UML-RT 到一个图形设计环境体系结构的映射*)

刘晓燕[1] 张云生[1] J-J. Schwarz[2] 李俊昌[1]
(昆明理工大学信自学院 昆明 650011)
(LIRIS, UCB Lyon1, IUT A, 69622 Villeurbanne Cedex France)

摘 要 为解决把 UML-RT 建模模型平滑过渡到实时系统的图形化的软构件设计开发环境的设计模型,本文提出了从 UML-RT 的结构模型映射到该环境下的体系结构模型的高层设计的映射方法。首先介绍 UML-RT 结构建模的概念及笔者研制的设计环境,其次给出从 UML-RT 映射到该设计环境体系结构模型的具体方法、约束和限制。

关键词 UML-RT,图形化建模,软件体系结构,映射方法

Mapping UML-RT to Architecture of a Graphical Design Environment

LIU Xiao-Yan[1] ZHANG Yun-Sheng[1] J-J. Schwarz[2] LI Jun-Chang[1]
(School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650011)[1]
(LIRIS, UCB Lyon1, IUT A, 69622 Villeurbanne Cedex France)[2]

Abstract Introduces guidelines of mapping structure model of UML-RT to architecture of a graphical design environment component-based for real-time application in order to overcome easy translation UML-RT model into the high-level model of this design environment. First, concepts of UML-RT modeling structure and the design environment are presented. Finally, specific methods and constrains of mapping UML-RT model to the architecture description of this design environment are given.

Keywords UML-RT, Graphical Modeling, Software architecture, Mapping guidelines

1 引言

复杂实时系统软件体系结构建模已经放到了一个相当重要的位置,许多研究者根据他们的实践提出了不同的体系结构描述语言[1],并有相应的设计工具支撑体系结构的建模设计。UML 语言以图形化的形式支持面向对象的通用系统的静态结构和动态行为的建模。为了更好地为实时系统建模,Rational 公司结合了 ROOM(Real-Time Object-Oriented modeling)[2]的概念,作为 UML 的扩展,推出了 CASE 工具 UML for Real-Time,通常称为 UML-RT。UML-RT 为实时系统的面向对象的分析和设计提供了一组图形符号,用于为电信、航天、航空、国防、嵌入式实时系统和工业控制等复杂实时系统领域建模,描述系统的结构模型和行为模型[3],为复杂实时系统的建模提供一个相对完整的解决方案。

笔者研制了一个基于实时多任务操作系统的图形化的分布式实时软构件设计开发环境 DRSCDE(Distributed Real-time Software Component Development Environment),用于为分布式的具有客户/服务器(Client/Server)结构的实时应用系统建模,DRSCDE 提供了一组图形符号支持实时软件的结构建模以及基于实时多任务操作系统的行为建模[4~6]。

鉴于 UML-RT 应用于复杂实时系统的日益广泛性,为解决把 UML-RT 建模模型平滑过渡到实时多任务操作系统的图形化的软构件设计开发环境 DESCDE 的设计模型中,本文提出了从 UML-RT 的结构模型映射到 DRSCDE 的体系结构模型的方法,以解决两个模型之间高层设计的映射。文中首先介绍 UML-RT 结构建模的概念及 DRSCDE 设计环境,其次给出从 UML-RT 映射到 DRSCDE 体系结构描述的具体方法、约束和限制。

2 UML-RT 简介

UML-RT 是一个实时系统的面向对象的建模标准,用于诸如电信、航天和工业控制或分布式复杂实时系统等领域建模,描述系统的结构模型和行为模型。其对标准 UML 扩展了五个构造型,为复杂实时系统的体系结构建模提供方便。本节重点介绍 UML-RT 为实时系统体系结构建模的这组概念及相应的图形表示符号。各构造型图形表示符号如图 1 所示。

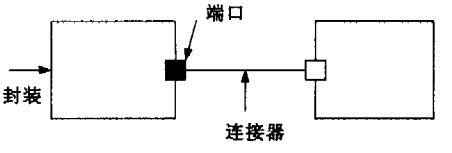

图 1 UML-RT 的各个构造型

(1)封装对象或封装(Capsules):是 UML-RT 用于表示复杂实时系统体系结构的主要元素,是复杂的客观存在的也可能是分布式的对象。它通过称为端口的接口对象和外部环境进行交互。封装对象由端口,连接器等基本构造子组成。

*)基金项目:云南省教育厅科学研究基金项目(04Y467D),云南省应用基础研究重点项目(2000F0004Z)。刘晓燕 博士生,副教授,主要研究方向为软件工程及软件开发环境。张云生 教授,博士生导师,主要研究方向为计算机控制及实时软件开发环境。J-J. Schwarz 教授,博士生导师,主要研究方向为软件工程及实时软件开发环境。

封装对象具有如下特征：(a)一个封装对象有一个或多个端口，通过端口和其它的封装对象通讯。除了端口之外，它没有操作或公共部分，端口是唯一的同外部世界交互的手段。(b)它总是一个活动对象。(c)它只能有一个状态机，该状态机通过该封装对象的终止端口发送和接收信号，用于控制内部结构的某些元素，因此该状态机被看作是实现所映射的行为，即控制封装对象自身操作的行为。(d)它可以包含一个或多个子封装对象，它们由连接器通过端口连接在一起。它与子封装对象有复合关系，这种内部的结构图称为 UML 的协作图。

(2)端口(ports)：是封装对象之间交互的一个中间体，是实现具体接口的对象。端口用于发送和接收信号或者是消息调用。它们被封装对象实例所拥有，随封装对象的创建而创建，和封装对象一起被销毁。每个端口有它自己的标识和状态，可以不同于动态的封装对象实例，既可以表示封装对象的结构的复合又可以表示行为(是封装对象的一部分)。

端口可以在某些协议中起着具体的角色作用。该协议角色定义了的端口类型，意味着该端口实现了相关协议角色的行为。

端口可分为两种：中继端口(relay ports)和终止端口(end ports)。中继端口是作为内部子封装的选择性输出“接口”；而终止端口是连接到封装的状态机上。它们的区别在于其内部连接：中继端口与子封装相连接，而终止端口是与封装的状态机的边界对象相连接。这两种端口都可位于封装的边界，并且它们是不可为外界所区分的。

(3)连接器(connectors)：连接器是将两个或多个端口互连起来的基于信号的通讯信道的一个物理对象。其功能仅仅是把信号从一个端口传递到另一个端口。连接器代表一个通信信道，为支持某个具体的基于信号的协议的传送提供方便。

(4)协议(protocols)：协议是对发生在连接器上的期望行为的规格说明即协议参与者间的契约式协定的明确的规格说明。这是对行为的说明而没有规定任何的结构元素。一个协议由一些参与者组成，每一个参与者扮演协议中的一个具体的角色。每个参与者都有唯一的名称，它们接收和发送信号(信号集也可以为空)。

(5)协议角色(protocol roles)：特指协议规格说明中的一方参与者，相当于 UML 的类元角色。

3 DRSCDE 的体系结构建模概念

随着面向基于构件的软件工程方法的应用，开发者越来越注重采用构件来开发应用系统，在实际开发应用系统中对软件构件设计工具的需要与日俱增。我们针对基于实时多任务操作系统的分布式实时应用系统设计开发了一个基于构件的图形化设计开发环境 DRSCDE，辅助基于构件开发的实时应用软件的架构设计、过程设计、中间语言描述文档及代码框架的自动生成，在图形设计过程中即产生相应构件的图形设计的规格说明及伪码语言描述文档。本节重点介绍该工具对高层的体系结构建模概念的定义。

3.1 构件

DRSCDE 针对分布式具有客户/服务器关系的实时应用系统建模。把对象或构件分散布置在客户机上和服务器上，是客户/服务器模型的特征。这些对象或构件被称作分布式对象或构件。服务器构件有可访问的接口，通过接口定义其提供的服务，客户构件可以发送服务请求，请求服务器为其完成相应的功能。而某些构件有双重作用：既是客户又是服务器。

应用对象 AO(Applicative Object)是 DRSCDE 定义的基于客户/服务器关系的高层类对象构件，是分布式实时应用构件的表示。刻画了分布式实时构件典型接口、功能、性能及其状态(激活和撤销等)，包含一系列活动，是可编程的。可由其它 AO 或底层原子对象复合构成，是个活动对象。在实际应用中，AO 可以是系统、子系统、对象或第三方开发的具有独立功能概念的部件。

AO 图形符号的定义(图 2)是用矩形代表构件，并由如下几个部分组成：(a)客户域：是向其它 AO 发送服务请求的区域，由此发送服务原语。(b)服务域：是服务请求到达的区域，代表 AO 对外部提供的服务接口。由服务原语来存取。由虚线矩形表示。(c)终端器：抽象地表示被实时应用系统监控或控制的外部实体，如代表现实世界中的监视器、传感器等。由实线矩形表示。(d)状态域：矩形的四边定义状态域，在这使用状态原语。矩形边界可被服务域以及终端器部分覆盖。

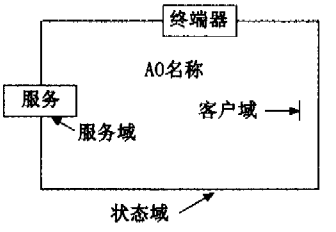

图 2 分布式实时应用构件 AO 的图形表示

3.2 服务原语及状态原语

AO 之间的交互协作关系可以被明确地表示为客户/服务器(C/S)关系。AO 之间的通信及动态连接关系由服务原语及状态原语表示。

服务原语表示访问构件提供的服务，其图形表示是客户域和服务域之间的直线连接，这条直线称为连接器。当客户构件通过连接器连接服务器构件时，就表示客户方请求服务器构件提供服务，这两个构件之间建立了 C/S 关系。

状态原语表示在实时应用环境中系统运行时对实时构件的动态调度行为。对所有的实时构件都是共用的。状态原语有：激活原语、删除原语、挂起原语、恢复原语、连接原语、撤销原语等。每个状态原语都有相应的图形表示。例如激活原语的图形表示是从一个 AO 的状态域指向另一个 AO 的状态域的单向箭头。

3.3 框架层

框架层定义 DRSCDE 软件设计上层的总体架构的图形表示，相当于软件设计中的体系结构图，定义组成分布式实时应用系统的构件及其分布性以及各构件之间的通信连接关系。通常要确定构件的分布性，规定构件的接口即向本地或远程构件对象所提供的服务集(功能或行为)，及构件之间的关系(复合或包容)，由服务原语和状态原语规定的构件对象之间的动态的相互通信关系及调度关系。

4 UML-RT 映射到 DRSCDE 的策略

UML-RT 提供一组对面向对象体系结构建模的概念及图形表达符号以及映射到实现的机制，见图 1 。已开发出相应的工具软件 RoseRT 来支持它。DRSCDE 是我们研究开发的基于构件的实时应用的图形化设计环境，定义了一组面向构件的体系结构的图形建模语言，见图 2。本文重点讨论

从 UML-RT 到 DESCDE 体系结构建模语言的手工映射方法。相应的映射工具在开发中，以支持两个开发工具的平滑过渡。

4.1 映射规则

我们考虑 UML-RT 的主要建模元素封装对象、端口和连接器等到 DRSCDE 中的应用对象、服务原语、状态原语等建模元素的映射关系。

UML-RT 中的封装对象是结构建模中的基本元素，是一个活动对象。一般而言，可将其直接映射为 DRSCDE 中的构件应用对象 AO。两个元素均表示实时系统中可动态建模的活动对象，均有接口及连接协议，可做层次化分解。UML-RT 中封装对象在协作图中的实例映射为 DRSCDE 的框架层复合图中的实例。

UML-RT 表示的实时系统外部设备的封装映射为 DRSCDE 中的终端器，外部设备在 DRSCDE 中建模为跨接在 AO 边界上的一个类对象，与该 AO 有直接关联关系，表示由该 AO 直接控制的外部硬件设备类。

UML-RT 的端口，一般而言映射为 DRSCDE 的服务域或客户域。二者皆为和其它对象交互的接口。由于 UML-RT 的端口分为终止端口和中继端口两类，我们从这两个方面来考虑映射：

1)终止端口：表示连接链的终止，是封装的状态机的边界对象。若带有状态机图符的终止端口表示硬件设备的封装时，则此终止端口映射为相应 AO 关联的终端器，不能映射为服务域。

2)中继端口：是和子封装连接的端口。为避免封装同子封装之间交互时线的交叉而引入的，负责信号的中继转播。有如下几种情况：

• 跨在子封装的边界上的中继端口，表示其对该封装的其它子封装交互，映射为 DRSCDE 中 AO 的服务域或客户域，这有不确定性，需要根据语义具体推断。

• 跨在子封装的边界上并和包含它的封装的中继端口相连的中继端口，表示该子封装同外部其它封装的交互，因而映射为 DRSCDE 中的 AO 客户域或服务域，表示向外发送服务请求或提供服务。

• 跨在子封装边界上的中继端口，它和一个带有状态机图符的表示硬件设备封装的终止端口相连，则此中继端口在转换时略去。

• 跨在封装边界上的中继端口，表示该封装对其它封装的信号传播，此中继端口在转换时略去，而将其中继的子封装的中继端口映射为 DRSCDE 中的服务域或客户域。

UML-RT 中的连接器映射为 DRSCDE 中的连接器，都是表示活动对象之间消息的发送和接收，表示对象之间的协作关系。

UML-RT 中的协议映射为 DRSCDE 中的 C/S 关系协作：服务原语。均表示活动对象之间的交互行为。

UML-RT 中的协议角色映射为 DRSCDE 中的服务原语类型。

UML-RT 中的协作图封装和具有标记{plug-ins}的封装相连的协议同时映射为 DRSCDE 中对应 AO 构件的服务原语类型和状态原语。标记{plug-ins}是 UML-RT 表示在运行时动态插入的封装对象，同时具有协议角色和动态调度的含义，因而有 AO 构件的服务原语类型和状态原语的语义。

UML-RT 的结构协作图映射为 DRSCDE 中的标题图和框架层结构图。

4.2 UML-RT 映射举例说明

为了说明映射方法，我们以一个已开发的某企业分布式实时数据采集及监控系统为例。该应用软件系统的主要功能是由若干台传感器进行分布式数据采集，实时数据入库及实时传输到监控中心。监控中心对数据进行综合分析处理，实施相应的控制。

该系统的 UML-RT 的结构模型如图 3 所示(以协作图表示)。我们采用前面所述的映射策略来说明两个模型的转换。

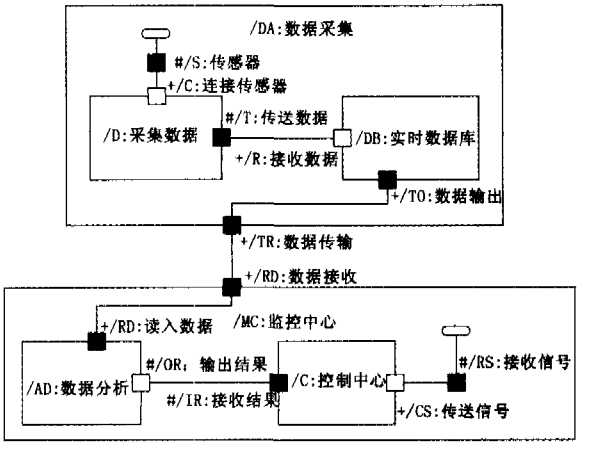

图 3 数据采集及监控系统的 UML-RT 结构模型

图 3 中的封装 DA 数据采集变换为 DRSCDE 模型(图 4 所示)中的分布式的应用对象 AO 实例数据采集。图 3 中的子封装 D 采集数据及 DB 实时数据库变换为图 4 中的子 AO：采集数据及实时数据库。终止端口 S 映射为子 AO 采集数据的命名为传感器的终端器。子封装 D 采集数据的中继端口 C 在转换时略去，D 的中继端口 T 和 DB 的中继端口 R，根据语义映射为相应 AO 采集数据的服务域和实时数据库的客户域。图 3 中的子封装 D 采集数据及 DB 实时数据库的连接器映射为图 4 中的相应 AO 采集数据及实时数据库之间的连接器，该连接器上的协议映射为两个相应 AO 之间的服务原语。封装数据采集 DA 的中继端口 TR 表示信号的中转，转换时略去。封装监控中心的变换采用同样方法可得。

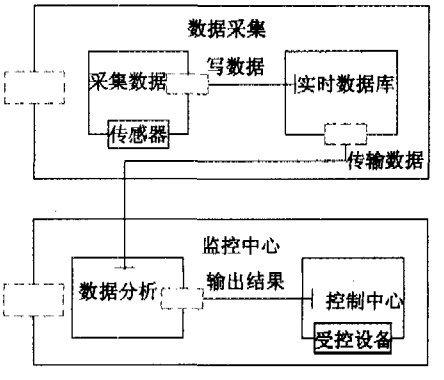

图 4 数据采集及监控系统的 DRSCDE 结构模型

5 不匹配规则

上节中我们给出了 UML-RT 和 DRSCDE 中大多数的元素可以直接映射的规则，但是仍然有一些建模元素在语义和使用方面不匹配。在本节中，将给出这些不匹配的说明和一些约束和限制。

一个 UML-RT 封装体内部的协作图由子封装实例显示了彼此之间的交互协作关系，表示该封装体和这些子封装实

例构成了复合关系，包括了动态建立的子封装，标记为{plug-ins}。因此，在给定类层次的封装只能有一个协作图。然而，DRSCDE中的AO构件可以出现在不同的框架层结构图中，也可以由若干个子AO构件组成。所以，UML-RT的协作图并没有提供像框架层这样的灵活性。

UML-RT的封装对象的协作图表示封装的分解，每个子封装是一个类而不是一个实例，所以分解的协作图表示子封装实例的一个样式(pattern)。语义上有点微弱区别的是DRSCDE的框架层通过复合关系支持AO构件的层次分解，但不是样式，表示的是构件实例。

在UML-RT中，某些标记为{plug-ins}的子类表示在封装的协作中在运行时动态引入的，和该子封装的连接协议相当于DRSCDE中的状态原语中的创建原语，但是UML-RT中的封装的动态调度关系没能显示表示，因而与DRSCDE中的状态域及状态原语在语义上不能精确匹配。

封装中带有和状态机相连的终止端口，在模型中显示的表示其动态行为，在DRSCDE中没有相应语义的建模元素。在DRSCDE中AO构件的动态行为由详细层表示，详细层定义了基于实时多任务操作系统的过程设计的抽象对象。如任务、邮箱、信号量等。由于篇幅所限，就不在此详细讨论从状态机到详细层的映射。

在UML-RT中，封装之间的通讯关系表示广泛意义的信号的接收和发送，而DRSCDE中的AO构件之间的通信关系显示表示C/S关系，语义相对较窄，因而我们在两个模型之间的通讯关系映射时作了限制，将UML-RT的通讯关系约束为DRSCDE中的C/S关系。

在UML-RT中，跨接在封装边界的端口既没有和外部其它封装的连接也没有和内部其它子封装的连接，则无法确定此端口是终止端口或是中继端口，其语义是模糊的具有不确定性，因而在语义上无法匹配DRSCDE中的服务域或状态域或是客户域，此为限制之一。另外UML-RT的端口的多重性(二元性或多元性)指示端口实例的个数，DRSCDE也不支持。

结束语 本文提出了从UML-RT的结构模型映射到DRSCDE的体系结构模型的方法、约束和限制，以解决两个模型之间高层设计的映射，通过一个实例说明了映射的具体方法。将要做的研究工作是探讨从UML-RT的行为模型到DRSCDE的AO构件的可执行模型的映射方法以及AO构件在CORBA环境中的发布方法。

参 考 文 献

1 Shaw M, Garlan D. Software Architecture. 北京：清华大学出版社，1998
2 Lyons A. UML for real-time overview. April 1998 http://www.uml.org.cn/UMLApplication/pdf/umlrt_overview.pdf
3 Selic B, Rumbaugh J. Using UML for modeling complex real-time systems. March 11, 1998, http://www-128.ibm.com/developerworks/rational/library/content/03July/1000/1155/1155_umlmodeling.pdf
4 刘晓燕，张云生，等. 复杂实时系统软构件对象设计. 计算机工程与应用，2003,39(31)：119～121
5 刘晓燕，张云生，等. 基于构件的分布式实时系统架构图形化建模. 计算机应用研究，已录用
6 Schwarz J J, Maranzana M, Skubich J J, Martinez Y. Applicative Objects Interconnection in a Graphical Design for Real-Time Applications [C]. 20th IFAC Workshop on Real Time Programming (WRTP'95), 6-10 November 1995, Florida, USA
7 Booch G, Rumbaugh J, Jacobson I. UML参考手册. 姚淑珍，等译. 机械工业出版社，2001
8 Rational software Corporation. Rational Rose Real-time 2002. http://www.rational.com/product/rosert, 2002

(上接第269页)

节描述的算法遍历抽象实现结构图，可以得到影响程序环路复杂性的节点：WHL、FOR和IFT，它们的出现次数各为1；并且IFT的条件表达式为复合语句“Graph[Vertex][i]==1 && Visited[i]=0”，逻辑运算符“&&”的数目为1。所以 $m=1+1+1=3$。根据式(2)及表1，可计算出该算法的环路复杂性的值：$V(G)=V(G_{WHL})+V(G_{FOR})+V(G_{IFT})=2+2+(2+1)-(3-1)=5$。

通过两种不同的方法，计算得到的该广度优先搜索算法的环路复杂性的值都为5，结果一致。

结束语 对代码直接进行环路复杂性度量，需要为源程序构造控制流图。这要求先对源程序进行语法分析处理，然后再根据所提取的信息来生成控制流图。假设采用广为使用的自下而上的分析法-算符优先分析法[12]对需要进行度量的程序进行语法分析，则由文[12]可知，该方法包括算符优先文法及优先表的构造、算符优先分析算法、分析中的出错处理等，其时间复杂度至少为$O(n^2)$；控制流图实质上是一个有向图，假定采用十字链表[13]作为其存储结构，则构造控制流图算法的时间复杂度为$O(n+e)$。而3.2节讨论的基于过程蓝图的环路复杂性计算算法的时间复杂度为$O(n)$；并由文[12]有词法分析函数时间复杂度为$O(n)$。度量过程的复杂性分析和实例研究均表明在过程蓝图上对代码进行程序复杂性度量，其效率较传统方法要高。

环路复杂性度量是基于语法树的检测重复代码的Kontogiannis度量方法的主要度量技术之一[3]。因为过程蓝图本质是一棵抽象语法树，所以可以直接在过程蓝图上进行重复代码的检测，并避免源代码到抽象语法树变换这一繁复的过程。同时，应用本文所述的算法，可简化程序环路复杂性度量过程，从而提高该重复代码检测的效率，为重构自动定位提供更好的技术。

参 考 文 献

1 丁炎炎，等. 程序复杂度系统pgrmetrics的设计与实现. 计算机应用研究，2004,21(1)：167～169
2 王振宇. 程序复杂性度量. 北京：科学出版社，1997
3 van Rysselberghe F. Vingerafdrukken van code om duplicatie op te sporen：[Master Thesis]. Universiteit Antwerpen, 2001-2002
4 单永明. 一种源程序到控制流图的自动生成方法. 小型微型计算机系统，1996,17(10)
5 杜子德. 程序控制流图：一种可视化的程序设计工具. 计算机研究与发展，1995,32(12)
6 龚世生，刘建宾. 软件工程. 广州：广东高等教育出版社，1999
7 刘建宾，郝克刚. 抽象概念结构图到JAVA过程蓝图的平滑过渡及一致性. 计算机科学，2001,28(8)
8 刘建宾. 过程蓝图设计方法学. 北京：科学出版社，2005
9 刘建宾. JAVA过程蓝图. 计算机科学，2000,27(7)
10 Shaffer C A. 数据结构与算法分析：Java版. 张铭，刘晓丹，译. 北京：电子工业出版社，2001
11 黄国瑜，叶乃子. 数据结构：Java语言版. 北京：清华大学出版社，2002
12 陈火汪，刘春林，等. 程序设计语言编译原理：第3版. 北京：国防工业出版社，2000
13 严蔚敏，吴伟民. 数据结构：C语言版. 北京：清华大学出版社，1997