systems

MDPI

Article

Ramping Up Customer-Centric Modular Design Projects: Mobile App Development for Pandemic Relief

Khaled Medini [1,*], Stefan Wiesner [2], Milad Poursoltan [3] and David Romero [4]

[1] Mines Saint-Etienne, Université Clermont Auvergne, CNRS, UMR 6158 LIMOS, Henri Fayol Institute, F-42023 Saint-Etienne, France
[2] BIBA—Bremer Institut für Produktion und Logistik GmbH, University of Bremen, Hochschulring 20, 28359 Bremen, Germany; wie@biba.uni-bremen.de
[3] University of Bordeaux, CNRS, IMS, UMR 5218, 33405 Talence, France; milad.poursoltan@u-bordeaux.fr
[4] Tecnológico de Monterrey, Mexico City 14380, Mexico; david.romero.diaz@gmail.com
* Correspondence: khaled.medini@emse.fr; Tel.: +33-4-77-42-93-17

Received: 30 September 2020; Accepted: 30 October 2020; Published: 3 November 2020

Abstract: Today's fast-growing demands at the global level for mobile applications (apps) cause customers to call for the customization of their apps to fit their individualized needs and business realms. Customization is a challenge for apps-development companies when they want to satisfy their numerous users in a crowded competitive market. Moreover, pursuing customization involves additional challenges when ramping up app development projects in order to meet demands at a larger scale. To address this challenge, we proposed a framework to systematize and support mobile apps' development consistently with a customer-driven approach and modular design philosophy. From a practical point of view, the proposed framework integrates quality function deployment (QFD), axiomatic design (AD) principles, and practices from the ITIL (Information Technology Infrastructure Library) framework. The framework supports a systematic process for translating customer needs into design parameters as well as supporting prioritization of ITIL practices for further development. The effectiveness of the framework was explored in practice through a case study about an app supporting relief in the 2020 COVID-19 pandemic, as well as a survey among potential users. The assessment of the framework indicated an average score ranging between 3.58 and 3.92 in a five-point Likert scale for all of the items used in the survey.

Keywords: quality function deployment; design projects; ITIL v4 foundation; customer-centric; mobile apps' development; pandemic

1. Introduction

Mobile apps receive much attention at a global level due to their various features and opportunities that can be offered for individuals and businesses. This type of software comes into sight in almost all spheres of our digital life such as entertainment, health and fitness, travel and hospitality, e-commerce and retail, and education and learning. This has led to the emergence of a new research field in software engineering that aims at developing various approaches and methodologies to help app development companies achieve their goals. These development companies provide mobile apps for specific vertical or horizontal markets. The vertical markets offer apps that typically target a group of users. By contrast, the horizontal markets include apps aiming at a large number of users with different levels of knowledge and skills, e.g., for web browsers. In both kinds of markets, development companies seek out methods and technologies to meet customers' demands in the short term with low budget and effort.

Offering differentiated functionalities to different users can be considered as a success in the mobile app market [1]. According to a study undertaken by Sitecore and Vanson Bourne with the contribution of more than 4500 customers and marketing decision makers from 11 countries, around a third of respondents said they expect their mobile apps to be more "personalized" (customized) [2]. Generally, developers tend to standardize as many components of their apps as feasible to enhance them for reusability, maintainability, reliability, and reduced development costs. However, with a lower customization level, standardization is likely to lead to a lower customer satisfaction level. Consequently, a trade-off should be identified in order to balance "standardization" vs. "customization".

Mass customization (MC) is a well-known concept that allows meeting individual customers' needs with near mass-production efficiency [3]. In contrast to the one-size-fits-all approach, customization aims at providing distinctive experiences for users regarding their specific expectations. Early and diverse users' involvement is required to create a good user experience [4]. In this regard, a great emphasis has been laid on user-centered design approaches as well as on agile software development processes.

Schnall et al. [5] have used an information systems research (ISR) framework as a user-centered model that allows end-user feedback and expert opinion to be considered in an app development process. Furthermore, Lopes et al. [6] have deployed user-centered techniques for apps' development to apply personas and scenarios tools for customers' requirements' elicitation as well as interaction models for identifying and locating usability issues in the app design phase.

Several customized agile practices have been introduced for mobile app development such as (1) "Mobile-D" [7], (2) Mobile Application Software development—based on Agile Methodology [8], and (3) Hybrid Methodology Design Process [9].

Increasing use of user-centered and agile approaches have created new challenges associated with time and effort estimation for large app-developer companies [4]. To meet both, evolving and diverse customer needs and company needs in terms of profitability, standardization, and customization should be balanced in a way that allows achieving economies of scale and economies of scope. This is quite consistent with the app developer's needs to reduce effort and shorten lead time, and also improve customer satisfaction. Hence, MC is gaining great interest in software development [10–12].

The current research was concerned with two challenges, namely: (1) How to balance customization (customer driven) and standardization (for efficiency and efficacy), and (2) how to ramp up customer-centric design (addressing large base of customers and generating economies of scale). To contribute toward meeting these challenges, the current paper aimed to reinforce customer-centric modular app development through capturing the benefits of coupling Quality Function Deployment (QFD), independence axiom from axiomatic design, and ITIL v4 SVS (Information Technology Infrastructure Library, version 4, Service Value System) practices. A Design Science Research (DSR) approach was adopted to address this problem and is described in Section 2.

In line with the research methodology, challenges and solution approaches for app development are identified in Sections 3 and 4, respectively. Three propositions were derived and are listed in Section 4, laying the foundation for the proposed framework. The propositions are as follows: (1) QFD provides an appropriate framework for developing customer-centric solutions through systematizing the progressive translation of customer needs into technical solutions, (2) independence axiom from the Axiomatic Design (AD) theory supports the improvement of system design toward modular architecture, and (3) ITIL v4 SVS provides a standard and flexible framework, supporting the design and management of value-driven systems of products and services. More specifically, practices are flexible and can be customized to different contexts.

Subsequently, the proposed framework is outlined in Section 5, which aimed at supporting user-centered modular mobile apps' development. This framework allows capturing the benefits of coupling QFD, independence axiom from axiomatic design, and ITIL v4 SVS practices to reinforce customer-centric modular app development. Section 6 presents an illustrative case study. Concluding remarks are summarized in Section 7.

2. Research Methodology

A DSR approach was adapted for developing the research work presented in this paper. DSR supports a synthesis-evaluation process of possible solutions to a given problem [13]. Accordingly, a Design Science Research Process (DSRP) model was selected to operationalize DSR in the context of the current research. The DSRP model was developed consistently with DSR theory and based on existing research, particularly in the Information Systems (IS) domain [14].

Figure 1 shows the DSRP steps. It is noteworthy that, in comparison to the work of Pfeffers et al. [14], the "communication" step was omitted as it refers basically to the scholarly and professional publication and it has no direct impact on the currently presented research development.

Problem identification and motivation and the objectives of a solution are developed in Sections 3 and 4 focusing, respectively, on the main challenges for developing mobile apps as well as potential and well-tested solution approaches. Section 5 supports design and development by outlining a framework coupling the ITIL v4 Service Value System and QFD. Section 6 reports on a case study highlighting practicality and improvement perspectives, which supports both demonstration and evaluation. Section 7 supports particularly the evaluation based on an assessment of the proposed framework by academics and practitioners.

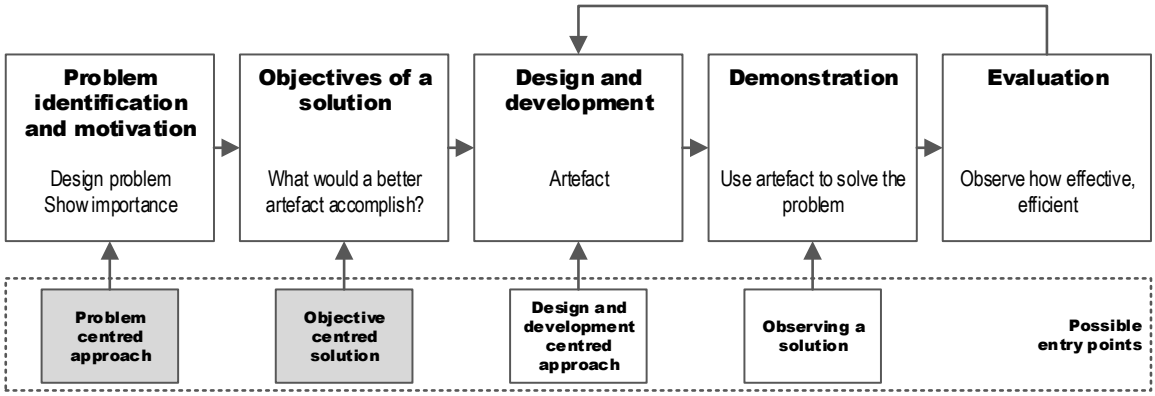

Figure 1. Design Science Research Process (DSRP) model (adapted from [13]).

3. Mobile Apps' Ramp Up: Development Challenges

Software development projects are challenged by changing customer requirements and market conditions. Therefore, time to market and time to volume became major concerns for companies regardless of their specializations. These factors add to the complexity of the shift from design to a stable operation phase generally known as "ramp up" [15–17]. Ramp up, as a value-creation phase located between development and stable production/operation, is critical for the successful introduction of products or services into the market. The proper management of ramp up projects is likely to ensure a timely introduction into the market at reasonable costs and with a satisfactory quality of the proposed solution [18,19]. Yet, meeting these objectives is challenged by the increasing complexity emerging from customers' requests for service and product customization. Consequently, the development process complexity and time need to be reduced in different ways. Particularly, successful development practices and processes need to be "standardized" and "expanded" in a way to benefit newly launched projects from past experiences.

Mobile apps are confronted with the challenge of balancing standardization vs. customization, which is heightened by the high process complexity. For instance, in terms of technical development, apps are classified into three different categories:

(1) Native apps developed for a specific operating system,
(2) Mobile web apps that refer to web applications running in mobile devices, and
(3) Hybrid apps, which evolved as a mixture of both native and mobile web apps.

Across these categories, customers are always looking for highly customized apps that meet their own expectations. Furthermore, cost, time, and quality are the primary focus of app development companies.

Although there are fundamental similarities between app and desktop software developments, there are still many differences between their features and challenges [20]. For example, it has been pointed out that a prominent challenge for developing native apps is to consider multiple mobile platforms [21]. The variety of mobile operating systems or platforms on different devices leads often to one app working on specific mobile devices [22]. This is owed to differences in user interfaces (UI), user eXperience (UX), human–computer interaction (HCI) standards, and supported development frameworks or tools, among different mobile platforms [21]. Also, frequent updating of operating systems can result in higher costs of maintenance, tests, pushing out updates, etc. Conversely, users seek high performance of functionalities regardless of the operating system (OS) or platform that they use. Subsequently, choosing the right type of mobile app for development is a crucial action to deal with this challenge.

In software development, requirements are usually classified as functional and nonfunctional requirements. Nonfunctional requirements are concerned with the identification of intended system behavior, while functional requirements focus on "what" the software does [23]. Meeting these requirements is closely related to consistent quality management and control efforts. Despite being time consuming and costly, software testing is an essential process that helps to detect failures in system quality and acceptability. Different levels of testing criteria have been identified and applied such as acceptance testing, system testing, integration testing, module testing, and unit testing [24]. Due to app development peculiarities, such as device availability and mobile network operators, more types of quality and performance tests are needed, such as interrupt testing, location testing, and outdated software testing. Moreover, in today's fast-growing demands for mobile apps, users expect quick and frequent app releases with new features, with no defects. However, testing can show only the presence of failures rather than their absence. This implies that the development process of mobile apps should be logically structured in order to limit failure.

4. Solution Approaches for Mass Customizing Apps

Based on the discussion in the previous section, moving toward a combination of "from-the-shelf" and "tailor-made" items is likely to mitigate complexity and improve efficiency. Furthermore, standardizing development processes and expanding best practices are likely to systematize the development process and foster service and products' ramp up. This section deals specifically with an integrated mass-customization (MC) and ITIL approach as a relevant solution for customer-centric modular design of apps (see Figure 2).

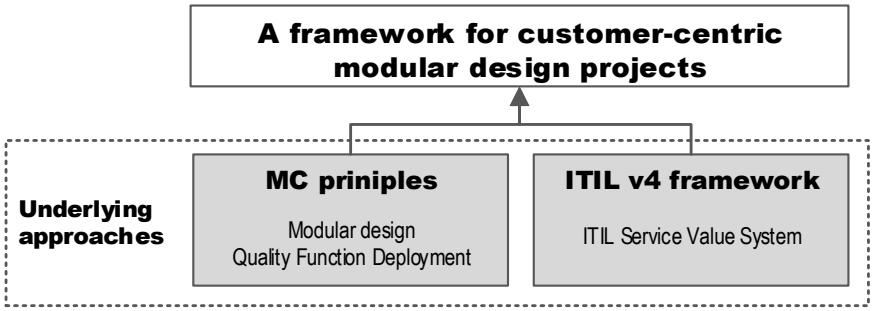

Figure 2. Framework building and development rationale.

MC in software development focuses on efficiency to build a wide variety of software modules or products by managing their communities and differences. Software MC is seen as a gainful strategic model for expanding new market segments and responding to customer demands. MC concepts' adoption in software development can be witnessed in various research works throughout the last few

years, e.g., [10–12] Kang and Namkung [12] have studied the relationship between "personalization" and "perceived benefit" for food-service mobile app.

Common strategies derived from software MC adopted to app development include the following [11]:

- Pure app standardization—development based on a one-size-fits-all principle.
- Segmented app, where multiple clusters of users are served using multiple app variants (derived app versions for customer-specific requirements).
- Customized app standardization—option of selecting the user's own set of components and functionality in the range of available components.
- Tailored app customization—modification of a standard design for a particular group of users.
- Pure app customization—from scratch development and implementation are realized as per user specification.

Identification of requirements is a key step to meet customer needs through an MC approach. To this end, QFD has been deployed as one of the well-known tools for capturing customer needs usually not formally represented and translating them into functional requirements [25]. QFD has been reported in many studies in software development to provide a full understanding of customer heterogeneous needs and transform them into engineering characteristics [26]. Unlike classical QFD that generally addresses physical characteristics, software QFD focuses on behavioral characteristics. Moreover, the production result is valued not for what it is, but for what it does [26]. Subsequently, the following proposition was established.

Proposition 1. *QFD provides an appropriate framework for developing customer-centric solutions through systematizing the progressive translation of customer needs into technical solutions.*

To consistently develop reusable components and enhance standardization while meeting customer requirements, "modularity" proves to be one of the most promising approaches. The International Organization for Standardization (ISO) and the International Electro-technical Commission (ISO/IEC 25010), define modularity as a degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components [27]. Modularity as a means to decrease complexity is characterized by two basic general features, "cohesion" and "coupling" [28]. Higher cohesion indicates lower complexity, while, contrarily, high coupling refers to higher complexity. Proactively implementing modularity during app development requires methodological guidance. Principles of axiomatic design, called "axioms", exhibit a potential for this challenge. Axiomatic design is a systematic model providing a general design framework spanning the following engineering sequence: (1) Customer needs (CNs), (2) functional requirements (FRs), (3) design parameters (DPs), and (4) process variables (PVs) [29]. The expected output of the software design is to satisfy FRs and functional constraints (Cs) [30]. Axiomatic design relies on two axioms, (1) independence, referring to maintaining FRs, and (2) information, referring to minimizing information content in the design. A design in which each DP covers a single FR is seen to be perfectly fulfilling the "independence" axiom. The following proposition can then be derived.

Proposition 2. *Independence axiom from the axiomatic design theory supports the improvement of system design toward modular architecture.*

While approaches and theories such as QFD and axiomatic design have the potential to support requirements of engineering and design of customer-centric modular solutions, "ramping up" the development projects of these solutions still requires standard and flexible frameworks [15]. In the field of Information Technology (IT) service domain, a major well-known framework is ITIL [31]. While ITIL applies to IT services in the first place, the business context led to an increasing reinforcement of the role of value and digital transformation in managing products and services. This is particularly

obvious in the fourth edition of ITIL bringing service management practices in a broader context of customer experience and value streams [31]. In this sense, value co-creation with customers, partners, and suppliers is one of the key concepts addressed by the ITIL framework. Accordingly, the four dimensions that should be holistically considered for proper management of products and services are (1) reorganization and people, (2) information and technology, (3) partners and suppliers, and (4) value stream and processes. How these dimensions interact and how value is created are covered by the ITIL v4 Service Value System (SVS) (see Figure 3). ITIL guiding principles represent a set of recommendations for promoting collaboration and cooperation. Governance represents the means of how the organization is directed. Practices refer to organizational resources supporting the achievement of the objectives. These are sets of generic and adaptable recommendations, which may apply to different organizations regardless of their sizes and sectors. The service value chain is at the core of the ITIL v4 SVS and represents interconnected activities to deliver a valuable product or service. These activities are triggered by an opportunity or a demand to create "value". Not less importantly, continual improvement supports increased performance and that stakeholders' expectations are met. Based on the aforementioned characteristics of ITIL v4 SVS, the following proposition can be inferred.

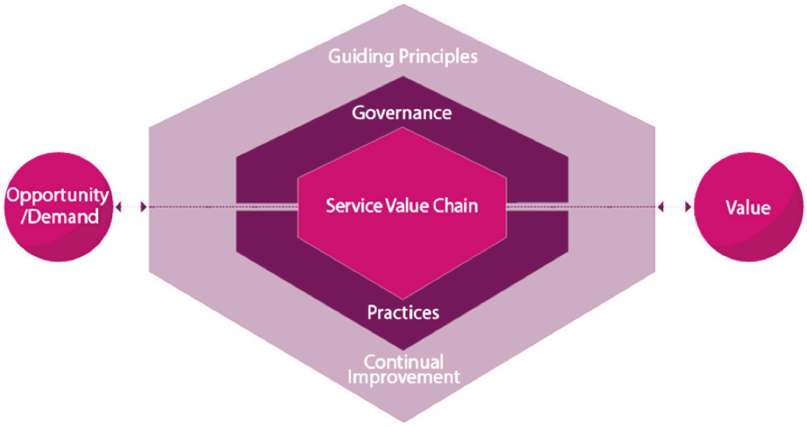

Figure 3. ITIL v4 Service Value System (SVS) [30].

Proposition 3. *ITIL v4 SVS provides a standard and flexible framework, supporting the design and management of value-driven systems of products and services. More specifically, practices are flexible and can be customized to different contexts.*

5. Customer-Driven Modular App Development

This section elaborates on a framework capturing the benefits of coupling QFD, independence axiom from axiomatic design, and ITIL v4 SVS practices to reinforce customer-centric modular app development. The framework supports requirements' elicitation, solution space development, and solution space evaluation (see Figure 4).

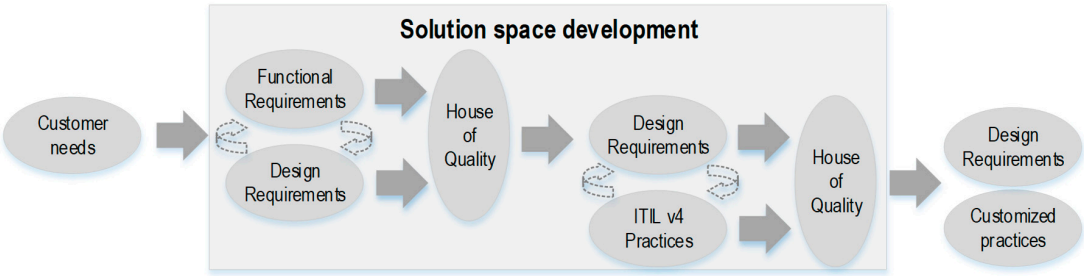

Figure 4. A framework for customer-centric modular design projects.

House of Quality as a backbone of QFD helps to direct the design process toward customer needs. House of Quality (HoQ) is a comprehensive visualization tool in the form of a matrix used to translate customers' needs into functional requirements and, subsequently, into design parameters, thus supporting process plans and production requirements [32]. Figure 5 shows a typical matrix of a House of Quality. The example illustrated in this figure involves *m* customer needs (CN*i*) and *n* functional requirements (FR*j*) derived from these customer needs. The relationships within the matrix range from "Weak" to "Strong". These are translated using a rating scale such as 1–3–9 or 1–5–9. The roof of the matrix represents correlations among FR*j*.

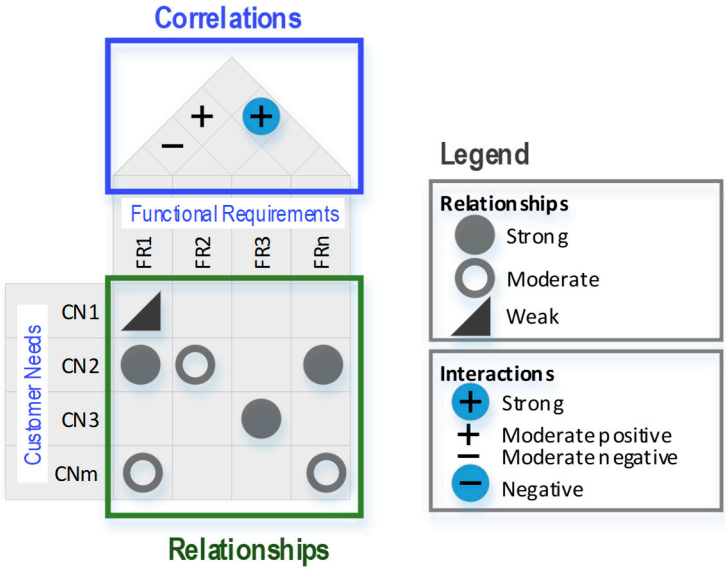

Figure 5. Example of House of Quality (HoQ) matrix.

Independence axiom is checked when deriving functional requirements and then design requirements, using House of Quality, to ensure a modular design as much as feasible. After design requirements are identified, practices from ITIL are identified and prioritized to guide subsequent development steps.

Prerequisite for proceeding with the design is gathering and processing customers' needs. User requirements are often inconsistent and incomplete. Therefore, requirements' elicitation plays a major role in integrating the voice of customer (VOC) in the design process, through customer needs. To this end, several data sources (e.g., expert judgment, commercial databases) and collection methods (e.g., brainstorming, interviews, focus groups, surveys) can be used complementarily. In addition to data gathering, requirements need to be analyzed and prioritized to mitigate the complexity of a mobile app development process. Analysis and prioritization rely on similar techniques as data gathering such as expert judgment and multi-criteria decision making.

Based on the identified customer needs, the framework further supports building a "solution space" to meet these needs. It relies on an iterative development process allowing for progressively defining functional requirements and design parameters of the mobile app.

The process is as follows. A set of FRs is identified and which supports the achievement of CNs resulting from requirements' elicitation. For each module in FRs, at least one corresponding DP is needed. Second, the House of Quality is used to evaluate the relationships between FRs and DPs, the importance of DPs, and the position of the solution in regard to competition. This represents one iteration, as shown in Figure 5. Further iterations result from breaking down DPs and FRs consistently with the axiomatic design zigzagging principle. According to this principle, FR is broken down into sub-FRs, and then the corresponding DP is broken down into sub-DPs corresponding to newly defined sub-FRs. This process leads to building tree structures of FRs and DPs (see Figure 6). Different types

of relations can be established between nodes such as mandatory, optional (OR), alternative (XOR), require, and exclude [33].

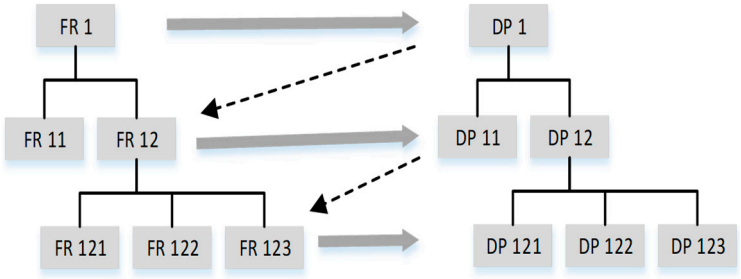

Figure 6. Hierarchical decomposition and zigzagging principle.

The Zigzagging between nodes allows for ensuring independence axiom is met as far as feasible (consistently with axiomatic design approach) and supports the modular design of the mobile app. Furthermore, it supports a controllable design process, as it reveals the relationships that exist between FRs, DPs (mapping), parent and children FR nodes, and parent and children DP nodes.

The subsequent step consists of customizing the ITIL v4 SVS practices to the development project context. This step is iterative and is supported by House of Quality, allowing to derive the relationships between DPs and ITIL v4 SVS practices to ultimately prioritize these latter for subsequent development steps. The basic idea of this mapping between DPs and ITIL v4 SVS practices is to make sure that the value stream activities of the mobile app are efficiently and effectively conducted to keep delivering high value to the end customer and all stakeholders. Once ITIL v4 SVS practices are prioritized, proper technical solutions are needed to implement them in the context of the developing company. The ITIL v4 SVS practices related to service management and technical management in ITIL are listed in Table 4 [31].

Considering these practices during the design of the mobile app enables the company to proactively deal with potential risks and opportunities that may occur during the development and operation phases, hence, ensuring a better customer focus.

6. Case Study and Discussion: COVID-19 Tracking App

To briefly illustrate the proposed framework, it was chosen to focus on an example of urgent needs requiring an agile development process. The current COVID-19 pandemic outbreak is perfectly consistent with these requirements. High contagion and morbidity rates add to the complexity of tracking this pandemic outbreak. Depending on target users' professions, risk exposure, and whether they are infected, they may have different expectations from such a mobile app. Customers could include healthcare personnel, confined adults, or researchers in healthcare management systems. Data about customer needs was collected using surveys. The main questions were what potential customers expect from the mobile app for informing and helping people during the pandemic. A total of four respondents representing six different customer profiles answered the survey. Survey results were processed by the authors to filter and prioritize CNs (see Table 1). The importance ω_i of a given customer need i was derived from the occurrence number of i within the total number of expressed customers' needs N (see Equation (1)). The respondents formulated their own needs by answering open-ended questions included in the survey. After collecting all answers, the needs were analyzed and reformulated (e.g., combining similar ones), resulting in eight CNs. The new versions of the CNs were checked with respondents to make sure they still reflect their needs.

$$\omega_i = {}^{o_i}/_{N} \times 100\% \quad (1)$$

Unsurprisingly, the overarching customer need was directly related to the question asked and consisted of tracking the number of infections in the users' neighborhood. However, several other

needs arose, such as open businesses and help offers. The survey uncovered also other expectations from potential customers regarding updates on new developments.

After CNs were identified and analyzed, a set of comprehensive FRs were derived using House of Quality while trying to be consistent with the independence axiom as much as "feasible". Figure 6 shows FRs mapping to CNs as well as the correlations among FRs. Full-textual descriptions of FRs are shown in Table 2. The identification of FRs was relatively straightforward since they were derived from CNs, so there was no need to apply creativity methods. In Figure 7, for example, there is a strong relationship between FR1 and CN1 since collecting infection and healing numbers is a prerequisite for allowing to track the number of infections. On the contrary, FR9 and CN2, for example, were only weakly related because visualizing charts and diagrams is not necessary for meeting customer needs in terms of alerts and news. Furthermore, FR1 had a strong positive interaction with FR3 and FR4 since these functions all relate to data collection. In terms of correlation, a negative correlation was identified between FR5 and FR10, as collecting detailed data about user symptoms (FR5) hinders limiting user inputs (FR10).

Table 1. Customer needs (CNs).

Code	Overview	Importance (%)
CN1	Allow tracking number of infections in my neighborhood.	31%
CN3	Overview of open businesses and help-offers in my neighborhood.	15%
CN6	Newsflash about new developments (treatments, vaccines, etc.).	12%
CN8	Should be simple to use.	12%
CN4	Self-assessment for COVID-19 symptoms, arrange testing if necessary.	8%
CN5	Prognosis is based on the current numbers of infections (e.g., illustrate the curve).	8%
CN2	Overview and flash alerts about current governmental regulations and news.	7%
CN7	Healing statistics.	7%

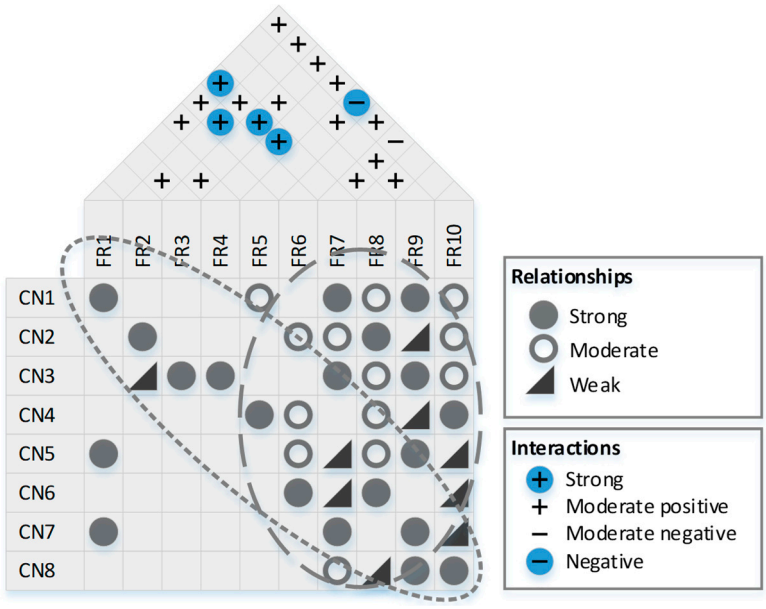

Figure 7. House of Quality—FRs' identification based on CNs.

Applying the independence axiom resulted in strong relationships across the diagonal of the House of Quality matrix. However, a one-to-one mapping, reflecting perfect independency, was not achieved for any of the relationships between CNs and FRs. In particular, a second area characterized with several strong, moderate, and weak relationships can be seen on the right side of the House of Quality (see Figure 7). This was owed to FRs (e.g., from FR 7 to FR10) being common to several CNs. This indicated a relatively high coupling among FRs and CNs and, thus, difficulties to ensure

consistency with independence axiom. These FRs support rather back-office requirements enabling customer services. This was supported also by the strong correlations particularly between FR7 and FR1, FR3, and FR4. Generally, this reflected a relatively high commonality among the solution space, as a limited set of FRs can be used for several CNs. Interestingly, this supports the idea of economies of scale rather than low modularity (because of the decoupled matrix).

The importance of a given functional requirement j referred to by φ_j is calculated according to Equation (2), such that the importance ω_i is the importance of a given customer need i, h_{ij} is the coefficient of the mapping matrix representing the strength of the relationships between customer need i and functional requirement j, and N and F are, respectively, the total number of expressed CNs and the total number of FRs.

$$\varphi_i = \left. \sum_{i=1}^{N} \omega_i h_{ij} \right/ \sum_{j=1}^{F} \sum_{i=1}^{N} \omega_i h_{ij} \quad (2)$$

It can be clearly seen that visualization (FR9) and filtering data according to user location (FR7) are the most important FRs to be carefully considered in subsequent steps. On the opposite, government regulation updates (FR2), collecting data about symptoms (FR5), and displaying news and updates (FR8) are seen to have very low importance. This is partly explained by the low importance of the corresponding CNs (cf. Table 1). Practical implications of this situation include decisions such as planning for separate DPs to meet these FRs to include such services only if customers request it, thus ensuring meeting individual customer requirements at lower costs. Other implications involve decisions such as removing these FRs if their realization involves high potential costs that impede economies of scales.

Table 2. Functional requirements (FRs).

Code	Overview	Importance
FR9	Visualize charts and diagrams	25%
FR7	Filter data according to user location	20%
FR1	Collect infection and healing number per city in a given country	16%
FR10	Limit user input	14%
FR6	Collect data about OMS and government updates	6%
FR3	Collect data about stores opening time	5%
FR4	Collect data about available persons per city for help	5%
FR2	Collect government regulation updates	3%
FR5	Collect user data about symptoms and display results	3%
FR8	Display news and updates	3%

The subsequent step consists of identifying design parameters (DPs) with the objective to remain consistent with the independence axiom as much as possible. This step was performed by the authors with the help of a subject matter expert. Figure 8, depicting a House of Quality, shows the assessment of the relationships among DPs and FRs as well as the correlations among DPs. For example, DP2 has a moderate relation with FR1 as integrating maps can be useful for collecting infection and healing information but are not necessary if the geographical location is not needed. However, maps' integration is required to ensure FR7, for example, about filtering data according to user location. The identification of DPs results in different interaction forms among them. For instance, a strong positive interaction occurs between DP3 and DP5 as user graphical interface and online forms are mutually impacted.

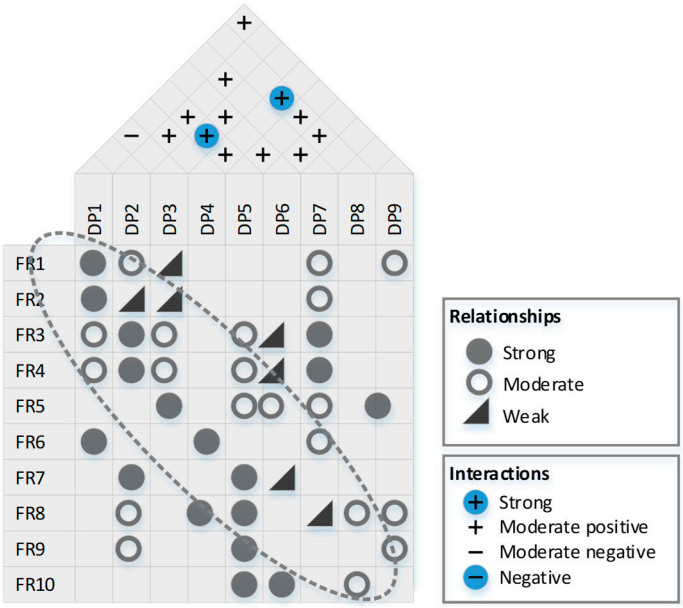

Figure 8. Results of requirements' analysis FRs vs. DPs.

While it can be seen that there are strong relationships across the matrix diagonal, the independence axiom is not fully respected since one DP is generally related to more than one FR (e.g., DP5 and FR7 to FR10). Thus, the proposed solution can be improved further, based on discussions involving subject matter experts. The upper side of the House of Quality shows the correlations among the DPs, which uncover another important aspect to address in order to ease subsequent development steps. In fact, frequent strong (positive or negative) correlations could lead to higher design complexity; thus, it is needed to limit these relationships as much as feasible, or at least to consider them when moving forward with the app development.

The importance δ_k of a given design parameter k is calculated according to Equation (3), such that φ_j is the importance of a functional requirement j, h'_{jk} is the coefficient of the mapping matrix representing the strength of the relationships between functional requirement j and design parameter k, and F and D are, respectively, the total number of FRs and the total number of identified DPs.

$$\delta_k = \Sigma_{j=1}^{F} \varphi_j h'_{ij} \Big/ \Sigma_{k=1}^{D} \Sigma_{j}^{F} \varphi_j h'_{jk} \tag{3}$$

The resulting relative importance of each of the DPs is shown in Table 3. The coupled effect of the importance of CNs, FRs resulted in the user interface (UI) design being the most important aspect to consider. "Maps' integration" and "search engine" should also be carefully addressed.

Table 3. Design parameters (DPs).

Code	Overview	Importance
DP5	User-Interface design	35%
DP2	Maps integration	20%
DP1	Search engine	15%
DP6	On-boarding process	8%
DP9	Calculator	7%
DP7	Links and booking systems	6%
DP4	Push notifications	4%
DP8	Splash screen	3%
DP3	Forms	2%

Despite several decoupled matrices, the example still shows how the method supports the improvement of the design through centric and modular perspectives. It also uncovers potential improvement areas toward a modular design.

To enlighten decision makers with the most valuable service management practices for subsequent development steps, a mapping of the DPs to the ITIL v4 SVS practices was realized using the House of Quality (see Figure 9). For example, SP17 is strongly linked to DP3 and DP5, since testing and validation, as an ITIL practice, are required to make sure forms and user interface design are appropriate and there are no failures. It can also be seen that SP1, referring to service availability, has a strong positive correlation with most of the other practices, which means that SP1 should be carefully addressed and proper resources should be allotted to it. While all practices are generally relevant to DPs, service validation and testing are seen to have paramount importance for the development of the COVID-19 mobile app. Looking at the DPs, it can be seen that the user interface (UI) design is impacted by most of the ITIL v4 SVS practices. These observed trends are confirmed by the relative importance of each of the practices, as shown in Table 4.

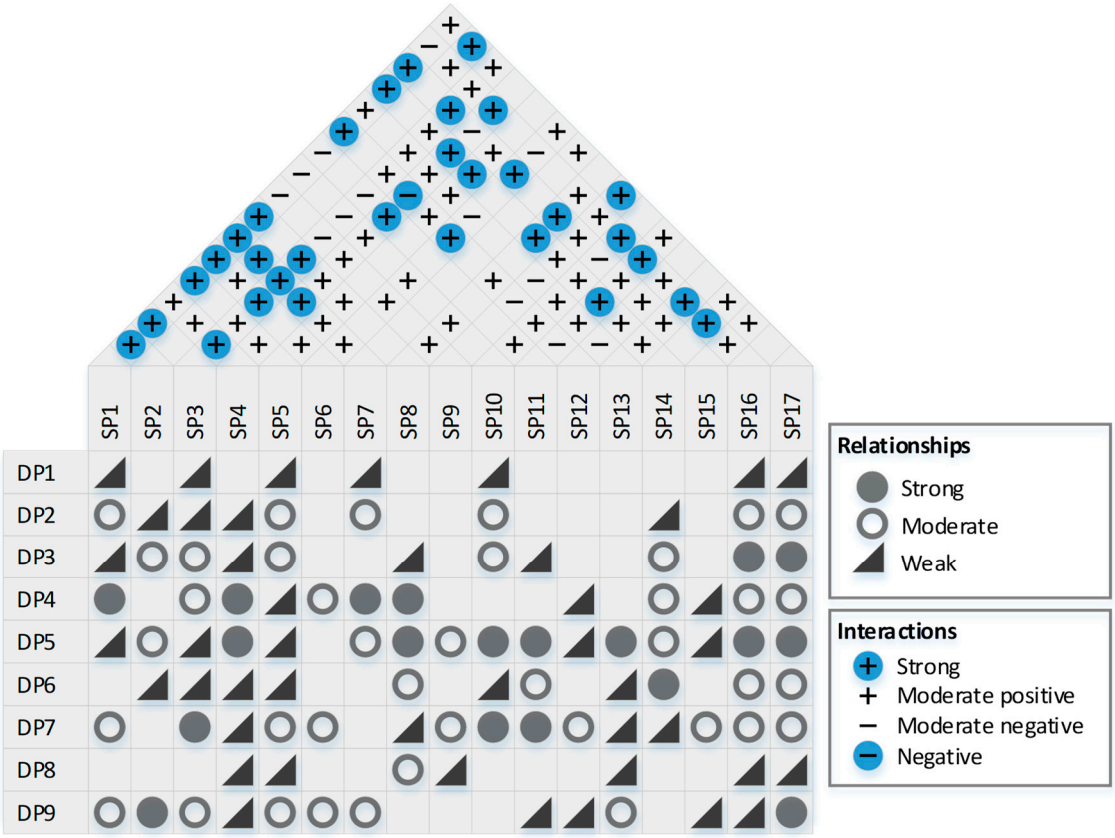

Figure 9. Results of requirements' analysis DPs vs. practices.

The prioritization of the ITIL v4 SVS practices supports the subsequent planning activities of the mobile app development. Special attention should be given to the highly ranked practices. For example, service validation and testing (SP17), how to translate this into practice, would depend on the development company context and whether it pursues process standardization and best practices' adoption such as the ones provided by ITIL v4 SVS.

For example, for newly developed applications in contexts with no past experience with ITIL v4 SVS, it might be suitable to select only a few numbers of easily manageable practices. However, regardless of the situation of the company, the process to adopt and implement these practices should be progressively consistent with the continual improvement principle from the ITIL v4 framework.

Table 4. ITIL v4 SVS practices.

Code	Overview	Importance
SP17	Service validation and testing	12%
SP10	Service catalog management	10%
SP16	Service request management	10%
SP4	Change enablement	9%
SP8	Problem management	9%
SP11	Service configuration management	9%
SP13	Service design	8%
SP2	Business analysis	5%
SP7	Monitoring and event management	5%
SP14	Service desk	5%
SP1	Availability management	4%
SP3	Capacity and performance management	4%
SP5	Incident management	4%
SP9	Release management	3%
SP6	IT asset management	1%
SP12	Service continuity management	1%
SP15	Service level management	1%

To conduct a proof of concept, the development of a prototype for illustration purposes was conducted. The developer in charge of this activity was not involved in the initial steps of the method. This allowed us to check both the clarity and comprehensiveness of the DPs.

Figure 10 shows one of the GUIs (Graphical User Interfaces) from the ongoing prototype. The development of this mock-up went through several iterations and each of them involved a "back-and-forth" between technical development, DPs and FRs: implementing functions progressively, getting feedback, taking further (improvement) decisions, etc. As such, the proposed framework was consistent with agile development.

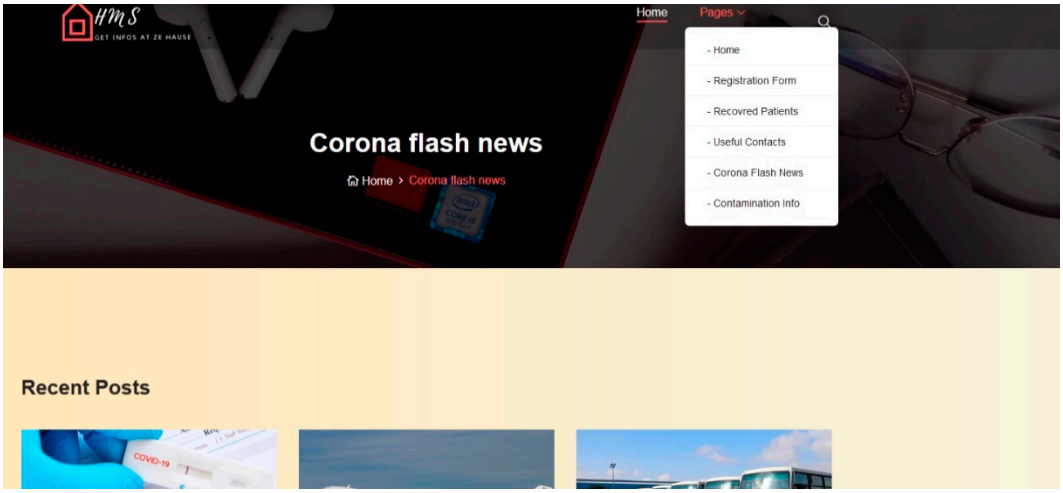

Figure 10. Screenshot of the working prototype.

7. Assessment of the Proposed "Customer-Centric Modular Design Projects" Framework

The proposed framework supported and systematized customer-centric modular design through coupling QFD and modular design, as well as adopting ITIL v4 practices for service management. The illustrative example showed the applicability of the framework to develop a mobile app for pandemic relief in the context of COVID-19. A questionnaire was designed to assess the usability of the framework by potential users and its effectiveness with regards to its objectives. These latter form the questionnaire items: (1) The framework supports a customer-centric design, (2) the framework supports

a modular design, (3) the framework is effective for mobile app development, (4) the framework provides insights into subsequent development steps through ITIL practices, and (5) the framework supports ramping up customer modular design (see Figure 11). The questionnaire included an overview of the framework, as shown in Figure 4, and a short illustrative example with comments on the steps, including three figures representing the House of Quality matrices and three tables summarizing the results.

The questionnaire was implemented online and spread among more than 300 potential users of the framework. The questions relied on a five-level Likert scale indicating the respondent's level of agreement: 1 = strongly disagree, 2 = disagree, 3 = neither agree nor disagree, 4 = agree, and 5 = strongly agree. After two reminders within a one-month period, a total of thirty-six valid replies were collected. The respondents were in the fields of business analysis, software development, requirements' engineering, and industrial and system engineering. The number of responses was assumed to be acceptable for this study as it allowed us to perform descriptive statistics and collect general insights into the framework. The results of the survey are summarized in Figure 11, representing the scores of the five items (i.e., an average of the scores assigned by respondents).

In general, the framework proved to meet the objectives for which it was designed as the scores of all five items were in the range 3.58–3.92. An agreement was observed about the fact that the framework supported customer-centric design. This is witnessed by the high score of the item amounting to 3.92 and the relatively low standard deviation (0.99). The survey supported the idea that the framework is effective for ramping up modular design (average score of 3.70) and customer-centric design (average score of 3.64). The coupling with ITIL practices was also perceived positively by the respondents in the sense that provided insights into subsequent development steps, thus helping to prioritize the tasks and structuring the subsequent process.

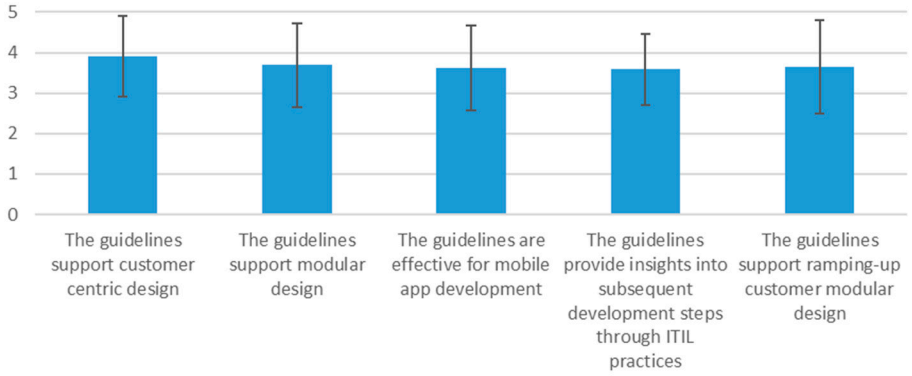

Figure 11. Assessment results.

Both the illustrative example and survey suggest that the proposed framework can be a relevant support tool toward the general objective of ramping up the customer-centric modular design of a mobile app. The coupling of QFD and ITIL supports a structured process to systematize a mobile app development and system development at large.

The illustration from the pandemic context shows the ease of use of the method to quickly move from requirements to technical solutions' identification and refinement. In this sense, this paper extends the literature (e.g., [34]) in system and software development through unleashing the potential of coupling QFD and ITIL v4 SVS practices to systematize the development process considering a customer-centric perspective. Furthermore, the results are in line with previous research, for example, with regards to the key role of a user interface in sectors such as the apparel industry [35,36]. As such, the current case study witnesses the validity of these results in apps' development for pandemic relief. Furthermore, in terms of required functions, the results from the current case study are in line with the research published in [37] regarding limiting user input in the app to get results.

8. Conclusions

The shift to more and more individual solutions is also affecting software development, especially the design of mobile apps. It has been shown that the required customization to satisfy their diverse users in a competitive market is a challenge for the app providers. It creates specific problems for development projects in order to meet demands at a larger scale. Due to the rising complexity, customer-driven projects and operations' management need to be supported by effective tools to operationalize enterprise-wide strategies.

Such an approach has been developed in this paper, employing QFD and axiomatic design to deal with the complexity using a modular architecture. It provides a step forward in this area by establishing a framework to smoothly translate customer requirements into design parameters consistently with a modular design principle. The framework was exemplarily validated in a case study on a COVID-19 tracking app. The framework needs, however, to be intensively tested in collaborative development projects for further improvement. Joint application design (JAD) exhibits a high potential to proceed with such tests and improvements. These improvements can benefit from recent research works (e.g., [38]). Ongoing research involves the validation of the method at a larger scale. This effort is being conducted within the VARIETY project (VARIETY and Complexity Management in the Era of Industry 4.0).

Author Contributions: Conceptualization, K.M., M.P., and S.W.; methodology, K.M., S.W., M.P., and D.R.; validation, K.M. and D.R.; writing—original draft preparation, K.M., M.P., and S.W.; writing—review and editing, D.R. and K.M.; funding acquisition, K.M. and S.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research is partly supported by the collaborative project VARIETY (VARIETY and Complexity Management in the Era of Industry 4.0) funded by Région Auvergne-Rhône-Alpes (AURA) (https://variety.wp.imt.fr/).

Acknowledgments: The authors would like to thank the media manager of the InFo+ club for the valuable contribution to the proof of concept during the period June–August 2020.

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

References

1. Nagappan, M.; Shihab, E. Future trends in software engineering research for mobile apps. In Proceedings of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Suita, Osaka, Japan, 14–18 March 2016; Volume 5, pp. 21–32.
2. Bourne, V. *How to Keep Pace with Mobile Consumer Expectations*; Sitecore: San Francisco, CA, USA, 2016.
3. Pine, B.J. *Mass Customization*; Harvard Business School Press: Boston, MA, USA, 1993.
4. Lusky, M.; Powilat, C.; Böhm, S. Software cost estimation for user-centered mobile app development in large enterprise. In Proceedings of the International Conference on Applied Human Factors and Ergonomics, Los Angeles, CA, USA, 17–21 July 2017; Springer: Cham, Switzerland, 2017; pp. 51–62.
5. Schnall, R.; Rojas, M.; Bakken, S.; Brown, W.; Carballo-Dieguez, A.; Carry, M.; Gelaude, D.; Mosley, J.P.; Travers, J. A User-centered model for designing consumer mobile health (mHealth) applications (Apps). *J. Biomed. Inform.* **2016**, *60*, 243–251. [CrossRef] [PubMed]
6. Lopes, A.; Valentim, N.; Moraes, B.; Zilse, R.; Conte, T. Applying user-centered techniques to analyze and design a mobile application. *J. Softw. Eng. Res. Dev.* **2018**, *6*, 5. [CrossRef]
7. Abrahamsson, P.; Hanhineva, A.; Hulkko, H.; Ihme, T.; Jaalinoja, J.; Korkala, M.; Koskela, J.; Kyllonen, P.; Salo, O. Mobile-D: An agile approach for mobile application development. In Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications, Vancouver, BC, Canada, 24–28 October 2004; pp. 174–175.
8. Jeong, Y.-J.; Lee, J.-H.; Shin, G.-S. Development process of mobile application SW based on agile methodology. In Proceedings of the 2008 10th International Conference on Advanced Communication Technology, Gangwon-Do, South Korea, 17–20 February 2008; IEEE: Piscataway, NJ, USA, 2008; Volume 1, pp. 362–366.

9. Rahimian, V.; Ramsin, R. Designing an agile methodology for mobile software development: A hybrid method engineering approach. In Proceedings of the 2008 2nd International Conference on Research Challenges in Information Science, Marrakech, Moroccom, 3–6 June 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 337–342.
10. Kuo, T.C. Mass customization and personalization software development: A case study eco-design product service system. *J. Intell. Manuf.* **2013**, *24*, 1019–1031. [CrossRef]
11. Verdouw, C.; Beulens, A.; Wolfert, S. Towards software mass customization for business collaboration. In Proceedings of the 2014 Annual SRII Global Conference, San Jose, CA, USA, 23–25 April 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 106–115.
12. Kang, J.-W.; Namkung, Y. The role of personalization on continuance intention in food service mobile *Apps*. *Int. J. Contemp. Hosp. Manag.* **2019**, *31*, 734–752. [CrossRef]
13. Van Aken, J.E.; Romme, A.G.L. A design science approach to evidence-based management. In *The Oxford Handbook of Evidence-Based Management*; Oxford University Press: Oxford, UK, 2012; pp. 43–57.
14. Pfeffers, K.; Tuunanen, T.; Gengler, C.E.; Rossi, M.; Hui, W.; Virtanen, V.; Bragge, J. The design science research process: A model for producing and presenting information systems research. In Proceedings of the First International Conference on Design Science Research in Information Systems and Technology (DESRIST 2006), Claremont, CA, USA, 24–25 February 2006; pp. 83–106.
15. Kroll, P.; Maclsaac, B. *Ramp up Development Projects More Quickly with Reusable Methods and Processes*; IBM Corporation: New York, NY, USA, 2008.
16. Skobelev, P.; Eliseev, V.; Mayorov, I.; Travin, V.; Zhilyaev, A.; Simonova, E.V. Designing distributed multi-agent system for aggregate and final assembly of complex technical objects on ramp-up stage. In Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018), Funchal, Portugal, 16–18 January 2018; pp. 250–257.
17. Medini, K.; Erkoyuncu, J.A.; Cornet, C. A model for cost-benefit analysis of production ramp-up Strategies. In Proceedings of the IFIP Conference on Advances in Production Management Systems (APMS), Novi Sad, Serbia, 30 August–3 September 2020; pp. 731–739.
18. Slamanig, M.; Winkler, H. An exploration of ramp-up strategies in the area of mass customisation. *Int. J. Mass Cust.* **2011**, *4*, 22–43. [CrossRef]
19. Schmitt, R.; Heine, I.; Jiang, R.; Giedziella, F.; Basse, F.; Voet, H.; Lu, S. On the future of ramp-up management. *CIRP J. Manuf. Sci. Technol.* **2018**, *23*, 217–225. [CrossRef]
20. Corral, L.; Sillitti, A.; Succi, G. Software development processes for mobile systems: Is agile really taking over the business? In Proceedings of the 2013 1st International Workshop on the Engineering of Mobile-Enabled Systems (MOBS), San Francisco, CA, USA, 25 May 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 19–24.
21. Joorabchi, M.E.; Mesbah, A.; Kruchten, P. Real challenges in mobile app development. In Proceedings of the 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, Baltimore, MD, USA, 10–11 October 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 15–24.
22. Cao, H.; Zhang, H.; Liu, M. Global modular production network: From system perspective. *Procedia Eng.* **2011**, *23*, 786–791. [CrossRef]
23. Clarkson, J.; Eckert, C. *Design Process. Improvement: A Review of Current Practice*; Springer: Berlin/Heidelberg, Germany, 2010.
24. Ammann, P.; Offutt, J. *Introduction to Software Testing*; Cambridge University Press: Cambridge, UK, 2016.
25. Gonçalves-Coelho, A.M.; Mourão, A.J.; Pereira, Z.L. Improving the use of QFD with axiomatic design. *Concurr. Eng.* **2005**, *13*, 233–239. [CrossRef]
26. Herzwurm, G.; Schockert, S.; Tauterat, T. Quality function deployment in software development-state-of-the-art. In Proceedings of the 21st International Symposium on Quality Function Deployment (ISQFD), Hangzhou, China, 22 August–26 September 2015.
27. ISO/IEC 25010 Systems and Software Engineering. *Systems and Software Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models*; ISO: Geneva, Switzerland, 2011.
28. Dörbecker, R.; Böhmann, T. The concept and effects of service modularity—A literature review. In Proceedings of the 2013 46th Hawaii International Conference on System Sciences, Wailea, HI, USA, 7–10 January 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1357–1366.
29. Park, G.-J. *Analytic Methods for Design Practice*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2007.

30. Suh, N.P.; Do, S.-H. Axiomatic design of software systems. *CIRP Ann.* **2000**, *49*, 95–100. [CrossRef]
31. Axelos. *ITIL Foundation ITIL 4 Edition*; Axelos: London, UK, 2019.
32. Park, T.; Kim, K.-J. Determination of an optimal set of design requirements using house of quality. *J. Oper. Manag.* **1998**, *16*, 569–581. [CrossRef]
33. Togay, C.; Caniaz, E.S.; Dogru, A.H. Rule-based axiomatic design theory guidance for software development. In Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops, Izmir, Turkey, 16–20 July 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 544–552.
34. Chen, R.Y. Feedback-based eco-design for integrating the recency, frequency, and monetary value of eco-efficiency into sustainability management. *Systems* **2016**, *4*, 30. [CrossRef]
35. Blazek, P.; Pilsl, K. User interface trends for mobile-optimized product configurators. In *Customization 4.0*; Springer: Cham, Switzerland, 2018; pp. 357–373.
36. Manca, M.; Paternò, F. Customizable dynamic user interface distribution. In Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, Brussels, Belgium, 21–24 June 2016; pp. 27–37.
37. Tobing, R.D.H.; Pardede, L.V.D.; Panjaitan, I.S.; Sianturi, E.Y. Customizable commerce mobile application. In Proceedings of the 3rd International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, Malaysia, 15–17 August 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 174–178.
38. Wall, J.; Bertoni, M.; Larsson, T. The model-driven decision arena: Augmented decision-making for product-service systems design. *Systems* **2020**, *8*, 22. [CrossRef]