Available online at www.sciencedirect.com

ScienceDirect

CAAI Transactions on Intelligence Technology 1 (2016) 150–161
http://www.journals.elsevier.com/caai-transactions-on-intelligence-technology/

Original Article

Tenant-centric Sub-Tenancy Architecture in Software-as-a-Service

Wei-Tek Tsai*, Peide Zhong, Yinong Chen

School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287, USA

Available online 13 October 2016

Abstract

Multi-tenancy architecture (MTA) is often used in Software-as-a-Service (SaaS) and the central idea is that multiple tenant applications can be developed using components stored in the SaaS infrastructure. Recently, MTA has been extended to allow a tenant application to have its own sub-tenants, where the tenant application acts like a SaaS infrastructure. In other words, MTA is extended to STA (Sub-Tenancy Architecture). In STA, each tenant application needs not only to develop its own functionalities, but also to prepare an infrastructure to allow its sub-tenants to develop customized applications. This paper applies Crowdsourcing as the core to STA component in the development life cycle. In addition, to discovering adequate fit tenant developers or components to help build and compose new components, dynamic and static ranking models are proposed. Furthermore, rank computation architecture is presented to deal with the case when the number of tenants and components becomes huge. Finally, experiments are performed to demonstrate that the ranking models and the rank computation architecture work as design.

Keywords: SaaS; MTA; STA; Tenant; Sub-tenant; Crowdsourcing; Ranking

1. Introduction

Cloud platforms often have three main components: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). SaaS is the software deployed over the internet [1], where users subscribe services from SaaS providers and pay by a way of "pay-as-you-go". In SaaS, software is maintained and updated on a cloud, and presented to the end users as services on demand. Multi-Tenancy Architecture (MTA) of SaaS allows tenant developers to develop applications using the same code that is based stored in the SaaS infrastructure. MTA is often integrated with databases and supports tenant application customization by composition of existing or new software components stored in the SaaS or supplied by tenant developers.

However, current MTA has the following limitations:

1) While a SaaS infrastructure support tenant applications using services and data stored in the infrastructure, a tenant application does not allow its users to use its own services or data to develop new applications.
2) It is difficult for a tenant application to share service or data with other tenant applications. Often, a SaaS platform provides security mechanisms to isolate tenant applications so that tenants cannot access data that belong to other tenants. Even though tenant code and data are stored in the same database, the SaaS security mechanism isolates a tenant from other tenants.
3) Most SaaS systems do not support tenants to customize their applications already customized by other tenants.

To address those issues, Tsai in [2] introduced a STA (Sub-Tenancy Architecture) to allow tenants to offer services for sub-tenant developers for customizing their applications. As SaaS component building often needs different technologies such as frontend UI and database, the tenant or sub-tenant developers are often not good at those technologies. Therefore, it can be difficult for them to build SaaS components from the scratch. Hence, this paper introduces Crowdsourcing

* Corresponding author.
E-mail addresses: wtsai@asu.edu (W.T. Tsai), Peide.Zhong@asu.edu (P. Zhong), Yinong.Chen@asu.edu (Y. Chen).
Peer review under responsibility of Chongqing University of Technology.

http://dx.doi.org/10.1016/j.trit.2016.08.002

to make use of the public wisdom and assign tasks to specific experts who are good at those required technologies. To help find adequate tenants, we developed models in this paper. The rest of the paper is organized as follows. Section 2 reviews related SaaS models and technologies; Section 3 analyzes life cycles of tenant-centric application development; Section 4 introduces component and tenant rank; Section 5 presents feature implementation selection model; Section 6 describes rapid application building process. Section 7 presents the experiment that illustrates the rank models, and Section 8 concludes the paper.

2. Related work

2.1. MTA in SaaS

In the current practice, MTA are implemented via the following ways:

1) Integration with Databases: Weissman and Bobrowski proposed a database-based and metadata-driven architecture to implement MTA in Ref. [3]. In Ref. [3], where heavy indexing was used to improve the performance, and a runtime application generator is used to dynamically build applications in response to specific user requests. As all tenants shared the same database, a flexible schema design was used. Aulbach [4] developed five techniques for implementing flexible schemas for SaaS.
2) Middleware Approach: In this approach, an application request is sent to a middleware that passes the request to databases behind the middleware. As all databases are behind the middleware and all application requests to databases are managed and directed by the middleware, the applications can be transformed into a MTA SaaS rapidly with minimum changes to the original applications. Cai [5] described a transparent approach of making existing Web applications to support MTA and run in a public cloud.
3) Service-oriented SaaS: This is an approach to implement MTA by SOA (Service-Oriented Architecture) [6]. SaaS domain knowledge is separated from SaaS infrastructure to facilitate different domains. EasySaaS [7] proposed a development framework to simplify SaaS development by harnessing both SOA and SaaS domain ontology. Azeez [8] proposed an architecture for achieving service-oriented MTA that enabled users to run their services and other SOA artifacts in a MTA service-oriented framework as well as provided an environment to build MTA applications. As this MTA is based on SOA, it can harness both middleware and SOA technology.
4) PaaS-based approach: The SaaS developers use an existing PaaS such as GAE [9], Amazon EC2 [10], or Microsoft Azure [11] to develop SaaS applications. In this approach, developers use the MTA features provided by a PaaS to develop SaaS applications, and most of SaaS features such as code generation, and database access are implemented by the PaaS. Tsai [12] proposed a model-driven approach on a PaaS to develop SaaS.
5) Object-oriented approach: Workday [13] proposed an object-oriented approach for tenant application development and configuration. In addition [13], also conducted a study on MTA models, specifically it addressed the architecture of MTA and its impact on customization, scalability, and security.

2.2. Crowdsourcing

The purpose of Crowdsourcing is to make use of public wisdom and let the crowd with domain knowledge to complete specific tasks. Howe first defined the term “crowdsourcing” in a companion blog post [14]. Merriam-Webster [15] defines Crowdsourcing as the practice of obtaining needed services, ideas, or content by soliciting contributions from a large group of people, and especially from an online community, rather than from the traditional employees or suppliers. Kittur in [16] investigated the utility of a micro-task market for collecting user measurements, and discussed design considerations for developing remote micro user evaluation tasks. Peng in [17] provided an overview of current technologies for crowdsourcing.

2.3. Variation point

Variation points are locations where variation occurs, and variants are the alternatives that can be selected. Software product families introduce variability management to deal with these differences by handling variability. Kang [18] described a method for discovering commonality among different software systems. Coplien [19] presented how to perform domain engineering by identifying the commonalities and variabilities within a family of products. Webber [20] described a systematic method for providing components that could be extended through variation points, which allowed the user or application engineer to extend components at pre-specified variation points to create more flexible set of components. Mietzner [21] presented a variability descriptor and described that its transformation into a WS-BPEL process model to guide customizations. In addition, Mietzner [22] explained how variability modeling techniques could support SaaS providers in managing the variability of SaaS applications and proposed using explicit variability models to derive customization for individual SaaS tenants.

2.4. Customization in SaaS

Customization is an important SaaS feature as tenants may have different business logic and interface yet they share the same code base. Chong [23] proposed a SaaS maturity model that classifies SaaS into four levels including ad-hoc/custom, customizable or configurable, multi-tenant efficient, and scalable. Tsai introduced ontology into SaaS to help customize applications [24]. A SaaS tenant application has components