计算机科学2002Vol.29№.10

B语言和方法与Z、VDM的比较

Comparison of Z and VDM with B

邹盛荣　郑国梁

（南京大学计算机科学系　南京210093）

Abstract　This paper introduces briefly three influential formal specification languages and methods—VDM、Z and B. We analyze and compare their advantages and weaknesses,and provide a comparison of differences:underlying logic、syntax、specification style,etc.

Keywords　B,Z,VDM,Formal method,Specification language

1　引言

形式化方法是建立在严格数学基础上的软件开发方法。软件开发的全过程中，从需求分析、规格说明、设计、编程、系统集成、测试、文档生成直至维护各阶段，凡是采用严格的数学语言、具有精确的数学语义的方法都称为形式化方法。

形式化方法的一个重要研究内容是形式规格说明，即用具有精确语义的形式化语言书写的程序功能描述，它是论证程序是否正确的依据。形式化方法需要形式规格说明语言的支持，也可以说形式化方法的关键在于形式规格说明语言。形式规格说明语言提供了一个称为语法域的记号系统和一个称为语义域的目标集合，以及一组精确地定义哪些目标系统满足哪个规格说明的规则。根据对目标软件系统进行说明的方式分三种规格说明语言：

（1）*面向模型的方法*　其基本思想是，利用一些已知特性的数学抽象来为目标软件系统的状态特征和行为特征构造模型。如VDM、Z、B等都是面向模型的方法。

（2）*代数方法*　它为目标软件系统的规格说明提供了一些特殊的机制，以允许对目标软件系统描述的结构化，并支持目标软件系统中公共元素的重用。常见的代数方法有CLEAR,OBJ,OBJ-2等。

（3）*描述并发的进程代数*　提供了描述抽象概念并进行进程间联接和推理的方法。如CSP,CCS等。

下面介绍基于模型的三种形式化语言和方法的基本原理和内容，然后再对其主要优缺点进行分析，最后对这三种语言和方法从七个方面加以分析比较，并列表总结。

2　三种形式化语言和方法

2.1　VDM

VDM是在1969年为开发PL/1语言时，由IBM公司维也纳实验室的研究小组提出的，VDM是一种功能构造性规格说明技术，它通过一阶谓词逻辑和已建立的抽象数据类型来描述每个运算或函数的功能，这种方法在90年代初在欧美许多研究机构或大学得到了广泛的应用。

VDM技术的基本思想是运用抽象数据类型、数学概念和符号来规定运算或函数的功能，而且这种规定的过程是结构化的，其目的是要在系统实现之前简短而明确地指出软件系统要完成的功能，由于这种形式化规格说明中采用了数学符号和抽象数据类型，从而可使软件系统的功能描述在抽象级上进行，完全摆脱了实现细节，这样为软件实现者提供了很大的灵活性，此外，这种形式化规格说明还为程序正确性证明提供了依据。应用VDM技术进行系统开发包含了形式化规格说明、程序实现和程序正确性证明三个部分。

使用VDM规定形式化规格说明具有以下三个明显的优点：(1)只告诉计算机做什么；(2)提供了程序正确性证明的依据；(3)使规格说明描述简练、精确。

此外，使用VDM还可以培训程序设计者牢固树立先抽象、后具体的不断证明其正确性的逐步分解的自顶向下的开发思想，从而在整个程序开发的全过程中用系统而严密的方法保证所开发程序的正确性。但是，VDM也存在一些不足之处：

(1)由于VDM对抽象数据类型预先定义了运算，而某些用户定义的类型在规格说明描述中无需这么多运算，因而产生了运算冗余。

(2)VDM目前还未能建立一整套描述机制，将一个大型系统分解为许多运算而描述出这些运算之间的关系。

(3)由于采用数学符号和抽象数据类型，VDM形式规格说明过于形式化往往不容易理解，这有可能造成未读懂形式规格说明而错误地实现其软件的情况。

2.2　Z

Z以经典集合论和一阶谓词逻辑为基础，提供了一种称为模式的结构，以此来描述一个规格说明的状态空间和操作。Z规格说明由一系列模式组成，每个模式定义一个抽象对象或操作，并用谓词判定描述给出新的对象或操作的语义约束。Z模式说明可以组合成新的Z模式，新的Z模式继承其成分模式的一切属性和约束。这样，软件系统的Z模式规格说明可以按一定的层次结构给出。模式的使用为规格说明提供了一种演算，通过这种演算，无论多么大型系统的规格说明都可以通过一个个小的部分来构成。

基于一阶谓词和集合论的形式规格说明语言Z，利用模式和模式演算对目标软件系统的结构和行为特征进行抽象描述，其中状态模式对目标软件系统的结构特征进行抽象描述，操作模式对目标软件系统的行为特征进行抽象描述。这是一种具有特色的有效的形式化方法，但是Z语言还存在如下缺点：

(1)Z语言对大型系统的模块化能力不足。因为在Z语言中，目标软件系统的结构和特征都用模式来描述，随着系统的增大，模式也会越来越多，而Z语言中没有更加有效的机制来管理这些模式，最终导致Z规格说明难以阅读。

(2)难以识别影响某一状态模式的所有操作模式。因为在

Z语言中，一个操作模式可能涉及多个状态模式，为了确定能影响特定状态模式的所有操作模式，就需要逐个检查全部操作模式的声明部分，这对于大型软件系统的规格说明来说是不实际的。

(3)不能支持规格说明的重用。Z语言中没有提供重用机制。

(4)Z语言难以由计算机直接处理。因为在设计Z语言时，只是考虑到把Z语言作为一种严格的描述手段并没有考虑到将来由计算机辅助进行Z语言应用，所以许多Z语言符号在计算机中没有对应的键，难以进行规格说明的输入和自动化处理。

为了克服Z语言这些缺陷，80年代末到90年代初相继提出了一些Z语言的扩展方案，也进行了一些标准化工作，国际上在90年代初提出形式化方法和面向对象方法相结合的思想，虽然有所发展，但因缺少商品化的工具支持等诸多原因，而不能大量地实际应用，因而还需进一步完善。

2.3 B

(1)概要 B是计算机辅助软件工程中B技术、方法和工具集的简称，B(包含B方法，B工具，B工具盒)是一种坚实的面向实际软件过程的基于数学理论的技术；B方法所用的语言和方法支持大部分的软件过程：需求分析、规格说明、软件设计、实现和维护，分层软件的逐步构造伴随着逐步的验证和确认是B方法的指导性原则；B工具盒包含有大量的工具，所有的工具集成在一个基于窗口的软件开发环境中，能交互或自动地运行，因而支持运用B方法开发软件的整个软件过程，B工具支持软件的逐步构造，确认过程中静态分析可用类型检测，动态分析采用模拟技术，正确性证明则使用集成的定理证明器。

(2)B方法 B方法用一种简单的伪程序语言来描述需求模型、规格说明，并进行中间设计和实现，这个语言就是B语言，B语言支持规格说明的类型检测、动态验证、数学证明等来确保设计过程的正确。分步开发可以减少大型软件开发的复杂性，分层次的方法可以将高层实现表示成低层的规范，一个完整的开发就是逐步实施的规范/实现的过程，规范/实现过程在最低级的实现可以从预先实现的可重用的部件库中得到，高级的复用也可通过不断扩展新的部件库从而支持整个开发过程，每一层的实现过程是将规范翻译成可维护的独立编译的源代码和可执行指令的过程，B语言中结构化的机制象其它面向对象方法一样，增强了信息隐藏和数据封装，严密的部件接口控制确保了大型开发中各个部件的独立开发。

(3)B软件过程 B严密管理控制整个软件过程，在软件开发早期规格说明阶段，开发人员将对照用户的需求来验证测试系统需求模型的一致性，在每个开发阶段，每个设计描述都将与规格说明的需求文档对照其一致性，最低一级的设计部件由预先定义的可靠的部件库组装得到，并翻译成可执行代码，可靠的预定义的部件通过测试或重用生成(许多部件已经为B工具集所应用)，新的低层部件也可以进一步添加，这些添加的部件可通过已存在的可靠的部件开发，并测试得到它们的属性。

(4)B方法的特点

① 简单熟悉的符号表示法。这种符号表示法用广义替换表示状态转换，从规格说明到编码，这种统一的形式减少了学习的难度和转换中的语法错误，设计这种"数学程序设计语言"使人们使用一种非常具体的规格说明形式，而且对软件工程师来说是极为有益的。

② 模块化构造。从规格说明到实现的模块化构造允许将规格说明和验证过程分解为多个子任务进行，这个优点相比于其它类似的规格说明语言是非常突出的，而且比ADA和C++中的结构化构造更容易学习。

③ 大量实用的工具支持。现有大量的实用工具支持了B方法软件开发周期的所有阶段，包括动画和文档生成，其它的形式化方法中还没有类似的集成工具。

④ 成功的工业应用。B语言和方法已在很多的工业领域得到成功应用，包括实时、仿真、信息处理和工程等。

3 B和Z、VDM的比较

相对于LARCH、OBJ等代数方法而言，B、Z、VDM都是基于模型方法的形式化规格说明方法。

VDM是B、Z、VDM中最早的一种语言，其实VDM不仅是一种语言，还是一种开发方法，VDM的数学符号并不是建立在公理集合论的基础上，而是建立在谓词逻辑和集合论的基础上，在证明过程中还使用了三值逻辑(真、假、未定义的)，但VDM没能提供组装/分解规格说明和精化的机制。

Z是描述规格说明的主要表示方法，Z建立在集合理论的基础上，但没有清楚的指定公理描述，尽管Z提供了一种称为模式的结构，但还不能算是一种开发方法，模式的层次结构形成了系统的规格说明，按照创造者的意图，模式之间的联接还必须用一些非正式的文本加以注释。

VDM和Z一样，用前置/后置条件定义了规格说明(或精化)的动态转换，这两种情况下，都必须指明那些没有改变的部分。

B是三种语言中最新的一种，是一种覆盖了从规格说明到编码的所有阶段的开发方法，B建立在Zermelo-Frankel集合论的基础上，B包含一种"抽象机器"的结构化机制(从规格说明到精化再到实现)，这种开发方法建立在数学理论基础上，还包括了广义替换、精化、软件的层次体系结构理论。系统的动态定义不是用前置/后置条件而是用建立在谓词逻辑基础上的转换来实现的。

以下，从基础逻辑、语法等七个方面作进一步的比较：

(1)基础逻辑 B和Z建立在相同的经典集合论上，VDM则有它自己的3值逻辑，这种逻辑允许有未定义的成分存在。B的语义是建立在Abrial广义替换语言和用谓词变换和扩展的Dijkstra最弱前置条件演算基础上。

(2)语法 在B中没有应用Z的图形模式的符号，B同VDM一样是建立在关键字的基础上，并和VDM一样，有两种符号表示方法，一是用ASCII码，另一种是用数学符号。

(3)书写风格 B建立在规格说明的谓词转换器的风格上，这表示在VDM中使用前置/后置条件的地方，在B中将用替换来表明状态的改变，如：

```
VDM:
      insert(elem:TYPE)=
      wr var:set(TYPE)
      pre~(elem :var)
      post var = var' \/ {elem}
B:
    insert(elem)=
    PRE
        elem:TYPE &
        elem /:var
    THEN
        var := var \/ {elem}
    END
```

B中的后置条件象一个赋值，因而使得规格说明看起来

象伪程序，其实它的语义恰如状态的替换，Abrial 已经证明了这种赋值和 VDM 中前置/后置条件功能是相等的，如：

```
VDM:
      sort ::
        wr list:seq(TYPE)
      pre true
      post is_sorted(list)&
           is_permutation(list,list')
B:
    sort ::
    ANY newlist WHERE
          is_sorted(newlist)&
          newlist :perm(rng(list))
    THEN
           list := newlist
    END
```

(4)*前置条件、不变式和证明法则*　Z、VDM 和 B 在前置条件和不变式的使用上都有不同的方法，在 Z 中，前置条件没有明确的陈述，它可从 delta 模式的定义中计算得到。

在 VDM 和 B 中，前置条件是明确陈述的，冗余的前置条件需要一致性检测，即这些前置条件是这个操作所必需的吗？

B 和 VDM 的不同之处在于它们处理不变式的方法上，在 VDM 中不变式被假定为任何前置或后置条件的一种暗示，因此关于该操作唯一要证明的就是其可行性。

在 B 中，不变式是冗余的，不变式的存在与否都不会改变操作的定义，它并不是操作后置条件的一部分，因而需要证明该操作是否必须保留不变式，而在 VDM 中则应该明确无误地保留。

(5)*语言应用范围*　Z 是一种相当严格的规格说明语言。

VDM 和 B 是"广谱"的，在两类语言中，包含了必要的程序构造。

B 包含了一个很小的语言子集，分6个构造部分，所有的数据类型被封装在抽象机器中，这就保证了精化到代码的过程都在一个统一的语义框架上进行。

(6)*结构化规格说明和实现*　B 比 VDM 有着更强大的结构化构造，而且也很不同于 Z 的模式演算，B 是以对象为基础，包含有各个种类的信息被增强了，因此操作的状态封装很易实现，然而还缺少真正意义上的继承和多态性等相关概念，所以它还不是太"面向对象"的。

B 方法的分层开发的观念是非常强有力的，从而允许一个复杂的开发被分解为一系列小的构造，以此满足工业化应用系统的精化需求。

(7)*工具支持*　B 因为有工具支持而获得相当大的优势，象 B－Toolkit 这种高质量的商业软件有一套完整的开发过程的支持用来完成如下一些任务：类型检测、动画、证明法则生成、交互或自动的证明支持、代码翻译、文档以及带有配置管理功能的综合开发环境支持。

结束语　上文中，我们简要介绍 VDM、Z、B 三种语言和方法及其优缺点，并从七个主要方面作了分析比较，作为总结，表1从基础、开发阶段等五个方面对上述三种规格说明方法进行比较。

表1　B 与 VDM、Z 的比较

属性	VDM	Z	B
基础	部分函数、集合论	谓词演算、集合论、模式	最弱前置条件、集合论
开发阶段	规格说明、设计	规格说明	规格说明、设计、实现
风范	前置/后置条件、函数	模式符号表示、关系	严格的程序设计语言
工具支持	在规格说明级	在规格说明级	所有开发阶段
培训支持	图书、课程	图书、课程	实例研究、课程

从上表可以看出，B 相对于 Z 和 VDM 来说，有很大的优越性，再加上商业工具的大力支持，因而在英、美、法等国得到了较广泛的应用，然而 B 语言还存在一些不足之处，下一步我们还将就 B 语言的扩充做进一步深入的研究，希望 B 语言和方法在国内也能得到广泛、深入的应用。

参 考 文 献

1　袁晓东，郑国梁. Z 的面向对象扩充 COOZ 的设计. 软件学报，1997(9)：694～700

2　袁晓东，许皓，胡德强，李勇，郑国梁. 现有 Z 面向对象扩充语言的比较. 计算机科学，1997(3)：58～61

3　Lano K. The B Language and Method. Springer-Verlag London Limited. 1996

(上接第127页)

CMM 来评估并且改进它们的软件生产过程，也有很多研究 CMM 实践的文章发表，总结 CMM 在实践中的经验与教训[5,6]。文[5]的研究表明，有效的过程评估(以及相应的提高)的模型/方法必须要追求组织所有方面的一致性的改进而不仅仅是工程方面的改进。要实现组织范围内的软件过程改进，必须要综合考虑其中的各种因素。

必须强调的一点是：CMM 描述的是过程应当解决的问题而非应当如何去执行，因此在过程改进中需要确定很多"执行细节"[7]。由于 CMM 重点在于软件问题，其他一些属于整入进大纲范畴的问题，在 CMM 中仅仅是提以而已。这些问题包括战略业务规划、建立产品线、采用有效技术以及人力资源管理等。

参 考 文 献

1　Schuppan V，Rußwurm W. A CMM-Based Evaluation of the V-Model 97，Software Process Technology，7th European Workshop，EWSPT2000，Springer

2　V-Model--Brief Description. http://WWW. v-modell. iabg. de

3　Paulk M C. Toward Quantitative Process Management With Exploratory Data Analysis，1999 International Conference on Solftware Quality http://WWW. sei. cmu. edu

4　杨一平，等. 软件能力成熟度模型 CMM 方法及其应用. 人民邮电出版社，2001

5　Cattaneo F，Fuggetta A，Sciuto D. Pursuing Coherence in Software Process Assessment and Improvement，Software Process Improvement Practice 2001，John Wiley & Sons，Ltd.

6　Blanco M，et al. SPI Patterns：Learning from Experience，European Software Institute，IEEE Software，May/June 2001

7　SEI：The Capability Maturity Model：Guidlines for Improving the Software Process