基于 Object-Z 的 ReflectiveArchitecture 形式化研究

罗巨波[1] 应 时[2]
(扬州大学新闻与传媒学院 扬州 225009)[1] (武汉大学软件工程国家重点实验室 武汉 430072)[2]

摘 要 将元信息、元建模、反射和软件体系结构结合起来,提出了一种基于反射机制的软件体系结构重用方法,该方法是一种更通用、更便捷的重用方法;该方法定义并构造了一种在设计阶段支持软件体系结构重用的反射机制 RMRSA。描绘了基于反射机制 RMRSA 的反射式软件体系结构 ReflectiveArchitecture 的元级体系结构模型。运用形式规格说明语言 Object-Z 对元级体系结构模型进行了完整的描述;选取链接模式 Link,给出了它的初始化定理及其证明过程,以此为范例证明了被形式化的反射式软件体系结构的正确性。

关键词 软件体系结构重用,反射式软件体系结构,元级体系结构模型,形式化
中图法分类号 TP311.5 **文献标识码** A

Research on ReflectiveArchitecture Formalism Based on Object-Z

LUO Ju-bo[1] YING Shi[2]
(School of Journalism and Communication, Yangzhou University, Yangzhou 225009, China)[1]
(State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)[2]

Abstract Banding together meta information, meta modeling, reflection and software architecture, we presented a method of reusing software architecture based on reflection mechanism. It is a more versatile and convenient method. This method has defined and constructed a reflection mechanism RMRSA for software architecture reuse at software design phase. This paper described the meta-level architecture model of the reflective software architecture based on RMRSA. Moreover, it formalized the meta-level architecture model using the formal specification language--Object-Z completely. Taking the Link schema as an example, this paper also gave the initial theorem and its testified process, so as to testify the correctness of the formalized reflective software architecture.

Keywords Reuse of software architecture, Reflective software architecture, Meta-level architecture model, Formalization

1 引言

特定领域的软件体系结构[1]、软件体系结构模式[2]、软件体系结构风格[3]和体系结构框架[4]等在一定程度上,可以帮助人们重用软件体系结构,但是这些重用方法存在的主要问题是缺乏统一的体系结构建模方法、基于不同的体系结构描述语言和缺乏支持重用过程的信息。

文献[5]定义并构造了一种在设计阶段支持软件体系结构重用的反射机制 RMRSA(Reflection Mechanism for Reusing Software Architecture)。基于 RMRSA 的体系结构重用方法是一种更通用、更便捷的体系结构制品本身的重用方法,它具有统一的体系结构建模方法的信息,给出了统一的元级体系结构描述语言 MetaADL。RMRSA 利用元级体系结构中的元信息,可以为体系结构设计人员和设计工具执行重用基级体系结构的操作和过程提供支持。RMRSA 重用体系结构的基本方法是:将通用的体系结构语言所描述的体系结构作为基级,将体系结构重用元信息在元级中建模,通过定义抽取基级体系结构的重用元信息的具体化操作和利用元级体系结构元信息描述来获得基级体系结构的反射操作,体系结构设计人员和工具可以在元级对体系结构重用元信息这一高抽象层次设计元素来进行组合与重用,从而实现重用已有的体系结构设计结构,构造满足需求的软件体系结构。

本文将在此基础上做如下工作:给出反射式软件体系结构 ReflectiveArchitecture 的定义并介绍其组成部分,用形式规格说明语言 Object-Z 对元级体系结构模型进行完整的描述,选取其中一个具有代表性的模式,给出它的初始化定理及其证明过程,以此为范例证明了反射式体软件系结构基于 Object-Z 规格说明的正确性。

2 反射式软件体系结构及其形式化描述

2.1 反射式软件体系结构 ReflectiveArchitecture

我们把基于反射机制 RMRSA 的软件体系结构称作反射

到稿日期:2009-12-23 返修日期:2010-03-05 本文受国家自然科学基金项目(60473066),教育部人文社会科学研究项目(09YJC870024)资助。

罗巨波(1976-),男,博士,讲师,主要研究方向为软件体系结构和模式、软件复用,E-mail:whuluocheng@126.com;应 时(1965-),男,博士,教授,博士生导师,主要研究方向为基于组件的软件工程方法学、软件体系结构和模式、软件可重用性和互操作性等。

式软件体系结构 ReflectiveArchitecture。反射式软件体系结构由基本级体系结构、元级体系结构和 PMB(Protocol for connecting Meta-level architecture and Base-level architecture)协议 3 部分组成[5]。PMB 协议包括:定义元级体系结构和基级体系结构之间的交互和互操作(包括反射操作和具体化操作,这两种操作都为支持软件重用的操作服务)是动态的;定义元级和基级的静态因果关联是静态的。PMB 协议保证了元级体系结构的改变能同时改变基级结构,基级体系结构的改变也能在元级软件体系结构中得到相应的改变,确保 RMRSA 中元级与基级之间的因果关联始终是正确的。

反射式软件体系结构的基级体系结构模型我们以 C2SADEL 为例进行说明,它分为 3 大部分:组件类型定义部分 component_types:定义了一组组件类型;连接器类型定义部分 connector_types:定义了一组连接器类型;拓扑结构定义部分 archietectural_topology:定义了一组组件 component_instances、一组连接器 connector_instances 以及这些组件与连接器之间的连接关系 connections。

反射式软件体系结构的元级体系结构模型是使用基级体系结构中的元信息构造出来的。它包括元组件、元连接器和元组合件,分别对应基级体系结构模型的组件、连接器和基级体系结构本身。其中元组合件包含体系结构的外观、构成和配置。元级体系结构模型如图 1 所示[5]。

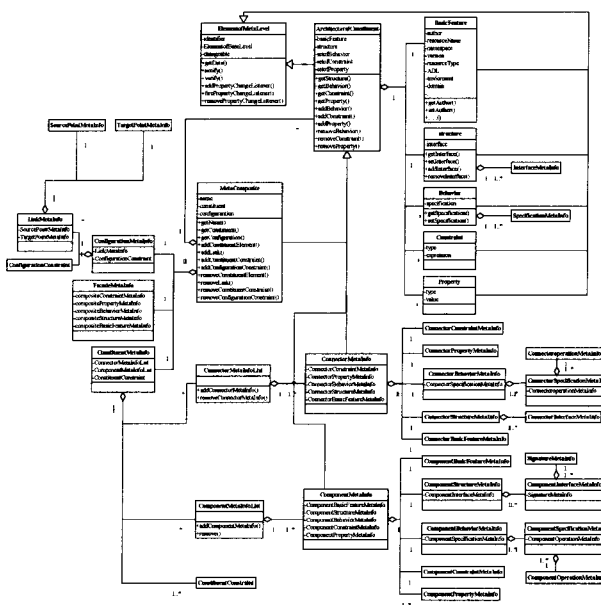


图 1　元级体系结构模型

2.2　反射式体软件系结构的形式化描述

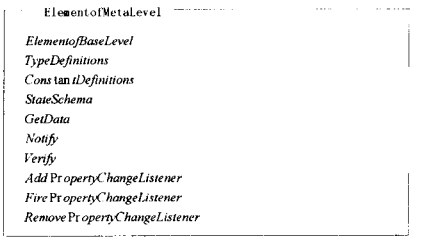
在文献[5]中,用形式规格说明语言 Object-Z 对基级体系结构模型进行了完整的描述。

在此,用形式规格说明语言 Object-Z 对元级体系结构模型进行完整的描述,如下所示。

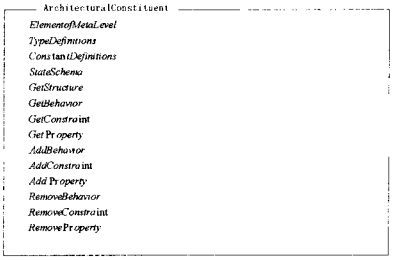
2.2.1　元组合件描述

在给出元组合件类前,先定义元级元素类 ElementofMetaLevel和元级体系结构组成成分类。

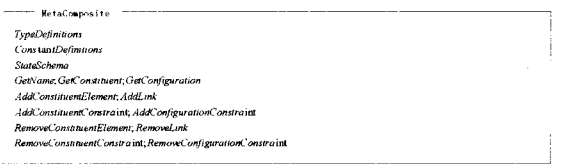
(1)元级元素 ElementofMetaLevel 是所有元级元素的超类,它的类 ElementofMetaLevel 的形式化描述如下:



(2)元级体系结构组成成分类 ArchitecturalConstituent 的形式化描述如下:



(3)元组合件类 MetaComposite 的形式化描述如下,其中声明的分量 *architecturelconstituent*,*metacompofacade*,*metacompoconstituent* 和 *metacompoconfig* 分别是元级体系结构组成成分类 *ArchitecturalConstitutent*、元组合件的外观类 *MetaCompoFacade*、元组合件构成类 *MetaCompoConstituent* 和元组合件配置类 *MetaCompoConfig* 的变量。



下面分别定义元组合件的外观类 MetaCompoFacade、元组合件构成类 MetaCompoConstituent 和元组合件配置类 MetaCompoConfig。

(4)元组合件的外观类 MetaCompoFacade 的形式化描述如下,其中声明的分量 *metacompostructure*,*metacompobehavior*,*metacompositeconstra int* 和 *metacompositeProperty* 分别是元组合件结构类 *MetaCompoStructure*、元组合件行为类 *MetaCompoBehavior*、元组合件约束类 *MetaCompositeComstraint* 和元组合件属性类 *MetaCompositeProperty* 的变量。

(5)元组合件构成类 MetaCompoConstituent 形式化描述如下,其中声明的分量元组件列表 *list_metacomponent*、元连接器列表 *list_metaconnector* 分别是 *List_MetaComponent*,*List_MetaConnector* 的变量。

下面分别定义元组合件构成成分的元组件列表类 List_MetaComponent 和元组件列表初始模式 InitList_MetaComponent、元连接器列表类 List_MetaConnector 和元连接器列

表初始化模式 InitList_MetaConnector、元组合件构成约束状态类 MetaCompoConstiConstr。

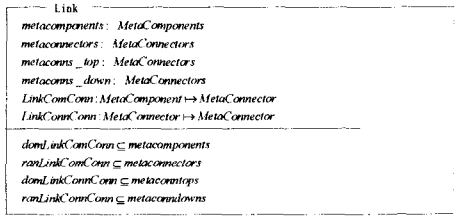
(6)元组件列表包含了所有元组件的集合，其中 *AddComponentMetaInfo* 和 *RemoveComponentMetaInfo* 是它的增加组件和删除组件操作模式。元组件列表类 List_MetaComponent 定义如下：

(7)元连接器列表类 List_MetaConnector 的形式化描述如下：

(8)元组合件构成约束类 MetaCompoConstiConstr 的形式化描述如下：

(9)元组合件配置类 MetaCompoConfig 定义如下：

(10)链接元信息描述了重用过程中，对组合件中元素之间的链接关系进行修改所需的信息。链接类 Link 的形式化描述如下，其中 *LinkComConn*:*MetaComponent* ⊢→*MetaConnector* 定义了元组件 *MetaComponent* 到元连接器 *MetaConnector* 的映射关系。*LinkConnConn*:*MetaConnector* ⊢→*MetaConnector* 定义了从元连接器 *MetaConnector* 到元连接器 *MetaConnector* 的映射关系。



2.2.2 元组件描述

(1)元组件类 MetaComponent 的形式化描述如下，其中声明的分量 *metacomstructure* 和 *metacomponentbehavior* 分别是元组件结构类 *MetaComStructure* 和元组件行为类 *MetaComponentBehavior* 的类型。

basicfeature，metacomstructure，metacomponenbehavior，metacomconstraint，metacomproperty 是元组件类 MetaComponent 的分量，它们分别是元组件基本特征类 BasicFeature、元组件结构类 MetaComStructure、元组件行为类 MetaComponentBehavior、元组件约束类 MetaComConstraint 和元组件属性类 MetaComProperty 的类型。

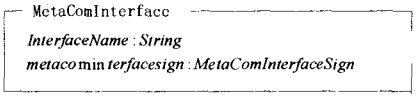
(2)元组件结构类 MetaComStructure

组件的结构元信息 ComponentStructureMetaInfo 描述组件的结构方面的元信息，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。SignatureMetaInfo 包括若干的接口元素元信息 InterfaceElementMetaInfo，它们分别描述每个接口元素的元信息。一组 ParameterMetaInfo 分别描述了每个接口元素的参数元信息。

根据上面详细介绍的组件的结构元信息，我们给出元组件结构类 MetaComStructure 描述如下：

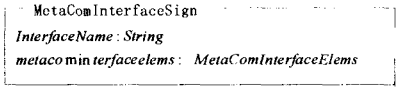
(3)元组件接口模式 MetaComInterface 是元组件结构类 MetaComStructure 的接口分量的类型，元组件接口签名模式 MetaComInterfaceSign 是元组件接口模式 MetaComInterface 的接口签名分量的类型，元组件接口元素模式 MetaComInterfaceElem 是元组件接口签名模式 MetaComInterfaceSign 的接口元素分量的类型，元组件接口元素参数模式 MetaComInterfaceElemPara是元组件接口元素模式 MetaComInterfaceElem 的接口元素参数分量的类型，分别对其定义如下：

元组件接口模式 MetaComInterface



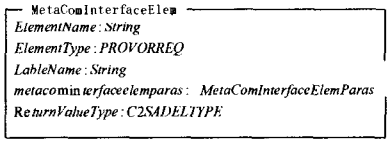
(4)元组件接口签名模式 MetaComInterfaceSign

元组件接口签名 MetaComInterfaceSign 包含若干个元组件接口元素 MetaComInterfaceElem。

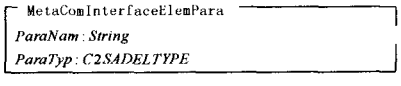


(5)元组件接口元素模式 MetaComInterfaceElem

一个元组件接口元素 MetaComInterfaceElem 包含若干个元组件接口元素参数 MetaComInterfaceElemPara。



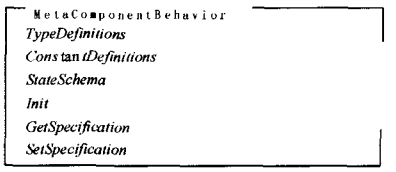
(6)元组件接口元素参数模式 MetaComInterfaceElemPara



(7)元组件行为类 MetaComponentBehavior

组件的行为元信息 ComponentBehaviorMetaInfo 描述组件的行为方面的元信息。它进一步包括行为规格元信息 SpecificationMetaInfo 以及可变性 Changeable。组件行为规约元信息(SpecificationMetaInfo)。C2SADEL 组件行为是由一组操作(operation)描述的。组件中定义了 map 成份,它定义了一组映射规则,主要是说明组件行为中的每个操作与组件接口中接口元素之间的映射关系。SpecificationMetaInfo 又包括一组操作元信息 OperationMetaInfo 和不变式 Invariant。MapMetaInfo 分别描述映射规则中的每一个与操作相关的映射关系的元信息。ParameterMapMetaInfo 描述了操作和接口元素的参数之间的映射关系的元信息。

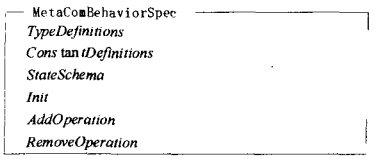
根据上述元组件结构元信息,元组件结构类 MetaComStructure 描述如下,其中声明的分量元组件行为规约是元组件行为规约类 MetaComBehaviorSpec 的变量。



元组件行为类 MetaComponentBehavior 的分量 metacombehaviorspec 属于元组件行为规约类 MetaComBehaviorSpec 的类型,元组件行为操作模式 MetaComBehaviorOper 包含元组件行为规约模式 MetaComBehaviorSpec 的分量 metacombehavioroper 属于元组件行为操作模式 MetaComBehaviorOper,元组件行为操作变量模式 MetaComBehaviorOperVar、元组件行为的操作与元组件接口元素的映射关系模式 MetaComBehInterfaceMap 所分别定义的变量 metacombehavioropervar、metacombehInterfacemap 是元组件行为操作类 MetaComBehaviorOper 的分量。

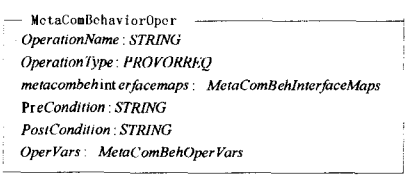
(8)元组件行为规约类 MetaComBehaviorSpec

元组件行为规约 MetaComBehaviorSpec 包含若干个元组件行为操作 MetaComBehaviorOper。

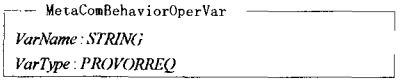


(9)元组件行为操作模式 MetaComBehaviorOper

每个元组件行为操作 MetaComBehaviorOper 包含若干个元组件行为操作变量 MetaComBehaviorOperVar 和若干个元组件行为与接口的映射关系 MetaComBehInterfaceMap。

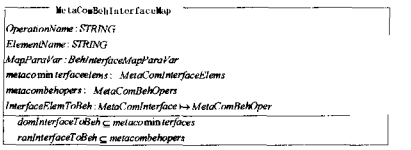


(10)元组件行为操作变量模式 MetaComBehaviorOperVar



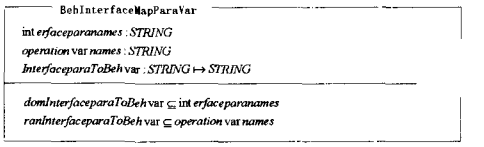
(11)元组件行为的操作与元组件接口元素的映射关系模式 MetaComBehInterfaceMap

元组件行为与接口的映射关系模式 MetaComBehInterfaceMap 的形式化描述如下,其中定义了从元组件接口元素到元组件行为的操作的映射关系。



(12)接口元素的参数和操作的变量之间的映射关系模式 MetaComBehOperMapPara

$Interfaceparatobehavr: STRING \mapsto STRING$ 定义了接口元素的参数和操作的变量之间的映射关系。该映射关系模式可以定义如下:



(13)元组件约束模式 MetaComConstraint 和元组件属性模式 MetaComProperty 的定义省略。

2.2.3 元连接器描述

(1)*Architecturalconstituent* 是元连接器的超类,用来描述体系结构的构成。

元连接器类 MetaConnector 的分量包括元连接器基本特征类 BasicFeature、元连接器结构类 MetaConnStructure、元连接器行为类 MetaConnBehavior、元连接器约束类 MetaConnConstraint 和元连接器属性类 MetaConnProperty 所定义的分量。

(2) 元连接器结构类 MetaConnStructure

连接器接口元信息(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。SignatureMetaInfo 的作用是描述连接器接口签名的元信息。

根据上述元连接器结构元信息,元连接器结构类 MetaConnStructure 定义如下:

(3)元连接器接口模式 MetaConnInterface

(4)元连接器行为模式 MetaConnBehavior、元连接器约

束模式 MetaConnConstraint 和元连接器属性模式 MetaConnProperty 的描述省略。

3 反射式软件体系结构 Object-Z 规格说明的初始化定理及其证明

选取元级元素模型基于 Object-Z 形式化描述的一个代表性的状态模式，给出它的初始化定理及其证明过程，以此为示范来证明本文给出的反射式软件体系结构基于 Object-Z 规格说明的正确性。这里我们选取元组合件配置链接状态模式 Link。

元组合件配置链接状态模式 Link 和链接初始状态模式 InitLink 的初始化定理为：

$\vdash \exists Link' \cdot InitLink$

这个定理的含义是，确实存在一个 $Link'$ 系统状态满足 $InitLink$ 的谓词。

该定理可以展开为：

$\vdash \exists metacomponents' : \mathbb{F} MetaComponents;$
$metaconnectors' : \mathbb{F} MetaConnectors;$
$metaconns\_top' : \mathbb{F} MetaConnectors;$
$metaconns\_down' : \mathbb{F} MetaConnectors;$
$LinkComConn' : MetaComponent \rightarrowtail MetaConnector;$
$LinkConnConn' : MetaConnector \rightarrowtail MetaConnector$
$\mid domLinkComConn' \subseteq metacomponents'$
$\wedge\ ranLinkComConn' \subseteq metaconnectors'$
$\wedge\ domLinkConntop' \subseteq metaconns\_top'$
$\wedge\ ranLinkConndown' \subseteq metaconns\_down'$
$\cdot\ metacomponents' = \emptyset \wedge metaconnectors' = \emptyset \wedge metaconns\_tops' = \emptyset$
$\wedge\ metaconns\_down' = \emptyset \wedge LinkComConn' = \emptyset \wedge LinkComConn' = \emptyset$

根据定律$(\exists J \mid P \cdot Q) \Leftrightarrow (\exists J \cdot P \wedge Q)$可删除竖杠$\mid$，因此上面的展开式等价于：

$\vdash \exists metacomponents' : \mathbb{F} MetaComponents;$
$metaconnectors' : \mathbb{F} MetaConnectors;$
$metaconns\_top' : \mathbb{F} MetaConnectors;$
$metaconns\_down' : \mathbb{F} MetaConnectors;$
$LinkComConn' : MetaComponent \rightarrowtail MetaConnector;$
$LinkConnConn' : MetaConnector \rightarrowtail MetaConnector$
$\cdot\ domLinkComConn' \subseteq metacomponents'$
$\wedge\ ranLinkComConn' \subseteq metaconnectors'$
$\wedge\ domLinkConntop' \subseteq metaconns\_top'$
$\wedge\ ranLinkConndown' \subseteq metaconns\_down'$
$\wedge\ metacomponents' = \emptyset \wedge metaconnectors' = \emptyset \wedge metaconns\_tops' = \emptyset$
$\wedge\ metaconns\_down' = \emptyset \wedge LinkComConn' = \emptyset \wedge LinkComConn' = \emptyset$

在该式中，存在量词变量是 $metacomponents'$，$metaconnectors'$，$metaconns\_top'$，$metaconns\_down'$，$LinkComConn'$ 和 $LinkConnConn'$，都为空集。因此我们就可以 6 次使用点规则，从而获得一个更简单的等价式。要应用点规则，就必须：

(1) 说明用来替换的项是正确类型的对象；

(2) 对谓词中这些变量的出现进行替换。

由(1)，可以得到一个新的等价式中的 6 个部分：

$\emptyset \in \mathbb{F} MetaComponents;$
$\emptyset \in \mathbb{F} MetaConnectors;$
$\emptyset \in \mathbb{F} MetaConnectors;$
$\emptyset \in \mathbb{F} MetaConnectors;$
$\phi \in MetaComponent \rightarrowtail MetaConnector;$
$\emptyset \in MetaConnector \rightarrowtail MetaConnector$

由(2)，可以得到进一步的限制：$dom\emptyset \subseteq \emptyset \wedge ran\emptyset \subseteq \emptyset \wedge \emptyset = \emptyset$。

所以，前面所描述的初始化定理现在就成了：

$\vdash \emptyset \in \mathbb{F} Component\_instances \wedge \emptyset \in \mathbb{F} Connector\_instances \wedge \emptyset \in \mathbb{F} Connector\_instances$
$\wedge\ \emptyset \in \mathbb{F} Connector\_instances \wedge \emptyset \in Component\_instance \rightarrowtail Connector\_instance$
$\wedge\ \emptyset \in Connector\_instance \rightarrowtail Connector\_instance \wedge dom\emptyset \subseteq \emptyset \wedge ran\emptyset \subseteq \emptyset \wedge \emptyset = \emptyset$

第 1 个到第 4 个子目标可由定律 $\emptyset \in \mathbb{F} S$(L24)立即得到证明，第 5 个和第 6 个子目标可由定律 $\emptyset \in S \rightarrowtail T$(L13)得到证明，第 7 个子目标可由定律 $dom\emptyset = \emptyset$(L6)得到证明，第 8 个子目标可由定律 $ran\emptyset = \emptyset$(L7)得到证明，最后 $\emptyset = \emptyset$ 显然成立。所以定理成立，即对于该规格说明，初始状态是存在的，也就是说我们验证了该规格说明的正确性。

结束语 本文的主要贡献：详细给出了基于反射机制 RMRSA 的反射式软件体系结构 ReflectiveArchitecture 的元级体系结构模型；用形式规格说明语言 Object-Z 对元级体系结构模型进行了完整的描述；选取元组合件配置链接状态模式 Link，给出了它的初始化定理及其证明过程，以此为示范证明了反射式软件体系结构 Object-Z 规格说明的正确性。

今后的研究工作：基级体系结构模型和元级体系结构模型中类模式的操作模式的定义及形式化；提供元信息模型信息结构，提供相应的方法来描述这些随着设计方案(体系结构)不同而在内容上完全不同的元信息；基于反射式软件体系结构，对软件可信性进行管理和验证的研究。

参 考 文 献

[1] Binns P, Engelhart M. Vestal. Domain-Specific Software Architectures for Guidance, Navigation, and Control [J]. Software Eng. and Knowledge Eng., 1996, 6(2): 1011-1017

[2] Shaw M. Some Patterns for Software Architecture, Pattern Languages of Program Design [M] // Vlissides, Coplien and Kerth, eds. Addison-Wesley, 1996: 255-270

[3] Schmerl B, Garlan D. AcmeStudio: Supporting Style-Centered Architecture Development [C] // Proceedings of International Conference on Software Engineering. Edinburgh, Scotland, May 2004

[4] Froehlich G, Hoover H J, Liu Ling, et al. Designing object-oriented frameworks, In CRC Handbook of Object Technology [M]. CRC Press, 1998

[5] 罗巨波，应时. 一种支持软件体系结构重用的反射机制及其形式化[J]. 计算机科学，2009，36(8)：145-148