```
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*

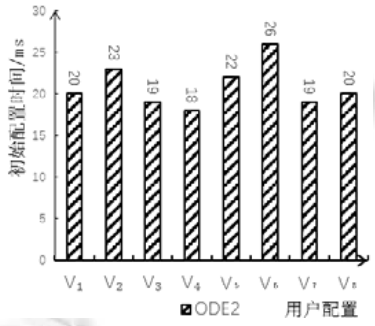

Fig.16 Initial configuration time

图 16 初始配置时间

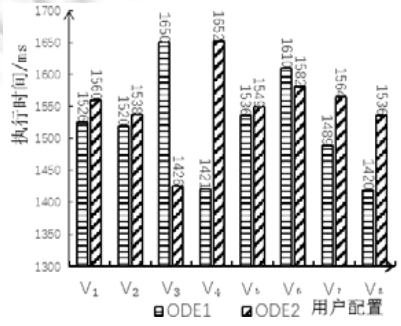

Fig.17 Execution time

图 17 执行时间

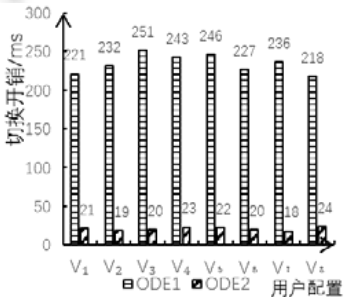

Fig.18 Switching overhead

图 18 切换开销

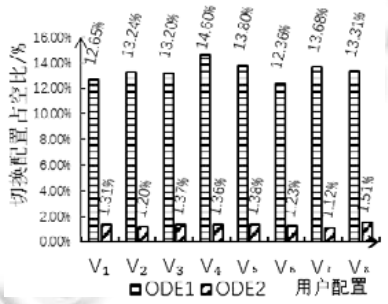

Fig.19 Duty ratio of configuration switching

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.