**提示信息：**为保证本文中的链接能够使用，请您先将本文件按规定的命名方式命名，并与“[USTB硕士学位论文规范及论文制作指南.exe](file:///C:\Users\DELL\AppData\Roaming\Microsoft\Word\USTB硕士学位论文规范及论文制作指南.exe)”保存于同一目录。

不用此信息时，删除此框。

（鼠标移到此框四边，鼠标变为十字箭头，点击边框选中此框，然后按Del删除）

北京科技大学

面向全生命周期的适应性服务组装方法的研究

张在兴

**密　　　　级：\_\_\_\_\_\_\_\_\_\_\_\_\_**

论文题目：面向全生命周期的适应性服务组装方法的研究

G20158650

学　 　号：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

张在兴

作　 　者：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

计算机技术

专 业 名 称：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2017年11月10日

面向全生命周期的适应性服务组装方法的研究

Research on Full Lifecycle-Oriented Adaptive Service Composition Technique

研究生姓名：张在兴

指导教师姓名：孙昌爱

北京科技大学计算机与通信工程学院

北京100083，中国

Master Degree Candidate： Zhang Zaixing

Supervisor： Sun Chang-ai

School of Computer and Communication Engineering

University of Science and Technology Beijing

30 Xueyuan Road，Haidian District

Beijing 100083，P.R.CHINA

分类号：\_\_\_\_\_\_\_\_\_\_\_\_ 密　　级：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

004.41

TP311

１０００８

ＵＤＣ：\_\_\_\_\_\_\_\_\_\_\_\_ 单位代码：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**北京科技大学硕士学位论文**

面向全生命周期的适应性服务组装

方法的研究

**论文题目：**

张在兴

**作者：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

北京科技大学

孙昌爱教授

**指 导 教 师： 单位：**

**指导小组成员： 单位：**

**单位：**

**论文提交日期：**2017年 11月 13日

**学位授予单位：北 京 科 技 大 学**

致 谢

岁月如歌，光阴似箭，转眼之间两年半的研究生生活已经接近尾声，回首这两年半的求学历程，往事历历在目，每一次的进步，都离不开同学和老师的支持与帮助，对那些引导我、帮助我、激励我的人充满了感激，在此向他们表达我最衷心的感谢。

首先我要感谢我的研究生导师孙昌爱教授，孙老师不仅从学术上给予我指导，在生活中还给予我无微不至的关怀，同时教给我做人的道理，让我感念至深。在这两年半的时光里，孙老师渊博的专业知识，严禁的求学态度，精益求精的工作作风和诲人不倦的高尚师德对我影响深远，为我以后的工作和学习树立了优秀的学习榜样。从初到课题组的课题研究，再到毕业论文的选题，最后到论文的定稿，都是在孙老师的悉心指导完成的，每个环节都倾注了导师大量的心血，在此谨向孙老师致以最诚挚的感谢。

在论文的完成过程中，何啸老师给我了很多的指导和建议，为我点拨迷津，帮助我解决了很多问题，何老师为人师表的风范令我敬仰，在此向何老师致以由衷的敬意。

其次感谢实验室中和我一起度过这既艰辛又快乐的两年半时光的同学们，正是有了你们的鼓励和陪伴，我才能克服生活困难，突破学习中的瓶颈，同时你们积极乐观的生活的态度让我受益匪浅。愿在今后的人生道路上，我最亲爱的同学们可以策马扬鞭，奋勇向前。同时感谢王真同学对课题工作的大力支持和帮助。

最后，向百忙之中抽出宝贵时间评阅本论文的老师们致以诚挚的谢意。

摘 要

近年来，面向服务的架构（SOA）逐渐成为分布式环境下的应用程序开发的一种主流范型或方法学。Web服务为实现SOA架构提供了基础，但是由于单一的Web服务的功能有限，因此服务组装被广泛应用于构造分布式的应用程序。课题组在前期工作中通过扩展BPEL开发了适应性服务组装语言VxBPEL，通过将业务流程中的变化建模为变体并在运行时支持其动态配置，实现灵活的业务流程。但是在当前的服务组装过程中，需求分析和流程建模糅杂在一起，整个服务组装过程缺少需求层次的指导，并且业务流程的可变性只存在于设计阶段，缺少对需求层次可变性的考虑。

针对上述问题，本文探索一种面向全生命周期的适应性服务组装方法，以特征模型驱动服务组装开发的过程，提供相应的特征模型建模工具和管理工具支持。取得的主要研究成果如下：

1. **提出一种面向全生命周期的适应性服务组装方法：**该方法将需求分析和流程建模分离开来，由需求分析得到领域的特征模型，并将特征模型转换为抽象服务组装模型，从而以特征模型来驱动服务组装开发过程中的需求分析、流程建模、组装、部署和执行。
2. **开发了一个特征模型建模工具SCFeatureModel：**该工具以插件的形式运行于Eclipse平台，支持服务组装领域特征模型的建立。
3. **设计并实现了一个****特征模型和服务组装管理工具FM2VxBPEL：**该工具提供对适应性服务组装全生命周期过程的支持，包括将特征模型转换后得到抽象服务组装模型转换为符合标准语言的VxBPEL、用户配置方案的创建和验证以及测试用例的执行等。
4. **实例研究：**模拟汽车组装领域系统验证本文所提出的面向全生命周期的适应性服务组装方法的有效性，并测试了所开发的SCFeatureModel和FM2VxBPEL两个工具的实用性。

本文所提出的方法将服务组过程中需求分析阶段和流程设计阶段明确分离开来，流程建模过程不再糅杂需求分析，并引入特征模型，将特征模型作为需求分析的产物，以特征模型驱动服务组装的过程。本文提出的面向全生命周期的适应性服务组装方法和支持工具对通过服务组装实现的具有复杂需求的软件系统具有重要的意义。

关键词： 服务组装，适应性服务组装，特征模型，模型驱动架构

Research on Full Lifecycle-Oriented Adaptive Service

Composition Technique

Abstract

In recent years, Service Oriented Architecture (SOA) has become an application development paradigm widely recognized in both academia and industry. Web services provide the foundation for the implementation of SOA. However, single Web services have limited functionality, so service compositions are widely used to construct various distributed applications. Our group has developed a variability-supported service composition language, VxBPEL, by extending BPEL, and developed an adaptive service composition approach, which models possible alternative treatments of a business process as variants, and supports the dynamic configuration of variants at runtime. In the current service composition, requirement analysis and business process modeling are mixed together, and the whole service composition process lacks the guidance of the hierarchy of requirements. The variability of business processes only exists in the design phase, lack consideration for variability in requirement analysis phase.

In order to address the above limitation, this thesis explores a full lifecycle-oriented adaptive service composition technique, and feature model drive service composition. We also developed the corresponding feature model modeling tool and management tool to facilitate the proposed technique. The main contributions made in this thesis are as follows.

1. **Lifecycle-oriented adaptive service composition technique:** In the technique, requirements analysis and process modeling will be separated, and we can gain domain feature model by requirements analysis. Next, we will convert the feature model into abstract service composition model, the feature model can be used to drive all the phases in service composition, including requirement analysis, business flow modeling, service composition, business flow deployment and execution.
2. **A feature model modeling tool:** It was developed by using Eclipse Modeling Framework and can run as plug-in in Eclipse platform. What's more, we can create feature model in service composition domain by using the tool.
3. **A feature model and service composition management tool:** It provides the support for development process of adaptive service composition, including converting abstract service composition into standard grammar VxBPEL, creating and validating user configuration plan and executing test cases.
4. **An empirical evaluation:** It was conducted to validate of the feasibility of the lifecycle-oriented adaptive service composition technique by simulating automobile assembly domain and evaluate the practicality of the supporting platform,SCFeatureModel and FM2VxBPEL.

In summary, the proposed lifecycle-oriented adaptive service composition technique introduce the domain feature model, the feature model will as the product of requirements analysis, requirements analysis and process design will be separated in service composition, and feature model will drive all the phases in whole service composition process. The approach and related supportive tools presented in this thesis is very helpful for service composition software systems with complex requirements.

Key Words： Service Composition, Adaptive Service Composition, Feature Model, Model Driven Architecture

目 录

[致 谢 V](#_Toc498438656)

[摘 要 VI](#_Toc498438657)

[Abstract IV](#_Toc498438658)

[目 录 VI](#_Toc498438659)

[1 引言 1](#_Toc498438660)

[2 研究背景 3](#_Toc498438661)

[2.1 相关概念与关键技术介绍 3](#_Toc498438662)

[2.1.1 相关概念 3](#_Toc498438663)

[2.1.2 关键技术介绍 8](#_Toc498438664)

[2.2 国内外研究现状 11](#_Toc498438665)

[2.3 课题研究背景 14](#_Toc498438666)

[2.4 主要研究成果 14](#_Toc498438667)

[2.5 论文的组织结构 15](#_Toc498438668)

[3 特征模型驱动的适应性服务组装方法 16](#_Toc498438669)

[3.1 特征模型驱动的适应性服务组装方法框架 16](#_Toc498438670)

[3.2 特征与特征模型 17](#_Toc498438671)

[3.3 抽象服务组装模型 22](#_Toc498438672)

[3.4 特征模型到抽象服务组装模型的映射 24](#_Toc498438673)

[3.5 小结 26](#_Toc498438674)

[4 特征模型建模工具SCFeatureModel的设计与实现 27](#_Toc498438675)

[4.1 特征模型Ecore模型 27](#_Toc498438676)

[4.2 SCFeatureModel的需求分析 30](#_Toc498438677)

[4.3 特征模型建模工具的实现 30](#_Toc498438678)

[4.3.1 特征模型Ecore模型的创建 30](#_Toc498438679)

[4.3.2 特征模型建模工具插件的生成 31](#_Toc498438680)

[4.4 特征模型建模工具的运行与演示 33](#_Toc498438681)

[4.5 小结 34](#_Toc498438682)

[5 特征模型到抽象服务组装模型的转换与实现 35](#_Toc498438683)

[5.1 VxBPEL的元模型 35](#_Toc498438684)

[5.2 特征模型到抽象服务组装模型转换的机制 37](#_Toc498438685)

[5.3 小结 40](#_Toc498438686)

[6 特征模型和服务组装管理工具FM2VxBPEL的设计与实现 42](#_Toc498438687)

[6.1 FM2VxBPEL的需求分析 42](#_Toc498438688)

[6.2 FM2VxBPEL系统设计 44](#_Toc498438689)

[6.2.1 系统架构 45](#_Toc498438690)

[6.2.2 特征模型管理器 45](#_Toc498438691)

[6.2.3 流程管理器 46](#_Toc498438692)

[6.2.4 配置管理器 46](#_Toc498438693)

[6.2.5 引擎管理器 46](#_Toc498438694)

[6.3 FM2VxBPEL的实现 47](#_Toc498438695)

[6.4 工具演示与验证 49](#_Toc498438696)

[6.4.1 特征模型管理 49](#_Toc498438697)

[6.4.2 流程管理和配置管理 50](#_Toc498438698)

[6.5 小结 52](#_Toc498438699)

[7 实例验证 53](#_Toc498438700)

[7.1 实验目的及实验设计 53](#_Toc498438701)

[7.2 汽车组装系统 53](#_Toc498438702)

[7.2.1 汽车组装系统领域需求分析 53](#_Toc498438703)

[7.2.2 实验评估 54](#_Toc498438704)

[7.3 实验结果与结论 58](#_Toc498438705)

[8 结论 60](#_Toc498438706)

[作者简历及在学研究成果 61](#_Toc498438707)

[独创性说明 62](#_Toc498438708)

[关于论文使用授权的说明 62](#_Toc498438709)

[参考文献 63](#_Toc498438710)

# 引言

近年来，随着软件开发技术的快速发展，面向服务的架构（Service-Oriented Architecture，SOA）作为一种组件模型，渐渐地成为Internet异构环境下分布式应用程序开发的一种主流方法学[[[1]](#endnote-2)]。SOA是一种粗粒度、耦合度比较低的服务架构，其将应用程序的不同功能单元（称为服务）作为基本构件，服务之间通过简单的、精确定义的接口进行通讯，实现了与底层编程接口、通讯模型的分离，在异构环境中可快速地进行应用程序的开发。

Web服务具有平台和语言的独立性，是SOA架构典型实现方式。由于单个的Web服务所提供的功能有限，需要将多个Web服务组合起来提供更强大的服务功能。从业务流程的角度来看，将多个Web服务按照一定的规格说明组装起来起来去支持灵活的、复杂的业务过程或构造新的Web服务的方式称为Web服务组装（也称Web服务组合）[[[2]](#endnote-3)][[[3]](#endnote-4)]。Web服务组装已经成为异构环境下应用软件开发的重要方式之一[[[4]](#endnote-5)]。通过服务组装能够提高开发效率并实现功能更加复杂、灵活度更高的应用软件。

然而面向服务的系统运行环境具有高度变化性和不可预测性，比如说当前系统的一个或者多个服务由于某种原因暂时失效，此时系统面临不可使用的风险。为了保证系统的服务质量（QoS）并且满足时刻发生变化的业务需求，基于服务的系统应具有一定的灵活性和适应性来对这些变化做出快速反应[[[5]](#endnote-6)]。

WS-BPEL（Web Service Business Process Execution Language）是基于SOA架构且支持面向过程的可执行服务组装语言[[[6]](#endnote-7)]，其通过服务编制（Orchestration）的方式实现Web服务组合，WS-BPEL提供一个总控流程，协调不同Web服务之间不同操作的执行。但是WS-BPEL对服务组装系统的灵活性和适应性的支持能力有很大的局限性，无法有效支持服务组装的的可变性设计，其局限性主要表现在：

1. 使用标准的BPEL语法所描述的业务流程执行语言，一旦部署到BPEL执行引擎以后，如果需要更改流程逻辑，需要重新修改业务流程并且在修改完成后重新部署，开发效率低；
2. 为了应对多变的业务需求，如果有可供选择的方式来实现目标，那么这些选项必须在流程规格说明中进行预定义，不可避免的需要在流程中引入大量的分支选择，从而导致业务流程过于复杂和庞大，不利于系统的维护。

为改善传统业务流程对灵活性和适应性支持能力的不足，课题组在前期的工作中探讨了如何从服务组装的规格说明考虑适应性问题，提出了一种支持可变性设计的适应性服务组装方法[[[7]](#endnote-8)][[[8]](#endnote-9)][[[9]](#endnote-10)][[[10]](#endnote-11)]，通过扩展标准的服务组装语言BPEL开发了支持适应性服务组装的语言VxBPEL[[[11]](#endnote-12)]。VxBEPL从服务组装的规格说明层考虑适应性问题，弥补了标准BPEL对系统可变性支持上的不足，并且对传统的BPEL执行引擎进行了改造，以支持VxBPEL对可变性设计与配置的解释与执行[[[12]](#endnote-13)]。同时为了提高VxBPEL对复杂服务依赖关系的表达能力，引入了一种基于约束的可变性设计与配置方法[[[13]](#endnote-14)]；为了提高业务流程对无法预期的变化的支持能力引入了动态绑定技术和抽象代理机制[[[14]](#endnote-15)]；为了能够支持SaaS模式第四级成熟度模型，扩展了基于可变性管理的适应性服务组装方法，使得运行时流程实例多态共存、互不影响，并基于此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[[[15]](#endnote-16)]。

传统的服务组装流程开发过程如图1-1所示，主要包括流程建模、流程组装、流程部署和流程运行。但是现在还没有一套相对成熟的体系、系统化的组装方案用于指导服务组装的整个开发过程。当前的服务组装开发过程在设计层考虑可变性和适应性，并没有在需求层次考虑需求的共性和可变性问题，同时需求层和设计层缺少映射关系，需求分析和流程建模柔和在一起，没有明确分离开来，整个服务组装的开发过程缺乏需求分析的指导；对于需求分析的结果也缺少形式化的表达；当有新的需求加入或者原有的业务需求发生改变的时候，都需要在设计层次进行调整或者对业务逻辑重新构造，开发效率相对较低。



图1-1传统的服务组装开发过程活动图

针对上述问题，本文引入了领域特征模型和模型驱动技术，提出了一种面向全生命周期的适应性服务组装方法。通过领域需求分析过程中得到特征模型作为贯穿整个Web服务组装过程中的模型工具，利用特征模型来驱动整个业务流程的开发过程。

# 研究背景

首先介绍本文所提出的方法设计到的相关概念和开发工具所使用到的关键技术，然后对国内外的研究现状、课题研究背景及本文主要研究成果作了简述，最后给出本文的组织结构。

## 相关概念与关键技术介绍

### 相关概念

本小主要节介绍可变性管理、SOA、领域分析和模型驱动等相关概念。

1. **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[[[16]](#endnote-17)]，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[1]。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。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[[[17]](#endnote-18)]。

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[[[18]](#endnote-19)]。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，因此不能很好的满足用户需求。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。

例如，一个旅行社系统提供机票预定和酒店预订服务，每个服务都有两个来自不同服务提供商提供的具体实现。根据对旅行开销的需求，该系统提供两种服务组合，一种实现了低成本的服务组合方案，其旨在提供高性价比的旅行体验，另一种服务组合方案旨在提供更好质量的旅行体验，从而选择价格相对较高的服务。在这个业务流程中，有两个低层次变异点分别代表两个服务调用的位置，即机票预订和酒店预订，每个变异点下有两个变体分别表示来自不同服务提供商提供的具体服务。可以看出，该流程存在4(2^2)种描述不同场景的服务组合方案，其中两种服务组合方案并没有反应真实的业务需求。为提供一个有效的选择方案集合，我们将两个低层变异点间的依赖关系实现为一个高层变异点，进而提供两种能够直接反映业务需求的有效变体组合方案。这种在高层变异点中实现低层变异点间依赖关系的过程被称为“实现关系”。

1. **SOA相关概念**

面向服务架构（SOA）是一种分布式应用程序开发的主流范式，以服务为基本构件，用于快速、低成本、多重用地在异构环境中构建应用程序来满足新的需求[1]。由于SOA是以服务为基本单位，具有松耦合、粗粒度的特点，它可以根据需求，将位于网络上任何地点的服务进行组装，每个服务可以作为其它服务的一部分，也可以直接为最终用户提供服务。Web服务是SOA的一种典型的实现方式，将Web服务作为SOA软件的基本构件，可以使用现有的Web服务，快速地构建更加灵活、复杂的业务系统，从而有效解决Web服务的复用、异构程序的集成、软件灵活度的提高等问题[4]。

Web服务是能够支持网络上机器间互操作的软件系统[[[19]](#endnote-20)]。每个Web服务都需要有各自的Web服务描述文件，这是一种使用WSDL(Web Services Description Language)[[[20]](#endnote-21)]书写的Web服务规格说明文件，用于描述本服务的接口，包括定义信息格式、数据类型、传输协议等，请求者通过WSDL文件可以明确对服务的请求方式，服务之间的通信使用SOAP（Simple Object Access Protocol）[[[21]](#endnote-22)]。Web服务为业务流程的开发和部署提供了基础支持，使用具有标准接口和协议的Web服务能够准确、方便地构建业务流程。

实现服务组装的方式可以分为编制（Orchestration）和编排（Choreography）两种[[[22]](#endnote-23)]。在编制方式中，流程的规格说明在服务组装过程中充当核心控制者的角色，该规格说明定义业务总体目标、相关操作和他们的调用序列。在这种集中管理环境下，Web服务能够在不了解彼此情况下进行添加和删除，并在发生异常和错误时进行补偿处理[[[23]](#endnote-24)]。而以编排方式实现的服务组装并不依赖一个总控流程，每个涉及其中的Web服务都知道何时执行自己的操作，和谁交互。

* **WS-BPEL**

WS-BPEL是一种面向服务、基于XML的可执行的业务流程编程语言。WS-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。WS-BPEL流程的规格说明由伙伴链接声明、变量声明、处理器（handler）声明和业务逻辑四部分组成。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）。WS-BPEL的结构化活动是按照某种结构形成的一组活动，描述了业务流程是怎样通过把它执行的基本活动组成结构而被创建的，这些结构表达了涉及业务协议的流程实例间的控制形式、数据流程、故障和外部事件的处理以及消息交换的协调。结构化活动包括“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）。一个简单的流程实例如图2-1所示。

<process name="warehouse">

<sequence>

<receive partnerLink="retailer" operation="request" …/>

<invoke name="Warehouse" partnerLink="warehousePL"

portType="waa:WarehousePT" operation="InquireGoods"

inputVariable="input" outputVariable="warehousemessage">

</invoke>

<assign>

<copy>

<from>’warehouse can not receive the bill.’</form>

<to part="orderResponse" variable="output" />

</copy>

<reply name="reply" operation="OrderGoods" partnerLink="ordergoodsPL"

portType="order:OrderGoodsPT" variable="output">

</reply>

</sequence>

</process>

图2-1 简单的WS-BPEL流程实例

* **Apache ODE**

WS-BPEL语言主要是通过BPEL引擎来解释执行，目前工业界已经有一系列成熟的引擎，主要有Apache ODE(Orchestration Director Engine)[[[24]](#endnote-25)]、ActiveBPEL[[[25]](#endnote-26)]、Oracle BPEL Process Manager[[[26]](#endnote-27)]、WebSphere Process Server[[[27]](#endnote-28)]等。本课题使用的引擎主要是Apache ODE，Apache ODE是一款能够执行遵循WS-BPEL标准的开源执行引擎，支持现有WS-BPEL规范（1.0、1.1和2.0）。Apache ODE主要功能是支持标准的服务组装语言WS-BPEL描述的业务流程的执行，他提供长期和短期流程执行机制、高层次的API，允许任何通信层与内核进行集成，并支持过程的热部署。Apache ODE系统架构的关键模块包括ODE BPEL编译器、ODE BPEL运行时、ODE数据访问对象（DAOs）、ODE集成层（ILs）和用户工具。由于Apache ODE是开源引擎，因此可以对该引擎的源码进行修改实现其功能的扩展。

* **VxBPEL**

业务流程通常由业务流程执行语言（WS-BPEL）实现，为适应快速变化的需求并从根本上实现对变化的建模，需要对业务流程执行语言WS-BPEL进行扩展。课题组在前期工作中对WS-BPEL进行了扩展，设计了一种支持可变性设计的服务组装语言VxBPEL[11]，VxBPEL提供变异点、变体、实现关系等可变性构造子，同时对BPEL的执行引擎Apache ODE进行了扩展[12]，以支持VxBPEL语言。

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2-2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>

图2-2 VxBPEL中的变体定义语法

图2-3 VxBPEL中的变异点定义语法

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**>

<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2-3所示。

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**>

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:ConfigurableVariati-onPoint>构造子进行定义，通常置于规格说明的</process>元素前。图2-4示意了实现关系的语法定义。其中，“id”属性作为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。

图2-4 VxBPEL中的可变性配置变异点定义

1. **领域分析**

许多学者对领域进行了不同的定义，本方法沿用Krut和Zalman对领域的定义。Krut和Zalman认为领域是指一组具有相似或者相近软件需求的应用系统所覆盖的功能区域[[[28]](#endnote-29)]，领域又称为程序家族（Program Family）或软件产品线（Software Product Line）。由于软件制品的使用环境往往十分相近，领域工程（Domain Engineering）是指利用领域知识和逻辑功能的相似、内聚及相对稳定等特点为某一特定问题域设计和实现可复用软件制品的过程，领域工程的最高目标就是系统化的软件重用。

领域工程阶段的主要任务是确定领域范围，识别并定义领域内共性和可变性，共性作为领域产品的公共部分是软件系统的基础，而不同领域产品的差异性体现了软件产品的可变性。而领域分析作为领域工程的起始阶段，与需求分析区别在于其从单个应用的分析转向了一个领域内相关的多个应用的需求的收集和分析。领域分析需要相关专家以领域知识为切入点，通过对比、分析以往不同时间上的软件系统实例，并结合可预见的领域功能演化，总结领域的问题空间，并在问题空间内区分功能性需求和非功能性需求，在领域范围内提取共性需求和可变性需求。同时，需求分析人员需要根据领域分析的结果对领域需求进行建模，对需求的建模能够便于设计人员对需求的理解，需求模型为以后领域软件制品的开发提供指导。

随着领域工程几十年的发展，领域分析技术已经取得了一些学术性和工程性的进展。当前领域分析方法有很多，包括面向目标的领域分析方法[[[29]](#endnote-30)]、面向对象的领域分析方法[[[30]](#endnote-31)]、基于本体的方法[[[31]](#endnote-32)]等等。Kang等人于1990年提出了面向特征的领域分析方法（Feature-Oriented Domain Analysis，FODA）[[[32]](#endnote-33)]，FODA方法被公认为是最实用的领域建模技术。

1. **模型驱动**

随着当前软件系统越来越复杂，随之而来的便是软件复杂度越来越高，其业务逻辑与具体的实现技术的高度耦合将降低软件系统的正确性、稳定性以及后期的可维护性。如果业务需求和领域知识混杂于具体的实现代码中，那么当新的业务变更必然会造成程序流程的修改，由于代码的非直观性和复杂性等缺陷，修改代码不可避免的会导致系统出现新的缺陷和错误，使得软件生命周期的各个阶段（需求分析、软件设计、实现代码）间产生不一致，严重影响所开发系统的正确性。

为了解决上述问题，以模型为中心的模型驱动软件开发方法逐渐被广泛应用[[[33]](#endnote-34)]。模型驱动的软件开发方法核心思想是将模型看作是系统的第一类软件制品，通过直观的、贴近人的思维模式的方式描述系统功能和逻辑，一切系统构建活动都可以归结为模型的创建和转换。软件开发的过程中，设计人员以模型为指导进行开发，业务逻辑和具体实现技术的耦合性大大降低，有效增强系统的正确性、可靠性和健壮性。

从模型驱动技术的描述中可以看出，模型是模型驱动的核心关注点，是系统功能、结构、行为的形式化的规范。对于设计人员来说，软件开发关注的重点不再是程序，而是模型。

随着模型驱动技术的提出和发展，许多的研究机构和大型公司提出了自己的模型驱动开发方法，包括对象管理组织（OMG）提出的模型驱动架构（Model-Driven Architecture，MDA）[[[34]](#endnote-35)]、统一建模语言（Unified Modeling Language，UML[[[35]](#endnote-36)]和Eclipse建模架构（Eclipse Modeling Framework，EMF）[[[36]](#endnote-37)]。

### 关键技术介绍

本小结主要介绍本文用到的关键技术，包括Eclipse建模框架和模型转换语言ATL。

1. **Eclipse建模框架（Eclipse Modeling Framework，EMF）**

Eclipse建模架构（Eclipse Modeling Framework，EMF）[36]是一种基于Eclipse的创建模型和数据集成框架，作为模型驱动的体系结构（Model-Driven Architecture，MDA）的重要组成部分，它已经成为了开发Eclipse项目的基础，例如图形化编辑框架（Graphical Editing Framework，GEF）、BPEL设计器（BPEL Designer）等。

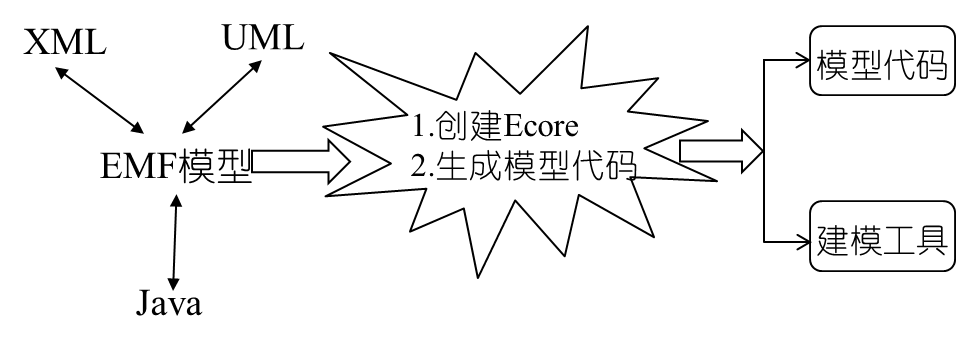


图2-5 EMF使用场景

众多周知，开发人员可以使用Java（接口）、XML或者UML去进行某个领域的建模，而EMF作为一个非常强大的代码和架构的生成工具，它统一了Java、XML和UML这三种经常用来定义概念模型的方法，即EMF允许开发人员用这三种技术定义模型，同时这三种模型都符合一定的设计规则，通过EMF模型可以实现这三种模型的转换，并且采用XML元数据互换（XML Metadata Interchange）作为ECore模型的串行化形式。换句话说，无论开发人员使用何种技术定义某个领域的模型，EMF模型都是将这些模型联系在一起的共同高级表示，图2-5显示了EMF是如何统一Java、XML、和UML这三种技术及其使用方式。

在软件系统领域有不同的模型，例如功能模型、类模型、目标模型等等，而元模型（Meta Model）描述了组成模型的元素以及元素之间的关系的规范，是为了描述模型而提出的，换句话说，元模型就是模型的模型。如图2-6所示，元模型的概念是相对于模型的，离开了模型，元模型这个概念也就失去了意义。在EMF中，元模型称之为Ecore模型，而Ecore模型实质也是一个模型，并且其元模型是它本身。

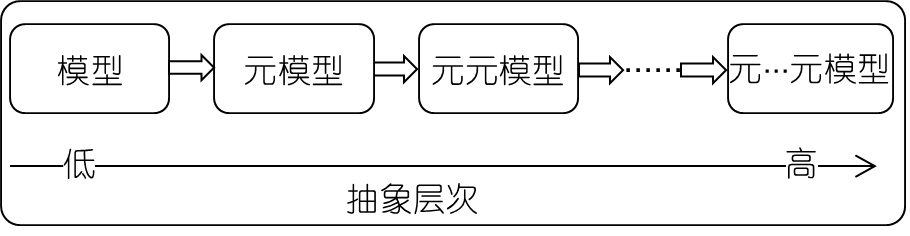


图2-6 模型与元模型

完整的Ecore模型基类很多，为了便于读者理解，本文对Ecore模型进行了简化（如图2-7所示）。从图2-7中我们可以看到，Ecore模型主要包括4个Ecore类，分别是EClass、EAttribute、EReferences、EDateType，表2-1是对主要Ecore类元素的详细介绍。

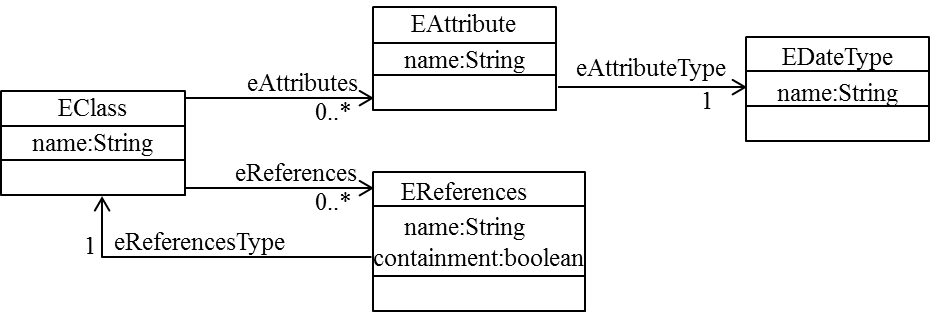


图2-7 Ecore元模型的简化子集

表2-1 组成Ecore类的主要元素

|  |  |
| --- | --- |
| **Ecore类** | **描述** |
| EClass | 表示建模类，包括名称(name)、零个或者多个属性(eAttributes)、零个或者多个引用(eReferences) |
| EAttribute | 表示模型属性，包括名称(name)和属性类型(eAttributeType) |

表2-1 组成Ecore类的主要元素（续）

|  |  |
| --- | --- |
| **Ecore类** | **描述** |
| EReferences | 表示类之间关联的一端，包括名称(name)、用来指示是否包容的布尔标记(containment)、引用类型(eReference Type) |
| EDataType | 表示属性的数据类型，包括基本数据类型(int、float、double等)和对象类型(java.util.Data) |

除了上表中所列举的组成Ecore类的主要元素外，其它的元素还包括EAnnotation(表示模型批注)、EOperation（表示模型行为）、EEnum（表示枚举类型，是一种特殊的数据类型），通过实例化Ecore元模型中定义的类来定义自己所需要的应用程序模型时，将创建Ecore模型实例，并且Ecore模型实例的以XML元数据交换格式（XML Metadata Interchange，XMI）进行序列化。

1. **模型转换语言（ATLAS Transformation Language，ATL）**

在软件工程领域，上并没有源模型和目标模型两个概念，也没有统一的建模标准，从本质上来说，这两个模型都是以某种形式化的方式来描述软件。而在模型转换的过程中，为了能够用来区分输入模型和输出模型便衍生出了源模型和目标模型这两个概念，并且源模型和目标模型必须具有相对应的元模型。通常来说，以模型转换机制为分割点，输入到模型转换机制的模型就是源模型，而模型转换机制通过预定义的一系列转换规则得到的新的模型就是目标模型，此时源模型和目标模型原本两个毫无关系的模型便存在了必不可少的关系，比如两个模型之间的数量应对关系（一对一、一对多、多对一或者多对多）、层次映射关系（比如元模型映射）。

ATL是一种声明式和命令式混合的编程语言，它的声明式定义了源模型元素和目标模型元素之间的关系，将源模型和目标模型复杂的映射关系进行了简化；而它的命令式作为声明式的补充，详细描述了从源模型转换到目标模型的步骤。软件开发人员定义源模型到目标模型的转换规则，使用ATL提供的模型转换机制，将大大节省模型转换时间，并且提高模型转换的效率。

模型转换的过程就是将源模型作为模型转换机制的输入模型，随后模型转换机制按照预定义的一系列转换规则与源模型所提供的信息创建目标模型的信息，并由这些信息构建目标模型，预定义的转换规则是基于源模型和目标模型的元模型。ATL模型转换框架如图2-8所示，首先，源元模型（Source MetaModel）和目标元模型（Target MetaModel）的元模型由Ecore定义的，源模型和目标模型分别根据源元模型和目标元模型建立；然后，模型转换语言源模型与目标模型之间的一系列转换规则都由模型转换语言来进行定义，模型转换语言ATL实质也是一个模型，因此也要符合其元模型的标准；最后，符合源元模型的源模型作为ATL模型转换引擎的输入，ATL模型转换引擎将对所预定义的转换规则进行解释与执行，自动生成符合目标元模型的目标模型，从而完成两种异构模型的转换。

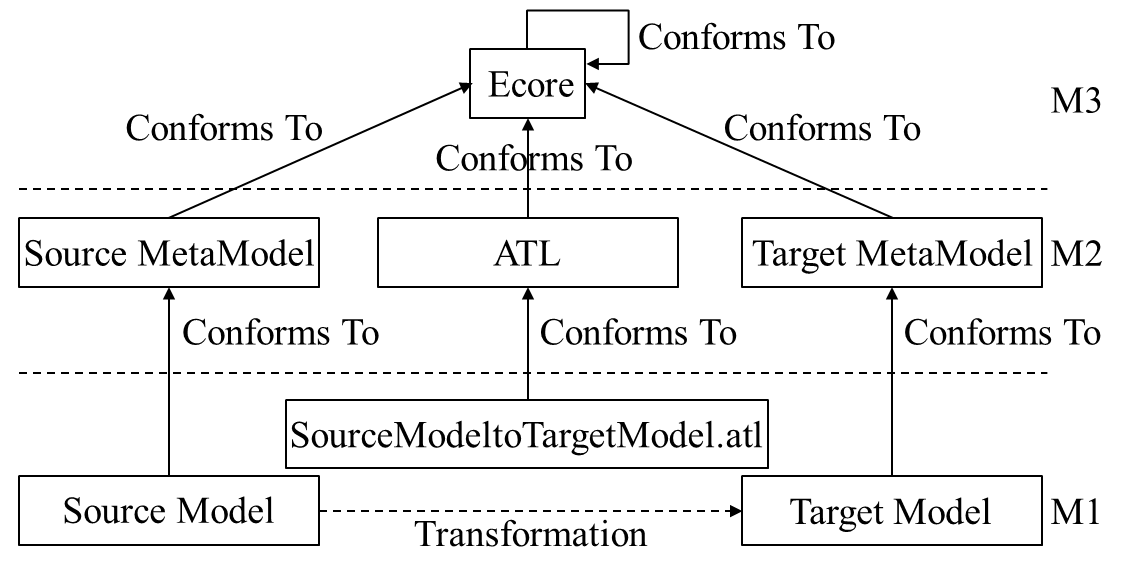


图2-8 ATL模型的转换框架

## 国内外研究现状

1. **适应性服务组装技术**

当前面向服务的架构已被广泛应用于业务环境中，用来构建异构系统，但是业务环境的动态特征会导致应用程序需求的变化，因此，服务组装必须要具有通过改变自身结构适应环境或者需求的变化的能力，具有这种能力的服务组装我们称之为适应性的服务组装。在适应性服务组装领域出现了众多的研究成果。

Mazzara和Dragoni[[[37]](#endnote-38)]等人提出一种通过工作流的重配置实现适应性服务组装的方法。由于WS-BPEL只是一种业务流程描述语言，设计人员不能在运行时动态地改变业务流程，因此作者设计了WS-BPEL的重构框架，并使用BPMN（Business Process Model and Notation）作为业务流程的图形化表示，随后根据预定义的转换规则将BPMN转换为WS-BPEL，并在WS-BPEL重构框架协调下，完成工作流的重新配置。但是该方法仍然存缺陷，比如该方法得到的新的工作流的配置仍然是静态的，不支持运行时工作流的重新配置。

Cherif和Djemaa等人通过对WS-BPEL扩展实现了SABPEL[[[38]](#endnote-39)]，SABPEL允许业务流程在运行环境下快速对服务失效或者需求变化等情况做出响应。同时作者提出了基于SABPEL的重配置机制，首先预定义将会触发该机制的上下文事件，然后在SABPEL中预定义自适应策略，即触发重配置机制后要进行的操作，最后由引擎执行过程中，当突发事件触发自适应机制时，引擎会执行相应的自适应策略，并且不需要重新部署业务流程。

Krishnamurty和Natarajan等人提出了一种基于重配置框架AO4BPEL的适应性服务组装方法[[[39]](#endnote-40)]，该方法使用AOP技术在运行时动态地改变WS-BPEL流程。AO4BPEL框架借助了面向Aspect的编程扩展了标准的WS-BPEL框架，WS-BPEL的执行引擎在运行时可以通过动态代理机制为程序添加新的功能。同时，该方法对AO4BPEL框架进行了扩展，使用Apache Jmeter监控引擎的运行过程，当WS-BPEL业务流程的服务失效或者某个服务需要替换时，该方法能够对候选服务进行QoS评分，选择最佳的服务绑定到业务流程中。

Wang和Chen等人提出了一种基于多代理的强化学习适应性服务组装模型TMG-WSC（Team Markov Games-Web Service Composition）[[[40]](#endnote-41)]，该模型适用于高度动态环境下的大规模服务组装软件。作者所提出的适应性服务组装模型集成了强化学习策略和马尔可夫决策过程MDP，其中强化学习策略能够使服务组装软件能够适应高度动态环境，该策略能够提高在面对大规模的候选服务时的对服务的选择效率，MDP能够使得多个代理为一个共同任务工作。同时，作者还对单代理的强化学习算法Sarsa进行了扩展得到多代理的强化学习算法Multi-Sarsa，通过该算法可以找到最优的服务组合方案。

Ma和Zhang等人提出了一种适应性服务组装框架[[[41]](#endnote-42)]，该框架利用多代理和JADE，实现了该框架的主要功能，包括执行引擎、监控器、分析器和计划器。作者使用上下文感知的Web服务组合语言CA-CCML来描述服务组装，执行引擎将会解析CA-CCML文件，根据解析到的功能信息从UDDI服务注册中心寻找满足需求的候选服务，同时引擎会根据最优选取策略从候选的服务中选择最佳的服务序列来执行。在引擎执行的同时，监控器将会监控用户的非功能需求和环境的改变，当监控到的信息发生改变时，监控器将监控的信息传递给分析器，随后分析器将会判断改变后的信息是否在预定义的范围内，如果改变后的信息在预定义的范围，分析器会把分析结果传递给计划器，计划器会通知引擎暂停执行过程，引擎会通过日志文件记录已经执行过的服务，计划器从候选服务中选择包含已经执行过的服务的最优服务序列，随后通知引擎重启执行更新后的服务序列。

Ezenwoye和Sadjadi等人首先提出一种更加健壮的业务流程设计语言RobustBPEL[[[42]](#endnote-43)]，通过在BPEL的业务流程中添加监控流程片段，当流程的执行过程中发现了失效服务时，可以通过代理服务进行替换，增强了服务组装的容错性和适应性。但是由于这种代理仍然是静态的去绑定有限数量的Web服务，随后作者对RobustBPEL进行了扩展提出了RobustBPEL-2[[[43]](#endnote-44)]，通过设计监控器监控这些预先被标记过的服务，当监控到服务调用超时等异常时，则启用动态代理机制，该机制将会根据监控到的上下文信息去动态的寻找服务去替换失效服务。

1. **面向特征的领域分析技术和模型驱动的服务组装技术**

基于特征的建模方法可以提供产品家族产品特征的简洁表示，特征模型的结构良好，通过识别特征之间的关系，用特征模型建模整个家族的变化性以支持对软件产品外在行为的变化性建模。FODA方法是最早提出的领域建模技术[32]，该方法是Kang等人在1990年提出的一种面向特征的领域分析方法，该方法支持某领域中系统共性和可变性需求的发现、分析和文档记录。领域分析包括三个基本阶段，场景分析：定义进行分析的领域的边界；领域建模：提供一种对领域问题域的描述；架构建模：构建一个可以将解决方法实施到领域具体问题的软件体系结构。FODA定义的特征模型由特征图和必要的附加信息组成，模型中的特征图以树形结构组织特征，该方法将特征分为两类：功能特征和非功能特征，通过特征间的组合关系共同构成特征树，高层次的特征由低层次的特征复合而成，附加信息包括特征说明、组合规则、问题等。

Kang等人通过扩展和完善FODA方法提出了FORM（Feature Oriented Reuse Method）[[[44]](#endnote-45)]。FODA方法关注的重点是识别共性需求、差异性需求以及创建特征模型的过程，然而该方法缺少对非功能性需求的考虑，并且缺少系统化的方法指导如何使用特征模型创建新的领域产品。而FORM在FODA的基础上，扩展了特征模型，使特征模型也能够表达非功能性需求。面向特征的领域分析方法发展至今，很多学者均是在FODA和FORM方法的基础上进行了研究和扩展。

Zhang和Jarzabek [[[45]](#endnote-46)]提出一种基于XVCL的软件产品线开发方法，该方法首先通过领域分析捕获软件产品领域中的公共需求和可变性需求创建特征图，随后通过设计软件产品线的架构，同时专门设计了一种文本标记语言XVCL，用于在核心软件制品和修改中直接标注出特征模型和这些可复用制品间的追踪关系。作者所提出的这种机制能够有效处理软件产品线中的变化。

国内对面向特征的领域分析技术研究起步较晚，北京大学的张伟、梅宏等人在已有特征模型的基础上提出的了一种新的面向特征的领域模型FODM[[[46]](#endnote-47)]，从功能、行为特点等方面深入分析了不同类型的特征，进一步研究不同类型特征之间的关系，从而获得了一种特征模型的具体形式。同时，FODM也对模型的组织框架及剪裁机制进行了进一步的抽象，给出了其对应的统一抽象描述。

1. **模型驱动的服务组装技术**

UML是广泛使用的系统建模语言，利用UML提供的建模机制可以进行系统的可变性建模。He和Fu等人提出一种基于可变性管理的模型驱动服务组装方法[[[47]](#endnote-48)]，该方法对UML扩展实现了VxUML，随后使用VxUML对体系结构进行建模，该模型能够从体系结构层表示服务组装的可变性问题，是一种更高层次的建模，然后通过定义一系列转换规则，将VxUML转化成VxBPEL模型。在VxUML转化成VxBPEL模型的过程中，在VxUML定义的可变性元素将会被映射为VxBPEL模型变异点和变体。

邢岩和谷放等人提出一种特征模型驱动的Web服务组装方案FWSC [[[48]](#endnote-49)]，使用特征模型指导Web服务组装过程，同时开发相应的支持平台BiFrost。该方案首先通过需求分析建立特征模型，随后对UDDI Web服务库扩展实现了特征模型库，利用特征模型和特征模型库维护需求模型和运行时刻模型之间的映射关系，该方法可以有效的提高Web服务组装系统的建模、开发速度和质量，增强了Web服务组装系统在需求发生变化时的动态调整和演化能力。

## 课题研究背景

当前，随着基于SOA架构的分布式应用越来越广泛，基于WS-BPEL的服务组装广泛应用于企业内异构系统间的集成[[[49]](#endnote-50)]。为应对分布式环境下多变的环境和需求，课题组在前期的工作中提出一种基于可变性管理的适应性服务组装方法，通过扩展WS-BPEL开发了适应性服务组装语言VxBPEL，使用VxBPEL能够在服务组装的规格说明中预定义变异点、变体以及相关的业务流程配置方案，同时扩展Apache ODE引擎以支持VxBPEL语言的解释与执行，实现运行时的可变性管理，并且将适应性服务组装应用到SaaS软件的开发，使其满足第四级成熟度。然而，当前的服务组装过程没有明确区分需求分析和流程设计，使得需求分析和流程设计糅杂在一起，并只是在设计层次进行了可变性设计，缺少对需求层次可变性的考虑。

本文探索研究一种全生命周期的适应性服务组装方法，在该方法中引入领域分析和模型驱动，以领域分析得到的特征模型作为需求模型，实现需求层次与流程设计层次的分离，同时以特征模型驱动业务流程的设计以及后续的服务组装系统的开发过程，并开发了特征模型建模工具SCFeatureModel、特征模型和服务组装管理工具FM2VxBPEL。

## 主要研究成果

本文的研究成果和创新点如下：

1. **一种面向全生命周期的适应性服务组装方法：**在服务组装的需求分析阶段引入领域分析的特征模型，并通过一系列转换规则实现特征模型到抽象服务组装模型的转换，将需求分析和流程设计阶段分离开来，以特征模型驱动后续的开发过程，增强了系统的正确性、可靠性和可扩展性。
2. **支持特征模型的建模工具SCFeatureModel：**基于Eclipse模型框架EMF开发了特征模型建模工具SCFeatureModel，该工具以特征模型的Ecore模型为基础，该工具能够以插件的形式运行于Eclipse平台。
3. **特征模型和服务组装管理工具FM2VxBPEL：**该工具支持领域特征模型图的创建、标准VxBPEL语言的转换以及用户配置方案的创建和验证，同时为流程部署、运行提供自动化支持，大幅度提高服务组装系统开发的效率。

## 论文的组织结构

论文的组织结构安排如下：

第一章，简述本文要解决的问题和研究方向。

第二章，介绍课题相关概念和关键技术、国内外研究现状和课题研究背景。

第三章，提出一种面向全生命周期的适应性服务组装方法。

第四章，讨论特征模型建模工具SCFeatureModel的设计与实现。

第五章，讨论特征模型到抽象服务组装模型的转换的具体实现。

第六章，讨论特征模型和服务组装管理工具FM2VxBPEL的设计与实现。

第七章，讨论实例研究的设计、过程和结果，评估所提出的方法和开发的支持工具的有效性。

第八章，总结全文，讨论存在的不足。

# 特征模型驱动的适应性服务组装方法

首先介绍特征模型驱动的适应性服务组装方法框架，随后介绍方法框架中涉及到的特征与特征模型、抽象服务组装模型，最后讨论特征模型与抽象服务组装模型转换的一般过程。

## 特征模型驱动的适应性服务组装方法框架

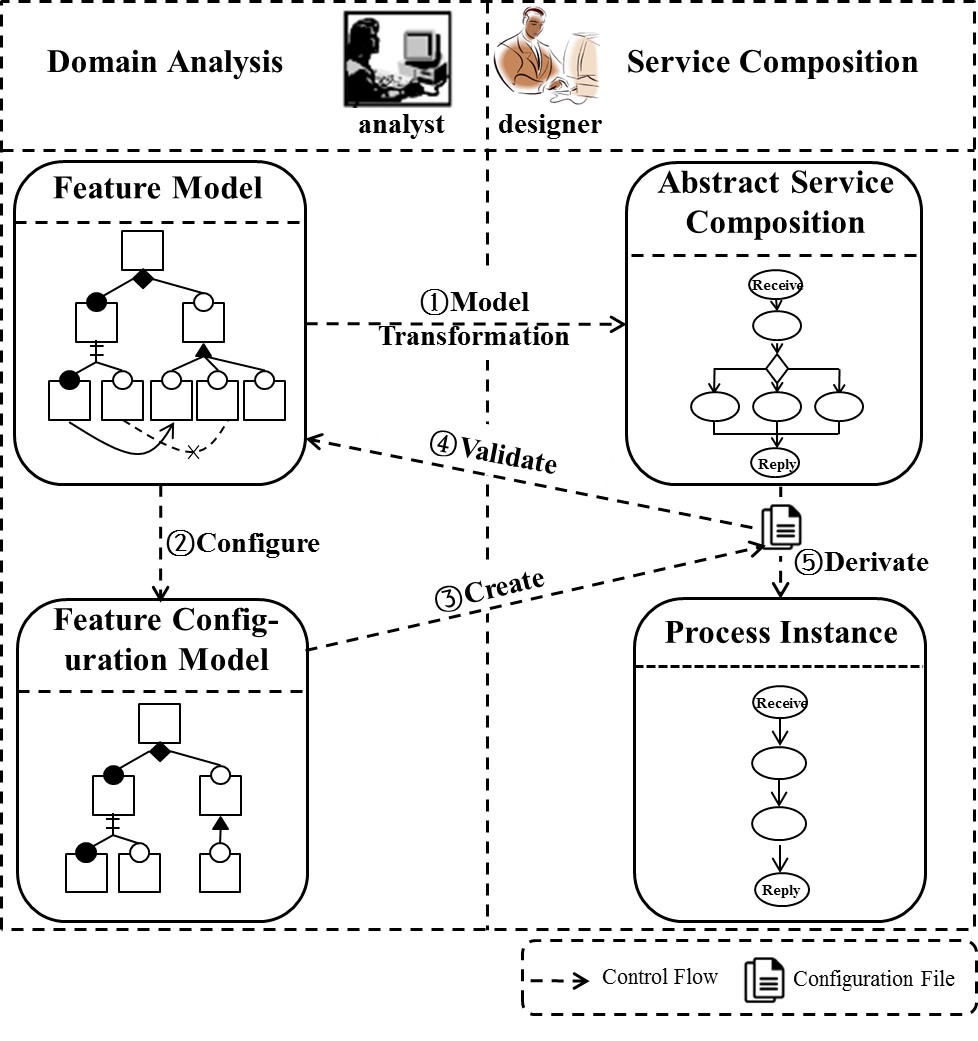


图3-1特征模型驱动的适应性服务组装方法框架

传统的服务组装开发过程中，流程设计和需求分析糅杂在一起，并且只在设计层次引入可变性设计，缺少对需求层次可变性的考虑，为此本文提出了如图3-1所示的方法框架：

1. 在服务组装开发过程的前期引入了领域分析，需求分析人员根据领域知识，对需求的问题空间进行划分，提取领域内的共性需求和可变性需求。随后需求分析人员可以利用特征模型建模工具SCFeatureModel进行需求模型的建模，得到领域的特征模型实例，该模型实例以XMI形式进行序列化，因此对特征模型实例解析所得到的特征模型图是一个类似于树的层次结构。此时得到的特征模型实例仍然是一个静态的、抽象的需求表示，无法与抽象服务组装模型建立联系，为此本方法定义了一系列的特征模型到抽象服务组装模型的转换规则，通过模型转换机制（ATL）得到抽象服务组装模型。该抽象服务组装模型只表达一个简单的业务流程结构，是一个不完整的VxBPEL规格说明，因此通过课题组前期开发的VxBPEL Designer[[[50]](#endnote-51)]以已有简单的业务流程结构为基础，进行完整的业务流程的设计，得到完整的VxBPEL业务流程。
2. 在进行特征模型到抽象服务组装模型转换的同时，需求分析人员根据用户的需求对领域的特征模型进行定制，即用户在满足特征模型约束的条件下选择相应的特征后，需求分析人员使用本方法所开发的工具FM2VxBPEL工具对特征模型进行配置生成特征配置模型，特征配置模型表达的单个用户的需求，并且仍以XMI进行特征配置模型的序列化。
3. 用户的特征配置模型创建完成后，需求分析人员继续使用FM2VxBPEL对特征配置模型进行解析，并根据解析结果创建用户配置文件，用户配置文件中保存了对抽象服务组装模型中变异点和变体的选择。
4. 用户配置文件创建完成后，为了进一步保证用户需求的正确性，设计人员利用FM2VxBPEL对用户配置文件进行验证，主要验证用户的需求是否符合特征模型的约束关系。
5. 在得到正确的用户配置文件后，执行引擎VxBPEL ODE将会根据用户配置文件派生出具体的流程实例。

## 特征与特征模型

1. **特征**

“特征”一词最早出现在电信领域，用来作为描述电信系统的基本功能单元[[[51]](#endnote-52)]。而在软件工程领域中，从不同的视角来看对特征有着不同的定义，Davis等人将特征定义为系统需求规约的组织方式[[[52]](#endnote-53)]，Zhang和Mei等人则认为特征是具有用户或者客户价值的软件特点[[[53]](#endnote-54)]，在本文中，对于特征的定义沿用Kang等人在FODA[27]方法中对“特征”的定义，特征是系统中用户可见的、显著的或具有特色的方面、品质和特点。

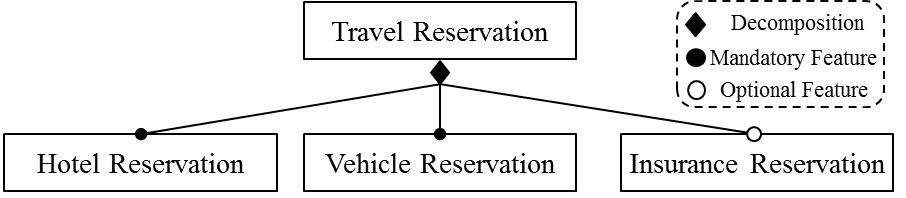


图3-2 旅行预订系统领域具有的特征

如图3-2所示，根据FODA方法中特征的定义，对旅行预订系统领域（Travel Reservation）进行领域需求分析，该系统旨在为用户提供旅行过程中用到的一系列服务，从用户的角度来看，该系统所具有的特征主要包括酒店预订（Hotel Reservation）、交通工具预订（Vehicle Reservation）、保险预订（Insurance Reservation）。为了标识领域的共性需求和可变性需求，将特征分为强制特征（Mandatory Feature）和可选特征（Optional Feature），定义如下：

1. 强制特征：强制特征也称为必选特征、共性特征，它代表了领域的共性，构成了领域内软件制品基础，换言之领域内的所有产品都要具有这些特征。在如图3-2所示的旅行预订系统中，酒店预订和交通工具预订这两个服务是用户必要的服务，缺一不可，因此这两个特征是该领域的强制特征。
2. 可选特征：可选特征也称为可变性特征，它代表了领域的变化性，使领域内软件制品各具特色，功能或者特点各不相同。如图3-2所示，在旅行预订系统领域，用户可以根据自己的需求选择是否需要该系统提供保险服务，因此该特征是领域的可选特征。
3. **特征模型**

特征模型是对领域内一系列软件产品的共性、变化性需求的抽象表示[[[54]](#endnote-55)]，它使用特征作为描述问题空间的一阶实体，由一组特征以及这组特征之间的关系组成。为了能够更好的描述特征模型，首先要建立特征模型的元模型，以后特征模型的建立都以元模型为准则，为以后的建模提供一个统一的标准，避免不同特征模型之间发生歧义。元模型顾名思义，就是模型的模型。传统的特征模型的元模型对描述面向服务的领域分析的需求模型不够充分，因此本文对特征模型的元模型进行了重新的定义。

特征模型元模型的UML表示如图3-3所示，特征模型（Feature Model）由特征（Feature）以及特征之间的关系（Relation）两部分组成，因此二者与特征模型之间是强聚合；前面已经提到，特征分为两种，因此特征泛化为强制特征（Mandatory Feature）和可选特征（Optionality Feature）；关系泛化为特征之间的约束关系（Constraint）和不同层次特征之间的精化关系（Refinement）；约束关系是在对特征模型中的特征之间的关系进行明确表达，其主要包括依赖约束（Require）和排斥约束（Exclude），这两种约束的定义如表3-1所示，由于两个特征之间才会约束关系，因此在特征和约束的双向关联中，特征与约束是2对1的关系；精化关系将不同抽象层次的特征组织成类树状结构（树状结构中增加了约束关系，因此是类树状结构），主要包括分解关系（Decomposition）、刻画关系（Characterization）、特化关系（Specialization），这三种精化关系的定义如

表3-2所示，为了便于读者理解，本文将更高抽象层次的特征称为父特征，精化后得到的特征称为子特征，不同的精化关系导致的父子特征在特征模型中扮演的角色也不同，具体角色分类如表3-2所示。

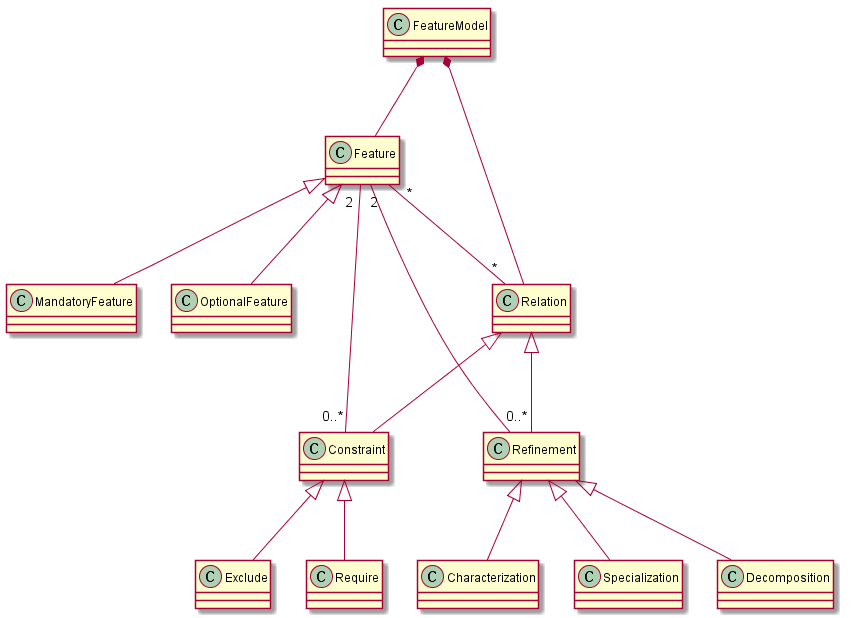


图3-3 特征模型元模型的UML类图

表3-1 约束关系(Constraint)的分类与定义

|  |  |
| --- | --- |
| **约束(Constraint)** | **定义** |
| 依赖(Require) | 若特征X依赖于特征Y，记作“X→Y”，其表示选择了特征X，必须要选择特征Y，但是选择了特征Y并不一定要选择特征X |
| 排斥(Exclude) | 若特征X和特征Y存在排斥关系，记作“X Y”，其表示特征X和特征Y只能存在一个 |

表3-2 精化关系(Refinement)的分类和角色分类

|  |  |  |  |
| --- | --- | --- | --- |
| **精化(Refinement)** | **定义** | **父特征角色** | **子特征角色** |
| 分解(Decomposition) | 一个特征能被分解成一组特征 | 整体 | 部分 |

表3-2 精化关系(Refinement)的分类和角色分类（续）

|  |  |  |  |
| --- | --- | --- | --- |
| **精化(Refinement)** | **定义** | **父特征角色** | **子特征角色** |
| 刻画(Characterization) | 通过识别特征的属性进而刻画成抽象层次更低的特征 | 实体 | 属性 |
| 特化(Specialization) | 将一个一般特征进一步精化为更具体的特征 | 一般实体 | 具体实体 |

为了让建模人员更好的理解特征模型，减少建模人员和设计人员的错误沟通，同时保证特征模型到抽象服务组装模型映射时的一致性和正确性，该方法对特征模型的建立定义了如表3-3所示约束：

表3-3 特征模型建立约束表

|  |  |
| --- | --- |
| **序号** | **描述** |
| ① | 最顶层特征（抽象层次最高的特征）一定是必选特征 |
| ② | 每个特征和其子特征之间有且只能有一种精化关系 |
| ③ | 如果父特征和子特征之间是分解关系，则子特征至少有一个是必选特征 |
| ④ | 如果父特征和子特征之间是刻画关系，则子特征可以是必选特征，也可以是可选特征 |
| ⑤ | 如果父特征和子特征之间是特化关系，则子特征全部为可选特征 |
| ⑥ | 如果父特征和子特征之间是刻画关系，则子特征与该子特征的子特征的精化关系可以为刻画关系或者特化关系，但不能是分解关系 |
| ⑦ | 如果父特征和子特征之间是特化关系，则子特征不可再进行任何关系的精化 |
| ⑧ | 依赖约束和排斥约束只能存在于可选特征之间，不能存在于必选特征和可选特征或者必选特征和必选特征之间 |

遵守如表3-3的约束所构造出来的领域特征模型结构清晰、逻辑正确，可以排除掉大部分特征之间错误的结构关系，同时减少复杂的特征模型存在多层次精化关系这种情况的出现。

举例来说，如图3-4所示，在某个领域，A、B、C、D是组成Root特征的基础，缺一不可，因此这四个特征是强制特征，构成了领域软件的基础，而E特征作为变化性特征，代表了领域软件功能的延伸，由于E特征依赖于用户的选择而存在，因此该特征是可选特征。B特征作为一个一般的实体特征，其可以通过B1或者B2两个具体的特征来实现，因此B1、B2与B特征之间是特化关系。D特征作为一个实体特征，从其属性角度分析，其又可以刻画为D1、D2、D3这三个特征，其中D1是强制特征，而D2和D3属于可选特征。同时，我们对该领域做出如下假设：

① 如果选择由B1特征实现B特征，则必须选择D2特征，因此B1特征和D2特征存在依赖约束关系；

② 如果选择B2特征实现B特征，则不能选择D3特征，即B2特征和D3特征不能同时存在。

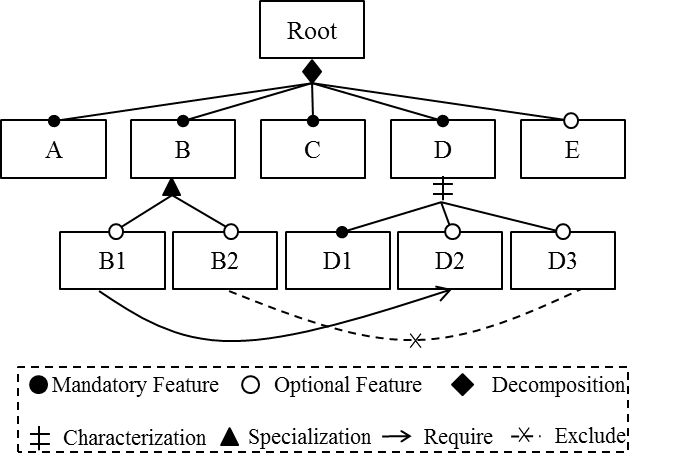


图3-4 模拟某领域的特征模型

1. **基于巴克斯范式的特征模型描述**

本文采用巴克斯范式（Backus-Naur Form，BNF）描述特征模型。巴克斯范式最早是由Backus和Naur提出的使用形式化符号来描述给定语言的语法，由于其逻辑比较严谨，并且使用一维的字符串表示，从而避免了通常在描述复杂的语法关系时所常用的二维图示法，因此被广泛应用。表3-4解释了巴克斯范式指定的形式化常用符号的含义。

表3-4 巴克斯范式形式化符号和符号含义

|  |  |
| --- | --- |
| **BNF符号** | **含义** |
| ::= | 该符号是“被定义为”的意思 |
| <> | 尖括号内包含的为必选项 |
| [] | 方括号内包含的为可选项 |
| {} | 花括号内包含的为可重复0至无数次的选项 |
| | | 竖线表示其左右两边任选一项，相当于逻辑运算“OR”的意思 |
| “” | 双引号中的字符，代表字符本身 |
| ““”” | 双重引号代表双引号本身 |

为了更好的描述特征模型，引入了根特征和叶子特征两个概念，其定义如表3-5所示：

表3-5 根特征与叶子特征定义

|  |  |
| --- | --- |
| **概念** | **定义** |
| 根特征 | 特征模型最顶层的特征（即抽象层次最高的特征） |
| 叶子特征 | 若某特征下再无任何精化关系，则该特征即为叶子特征 |

特征模型有ID、Name两个属性，由根特征RootFeature、约束Constraints和一个可选的描述信息Description组成；对于根特征，其有ID、Name、IsLeafFeature和FeatureOptional四个属性，IsLeafFeature用来标识该特征是否为叶子特征，FeatureOptional标识特征的可选性，由组Group和可选的Description组成，其中Group表示某个特征下的子特征及其子特征下的所有特征的集合；对于组Group，由ID、Num、RefinetmentType三个属性，Num表示用来标识当前Group下包含Group的个数，如果当前Group下的特征全部为叶子特征，则Num的值为-1；RefinetmentType标识该组与上一层特征的精化类型，由特征Feature和可选的Description组成；Feature包括ID、Name、IsLeafFeature、FeatureOptional四个属性，由Group和可选的Description组成。

根据以上描述，基于巴克斯范式的特征模型描述如图3-5所示：

FeatureModel ::= <ID><Name><RootFeature><Constraints>[Description]

RootFeature ::= {Group}<ID><Name>< IsLeafFeature ><FeatureOptional>[Description]

Group ::= <Feature><ID><Num><RefinetmentType>[Description]

Feature ::= {Group}<ID><Name><IsLeafFeature><FeatureOptional>[Description]

IsLeafFeature ::= True|False

True ::= "true"

False ::= "false"

FeatureOptional ::= Mandatory| Optionality

Mandatory::= "Mandatory"

Optionality::= "Optionality"

Constraints ::= <ID><ConstraintType><XFID><YFID>

ConstraintType ::= Require|Exclude

Require ::= "Require"

Exclude::= "Exclude"

XFID ::= <ID>

YFID ::= <ID>

Description ::= <Text>

Text ::= {a|…|z|A|…|Z}

ID ::= {a|…|z|A|…|Z}

Num ::= < -1|0|1|2|…|N>

图3-5 特征模型巴克斯范式

## 抽象服务组装模型

课题组前期提出了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[15]，如图3-6所示。首先，将服务组装看作底层服务集合的一种抽象，引入了基于可变性设计的抽象服务组装模型，如图3-7(a)所示，即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。当抽象服务组装模型建模完成以后，在运行阶段执行引擎会根据不同用户需求（即用户配置文件）对抽象服务组装模型进行配置，派生（如图3-7(b)所示）出符合每个用户需求的具体流程实例（即A1=A2、B1=B2，但A1≠B1、A1≠C1，而且B1≠C1，如图3-7(c)所示），并且运行时流程实例多态共存、互不影响（即A1与A2并发执行、互不影响，B1与B2并发执行、互不影响，而且A1与A2和B1与B2互不影响）。

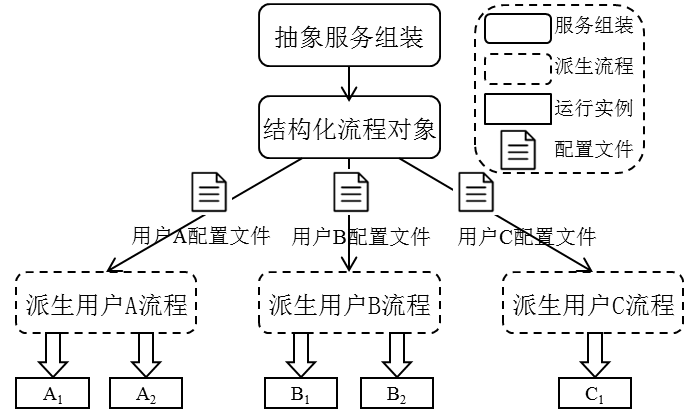


图3-6 基于可变性模型的服务组装建模方法

活动

变异点

舍弃

Receive

invokeA1

invokeA2

invokeA3

invokeB1

invokeB2

invokeB3

invokeC

Reply

Receive

invokeA1

invokeA2

invokeA3

invokeB1

invokeB3

invokeC

Reply

Reply

invokeC

invokeB3

invokeA2

Receive

invokeB2

(a)抽象服务组装模型

(b)派生过程

(c)具体业务流程

图3-7 抽象服务组装模型到具体业务流程的派生过程

抽象服务组装模型针对的是某个问题空间不同用户的共性需求和差异性需求。例如，如图3-2所示的旅行预订系统，其抽象服务组装模型如**错误!未找到引用源。**所示。交通工具预订（Vehicle）和酒店预订（Hotel）作为必选服务是共性需求，但是该系统为用户提供商务和经济两种类型的酒店，用户需要选择其一，而保险服务（Insurance）完全由用户根据自己的需求进行决定是否需要提供。根据以上描述，在该系统的抽象服务组装模型中，酒店预订作为共性需求抽象为变异点，两种类型的酒店作为具体的实现抽象为变体，并且此变异点下的两个变体是独选约束的关系（即H-Costly被选择而H-Cheap不被选择或者H-Cheap被选择而H-Costly不被选择），保险预订作为差异性需求抽象为变异点，该变异点下只有一个变体。

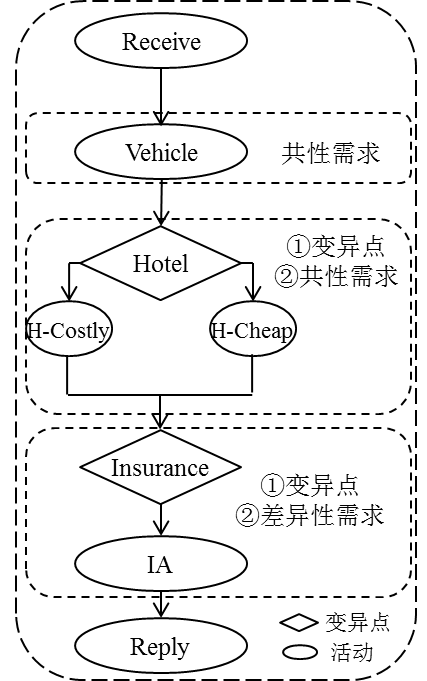


图3-8 旅行预订系统的抽象服务组装模型

## 特征模型到抽象服务组装模型的映射

为了能够更清晰的描述特征模型到抽象服务组装的映射过程，同样使用巴克斯范式描述VxBPEL。

1. **基于巴克斯范式的VxBPEL描述**

前期课题组扩展BPEL语言得到的VxBPEL语言中，增加了12个结构化活动，分别是VariationPoint、Variants、Variant、VariabilityConfigurationInformation、ConfigurableVariationPoints、ConfigurableVariationPoint、Rationale、VariantInfo、RequiredConfiguration、VPChoices、VPChoice、Constraint，每个活动的含义如下：

1. VariationPoint：代表变异点发生变化的位置。
2. Variants：VariationPoint的子元素，代表一组变体的集合，VariationPoint下可以有多个Variants。
3. Variant：Variants的子元素，代表某个变体，表示一种可被代替的动能或者操作，Variants可以有多个Variant，默认情况下，多个Variant之间存在替代约束。
4. VariabilityConfigurationInformation：代表不同变异点下的变体间特定业务需求而存在的约束关系，即可变性的配置信息。
5. ConfigurableVariationPoints：VariabilityConfigurationInformation的子元素。
6. ConfigurableVariationPoint：ConfigurableVariationPoints的子元素，代表多种配置方案的集合，通过id和defaultVariant两个属性唯一的标识该配置方案，defaultVariant其中用来表示在没有指明选择配置方案的情况下默认的配置方案的名称。
7. Rationale：ConfigurableVariationPoint的子元素，代表文本信息。
8. VariantInfo：Variant的子元素，代表对某种配置方法的文本描述。
9. RequiredConfiguration：Variant的子元素，代表配置方案。
10. VPChoices：RequiredConfiguration的子元素，代表多个变异点的选择
11. VPChoice：VPChoices的子元素，有vpname和variant两个属性，vpname的属性值为选择的变异点的name，variant的属性值为vpname标识的变异点下variant的name，每一个VPChoice代表一个变异点下变体的选择情况，VPChoices包含多个VPChoice。
12. Constraint：代表不同变异点下变体之间的约束关系[13]，有CType、varianta和varianta三个属性，其中CType是枚举类型，代表约束类型，变体之间的约束关系主要分为包含约束(Inclusion Constraint)、独占约束(Exclusive Constraint)、排除约束(Exclusion Constraint)、替代约束(Substitution Constraint)、并存约束(Corequisite Constraint)。

基于巴克斯范式的VxBPEL描述如图3-9所示：

VxBPEL ::=<Process>{WSDL}

Process ::=<Name of Process>{variable}{PartnerLink}{faultHandle}{eventHandle}{activity}{Constraint}

PartnerLink ::=<Name of PartnerLing>[myrole | partnerrole]

Myrole ::=<Name of role>

partnerrole ::=<Name of partnerrole>

variable ::=<Name of Variable><Type Name of Message>

activity ::=[elementary activity | structured activity]

elementary activity ::=<invoke | receive | reply | empty | assign | wait | throw>

structured activity ::=<sequence | flow | if | pick | while | VariationPoint |Variants | Variant | VariabilityConfigurationInformation | ConfigurableVariationPoints | ConfigurableVariationPoint | Rationale | VariantInfo | RequiredConfiguration | VPChoices | VPChoice | Constraint>

VariationPoint ::=<Variants>{activity}

Variants ::=<Variant>

Variant ::=[VariantInfo][RequiredConfiguration]{activity}

ariabilityConfigurationInformation ::=<ConfigurableVariationPoints>

ConfigurableVariationPoints ::=<ConfigurableVariationPoint>

ConfigurableVariationPoint ::=<Variants>[Rationale]

RequiredConfiguration ::=[VPChoices]

VPChoices ::={VPChoice}

VPChoice ::=<vpname><variant>

Constraints ::=<CType><varianta><variantb>

CType ::=<Inclusion | Exclusive | Exclusion | Substitution | Corequisite>

图3-9 VxBPEL的巴克斯范式

特征模型到抽象服务组装模型的转换算法如下所示：

|  |
| --- |
| **Algorithm 1.** 特征模型到抽象服务组装模型的转换算法 |
| **Input:** 特征模型FeatureModel  **Output:** 抽象服务组装模型VxBPEL  **PROCEDURE**：   1. 分析FeatureModel，读取特征模型根特征信息，建立VxBPEL的process节点，并在process节点下添加sequence节点， 2. 判断根特征是否为叶子特征，若根特征的为叶子特征，则在sequence节点下添加invoke活动，且模型转换结束；若根特征非叶子特征，继续执行； 3. 读取当前特征下的每一组Group，并对每一组Group进行处理，如果当前Group下的子特征全部为叶子特征，且Group下的子特征与其父特征是分解关系或者刻画关系，对于强制类型的子特征，则在当前的节点下添加invoke节点，对于可选类型的子特征，则对每个子特征进行可变性设计，即在当前的节点下添加VariationPoint、Variants、Variant、invoke节点；如果当前Group下的子特征全部为叶子特征，且Group下的子特征与其父特征是特化关系，则对Group进行可变性设计，Group下的每个特征都映射为一个变体，该Group处理完毕；如果当前Group下的子特征非叶子特征，对于强制类型的非叶子特征，在当前的节点下添加sequence节点，重复执行步骤3，对于可选类型的非叶子特征进行可变性设计，在当前的节点下添加VariationPoint节点，重复执行步骤3，当FeatureModel下的所有Group处理完，模型转换结束。 |

## 小结

本章主要介绍了面向全生命周期的适应性服务组装方法的框架，首先介绍了特征和特征模型，并对特征模型进行了形式化的定义，随后介绍了抽象服务组装模型和VxBPEL的形式化定义，最后提出了特征模型到抽象服务组装模型的转换算法。

# 特征模型建模工具SCFeatureModel的设计与实现

本章主要介绍服务组装领域特征模型建模工具SCFeatureModel的设计与实现，包括特征模型的Ecore模型的详细介绍、工具的开发过程和工具的使用说明。

## 特征模型Ecore模型

在很多软件领域通常使用UML来进行建模，虽然UML在可视化建模领域作用巨大，但是其客观存在一定的局限性，这些局限性主要表现在：1.UML模型不能很好的满足对于工程领域的建模需求；2.建立的UML模型不可执行，不利于需求分析和模型验证。

因此本课题选择使用EMF进行特征模型建模工具的开发，首先创建特征模型的元模型，即创建特征模型的Ecore模型（如图4-1所示）。

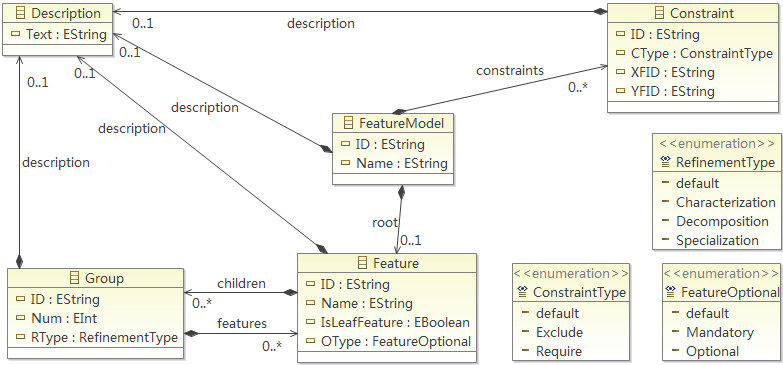


图4-1 特征模型元模型

特征模型Ecore模型涉及到的EClass主要包括FeatureModel、Feature、Group、Description、Constraint，涉及到的EEnum包括FeatureOptional、RefinementType、ConstraintType，表4-1到表4-8是特征模型的Ecore模型的详细介绍。

表4-1 FeatureModel EClass

|  |  |  |
| --- | --- | --- |
| **FeatureModel** | **类型** | **描述** |
| FeatureModel | EClass | 名为FeatureModel的建模类，是对某个领域的高度抽象描述 |
| ID | EAttribute | 名为ID的模型属性，唯一标识FeatureModel |
| Name | EAttribute | 名为Name的模型属性，用来表示领域的名称 |

表4-1 FeatureModel EClass（续）

|  |  |  |
| --- | --- | --- |
| **FeatureModel** | **类型** | **描述** |
| description | Aggregation | FeatureModel和Description建模类之间是聚合关系，表示一个FeatureModel包含0个或者1个Description，用description标识 |
| root | Aggregation | FeatureModel和Feature建模类之间是聚合关系，表示一个FeatureModel包含0个或1个Feature，用root标识 |
| constraint | Aggregation | FeatureModel和Constraint建模类之间是聚合关系，表示一个FeatureModel包含0个或者多个Constraint，用constraints标识 |

表4-2 Feature EClass

|  |  |  |
| --- | --- | --- |
| **Feature** | **类型** | **描述** |
| Feature | EClass | 名为Feature的建模类，领域内某个特征的抽象描述 |
| ID | EAttribute | 名为ID的模型属性，唯一标识Feature |
| Name | EAttribute | 名为Name的模型属性，用来表示某个特征名称 |
| IsLeafFeature | EAttribute | 名为IsLeafFeature的模型属性，用来标识该特征是否为叶子特征 |
| OType | EEnum | 名为OType的模型属性，用来标识该特征是强制特征或者可选特征 |
| description | Aggregation | Feature和Description建模类之间是聚合关系，表示一个Feature包含0个或1个Description，用description标识 |
| children | Aggregation | Feature和Group建模类之间是聚合关系，表示一个Feature包含0个或者1个Group，用children标识 |

表4-3 Group EClass

|  |  |  |
| --- | --- | --- |
| **Group** | **类型** | **描述** |
| Group | EClass | 名为Group的建模类，表达特征模型中多层次关系 |
| ID | EAttribute | 名为ID的模型属性，唯一标识Group |
| Num | EAttribute | 名为Num的模型属性，如果该Group下的子特征全为叶子特征，则Num值为-1，否则Num值为Group下包含Group的个数 |

表4-3 Group EClass（续）

|  |  |  |
| --- | --- | --- |
| **Group** | **类型** | **描述** |
| RType | EEnum | 名为RType的模型属性，用来标识该Group下的特征与其父特征的精化关系 |
| description | Aggregation | Feature和Description建模类之间是聚合关系，表示一个Feature包含0个或1个Description，用description标识 |
| children | Aggregation | Feature和Group建模类之间是聚合关系，表示一个Feature至少包括一个Group，用children标识 |

表4-4 Constraint EClass

|  |  |  |
| --- | --- | --- |
| **Constraint** | **类型** | **描述** |
| Constraint | EClass | 名为Constraint的建模类，表达特征模型中约束关系 |
| ID | EAttribute | 名为ID的模型属性，唯一标识Constraint |
| CType | EAttribute | 名为CType的模型属性，标识约束类型 |
| XFID、YFID | EAttribute | 名为XFID和YFID的模型属性，标识约束的两方 |

表4-5 Description EClass

|  |  |  |
| --- | --- | --- |
| **Description** | **类型** | **描述** |
| Description | EClass | 名为Description的建模类，表达描述信息 |
| Text | EAttribute | 名为Text的模型属性，表达描述信息文本内容 |

表4-6 Refinement EEnum

|  |  |  |
| --- | --- | --- |
| **Refinement** | **类型** | **描述** |
| Refinement | EEnum | 名为Refinement的枚举类，用来标识三种精化关系 |
| Decomposition | Eenumliteral | 名为Decomposition的枚举值，标识分解关系 |
| Characterization | Eenumliteral | 名为Characterization的枚举值，标识刻画关系 |
| Specialization | Eenumliteral | 名为Specialization的枚举值，标识特化关系 |

表4-7 FeatureOptional EEnum

|  |  |  |
| --- | --- | --- |
| **FeatureOptional** | **类型** | **描述** |
| FeatureOptional | EEnum | 名为FeatureOptional的枚举类，用来标识特征的可选性 |
| Mandatory | Eenumliteral | 名为Mandatory的枚举值，标识强制特征 |
| Optionality | Eenumliteral | 名为Optionality的枚举值，标识可选特征 |

表4-8 ConstraintType EEnum

|  |  |  |
| --- | --- | --- |
| **ConstraintType** | **类型** | **描述** |
| ConstraintType | EEnum | 名为ConstraintType的枚举类，用来标识特征之间的约束关系 |
| Exclude | Eenumliteral | 名为Exclude的枚举值，标识排斥约束 |
| Require | Eenumliteral | 名为Require的枚举值，标识依赖约束 |

## SCFeatureModel的需求分析

SCFeatureModel核心功能是特征模型的建模，其用例图如图4-2所示。

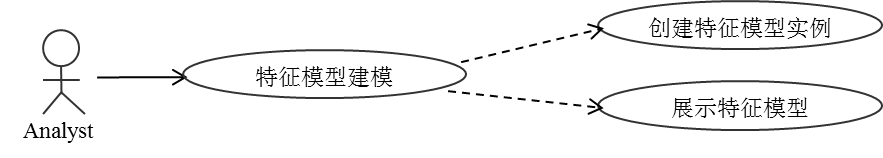


图4-2 SCFeatureModel工具的用例图

对与特征模型建模这个用例来说，领域需求分析人员能够对某个领域进行需求分析，并且使用该工具创建出符合特征模型定义的特征模型实例，该特征模型实例是以XMI文件的形式存在的，需求分析人员能够使用该工具以XMI的形式展示特征模型。

## 特征模型建模工具的实现

前面我们已经简单介绍了EMF，由于EMF作为插件已经集成到了Eclipse当中，因此我们可以直接使用Eclipse进行EMF项目的开发，本次开发环境使用的是Eclipse3.7.0。

### 特征模型Ecore模型的创建

特征模型Ecore模型的创建主要包括三个步骤，分别是EMF项目的创建、Ecore模型文件的创建和Ecore模型文件的编辑。

1. **EMF项目的创建**

打开Eclipse，使用“New>Other”打开向导窗口，然后在向导窗口中打开“Eclipse Model Frameword”节点，选择“Empty EMF Project”，点击“Next”，输入工程名“SCFeatureModel”完成EMF项目的创建。

1. **Ecore模型文件的创建**

在SCFeatureModel项目中，包含有一个名为“model”的文件夹，在此文件夹上右击选择“New>Other”打开向导窗口，然后在向导窗口中打开“Eclipse Model Frameword”节点，选择“Ecore Model”，点击“Next”，输入文件名“SCFeatureModel”完成Ecore模型的创建（文件扩展名为ecore）

1. **Ecore模型文件的编辑**

右击该Ecore模型文件，从弹出菜单中选择“Open With>Sample Ecore Model Editor”，使用模型编辑器Model Editor打开Ecore模型，随后即可通过编辑特征模型结构树视图完成特征模型文件的创建。由于特征模型结构树的直观性比较差，可以对Eclipse安装Ecore Tools插件，在“SCFeatureModel.ecore”文件右键单击选择“Initialize Ecore Diagram File”生成特征模型元模型文件的图形化文件“SCFeatureModel.ecorediag”，并且Ecore Tools工具为这个文件提供了可视化编辑器，对“SCFeatureModel.ecorediag”的修改会同步到“SCFeatureModel.ecore”文件中，最终得到的特征模型的元模型文件“SCFeatureModel.ecore”，如图4-3所示。在特征模型的Ecore创建完成以后，右击Ecore模型文件，可以从弹出菜单中选择“Open With>Text Editor”查看Ecore模型文件XMI格式。SCFeatureModel.ecorediag文件打开后如图4-1所示。

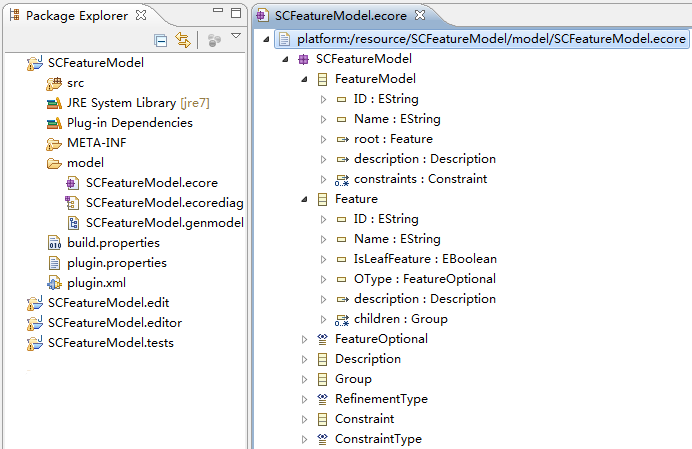


图4-3 特征模型Ecore模型

### 特征模型建模工具插件的生成

特征模型建模工具插件的生成主要包括两个步骤，生成器模型的创建和插件代码的生成。

1. **生成器模型的创建**

为了能够将特征模型建模工具以插件的形式运行在Eclipse中，我们需要通过特征模型的Ecore模型创建生成器模型（文件扩展名为genmodel）。首先在“SCFeatureModel.ecore”文件下右击，选择“New>Other”打开向导窗口，然后在向导窗口中打开“Eclipse Model Framework”节点，选择EMF Generator Model，点击“Next”，其默认文件名与“SCFeatureModel.ecore”文件名相同，点击“Next”，指定所需要的模型导入器为“Ecore model”，点击“Next”，指定Ecore模型的URI，然后选择“Load”按钮完成“SCFeatureModel.ecore”模型的加载，最后选择“Finish”按钮完成生成器模型文件SCFeatureModel.genmodel的创建。

1. **插件代码的生成**

在生成器模型创建完成后，打开SCFeatureModel.genmodel文件，右击生成器结构树中任何对象，选择“Generate All”按钮将会生成可运行的插件的代码，接下来便可以通过Eclipse运行此插件。

## 特征模型建模工具的运行与演示

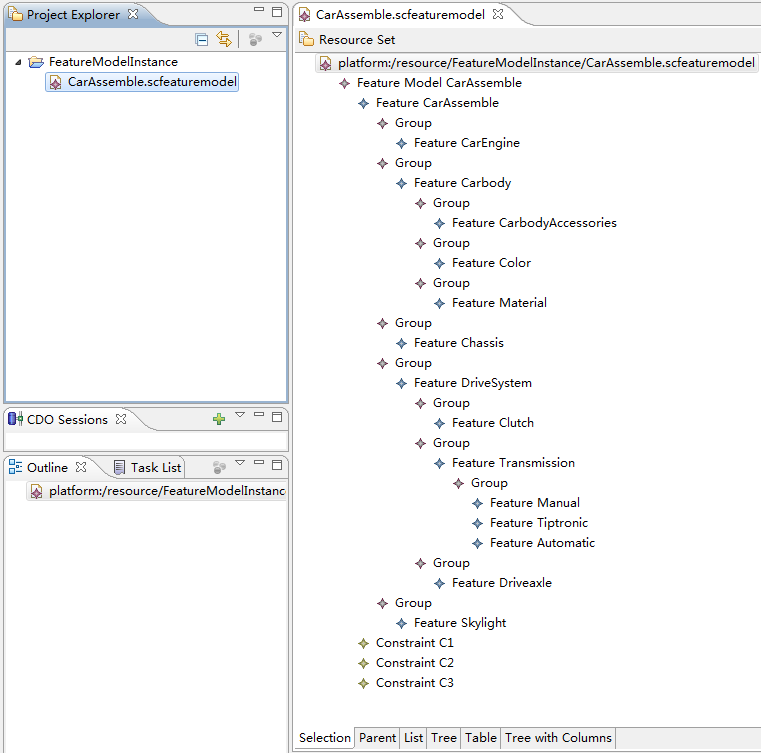


图4-4 特征模型结构化视图

在当前的Eclipse实例下，选择一个生成的项目，随后从“Run”下拉菜单中选择“Run As”命令，再选择“Eclipse Application”选项，即可启动Eclipse第二个实例。在Eclipse的第二个实例中，首先创建一个项目来保存特征模型的实例，右击选择“New>Other”打开向导窗口，然后在向导窗口中打开“General”节点，选择“Project”，输入项目名，完成项目的创建。随后右击该项目在弹出的菜单中选择“New>Other”打开向导窗口，然后在向导窗口中打开“Example EMF Model Creation Wizards”节点，即可看到所有建模工具，选择“SCFeatureModel Model”，点击“Next”进入向导的下一个页面，输入文件名，继续点击“Next”，进入向导的最后一个页面，在这个页面上，必须选择一个模型中定义的类，将其用作根对象的类型，因此在此处选择FeatureModel类作为创建特征模型实例的起点，点击“Finish”按钮完成实例文件的创建。右键单击该实例文件，从弹出菜单中选择“Open With>SCFeatureModel Model Editor”，使用模型编辑器Model Editor打开实例文件，该编辑器提供了模型的结构树视图，如图4-4所示，根级别的元素对应资源，展开后即可看到一个子对象，即FeatureModel类型的模型对象，右击后可以使用弹出菜单添加子对象或者兄弟对象，这些子对象或者兄弟对象必须要符合元模型的定义，所以该菜单只提供有效选项。图4-5在特征模型实例创建完成后，右击该实例文件，可以从弹出菜单中选择“Open With>Text Editor”，查看特征模型的XMI格式。

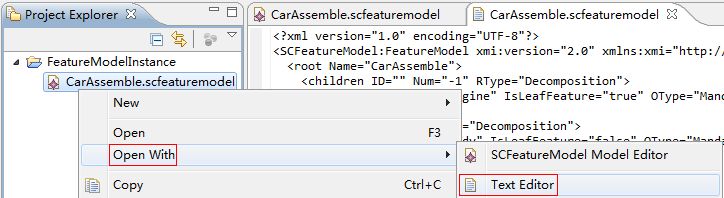


图4-5 查看特征模型的XMI格式

## 小结

本章介绍了特征模型建模工具的设计与实现，涵盖了特征模型Ecore模型的详细信息、特征模型建模工具的开发过程以及如何运行、使用该工具。

# 特征模型到抽象服务组装模型的转换与实现

本节主要介绍扩展WS-BPEL元模型实现的VxBPEL元模型以及使用模型转换机制ATL实现特征模型到抽象服务组装模型的具体转换过程。

## VxBPEL的元模型

第2.2节中已经简单介绍了本方法所用到的模型转换机制ATL，特征模型作为源模型，源模型的元模型，即特征模型的Ecore模型在第五节中已经详细描述。而抽象服务组装模型作为目标模型，由于VxBPEL是对BPEL的扩展，为了统一元模型的标准，因此本方法需要对BPEL的Ecore模型进行扩展去创建目标模型的元模型。

在2.2小节中本文对WS-BPEL做了详细的描述，由于WS-BPEL的Ecore模型由大量的建模类组成，并且建模类之间关系十分复杂，为了便于读者理解，图5-1只显示了WS-BPEL的Ecore模型中重要的建模类以及这些建模类之间的关系。

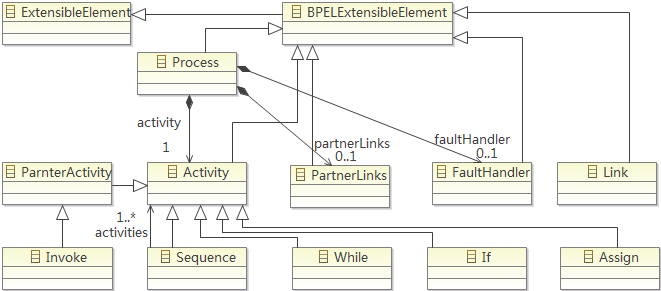


图5-1 WS-BPEL的Ecore模型重要的建模类

通过对WS-BPEL的扩展我们实现了VxBPEL，VxBPEL主要扩展了BPEL的结构化活动（activity），根据第3.3小结中对VxBPEL扩展部分的描述和图3-9所示的VxBPEL的巴克斯范式的描述，VxBPEL扩展部分的Ecore模型如图5-2所示：

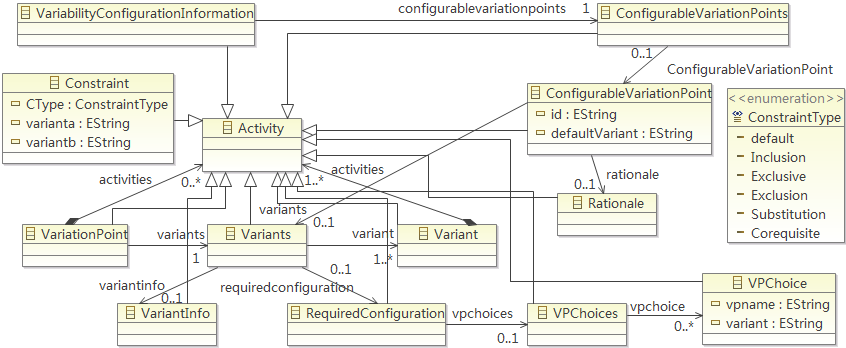


图5-2 VxBPEL扩展部分的Ecore模型

如图5-3所示，我们可以在VxBPEL的模型实例中创建新增的活动。

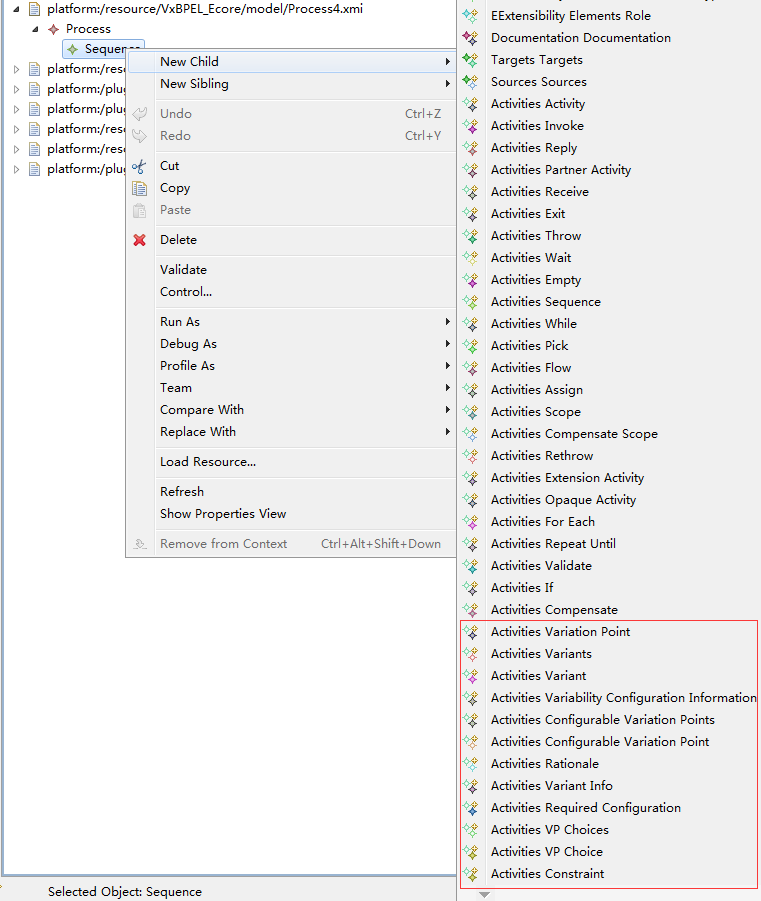


图5-3 VxBPEL模型实例

## 特征模型到抽象服务组装模型转换的机制

通过前面已经建立的特征模型元模型可以对某个领域进行需求分析从而建立特征模型，此时得到的领域特征模型与抽象服务组装模型仍然是两个毫无关系的模型，为了能够从特征模型得到抽象服务组装模型，本文使用ATL模型转换机制来进行这两个模型的转换，转换的框架如图5-4所示。

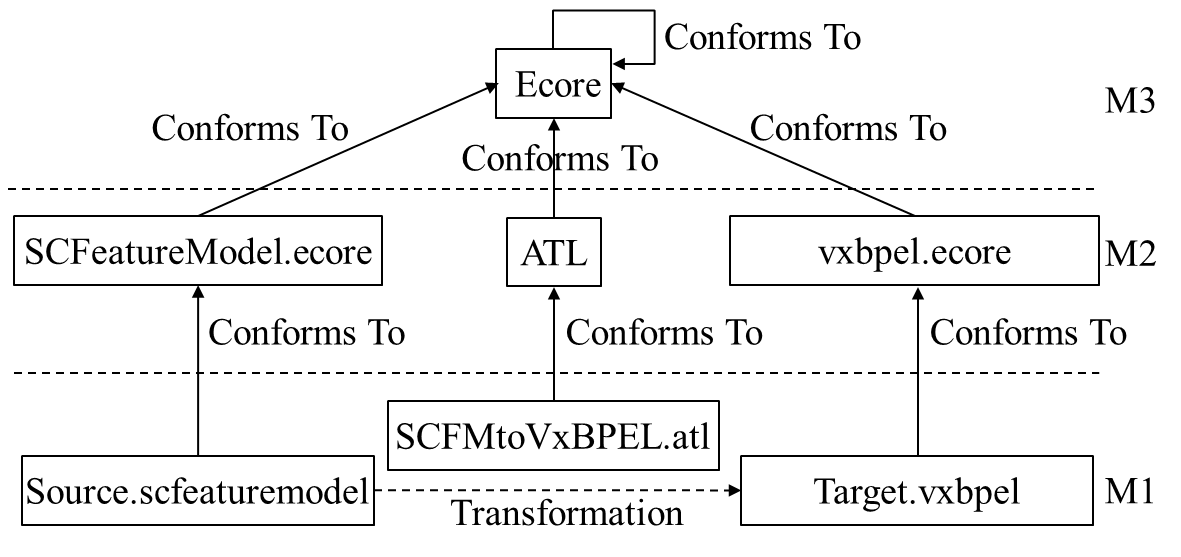


图5-4 特征模型与抽象服务组装模型的转换框架

图5-4所示的模型转换框架中涉及到的概念如表5-1所示：

表5-1 特征模型与VxBPEL模型转换框架中的概念

|  |  |
| --- | --- |
| **概念** | **描述** |
| Source.scfeaturemodel | 源模型，即输入模型 |
| Target.vxbpel | 目标模型，即输出模型 |
| SCFeatureModel.ecore | 特征模型元模型 |
| vxbpel.ecore | vxbpel元模型 |
| SCFMtoVxBPEL.atl | 特征模型与vxbpel模型转换规则文件 |

其中SCFeatureModel.ecore是源模型的元模型，即源元模型；Source.scfeaturemodel是某个领域需求分析得到的特征模型，作为源模型，也就是ATL模型转换引擎的输入模型，该模型可以通过第四节介绍的特征模型建模工具SCFeatureModel得到；SCFMtoVxBPEL.atl文件中定义了特征模型到抽象服务组装模型一系列的转换规则，是整个模型转换框架的核心，该文件是由ATL Language所描述；vxbpel.ecore是VxBPEL的元模型；Target.vxbpel是由输入模型经过ATL引擎转换后得到的输出模型，也就是目标模型。整个模型转换过程示意图5-5如下所示：

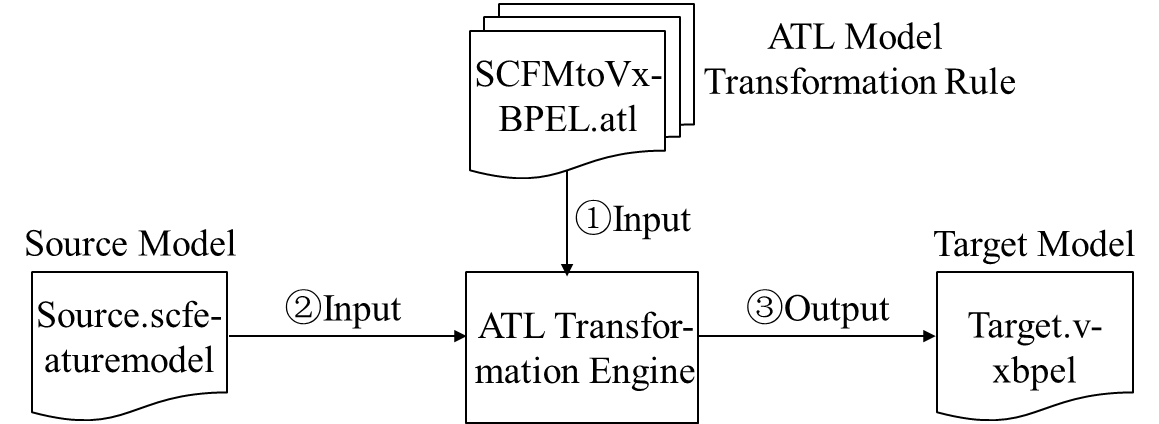


图5-5 特征模型到抽象服务组装模型的转换过程

ATL模型转换引擎（ATL Transformation Engine）是整个模型转换过程的核心，其主要通过解析输入的模型转换规则文件（又叫模型转换规则集，其文件的后缀为.atl）得到的信息来调用源模型和目标模型的结构控制程序以及元素的读写创建程序，模型转换引擎根据转换规则输出目标模型。

模型转换规则文件详细定义了模型转换的过程中类与类之间的映射关系，即模型转换的源码，首先在模型转换文件中定义了四个方法：

1. isLeafFeature()用来判断当前的特征是不是叶子特征，若当前特征的IsLeafFeature属性值为“true”，即该特征为叶子特征，则返回true；若当前特征的IsLeafFeature属性值为“false”，即该特征非叶子特征，则返回false。
2. isOptional()用来判断当前特征是不是可选特征，若当前的特征的OType属性值为“Optionality”，即该特征是可选特征，则返回true； 若当前特征的OType属性值为“Mandatory”，即该特征是强制特征，则返回false。
3. isSpecialization()用来判断当前的一组子特征与其父特征之间的关系是否为特化关系，若当前Group的RType属性值为“Specialization”，即当前一组子特征与父特征之间的关系为特化关系，则返回true；若当前Group的RType属性值为“Decomposition”或者“Characterization”，即当前一组子特征与父特征之间的关系为分解关系和刻画关系，则返回false。
4. isLastGroup()用来判断当前的Group下的特征是否全为叶子特征，若当前的Group的Num属性值为-1，即当前Group下的特征全部为叶子特征，则返回true，否则返回false。

转换规则1-7详细介绍了使用模型转换机制ATL实现模型转换过程主要的映射规则，需要注意的是由于VxBPEL有一个名为“Sequence”的类，同时“Sequence”又是ATL的关键字，重名会产生冲突，在本方法中，将VxBPEL的元模型中Sequence类名暂时改为了Sequence1，后面会使用FM2VxBPEL工具将“Sequence1”改为“Sequence”

**转换规则1** FeatureMode转化为VxBPEL的Process节点。

rule SCFeatureModel\_to\_vxbpelprocess {

from

sc : SCFeatureModel!FeatureModel

to

vx\_process : VxBPEL!Process (

name <- sc.root.Name,

activity <- sc.root.children

)

}

**转换规则2** 如果当前Group下的子特征存在非叶子特征（isLastGroup()方法返回值为“false”），则将Group转换为Process的Sequence节点。

rule Group\_to\_Sequence {

from

g : SCFeatureModel!Group(not g.isLastGroup())

to

s : VxBPEL!Sequence1 (

name <- 'Group'+g.ID,

activities <- g.features

)

}

**转换规则3** 如果当前Group下的特征全部为叶子特征（isLastGroup()方法返回值为“true”），则将Group转化为Sequence节点。

rule Group\_to\_Sequence1 {

from

g : SCFeatureModel!Group(g.isLastGroup() and (not g.isSpecialization()))

to

s : VxBPEL!Sequence1 (

name <- 'Group'+g.ID,

activities <- g.features

)

}

**转换规则4** 如果当前Group下的子特征全为叶子特征（isLastGroup()方法返回值为“true”）并且Group下的子特征与其父特征的精化关系是Specialization（isSpecialization()方法返回值为“true”），则将Group转化为VariationPoint和Variants节点。

rule Group\_to\_Variants {

from

g : SCFeatureModel!Group(g.isLastGroup() and (g.isSpecialization()))

to

vp : VxBPEL!VariationPoint (

name <- 'VPGroup'+g.ID,

variants <- g

),

vs : VxBPEL!Variants (

name <- 'VsGroup'+g.ID,

variant <- g.features

)

}

**转换规则5** 如果Feature不是叶子特征（isLeafFeature()方法返回值为“false”），则将feature转换为vxbpel中的Sequence节点。

rule Feature\_to\_Sequence {

from

f : SCFeatureModel!Feature(not f.isLeafFeature())

to

s : VxBPEL!Sequence1 (

name <- f.Name,

activities <- f.children

)

}

**转换规则6** 如果Feature是叶子特征（（isLeafFeature()方法返回值为“true”）），并且是强制特征（f.isOptional()方法返回值为“false”），则将feature转换为Invoke节点

rule Feature\_to\_Invoke {

from

f : SCFeatureModel!Feature(f.isLeafFeature() and (not f.isOptional()))

to

i : VxBPEL!Invoke (

name <- f.Name

)

}

**转换规则7** 如果Feature是叶子特征（isLeafFeature()方法返回值为“true”），并且该特征是可选特征（f.isOptional()方法返回值为“true”），将feature转换为Variant节点

rule Feature\_to\_VariationPoint {

from

f : SCFeatureModel!Feature(f.isLeafFeature() and f.isOptionality())

to

vp : VxBPEL!VariationPoint (

name <- f.Name

)

}

## 小结

本节介绍了本文用到的模型转换机制ATL以及详细的模型转换规则，通过Eclipse的ATL插件和预定义的转换规则文件，由ATL引擎实现特征模型到抽象服务组装模型的转换。

# 特征模型和服务组装管理工具FM2VxBPEL的设计与实现

本节主要介绍特征模型和服务组装管理工具FM2VxBPEL的设计与实现。FM2VxBPEL提供特征模型到抽象服务组装模型的转换以及运行阶段的对流程动态配置的支持。

## FM2VxBPEL的需求分析

FM2VxBPEL核心功能主要由特征模型管理、流程管理、流程配置和引擎管理四部分组成，其用例图如图6-1所示。

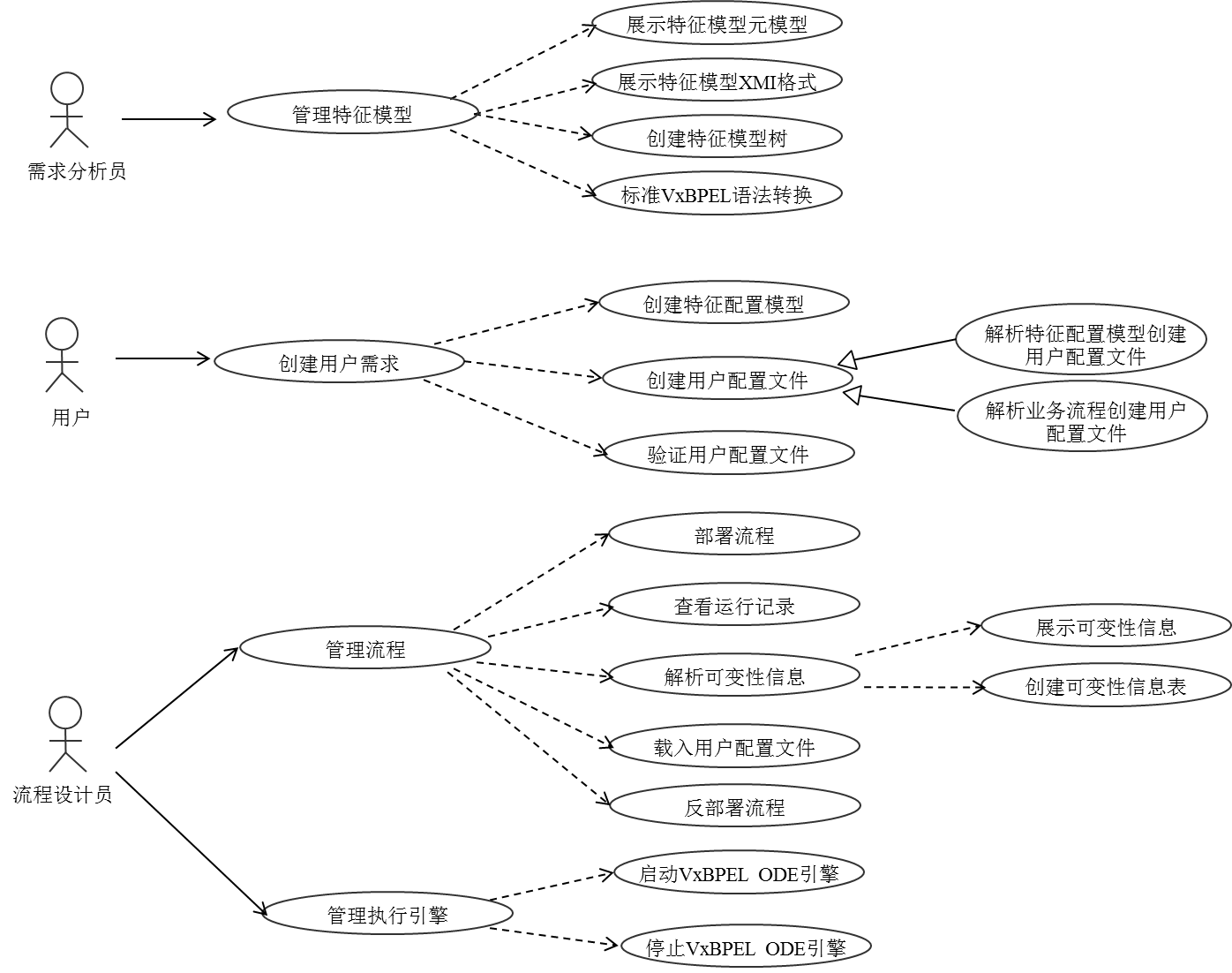


图6-1 FM2VxBPEL工具用例图

该工具的用例说明如下：

1. **管理特征模型：**需求分析人员能够通过该工具对使用SCFeatureModel建模工具得到的特征模型实例进行管理。该用例主要包括展示特征模型元模型、展示特征模型XMI格式、创建特征模型树和标准VxBPEL语法转换四个子用例。
2. 展示特征模型元模型：该用例将特征模型Ecore模型图展示出来，主要为了帮助需求分析人员能够正确的理解特征模型。
3. 展示特征模型XMI格式：使用SCFeatureModel建模工具得到的特征模型实例是以XMI文件的形式存在，需求分析人员可以使用FM2VxBPEL上传该XMI文件，上传过程中的路径以及上传是否成功的结果等日志信息将会展示在“Logger Info”窗口中，同时该文件的XMI源码将会展示在“FM XMI”窗口中。
4. 创建特征模型树：需求分析人员将特征模型文件上传到该工具后，该工具将会自行解析特征模型文件，并创建特征模型树，该特征模型树能够表达领域内的共性需求和差异性需求。
5. 标准VxBPEL语法转换：模型转换得到的目标模型中嵌套了大量的Sequence节点，该用例主要用来帮助需求分析人员将目标模型进行简化，并解析为符合VxBPEL标准语法的抽象服务组装模型。但经过FM2VxBPEL工具转换后得到的文件仍然是一个不完整的VxBPEL工程文件，该文件只表达了一个简单的流程结构，后续仍需要VxBPEL Designer对该VxBPEL工程进行完整地设计。
6. **创建用户需求：**该用例帮助用户实现对需求的配置，主要包括创建特征配置模型、创建用户配置文件和验证用户配置文件三个子用例。
7. 创建特征模型配置模型：用户通过特征模型树进行符合自己需求的特征配置模型的创建，创建过程即为对可选特征的选择。
8. 创建用户配置文件：该用例根据用户的特征配置模型信息或者业务流程中的可变性信息创建用户配置文件，用户配置文件是执行引擎派生流程实例的基础。
9. 验证用户配置文件：在得到用户配置文件后，该用例主要帮助用户验证变异点、变体的选择是否符合特征模型的约束。
10. **管理流程：**该用例帮助流程设计人员管理业务流程的部署，主要包括流程部署、查看部署记录、解析可变性信息、载入用户配置文件、反部署流程五个子用例
11. 流程部署：在启动引擎后，流程设计人员可以将完整的VxBPEL工程文件上传到服务器上。
12. 查看运行记录：该用例允许流程设计人员能够以日志形式查看流程部署、执行过程。
13. 解析可变性信息：该用例帮助流程设计人员解析指定路径下的VxBPEL规格说明，得到可变性信息。该用例包含两个子用例：

* 展示可变性信息：将解析后得到的可变性信息显示在“Variability Info”窗口中。
* 创建可变性信息表：将解析后得到的可变性信息存储在表格中，流程设计人员允许用户可以通过对变异点和变体的勾选来配置流程并创建用户配置文件。

1. 载入用户配置文件：该用例帮助流程设计人员将用户配置文件载入到引擎中相应的应用根目录下。在流程的运行阶段，当用户向流程发出服务请求时，VxBPEL ODE执行引擎将会读取相应的用户配置文件，并根据用户配置文件中保存的变异点和变体的信息派生出具体的流程实例。
2. 反部署流程：当流程不再需要、流程失效或者因为其他原因而需要卸载时，该用例帮助流程设计人员将指定流程部署文件从服务器中删除。
3. **管理引擎：**该用例帮助流程设计人员快速的启动或者停止VxBPEL ODE引擎的执行。该用例包含启动VxBPEL ODE引擎和关闭VxBPEL ODE引擎两个子用例。

## FM2VxBPEL系统设计

本节介绍运行时管理工具FM2VxBPEL系统结构及主要模块设计。

### 系统架构

FM2VxBPEL主要由特征模型管理、流程管理、流程配置和引擎管理四个模块组成。其系统架构如下图6-2所示。

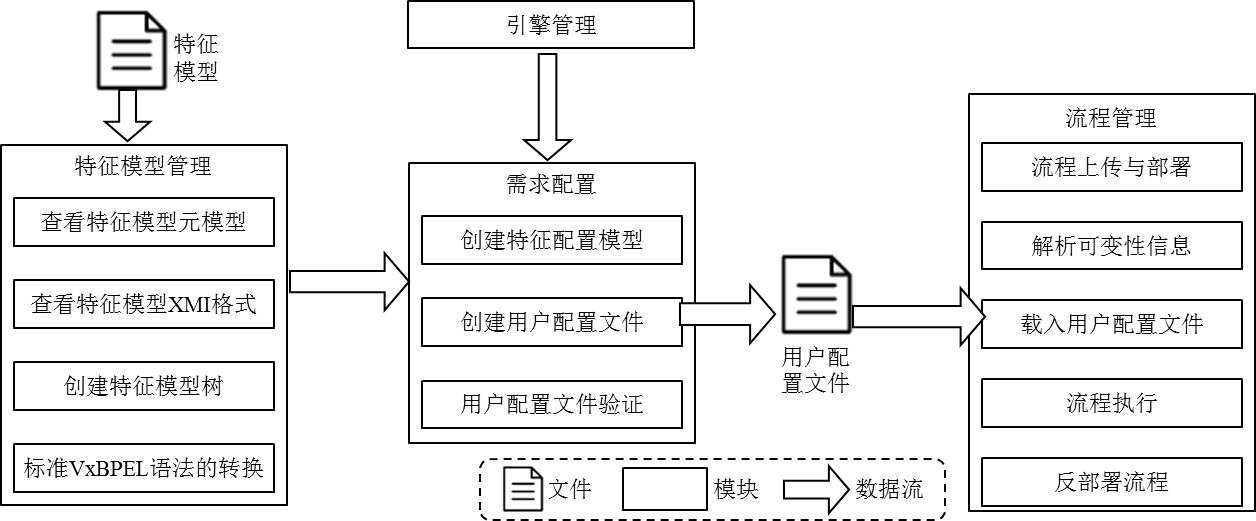


图6-2 FM2VxBPEL的系统架构

### 特征模型管理

特征模型管理模块主要负责查看特征模型元模型、查看特征模型XMI格式、创建特征模型树和标准VxBPEL语法标准文件转换四部分。

* 查看特征模型元模型模块：此模块允许需求分析人员随时查看特征模型的Ecore模型
* 查看特征模型XMI格式模块：由于特征模型以XMI方式进行保存，此模块读取特征模型文件，并将文本内容展示在“FM XMI”窗口中。
* 创建特征模型树模块：此模块主要通过使用Dom4j对特征模型进行解析，最后根据解析出来的结果结合Java的JTree技术进行特征模型树的创建。
* 标准的VxBPEL语法转换模块：前期由特征模型转通过ATL转换得到的VxBPEL抽象服务组装模型嵌套了大量的Sequence节点，其语法也不符合标准的VxBPEL语法，通过这个模块可以删除抽象服务组装模型中冗余的Sequence节点，并将其转换为标准的VxBPEL语法。

### 需求配置

需求配置模块主要负责创建用户的需求，包括创建特征配置模型、创建用户配置文件和用户配置文件验证三部分。

* **创建特征配置模型**：此模块主要负责用户对特征模型树的配置，创建特征配置模型。
* **创建用户配置文件：**此模块主要负责根据用户的特征配置模型创建用户配置文件。
* **用户配置文件验证：**此模块主要负责验证用户配置文件中对变异点和变体的选择是否符合特征模型的约束条件。

### 流程管理

流程管理模块主要负责对引擎中的流程进行管理并解析流程中的可变性信息，包括流程的上传与部署，解析可变性信息、载入用户配置文件、流程的执行以及反部署流程五部分。

* **流程的上传与部署**：此模块主要负责上传业务流程到引擎中并进行流程部署。
* **解析可变性信息**：此模块主要负责将部署到引擎的业务流程中文件扩展名为bpel的文件进行变异点和变体的解析，解析的结果将会展示在 “Variability Info”窗口中。
* **载入用户配置文件**：此模块主要负责将用户配置文件上传到引擎中相应的应用根目录下。
* **流程执行**：用于用户完成流程配置后运行流程，该模块使用Java SOAP API创建SOAP消息模板，获取到用户输入后，加入到消息模板中，并将消息发送到引擎。
* **反部署流程**：此模块主要负责当引擎下的业务流程失效或者不再需要时，流程设计人员对指定流程文件进行卸载。

## FM2VxBPEL的实现

本工具使用Java语言开发，工程总计2257行代码，开发环境为IDEA2017.1.4版本，界面开发主要使用的是Swing技术，特征模型树的创建、特征配置模型的创建、标准VxBPEL语法的转换、可变性信息的解析和用户配置文件的创建主要使用DOM4J插件技术，模拟客户端执行测试用例主要使用Axis2插件技术。下面介绍工具中关键模块的实现。

1. **特征模型管理模块**

特征模型管理模块主要功能实现特征模型树的创建和标准VxBPEl语法的转换，用到的类分别是CreateFMTree和Convert2VxBPEL。

* CreateFMTree类：负责特征模型树的创建。该类会获取到特征模型文件的路径信息，并调用createDocument()方法将特征模型文件转化为Document对象，然后使用DOM4J插件解析Document对象，由于XMI本身是一个层次结构，通过DOM4J迭代读取节点信息并结合JTree技术创建特征模型树。
* Convert2VxBPEL类：负责标准VxBPEl语法的转换。该类获取到不符合标准VxBPEL语法文件的路径信息，并调用createDocument()方法将特征模型文件转化为Document对象，然后调用createNewFile()方法将使用DOM4J插件解析后创建的符合标准VxBPEL语法的XMI语句保存在到新文件中。

1. **需求配置模块**

需求配置模块主要功能创建用户配置文件，用到的类分别是CreateCnfFM、CreateUCCFFile和Validate。

* CreateConFM类：负责特征配置模型的创建。该类获取用户对特征模型树的选取信息，并调用createCnfFM()方法创建特征配置模型。
* CreateUCCFFile类：负责用户配置文件的创建。该类首先创建用户配置文件的模板，随后通过DOM4J解析特征配置模型，并根据解析得到的信息创建用户配置信息，最后调用createUCCFFile()方法存储用户配置信息到用户配置文件中。
* ValidateUCCF类：负责验证用户配置文件中对变异点和变体的选择是否符合特征模型的约束条件。该类首先读取用户配置文件中的信息，随后调用validateUCCF()方法进行验证。validateUCCF()方法借助Java的ArrayList集合类存储每一条约束的约束类型、约束的双方，然后用每个ArrayList集合类去验证用户配置信息中是否存在不符合约束条件的变体。如果用户配置信息符合所有的约束关系，则说明这是一个正确的用户配置文件。

1. **流程管理模块**

流程管理模块主要功能包括业务流程部署和反部署、可变性信息的解析、业务流程的执行，用到的类主要包括ODEManager、ParseVPInfo以及SoapAction。

* ODEManager类：负责业务流程的部署和反部署。通过复用Apache ODE自身提供的流程管理系统，该类调用openDefaultBrowser()方法，在当前系统的默认浏览器下请求“http://localhost:8080/ode/deployment.html”页面进行业务流程的管理。
* ParseVPInfo类：负责可变性信息的解析。该类根据流程设计人员提供的流程文件（文件扩展名为bpel）的路径，调用parseVPInfo()方法通过DOM4J解析技术读取流程文件中的变异点和变体信息，并将所有的变异点和变体信息存储到可变性信息表中，可变性信息表由JTable来实现，通过对该JTable的设置，可以对可变性信息表进行配置，即可实现对变异点和变异体的选择。
* SoapAction类：负责模拟Web服务客户端对业务流程进行调用。该类通过DOM4J对业务流程的WSDL文件进行解析，由解析得到的服务的地址、命名空间、操作名、请求消息及其类型创建SOAP消息模板。在用户输入请求消息后，该类将获取到的请求消息添加到SOAP消息模板中，并且将SOAP消息发送给引擎，得到引擎的返回结果，即服务的调用结果。

1. **引擎管理模块**

引擎管理模块负责引擎的启动和停止，用到的类是EngineManage。

* EngineManage类：负责引擎的启动和停止。该类分别调用startODE()和stopODE()方法，实现当前系统环境变量下的引擎的启动和停止。

## 工具演示与验证

FM2VxBPEL工具主要功能是管理特征模型管理、创建用户需求和管理流程，本节以汽车组装系统为例进行工具的演示。

### 特征模型管理和需求配置

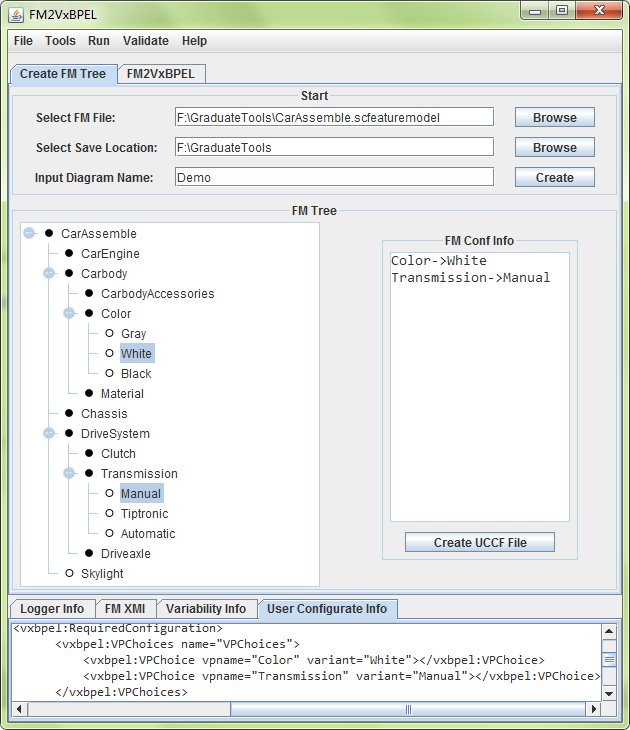


图6-3 特征模型管理界面

图6-3是FM2VxBPEL工具中特征模型管理主界面，首先需求分析人员通过点击“Help”菜单下的“FeatureModel Metamodel”子菜单，将会弹出新的窗口来显示特征模型的元模型。其次需求分析人员通过界面提示上传特征模型到该工具，点击“Create”按钮，该工具完成特征模型树创建。由于特征模型树不能完整的展示领域需求分析并且不够直观，因此在创建特征模型树的同时，该工具会根据解析到的信息创建特征模型图（如图6-4所示），并允许需求分析人员自定义特征模型图的保存路径和图片名称。

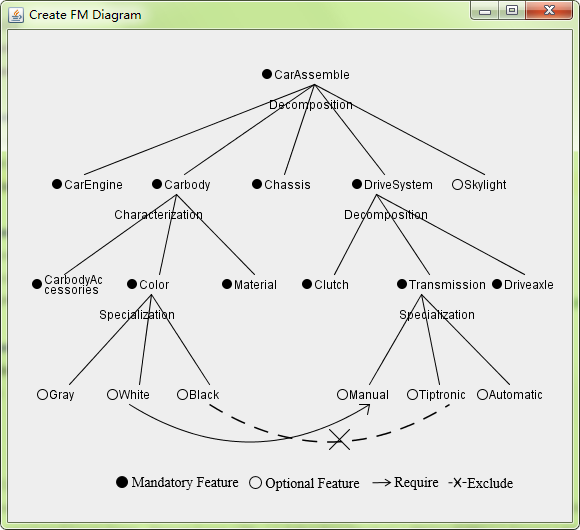
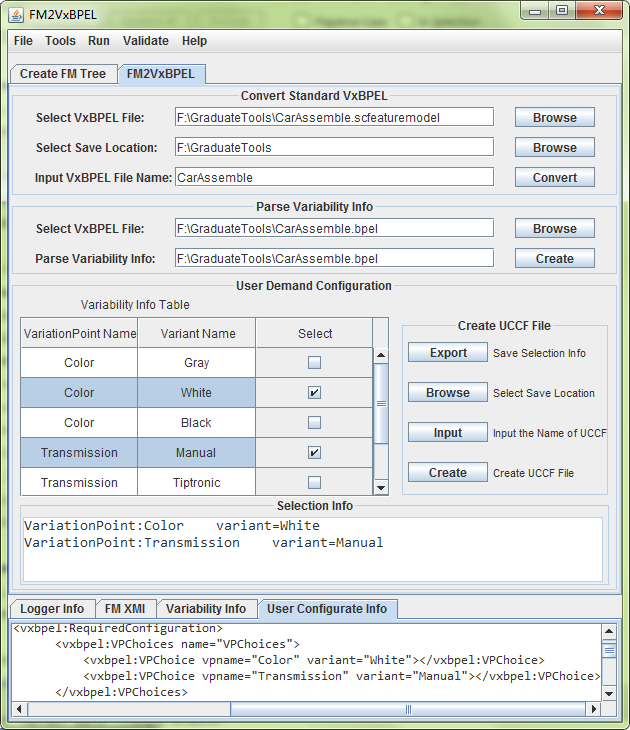


图6-4 特征模型图

在如图6-3所示的界面中，用户能够对特征模型树进行配置创建特征配置模型，特征模型树中实心圆圈标识强制特征，空心圆圈标识可选特征，用户只需要对可选特征进行选择即可，被选中的特征有颜色的变化作为标识，并且相应的特征信息显示在“FM Conf Info”区域中。在用户完成特征模型的配置后，点击“Create UCCF File”按钮将会弹出窗口补充用户配置文件名称和路径等信息。在用户配置文件创建完成后，用户配置文件的内容会显示在“User Configurate Info”窗口。



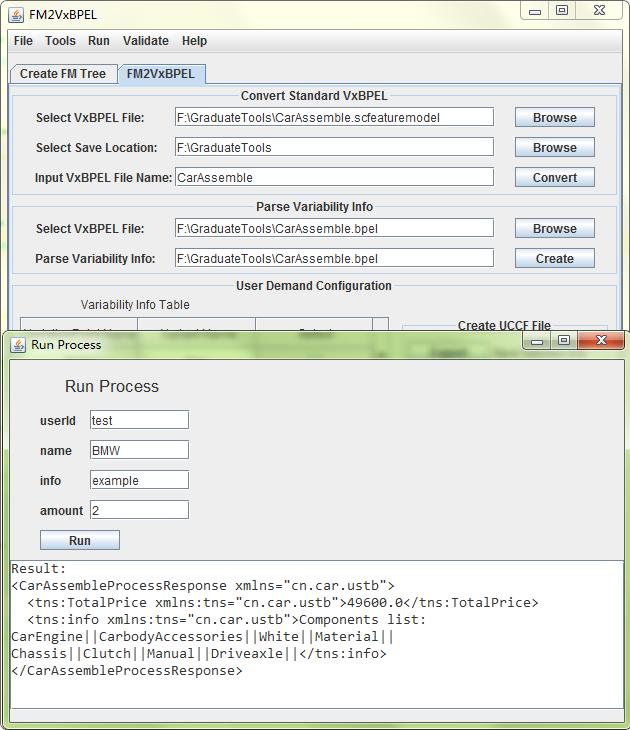


图6-5 特征模型管理界面

### 流程管理和配置管理

如图6-4所示主界面是流程管理和配置管理的主界面，首先用户可以通过step1、step2和step3生成符合标准VxBPEL语法的业务流程，再由VxBPEL Designer将完整的业务流程创建完成后，通过step4和step5完成可变性信息的解析和可变性信息表（Variability Info Table）的创建，可变性信息将会展示在Variability窗口中，用户通过可变性信息表中Select一栏进行变异点与变体的选择，被选中的一行将会有颜色的变化，工具将会记录用户的选择，并创建用户配置文件，此时用户可以自定义用户配置文件的路径和名称，完整的用户配置文件内容将会展示在User Configurate Plan窗口中。

在用户配置文件创建完成后，选择菜单栏Validate下的Validate UCCF子菜单进行用户配置文件的验证。

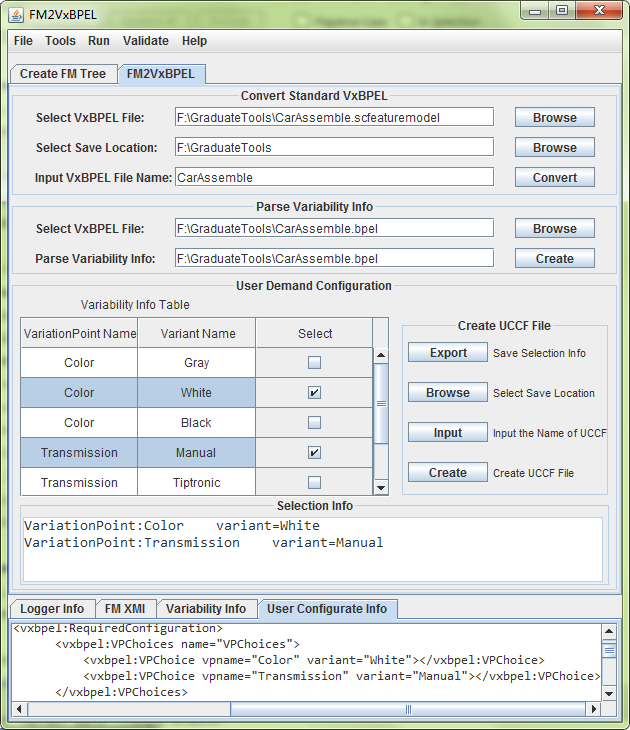


图6-6 流程配置和流程管理界面

Tools菜单下包括“ODE Manage”子菜单和“Run Process”子菜单，点击“ODE Manage”子菜单会打开默认浏览器的“http://localhost:8080/ode/deployment.html”地址进行服务的部署和反部署；如图6-5所示，点击“Run Process”子菜单会弹出新的窗口模拟SOAP客户端执行测试用例。

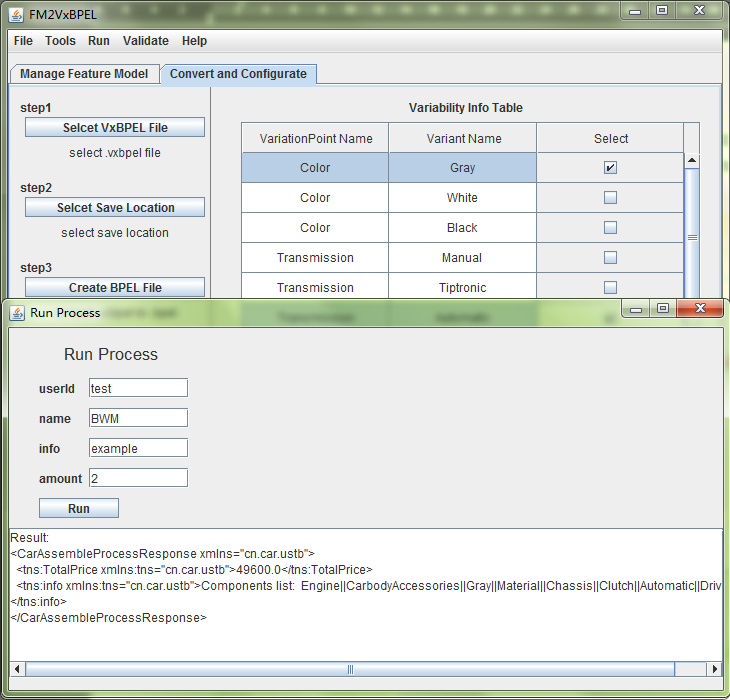


图6-7 流程运行界面

## 小结

本章主要介绍了管理工具FM2VxBPEL，涵盖了需求分析、系统设计、工具实现和演示与验证等方面的介绍。该工具主要使用Swing插件、DOM4J插件、Axis2插件进行了开发，实现了需求分析阶段特征模型图的生成以及对抽象服务组装模型文件的部分转换，并且对业务流程部署、配置、运行、卸载全过程支持，用户也可以通过工具进行个性化选择，该工具对全生命周期的适应性服务组装过程提供了帮助。

# 实例验证

模拟汽车组装领域系统来验证本论文所提出的面向全生命周期的适应性服务及其支持平台的可行性和性能。

## 实验目的及实验设计

实例研究有四个目的：（1）验证特征模型是否能够充分表达领域的共性需求和可变性需求；（2）验证特征建模工具SCFeatureModel对特征模型建模的支持能力；（3）验证特征模型转换后得到的抽象服务组装模型与需求的一致性；（4）验证所开发的特征模型和服务组装管理工具FM2VxBPEL对全生命周期的适应性服务组装过程的支持能力。

根据全生命周期的适应性服务组装过程，我们将实验分为一下几个步骤

1. **领域的需求分析和特征模型的建模：**通过特征模型建模工具SCFeatureModel对某个领域进行特征模型建模，验证特征模型能否充分表达领域的共性需求和可变性需求，同时验证特征模型建模工具SCFeatureModel的有效性。
2. **抽象服务组装模型与需求的一致性验证：**为了能够保证服务组装系统的功能与需求的一致性，需要首先验证抽象服务组装模型与需求的一致性，即验证通过特征模型经过转换后得到的抽象服务组装模型所表达的业务流程是否能够充分表达需求。
3. **特征模型与服务组装管理工具FM2VxBPEL工具的使用：**验证FM2VxBPEL工具管理特征模型功能、流程部署、创建用户配置文件、验证用户配置文件、模拟客户端执行测试用例等功能。FM2VxBPEL为用户提供可视化的界面，通过解析VxBPEL的规格说明创建可变性信息表，用户通过可变性信息表进行用户配置文件的创建，若用户的需求符合特征模型的约束，则通过该工具为用户模拟客户端执行流程。

## 汽车组装系统

### 汽车组装系统领域需求分析

随着经济的发展，汽车已经成为了我们生活中必不可少的交通工具。“汽车组装”业务流程用于辅助如何根据用户不同需求组装出不同配置的汽车。对于一个汽车生产商来说，一个汽车生产线不只完成单一配置的单一车型，不同配置的车型的生产装配过程中有大量相同的过程。为每一种不同配置的车型开发一个专属的控制流程不可避免的会产生大量的冗余，而且增加了维护复杂度，严重影响生产的效率。

通过对汽车组装领域进行分析，一辆完整的汽车其无论什么配置，必须由引擎、车身、底盘和传动系统四部分组成，这四个特征即为强制特征，而天窗是汽车的特色配置和功能的延伸，其作为一个可选的特征存在于汽车组装领域当中；对于车身特征来说，根据本文对特征的定义[32]，从用户角度来看，其具有三个属性，分别是颜色、材质和车身附件，并且这三个特征都是车身所必须具有的，对于车身颜色来说，必须要从可选的颜色中选择一种颜色以实现这个颜色特征；对于传动系统特征来说，其由离合器、变速器和半轴三个特征组成，这三个特征缺一不可，其中变速器有三种实现方式，分别是手动挡、手自一体和自动挡。为了能模拟特征模型中的约束关系，我们假设①如果用户选择了白颜色的车身，则变速器的实现方式必须指定为手动挡；②如果用户选择了黑颜色的车身，则变速器实现方式不能为手自一体。

### 实验评估

1. **特征模型对需求的表达能力和特征建模工具有效性的评估**

通过对汽车组装领域的需求分析，图4-4展示了使用SCFeatureModel对该领域进行特征模型建模的结果，该模型以XMI作为序列化的格式，因此通过分析该模型文件，我们得到如图7-1所示的特征模型树状图。

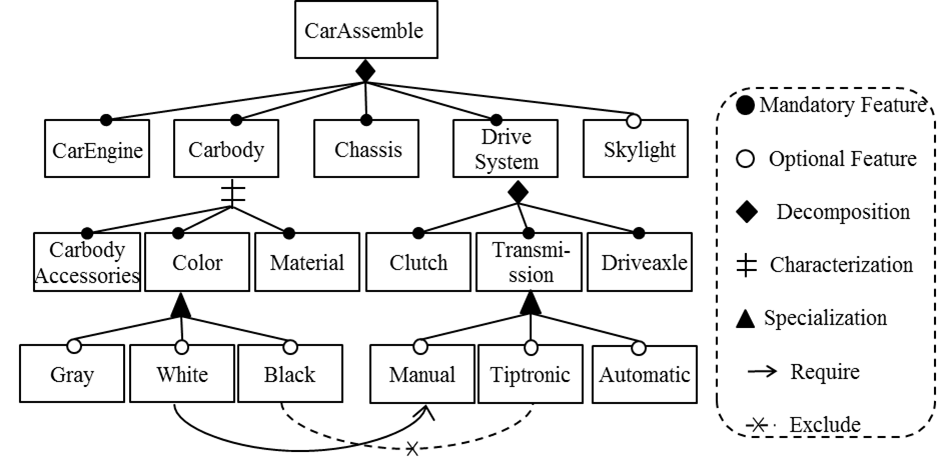


图7-1 汽车组装领域特征模型树状图

由图7-1可以看出，本方法所定义的基于元模型的特征模型完全可以正确的表达领域内的共性和可变性的需求，这为下一步与抽象服务组装模型的转换提供了良好的基础，同时，本方法所开发的建模工具对服务组装领域特征模型的建模提供了有效的支持。

1. **抽象服务组装模型与需求一致性的评估**

通过本文所开发的建模工具，得到的抽象服务组装模型嵌套了大量的sequence节点并且不符合标准的VxBPEL的语法，为此，通过FM2VxBPEL工具，我们可以将其转化成简单的符合VxBPEL语法的抽象服务组装模型，通过VxBPEL Designer工具，可以得到如图7-2所示的业务流程。

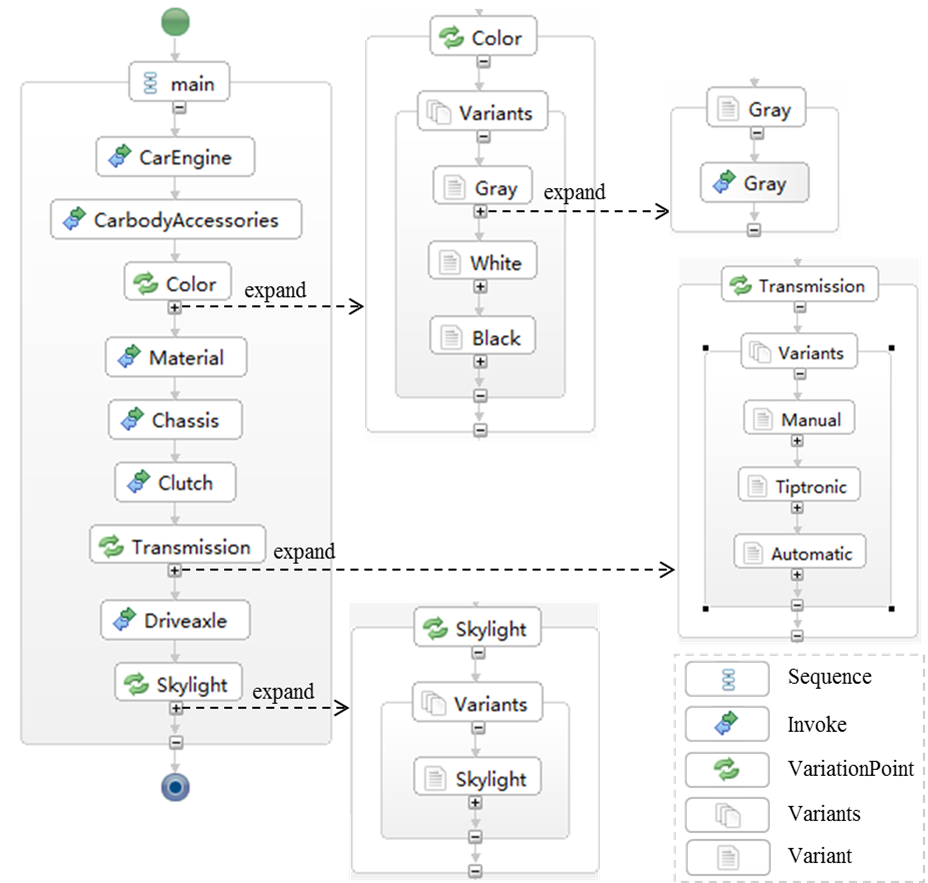


图7-2 转换后得到的简单业务流程

由于此时得到的业务流程仍然是一个不完整的工程文件，需要继续使用VxBPEL Designer完成整个业务流程的开发。通过图7-2，此时的抽象服务组装模型能充分表达需求分析，即其与特征模型所表达的需求分析结果是一致的。

1. **FM2VxBPEL工具有效性的评估**

通过分析汽车组装的业务流程，得到活动图，如图7-3所示。该过程共包含9个活动，为了生产不同配置的车型，在存在不确定性因素的活动下设置变异点，每个变异点下存在变体。该实例流程中3处活动存在不确定性因素，根据汽车组装领域的特征模型，在选择车身颜色这个变异点下，将会设置三个变体，分别代表提供白色车身、灰色车身和黑色车身三个服务，该业务流程片段的VxBPEL代码如图7-4所示。

安装引擎

Engine

◊选择车身颜色

Color

安装车身附件

Carbody Accessories

安装离合器

Clutch

安装半轴

Driveaxle

◊安装天窗

Skylight

◊

存在不确定

性因素活动

选择车身材质

Material

◊安装变速器

Transmission

安装底盘

Chassis

图7-3 引入可变性设计的汽车组装活动图

**Color**

**White**

**Gray**

**Black**

含可变性设计的选择车身颜色活动

VxBPEL代码

<vxbpel:**VariationPoint** name="**Color**">

<vxbpel:**Variants**>

<vxbpel:**Variant** name="**Gray**">

<bpel:sequence name="Color\_Gray">

······

</bpel:sequence>

</vxbpel:**Variant**>

<vxbpel:**Variant** name="**White**">

<bpel:sequence name="Color\_White">

······

</bpel:sequence>

</vxbpel:**Variant**>

<vxbpel:**Variant** name="**Black**">

<bpel:sequence name=" Color\_Black ">

······

</bpel:sequence>

</vxbpel:**Variant**>

</vxbpel:**Variants**>

</vxbpel:**VariationPoint**>

图7-4 流程片段

同时使用FM2VxBPEL对可变性信息进行解析，创建可变性信息表，用户通过可变性信息表进行变异点和变体的选择，随后工具即可创建用户配置文件，生成的配置文件如图7-5所示，由该配置文件可知，该用户根据自己的需求指明了变异点的选取情况和不同变异点下变体选择情况，例如在选择车身颜色变异点“Color”指明选择了“Gray”变体。不同的配置文件对变异点的选取情况和不同变异点变体选择情况不同。

<UserConfig name="default" xmlns:vxbpel="http://vxbpel.rug.org" >

<vxbpel:VariabilityConfigurationInformation name="VariabilityConfigurationInformation">

<vxbpel:ConfigurableVariationPoints>

<vxbpel:ConfigurableVariationPoint>

<vxbpel:Variants>

<vxbpel:Variant name="PlanA">

<vxbpel:RequiredConfiguration>

<vxbpel:VPChoices name="VPChoices"><vxbpel:VPChoice vpname="**Color**" variant="**Gray**"></vxbpel:VPChoice><vxbpel:VPChoice vpname="**Transmission**" variant="**Automatic**"></vxbpel:VPChoice>

</vxbpel:VPChoices>

</vxbpel:RequiredConfiguration>

</vxbpel:Variant>

</vxbpel:Variants>

</vxbpel:ConfigurableVariationPoint>

</vxbpel:ConfigurableVariationPoints>

</vxbpel:VariabilityConfigurationInformation>

</UserConfig>

图7-5 用户配置文件

在用户配置文件生成后，还需对配置文件进行验证是否符合特征模型的约束关系，如果符合约束关系，接下来便可继续使用该工具模拟客户端进行流程的执行，如果不符合约束关系，则会提示配置失败，需要重新配置，但是在每次配置完成后，都必须要进行验证。

表7-1 试验环境配置

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

接下来验证FM2VxBPEL工具对运行时配置的支持能力。表7-1展示了实验环境的配置和相关参数，本文采用VxBPEL ODE引擎部署所开发的汽车组装业务流程，然后采用FM2VxBPEL工具随机生成8种符合特征模型约束的用户配置方案，由FM2VxBPEL模拟客户端对引擎发送请求信息，引擎接受到请求信息后，根据用户配置文件派生出具体的流程执行，并且得到将返回结果。

表7-2列出了上述8种配置方案的流程配置及其运行结果，VP1表示变异点“Color”，V1a表示变体“Gray”、V1b表示变体“White”、V1c表示变体“Black”；VP2表示变异点“Transmission”，V2a表示变体“Manual”、V2b表示变体“Tiptronic”、V2c表示变体“Automatic”；VP3表示变异点“Skylight”，V3a表示变体“Skylight”。“配置文件”栏表示当前用户配置文件名；“配置方案/运行结果(变异点和变体的选择)”栏列出了当前用户配置文件的配置方案实验结果；“运行时间”栏列出模拟客户端从开始调用流程到获取返回消息的平均执行时间开销。

表7-2 配置方案和运行结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **配置文件** | **配置方案/运行结果(变异点和变体的选择)** | | | **运行时间(ms)** |
| VP1 | VP2 | VP3 |
| test1.uccf | V1a | V2a | NA | 887 |
| test2.uccf | V1a | V2a | V3a | 925 |
| test3.uccf | V1b | V2a | NA | 902 |
| test4.uccf | V1b | V2a | V3a | 913 |
| test5.uccf | V1a | V2b | NA | 810 |
| test6.uccf | V1a | V2b | V3a | 942 |
| test7.uccf | V1c | V2c | NA | 856 |
| test8.uccf | V1c | V2c | V3a | 932 |

实验结果表明FM2VxBPEL工具能够根据用户需求创建不同的配置文件，并且能够模拟客户端执行测试用例。

## 实验结果与结论

通过上述的汽车组装领域，本文对特征模型对需求的表达能力、特征模型的建模工具、抽象服务组装与需求的一致性以及全生命周期的适应性服务组装管理工具进行了充分的验证，实例研究结果表明：

1. 特征模型能够充分表达领域内的共性需求和可变性需求。在汽车组装领域，汽车车身颜色、变速器和天窗对于用户来说可以根据自己的需求进行选择额，因此是可变性需求，其它的特征作为汽车组装的基本元素，对于用户来说是共性需求。
2. 通过使用特征模型建模工具，对汽车组装领域特征模型进行了建模，该工具能够有效的支持服务组装领域系统建模。
3. 通过对汽车组装的需求进行分析，特征模型转换后得到的抽象服务组装所表达的业务流程与汽车组装的需求分析结果基本一致。
4. 通过汽车组装这个实例，我们验证了所开发的特征模型和服务组装管理工具FM2VxBPEL能够正确并且高效的实现流程部署，并能够通过解析得到正确的可变性信息。同时该工具支持用户对可变性信息进行配置来表达自己的需求，并为用户提供了客户端执行业务流程。
5. 由表7-2“运行时间”我们可以看出，选择的变异点的数目越多，业务流程的执行时间相对较长，这是因为VxBPEL ODE执行引擎会根据用户配置文件去选择相应的活动来执行，执行的活动越少，执行时间则越短。

# 结论

针对传统的服务组装过程中业务流程设计与需求分析糅杂在一起的问题，本文在课题组前期工作中提出的基于可变性管理的适应性服务组装方法的研究基础上，进一步提出一种面向全生命周期的适应性服务组装方法，并且开发了相应的支持平台。

本文取得的主要研究成果如下：

1. 提出一种面向全生命周期的适应性服务组装方法，在基于可变性的适应性服务组装方法的研究基础上，将需求分析从业务流程设计中明确分离出来，以特征模型作为需求分析的产物，该模型能够表达领域内的共性需求、可变性需求以及需求的约束，随后根据本文所提出的模型转换规则，将特征模型转换为抽象服务组装模型，最后以特征模型驱动整个业务流程的开发过程。
2. 设计并实现了特征模型建模工具，该工具支持服务组装系统领域特征模型的建模，所建立的模型层次分明、结构清晰，能够充分表达领域需求。
3. 设计并实现了特征模型和服务组装管理工具，该工具支持全生命周期适应性服务组装过程，包括特征模型图的创建、业务流程的管理（部署、反部署等）、用户配置文件的验证与创建，并且为用户模拟客户端执行业务流程。
4. 通过实例研究，验证了面向全生命周期的适应性服务组装方法的可行性和有效性，所开发支持平台的可用性、易用性等特点。

当前工作存在的不足和未来工作的研究方向如下所述：

1. 本文所提出的方法只能由特征模型转换为抽象服务组装模型，无法实现由抽象服务组装模型到特征模型的逆向转换，同时，转换后得到的抽象服务组装模型只能表达简单的流程结构，仍需要VxBPEL Designer进行进一步的补充，未来工作可以通过构造辅助模型，由特征模型直接转换为可以运行的VxBPEL业务流程。
2. 目前本文所提出的约束只有两种，并且这两种约束关系都是静态的，本方法未能实现动态的约束关系以及约束关系的扩展，未来工作可以研究服务组装领域更为复杂的约束关系以及如何实现动态的约束关系。
3. 通过更多的实例全面的验证所提出的方法及所开发的支持工具的可行性、有效性和健壮性。

作者简历及在学研究成果

1. 作者入学前简历

|  |  |  |
| --- | --- | --- |
| 起止年月 | 学习或工作单位 | 备注 |
| 2011.09-2015.06 | 山东理工大学 |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. 在学期间从事的科研工作
2. 在学期间所获的科研奖励
3. 2016年第七届中国计算机学会服务计算学术会议（NCSC2016）软件服务创新大赛一等奖，作品《VxSaaS:基于可变性模型的可定制SaaS软件支撑平台》，指导教师孙昌爱教授，作者张在兴、张鑫、贾婧婷。
4. 2017年第八届中国计算机学会服务计算学术会议（NCSC2017）软件服务创新大赛二等奖，作品《MT4WS:面向Web服务的脱变测试系统》，指导教师孙昌爱教授，作者付安、张媛钰、张在兴。
5. 在学期间发表的论文
6. 孙昌爱, 张在兴, 张在兴. 一种基于可变性模型的可复用与可定制SaaS软件开发方法[J]. 软件学报, 2017. (已录用)

独创性说明

本人郑重声明：所呈交的论文是我个人在导师指导下进行的研究工作及取得研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写的研究成果，也不包含为获得北京科技大学或其他教育机构的学位或证书所使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

签名：\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_

关于论文使用授权的说明

本人完全了解北京科技大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

**（保密的论文在解密后应遵循此规定）**

签名：\_\_\_\_\_\_\_\_\_\_\_ 导师签名：\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_

# 参考文献

1. []Papazoglou M, Traverso P, Dustdar S, Leymann F. Service-oriented computing: a research roadmap[J]. International Journal on Cooperative Information Systems, 2008, 17(2): 223-255. [↑](#endnote-ref-2)
2. [] Peltz C. Web Services Orchestration-a review of emerging technologies, tools and standards Technical report. Technical white paper[EB/OL]. http://devr-esource.hp.com/drc/technical\_white\_papers/WSOrch/WSOrchestration.pdf, 2003. [↑](#endnote-ref-3)
3. [] Aalst WMP, Dumas M,.Hofstede A. Web service composition languages: in new bottles?[C] . Proceedings of the 29th Euromicro Conference, Washington: IEEE Computer Society, 2003: 298-305. [↑](#endnote-ref-4)
4. [] 孙昌爱, 王冠, 赵永梅. Web 服务的开发过程框架及其实例研究[J]. 中国科技论文在线精品论文, 2011, 4(10): 1835-1839. [↑](#endnote-ref-5)
5. [] Xiao Z, Cao DG, You C, Mei H. Towards a constraint-based framework for dynamic business process adaptation[C]. Proceedings of the 2011 IEEE International Conference on Services Computing (SCC 2011), Washington: IEEE Computer Society, 2011:685-692. [↑](#endnote-ref-6)
6. [] Jordan D, Evdemon J, Alves A, Arkin A, Askary S, Barreto C, Bloch B, Curbera F. Web services business process execution language version 2.0[S].http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html [↑](#endnote-ref-7)
7. [] Tsai WT, Song WW, Paul R, Cao ZB, Huang H. Services-oriented dynamic reconfiguration framework for dependable distributed computing[C]. Proceedings of the 36th Annual Computer Software and Applications Conference (COMPSAC 2004), Washington: IEEE Computer Society, 2004:554-559. [↑](#endnote-ref-8)
8. [] Sun CA, Rossing R, Sinnema M, Bulanov P, Aiello M. Modeling and managing the variability of Web service-based systems [J]. Journal of Systems and Software, 2010, 83(3): 502-516. [↑](#endnote-ref-9)
9. [] Sun CA, Aiello M. Towards variable service compositions using VxBPEL[C]. Proceedings of the 10th International Conference on Software Reuse (ICSR 2008), Berlin: Springer-Verlag, 2008: 257-261. [↑](#endnote-ref-10)
10. [] Sun CA, Xue T, Aiello M. ValySeC: A variability analysis tool for service compositions using VxBPEL[C]. Proceedings of the 5th IEEE Asia-Pacific Services Computing Conference (APSCC 2010), Washington: IEEE Computer Society, 2010: 307-314.(缺少断字) [↑](#endnote-ref-11)
11. [] Koning M, Sun CA, Sinnema M, Avgeriou P. VxBPEL: Supporting variability for Web services in BPEL[J]. Information and Software Technology, 2009, 51(2): 258-269. [↑](#endnote-ref-12)
12. [] Sun CA, Wang P, Zhang X, Aiello M. VxBPEL\_ODE: A variability enhanced service composition engine. Proceedings of the Asia-Pacific Web Conference 2014 Workshops on Web Technologies and Applications (APWeb 2014), Switzerland: Springer International Publishing, 2014: 69-81. [↑](#endnote-ref-13)
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[D]. 北京：北京科技大学, 2014: 1-79. [↑](#endnote-ref-14)
14. [] 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[D]. 北京：北京科技大学, 2015: 1-67. [↑](#endnote-ref-15)
15. [] 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[D]. 北京：北京科技大学, 2016: 1-78. [↑](#endnote-ref-16)
16. [] Griss ML. Software reuse: architecture, process and organization for business success[C]. Proceeding of the 8th Israeli Conference on Computer Systems and Software Engineering. Washington: IEEE Computer Society, 1997: 86-89. [↑](#endnote-ref-17)
17. []Topaloglu NY, Capilla R. Modeling the variability of web services from a pattern point of view[C]. Proceeding of the European Conference on Web Services (ECOWS 2004). Berlin: Springer-Verlag, 2004: 128-138. [↑](#endnote-ref-18)
18. []Sinnema M, Deelstra S, Nijhuis J, Bosch J. COVAMOF: A framework for modeling variability in software product families[C]. Proceeding of the 3rd International Conference on Software Product Lines (SPLC 2004). Berlin: Springer-Verlag, 2004: 197–213. [↑](#endnote-ref-19)
19. [] Weerawarana S, Curbera F, Leymann F, Storey T, Ferguson D, Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more[M]. New Jersey: Prentice Hall, 2005:1-265. [↑](#endnote-ref-20)
20. [] Christensen E, Curbera F, Meredith G, Weerawarana S. Web services description language (WSDL)[S]. https://www.w3.org/TR/wsdl. [↑](#endnote-ref-21)
21. [] Box D, Ehnebuske D, Kakivaya G, Layman A, Mendelsohn N, Nielsen HF,Thatte S, Winer D. Simple object access protocol (SOAP)1.1[S]. https://www.w3.org/TR/2000/NOTE-SOAP-20000508/. [↑](#endnote-ref-22)
22. []Sadiq S, Maria E, Sadiq W. Specification and validation of process constraints for flexible workflows[J]. Information Systems. 2005, 30(5): 349–378. [↑](#endnote-ref-23)
23. [] Emmerich W, Butchart B, Chen L, Wassermann B, and Price S. Grid service orchestration using the Business Process Execution Language (BPEL)[J]. Journal of Grid Computing, 2005, 3(3): 283-304. [↑](#endnote-ref-24)
24. []Apache ODE[EB/OL]. http://ode.apache.org/developerguide/architectural over-view.html, 2003.(左对齐，允许断行) [↑](#endnote-ref-25)
25. []ActiveBPEL development server user’s guide[EB/OL]. http://sse.cs.ucl.ac.uk/omii-bpel/docs/server/AdministrationConsole.pdf, 2007. [↑](#endnote-ref-26)
26. []Oracle BPEL process manager[EB/OL]. http://www.oracle.com/technetwork/middleware/bpel/overview/index.html. (左对齐，允许断行) [↑](#endnote-ref-27)
27. []WebSphere process server[EB/OL]. https://www.ibm.com/software/integration/wps/.(左对齐，允许断行) [↑](#endnote-ref-28)
28. []Krut R, Zalman N. Domain analysis workshop report for the automated prompt and response system domain[R]. Pittsburgh: Software Engineering Institute, Carnegie Mellon University. 1996. [↑](#endnote-ref-29)
29. []Kim S, Kim M, Park S. Service identification using goal and scenario in service oriented architecture[C]. Proceedings of the 15th Asia-Pacific Software Engineering Conference (APSEC 2008), 2008: 419-426. [↑](#endnote-ref-30)
30. []孙纯怡, 徐福缘. 利用面向对象领域分析方法开发软件的构件与构架[J]. 上海理工大学学报, 2003, 25(2):177-180. [↑](#endnote-ref-31)
31. []吴步丹, 金芝, 赵彬. 面向服务的建模:一种全过程复用的方法[J]. 计算机学报, 2008, 31(8):1293-1308. [↑](#endnote-ref-32)
32. []Kang KC, Cohen SG, Hess JA, Novak WE. Feature-oriented domain analysis (FODA) feasibility study[M]. Pittsburgh: Carnegie Mellon University, 1990: 1-52. [↑](#endnote-ref-33)
33. []Mellor S, Scott Kendall, Uhl A, Weise D. Model-driven architecture[C]. Proceeding of the International Conference on Object-Oriented Information, Berlin: Springer-Verlag, 2002: 229-230. [↑](#endnote-ref-34)
34. []Object Management Group. MDA specifications[S]. http://www.omg.org/mda/specs.htm. (左对齐，允许断行) [↑](#endnote-ref-35)
35. []Mellor SJ, Balcer MJ. Executable UML: A foundation for model-driven architecture[M]. State of New Jersey: Addison-Wesley, 2002: 1-368. [↑](#endnote-ref-36)
36. []Steinberg D, Budinsky F, Paternostro M, Merks E. Eclipse Modeling Fram-ework2.0[M]. State of New Jersey: Addison Wesley, 2008: 1-252. (左对齐，允许断行，手动断字) [↑](#endnote-ref-37)
37. []Mazzara M, Dragoni N, and. Zhou M. Implementing work-flow reconfiguration in ws-bpel[J]. Journal of Internet Services and Information Security(JISIS), 2012, 2(1/2): 73-92. [↑](#endnote-ref-38)
38. []Cherif S, Djemaa RB, Amous I. SABPEL: Creating self-adaptive business processes[C]. Proceeding of the 14h IEEE International Conference on Computer and Information Science (ICIS 2015), Washington: IEEE Computer Society, 2015: 619-626. [↑](#endnote-ref-39)
39. []Krishnamurty V, Natarajan R, Babu C. Monitoring and reconfiguring the services in service oriented system using AOBPEL[C]. Proceeding of the 2013 International Conference on Recent Trends in Information Technology, Washington: IEEE Computer Society, 2013:423-428. [↑](#endnote-ref-40)
40. []Wang HB, Chen X, Wu Q, Yu Q, Hu XG, Zheng ZB. Integrating on-policy reinforcement learning with multi-agent techniques for adaptive service composition[J]. ACM Transactions on Autonomous & Adaptive Systems, 2017, 12(2):8:1-8:42. [↑](#endnote-ref-41)
41. []Ma Z, Zhang XG, Zhu ZY, Wang H. The implementation of adaptive web service composition[C]. Proceeding of the 2015 International Conference on Computer Science and Applications (CSA 2015). Washington: IEEE Computer Society, 2015: 303-307. [↑](#endnote-ref-42)
42. []Ezenwoye O, Sadjadi SM. Enabling robustness in existing BPEL processes[J].Proceeding of the 8th International Conference on Enterprise Information Systems(ICEIS 2006),2006:95-102. [↑](#endnote-ref-43)
43. []Ezenwoye O, Sadjadi SM. RobustBPEL2: Transparent autonomization in business processes through dynamic proxies[C]. Proceeding of the 8th International Symposium on Autonomous Decentralized Systems (ISADS 2007). Washington: IEEE Computer Society, 2007: 17-24. [↑](#endnote-ref-44)
44. []Kang KC, Kim S, Lee J,Kim K,Shin E, Huh M. FORM: A feature-oriented reuse method with domain-specific reference architectures[J]. Annals of Software Engineering, 1998, 5(1):143-168. [↑](#endnote-ref-45)
45. []Zhang HY, Jarzabek S. An XVCL-based Approach to Software Product Line Development[C]. Proceedings of the 15th International Conference on Software Engineering & Knowledge Engineering (SEKE 2003), 2003: 267-275. [↑](#endnote-ref-46)
46. []张伟, 梅宏. 一种面向特征的领域模型及其建模过程[J]. 软件学报, 2003, 14(8): 1345-1356. 左对齐 [↑](#endnote-ref-47)
47. []He X, Fu YM,Sun CA, Ma ZY, Shao WZ. Model-Driven variability-vased flexible service compositions[C]. Proceeding of the 39th IEEE Annual Computer Software and Applications Conference (COMPSAC 2015), Washington: IEEE Computer Society, 2015: 298-303. [↑](#endnote-ref-48)
48. []邢岩, 谷放, 梅宏. 特征模型驱动的Web Services组装方案及其工具支持[J]. 软件学报, 2007, 18(7):1582-1591. [↑](#endnote-ref-49)
49. []Milanovic, N, Malek M. Current solutions for web service composition[J]. Internet Computing, 2004, 8(6): 51–59. [↑](#endnote-ref-50)
50. []薛铁恒. 基于VxBPEL的适应性服务组装平台的研究与实现[D]. 北京：北京科技大学, 2013: 1-68. [↑](#endnote-ref-51)
51. []Bowen TF, Dworack FS, Chow CH, Griffeth N, Herman GE, Lin YJ. The feature interaction problem in telecommunications systems[C]. Proceeding of the 7th International Conference on Software Engineering for Telecommunications Switching Systems, London: Institution of Electrical Engineers, 1989: 59−62. [↑](#endnote-ref-52)
52. []Davis AM. The design of a family of application-oriented requirements languages[J]. IEEE Computer, 1982,15(5):21−28. 左对齐 [↑](#endnote-ref-53)
53. []Zhang W, Mei H, Zhao H. A feature-oriented approach to modeling requirements dependencies[C]. Proceeding of the 13th International Conference on Requirements Engineering, Washington: IEEE Computer Society, 2005: 273-282. [↑](#endnote-ref-54)
54. []于文静, 赵海燕, 张伟, 金芝. 基于特征模型的软件产品自动导出方法综述[J]. 软件学报, 2016, 27(1):26-44. [↑](#endnote-ref-55)