基于 TLA 的 SaaS 业务流程定制及验证机制研究

史玉良[1)] 栾 帅[1)] 李庆忠[1)] 董晋利[1)] 刘方方[2)]
[1)](山东大学计算机科学与技术学院 济南 250101)
[2)](上海大学计算机工程与科学学院 上海 200072)

摘 要 SaaS 模式已成为当前流行的软件服务形式. 为满足不同租户个性化的业务服务需求, SaaS 模式必须提供灵活的定制机制. 为此, 提出了一个支持租户业务流程定制行为建模及验证的框架. 该框架以层次定制行为约束图作为定制指导, 通过 TLA(Temporal Logic of Actions)对各层的原子定制活动建模, 并以此为基础构建租户的全局定制行为, 然后基于应用的业务规则约束设计算法验证全局定制行为的正确性; 为有效提高租户的定制效率, 基于对多租户已有正确定制结果的统计分析, 设计定制推荐算法, 合理减少验证次数. 仿真实验结果证实了该机制的高效性及可靠性.

关键词 软件即服务; 业务流程; 定制; 验证; 活动时序逻辑
中图法分类号 TP311 **DOI** 号: 10.3724/SP.J.1016.2010.02055

TLA Based Customization and Verification Mechanism of Business Process for SaaS

SHI Yu-Liang[1)] LUAN Shuai[1)] LI Qing-Zhong[1)] DONG Jin-Li[1)] LIU Fang-Fang[2)]
[1)](*School of Computer Science and Technology, Shandong University, Jinan* 250101)
[2)](*School of Computer Engineering and Science, Shanghai University, Shanghai* 200072)

Abstract SaaS model has become a popular form of software services. In order to meet personalized business service needs of tenants, SaaS model must provide flexible business process customization mechanism. This paper proposes a modeling and verification framework for business process customization. Following the guidance of Hierarchical Customized Behavior and Dependency Graph, the framework models the atomic customization action based on TLA (Temporal Logic of Actions), constructs the global customization behavior and designs the verification algorithm for tenants' global customization based on business rule constraints. Based on multi-tenants' feature and historical verified customization, this framework also constructs a customization recommendation algorithms to improve the efficiency of customization for tenants. Experiments demonstrate the efficiency and reliability of this customization and verification mechanism.

Keywords software as a service; business process; customization; verification; temporal logic of actions

1 引 言

软件即服务(Software as a Service, SaaS)是随着互联网技术的发展和应用软件的成熟而逐渐兴起的一种新型软件服务形式. SaaS 模式的一个典型特征是"单实例多租赁", 即多个租户共享服务提供商的一个应用实例, 不同租户的数据、服务在物理上共

收稿日期: 2010-06-08; 最终修改稿收到日期: 2010-09-01. 本课题得到国家自然科学基金(90818001, 60803143)、国家科技支撑计划(2009BAH44B02)、山东省自然科学基金(ZR2010FQ026, 2009ZRB019YT, Y2007G38)及山东省科技攻关计划(2010GGX10105)资助.
史玉良, 男, 1978 年生, 博士, 讲师, 主要研究方向为服务计算、可信计算、数据库. E-mail: liangyus@sdu.edu.cn. 栾 帅, 男, 1984 年生, 硕士研究生, 主要研究方向为服务计算、数据库. 李庆忠, 男, 1965 年生, 博士, 教授, 博士生导师, 主要研究领域为数据库、可信计算、面向语义的应用集成和信息集成. 董晋利, 男, 1985 年生, 硕士研究生, 主要研究方向为服务计算、数据库. 刘方方, 女, 1979 年生, 博士, 副教授, 主要研究方向为 Web 服务合成、服务发现.

享,而在逻辑上完全隔离,对于每个租户来说,这个实例只像是为自己服务. SaaS 这种统一的软件服务形式要满足业务需求各异的不同租户,必须提供灵活的按需定制机制,以便让租户在共享实例的基础上构建专属于自己的业务流程.

SaaS 模式下的业务流程定制不同于传统模式下面向服务组合的流程定制,其定制一般涉及到流程、服务、数据等不同层次. 这些层次的定制活动之间存在着诸多的依赖约束关系,如定制流程中的某个服务通常需要定制这个服务所操作的数据对象,而删除流程中的某个服务需要先考虑是否有其它服务依赖于这个服务,租户的定制行为必须同这些约束一致,此外,租户的定制行为还需要遵守 SaaS 服务提供商定义的一些外部业务规则,否则将会导致错误的定制结果;同时,SaaS 模式下,如何利用规模效应,基于其他租户已有的正确的定制结果,降低租户定制的复杂度,提高租户定制的效率,也是一个值得研究的问题. 这些都对 SaaS 模式下业务流程的个性化定制提出了新的挑战.

目前,BPEL4WS[1]、WSCI[2] 等描述业务流程的标准规范已被广泛采用,但这些规范多是用来描述流程的静态业务逻辑,执行时的灵活性较差;在流程的个性化定制研究中,已有工作多是围绕服务的动态选择及最优组合展开[3-6],而 SaaS 模式下流程定制的重点是如何在流程、服务、数据多层次的定制活动中确保各个租户定制结果的正确性;在流程定制的正确性研究中,已经提出了自动机[7]、进程代数[8]、Petri 网[9-10] 等多种形式化的验证方法,但这些方法关注的多是不同服务接口的匹配、流程自身逻辑结构的正确性等问题. 而 SaaS 模式下的业务流程定制,流程、服务、数据各个层次内部和层次之间存在着一定的约束关系,这些约束关系具有一定的时序性、逻辑性等特点,一个定制正确的流程必须要满足这些关系,但以上这些模型对这些关系的描述难以提供良好的支持.

针对以上问题,本文设计开发了一个面向 SaaS 应用的业务流程定制及验证框架. 在该框架中,首先制定了层次定制行为约束图(Hierarchical Customized Behavior Constraint Graph, HCBCG),该图描述了在流程、服务、数据等层次可能进行的所有原子定制活动及其之间的依赖关系;进而在租户的定制过程中,以 HCBCG 作为指导,基于 TLA(Temporal Logic of Actions)对 HCBCG 中的各个定制活动建模,并通过合理的逻辑顺序无损构建租户的复杂定制活动,形成全局定制行为模型;最后设计相应的验证算法,确保租户的全局定制行为既满足应用内部的依赖关系,又不违背服务提供商的外部业务规则约束,从而避免其对整个应用产生难以预测的负面影响;为充分利用 SaaS 应用的"单实例多租赁"特点,提高租户的定制效率,框架还提供了推荐机制,基于对多租户已有正确定制结果的统计分析,为当前租户推荐合适的服务,合理减少验证次数.

本文第 2 节是相关工作;第 3 节是定制验证框架的整体结构描述;第 4 节是基于 TLA 的定制行为建模分析及相应的算法描述;第 5 节介绍一个考试业务流程定制的应用示例,并通过原型系统的仿真实验结果证实机制的高效性;最后是本文的总结及下一步工作的展望.

2 相关工作

目前,传统模式下流程定制的相关研究多是从服务动态选择及最优组合、流程的形式化验证、流程的动态改变等几方面展开,而 SaaS 模式下面向多租户应用的业务流程定制研究刚刚起步,研究相对较少.

在服务的动态选择及最优组合研究中,其主要思想是通过各种算法或模型使服务组合整体的 QoS 达到最优. 文献[3] 采用模式选择算法确定每个流程结点的最优服务;文献[4] 设计的 QoS 代理体系及效用函数实现了服务的最优选择;文献[5] 基于 MIP 方法实现了服务组合的质量最优;文献[6] 利用多目标遗传算法的智能优化原理产生了满足约束条件的优化服务组合流程集.

在流程的形式化验证研究中,文献[7] 提出了一个基于模型的方法验证流程中服务的相互协作关系,在验证过程中使用有限状态机分析服务组合是否正确;文献[8] 使用进程代数作为形式化的工具描述 Web 服务,对合成中的行为特征进行推导,分析异步交互对 Web 服务合成设计与实现产生的影响,在此之上,基于服务合成正确性的定义,提出了一种一致性关系;文献[9-10] 以 Petri 网为基础模型,模拟服务的组合过程,并对组合后的流程进行分析验证,确保服务组合的正确性.

为了支持流程的动态改变,文献[11] 提出了一个支持业务流程动态改变的管理机制,形式化地定义了服务选择约束与服务调度约束,在构建业务流程时,能够动态地进行约束验证,但该机制只涉及局

部的服务选择和调度，无法从整体上对组建的流程进行验证，本文将借鉴文献[11] 所列出的服务之间的关系作为租户定制服务的约束；文献[12] 提出了一个正规模型，模型用事实规则来描述业务层和 IT 层中各元素之间的关系，并提供具有普遍适用性的影响传播规则，当系统产生新的定制请求时，通过事实规则和影响传播规则可以方便地判断影响到的元素，这种方法不允许规则的实时添加，缺乏一定的灵活性，但可以看出规则本身对流程的柔性定制具有很好的支持作用，本文也将采用业务规则作为 SaaS 业务流程定制验证的基础.

SaaS 模式下，文献[13] 提出一种策略驱动的定制机制，若租户的定制请求同定制策略一致，服务提供方将对服务进行相应升级，这种方法可以确保定制的正确性，但同时增加了服务提供商的负担，并且定制响应周期过长. 文献[14] 分析了 SaaS 模式下的配置和定制，给出了 SaaS 配置能力模型，同时示例描述了不同层次之间定制的依赖关系，但是没有给出其形式化定义，也没有展开对其定制验证工作研究. 文献[15] 中提出了可变点的概念，用来在 SaaS 模式下进行 BEPL 流程的定制化处理，定制考虑了跨层之间的依赖关系，给出了统一的描述规范. 文献[16] 进一步给出了可变点模型，从基于租户的可变点和基于实现的可变点两方面支持 SaaS 服务商管理 SaaS 应用中的定制需求和可变点. 但是，已有 SaaS 模式下的业务流程定制缺乏对 SaaS 业务流程中各个层次间及跨层的定制的形式化定义，没有针对各种依赖约束对定制结果进行验证检查. 而且，也没有考虑 SaaS 模式多租户特征对业务流程定制的作用和影响.

3　面向 SaaS 的业务流程定制及验证框架

SaaS 业务流程及验证框架的整体结构如图 1 所示，包括层次定制行为约束图 HCBCG、规则管理、租户定制、定制建模、定制验证和推荐 6 大模块. 这些模块之间既能够独立完成相应任务，又能相互通信，协同工作.

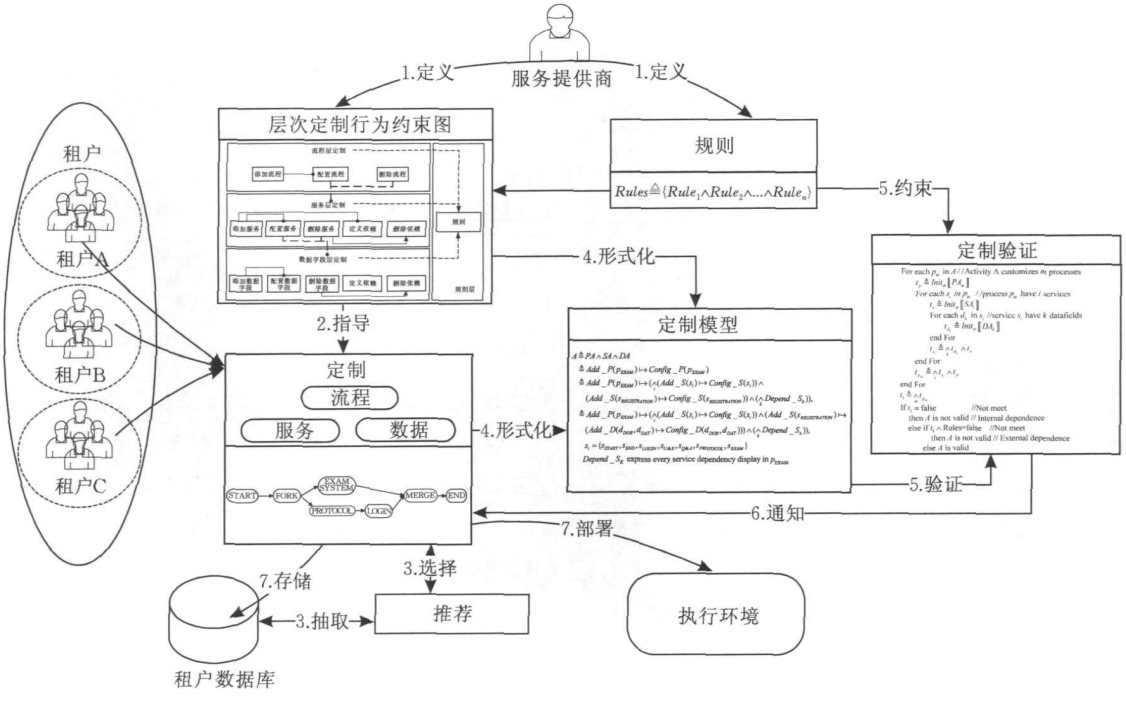

图 1　框架的整体结构图示

框架的整体运行过程如下：

(1) 在应用开始之初，服务提供商通过定义 HCBCG，详细阐述了租户在流程、服务和数据层能够进行的所有原子定制活动以及活动间的约束关系，该图作为定制活动的基石，能够在租户定制过程中起到指导和约束作用.

(2) 在此基础上，租户可以使用定制模块对服务商提供的 SaaS 应用进行多方面的个性化定制，每一次的定制活动都可以看作是由一个或多个原子定制活动组合而成.

(3)推荐模块将利用 SaaS 应用的"单实例多租赁"特点,对各个租户定制的服务进行统计分析,进而为当前租户下一步的定制提供推荐服务,如果租户选择模块所推荐的服务,则无需进行检验可直接跳到下一步定制.

(4)对于不采用推荐模块的每一步定制,定制建模组件使用 TLA 公式进行合理化编排,将其形式化并传递给定制验证模块.

(5)定制验证模块将通过验证算法,以服务提供商定义的约束性规则和形式化的定制活动为输入,确定租户的定制活动是否合理.

(6)对于租户的定制行为,将返回最终的验证结果.

(7)只有验证通过的定制活动才能被存储和部署执行,随着租户数量和定制次数的增多,推荐模块对于提高租户定制效率将会发挥更大的作用.

3.1 层次定制行为约束图

框架中的层次定制行为约束图 HCBCG 如图 2 所示,该图详细列出了 SaaS 应用中流程、服务和数据各层的定制活动以及相关约束.

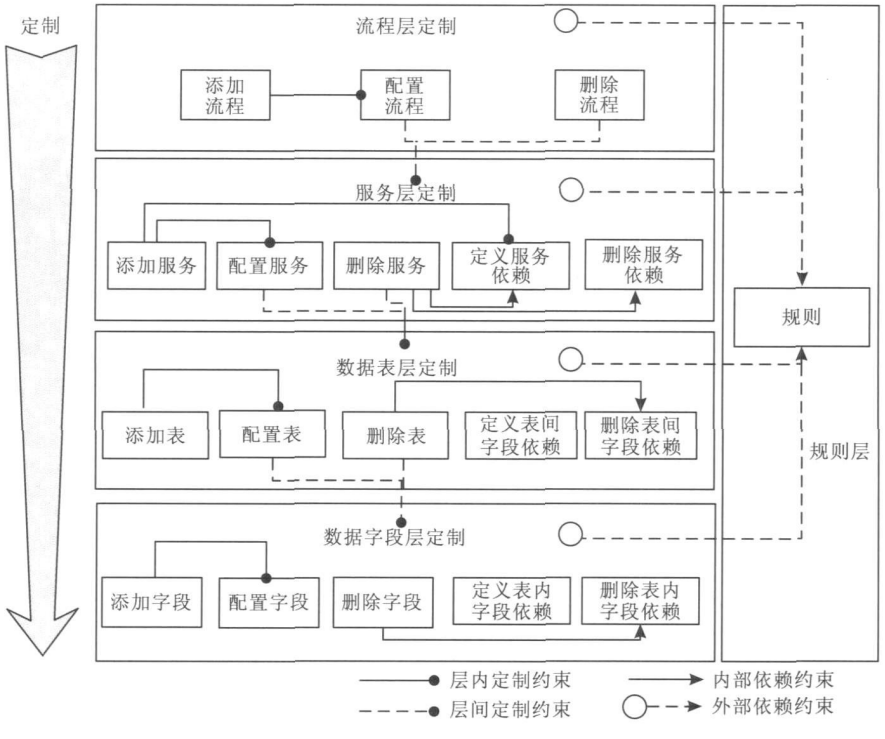

图 2 定制行为约束图 HCBCG

在 HCBCG 中,每一层的定制活动不仅可能会影响层次内部的其它活动,而且也会对其它层产生影响;除此之外,服务提供商通常会在一个特殊的层次(图 2 所示规则层)上定义一些额外的业务规则,以此限定租户在各个层次上的定制活动.

HCBCG 中共有层内定制约束、层间定制约束、内部依赖约束、外部依赖约束 4 种约束关系.其中定制约束是指租户的定制行为必须按照这些约束进行,否则无法进行定制;依赖约束是指租户的定制行为必须满足这些约束,否则定制结果将是错误的.例如"删除服务"活动和"删除服务依赖"、"定义服务依赖"活动之间存在着内部依赖约束关系,意味着租户删除的服务如果同其它服务存在着依赖关系,则必须删除对应这个服务的依赖关系,否则流程中将会出现冗余的服务依赖关系;此外,在流程删除这个服务及其依赖关系后,如果存在着没有依赖关系的服务,也必须定义新的服务依赖关系.

在 HCBCG 中,每一个数据字段可以看作一个特殊的数据表(只含一个字段),每一个数据表可以看作是一个特殊的服务(只存储字段信息),而每一个服务又可以看作是一个特殊的流程(只有一个服务),故不论是数据、服务或是流程的定制,都可以看作是以流程定制为开始,因此 HCBCG 定制行为是自上而下的.

在 HCBCG 中,添加流程之后就要对流程进行配置,而配置流程意味着要对流程中包含的所有服务以及服务之间的关系进行定制,服务的定制又涉及到服务所包含的数据表以及表间的关系,最终涉

及表中的数据信息以及字段间的依赖，这种有序定制行为也在一定程度上保证了定制的合理性. 因为每一层的定制都是要以上一层的存在作为前提的，譬如不能没有进行服务定制，就直接修改或添加服务中涉及的数据. 针对这种定制行为，可以通过时序逻辑对租户的个性化定制活动进行合理的建模分析，确保每一步的定制活动不会对系统产生难以预料的影响.

3.2　规则层

规则中的约束元素主要是服务和数据，基于实际的业务需求，规则可以灵活地添加、删除、修改. 本文所列规则主要分为存在性规则和调度性规则两类. 存在性规则主要涉及到两个或多个服务(数据)间的存在关系，如图 3 所示.

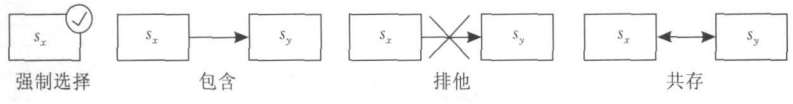

图 3　存在性规则图示

存在性规则包括：

强制选择规则($ManS(s_x)$). 服务(数据)必须存在于某个流程(服务)中.

包含规则($Inc(s_x, s_y)$). 定义了两个服务(数据)间的包含性依赖关系. 如 $Inc(X, Y)$，意味着有 X 必须有 Y，而有 Y 不一定有 X.

排他规则($Exc(s_x, s_y)$). 定义了两个服务(数据)间的排他性依赖关系. 如 $Exc(X, Y)$，意味着有 X 就不能有 Y.

共存规则($Cor(s_x, s_y)$). 定义了两个服务(数据)间的共存性依赖关系. 如 $Cor(X, Y)$，意味着在流程中要么 X、Y 都存在，要么都不存在.

调度性规则主要涉及两个或多个服务(数据)间的处理顺序关系，如图 4 所示.

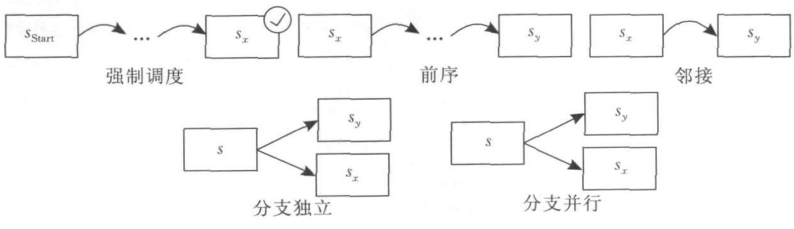

图 4　调度性规则图示

调度性规则包括：

强制调度规则($ManD(s_x)$). 定制的服务必须能够在流程中被执行.

前序规则($Bef(s_x, s_y)$). 两个定制服务(数据)之间执行时，其中一个必须要在另一个之前，但可以不邻接.

邻接规则($Mee(s_x, s_y)$). 两个定制服务(数据)之间执行时，其中一个必须要在另一个之前，并且服务邻接.

分支独立规则($Ind(s, s_x, s_y)$). 某个服务分支执行时，只能执行其中的一个分支.

分支并行规则($Par(s, s_x, s_y)$). 某个服务分支执行时，可以执行一个分支或两个都执行.

4　基于 TLA 的定制行为建模分析

SaaS 的个性化定制涉及流程、服务和数据多个层次，活动离散，有一定的时序性. 针对这些特性，为了确保定制活动的正确性，按照框架的运行机制，本文采用 TLA 这种时序逻辑来建模、分析和验证租户定制行为.

TLA 是描述离散时序动态系统的一种重要的时序逻辑，它结合了标准时序逻辑和活动逻辑的特点，表达能力强，且采用模块化、层次化的描述方法使得条理清晰、脉络分明，从已有的文献[17]中可以看出 TLA 对活动的时序建模具有很好的支持作用.

TLA 的语义解释基于行为(behavior)、状态(state)和动作(action)3 个概念. 行为是一个无限的状态序列，状态是将值赋予变量的映射，活动是一个包含变量、prime 变量和常量符号的布尔表达式. TLA 的详细的知识可见参考文献[18].

本文将使用行为表示租户的一次完整的个性化定制过程，行为中的状态表示了租户个性化定制时系统中流程、服务、数据的表征变化，最后使用活动表示租户为达到最终目标而进行的每一次定制操作.

为了描述租户的个性化定制，首先定义了定制活动的初始状态.

定义 **1**. 租户定制活动的开始状态为 $Init_\sigma$，描述如下：

$$Init_\sigma \triangleq P \wedge S \wedge D \wedge (P_{CIN}=S_{CIN}=D_{CIN}=\varnothing) \wedge (P_{CED}=S_{CED}=D_{CED}=S_{DEP}=E_{DEP}=D_{DEP}=\varnothing),$$

其中，P、S、D 分别表示应用初始的流程、服务和数据集合；P_{CIN}、S_{CIN}、D_{CIN} 分别表示租户正在定制的流程、服务和数据集合；P_{CED}、S_{CED}、D_{CED} 分别表示租户已经定制的流程、服务和数据集合；S_{DEP}、E_{DEP}、D_{DEP} 分别表示应用中租户定义的服务依赖关系、表间字段依赖关系、表内字段依赖关系集合.

下面将针对每一层的定制活动，给出相应的 TLA 描述，这些描述将作为验证的输入.

4.1 流程层建模

(1) 添加流程. 租户可以根据业务需求，以应用提供的各种元素为基础，自己定制一个新的流程. 需要注意的是，租户添加流程的活动势必要导致对新流程的配置，从另一个角度上说，租户添加流程的活动是建立在新流程配置的基础上的.

添加流程活动用 Add_P 表示，其中 p_i 表示要添加的流程.

$$Add\_P(p_i) \triangleq (p_i \notin P) \wedge (P'_{CIN}=P_{CIN} \cup \{p_i\}) \wedge (P'_{CED}=P_{CED}).$$

(2) 配置流程. 租户配置流程就是配置流程中的各个服务，包括服务的添加、配置、删除以及服务之间依赖关系的创建和删除，流程的配置结束当且仅当流程中的各个服务配置完毕.

租户配置的流程可以是新添加的流程，也可能是系统原有的流程. 配置流程活动用 $Config\_P$ 表示，其中 SA 表示服务层的定制活动，$s_1, s_2, \cdots, s_m$ 表示流程 p_i 中的服务.

$$Config\_P(p_i) \triangleq (P'_{CIN}=P_{CIN} \cup \{p_i\}) \wedge \Box(\bigwedge_{k=1}^{m} SA(s_k) \text{ a } CP(p_i)),$$

$$CP(p_i) \triangleq (P'_{CIN}=P_{CIN}-\{p_i\}) \wedge (P'_{CED}=P_{CED} \cup \{p_i\}).$$

(3) 删除流程. 删除流程表示删除该流程中的所有服务以及服务间的依赖关系.

删除流程活动用 $Delete\_P$ 表示，其中，$s_1, s_2, \cdots, s_m$ 表示流程 p_i 中的所有服务，$Delete\_SDep$ 表示删除流程中服务间的依赖关系，$Depend\_S_1, \cdots, Depend\_S_n$ 表示流程中所有的服务依赖关系，$Delete\_S$ 表示删除服务.

$$Delete\_P(p_i) \triangleq (p_i \in P_{CED}) \wedge (P'_{CIN}=P_{CIN}) \wedge (P'_{CED}=P_{CED}-\{p_i\}) \bigwedge_{k=1}^{n} Delete\_SDep(Depend\_S_k) \bigwedge_{k=1}^{m} Delete\_S(s_k).$$

定义 **2**. 整个流程层的定制活动为 PA，描述如下：

$$PA \triangleq (Add\_P \text{ a } Config\_P) \vee Config\_P \vee Delete\_P.$$

4.2 服务层建模

(1) 添加服务. 租户可以根据业务需求，从服务商提供的所有服务中选择一个添加到自己定制的流程中，与添加流程相似，租户对服务的添加也势必将导致对服务的配置. 此外，为确保流程描述的合理性，添加服务同时也要定义服务的依赖关系.

添加服务活动用 Add_S 表示，其中，s_i 表示要添加的服务.

$$Add\_S(s_i) \triangleq (s_i \in S) \wedge (S'_{CED}=S_{CED}) \wedge (S'_{CIN}=S_{CIN} \cup \{s_i\}).$$

(2) 配置服务. 租户选择配置服务就是配置服务中的数据，包括数据表的修改、数据字段的添加、配置、删除以及其依赖关系的创建和删除，服务的配置结束当且仅当服务中的各个字段配置完毕.

配置服务活动用 $Config\_S$ 表示，其中 EA 表示数据表层的定制活动，$e_1, e_2, \cdots, e_m$ 表示服务 s_i 中的数据表.

$$Config\_S(s_i) \triangleq (S'_{CIN}=S_{CIN} \cup \{s_i\}) \Box (\bigwedge_{k=1}^{m} EA(e_k) \text{ a } CS(s_i)),$$

$$CS(s_i) \triangleq (S'_{CIN}=S_{CIN}-\{s_i\}) \wedge (S'_{CED}=S_{CED} \cup \{s_i\}).$$

(3) 定义服务依赖. 服务和服务之间会有依赖关系，这种关系确定了服务执行的先后顺序，同时也具有传递性.

租户可以定义服务间的依赖关系，活动用 $Depend\_S$ 表示，其中，s_i，s_j 表示具有依赖关系的两个服务，s_i 要在 s_j 前被执行.

$$Depend\_S(s_i, s_j) \triangleq (s_i, s_j \in S_{CED}) \wedge (S'_{CIN}=S_{CIN}) \wedge (S'_{CED}=S_{CED}) \wedge (S'_{DEP}=S_{DEP} \cup \{s_i \text{ a } s_j\}).$$

基于两两服务之间的 $Depend\_S$，用 $Depend\_SAL$ 表示了所有的服务依赖关系：

$$Depend\_SAL \triangleq Depend\_S_1 \wedge Depend\_S_2 \wedge \cdots \wedge Depend\_S_n.$$

(4) 删除服务依赖. 租户也可以删除已经定制的服务之间的依赖关系，活动用 $Delete\_SDep$ 表示：

$$Delete\_SDep(Depend\_S(s_i, s_j)) \triangleq (S'_{CIN}=S_{CIN}) \wedge (S'_{CED}=S_{CED}) \wedge (S'_{DEP}=S_{DEP}-\{s_i \text{ a } s_j\}).$$

(5) 删除服务. 删除服务表示删除了该服务中的所有数据表，删除服务活动用 $Delete\_S$ 表示，其

中，e_1，e_2，…，e_m 表示服务 s_i 中的数据表，$Delete\_E$ 表示删除服务所包含的数据表.

$$Delete\_S(s_i) \triangleq (s_i \in S_{CED}) \bigwedge_{k=1}^{m} Delete\_E(e_k) \wedge (S'_{CIN} = S_{CIN}) \wedge (S'_{CED} = S_{CED} - \{s_i\}).$$

定义 **3**.　整个服务层的定制活动为 SA，描述如下：

$$SA \triangleq (Add\_S \text{ a } (Config\_S \wedge Depend\_S)) \vee Config\_S \vee Delete\_S \vee Depend\_SAL \vee Delete\_SDep.$$

4 3　数据层建模

4 3 1　数据表层建模

(1) 添加表. 租户可以根据数据管理需求，添加一个新的业务数据表，在添加数据表之后，也将对数据表进行配置.

添加表活动用 Add_E 表示，其中，e_i 表示要添加的表.

$$Add\_E(e_i) \triangleq (e_i \notin D) \wedge (D'_{CIN} = D_{CIN} \cup \{e_i\}) \wedge (D'_{CED} = D_{CED}).$$

(2)配置表. 租户选择配置表就是配置表中的数据，包括数据字段的添加、删除以及其表内字段依赖关系的创建和删除，配置结束当且仅当表中的各个字段配置完毕.

配置表活动用 $Config\_E$ 表示，其中 DA 表示数据字段层的定制活动，d_1，d_2，…，d_m 表示数据表 e_i 中的所有数据字段.

$$Config\_E(e_i) \triangleq (D'_{CIN} = D_{CIN} \cup \{e_i\}) \square (\bigwedge_{k=1}^{m} DA(d_k) \text{ a } CP(e_i)),$$

$$CP(e_i) \triangleq (D'_{CIN} = D_{CIN} - \{e_i\}) \wedge (D'_{CED} = D_{CED} \cup \{e_i\}).$$

(3) 定义表间字段依赖. 租户可以定义不同数据表中数据字段间的依赖关系，活动用 $Depend\_E$ 表示：

$$Depend\_E(e_i, d_i, e_k, d_k) \triangleq (d_i \in e_i) \wedge (d_k \in e_k) \wedge (D'_{CIN} = D_{CIN}) \wedge (D'_{CED} = D_{CED}) \wedge (E'_{DEP} = E_{DEP} \cup \{d_i \text{ a } d_k\}).$$

基于 $Depend\_E$ 的描述，数据表 e_i 中所有的表间字段依赖关系用 $Depend\_EAL(e_i)$ 表示：

$$Depend\_EAL(e_i) \triangleq \bigwedge_{k=1}^{m} Depend\_E_k(e_i).$$

基于 $Depend\_EAL(e_i)$ 的描述，所有数据表中数据字段依赖关系的集合用 $Depend\_EAL$ 表示：

$$Depend\_EAL \triangleq \bigwedge_{i=1}^{n} Depend\_EAL(e_i).$$

(4) 删除表间字段依赖. 租户删除表间字段依赖活动用 $Delete\_EDep$ 表示：

$$Delete\_EDep(Depend\_E(e_i, d_i, e_k, d_k)) \triangleq (d_i \in e_i) \wedge (d_k \in e_k) \wedge (D'_{CIN} = D_{CIN}) \wedge (D'_{CED} = D_{CED}) \wedge (E'_{DEP} = E_{DEP} - \{d_i \text{ a } d_k\}).$$

(5) 删除表. 删除表活动将删除该表中的所有数据字段，删除表活动用 $Delete\_E$ 表示：

$$Delete\_E(e_i) \triangleq (\bigwedge_{k=1}^{m} d_k \in e_i) \text{ a } (\bigwedge_{k=1}^{m} Delete\_d(d_k)).$$

定义 **4**.　整个数据表层的活动为 EA，其描述如下：

$$EA \triangleq (Add\_E \text{ a } Config\_E) \vee Config\_E \vee Delete\_E \vee Depend\_EAL \vee Delete\_EDep.$$

4 3 2　数据字段层建模

(1)添加字段. 租户可以在数据表中添加指定类型的数据字段，活动用 Add_D 表示，其中，d_i 为要添加的字段，d_{Ai} 表示了添加字段的类型，D_A 表示服务中所有的字段类型.

$$Add\_D(d_i, d_{Ai}) \triangleq (d_{Ai} \in D_A) \wedge (D'_{CIN} = D_{CIN} \cup \{d_i\}) \wedge (D'_{CED} = D_{CED}).$$

(2)配置字段. 租户可以配置数据表中已存在或是自定义的数据字段，活动用 $Config\_D$ 表示：

$$Config\_D(d_i, d_{Ai}) \triangleq (d_{Ai} \in D_A) \wedge (D'_{CIN} = D_{CIN} - \{d_i\}) \wedge (D'_{CED} = D_{CED} \cup \{d_i\}).$$

(3)定义表内字段依赖. 数据表内的字段和字段之间会有依赖关系，这种关系确定了字段被处理的先后顺序，同时也具有传递性. 租户定义表内字段依赖关系的活动用 $Depend\_D$ 表示：

$$Depend\_D(d_i, d_j) \triangleq (d_i, d_j \in D_{CED}) \wedge (D'_{CIN} = D_{CIN}) \wedge (D'_{CED} = D_{CED}) \wedge (D'_{DEP} = D_{DEP} \cup \{d_i \text{ a } d_j\}).$$

基于 $Depend\_D$ 的描述，用 $Depend\_DAL$ 表示所有的字段依赖关系：

$$Depend\_DAL \triangleq Depend\_D_1 \wedge Depend\_D_2 \wedge \cdots \wedge Depend\_D_n.$$

(4)删除表内字段依赖. 租户删除表内字段依赖关系的活动用 $Delete\_DDep$ 表示：

$$Delete\_DDep(Depend\_D(d_i, d_j)) \triangleq (D_{CIN} = D_{CIN}) \wedge (D'_{CED} = D_{CED}) \wedge (D'_{DEP} = D_{DEP} - \{d_i \text{ a } d_j\}).$$

(5)删除字段. 租户删除服务中自定义字段的活动用 $Delete\_D$ 表示：

$$Delete\_D(d_i) \triangleq (d_i \in D_{CED}) \wedge (D'_{CIN} = D_{CIN}) \wedge (D'_{CED} = D_{CED} - \{d_i\}).$$

定义 **5**.　整个数据字段层的定制活动为 DA，其描述如下：

$$DA \triangleq (Add\_D \text{ a } Config\_D) \vee Config\_D \vee$$

$Delete\_D \vee Depend\_DAL \vee Delete\_DDep$.

4.4 规则层建模

规则层也可以通过 TLA 形式化地描述. 表 1、表 2 是服务存在性规则和调度性规则的 TLA 描述, 这些规则对于数据来说同样适用.

表 1 规则层语义-存在性规则

存在性规则	
$ManS(s_X)$	$(s_X \in S_{CED}) \equiv \text{True}$
$Inc(s_X, s_Y)$	$(s_X \in S_{CED}) \text{ a } (s_Y \in S_{CED})$
$Exc(s_X, s_Y)$	$(s_X \in S_{CED} \wedge s_Y \notin S_{CED}) \vee (s_X \notin S_{CED} \wedge s_Y \in S_{CED})$
$Cor(s_X, s_Y)$	$(s_X \in S_{CED} \wedge s_Y \in S_{CED}) \vee (s_X \notin S_{CED} \wedge s_Y \notin S_{CED})$

表 2 规则层语义-调度性规则

调度性规则	
$ManD(s_X)$	$((s_X \in S_{CED}) \equiv \text{True}) \wedge (\Box(s_{start} \text{ a } s_X))$
$Bef(s_X, s_Y)$	$\Diamond(s_X \text{ a } s_Y)$
$Mee(s_X, s_Y)$	$s_X \text{ a } s_Y$
$Ind(s, s_X, s_Y)$	$((s \text{ a } s_X) \wedge (s \text{ a } s_Y)) \vee ((s \text{ a } s_Y) \wedge (s \text{ a } s_X))$
$Par(s, s_X, s_Y)$	$(s \text{ a } s_X) \vee (s \text{ a } s_Y)$

定义 **6**. 租户的个性化定制行为模型为 σ, 其描述为

$$\sigma = Init_\sigma \wedge \Box[A]_f \wedge SF_f(A),$$

其中, $Init_\sigma$ 表示定制活动的开始状态, $A \triangleq PA \wedge SA \wedge EA \wedge DA$ 表示租户在各个层上执行的定制活动.

$f \triangleq f\langle P_{CIN}, S_{CIN}, D_{CIN}, P_{CED}, S_{CED}, D_{CED}, S_{DEP}, E_{DEP}, D_{DEP}\rangle$ 表示状态函数, 用以改变系统中各个状态变量的值, $SF_f(A)$ 表示最后一个定制活动的结束标志着本租户定制行为的结束. 对于服务商规定的外部约束性规则 $Rule$, 也可以使用基于系统状态变量的函数 f_{Rule} 来表示, 如果规定了多条规则, 则租户的定制活动必须满足所有的约定.

4.5 定制验证算法

使用 TLA 对租户的定制活动建模后, 需要设计相应的验证算法, 以确保租户定制行为的正确性. 在介绍算法前, 先定义如下定理作为租户个性化定制验证的依据.

定理 **1**. 对于当前系统状态集合 q (q 为真)和定制活动 A, 如果存在状态集合 t ($t \triangleq q[\![A]\!] \wedge Rules$)为真, 则定制活动 A 合理. 其中 A 为租户定制活动, $Rules$ 为服务商的约束性规则集合, 其描述为 $Rules \triangleq \{Rule_1 \wedge Rule_2 \wedge \cdots \wedge Rule_n\}$, 如果服务商没有规定约束规则, 则 $Rules \equiv \text{true}$.

证明. 为证明定理 1 的正确性, 我们首先对开发商指定的约束性规则 $Rules$ 按照服务层、数据表层、数据字段层等不同层次的约束划分为 RS、RE、RD 3 类, 即 $Rules \triangleq RS \wedge RE \wedge RD$, 同时, 由于 $A \triangleq PA \wedge SA \wedge EA \wedge DA$, 那么我们可以得到如下公式:

$$\begin{aligned} q[\![A]\!] \wedge Rules &\triangleq (q[\![PA]\!] \wedge q[\![SA]\!] \wedge q[\![EA]\!] \wedge q[\![DA]\!]) \wedge (RS \wedge RE \wedge RD) \\ &\triangleq q[\![PA]\!] \wedge (q[\![SA]\!] \wedge RS) \wedge (q[\![EA]\!] \wedge RE) \wedge (q[\![DA]\!] \wedge RD). \end{aligned}$$

如果状态集合 t 为真, 那么 $q[\![A]\!] \wedge Rules$ 将为真, 这意味着 $q[\![PA]\!]$、$q[\![SA]\!] \wedge RS$、$q[\![EA]\!] \wedge RE$ 以及 $q[\![DA]\!] \wedge RD$ 这些状态也为真, 这说明租户按照定制模型进行的定制活动都满足了外部依赖约束关系.

进一步分析, 我们还可以推导得出 $q[\![PA]\!]$、$q[\![SA]\!]$、$q[\![EA]\!]$、$q[\![DA]\!]$ 这些状态也为真, 这意味着租户的活动 PA、SA、EA、DA 都不会与之前的系统状态产生矛盾, 均遵守定制活动的内部依赖约束关系.

综上所述, 如果状态 $t \triangleq q[\![A]\!] \wedge Rules$ 为真, 那么活动 PA、SA、EA、DA 均合理, 它们均满足内部依赖约束关系及外部依赖约束关系. 由 $A \triangleq PA \wedge SA \wedge EA \wedge DA$ 可得活动 A 合理. 证毕.

根据定理 1, 设计如下的验证算法.

For each p_m in A //活动 A 定制了 m 个流程
  $t_p \triangleq Init_\sigma[\![PA_m]\!]$;
  For each s_i in p_m //流程 p_m 有 i 个服务
    $t_s \triangleq Init_\sigma[\![SA_i]\!]$;
    For each e_j in s_i //服务 s_i 有 j 个数据表
      $t_{e_j} \triangleq Init_\sigma[\![EA_j]\!]$;
      For each d_k in e_j //数据表 e_j 有 k 个数据字段
        $t_{d_k} \triangleq Init_\sigma[\![DA_k]\!] \triangleq DA_k \wedge D_{DEP}$;
      end For
      $t_{e_j} \triangleq \left(\bigwedge_k t_{d_k} \wedge t_e\right) \wedge (EA_j \wedge E_{DEP})$;
    end For
    $t_{s_i} \triangleq \left(\bigwedge_j t_{e_j} \wedge t_s\right) \wedge (SA_i \wedge S_{DEP})$;
  end For
  $t_{p_m} \triangleq \bigwedge_i t_{s_i} \wedge t_p$;
end For
$t_1 \triangleq \bigwedge_m t_{p_m}$;
If $t_1 =$ false //没有满足内部依赖约束
  then return A is not valid;
$Result = \{P_{CED}, S_{CED}, D_{CED}, S_{DEP}, E_{DEP}, D_{DEP}\}$;
//满足内部依赖约束后的定制结果需要进行规则验证
$Rules \triangleq ExistingRule \wedge SchedulingRule$;
//规则分为存在性规则和调度性规则两类
For each $Rule$ in $Rules$
  For each $Rule \in ExistingRule$
    if $\{P_{CED}, S_{CED}, D_{CED}\} \wedge Rule =$ false
      then return A is not meet $ExistingRule$;
  end For
  For each $Rule \in SchedulingRule$
    if $\{S_{DEP}, E_{DEP}, D_{DEP}\} \wedge Rule =$ false

```
        then return A is not meet SchedulingRule;
    end For
end For
return A is valid.
```

算法复杂度分析: 对于租户的每一个原子定制活动而言, 其操作结果是通过替换的方式改变系统已有的状态变量集合 $State=\{P_{CIN}, S_{CIN}, D_{CIN}, P_{CED}, S_{CED}, D_{CED}, S_{DEP}, E_{DEP}, D_{DEP}\}$, 因此每个原子定制操作的时间复杂度为 $O(n)$. 而租户的定制活动 A 是由流程、服务、数据表、数据字段 4 层的原子定制活动组合而成的, 所以 A 的时间复杂度为 $O(n^4)$. 服务商提供了有限的规则, 对系统状态集合做出了约束, 故 *Rules* 的时间复杂度也为 $O(n)$, 因此最终算法的时间复杂度为 $O(n^4)$.

4.6　服务推荐算法

由于租户是基于一个应用实例进行定制的, 在不同租户之间定制的服务是有相关性的, 即存在多数租户共同定制一组服务. 对此, 框架提供了推荐模块, 使用其收集其他租户的定制信息, 为当前租户的定制提供推荐服务.

推荐模块介于数据存储和租户定制之间, 通过推荐提取算法, 从数据库中选择出具有代表性的服务组集合, 如$\{(S_1, S_2), (S_1, S_3)\}$. 该集合表明具有一定数量的租户已经定制了其中的一个或多个服务组合, 其中 S_1 是当前租户已经定制的服务, S_2 和 S_3 均表示模块为租户推荐的下一步定制的服务.

本文设计的推荐定制算法是由数据挖掘的经典算法 Apriori 改进而来, 具体描述如下:

```
list l1, l2, l3, Srem
//集合 l1 代表满足最小支持度要求的服务,
//集合 l2 代表以当前服务为首的服务依赖组合,
//集合 l3 代表满足最小支持度要求的服务依赖组合
for each service si in S        // S 为所有服务的集合
  for each Tenant ti in SCED
    if ti customized si then //租户 ti定制了 si
      si.count++;
  if si.count≥min_sup then //服务 si超过最小支持度
    l1.add(si);                 //将 si放入 l1列表中
for(i=0; i< l1.length; i++)
  if l1[ i] is not current service then
                          //要根据当前服务进行推荐
    continue;                   //所以必须确定当前服务
  else
    for(j=0; j< l1.length; j++)
      if(i≠j) then
        l2.add(l1[ i], l1[ j]);
                    //得到类似(si, sj)的服务组集合 l2
for(k=0; k< l2.length; k++)
  for each Tenant ti
    if ti customized l2[ i] then
                  //如果租户 ti同时定制了组合(si, sj)
      l2[ i].count++; //所包含的两个服务
  if (l2[ i].count≥min_sup∧ l2[ i] ⊂SDEP) then
                          //依赖组合(si, sj)大于
    l3.add(l2[ i]);             //最小支持度
for each (si, sj) in l3
  if((si, sj).count / si.count ≥min_con) then
                        //组合(si, sj)大于最小可信度
    Srem.add(sj);       //得到推荐服务集合 Srem
return Srem
```

在算法中, *min_sup*、*min_con* 分别是最小服务支持度和最小服务可信度, 它们均可由租户设定. 其中, *min_sup* 表示选择推荐服务组合的租户人数不能小于指定的一个阈值; *min_con* 表示选择推荐服务组合的租户人数同只选择组合中前一个服务的租户人数的百分比不能小于指定的一个阈值.

例 **1**.　假设服务集合 S 中包含 $s_1 \sim s_5$ 5 个候选服务, 有 6 个租户 $u_1 \sim u_6$ 的定制结果如下:

u_1: $\{s_1, s_2, s_5\}$;　　u_2: $\{s_1, s_2, s_4, s_5\}$;

u_3: $\{s_2, s_4, s_5\}$;　　u_4: $\{s_1, s_2, s_4, s_3\}$;

u_5: $\{s_1, s_3, s_5\}$;　　u_6: $\{s_1, s_4, s_2, s_5\}$.

推荐模块中租户定义的服务最小支持度为 3, 最小可信度为 0.5. 假定租户 u_7 已定制的服务为 s_1, 下面给出算法对推荐服务集合的计算过程.

(1) 首先对每个服务计算初期支持度:

$s_1.count=5$,　$s_2.count=5$,　$s_3.count=2$,

$s_4.count=4$,　$s_5.count=3$.

可见 s_3 不满足最小支持度 3 的要求, 由此产生服务集合 $l_1=\{s_1, s_2, s_4, s_5\}$.

(2) 以服务 s_1 为首做笛卡尔积, 得到集合

$$l_2=\{(s_1, s_2), (s_1, s_4), (s_1, s_5)\}.$$

(3) 根据已定制的服务组合计算 l_2 中每个组合的支持度, 并通过服务依赖集合数据库 S_{DEP} 对其进行验证.

$(s_1, s_2).count=4$, $(s_1, s_4).count=3$, $(s_1, s_5).count=4$.

3 个组合均满足最小支持度 3 的要求, 假设 S_{DEP} 中描述服务 s_5 不是 s_1 之后所连接的服务, 产生集合 $l_3=\{(s_1, s_2), (s_1, s_4)\}$.

(4) 最后对 l_3 中每个组合计算可信度如下:

$$\frac{(s_1, s_2).count}{s_1.count}=0.8,\quad \frac{(s_1, s_4).count}{s_1.count}=0.6.$$

两者的可信度均满足最小可信度 0.5 的限制.

故最终的推荐服务集合 $S_{rem}=\{s_2, s_4\}$.

算法复杂度分析:假设候选服务数为 n,租户数为 m.算法包含 4 个步骤:第 1 步是计算每个服务的支持度,复杂度为 $O(nm)$;第 2 步是寻找当前服务的依赖组合,复杂度为 $O(n^2)$;第 3 步是计算每个组合的支持度,复杂度为 $O(nm)$;第 4 步是计算每个组合的可信度,复杂度为 $O(n)$,因此最终算法的时间复杂度为 $O(n(m+n))$.

推论 **1**. 对于当前系统状态集合 q(q 为真),如果通过上述算法得到了推荐服务定制活动 $S\{s_i, s_j\}$,并且 PA、EA、DA 均为真,则租户的定制活动必定合理.

推论 1 可通过定理 1 推导证明.

5 流程定制验证示例及实验分析

5.1 考试应用定制示例分析

本节通过一个简单的考试应用示例说明租户个性化定制及验证的过程.

例 **2**. 图 5 上半部显示了一个 SaaS 服务商为租户提供的关于考试应用的基本元素,简单起见,只列出相关部分,下半部显示了租户基于该应用中的元素定制的一个正确的考试流程.在流程中,新用户首先需要保证遵守相关协议,然后填写信息进行注册(只增加了一个出生日期字段)并登陆考试系统,而已注册的用户可以直接登陆到考试系统中.

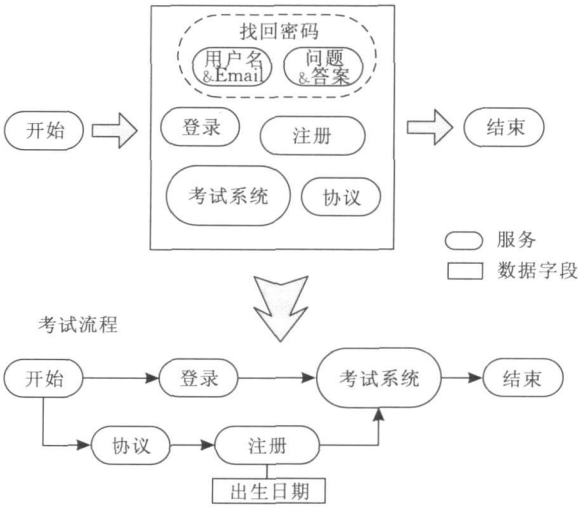

图 5 考试流程的定制

下面描述了在流程 p_{EXAM} 的基础上,租户对该流程的两种不合理定制活动以及相应的验证情况.

情况 **1**. 租户的定制活动违反内部依赖约束关系,如图 6 所示.

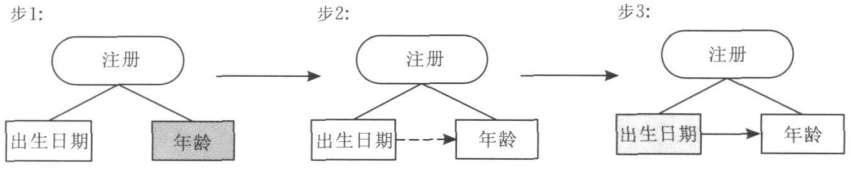

图 6 租户定制过程 1

1. 租户在 $s_{REGISTRATION}$ 服务下定义并配置了一个新的数值型字段 d_{AGE}(该字段属于数据表 $Info$),假设定制初始状态集合为 t,则租户的定制活动 A_1 及相应的状态表示如下(假设服务商没有定义规则,下同):

$$A_1 \triangleq PA \wedge SA \wedge EA \wedge DA$$
$$\triangleq Config\_P(p_{EXAM}) \wedge Config\_S(s_{REGISTRATION}) \wedge Config\_E(Info) \wedge (Add\_D(d_{AGE}, d_{NUM}) \text{ a } Config\_D(d_{AGE}, d_{NUM})),$$
$$t_1 \triangleq t[\![A_1]\!] \triangleq (P_{CED}=\{p_{EXAM}\}) \wedge S_{CED} \wedge S_{DEP} \wedge (D_{CED}=\{E_{Info}, d_{DOB}, d_{AGE}\}) \wedge (D_{DEP}=\varnothing).$$

2. 租户在 $s_{REGISTRATION}$ 服务下定义了一个字段依赖关系,通过公式保证用户输入出生日期字段后,系统会自动生成用户的年龄字段.租户的定制活动 A_2 及相应的状态表示如下:

$$A_2 \triangleq PA \wedge SA \wedge EA \wedge DA$$
$$\triangleq Config\_P(p_{EXAM}) \wedge Config\_S(s_{REGISTRATION}) \wedge Config\_E(Info) \wedge Depend\_D(d_{DOB}, d_{AGE}),$$
$$t_2 \triangleq t_1[\![A_2]\!] \triangleq (P_{CED}=\{p_{EXAM}\}) \wedge S_{CED} \wedge S_{DEP} \wedge (D_{CED}=\{E_{Info}, d_{DOB}, d_{AGE}\}) \wedge (D_{DEP}=\{d_{DOB} \text{ a } d_{AGE}\}).$$

3. 租户在 $s_{REGISTRATION}$ 服务中删除了字段 d_{DOB},由于 d_{DOB} 与 d_{AGE} 之间的依赖关系尚未解除,因此该活动会导致系统的额外信息丢失,不应被允许,将通过公式的方式验证指出.租户的定制活动 A_3 及相应状态如下:

$$A_3 \triangleq PA \wedge SA \wedge EA \wedge DA$$
$$\triangleq Config\_P(p_{EXAM}) \wedge Config\_S(s_{REGISTRATION}) \wedge Config\_E(Info) \wedge Delete\_D(d_{DOB}),$$
$$t_3 \triangleq t_2[\![A_3]\!] \triangleq (P_{CED}=\{p_{EXAM}\}) \wedge S_{CED} \wedge S_{DEP} \wedge (D_{CED}=\{E_{Info}, d_{AGE}\}) \wedge (D_{DEP}=\{d_{DOB} \text{ a } d_{AGE}\}).$$

从 t_3 中可以看到,D_{CED} 和 D_{DEP} 之间存在矛盾,因此 t_3 的结果为假,这意味着活动 A_3 是一个不合理的定制,在没有删除字段依赖关系之前,不能进行该定制活动.

情况 **2**. 租户的定制活动违反外部依赖约束，如图 7 所示.

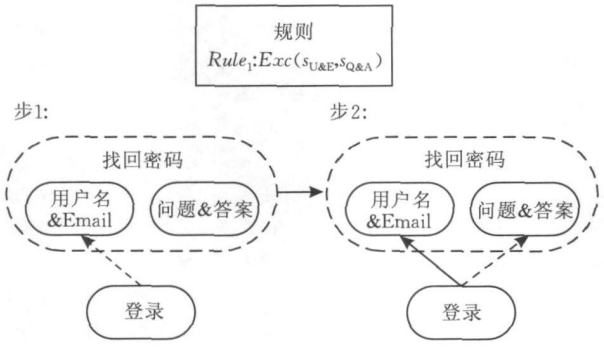

图 7　租户定制过程 2

SaaS 服务商还额外为租户提供了两种登陆时忘记密码时的补偿服务：用户名 &Email 方式和问题 &答案方式，租户可以根据需要将该服务定制到考试流程中. 但服务商同时规定了租户仅可以选择两种服务中的一种进行定制，即 $Rule_1 = Exc(S_{U\&E}, S_{Q\&A})$.

1. 租户选择 $s_{U\&E}$ 服务，建立其与 s_{LOGIN} 之间的依赖关系，定制活动 A_1 及其状态为

$$A_1 \triangleq PA \wedge SA$$
$$\triangleq (Config\_P(p_{EXAM}) \wedge (Add\_S(s_{U\&E}) \,a\, Config\_S(s_{U\&E})) \wedge Depend\_S(s_{LOGIN}, s_{U\&E}),$$
$$t_1 \triangleq t[\![A_1]\!] \wedge Rule_1$$
$$\triangleq (P_{CED} = \{p_{EXAM}\}) \wedge (S_{CED} = S_{CED} \cup \{s_{U\&E}\}) \wedge (S_{DEP} = S_{DEP} \cup \{s_{LOGIN} \,a\, s_{U\&E}\}) \wedge Rule_1.$$

2. 租户选择 $s_{Q\&A}$ 服务，建立其与 s_{LOGIN} 之间的依赖关系，定制活动 A_2 及其状态为

$$A_2 \triangleq PA \wedge SA$$
$$\triangleq (Config\_P(p_{EXAM}) \wedge (Add\_S(s_{Q\&A}) \,a\, Config\_S(s_{Q\&A})) \wedge Depend\_S(s_{LOGIN}, s_{Q\&A}),$$
$$t_2 \triangleq t_1[\![A_2]\!] \wedge Rule_1$$
$$\triangleq (P_{CED} = \{p_{EXAM}\}) \wedge (S_{CED} = S_{CED} \cup \{s_{U\&E}, s_{Q\&A}\}) \wedge (S_{DEP} = S_{DEP} \cup \{s_{LOGIN} \,a\, s_{U\&E}, s_{LOGIN} \,a\, s_{Q\&A}\}) \wedge (s_{U\&E} \in S_{CED} \wedge s_{Q\&A} \notin S_{CED}) \vee (s_{U\&E} \notin S_{CED} \wedge s_{Q\&A} \in S_{CED}).$$

从 t_2 可以看出 S_{CED} 这个状态集合将为假，从而可知 t_2 为假，即活动 A_2 无法执行，因为其违背了服务商的规则 $Rule_1$.

基于这个示例，充分说明了通过对租户个性化行为的建模和相应算法，能够有效验证各种定制活动的正确性.

5.2　实验分析

定理 1 确保了定制结果验证的正确性和合理性. 基于提出的 SaaS 业务流程定制及验证框架，开发了相应的原型系统，并针对原型系统进行实验，分析基于 TLA 的业务流程定制、验证、推荐及整个框架运行的效率.

原型系统开发工具包括 Java 5、Apache Axis2 1.4，流程引擎为 ActiveBPEL 5.0.2，数据库为 MySQL 5.1.22，使用 JBossESB 4.3 作为服务总线，有效地串联起各个模块. 系统的实验评估是通过运行在两台局域网相连的计算机上实现的，操作系统为 Microsoft Windows XP Professional SP 3，其中，一台(Intel Core 2 Duo 2.33GHz，4GB 内存)运行流程引擎及数据库，另一台(Intel Core 2 Duo 2.33GHz，2GB 内存)运行流程定制框架.

针对多租户下的流程定制及验证进行实验，分析在不同租户的情况下，无定制验证和有定制验证时的平均响应时间. 其中，租户定制的服务数目为 10 个，分为 3 类情况进行模拟：第 1 类是不加入约束规则，即 10 个服务可以任意定制；第 2 类是加入了 10 个约束规则；第 3 类是加入了 20 个约束规则，图 8 给出了该实验的结果，从图中可以看出，定制的租户数量和规则越多，相应的定制响应时间会增加，但变化幅度不大.

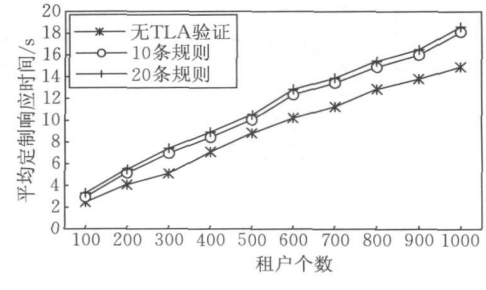

图 8　多租户下的流程定制及验证

考虑多租户下的服务推荐，分析在不同租户、不同定制服务个数的情况下的平均定制响应时间. 其中，租户最多可以定制的服务数目为 10 个，包含了 10 条约束规则，图 9 给出了该实验的结果.

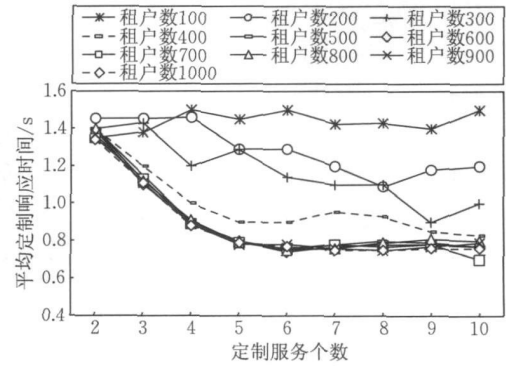

图 9　多租户下的服务推荐

从图 9 可以看出，在应用开始之初，租户数量较少，租户定制的响应时间较长且稳定，随着租户数量

的增多和定制次数的频繁,当新租户进行个性化定制时,推荐模块能够从足够的数据中提取合适的候选服务供租户选择.租户选择推荐服务后,无需进行验证即可进行下一步的定制,因此从总体上降低了定制响应时间,进一步提高了租户的定制效率.

针对定制正确的流程,分析其执行时间开销.共分4组实验,其中:第1组中包含一个定制流程 A,该流程中包含5个服务;第2组包含一个定制流程 B,包含10个服务;第3组包含一个定制流程 C,包含20个服务;第5组包含3个定制流程 A、B、C.从图10中的实验结果可以看出,多租户单流程运行开销相差不大,多租户多流程运行执行的开销相对于多租户单流程定制时增加不大,框架的执行情况总体上比较令人满意.

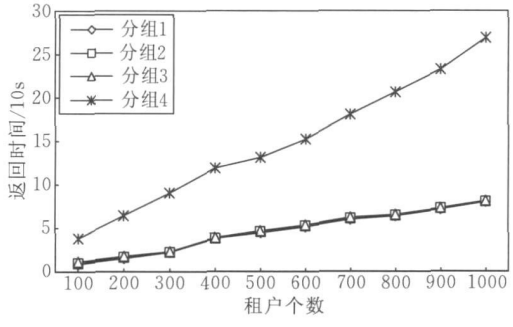

图10 多租户框架的运行开销

6 总 结

本文提出了一个面向SaaS应用的业务流程定制及验证框架,通过TLA对租户个性化定制行为进行建模,形式化地描述了租户在流程、服务及数据层次上可能进行的所有定制活动,并以此为基础设计验证算法,确保租户即将执行的定制活动满足已存在的各项约束条件,进而避免其对整个应用产生难以预测的负面影响.除此之外,通过推荐模块为租户提供候选服务,提高了定制的效率.

下一步的工作主要是对定制行为模型的扩展和完善,使其不仅能够验证租户的定制活动,而且能够对错误的活动进行定位,并提出合理的解决方案.然后将该验证机制集成到一个SaaS平台中,形成一个完整的SaaS应用体系.

参 考 文 献

[1] IBM. Business Process Execution Language for Web Services (BPEL4WS). http://www.ibm.com.developworkers/library/ws-bpel, 2002

[2] W3C. Web Service Choreography Interface (WSCI), World W3C. http://www.w3.org/TR/w-sci, 2002

[3] Jaeger M C, Muhl G, Golze S. QoS-aware composition of Web services: A look at selection algorithm//Proceedings of the 2005 IEEE International Conference on Web Services (ICWS). Orlando, FL, USA, 2005: 807-808

[4] Menasc'e D A, Dubey V. Utility-based QoS brokering in service oriented architectures//Proceedings of the 2007 IEEE International Conference on Web Services (ICWS). Salt Lake City, Utah, USA, 2007: 422-430

[5] Alrifai M, Risse T. Combining global optimization with local selection for efficient QoS-aware service composition//Proceedings of the 18th International World Wide Web Conference (WWW). Madrid, Spain, 2009: 881-890

[6] Liu Shu-Lei, Liu Yun-Xiang, Zhang Fan, Tang Gui-Fen, Jing Ning. A dynamic Web services selection algorithm with QoS global optimal in Web services composition. Journal of Software, 2007, 18(3): 646-656(in Chinese)
(刘书雷, 刘云翔, 张帆, 唐桂芬, 景宁. 一种服务聚合中QoS全局最优服务动态选择算法. 软件学报, 2007, 18(3): 646-656)

[7] Foster H, Uchitel S, Magee J, Kramer J. Compatibility verification for Web service choreography//Proceedings of the IEEE International Conference on Web Services (ICWS). San Diego, California, USA, 2004: 738-741

[8] Liu Fang-Fang, Shi Yu-Liang, Zhang Liang, Shi Bo-Le. Substitution analysis of Web service composition via process algebra. Chinese Journal of Computers, 2007, 30(11): 2033-2039(in Chinese)
(刘方方, 史玉良, 张亮, 施伯乐. 基于进程代数的Web服务合成的替换分析. 计算机学报, 2007, 30(11): 2033-2039)

[9] Tang Xian-Fei, Jiang Chang-Jun, Ding Zhi-Jun, Wang Cheng. A Petri net-based semantic Web services automatic composition method. Journal of Software, 2007, 18(12): 2991-3000(in Chinese)
(汤宪飞, 蒋昌俊, 丁志军, 王成. 基于Petri网的语义Web服务自动组合方法. 软件学报, 2007, 18(12): 2991-3000)

[10] Li Xi-Tong, Fan Yu-Shun. Analyzing compatibility and similarity of Web service processes. Chinese Journal of Computers, 2009, 32(12): 2429-2437(in Chinese)
(李喜彤, 范玉顺. Web服务流程相容性和相似性分析. 计算机学报, 2009, 32(12): 2429-2437)

[11] Ruopeng L, Shazia S, Guido G. On Managing business processes variants. Data and Knowledge Engineering, 2009, 68(7): 642-664

[12] Pietro M, Biplav S. Business driven SOA customization//Athman B, Ingolf K, Tiziana M eds. Proceedings of the 6th International Conference on Service-Oriented Computing. LNCS 5364. Berlin: Springer, 2008: 286-301

[13] Zhang Kuo, Zhang Xin, Sun Wei, Liang Hai-Qi, Huang Ying. A policy-driven approach for software-as-services customization//Proceedings of the 9th IEEE International Con-

ference on E-Commerce Technology (CEC) and the 4th IEEE International Conference on Enterprise Computing (EEE). Tokyo, Japan, 2007: 123-130

[14] Sun Wei, Zhang Xin, Guo Chang-Jie, Sun Pei, Su Hui. Software as a service: Configuration and customization perspectives//Proceedings of the 2008 IEEE Congress on Service Part II. Beijing, China, 2008: 18-24

[15] Ralph M, Frank L. Generation of BPEL customization processes for SaaS applications from variability descriptors// Proceedings of the 2008 International Conference on Services Computing (SCC 2008). Honolulu, Hawaii, USA, 2008: 359-366

[16] Ralph M, Andreas M, Frank L, Klaus P. Variability modeling to support customization and deployment of multi-tenant-aware software as a service applications//Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems (PESOS 2009). Vancouver, Canada. 2009: 18-25

[17] Yuan W, Yushun F. Using temporal logics for modeling and analysis of workflows//Proceedings of the IEEE International Conference on the E-Commerce Technology for Dynamic E-Business (CEC-East). Beijing, China, 2004: 169-174

[18] Lamport L. The temporal logic of actions. DEC Systems Research Center, Palo Alto, CA: Technical Report 79, 1991

SHI Yu-Liang, born in 1978, Ph.D., lecturer. His research interests include service computing, trusted computing and database.

LUAN Shuai, born in 1984, M.S. candidate. His research interests include service computing and database.

LI Qing-Zhong, born in 1965, Ph.D., professor, Ph.D. supervisor. His research interests include database, trusted computing, semantic oriented application integration and information integration.

DONG Jin-Li, born in 1985, M.S. candidate. His research interests include service computing and database.

LIU Fang-Fang, born in 1979, Ph.D., associate professor. Her research interests mainly include Web service, service composition and service discovery.

Background

With the rapid development of Internet and the gradual maturity of SOA, SaaS has become a popular software service mode. SaaS model has several unique technique features, such as single-instance multi-tenancy and customization. The single-instance multi-tenancy refers to enabling single software instance to server multiple tenants. In order to meet personalized business service needs of tenants, SaaS model must provide flexible business process customization mechanism, including process customization, service customization and data customization. Process customization permits tenants to organize their own business process according to their specific business requirements. Service customization permits tenants to select and configure services provided and add them to the process. Data customization permits tenants to add, delete and modify data or data table in data table and process respectively.

Customization support of SaaS gives convenience to tenants for personalization, but it also brings new challenges. Firstly, tenants may ignore internal association of customizations between process, service and data when customization, which cause unpredictable effects. Secondly, tenants may ignore external constraint rule designed by service providers, which makes customization incorrect.

This paper proposes a modeling and verification framework for business process customization. Following the guidance of Hierarchical Customized Behavior Constraint Graph, the framework models the atomic customization action based on TLA (Temporal Logic of Actions), constructs the global customization behavior and designs the verification algorithm for tenant's global customization based on business rule constraints. Based on multi-tenants' feature and historical verified customization, this framework also constructs a customization recommendation algorithms to improve the efficiency of customization for tenants. Experiments demonstrate the efficiency and reliability of this customization and verification mechanism.

The research is supported by the National Natural Science Foundation of China under Grant Nos 90818001 and 60803143, the National Key Technologies R&D Program under Grant No 2009BAH44B02, the Natural Science Foundation of Shandong Province of China under Grant Nos ZR2010FQ026, 2009ZRB019YT and Y2007G38, the Key Technology R&D Program of Shandong Province under Grant No 2010GGX10105.