第 29 卷 第 7 期
2006 年 7 月
计 算 机 学 报
CHINESE JOURNAL OF COMPUTERS
Vol. 29 No. 7
July 2006

一种面向按需集成服务的业务模型构造方法

李长云[1),2)] 阳爱民[1)] 满君丰[1)] 应 晶[2)]

[1)](湖南工业大学计算机科学与技术系 株洲 412008)

[2)](浙江大学计算机科学与技术学院 杭州 310027)

摘 要 针对如何从初始需求出发构造业务模型的问题,提出了一种基于场景的设计方法.该方法强调将显性需求和专家知识有机融合.客户化的场景在设计模式、插件库和本体库的支持下被交互式地创建、求精,最终成为内容完备、结构良好的业务模型,为服务集成、任务协调提供支持.文章也给出了将质量需求分配到软件功能块的策略.最后,举例说明了该方法的应用.

关键词 Web 服务;服务集成;场景;业务模型;UCM

中图法分类号 TP311

An Approach to Constructing Business Model for Services Integration According to Requirement

LI Chang-Yun[1),2)] YANG Ai-Min[1)] MAN Jun-Feng[1)] YING Jing[2)]

[1)](*Department of Computer Science and Technology, Hunan University of Technology, Zhuzhou* 412008)

[2)](*College of Computer Science, Zhejiang University, Hangzhou* 310027)

Abstract Business model not only is the logical scheme for solving the requirements of users, also is the dependence and clue of Web services discovery and execution, so it is the key of services integration according to requirements. In order to construct business model from initial requirements of users, a design approach based on scenario is presented. This approach is to emphasize on the synthesis of explicit requirement and the experts' knowledge efficiently. In the process of the approach the customization scenario is created and refined in interactive fashion with the support of design patterns, plug and ontology library, and becomes a business model with complete content and good structure to provide basis for integrating services and coordinating action. The paper also presents a strategy to allocate the quality requirements to software function blocks. Finally, a example is given to illustrate the application of the approach.

Keywords Web services; service integration; scenario; business model; UCM

1 引 言

Web 服务是一种新兴的应用模式和分布式计算模型,是 Web 上业务数据和信息集成的有效机制.首先,SOAP、WSDL 和 UDDI 为 Web 服务的消息传输、服务定义和发布制定了规范[1].其次,基于构件的软件开发技术的成熟为 Web 服务的快速实现、分发提供了支持[2].一个更具挑战性的问题是动态地按需集成 Web 服务,即当需要的功能不能通过

收稿日期:2006-02-03;修改稿收到日期:2006-04-19.本课题得到霍英东青年教师基金(94030)和湖南省自然科学基金(05JJ30122)资助.
李长云,男,1972 年生,博士,副教授,研究方向为软件体系结构和语义 Web 服务. E-mail: lcy469@163.com. 阳爱民,男,1970 年生,博士,副教授,CCF 高级会员,研究方向为数据与知识工程. 满君丰,男,1976 年生,硕士,讲师,CCF 学生会员,研究方向为协同计算和工作流. 应 晶,男,1971 年生,博士,教授,博士生导师,CCF 高级会员,研究领域为软件自动化.

已有的服务实现时，已有的服务能被合理地组合来满足这个要求.

按需集成 Web 服务实质蕴含两层意思：(1)如何方便、准确地表达用户的需求；(2)如何选择、集成合适的 Web 服务来满足和解决用户的需求. 需求具有不同的抽象层次，目标、任务、功能和过程等可在不同的抽象层次表达需求. 由于用户的需求是主观的和隐式的，不可能一开始就将它们进行完整描述. 一般说来，用户最初提出的需求可能是较抽象的、宏观的，而单个的 Web 服务只能满足具体的、微观的需求，这就造成了两者之间的鸿沟. 业务模型是连接两者的桥梁和纽带. 业务模型是对应用域业务的描述，反映组织的目标和职能，规定业务流程、实体、规则及其之间的关系. 因此业务模型既是对用户初始需求求解的逻辑解决方案，独立于 Web 服务之外，又是选择和集成 Web 服务的依据和线索. 对初始需求进行求精和设计决策，可以形成具有足够丰富细节的业务模型. 依据业务模型可以发现、集成和执行 Web 服务，最终满足和解决用户的需求. 如何由初始需求构造语义清晰、内容完备、结构良好的业务模型成为服务集成中关键的问题.

目前，集成服务普遍的方法是以某种过程描述语言(如 BPEL4WS，WSFL)描述业务过程规格，然后将过程活动绑定到 Web 服务，形成松散耦合的应用程序[3,4]. 但这种方法存在两个问题：(1)过程建模语言本身并不解决从初始需求构造业务过程规格的问题；(2)BPEL4WS 和 WSFL 不能够表达过程中各个活动的语义，使得为完成业务流程所需的服务选取、配置和交互存在障碍.

本文提出的 SBMC(Scenario Based Model Construction for service integration)方法正是针对上述问题而提出的，其中场景(系统执行时的一系列可观察的行为序列)处于核心地位. 场景不仅仅可以诱导、表示需求和产生规格，还能驱动设计和系统的演化. 该方法使客户能够通过需求求精和模式应用来逐步构造业务模型，为服务的按需集成提供依据. 本文第 2 节给出 SBMC 的基本思想；第 3 节介绍 SBMC 的方法和过程，重点阐述设计模式的 UCM(Use Case Maps)表示、搜索选择和使用；第 4 节以实例说明 SBMC 的应用；第 5 节讨论国内外的相关研究；最后是结论和下一步的工作.

2 基本思想

传统的软件开发基本过程是经需求分析、概要设计和详细设计得到设计规范，然后人工编码实现为可执行的程序体；CBSD(Component Based Software Development)方法则是经由需求分析得到软件的宏观体系结构，以软件体系结构为范本进行构件的选择、组装；最近由 OMG 提出的 MDA(Model Driven Architecture)方法则试图从需求规范或设计模型自动生成程序，使人们从手工编码解脱出来[5]. 所有这些表明，在软件过程中的设计决策和其它开发活动日益关注需求的分析和模型的设计.

在软件工程中，尽可能重用已有的软件元素，有助于加快开发进度，同时也有利于提高软件质量. 软件复用技术总体上可以分为两类：微观的和宏观的. 前者(如函数库、类库和构件)侧重于代码及实现阶段软件“产品物件”的复用；后者(如设计模式、体系结构风格和软件框架)侧重于分析和设计阶段的知识和方法的复用. 基于 Internet 的 Web 服务的出现，从软件工程的角度可以认为是提供了无穷的微观可复用构件. 在按需集成服务的过程中，为提高从初始需求构造业务模型的生产率和质量，还应该能够复用公共的设计决策和专家经验知识，即宏观复用.

SBMC 方法的基本思路是：首先用户对需求进行初始的、抽象的描述；然后对其进行半自动的求精和设计，在此过程中强调将问题解决经验与客户的显性需求进行集成，通过融合设计模式和领域解决方案以进行宏观复用，形成具体的业务模型. 这样的业务模型规定了业务的过程、实体及其协作关系，包括责任分配、控制流和数据流，最终依据业务模型在语义层次上进行 Web 服务的发现和集成，形成可执行的业务过程.

需求必须描述系统功能特征和概念结构. 在 SBMC 方法中，使用 UCM[6] 描述需求. UCM 提供一系列表示场景的可视化标记，通过抽象构件之间的路径来集成功能和建立结构视图. 同时，由于 UCM 的求精能力，场景可以被逐步结构化和分层表示，我们也使用 UCM 描述业务模型. UCM 还可以描述和推导关于大粒度的行为模式，支持高层次的体系结构设计决策，因此我们也使用 UCM 来描述设计模式. 一致的表示视图使得需求的求精、模型的构造更加顺畅和方便，提升了软件的复用性.

除了功能性需求，也有非功能性质量需求. 随着软件系统大小和复杂性的增加，质量需求的解决日益重要，也更难处理. 在业务模型的构建过程中，一方面，质量需求涉及到系统的整体性问题，软件结构代表这种整体性问题，它的设计在很大程度上确定

了系统是否能达到要求的质量. 另一方面，软件结构并不能单独保证系统所要求的质量. 为尽可能地满足质量需求，需要将没有或不充分满足的质量需求合理地分配到每个软件功能块上去. 附属了质量约束的软件功能块规格将成为服务发现和匹配的依据.

3　SBMC 方法

3.1　UCM 表示法介绍

UCM 是一种表示场景的图示语言，由 Carleton 大学的 Buhr 教授和他的团队开发，已经应用在一些复杂系统的开发中[7]. UCM 主要由路径(path)、构件(component)、责任(responsibility)和存根(stub)构成. 路径表示场景流，连接起始点、责任点和终点；场景构件是责任点的逻辑执行者，可以代表不同的系统实体，如软构件、组织和伙伴；责任点表示处理(行为、任务或功能)，如果一个责任点位于一个构件内，意味着责任被分配给构件，被构件执行；存根表示暂时不能确定的行为，起“占位符”的作用. 在设计时选取或设计一个插件(plug，子 UCM 图)来具体化存根. 图 1 说明了 UCM 的基本元素.

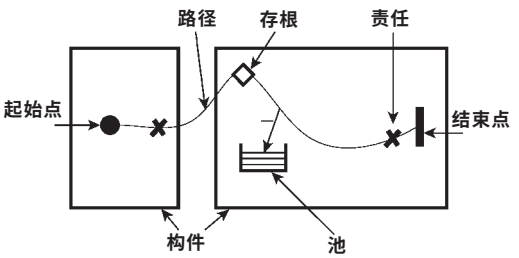

图 1　UCM 的基本标记

传统 UCM 本身是非形式化的，责任点、存根等一般使用自然语言说明. 而在按需集成 Web 服务应用中，要求业务模型的语义尽可能清晰，表示尽可能规范. 我们对传统 UCM 作了一些扩展. 责任点使用元组〈name，rq-function，rq-performance〉描述，name 为责任点命名；rq-function 表示责任点需要完成的功能，是一个三元组〈action，in，out〉，action 是功能要执行的动作名，in 和 out 分别表示功能的输入、输出参数；rq-performance 表示施加于责任点之上的质量需求，是一个二元组集合：$\{\langle Nnfp, Vnfp\rangle\}$，$Nnfp$ 为质量属性，如性能、可靠性等，$Vnfp$ 表示分配在此责任点上质量属性的要求值，初始赋值为 0.

除此之外，当场景变的太复杂而不能作为一个简单 UCM 来描述时，定义和结构化子映射的机制变得很必要. 高层次 UCM 中的 stub 作为一个根映射提出，作为子映射的 plug 用来具体化 stub，从而允许复杂 UCM 场景的层次分解. Stub 使用三元组〈S-name，rq-target，description〉表示，S-name 是 stub 名字，rq-target 定义 stub 的需求目标，description 是一般性描述. Plug 使用四元组〈P-name，ex-target，子 UCM，description〉表示，P-name 是 plug 名称；ex-target 为 plug 的操作目标；子 UCM 描述 plug 的内部实现；description 是一般性描述.

3.2　基于场景的按需集成服务框架

SBMC 本质上是一种客户驱动的模型设计方法. 这种方法是客户直接参与和驱动的交互式过程，通过对初始场景的演化来满足多样化的需求. 在 SBMC 方法中，集成 Web 服务的框架如图 2 所示，它反映了对 Web 服务按需集成特点的认识.

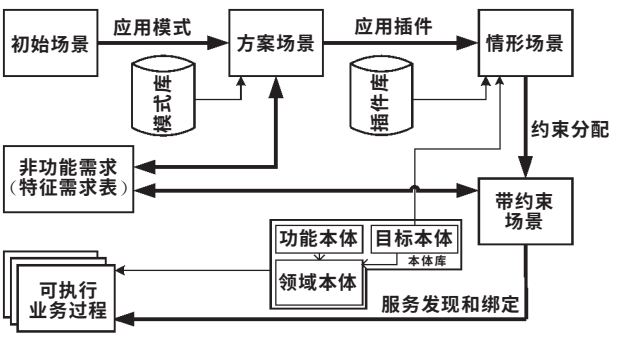

图 2　基于场景的按需集成服务框架

SBMC 方法中的特征需求表表示质量属性需求. 特征需求表是一个三元组集合：$\{\langle Nnfp, Vnfp, SaRa\rangle\}$，其中 $Nnfp$ 表示质量属性的名称；$Vnfp$ 表示质量属性的取值. 一般说来，针对不同的质量属性，取值的范围和形式也应不相同，为简化计算，规定所有取值为整型，范围为[0，10]；$SaRa$ 表示质量属性需求已被满足的程度，取值为[0，1]之间，初始赋值为 0. 在业务模型的构建过程中，质量需求将逐渐被映射到软件结构和软件功能块的质量属性上，$SaRa$ 的值不断增加，直至为 1. 因此，特征需求表将质量需求和问题解决空间连接起来，并在质量属性分配上起着主要作用.

为了使客户能够直接定制和驱动业务模型的构造，必须提供足够的知识支持. 模式库存储各类使用 UCM 表示的设计模式. 设计模式可应用于多种不同场合，提供设计问题的抽象描述和怎样用一个具有一般意义的元素组合来解决这个问题. 插件库存储插件(plug)，插件描述了可重用的、标准化的过程构件(表现为子 UCM 图)，可作为构建业务模型的逻辑单元.

本体库提供共享知识，存储三类本体：领域本体、目标本体和功能本体. 领域本体提供应用领域中概念及其关系的标准定义，确定该领域内共同认可的词汇，并给出这些词汇之间相互关系的明确定义. 目标本体反映了我们希望达到的状态，或希望通过采取某项行动后收到的效果. 前者被称为“需求目标”，后者被称为“操作目标”. 每一个 stub 都包含一个需求目标，每一个 plug 都执行一个操作目标. Stub 中的需求目标通过具有与其相匹配的操作目标的 plug 实现. 功能本体定义行为的功能术语及其之间的关系. 在场景中，每一个责任点都规定了一个功能需求，而每一个 Web 服务操作都实现一个功能. 责任点的功能需求通过执行与其匹配的 Web 服务来完成.

在这个框架中，首先终端用户将初始的功能性需求和系统行为以 UCM 图描绘出来，确定好责任点及承担责任点的场景构件，不能确定的行为路线以 stub 符号表示；同时，质量需求以特征需求表的形式表示. 依据特征需求表，用户到模式库中搜索一个或多个合适的设计模式，根据模式使用方法，应用模式到初始场景，得到蕴含专家解决办法的方案场景，同时评估模式结构对质量属性需求的影响，修改特征需求表中每个质量属性的满足度. 然后，根据方案场景中 stub 符号定义的需求目标，搜索插件库中具有与其匹配的操作目标的 plug，或由用户定制设计一个能实现 stub 需求目标的 plug. 将这些 plug 替换相应 stub 符，得到情形化场景. 接着，将没有充分满足的质量属性需求权衡分配到情形化场景的每一处责任点上，使得责任点不仅表征功能和任务，还附属了质量属性约束，得到带约束场景. 依据带约束场景对每一个责任点进行服务选择、匹配和绑定，使得每个责任点都有一个执行体. 鉴于场景的路径定义了系统之间的控制流和数据流，从业务过程的视图看，每条路径都可视作一个业务过程. 其中绑定了服务操作的责任点转化为过程活动，场景构件可表征为执行活动的组织和承担者. 最终，这多个业务过程可在过程引擎中得到执行和监控.

以下分节讨论上述 SBMC 方法的步骤，重点介绍设计模式的应用方法.

3.3 应用设计模式

设计模式是针对反复出现的设计问题给出的可复用设计决策，通常被认为是构成一个整体的组成成分间的一种结构. 它记录了专家的设计知识，代表对特定环境中出现的问题的专业解决方案. 在面向对象方法中，使用类图表示模式，本文使用 UCM 表示模式.

3.3.1 设计模式的 UCM 表示

一个模式是一个三元组〈Name，UCM，Consequences〉，由一组协作的构件构成，定义各部分的职责和协作关系，其中：

Name 表示模式名称；

UCM 图描述解决方案和设计的组成成分，他们之间的相互关系及各自的职责、协作方式和控制流程. 在模式 UCM 图中，每一个责任点都有一个标注. 标注有三种选项：[mandatory]、[optional] 或 [extension]. [mandatory] 表示这是一个强制的、必不可少的责任点，作为模式应用时的功能嵌入点；[optional] 表示可选责任点；[extension] 表示扩展、附加的责任点.

Consequences 描述模式应用的效果. 显示地列出这些效果对理解和应用模式很有帮助. 效果反映模式对软件质量属性的影响，包括系统的性能、可靠性、扩展性等. 效果是一个二元组集合：$\{\langle Nnfp, PVnfp\rangle\}$，其中 $Nnfp$ 表示质量属性的名称，类型为 String；$PVnfp$ 表示模式对质量属性的影响取值，规定取值为整型，范围为[0，10].

举例说明，Observer 设计模式描述如下：

〈Observer，图 3 所示的 UCM，(〈expansibility，8〉，〈reliability，6〉，〈performance，4〉)〉.

图 3 为 Observer 模式的 UCM 图，表明多个观察者向出版者订阅信息，一旦有观察者订阅的信息，出版者主动通知每个观察者. 设计模式的 Consequences 部分说明 Observer 模式的扩展性好、可靠性一般而性能较差.

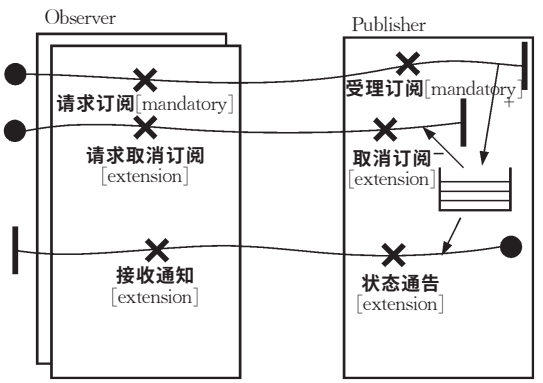

图 3 观察者设计模式的 UCM 表示

3.3.2 模式的搜索方法

在软件设计过程中人们所面临的问题不单是考虑系统的功能需求，而且要解决更难处理的非功能性质量需求. 将设计模式应用到场景中以确定整个

系统的总体框架和良好的软件结构，归根到底，是为了解决软件的质量需求. 因此，从模式库搜索模式依据的是模式的效果对质量属性需求的满足度. 满足度的计算算法如下：

对于每一个质量属性需求〈$Nnfp$，$Vnfp$，$SaRa$〉(来自特征需求表)，模式提供的对应效果为〈$Nnfp$，$PVnfp$〉，在质量属性 $Nnfp$ 上定义满足度函数：

$$PSat=\delta\times(10-Rat)/10 \tag{1}$$

其中 δ 表示模式结构在此维属性上对系统的影响因子，依据不同的属性和具体的应用特点权衡拟定，取值范围为[0，1]；

$$Rat=\begin{cases}Vnfp-PVnfp, & Vnfp\geq PVnfp\\ 0, & Vnfp<PVnfp\end{cases}.$$

满足度函数反映模式对某一质量属性需求的满足度.

模式对质量需求总的满足度计算如下：

$$PSat_{\mathrm{sum}}=\sum_{n}^{i}(w_i\times Psat_i) \tag{2}$$

其中，n 代表了属性总数，$w_i\in[0,1]$是依据用户偏好对属性 $Nnfp_i$所拟的权重，$\sum w_i=1$；$PSat_i$表示模式对属性 $Nnfp_i$的满足度.

满足度高的模式被选中，同时修订特征需求表{〈$Nnfp$，$Vnfp$，$SaRa$〉}，使得属性 $Nnfp_i$的 $SaRa$ 等于其对应的满足度 $PSat$.

由于客户的需求是很复杂的，许多需求是隐性的，因此，利用该模型得到的质量度量对于客户而言只是一种参考，并不全面，但相对其它方法简单而有效，反映了在结构方面对质量属性需求的满足度(体现在特征需求表中的满足度维上).

3.3.3　模式的应用方法

在选定了设计模式后，用户将模式应用到初始场景中去，分为 3 个步骤：

(1) 构件融合. 使模式中至少一个构件匹配初始场景中的某个构件. 这种匹配由用户手工指定，其依据是构件在模式和初始场景中担当角色和代表的协作伙伴的一致. 至少有一个模式中的构件被指定，但不需要模式中的每个构件都在初始场景中有匹配的构件. 匹配了的模式构件融入初始场景中所指定的构件，即这两个构件合二为一.

(2) 责任点融合. 对于模式中被匹配了的构件中所有标注为强制([mandatory])的责任点都必须在初始场景相应构件中具有功能相容的责任，并且这两个责任点合二为一，即模式责任点融入初始场景的对应责任点，其定义维持初始场景的责任点规格. 所谓功能相容，意指两个责任点间具有以下关系：要么一个责任点的功能包含了另一个责任点的功能，要么一个责任点的功能是另一个责任点功能的强化(即一个功能可以通过另一个功能的满足而得到满足). 对于被匹配了的模式构件中标注为可选([optional])的责任点如果在初始场景相应构件中存在有类型相容的责任点，则融入初始场景的对应责任点，否则丢弃.

(3) 模式嵌入. 如果责任点融合成功，则进行模式嵌入任务. 标注为扩展([extension])的模式责任点与可选责任点相比，不需要任何初始场景中的责任点被提供，扩展责任点永远被保留. 没有被匹配的模式构件及包含其中的责任点、模式中的所有路径、Stub 和其它图符定义都被保留并嵌入初始场景.

在执行这 3 个步骤时应该遵循两个规则：

(1) 结构规则. 如果一个 UCM 路径连接两个构件，意味着这两个构件彼此通信，这两个构件间存在一个结构关系. 通信可以是直接的或间接的，结构关系可分为 1 对 1 或 1 对多. 在应用模式时应维持结构关系的前后一致，初始场景的 1 对 1 关系可以采纳模式中 1 对多关系，反之不行. 在应用模式时应维持通信方向的一致性，同时初始场景中的直接通信在应用模式后可以转变为间接通信，反之则不行.

(2) 边界规则. 所有被封装在模式中一个构件内的责任点在应用到初始场景后必须被置于一个构件里. 这种机制确保内聚的责任点不会被分散到多个构件中.

显然，通过模式的应用，专家知识和公共设计决策被引入了初始场景中，初始场景演化为方案场景.

3.4　应用插件

UCM 也能表达结构化分层 UCM. Stub 反映了一个 plug 在协作中所要达到的目标，它起到“占位符”的作用；plug 用来填充 stub，使复杂映射能够分层递解. Stub 引用的目标称为需求目标，plug 引用的目标称为操作目标，stub 中的需求目标通过选择那些具有与其相匹配的操作目标的 plug 来实现. 为求精 stub，参照目标本体库，从 plug 库中搜索符合 stub 目标要求的一个 plug，替换相应的 stub.

文献[8]针对 Web 服务领域提出了一个目标本体模型. 在这个模型中，目标包含动作、对象、方法和质量四个方面. 鉴于 stub 和 plug 的目标只需表示“做什么”(动词和对象)的问题，并不涉及到“如何做”(方法和质量，由 UCM 图结构和 UCM 责任点规格所共同定义)，我们对文献[8]的目标本体模型做了简化和适应性修改. 在 SBMC 的目标本体模型中，目标(target)由动作(action)、对象(object)构成，表示成二元组〈action，object〉. 动作表示目标需要实现的意图，对象由领域本体表示，它指定了动作

所操作的实体.

在 SBMC 方法中,本体使用 Web 本体语言 OWL[9]描述. OWL 包含以下元语:类(class,表示概念)、关系(relation)和实例(instance)等. 目标本体使用 OWL 表示如下:

```
<owl:Class rdf:ID="Target"/>
<owl:Class rdf:ID="Action"/>
<owl:Class rdf:ID="Object"/>
<owl:ObjectProperty rdf:ID="Target-Act">
  <rdfs:domain rdf:resource="#Target"/>
  <rdfs:range rdf:resource="#Action"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Target-Obj">
  <rdfs:domain rdf:resource="#Target"/>
  <rdfs:range rdf:resource="#Object"/>
</owl:ObjectProperty>
```

OWL 也提供了描述本体间基本关系的元语如 sameAS(同一关系)、Instance Of(例化关系)、subClassOf(泛化关系)和(Intersection)交集关系. 基于描述逻辑,OWL 也具备这些关系的判定能力. 关于本体间关系的判定规则和算法可参见文献[10].

从 plug 库中搜索 plug 的具体过程是:首先参照目标本体库并依据 OWL 的推理规则(可借助某种自动化工具如 Racer)对 stub 和 plug 的目标相容性进行筛选. 要使得 plug 能够实现 stub,必要的条件就是 plug 的操作目标 g 和 stub 的需求目标 t 存在如下关系之一:

$$g\ sameAS\ t;$$
$$g\ instance\ of\ t;$$
$$g\ subClassOf\ t.$$

道理是显而易见的,例如目标"购买车"是"购买摩托车"的泛化,即〈purchase,motorcycle〉*subClassOf*〈purchase,vehicle〉,自然地,操作目标为"购买摩托车"的 plug 可实现需求目标为"购买车"的 stub.

接下来就是按照路径做进一步的筛选,要求 plug 和 stub 的进入和出去的路径段能保持一致,即从外部观察者角度看,两者的进入路径数量相同,出去路径数量也相同. 经过这些筛选后,如果获得的 plug 组件数目还是比较多,用户可以通过某种策略进行最终选择,例如随机化方法.

除了从 plug 库中搜索 plug 的方法之外,客户还可以直接设计定制 plug 来实现 stub 的需求目标. Plug 本身也可以含有 stub,这些 stub 也通过另外的 plug 来实现,从而形成场景的分层递解和细化. 经过对方案场景中所有 stub 的求精和具体化,越来越多的细节被增加,方案场景演化为情形场景.

3.5 质量约束分配

优良的软件结构设计一定程度上满足了非功能性质量需求,但并不能单独保证系统所要求的质量. 质量需求一方面被映射到软件结构上,另一方面也将被细化分解到软件构造块的质量属性上. 为尽可能地满足质量需求,需要将没有或不充分满足的质量需求合理地分配到每个责任点上去.

这个分配过程是由客户推动的,难以自动化进行,因为没有可定义的分配规则可应用. 然而参考软件工程学中关于软件质量属性保证的相关成果,有一些可借鉴的方法,这里推荐质量展开法来保证分配的合理. 质量展开法的基本思想是在定义和开发各阶段,运用质量表,将用户的质量需求分布到软件的各个功能部分,作为各个部分的质量属性.

建立反映质量属性与责任点对应关系的矩阵,形成质量表,如表 1 所示.

表 1 质量属性与责任点对应矩阵

责任点	质量属性								
	性能			可靠性			扩展性		
	权值 (w)	满足度 ($RSat$)	分配值 ($VnFp$)	权值 (w)	满足度 ($RSat$)	分配值 ($VnFp$)	权值 (w)	满足度 ($RSat$)	分配值 ($VnFp$)
责任点 1									
责任点 2									
责任点 3									
…									
总和	1			1			1		

对于每一个质量属性,要衡量其在每一个责任点上的权值和满足度. 权值反映该责任点对非功能质量属性的重要程度;满足度权衡分配有关责任对质量属性需求的满足度. 首先拟定责任对质量属性的权值,所有责任对某一维属性的权值总和应为 1. 接着分配责任对属性的满足度. 所有责任对某维属性的满足度计算如下:

$$RSat_{\mathrm{sum}} = \varepsilon \times \sum_{n}^{i} (w_i \times Rsat_i) \tag{3}$$

其中 ε 表示质量展开在此维属性上对系统的影响因子,依据不同的属性和具体的应用特点拟定,取值范围为[0,1];n 是责任点数,$w_i \in [0,1]$是责任点 i 在

此维属性上的权值；$RSat_i$是责任点 i 对此维属性的满足度. 通过调整 $RSat_i$取值，使得 $RSat_{sum}$大于或等于此维属性在特征需求表中的$(1-SaRa)$值(此时的 $SaRa$ 值反映了结构层次方面对质量需求的满足度，$(1-SaRa)$反映还待满足的质量需求). 确定 $RSat_i$后，计算并修改责任点 i 的质量要求值 $VnFp_i$，公式为 $VnFp_i=RSat_i\times 10$. 这样就将质量属性落实到各个责任点上去了，得到带约束场景.

带约束场景既融入了领域解决方案，体现了客户化定制的能力，又是一个具体而微、有丰富语义的逻辑业务模型，从而可以对每一个责任点进行服务选择和匹配，成为服务发现的依据和服务执行的线索.

3.6　服务发现

服务发现首先需要获得服务的描述. 文献[11]给出了一个较通用的 Web 服务描述模型$\{S,C,P\}$，其中 S 是基本描述，即服务名称和文本描述，C 是服务功能描述，P 是服务质量描述. 大多数 Web 服务描述语言如 WSDL 和 OWL-S 都在一定程度上自发地遵循$\{S,C,P\}$模型. 不失一般性，本文的服务发现机制以$\{S,C,P\}$模型作为讨论基础，并假定服务描述采用本体标注.

场景中的每个责任点都应该被一个 Web 服务实现，服务发现的任务就是为带约束场景中的每一个责任点选择一个最优的服务. 由于带约束场景中的责任点明确规定了功能要求和质量要求，因此 SBMC 服务发现机制的基本思想是：首先责任点和服务进行基于功能的匹配，产生一个候选服务集，然后在此服务集合上进行基于 QoS 的匹配，从而发现最优的服务. 服务的自动发现过程具体如下：

1. 依据带约束场景中责任点的功能规格筛选出能够执行该功能的 Web 服务. 一个服务能够实现指定的场景责任点，必要条件就是这个服务的功能 C 和责任点的功能规格 rq-function 间存在下述关系之一，

C *sameAS*　rq-function；

C *instance of*　rq-function；

C *subClassOf*　rq-function.

这个条件的检查可以参照功能本体库由机器自动进行. 功能本体库定义了功能概念及其之间的关系. 功能本体的定义类似于目标本体，由动作 Action、输入参数 Input、输出参数 Output 构成，其中输入、输出参数引用领域本体表示. 功能本体使用 OWL 表示为

```
〈owl：Class rdf：ID="Function"/〉
〈owl：Class rdf：ID="Action"/〉
〈owl：Class rdf：ID="Input"/〉
〈owl：Class rdf：ID="Output"/〉
〈owl：ObjectProperty rdf：ID="Fun-Act"〉
  〈rdfs：domain rdf：resource="#Function"/〉
  〈rdfs：range rdf：resource="#Action"/〉
〈/owl：ObjectProperty〉
〈owl：ObjectProperty rdf：ID="Fun-In"〉
  〈rdfs：domain rdf：resource="#Function"/〉
  〈rdfs：range rdf：resource="#Input"/〉
〈/owl：ObjectProperty〉
〈owl：ObjectProperty rdf：ID="Fun-Out"〉
  〈rdfs：domain rdf：resource="#Function"/〉
  〈rdfs：range rdf：resource="#Output"/〉
〈/owl：ObjectProperty〉
```

服务的功能描述 C 包含要实现的功能动作名、输入和输出参数几大部分，由 3.1 节知责任点的功能规格 rq-function 由三元组〈action，in，out〉形成，如果功能本体库中容纳了足够应用领域里的功能术语及其之间的关系描述，显而易见，参照功能本体库并借助 OWL 的推理工具可自动判定服务的功能描述和责任点的功能要求间是否存在上述关系，从而获得候选的 Web 服务集合.

2. 在候选服务集上进行基于 QoS 的匹配. 某一候选服务与责任点的 QoS 参数匹配结果是通过计算该服务提供的质量描述对责任点的质量要求的满足度并进行加权平均得到的，其计算公式为

$$SSat=\sum_{n}^{i}(w_i\times SSat_i),$$
$$\sum_{n}^{i}w_i=1,\ 0\le w_i\le 1,\ i=1,2,\cdots,n \tag{4}$$

其中，n 代表属性总数，w_i是依据用户偏好对第 i 个质量属性所拟的权重，$SSat_i$表示责任点对于服务的第 i 个质量属性的满足度，

$$SSat_i=(10-Rat_i)/10,$$
$$Rat_i=\begin{cases}VnFp_i-SVnFp_i, & VnFp_i\ge SVnFp_i\\ 0, & VnFp_i<SVnFp_i\end{cases} \tag{5}$$

其中，$VnFp_i$是责任点在第 i 个质量属性上的要求值，$SVnFp_i$是服务的描述模型$\{S,C,P\}$中 P 提供的第 i 个质量属性上的值，反映服务满足第 i 个质量属性的真实程度，取值范围为[0，10].

通过 QoS 匹配操作，将得到候选服务集中的每个服务的对责任点的质量属性满足度，其中值最高的为最优匹配，选择该服务并将其绑定到责任点上以完成责任点的功能要求.

4　案例研究

设一电子购物系统的业务模型设计过程如图 4 所示. 图 4(a)是用户使用 UCM 描绘的初始需求：顾客向电子零售商注册帐号成为购物客户；购物客户向零售商发出购物请求，零售商处理请求(处理细节待定). 将观察者模式(图 3 所示)应用到初始场景：

指定观察者对应客户构件，出版者对应零售商构件. 在观察者模式中，“请求订阅”和“受理订阅”责任点被标识为[mandatory]. 假设在初始场景中“注册信息”包含顾客希望了解的商品动态和购买意向等信息，则“请求订阅”和“受理订阅”责任点的功能分别被“请求注册”，“受理注册”责任点的功能所包含. 使得“请求订阅”责任点融入“请求注册”责任点，“受理订阅”责任点融入“受理注册”责任点，然后嵌入观察者模式的其它部分，得到方案场景，如图 4(b)所示. 此时场景增加了“取消注册”和“商品信息通知”(一旦客户所订阅的商品信息发生变动，零售商主动通知客户)的路线过程，体现了专家经验知识的复用. 图 4(c)是一个领域 plug，表征零售商处理购物请求的内部过程：如果自身库存能满足客户请求，则直接响应客户，否则向供货商咨询供货信息. 此 plug 的操作目标刚好满足图 4(b)的 stub 需求目标，则使此 plug 替换 stub 后得情形场景，如图 4(d)所示. 进一步，对情形场景作质量特征约束分配，就将得到带约束场景. 依据带约束场景可以发现、匹配并集成 Web 服务.

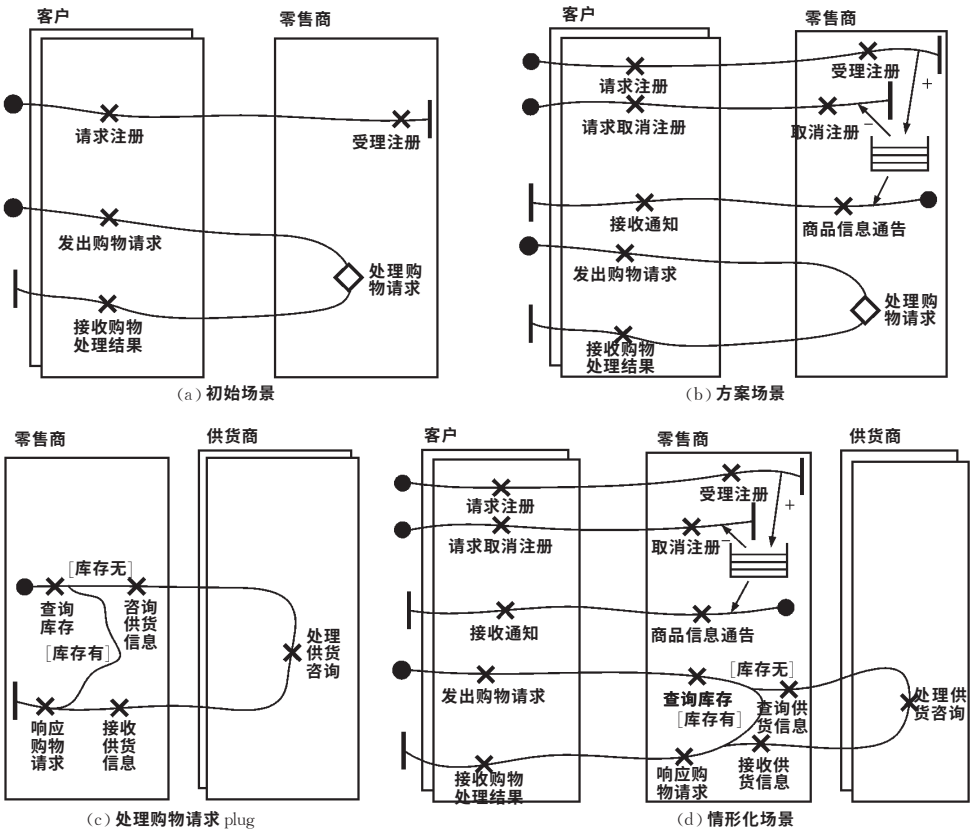

图 4 一个电子购物系统的场景演化

由案例可见，SBMC 是一种以场景为中心，由客户直接参与和驱动的交互式过程，通过需求求精和模式应用来逐步构建服务集成方案.

5 相关工作和比较

传统的业务建模方法，如 ARIS、CIMOSA、GRAI/GIM 等[12]，通常将模型的重点放在如何使用不同视图对企业进行完备描述，其共同特点是对业务模型所涉及的实体、过程、目标和功能进行充分和全面的表达. 例如 ARIS 使用 4 个视图(组织、数据、控制、功能)对业务进行描述，并通过控制视图来关联实体、数据和功能. 近来人们也较广泛使用 UML 并采取 4+1 视图(过程、设计、实现、配置和用例视图)对业务模型进行描述. 有些研究也以业务规则的形式体现模型的灵活可重构性[13]，但缺少足够的业务细节，没有形成相对清晰和完整的体系. 而作为服务集成线索的业务模型不仅要求足够的业务细节，而且要是一个集成化的可执行模型，因此上述方法并不完全适合服务集成场合.

目前国内外对 Web 服务集成框架的研究可概括为两种类型：第 1 种是从工作流的角度，将 Web 服务看作工作流活动；第 2 种是从软件工程的角度，将 Web 服务看作分布的软件构件，即基于构件的服务集成. 前者普遍的方法是在 WSDL 基础上，使用过程定义语言如 BPEL4WS 将这些服务集成为一个

整体[3,4]. BPEL4WS 的目标是在过程中将服务的引用从真正的服务实现中抽象出来，有助于业务逻辑和服务相分离并实现服务的动态绑定. 语义 Web 服务的研究也方兴未艾，直接应用语义 Web 服务语言 OWL-S(DAML-S)的过程模型和推理能力进行服务集成也得到了广泛的研究[11,14]. 其优点是对 Web 服务的语义描述和 Web 服务的集成从一个视角看待，使用统一的语言处理. 但无论是 BPEL4WS 还是 OWL-S，其本身并不解决从初始的需求构造业务过程规格的问题.

从软件工程的角度，为集成这些服务构件，构造软件体系结构作为集成蓝图成为必要. 体系结构刻画了系统的集成信息，其中包含构件接口规约和构件之间的连接关系. 人们在进行软件总体设计时，首先根据用户需求和实现环境的要求，确定系统的体系结构；在体系结构层次上，构件接口规约对外唯一地代表了构件，故从服务构件库中查找符合接口规约的服务构件；最后，通过组装工具将构件接口处的集成映射到服务构件的集成，即将原子服务构件组装成复合服务构件或完整的系统. 例如文献[15]使用图来描述体系结构，为服务集成提供上下文.

总之，基于工作流的服务集成从业务过程的角度定义业务逻辑，关注系统的行为视图和控制流程，具有较好的灵活性和动态性，但忽略了系统的总体结构和难于复用成功的设计方案. 基于构件的服务集成从宏观上规定了系统的实体结构和协作行为，易于引进专家经验和知识，提升了软件的复用性. 本文工作结合了这两方面的优势，UCM 既是一个体系结构视图，明确了构件之间的责任分配和协作关系，又是面向工作流的，每一条路径都定义了一个业务过程.

另外还有基于 PSM(Problem-Solving Method)的服务集成框架[16]、基于计划的服务集成方法[17]. 基本的思想是将用户的需求表示为目标或问题，使用人工智能中的规划和推理技术形成达到目标或解决问题的方案，最终将方案映射到一系列的服务上从而实现方案. 例如 WSMF(Web Service Modeling Framework)[16]使用 UPML(The Unified Problem-Solving Method development Language)来构建业务方案和作为服务集成时的协调者. UPML 是一种 PSM 开发语言，提供构件、适配器和连接配置的概念来描述和实现 PSM 框架. 本文描述的服务集成框架与这些工作的相同点在于：都是从满足客户的需求为出发点，构建业务模型和需求解决方案，并通过服务的选择、匹配和执行而实现需求. 本文工作的不同点在于：该方法不是基于规划方法的，它是由客户直接参与和驱动、可定制的交互式过程；统一使用 UCM 作为描述客户功能需求、设计模式和业务模型的语言，同时强调将问题解决经验与客户的显性需求知识进行集成.

6　结论与展望

动态地、按需集成 Web 服务首先要求解决业务需求的表示和求精，其次应该对服务的发现、选取和集成提供足够的依据和线索. 业务模型是连接两者之间的桥梁. 本文提出的 SBMC 方法解决了如何从用户的初始需求出发设计语义清晰、内容完备、结构良好的业务模型问题. 该方法强调将显性需求和专家知识有机集成，使客户能够通过需求求精和模式应用来逐步构建服务集成方案. 在该方法中，统一使用 UCM 作为描述功能需求、设计模式和业务模型的语言，一致的表示视图使模型的建立更为方便，软件的复用性得到提高.

我们已经开发了一些工具来支持该方法. 将来的工作将包含以下方面的内容：(1)使用更为合理的方法来进行责任的质量约束分配. 目前的质量展开法需要用户准确衡量质量属性在每一个责任上的权值和满足度，这对一般用户未免要求太高，不足以应付复杂的情形；(2)进一步改进设计模式的表示和使用方法，例如目前使用三种标注来说明模式中责任类型，以后可能设计更多种标注来分类责任并提高模式应用的自动化程度.

参　考　文　献

1　Curbera F., Duftler M., Khalaf R., Nagy W., Mukhi N., Weerawarana S.. Unraveling the Web services Web: An introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing, 2002, 6(2): 86～93

2　Brown A., Johnston S., Kelly K.. Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications. IBM(American), A Rational Software Whitepaper from IBM, 2003

3　Mandell D. J., McIlraith S. A.. Adapting BPEL4WS for the semantic Web: The Bottom-Up approach to Web service interoperation. In: Proceedings of the 2nd International Semantic Web Conference, Florida, 2003, 227～241

4　Leymann F., Roller D., Schmidt M. T.. Web services and business process management. IBM Systems Journal, 2002, 41(2): 198～211

5　David S., Frankel. Applying MDA to Enterprise Computing. New York: John Wiley & Sons Press, 2002

6 Buhr R. J., Casselman R. S.. Use Case Maps for Object-Oriented Systems. USA: Prentice Hall, 1995

7 Buhr R. J.. Use case maps as architectural entities for complex systems. IEEE Transactions on Software Engineering, 1998, 24(12): 1131～1155

8 Cao Jian, Zhang Shen-Sheng, Li Ming-Lu. A goal driven and process reuse based Web services customization model. Chinese Journal of Computers, 2005, 28(4): 721～730(in Chinese)
(曹 健,张申生,李明禄.基于目标驱动和过程重用的 Web 服务客户化定制模型.计算机学报, 2005, 28(4): 721～730)

9 Dean M., Schreiber G. *et al*. OWL Web Ontology Language Reference. W3C Candidate Recommendation. http://www.w3c.org/TR/owl-ref/. 2003

10 Baader F., Nutt W.. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge: Cambridge University Press, 2002

11 Hu Jian-Qiang, Zou Peng, Wang Huai-Min, Zhou Bin. Research on Web service description language QWSDL and service matching model. Chinese Journal of Computers, 2005, 28(4): 505～513(in Chinese)
(胡建强,邹 鹏,王怀民,周 斌. Web 服务描述语言 QWSDL 和服务匹配模型研究.计算机学报, 2005, 28(4): 505～513)

12 Fan Yu-Shun, Wang Gang, Gao Zhan. An Introduction to the Theory and Methodology of Enterprise Modeling. Beijing: Tsinghua University Press, 2001(in Chinese)
(范玉顺,王 刚,高 展.企业建模理论与方法学导论.北京:清华大学出版社, 2001)

13 Hu Jin-Min, Zhang Shen-Sheng, Yu Xin-Ying. A workflow model based on ECA rules and activity decomposition. Journal of Software, 2003, 13(4): 761～767(in Chinese)
(胡锦敏,张申生,余新颖.基于 ECA 规则和活动分解的工作流模型.软件学报, 2002, 13(4): 761～767)

14 McIlraith S., Son T. C., Zeng H.. Semantic Web services. IEEE Intelligent Systems, Special Issue on the Semantic Web, 2001, 16(2): 46～53

15 Ma Xiao-Xing, Cao Jian, Lu Jian. Architecting distributed Web application: A graph-oriented approach. Chinese Journal of Computers, 2003, 26(9): 467～477(in Chinese)
(马晓星,曹建农,吕 建.一种面向图的分布 Web 应用架构技术.计算机学报, 2003, 26(9): 467～477)

16 Fensel D., Bussler C.. The Web service modeling framework WSMF. Electronic Commerce Research and Applications, 2002, 1(2): 113～137

17 Aiello M., Papazoglou M., Yang J.. A request language for Web-services based on planning and constraint satisfaction. Lecture Notes in Computer Science 2444. Berlin: Springer Verlag, 2002, 76～85

LI Chang-Yun, born in 1972, Ph.D., associate professor. His research interests include software architecture and semantic Web services.

YANG Ai-Min, born in 1970, Ph. D., associate professor. His research interests include data and knowledge engineering.

MAN Jun-Feng, born in 1976, M. S., lecturer. His research interests include cooperative computing and workflow.

YING Jing, born in 1971, Ph. D., professor, Ph. D. supervisor. His research interests include software automation.

Background

This work is supported by the Fork Ying Yung Science Foundation for Yong Teacher under grant No. 94030 (Research on Architecture-Oriented Software Evolution) and the Natural Science Foundation of Hunan Province of China under grant No. 05JJ30122 (Research on the Framework of Integrating Semantic Web Services Using Process Ontology).

There is a common task among these projects, how to construct business model, and the key issue is to introduce experts' knowledge during the design phase of model. This paper is devoted to solve how to construct server-oriented business process model from the requirement of users and design patterns, and is considered as an important part of those project listed above.