13.5.3 经验的作用

拥有丰富经验的设计人员对于一个操作系统项目来说至关重要。Brooks指出，大多数错误不是在代码中，而是在设计中。程序员正确地做了吩咐他们要做的事情，而吩咐他们要做的事情是错误的。再多测试软件都无法弥补糟糕的设计说明书。

Brooks的解决方案是放弃图13-11a的经典开发模型而采用图13-11b的模型。此处的想法是首先编写一个主程序，它仅仅调用顶层过程，而顶层过程最初是哑过程。从项目的第一天开始，系统就可以编译和运行，尽管它什么都做不了。随着时间的流逝，模块被插入到完全的系统中。这一方法的成效是系统集成测试能够持续地执行，这样设计中的错误就可以更早地显露出来。实际上，拙劣的设计决策导致的学习过程在软件生命周期中应该更早就开始。

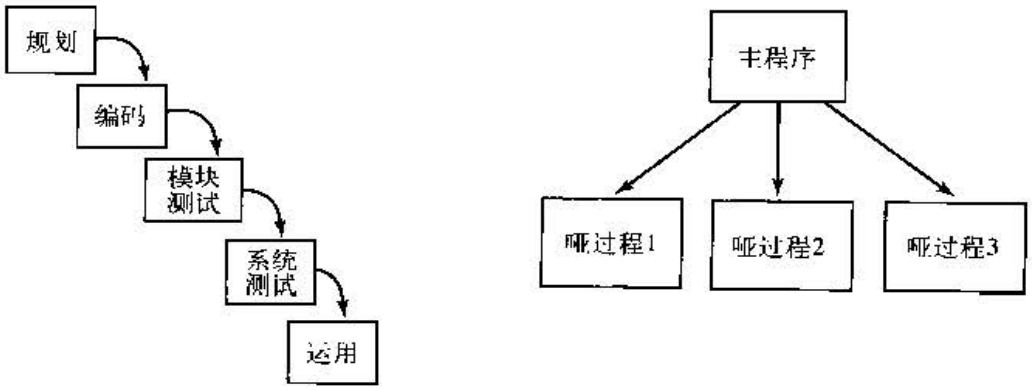

图13-11 a) 传统的分阶段软件设计过程；b) 另一种设计在第一天开始就产生一个（什么都不做的）工作系统

缺乏知识是一件危险的事情。Brooks注意到被他称为第二系统效应（second system effect）的现象。一个设计团队生产的第一件产品经常是最小化的，因为设计人员担心它可能根本就不能工作。结果，他们在加入许多功能特性方面是迟疑的。如果项目取得成功，他们会构建后续的系统。由于被他们自己的成功所感动，设计人员在第二次会包含所有华而不实的东西，而这些是他们在第一次有意省去的。结果，第二个系统臃肿不堪并且性能低劣。第二个系统的失败使他们在第三次冷静下来并且再次小心谨慎。

就这一点而言，CTSS和MULTICS这一对系统是一个明显的例子。CTSS是第一个通用分时系统并且取得了巨大的成功，尽管它只有最小化的功能。它的后继者MULTICS过于野心勃勃并因此而吃尽了苦头。MULTICS的想法是很好的，但是由于存在太多新的东西所以多年以来系统的性能十分低劣并且绝对不是一个重大的商业成功。在这一开发路线中的第三个系统UNIX则更加小心谨慎并且更加成功。

13.5.4 没有银弹

除了《人月神话》，Brooks还写了一篇有影响的学术论文，称为“No Silver Bullet”（没有银弹）(Brooks, 1987)。在这篇文章中，他主张在十年之内由各色人等兜售的灵丹妙药中，没有一样能够在软件生产率上产生数量级的改进。经验表明他是正确的。

在建议的银弹中，包括更好的高级语言、面向对象的程序设计、人工智能、专家系统、自动程序设计、图形化程序设计、程序验证以及程序设计环境。或许在下一个十年将会看到一颗银弹，或许我们将只好满足于逐步的、渐进的改进。

13.6 操作系统设计的趋势

做预测总是困难的——特别是关于未来。例如，1899年美国专利局局长Charles H. Duell请求当时的总统McKinley（麦金利）取消专利局（以及他的工作！），因为他声称“每件能发明的事物都已经发明了”(Cerf and Navasky, 1984)。然而，Thomas Edison（托马斯·爱迪生）在几年之内就发明了几件新的物品，包括电灯、留声机和电影放映机。让我们将新电池装入我们的水晶球中，并且冒险猜测一下在最近的未来操作系统将走向何方。