计算机科学1999Vol. 26№. 10

# 软件体系结构的可视化描述与实现*)

Visual Description and Implementation of Software Architecture

于 卫 杨卫东 蔡希尧

(西安电子科技大学软件工程研究所 西安710071)

**Abstract** The key of software system reuse is the design and specification of overall system structure. This paper combines UML and software architecture description language, presents the core model of software architecture. Finally, an example implemented in ROSE98 is given.

**Keywords** Software reuse, UML, Software architecture, Architecture description language, Component, Connector, Domain specific software architecture

面向对象的软件开发方法使用对象或类作为建模的基本实体,将消息传递作为对象间通讯的基本机制,描述了软件建模的一般概念、相关建模技术及一般设计指导规则。尽管这种方法也可以描述一些软件体系结构的设计问题,但与软件体系结构及其相关工具相比,有很大差距,因为后者以构件和连接作为基本的建模实体,对构件和连接提供了多重接口,可以描述构件间丰富的交互语义,较好地支持软件系统一级的重用,可以分析软件的某些特性(如:死锁检测等)。当前软件体系结构的描述方法主要有框线图和ADL(体系结构描述语言),但难以被开发人员理解,并且没有应用于工业界。如果将软件体系结构与面向对象的开发方法统一起来,则既可以在软件开发的设计阶段分析软件的某些特性、重用软件的整个组织结构,又可以利用面向对象方法的支持工具。

本文以UML作为基础,利用UML的半形式化特性(OCL、对象约束语言)和嵌入的扩充机制描述了软件体系结构。

## 1 UML及其可扩充机制

软件体系结构所支持的构件及构件之间的语义非常丰富,仅用图和自然语言的方法(如:OMT方法、Booch方法等)很难描述清楚。统一建模语言(UML)是当前主流的面向对象软件开发方法,它的一个重要的特点是引入了形式化定义(对象约束语言OCL),有利于描述软件体系结构,同时UML又具有很好的扩充机制,众多的工具支持,且与具体程序设计语言和开发过程无关,因此我们选择UML做为描述软件体系结构的基础。

UML是一种明确定义了语法和语义的可视化建模语言[1],它基于主流的软件开发方法及开发经验,语法和语义由元模型、自然语言和约束来说明。UML是四层元模型的体系结构,即用户对象层、模型层、元模型层、元元模型层。UML的结构主要体现在元模型中,分为三个逻辑包:基础包、行为元素包、一般机制包,这些包依次又分为若干个子包。语义约束用对象约束语言(OCL)表示,OCL基于一阶谓词逻辑,每一个OCL表达式都以UML模型元素为背景。UML模型从以下几个方面说明软件系统:类及其属性、操作和类之间的静态关系,类的包和其依赖关系,类的状态及其行为,对象之间的交互行为,使用事例,原码的结构以及执行时的实施结构。

UML提供了丰富的建模概念和表示符号以满足典型的软件开发。但是用户有时需要另外的概念或符号来表示其特定领域的需求,因此需要UML具有一定的扩充能力,UML提供了三种嵌入的扩充机制:Stereotypes、Constraints、Tagged values。

(1)Stereotypes是UML中最重要的扩充机制,提供了一种在模型层中加入新的建模元素的方式,可以在Stereotypes中定义相关的Constraints和Tagged values以说明特定的语义和特征。定义Stereotypes必须满足以下规则:

- Stereotype名不能与其基类重名;
- Stereotype. oclAllInstances → forAll (st | st.

*)本文得到国防科技预研基金项目《C3I系统应用软件开发工程化研究》(6.2.1.4)的支持。于卫 在职博士生。杨卫东 博士生。蔡希尧 教授、博士导师。

baseClass()self. name);

● Stereotype 名不能与它所继承的 Stereotype 重名;

● self. allSupertypes→forAll(st:Stereotype|st. name()self. name);

● Stereotype 名不能与元类命名空间冲突;

● Stereotype 所定义的 Tag 名不能与其基类元素的元素性命名空间冲突,也不能与它所继承的 Stereotype 的 Tag 名冲突。

(2)对于模型中的建模元素,Tag values 允许加入新的属性,Tag 表明了建模元素的可扩展特性的名称,value 可以是任意的值,值的范围取决于用户或工具对 tag 的解释。对每一个 Tag 名,一个建模元素至多有一个给定的 Tag Valul。

self. tagged Value→forAll(t1,t2:TaggedValue|
t1. tag=t2. tag implies t1=t2)

(3)Constraints 是对建模元素的语义上的限制。

## 2 用 UML 描述软件体系结构

随着软件系统的复杂度和规模的增加,整个系统结构的说明和设计显得更为重要。软件体系结构在较高层次将系统定义为一组相互交互的构件和连接,包括系统各构件的组织、全局控制结构、通讯的协议、设计元素的功能、物理分布等。在目前通用的软件开发方法中(OMT, Booch, Fusion, UML 等),其描述通常是用非形式化的图和文本,难以被开发人员理解,更不能用来分析其一致性和完整性等特性。针对这些问题,一些工业界和学术界的研究者提出了体系结构的形式化描述:体系结构描述语言(ADL),它可以用来表示和分析体系结构的设计,同时提供了显示、分析、模拟软件体系结构的相应工具。目前主要的体系结构描述语言有:Aesop、Meta-H、C2、Rapide、SADL、Unicon 和 Wright,尽管它们都描述软件体系结构,却有不同的特点:Aesop 支持体系结构风格的应用,Meta-H 为设计者提供了关于实时电子控制软件系统的设计指导,C2 支持基于消息传递风格的用户界面系统的描述,Rapide 支持体系结构设计的模拟并提供了分析模拟结果的工具,SADL 提供了关于体系结构加细的形式化基础,Unicon 支持异构的构件和连接类型并提供了关于体系结构的高层编译器,Wright 支持体系结构构件之间交互的说明和分析。这些 ADL 强调了体系结构不同的侧面,对体系结构的研究起到了重要作用,但也有负面的影响,每一种 ADL 都以独立的形式存在,描述语法不同且互不兼容,这使设计人员很难选择一种合适的 ADL,若设计特定领域的软件体系结构(DSSA)又需要从头开始描述,因此,我们将它们的共同特征提取出来,作为体系结构描述的核心模型,其优点在于:可以作为各种 ADL 在 UML 中进行描述的共同基础,若要描述某一 ADL,只需描述与其相关的约束,也为描述 DSSA 奠定了基础。

### 2.1 软件体系结构的核心模型

体系结构的核心模型由五种元素组成:构件、连接、配置、端口和角色,其中构件、连接和配置是最基本的元素:

●构件是具有某种功能可重用的软件模板单元,表示了系统中主要的计算元素和数据存储。构件有两种:复合构件和原子构件,复合构件由其它复合构件和原子构件通过连接构成;原子构件是不可再分的构件,底层由实现该构件的类组成。这种构件的划分提供了体系结构的分层表示能力,有助于简化体系结构的设计。典型的构件如:client, server, filter, database 等。

●连接表示了构件之间的交互,简单的连接如:pipes, procedure call, event broadcast 等,更为复杂的交互如:client-server 通讯协议,数据库和应用之间的 SQL 连接。

●配置表示了构件和连接的拓扑逻辑和约束。

另外,构件作为一个封装的实体,只能通过其接口与外部环境交互,构件的接口由一组端口组成,每个端口表示了构件和外部环境的交互点,通过不同的端口类型,一个构件可以提供多重接口。一个端口可以非常简单,如过程调用,也可以表示更为复杂的界面(包含一些约束),如必须以某种顺序调用的一组过程调用。连接作为建模软件体系结构的主要实体,同样也有接口,连接的接口由一组角色组成,连接的每一个角色定义了该连接表示的交互的参与者,二元连接有两个角色,如:RPC 的角色为 caller 和 callee, pipe 的角色是 reading 和 writing,消息传递连接的角色是 sender 和 receiver。有的连接有多于两个的角色,如事件广播有一个事件发布者角色和任意多个事件接受者角色。

### 2.2 在 UML 中的描述

使用 UML 描述软件体系结构,可以通过定义新的元素做为 UML 元模型中某元类的子类,但是这种方法需要修改 UML 元模型,难以获得 UML 相关工具的支持,而使用 UML 嵌入的扩充机制对 UML 进行定制,则可以避免修改 UML 元模型。因此,我们首先在 UML 中选择与软件体系结构元素语义相近的元类,定义其上的 StereoType 做为该元类的实例,并用对象约束语言(OCL)描述软件体系结构的约束,这样就可以与 UML 元模型一致并可重用现有的 UML 相关工具。

为了使体系结构具有分层的表示能力以减少设计复杂度,构件分为复合构件和原子构件,由于软件体系

结构的描述建立在UML之上，以对象模型作为基础，原子构件的下一层是实现该构件的类，因此，我们把构件定义为UML中元类"Package"的实例。连接表示了构件之间的通讯机制。在某些软件体系结构的研究中(如：Unicon等)，连接只是概念上的抽象，其具体实现嵌入在构件之中，一般没有相应的实体与之对应(如Unix构件从管道中读、写)，而连接作为体系结构建模的重要实体，也应该独立出来以描述复杂的交互和利于重用[4]，因此，我们把连接定义为UML中元类"Class"的实例。软件体系结构的约束在UML中定义如下：

(1)软件体系结构仅由其构件元素构成：

```
self. oclType. elements(forAll(e|
e. stereoType. =Component or
e. stereoType =Connector)
```

(2)每一个构件具有一个Tagged Value：kindOfComponent，标志它是原子构件或是复合构件。kindOfComponent：enum{Composite component，Primitive Component}

(3)构件的接口由一组端口组成：self. oclType. interface→forAll(1|1. storotype=port)

(4)构件只能通过端口与其它连接相关联，而不能与其它构件相关联。self. oclType. assoEnd. stereoType=Port

(5)连接通过角色与其他构件相关联。self. oclType. assoEnd. stereoType=Role

(6)每一个构件不能没有端口，self. oclType. assoEnd. port(size)>0

(7)每一个连接不能没有角色。self. oclType. assoEnd. role(size)>0

(8)构件在执行时可以有多个实例。self. allInstances(size)>=1

(9)每一个连接至少与两个构件相连。self. oclType. assocEnd(size)>=2

(10)复合构件的子构件只能是由连接相连的复合构件或原子构件。

```
Let ips=self. subComponents
ips(forAll(c|c. kindOfComponent = Composite component or c. kindOfComponent = Primitive Component)
```

(11)原子构件不能再包含其它构件(原子构件或子构件)。self. subComponents. size=0

(12)每一个Port至多只能与一个连接关联。

```
Let comps = self. oclType. elements(select(e | e. stereoType=Component)
comps. assicEnd(for All(ae|ae. linkEnd(size=1))
```

(13)软件体系结构的构件和连接不参加其语义范围以外的任何关联。

```
Let comps = self. oclType. elements(select(me | me. stereoType=Component)
comps. assoEnd(forAll(a|a. stereoType=Port or
    a. stereoType=Component
      a. stereoType=Connector)
```

### 2.3 多视图描述

从多个视图描述软件体系结构时，每一个视图描述了软件体系结构的不同特征，有助于减少体系结构建模的复杂度，也有助于设计人员对体系结构的理解。文[2]提出了软件体系结构的4+1视图描述，文[3]在第五章中提出了多维设计空间的概念用来描述软件体系结构的不同特征，分为功能设计空间(表示功能和性能需求)和结构设计空间(表示系统的初始分解结果)，相比较而言，文[2]较为全面地描述了软件体系结构，但它以Booch方法的图形描述为基础，建立在对象模型之上，缺少形式化描述基础，因此我们借鉴文[2]，在UML的基础上，利用其扩充机制和对象约束语言，从逻辑视图、使用事例视图和物理视图等三个角度描述软件体系结构。其中逻辑视图表示了结构和行为方面的信息，软件体系结构由构件和连接组装而成，构件和连接的行为由状态图和交互图表示(文[5]详细论述了CSP到状态图的转换)；使用事例视图和物理视图类似于UML，区别在于它们是以构件而不是以类为基础。

## 3 军事指挥中心DSSA在ROSE98中的实现[6]

通过ROSE98的可扩充性接口，我们在ROSE98中实现了对软件体系结构模型的分层描述。图1表示了军事指挥中心模拟DSSA的顶层逻辑构件图。我们建构了一族解决军事指挥中心领域的应用系统的参考模型，包括四种预定义构件类型：特定应用构件、元数据构件、进程管理构件、数据发送与接收构件；五种预定义连接类型：管道(Pipe)，共享内存(Share Memory)，本地过程调用(LPC)，事件(Event)，分发(Dispatch)。

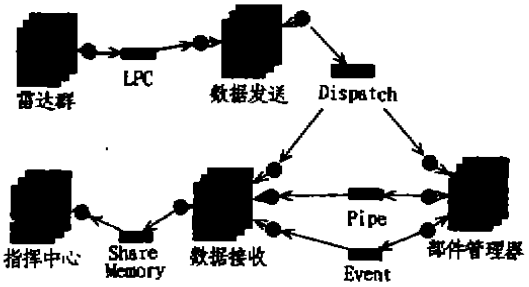


图1 军事指挥中心模拟DSSA的顶层逻辑构件视图

**结论** 软件体系结构与主流软件开发方法的结合提高了软件重用的抽象层次及软件的开发效率，本文

(下转第31页)

性——或者两者同时提交，或者两者都回滚，不会出现只有一项提交的情况)。这意味着系统不能保证工作流实例能够正常运行，也无法保证系统能够正常运转。而且，当工作流非正常中断时，工作流产品也无法保证能够正确恢复数据。已经有一些供应商考虑到了这个问题，并做了一些努力，但工作流产品在这方面还有待改进。

·工作流模型的分析和仿真工具：目前越来越多的用户要求能够对模型进行评估和特性测试，仿真和性能监控工具将不可避免地成为新一代工作流产品的发展趋势。

在应用方面工作流技术也将有很大的发展余地。我们仅以 CIMS 的信息环境为背景，讨论工作流产品应用的发展趋势：

·应用于并行工程中：在并行工程中，产品设计过程是工作流技术的典型应用。工作流技术可以描述其产品设计数据在设计组/设计者之间有序流动的主要流程，而且，对于不同部件/零件的产品数据间的复杂逻辑关系，工作流也可以通过不同设计活动之间的逻辑关系来表示。对于流程执行过程中的一些典型模式，如产品设计数据的评审、发布；产品设计、开发、试生产不同阶段的反馈；设计流程中的冲突协调等，工作流管理系统都能够进行有效地管理。

·应用于企业建模与过程集成：工作流管理系统能够对整个企业的运作方式进行建模，并可以对模型进行仿真、修改和优化。而且，工作流技术还可以实现过程集成，使企业能够提高效率，把握住机遇，提高市场竞争力。

**结论** 工作流技术综合了计算机科学和管理科学中多个研究领域的原理、方法和技术：数据库管理、C/S 技术、编程语言、图形化用户界面、系统集成、消息传递、文档管理、仿真等等。最近几年企业对于过程建模和 BPR 工具的需求为工作流提供了一个广阔的市场，使得工作流产品得以迅速发展。而且，工作流产品的供应商不断将信息技术、web 等研究中的最新成果应用于自己的产品开发中，使得其能够迅速普及。尽管如此，目前的工作流产品还存在很多问题有待解决。随着工作流技术的进一步发展，它必将在提高企业效率和竞争力，更好地适应市场变化等方面起到举足轻重的作用。

## 参 考 文 献


1 Workflow Management Coalition. Workflow Management Coalition Terminology & Glossary [WfMC000], 1994

2 Workflow Management Coalition. The Workflow Reference Model [WfMC1003]WFMC TC00-1003.1994

3 麦中凡，薛瑜. 工作流管理系统.概念、内容和现状. 计算机工程与应用，1999 增刊.24～28

4 Workflow Market Reference Point 97. Available at: URL: http. // www delphigroup. com/pubs/sample/WF-REFERENCE-1998-03. PDF

5 Mohan C. Recent Trends in Workflow Management Products, Standards and Research. Available at URL: http: // www. almaden. ibm. com/u/mohan/wfnato97. ps

6 Georgakopoulos D, Hornick M, Sheth A. An Overview of Workflow Management. From Process Modeling to Workflow Automation Infrastructure. Distributed and Parallel Databases, 1995.3(2):119～153

7 石伟，吴澄，范玉顺. CIMS 应用集成平台中的工作流技术研究. 清华大学学报，1998.38(8):125～128

8 Workflow/BPR Tools Vendors. Available at: URL: http: // www. waria. com/gw4wfven. html

9 Workflow and BPR Consultants. Available at: URL: http: // www. waria. com/gw3wfcon. html


(上接第 83 页)

讨论的描述方法对这一方面的研究作了有益的探索，具有一定的理论指导意义。

进一步的研究有：对 UML 描述的软件体系结构模型的进一步细化，如代码产生工具及软件体系结构分析工具等的研究与开发。

## 参 考 文 献


1 UML DOCUMENTS, version 1.1, URL: http: // www. rational. com/uml, 1997

2 Kruchten P B. The 4＋1 View Model of Architecture. IEEE Software, 1994

3 Show M, Garlan D. Software Architecture, perspectives on an emerging discipline, 1996, Prentice Hall, Inc

4 Garlan D. Higher-order Connectors. URL. http: // www. sei. cmu. edu, 1998

5 Robbins J E. Medvidovic N. Integrating Architecture Description Languages with a Standard Design Method, 1997

6 杨卫东，于卫，陈平. 软件体系结构在 ROSE 中的表示：[技术报告]. 西安电子科技大学软件工程研究所，1998