计算机科学 1998 Vol. 25№. 2

面向对象开发方法的最新进展*)

Newest Advances in Object-Oriented Development Methodology

姜鸿飞　全炳哲　金淳兆

(吉林大学计算机科学系　长春 130023)

摘　要　In this paper, we review the past and current situation of object-oriented development methodology. We also introduce the newest development tendency of object-oriented development methodology and Modeling Language which represents this tendency: UML (Unified Modeling Language)

关键词　Object-oriented development methodology, Modeling language

一、发展历史回顾

面向对象程序设计方法起源于六十年代末期的语言 SIMULA'67，到了七十年代末期，软件行业正受到软件危机的困扰，结构化的开发方法不能够很好地解决软件危机。面向对象语言 Smalltalk 的出现，进一步发展和完善了面向对象的程序设计语言，从此面向对象也和开发方法开始结合，出现了面向对象的开发方法。到八十年代末期，面向对象的开发方法已将近十种，著名的有 Shlaer-Mellor[9,10]、Coad/Yourdon[5,6]和 Booch'86[1]等方法，这些都是早期的，在一定程度上存在局限性。Shlaer-Mellor 方法把待开发的系统分为四个相互独立的领域(domain)：应用领域、服务领域、体系结构领域和实现领域，进行分别开发。该方法的开发过程没有明显区分分析和设计阶段。Coad/Yourdon 方法把系统的开发分为分析和设计两个阶段，分析阶段采用类、对象、结构、主题、服务、属性等概念来描述系统，设计阶段的主要工作则是细化分析阶段的成果。Coad/Yourdon 方法对系统的动态行为描述不很全面，另外对静态模型的表示也不很丰富。Booch'86 方法的侧重点在设计阶段。这是面向对象开发方法的早期发展阶段。

从 1988 年到 1994 年，面向对象开发方法的发展进入百花齐放阶段。新的方法层出不穷，先后有五十余种面向对象开发方法出现，其中涌现了一些相当出色的方法，如 OMT[8] 方法、OOSE[7] 方法、RDD[12]方法和 Booch'93[3]方法。这些方法都有自身独到的特点，并且在一定范围内具有较强的开发能力。

OMT(Object Modeling Technique)方法覆盖了应用开发的全过程，包括分析、设计和实现。在分析阶段，OMT 方法强调对系统和相关领域的理解，并在此基础上建立模型。通过分析，确立对象、关系、事件流和功能。分析阶段的模型由三部分组成：对象模型、动态模型和功能模型。对象模型反映系统的静态结构，通过对象图，显示系统内部的对象、对象间的关系、对象内部的属性和操作。动态模型反映系统的行为、控制等动态方面，它包括事件图和状态图。OMT 方法使用脚本(scenario)描述对象间的相互作用，事件图描述参与某个脚本的对象和事件；状态图描述系统中对象的各种状态以及触发它们之间相互转换的事件。功能模型使用数据流图描述对象操作的具体含义。OMT 方法的设计阶段由两个部分构成：系统设计和对象设计。系统设计负责划分子系统，确定系统的体系结构。对象设计的主要任务是细化分析阶段得到的模型，实现问题领域到计算机领域的转换。实现阶段的细节和具体的实现环境有关。OMT 方法突出的特点是在分析阶段，它可以较为全面地描述系统的静态结构，所以 OMT 方法适合于

*)本文研究得到国家"九五"攻关项目资助。姜鸿飞　硕士研究生，主要研究领域为面向对象开发方法与 CASE 工具。全炳哲　副教授，主要研究领域为软件自动化、面向对象技术、软件重用、软件工程。金淳兆　教授，博士生导师，主要研究领域为软件工程。

数据密集型的信息系统的开发。

OOSE(Object-Oriented Software Engineering)方法是一个使用事例(use case)驱动的方法,它建立的所有模型都是以使用事例模型为基础的。这种关系如图1所示。使用事例就是"用户和系统在一次相互作用中相关事务的一个特殊序列"。OOSE方法的主要特点是能够较好地描述系统的需求。该方法适合于商务处理方面的应用开发。

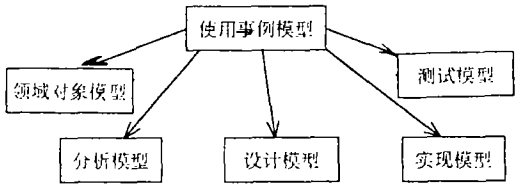

图1

RDD(Responsibility-Driven Design)方法强调对象行为以及对象间的关系,它用拟人的手法把对象的行为比喻为责任,对象间的关系比喻为合作,具有责任的对象比喻为合作者。RDD方法认为合作者在系统中的角色不是顾客就是服务者。顾客向服务者请求某项服务,服务者则根据不同的请求提供相应的服务。RDD方法把所有请求按照所请求服务的种类分类,组合成为合约(constract),每个合约都详细说明了相关服务的每一步操作。RDD方法把开发过程分为两个阶段:探查(Exploratory)阶段和分析阶段。探查阶段的主要任务是寻找类,确定责任和合作。分析阶段的工作主要是细化对象的行为和服务。具体包括定义类接口、实现操作规范、定义类的多态性和细化服务规范。RDD方法使用了CRC(Class-Responsibility-Collaboration)卡片来确定类的责任和它们的合作者。该方法对于小型项目具有较好的灵活性和适应性。但是对于大型系统,这个方法则显得力不从心了。

Booch'93方法把系统的开发工作分为两个部分:微观过程和宏观过程。微观过程主要用于建立一个反复、递增的开发框架,而宏观过程用来对微观过程进行控制。Booch'93方法的微观过程是由脚本和产品的体系结构驱动的,它由四步组成,如图2所示。

Booch'93方法的宏观过程控制着开发过程中的许多活动,这些活动有益于开发人员评估开发风险并且及时纠正微观过程中的各种错误。宏观过程关心的是开发过程中的管理方面。宏观过程由五步组成。如图3所示。

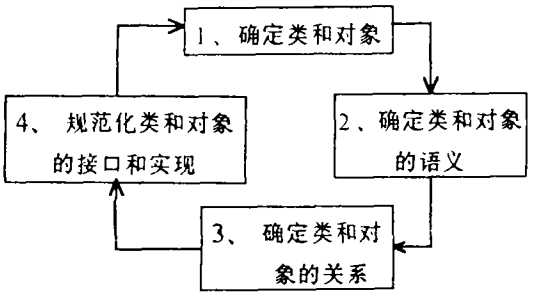

图2

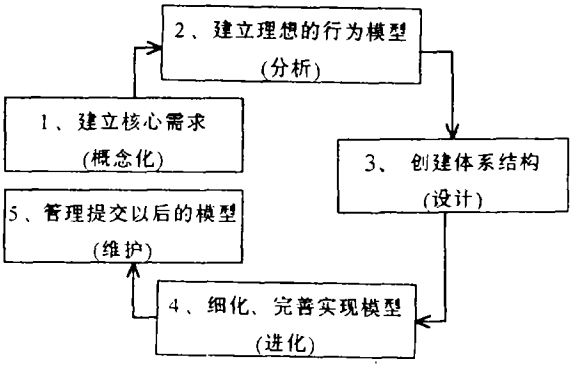

图3

Rooch'93方法沿袭了Booch'86方法的特点,它把工作集中在开发过程中的设计阶段,该方法对于开发的各阶段没有明确的划分,并且该方法没有规范需求说明书阶段。

进入九十年代中期,各种面向对象的开发方法开始取长补短,相互融合,出现了一些新的方法。另外,原有的一些面向对象方法也进行了修改或更新。这些方法,无论是原有的,还是新生的,都力求在表示手段、代表语义和开发过程上都有所改进。其中具有代表性的方法是VMT[11]方法。

VMT(Visual Modeling Technique)方法是IBM公司于1996年公布的。作为第三代面向对象开发方法,VMT方法结合了OMT,OOSE,RDD等方法的优点,并且结合了可视化编程和原型技术。VMT方法选择OMT方法作为整个方法的框架,并且在表示上也采用了OMT方法的表示。但是由于OMT方法中的功能模型采用的是数据流图,这与整体方法有些脱节,所以VMT方法摒弃了这部分内容,取而代之的是RDD方法中的CRC卡片这一表示手段,用它来定义各个对象的责任(操作)以及对象间的合作(关系)。此外,为加强对需求的分析,VMT

方法引入了 OOSE 方法中的使用事例概念，用以描述用户与系统之间的相互作用，确定系统为用户提供的服务，从而得到准确的需求模型。VMT 方法的开发过程分为三个阶段：分析、设计和实现。分析阶段的主要任务就是建立分析模型。分析模型中包括使用事例模型、分析阶段原型、对象模型、动态模型和 CRC 卡片。它们之间的关系如图 4 所示：

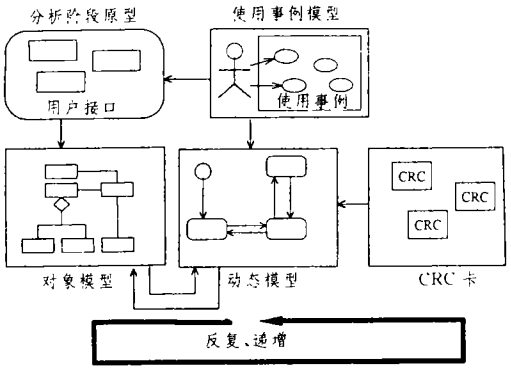

图 4

VMT 方法的设计阶段由三部分组成：系统设计、对象设计和永久性对象设计。系统设计的主要工作是划分子系统，设计系统的体系结构。对象设计的主要工作是从设计的角度细化分析阶段得到的对象模型，动态模型和 CRC 卡片。永久性对象设计主要考虑的是和数据库相关的设计。如果系统开发不涉及到数据库，则这一部分可以忽略。

VMT 方法的实现阶段的主要工作是用某一种环境实现系统。IBM 公司在推出 VMT 方法的同时也推出了支持 VMT 方法的可视化编程工具 VisualAge，通过它可以方便地实现设计阶段建立的模型。

二、新近发展

1. 统一化、标准化

软件在工业中的战略地位，要求软件生产尽快实现自动化。要想实现软件生产自动化，还需要解决许多问题。实现开发方法的统一化、标准化是基本的一条。实现面向对象开发方法的标准化以后，就可以避免不同方法的差异，提高软件重用效率。此外，统一面向对象开发方法，也就是综合各个面向对象方法的优点，最大限度地发挥各种方法的长处。另外，标准化后的面向对象方法也有利于在方法中加入其他先进的技术。

2. 软件开发方法中的表示手段、代表语义和开发过程的分离

随着对面向对象方法研究的逐渐深入，人们注意到开发过程受到开发人员、文化背景和问题领域的制约。适合于这个环境的开发过程可能就不适合于那个环境，所以开发过程的选择在很大程度上依赖于诸如问题领域、实现技术和开发小组等方面。由于开发过程的这种差异，使得面向对象开发方法的标准化工作很难进行。因此，有必要把开发过程从开发方法中抽取出来，这样，剩下的表示手段和代表语义就完全可以实现标准化了。表示手段和代表语义组合在一起就是建模语言。这种语言用来规范、表示、构造软件系统的模型。它必须包括三种成份：模型元素，用于建模的基本概念和语义；表示手段(Notation)，模型元素的直观表示；指南，建模语言的使用指导。

以上两种发展趋势的代表就是建模语言 UML (Unified Modeling Language)[4]。

三、建模语言 UML

建模语言 UML 最初仅仅是 OMT 方法、Booch'93 方法的统一。1995 年 10 月，Grady Booch 和 Jim Rumbaugh 联合推出了 Unified Method 0.8 版本(当时的名称)。这个方法力图实现 OMT 方法和 Booch'93 方法的统一。同年秋天，Ivar Jacobson 加入了 Booch 和 Rumbaugh 所在的 Rational 软件公司，于是 OOSE 方法也加入了统一的过程中。在 1996 年的 6 月和 10 月，Booch，Rumbaugh 和 Jacobson 所在的 Rational 软件公司推出了 UML 0.9 版本和 UML 0.91 版本。Rational 软件公司推出这两个版本的主要目的是希望得到各方面的反馈信息，进而改进 UML，推出正式的 UML 版本。UML 0.9 和 UML 0.91 版本推出以后，受到了普遍的关注，得到了许多计算机公司的支持，一些知名的公司加入到了 UML 1.0 版本的制定工作中。1997 年 1 月 13 日，UML 的正式版本 UML 1.0 问世。在 UML 1.0 问世的同时，它就被提交到了 OMG (Object Management Group)组织，希望在 1997 年的年中得到采纳，成为标准。

UML 作为一种建模语言，具有如下的特点：

• UML 结合了 Booch'93 方法、OMT 方法和 OOSE 方法的概念，是一个单一的通用的建模语言。UML 适合于使用上述三种面向对象方法或其他方法的用户。

• UML 的建模能力比其他面向对象方法更强，

不仅适合于一般系统的开发，更擅长于并行，分布式系统的建模，这是因为 UML 包含着描述这些领域的元素。

• UML 是一种标准的建模语言，而不是一个标准的开发过程。尽管 UML 必须应用在某种开发过程中，但它是完全独立于开发过程的。

UML 主要包括两部分内容：表示和语义。UML 的表示主要分两大类。一类是支持建立模型的各种图，另一类是通用的表示。UML 中支持建立模型的各种图及其相互关系如图 5 所示（图中的表示采用 OMT 方法中的聚合的表示法，即表示包含的含义）。

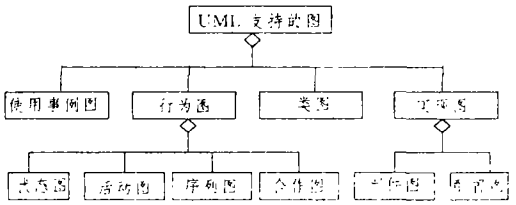

图 5

使用事例图和 OOSE 方法中的使用事例图相似，用来反映系统中使用事例和角色（actor）之间的关系。

类图展现了模型的静态结构，不仅包括模型中类和类型的内部结构，还包括它们之间的关系。UML 的类图以 OMT 方法和 Booch 方法中的类图（对象图）为基础，融入了其他方法中类图的优点。它的表示手段包括类和类的变体（模板和参数化类）、类之间的关系、类的属性和操作以及组织类的包（package）。

行为图展现了模型的动态结构，由四个图构成：状态图、活动图、序列图和合作图，这四个图分别展现了模型的动态结构的不同侧面。状态图显示一个对象或参与一个相互作用时各个对象在其一生中的状态转换，以及在状态转换过程中的响应事件和响应动作。UML 的状态图的表示以 David Harel 的状态机为基础，在其上进行了微小的修改。活动图是状态图的一个特例，它的状态是动作状态。所谓动作状态就是这些状态的转换必须由源状态中的 不可中断的动作触发。活动图的用途在于描述由工作流驱动的内部过程，它和其它方法包括非面向对象的方法中的工作流图相似。序列图和其他方法中的交互图、消息追踪图、事件追踪图是类似的，用来显示按时间排列的相互作用。序列图只显示参与相互作用的对象及其之间的消息，而不显示对象间的静态关系。合作图是从 Booch'93 方法和 Fusion 方法中转变过来的，它通过把相关对象组织在一起来描述相互作用，它本身包含对象间的关系。在合作图中，事件的先后顺序或并行关系由序号来表示。

实现图是在实现阶段使用的图，是从 Booch 方法中的模块和过程图派生而来的。实现图以部件为中心，它包括两个图：部件图和配置图，部件图描述了实现系统中各个部件及其相互关系；配置图描述了实现系统中各个部件的物理分布。

UML 中的通用表示有：

• 字符串：表示有关模型的信息。

• 名字：在一定范围内表示模型元素。

• 标号：这里的标号同编程语言中的标号不同，它是缚着于图形符号的字符串。

• 特性字符串：表示缚着于某一模型元素的特性。

• 类型表达式：声明属性、变量和参数。它的含义和普通的编程语言是一致的。

• 定制（Stereotype）：实际上是 UML 的扩展机制。运用定制可以利用已有的模型元素来定义新类型的模型元素。

UML 的语义部分是对 UML 的准确解释，其内容由三部分组成：

1）元素（Common Element）：主要描述的是 UML 中的各种元素的语义。元素是 UML 中的基本构造单位，包括模型元素和视图元素，模型元素用来构造系统，而视图元素则用来构成系统的表示部分。

2）机制（Common Mechanism）：主要描述的是使 UML 保持简单和概念上的一致的各种机制的语义。这些机制包括定制、标记值、注记、约束、依赖关系和类型/实例、类型/类的对应关系。

3）类型（Common Type）：主要描述的是 UML 支持的各种类型的语义，这些类型包括布尔类型、表达式类型、列表类型、阶类型（Multiplicity）、名字类型、坐标类型、字符串类型、时间类型和用户定义类型（Uninterpreted）。

UML 中语义的各个部分不是相互独立的，彼此之间紧密相连，互有重叠，组合在一起构成了 UML 的完整语义。

结束语 虽然 UML 刚刚诞生，但是它具有强大的生命力。围绕着 UML 的工作将会陆续展开，这其中包括 UML 本身的完善，支持 UML 的开发过程的研究以及支持 UML 的 CASE 工具的开发。这些工作的展开必将有助于实现软件自动化。

（下转第 59 页）

积极研究具有良好用户界面的、容易学习和操作简单的形式化方法的支持工具。④鉴于形式规范与实际代码的差异，借鉴软件重用的基本思想，作者认为，我们需要研究一种介于规范语言和程序语言之间的语言，这种中间语言主要完成在程序部件库中，寻找相应的部件，并完成部件返送的工作。对此进行的研究尚刚刚开始。

主要参考文献

[1] Constance Heitmeyer et al. ,Formal methods for Real-Time Computing,John,Wiley & Sons Ltd

[2] Dan Craigen et al. ,Formal Methods for Trustworthy Computer Systems,Springer-Verlag

[3] P. T Ward and S. J. Mellor,Structured Development for Real-Time System,Yourdon Press,1985

[4] D. Harel,Statecharts:A Visual Formalism For Complex Systems,Science of Computer Programming,8 (1)1987

[5] F. Jahanian et al. ,A Specification Language For Real-time Systems,IEEE Trans. on Software,Eng,20 (10)1994

[6] Ingo Claben et al. ,Algebraic Specification Techniques and Tools for Software Development:ACT Approach,1993

[7] 全炳哲、余江、金淳兆，可重用构件及其描述语言，软件学报,5(1),1994

[8] A. Coen-Porisini et al. ,A Formal Method for ASTRAL Intralevel Proof Obligations,IEEE Tran. On Software Eng. ,20(8)

[9] L. Semmens et al. ,Integrated Structured Analysis and Formal Specification Techniques,The Computer Journal,35(6)1992

[10] Dick A. J. J. et al. ,Computer Aided Transformation of Z into Prolog, Proc. 4th Z Users Meeting, Springer-Verlag,1989

[11] Johnson M. and Sanders P. ,From Z Specification to Functional Implementations,Same to[10]

[12] 林凯、孙永强、陆朝俊，基于重写方法的程序开发系统的设计和实现，计算机学报,19(9)1996

[13] Mandayam Srivas and Albert Camilleri(Eds.),Formal Methods in Computer-Aided Design,FMCAD' 96,Nov. 1996,Proc. LNCS 1166

[14] Authony Hall,Seven Myths of Formal Methods, IEEE Software,Sep. 1990

[15] Jean-Raymond Abrial et al. ,(Eds.),Formal Methods for Industrial Applications,LNCS,Springer

---

(上接第 54 页)

参考文献

[1] G. Booch, Object-Oriented Development, IEEE Trans. on Soft. Engi. SE-1(2)1986

[2] G. Booch,Object-Oriented analysis and design. ,The C++ Report,3(8)1991

[3] G. Booch,Object-Oriented Analysis and Design with Applications,Benjamin/Cummings,1994

[4] G. Booch et al. ,The Unified Modeling Language, http://www.rational.com/uml

[5] Peter Coad and Edward Yourdon,Object-Oriented Analysis,Yourdon Press,1990

[6] Peter Coad and Edware Yourdon,Object-Oriented Design,Yourdon Press,1991

[7] I. Jacobson et al. ,Object-Oriented Software Engineering:A Use Case Driven Approach, Addison-Wesley,1992

[8] J. Rumbaugh et al. ,Object-Oriented Modeling and Design,Prentice Hall,1991

[9] S. Shlaer and S. J. Mellor,Object-Oriented Systems Analysis:Modeling the World in Data,Englewood Cliffs,NJ:Yourdon Press,1988

[10] S. Shlaer and S. J. Mellor,Object Lifecycles:Modeling the World in States,Englewood Cliffs,NJ:Yourdon Press,1992

[11] Daniel Tkach et al. ,Visual Modeling Technique:Object Technology Using Visual Programming,Addison-Wesley,1996

[12] R. Wirfs-Brock et al. ,Designing Object-Oriented Software,Prentice Hall,1990