软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(11):3435−3454 [doi: 10.13328/j.cnki.jos.005294]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于可变性模型的可复用与可定制 SaaS 软件开发方法*

孙昌爱, 张在兴, 张 鑫

(北京科技大学 计算机与通信工程学院,北京 100083)

通讯作者: 孙昌爱, E-mail: casun@ustb.edu.cn

摘 要: 云计算环境下,软件通过互联网向租户提供服务,这种基于互联网的软件交付模式称为 SaaS(软件即服务).与传统软件交付模式相比,SaaS 软件通常运行于软件供应商的服务器端,同时为多个租户提供服务.由于需要支持不同租户的个性化需求,SaaS 软件应具备足够的灵活性,以应对快速变化的租户需求;而且针对某一个租户的变更,不应影响其他租户.通过扩展课题组前期开发的基于可变性管理的适应性服务组装方法及其支持平台,提出了一种云计算环境下可复用、可定制的 SaaS 软件开发方法,开发了相应的支持平台,包括支持 SaaS 模式的服务组装引擎和远程定制工具.该方法针对不同租户的共性需求,提供一个抽象服务组装模型,支持平台在运行阶段解释执行抽象服务组装模型,根据租户的个性化需求派生不同的流程实例,这些运行时流程实例多态共存、互不影响.采用一个特定领域的 SaaS 软件实例来验证该方法的可行性,评估了支持平台的性能.实验结果表明,该方法及其支持平台可以支持多实例多租户的交付模式.

关键词: Web 服务;服务组装;可变性管理;VxBPEL;SaaS

中图法分类号: TP311

中文引用格式: 孙昌爱,张在兴,张鑫.基于可变性模型的可复用与可定制 SaaS 软件开发方法.软件学报,2018,29(11):3435−3454. http://www.jos.org.cn/1000-9825/5294.htm

英文引用格式: Sun CA, Zhang ZX, Zhang X. Reusable and customizable saas software development approach based on variation model. Ruan Jian Xue Bao/Journal of Software, 2018,29(11):3435−3454 (in Chinese). http://www.jos.org.cn/1000-9825/5294.htm

Reusable and Customizable SaaS Software Development Approach Based on Variation Model

SUN Chang-Ai, ZHANG Zai-Xing, ZHANG Xin

(School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China)

Abstract: In the context of cloud computing, software is delivered as a service to the customer through the Internet, and such a software delivery mode is called SaaS (software as a service). Unlike the traditional software delivery mode, SaaS software is usually running on the server side, which provides services to multiple tenants at the same time. As a result, SaaS software should be designed to meet the individual needs of different tenants, they should be flexible enough to cater for the rapidly changing tenant's requirements, and the response to a talent's change should not affect other tenants. Using the adaptive service composition method based on variability management and its supporting platform, a reusable and customizable SaaS software development method for the context of cloud computing is proposed, and a supporting platform is developed to facilitate the adoption of the proposed method. The platform includes a SaaS mode supporting service composition engine and a remote customization tool. The proposed method first creates an abstract service composition model to meet the common requirements of different tenants, and then the supporting platform is used to interpret the model and derive multiple different process instances at runtime, which are concurrently executed and isolated. A case study is conducted to

* 基金项目: 国家自然科学基金(61872039, 61370061); 北京市自然科学基金(4162040); 航空科学基金(2016ZD74004); 北京市优秀人才培养资助项目(2012D009006000002)

Foundation item: National Natural Science Foundation of China (61872039, 61370061); Beijing Natural Science Foundation (4162040); Aeronautical Science Foundation (2016ZD74004); Beijing Municipal Training Program for Excellent Talents (2012D009006000002)

收稿时间: 2017-01-09; 修改时间: 2017-02-20; 采用时间: 2017-04-23; jos 在线出版时间: 2018-04-16

CNKI 网络优先出版: 2018-04-16 10:59:42, http://kns.cnki.net/kcms/detail/11.2560.TP.20180416.1059.003.html

validate the feasibility of the proposed method and evaluate the performance of the supporting platform using a domain specific SaaS software. Experimental results show that the proposed method and platform present a viable alternative for multi-tenant and multi-instance delivery mode for SaaS software.

Key words: Web services; service composition; variability management; VxBPEL; SaaS

传统的软件运营模式下,软件的购买、部署、使用和维护全部离散地分布在软件用户本地,软件运行的硬件设备、数据维护及备份工作都需要用户自己解决,软件供应商需要到现场为用户进行维护、升级等工作.SaaS (software as a service)是云计算的一种软件交付模式(如图 1 所示),其核心是软件服务化[1].该模式下的软件如同电、燃气、暖气等生活资源一样按需提供给租户,租户可通过网络远程按需计费租用这种服务.

尽管 SaaS 软件的多个租户的需求绝大部分是相同的,但各个租户的需求之间仍存在差异性.开发具有多租户特性的应用软件,如果能够采用“一次开发、一次部署、多租户共享”的方式,则能够降低软件维护成本与管理复杂性.为了实现所有租户需求之间能够“求同存异”[2],SaaS 软件应允许个性化定制,以满足不同租户的差异性需求.针对每个租户进行个性化定制时,应不影响对其他租户需求的响应,我们将这样的特性称为“需求隔离”.经过个性化定制后,满足不同租户需求的 SaaS 软件之间存在一些差异,如图 2 所示.为了支持多个租户并发访问,这些存在差异的 SaaS 软件需要同时部署与运行于云端服务器,我们将这种特性称为 SaaS 软件的“多态共存”.SaaS 软件以服务为单元提供给租户使用,通常一个服务可以满足多个租户的需求.理想情况下,SaaS 软件能够以服务组装的方式同时向多个租户提供复杂的应用软件,软件维护、升级等工作只需在一个软件版本上进行[3].

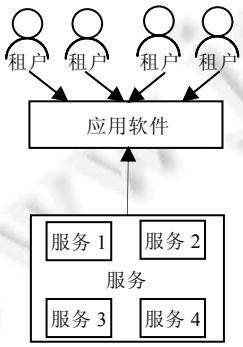


Fig.1 Multi-Tenant nature

图 1 多租户特性

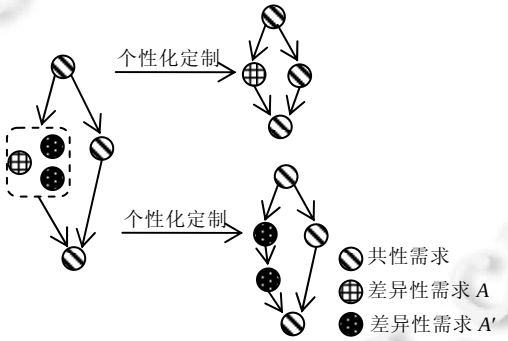


Fig.2 Requirement isolation

图 2 需求隔离

课题组前期研究工作从规格说明层考虑服务组装的适应性问题,提出了基于可变性管理的适应性服务组装方法与支持平台[4−13].通过对标准的服务组装语言 BPEL 进行扩展,引入了多种可变性构造子,开发了支持可变性设计的服务组装语言 VxBPEL[9].为了提高 VxBPEL 对复杂变体依赖关系的表达能力,引入基于约束关系的可变性设计与配置方法[13].为了支持服务组装中无法预期的运行时变化,引入了抽象代理与动态绑定机制[11].我们开发了基于 VxBPEL 的服务组装支持平台,对 VxBPEL 流程分析、设计、部署和运行阶段提供全面的支持,包括可视化分析工具[6]、设计工具[12,13]、引擎[7,8]与运行时管理工具[13].上述研究工作较为系统地解决了服务组装的适应性问题.

然而,前期工作提出的适应性服务组装方法无法直接解决 SaaS 软件的“需求隔离”和“多态共存”问题.一方面,当用户需求发生变化时,基于可变性管理的适应性服务组装方法通过运行时切换可变性配置来响应用户的新需求,但是这种切换操作的影响是全局性的.也就是说,为了满足一个用户的需求而做出的改变将影响到其他用户,从而引起用户需求冲突.另一方面,基于可变性管理的适应性服务组装方法通过切换可变性配置响应不同的用户需求,尽管这个切换过程不需要重启服务器,但需要重新编译生成新流程对象.当运行复杂的 SaaS 软件时,这段时间相比于软件运行时间而言较长,即配置切换占空比较高.

本文试图解决基于服务组装的 SaaS 软件开发中的可复用与可定制的问题.为了支持 SaaS 软件的“需求隔离”和“多态共存”特性,本文提出一种可复用与可定制的 SaaS 软件开发方法,引入基于可变性设计的抽象服务组装模型以满足 SaaS 软件的多态共存的需求,增强软件的可复用性与可定制性,实现“一次开发、一次部署、多次使用”的 SaaS 软件开发目标;我们还开发了相应的支持平台,进一步增强所提方法的可行性与实用性,提高了 SaaS 软件运行时间占空比.本文主要研究成果如下.

(1) 提出了一种基于服务组装的可复用、可定制的 SaaS 软件开发方法:该方法针对不同租户的共性需求提供一个抽象服务组装模型(无须维护不同流程实例),在运行时刻根据租户的个性化需求派生不同的流程实例,不同的流程实例多态共存、互不影响.

(2) 开发了一个支持 SaaS 模式的服务组装引擎:采用推迟编译可变性配置的方法,对 VxBPEL 服务组装引擎 VxBPEL_ODE 进行扩展,允许租户在运行时对业务流程进行定制,对不同业务流程进行隔离,从而支持 SaaS 模式的服务组装的解释执行.

(3) 开发了一个运行时管理工具 RTM4B:支持租户对 SaaS 软件进行远程个性化定制,包括远程读取业务流程、提取业务流程中可变性信息、远程配置与上传配置文件、远程执行测试用例等功能.

(4) 采用一个实例系统验证了本文提出的可复用、可定制的 SaaS 软件开发方法的可行性,评估该方法的开发效率和多租户模式下 SaaS 软件的执行效率.

本文第 1 节介绍本文研究工作的背景知识.第 2 节提出一种可复用、可定制的 SaaS 软件开发方法.第 3 节讨论支持平台的设计与实现.第 4 节验证本文所提的方法与支持平台.第 5 节介绍相关研究工作.最后总结全文.

1 研究背景

介绍服务组装语言 BPEL、可变性管理、支持可变性设计的服务组装语言 VxBPEL,SaaS 的基本原理与相关概念.

1.1 BPEL

Business Process Execution Language(BPEL)是一种面向服务、基于 XML 的可执行的业务流程编程语言.BPEL 流程可以指定调用哪些 Web 服务以及调用的顺序,提供一种将多个 Web 服务组合起来以实现复杂的业务流程的手段.BPEL 流程的规格说明由伙伴链接声明、变量声明、处理器(handler)声明和业务逻辑这 4 部分组成.伙伴连接声明部分定义了一系列参与交互的服务,变量声明部分定义了流程中使用的消息和 XML 文件格式,处理器声明部分通常指明异常、特殊事件及补偿相关的处理方法,业务逻辑部分由一组交互活动构成.活动可以分为基本活动和结构化活动.基本活动定义了流程基本功能操作,包括服务调用、操作、数据传输等.BPEL 的基本活动包括:“receive”“invoke”“reply”活动(流程与外界通信的基本活动);“assign”活动(流程中数据处理的基本活动);“throw”活动(用于发出故障信号);“terminate”活动(停止当前流程的执行);“wait”活动(等待一段时间或到达某个截止期限后再执行);“empty”活动(不执行任何动作)、“compensate”活动(与 scope 联合使用执行补偿动作).BPEL 的结构化活动是按照某种结构形成的一组活动,包括“sequence”“switch”和“while”活动(提供流程控制);“flow”活动(支持活动间的并发和同步);“pick”活动(基于外部事件选择不同的动作).

1.2 可变性管理

近年来,越来越多的软件系统采用基于服务的开发方式[14],松散耦合的特性有助于实现这类系统的可重用性和适应性[4].为适应快速变化的业务需求与运行时环境,实现业务过程的服务组装需要具有足够的灵活性[15].例如,业务流程应能够快速、动态地改变,在满足接口一致性前提下具备改变流程的能力,或者通过改变其接口实现外在行为的改变[16].

可变性是指一个软件系统具有能够根据环境进行扩展、改变、定制或者配置的能力[17].可以通过指定软件系统的一部分为可变因素,然后根据需求派生出不同版本的软件系统,从而使得软件系统具有可变性.可变性建模的主要概念包括变异点、变体和实现关系.变异点是对可能发生变化位置的抽象,变体是对变化备选方案的

抽象.变异点包含该类变化的多个备选方案,每个备选方案都是一种特定场景下的业务逻辑的实现.可变性的抽象可能存在于业务流程中的不同层次.低层次的可变性抽象表现为将系统分为不同功能模块的变异点,每个变异点由一组实现不同功能或性能的变体组成,低层次可变性抽象为系统提供了实现灵活性和配置性的基础.大量变异点可能导致配置过程复杂且易错,因此不能很好地满足用户需求.通过提高变化的抽象层次,简化针对不同用户需求的可变性配置,可以隐藏底层可变性实现的复杂性.高层次的可变性抽象侧重关注业务需求,通过指定不同低层次变异点下变体间的依赖关系来响应不同的需求.总之,包含可变性设计的业务流程能够通过选择每个变异点下的变体来响应需求或环境的变化.

1.3 VxBPEL

课题组在前期工作中对现有的服务组装语言BPEL进行扩展,设计了一种支持可变性设计的服务组装语言VxBPEL[9].VxBPEL提供变异点、变体、实现关系等可变性构造子.

在VxBPEL中,将变体的概念实现为一个新元素〈Variant〉.变体用于定义由一组活动构成的反映某一场景的行为.每个变体元素表示一种可被替代的功能或操作,换句话说,变体提供了可变性选择的具体内容.变体的语法定义如图 3 所示.标签的“name”属性唯一地标识了变体过程的代码片段,在进行可变性配置时使用“name”属性实现变体选择.〈Variant〉的内容可以是 BPEL 代码,也可以是在〈VariantPoint〉嵌套定义中实现更复杂的可变性设计元素.

```
1. 〈vxbpel:Variant name=“default”〉
2.     (bpel code or 〈VariaionPoint〉 elements)
3. 〈/vxbpel:Variant〉
```

Fig.3 Syntax of variant in VxBPEL

图 3 VxBPEL 中变体定义的语法格式

〈VariationPoint〉用于定义变化可能发生的位置,包含多种可相互替代的行为,其中每个可被替代的行为都被封装到一个〈Variant〉元素中,其语法定义如图 4 所示.

```
1.  〈vxbpel:VariationPoint name=“vp1”〉
2.    〈vxbpel:Variants〉
3.      〈vxbpel:Variant name= “v1”〉
4.        (bpel code or 〈VariaionPoint〉 elements)
5.      〈/vxbpe:Variant〉
6.        〈vxbpel:Variant name=“v2”〉
7.        (bpel code or 〈VariaionPoint〉 elements)
8.      〈/vxbpel:Variant〉
9.    〈/vxbpel:Variants〉
10. 〈/vxbpel:VariationPoint〉
```

Fig.4 Syntax of VariationPoint in VxBPEL

图 4 VxBPEL 中变异点定义的语法格式

在 VxBPEL 中,实现关系用于定义不同变异点下的变体间因特定业务需求而存在的约束关系,采用〈vxbpel:Configurable-VariationPoint〉构造子定义,通常置于规格说明的〈/process〉元素前.图 5 示意了实现关系的语法定义.其中,“id”属性是实现关系的唯一标识,“defaultVariant”属性表示默认的流程实现.〈Variants〉标签下定义的每个〈Variant〉指定流程变体的高层变异点,在〈VPChoices〉元素中实现了高层变异点和低层变体间的映射关系.〈Rational〉和〈VariantInfo〉标签提供对配置的描述,指导运行时配置的选择.

```
1. ⟨vxbpel:ConfigurableVariationPoint id="1"
2.          defaultVariant="default"⟩
3.   ⟨vxbpel:Name⟩...⟨/vxbpel:Name⟩
4.   ⟨vxbpel:Rationale⟩...⟨/vxbpel:Rationale⟩
5.   ⟨vxbpel:Variants⟩
6.     ⟨vxbpel:Variant name="default"⟩
7.     ⟨vxbpel:VariantInfo⟩ ⟨/vxbpel:VariantInfo⟩
8.     ⟨vxbpel:RequiredConfiguration⟩
9.       ⟨vxbpel:VPChoices⟩
10.        ⟨vxbpel:VPChoice vpname="VP1" variant="default"/⟩
11.        (other ⟨vxbpel:VPChoice⟩ elements)
12.      ⟨/vxbpel:VPChoices⟩
13.    ⟨/vxbpel:RequiredConfiguration⟩
14.  ⟨/vxbpel:Variant⟩
15.  (other ⟨vxbpel:Variant⟩ elements)
16.  ⟨/vxbpel:Variants⟩
17. ⟨/vxbpel:ConfigurableVariationPoint⟩
```

Fig.5 Syntax of ConfigurableVariationPoint in VxBPEL

图 5 VxBPEL 中可配置变异点定义的语法格式

1.4 Software as a Service (SaaS)

依据复用程度、可配置性、可伸缩性、高性能等评价指标,将 SaaS 软件的成熟度模型划分为 4 个等级[18].

(1) 第 1 级:软件供应商为每个租户定制专属软件,并部署在服务器端供租户远程调用,每个租户独占一个运行实例.不同流程之间无法共享服务.

(2) 第 2 级:在设计时考虑软件的扩展问题,能够对软件中大部分功能进行复用.当租户有不同的需求时,能够在模板的基础上进行配置,产生一个新的软件,并再次部署到服务端,实现了一次开发多次部署.

(3) 第 3 级:支持多个租户共享同一个实例软件,通过运行时动态配置,满足不同租户之间的个性化需求,提出了满足单实例多租户的应用架构.

(4) 第 4 级:通过一个中间调度层,将租户分配到不同的运行实例上,不同的运行实例能够运行在不同的硬件设备上,将单实例多租户模式扩展为多实例多租户模式.

2 基于可变性模型的可复用与可定制的 SaaS 软件开发方法

介绍一种基于可变性模型的可复用与可定制的 SaaS 软件开发方法,包括基本原理、框架与运行过程.该方法支持第 4 级成熟度模型 SaaS 软件.

2.1 基本原理

采用传统服务组装实现的软件交付模式中,运行时刻不同用户只能执行相同的软件实例,如图 6 所示.A,B,C 这 3 个用户执行的所有实例均直接来自流程定义对象,即 $A_1=A_2=B_1=B_2=C_1$.若用户 A 出现需求变动,为了不影响其他用户的需求,我们需要对之前版本的流程定义对象进行较小的改动.由此可见:采用传统服务组装实现该软件时,尽管不同流程实例之间可以复用一些共同服务,这些服务通常部署在服务容器中,支持不同用户的并发访问,但是流程定义本身无法复用(即需要维护不同的流程实例).因此,该模式存在开发效率低与维护困难等问题.

为了增强上述基于服务组装实现的 SaaS 软件的可复用性与可定制性,本文在服务组装中引入可变性模型,提出一种基于可变性模型的 SaaS 软件开发方法,其原理如图 7 所示.该方法首先针对用户的共性需求设计一个抽象服务组装模型,该模型能够体现同一个问题领域内不同用户的共性需求.假设共性需求作为软件必选服务的需求(记 V_0),差异性需求作为软件的可选服务需求(记做$\Delta_1,\Delta_2,\Delta_3,\ldots,\Delta_n$),则待开发的 SaaS 软件的需求集为 $V_{总}=V_0+\Delta_1+\Delta_2+\ldots+\Delta_n$.然后,依据不同租户的需求(即用户配置文件)对该抽象服务组装模型进行定制,派生出不同的具体流程实例.即 $V_i=V_0+\sum_{j=p}^{q}\Delta_j$ $(0<p\leqslant q\leqslant n)$,其中,$V_i$ 表示不同租户需求对应的服务集.在运行阶段,采用相应的支持平台解释执行抽象服务组装模型,根据用户的个性化需求创建不同的流程实例,即 $A_1=A_2$,$B_1=B_2$,但

$A_1 \neq B_1, A_1 \neq C_1$,而且 $B_1 \neq C_1$.这些运行时流程实例多态共存、互不影响(即 A_1 与 A_2 并发执行、互不影响,B_1 与 B_2 并发执行、互不影响,而且 A_1 与 A_2 和 B_1 与 B_2 互不影响).因此,提出的 SaaS 软件开发方法支持多实例多租户的交付模式,满足 SaaS 软件第四级成熟度模型.

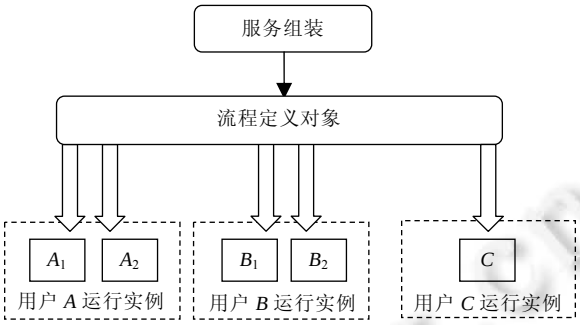


Fig.6 Software delivery mode supported by conventional service compositions

图 6 传统的服务组装支持的软件交付模式

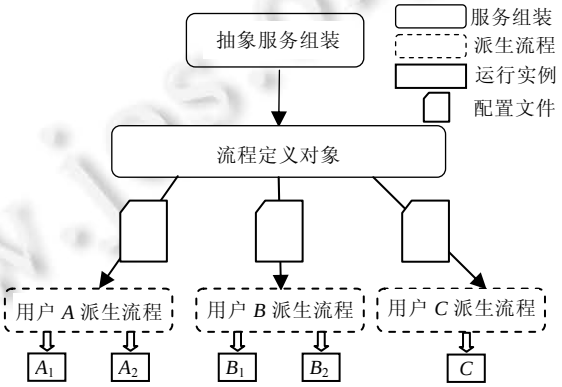


Fig.7 Software delivery mode supported by service compositions based on variation model

图 7 基于可变性模型的服务组装支持的软件交付模式

2.2 基于可变性模型的SaaS软件开发框架

在课题前期研究工作中,我们扩展标准的服务组装语言 BPEL 设计了 VxBPEL[9−11],开发了基于 VxBPEL 服务组装的支持平台,包括可视化的可变性分析工具 ValySec[6]、可视化的 VxBPEL 设计工具 VxBPEL Designer[10,12]、VxBPEL 引擎(VxBPELEngine[8]和 VxBPEL_ODE[7])、运行时可变性管理工具 MX4B[13].本文采用 VxBPEL 设计抽象服务组装模型,VxBPEL_ODE 支持基于 VxBPEL 的服务组装的解释执行,但无法支持多实例多租户的交付模式,本文对其进行扩展并开发支持 SaaS 模式的 VxBPEL 引擎 VxBPEL_SaaS.

提出的基于可变性模型的 SaaS 软件开发框架,如图 8 所示.

该框架在服务组装实现的 SaaS 软件中引入可变性设计,支持不同租户对同一 SaaS 软件的不同版本的流程设计的复用.通过在部署阶段对流程中全部变体进行无差别编译和在运行阶段根据租户的配置文件生成相应的流程实例,支持面向不同租户的具体流程的派生.

- 多租户需求分析阶段的主要任务是提取共性需求和差异性需求.采用已有领域软件需求分析技术,获取并建立待开发的 SaaS 软件的需求集.根据预期租户的个性化需求,建立特征模型,描述不同租户对需求项的选择说明.
- 创建抽象服务组装模型阶段的主要任务是依据特征模型、需求与服务之间的映射关系进行面向服务

的垂直组装,采用 VxBPEL 建立抽象服务组装模型.可有两种方法创建抽象服务组装模型:第 1 种方法采用 BPEL 创建一个基本的业务流程的规格说明,然后,在此规格说明中增量地引入可变性设计,得到 VxBPEL 规格说明;第 2 种方法直接采用 VxBPEL 进行业务流程的设计,在预期发生变化的位置设置变异点,变异点下设置多个变体.该阶段通过在基本流程中引入可变性的流程定义,得到一个能够满足所有租户需求的 VxBPEL 规格说明.

- 服务组装规格说明的部署与执行阶段的主要任务是部署与执行 VxBPEL 规格说明,依据租户的差异性需求,派生出面向不同租户的流程实例.设计阶段的 VxBPEL 规格说明本身无法明确地区分不同租户的需求,为了在运行时区分不同租户的业务流程,需要在变体标签中添加租户信息的描述.为了便于管理与描述租户需求,我们采用基于 XML 的租户配置文件(自定义扩展名“uccf”标识)描述租户信息,其格式如图 9 所示.其中,⟨client⟩标签指明当前配置所属租户;⟨configuration⟩标签指明当前流程信息;⟨VPChoices⟩标签指明在每个变异点下变体的选择,即多个⟨VPChoice⟩标签的集合.通过将抽象服务组装模型与租户配置文件进行绑定分析,在运行时,能够根据租户信息选取相应的变体,为每个租户派生出一个具体流程实例.

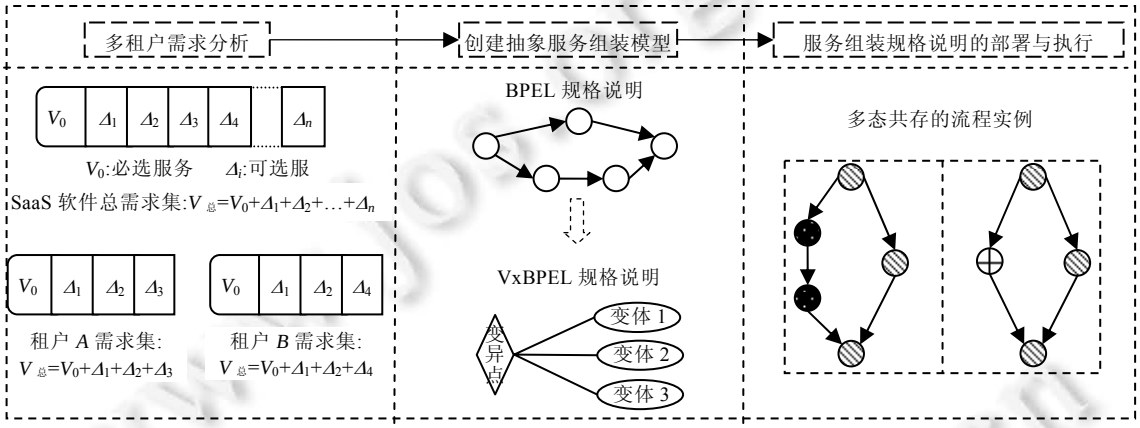


Fig.8 Reusable and customizable SaaS software development framework based on variation model

图 8 基于可变性模型的可复用与可定制 SaaS 软件开发框架

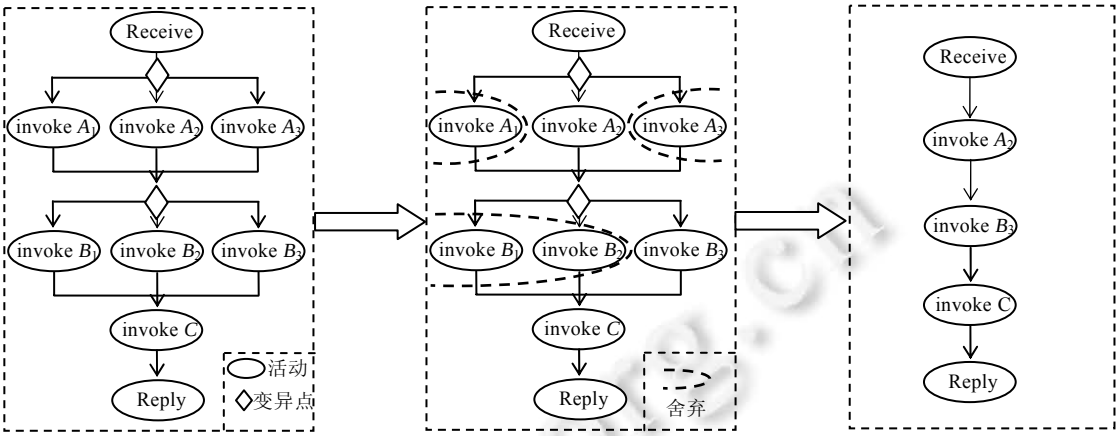
```
1.  ⟨? xml version="1.0" encoding="UTF-8"?⟩
2.  ⟨rundata⟩
3.    ⟨client username="userA"⟩
4.    ⟨configuration name="ProcessName"⟩
5.      ⟨VPChoices⟩
6.        ⟨VPChoice vpname="VariantPoint1" variant="variant1"/⟩
7.        ⟨VPChoice vpname="VariantPoint2" variant="variant3"/⟩
8.        ⟨VPChoice vpname="VariantPoint3" variant="variant4"/⟩
9.        ⟨VPChoice vpname="VariantPoint4" variant="variant2"/⟩
10.     ⟨/VPChoices⟩
11.   ⟨/configuration⟩
12.   ⟨/client⟩
13. ⟨/rundata⟩
```

Fig.9 Syntax for tenant configuration

图 9 租户配置定义的语法格式

图 10 示意了具体流程的派生过程.图 10(a)示例了一个采用抽象服务组装建模的抽象业务流程,该业务流程中预置 2 个变异点,每个变异点设置 3 个子流程变体,刻画了不同用户的个性化需求.在流程部署后,依据不同租户提供的用户配置文件对抽象业务流程进行配置.在每个变异点选择一个变体,派生出一个具体流程,如图

10(c)所示.在派生具体流程时,舍弃那些变异点下未被选择的变体,如图 10(b)所示.因此,对于不同的租户而言,抽象业务流程对他们是不可见的.需要指出的是,由于服务通常不需要用户界面,因此提出的方法侧重讨论服务组合实现的后台业务逻辑,而不关注如何开发软件的界面问题.



(a) 抽象业务流程 (b) 派生过程 (c) 具体业务流程

Fig.10 Process of deriving specific business process from abstract business process

图 10 具体业务流程的派生过程

2.3 基于可变性模型的SaaS软件的运行过程

上述基于可变性模型的 SaaS 软件开发框架可以用于构造可复用和可定制的 SaaS 软件.下面讨论如何借助本文开发的支持平台 VxSaaS 执行基于可变性模型的 SaaS 软件的一般过程(如图 11 所示).VxSaaS 的实现原理与技术将在第 3 节讨论.

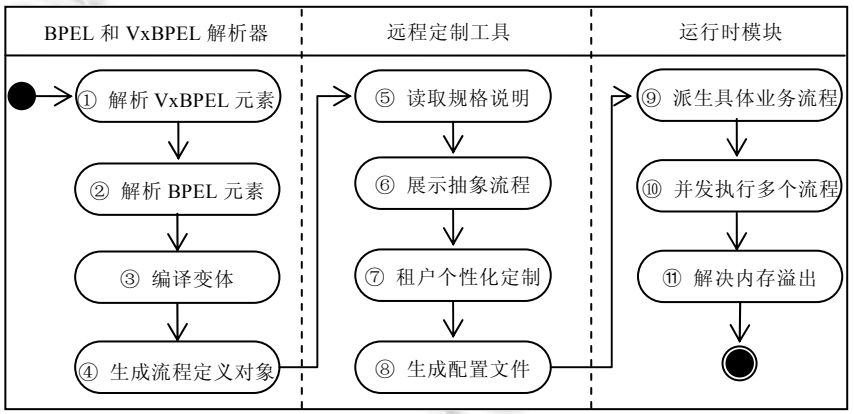


Fig.11 A general process of running SaaS Software based on variation model

图 11 基于可变性模型的 SaaS 软件运行的一般过程

第 1 阶段:解析抽象服务组装模型(基于 XML 的 VxBPEL 文件),包括 VxBPEL 解析和 BPEL 解析两部分,分别由引擎 VxBPEL_SasS 中的 VxBPEL 解析器和 BPEL 解析器完成.使用 VxBPEL 解析器对抽象服务组装中的可变性因素进行解析,提取变异点下的变体集合,存储变异点与变体之间的一对多的映射关系(步骤①);BPEL 解析器负责解析标准的 BPEL 元素(步骤②).解析过程将抽象服务组装中的所有变体无差别编译为 BPEL 对象(步骤③),生成一个抽象业务流程定义对象(步骤④),得到的抽象业务流程定义对象包括所有变体的信息.

第 2 阶段:远程定制租户的具体业务流程.需要借助远程定制工具 RTM4B 完成.首先,RTM4B 读取已经部署的抽象服务组装规格说明(步骤⑤),解析其中的可变性信息,以树形图的方式展示抽象流程(步骤⑥),供不同租户对流程进行个性化定制(步骤⑦).该工具收到租户请求后,首先对租户身份进行验证,然后根据租户的需求生成租户配置文件(步骤⑧).

第 3 阶段:并发执行具体流程.当接收从远程客户端发来的请求时,引擎 VxBPEL_SasS 中 BPEL 运行时模块的配置子模块根据租户身份,读取并解析租户提供的租户配置文件(语法格式如图 9 所示).根据配置文件对可变性流程定义文件进行剪裁,生成租户需求对象,从抽象流程模型中派生出新的具体流程(步骤⑨).依据租户的配置文件派生具体流程时,某个变异点下的变体之间是排他的,即在派生具体流程的过程中,某个变异点下只能选择一个变体,其他变体被舍弃.该阶段可以接受不同租户发来的请求,而且一个租户的需求不会对其他租户的流程产生任何影响.在验证租户身份后,并发执行相应的具体流程(步骤⑩).为了防止大量租户同时访问造成内存溢出,具体流程在执行后立即被销毁,再次执行时根据租户需求对象重新派生具体流程.该策略虽然可以有效防止因具体流程过多而导致内存泄露,但也导致流程执行的性能下降(因为每次执行流程之前都需要派生新的流程).为此,我们引入了优先销毁“最近最少使用的”策略(LRU 策略)对派生流程进行管理(步骤⑪),可以通过设定引擎的相关参数指明可以保留的具体流程的数量.如果某个租户希望改变配置文件或者有新租户加入时,则需返回远程定制阶段.

3 支持平台

我们实现了一个基于可变性模型的 SaaS 软件支持平台 VxSaaS,该平台由支持 SaaS 模式的服务组装引擎 VxBPEL_SaaS 和远程定制工具 RTM4B 组成.VxBPEL_SaaS 是对 VxBPEL 服务组装引擎 VxBPEL_ODE 的扩展,支持 SaaS 模式的服务组装的部署与解释执行,RTM4B 支持租户对 SaaS 软件进行远程个性化定制.

3.1 支持SaaS模式的服务组装引擎VxBPEL_SaaS

在课题组的前期研究工作中,通过扩展开源 BPEL 引擎 Apache ODE 开发了 VxBPEL 引擎 VxBPEL_ODE[7].本文通过扩展 VxBPEL_ODE 引擎中编译器模块和 BPEL 运行时模块,实现支持 SaaS 模式的服务组装引擎 VxBPEL_SaaS.在编译、部署时,保存了抽象业务流程的可变性设计信息;在执行阶段,根据用户配置文件解析并建立可变性元素之间映射关系,根据此映射关系执行具体流程,满足 SaaS 模式下“多态共存”与“需求隔离”的需求.

3.1.1 VxBPEL_SaaS 的系统架构

图 12 示意了 VxBPEL_SaaS 的系统架构.Apache ODE 引擎由编译器(compiler)、BPEL 运行时(ODE BPEL runtime)、集成层(ODE integration layer)和数据库系统(DBMS)等 4 个模块构成.为了能够解释执行 VxBPEL 服务组装,对编译器模块和 BPEL 运行时模块进行了扩展.具体来说,

- 编译器包括 BPEL 编译器(BPEL compiler)和 VxBPEL 编译器(VxBPEL compiler).BPEL 编译器负责将 BPEL 规格说明、WSDL 文件和各种数据结构定义(schema)编译成一个 Java 对象模型,该对象模型内封装了 BPEL 规格说明中所有的 BPEL 活动对象,各个活动对象通过接口、抽象类的继承等技术实现 BPEL 规格说明中流程活动之间的结构关系.VxBPEL 编译器负责解析 VxBPEL 定义的各种构造子活动,提取业务流程中可变性设计与配置信息,生成一对多的映射关系,并将其存储到数据结构 VPMap 中.VPMap 是一种列表结构,记录变异点名称与该变异点封装的所有变体信息.编译器为每个活动创建一个对象,当遇到变异点时,则在存储可变性信息的 VPMap 中查找,将获取到的所有变体单独编译成 Java 对象.
- BPEL 运行时是流程执行的核心模块,负责创建流程实例、创建运行时上下文、管理流程活动调用关系、消息路由等逻辑处理功能.其中,JaCOb(Java concurrent objects)提供不依赖线程的应用级并发机制,BPEL 流程调用执行过程中依赖 JaCOb 框架来实现 BPEL 的层次结构.该框架提供了必要的机制解决实现 BPEL 结构的两个关键问题:运行状态持久化和并发性.JaCOb 对象的所有处理过程全部发生在

虚拟处理单元(VPU)中,VPU 为每次请求创建一个供 BPEL 活动对象和 ODE 引擎进行交流的通道,然后创建一个执行队列作为 BPEL 活动对象的容器,此容器一般由通道进行管理,每次创建新的活动对象时,都会将新对象加入到此执行队列中.ODE 数据访问对象(ODE data access object)负责协调 BPEL 运行时与底层数据库之间的交互.通常情况下,数据源使用的是 JDBC 关系型数据库(DBMS),BPEL 运行时,通过数据访问对象(DAO)跟踪已创建的实例、路由消息到正在等待的实例、维护每个实例的 BPEL 变量和伙伴链接、实现 JaCOb 的持久化虚拟机的状态序列化等.

- 集成层负责外部 Web 服务的调用和与外界消息的传递过程.ODE 引擎提供针对 Axis2 和 JBI 的集成层.其中,Axis2 能够提供 Web 服务与 BPEL 运行时之间的通信通道和服务器端与客户端的 SOAP 操作框架.本文采用 Axis2 实现 Web 服务与 ODE 引擎中 BPEL 运行时之间的通信交互.

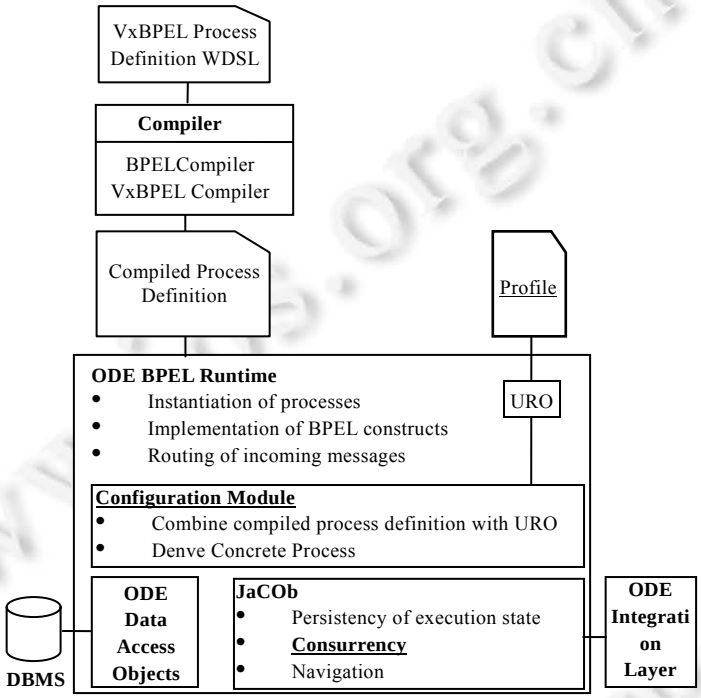


Fig.12 Architecture of VxBPEL_SaaS

图 12 VxBPEL_SaaS 架构

在讨论 VxBPEL_SaaS 总体架构之后,我们重点讨论如何支持流程运行时动态配置和多态共存的问题.

3.1.2 编译器模块的扩展与实现

VxBPEL_SaaS 引擎对 VxBPEL 流程进行编译与部署的过程如下.

(1) 启动轮询器检测新流程:引擎启动完成后,开启流程部署轮询器,根据部署文件夹的配置,每隔固定时间轮询检查部署路径下是否有新工程.若出现新工程,则检查工程文件夹下是否有新部署文件.若存在新部署文件,则进入解析流程阶段.

(2) 解析流程:以深度优先方式提取流程中的组成元素,并针对这些元素创建对应的活动对象,形成自顶向下的 BPEL 结构树.VxBPEL 编译器在 BPEL 结构树建立完成后启动,提取变体与变异点信息,存储到一个全局变量中.变异点和变体映射关系建立完成后,BPEL 编译器继续工作,将 BPEL 结构树从根结点开始遍历,根据注册的 BPEL 标签将 BPEL 活动标签转化成活动对象,并添加到过程对象中.该过程中遇到变异点时,则查找之前创建的变异点和变体集映射关系,将变异点转化成变体集合添加到执行队列中;同时,使用 Java 序列化技术,将过程对象序列化生成一个二进制文件,保存在数据库.上述编

译完成的抽象服务组装定义的业务流程产生两个备份:一份存在于运行时内存中,另一份存在于数据库中的序列化二进制文件.当前工程未被修改的情况下,下次启动引擎时则直接反序列化此二进制文件,生成过程对象载入到内存,不需要重新编译.过程对象创建完成后,返回流程对象集;收到此对象集后,则注册流程并得到一个流程 ID;最后,向轮询器返回流程 ID,编译工作完成.

(3) 工程监控:轮询器收到返回的流程 ID 后,为此流程创建一个标识文件,该流程部署成功.当工程监控器发现工程文件发生改变的时候,引擎自动启动编译器重新编译.当流程文件夹被删除的时候,如轮询器发现部署路径下有一个独立存在的部署标识文件,则启动反部署工具移除该流程.

通过上述编译与部署步骤生成了包含可变性信息的流程定义对象.可变性信息为多实例多租户运行模式下“多态共存”特性提供了支撑.然而,VxBPEL 服务组装引擎 VxBPEL_ODE 在无法支持 SaaS 模式下的“多态共存”的特性,因为在 VxBPEL 对象序列化的过程中,VxBPEL_ODE 根据可变性配置信息对相关变体进行选择性编译,本次未被选择的变体信息直接丢弃,从而导致每当流程动态切换时,需要重新编译 VxBPEL 流程.在实现 VxBPEL_SaaS 时,我们对 VxBPEL 编译器的实现策略进行了修改:在对象序列化的过程中不对变体信息进行舍弃,而是将这一过程延迟到运行时.因而,VxBPEL_SaaS 能够实现抽象流程的多个流程实例“多态共存”,而且在多实例多租户模式下切换流程时无需再次进行编译,实现了“一次开发、一次编译,一次部署、多次运行”的目标.

3.1.3 运行时模块的扩展与实现

对运行时模块扩展的目的在于实现引擎运行时,能够根据用户需求从抽象业务流程中派生出多个业务流程实例.为此,我们对 VxBPEL_ODE 进行了如下两个方面的扩展.

- 在图 12 中示意的 BPEL 运行时模块中增加一个配置子模块(configuration module).该模块能够拦截服务请求并读取请求中的身份信息,根据身份信息查找否存在此用户需求对象(URO).若存在,则从用户需求对象中获取用户关于此流程的配置方案,并将其与抽象流程定义进行绑定,派生出具体流程,将具体流程加载到内存中,最后释放服务请求.运行时模块收到请求后,则执行上一步生成的具体流程.若未找到用户需求对象,则说明此用户配置方案不存在(在编译时流程部署路径下的用户配置文件被编译生成相应的用户需求对象),引擎抛出“用户不存在”异常,停止此次服务请求过程.
- 尽管 BPEL 运行时模块中的 JaCOb 子模块提供了并发机制,但无法支持流程的多态并存.具体说来,虚拟处理单元允许多个通道同时执行,为流程并发执行提供了支撑.但是虚拟处理单元为每个流程只创建一个通道,所有用户共享通道中的执行队列,这导致不同租户之间的业务流程无法隔离.为了支持 SaaS 模式下不同租户之间的流程并发执行,我们对 JaCOb 的通信通道进行修改,使其每次为同一个流程的不同租户创建新的执行队列.

经过上述对 BPEL 运行时的扩展,VxBPEL_SaaS 在流程执行阶段能够动态创建不同的流程实例,并允许不同流程实例多态共存.VxBPEL_SaaS 执行阶段的主要过程包括:

- 扩展后的引擎在收到用户请求消息后,首先拦截该消息并获取消息中的用户身份信息.根据此信息,查找是否存在此用户的用户需求对象:若存在,则将该消息放行;否则,根据用户配置文件建立用户需求对象.当某个抽象业务流程编译、部署之后,租户可以通过远程定制工具 RTM4B 对流程进行个性化定制,产生配置文件;否则,引擎按照默认配置进行处理(不执行任何活动).
- 根据用户需求对象创建相应的引擎服务,将用户需求对象与抽象业务流程绑定,动态配置生成具体业务流程.我们对 VxBPEL 编译器的实现策略进行了修改,将创建具体业务流程的过程延迟到流程执行阶段,因此,BPEL 运行时模块中始终保存原始的抽象业务流程与可变性信息.
- 扩展后的 JaCOb 子模块通过多任务并发处理机制,为所有请求的具体业务流程分别创建不同的执行对列,实现了运行时不同流程实例的多态共存.

3.2 远程定制工具RTM4B

RTM4B 支持租户远程完成流程的部署、配置与调用,对业务流程部署、配置、运行、卸载提供全过程支持.该工具使用 Struts2 框架实现,提供如下功能.

- 用户与权限管理:工具将系统的用户角色分为系统管理员、应用管理员和普通租户.系统管理员是服务器的拥有者,应用管理员是应用的拥有者并租用服务器,普通租户是应用的使用者.不同角色的用户对流程和服务器的管理有着不同权限.系统管理员拥有最高权限;应用管理员管理属于自己的流程和用户;通过认证的租户拥有配置自己流程的权限,每个租户都有属于自己的流程,并对已租用的流程进行管理.
- 流程管理:包括购买流程、展示流程、部署流程、查看运行记录、反部署流程等.展示流程:以树形图的形式展示抽象流程及可变性信息,租户借助该树形图获知如何配置流程满足自己的需求;查看运行记录:以日志的形式查看流程部署、执行过程;反部署流程:当流程不需要或者其他原因需要销毁时,租户可以选择指定流程部署文件实现流程卸载.
- 配置流程:租户可以在运行时改变流程的配置,满足不同的业务需求.浏览已有配置:在树形图中高亮显示被选的变体.有两种创建新配置的支持方式:(1) 流程树配置:以树形图的形式展示所有的变异点和变体,对变异点与变体进行选择,验证流程的完整性;(2) 文件配置:按照标准的配置文件格式,创建新的配置文件.
- 运行流程:租户配置完流程之后,可以在此运行自己定制的流程.

4 实例研究

采用一个汽车组装实例系统验证所提 SaaS 软件开发方法的可行性及其支持平台的性能.

4.1 实验目的及实验设计

通过实例研究,我们希望回答如下 3 个问题.

(1) 所提基于可变性模型的可复用与可定制 SaaS 软件开发方法能否支持“一次开发、一次部署、多租户多实例”的目标?采用 VxBPEL 设计与实现支持“汽车组装”业务流程的 SaaS 的软件,讨论如何支持不同租户的需求构造出多个实例业务流程(第 4.2 节).

(2) 开发的支持平台能否在运行时支持多个租户执行不同业务流程实例,不同流程实例之间能否隔离、互不影响?采用支持平台部署与执行基于 VxBPEL 的“汽车组装”业务流程,即在云端服务器,采用 RTM4B 工具部署与管理开发的 VxBPEL 规格说明,采用 VxBPEL_SaaS 引擎解释执行的 VxBPEL 规格说明.远程租户通过 RTM4B 工具读取业务流程树,对流程中的变异点下的变体进行配置,生成配置文件并导出到流程工程路径下.验证 VxBPEL_SaaS 引擎能否依据配置文件执行不同的业务流;采用多个租户同时访问同一个业务流程的时候,验证能否实现用户需求隔离(第 4.3 节).

(3) 开发的支持平台在 SaaS 软件部署与运行时切换的时间开销如何?分别采用前期版本的 VxBPEL_ODE 引擎和扩展后的 VxBPEL_SaaS 引擎对基于 VxBPEL 的“汽车组装”流程不同租户配置方案进行测试,记录“汽车组装”的部署时间、初始配置开销(租户在运行前对流程配置的时间)、执行时间(测试用例调用流程的平均执行时间)、切换开销(从切换配置到流程执行的时间)以及切换配置占空比(切换配置的时间占总时间的比例),评估所开发平台的运行效率(第 4.4 节).

4.2 实例系统及其基于VxBPEL的构造

“汽车组装”业务流程用于辅助汽车装配过程中各个零部件的订购与组装.通常,一个汽车生产线不只完成单一车型的生产,不同车型的生产装配流程中有大量相同的过程.为每一种车型的装配开发一个专属的控制流程不仅产生大量的冗余,而且增加维护复杂度.因此,有必要采用一个通用装配流程支持不同车型的装配过程.我们采用基于 VxBPEL 的服务组装方式模拟实现这样的装配过程,将该业务流程部署到云端服务器,不同的租户根据自己的需要运行流程实例,实现不同车型的模拟装配过程.

基于可变性模型的汽车装配的业务流程如图 13 所示.该过程包含 10 个活动,其中,“安装油箱”和“安装内饰”“安装座椅”和“安装方向盘”均属于并发流程.为了支持多种车型的装配,在不同的装配活动处设置变异点,每

个变异点下设置多个变体.该示例流程中,8 处活动存在不确定性因素,不妨在示例流程中每个变异点下设置 3 个变体.以安装油箱(FuelTank)活动为例,该变异点下设置 3 个变体 $FuelTank_A$,$FuelTank_B$ 和 $FuelTank_C$,该业务流程片段的 VxBPEL 代码如图 14 所示.类似的,我们可以定义不同变体之间的约束关系,从而得到基于 VxBPEL 的汽车装配业务流程.

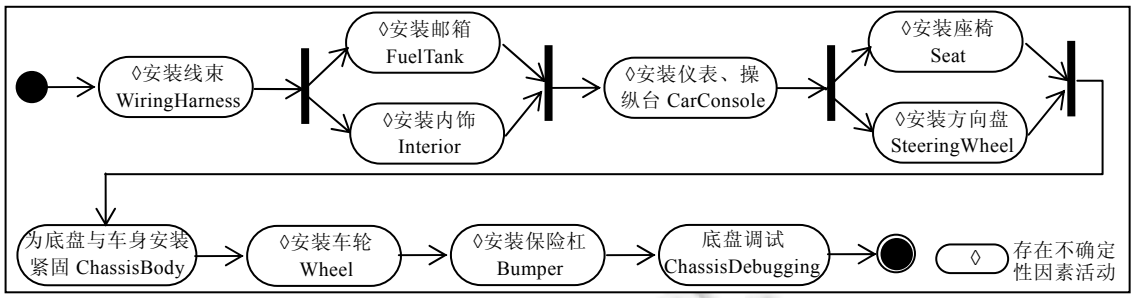


Fig.13 Workflow of automobile assembly line with variation design

图 13 具有可变性设计的汽车组装生产线业务流程

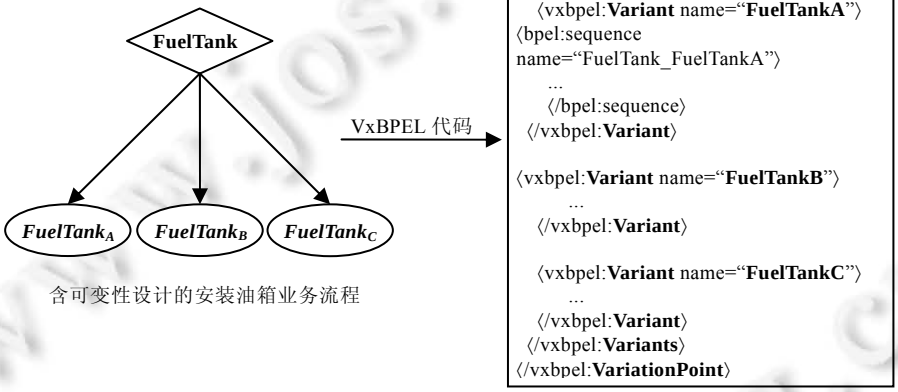


Fig.14 Illustration of VxBPEL code for FuelTank activitity

图 14 安装油箱活动的 VxBPEL 代码示例

我们采用服务组装引擎 VxBPEL_SaaS 部署与执行所开发的 VxBPEL 规格说明.在流程部署后,租户采用 RTM4B 工具对部署的汽车组装流程进行配置.图 15 示意了一个租户的配置文件,由该配置文件可知,抽象汽车组装模型包含"WiringHarness""FuelTank""Interior""CarConsole""Seat""SteeringWheel""Bumper""Bumper"这 8 个变异点,指明了不同变异点下变体的选择情况.

例如,安装油箱活动变异点"FuelTank"选择了"FuelTankB"变体.为简明起见,我们用"VP_1",...,"VP_8"分别代表变异点"CarConsole""Interior""Seat""Bumper""WiringHarness""FuelTank""SteeringWheel"和"Wheel";用"*A*""*B*""*C*"分别表示选择当前变异点处的"变体 *A*""变体 *B*"和"变体 *C*".VxBPEL_SaaS 通过分析该配置文件,为此租户派生出一个具体流程.类似的,其他租户通过各自的配置文件进行远程流程定制.

上述实例研究中,我们仅设计并维护一个抽象服务组装模型,针对不同的汽车装配需求,通过支持平台派生出不同的具体装配流程,实现不同车型的装配.实例研究结果表明,提出的方法采用服务组装的方式构造 SaaS 软件,首先设计并部署一个能满足共性需求的抽象服务组装模型;然后,针对不同租户的个性化需求,从抽象服务组装模型中动态派生出相应的具体流程实例,实现了"一次开发、一次部署、多租户多实例"目标.

```
1. ⟨vxbpel:VPChoices name="VPChoices"⟩
2.     ⟨vxbpel:VPChoice vpname="CarConsole" variant="CarConsoleA"/⟩
3.     ⟨vxbpel:VPChoice vpname="Interior" variant="InteriorB"/⟩
4.     ⟨vxbpel:VPChoice vpname="Seat" variant="SeatA"/⟩
5.     ⟨vxbpel:VPChoice vpname="Bumper" variant="BumperA"/⟩
6.     ⟨vxbpel:VPChoice vpname="WiringHarness" variant="WiringHarnessB"/⟩
7.     ⟨vxbpel:VPChoice vpname="FuelTank" variant="FuelTankB"/⟩
8.     ⟨vxbpel:VPChoice vpname="SteeringWheel" variant="SteeringWheelA"/⟩
9.     ⟨vxbpel:VPChoice vpname="Wheel" variant="WheelA"/⟩
10. ⟨/vxbpel:VPChoices⟩
```

Fig.15 Illustration of a tenant profile

图 15 租户配置文件示例

4.3 执行多租户多实例的汽车组装流程

为了验证开发的支持平台能否在运行时支持多个租户执行不同业务流程实例、不同流程实例之间能否隔离,我们在客户端模拟 10 个租户,随机生成 10 种不同配置方案,将这些配置方案存为不同的用户配置文件.首先,采用 VxBPEL_SaaS 部署与执行开发的 VxBPEL 规格说明;然后,采用 RTM4B 工具读取随机生成的 10 种配置文件并向 VxBPEL_SaaS 发送请求;VxBPEL_SaaS 接受租户的请求后,依据用户配置文件为其派生具体流程,并发执行这些流程.表 2 列出了上述 10 种汽车组装的流程配置方案及其运行结果,其中,用"用户名"栏表示租户的 ID;"配置文件"栏表示当前租户的配置文件名;"配置方案/运行结果(变体选择)"栏列出了当前租户的配置方案及运行结果,即不同变异点处的变体选取情况;"运行时间(ms)"栏列出当前租户的运行时间(实验环境配置见表 1).实验结果表明,开发的支持平台能够根据租户配置文件派生出不同的流程实例,而且各个流程实例的执行是相互隔离的、互不影响.

Table 1 Setting of experimental environment

表 1 实验环境配置

配置项	参数值
CPU	3.60*4GHz
内存	4GB
硬盘	500GB
操作系统	Windows7-64bit

Table 2 Multi-tenant and multi-instance process configuration schema and their performance

表 2 多租户多实例流程配置方案及其性能

用户名	配置文件	配置方案/运行结果(变体选择)								运行时间(ms)
		VP_1	VP_2	VP_3	VP_4	VP_5	VP_6	VP_7	VP_8	
testUser00	testUser00.uccf	*A*	*B*	*C*	*C*	*B*	*A*	*C*	*C*	773
testUser01	testUser01.uccf	*A*	*C*	*A*	*C*	*A*	*A*	*A*	*B*	686
testUser02	testUser02.uccf	*C*	*C*	*A*	*B*	*C*	*A*	*A*	*C*	636
testUser03	testUser03.uccf	*A*	*A*	*B*	*A*	*A*	*B*	*A*	*C*	617
testUser04	testUser04.uccf	*A*	*B*	*C*	*A*	*B*	*C*	*B*	*C*	685
testUser05	testUser05.uccf	*B*	*A*	*B*	*A*	*C*	*C*	*B*	*C*	950
testUser06	testUser06.uccf	*C*	*B*	*A*	*C*	*C*	*C*	*C*	*C*	547
testUser07	testUser07.uccf	*B*	*A*	*C*	*A*	*B*	*A*	*C*	*B*	511
testUser08	testUser08.uccf	*B*	*C*	*C*	*A*	*A*	*A*	*C*	*B*	541
testUser09	testUser09.uccf	*C*	*B*	*B*	*C*	*C*	*A*	*A*	*A*	727

4.4 支持平台的性能评估

为了进一步评估 VxBPEL_SaaS 和 VxBPEL_ODE 的执行效率,我们采用两个版本的引擎分别执行同一租户的"汽车组装"不同流程配置(实验环境配置见表 1)."汽车组装"流程中共设置 8 个变异点 $VP_i(1\leqslant i\leqslant 8)$,即使每个变异点下设置 3 个变体,则有 3^8 种配置方案.由于数量庞大,采用抽样测试方法,选取了表 3 所示的 8 种配置方案 $V_i(1\leqslant i\leqslant 8)$.

实验中,我们分别采用 VxBPEL_SaaS 和 VxBPEL_ODE 部署与执行基于 VxBPEL 的汽车组装规格说明,按照表 3 示例的配置方案执行不同的组装流程.从流程部署、执行和切换配置的时间开销几个方面评估 VxBPEL_SaaS 的性能.部署时间指流程发布到流程部署成功的时间开销,由于 VxBPEL_SaaS 和 VxBPEL_ODE 对于不同配置都是一次部署,两个引擎对于相同的 VxBPEL 规格说明的时间开销是一致的,因此省去二者部署时间的比较.分别记录 VxBPEL_SaaS(ODE2)和 VxBPEL_ODE(ODE1)的初始配置时间开销、执行时间开销、切换时间开销以及切换配置的占空比,实验结果如图 16~图 19 所示.

Table 3 Configuration schema based on tenant profiles

表 3 租户配置方案

Scheme	VP_1	VP_2	VP_3	VP_4	VP_5	VP_6	VP_7	VP_8
V_1	*A*	*A*	*A*	*A*	*A*	*A*	*A*	*A*
V_2	*B*	*B*	*B*	*B*	*B*	*B*	*B*	*B*
V_3	*C*	*C*	*C*	*C*	*C*	*C*	*C*	*C*
V_4	*A*	*A*	*A*	*A*	*B*	*B*	*B*	*B*
V_5	*B*	*B*	*B*	*B*	*C*	*C*	*C*	*C*
V_6	*A*	*A*	*A*	*A*	*C*	*C*	*C*	*C*
V_7	*A*	*B*	*C*	*A*	*B*	*C*	*A*	*B*
V_8	*C*	*B*	*A*	*C*	*B*	*A*	*C*	*B*

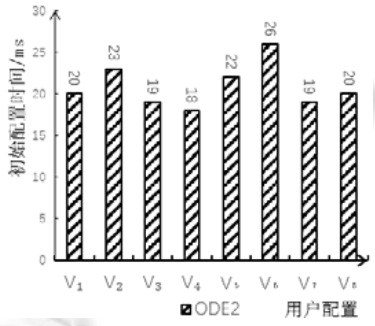


Fig.16 Initial configuration time

图 16 初始配置时间

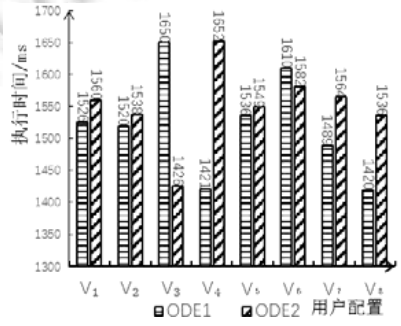


Fig.17 Execution time

图 17 执行时间

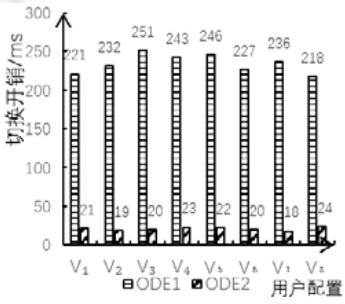


Fig.18 Switching overhead

图 18 切换开销

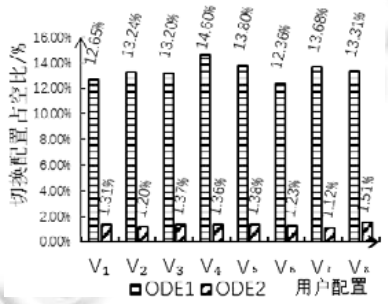


Fig.19 Duty ratio of configuration switching

图 19 配置切换占空比

初始配置时间指运行前对 VxBPEL 流程进行配置的时间开销.由于 VxBPEL_ODE 配置方案预置在规格说明中,不需要配置便可直接运行;VxBPEL_SaaS 必须先进行配置才可以运行,时间开销指将配置文件导出到部署路径的时间.执行时间指使用调用 VxBPEL 流程的平均执行时间开销.配置切换时间指用户从开始切换配置到流程能够开始执行的时间开销.配置切换占空比指切换配置时间占总时间的比例,可由如下计算公式得到:

切换配置占空比=切换配置时间/(切换配置时间+执行时间+初始配置时间).

该值越小,则表明部署的流程可用性越高.

由图 16~图 19,我们可以得到如下结论.

(1) 在初始配置方面,VxBPEL_SaaS 存在很小的时间开销(20ms 左右),VxBPEL_ODE 则无需进行初始化配置,因此时间开销为 0.

(2) 在执行性能方面,VxBPEL_SaaS 与 VxBPEL_ODE 的执行时间相当,因为流程执行的主要时间开销在于流程执行过程中的服务调用,而变体选择的时间开销相对较小.部分配置方案下(V_1,V_2,V_4,V_5,V_7 和 V_8),VxBPEL_ODE 小于 VxBPEL_SaaS;部分配置方案下(V_3 和 V_6),后者小于前者(与服务调用的响应时间的随机性有关).

(3) 在配置切换方面,VxBPEL_SaaS 的配置切换时间明显少于 VxBPEL_ODE.

(4) 在切换配置占空比方面,VxBPEL_ODE 是 VxBPEL_SaaS 的 10 倍左右.

上述实验结果表明,本文开发的支持 SaaS 模式的服务组装引擎 VxBPEL_SaaS 与课题组前期开发的服务组装引擎 VxBPEL_ODE 相比,前者总体性能相当或稍优于后者,但在切换配置占空比方面,前者远远优于后者.本文开发的支持平台能够支持在具体流程之间进行更快速切换,对不同租户的响应时间更短.

4.5 小 结

采用“汽车组装”流程验证了基于可变性模型构造可复用与可定制 SaaS 软件的可行性,评估了支持平台的性能.实例研究结果表明:

(1) 基于可变性模型的 SaaS 软件开发方法提供了一种可复用、可定制的 SaaS 软件开发途径.该方法采用可变性模型表达不同租户之间的共性需求与差异性需求,首先,基于抽象服务组装模型实现抽象业务流程;然后,针对租户差异性需求派生不同的具体流程,不同流程之间共享领域服务集.增加新租户时,只需针对新租户的配置文件进行远程流程定制,因此,该方法能显著提升多租户环境下软件开发效率.

(2) 开发的支持平台 VxSaaS 支持多租户多实例运行模式,具有良好的性能.开发的支持平台能够部署、执行和定制基于可变性模型的抽象服务组装模型,在运行时刻,动态派生面向不同的租户的流程实例、不同租户之间的配置切换过程互相透明、支持同一流程的“多态共存”.

5 相关工作

从可配置工作流技术[19]与基于 SOA 的 SaaS 软件开发技术两个方面介绍与本文紧密相关的研究工作.

5.1 可配置工作流技术

Gottschalk 等人提出了一种可配置的工作流建模方法[20],该方法对主流的工作流建模语言进行扩展,引入可配置性元素.采用该方法首先得到包含预先定义可配置元素的工作流模型,运行时,根据需求对可配置性元素进行选择.通过在可配置性元素之间设置依赖关系,实现不同模型之间的切换.

Hallerbach 等人提出了一种业务流程生命周期过程中变体集合的管理方法 Provop[21].在建模阶段,首先针对一个基本的业务流程模型识别出各种变体流程模型,通过操作(插入、删除、移动、改变)描述基本模型与派生的变体流程模型之间的不同之处.采用可视化方式支持复杂的变体流程建模,即在基本模型中,将能够发生特定变化的地方标识为可调整点,将可调整点可用来得到变体流程模型的操作展示出来.

Pesic 等人提出了一种基于约束的流程建模框架[22],该框架采用声明式的建模方法,避免流程中出现过于繁琐的规格说明,并能支持无法避免的变化.在此基础上,提出一种基于约束的流程建模语言 ConDec,使用一个约束模板定义活动间的关系,可以通过增加、移除和组合约束条件实现运行时的可变性配置.与传统的流程建模框架相比,该框架具有更好的灵活性和动态适应性.

类似的,Lu 等人提出一种支持基于约束的灵活流程管理框架[23],该框架由业务流程约束和流程变体库两个部分组成,前者使用流程约束作为基本概念以支持流程可变性,开发者需要在设计阶段定义流程中的约束关系,为每个流程变体设置必要的约束关系并生成约束集,在运行时可以根据事先定义的约束集动态调整实例模版

并通过活动池得到可用的流程活动;后者为变体的存储和评价提供支持,以实现高效率的查找和检索.

赵俊峰等人提出了一种支持领域特性的 Web 服务组装方法[24],该方法针对一组具有相似或相近软件需求的应用系统所需要的 Web 服务,通过领域组装模型和用户的需求来决定软件中所需要的服务.需求之间存在的依赖、互斥等关系可以帮助指导软件的组装,采用线性规划模型将问题转化为一组线性约束条件和最优线性目标函数,所求最优解即可用来指导进行服务组装.

已有工作从约束或过程演化等角度探讨了工作流系统中的可配置性问题,本文工作则从可变性管理的角度探索基于服务组装的 SaaS 软件的可配置问题.传统的工作流系统通常单独部署与执行,而 SaaS 软件通过协调部署在云端的服务实现各种业务流程.已有工作流可配置技术无法直接用来解决 SaaS 软件的可配置性问题,为此,本文引入基于可变性设计的抽象服务组装模型,基于该模型,无需维护不同流程实例,通过运行时可变性配置的切换满足不同租户的需求.因此,采用本文提出的方法开发的 SaaS 软件不仅具有良好的可配置性,而且具有高度的可复用性(流程定义层的复用).

5.2 基于SOA的SaaS软件开发技术

采用 SOA 架构的 SaaS 软件通过指定服务调用的顺序以及如何调用相关服务满足新的需求,因而易于集成现有的松散耦合的服务.当不同用户对该实例软件的需求不同时,单实例多租户的 SaaS 软件如何解决“多态共存”的问题非常突出.部分研究工作探讨了基于 SOA 的业务流程定制问题.

Kapuruge 等人提出了一种面向 SaaS 软件业务过程建模方法,支持以不同的版本交付给多个用户使用[2],重点讨论了如何让需要相同服务、但是需求略有不同的用户共享同一个服务的解决方案.提出的 RoSaaS 实例系统分为 3 层:第 1 层为结构层,定义了所有的服务;第 2 层为行为层,依照预定义的约束关系将相关服务组合成一个行为,供不同的流程调用,一个行为会被多个流程所使用;最高层则是包装层,根据客户的不同需求,将所需要的行为组装成一个流程定义,不同流程定义可能用到同一个行为,不同行为也可能用到同一个服务.当不同用户对同一个行为有着不同需求的时候,则重新定义一个行为并继承原行为,并修改部分行为以满足不同的需求.该方法有效解决了如何让需要相同服务但需求略有不同的用户共享同一个服务,即如何让服务“求同存异”,但并未考虑如何让按照不同的用户的需求对业务流程进行定制.本文提出的方法不仅支持服务层的复用,而且解决了流程定义层的复用问题.

Mietzner 提出了一种应用程序模板的概念[25],通过事先明确 SaaS 软件中的可变性,在流程中定义变异点并附以应用程序模板,通过裁剪应用程序模板来满足不同的用户需求,从而实现软件的可定制化.该方法中没有定义变体之间的约束关系,因此变体的选择是任意的,变体之间的潜在冲突将影响到应用程序的服务质量.本文提出的 SaaS 软件方法基于我们早期工作中提出的 VxBPEL,VxBPEL 不仅提供了描述不同变体之间依赖关系的构造子,还支持变体之间的多种约束关系的表达.

Shi 等人设计了一种采用 BPEL 定义 SaaS 业务流程的框架[26],该框架便于部署与运行时的服务和流程的可定制化,完全依赖 BPEL 引擎和 Web 服务,允许用户在定义流程的时候自定制服务和流程,在部署时根据用户提供的规则验证和检查流程逻辑上的正确性、动态定制、动态替换以及处理异常.类似的,Pathirage 等人提出一种基于 BPEL 的多租户架构[27],通过允许租户自己在公开的引擎上部署自己的流程来实现多租户,但不支持已部署业务流程的定制.

熊伟等人提出了一种基于 O-RGPS 需求元建模的 SaaS 的架构[28],该架构将租户看作主体,将服务作为主导,根据用户需求信息搜索相关的服务进行组合,并传递给形式化验证引擎,逐层向上验证,最终生成相关层的描述信息.通过这种方式,有效降低了服务定制的复杂性.

葛秀豪等人提出了基于 SaaS 模式的流引擎服务模型[1],该模型的用户通过 Web 展现层与流程引擎接口层进行交互,业务流程的管理、部署监测等任务由用户完成.租户通过 BPEL 引擎对服务进行编排和组合,形成用户所需要的业务流程,这里的服务不仅是本地服务,也可以是云端的各种服务,从而保证用户能够更方便、快捷地获得所需要的服务.Web 服务供应商部分的核心是 BPEL 引擎,能够根据用户配置创建流程实例.

OSF[29]是一种离线 SaaS 应用框架,该框架能够感知外部网络连接的动态变化,根据网络连接情况改变自己

的运行方式,能够持续不断地为用户提供服务.在网络拥挤或者完全离线的情况下,为所有操作生成了离线实例,这些离线实例存在于客户端的缓存中.在连接恢复时,保存在服务器中的在线实例使用操作同步构件进行冲突检测和处理,将在线实例和离线实例同步.

本文提出的基于可变性模型的 SaaS 软件开发方法是对基于可变性管理的适应性服务组装方法的继承与扩展:一方面,我们采用前期工作中开发的 VxBPEL 构造抽象服务组装模型;另一方面,我们重点解决了基于 VxBPEL 构造 SaaS 软件时需要解决的"需求隔离"和"多态共存"两个关键问题,并通过对 VxBPEL 引擎的扩展开发了 SaaS 软件开发支持平台.

6 总 结

为了提高云计算环境下 SaaS 软件的可复用性与可定制性,本文提出了一种基于可变性模型的 SaaS 软件开发方法.该方法采用基于服务组装方式构造的 SaaS 软件,在设计阶段引入了抽象服务组装模型与运行时动态配置实现个性化定制,为不同租户派生出不同的流程实例,能够在不修改 SaaS 软件的情形下满足多个租户的不同需求.本文提出的 SaaS 软件开发方法不仅增强了 SaaS 软件的可复用性,还增强了 SaaS 软件的可定制性.一方面,通过分析 SaaS 软件不同租户的差异化需求,在抽象服务组装模型中预期发生变化的地方设置变异点,变异点下设置多个供运行时选择的变体,解决在运行时不同租户对 SaaS 软件的动态定制问题;另一方面,该方法能够支持面向某个用户业务流程的运行时切换,不影响到其他用户,即在运行时刻,不同租户执行不同的业务流程.这解决了运行时面向不同租户的业务流程并发执行与隔离问题.采用该方法开发的 SaaS 软件满足"多态共存"与"需求隔离"的特性.

本文通过扩充基于可变性管理的适应性服务组装平台,开发了 SaaS 软件支持平台.该平台为基于服务组装开发的 SaaS 软件提供了部署、运行时配置、解释执行的支撑环境,包括支持隔离不同租户业务流程的引擎 VxBPEL_SaaS 和支持运行时业务流程远程配置工具 RTM4B.采用一个 SaaS 软件实例评估了本文提出的 SaaS 软件开发方法与支持平台,实验结果表明了该方法是可行的,且支持平台具有良好的性能.

我们将在如下几个方面进一步完善 SaaS 软件开发方法与支持平台:(1) 增加对多租户业务流程执行的性能监控,通过考虑虚拟机负载均衡,进一步支持多实例多租户 SaaS 软件的运行时性能优化问题;(2) 解决多实例多租户 SaaS 软件的业务流程执行的异常恢复问题,即当引擎运行不同租户的业务流程出现故障时,保存不同租户的业务流程的运行状态、自动执行故障点之前的流程步骤.近年来,微服务及容器技术等在云服务开发方面有着广泛的应用,如何将本文的方法应用于这些新的技术体系,是一个值得研究的方向.

References:

[1] Ge XH, Lu HH, Ding AX. Study on process engine service model based on SaaS. Telecommunications Information: Networking and Communications, 2010,265(12):27−30 (in Chinese with English abstract).

[2] Kapuruge M, Colman A, Han J. Achieving multi-tenanted business processes in SaaS applications. In: Proc. of the 12th Int'l Conf. on Web Information System Engineering (WISE 2011). Berlin: Springer-Verlag, 2011. 143−157.

[3] Grivas SG, Kumar TU, Wache H. Cloud broker: Bringing intelligence into the cloud. In: Proc. of the IEEE 3rd Int'l Conf. on Cloud Computing (CLOUD 2010). Washington: IEEE Computer Society, 2010. 544−545.

[4] Sun CA, Rossing R, Sinnema M, Bulanov P, Aiello M. Modeling and managing the variability of Web service-based systems. Journal of Systems and Software, 2010,83(3):502−516.

[5] Sun CA, Aiello M. Towards variable service compositions using VxBPEL. In: Proc. of the 10th Int'l Conf. on Software Reuse: High Confidence Software Reuse in Large Systems (ICSR 2008). Berlin: Springer-Verlag, 2008. 257−261.

[6] Sun CA, Xue TH, Aiello M. ValySeC: A variability analysis tool for service compositions using VxBPEL. In: Proc. of the 5th IEEE Asia-Pacific Services Computing Conf. (APSCC 2010). Washington: IEEE Computer Society, 2010. 307−314.

[7] Sun CA, Wang P, Zhang X, Aiello M. VxBPEL_ODE: A variability enhanced service composition engine. In: Proc. of the Asia-Pacific Web Conf. 2014 Workshops on Web Technologies and Applications (APWeb 2014). Springer Int'l Publishing, 2014. 69−81.

[8] Sun CA, Xue TH, Hu CJ. VxBPELEngine: A change-driven adaptive service composition engine. Chinese Journal of Computers, 2013,36(12):2441−2454 (in Chinese with English abstract).

[9] Koning M, Sun CA, Sinnema M, Avgeriou P. VxBPEL: Supporting variability for Web services in BPEL. Information & Software Technology, 2009,51(2):258−269.

[10] Zhang X. Research on design technique for reusable and customizable business processes in the cloud computing context and its supporting tool [MS. Thesis]. Beijing: University of Science & Technology Beijing, 2016. 1−78 (in Chinese with English abstract).

[11] Wang PP. An adaptive service composition technique via variability management and dynamic binding [MS. Thesis]. Beijing: University of Science & Technology Beijing, 2015. 1−67 (in Chinese with English abstract).

[12] Xue TH. Research and implementation on a supporting platform for VxBPEL-based adaptive service compositions [MS. Thesis]. Beijing: University of Science & Technology Beijing, 2013. 1−68 (in Chinese with English abstract).

[13] Wang K. Research on variability design and management technique for service compositions and its supporting platform [MS. Thesis]. Beijing: University of Science & Technology Beijing, 2014. 1−79 (in Chinese with English abstract).

[14] Griss ML. Software reuse: Architecture, process and organization for business success. In: Proc. of the 8th Israeli Conf. on Computer Systems and Software Engineering. Washington: IEEE Computer Society, 1997. 86−89.

[15] Tsai WT, Song WW, Paul R, Cao ZB, Huang H. Services-Oriented dynamic reconfiguration framework for dependable distributed computing. In: Proc. of the 28th Annual Int'l Computer Software and Applications Conf. (COMPSAC 2004). Washington: IEEE Computer Society, 2004. 554−559.

[16] Topaloglu NY, Capilla R. Modeling the variability of Web services from a pattern point of view. In: Proc. of the European Conf. on Web Services (ECOWS 2004). Berlin: Springer-Verlag, 2004. 128−138.

[17] Sinnema M, Deelstra S, Nijhuis J, Bosch J. COVAMOF: A framework for modeling variability in software product families. In: Proc. of the 3rd Int'l Conf. on Software Product Lines (SPLC 2004). Berlin: Springer-Verlag, 2004. 197−213.

[18] Chong F, Carraro G. Architecture strategies for catching the long tail. 2006. https://msdn.microsoft.com/en-us/library/aa479069.aspx

[19] Bachmann F, Bass L. Managing variability in software architectures. In: Proc. of the Symp. on Software Reusability: Putting Software Reuse in Context (SSR 2001). New York: ACM Press, 2001. 126−132.

[20] Gottschalk F, Van Der Aalst WMP, Jansen-Vullers MH, Rosa ML. Configurable workflow models. Int'l Journal of Cooperative Information Systems, 2008,17(2):177−221.

[21] Hallerbach A, Bauer T, Reichert M. Managing process variants in the process lifecycle. In: Proc. of the 10th Int'l Conf. on Enterprise Information Systems (ICEIS 2008). 2008. 154−161.

[22] Pesic M, Schonenberg MH, Sidorova N, Van Der Aalst WMP. Constraint-Based workflow models: Change made easy. In: Proc. of the CoopIS, DOA, ODBASE, GADA, and IS on the Move to Meaningful Internet Systems 2007 (OTM 2007). Berlin: Springer-Verlag, 2007. 77−94.

[23] Lu R, Sadiq S, Governatori G. On managing business processes variants. Data & Knowledge Engineering, 2009,68(7):642−664.

[24] Zhao JF, Xie B, Zhang L, Yang FQ. A Web services composition method supporting domain feature. Chinese Journal of Computers, 2005,28(4):731−738 (in Chinese with English abstract).

[25] Mietzner R, Metzger A, Leymann F, Pohl K. Variability modeling to support customization and deployment of multi-tenant-aware software as a service applications. In: Proc. of the ICSE Workshop on Principles of Engineering Service Oriented Systems (PESOS 2009). Washington: IEEE Computer Society, 2009. 18−25.

[26] Shi YL, Luan S, Li QZ, Wang HY. A flexible business process customization framework for SaaS. In: Proc. of the WASE Int'l Conf. on Information Engineering (ICIE 2009). Washington: IEEE Computer Society, 2009. 350−353.

[27] Pathirage M, Perera S, Kumara I, Weerawarana S. A multi-tenant architecture for business process executions. In: Proc. of the IEEE Int'l Conf. on Web Services (ICWS 2011). Washington: IEEE Computer Society, 2011. 121−128.

[28] Xiong W, Li B, He P, Huang Y, Dong DD. A kind of innovation model for construction of SaaS service. Microelectronics & Computer, 2012,29(9):141−144 (in Chinese with English abstract).

[29] Song J, Hou HY, Zhu ZL. OSF: A component-based framework supporting offline SaaS application. Chinese Journal of Computer Science, 2012,39(10):125−130 (in Chinese with English abstract).

附中文参考文献:

[1] 葛秀豪,卢捍华,丁傲西.基于 SaaS 模式的流程引擎服务模型研究.电信快报:网络与通信,2010,265(12):27−30.

[8] 孙昌爱,薛铁恒,胡长军.VxBPELEngine:一种变化驱动的适应性服务组装引擎.计算机学报,2013,36(12):2441−2454.

[10] 张鑫.云计算环境下可复用与可定制业务流程设计技术与支持工具研究[硕士学位论文].北京:北京科技大学,2016.1−78.

[11] 王攀攀.基于可变性管理与动态绑定相结合的适应性服务组装方法研究[硕士学位论文].北京:北京科技大学,2015.1−67.

[12] 薛铁恒.基于 VxBPEL 的适应性服务组装平台的研究与实现[硕士学位论文].北京:北京科技大学,2013.1−68.

[13] 王可.服务组装中的可变性设计与管理技术及其支持平台研究[硕士学位论文].北京:北京科技大学,2014.1−79.

[24] 赵俊峰,谢冰,张路.一种支持领域特性的 Web 服务组装方法.计算机学报,2005,28(4):731−738.

[28] 熊伟,李兵,何鹏,黄媛,董丹丹.一种创新的 SaaS 服务的构建模型.微电子学与计算机,2012,29(9):141−144.

[29] 宋杰,侯泓颖,朱志良.OSF:一种支持 SaaS 应用的构件框架.计算机科学,2012,39(10):125−130.

孙昌爱(1974－),男,江苏盐城人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件测试,程序分析,服务计算,软件体系结构.

张在兴(1993－),男,硕士,主要研究领域为服务计算.

张鑫(1990－),男,硕士,主要研究领域为服务计算.