Article

A Coding-Free Software Framework for Developing Lightweight Web Data Management Systems

Can Yang ^{1,*} , Shiyang Pan ^{1,*}, Runmin Li ¹, Yu Liu ² and Lizhang Peng ³¹ Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China; 15521115851@163.com² Shenzhen Tencent Computer Systems Co., LTD., Shenzhen 518057, China; rainydayliu@tencent.com³ Vipshop Holdings Limited, Guangzhou 510370, China; pengzh@gmail.com

* Correspondence: cscyang@scut.edu.cn (C.Y.); 201721041376@mail.scut.edu.cn (S.P.)

Received: 2 January 2020; Accepted: 21 January 2020; Published: 27 January 2020

Abstract: Increasingly more enterprises are intending to deploy data management systems in the cloud. However, the complexity of software development significantly increases both time and learning costs of data management system development. In this paper, we investigate the coding-free construction of a data management system based on Software-as-a-Service (SaaS) architecture, in which a practical application platform and a set of construction methods are proposed. Specifically, by extracting the common features of data management systems, we design a universal web platform to quickly generate and publish customized system instances. Then, we propose a method to develop a lightweight data management system using a specific requirements table in a spreadsheet. The corresponding platform maps the requirements table into a system instance by parsing the table model and implementing the objective system in the running stage. Finally, we implement the proposed framework and deploy it on the web. The empirical results demonstrate the feasibility and availability of the coding-free method for developing lightweight web data management systems.

Keywords: data management system; requirements-table-driven; SaaS; spreadsheet

1. Introduction

With the development of information technology, enterprises often need to develop their own information systems [1]. However, for many years, the development process has been difficult for enterprises because of the high complexity of both time and learning costs of data management system development, especially for small- and medium-sized enterprises. The recent statistical results show that by the end of the last century, only 26% of information systems' IT projects were successfully used in practical applications [2]. Moreover, due to the high complexity of software and system development, it is notable that non-programmers [3] often endure the difficulty of developing data management software in isolation. As a consequence, over the years, there have been far fewer programmers than non-programmers [4].

Software development has transitioned from a client-based environment to a web-based one since the 1990s [5]. In software systems, Software-as-a-Service (SaaS) and multi-tenancy are currently leading to dramatic changes [6], where the SaaS model refers to enterprises or individuals using software hosted and managed by SaaS providers [7], who aim to provide economic benefits by taking advantage of the commonality of large-scale software [8]. Businesses, also called tenants, of SaaS subscribe to hosted services and access these services via web browsers or client programs [9]. Today, cloud-oriented SaaS provides a viable technical approach for coding-free software development. According to the Gartner report, SaaS is forecasted to grow to \$75.73 billion by 2021, with a CAGR of 18.4%. Due to the benefits of SaaS, enterprises are increasingly deploying information systems to the SaaS cloud, and different technologies for customizable SaaS platforms had been proposed in

recent years. In [10], Tsai et al. proposed a SaaS development framework named EasySaaS designed to simplify SaaS development. In [11], Tsai et al. also proposed a model-driven SaaS application platform to develop a complete application through the model description file, providing a convenient and quick solution for program customization. More recently, Tsai and Sun [12] proposed an innovative customization approach that studies similar tenants' customization choices and provides guided semi-automated customization processes for future tenants. They divided SaaS applications into four layers: GUI, business logic, service and data. Different aspects of the application are customized and modelled using the orthogonal variability model (OVM), and a guided framework was proposed. However, they did not focus on the scalability of the platform. Zhang et al. [13] implemented a big data intelligent SaaS platform, which supports the dynamic construction and operation of intelligent data analysis applications. In addition, they used cloud workflow technology to realize the rapid development and process deployment of business analysis processes. Recently, Xu et al. [14] proposed a demand-driven data-centric web service configuration method, which represented data resources as data-centric web services. They implemented a scalable dynamic platform, which enabled people to automatically package information resources as web services and expose them at the business level. The information resources of different tenants are isolated through REST services. They also modelled the system first, and then mapped the model onto the platform.

In dealing with information system development, it is notable that information management commonly involved reports based on spreadsheets or EXCEL [15] due to the popularity and ease of use of spreadsheets, in which the spreadsheets are ubiquitous information management tools that cross technical and cultural divides. According to the 2012 United States Census, over 55 million people in the United States “manipulate data with either spreadsheets or databases at work” [16]. Therefore, it is very advantageous to leverage spreadsheets in a SaaS environment to make it easier to customize and deploy SaaS applications.

In this paper, we will focus our attention on the lightweight data management system based on SaaS architecture, in which we propose a method to develop a lightweight data management system. To this end, we first introduce a general lightweight development framework for data management systems based on SaaS, where the framework mainly involves two parts: a data management system web platform (DMSWP) and a system requirements-table-driven development method (RTDDM):

- The DMSWP platform provides a scalable, multi-tenant software application model that provides both tenants (system administrators) and ordinary users access to the system, which supports tenants in developing software systems based on the RTDDM and supports tenants in interactively customizing and configuring software systems. Based on DMSWP, we provide a set of general data web interfaces (GDWIs) for users to exchange data or do more. It is notable that unlike the traditional DMSWP, our proposed framework focuses more on the logic at the data operation level, while they focus on the logic at the business level.
- The RTDDM method supports an offline table to drive the development of data management systems. This is a coding-free method that enables users to develop and deploy data management systems on the SaaS platform efficiently by directly customizing the system requirements into a given specific table. Based on this method, we design a scalable and RTDDM-enabled system requirements table (ReTa), which can completely describe a software system with data management as the core requirement. In addition, the RTDDM method that we proposed is easier to understand and employ by users without a programming background. To the best of our knowledge, no method similar to the RTDDM has previously been presented in the literature.

The remainder of this paper is organized as follows: In Section 2, we overview some related works. In Section 3, we present the proposed system architecture. In Sections 4 and 5, we present the architecture and the implementation of DMSWP, RTDDM and ReTa. In Section 6, we evaluate of the proposed platform and method. Finally, in Section 7, we summarize the work and propose future research directions.

2. Related Work

In this section, we will overview some of the works with respect to SaaS and spreadsheets that are related to the software development of this paper.

2.1. System Software Development Based on SaaS

In the system software development based on SaaS, most studies adopt the model-driven architecture (MDA), in which the MDA model is the core and the development is driven by model mapping. For example, Mohamed et al. [17] proposed an integration platform based on service orientation, software product lines (SPLs) and MDA, in which the platform supports configuration customization for runtime tenants. Sharma and Sood [18] explored the MDA approach for developing SaaS applications, where they proposed a development method from a platform-independent model (PIM) to a platform-specific model (PSM) and then binding from PSM to the execution platform. Cai et al. [19] combined MDA ideas to promote SaaS application development and proposed a model-driven development model based on semantic reasoning mechanisms. However, their work is mostly focused on Cloud of Things (CoT) applications. In [20], Ma et al. proposed data middleware to customize a multi-tenant database, generated SaaS applications from the model with the help of model transformation, and rapidly generated application code through modelling. Despite the wide applications, MDA still needs to master professional techniques and grammars, and it is still difficult for non-professionals to quickly produce software systems through MDA.

In addition to MDA, several researchers have proposed different ways to quickly customize software on SaaS platforms. Mietzner et al. [21] proposed a multi-tenant pattern that can be used to design, develop, and deploy service-oriented applications that support processes. They described how different SaaS application components can be combined horizontally or vertically. Scheibler et al. [22] proposed a platform-independent approach to modelling, describing and implementing enterprise application integration (EAI) patterns in a service-oriented architecture to integrate different applications into a SaaS environment. Liu et al. [23] and Zhang et al. [24] used WSCL to build SaaS services and describe SaaS models. Zhu and Wang [25] combined MDA thoughts with SOA, proposed an STML-based method for describing business requirements, and implemented integrated development tools to enable users to quickly build SaaS applications. Zhang et al. [26] proposed a requirements-based, service-oriented, model-driven architecture approach, which applies MDA meta-level analysis to facilitate the construction of SaaS mashup applications. Etedali et al. [27] proposed a method to support a multi-tenant SaaS system with feature modelling and extensible markup language (XML) filtering technology. Their main work was to realize the automatic encoding of XML documents into the format required by Yfilter technology and improve the efficiency of developing and deploying SaaS applications. Jiang et al. [28] proposed a new multi-tenant architecture design pattern, model-driven layered architecture (LMDA), which divides the model into four layers: a domain-independent component layer, a domain-related component layer, an abstract business layer and an abstract SaaS application layer. They also suggested a way to build SaaS tenant applications using each layer model. Nevertheless, it is notable that the assembly environment in LMDA and most of the aforementioned studies require specific plug-ins, such that they are unable to avoid tools that use the C/S architecture in modelling.

2.2. Spreadsheet-Driven Development

Spreadsheets are an information management tool widely employed by end users [16], which are easily accessible and familiar to end users [29]. In recent years, many researchers have developed the methods of using spreadsheets to drive development. For instances, Benson et al. [16] developed a system for creating basic web applications using spreadsheets instead of servers and described the client-side UI using HTML. The result was a responsive read-write compute webpage. Chang and Myers [30] proposed the Gneiss system, which tightly integrated the spreadsheet editor with the web

interface builder such that users could demonstrate the binding between web GUI elements and cell attributes in spreadsheets when dealing with actual web service data. Kongdenfha et al. [31] proposed a spreadsheet-based web mashup development framework. Moreover, Hoang et al. [29,32] proposed a universal MashSheet based on a spreadsheet, which enabled users to build mashup using spreadsheet examples. Honkisz et al. [33] proposed a presentation approach using spreadsheets as middleware for transforming natural language text into business process models. Similarly, Wisniewski et al. [34] proposed a way to describe business processes through spreadsheets. They modelled BPMN processes through spreadsheets, mapping process model elements to a spreadsheet representation.

3. Framework Architecture and Formal Modelling

In this section, we will introduce the formal modelling for data management systems with common characteristics, as well as the system architecture and working modules of the proposed framework. Figure 1 shows the system architecture, which is composed of the following three main parts, i.e., RTDDM, DMSWP and GDWI: (1) The DMSWP part aims to extract the core commonalities of the data management system and to design a lightweight web framework; (2) The RTDDM part is a method for offline customization using spreadsheets, in which we design a set of requirements table called ReTa. The RTDDM engine analyses and executes based on the system requirements table and derives the corresponding system instance on the DMSWP; (3) The GDWI module provides users with an interface for data services and an operating environment. This module is based on the DMSWP and is compatible with the RTDDM.

Figure 1. Framework Module and Architecture.

Before implementing the aforementioned framework, a unified architecture must be designed for the data management system (DMS). To this end, we extract the core commonalities of the DMS and proceed with formal modelling. We abstract the commonalities of the DMS using a five-tuple defined in the following Definition 1:

Definition 1. *DMS is a five-tuple components:*

$$\Psi = F(\text{Tenant}, \text{Group}, \text{User}, \text{Schema}, \text{Data}) \quad (1)$$

where F represents the engine transformation function, and *Tenant* is the system administrator who uses the framework to create the system; we will use the term *tenant* instead of *administrator* below. A data management system consists of five parts: *tenant*, *user*, *group*, *schema*, and *data*.

Figure 2 shows the hierarchy of the general DMS architecture, where the dotted line represents inclusion, the solid line that does not have an arrow indicates membership, and the solid line that has an arrow for further explanation and representation. For example, the tenant information is specifically expressed as id, password, and name, respectively. The notations G_u denote the group to which the user belongs, in which a user may belong to one or more groups. Any user's permissions are the union set of the authority of group to which it belongs. Additional explanations are given as follows:

- The gpermission and opermission in the schema representation are used to distinguish the permissions of different groups on the schema. For any group that has the permissions of schema, it can have the permissions of any subset of schema.
- FI refers to the field information in the schema. Field information can be further interpreted as field names, field types, and other attributes of fields (OAs).
- Each piece of data belongs to a schema. For each data item belonging to schema, the attribute values of each field must match the attributes of the corresponding schema's field.

Figure 2. The Hierarchy of DMS.

Additionally, the DMS has uniqueness and isolation. That is, only one system administrator with the highest authority corresponds to a system, and multiple systems are independent of each other.

4. DMSWP in the Cloud

In this section, we will specify the DMSWP platform whose architecture is shown in Figure 3.

Figure 3. The Three-Tier Architecture of the DMSWP.

For the DMSWP, the end users can be divided into three categories, i.e., administrators, tenants and ordinary users. For different categories of users, the DMSWP provides different business logic to satisfy the service on demand. For example, for the administrators, the corresponding business logic layer service involves tenant management. That is, the administrator manages the system registered on the DMSWP uniformly. For the tenants, the corresponding business logic layer services include user management, group management, schema management and metadata management. For the ordinary users, namely, users under a tenant, the corresponding service of the business logic layer is data management. The platform administrator is unique and can manage multiple tenants. Each tenant manages a data management system, which can have multiple ordinary users.

From the system architecture point of view, the DMSWP is roughly divided into three layers, i.e., the user interface layer, the business logic layer and the data storage layer. To facilitate maintenance and migration, the DMSWP adopts a loosely coupled design between layers. In what follows, we will specify the three-layer structure.

User interface layer (UIL) provides customers with access to the DMS on the Internet and supports interactive operations between customers and the system. The UIL is implemented by Google Web Toolkit (GWT) and SmartGWT, and its separation from the business logic layer is realized through remote procedure call (RPC) communication. We use GWT and SmartGWT because they are very interactive and have their own UI library that allows developers to build web applications quickly [35].

Business logic layer (BLL) contains modules, as shown in Figure 3. For the data management module, we also implement the GDWI module. In the GDWI module, we implement advanced functions such as multi-condition combined query, statistics and data migration. The data migration function includes import and export. The import function includes imports from spreadsheets and traditional relational databases such as MySQL. That is, it can quickly integrate the enterprise's resources in different databases into the system, such as SQLServer, MySQL, etc. An export function can export the data from a DMS into a spreadsheet.

The BLL implements the respective Manager classes and defines the I*Manager interface to implement specific operational functions. The parameters passed into the parsing API are provided by the ParameterInParser class. The results returned by the parsing API are provided by the ParameterOutParser class. The permission verification is obtained by calling the PermissionManager class. The conceptual UML class diagram for the BLL is shown in Figure 4.

Data storage layer (DSL) includes the database implementation and unified data access module. In terms of database implementation, the platform adopts the schemaless database MongoDB to establish the database model of a multi-tenant application framework. Because of MongoDB's schemaless nature, data of different structured data tables can coexist within the same table.

Figure 4. Business Logic Layer Conceptual UML Class Diagram.

5. RTDDM

Given that spreadsheets have a wide audience and low learning costs [16], we attempted to move the production system process to custom spreadsheets. Based on the characteristics of the DMSWP, we propose a development method called the system requirements-table-driven method to develop a DMS and design a set of scalable system ReTa. In this section, we elaborate on how to use the system ReTa to describe a system instance on the DMSWP and describe the workflow for executing this method on the DMSWP.

5.1. The Hierarchical Structure of RTDDM

The RTDDM can customize the personalized DMS through a spreadsheet in a specific format. The workflow of the specific RTDDM is shown in Figure 5. As shown in the figure, the RTDDM includes three main layers: presentation layer, drive engine layer and application system layer.

Figure 5. The Work Execution Process and Three-tier Architecture of the RTDDM.

- **Presentation Layer:** The presentation layer is the layer where the user interacts directly with the RTDDM. In the RTDDM, the presentation layer takes a spreadsheet as input such that users have a convenient and simple entity through which to interact with the DMSWP. The presentation layer describes users' requirements for a DMS by designing a set of system ReTa. Users configure the system ReTa to customize the DMS. In addition, this layer assembles services on demand from a user perspective.
- **Drive Engine Layer:** The drive engine layer is the bridge that converts the system ReTa from the presentation layer to the system instances on the application system layer. The main task of the drive engine layer is to parse the system ReTa customized by the tenant. The system ReTa passed from the presentation layer is read, and the information is verified. After passing the verification, the information is interpreted into language that can be parsed and executed on the DMSWP. ReTa describes the Tenant, User, Group, Schema, and Data in detail, and the drive engine layer is the language that parses the contents of these five components into database recognition. Here, we parse ReTa into JSON data and inject the data into different collections that compose the underlying MongoDB database of our system.
- **Application System Layer:** The application system layer is responsible for instantiation of the system. The DMS customized by the user in the system ReTa is transformed and parsed by the drive engine layer, executed on the DMSWP, and then generated and derived on the web. Users can access the customized system on the web.

Figure 6 is a sequence diagram of users employing the RTDDM, which describes how users can develop a system through the RTDDM. After users register and log in, the system can be generated by uploading spreadsheet files. Then, users can manage the system through the DMSWP.

Figure 6. RTDDM UML Sequence Diagram.

5.2. Requirements Tables of RTDDM

The RTDDM aims to extract the common characteristics of DMS and then models the system, which includes two types of data requirements tables, i.e., a system metadata table and a data exchange table, which are respectively specified in what follows:

- **System Metadata Table:** The system metadata table has a detailed description of users, groups and schemas in the DMS.

Table 1 shows the format of the system metadata table. From Table 1, row 1 and 2 are used for the security verification of tenant T, whereas row 3 and 4 are used to describe the group G of the DMS and incrementing the cell to the right to add the group. Row 5 begins describing user U of the system, with one user record per line. Row 7 and row 8 describe the basic information of the schema S of the system. Row 9 begins to describe the field information FI within the schema. Users can continue to extend the table to add a new model or field. The symbol “+” in the table means supporting extension. In fact, for the entire platform, each content followed by “+” can be extended indefinitely as long as the physical space is sufficient.

- **Data Exchange Table:** Table 2 shows the format of the data exchange table, which provides a detailed description of structured data D in a DMS. In each data exchange table, we can inject only data items within the same data structure. The first row of each data exchange table is a description of the data structure, the other rows represent a piece of the data record that conforms to the data structure, and each column corresponds to a field.

Table 1. System Metadata and Format Constraints.

1	T	id	password	systemname		
2	+					
3	G	groupid				
4	+					
5	U	userid	username	userpwd	subG	
6	+					
7	S	schemaid	groupid	entry	gpermission	opermission
8	+					
9	FI	ftype	fname	F _{oa}		
10	+					

Table 2. Data Exchange Table.

	f ₁	f ₂	...	f _k
1	v ₁₁	v ₁₂	...	v _{1k}
2	v ₂₁	v ₂₂	...	v _{2k}
...	...	...	...	...
n	v _{n1}	v _{n2}	...	v _{nk}

6. Evaluation

In this section, we conducted an experiment to verify the feasibility of the proposed framework, through which we aim to answer the following three major questions: (1) Is the framework we proposed easy to learn? (2) Can the platform we have implemented quickly produce a data management system? (3) Can the method we designed improve the efficiency of the production data management system?

To evaluate the proposed system and methods, we first recruited 12 subjects to conduct the experiment, where the ages of the subjects were between 20 and 25 years old, with 3 years or more of programming experience. Among the 12 subjects, ten of them were proficient in JAVA, 6 of them were proficient in Python, and 2 of them were proficient in C/ C++. On the other hand, 10 of the 12 subjects had mastered MySQL and 2 of them had mastered MongoDB. Moreover, all of the 12 subjects had no experience in completing a data management system independently.

In the process of selecting subjects, we randomly recruited subjects who mastered different programming languages and database skills, so as to eliminate the threats of different programming languages and database skills on results. We did not recruit people without relevant knowledge as subjects, because it's obvious that they can use our framework faster than relearning relevant knowledge. In order to reduce the deviation in time cost, we only counted the effective working hours of subjects, but did not completely eliminate the slight deviation of subjects in this process. Because different data management systems have different requirements, we assigned two tasks with different requirements to eliminate the threats of this factor. However, subjects all had more than 3 years of programming experience, so we could not eliminate the threats of programming experience on results. Also, because none of subjects had any experience in developing a data management system independently, we could not eliminate the threat of development experience. In addition, while subjects all had programming experience, since no additional programming knowledge is required in our framework, it is also applicable to those without programming experience.

Since all the subjects had more than 3 years of programming experience, instead of teaching them separately, we designed an experiment by using a simple sample document to obtain a quick explanation about how the subjects familiarize with the framework platform. At the same time, we measured how long they were familiar with the framework. The results of this experiment are shown in Figure 7a, from which we can see that the subjects took less than one hour to become familiar with the framework, which indicates that our proposed method is easy to start working with because the time cost of learning it is not high.

On the other hand, to evaluate the efficiency of the subjects in using both online and offline spreadsheet production methods, we assigned two tasks to all subjects and customized the requirements of the tasks in detail, including the use of case diagrams and data structures. In dealing with the task 1, we asked subjects to complete a warehouse management system and to use an online production method. Some specific requirements are as follows:

- The system covers at least two types of users with privileges. There should be at least an administrator of the management system and users who can enter, modify and query data.
- The system should at least cover the form of equipment information, equipment outbound, equipment warehousing, equipment inventory, equipment purchase requirements, etc., which can clearly represent the information related to warehouse equipment.

In dealing with the task 2, we asked subjects to complete a student management system and to produce the system using an offline spreadsheet. The specific requirements are as follows:

- The system should cover at least four major modules, including teachers, students, classes and courses.
- The system should cover at least three users with different privileges, such as administrators, teachers and students. At the same time, users with different permissions in the system should have clearly divided operation permissions for different data.
- The system should at least be able to manage the information of students, teachers and classes, as well as course meta-information and information related to the course, such as the students enrolled in a certain course, their scores and attendance.
- The system should be able to perform statistical analysis and multi-condition combination analysis, for example, calculate the average score of students who are absent more than one time in a certain course.

The experimental results are shown in Figure 7b, from which we can see that all subjects completed task 1 in less than 60 min, and more than half of them completed task 1 in less than 30 min. In addition, all subjects in task 2, which was completed using an offline spreadsheet, completed the task in less than 30 min, and half completed the task in less than 15 min.

Figure 7. Evaluation Results. (a) All of the participants were familiar with the system in less than 60 min. (b) All participants completed the task in less than 60 min, which was further shortened by using an offline spreadsheet to complete task2.

7. Discussion and Conclusions

In this paper, we proposed a coding-free construction of a data management system based on SaaS architecture to deal with the high complexity problem of software development. We conducted an experiment to show the effectiveness and feasibility of our framework. Specifically, our experimental results show that the proposed framework is easy to start working with, requires no manual programming and extra development costs and can quickly integrate multiple databases and files in different formats. Moreover, we also proposed a spreadsheet-driven production approach that expands the applicable users, and reduces additional labour costs compared to the traditional development method. The experimental results indicate that the offline spreadsheet method can improve the efficiency of the production system and reduce the time cost. In addition, we also show that, compared to online production on the SaaS platform, the proposed platform can further improve the productivity of users.

Author Contributions: C.Y. proposed the methodology as well as did software architecture, system analysis, and validation. S.P. did the software design, programming, validation as well as the draft writing and editing. R.L. did the software component development and validation. Y.L. did validation and data migration. L.P. did the design and implementation of the system. All authors have read and agree to the published version of the manuscript.

Funding: This research was partially funded by the National Natural Science Foundation of China (NSFC) grant number U1611461, Guangzhou Science and Technology Program key project grant number 201704030124, and the Science and Technology Plan Project of Guangdong grant number 2014B010115002.

Acknowledgments: The authors thank Kemin Qu, the editors and reviewers to improve the paper, as well as the volunteers of taking part in our empirical experiments.

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

References

- Chen, S.L.; Chen, Y.Y.; Hsu, C. A new approach to integrate internet-of-things and software-as-a-service model for logistic systems: A case study. *Sensors* **2014**, *14*, 6144–6164. [\[CrossRef\]](#) [\[PubMed\]](#)
- Onokoy, L.; Lavendels, J. Evolution and Development Prospects of Information System Design Methodologies. *Appl. Comput. Syst.* **2018**, *23*, 63–68. [\[CrossRef\]](#)
- Namoun, A.; Tobias, N.; De Angeli, A. Service composition for non-programmers: Prospects, problems, and design recommendations. In Proceedings of the 2010 Eighth IEEE European Conference on Web Services, Ayia Napa, Cyprus, 1–3 December 2010; pp. 123–130.
- Maćkowiak, M.; Nawrocki, J.; Ochodek, M. On some end-user programming constructs and their understandability. *J. Syst. Softw.* **2018**, *142*, 206–222. [\[CrossRef\]](#)
- Park, S.; Hwang, M.; Lee, S.; Park, Y. A generic software development process refined from best practices for cloud computing. *Sustainability* **2015**, *7*, 5321–5344. [\[CrossRef\]](#)

6. Preuveneers, D.; Heyman, T.; Berbers, Y.; Joosen, W. Systematic scalability assessment for feature oriented multi-tenant services. *J. Syst. Softw.* **2016**, *116*, 162–176. [[CrossRef](#)]
7. Aleem, S.; Rabia, B.; Faheem, A.; Asad, K.; Raja, M.U.U. Architecture guidelines for SaaS development process. In Proceedings of the 2017 International Conference on Cloud and Big Data Computing, London, UK, 17–19 September 2017; pp. 94–99.
8. Makki, M.; Van Landuyt, D.; Lagaisse, B.; Joosen, W. A comparative study of workflow customization strategies: Quality implications for multi-tenant SaaS. *J. Syst. Softw.* **2018**, *144*, 423–438. [[CrossRef](#)]
9. Luo, Y.; Zhou, S.; Guan, J. LAYER: A cost-efficient mechanism to support multi-tenant database as a service in cloud. *J. Syst. Softw.* **2015**, *101*, 86–96. [[CrossRef](#)]
10. Tsai, W.T.; Huang, Y.; Shao, Q. EasySaaS: A SaaS development framework. In Proceedings of the 2011 IEEE International Conference on Service-Oriented Computing and Applications(SOCA), Irvine, CA, USA, 12–14 December 2011; pp. 1–4. [[CrossRef](#)]
11. Tsai, W.T.; Li, W.; Esmaeili, B.; Wu, W. Model-driven tenant development for PaaS-based SaaS. In Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, Taipei, Taiwan, 3–6 December 2012; pp. 821–826. [[CrossRef](#)]
12. Tsai, W.T.; Sun, X. SaaS Multi-Tenant Application Customization. In Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, Redwood City, CA, USA, 25–28 March 2013; pp. 1–12. [[CrossRef](#)]
13. Zhang, B.; Yu, L.; Feng, Y.; Liu, L.; Zhao, S. Application of Workflow Technology for Big Data Analysis Service. *Appl. Sci.* **2018**, *8*, 591. [[CrossRef](#)]
14. Xu, B.; Xu, L.D.; Fei, X.; Jiang, L.; Cai, H.; Wang, S. A method of demand-driven and data-centric Web service configuration for flexible business process implementation. *Enterp. Inf. Syst.* **2016**, *11*, 988–1004. [[CrossRef](#)]
15. Shao, T.; Xu, Y.; Li, J. SaaS BI for Chinese SMEs: Case study on Zhongli intellectual technology. In Proceedings of the 2018 International Conference on Information Management & Management Science, Chengdu, China, 25–27 August 2018; pp. 38–41.
16. Benson, E.; Zhang, A.X.; Karger, D.R. Spreadsheet-driven web applications. In Proceedings of the 27th annual ACM symposium on User interface software and technology ACM, Honolulu, HI, USA, 5–8, October 2014; pp. 97–106. [[CrossRef](#)]
17. Mohamed, F.; Mizouni, R.; Abu-Matar, M.; Al-Qutayri, M.; Whittle, J. An Integrated Platform for Dynamic Adaptation of Multi-Tenant Single Instance SaaS Applications. In Proceedings of the 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud) IEEE, Prague, Czech Republic, 21–23 August 2017; pp. 257–264. [[CrossRef](#)]
18. Sharma, R.; Sood, M. Enhancing Cloud SAAS Development with Model Driven Architecture. *Int. J. Cloud Comput. Serv. Arch. (IJCCSA)* **2011**, *1*, 89–102. [[CrossRef](#)]
19. Cai, H.; Gu, Y.; Vasilakos, A.V.; Xu, B.; Zhou, J. Model-driven development patterns for mobile services in cloud of things. *IEEE Trans. Cloud Comput.* **2018**, *6*, 771–784. [[CrossRef](#)]
20. Ma, K.; Yang, B.; Abraham, A. A template-based model transformation approach for deriving multi-tenant saas applications. *Acta Polytechnica Hungarica* **2012**, *9*, 25–41.
21. Mietzner, R.; Leymann, F.; Unger, T. Horizontal and vertical combination of multi-tenancy patterns in service-oriented applications. *Enterp. Inf. Syst.* **2010**, *5*, 59–77. [[CrossRef](#)]
22. Scheibler, T.; Mietzner, R.; Leymann, F. EMod: Platform independent modelling, description and enactment of parameterisable EAI patterns. *Enterp. Inf. Syst.* **2009**, *3*, 299–317. [[CrossRef](#)]
23. Ying, L.; Bin, Z.; Guoqi, L.; Deshuai, W.; Yan, G. Personalized Modeling for SaaS Based on Extended WSCL. In Proceedings of the 2010 IEEE Asia-Pacific Services Computing Conference IEEE, Hangzhou, China, 6–10 Deceber 2010; pp. 355–362. [[CrossRef](#)]
24. Zhang, Y.C.; Zhang, B.; Liu, Y. Research and Implementation of Multi-tenancy Business Customization System based on SBTM Model. *J. Chin. Comput. Syst.* **2013**, *34*, 127–133. [[CrossRef](#)]
25. Zhu, X.; Wang, S. Software customization based on model-driven architecture over SaaS platforms. In Proceedings of the 2009 International Conference on Management and Service Science IEEE, Wuhan, China, 20–22 September 2009; pp. 1–4. [[CrossRef](#)]
26. Zhang, X.; He, K.; Wang, J.; Liu, J.; Wang, C.; Lu, H. On-Demand Service-Oriented MDA Approach for SaaS and Enterprise Mashup Application Development. In Proceedings of the 2012 International Conference on Cloud and Service Computing IEEE, Shanghai, China, 22–24 November, 2012; pp. 96–103. [[CrossRef](#)]

27. Etedali, A.; Lung, C.H.; Ajila, S.; Veselinovic, I. Automated Constraint-Based Multi-tenant SaaS Configuration Support Using XML Filtering Techniques. In Proceedings of the 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC) IEEE, Torino, Italy, 4–8 July, 2017; pp. 413–418. [\[CrossRef\]](#)
28. Jiang, X.Y.; Zhang, Y.; Liu, S.J.; Meng, X.X. Layer Multi-tenants Architecture Based on Component and Metadata. *J. Chin. Comput. Syst.* **2013**, *34*, 2724–2728. [\[CrossRef\]](#)
29. Hoang, D.D.; Paik, H.Y.; Dong, W. Mashsheet: Mashups in your spreadsheet. In *International Conference on Web Information Systems Engineering*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 332–333. [\[CrossRef\]](#)
30. Chang, K.S.P.; Myers, B.A. Creating interactive web data applications with spreadsheets. In Proceedings of the 27th annual ACM symposium on User interface software and technology ACM, Honolulu, HI, USA, 5–8 October, 2014; pp. 87–96. [\[CrossRef\]](#)
31. Kongdenfha, W.; Benatallah, B.; Vayssire, J.; Saint-Paul, R.; Casati, F. Rapid development of spreadsheet-based web mashups. In Proceedings of the 18th international conference on World wide web ACM, Madrid, Spain, 20–24 April 2009; pp. 851–860. [\[CrossRef\]](#)
32. Hoang, D.D.; Paik, H.Y.; Ngu, A.H. Spreadsheet as a generic purpose mashup development environment. In *International Conference on Service-Oriented Computing*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 273–287. [\[CrossRef\]](#)
33. Honkisz, K.; Kluza, K.; Winiewski, P. A Concept for Generating Business Process Models from Natural Language Description. In *International Conference on Knowledge Science, Engineering and Management*; Springer: Cham, Switzerland, 2018; pp. 91–103.
34. Winiewski, P.; Kluza, K.; Kucharska, E.; Ligza, A. Spreadsheets as Interoperability Solution for Business Process Representation. *Appl. Sci.* **2019**, *9*, 345. [\[CrossRef\]](#)
35. Kurtz, B.L.; Fenwick, J.B., Jr.; Meznar, P. Developing microlabs using Google web toolkit. In Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, Raleigh, NC, USA, 29 February–3 March 2012; ACM: New York, NY, USA, 2012; pp. 607–612.

 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).