一种支持软件体系结构重用的元信息模型

叶 鹏[1,2] 应 时[1,3] 袁文杰[1] 姚俊峰[1] 罗巨波[1]
(武汉大学软件工程国家重点实验室 武汉 430072)[1] (武汉科技学院计算机科学学院 武汉 430073)[2]
(武汉大学计算机科学学院 武汉 430072)[3]

摘 要 软件体系结构元信息组织和管理是利用反射机制实现软件体系结构重用的一个重要问题。提出了一种体系结构元信息模型,以有效地组织和管理支持体系结构重用的元信息。同时,基于XML设计了一种体系结构元信息描述语言,以充分地描述这个模型中定义的各种元信息,达到体系结构设计人员可以在软件设计阶段通过操作这些元信息高效地重用软件体系结构的目的。

关键词 软件体系结构,重用,元信息模型,反射,XML
中图法分类号 TP311 **文献标识码** A

Meta Information Model for Reusing Software Architecture

YE Peng[1,2] YING Shi[1,3] YUAN Wen-jie[1] YAO Jun-feng[1] LUO Ju-bo[1]
(State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)[1]
(College of Computer Science, Wuhan University of Science and Engineering, Wuhan 430073, China)[2]
(College of Computer Science, Wuhan University, Wuhan 430072, China)[3]

Abstract A critical issue for realizing reuse of software architecture by reflection mechanism is to organize and manager meta-information of software architecture. This paper argued for a meta-information, which can effectively organize and manager meta-information for reusing architecture. At the same time we utilized XML to design an architectural meta-information description language which is able to sufficiently describe all kinds of meta-information defined in this meta-information model in order that architects could accomplish the reusing of software architecture artifacts efficiently at software design stage, by manipulating meta-information.

Keywords Software architecture, Reuse, Meta-information model, Reflection, XML

软件重用是提高软件开发效率和软件产品质量的有效途径[4]。近些年来,随着组件技术的出现,软件重用技术取得了长足的进步。人们在重用代码组件方面已取得了很大的成功,但在设计阶段重用大粒度软件资源(如软件体系结构、软件模型等)方面仍进展缓慢[3]。目前,在设计阶段重用软件体系结构所面临的主要问题是:虽然存在大量的具有重用价值的体系结构制品(通常是使用特定ADL描述的软件系统的总体设计方案),但是这些制品却难以得到成功的重用。我们认为难以显式地描述并使用支持体系结构制品重用操作和过程的信息,缺乏有效的重用方法,是导致这个问题的最根本原因。

为了解决这个问题,达到在设计阶段有效地重用体系结构制品的目的,本文基于体系结构反射[8],提出了一种支持体系结构重用的元信息模型MIM4RSA(Meta Information Model for Reusing Software Architecture)。对于软件体系结构制品及其各种组成元素,MIM4RSA模型系统地定义了支持体系结构制品重用过程和各种重用操作所需的各种元信息。同时,为了能让计算机处理MIM4RSA中定义的元信息,使得体系结构设计人员能使用软件工具操作这些元信息,设计出新的体系结构,我们基于XML设计了体系结构元信息描述语言MetaADL来描述这些元信息。

本文第1节介绍相关知识,第2节概述MIM4RSA模型的设计,第3节介绍MetaADL,第4节进行案例研究,最后总结本文的贡献和今后要做的研究工作。

1 相关知识

1.1 体系结构反射

体系结构反射是由一个系统执行的关于它自身体系结构的计算,这个计算过程在使用体系结构元信息的基础上进行。软件体系结构可以利用体系结构反射进行说明[10]。基于体系结构反射机制构造的软件体系结构称为反射式体系结构。反射式体系结构由两部分组成:一个是基级,一个是元级[8]。

到稿日期:2008-07-22 本文受国家重点基础研究发展规划(973)(2007CB310800),国家自然科学基金(60773006),北京市教育委员会科技发展计划面上项目(KM200710772003)资助。

叶 鹏(1976—),男,博士生,讲师,主要研究方向为软件体系结构、软件重用、反射理论等,E-mail:whuyp@126.com;应 时(1965—),男,博士,教授,博士生导师,主要研究方向为软件体系结构、软件的可重用性与互操作性、面向Aspect的软件开发等;袁文杰(1982—),男,博士生,主要研究方向为软件体系结构、反射理论、软件形式化方法等;姚俊峰(1973—),男,博士生,主要研究方向为软件体系结构、软件测试等。

元级由描述基级的元信息构成，根据不同的使用目的，这些元信息可以是描述基级结构、状态和行为方面的信息，也可以是描述基级性能、安全特性等方面的信息。元级和基级之间具有因果关联。元级中的内容及其变化可以通过反射过程反映到基级中，基级中的内容及其变化也可以通过具体化过程反映到元级中。元级和基级之间的因果关联是保证两者一致性的途径。元级中的元信息被显式地描述，并能被处理。通过修改元信息，可以改变体系结构的特性。

1.2 体系结构的元信息

元信息[7]是关于信息的信息，用于描述信息的结构、语义、用途和用法等。反射[8]利用元信息为管理、控制和使用复杂信息提供了一种高效的途径。通过从元信息到信息的反射机制，用户可以以更简单、更灵活、更自动化的方式使用信息本身。元信息和元信息建模可以用于软件体系结构的描述和使用过程中。

体系结构元信息是关于体系结构的信息，可用于描述体系结构的结构、语义、用途和用法等。根据不同的使用目的，可以定义不同的体系结构元信息。为了支持体系结构的重用，本文中的体系结构的元信息指的是支持软件体系结构重用的元信息。

目前用 ADL 设计软件体系结构时所关注的元素包括组件、连接器和体系结构配置等[5]。有鉴于此，我们从组件、连接器和组合件(即体系结构的整体)3 个方面定义体系结构元信息。对于其中组件的元信息和连接器的元信息，我们又分别从基本特征、结构、行为、约束、属性这 5 个方面去归纳和定义更详细的元信息。

- 基本特征元信息：支持在体系结构资源库中进行检索操作所需的元信息。
- 结构元信息：支持重用操作所需的体系结构元素在结构方面的元信息。
- 行为元信息：支持重用操作所需的体系结构元素在行为方面的元信息。
- 约束元信息：体系结构元素在被重用时应遵守的约束条件。
- 属性元信息：支持重用操作所需的体系结构元素在属性方面的元信息。体系结构元素某些特征不能简单地归类到基本特征、结构、行为和约束元信息，这时候可以把它定义为体系结构属性。常见的属性有体系结构的质量属性、实现约束等。

对于组合件的元信息，我们分别从外观、构成和配置方面去归纳和定义更详细的元信息。

- 外观元信息：将体系结构组合件看成一个黑盒时，与组件一样，它应具有基本特征、结构、行为、约束和属性方面的元信息，这些元信息称为体系结构的外观元信息。
- 构成元信息：在重用过程中，支持体系结构中构成元素的增减操作所需的元信息，如组件、连接器和组合件的列表。
- 配置元信息：在重用过程中，支持对体系结构中元素之间链接关系的修改操作所需的元信息。

2 MIM4RSA 模型的设计

2.1 总体设计原则与思路

设计 MIM4RSA 的根本目的是要解决难以显式地描述并使用支持体系结构制品重用操作和过程的信息，因此 MIM4RSA 模型中应定义支持软件体系结构重用的元信息。我们通过将一般 ADL 描述的体系结构设计方案中缺乏的、隐含的、不充分的支持体系结构重用的信息，用显式的、可以被计算机理解、表示和处理的方式充分地表达出来，形成元信息模型。

对于设计同一个系统的体系结构来说，使用不同的 ADL 设计得到的体系结构制品是不同的，它们之间既有共同点也有不同点。因此在定义 MIM4RSA 模型时要考虑到通用的体系结构元信息，它们主要用于针对体系结构中一些通用概念的描述，例如组件的名称等，同时也要考虑与特定 ADL 相关的元信息，它们主要用于针对不同 ADL 描述体系结构时有差异的某些方面，例如组件的接口定义等。目前，MIM4RSA 模型中只定义了与 C2SADEL[9] 相关的体系结构元信息，但当采用其它 ADL 描述体系结构时，需要将该部分相应地替换为其它 ADL 相关的元信息。

2.2 设计过程

从上述的体系结构元信息的归纳和定义过程，我们设计出如图 1 所示的具有层次结构的 MIM4RSA 模型。MIM4RSA 是一个层次结构的模型，这里只展示了它的前 3 层。体系结构元信息从总体上看包括组件的元信息、连接器的元信息和组合件的元信息。MIM4RSA 的第 1 层 ArchitectureMetaInfo 表示体系结构元信息。第 2 层中 ComponentMetaInfo 表示组件的元信息，ConnectorMetaInfo 表示连接器的元信息，CompositeMetaInfo 表示组合件的元信息。第 3 层进一步从基本特征、结构、行为、约束和属性 5 个方面分别说明组件的元信息、连接器的元信息，并且从外观、构成和配置 3 个方面说明组合件的元信息。组件的元信息和连接器的元信息比较类似，因此这里只对组件的元信息和组合件的元信息进行说明。

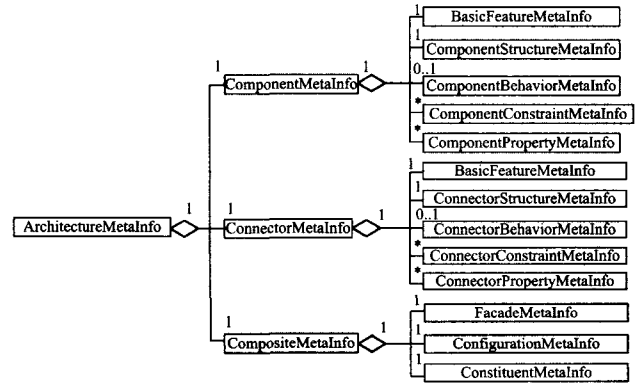

图 1 体系结构的元信息模型 MIM4RSA 的总体结构图

2.2.1 组件的元信息

组件的元信息包括组件的基本特征元信息、组件的结构元信息、组件的行为元信息、组件的约束元信息和组件的属性元信息，以下分别加以介绍。

(1)组件基本特征元信息

组件、连接器以及组合件的基本特征元信息所包含的内容是相同的。基本特征元信息的层次结构图如图 2 所示，它展示了基本特征元信息 BasicFeatureMetaInfo 包括作者(Author)、版本(Version)、命名空间(Namespace)、完成日期(DoneDate)、入库日期(StoredDate)、资源的类型(Resource-

Type)、体系结构描述语言(ADL)、体系结构风格(Style)、使用环境(Environment)、应用领域(Domain)、用途(Purpose)、功能(Functions)。

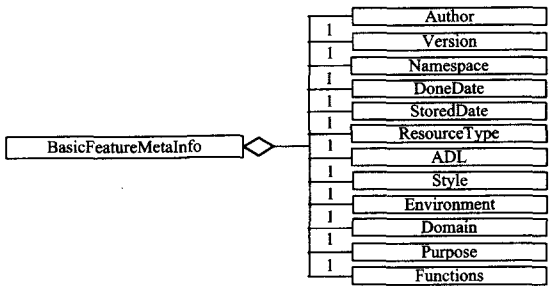

图 2　组件的基本特征元信息层次结构图

(2)组件的结构元信息

如图 3 所示,组件的结构元信息 ComponentStructureMetaInfo 描述组件结构方面的元信息,它进一步包括组件名称 ComponentName、组件类型名称 TypeName、接口元信息 InterfaceMetaInfo 以及可变性 Changeable。

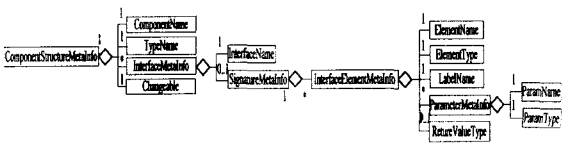

图 3　组件的结构元信息层次结构图

① 组件的名称(ComponentName):用字符串表示组件的名称。

② 组件类型的名称(TypeName):用字符串表示组件类型的名称。

③ 组件接口的元信息(InterfaceMetaInfo)包括接口名称 InterfaceName 和接口签名元信息 SignatureMetaInfo。InterfaceName 的作用是描述组件接口的名称,使用字符串类型的数据表示。SignatureMetaInfo 的作用是描述组件接口签名的元信息。由于不同的 ADL 定义组件接口的方式是不同的,因此 SignatureMetaInfo 会根据不同的 ADL 而有所不同,它是一个与特定 ADL 相关的元信息(ADL-specific Meta information)。因此,图 3 中 SignatureMetaInfo 部分描述的是 C2SADEL 的组件接口签名的元信息。当采用其它 ADL 描述体系结构时,需要将该部分相应地替换为其它 ADL 相关的元信息描述形式。

④ Changeable 指示组件在重用时其结构方面的信息能否被修改,可以用布尔类型的值来设置。以下出现的 Changeable 的作用与此类似,不再赘述。

(3)组件的行为元信息

如图 4 所示,组件的行为元信息 ComponentBehaviorMetaInfo 描述组件行为方面的元信息,它进一步包括行为规约元信息 SpecificationMetaInfo 以及可变性 Changeable。

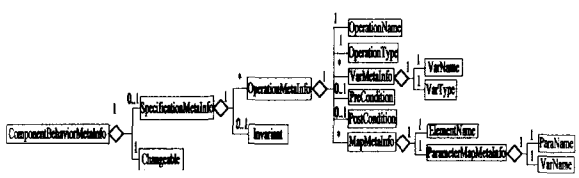

图 4　组件的行为元信息层次结构图

组件行为规约元信息(SpecificationMetaInfo)的作用是描述在重用过程中对组件行为的规约进行修改时所需的元信息。由于不同的 ADL 定义组件行为的方式是不同的,因此 SpecificationMetaInfo 会根据不同的 ADL 而有所不同,它也是一个与特定 ADL 相关的元信息。本文以 C2SADEL 作为研究对象,因此图 4 中的 SpecificationMetaInfo 部分描述的是针对 C2SADEL 定义的组件行为规约的元信息。当采用其它 ADL 描述体系结构时,需要将该部分相应地替换为其它 ADL 相关的元信息描述形式。

(4)组件的约束元信息

组件的约束元信息 ComponentConstraintMetaInfo 描述组件在被重用时应遵守的约束条件。如图 5 所示,约束元信息包括约束名称、约束类型以及约束表达式。

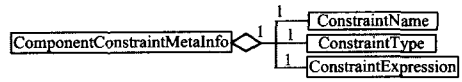

图 5　组件的约束元信息层次结构图

① 约束名称(ConstraintName):描述约束的名称,使用字符串表示。

② 约束类型(ConstraintType):描述约束的类型,分为强制类型 Invariant 和非强制类型 Heuristic 两种。

③ 约束表达式(ConstraintExpression):描述具体的约束条件,使用一阶逻辑表达式来表示。例如,对组件的接口的数量限定在 3 个以内,用逻辑表达式可表示为:self. interfaces. size<=3。

(5)组件的属性元信息

组件的属性元信息 ComponentPropertyMetaInfo 描述组件中除了上述的基本特征、结构、行为、约束以外的其他方面的元信息。如图 6 所示,属性元信息包括属性名称 PropertyName、属性类型 PropertyType 和属性的值 PropertyValue。

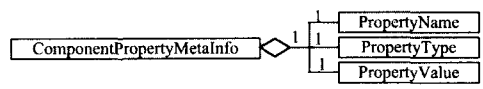

图 6　组件的属性元信息层次结构图

在一些 ADL 描述组件时,要描述一些除了基本特征、结构、行为、约束以外的其他方面的内容,例如在 C2SADL 组件中可以定义组件中使用的状态变量,状态变量就可以使用属性元信息来描述。例如 PropertyName 表示状态变量的名称;PropertyType 表示状态变量的类型,取值为 C2SADEL 中定义的数据类型。PropertyValue 表示状态变量的值。

2.2.2　组合件的元信息

组合件元信息包括组合件的外观元信息、组合件的构成元信息和组合件的配置元信息。下面我们分别说明这 3 种元信息。

(1)组合件的外观元信息

因为从系统外部看组合件就是一个组件,那么组合件的外观元信息中的各个部分与上述的组件元信息的各个部分描述方法相同。如图 7 所示,组合件的外观元信息包括组合件的基本特征元信息 BasicFeatureMetaInfo、组合件的结构元信息 CompositeStructureMetaInfo、组合件的行为元信息 CompositeBehaviorMetaInfo、组合件的约束元信息 CompositeConstraintMetaInfo 和组合件的属性元信息 CompositePropertyMetaInfo。

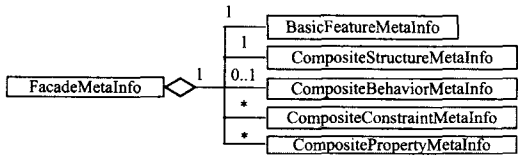

图 7　组合件外观元信息层次结构图

(2)组合件的构成元信息

组合件的构成元信息 constituentMetaInfo 描述构成组合件的组件、连接器的元信息。如图 8 所示，组合件的构成元信息包括组件元信息列表、连接器元信息列表和构成约束元信息。

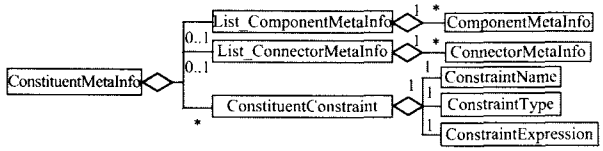

图 8　组合件构成元信息层次结构图

① 组合件元信息列表(List_ComponentMetaInfo)：包括组合件中包含的各个组件的元信息。

② 连接器元信息列表(List_ConnectorMetaInfo)：包括组合件中包含的各个连接器的元信息。

③ 构成约束元信息(ConstituentConstraint)：描述重用过程中对组合件构成成分的约束。例如，可以规定在重用过程中约束构成元素列表中某个组件不能被删除。

(3)组合件的配置元信息

组合件的配置元信息 ConfigurationMetaInfo 描述组合件的有关配置信息，主要是组件、连接器接口之间的链接关系和配置约束。链接关系可以分为两类：组合件外部接口与内部组件或者连接器接口之间的链接、组合件内部的组件接口与连接器接口之间的链接。

如图 9 所示，ConfigurationMetaInfo 包括若干链接元信息 LinkMetaInfo 和配置约束元信息 ConfigurationConstraint。

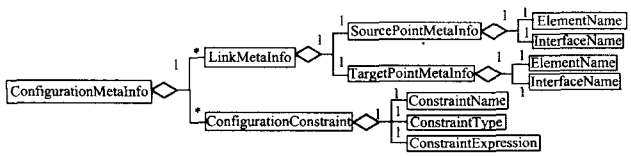

图 9　组合件配置元信息层次结构图

① 链接元信息(LinkMetaInfo)：描述了重用过程中对组合件中元素之间的链接关系进行修改所需的信息。LinkMetaInfo 包括起始端元信息 SourcePointMetaInfo 和目标端元信息 TargetPointMetaInfo。起始端元信息和目标端元信息都包括元素名称 ElementName 和接口名称 InterfaceName 两部分。ElementName 描述了进行链接的元素(某个组件、连接器)名称，InterfaceName 描述了该元素进行链接的接口名称。

② 配置约束元信息(ConfigurationConstraint)：描述重用过程中对组合件的内部拓扑结构的约束。

3　体系结构元信息描述语言 MetaADL

为了给设计人员提供一种建模规范，以准确、充分地描述 MIM4RSA 中的各种体系结构元信息，我们设计了体系结构元信息描述语言 MetaADL。MetaADL 语言为描述 MIM4RSA 模型中定义的各类元信息分别提供了相应的语法设施。MetaADL 既有用于描述通用元信息的语法成分，也有用于描述与特定 ADL 相关元信息的语法成分。目前，MetaADL 中只定义了与 C2SADEL 相关元信息的语法成分。

MetaADL 语言的语法定义是通过定义一组 XML schema 来完成的。XML schema 主要是适用于机器处理的，因此不易阅读。为了简要说明该语言，我们将使用 BNF 风格的 pseudo-schemas 来说明 MetaADL 的语法结构。Pseudo-schemas 接近 MetaADL 文档的最终结构，并采取了类似 BNF 范式的语法符号来描述文档的语法，易于阅读和理解。在 Pseudo-schemas 中，“?”表示可选(发生零次或者一次)，“*”表示发生零次或者多次，“+”表示发生一次或者多次，“[”和“]”表示一组元素，“|”分开几个可选择项。属性或者包含简单内容的元素通常指定一个表示其类型的值，如“xs:string”表示字符串类型。

在 MetaADL 中我们对应地定义了三类结构体(construct)，分别是：

• 元组件(metaComponent)：描述 MIM4RSA 模型中组件元信息的结构体。

• 元连接器(metaConnector)：描述 MIM4RSA 模型中连接器元信息的结构体。

• 元组合件(metaComposite)：描述 MIM4RSA 模型中组合件元信息的结构体。

3.1　元组件

```
< metaComponent id="xs:QName" >
        <basicFeature>
                <author>xs:string</author>?
                <resourceName>xs:string</resourceName>?
                <namespace>xs:string</namespace>?
                <version>xs:string</version>?
                <doneDate>xs:date</doneDate>?
                <storedDate>xs:date</storedDate>?
                <resourceType>?[architecture|component|connector|family]</resourceType>
                <ADL>xs:string</ADL>?
                <style>xs:string</style>?
                <environment>xs:string</environment>?
                <domain>xs:string</domain>?
                <purpose>xs:string</purpose>?
                <functions>xs:string</functions>?
        </basicFeature>?
        <structure typeName="xs:string" changeable="xs:boolean" name="xs:string">
                <interface id="xs:QName" name="xs:string">
                        <signature>
                                <interfaceElement id="xs:QName" name="xs:string">
                                        <elementType>[prov|req]</elementType>
                                        <labelName>xs:string</labelName>
                                        <paramDecl>
                                                <paramName>xs:string</paramName>
                                                <paramType>xs:string</paramType>
                                        </paramDecl>?
                                        <returnValueType>xs:string</returnValueType>?
                                </interfaceElement>*
                        </signature>
                </interface>*
        </structure>
        <behavior id="xs:QName" changeable="xs:boolean">
                <specification>
                        <operation id="xs:QName" name="xs:string">
                                <optType>[prov|req]</optType>
                                <localVarDecl>
                                                                <varName>xs:string</varName>
                                                                <varType>xs:string</varType>
                                </localVarDecl>*
                                <pre>xs:string</pre>
                                <post>xs:string</post>
                                <map id="xs:QName">
                                        <elementName>xs:QName</elementName>
                                        <parameterMap>
                                                                <paraName>xs:string</paraName>
                                                                <varName>xs:string</varName>
                                        </parameterMap>?
                                </map>*
                        </operation>*
                        <invariant>xs:string</invariant>?
                </specification>
        </behavior>?
        <constraint id="xs:QName">
                <name>xs:string</name>
                <type>xs:string</type>
                <expression>xs:string</expression>
        </constraint>*
        <property id="xs:QName">
                <name>xs:string</name>
                <type>xs:string</type>
                <value>xs:string</value>
        </property>*
</ metaComponent>
```

图 10　元组件的文档结构

元组件描述体系结构元信息模型 MIM4RSA 中组件的元信息。在 MIM4RSA 中，组件的元信息分为基本特征元信息、结构元信息、行为元信息、约束元信息和属性元信息 5 个部分。与之相对，如图 10 所示，在 MetaADL 中元组件定义了 5 个子元素 basicFeature，structure，behavior，constraint 和 property，它们分别描述上述的 5 种元信息。为了便于计算机

处理，在 MetaADL 中为各种体系结构元素都设定了一个唯一的标识符。例如 metaComponent 就有一个属性 id 来表示其唯一标识。在以下将要介绍的各个元素中，它们属性 id 的作用与此相同。

3.2 元连接器

元连接器描述体系结构元信息模型 MIM4RSA 中连接器的元信息，与元组件类似。如图 11 所示，在 MetaADL 中元连接器中定义了 5 个子元素 basicFeature，structure，behavior，constraint 和 property，它们分别描述连接器的基本特征元信息、结构元信息、行为元信息、约束元信息和属性元信息。

```
<metaConnector id="xs:QName">
    <basicFeature>
        <author>xs:string</author>?
        <resourceName>xs:string</resourceName>?
        <namespace>xs:string</namespace>?
        <version>xs:string</version>?
        <doneDate>xs:dnte</doneDate>?
        <storedDate>xs:date</storedDate>?
        <resourceType>[architecture|component|connector|family]</resourceType>
        <ADL>xs:string</ADL>?
        <style>xs:string</style>?
        <environment>xs:string</environment>?
        <domain>xs:string</domain>?
        <purpose>xs:string</purpose>?
        <functions>xs:string</functions>?
    </basicFeature>?
    <structure typeName="xs:string" changeable="xs:boolean" name="xs:string">
        <interface id="xs:QName" name="top">
        </interface>
        <interface id="xs:QName" name="bottom">
        </interface>
    </structure>
    <behavior id="xs:QName" changeable="xs:boolean">
    </behavior>?
    <constraint id="xs:QName">
        <name>xs:string</name>
        <type>xs:string</type>
        <expression>xs:string</expression>
    </constraint>*
    <property id="xs:QName">
        <name>xs:string</name>
        <type>xs:string</type>
        <value>xs:string</value>
    </property>*
</metaConnector>
```

图 11 元连接器的文档结构

3.3 元组合件

如图 12 所示，元组合件中定义了 3 个子元素 facade，constituent 和 configuration。其中元素 facade 描述组合件的外观元信息，元素 constituent 描述组合件的构成元信息，元素 *configuration* 描述组合件的配置元信息。

```
<metaComposite id="xs:QName">
    <facade id="xs:QName">
        ……
    </facade>
    <constituent>
        <metaComponentList>
            <metaComponent id="xs:QName">
            ……
            </metaComponent>*
        </metaComponentList>
        <metaConnectorList>
            <metaConnector id="xs:QName">
            ……
            </metaConnector>*
        </metaConnectorList>
        <constraint>
            <name></name>
            <type></type>
            <expression></expression>
        </constraint>*
    </constituent>
    <configuration>
        <link id="xs:QName">
            <point>
                <elementName>xs:QName</elementName>
                <interfaceName>xs:QName</interfaceName>
            </point>
            <point>
                <elementName>xs:QName</elementName>
                <interfaceName>xs:QName</interfaceName>
            </point>
        </link>*
        <constraint>
            <name></name>
            <type></type>
            <expression></expression>
        </constraint>*
    </configuration>
</metaComposite>
```

图 12 元组合件的文档结构

4 案例研究

本部分介绍使用上述元信息模型和 MetaADL 语言进行体系结构重用的实际案例。我们通过重用两个已有的体系结构，完成一个银行客户业务管理系统的体系结构设计。案例的业务目标是设计一个银行客户业务管理系统的体系结构。这个银行客户业务管理系统的需求包括能支持对多种帐户（储蓄帐户、支票帐户和商业帐户）和客户的管理，并且提供通过银行出纳员和 ATM 访问帐户的途径，这个系统还能支持日常计息、取款、存款、余额查询和转帐的功能。

银行客户业务管理系统的设计是通过重用两个已有的 C2SADEL 描述的体系结构实现的：一个是银行帐户管理系统的体系结构，如图 13 所示。这个系统能完成对储蓄帐户、支票帐户、商业帐户以及客户信息的管理功能。另一个是银行事务管理系统的体系结构，如图 14 所示。这个系统能完成查询余额、取款、存款、计息和转帐的银行业务功能。通过重用这两个已有的体系结构，设计出目标体系结构。

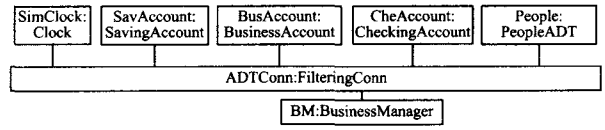

图 13 银行帐户管理系统的体系结构的图形化表示

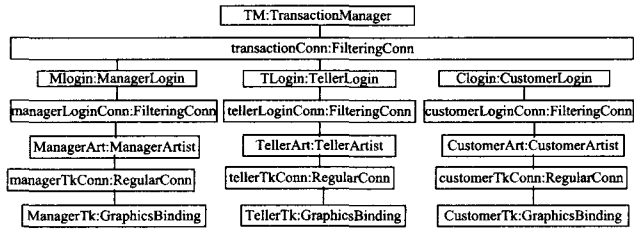

图 14 银行事务管理系统的体系结构的图形化表示

重用这两个体系结构的前提是先把这两个体系结构制作成反射式的体系结构制品：基级仍然是原来的 C2SADEL 描述的体系结构，而元级则是由 MetaADL 所描述的基级的元信息构成的。制作后得到的两个可重用的反射式体系结构制品存入可重用体系结构资源库中。目标体系结构（也就是银行客户业务管理系统的体系结构）的设计是通过重用上述两个体系结构完成的：通过在体系结构资源库中检索，可以找到这两个反射式体系结构制品，然后将这两个制品组合起来，得到目标体系结构。

4.1 制作可重用的反射式体系结构制品

我们以银行帐户管理系统体系结构为例，介绍反射式体系结构制品制作的方法和过程。

银行帐户管理系统的体系结构包括 6 个组件：SavAccount 表示储蓄帐户、CheAccount 表示支票帐户、BusAccount 表示商业帐户、SimClock 表示系统的计时器、BM 表示业务管理、People 表示客户信息。连接件 ADTConn 负责传递组件间的消息，BM 负责维护帐户和客户的信息。这个系统能完成对储蓄帐户、支票帐户、商业帐户以及客户信息的管理功能。图 15 中，用 C2SADEL 描述的银行帐户管理系统的体系结构作为反射式体系结构制品的基级。

```
architecture AccountManagerSystem is {
  component_types {
    component Clock is extern {AccountManager/Clock.c2;}
    component SavingAccount is extern { AccountManager/SavingAccount .c2; }
    component BusinessAccount is extern
{AccountManager/BusinessAccount .c2;}
    component CheckingAccount is extern
{AccountManager/CheckingAccount .c2;}
    component PeopleADT is extern { AccountManager/PeopleADT .c2; }
    component BusinessManager is extern
{ AccountManager/BusinessManager .c2;}
  }
  connector_types{
    connector FilteringConn is { message_filter message_filtering;}
  }
  architectual_topology {
    component_instances {
      SavAccount1:SavingAccount ; CheAccount:CheckingAccount ;
      BusAccount:BusinessAccount ; SimClock:Clock;
      People:PeopleADT ; BM:BusinessManager;
    }
    connector_instances {
      ADTConn:FilteringConn;
    }
    connections {
      connector ADTConn
        {top SavAccount ,CheAccount, People,BusAccount ,SimClock ; bottom
BM;}
    }
  }
}
```

图 15 银行帐户管理系统的体系结构的 C2SADEL 描述

同时我们使用 MetaADL 描述上述基级的元信息，得到反射式体系结构制品的元级。元级是由基级的元信息组成的，因此我们使用 MetaADL 语言描述银行帐户管理系统的体系结构的元信息，包括外观元信息（如图 16 所示）、构成元信息（如图 17 所示）和配置元信息（如图 18 所示），从而得到对反射式体系结构制品的元级的描述。

```
<facade id="metaArch01">
    <basicFeature>
        <author >张三</author>
        <resourceName>AccountManagerSystem </resourceName>
        <namespace>http ://www. sklse .org/reflection /zhang</
namespace>
        <version >1.0</version>
        <doneDate >2007 -09- 18</doneDate>
        <storedDate >2007 - 10 - 01</storedDate>
        <resourceType >architecture</resourceType>
        <ADL>C2SADEL</ADL>
        <style >C2</style>
        <environment />
        <domain>银行帐户管理系统</domain>
        <purpose >管理各种客户帐户信息</purpose>
        <functions />
    </basicFeature>
    <structure changeable ="true" name="AccountManagerSystem ">
        <interface id ="outerInterface " name=" interface"></interface>
    </structure>
</facade>
```

图 16　银行帐户管理系统体系结构的外观元信息描述

```
<constituent >
  <metaComponentList >
    <metaComponent id ="MSavingAccount "
      <structure changeable ="true" typeName ="SavingAccount " name="SavAccount" >
        <interface id ="MSavingAccount interface01" name ="interface">
          <signature
            <interfaceElement id ="elm01" name ="SetPwd" >
              <elementType :prov :elementType >
              <labelName>ip setpwd </labelName >
              <paramDecl>
                <paramName>p</paramName>
                <paramType>String</paramType>
              </paramDecl>
            </interfaceElement>
            .....
          </signature >
        </interface >
      </structure >
      <behavior id ="MSavingAccount behavior" changeable ="true" >
        <specification >
          <operation id ="op01" name="op_setpwd" >
            <optType :prov :optType >
            <localVarDecl >
              <varName>pw</varName>
              <varType >String</varType>
            </localVarDecl>
            <pre> <password> pw</pre>
            <map>
              <elementName> SetPwd </elementName >
              <parameterMap>
                <paraName>p</paraName>
                <varName>pw</varName>
              </parameterMap>
            </map>
          </operation >
          .....
          <invariant (balance >agcreater 0) and (password <> NULL ) </invariant >
        </specification >
      </behavior >
      <property id ="property 01 "
        name accountNo </name
        type String </type
        value </value
      </property >
      .....
    </metaComponent >
    .....
  </metaComponentList >
  <metaConnectorList >
    <metaConnector id ="MADTConn ":
      <structure changeable ="true" typeName ="FilteringConn " name ="ADTConn ">
        <interface id "Conn interface01" name "top" >
        <interface id "Conn interface02" name "bottom ">
      </structure >
      <constraint >
        name Conn constraint 01 </name>
        type invariant </type>
        <expression (self .interfaces size = 2) and
          (exists i in self .interfaces implies (i.name =top)) and
          (exists i in self .interfaces implies (i.name = bottom ))
        </expression
      </constraint >
      <property id = "property 08"
        name filtering -policy </name>
        type enum </type>
        value "message filtering "</value>
      </property >
    </metaConnector >
  </metaConnectorList >
</constituent >
```

图 17　银行帐户管理系统体系结构的构成元信息描述

4.2　基于重用的目标系统体系结构的设计

对以上制作出的两个反射式体系结构进行组合重用操作，得到一个银行客户业务管理系统的体系结构，即目标系统的体系结构。步骤如下：

（1）检索——从可重用资源库中检索这两个反射式体系结构制品。

按照目标系统的需求，我们可以在体系结构资源库中查找可重用的体系结构制品。由于反射式体系结构制品的元级中有它们的基本特征元信息的描述，体系结构制品按照这些基本特征信息在体系结构资源库中进行分类和注册。因此，这些元信息是用于检索操作。通过关键字检索的方法可以检索出这两个反射式体系结构制品。

（2）组合——将这两个反射式体系结构制品组合起来，形成目标系统的体系结构。

```
<configuration >
        <link id = "link 01">
            <point >
                <elementName>SimClock</elementName>
                <interfaceName>interface</interfaceName>
            </point>
            <point >
                <elementName>ADTConn </elementName>
                <interfaceName>top</interfaceName>
            </point>
        </link>
        <link id = "link 02">
            <point >
                <elementName>BusAccount </elementName>
                <interfaceName>interface</interfaceName>
            </point>
            <point >
                <elementName>ADTConn </elementName>
                <interfaceName>top</interfaceName>
            </point>
        </link>
        <link id = "link 03">
            <point >
                <elementName>CheClock</elementName>
                <interfaceName>interface</interfaceName>
            </point>
            <point >
                <elementName>ADTConn </elementName>
                <interfaceName>top</interfaceName>
            </point>
        </link>
        <link id = "link 04">
            <point >
                <elementName>SavAccount</elementName>
                <interfaceName>interface</interfaceName>
            </point>
            <point >
                <elementName>ADTConn </elementName>
                <interfaceName>top</interfaceName>
            </point>
        </link>
        <link id = "link 05">
            <point >
            <elementName>People</elementName>
            <interfaceName>interface</interfaceName>
            </point>
            <point >
                <elementName>ADTConn </elementName>
                <interfaceName>top</interfaceName>
            </point>
        </link>
        <link id = "link 06">
            <point >
                <elementName>BM</elementName>
                <interfaceName>interface</interfaceName>
            </point>
            <point >
                <elementName>ADTConn </elementName>
                <interfaceName>bottom </interfaceName>
            </point>
        </link>
        <link id = "link 07">
            <point >
                <elementName>BM</elementName>
                <interfaceName>interface</interfaceName>
            </point>
            <point >
                <elementName>AccountManagerSystem </
elementName>
                <interfaceName>interface</interfaceName>
            </point>
        </link>
</configuration >
```

图 18　银行帐户管理系统体系结构的配置元信息描述

我们通过增加一个新的连接器 dispatchConn，从而将这两个体系结构连接起来，形成一个有机的整体，得到如图 19 所示的新体系结构。然后通过一系列操作，可以得到银行客户业务管理系统的体系结构制品的元级的 MetaADL 文档和基级的 C2SADEL 文档，如图 20 所示。

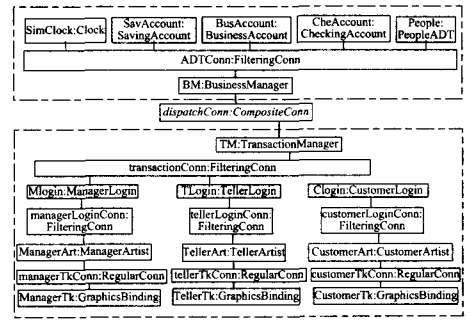

图 19　银行客户业务管理系统的体系结构的图形化表示

```
architecture BankApplicationSystem is {
  component_types {
    component Clock is extern {AccountManager /Clock .c2;}
    component SavingAccount is extern { AccountManager /SavingAccount .c2; }
    component BusinessAccount is extern {AccountManager / BusinessAccount .c2;}
    component CheckingAccount is extern {AccountManager /CheckingAccount .c2;}
    component PeopleADT is extern { AccountManager /PeopleADT .c2; }
    component BusinessManager is extern { AccountManager /BusinessManager .c2;}
    component TranscationManager is extern
{TranscationManager /TranscationManager .c2}
    component ManagerLogin is extern {TranscationManager /ManagerLogin .c2}
    component TellerLogin is extern {TranscationManager /tellerLogin .c2}
    component CustomerLogin is extern {TranscationManager /CustomerLogin .c2}
    component MananagerArtist is extern {TranscationManager /ManangerArtist .c2}
    component TellerArtist is extern {TranscationManager /TellerArtist .c2}
    component CustomerArtist is extern {TranscationManager /CustomerArtist .c2}
    component GraphicsBinding is virtual {}
  }
  connector types {
    connector FilteringConn is { message_filter message_filtering ;}
    connector RegularConn is {message_filter no_filtering ;}
    connector CompositeConn is {message_filter no_filtering ;}
  }
  architectual_topology {
    component_instances {
      SavAccount :SavingAccount ; CheAccount :CheckingAccount ;
      BusAccount :BusinessAccount ; SimClock :Clock;
      People :PeopleADT ; BM :BusinessManager ;
      TM :TranscationManager ; TLogin :TellerLogin ;
      MLogin :ManagerLogin ; CLogin :CustomerLogin ;
      ManagerArt :MananagerArtist ; CustomerArt :CustomerArtist ;
      TellerArt :TellerArtist ; TellerTk :GraphicsBinding ;
      ManagerTk :GraphicsBinding ; CustomerTk :GraphicsBinding ;
    }
    connector_instances {
      ADTConn :FilteringConn ;
      transactionConn :FilteringConn ; managerLoginConn :FilteringConn ;
      tellerLoginConn :FilteringConn ; tellerTkConn :RegularConn;
      customerLoginConn :FilteringConn ; customerTkConn :RegularConn;
      managerTkConn :RegularConn ; dispatchConn :CompositeConn ;
    }
    connections {
      connector ADTConn
        {top SavAccount ,CheAccount, People,BusAccount ,SimClock , bottom BM ;}
      connector transactionConn
        {top TM ; bottom TLogin ,Clogin ,MLogin ;}
```

（下转第 186 页）

[J]. IEEE Transactions on Automatic Control, 2005, 50(3): 406-410

[9] Lin Hai, Antsaklis P J. Robust Invariant Control Synthesis for Discrete-Time Polytopic Uncertain Linear Hybrid Systems[A]//Proc. of ACC03[C]. 2003

[10] Bemporad A, Morari M. Control of Systems Integrating Logic, Dynamic, and Constraints [J]. Automatica, 1999, 35(3): 407-427

[11] 张聚. 混杂系统理论及在非线性系统中的应用研究[D]. 杭州: 浙江大学, 2005. 2

[12] Richards A, How J. Mixed-integer Programming for Control [A]//ACC[C]. 2005

[13] Bemporad A. Hrid Toolbox For Real-Time Applications. October 2006

[14] Grieder P, Kvasnica M, Baotic M, et al. Low complexity control of piecewise affine systems with stability guarantee. ACC. 2004

[15] Fletcher R, Leyffer S. Numerical Experience With Lower Bounds for MIQP Brand-and-Bound[R]. Dept. of Mathematics, University of Dundee, Scotland, U. K. SIAM J. Optim., submitted, 1995

[16] Torrisi F D, Bemporad A. HYSDEL 2.0-User Manual, 2002

[17] Wonham W M. Linear Multivariable Control: a Geometric Approach[M]. New York: Springer Verlag, 1985

[18] 胡一凡. 飞行机器人的建模和控制[D]. 广州: 华南理工大学, 2004

[19] 席裕庚, 王凡. 非线性系统预测控制的多模型方法[J]. 自动化学报, 1996, 22(4): 456-461

[20] Rakovic S, Grider P, Kvasnica M, et al. Computation of Invariant Sets for Piecewise Affine Discrete Systems Subjects to Bounded Disturbances[C]//CDC. 2004

[21] Bertsekas D P. Infinite-time Reachability of State-Space Regions by Using Feedback Control[J]. IEEE Transaction on Automatic Control, 1972, 17(5): 604-613

[22] 李坚强, 裴海龙. 一类非线性系统最大可控不变集求解[J]. 控制工程, 2009(2)

(上接第 150 页)

```
        connector tellerLoginConn {top TLogin ;bottom TellerArt ;}
        connector customerLoginConn {top CLogin ;bottom CustomerArt ;}
        connector managerLoginConn {top Mlogin ;bottom ManagerArt ;}
        connector tellerTkConn {top TellerArt ;bottom TellerTk ;}
        connector customerTkConn {top CustomerArt ;bottom CustomerTk ;}
        connector managerTkConn {top ManagerArt; bottom managerTk ;}
        connector dispatchConn {top BM ; bottom TM ;}
    }
}
```

图 20 银行客户业务管理系统的体系结构的 C2SADEL 描述

结束语 本文的主要贡献在于建立了体系结构元信息模型 MIM4RSA，系统地定义了支持重用的体系结构元信息。该模型也是一个具有良好的可扩展性的信息模型框架，我们可以针对不同的 ADL，来对该模型框架中的组成成分进行修改，以建立适应不同 ADL 的元信息模型。同时更重要的是，我们可以利用 MetaADL 将体系结构元信息描述出来，使得计算机可以处理这些元信息，为实现体系结构设计人员通过使用体系结构元信息，高效地重用软件体系结构奠定了良好的基础。

作为今后的工作，我们将不断完善 MIM4RSA 模型和 MetaADL，具体来说就是针对除了 C2SADEL 之外的其它 ADL，在元信息模型 MIMSA 中定义与这些 ADL 相关的体系结构元信息，相应地在 MetaADL 中定义支持描述这些元信息的语法成分；我们还需要开发一个工具，以支持体系结构设计人员通过操作元信息来实现软件体系结构重用。

参 考 文 献

[1] Shaw M, Garlan D. Software Architecture: Perspectives on an Emerging Discipline[M]. Prentice Hall, 1996

[2] Mili H, Mili A, Yacoub S. Reuse-based Software Engineering: Techniques, Organization, and Controls. New York: Jonh Wiley & Sons, 2002

[3] Keller R K, Schauer R. Design Components: Towards Software Composition at the Design Level[C]// Proceedings of the 20th International Conference on Software Engineering (ICSE'98). New York: ACM Press, 1998: 302-311

[4] Mili H, Mili A, Yacoub S. Reuse-based Software Engineering: Techniques, Organization, and Controls. Jonh Wiley & Sons Ltd., 2001

[5] Medvidovic N, Taylor R N. A Classification and Comparison Framework for Software Architecture Description Languages [J]. IEEE Transactions on Software Engineering, 2000, 26 (1): 70-93

[6] Maes P. Concepts and Experiments in Computational Reflection [C]// Proceedings of OOPSLA87, ACM SIGPLAN Notices. New York: ACM Press, 1987: 147-155

[7] Cazzola W, Savigni A, Sosio A, et al. Architectural reflection: Concepts, design, and evaluation[R]. RI-DSI 234-99. DSI, University degli Studi di Milano, May 1999

[8] Cazzola W, Savigni A, Sosio A, et al. Explicit Architecture and Architectural Reflection[C]// Proceedings of the 2nd International Workshop on Engineering Distributed Objects (EDO 2000), LNCS. Springer-Verlag, 2000

[9] Oreizy P, Medvidovic N, Taylor R N. Architecture-Based Runtime Software Evolution[C]// Proceedings of the 20th International Conference on Software Engineering (ICSE'98). New York: ACM Press, 1998: 177-186

[10] Dowling J, Cahill V. The K-Component Architecture Meta-Model for Self-Adaptive Software[C]// Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns, Lecture Notes In Computer Science. Vol. 2192. London: Springer-Verlag, 2001: 81-88