目录

1. 背景介绍

近年来，随着软件开发技术的快速发展，面向服务的架构（SOA）渐渐地成为异构环境下分布式应用程序开发的一种主流方法学。SOA 是一种粗粒度、松耦合的服务架构，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，在异构环境中开发人员通过服务可快速地进行应用程序的设计与开发。

Web 服务是 SOA 架构的典型实现方式。由于单个的 Web 服务所提供的功能有限，开发人员需要将多个 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）。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，通过服务组装能够提高更加复杂、灵活度更高分布式系统的开发效率。

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。例如，当前系统的一个或者多个服务由于某种原因暂时失效，系统将面临不可用的风险。为了在保证服务质量的同时，能够快速应对变化的业务需求和各种异常情况，该系统必须具有一定的灵活性和适应性对这些变化做出快速反应。

1.1 服务组装语言

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。该语言通过服务编制的方式实现 Web 服务组合。WS-BPEL 将多个 Web 服务组织起来，并提供一个总控流程，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可变性设计。WS-BPEL 的局限性主要表现在以下两点：

（1）使用标准的 WS-BPEL 语法所描述的服务组装，一旦部署到 WS-BPEL 执行引擎以后，流程设计人员如果由于某种原因需要更新业务流程的功能或者替换某失效服务，需要重新修改业务流程逻辑并且在修改完成后重新部署。这将导致服务组装系统开发效率低，业务流程的重复部署也增加了开发成本。

（2）WS-BPEL 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。这种应对方式将导致流程

设计中出现大量的选择分支，过于复杂的业务流程会降低系统的正确性和可维护性。

为改善传统服务组装技术对服务组装系统的灵活性和适应性支持能力的不足，我们在前期的工作中探讨了如何从服务组装的规格说明层考虑适应性问题，提出了一种基于可变性管理的适应性服务组装方法。该方法通过扩展标准的 WS-BPEL 语言开发了支持可变性表达的服务组装语言 VxBPEL。

VxBPEL 语法与 WS-BPEL 基本一致，提供变体、变异点、实现关系等可变性构造子。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。变异点定义变化可能发生的位置，包含多种可相互替代的行为，每个可被替代的行为都被封装到变体元素中。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。

VxBPEL 从服务组装的规格说明层考虑可变性问题，弥补了标准 WS-BPEL 对系统可变性支持上的不足。但仍然是在设计层考虑可变性和适应性，缺乏在需求层次对于可变性的考虑，应对业务需求变换的能力不足。

1.2 可变性管理

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。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。变异点是对发生变化位置的抽象，变体是对备选方案的抽象。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，包含可变性设计的服务组装能够通过选择每个变异点下的变体来响应需求或环境的变化。

1.3 领域分析

“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，领域又称为程序家族或软件产品线。领域工程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，为某一特定问题空间设计和实现可复用软件制品的过程，它的最高目标是实现系统化的软件重用。

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。领域分析作为领域工程的起始阶段，与需求分析区别在于领域分析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，结合可预见的领域功能演化，总结领域的问题空间，在问题空间内区分功能性需求和非功能性需求，并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，也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、面向对象的领域分析方法、基于本体的方法等。Kang 等人于 1990 年提出了面向特征的领域分析方法，被公认为是最实用的领域建模技术。

1.4 模型驱动

随着当前软件系统越来越庞大，复杂度也变得越来越高。软件系统的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，那么新的业务变更必然会造成程序代码的修改。由于代码的非直观性和复杂性等缺陷，修改代码将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、代码实现）间产生不一致，严重影响所开发系统的正确性和开发效率。

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。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，许多研究机构和大型公司提出各自的模型驱动开发方法，包括对象管理组织提出的模型驱动架构、统一建模语言和 Eclipse 建模架构等。

课题组在前期工作中提出了基于可变性管理的适应性服务组装方法。该方法通过扩展 WS-BPEL 开发了支持可变性表达的服务组装语言 VxBPEL，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。在上述研究的基础上，课题组提出了面向全生命周期的基于可变性管理的适应性服务组装方法。为了高效地运用提出的方法，可提出开发相应的支持系统 FM2VxBPEL。

2. FM2VxBPEL 系统的分析与设计

2.1 需求分析

传统服务组装的开发过程如图 1 所示，主要包括流程建模、流程组装、流程部署和流程运行四步。但是目前还没有一套相对成熟的方案用于指导服务组装的整个开发过程。

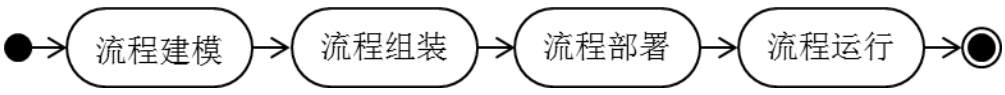

图 1 服务组装开发过程活动图

当前基于可变性管理的适应性服务组装方法仅在设计层考虑可变性和适应性，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。该方法缺乏在需求层对于可变性的考虑，整个服务组装的开发过程缺乏需求分析的指导。

针对上述问题，我们提出了一种面向全生命周期的适应性服务组装方法，并开发了相应的支持平台 FM2VxBPEL。该方法在需求层面考虑可变性问题，引入了领域特征模型和模型驱动技术，定义了服务组装系统领域的特征模型。该方法通过特征模型进行需求建模，以显示化的表达领域的共性需求、差异性需求和约束关系，并利用 FM2VxBPEL 将特征模型转换为设计阶段的抽象服务组装模型，

最终实现由特征模型驱动服务组装的可变性设计。

FM2VxBPEL 是一个特征模型驱动的适应性服务组装构造支持系统，它的核心功能是实现模型转换，同时该系统能够帮助需求分析人员刻画特征模型图，并解析成可配置的特征模型树，帮助用户实现需求的可视化定制。该系统还可以根据用户对特征模型定制的结果创建用户配置文件、模拟 SOAP 客户端执行测试用例。FM2VxBPEL 系统的用例图如图 2 所示。

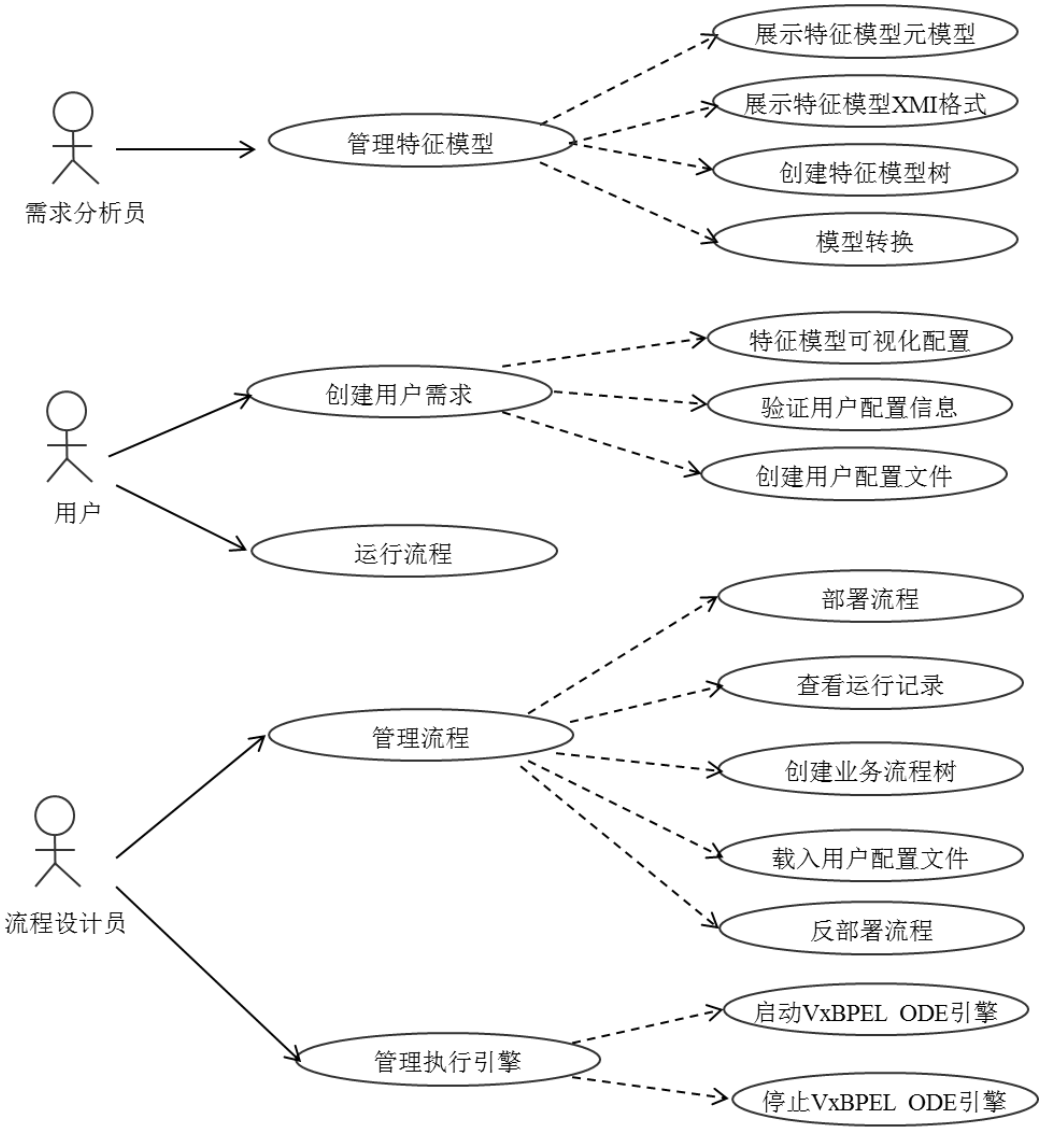

图 2 FM2VxBPEL 系统用例图

该系统的用例说明如下：

(1) 管理特征模型：需求分析人员能够通过该系统对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。

1) 展示特征模型元模型：该用例将特征模型展示出来，主要为了帮助需求分析人员能够正确的理解特征模型。
2) 展示特征模型 XMI 格式：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 系统上传该 XMI 文件，上传过程中的路径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。
3) 创建特征模型树：需求分析人员将特征模型文件上传到该系统后，该系统将会自行解析特征模型文件，并创建特征模型树。特征模型树能够表达领域内的共性需求和差异性需求，该系统允许用户对特征模型树进行配置。
4) 模型转换：该用例帮助需求分析人员将特征模型转换为基于 VxBPEL 的抽象服务组装模型。经过 FM2VxBPEL 系统转换后得到的模型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，流程设计人员后续仍需要对该 VxBPEL 工程进行完整地设计。

(2) 创建用户需求：该用例帮助用户实现对需求的配置，主要包括特征模型可视化配置、验证用户配置信息以及创建用户配置文件三个子用例。

1) 特征模型可视化配置：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。
2) 验证用户配置信息：该用例验证用户对特征模型的配置信息是否符合特征模型的约束条件。
3) 创建用户配置文件：该用例帮助用户创建用户配置文件，用户配置文件中保存了用户在特征模型中所选择的特征所匹配的变异点和变体信息，同时该用例允许用户自定义用户配置文件的保存路径和文件名。

(3) 运行流程：该用例模拟 Soap 客户端，允许用户输入请求信息并执行业务流程实例，得到返回结果。

(4) 管理流程：该用例帮助流程设计人员管理业务流程，主要包括流程部署、查看部署记录、解析业务流程信息、载入用户配置文件、反部署流程五个子用例。

1) 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。
2) 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。

3) 创建业务流程树：该用例帮助流程设计人员解析指定路径下的 VxBPEL 工程文件的业务流程信息，并以树的形式展示出来。

4) 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上传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。

5) 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。

(5) 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。该用例包含启动 VxBPEL ODE 引擎和关闭 VxBPEL ODE 引擎两个子用例。

2.2 系统设计

FM2VxBPEL 系统架构主要包括特征模型管理、需求配置、流程运行、流程管理和引擎管理五个模块，其系统架构如下图 3 所示。

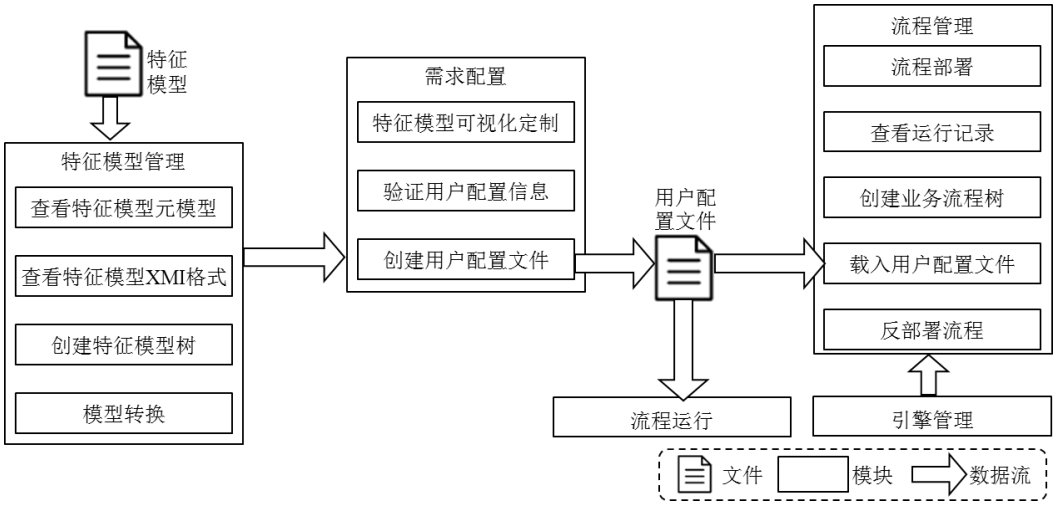

图 3 FM2VxBPEL 的系统架构

本系统使用 Java 语言开发，共计 3217 行代码，开发环境为 IDEA2017.1.4 版本，JDK 版本为 1.7.0_51，界面开发主要使用的是 Swing 技术、AWT 技术；

特征模型图生成、可配置特征模型树创建、模型转换、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。下面介绍系统中关键模块的设计。

(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图、可配置特征模型树的创建和模型转换，用到的类分别是 DrawFMDiagram、CreateFMTree 和 ModelTran。

- DrawFMDiagram 类：负责特征模型图的创建。该类会根据获取到的特征模型文件的路径信息，调用 createDocument()方法将特征模型文件转化为 Document 对象，然后调用 drawFMDiagram()方法生成特征模型图。由于特征模型以 XMI 格式存储，因此 drawFMDiagram()方法通过递归调用方式创建特征模型图。该类所创建的特征模型图结构清晰、层次分明、表达信息完整，同时该类支持需求分析人员自定义特征模型图的保存路径和图片名称。
- CreateFMTree 类：负责可配置特征模型树的创建。该类会根据获取到特征模型文件的路径信息，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 技术即可创建可配置的特征模型树。
- ModelTran 类：负责特征模型到抽象服务模型的模型转换。特征模型实例文件是以 XMI 格式进行保存，可以通过 DOM4J 技术递归遍历到 XMI 文件的每一个节点，即特征模型的每一个特征。随后该类调用 toVxBPEL()方法，toVxBPEL()方法是对算法 1 和算法 2 的具体代码实现，其本质是根据转换规则将特征模型文件的节点转换为抽象服务组装模型文件的节点，即抽象服务组装模型的活动，并由这些转换后的节点创建新的 Document 对象，将新建的 Document 对象写入到文件中即可完成抽象服务组装模型文件的保存。同时该类允许自定义抽象服务组装模型文件的保存路径和文件名。

(2) 需求配置模块

需求配置模块主要功能支持特征模型的可视化定制、验证用户配置信息、创建用户配置文件，用到的类分别是 Validate 和 CreateUCCF。

- Validate 类：负责验证用户对特征模型的配置信息是否符合特征模型的约束条件。该类借助 Java 的 ArrayList 集合类存储每一条约束的约束类型、约束的双方，然后读取每个 ArrayList 集合类元素去验证配置信息中是否

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配置。

- CreateUCCF 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 技术将由特征模型配置信息匹配到的可变性配置信息写入到用户配置文件中。用户配置文件内容以 XML 语法进行描述。

(3) 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、创建业务流程树、载入用户配置文件，用到的类主要包括 BusinessManager、ProcessTree。

- BusinessManager 类：负责业务流程的部署和反部署。该类通过复用 Apache ODE 自身提供的流程管理系统，调用 openDefaultBrowser()方法。openDefaultBrowser() 方法使用当前系统的默认浏览器打开“http://localhost:8080/ode/deployment.html”页面进行业务流程的管理，实现业务流程部署和反部署。
- ProcessTree 类：负责创建业务流程树。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，结合 DOM4J 解析技术和 JTree 技术递归遍历流程文件，得到创建业务流程树。

(4) 流程运行模块

流程运行模块负责模拟 Soap 客户端，由用户对业务流程进行调用。用到的类 SoapAction 类。

- SoapAction 类：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。

(5) 引擎管理模块

引擎管理模块负责 Apache ODE 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。

3. FM2VxBPEL 系统的实现与演示

3.1 系统功能

特征模型驱动的适应性服务组装构造支持系统 FM2VxBPEL 的主要功能包括：模型转换、用户需求配置、管理流程、运行流程、管理引擎：

(1) 模型转换：面向全生命周期的适应性服务组装方法在开发过程中引入了领域分析。需求分析人员根据领域知识进行需求建模得到领域的特征模型实例。该特征模型实例是一个静态、抽象的需求表示，无法与 VxBPEL 的抽象服务组装模型建立联系。为此 FM2VxBPEL 根据方法中提出的模型转换算法实现了由特征模型到抽象服务组装模型的转换。该模型转换算法的核心思想是将不同类型的特征映射为抽象服务组装模型中相应的活动。

(2) 用户需求配置：在特征模型到抽象服务组装模型转换的同时，用户可以使用 FM2VxBPEL 系统对领域的特征模型进行可视化定制，创建用户的特征配置模型。特征配置模型表达单个用户的个性化需求，即用户根据自身需求对可选特征进行选择。同时该系统还可以验证配置信息的正确性，判断是否符合特征模型的约束条件。

(3) 流程管理：管理功能包括五部分：

1) 流程部署：将完整 VxBPEL 工程部署到服务器端。
2) 部署记录：查看流程部署和执行日志。
3) 流程解析：解析 VxBPEL 工程中的业务流程信息，生成业务流程树。
4) 载入配置：载入用户的特征配置文件。
5) 流程反部署：现有流程不能满足需求、失效或因其他原因需要卸载时，通过反部署功能从服务器端删除。

(4) 运行流程：该功能会执行 VxBPEL ODE 引擎，根据用户的特征配置信息派生出具体的流程实例，同时模拟 SOAP 客户端，允许用户输入请求信息，并得到返回结果。

(5) 引擎管理：快速启动或停止 VxBPEL ODE 引擎。

3.2 系统演示

本节采用汽车组装生产线作为实例，演示 FM2VxBPEL 系统的使用。对于

厂商来说，一个汽车生产线会根据需求进行多种配置的车型的组装，不同配置车型的生产装配过程存中也存在大量相同的操作。如果为不同配置的车型开发相应的专属控制流程，则不可避免的会产生大量冗余，进而占用系统资源，增加复杂度，影响生产效率。

通过对汽车组装领域进行需求分析，得到其特征模型如图 4 所示：一辆完整的汽车必须由引擎、车身、底盘、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，作为一个可选特征存在于领域当中。对于车身特征来说，其具有三个属性：车身类型、颜色和材质，这些都是车身的必选特征。对于颜色，则是必须要在可选的颜色特征中选择一种。同理还有传动系统和变速器。为了能体模拟特征模型中的约束关系，我们假设：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；如果用户选择了黑颜色的车身，则变速器实现方式不能选择手自一体。

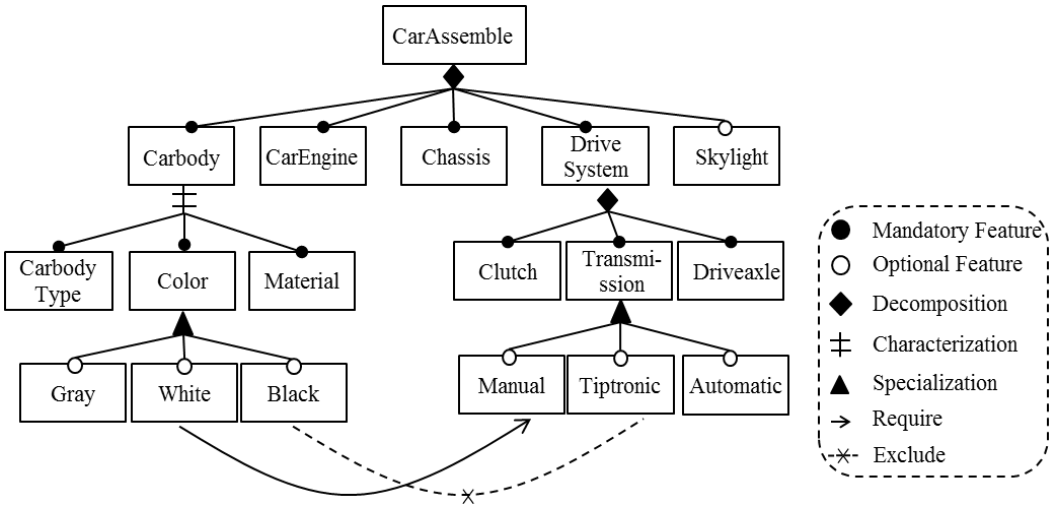

图 4 汽车组装领域特征模型树状图

(1) 实验环境及配置

表 5 展示了 FM2VxBPEL 系统的实验环境和配置参数，运行该系统还需在 window7-64bit 系统上部署 Apache ODE 引擎。

表 5 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

(2) FM2VxBPEL 系统首页

如图 6 所示，FM2VxBPEL 系统首页包含菜单栏、文件选择区、功能区和日志区四部分。

菜单栏包含“File”、“Engine”、“Run”、“Tools”、“Help”五个菜单。“File”菜单下“Close”子菜单用来关闭该系统；“Engine”菜单下“Start”子菜单和“Stop”子菜单支持当前系统环境变量设置下的执行引擎的开启和停止；“Run”菜单下的“Run Process”子菜单模拟 Soap 客户端对服务组装实例发起请求；“Tools”菜单下的“Business Manage”子菜单用来调用 Apache ODE 自身提供的流程管理系统实现业务流程的部署和反部署；“Help”菜单下的“FMMetaModel”子菜单用来展示特征模型的 Ecore 模型图。文件选择区可以选择加载特征模型文件的位置和新文件保存的位置。功能区针对不同的功能可分别实现特征模型的可视化定制、模型转换、可变性信息解析和用户配置文件生成等功能。日志区可以展示特征模型 XMI 文件和系统执行过程中的日志信息。

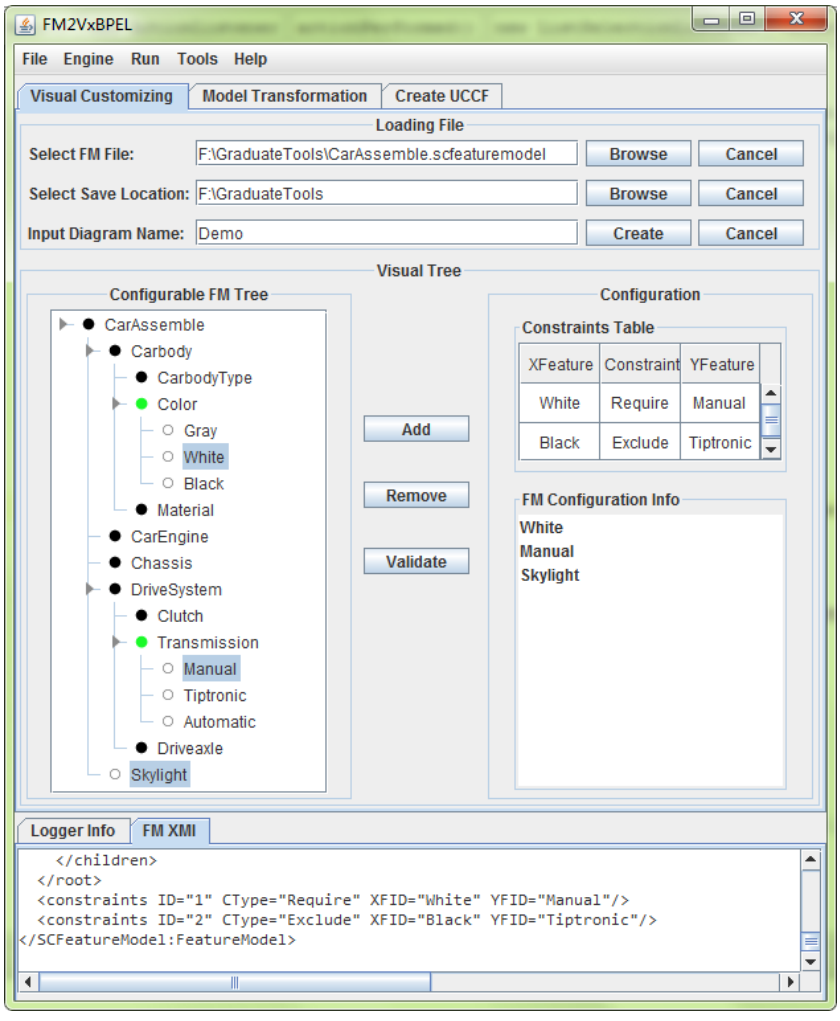

图 6 特征模型可视化配置界面

(3) 特征模型的可视化定制

图 6 功能区所示是 FM2VxBPEL 系统特征模型可视化定制界面。首先，需求分析人员在文件选择区上传特征模型文件到该系统中，并且依次选择并输入特征模型图的保存路径和文件名，点击“Create”按钮，该系统将会解析并弹出如图 7 所示的特征模型图，并创建可配置的特征模型树。特征模型实例的约束关系将会展示在功能区右侧“Configuration”区域，XMI 内容将会展示在日志区“FM XMI”窗口。

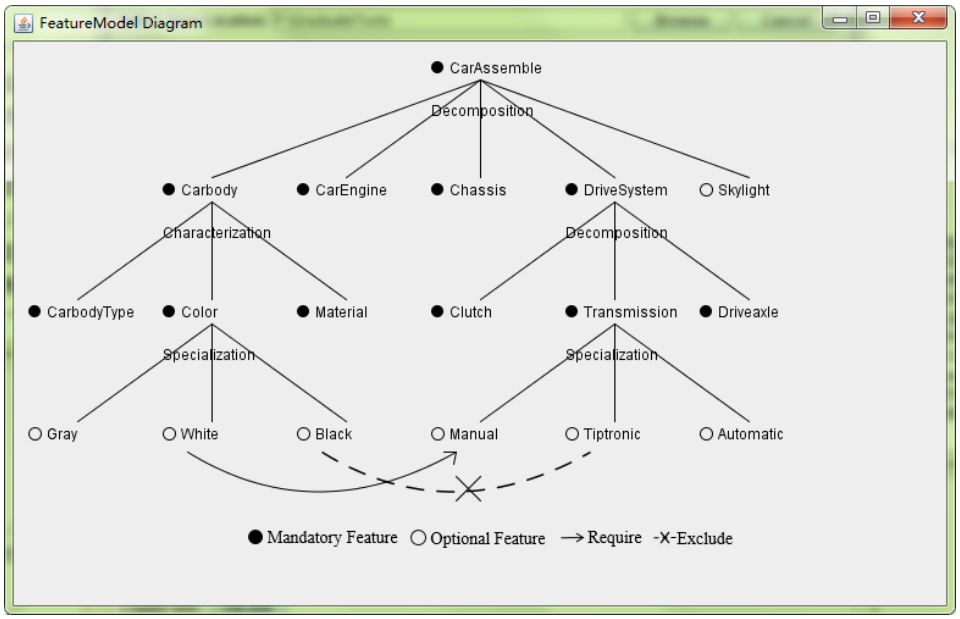

图 7 特征模型图

在功能区左侧的可配置特征模型树中，黑色圆圈表示强制特征；白色圆圈表示可选特征；绿色实心圆圈表示该特征为强制特征，并且该特征与其子特是特化关系；绿色空心圆圈表示该特征为可选特征，并且该特征与其子特是特化关系。用户可以通过“Add”和“Remove”按钮对可配置的特征模型树中的可选特征进行选择，选择的结果将会展示在“FM Configuration Info”区域。用户配置完成后，点击“Validate”按钮验证所选择特征是否符合特征模型实例的约束关系，只有符合约束关系才可以进行下一步操作，否则将会弹出错误提示框。

(4) 模型转换与可变性信息的解析

如图 8 所示是 FM2VxBPEL 系统的模型转换与可变性信息解析界面。在该界面，通过文件选择区上传特征模型的实例文件，并自定义经过模型转换后得到的抽象服务组装模型文件的保存路径和文件名，点击“Convert”按钮，即可完成模型转换。

此时得到的基于 VxBPEL 的抽象服务组装模型仍然是一个不完整的工程文件。在流程设计人员经过补充设计后，FM2VxBPEL 系统能够对服务组装系统的流程信息进行解析，也就是对 VxBPEL 工程的.bpel 文件进行解析并创建业务流程树。业务流程树功能区左侧在“Process Tree”区域展示。功能区右侧上方“FM Configuration Info”区域保存了在可视化定制界面用户对特征模型的定制信息，点击“Match”按钮，该系统将会根据用户特征模型定制信息去匹配流程树中的变异点和变体，并将可变性配置信息结果展示在功能区右侧下方的“VariationPoint && Variant Conf”区域。

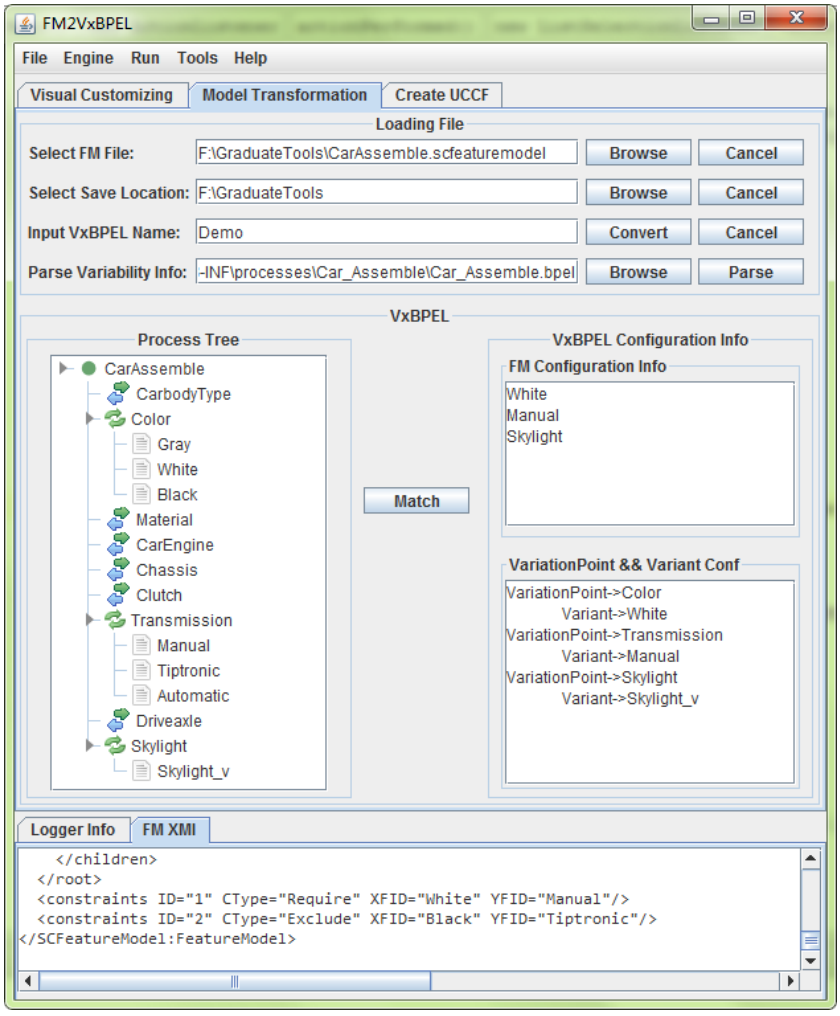

图 8 模型转换和可变性信息解析界面

(5) 用户配置文件的创建

可变性配置信息匹配完成后，在创建用户配置文件界面，用户可以再文件选择区自定义用户配置文件的路径和文件名，但是最终用户配置文件都要放在引擎中相应的应用根目录下。用户配置文件的内容将会展示在功能区“UCCF Info”区

域下，如图 9 所示。

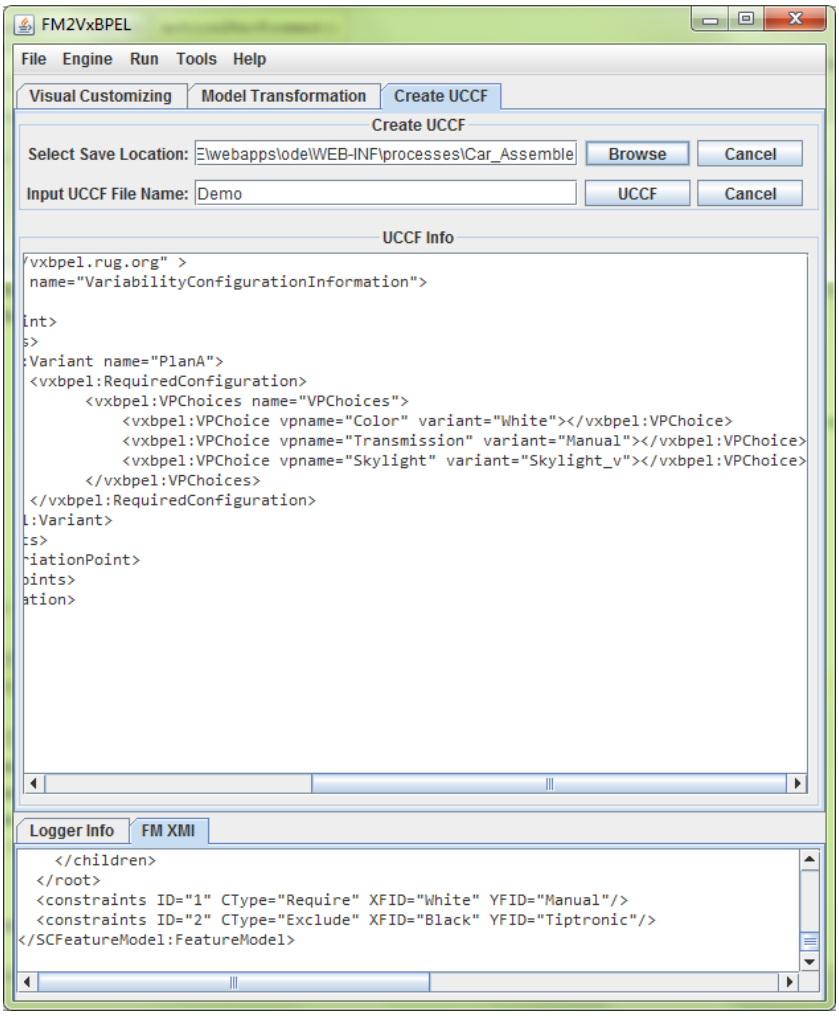

图 9 创建用户配置文件界面

(6) 模拟 Soap 客户端执行服务组装实例

用户配置文件创建完成后，点击菜单栏“Run”下“Run Process”子菜单打开模拟 Soap 客户端界面。如图 10 所示，该界面支持对 WS-BPEL 工程和 VxBPEL 工程进行服务的调用。以调用 VxBPEL 工程服务为例，在该界面加载 VxBPEL 工程的 WSDL 文件，点击“VxBPEL”按钮，在“Request”区域输入用户请求信息。其中“userId”即为用户配置文件的文件名。用户请求信息输入完成后，点击“Run”按钮，执行引擎会根据请求信息执行业务流程，得到响应消息以及响应时间（Response Time），响应消息和响应时间并展示在“Result”区域。

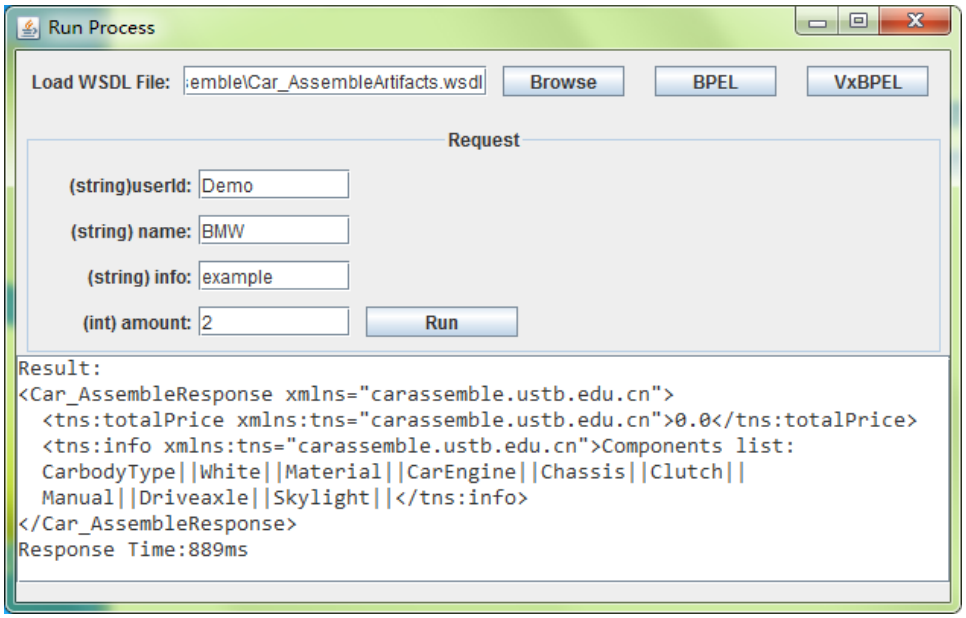

图 10 模拟 Soap 客户端执行业务流程界面