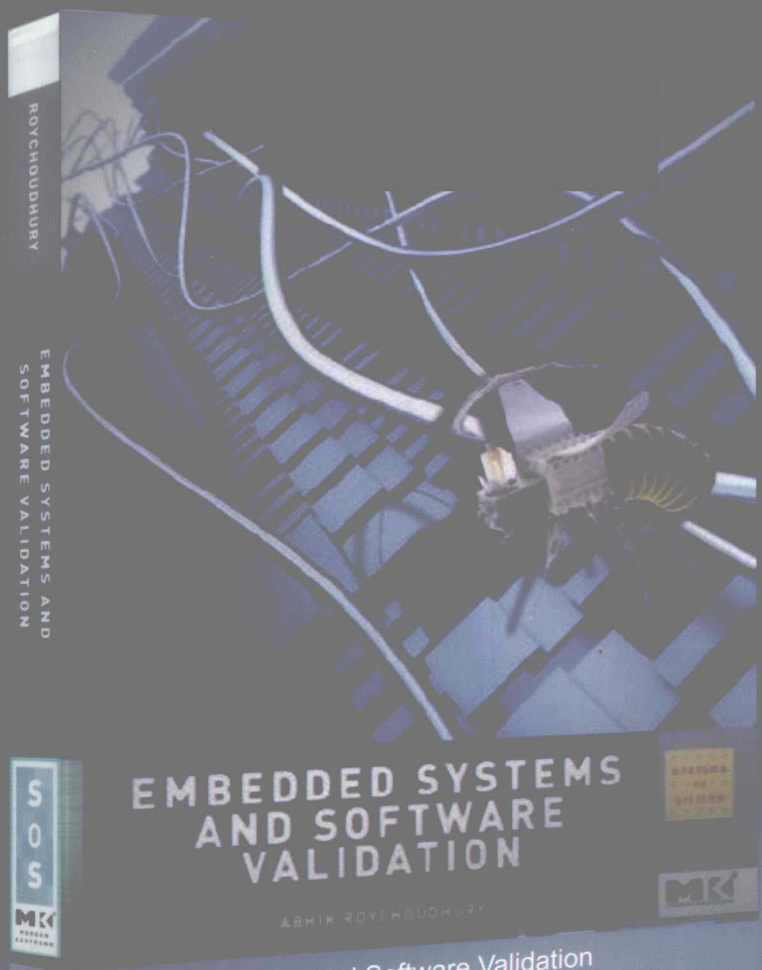
国外计算机科学经典教材

嵌入式系统设计的验证与调试技术

(印度) Abhik Roychoudhury 著　田尊华 译



Embedded Systems and Software Validation

ELSEVIER

MK MORGAN KAUFMANN

清华大学出版社

国外计算机科学经典教材

嵌入式系统设计的验证与调试技术

(印度) Abhik Roychoudhury 著

田 [redacted] 译

清华大学出版社

北 京

Embedded Systems and Software Validation
Abhik Roychoudhury
EISBN: 978-0-12-374230-8

Authorized Simplified Chinese translation edition published by Elsevier (Singapore) Pte Ltd Press and Tsinghua University.
ISBN: 978-9812725011

北京市版权局著作权合同登记号　图字：01-2009-7470

图书在版编目(CIP)数据

嵌入式系统设计的验证与调试技术/(印)罗伊乔杜里(Roychoudhury，A.)著；田尊华 译.
—北京：清华大学出版社，2010.7
(国外计算机科学经典教材)
书名原文：Embedded Systems and Software Validation
ISBN 978-7-302-23072-4
I. 嵌… II. ①罗… ②田… III. 微型计算机—系统设计 IV. TP360.21
中国版本图书馆 CIP 数据核字(2010)第 113950 号

责任编辑：王　军　赵利通　　**装帧设计**：孔祥丰
责任校对：成凤进　　**责任印制**：王秀菊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：185×230　**印　张**：13.5　**字　数**：278 千字
版　　次：2010 年 7 月第 1 版　　**印　　次**：2010 年 7 月第1 次印刷
印　　数：1～4000
定　　价：29.00 元

产品编号：034803-01

出 版 说 明

近年来，我国的高等教育特别是计算机学科教育，进行了一系列大的调整和改革，亟需一批门类齐全、具有国际先进水平的计算机经典教材，以适应我国当前计算机科学的教学需要。通过使用国外优秀的计算机科学经典教材，可以了解并吸收国际先进的教学思想和教学方法，使我国的计算机科学教育能够跟上国际计算机教育发展的步伐，从而培养出更多具有国际水准的计算机专业人才，增强我国计算机产业的核心竞争力。为此，我们从国外多家知名的出版机构 Pearson、McGraw-Hill、John Wiley & Sons、Springer、Cengage Learning 等精选、引进了这套“国外计算机科学经典教材”。

作为世界级的图书出版机构，Pearson、McGraw-Hill、John Wiley & Sons、Springer、Cengage Learning 通过与世界级的计算机教育大师携手，每年都为全球的计算机高等教育奉献大量的优秀教材。清华大学出版社和这些世界知名的出版机构长期保持着紧密友好的合作关系，这次引进的“国外计算机科学经典教材”便全是出自上述这些出版机构。同时，为了组织该套教材的出版，我们在国内聘请了一批知名的专家和教授，成立了专门的教材编审委员会。

教材编审委员会的运作从教材的选题阶段即开始启动，各位委员根据国内外高等院校计算机科学及相关专业的现有课程体系，并结合各个专业的培养方向，从上述这些出版机构出版的计算机系列教材中精心挑选针对性强的题材，以保证该套教材的优秀性和领先性，避免出现“低质重复引进”或“高质消化不良”的现象。

为了保证出版质量，我们为该套教材配备了一批经验丰富的编辑、排版、校对人员，制定了更加严格的出版流程。本套教材的译者，全部由对应专业的高校教师或拥有相关经验的 IT 专家担任。每本教材的责编在翻译伊始，就定期不间断地与该书的译者进行交流与反馈。为了尽可能地保留与发扬教材原著的精华，在经过翻译、排版和传统的三审三校之后，我们还请编审委员或相关的专家教授对文稿进行审读，以最大程度地弥补和修正在前面一系列加工过程中对教材造成的误差和瑕疵。

由于时间紧迫和受全体制作人员自身能力所限，该套教材在出版过程中很可能还存在一些遗憾，欢迎广大师生来电来信批评指正。同时，也欢迎读者朋友积极向我们推荐各类优秀的国外计算机教材，共同为我国高等院校计算机教育事业贡献力量。

清华大学出版社

国外计算机科学经典教材

编审委员会

译 者 序

21 世纪是信息时代，尤其是计算机时代，也可以讲 21 世纪是嵌入式系统时代。从复杂的航天系统到简单的遥控器，无不与嵌入式系统相关。嵌入式系统已经与我们的日常生活密切相关，例如，移动电话、PDA、电子阅读器、各种家电、汽车、门禁系统、各种 IC 卡等等，可以讲，如果现在离开了嵌入式系统，我们的日常生活工作将无法正常进行。嵌入式计算就是在这种背景下提出来的，这也足以说明嵌入式系统的普及程度和重要性。

从安全可靠的角度来讲，嵌入式系统可以分为安全关键的(safety-critical)和非(或一般)安全关键的。属于安全关键的嵌入式系统有车辆、飞行、核电厂和医疗设备等方面的控制系统。属于非(或一般)安全关键的嵌入式系统有移动电话、HDTV、家电控制器等。不管嵌入式系统是否为安全关键的，可靠性都很重要。安全关键的系统出现问题可能意味着巨大的生命财产损失，而对于非(或一般)安全关键的嵌入式系统来说，出现问题时即便不会造成任何生命财产损失，也意味着对用户使用信心的打击，而这将直接关系到相关产品的市场前景。那么如何保证嵌入式系统和软件的可靠性呢？本书正是为此而编写。本书系统地讲述了嵌入式系统设计流程中各个阶段的验证问题。当然，根据嵌入式系统不同的安全需求，考虑到成本因素，在验证上可以区分对待。相比一般系统而言，对于安全关键的系统必须进行更加严格的验证。

本书的显著特征是针对嵌入式系统和软件这个特定的领域，从不同的层面系统地讨论了各种验证方法。本书具有的第二个特点是，与一般讲理论的书不同，本书在讲述过程中紧密结合具体示例(如空中交通管制系统和汽车控制系统)和标准案例(西门子基准测试套件中的代码段)。此外，本书还有针对性地给出了少而精的习题，为巩固相关知识和激发读者思考提供了良好的题材。

本书由国防科学技术大学的田尊华翻译，由肖国尊负责监督本书的翻译质量和进度。鉴于译者水平有限，时间仓促，不足和疏漏之处在所难免，敬请广大读者提供反馈意见。读者可以将意见发送到 wkservice@vip.163.com，我们会仔细查阅读者发来的每一封邮件，以求进一步提高今后译著的质量。

译者

2010 年 3 月

前　言

本书针对的主要是嵌入式软件和系统的验证问题。有很多关于该主题的书籍，用适当的搜索术语在 Web 上搜索一下，就可以证明这一点。既然如此，为什么还要编写本书呢？

可以从几个方面来回答这个问题。首先，也是最直接的回答就是，现有的有关书籍大多都是关于嵌入式系统编程或协同设计方面的。在这些书籍中，验证通常作为附带内容在最后才进行讨论。在本书中，我们将验证看作设计过程中的首要内容，并将验证过程与设计过程本身结合在一起。

本书的重点在于验证，但这里的验证是着眼于嵌入式软件和系统的。本书涉及的方法(测试/模型校验)还可以从完全通用的角度来讨论，只关注技术，而不考虑如何将这些技术用于系统设计过程。但是，我们没有这样做。虽然本书的重点在于验证方法，但是我们仍然清楚地讲述了如何将它们用于系统设计。作为一个示例，我们分别两次以不同的方式提出并讨论了模型校验方法，即分别在系统模型的层面上(第 2 章)和在系统实现的层面上(第 5 章)。

最后，本书的重点基于嵌入式软件和系统，但并不限于功能验证。我们至少还讨论了另外两个方面的内容，即性能和通信行为调试。因此，本书包含了很难在一本书中找到的分析方法，包括测试(非形式化验证)、模型校验(形式化验证)、最差情况执行时间分析(程序性能静态分析)、可调度性分析(系统级性能分析)等等，它们有机地交叉融合在一起，以便帮助您实现可靠的嵌入式系统设计。

本书的第 1 章对嵌入式系统验证中的问题进行了一般性介绍，并从整体的角度对功能和性能验证的差异进行了讨论。

第 2 章讨论了模型级(level of model)验证。我们从对系统结构和行为的一般性讨论开始，然后逐步讲解类似有限状态机(Finite-State-Machine，FSM)和消息顺序图(Message Sequence Chart，MSC)这样的具体行为建模概念，并对这些模型的仿真(simulation)、测试和形式化验证进行了讨论。我们讨论了基于模型的测试，在这种测试中使用由模型生成的测试用例(test case)对系统实现进行试验。另外还介绍了属性校核(property verification)和常用的模型校验方法，并在该章最后对 SPIN 和 SMV 这样的实用验证工具进行了很好的讨论。由此可见，该章的内容对应于模型级调试(model-level debugging)。

第 3 章讨论了解决嵌入式系统组件之间的通信不兼容性的问题。我们讨论了解决这类

不兼容问题的不同策略，如用合适的接口增强组件，或构建一个集中通信协议转换器。由此可见，该章内容对应于通信调试(communication debugging)的内容。

第 4 章讨论了系统级性能验证。我们首先介绍了软件时间分析，并着重讨论了最差情况执行时间(Worst-Case Execution Time，WCET)分析。接着对程序执行过程中不同干扰导致的时耗估计进行了讨论，其中干扰可能来自外部环境，也可能来自相同或不同处理单元上执行的其他程序。另外讲解了用于估计这些干扰导致的时耗的合适分析方法。然后描述了通过系统级支持来解决执行时间不可预测性的机制。尤其是，描述了编译器控制的存储器或中间变量存储器。该章最后还讨论了新兴应用程序中的时间可预测性问题。由此可见，该章的内容对应于性能调试(performance debugging)。

第 5 章讨论了嵌入式软件的功能调试。我们对形式化和非形式化方法都进行了讨论，并且对测试和形式化校验的讨论所占的比重相近。该章的前半部分讲解了构建在测试或动态分析之上的验证方法进行了讨论，后半部分则重点讨论了形式化校验，尤其是软件模型校验。该章最后还讨论了形式化校验与测试的结合。由此可见，该章的内容对应于软件调试(software debugging)。

除了在第 2 章和第 5 章中有一些调试/验证方法是共同的之外，您可以单独阅读这些章节。但是，关于该主题的大学高年级或研究生课程可以按顺序阅读这些章节。

目　录

CHAPTER 1

嵌入式系统验证简介

嵌入式软件和嵌入式系统已经逐步成为我们日常生活中使用计算机和计算的主导方式。计算机不再是放在桌子上的独立实体，相反，通过我们直接或间接使用的小型设备，计算机已经完全融入了我们的日常生活，这些设备包括移动电话、洗衣机、微波炉、汽车控制器和飞行控制器。实际上，嵌入式系统是如此普及，以至于今天它们可以完成大量的计算任务，从而使“嵌入式计算”成为一个新的研究领域。本书主要关注嵌入式软件和系统的验证，其目的是要开发出具有可靠功能和时间特性的嵌入式系统。

并非所有的嵌入式系统都是安全关键的。一方面，有很多安全关键的嵌入式系统，如汽车、交通(火车)控制、飞行控制、核电厂和医疗设备。另一方面，还存在更多的一般或不是那么安全关键的嵌入式系统，如移动电话、HDTV(高清晰度电视机)、家用设备(如洗衣机、微波炉和空调)控制器、智能衬衫等等。毫无疑问，不管嵌入式系统是不是安全关键的，将验证过程集成到嵌入式系统设计流程中的每个阶段都是至关重要的。当然，对于安全关键的嵌入式系统，有必要进行更加严格的验证。正因如此，所以要使用形式化分析方法，因为这样可以为系统的功能/时间特性提供数学保证，至少在设计的某些阶段需要这样。

在本书中，我们所关注的重点在于验证方法，以及如何有效地将这些方法与嵌入式系统设计过程结合起来。在深入讨论这些内容之前，首先直观地解释一下出现在验证过程中的一些通用术语，包括测试、仿真、校验和性能分析。

- 测试(testing)是指对于给定的系统输入，校验系统的行为是否与期望的行为一致的过程。在此，被校验的系统可以是即将运行的实际系统。但是需要注意，只对给定的输入进行校验，并非针对所有的输入。
- 仿真(simulation)是指基于给定的输入来运行系统。但是，仿真在以下一个或两个方面不同于实际系统的执行：
  - 被仿真的系统可能只是实际执行系统的一个模型。这一点对于功能仿真是有益的，因为可以在建立实际系统前，检查在所选输入情况下系统模型的功能。

- 系统执行的平台是仿真的，它不同于实际的执行平台。这种情况对于性能仿真非常普遍。实际系统的执行平台可能根本就不存在，或者它需要通过性能仿真过程才能确定下来。典型情况下，执行平台的软件模型是可以用于性能仿真的。
- 形式化校核(formal verification)是指对于所有可能的输入检查系统的行为是否与期望的行为一致的过程。穷举测试是低效的，甚至是不可行的，而校核是可以通过静态分析系统模型来实现的(系统模型可以用类似有限状态机这样的结构表示)，因而可以避免这种低效。
- 最后，需要注意，形式化校核方法通常用于为系统功能提供严格的数学保证。但是，为了给出性能的严格保证(例如，给出特定软件执行时间的上限)，需要应用估计性能的数学分析方法。这些方法通常都冠以性能分析的名称。

为了了解将验证过程集成到嵌入式系统设计流程中的可能性和时机怎么样，可以看一下汽车行业。普遍认为，汽车电子设备具有广阔的市场，现代汽车中越来越多的功能都是由软件控制的。实际上，汽车软件上的创新可以带来新的设计，这通常也是汽车制造商他们自己认同的观点。当今有一些著名的观点可以证明在现代汽车设计中嵌入式软件/系统的重要性，如“现在生产的汽车中超过 90%的创新来自于软件”。自然，由于在汽车驾驶过程中，各种汽车部件功能“正确”十分重要，因此对控制这些部件的硬件/软件的严格验证是极其关键的。换言之，集成了各种调试/验证过程、具有可靠和健壮特性的嵌入式系统设计流程是必不可少的。

为了深入理解对于汽车而言嵌入式系统验证的重要性，我们可以更进一步地研究汽车的各个组成部件，其中有些可能是由计算机控制的。大致来讲，这些部件可以分为 3 类，即引擎功能、驾驶室功能和娱乐功能。显然，引擎功能是最安全关键的，而与车内娱乐相关的功能是最不安全关键的。引擎功能包括像刹车和方向盘这样的关键功能；通常这些功能与硬实时约束相关。驾驶室功能包括关键程度较低(但很重要)的功能，如电动车窗和空调装置。娱乐或信息功能包括车内设备的控制，如 GPS 导航系统、CD 播放器和车内电视，以及这些设备间的通信功能。很显然，控制引擎功能的计算部件(如刹车)需要非常严格的验证，以至于必须对这些计算部件的行为进行形式化建模和校核。对于汽车驾驶室功能，至少需要对控制驾驶室功能的计算部件进行建模和广泛的测试。对于信息功能，我们需要使用性能分析方法来保证满足必要的软实时约束。

由此可见，与我们在汽车软件这个特定领域中讨论的一样，对于复杂的嵌入式系统往往需要进行不同类型的验证。对于系统中安全关键程度较高的部件，可能需要严格的建模和形式化校核，而对于系统中安全关键程度较低的部件，进行更多的测试可能就足够了。而且，对于控制或保证对环境的实时反应的系统部件，需要进行详细的性能验证。为此，我们采用的验证方法包括从形式化方法(如模型校验)到非形式化验证(如测试)的各种方法。

而且，我们可能采用具有不同抽象层次的验证方法，分别为模型级验证；或高层实现验证(在这个层次的验证中，只考虑组件之间的行为，而不考虑组件内部的行为)；或低层实现验证(也考虑系统组件内部的行为)。最后，验证准则也可能有所不同，我们可能需要在不同的层次上完成验证，以检查存在的功能错误、时序错误等。

图 1-1 直观地描述了嵌入式系统验证的复杂关系。尤其是在图 1-1(a)中，显示了系统验证的不同抽象级别(模型/实现)和准则(性能/功能)。

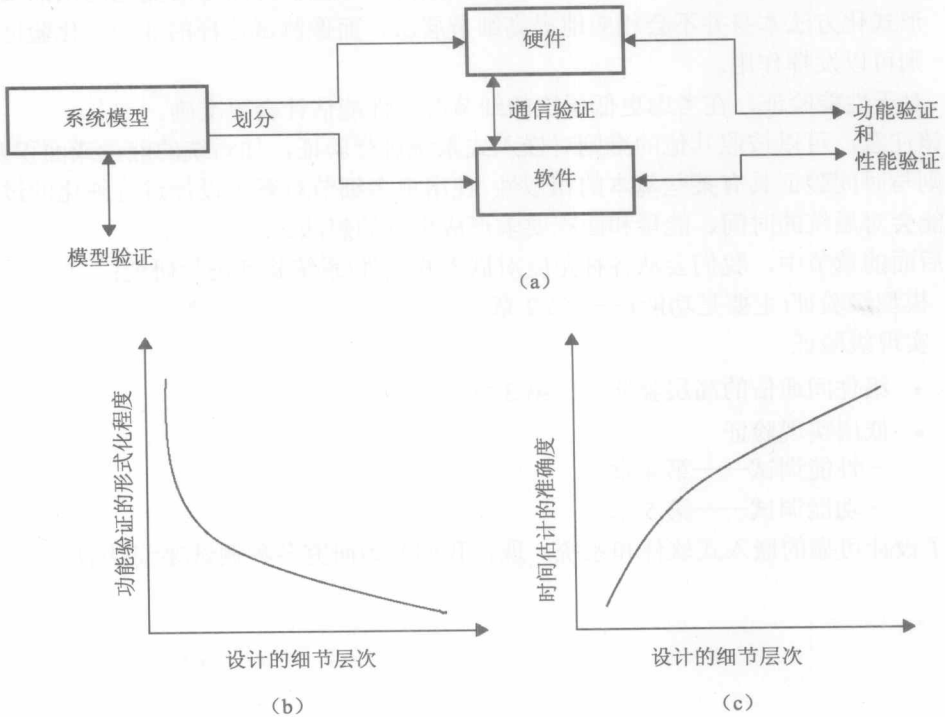


图 1-1　嵌入式系统功能和时间验证中的争论点

图 1-1(b)图解说明了功能验证的复杂性。对于希望构建的嵌入式系统，我们可能需要设计它，并在不同的细节层次(或不同的抽象层次)上对其进行精化。如果我们正在进行功能验证，那么细节层次越高，验证方法的形式化程度就越低。因此，对于较高抽象级别的系统设计，我们可以完全使用形式化验证方法。另一方面，一旦我们开始充实所构建系统的实现细节时，就可能需要更多的非形式化验证方法，如广泛测试。

与功能验证相比，时间验证的图形有所不同，如图 1-1(c)所示。这很好理解，嵌入式

系统通常包括对系统与外部环境交互(在这里就是与系统不同组件之间的交互)的硬实时和软实时约束。所以，时间验证包括建立“系统响应时间”(对应于环境中出现的某些事件)的精确估计。很显然，在充实嵌入式系统的细节时，我们可以建立更准确的时间估计，从这种意义上讲，就是可以执行更详细的时间验证。

由此可见，图 1-1 中给出了验证功能与验证时间特性中的问题，在嵌入式系统设计流程中，二者都非常重要。在此，图 1-1 强调的是两个不同的方面：

- 在较高抽象层次上执行功能形式化验证更好。在我们开始考虑更低层次的细节时，形式化方法本身并不会相应地提高细节层次，而像测试这样的非形式化验证方法则可以发挥作用。
- 对于性能验证，在考虑更低层次的细节时，性能估计会更准确。

应该注意，可以按照其他的准则对嵌入式系统进行验证，如系统的能量或面积要求，这些准则与时间验证具有某些基本的相似性。在用更多细节对系统设计进行细化的过程中，我们可能会对系统的时间、能量和面积要求形成更好的想法。

在后面的章节中，我们会从各种角度对嵌入式软件/系统验证进行研究：

- 模型级验证(主要是功能)——第 2 章
- 实现级验证
  - 组件间通信的高层验证——第 3 章
  - 低层实现验证
    - 性能调试——第 4 章
    - 功能调试——第 5 章

为了设计可靠的嵌入式软件和系统，现在我们开始研究各种调试/验证方法。

CHAPTER 2

模型验证

现在，我们开始设计过程的第一步——系统建模。为了描述这一步，我们需要阐述什么是模型。在清楚地理解了系统模型以后，我们还可能要描述建模过程(就整个系统设计生命周期而言)中校核与验证的作用。事实上，构成系统模型的东西通常是引起混淆的主要根源。引起这种混乱的主要根源之一就是设计的嵌入式系统的行为和体系结构之间的差异。

按照最简单的形式来讲，*系统结构*(system architecture 或 system structure)是指组件之间的互联，而*系统行为*(system behavior)则指的是组件如何改变状态，这可能是通过组件之间的通信实现的。这种理解尽管直观，但经常引起更严重的问题，因为我们需要澄清“组件”的概念。在本书中，我们将设计中独立的实体定义为*进程*(process)，或*活动对象*(active object)，或*组件*(component)。进程具有自己的控制流，可以通过执行独立的动作或通过与其他进程通信来改变状态。系统结构指系统组件之间的结构关系，而系统行为则指组件是如何改变状态的。

现在列举一个具体的示例来进一步说明系统结构与系统行为之间的差异。考虑空中交通管制器(Air-Traffic Controller，ATC)的示例，ATC 需要与多个客户端通信。每个客户端都可以表示为一个微型控制器，每个新到达的飞机都有一个微型控制器，用于为到来的飞机提供输入信息。中心控制器与这些客户端交互以提供重要的更新信息(如天气信息)，这些信息由新到达的飞机用作输入以进行决策(如速度和跑道计算)。基本的系统结构可以直观地表示为图 2-1。

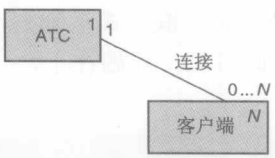


图 2-1　空中交通管制系统的基本系统结构

图 2-1 表示空中交通管制系统中有一个 ATC 和任意数量的客户端(例如 N 个客户端)，而且一部分客户端正在接收来自 ATC 的更新信息，在图中是用连接关系(连接的基数为 0...N)来表示的。这正是系统结构的含义。但是，图 2-1 并没有给出 ATC 与客户端之间的交互协议，即 ATC 如何响应客户端发出的请求，以及客户端如何发出请求，这些都属于系统“行为”。

如果您熟悉统一建模语言(Unified Modeling Language，UML)[54]，那么我们可以给出如下类比。系统结构用 UML 类图描述，包括系统中的类、每个类中对象的数量、类之间的关系和关系的基数(cardinality)。另一方面，系统行为是指系统对象会因为控制流和对象间的交互而发生状态变化。这种状态变化可以通过 UML 状态图(state diagram)和 UML 顺序图(sequence diagram)进行描述。典型情况下，每个类都会有一个状态图，用于描述类中对象状态的变化。此外，可以用顺序图描述对象间的交互。

如果您并不熟悉 UML，那也没有关系，我们很快就会详细地介绍这些术语。但在介绍这些术语之前，我们先要研究一下示意性的空中交通管制(ATC)示例，由此彻底弄明白系统行为的概念，以及对其建模的重要性。

2.1 平台与系统行为

到此为止，我们已经将行为定义为有状态对象间的交互引起状态变化的机制[1]。然而，这个定义太宽泛，无法区分设计中的系统和实现系统所使用的平台。在进一步阅读后面的内容之前，澄清这种差异是很重要的。

在示意性空中交通管制应用的示例中，“设计中的系统”是指 ATC 和客户端。系统行为则是指 ATC 与客户端之间的交互协议。但是，这并没有说明如何实现这些交互协议。ATC 和这些客户端有可能是在分离的电子控制单元(Electronic Control Unit，ECU)中实现的，而这些 ECU 则是通过总线进行通信的。在此，总线和总线访问协议(决定谁以及什么时候可以使用总线传输信息)是平台描述的一部分。显然，总线访问协议指的是平台行为(platform behavior)。

在谈到行为建模时，在形式化方法或软件工程领域中，通常是指对设计中的系统行为进行建模。平台在设计中只会起到很少作用，甚至不起作用，因为初步抽象出来的系统描述通常是假定与平台无关的。但是，对于嵌入式系统则不能这么认为，嵌入式系统通常是与平台相关的(platform-aware)。由此可见，问题在于，当我们描述行为时，应该如何将平台建模与系统模型结合起来？

1 有状态对象是指自身具有局部状态和控制流的实体。