模型驱动的构件组装的研究与实现①

Research and Implementation of Model Driven Component Composite

杨兴涛[1]　苏桂平[2]　王瑞芳[3]　王小芳[4]

1 (中国科学院研究生院 工程教育学院 北京 100049)

2 (中国科学院研究生院 信息科学与工程学院 北京 100049)

3 (大连大学 信息工程学院数学系 辽宁 大连 116622)

4 (吉林大学 数学研究所 吉林 长春 130012)

摘　要： 构件组装是提高软件开发效率与产品质量的有效途径。本文在深入研究构件组装与 DSM 以及代码生成等技术后，提出一种普适性的构件组装机制，建立起构件与连接件元模型，并对它们的性质进行了归纳总结，在此基础之上建立起元模型的实现，并介绍了以构件与连接件元模型为输入，通过 GMF 快速配置生成构件组装建模工具的过程。最后，提出一种代码模板制作与代码生成的方法与实现，并介绍了通过构建组装模型与代码模板生成完整的可运行程序代码的过程。

关键词： 构件 构件组装 连接件 特定领域建模 元模型 代码模板 代码生成

1　引言

构件组装是将传统产业的生产模式，即符合标准的零部件(构件)生产以及基于标准零部件的产品生产(组装)，运用到软件开发领域，促使软件开发向工业化生产方向发展的先进思想[1]。它以构件组装的方式快速构建软件系统，以构件增减的方式快速响应需求的变化。

构件组装是提高软件开发效率与产品质量的有效途径[1]。构件是系统构成成分的标准形态，组装是构件演化的原则与方式，以及自底向上的系统构建方法。构件组装成功运用的前提是标准化的构件与构件组装机制。代码级构件无法直接在接口处显式定义组装[2]，所以构件组装模型是实现构件组装显式定义的辅助、也是必需的手段。目前，构件组装的研究主要集中在软件体系结构与构件组装机制上，但对于如何从高层构件组装模型产生完整的程序代码尚没有系统化的指导方法。

本文在深入研究构件组装与 DSM(Domain-Specific Modeling)以及代码生成等技术后，提出一种普适性的构件组装机制，建立起构件与连接件元模型，并对它们的性质进行了总结归纳，在此基础上建立起元模型的实现，并介绍了以构件与连接元模型为输入，通过 GMF(Graphical Modeling Framework)快速配置生成构件组装建模工具的过程。最后，提出一种代码模板制作与代码生成的方法与实现，并介绍了通过构建组装模型与代码模板，生成完整的可运行程序代码的过程。本文通过构建组装建模与代码生成，实现从高层的构件组装建模到程序代码的完整映射。

2　构件元模型及特性

2.1 构件元模型

构件元模型是对构件及其特性的高层描述，关注构件作为零部件进行组装的特性，而并不关心其内部实现。为实现高层抽象组装，通常对每个代码级构件都建立构件模型，用以指导组装。

如图 1 所示，本文提出一种构件元模型，包括如下 5 部分内容：

(1) Descriptions，构件描述信息，包括名称、类

① 基金项目:国家 863 计划项目(2001AA141010)
收稿时间:2008-10-09

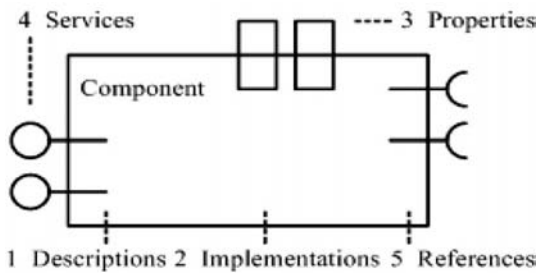

图 1 构件元模型

型、状态、功能/性能/执行环境说明、作者、修改日志等内容。

(2) Implementations，构件实现信息，包括技术类型、引用位置、内容、说明等内容。

(3) Properties，构件属性信息，包括名称、类型、值、说明等内容。

(4) Services，构件提供服务接口，包括名称、类型、接口规格、功能说明等内容。

(5) References，构件引用服务接口，包括名称、类型、接口规格、功能说明等内容。

2.2 构件元模型的特性

本文提出的构件元模型具备如下特性:

(1)构件是系统构成成分的唯一形态

构件不区分原子构件与复合构件，只在粒度或复杂度上存在差别。通常由小构件组装成大构件，由简单构件组装成复杂构件，形成系统的结构层次。

(2)服务接口是构件与外界交互的唯一途径

构件通过服务接口封装内部实现，并定义与外界交互的全部内容。构件通过属性标识构件状态，通过与其它构件交互而使状态产生变化。

(3)构件模型与代码级构件一一对应

构件实现信息描述对已有代码级构件的引用或具体代码模板的绑定。对尚未建立的代码级构件可以通过代码生成或编程手段进行创建。

(4)构件建立的正交性与完备性

正交性指某构件发生变化而不会影响其它构件。完备性指建立的构件应该是清晰、简单、充分、完整的。

2.3 构件元模型的的实现

与 MDA 相对，DSM 从大幅度提高软件生产力的角度提出特定领域建模，它强调“是否领域特定，是否能生成 100%的代码”。但 DSM 没有提供具体的技术实现。本文使用 EMF(Eclipse Modeling Framework)创建构件元模型，作为 GMF 配置生成构件组装建模工具的输入。EMF 支持 3 种类型输入，本文使用 XML 模式。

本文使用 XML 模式描述构件元模型，具体如下:

(1)分别将构件元模型的 5 个组成部分描述为 XML 模式的 Element。

(2)分别将每个组成部分的具体内容描述为 Attribute。

(3)分别为 Element 与 Attribute 定义简单或复杂类型。

3 构件组装机制与连接件元模型

SA(Software Architecture)将系统的整体结构作为研究对象，在高层显式的描述系统构成元素及其之间的交互关系[3]。从构件组装的角度看，SA 描述的是软件系统的基本组织，包括构件、构件之间、构件与环境之间的关系及相关设计与演化原则。SA 研究的贡献在于将构件之间的交互显式的定义为连接件[3]，使得接口不匹配的构件之间可以通过连接件进行组装，发展了接口匹配的构件组装。

3.1 构件组装机制

构件组装就是将被组装构件的多个服务接口绑定到组装后的构件的服务接口上，并建立起被组装构件的多个服务接口之间的交互方式。它的本质就是在构件之间通过接口或连接件建立关联并协调行为[4]。接口连接发生在对偶接口之间，其实是连接件连接的特殊情况。连接件是构件组装的基础，构件组装机制反映的是连接件所表现的构件之间的交互方式。

本文提出 4 种具有普适性的基于连接件连接的构件组装机制，如下所示。

(1)顺序组装

构件 A 与构件 B 顺序组装，先执行 A。若 A 与 B 无依赖关系，则先执行 A，再执行 B；若 A 与 B 存在依赖关系且 A 依赖于 B，则在调用 A 后，先执行 B 以满足 A 的执行条件，再执行 A，最后执行 B；若 A 与 B 存在依赖关系且 B 依赖于 A，则先执行 A，再调用 B，然后执行 A 以满足 B 的执行条件，最后执行 B。

(2)选择组装

构件 A 与构件 B 选择组装。若 A 与 B 间无依赖关系，则根据条件选择执行 A 或 B；若 A 与 B 间存在依赖关系，则根据条件选择调用 A 或 B 后，先执行 B 或 A 以满足 A 或 B 的执行条件，再执行 A 或 B。

(3)循环组装

构件 A 与构件 B 循环组装，先执行 A。若 A 与 B

无依赖关系，则根据循环条件先执行 A，再根据循环条件执行 B；若 A 与 B 存在依赖关系且 A 依赖于 B，则在调用 A 后，要先根据循环条件执行 B 以满足 A 的执行条件，再根据循环条件执行 A；若 A 与 B 存在依赖关系且 B 依赖于 A，则根据循环条件先执行 A，则在调用 B 后，要先根据循环条件执行 A 以满足 B 的执行条件，再根据循环条件执行 B。

(4)并行组装

构件 A 与构件 B 并行组装。若 A 与 B 无依赖关系，则并行执行 A 与 B；若 A 与 B 存在依赖关系且 A 依赖于 B，则并行调用 A 与 B 后，先执行 B 以满足 A 执行的条件，再并行执行 A 与 B；若 A 与 B 存在依赖关系且 B 依赖于 A，则并行调用 A 与 B 后，先执行 A 以满足 B 执行的条件，再并行执行 A 与 B。

在本文提出的上述构件组装机制中，构件之间的依赖关系指某构件引用另外构件所提供的服务。依赖关系是隐藏且默认的构件之间的交互关系，组装机制是显式定义的构件之间的交互关系。

3.2 连接件元模型

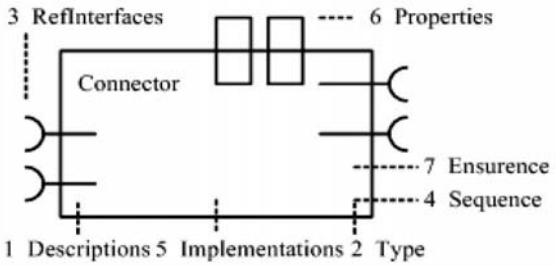

图 2　连接件元模型

针对上述 4 种构件组装机制，如图 2 所示，本文提出了连接件元模型，包括如下 7 部分内容：

(1)Descriptions，连接件描述信息，包括名称、类型、功能说明等内容。

(2)Type，连接件类型信息，包括顺序、选择、循环、并行 4 种类型。

(3)RefInterfaces，连接件所连接的构件服务接口，包括名称、类型、接口规格、功能说明等内容。

(4)Sequence，构件组装执行顺序，根据连接件类型设定构件之间的执行顺序，包括默认与条件执行顺序。

(5)Implementations，连接件实现信息，包括实现技术类型、引用位置、内容、说明等内容。

(6)Properties，连接件属性信息，包括名称、类型、值、说明等内容。

(7)Ensurence，构件组装保障机制，包括数据转换、事务、加密/解密、日志等内容。

3.3 连接件元模型的特性

本文提出的连接件元模型具备如下特性：

(1)连接件短暂的生命周期

连接件是构件组装的具体表现，粘接代码是连接件的最终实现。在构件组装产生新构件后，新构件期待被再次组装。所以在构件组装产生新构件后，连接件的生命周期即告结束。连接件只存在于构件组装的短暂过程中。

(2)连接件模型绑定代码模板

连接件实现信息描述连接件模型对代码模板的绑定，而对不同 SA 风格与平台环境及具体构件组装保障机制的适应制作入代码模板，实现连接件模型与具体实现技术的解耦。

(3)构件组装执行顺序描述构件组装的控制结构

根据连接件类型，设定所连接构件之间的执行顺序，并设定执行条件，如选择执行条件、循环执行条件等。

(4)构件组装保障机制描述保障构件组装的数据流转

数据转换保障流转数据对构件服务接口的匹配，事务保障多个构件处于同一个事务中，加密/解密保障流转数据的安全性等。

3.4 连接件元模型的的实现

与构件元模型的实现相同，本文使用 XML 模式描述连接件元模型，具体如下：

(1)分别将连接件元模型的 7 个组成部分描述为 XML 模式的 Element。

(2)分别将每个组成部分的具体内容描述为 Attribute。

(3)分别为 Element 与 Attribute 定义简单或复杂类型。

4　基于元模型创建构件组装建模工具

目前，通常使用 GMF 实现 DSM 中的 DSL (Domain-Specific Language)及图形化编辑器，即特定领域建模工具。GMF 提供图形化定制环境，开发人员可在图形环境中快速轻松定制生成自己的特定领域建模工具。

本文以上面所建立的构件与连接件元模型 XML 模式描述为输入，使用 GMF 建立构件组装建模工具，具体如下：

(1)以 XML 模式为输入，通过 GMF 中的 EMF 生成构件与连接件元模型 Ecore。

(2)建立图形定义模型，输入元模型 Ecore，定制图形内容，生成.gmfgraph 文件。

(3)建立工具定义模型，输入图形定义模型，定制工具内容，生成.gmftools 文件。

(4)建立映射定义模型，输入元模型 Ecore、图形定义模型、工具定义模型，定制 Ecore 与图形工具之间的映射，生成.gmfmap 文件。

(5)通过映射定义模型，自动生成完整的构件组装建模工具。

由 GMF 建立的构件组装建模工具具有图形化可拖拽的人机界面，可以非常方便的进行构件组装建模。

5 代码模板制作与代码生成

XSLT(eXtensible Stylesheet Language Trans-formation) 由 XSL(eXtensible Stylesheet Language)发展而来，是一种用于将 XML 文档转换成 XHTML 文档或其他文档的语言。目前广泛应用于代码生成领域。

本文使用 XSLT 实现构件组装的代码生成，如图 3 所示。

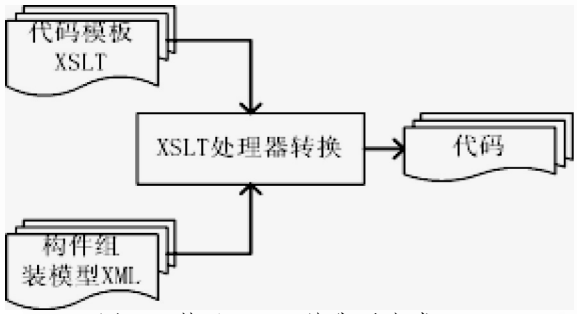

图 3 基于 XSLT 的代码生成

根据构件与连接件元模型，使用 XSLT 技术制作代码模板，并以构件组装模型作为代码模板的数据源，通过 XSLT 处理器转换形成代码。

本文使用 XSLT 制作代码模板的具体过程如下：

(1)分析设计系统架构，适应不同 SA 风格与平台环境，选择架构支撑技术。

(2)设计实现系统框架，抽象系统共性部分，建立起不同业务类型与表现风格类型的可运行样例代码。

(3)根据构件与连接件元模型，将样例代码使用 XSLT 技术制作成业务代码模板。

样例代码是代码模板制作的基础，所以样例代码应尽量完整且覆盖不同的业务类型与表现风格。为生成完整的可运行代码，代码模板制作应参照完整的样例代码。

代码模板制作应遵循如下原则：分离代码中的不变与变化部分，将不变的规则形成代码模板中的固定代码，将变化的业务对象以模板语言进行标记，即不变的部分为规则或逻辑，变化的部分为业务对象。

本文的构件组装代码模板包括：构件代码模板与连接件代码模板。代码模板均需要支持不同 SA 风格与平台环境。连接件代码模板还需要支持如过程调用、事件、流、适配器等类型的连接，同时结合系统框架支持构件组装保障机制。

本文使用 Apache Xalan 作为构件组装代码生成的具体 XSLT 处理器。Apache Xalan 可以通过命令行的形式方便的进行代码生成，如下：

java org.apache.xalan.xslt.Process -in xml -Source -xsl stylesheet -out outputfile

其中，xmlSource 指构件组装模型，stylesheet 指代码模板，outputfile 指生成的代码。

6 结束语

本文实现了从构件组装的高层抽象模型与技术实现之间的完整映射。但影响本文代码生成质量的主要因素在于代码模板对不同业务类型以及 SA 风格与不同平台环境的覆盖度上。如果覆盖度较低，则需要耗费时间来扩展代码模板。

模型验证与模拟是保证高质量建模的有效措施。模型验证是检测模型是否满足模型约束的技术，通常有语法检查、逻辑验证、状态验证、时态验证等。模型模拟是以模型验证为基础对模型进行运行态模拟，分析系统的动态行为，在高层指导验证系统设计，并进行错误分析。这将是本文以后研究的方向。

参考文献

1 杨芙清,王千祥,梅宏,陈兆良.基于复用的软件生产技术.中国科学(E 辑), 2001,31(4):363－371.

2 张世琨,张文娟,常欣,王立福,杨芙清.基于软件体系结构的可复用构件制作和组装.软件学报, 2001, 12(9):1351－1359.

3 梅宏,陈锋,冯耀东,杨杰.ABC:基于体系结构、面向构件的软件开发方法.软件学报, 2003,14(4):721－732.

4 任洪敏,钱乐秋.构件组装及其形式化推导研究.软件学报, 2003,14(6):1066－1074.

5 许毅,彭鑫,赵文耘.基于通用连接器模型的复合构件的组装.计算机工程, 2006,32(23):55－57.