# 形 式 化 方 法:

## 基于 B 方法的严格软件开发

## (15)

## B 方法、形式化方法和课程总结

裘宗燕

北京大学数学学院信息科学系

2010年春季

## 总结

形式化方法希望把软件开发过程中的分析、设计和实现活动纳入更加科学、严格轨道。为此需要

- 研究软件的各种组成部分的理论基础，包括
  - 数据的理论和模型
  - （计算）过程的理论和模型
  - 过程描述（程序）的理论，等等
- 总结人们在软件开发中积累的经验，抽象出规律性的东西，建立软件开发的理论基础
- 通过实践和总结，为严格的软件开发过程构建工具和环境

可以认为 B 方法是这方面的一个尝试，在其前面的工作有 VDM，Z 等等

正在进行的一个研究开发项目是 event-B，由 Jean-Raymond Abrial 提出，基本描述语言类似于 B，以集合论、广义代换、精化、证明义务等作为基础。但其主要目标是支持系统的分析和建模（与当前软件实践的发展一致）

# 总结

B 方法的主要想法：

- 以一阶逻辑和集合论作为描述软件规范的基础
- 以抽象机作为软件部件的描述单元
- 基于集合论的概念描述抽象机的状态空间
- 用不变式描述合法状态
- 用广义代换描述系统操作
- 用逻辑公式描述参数的约束，集合和常量的属性
- 基于各种约束、属性、不变式生成规范的证明义务。如果一个抽象机的所有证明义务都得到确认，这个抽象机就是协调一致的

抽象机的基本用途是建模。在 Atelier B 里创建项目，可以创建软件开发项目，也可以创建系统建模项目

差别在于系统建模项目只包含抽象机组件和精化组件，不包含实现组件



# 总结

如今的软件开发也特别强调建模，例如已经被软件工程界广泛接受和使用的 UML 就是 “统一建模语言”（Unified Modeling Language），人们强调在开发复杂的软件系统之前，先建立软件的模型

类似 UML 的建模语言是软件设计语言，来自实践并用于实践，把软件系统的严格语言描述（而不是自然语言描述）阶段向前推，是重要进步

但 UML 没有严格定义的语义基础，虽然可以做一些浅层的自动化的一致性分析，深入的语义方面的分析和检查还要靠开发者和检查者的理解

建模总是在某个抽象层次上做的，而不是在代码层次上做的。无论是否采用形式化方法。建立的系统模型都不是可执行的

如果我们能接受非形式化的或者半形式化的（如基于 UML）建模，那么也应该可以考虑基于形式化方法和语言（例如 B 方法和 B 语言）建模

其他形式化技术和描述形式都可以考虑，例如 VDM、Z、Event-B 等等

# 总结

基于形式化方法建模的优点，在于

- 形式化方法和语言有严格定义的数学基础和严格定义的推理规则
- 可以开发自动化的工具帮助我们检查模型的语义一致性

无论用任何方法建立模型，包括写程序，都不可能提供下面功能：

**“保证模型与被模拟 / 解决 / 处理的实际问题一致”**

这一问题只能在现实世界里，通过其他技术和方法另行检验

B 方法和语言为软件和系统的严格建模提供了一种形式化模型，它

- 有良好定义的数学基础
- 但在适合软件开发工作的需要，容易为今天和未来的软件工作者接受等方面还存在差距

这些说明，形式化方法还需要发展



# 总结

用形式化方法建模，有几类主要的应用

1. 为需要开发的软件系统建模。其中的工作重点是
   - 通过形式化的严格描述，澄清软件系统的需求
   - 验证软件模型的内在一致性
   - 通过严格描述和证明的过程，进一步认识所需要开发的系统，挖掘出在前期非形式化的需求分析中没有认识到的问题
2. 为已有的系统建模，主要的用途是
   - 帮助发现已有系统中潜藏很深的问题，或确认系统的性质
   - 作为逆向工程的工具，为今后的修改开发做准备
3. 建立某种系统的模型，与实际软件开发无关，只是为了帮助我们认识有关系统或过程的行为或性质

通过本课程的学习，大家初步掌握了一种有用的建模工具。这一工具可以应用到任何需要建模的领域中，而且有有效的自动化验证工具支持，能帮助我们检查模型中的遗漏或错误

# 总结

B 方法希望支持软件开发的全过程，为此，它在提供了基本的模型描述手段 AMN（Abstract Machine Notation）的基础上，进一步考虑了许多问题

1. 如何支持对于抽象规范的进一步开发
2. 如何支持从规范到实际的可执行程序的开发过程
3. 如何组织大型的软件规范

精化是两个抽象机组件之间的关系。B 方法为抽象机的精化建立了严格的数学基础：与精化关系相应的证明义务（精化定理）

一旦证明了所有的精化定理，我们就能保证 "精化抽象机" 的行为必定满足被精化的抽象机规范的要求

在精化的过程中，我们可能做

- 数据精化，用更接近实现的数据表示取代原来的数据表示
- 行为精化，消除抽象机行为中的非确定性



# 总结

通过精化，可以在不同层次上建立满足抽象规范的模型，其中的数据表示和操作都可以有不同考虑，以满足在不同层次上描述系统行为的需要

第一层抽象规范限定了系统部件的行为准则，精化为系统的行为方式和细节提供了灵活性（例如，设法利用已有的基本抽象机组件等）

随着系统规范的逐步精化，得到的规范描述越来越接近系统的实现

这一过程实际上利用了 B 方法 AMN 语言的两重性

- 作为规范语言，它是一种有严格意义、可以严格检查的数学模型语言
- 其中的一个子语言可以直接对应到常见的高级编程语言的结构

当一个软件系统的所有组件的规范都精化到用 B 的可实现的子语言描述时，这一系统的规范就直接对应于一个具体实现了

如果有生成可执行代码的自动工具，就可以从这种实现层规范自动生成某种高级程序设计语言的系统。如果没有自动工具，手工将其翻译到具体高级语言程序也已经很直接了

# 总结

无论是自动生成，或是手工翻译，最终都能得到可执行的实现

在这一开发过程中，对于自动支持工具，也有许多问题可以提出来：

- 自动化证明工具的证明可以信赖吗？
- 到具体高级语言的自动化翻译工具是否确实能保语义？等等

这些都不是简单问题，需要深入的理论研究工作和长期的实际应用的检验

即使上述工作有了肯定的回答，高级语言程序还需经过编译程序转换到机器语言程序（编译），这一层次的保语义性质也需要认真考虑

最近一项重要工作就是有人证明了一个编译器（从 C 语言的一个相当大的子集到 MIPS 机器语言）的正确性。在 Communications of the ACM 和《中国计算机学会通讯》上都有相关论文，可以参考

总之，要保证形式化开发的 “工具链"，还有许多工作需要去做



# 总结

要开发大型的软件系统，不仅要解决基础的语义层面的一个描述问题，还需要更高层次的规范组织。B 方法和语言的设计在这方面也有所考虑

- 高层的抽象层面的规范描述提供了 **INCLUDES**, **USES** 和 **SEES** 连接，用于组织复杂的抽象机规范和精化规范
- 在实现层面上提供了 **IMPORTS** 和 **SEES** 连接，支持通过一组实现组件构造复杂的可执行的软件系统

这些组织方式的可用性和价值，已由一些大型实际系统的开发得到验证。但其表达能力是否满足范围更广泛的软件（和硬件）系统开发的需要，还需要更多开发实践的进一步检验

（大家可以就自己的工作，对这方面的情况提出自己的看法）

B 方法在支持大型软件开发方面提出了一套想法，还需进一步开发和完善

由于软件和其他计算机化的系统在社会生产生活中扮演着越来越重要的地位，我们可以预计，B 或其他形式化方法和形式化研究会在未来各种系统的开发中扮演越来越重要的角色

**谢谢大家参加本课程的学习!**

希望大家能在今后的学习和工作中

考虑一下形式化方法

能否对解决你面临的问题有所帮助