一种带租户演化容忍度的SaaS服务演化一致性判定方法

王晓芳[1] 谢仲文[1,2] 李 彤[2] 成 蕾[1] 郑交交[1] 刘晓芳[1]
(云南大学软件学院 昆明 650091)[1] (云南大学软件工程重点实验室 昆明 650091)[2]

摘 要 随着云技术的不断发展与成熟,软件即服务(SaaS)模式成为未来软件应用发展的主要趋势。在多元开放的网络生态环境中,SaaS服务若要有效应对用户需求及外部变化,就须具备演化能力。演化一致性是指服务在演化后能保有原基础及与其他服务正常交互的能力。目前对演化一致性的判定多偏向于定性分析,且往往忽略了租户的感受,没有既定的显式标准对一致性进行定量度量并判定。针对此问题,从SaaS多租户单实例的应用模式出发,分层次细粒度地建立服务实例描述模型,引入一致性度量值来表示定量计算的结果,充分考虑租户的演化要求,提出一种带租户演化容忍度的判定方法,细粒度地判定演化一致性。最后,结合SaaS应用案例,采用所提方法对演化一致性进行分析判定,实际应用的反馈情况验证了该方法的可行性和有效性。

关键词 租户演化容忍度,服务演化,演化一致性,定量计算
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.05.025

Saas Service Evolution Consistency Checking with Tenant Tolerance

WANG Xiao-fang[1] XIE Zhong-wen[1,2] LI Tong[2] CHENG Lei[1] ZHENG Jiao-jiao[1] LIU Xiao-fang[1]
(College of Software, Yunnan University, Kunming 650091, China)[1]
(Key Laboratory for Software Engineering of Yunnan Province, Yunnan University, Kunming 650091, China)[2]

Abstract With the development and maturity of cloud computing technology, software as a service (SaaS) has become the main trend of software application. In multivariate open network of ecological environment, in order to effectively respond to user needs and external environment, SaaS service must possess evolution capacity. Being short of uniform standard and having no established explicit standard to quantitatively measure, the judgment of the consistency checking often overlooks the feel of tenant. In response to these shortages, this paper focused on tenant demand which is the core of SaaS, proposed a description model for SaaS service in aspects of structural layer and non-functional layer. Based on the model, it took full account of the evolutionary needs of tenants and introduced the extent of evolution consistency to analyze evolution consistency quantitatively. After that, it proposed three-tier method of consistency checking. At last, this method was used to judge the evolution consistency combining with SaaS application case, and the feasibility and validity of the method were verified according to the actual application feedback.

Keywords Tenant tolerance, Service evolution, Evolution consistency, Quantitative calculation

1 引言

面向服务计算(Service-Oriented Computing, SOC)是一种新型计算模式,它将服务当作基本组件,使服务可以低成本、快速、简单地分布,甚至在异构环境下组合[1]。随着面向服务计算的不断发展和互联网技术的逐渐成熟,软件即服务(Software as a Service, SaaS)作为一种完全创新的软件应用模式被提出[2]。近年来,SaaS作为云计算的主要服务提供模式,成为未来软件应用发展的主要趋势,SaaS应用正给软件工业带来一场革命。

软件演化是软件不断更新变化的过程,是软件的本质特征之一[3]。同时,SaaS服务处于不断变化的环境中,若要有效应对用户需求和外部环境的变化,必须具备演化的能力。学界一般用演化一致性[4]来定义服务在演化后还保持有与原服务基础交互及与其他服务或其他租户交互的能力。目前,SaaS演化应用研究中没有相对完整的规范化体系标准,且大多关注SaaS演化的实现,对演化一致性的讨论较为欠缺[5-8]。但是,一致性的保持是保证演化实施可靠性的重要条件,要建立SaaS演化的规范体系,就必须对一致性进行讨论。

现有SaaS服务大多建立在面向服务架构(Service Orien-

到稿日期:2017-03-07 返修日期:2017-06-05 本文受国家自然科学基金项目(61379032,61662085)资助。
王晓芳(1993-),女,硕士生,主要研究方向为领域软件工程,E-mail:1922657476@qq.com;**谢仲文**(1982-),男,博士,讲师,CCF会员,主要研究方向为软件工程、软件过程、软件演化等,E-mail:xiezw56@126.com(通信作者);**李 彤**(1963-),男,博士,教授,CCF会员,主要研究方向为软件工程、软件过程、软件演化等;**成 蕾**(1993-),女,硕士生,主要研究方向为软件工程理论与方法;**郑交交**(1992-),女,硕士生,主要研究方向为系统分析与集成;**刘晓芳**(1991-),女,硕士生,主要研究方向为领域软件工程。

ted Architecture,SOA)的基础上,指以服务组合形成的 SaaS 应用[2]。相比于一般 Web 服务组合,此种 SaaS 应用强调多租户、单实例、可定制的特性。文献[5]将 SaaS 应用与传统软件进行区别,归纳了 SaaS 应用的特点,并根据其特点将演化决策因素分类为服务本身驱动、管理员驱动和租户需求驱动 3 类。可见,研究 SaaS 服务演化一致性时必须考虑租户的演化容忍情况。

现有针对 SaaS 演化一致性的研究多为定性分析,且往往忽略租户感受,没有形成统一的标准来对演化一致性进行定量计算并判定[2,6-10]。国内外对单个 Web 服务演化一致性的研究已有显著成效,主要集中于子型理论与近似树匹配[11-16]两个方面。现有 SaaS 服务的实质是基于 SOA 架构的服务组合[17-20],因此单个服务的研究成果可对其研究提供帮助。通过综合分析发现,现有的方法主要存在以下不足:1)没有适用于 SaaS 服务的描述模型;2)现有研究成果可反映 Web 服务版本间的变化,却没有可清晰反映服务组合在演化过程中的变化的方法;3)现有研究对 SaaS 服务演化一致性的讨论仅仅停留在定性分析的层面,且多关注流程变更问题,并没有真正讨论演化的一致性和正确性[21-23];4)对 SaaS 服务一致性的定性分析过于粗略,并没有细致讨论一致性不能满足的原因,没有实际的可操作标准;5)讨论演化一致性往往忽略了租户这一核心因素;6)在一致性判定方面,大多方法只提供定性分析,个别方法虽然提出了定量计算,但只适用于单个 Web 服务,目前没有针对 SaaS 服务一致性的定量计算方法。

针对上述问题,本文结合文献[11,24],使用扩展的服务描述性语言 WSDL(Web Service Description Language)对 SaaS 应用服务实例和服务实例变迁进行建模,引入演化一致性度量值来解决演化一致性的量化和判定问题。首先,提出基于实例变迁的服务实例描述模型,并给出相应的层次性变化抽取方法;其次,基于抽取出的变化序列,引入一致性度量值的概念,提出租户参与的分层次细粒度的一致性定量计算方法;最后,参考租户自定义演化容忍度系数,对 SaaS 服务演化一致性进行分层次和综合性判定。

本文第 2 节讨论 SaaS 服务演化一致性判定的过程;第 3 节给出描述模型及其变化抽取方法;第 4 节给出一致性定量计算方法及一致性判定方法;第 5 节结合 SaaS 应用实例来验证一致性判定方法的正确性。

2 SaaS 服务演化一致性判定的过程

2.1 SaaS 服务演化的一般过程

在 SaaS 应用中,软件服务提供商为每一位客户制定并部署软件,多租户单实例(Multi-Tenant Single Instance)应用架构是满足此模式的真正意义上的 SaaS 应用架构。在多租户的环境中,应用运行在相同的一个或者一组服务器上,被称为“单实例”架构[2]。现有的 SaaS 应用是以 SOA 为技术平台框架,由单个 Web 服务组合而成的,此 Web 服务组合和一般服务组合并不完全相同,主要突出表现 SaaS 的特性,重点关注服务的内部结构,而非一般 Web 服务组合关注的服务聚合问题。SaaS 服务演化的实质是“单实例”变化,通常会产生新的“实例”,形成新的 SaaS 应用,即 SaaS 服务演化过程。SaaS 服务演化示例如图 1 所示。

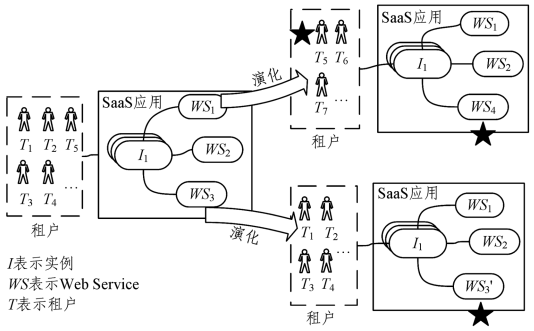


图 1 SaaS 服务演化过程示例

Fig. 1 Example of SaaS service evolution process

图 1 中,租户 T_1,T_2,T_3,T_4 接受实例 I_1 到 I_3 的迁移,适用于此种过程的 SaaS 应用演化;而 T_5 属于定制租户,不接受此种演化,考虑到租户 T_5 的需求,实例由 I_1 到 I_2 迁移,实现另一过程的 SaaS 应用演化。由图 1 可知,SaaS 应用中的服务组合强调多租户、单实例、可定制的特性,SaaS 服务的演化需要考虑租户因素。在演化过程中,称 I_2 和 I_3 为目标实例,称 I_1 为源实例。根据图 1 可知,SaaS 服务演化中的实例变迁是前后相继的:一个目标实例只会对应一个源实例(即使考虑到租户的演化需求,也只会存在一个目标实例有且只能对应一个源实例);同一个源实例对照不同的演化过程,会有不同的目标实例,而在同一演化过程中不会产生一个源实例对应多个目标实例或多个源实例对应一个目标实例的情况。

2.2 一致性判定过程

演化一致性应该包含两条基本性质:演化后的服务与环境能够正常交互;演化后的服务功能能够满足租户需求。对于定制租户,服务演化可能会涉及所定制服务的使用性,因此租户会参与服务演化的一致性分析过程,确保演化后服务还能继续满足定制租户的特殊需求。一般地,对服务实例内容的描述可从功能层次和非功能层次两方面进行[1]。功能层次又可从结构和行为两方面来描述,但由于服务行为主要是内部的业务流程和逻辑,对演化一致性具有深层次、不明显的影响[11],因此本文主要关注对 SaaS 服务演化一致性有显著影响的服务结构层次和非功能层次。本文所提出的演化一致性判定的过程如图 2 所示。

该过程主要分为以下 4 个部分。

1)服务实例和服务变迁的描述建模。要讨论服务演化一致性,就需要描述出服务实例。本文参照文献[9]的方法建立 SaaS 服务实例描述模型,并在此基础上建立服务实例变迁描述模型(具体参见 3.1 节和 3.2 节)。

2)实例变迁中的变化抽取。变化是引起演化的本质原因。为便于抽取变化向量,本文将服务实例描述模型建立为适合文本的结构,并基于建立的描述模型,通过变化抽取算法对变化向量进行抽取。

3)一致性度量值的定量计算。综合考虑租户的演化要求,通过引入服务演化前后保持一致性的程度,即演化一致性度量值,对 SaaS 服务演化一致性进行细致的定量计算。为便于后续一致性的判定,将一致性度量值归一化到[0.0,1.0]的范围内。

4)服务演化一致性判定。对于一致性的判定,租户起着关键性作用。本节结合租户的演化容忍要求,设定判定阈值,

将分别从结构层、非功能层和综合一致性 3 个方面给出一致性判定结果，以达到分层次、细粒度、全方位的一致性判定。

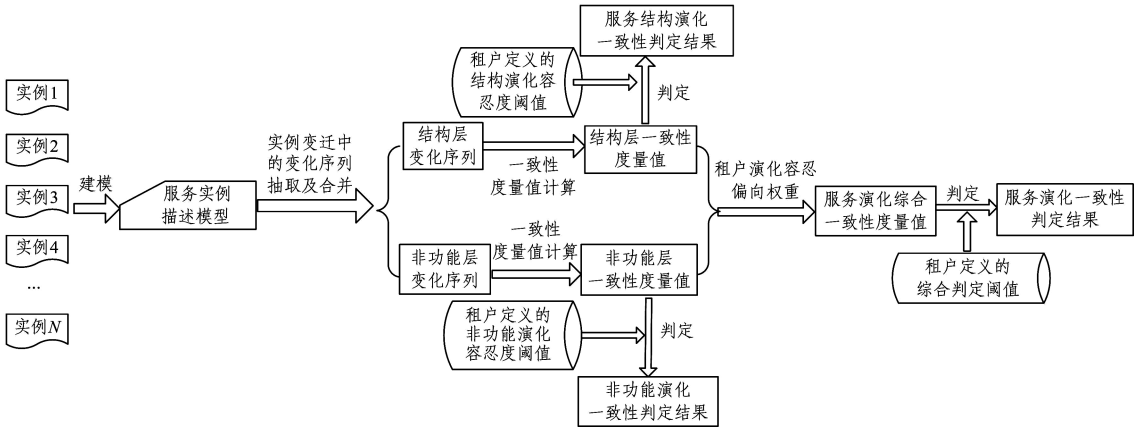


图 2　SaaS 服务演化一致性判定的过程

Fig. 2　Process of service evolution consistency checking on SaaS model

3　基于实例变迁的服务实例描述模型及变化抽取

为实现分层次细粒度的演化一致性判定，本节从结构层和非功能层建立服务实例描述模型，并给出层次性的变化向量抽取方法。

3.1　服务实例的描述模型

本文所提及的 SaaS 应用是以 SOA 为技术平台的服务组合，而服务由组件构成，组件又是若干操作的集合，操作对应着具体的程序函数，因此对 WSDL 和 QoS 标准进行适当扩展，定义符合 SaaS 应用的服务实例描述模型。

3.1.1　服务实例结构层描述模型

本文参照文献[11]，提出了适用于 SaaS 应用的服务实例描述模型。基于文献[11]对单个 Web 服务的描述，本文提出的描述模型侧重于服务实例及实例下的服务组合，单个 Web 服务仅作为整体结构的一部分。整个实例描述模型严格按照层次性的逻辑结构建立，实例结构层描述模型的定义如下。

定义 1（服务实例结构）　服务实例结构是整个 SaaS 服务的实例结构，用九元组 $Structure=\langle Case, Service, Port, PortType, Binding, Operation, Message, Part, DataType\rangle$ 表示。其中，*Case* 表示服务实例；*Service* 表示 Web 服务；*Port* 表示接口；*PortType* 表示接口类型；*Binding* 表示绑定；*Operation* 表示操作；*Part* 表示参数；*DataType* 表示数据类型。

定义 2（服务实例）　服务实例是单个实例节点，用三元组 $Case=\langle name, depict, Service\rangle$ 表示。其中，*name* 表示实例的名称；*depict* 表示实例的描述；*Service* 表示服务实例中包含的服务集合。

其中，*Service*，*Port*，*PortType*，*Binding*，*Operation*，*Message*，*Part*，*DataType* 的定义在文献[11]中已给出，此处不再赘述。

上述方法定义的服务实例是符合文本结构的，在此描述模型中，服务实例可以自然构建符合层次结构逻辑的服务实例结构树，如图 3 所示。

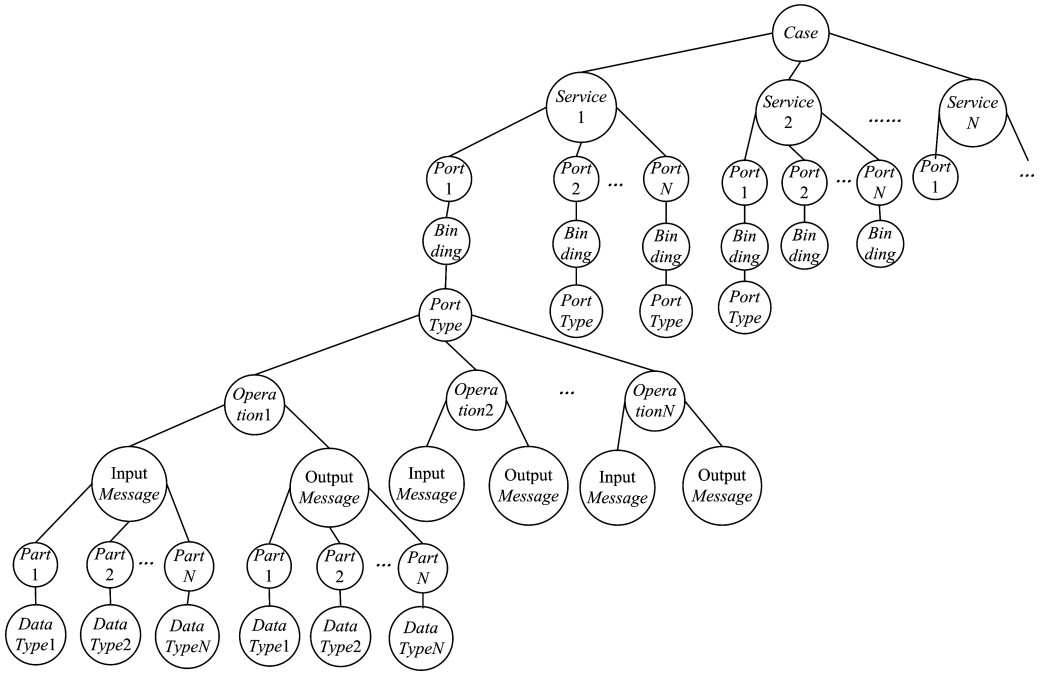


图 3　服务实例结构树

Fig. 3　Structure tree of service instance

3.1.2 服务实例非功能层描述模型

根据文献[24]中对 Web 服务组合的 QoS 评价,结合现有研究成果,对照结构层描述模型,建立以下非功能层描述模型。

定义 3(服务非功能层) 服务非功能层表示为服务实例的服务质量总和,即 $SUM\_Q=\langle QoS\rangle$。

定义 4(服务质量) 服务质量 $QoS=\langle depict, i\_name, i\_type, i\_value\rangle$。其中,*depict* 表示对 QoS 的描述;*i_name* 表示指标名称;*i_type* 表示指标属性,有正比属性(monotonic)和反比属性(antitonic)两类,QoS 的关键指标主要包括可用性、安全性、时延变化和费用等;*i_value* 表示对应的指标值。

3.2 实例变迁描述模型

2.1 节中已指出 SaaS 服务变迁是前后相继的。由此,实例变迁可分解为 3 个部分:变化源、变化操作、变化目标。变化源是指原始的实例内容;变化操作包括 3 种操作:增加、删除、修改;变化目标是指演化后的实例内容。SaaS 服务演化的实质是变化源在变化操作的作用下变为变化目标,从而使得源实例变为目标实例,完成服务演化。实例变迁中的变化应该是有方向的,本文将变化方向统一规定为从源实例到目标实例。

定义 5(实例变迁的变化向量) 实例变迁的变化向量表示为一个四元组 $dv=\langle O, S, F, D\rangle$。其中,$S\in$ {structure, non-function},表示服务实例两个层级的变化;$O\in$ {add, del, mod}表示操作变化;F 表示实例的服务变化源;D 表示实例的服务变化目标。当 $O=$add 时,表示对 F 执行增加操作;当 $O=$del 时,表示对 F 执行删除操作,对应的 $D=\emptyset$。操作也可以定义为 3 种操作的组合。

基于上述对实例变迁的变化向量的定义,引入变化向量序列来描述从一个实例变迁到另一个实例的所有有序的变化过程。

定义 6(实例变迁的变化向量序列) 实例变迁的变化向量序列是指实例变迁中所有变化向量的有序排列组合,$\Delta L=(dv_1, dv_2, dv_3, \cdots, dv_n)$。若服务实例 C_1 对应的变化向量序列为 ΔL_1,那么演化后的服务实例 $C_2=C_1 * \Delta L_1$。

由于本文分层次讨论演化一致性问题,变化向量同样被分为结构层变化向量和非功能层变化向量,因此变化向量序列还可以表示为 $\Delta L=\Delta L_S+\Delta L_N$,其中 ΔL_S 表示结构层的变化向量序列,ΔL_N 表示非功能层的变化向量序列。

定义 7(基于实例变迁的服务实例描述模型) 服务实例 $C_n=\langle C_{n-1}, \Delta L\rangle$,当 $n=1$ 时,$C_0=C_1$。

3.3 变化向量的抽取方法

上文定义了描述模型,但若要表示服务演化前后的变化,则需要抽取出变化向量。本节将给出抽取相应变化向量的方法。

3.3.1 实例结构层变化向量的抽取方法

服务实例为层次树结构,结构间的层次关系通过父与子的关系节点体现,同层次关系通过兄弟节点间的关系体现,且层次关系与兄弟节点的顺序无关。节点的属性信息中保留了结构的语法信息。本文提出的变化向量抽取方法先通过节点之间的层次递进关系来确定变化路径以及变化对象,再通过比较该节点属性来确定变化的内容。值得注意的是,不同的变化内容对一致性的影响程度不同。现假设某一操作节点 $Operation=\langle O_1, request\text{-}response, Encoder, MI, MO\rangle$ 有两种演化情况:一种为演化后的 *Operation* 变成了 $Operation'=\langle O_2, request\text{-}response, Encoder, MI, MO\rangle$,其中操作名称由 O_1 变为 O_2,其余属性保持不变;另一种为演化后的 *Operation* 变成了 $Operation''=\langle O_1, solicit\text{-}response, Encoder, MI, MO\rangle$,其中操作的消息操作类型由 *request-response* 变成了 *solicit-response*。以上两种演化情况中,第一种对一致性的影响程度大于第二种,因为服务在运行编译过程中是按照名称来进行连接的,由此可知即使是同一节点的内容发生变化,不同的变化属性对演化一致性的影响也是不同的。参照文献[11]的处理办法,将变化后对一致性影响较大的属性定义为主属性,在上述例子中 name 就是一个主属性。对应于上文定义的变化操作,将主属性的修改定义为 add 和 del 操作,进而其他非主属性的修改定义为 mod 操作,制定表 1 所列规则。

表 1 变化操作的应用规则

Table 1 Application rules of change operation

源实例主属性	目标实例主属性	变化操作
不存在	存在	add
存在	不存在	del
存在	存在(其他属性不同)	mod

根据以上形式化定义的操作变化规则,给出本文采用的实例结构层变化向量抽取算法的具体描述,如算法 1 所示。

算法 1 实例结构层变化向量抽取算法

输入:源实例结构树 SourceTree,目标实例结构树 TargetTree

输出:服务实例变迁的结构层变化序列 ΔL_s

```
∅→ΔLs;
for each level_i ∈ structure{
∅→MatchedStore;
for each Node_j ∈ TargetTree
for each Node_k ∈ SourceTree
  if Node_j主属性=Node_k 主属性{
    if 其他属性没有修改
      MatchedStore∪(Node_j, Node_k)→MatchedStore;
    else 其他属性有修改
      ΔLs∪⟨mod, structure, F, D⟩→ΔLs;}
    for each Node_j ∈ TargetTree ∧ Node_j MatchedStore
      ΔLs∪⟨add, structure, F, D⟩→ΔLs;
    for each Node_k ∈ SourceTree ∧ Node_k MatchedStore
      ΔLs∪⟨del, structure, F, D⟩→ΔLs;}
```

3.3.2 非功能层变化向量的抽取方法

参照文献[24]的处理方法对 QoS 的各项指标进行统一化和无量纲化,对 SaaS 服务实例 QoS 进行组合评价。对照上文的结构层变量抽取方法,将非功能层 QoS 的每个指标都看作一个节点,根据表 1 的变化操作规则,统一将非功能层 QoS 指标名称定义为主属性,延续结构层的变化向量抽取方法,对非功能层采用如算法 2 所示的变量抽取算法。

算法 2 实例非功能层变化向量抽取算法

输入:源非功能描述 SourceQoS,目标非功能描述 TargetQoS

输出:实例变迁的非功能层变化向量序列 ΔL_N

```
∅→ΔLN;
∅→MatchedStore;
```

for each $QoS_i \in TargetQoS$
for each $QoS_j \in SourceQoS$
  if QoS_i 主属性＝QoS_j 主属性{
    if QoS_i 的值＝QoS_j 的值
    MatchedStore∪(QoS_i,QoS_j) →MatchedStore;
    if QoS_i 的值≠QoS_j 的值
    ΔL_N ∪〈mod,non-function,F,D〉→ΔL_N;}
for each $QoS_i \in TargetQoS \wedge QoS_i$ MatchedStore
    ΔL_N ∪〈add,non-function,F,D〉→ΔL_N;
for each $QoS_j \in SourceQoS \wedge QoS_j$ MatchedStore
  ΔL_N ∪〈del,non-function,F,D〉→ΔL_N;}

4　分层次细粒度的一致性定量计算及一致性判定方法

4.1　服务演化一致性定量计算方法

考虑到租户的定制需求，本节将从结构层和非功能层细粒度两个方面给出与其相适应的一致性定量计算方法，最后结合租户演化容忍偏向，采用加权平均的方法给出综合演化一致性度量值。

4.1.1　结构层一致性度量

根据 3.1.1 节中的服务实例结构层描述模型，将结构层的一致性深度细化为两个方面：从实例角度出发，讨论单实例下的服务组合变化对一致性的影响；从单个 Web 服务出发，讨论服务结构中操作、绑定和数据的变化对一致性的影响。最后考虑租户的应用需求，综合计算结构层的一致性度量值。

1）服务一致性度量值 *ScCE*（Service combination Consistency Extent）

从实例出发，本文采用服务一致性度量值 *ScCE* 来度量单实例下服务组合变化对一致性的影响，主要涉及服务增加、删除对一致性的影响。采用式(1)来量化影响的程度：

$$ScCE(\Delta L_s)=\frac{N-B}{N}-\frac{A}{N+A}-\frac{B}{N-B} \tag{1}$$

其中，ΔL_S 指服务实例的结构层变化序列，N 指源服务实例中的服务总数，B 指服务实例演化后减少的服务个数，A 指服务实例演化后增加的服务个数。由式(1)可以看出，服务的增加和减少对一致性的保持都是负影响的，其中服务减少对一致性的影响更为明显，服务增加对一致性的影响相对较小，这与实际 SaaS 应用演化的反馈情况相符。服务减少会造成原 SaaS 应用实例功能的不完善，在实际应用中对租户的影响更加明显；而服务增加对原 SaaS 应用功能的完整性并没有显著影响。

$$TypE(T_i,T_j)=\begin{cases}1, & T_i \text{ 和 } T_j \text{ 为同一类型并且数据精度相同}\\ 0, & T_i \text{ 和 } T_j \text{ 为同一类型并且 } L(T_i)>L(T_j)\\ 1-e\left(\dfrac{L(T_j)-L(T_i)}{LN(T_i)}\right), & T_i \text{ 和 } T_j \text{ 为同一类型并且 } L(T_i)<L(T_j)\\ 1-f\left(\dfrac{L(T_i)}{LN(T_i)}+\dfrac{L(T_j)}{LN(T_j)}\right), & T_i \text{ 和 } T_j \text{ 为不同类型}\end{cases} \tag{10}$$

$$DaCE(\Delta L_{mod})=\frac{\sum_{i=0}^{n} ParE(dv_i)}{n} \tag{11}$$

2）服务结构一致性度量值 *SsCE*（Service Structure Consistency Extent）

式(1)给出了实例下服务组合变化对结构层一致性的影响，下面从单个服务出发讨论单个服务结构修改对一致性的影响。本文用服务结构一致性度量值 *SsCE* 来表示单个服务结构修改对一致性的影响程度。结合文献[11]，给出 *SsCE* 的计算公式为：

$$SsCE(\Delta L_s)=\begin{cases}0, & \exists dv_i \in \Delta L_{add} \cap dv_i \in IntC \mid \exists dv_i \in \Delta L_{del} \cap dv_i \in OutC\\ OpCE(\Delta L_{mod}) * BdCE(\Delta L_{mod}) * DaCE(\Delta L_{mod}) & \end{cases} \tag{2}$$

其中，操作一致性度量值 *OpCE*（Operation Consistency Extent）是对单个服务操作变化的一致性度量，涉及消息交换模式和消息编码规则的变化对一致性的影响；绑定一致性度量值 *BdCE*（Binding Consistency Extent）是对单个服务绑定变化的一致性度量，包含绑定风格、传输协议和绑定地址的变化对一致性的影响；数据一致性度量值 *DaCE*（Data Consistency Extent）是对单个服务数据变化的一致性度量，涉及数据类型和数据级别（精度）的变化对一致性的影响。ΔL_{mod} 是指结构层变化向量序列中 mod 操作的序列，*IntC* 表示输入相关变化集，*OutC* 表示输出相关变化集。本文采用式(3)－式(11)进行计算，其相关定义在文献[11]中已有详细说明，此处不再赘述。

$$PerE(dv)=\begin{cases}1, & MP_{sou}=MP_{tar} \cap use_{sou}=use_{tar}\\ a, & MP_{sou}=MP_{tar} \cap use_{sou}\neq use_{tar}\\ 0, & MP_{sou}\neq MP_{tar}\end{cases} \tag{3}$$

$$OpCE(\Delta L_{mod})=\frac{\sum_{i=0}^{n} PerE(dv_i)}{n} \tag{4}$$

$$StrE(dv)=\begin{cases}1, & style_{sou}=style_{tar}\\ b, & style_{sou}\neq style_{tar}\end{cases} \tag{5}$$

$$TraE(dv)=\begin{cases}1, & transport_{sou}=transport_{tar}\\ c, & transport_{sou}\neq transport_{tar}\end{cases} \tag{6}$$

$$LocE(dv)=\begin{cases}1, & location_{sou}=location_{tar}\\ d, & location_{sou}\neq location_{tar}\end{cases} \tag{7}$$

$$BdCE(\Delta L_{mod})=\frac{\sum_{i=0}^{n}\{StrE(dv_i)+TraE(dv_i)+LocE(dv_i)\}}{3n} \tag{8}$$

$$ParE(dv)=\begin{cases}TypE(type_{sou},type_{tar}), & dv\in IntC\\ TypE(type_{tar},type_{sou}), & dv\in OutC\end{cases} \tag{9}$$

3）结构层一致性度量值 *SCE*（Structure Consistency Extent）

通过综合组合变化和单个服务结构变化，本文使用结构

层一致性度量值 SCE 对结构层变化进行定量评价，使用式(12)进行计算：

$$SCE(\Delta L_s)=\alpha * ScCE(\Delta L_s)+\beta * SsCE(\Delta L_s),\alpha+\beta=1 \tag{12}$$

其中，α 和 β 为租户自定义系数，其意义是租户可以自由控制对服务实例结构层的改变的容忍偏向，即更偏向容忍服务组合改变还是容忍单个服务结构变化。但现实中，租户往往只会关注功能是否实现，并不关注内部结构，而功能实现与结构息息相关，因此在实际应用中 α 和 β 通常根据租户的应用需求，使用经验统计科学或机器学习的方法得出。本文充分考虑了租户的定制需求，挖掘提取其中对一致性判定有效的信息。因需求挖掘并不是本文研究的重点，故此处不讨论用户需求挖掘的过程，下文同理。

4.1.2 非功能层一致性度量

文献[24]已给出完整的服务组合评价方法，对于非功能层一致性度量，本文参考文献[4]中的方法，通过比较各个 QoS 指标值的变化关系来判定非功能层的一致性，即判断目标 QoS 属性值 $value_{tar}$ 是否优于源 QoS 属性值 $value_{sou}$，表示为 $value_{sou}\leqslant value_{tar}$，具体定义如式(13)所示：

$$value_{sou}\leqslant value_{tar}\Leftrightarrow \begin{cases} value_{sou}\{=,<,s,fi,m,o\}value_{tar}(Type=\text{monotonic}) \\ value_{sou}\{=,>,si,f,mi,oi\}value_{tar}(Type=\text{antitonic}) \end{cases} \tag{13}$$

其中，$Type$ 是 QoS 属性的类型，有正比属性 monotonic 和反比属性 antitonic，在计算服务非功能层一致性时，需要分开考虑；f,s,o,m 是艾伦区间代数[25]，fi,si,oi,mi 分别表示对应的逆关系。采用式(13)判定得出的 QoS 一致性是完全严格的一致性，但从应用角度出发，实际情况往往并不需要如此严格，对服务质量在一定范围内的变化都是可以容忍的。参照文献[11]的方法，提出了适用于 SaaS 应用的服务质量一致性度量方法，用式(14)对 QoS 指标变化进行评价。

$$QoSE(dv)=\begin{cases} 1, & value_{sou}\leqslant value_{tar} \\ 1-g\dfrac{value_{tar}^{R}-value_{sou}^{R}}{value_{sou}^{L}+value_{sou}^{R}}, & value_{sou}>value_{tar}\text{且} \\ & Type=\text{antitonic} \\ 1-g\dfrac{value_{tar}^{L}-value_{sou}^{L}}{value_{sou}^{L}+value_{sou}^{R}}, & value_{sou}>value_{tar}\text{且} \\ & Type=\text{monotonic} \end{cases} \tag{14}$$

其中，g 为租户自定义系数，可以用于调节对 QoS 变化的容忍，实际应用中 g 的大小通过挖掘租户的需求或统计经验数据获得。根据式(14)，本文提出非功能层的一致性度量方法，并通过式(15)进行计算：

$$NCE(\Delta L_N)=\begin{cases} 0, & \exists dv\in\Delta L_{del} \\ \sum\dfrac{QoSE(dv_i)}{n}, & dv_i\in\Delta L_{mod} \end{cases} \tag{15}$$

其中，ΔL_{mod} 是指服务实例非功能层变迁向量序列中的 mod 操作序列，n 是指该序列中 QoS 变化的个数。

4.1.3 服务演化一致性度量

基于上文结构层一致性度量和非功能层一致性度量，本文采用加权方法来计算服务演化的一致度：

$$CCE(\Delta L)=\gamma * SCE(\Delta L_s)+\delta * NCE(\Delta L_N),\ \gamma+\delta=1 \tag{16}$$

其中，γ 和 δ 为租户自定义演化容忍权重系数，租户可通过调节 γ 和 δ 的值来控制对 SaaS 应用一致性定量计算的偏向，即是更关注结构一致性还是非功能一致性。实际应用中，γ 和 δ 主要根据租户需求得出，其值通过挖掘租户定制需求，采用经验统计科学或者机器学习的方法给出。

4.2 SaaS 服务演化的一致性判定

由上文工作可以细粒度地得到 SaaS 服务演化结构层和非功能层的一致性度量值及综合一致性度量值。结合租户需求和得到的度量值，便可进行一致性判定。

4.2.1 演化结构一致性判定

判定准则 1(演化结构一致性) 给定服务实例结构变化向量序列 ΔL_S，如果 $SEC(\Delta L_s)\geqslant\chi(\chi>0)$，则称由源服务实例结构 S_i 演化到目标实例结构 S_j 是满足一致性要求的，演化结构一致性得到保证，记为 $S_i\ll S_j$。其中，χ 为演化结构层的演化容忍度系数，表示租户对演化结构一致性的最大容忍度。其值根据租户定制的功能需求，使用经验统计科学的方法得出，也可由租户自己给定。

4.2.2 演化非功能一致性判定

判定准则 2(演化非功能一致性) 给定服务实例非功能变化向量序列 ΔL_N，如果 $NCE(\Delta L_N)\geqslant\psi(\psi>0)$，则称由源服务实例非功能层 N_i 演化到目标实例非功能层 N_j 是满足一致性要求的，演化非功能一致性得到保证，记为 $N_i\ll N_j$。其中，ψ 为演化非功能层的演化容忍度系数，表示租户对演化非功能一致性的最大容忍度。其值根据租户定制的非功能需求，使用经验统计科学的方法得出，也可由租户自己给定。

4.2.3 服务演化一致性判定

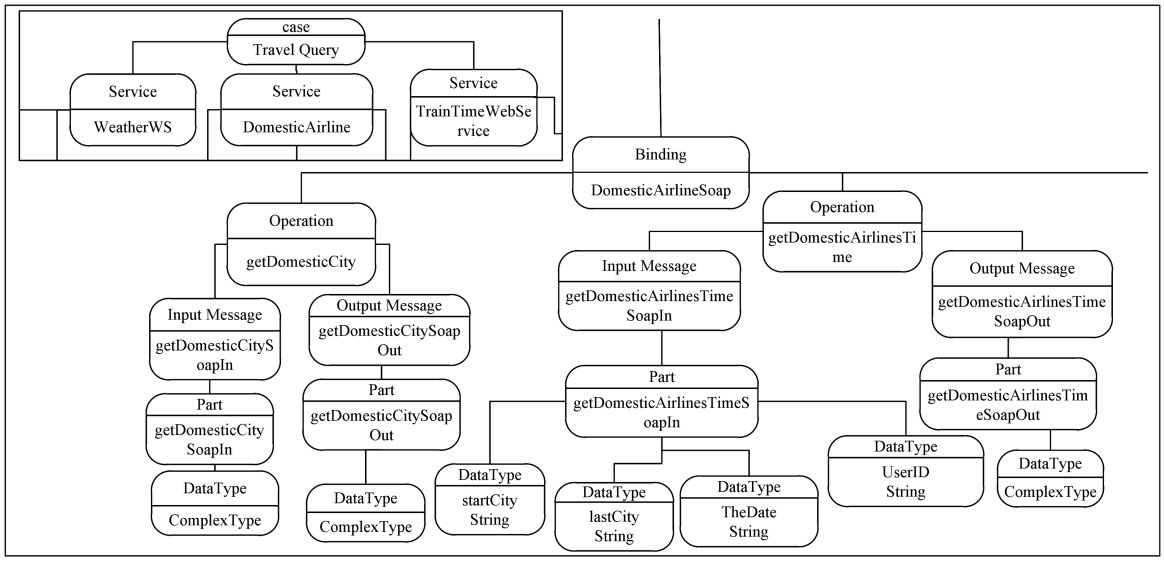
判定准则 3(服务演化一致性) 给定服务实例变化向量序列 ΔL，如果 $CCE(\Delta L)\geqslant\omega(\omega>0)$，则称由源服务实例 C_i 演化到目标实例 C_j 是满足一致性要求的，服务演化一致性得到保证，记为 $C_i\ll C_j$。其中，ω 为服务演化一致性判定阈值，表示租户对服务整体演化一致性的最大容忍情况。其值根据租户综合的定制需求，使用经验统计科学的方法得出，也可由租户自己给定。

5 案例分析

为分析和验证此 SaaS 服务演化一致性判定方法，基于一个实际的 SaaS 服务应用案例，设计 3 种演化场景，采用本文方法对一致性度量值进行计算，并对一致性做出判定。

5.1 演化场景的设置

本文基于一个实际的旅游出行查询 SaaS 服务案例来设计演化场景。该 SaaS 应用实例主要包括 3 个服务应用，分别是 2500 多个国内外城市的天气查询服务、国内火车时刻表查询服务、国内飞机航班时刻表查询服务。实例涉及 3 个服务、12 组绑定、64 个操作、128 个消息输入输出、200 多组消息参数集合，以及若干端口节点。由于该 SaaS 服务实例结构描述图过大，为了便于表述，图 4(a)给出该结构描述图的局部细节，图 4(b)给出服务实例的 QoS 指标。



(a)

执行时间(Execution Time):〈Time,antitonic,second,[0.35.1]〉
费用(Execution Cost):〈Cost,antitonic,yuan,[0.93,1]〉
可靠性(Reliability):〈Reliability,monotonic,percent,[87,88]〉
可用性(Availability):〈Availability,monotonic,percent,[89,92]〉
安全等级(Security):〈Security,monotonic,score,[4,5]〉
信誉度(Reputation):〈Reputation,monotonic,score,[3.5,5]〉

(b)

图 4　源实例 Travel Query 的结构及 QoS 评价指标

Fig. 4　Structure of source instance Travel Query and QoS evaluation index

演化场景 1　产生新服务实例 C_2,在 TrainTimeWebService 服务中,输入消息 getStationAndTimeByTrainCode 参数的 ID 类型由 int 变为 String,输出消息 getVersionTimeResponse 参数的 getVersionTimeResult 类型由 String 变为 int;在 DomesticAirline 服务中增加 getDomesticCity 操作;同时,QoS 的执行时间由[0.35,1]变为[0.35,1.3],费用由[0.93,1]变为[0.81,0.93]。

演化场景 2　产生新服务实例 C_3,在 TrainTimeWebService 服务中,输入消息 getStationAndTimeByTrainCode 参数的 ID 类型由 int 变为 String;在 DomesticAirline 服务中增加 getDomesticCity 操作;在 WeatherWS 服务中,输入消息 getWeather 的参数增加 date 类型变量;同时,QoS 的执行时间由[0.35,1]变为[0.77,2.0],可靠性由[87,88]变为[84,87],信誉度由[3.5,5]变为[3.4,4.1]。

演化场景 3　产生新服务实例 C_4,服务组合中减少 WeatherWS 服务,在 TrainTimeWebService 服务中,输入消息 getStationAndTimeByTrainCode 参数的 ID 类型由 int 变为 String;在 DomesticAirline 服务中增加 getDomesticCity 操作;同时,QoS 的执行时间由[0.35,1]变为(0,0.8],可靠性由[87,88]变为[84,100]。

5.2　实例分析

使用所提方法对 3 种演化场景的一致性进行判定,其中所使用到的参数都是根据案例用户的定制需求挖掘得出,具体设置如表 2 所列。为方便案例分析,将一致性的判定阈值统一设置为 0.85。

表 2　参数设置

Table 2　Setting of parameters

调节系数	值	调节系数	值
消息风格转换系数 a	0.9	数据类型转换系数 e	0.03
绑定风格转换系数 b	0.9	数据类别转换系数 f	0.1
传输协议转换系数 c	0.9	QoS 调节系数 g	1.0
绑定地址转换系数 d	0.9		
服务结构层调节系数	值	权重	值
服务组合容忍系数 α	0.5	结构层权重 γ	0.5
服务结构容忍系数 β	0.5	非功能层权重 δ	0.5
演化容忍度	值		
结构层演化容忍度系数 χ	0.9		
非功能层演化容忍度系数 ψ	0.9		
一致性判定阈值 ω	0.85		

1)演化场景 1 抽取出的变化向量如图 5 所示。

MOD Structure From
[TrainTimeWebService] TrainTimeWebService. TrainTimeWebServiceSoap. TrainTimeWebServiceSoapBinding. TrainTimeWebServiceSoap. getStationAndTimeByTrainCode. inputMessage. getStationAndTimeByTrainCodeSoapIn. parameters. getStationAndTimeByTrainCodeTo TrainTimeWebService. TrainTime WebServiceSoap. TrainTimeWebServiceSoapBinding. TrainTimeWebServiceSoap. getStationAndTimeByTrainCode. inputMessage. getStationAndTimeByTrainCodeSoapIn. parameters. getStationAndTimeByTrainCode
MOD No-fuction From
〈Time,antitonic,second,[0.35,1]〉 To 〈Time,antitonic,second,[0.35,1.3]〉
〈Cost,antitonic,yuan,[0.93,1]〉 To 〈Cost,antitonic,yuan,[0.81,0.93]〉

图 5　演化场景 1 中由实例 C_1 演化到实例 C_2 的变化向量

Fig. 5　Change vector sequence from C_1 to C_2 in evolutionary scenarione

演化场景 1 的判定结果为:结构层一致度为 0.9416;非功能层一致度为 0.9;服务演化一致度为 0.9208。根据设定的演化容忍度和一致性判定阈值,结构层一致性能够满足演化一致性,非功能层也能满足演化一致性。综合来看,服务演化一致性满足设定的阈值 0.85,因此演化场景 1 满足演化一致性。

2)演化场景 2 抽取出的变化向量如图 6 所示。

MOD Structure From[WeatherWS]WeatherWS. WeatherWSSoap. WeatherWSSoapBinding. WeatherWSSoap. getWeather. inputMessage. getWeatherSoapIn. parameters. getWeather To WeatherWS. WeatherWSSoap. WeatherWSSoapBinding. WeatherWSSoap. getWeather. inputMessage. getWeatherSoapIn. parameters. getWeather
[TrainTimeWebService] TrainTimeWebService. TrainTimeWebServiceSoap. TrainTimeWebServiceSoapBinding. TrainTimeWebServiceSoap. getStationAndTimeByTrainCode. inputMessage. getStationAndTimeByTrainCodeSoapIn. parameters. getStationAndTimeByTrainCode To TrainTimeWebService. TrainTimeWebServiceSoap. TrainTimeWebServiceSoapBinding. TrainTimeWebServiceSoap. getStationAndTimeByTrainCode. inputMessage. getStationAndTimeByTrainCodeSoapIn. parameters. getStationAndTimeByTrainCode
MOD No-fuction From
〈Time,antitonic,second,[0.35,1]〉 To 〈Time,antitonic,second,[0.77,2.0]〉
〈Reliability,monotonic,percent,[87,88]〉 To 〈Reliability,monotonic,percent,[84,87]〉
〈Reputation,monotonic,score,[3.5,5]〉 To 〈Reputation,monotonic,score,[3.4,4.1]〉

图 6 演化场景 2 中由实例 C_1 演化到实例 C_3 的变化向量

Fig. 6 Change vector sequence from C_1 to C_3 in evolutionary scenario two

演化场景 2 的判定结果为:结构层一致度为 0.4708;非功能层一致度为 0.9867;服务演化一致度为 0.7288。根据设定的演化容忍度和一致性判定阈值,结构层一致性不满足演化一致性,非功能层满足演化一致性。综合来看,服务演化一致性不满足设定的阈值 0.85,因此演化场景 2 不满足演化一致性。

3)演化场景 3 抽取出的变化向量如图 7 所示。

MOD Structure From
[Travel Query]Travel Query. WeatherWS. TrainTimeWebService. DomesticAirline TO Travel Query. TrainTimeWebService. DomesticAirline
[TrainTimeWebService] TrainTimeWebService. TrainTimeWebServiceSoap. TrainTimeWebServiceSoapBinding. TrainTimeWebServiceSoap. getStationAndTimeByTrainCode. inputMessage. getStationAndTimeByTrainCodeSoapIn. parameters. getStationAndTimeByTrainCode To TrainTimeWebService. TrainTimeWebServiceSoap. TrainTimeWebServiceSoapBinding. TrainTimeWebServiceSoap. getStationAndTimeByTrainCode. inputMessage. getStationAndTimeByTrainCodeSoapIn. parameters. getStationAndTimeByTrainCode
MOD No-fuction From
〈Time,antitonic,second,[0.35,1]〉 To 〈Time,antitonic,second,(0,0.8]〉
〈Reliability,monotonic,percent,[87,88]〉 To 〈Reliability,monotonic,percent,[84,100]〉

图 7 演化场景 3 中由实例 C_1 演化到实例 C_4 的变化向量

Fig. 7 Change vector sequence from C_1 to C_4 in evolutionary three

演化场景 3 的判定结果为:结构层一致度为 0.5541;非功能层一致度为 0.9914;服务演化一致度为 0.7728。根据设定的演化容忍度和一致性判定阈值,结构层一致性不满足演化一致性,非功能层满足演化一致性。综合来看,服务演化一致性不满足设定的阈值 0.85,因此演化场景 3 不满足演化一致性。

根据实验结果,演化场景 1 中,输入数据的类型由 int 变为 string,符合输入逆变,对一致性没有太大影响;输出数据的类型由 string 变为 int,符合输出协变,对一致性也没有太大影响,同理,增加输出操作 getDomesticCity 对结构层一致性的影响不大。演化场景 2 延续演化场景 1 的变化,不过在另一个服务中增加输入参数,由于在服务中没有提供该输入数据相应的机制,因此对一致性造成较大影响,使结构层一致性得不到满足;非功能层是对整个应用的评价,由数据显示,对非功能层并没有造成太大影响。整体来看,场景 2 不满足整体一致性,这与现实相符合。演化场景 3 中,服务实例比源实例减少了一个服务,此种情况一定不能满足用户需求。数据显示其对服务实例结构的影响很大,结构层一致性得不到满足。虽然非功能层的一致性得到满足,但综合来看,整个服务的一致性没能得到满足,这与实际的反馈情况相符。

对比实际应用演化一致性情况,上述演化场景的一致性判定结果与实际经验相符合,实际中仅有演化场景 1 反馈的情况能够达到相对的演化一致性;而场景 2 和场景 3 中的演化一致性反馈情况不理想,但并不能直观地得出一致性不满足,仅能根据用户需求以及实际应用情况作出定性的分析,得出演化一致性是不满足的。采用本文所提方法,可以量化一致性情况,直观地显示出一致性的保持情况;由于本文分层次量化演化一致性,能由量化数值得知演化一致性结构层和非功能层的一致性情况;同时,此方法在量化一致性保持情况和一致性判定过程中,考虑了核心租户的演化需求,与实际应用定性分析一致性过程的反馈情况相符合。

结束语 针对 SaaS 应用,充分考虑租户因素,提出了一种带租户演化容忍度的 SaaS 服务演化一致性判定方法。为解决问题,提出服务实例描述模型、变化向量抽取方法和一致性定量计算方法,以及一致性判定方法。采用本文方法可细粒度分层次解决 SaaS 服务演化带来的一致性判定问题。

本文的主要工作包括以下几个方面:

1)分析 SaaS 应用的特点以及 SaaS 服务演化的一般过程,提出服务演化一致性的判定需要考虑租户的演化容忍度。与现有研究相比,所提方法考虑到了 SaaS 核心的租户因素。

2)提出基于实例变迁的服务实例描述模型。与文献[11]的工作相比,本文实现了对服务实例的描述,且此描述模型还可以反映实例变迁,清晰地表示 SaaS 服务演化中的变化。

3)提出分层次的变化向量抽取方法,提出分层次细粒度的一致性定量计算方法,并充分考虑了租户演化容忍度,提出一致性分层次判定方法。

4)结合实际的 SaaS 服务应用案例进行分析,验证了所提方法的可行性和有效性。

本文仅讨论了实例中服务的增加和减少对演化一致性的影响,并没有对服务替换进行区分。下一步将关注服务替换

问题，深层次细粒度地对服务演化一致性进行定量计算并判定；进一步分析 SaaS“多租户单实例”的局限性，研究如何运用此方法规避一致性判定上的局限性等细节问题；计划将此方法用于实时在线演化的一致性检查中，为 SaaS 应用提供动态支持。

参 考 文 献

[1] 周宇辰. 面向服务的计算(SOC)：技术、规范与标准[M]. 北京：电子工业出版社，2010.

[2] FOX A，PATTERSON D. SaaS 软件工程[M]. 北京：清华大学出版社，2015.

[3] YANG F Q. Thoughts on the development of software engineering technology [J]. Journal of Software，2005，16(1)：1-7. (in Chinese)
杨芙清. 软件工程技术发展思索[J]. 软件学报，2005，16(1)：1-7.

[4] ANDRIKOPOULOS V，BENBERNOU S，PAPAZOGLOU M P. On the Evolution of Services[J]. IEEE Transactions on Software Engineering，2012，38(3)：609-628.

[5] HE J. Demand-driven evolution of SaaS services[D]. Kunming：Yunnan University，2013：1-7. (in Chinese)
何俊. 需求驱动的 SaaS 服务演化研究[D]. 昆明：云南大学，2013：1-7.

[6] WANG S Y. Adaptive evolutionary technology of SaaS platform supporting collaborative industry chain [J]. Journal of Southwest Jiaotong University，2012，47(1)：39-45. (in Chinese)
王淑营. 支撑产业链协同的 SaaS 平台自适应演化技术[J]. 西南交通大学学报，2012，47(1)：39-45.

[7] LIU S Q，WANG H Y，CUI L Z. Progressive Pattern Evolution Method Based on Data Dependency in SaaS Application [C]// Proceedings of the First National Conference on Service Computing(CCF NCSC 2010). 2010. (in Chinese)
刘士群，王海洋，崔立真. Saas 应用中基于数据依赖的渐进式模式演化方法[C]// 全国服务计算学术会议. 2010.

[8] ZHOU L，CAO J，CHEN J J. Automatic Evolution of Software and Service Process Model [J]. Computer Integrated Manufacturing Systems，2011，17(8)：1603-1608. (in Chinese)
周亮，曹健，陈姣娟. 软件即服务流程模型的自动演化[J]. 计算机集成制造系统，2011，17(8)：1603-1608.

[9] JIANG X D，XIE Z W，LI T，et al. Correlation analysis software behavior for dynamic evolution [J]. Small Computer Systems，2016，37(9)：1925-1929. (in Chinese)
蒋旭东，谢仲文，李彤，等. 面向动态演化的软件行为相关性分析研究[J]. 小型微型计算机系统，2016，37(9)：1925-1929.

[10] BAO A H. Semantic Web composite service evolution method and its key technology research [D]. Changsha：University of Defense Technology，2009. (in Chinese)
鲍爱华. 语义 Web 环境下组合服务演化方法及其关键技术研究[D]. 长沙：国防科学技术大学，2009.

[11] LI B，GAO Y，WANG B，et al. Coordination of Service Evolution Based on Change [J]. Computer Engineering and Science，2014，36(12)：2257-2266. (in Chinese)
李冰，高岩，王斌，等. 基于变化的服务演化一致性判定[J]. 计算机工程与科学，2014，36(12)：2257-2266.

[12] BECKER K，LOPES A，MILOJICIC D S，et al. Automatically Determining Compatibility of Evolving Services[C]// IEEE International Conference on Web Services. IEEE Computer Society，2008：161-168.

[13] FAN L，TANG J，LING Y，et al. Dynamic and quantitative method of analyzing service consistency evolution based on extended hierarchical finite state automata [J]. The Scientific World Journal，2014，2014(2)：793271.

[14] KLAI K，OCHI H. Checking compatibility of web services using SOGs[C]// Proc of IEEE 19th International Conference on Web Services(ICWS'12). 2012：670-671.

[15] HE L J，LIU L C，WU C. Relaxation matching of structured web services[J]. Journal of Tsinghua University(Science and Technology)，2011，51(3)：289-292. (in Chinese)
何玲娟，刘连臣，吴澄. 结构化 Web 服务的松弛匹配[J]. 清华大学学报(自然科学版)，2011，51(3)：289-292.

[16] HE L J，LIU L C，WU C. An improved method of similarity measure based on WSDL description[J]. Journal of Computers，2008，31(8)：1331-1339. (in Chinese)
何玲娟，刘连臣，吴澄. 一种改进的基于 WSDL 描述的操作相似性度量方法[J]. 计算机学报，2008，31(8)：1331-1339.

[17] DAM H K，GHOSE A. Supporting Change Propagation in the Maintenance and Evolution of Service-Oriented Architectures [J]. Asia Pacific Software Engineering Conference，2010，115(2)：156-165.

[18] CHANDA J，SENGUPTA S，KANJILAL A，et al. Behavioral and structural evolution of SOA from OO：an integrated approach[J]. Acm Sigsoft Software Engineering Notes，2013，38(5)：1-9.

[19] LEFEBVRE S，KUMAR S P，CHIKY R. Simizer：evaluating consistency trade offs through simulation[C]// The Workshop on Principles and Practice of Eventual Consistency. ACM，2014：6.

[20] HOLT B，BORNHOLT J，ZHANG I，et al. Disciplined Inconsistency with Consistency Types [C] // ACM Symposium on Cloud Computing. ACM，2016：279-293.

[21] ZOU J，LIU X，SUN H，et al. Live Instance Migration with Data Consistency in Composite Service Evolution [C] // Services. IEEE，2010：653-656.

[22] NABUCO O，BONACIN R，FUGINI M，et al. Web2Touch 2016：Evolution and Security of Collaborative Web Knowledge [C]// IEEE，International Conference on Enabling Technologies：Infrastructure for Collaborative Enterprises. IEEE，2016：214-216.

[23] LIAO L，QI S，LI B. Trust analysis of composite service evolution[C]// IEEE，International Conference on Software Engineering Research，Management and Applications. IEEE，2016：15-22.

[24] ZHU H N. Service-oriented portfolio service evaluation technology research [D]. Shenyang：Northeastern University，2009. (in Chinese)
朱红宁. 面向 Web 服务组合的服务 QoS 评价技术的研究[D]. 沈阳：东北大学，2009.

[25] ALLEN J F. Maintaining Knowledge about Temporal Intervals [J]. Readings in Qualitative Reasoning About Physical Systems，1983，26(11)：361-372.