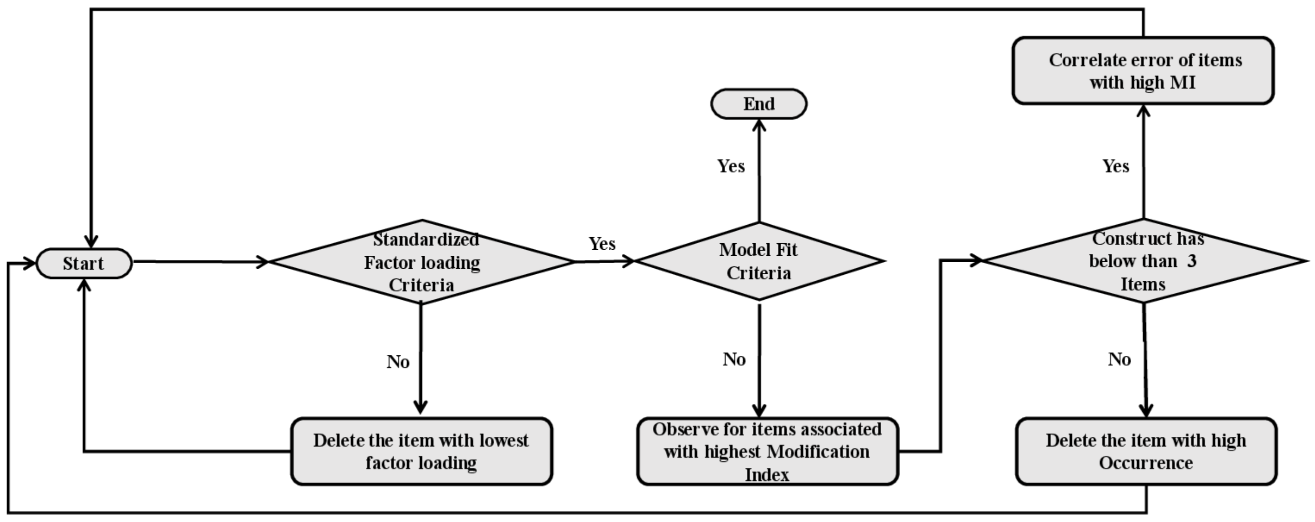
Start
Standardized Factor loading Criteria
Yes
No
Delete the item with lowest factor loading
Model Fit Criteria
Yes
End
No
Observe for items associated with highest Modification Index
Construct has below than 3 Items
Yes
Correlate error of items with high MI
No
Delete the item with high Occurrence

**Figure 2.** Model fit evaluation steps in this study.

### 4.3.4. Construct Reliability

As this study employs SEM, which prioritizes the indicators based on the individual indicator's loadings, composite reliability (CR) was more appropriate measure than Cronbach's Alpha [75]. An a-value above 0.7 indicates high internal consistency reliability [76,77]. In addition, any a-value above 0.70 is considered reliable and acceptable [78]. Thus, the construct reliability for each construct calculated as shown in Formula (1) [63]:

$$CR = \frac{\sum \lambda^2}{\sum \lambda^2 + \sum \delta} \tag{1}$$

where $\sum \lambda^2$ is the total of all squared standardized factor loadings and $\sum \delta$ represents the total of the measurement error.

### 4.3.5. Convergent Validity

Convergent validity refers to a set of indicators that presume to measure a construct [67]. Two values, average variance extracted (AVE) and CR are commonly used to assess the convergent validity of the research instrument [79]. AVE, the average amount of the squared standardized factor loadings, is calculated using Formula (2) [63]:

$$AVE = \frac{\sum \lambda^2}{n} \tag{2}$$

where $\sum \lambda^2$s is the total of all squared standardized factor loadings and $n$ represents the number of items. Values greater than 0.5 are acceptable and show how well latent variables have convergent validity. Overall, convergent validity is considered to exist if AVE for each individual construct is greater than 0.5 and CR is higher than 0.7 [63].

#### 4.3.6. Discriminant Validity

Discriminant validity refers to the extent to which a construct is truly distinct from other constructs [63]. To assess the construct discriminant validity, two common criteria, AVE analysis and the correlation coefficient are applied as follows: (1) the square root of AVE for each construct should be greater than the correlations of the construct with any of the other constructs [79], and (2) for the correlation coefficient, discriminate validity is satisfied if the correlation coefficient for any two constructs is less than 0.90 [63,79]. It is worth noting that the second criteria can be used to check whether multicollinearity among variables or constructs is said to exist [70,79]. Therefore, confirming the discriminate validity of a model remedies the multicollinearity problem as well.

### *4.4. Structural Model Assessment*

A slope test is performed to determine the individual impact of an exogenous variable on the endogenous variable. The p-value and the critical ratio (C.R.) should be less than 0.05 and more than 1.96, respectively, to establish that the exogenous variables have a significance effect on the endogenous variables [63].

The coefficient of determination ($R^2$) is considered to represents the variance in the dependents, as explained by the predictor variables. $R^2$ values be classified such that values above 0.67 are considered high, values in the range 0.33 to 0.67 are considered moderate, values in the range 0.19 to 0.33 are considered weak, and any value less than 0.19 is deemed unacceptable [80].

## **5. Analyses and Results**

This section presents and details the findings and the results of the analysis of this study. The results of the measurement model assessment and structural model assessment are presented in the following subsections.

### *5.1. Results of Measurement Model Assessment*

Two main procedures were utilized as a part of the CFA to test the measurement model. The first approach is testing model fit criteria and second, assessing the reliability and validity of the model.

#### 5.1.1. Model Fit Criteria

The fulfillment of at least one fitness index from absolute, incremental, and parsimonious fit categories and factor loading of more than 0.5 were applied to test overall measurement model fit. The model evaluation revealed that the measurement model with Relative Chi-Sq ($\leq$3) = 1.975; CFI ($\geq$0.9) = 0.867; RMSEA ($\leq$0.08) = 0.063 did not meet the model fit requirements, because none of the Incremental fit indices met the requirements. Moreover, all factor loadings were more than 0.5 except for Ext6 (factor loading = 0.07), Int8 (factor loading = 0.36), and Con5 (factor loading = 0.37).

Accordingly, the deletion should be made on Ext6, Int8, and Con5, respectively. After each deletion, we ran a new measurement model. It revealed that the modified measurement model; with Relative Chi-Sq ($\leq$3) = 1.972; CFI ($\geq$0.9) = 0.883; RMSEA ($\leq$0.08) = 0.063, still did not meet the model fit requirements. As all factor loadings exceed 0.5, we look for items associated with highest modification index. Items Ext3 (occurring 10 times) and Ext4 (occurring 11 times) show the highest MI (29.766). The occurrence for item Ext4 is the highest; thus, we delete item Ext4. The results of this step showed the modified measurement model with some improvement on fit indices (Chi-Sq ($\leq$3) = 1.878; CFI ($\geq$0.9) = 0.896; RMSEA ($\leq$0.08) = 0.063) but still does not meet the model fit requirements.

Consequently, a second round of modification index checks is conducted. Items QA10 (occurring 8 times) and QA 11 (occurring 11 times) show the highest MI (27.136). The occurrence for item QA11 is the highest; thus, we delete item QA11. As a result, the respective measurement model fit to the data showed improvement where Relative Chi-Sq ($\leq$3) = 1.814; CFI ($\geq$0.9) = 0.905; IFI ($\geq$0.9) = 0.906; RMSEA ($\leq$0.08) = 0.058. These results of the model fit indices and factor loadings indicate that the model fits well, meeting all the required criteria. Figures present all rounds of measurement model modifications through CFA are available in the Supplementary Materials.

#### 5.1.2. Construct Reliability

The construct reliability (CR) of each construct, calculated based on formula 1, should be greater than 0.7 to confirm the construct reliability. In this study, the CR was found to range from 0.803 to 0.942, as summarized in Table 3. The computed values of CR for the configuration (n = 7), composition (n = 4), extension (n = 4), integration (n = 7), modification (n = 5), and SaaS quality (n = 12) constructs were 0.892, 0.806, 0.839, 0.904, 0.929, and 0.937, respectively. Overall, the CR values for all six constructs were greater than 0.7 and subsequently all these constructs may be considered reliable.

#### 5.1.3. Convergent Validity

AVE and CR are used to calculate construct convergent validity. AVE is calculated using formula 2. The AVE results were found to range from 0.506 to 0.564 as summarized in Table 3. The computed values of AVE for the configuration (CR = 0.892), composition (CR = 0.806), extension (CR = 0.839), integration (CR = 0.904), modification (CR = 0.929), and SaaS quality (CR = 0.937) constructs were 0.547, 0.512, 0.578, 0.578, 0.724, and 0.555, respectively. The AVE results and the CR of each show that all the AVE values are above the minimum cut-off point (>0.5). Moreover, all constructs meet the criteria of CR > AVE, indicating the construct convergent validity.

**Table 3.** Factor loading, composite reliability (CR), average variance extracted (AVE), and model fit indices' statistics.

| Construct | Item | Factor Loading | CR | AVE | Construct | Item | Factor Loading | CR | AVE |
|---|---|---|---|---|---|---|---|---|---|
| SaaS Quality | QA 1 | 0.75 | 0.937 | 0.555 | Integration | Int 1 | 0.64 | 0.904 | 0.578 |
| | QA 2 | 0.77 | | | | Int 2 | 0.90 | | |
| | QA 3 | 0.69 | | | | Int 3 | 0.86 | | |
| | QA 4 | 0.73 | | | | Int 4 | 0.78 | | |
| | QA 5 | 0.78 | | | | Int 5 | 0.77 | | |
| | QA 6 | 0.76 | | | | Int 6 | 0.67 | | |
| | QA 7 | 0.79 | | | | Int 7 | 0.66 | | |
| | QA 8 | 0.78 | | | Extension | Ext 1 | 0.93 | 0.839 | 0.578 |
| | QA 9 | 0.77 | | | | Ext 2 | 0.87 | | |
| | QA 10 | 0.65 | | | | Ext 3 | 0.53 | | |
| | QA 12 | 0.70 | | | | Ext 5 | 0.64 | | |
| | QA 13 | 0.76 | | | Modification | Mod 1 | 0.80 | 0.929 | 0.724 |
| Configuration | Con 1 | 0.80 | 0.892 | 0.547 | | Mod 2 | 0.83 | | |
| | Con 2 | 0.85 | | | | Mod 3 | 0.88 | | |
| | Con 3 | 0.76 | | | | Mod 4 | 0.92 | | |
| | Con 4 | 0.74 | | | | Mod 5 | 0.82 | | |
| | Con 6 | 0.52 | | | Composition | Com 1 | 0.66 | 0.806 | 0.512 |
| | Con 7 | 0.69 | | | | Com 2 | 0.64 | | |
| | Con 8 | 0.77 | | | | Com 3 | 0.80 | | |
| | | | | | | Com 4 | 0.75 | | |
| **Model Fit Indices** | | | Relative Chi-Sq ($\leq$3) = 1.814; CFI ($\geq$0.9) = 0.905; IFI ($\geq$0.9) = 0.906; RMSEA ($\leq$0.08) = 0.058 | | | | | | |

#### 5.1.4. Discriminant Validity

Discriminant validity of the overall measurement model was established by examining the square root of AVE for each individual construct verse with its corresponding correlations, and checking correlation coefficient between any two constructs. Based on Table 4, the correlations between constructs ranged from 0.000 to 0.57 and the correlations among the constructs were less than the the square root of AVE, indicating each construct to be truly distinct from the others.

**Table 4.** Discriminant validity and correlations.

| | SaaS Quality | Configuration | Integration | Extension | Modification | Composition |
|---|---|---|---|---|---|---|
| SaaS Quality | **0.745** | | | | | |
| Configuration | 0.39 | **0.740** | | | | |
| Integration | 0.11 | 0.02 | **0.760** | | | |
| Extension | −0.15 | 0.06 | 0.11 | **0.76** | | |
| Modification | −0.03 | 0.57 | −0.02 | 0.000 | **0.851** | |
| Composition | 0.31 | 0.03 | 0.11 | 0.15 | 0.03 | **0.716** |

Note: Diagonal elements (bold figures) are the square roots of AVE values, and off-diagonal elements are the correlations among the constructs.

### *5.2. Results of Structural Model Assessment*

The results of the structural model assessment revealed that the coefficient of determination ($R^2$) value was 0.37 of the endogenous latent variables in the structural model, which is more than the minimum acceptable level of $R^2$ (See Figure 3). Thus, it can be concluded that 37% of the variation in the SaaS quality was explained by the variations in the configuration, integration, extension, modification, and composition variables at a 95% confidence level.

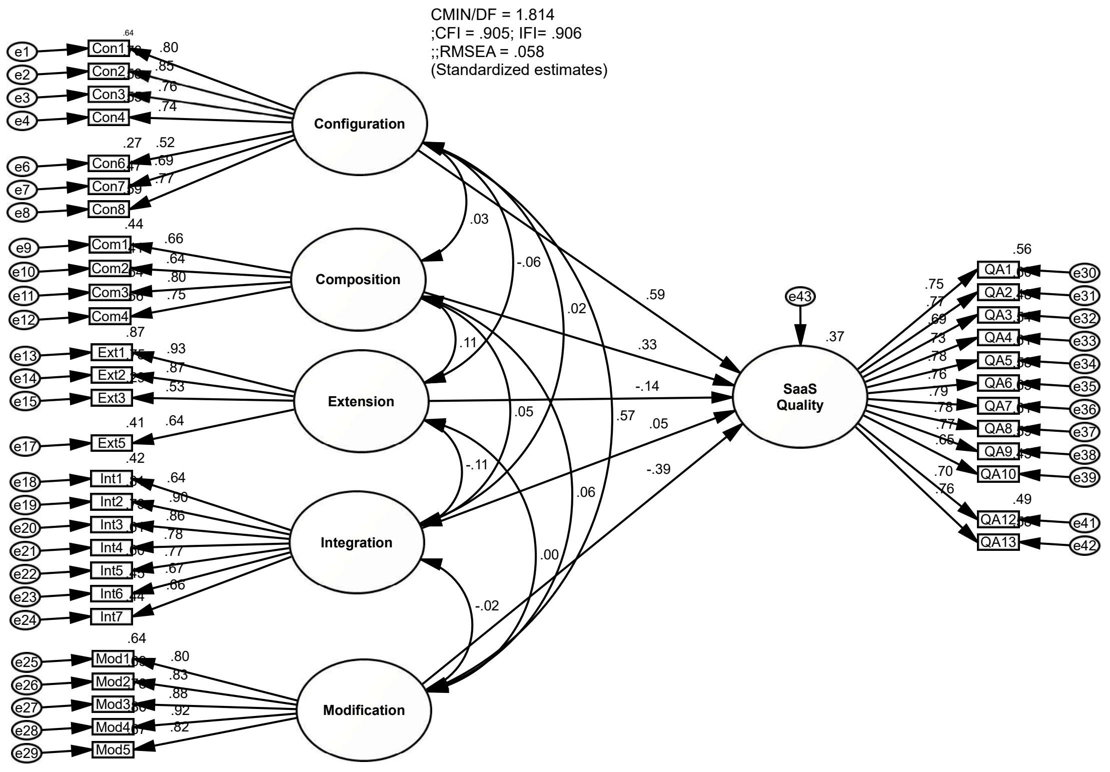

**Figure 3.** Results of structural model assessment.

Testing hypotheses using maximum likelihood estimation (MLE) technique revealed that four of the five hypotheses, H1, H2, H3, and H5 are statistically supported. The results show that standardized estimates for these hypotheses 0.589, 0.33, −0.144, and −0.386 respectively) are statistically significant, whereas one hypothesis, H4, with standardized estimate (0.052) was found to be statistically insignificant. This hypothesis, statistically, should be rejected. The overall results of hypothesis testing are presented in Table 5.

Subsequently, we test the structural model once more, by omitting the integration construct. It revealed the same results of $R^2$ and P values for H1, H2, H3, and H5 as that of the initial structural model. Furthermore, the Beta values implied that configuration, modification, composition, and extension, respectively, have the most impact on SaaS quality. Figures present detailed structural model with/out integration construct are available in the Supplementary Materials.

**Table 5.** Hypothesis testing results.

| Hypothesis | B | SE | Beta | C.R. | P |
|---|---|---|---|---|---|
| **H1:** Configuration===>SaaS Quality | 0.419 | 0.061 | 0.589 | 6.838 | ** 0.000 |
| **H2:** Composition===>SaaS Quality | 0.301 | 0.064 | 0.330 | 4.692 | ** 0.000 |
| **H3:** Extension===>SaaS Quality | −0.108 | 0.045 | −0.144 | −2.385 | * 0.017 |
| **H4:** Integration===>SaaS Quality | 0.047 | 0.053 | 0.0520 | 0.882 | 0.378 |
| **H5:** Modification===>SaaS Quality | −0.328 | 0.067 | −0.386 | −4.923 | ** 0.000 |

* Significant at the 0.05 level. ** Significant at the 0.001 level.

## 6. Discussion

To achieve the objectives of this study, SEM statistical technique were used. After running pooled CFA, the final measurement model resulted in Relative Chi-Sq ($\leq$3) = 1.814, CFI ($\geq$0.9) = 0.905, IFI ($\geq$0.9) = 0.906, RMSEA ($\leq$0.08) = 0.058; this indicates that the data fit the model. Moreover, all six constructs of the model have exhibited the required construct reliability, convergent validity, and discriminant validity. This indicates that the six-constructs model with 39 items had a good fit and exhibits the good construct reliability, convergent validity, and discriminant validity of the model prior to testing the research hypotheses.

This model was developed based on the results of a systematic mapping study [23] and then soundly validated via the iterative method in [24]. Therefore,it was not necessary to apply EFA in this study. Nevertheless, we assume that some readers and researchers prefer to perform EFA before CFA , because CFA does not show how the items are grouped and loaded in each construct [69,81]. Therefore, a detailed result for EFA is reported in the Supplementary Materials. The EFA results confirmed the measurement model structure used in this study.

Finally, a set of hypotheses was formulated to establish a relationship between each customization approach and SaaS quality. The hypotheses were empirically tested to determine if customization has a significance impact on SaaS quality. The findings of the structural model assessment show that all customization approaches, bar the integration approach, significantly influence the quality of SaaS application. Moreover, it revealed that the impact of configuration and composition approaches on SaaS quality is positive, while the impact of other approaches is negative.

The positive effects of the composition and configuration approaches could be attributed to the fact that numerous SaaS vendors offer built-in tools and support interfaces to allow such approaches to be implemented, as such facilities can boost application usage without changing the source code. In such customization approaches, the efforts needed to make changes will improve the maintainability and usability of the application by enabling the customers to performed some customization and maintenance jobs using the built-in tools and interfaces provided in these approaches. Since the maintainability and usability of SaaS application could be improved, the improvement in overall quality of SaaS could be considered as well.

conversely, the negative influence of the modification and extension approaches on SaaS quality is attributed to the significant work necessary to implement the modification or extension changes to the application code. In the SaaS application environment, the operator must change the application code and perform a redeployment for each tenant. Therefore, these approaches are bound to have a negative impact as successful multi-tenancy must have a stable code foundation.

A general observation of these findings is that all customization approaches that have a statistically significant effect on SaaS quality belong to the internal change category, whereas the integration approach belongs to the external change category as discussed in [82]. Considering this and the fact that all the data collected in this study are the respondents' own perceptions and interpretations, the impact of the integration approach was found statistically insignificant. Although the impact of the integration approach (external change) on SaaS quality was statistically not supported in this study, an initiative to empirically investigate this relation in independent study would be interesting.

## *6.1. Theoretical Implications*

As the model used in this study is the only model relating software customization to SaaS quality, this study sought to prove and improve its soundness by empirically assessing and evaluating its construct validity and reliability. Additionally, the establishment of empirical evidence on the effects of different software customization types on SaaS quality has not previously been examined, this study sought to fill the research gap which constitutes a true contribution to the literature for prospective researchers with similar research intentions.

Moreover, the improved model provides a broad classification of the multi-tenant SaaS customization space for academic research from different aspects (e.g, customization types, quality attributes, and potential impacts). Anticipating further research attention in this field, the improved model will also be an effective and useful tool to classify the current and forthcoming solutions for software and SaaS customization.

## *6.2. Practical Implications*

The findings of this study can improve the ability of SaaS providers to make better decisions on tenant customization requirements. For example, SaaS providers can outweigh between subscription fees received from tenants and the effect of implementing the new requirement on SaaS quality. If the implementation of the new feature has a negative impact on SaaS quality, then they lower its priority and try to communicate with the customers to inform them that this is not the direction they would like to take. Otherwise, the findings this study could help SaaS providers persuade customers to consider a more favorable approach (e.g., configuration approach) over other approaches (e.g., modification approach) that could cost more in the long run, especially their arguments would be backed with numbers and statistics.

Furthermore, rather than doing assumptions, the empirical results of this study will help SaaS developers and architects to understand the relationship between software customization and SaaS quality. Moreover, instead of assessing all the customization approaches and all quality attributes, the developed model mainly focuses on the most generic types of customization approaches and most-related quality attributes of SaaS applications, which make it easier for SaaS implementation teams to conduct control assessment. Being able to accurately identify the customization type that will affect SaaS quality will aid developers to make the decision on suitable customization type and practice without degrading the quality of the SaaS application.

## *6.3. Threats to Validity*

The four key threats to validity of this study and the mitigation steps espoused are discussed below.

1. Threats to external validity
   External validity is related to the ability of the researchers to generalize the results of academic research to industrial practice [83]. For SEM, the sample size of respondents in this study should be considered sufficient [61]. However, this number of respondents can lead to this study being criticized for having limited generalizability. In response, it is worth noting that respondents have a diversity of industrial backgrounds and experiences from different countries and were approached online and face-to-face.
2. Threats to internal validity
   One of the limitations of this study was the choice of customization approaches (independent variables) selected for an analysis of their association and impact on SaaS quality. There may be other customization approaches which have an impact on SaaS quality but we only considered those which were the results of previous studies including systematic mapping study [23] and academic-related experts' opinions [24]. Additionally, other factors that influence software and SaaS quality (e.g., software architecture and requirements volatility) were not considered in this study because the focus of this study was only on software customization approaches affecting the SaaS quality.
3. Threats to conclusion validity
   Another limitation of this study is that both the dependent and independent variables are measured from the same source, which may lead to incorrect conclusions about the relationship between variables. This is seen as a potential source of common method bias (CMB) which threatens the validity of the conclusions [84]. To mitigate this threat, the common latent factor (CLF) technique [85] was employed to quantitatively ascertain the presence of potential instrument bias issues. The standardized regression weights were calculated with and without the CLF, after which the differences were calculated. Subsequently, all the differences higher than 20% were identified and used to discover construct-associations affected by CMB problems. The results indicated that this model lacked bias. A thorough explanation of the CLF procedure can be seen in the Supplementary Materials.
4. Threats to construct validity
   However, the model used in this study was iteratively validated [24], which may offer some certainty of the construct validity, more testing had been conducted to evaluate its construct validity and reliability with a larger sample based on the industry environment. As using multiple types of data analysis can ensure construct validity, we reported both factor analysis types (long associated with construct validity). Another issue related to construct validity could be the gender bias, as the participants of this study are mostly male. However, the number of women software engineers and developers is gradually increased, software engineering domain is still dominated by men [86] where women represent only 21% of the total software development workforce [87]. Considering this fact and the fact that proportion of women's presence in our sample, 19.3% are women and the remaining 80.7% are men, considers higher than their presence in other software engineering research studies (e.g, [86,88]), the gender distribution of our sample considers acceptable and free from gender bias.

## 7. Conclusions and Future Research

The purpose of this study was to empirically assess the impact of software customization on the quality of software as a service. This involved the evaluation of the construct reliability and construct validity of software customization model for SaaS quality. A questionnaire-based survey was used to collect data from 244 software professionals with experience in the SaaS development life cycle. After rounds of modification through CFA, the required model fit, reliability, and validity were achieved. A significant relationship was found between all customization approaches and SaaS quality, but none between the integration approach and SaaS quality. Our ongoing research is to examine the effectiveness and usefulness of the presented model based on some success factors of software projects (e.g., reducing the time, cost, and uncertainty of customization assessment on SaaS quality).

Whilst carrying out this study, several ideas and enhancements were considered as possibilities for future work. First, this study was restricted to customization types, practices, and quality attributes of SaaS applications that are the results of systematic mapping studies and experts' opinions [23,24]. However, this work does not intend to claim that these are the only customization types, practices of SaaS customization, and SaaS quality attributes associated with customization. Future research could also be conducted to expand the model to include many different perspectives. Second, this study empirically reported the impact of each customization approach on SaaS quality, however; it did not consider the impact of each customization approach on each quality attribute of SaaS applications. Future investigations may explore these relationships—for example, the impact of composition approach on the SaaS security attribute (as one of the indicators of SaaS quality). Furthermore, though the impact of the integration approach (external change) on SaaS quality was statistically not supported in this study, an initiative to empirically investigate the relationships between integration approach and SaaS quality independently would be interesting. Third, an addendum to this study considering different methodologies (e.g., historical analysis of software changes or case studies) and replication are recommended to build upon and refine the reported findings of this study.

**Supplementary Materials:** The following are available online at https://www.mdpi.com/2076-3417/11/4/1677/s1, A supplementary document that supports and enhances the analysis and findings of this study is is available online at https://tinyurl.com/OnlineSupSaaSCustQ (accessed on 22 February 2021).

**Author Contributions:** Conceptualization, A.Q.A.; formal analysis, A.Q.A.; funding acquisition, A.B.M.S.; investigation, A.Q.A.; methodology, A.Q.A.; Resources, A.B.M.S.; supervision, A.B.M.S., A.A.A.G. and H.Z.; writing—original draft, A.Q.A.; writing—review and editing, A.B.M.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This study was funded by Universiti Putra Malaysia.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Walraven, S.; Landuyt, D.V.; Truyen, E.; Handekyn, K.; Joosen, W. Efficient customization of multi-tenant Software-as-a-Service applications with service lines. *J. Syst. Softw.* **2014**, *91*, 48–62. [CrossRef]
2. Araujo, V.M.; Vazquez, J.A. Business and technical requirements of Software-as-a-Service: Implications in portuguese enterprise business context. *Int. J. Found. Comput. Sci. Technol.* **2013**, *3*. [CrossRef]
3. Ali, A.Q.; Sultan, A.B.M.; Ghani, A.A.A.; Zulzalil, H. Customization of Software as a Service Application: Problems and Objectives. *J. Comput. Sci. Comput. Math.* **2018**, *8*, 27–32. [CrossRef]
4. Tsai, W.T.; Zhong, P.; Chen, Y. Tenant-centric Sub-Tenancy Architecture in Software-as-a-Service. *CAAI Trans. Intell. Technol.* **2016**, *1*, 150–161. [CrossRef]
5. Al-Shardan, M.M.; Ziani, D. Configuration as a service in multi-tenant enterprise resource planning system. *Lect. Notes Softw. Eng.* **2015**, *3*, 95. [CrossRef]
6. Tsai, W.; Shao, Q.; Li, W. OIC: Ontology-based intelligent customization framework for SaaS. In Proceedings of the 2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA), Perth, Australia, 13–15 December 2010; pp. 1–8. [CrossRef]
7. Ali, A.Q.; Sultan, A.B.M.; Ghani, A.A.A.; Zulzalil, H. The Five Ws Taxonomy on Customization of Software as a Service Applications. *J. Comput. Sci. Comput. Math.* **2018**, *8*, 43–48. [CrossRef]
8. Guo, C.J.; Sun, W.; Jiang, Z.B.; Huang, Y.; Gao, B.; Wang, Z.H. Study of Software as a Service Support Platform for Small and Medium Businesses. In *New Frontiers in Information and Software as Services*; Agrawal, D., Candan, K.S., Li, W.S., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 1–30. [CrossRef]
9. Shahin, A.A. Variability modeling for customizable SaaS applications. *Int. J. Comput. Sci. Inf. Technol.* **2014**, *6*, 39–49. [CrossRef]

10. Van Landuyt, D.; Walraven, S.; Joosen, W. Variability Middleware for Multi-tenant SaaS Applications: A Research Roadmap for Service Lines. In Proceedings of the 19th International Conference on Software Product Line, Nashville, TN, USA, 20–24 July 2015; ACM: New York, NY, USA, 2015; SPLC '15, pp. 211–215. [CrossRef]
11. Yang, S.; Yoo, B.; Jahng, J. Does the SaaS model really increase customer benefits. *Asia Pac. J. Inf. Syst.* **2010**, *20*, 87–101.
12. Samir, A.; Darwish, N.R. Reusability Quality Attributes and Metrics of SaaS from Perspective of Business and Provider. *Int. J. Comput. Sci. Inf. Secur.* **2016**, *14*, 295–312.
13. Xin, M.; Levina, N. Software-as-a Service Model: Elaborating Client-Side Adoption Factors. In Proceedings of the 29th International Conference on Information Systems, Paris, France, 14–17 December 2008; [CrossRef]
14. Sun, W.; Zhang, X.; Guo, C.J.; Sun, P.; Su, H. Software as a Service: Configuration and Customization Perspectives. In Proceedings of the 2008 IEEE Congress on Services Part II (Services-2 2008), Beijing, China, 23–26 September 2008; pp. 18–25. [CrossRef]
15. Joha, A.; Janssen, M. Design choices underlying the software as a service (saas) business model from the user perspective: Exploring the fourth wave of outsourcing. *J. Univ. Comput. Sci.* **2012**, *18*, [CrossRef]
16. Espadas, J.; Molina, A.; Jiménez, G.; Molina, M.; Ramírez, R.; Concha, D. A tenant-based resource allocation model for scaling Software-as-a-Service applications over cloud computing infrastructures. *Future Gener. Comput. Syst.* **2013**, *29*, 273–286. [CrossRef]
17. Ali, A.Q.; Sultan, A.B.M.; Ghani, A.A.A.; Zulzalil, H. Empirical studies on the impact of software customization on quality attributes: a systematic review. *J. Theor. Appl. Inf. Technol.* **2019**, *97*, 1747–1763.
18. Chaumun, M.; Kabaili, H.; Keller, R.K.; Lustman, F. A change impact model for changeability assessment in object-oriented software systems. *Sci. Comput. Program.* **2002**, *45*, 155–174. [CrossRef]
19. Parthasarathy, S.; Sharma, S. Impact of customization over software quality in ERP projects: An empirical study. *Softw. Qual. J.* **2017**, *25*, 581–598. [CrossRef]
20. Parthasarathy, S.; Sharma, S. Efficiency analysis of ERP packages-A customization perspective. *Comput. Ind.* **2016**, *82*, 19–27. [CrossRef]
21. Williams, B.J.; Carver, J.C. Characterizing software architecture changes: A systematic review. *Inf. Softw. Technol.* **2010**, *52*, 31–51. [CrossRef]
22. Lehman, M. Feedback, evolution and software technology. In Proceedings of the 10th International Software Process Workshop, Dijon, France, 17–19 June 1996; pp. 101–103. [CrossRef]
23. Ali, A.Q.; Sultan, A.B.M.; Abd Ghani, A.A.; Zulzalil, H. A Systematic Mapping Study on the Customization Solutions of Software as a Service Applications. *IEEE Access* **2019**, *7*. [CrossRef]
24. Ali, A.Q.; Md Sultan, A.B.; Abd Ghani, A.A.; Zulzalil, H. Development of a valid and reliable software customization model for SaaS quality through iterative method: perspectives from academia. *PeerJ Comput. Sci.* **2020**, *6*, e294. [CrossRef]
25. Franke, N.; Keinz, P.; Steger, C.J. Testing the value of customization: When do customers really prefer products tailored to their preferences? *J. Mark.* **2009**, *73*, 103–121. [CrossRef]
26. Etame, F.; Atsa, R. Survey on ERP's customization-driven requirements engineering. In *Applied Informatics*; Springer: Berlin/Heidelberg, Germany, 2018; Volume 5, p. 2, [CrossRef]
27. Gilmore, J.H.; Pine, B.J. The four faces of mass customization. *Harv. Bus. Rev.* **1997**, *75*, 91–102.
28. Davenport, T.H. Putting the Enterprise into the Enterprise System. *Harv. Bus. Rev.* **1998**, *76*, 121–131. [PubMed]
29. Brehm, L.; Heinzl, A.; Markus, M.L. Tailoring ERP systems: A spectrum of choices and their implications. In Proceedings of the 34th Annual Hawaii International Conference on System Sciences, Maui, HI, USA, 3–6 January 2001; p. 9, [CrossRef]
30. Rothenberger, M.A.; Srite, M. An Investigation of Customization in ERP System Implementations. *IEEE Trans. Eng. Manag.* **2009**, *56*, 663–676. [CrossRef]
31. Luo, W.; Strong, D.M. A framework for evaluating ERP implementation choices. *IEEE Trans. Eng. Manag.* **2004**, *51*, 322–333. [CrossRef]
32. Haines, M.N. Understanding Enterprise System Customization: An Exploration of Implementation Realities and the Key Influence Factors. *Inf. Syst. Manag.* **2009**, *26*, 182–198. [CrossRef]
33. Kurbel, K.E. *Enterprise Resource Planning and Supply Chain Management: Functions, Business Processes and Software for Manufacturing Companies*; Springer: Berlin/Heidelberg, Germany, 2013; [CrossRef]
34. Munkelt, T.; Völker, S. ERP systems: Aspects of selection, implementation and sustainable operations. *Int. J. Inf. Syst. Proj. Manag.* **2013**, *1*, 25–39. [CrossRef]
35. Tsai, W.; Sun, X. SaaS Multi-tenant Application Customization. In Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, San Francisco, CA, USA, 25–28 March 2013; pp. 1–12. [CrossRef]
36. Müller, J.; Krüger, J.; Enderlein, S.; Helmich, M.; Zeier, A. Customizing Enterprise Software as a Service Applications: Back-End Extension in a Multi-tenancy Environment. In *Enterprise Information Systems*; Filipe, J., Cordeiro, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 66–77. [CrossRef]
37. Kabbedijk, J.; Jansen, S. Variability in Multi-tenant Environments: Architectural Design Patterns from Industry. Advances in Conceptual Modeling. In *Recent Developments and New Directions*; De Troyer, O., Bauzer Medeiros, C., Billen, R., Hallot, P., Simitsis, A., Van Mingroot, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 151–160. [CrossRef]
38. ISO. 9126 Software product evaluation—Quality characteristics and guidelines for their use. *ISO/IEC Stand.* **2001**, *9126*. Available online: https://www.researchgate.net/figure/ISO-9126-Software-Product-Evaluation-Quality-Characteristics-and-Guidelines-for-their_fig3_228723822 (accessed on 22 February 2021).

39. ISO. Iec25010: 2011 systems and software engineering—Systems and software quality requirements and evaluation (square)—System and software quality models. *Int. Organ. Stand.* **2011**, *34*, 2910.
40. McCall, J.A. *Factors in Software Quality*; US Rome Air Development Center Reports; General Electric Co.: Sunnyvale, CA, USA, 1977.
41. Boehm, B. *Characteristics of Software Quality*; Notas de Matematica; North-Holland Publishing Company: Amsterdam, The Netherlands, 1978.
42. Dromey, R.G. Cornering the chimera [software quality]. *IEEE Softw.* **1996**, *13*, 33–43. [CrossRef]
43. La, H.J.; Kim, S.D. A Systematic Process for Developing High Quality SaaS Cloud Services. In *Cloud Computing*; Jaatun, M.G., Zhao, G., Rong, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 278–289. [CrossRef]
44. Khanjani, A.; Rahman, W.N.W.A.; Ghani, A.A.A.; Sultan, A.B.M. SaaS quality of service attributes. *J. Appl. Sci.* **2014**, *14*, 3613–3619. [CrossRef]
45. Lee, J.Y.; Lee, J.W.; Cheun, D.W.; Kim, S.D. A Quality Model for Evaluating Software-as-a-Service in Cloud Computing. In Proceedings of the 2009 Seventh ACIS International Conference on Software Engineering Research, Management and Applications, Haikou, China, 2–4 December 2009; pp. 261–266. [CrossRef]
46. Nadanam, P.; Rajmohan, R. QoS evaluation for web services in cloud computing. In Proceedings of the 2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12), Coimbatore, India, 26–28 July 2012; pp. 1–8. [CrossRef]
47. Cancian, M.H.; Hauck, J.C.R.; von Wangenheim, C.G.; Rabelo, R.J. Discovering Software Process and Product Quality Criteria in Software as a Service. In *Product-Focused Software Process Improvement*; Ali Babar, M., Vierimaa, M., Oivo, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 234–247. [CrossRef]
48. Duarte Filho, N.F.; de Souza Bermejo, P.H.; Zambalde, A.L.; de Barros, U.S. Saasquality-a method for quality evaluation of software as a service (saas). *Int. J. Comput. Sci. Inf. Technol.* **2013**, *5*, 101. [CrossRef]
49. Seethamraju, R. Adoption of software as a service (SaaS) enterprise resource planning (ERP) systems in small and medium sized enterprises (SMEs). *Inf. Syst. Front.* **2015**, *17*, 475–492. [CrossRef]
50. Ahn, B.; Ahn, H. Factors Affecting Intention to Adopt Cloud-Based ERP from a Comprehensive Approach. *Sustainability* **2020**, *12*, 6426. [CrossRef]
51. Valdebenito, J.; Quelopana, A. Conceptual Model for Software as a Service (SaaS) Enterprise Resource Planning (ERP) Systems Adoption in Small and Medium Sized Enterprises (SMEs) Using the Technology-Organization-Environment (TOE) Framework. In *International Conference on Information Technology & Systems*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 143–152.
52. Faasen, J.; Seymour, L.F.; Schuler, J. SaaS ERP adoption intent: Explaining the South African SME perspective. In *Enterprise Information Systems of the Future*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 35–47.
53. Light, B. The maintenance implications of the customization of ERP software. *J. Softw. Maint. Evol. Res. Pract.* **2001**, *13*, 415–429. [CrossRef]
54. Ng, C.S.P. A case study on the impact of customization, fitness, and operational characteristics on enterprise-wide system success, user satisfaction, and system use. *J. Glob. Inf. Manag.* **2013**, *21*, 19–41. [CrossRef]
55. Jung, H.W. Validating the external quality subcharacteristics of software products according to ISO/IEC 9126. *Comput. Stand. Interfaces* **2007**, *29*, 653–661. [CrossRef]
56. Miguel, J.P.; Mauricio, D.; Rodríguez, G. A review of software quality models for the evaluation of software products. *Int. J. Softw. Eng. Appl.* **2014**, *5*, 31. [CrossRef]
57. Moses, J. Should we try to measure software quality attributes directly? *Softw. Qual. J.* **2009**, *17*, 203–213. [CrossRef]
58. Mohagheghi, P.; Conradi, R. An empirical study of software change: Origin, acceptance rate, and functionality vs. quality attributes. In Proceedings of the 2004 International Symposium on Empirical Software Engineering, Redondo Beach, CA, USA, 19–20 August 2004; ISESE'04, pp. 7–16. [CrossRef]
59. Tongco, M.D.C. Purposive sampling as a tool for informant selection. *Ethnobot. Res. Appl.* **2007**, *5*, 147–158. [CrossRef]
60. Unkelos-Shpigel, N.; Sherman, S.; Hadar, I. Finding the missing link to industry: LinkedIn professional groups as facilitators of empirical research. In Proceedings of the 2015 IEEE/ACM 3rd International Workshop on Conducting Empirical Studies in Industry, Florence, Italy, 18 May 2015; pp. 43–46. [CrossRef]
61. Lei, P.W.; Wu, Q. Introduction to structural equation modeling: Issues and practical considerations. *Educ. Meas. Issues Pract.* **2007**, *26*, 33–43. [CrossRef]
62. Hoogland, J.J.; Boomsma, A. Robustness studies in covariance structure modeling: An overview and a meta-analysis. *Sociol. Methods Res.* **1998**, *26*, 329–367. [CrossRef]
63. Hair, J.F.; Black, W.C.; Babin, B.J.; Anderson, R.E.; Tatham, R.L. *Multivariate Data Analysis*; Pearson Education Limited: London, UK, 2013.
64. Ben-Gal, I. *Outlier Detection [w:] Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers, red. O. Maimon, L. Rokach*; Kluwer Academic Publishers: Boston, MA, USA, 2005.
65. Tabachnick, B.G.; Fidell, L.S. Multivariate analysis of variance and covariance. *Using Multivar. Stat.* **2007**, *3*, 402–407.
66. Byrne, B.M. *Structural Equation Modeling with AMOS: Basic Concepts, Applications, and Programming (Multivariate Applications Series)*; Taylor & Francis Group: New York, NY, USA, 2010; Volume 396, p. 7384.
67. Kline, R.B. *Principles and Practice of Structural Equation Modeling*; Guilford Publications: New York, NY, USA, 2015.

68. Sharma, S.; Mukherjee, S.; Kumar, A.; Dillon, W.R. A simulation study to investigate the use of cutoff values for assessing model fit in covariance structure models. *J. Bus. Res.* **2005**, *58*, 935–943. [CrossRef]
69. Schniederjans, D.G.; Hales, D.N. Cloud computing and its impact on economic and environmental performance: A transaction cost economics perspective. *Decis. Support Syst.* **2016**, *86*, 73–82. [CrossRef]
70. Awang, Z. *Structural Equation Modeling Using AMOS Graphic*; Penerbit Universiti Teknologi MARA: Shah Alam, Malaysia, 2012.
71. Costello, A.B.; Osborne, J. Best practices in exploratory factor analysis: Four recommendations for getting the most from your analysis. *Pract. Assess. Res. Eval.* **2005**, *10*, 7.
72. Afthanorhan, W.; Ahmad, S.; Mamat, I. Pooled Confirmatory Factor Analysis (PCFA) using structural equation modeling on volunteerism program: A step by step approach. *Int. J. Asian Soc. Sci.* **2014**, *4*, 642–653.
73. Awang, Z.; Afthanorhan, A.; Mohamad, M.; Asri, M. An evaluation of measurement model for medical tourism research: The confirmatory factor analysis approach. *Int. J. Tour. Policy* **2015**, *6*, 29–45. [CrossRef]
74. Samah, B. *Enhancing Extension Education Research Using Structural Equation Modelling*; Universiti Putra Malaysia Press: Serdang, Malaysia, 2016.
75. Werts, C.E.; Linn, R.L.; Jöreskog, K.G. Intraclass reliability estimates: Testing structural assumptions. *Educ. Psychol. Meas.* **1974**, *34*, 25–33. [CrossRef]
76. Sekaran, U.; Bougie, R. *Research Methods for Business: A Skill Building Approach*; John Wiley & Sons: Hoboken, NJ, USA, 2016.
77. Hinton, P.R.; McMurray, I.; Brownlow, C. *SPSS Explained*; Routledge: London, UK, 2014.
78. Nunnally, J.C. *Psychometric Theory 3E*; Tata McGraw-Hill Education: New Delhi, India, 1994.
79. Fornell, C.; Larcker, D.F. Evaluating structural equation models with unobservable variables and measurement error. *J. Mark. Res.* **1981**, *18*, 39–50. [CrossRef]
80. Falk, R.F.; Miller, N.B. *A Primer for Soft Modeling.*; University of Akron Press: Akron, OH, USA, 1992.
81. Kelloway, E.K. Structural equation modelling in perspective. *J. Organ. Behav.* **1995**, *16*, 215–224. [CrossRef]
82. Parhizkar, M. Impact Analysis of Enterprise Resource Planning Post-Implementation Modifications. Ph.D. Thesis, University of London, London, UK, 2016.
83. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering*; Springer: Berlin/Heidelberg, Germany, 2012. [CrossRef]
84. Podsakoff, P.M.; MacKenzie, S.B.; Lee, J.Y.; Podsakoff, N.P. Common method biases in behavioral research: A critical review of the literature and recommended remedies. *J. Appl. Psychol.* **2003**, *88*, 879. [CrossRef] [PubMed]
85. Matemba, E.D.; Li, G.; Maiseli, B.J. Consumers' Stickiness to Mobile Payment Applications: An Empirical Study of WeChat Wallet. *J. Database Manag.* **2018**, *29*, 43–66. [CrossRef]
86. Russo, D.; Stol, K.J. Gender differences in personality traits of software engineers. *IEEE Trans. Softw. Eng.* **2020**. [CrossRef]
87. Wang, Y.; Redmiles, D. Implicit gender biases in professional software development: An empirical study. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS), Montreal, QC, Canada, 25–31 May 2019; pp. 1–10. [CrossRef]
88. Qiu, H.S.; Nolte, A.; Brown, A.; Serebrenik, A.; Vasilescu, B. Going farther together: The impact of social capital on sustained participation in open source. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), Montreal, QC, Canada, 25–31 May 2019; pp. 688–699. [CrossRef]