计算机科学2002Vol. 29№. 1

建模语言及其研究进展初谈

Advances in Modeling Languages

戴桂兰

(清华大学计算机科学与技术系　北京100084)

Abstract Based on the common features of modeling languages, the paper attempts to present a definition, description content, application domains for modeling languages. In terms of their derivation and foundation, modeling languages can be divided into four kinds: formal modeling languages, graphic modeling languages, integration graphic and formalization, and meta-modeling languages, from which we outline their representative work, the state of the art, fundamental issues, and development directions for the further researches.

Keywords Software modeling, Modeling languages, Modeling methods, Meta-modeling technique

1 引言

建模语言的发展是随着计算机发展而发展的。在计算机发展的初期,程序设计被认为是一种技巧,那时所开发软件产品的部件不具有可重用性,难以维护及改进。这很不利于复杂的大型软件系统的开发。到了1980年代,DeMarco 提出了基于模型的软件开发方法,他认为软件系统的开发与大型复杂的工程系统的开发相类似,在系统开发之前首先建立系统的书面工作模型。利用这些模型,使得有关人员在系统成型之前对其有明确的认识。这样可以大大提高软件开发的效率,增强软件的正确性、可靠性以及可重用性。

目前,建模语言已成为软件工程的研究热点之一,同时已引起了许多学者的兴趣和一些公司的广泛重视,并研制了诸如 UML[21]、ROOM[5]等一批在理论、实用等方面有重要价值的建模语言。实际工作者将这些建模语言及其提供的支持工具集应用于实践中,取得了相当的成功[22]。在数十种建模语言出现后,为进一步掌握建模语言的内涵,以便于建模语言的进一步发展,本文试图从建模语言的基本共性出发,在阐明建模语言的定义、描述内容、适用范围及其基本特征的基础上,围绕着形式化建模语言、图形化建模语言和形式化建模语言的集成以及元建模语言四大类,总结其代表性工作、研究现状、存在的问题及进一步的研究方向。

2 建模语言的基本内容

软件建模是对目标系统的结构、行为、功能和性能等特征的抽象描述,以突出其主要因素(如系统的体系结构和行为等),忽略其次要因素(如具体实现细节等),所得的软件模型可以充当软件的最终用户、建模者与实现者之间交流和协商的桥梁,也是维护人员对软件系统进行维护的重要依据。软件模型是用建模语言描述的,建模语言是软件建模必不可少的重要工具。

2.1 建模语言定义

建模语言并没有标准的定义,但学术界基本接受 Rumbaugh 等人提出的定义[21]:

建模语言是用于规格说明、形象化和构造软件系统的语言,它能够提供适当的设施来描述问题空间和软件解空间的概念和结构。

2.2 建模语言适用阶段

建模语言主要应用于软件系统的需求分析和设计阶段,但用其描述的软件模型在软件生命期中从需求规格描述到系统完成后测试的不同阶段都有相当重要的作用,为软件设计者提供指导和支持。在需求分析阶段,利用建模语言捕捉系统的功能需求,分析、提取"客观世界"领域中的主要概念和机制以及描述它们的合作概貌。在设计阶段,建模的目的是通过考虑实现环境,将分析阶段的模型扩展和转换为可行的技术实现方案。在实现阶段,其主要任务是用高级程序语言实现设计模型中的主要抽象单元。在配置阶段,利用建模语言描述所构造的系统的软硬件配置情况。在测试和维护阶段,可利用前几个阶段所构造的模型来指导和协助测试和维护工作。

2.3 建模语言描述内容

建模语言能够提供适当的设施,使软件设计者能在适当的抽象层次上以适当的方式(图形化或文本化)描述各个抽象单元(如:类、过程、use-case 或程序单元等)以及它们之间的相互关系(如:继承/子类型、数据流或顺序图)。这些抽象单元以及它们之间的关系统称为软件系统的体系结构描述。由此可见,体系结构是建模语言所描述的主要内容之一。软件系统的行为也是建模语言所需要描述的一个重要内容,它主要包括如何完成每个抽象的主要功能以及成份间的相互作用何时发生等等。另由于建模语言用于对软件系统的完整描述,因此软件系统的功能和性能特征也是建模语言所应描述的重要内容。

3 建模语言与建模方法的关系

大部分建模方法是由一个建模语言和一种过程共同组成的。过程是指如何利用语言表示描述软件模型以及在软件建模过程中应采取哪些步骤所提出的建议。建模方法需要建模语言来支撑。建模语言是软件建模必不可少的重要工具,它影响着建模者的思维方式。虽然一个语言可以支撑多种开发过程,一个建模方法可以运用多种描述语言,但是,由于某个具体的建模方法需要特定的表示来体现,也即一个建模语言与某个具体的建模过程相结合,可以产生比较好的效果,因此建模语言与建模方法基本上是一一对应的。

戴桂兰　博士后,主要从事建模语言、建模方法、软件工程和编译技术等方面的研究工作。

另由于一些建模方法(如:OMT 方法)未定义完备的语言表示的语法模型,为与主流刊物上的称谓保持一致,因此在下文中,对于一些无完备语法模型的建模方法,我们用建模方法(如:OMT 方法)表示相应的建模语言(如:OMT 语言)。

4 建模语言与其它语言的区别

同样作为软件系统的描述工具,建模语言与传统程序设计语言、体系结构描述语言、需求语言有许多相同之处,但由于它们的设计目标不同,因此也存在一定的差别。这里我们主要从语言的抽象层次、关系描述、适用开发阶段、抽象的作用与规格说明的角度着重比较建模语言与传统程序设计语言的主要不同之处。比较所得的结果如表1所示。

另外,建模语言与体系结构描述语言(ADL)的主要区别在于后者集中在组件及其组件间进行交互的规则描述上,很少考虑某个具体组件的内部行为,而建模语言关注的是软件系统的整体结构和行为而不是某一部分特征;建模语言与需求语言的主要区别在于后者用于描述问题空间,而前者既用于描述问题空间,也用于描述解空间。

表 1 建模语言与传统程序设计语言的主要区别

	建模语言	传统程序设计语言
抽象层次	对现实世界各抽象层次信息的描述	主要对数据结构和算法的描述
关系描述	除具备传统程序设计语言的运算符外,还提供更丰富、更灵活的成份间关系描述设施	包括递归,条件结构和特定数据类型的运算符+,-,*,/
抽象的作用	用于处理系统复杂性问题,包括递归、增量建模、可重用等	主要利用抽象规则实现宏和过程定义
适用阶段	软件生命期的各个阶段(主要为系统分析与设计阶段)	实现阶段
规格说明	强调功能、性能、容错等多方面的规格说明	侧重于语义规格说明

5 建模语言的基本特征

为提高建模语言的适用性,建模语言应能描述软件系统的各种必要特性,如:动态的内部结构、实时反应性、并行性、分布性等,也就是说,建模语言应具有足够的表达能力。同时,为提高软件模型的质量,建模语言一般应具备以下几个基本特征:

·简单性,用户只有对建模语言充分理解之后,才能用其描述比较复杂的问题,因此,建模语言应尽可能地简单化;

·正交性,为便于使用,建模语言中描述某一类成份的设施应具有唯一性,也即建模语言中尽量避免冗余现象,这样可使语言的概念集合保持较小,且不需要解释何时用这个设施而不用另一个与之相类似的设施;

·一致性,这是相对于语言的设计目标而言的。建模语言所提供的设施都应加强对其设计目标的支持,对不支持其设计目标的设施应被抛弃;

·平滑性,为便于软件系统问题域抽象到软件解空间的直接映射、避免软件开发过程中的"缝隙",建模语言应具有平滑性,以便于产生可维护的软件;

·可逆向性,为使在一个阶段产生的软件模型的改变能自动地对另一个阶段的模型产生影响,且使软件的实现代码与其模型保持同步,建模语言应具有可逆向性;

·可支持性,为便于开发大型的软件系统,工具支持对模型的书写和维护是很重要的。这就要求建模语言的语法比较容易书写且易于显示在计算机屏幕上,语义能自动或半自动地产生实现代码;

·可靠性,为使目标软件能满足其规格说明、且当遇到未预料到或错误的输入时产生适当的反应,这要求建模语言能提供提高系统可靠性的设施。

当然,这些特征并不是所有的建模语言都能具备,但它们是我们评价建模语言的主要标准,也是进一步发展建模语言的努力方向。

6 研究现状

建模语言的表示形式有多种,它可以是自然语言、图表,也可以是数学公式。根据其起源及支撑基础的不同,建模语言可分为以下四大类:

6.1 形式化建模语言

形式化建模语言是在 Dijkstra 和 Hoare 的程序验证和 Scott、Stratchey 以及其他学者在程序语义方面的工作基础上发展起来的,已有近30年的研究与应用历史。形式化建模语言的表示是具有已知特性的数学抽象,如域、元组、集合、序列、包和映射等,因而具有完备的数学基础。形式化建模语言的主要优点在于形式简洁,说明问题明确,它们支持关于功能说明的形式推理并且提供了对于软件产品结果确认的基础。有代表性的工作主要有 VDM[23]、Z 以及 F-Logic[12]、DS-Logic[13]等。

至今为止,由于还没有对形式化建模语言提供完全自动化的、用户接口良好的、易学和易对功能进行扩展的完善的支持工具,且没有一种形式化建模语言能对软件生命周期的各个阶段提供全面的支持,因此仍无法被大多数软件开发者所接受。

6.2 图形化建模语言

由于图形化建模语言主要基于比较接近设计者的直觉,并提供了丰富的结构化机制,比较易于理解、运用以及与用户之间的交互,因此发展很快,且已得到广泛应用。又由于面向对象技术是解决软件系统复杂性最有效的技术之一,因此最常用的图形化建模语言是图形化面向对象建模语言。

图形化面向对象建模语言及其支撑方法出现于1970年代中期与1980年代后期之间,到目前为止已有60多种[21],有代表性的工作主要有 CRC 方法、OMT 方法[6]、OOSE、Booch 方法[4],ROOM[5]等。其中,ROOM 具有完备的语法模型,且用其描述的各级软件模型都是可编译执行的。

近年来,一些学术刊物上登载有各种图形化面向对象建模语言(方法)的表示体系,同时有的已应用于开发实际的软件系统。可以看出,它们都不外乎经过分析抽象出对象,建立

对象之间的关系，但由于其应用背景和侧重点不同，因此它们之间还存在一些差异，尤其是在设计思想上：有些建模语言（方法）是基于信息建模的扩充（如 OMT、Coad 和 Yourdon[3]），而有些则是基于行为和职责的建模（如 CRC 和 OOSE）。经过多年的实践，比较有影响力的有 OMT、Booch 和 OOSE。其中，OMT 表示描述分析模型能力较强，而描述设计模型的能力较弱；Booch'91则在设计领域较强，分析领域较弱；OOSE 有很强的行为表达能力，比较适用于实时系统，但其它方面的表达能力不强。Booch 表示主要适用于小粒度对象，难以用于高级结构建模，而 OOSE 却走向另一个极端，仅适用于高级结构的描述，而不便于系统的实现。

到1990年代中期，为能更好地支持 OO 软件开发，各种建模语言及其支撑方法纷纷引进别的方法中好的设计思想和技术来弥补自己的不足，例如：Booch 方法吸取了 J. Rumbaugh 和 I. Jacobson 等人提出的分析技术，产生了 Booch'93。OMT 方法利用了 Booch 方法中许多好的技术，形成了 OMT-2。这些方法虽不断交叠发展，但它们仍然沿用各自的表示体系，使软件开发市场的表示法日益繁多，从而给使用带来了一定的困难。

6.3 形式化建模语言与图形化建模语言的集成

图形化建模语言的表示缺少精确的语义，易于产生二义和不一致，而形式化建模语言虽有精确的语法和语义定义，但可能会使建模者把主要精力放在掌握语言表示本身而不是系统的模型上，从而导致创建的系统模型并不能恰当地反映系统。由于面向对象技术与形式化技术具有较高的相容性，这主要表现在：

(1)面向对象技术使用抽象的数据类型，而形式化技术能准确地描述这些的抽象；

(2)面向对象技术提供了结构化的规则，将形式化技术应用于大型系统的机制和开发中；

(3)对复杂的面向对象的机制，如对象的聚集或继承，形式化技术能给出比较准确的定义。

鉴于此，有人提出将直观的图形表示和精确的数学表示结合起来，在软件开发实践中引入精确的规格说明，同时保证它的可接收性和可使用性，这样所得的语言比纯粹的图形表示和形式化表示都要好。将面向对象技术和形式化技术结合起来已成为软件工程研究较热的一个领域，同时也吸引了世界许多学者及公司的兴趣。目前，在这个领域内有四个主要的研究分支：

(1)*利用形式化建模语言丰富图形化建模语言的概念*

借助形式化建模语言来提高图形化语言的描述能力，而在开发过程中主要采用图形化表示，这方面的代表性工作主要有 Syntropy[19]，Lano[20]和 Weber[16]等人利用 Z 表示来丰富相应的图形化表示的概念。

(2)*利用图形化建模语言中的概念对形式化建模语言进行扩展* 这种方法主要体现在利用面向对象范型中的比较接近应用域的概念，对一些形式化建模语言进行面向对象扩充，以使之更易于为软件开发者所理解和管理。主要代表性工作有 OOZ，Z^{++}，VDM^{++}，Object-Z[7]等。

(3)*利用图形化建模语言给形式化建模语言提供接口*

对于一个给定的形式化建模语言，开发一种与之表达能力相等价的图形表示，以便于模型的创建和可视化，这方面的工作主要有 OASIS[6]和 LTL[9]等。

(4)*利用形式化建模语言定义图形化建模语言的语义*

利用形式化建模语言来定义图形化建模语言的语义的主要工作就是建立一些规则，为图形化语言的语法结构与形式化定义的语义域建立对应关系，以提高图形化表示的精确度。这是形式化建模语言与图形化建模语言集成的最为广泛的方法，如：Hartrum 和 Semnens 等人利用 Z 和 VDM-Like 表示分别给出了 OMT、Yourdon、Fusion 等方法部分表示的形式语义[2,24]。

形式化建模语言与图形化建模语言的结合，能克服各自的缺点，起到取长补短的作用。虽然也存在一定的问题，但它们所具有的优点是显而易见的，必将对提高软件生产率和软件质量，进而对迈向软件生产自动化产生巨大的推动作用。

6.4 元建模语言

近年来，元建模已成为建立软件建模语言的一个重要的技术。其基本思想在于：将几个元模型集成为一个简单的"核"元模型，以建立一系列建模概念的"核"描述。

利用元建模技术建立新的建模语言有多种[15,11,14]，其中，最突出的代表为 UML。UML 是通过融合 OMT、OOSE 和 Booch 方法而产生的一个建模语言。经过不断的发展，UML1.1已成为面向对象行业公认的标准语言。UML 采用了四级元模型体系结构：元-元模型、元模型、模型、用户对象，以加强其形式化程度。元模型为 UML 的所有元素在语法和语义上提供了简单、一致、通用的定义性说明，使系统开发在语义上取得一致，消除了有二义性的表达方法所造成的影响。通过支持对元模型的扩展定义，使得 UML 可应用于各种应用域，成为一个通用的建模语言，同时对软件生命周期的各个阶段提供了全面的支持。

相应地，由33个研究者和方法学家组成的 OPEN 合作组，以 SOMA、MOSE 和 Firesmith 为基础，创建了一个完全的、以过程为中心的面向对象方法——OPEN 方法[1]，其支撑语言 OML 与 UML 相类似，利用元建模技术，创建了一个形式化语言结构。

7 存在问题及进一步研究方向

综上所述，建模语言的发展经历了在数量上快速增长时期，同时经历了下述过程：(1)从非形式化建模语言和图形化建模语言各自发展到形式化建模语言与图形化建模语言的集成，再到元建模语言；(2)从单一表示模式向多视点发展；从多元方法并存到向统一标准发展；(3)从自然语言到有严格语法而无精确定义的语义的语言，再到有严格语法和语义定义的语言发展。今后的研究工作的重点应放在以下几个方面：

在完备性方面，由于建模语言用于对软件系统模型的完整描述，因此它必须能够简洁、完整地描述系统的各种必要特性。而现有的建模语言在表达能力上或多或少存在着不足：虽然一些形式化建模语言作了面向对象扩充，如 Z^{++}、VDM^{++}，但它们仍不具备描述大型软件系统结构的能力；图形化面向对象建模语言虽然在描述大型复杂的软件系统的框架结构方面有较强的优势，但在描述某些细节方面存在着一定的局限性，且在对系统的并发性和不确定性的描述上也存在着不足。如何结合现有建模语言的长处，在强调语言表达能力的同时，还应考虑语言本身是否精确、不存在歧义，各成份间不存在不一致性，这是我们今后在研究建模语言的过程中应考虑的问题。

在针对性方面，由于软件模型是模型定义者、软件开发人员以及软件最终用户之间的一种交互和协商的桥梁，因而主

要面向软件最终用户和开发人员。而软件模型的正确与否是相对于软件需求而言的。问题在于，软件模型的作用是作为一个交互工具，还是既作为交互工具，同时又作为系统实现的输入。实际上，目前多数建模语言并未配以相应的转换工具，或者语言所描述的模型本身难以自动、半自动地转换为实现，系统的实现还需用手工编写。这样，软件模型只起到文档的作用，这是相当不够的。在研究建模语言时，一方面使语言具有较好的可读性和易理解性，同时尽量使语言结构便于变换为实现代码，且对相应的转换工具的研究尚需进一步加强。

在应用域方面，一个语言只有具有较强的适用性和通用性，才会有生命力。而现有的多数建模语言只适用于某一特定应用领域，如，ROOM 主要用于分布式实时系统，OCTOPUS 侧重于嵌入式系统。虽然已有通用的建模语言，但它们要么是抽取各种应用领域的所有概念作为自己的概念集合而包含一些冗余；要么是语言概念过于抽象而难以理解和使用。如何在寻求完备性的同时，使语言的基本结构尽可能简单，且具有较强的可扩充性，以适用于各种应用领域，也是建模语言进一步的发展趋势。

另外，软件模型一方面作为交互工具，它应具有较强的可读性，同时作为系统实现的输入，又应具有便于机器支持和自动处理等特点。而现有的形式化建模语言虽然精确程度高、易于验证，但可读性不强，使用时需要较高的数学素养；图形化面向对象建模语言虽然在可读性和效率方面弥补了前者的不足，但自身的精确化程度不高，不便于对软件模型进行自动验证。因此，对建模语言的功能与性能方面的研究仍需进一步加强。

致谢：本文得到了东南大学徐宝文教授、清华大学张素琴教授和蒋维杜教授的指教，在此表示真挚的谢意。

参 考 文 献

1 Henderson-sellers B, Firesmith D G. Comparing OPEN and UML: the Two Third-Generation OO Development Approaches. Information and Software Technology, 1999, 41: 139～156

2 Robert H B, Betty H C C. A Formal Semantics for Object Model Diagrams. IEEE Transaction on Software Engineering, 1995, 21 (10)

3 Edward Y, Carl A. Case Studies in Object Oriented Analysis & Design. Prentice Hall, 1996

4 Booch G. Object-Oriented Design with Application. Menlo Park, CA: Behjamin/commings, 1991

5 Selic B. Real-Time Object-Oriented Modeling. Katherine Schowdter, US, 1994

6 Maher A, Juha K, Jurgen Z. Object-oriented Technology for Real-Time Systems——A Practical Approach Using OMT and Fusion. Prentice-Hall PTR, 1996

7 Claus L, Thomas L. Formal Development of Reactive Systems. Spring-verlag, BerLin Heidelberg, 1995

8 Pastor O, Ramos I. Oasis 2.2: A Class-Definition Language to Model Information System Using an Object-Oriented Approach. SPUPV-95.788, Universitat P. de Valencia, 1996

9 Reggio G, Larosa M. A Graphic Notation for Formal Specification of Dynamic Systems. In: Proc. of FME97, Lecture Notes in Computer Science 1313, Spring-Verlag, 1997

10 Evans A, France R, Lano K, Rumpe B. Developing the UML as a Formal Modeling Notation. Muller and Bezivin editors, Lecture Notes in Computer Science 1618, Springer-Verlag, 1998

11 Sjaak B, Motoshi S, Farnk H. Meta-Modeling Based Assembly Techniques for Situational Method Engineering. Information System, 1999, 24(3): 209～228

12 Kifer M, Lausen G. F-Logic: A High Order Language for Reasoning About Objects, Inheritance and Scheme. Proc. of the ACM SIGMOD Symposium on Principles of Database Systems, SIGMOD RECORD, 1990, 18(6)

13 Wieringa R, Broersen J. Minimal Transition System Semantics for Lightweight Class and Behavior Diagrams. In PSMT Workshop on Precise Semantics for Software Modeling Techniques, Technische Universitat Munchen, Report TUM-I9803, 1998

14 Pons C, Baum G. Formal Engineering Methods 2000, ICFEM2000. In: Third IEEE Intl. Conf. 2000. 101～110

15 Richard F P. A Meta-method for Formal Method Integration. In: Proc. Formal Methods Europe'97, Lecture Notes in Computer Science 1313, Springer-Verlag, Sep. 1997

16 Weber M. Combining Statecharts and Z for the Design of Safety-Critical Control Systems. In: Proc. of Third international Symposium of FME96, Oxford, 1996

17 Kim S, Carrington D. Formalizing the UML Class Diagrams Using Object-Z. In: Proc. UML'99 Conf. Lecture Notes in Computer Science 1723, 1999

18 Evans A, et al. Towards a Core Metamodeling Semantics of UML, Behavioral Specifications of Businesses and Systems. H. Kilov editor. Kluwer Academic Publishers, 1999

19 Cook S, Daniels J. Let's Get Formal, Journal of Object-Oriented Programming (JOOP), July-Aug. 1994

20 Goldsack S, Kent S. Formal Methods and Object Technology, Chapter 3: LOTOS in the Object-Oriented Analysis Process. Editors S. J. Goldsack, S. J. H. Kent. Serie FACIT. Springer-Verlag, 1996

21 Rumbaugh J, Jacobson I, Booch G. The Unified Modeling Language Reference Manual. Addison-Wesley, New York, 1999

22 Femandes J M, Machado R J, Santos H D. Modeling Industrial Embedded Systems with UML, Hardware/Software Codesign, 2000. CODES 2000. In: Proc. of the Eighth International Workshop on, 2000. 18～22

23 Dawes J. The VDM-SL Reference Guide, Pitman, 1991

24 France R B, Bruel J M, Larrond-Petrie M M. An Integrated Object-Oriented and Formal Modeling Environment. Journal of Object-Oriented Programming, Nov./Dec. 1997. 25～34