计 算 机 科 学 丛 书

软件工程

Java语言实现

Software Engineering with Java

(美) Stephen R. Schach 著　　袁兆山 等译

机械工业出版社 China Machine Press　McGraw-Hill

计算机科学丛书

软件工程
——Java语言实现

(美) Stephen R. Schach 著

袁兆山 等译

袁晓辉 等审校

机械工业出版社
China Machine Press
McGraw-Hill

本书介绍经典的和面向对象的软件工程，强调理论、抽象和设计相结合，重视对软件工程学有指导作用的重要概念。本书分两部分，共14章及8个附录。第一部分包括第1~6章，向读者介绍软件工程的概念，给出了本书的框架。顺序讨论了软件工程的范围，软件过程及其问题、软件生命周期模型、逐步求精、CASE工具、测试原理，详细解释了类和对象，并且说明为什么面向对象的范型比结构化范型更成功。本书的第二部分包括第7~14章，详细介绍软件过程的各个阶段，如需求、规格说明、计划、设计、实现与集成、维护、最终退役。还包括用于开发和维护软件的工具和技术，并对各阶段中有关的CASE工具、度量和测试技术加以说明。各章末都附有大量的练习。

本书内容广泛新颖，深浅适宜，是大学计算机科学系高年级学生和研究生的较优秀的教科书，也是对从事软件开发的管理者、系统分析员、程序员具有指导作用和实用价值的著作。

本书版权登记号：图字：01-1999-1419

图书在版编目(CIP)数据

软件工程：Java语言实现/(美)沙切斯(Schach,S.R.)著；袁兆山等译.-北京：机械工业出版社，1999.9
(计算机科学丛书)
书名原文：Software Engineering with JAVA
ISBN 7-111-07355-X

Ⅰ.软… Ⅱ.①沙…②袁… Ⅲ. Java语言 Ⅳ.TP312

中国版本图书馆CIP数据核字(1999)第27034号

出 版 人：马九荣（北京市百万庄大街22号 邮政编码100037）
责任编辑：陈剑匾
北京牛山世兴印刷厂印刷·新华书店北京发行所发行
1999年9月第1版第1次印刷
787mm×1092mm 1/16·25.75印张
印数：0 001- 5 000册
定价：38.00元

译 者 序

当代计算机硬件更新换代迅速，成本迅速下降，性能提高迅速。计算机的应用更加普及，使得科学和技术各个领域，工业和社会各个部门对高性能计算机系统的依赖性日益增强。

遗憾的是，计算机在使社会生产力得到迅速解放、社会高度自动化和信息化的同时，却没有使计算机本身的软件生产得到类似的巨大进步，软件生产方式仍然落后。经典的软件开发方法植根于计算机硬件性能比较低下、计算机应用尚不广泛的年代，因而无法摆脱人工方式的范畴，所开发的软件无法重用，重复性劳动无法避免，既大量浪费了宝贵的人力资源，浪费了财力、物力，也严重阻滞了软件产业的发展，制约了软件生产率的提高。这促使软件专家在改进和完善基于经典开发范式涉及的理论、技术和方法的同时，转向探索新理论、新思想、新方法、新技术和新途径方面，目的是冲破这种限制，求得软件生产和维护从根本上得到改进。

软件工程学是这方面研究成果的总结。软件工程学强调软件产品的生产特性，采用工程化的方法，围绕生产过程自动化、软件产品的可靠性展开对软件的生产方式、生产管理、产品设计方法、生产工具系统和产品质量保证等的研究，研究的重点是软件设计方法论及工程开发技术，并将计算机科学理论和工程方法相结合，重视对软件工程学有指导作用的理论、抽象和设计等重要概念的定义。软件工程学是研究软件生命周期一切活动，包括需求分析、计划、设计、实现、测试、维护及管理的科学。软件工程学研究的最终目的是为经济有效地开发软件系统提供科学的方法和工具。软件工程学既强调软件(一般指大型软件)开发的工程特点，又强调软件设计方法的科学性、先进性。

随着软件技术的迅速发展，软件工程学的研究内容也在不断发展。由Stephen R.Schach编著的本书(Software Engineering with Java)，以经典的和面向对象的软件工程相结合，突出最新研究成果，不仅介绍了经典的和面向对象的软件工程，还对两者进行了比较和分析。强调面向对象的范型，使读者完全领会到为什么面向对象方法要优于经典的方法。

目前国内比较缺乏反映软件工程最新研究成果的论著。为满足国内对软件工程教学和软件开发的需要，立足培养能跟上国际计算机科学技术发展水平的学生，我们翻译了Stephen R. Schach编著的Software Engineering with Java这本在软件界有影响和有实用价值的著作。

本书分两部分，共14章及8个附录。第一部分包括第1～6章，向读者介绍软件工程的概念，给出了本书的框架。顺序讨论了软件工程的范围、软件过程及其问题、软件生命周期模型、逐步求精、CASE工具、测试原理，详细解释了类和对象，并且说明为什么面向对象的范型比结构化范型更成功。本书的第二部分包括第7～14章，详细介绍软件过程的各个阶段，如需求、规格说明、计划、设计、实现与集成、维护、最终退役。还包括用于开发和维护软件的工具和技术，并对各阶段中有关的CASE工具、度量和测试技术加以说明。各章末都附有大量的练习。

阅读本书的读者，值得注意的方面是：

1) 本书对类、对象、继承、多态性和动态联编等基本的面向对象概念做出了明确的定义。介绍了面向对象生命周期模型、面向对象的设计、管理关系以及面向对象软件的测试和维护、对象的度量、内聚性和耦合性。强调了面向对象的范型，使读者完全领会到为什么面向对象方法要优于经典的方法。

2) 把Java当作软件工程中的一个实例研究，论述了Java和模块化、结构化程序设计、可移

植性、安全性、可靠性、常量、面向对象技术、可重用性、可见性，作者之所以选用Java作为该书的主要语言是因为Java不但是应用于WWW上的最好的语言，更是一种通用编程语言，Java语言的设计反映了良好的软件工程的实践，遵循了软件工程的原则。

3) 强调测试并不是一个独立的阶段，它不是仅在将产品交付给用户之前才进行，甚至也不是在软件生命周期中每个阶段结束时进行，而是和所有的软件生产过程并行。强调实现阶段和集成阶段应该并行地执行。介绍了能力成熟度模型(CMM)，对ISO 9000系列进行讨论并将其与CMM进行了比较，介绍了Petri网、Z等描述技术，还介绍了实时软件、并发软件的开发方法和重用技术。

4) 在每章末附有练习和重要的实例研究。本书有四种类型的练习。一种练习是为了巩固重要的知识，所有练习的内容可在书中找到。其次是软件学期项目，它被设计成供一个三人组的学生去解决。第三类问题是软件工程文献的阅读，每章中都选择了一篇重要的文献，要求学生阅读这些文献，并回答与其相关的问题。第四类问题是新增加的，与实例研究相关。要求学生修改一现成的产品，这比从头开发一新产品所能学到的知识要多。因此，在实例研究所在的每一章都至少有3个问题要求学生对实例研究进行修改。为了便于实例研究(附录C和H)的源代码修改，源代码可通过匿名ftp，从ftp vuse.vanderbilt.edu(129.59.100.10)的 /pub/Software_Eng/Java目录中获得。

本书内容广泛新颖，深浅适宜，是大学计算机科学系高年级学生和研究生的较优秀的软件工程教科书，也是对从事软件开发的管理者、系统分析员、程序员具有指导作用和实用价值的著作。

本书主要由合肥工业大学袁兆山教授翻译。参加本书翻译工作的还有合肥工业大学的苗沛荣、方德春、袁晓辉、宋宇、袁晓靖、李向上、张健、袁兆勤、张强等。由袁兆山、袁晓辉、方德春审校定稿。

本书翻译过程中得到了合肥工业大学计算机与信息学院刘宗田教授、蔡智明副教授的支持，在此表示诚挚的谢意。限于译者水平和时间仓促，译文难免有不足之处，欢迎广大读者不吝指正。

1999年3月于合肥

本书英文影印版已由机械工业出版社出版
书号：ISBN7-111-06714-2/TP. 890
页码：618页　开本：16　定价：51.00元

前　言

Java是应用于WWW上的最好的语言，因此，它对于在Internet和信息高速公路上的应用来说也是如此。此外，它还是一种适于从事计算机新闻工作的人员和在线杂志的专栏作家使用的编程语言。由于信息高速公路所具有的巨大的经济潜力，对于那些对计算机一无所知的投资者来说，Java可能也是他们最好的编程语言。

但是，这并非是我出版本书(本书是第三版Classical and Object-Oniented Software Engineering 的 Java 版)的原因。正相反，我之所以选择Java，是因为Java遵循了软件工程的原则。这一点在本书的结束语中有详细的论述，其标题为“Java：软件工程的实例研究”。

Java可应用于WWW applet(小应用程序)。这种应用程序可通过Web访问并且在个人的计算机上运行。但远不止如此，实际上，Java是一种通用的编程语言，它能用于各种软件。在Classical and Object-Oriented Software Engineering 每章的结尾处，有一连续的用C++编写的实例研究。本书中，实例研究用Java来实现，以证明Java能用于重要的软件。本书中一些问题要求学生对实例研究做些修改。实例研究的源代码(附录C和H)可通过匿名ftp从ftp.vuse.vanderbilt.edu(地址129.59.100.10)处的目录/pub/Sofware_Eng/Java 处获得，或通过地址Richard D.Irwin,1333 Burr Ridge Parkway,Burr Ridge,IL 60521得到磁盘。

学生也需要用Java来实现本书中的学期项目习题。指导手册(可从Richard D.Irwin 处获得)中详细的解答也表明了Java是真正通用的编程语言。

第3版Classical and Object-Oriented Software Engineering序言

第2版Software Engineering 于1993年出版。那时进行软件开发有两种主要的途径，即结构化范型和面向对象范型。结构化范型是一种可靠的途径，但它并非总是成功的。而面向对象范型似乎更有前途，但也不比前者好多少。第2版表明了这种态度。这本书当然包括了有关对象和面向对象设计的材料，但在这之前，则有点过早地强调了面向对象是一种新的范型，当时还没有证据表明该新范型比结构化范型更优秀。

在第2版出版后的3年时间中，逐渐地不断有证据表明面向对象范型比经典的软件工程范型要优秀。实际上，专门致力于面向对象软件工程的教课书将重新出版。

如果是这样的话，那么为何这本书的名字叫做Classical and Object-Oriented Software Engineering(经典的和面向对象的软件工程)呢？为什么经典技术仍被提及呢？这有两个原因。

首先，该书是大学高年级学生或研究生一年级的教科书。很可能使用本书的许多学生以后将为那些仍在使用经典软件工程技术的组织工作。此外，即使某个组织现在使用面向对象的方法来开发新的软件，但现存的软件仍需要加以维护，而这些软件却并非是面向对象的。因此，还得编入有关经典技术的材料。

第二个原因是，如果没有完全理解经典方法以及这些方法如何有别于面向对象的方法的话，那么就不可能理解为什么面向对象技术优于经典技术。因此，本书不仅描述了经典的和面向对象的方法，还将这些方法进行了比较和分析。这将使读者完全领会到为什么有如此多的专业软件人员认为面向对象方法要优于经典的方法。此外，如果学生受雇于一个还未采用面向对象技术的组织，他将能就新的范型的优缺点向组织提出建议。

因此此次版本的主要变化是强调了面向对象的范型。在第1章就介绍了对象，并通贯全书对其进行讨论。第6章题为“对象”，对例如类、对象、继承、多态性和动态联编等基本的面向对象概念做出了明确的定义(该章是第2版的扩展内容)。在面向对象的分析方面，另编写了一章，讨论了在第2版涉及的主题。另外，要特别注意面向对象范型中的面向对象生命周期模型、面向对象的设计、管理关系以及面向对象软件的测试和维护。本书也包含了对象的度量(metrics)、内聚性和耦合性。此外，还有许多有关对象的摘要，通常为一段话或一句话。原因是面向对象范型与其说与如何完成各阶段任务有关，还不如说它遍布于软件工程的方方面面。因此，面向对象的技术遍布于本书的各章节之中。

软件过程作为一个整体仍是这本书的基础。为了控制该过程，我们必须估量项目中会有什么事发生，所以要强调维护和度量。

第3版继续并扩展了原来各版本的其他主题。例如，第2版有关于能力成熟度模型(CMM)的讨论以及如何将其用于改善软件过程以提高生产率。在这一版本中，也对ISO 9000系列进行了讨论并将其与CMM进行了比较。

第3版在计算机辅助软件工程(CASE)领域有了不少新的进展。一方面，一些组织已对CASE失去信心，而另一些组织却在引入CASE，并在例如生产率、软件质量、员工士气等方

面看到了显著的进展。本书从中间角度来看待CASE，解释各组织为什么对CASE的态度上有这样的分歧。另外本书也包括了面向对象范型的CASE工具。

本书一直强调的主题包括维护的重要性和任何时候对完整正确的文档的需要，并且仍对软件重用的重要性进行了强调，但是在对象的背景中体现这点。

本书本质上仍遵循语言独立性。很少量的例子代码是用C++编写的。更确切地说，是无论在哪里都可能得到C++的C子集。此外，也注意了尽可能地少用C语言，以使那些对C语言不甚了解的人也能读懂本书。唯一使用C++(不是C)的章节是第6章，该章提供了详细的C++特定结构的说明。此外，在附录I中的实例研究的实现也用了一些C++结构。

就先决条件而言，该书假定读者熟悉一门高级编程语言，例如，Pascal、C、BASIC、COBOL或FORTRAN。尽管大多数例子是用C编写的，但却不必事先具有C的知识。另外，也假设读者已学过“数据结构”这门课程。

第3版如何组织

本书中章节安排的次序表示了软件生命周期各阶段的次序。特别是，本书的第二部分(第7章到第14章)是由软件生命周期的各个处理阶段所组成的，从需求阶段开始，以维护阶段结束。为了使读者有个准备，第一部分讲述了一些背景资料以便于第二部分的理解。例如，第一部分带着读者接触了CASE工具、度量和测试，因为第二部分中每一章都有相应阶段的CASE工具的一节、有关度量的一节、有关测试的一节。

为了确保第二部分中关键的软件工程技术能真正地被理解，每种技术都出现两次。首先，在介绍一种技术时，通过电梯问题举例说明。电梯问题的规模对读者来说是合适的，读者可从中看到应用该技术来解决一个完整问题的全过程，该例子也能足够精确地显示出所述技术的优缺点。然后，在每章末有一连续的重要的实例学习的详细解决办法。实例学习的每个阶段的资料一般来说都太多而无法在相应章节中给出。但是，解决办法的关键点写在相应的章节中，完整的材料在书末给出(附录C~I)。

问题集

第3版有四种类型的练习。就像以前一样，首先在每章的结尾有一些练习，以巩固重要的关键问题。这些练习是完备的，所有练习的技术信息可在书中找到。

其次，有一软件学期项目。它被设计成供一个三人组的学生去解决。三人小组是人数最少的小组，其中的成员无法在普通的电话中进行协商。该项目由14个独立的部分组成，每一部分都与相应的章节联系。例如，设计是第11章的主题，因此，在那一章中，该项目的组成部分与用该项目的软件设计相关。通过将大项目分解成更小的、明确的小块，导师能更密切的监控着课题的进展。项目的结构安排使得导师可自由地将14部分介绍的知识应用到他们所选择的任何其他项目中去。

因为本书是为研究生和大学高年级学生编写的，第三类问题是基于软件工程著作中的研究报告而提出的。每章中都选择了一篇重要的报告；在任何可能的地方，都选择了一篇与面向对象的软件工程有关的报告。对学生的要求是阅读报告，并回答与其相关的问题。当然，导师可自由地选择其他的研究报告；在每章后面“进一步的阅读”一节包含有各种相关的报告。

本书第四类问题是新增加的，即这些问题与实例研究相关。不少导师告诉我说，他们认为

学生通过修改一现成的产品比通过从开始开发一新产品所能学到知识要多。许多资深的软件工程师也同意这一观点。因此，在实例研究所在的每一章都至少有3个问题要求学生对实例研究进行修改。例如，在其中一章，要求学生使用与在实例研究中所使用的不同技术来重新设计该实例研究。在另一章，要求学生回答以不同的次序来进行面向对象分析的各步骤会有什么结果。为了使得修改实例研究的源代码(附录C和I)更加容易，源代码可用匿名ftp从ftp. vuse. vanderbilt. edu (129.59.100.10) 处的 /pub/Software_Eng/Third_Edition处得到，或通过地址Richard D.Irwin,1333 Burr Ridge Parkway,Burr Ridge,Illinois 60521得到磁盘。

指导手册包含有所有练习的详细的解决方法。它也可从Richard D.Irwin处获得。该书中所有的图表也可如此获得。

目　录

第二部分 软件过程的各个阶段

第一部分 软件过程

本书前6章有两个作用，即向读者介绍软件工程的概念并给出了本书的框架。软件过程是生产软件的方法，它从需求和概念考察出发，直到软件产品最终退役。在此期间，软件产品经历了一系列的阶段，如需求分析、规格说明、计划、设计、实现、维护、最终退役。软件过程还包括用于开发和维护软件的工具和技术，以及软件专业人员。

第1章“软件工程的范围”指出，软件产品开发技术必须经济有效，而且必须促进软件开发成员之间建设性的交互。这一章强调了对象这一概念的重要性，这也是全书所强调的一点。

第2章“软件过程及问题”叙述了软件工程中的许多问题，但没有提出解决方案，只是告诉读者各个问题在本书中哪个部分解决。因此，本章起到引导读者阅读本书的作用。

第3章“软件生命周期模型”详细介绍了各种不同的软件生命周期模型，其中包括：瀑布模型、快速原型、增量模型和螺旋模型。为使读者能对特定的工程项目选用适合的模型，本章对不同的生命周期模型做了类比分析和比较。

第4章题为“逐步求精、CASE和其他商业工具”。一个软件工程师必须具备使用各种不同的理论和实践工具的能力。本章将向读者介绍多种软件工程工具。逐步求精是工具之一，这项技术是把一个大问题分解成多个较小的、更易处理的问题。另一个工具是成本效益分析，该工具可判断一个软件方案是否在经济上可行。然后是计算机辅助软件工程(CASE)工具。CASE工具是一个帮助软件工程师开发和维护软件的软件产品。最后，为了实现对软件工程的管理，还必须进行各种分析，以判断工程是否处在正确的轨道上。度量(尺度)对一个工程的成功起着关键的作用。第4章的最后两个话题是，CASE工具和度量。这两个话题还将在第7章到14章详细讨论。这8章还将介绍软件生命周期的各个特定的阶段，其中讨论了支持每个阶段的CASE工具，并描述了管理各个阶段所需要的度量。

本书的一个重要的主题是，测试并不是一个独立的阶段，它不是仅在将产品交付给用户之前才进行，甚至也不是在软件生命周期中每个阶段结束时进行，而是和所有的软件生产过程并行。第5章“测试原理”讨论了测试中的一些基本概念，而针对生命周期中各个独立阶段的专门测试技术将在第7章到第14章介绍。

第6章“对象”是对象的引言，是第一部分的最后一章。本章详细解释了类和对象，并且说明为什么面向对象的范型比结构化范型更成功。本章的思想将在书中其余部分运用，尤其是第9章“面向对象的分析阶段”和第11章“设计阶段”(该章演示了面向对象的设计)。

第1章 软件工程的范围

有一个广为流传的故事。有一次，一位经理收到一张计算机打印出的帐单，上面标出他欠了0.00美元。在和朋友们嘲笑了“傻瓜计算机”之后，这位经理将帐单扔掉了。一个月之后，又来了一张类似的帐单，这一次标明他已欠款30天。其后又来了第3次帐单。第4次帐单在一个月之后来了，帐单上提醒他，如果他不一次付清0.00美元，他将可能承担法律责任。

第5张帐单标明他已欠款120天了，上面没有提示任何东西，只有一条粗暴而直率的消息威胁他，如果不立即付款，将对他采取所有可能的法律行动。由于害怕他的公司在这台发疯的机器控制下要付贷款利息，这位经理打电话给他的一位做软件工程师的熟人，向他叙述了这个难受的故事的全过程。那位软件工程师一边笑，一边告诉这位经理发送一张0.00美元的支票即可。这一招收到了很好的效果。几天以后，他收到一张0.00美元的收据。这位经理仔细地将收据收藏好，以防那台计算机在某个时候又说他还欠0.00美元。

这个广为人知的故事还有一个鲜为人知的结局。几天后，这位经理被他的银行管理人员召见。管理人员拿着一张支票问他，“这是你的支票吗？”

经理说，是的。

银行管理人员又问：“你不介意告诉我，为什么你要签一张0.00美元的支票呢？”

于是经理将整个故事重述了一遍。当经理说完之后，银行管理人员看着他，平静地问：“你知道你的$0.00支票对我们的计算机系统有何影响吗？”

计算机专业人员会嘲笑这个故事，但会有点感触。毕竟，我们每个人设计或实现的软件产品，在最初阶段也常会出现和送一封要收0.00美元的信一样愚蠢的结果。迄今为止，我们一直在测试过程中捕捉这类故障。但我们在笑这一故障时有些勉强，因为在潜意识中，我们担心某一天，在直到将产品交付给用户之后，我们才发现有故障。

在1979年9月9日，发生了一个完全不好笑的软件故障。这一天，美国战略空军司令部警钟长鸣，一片混乱，因为全球军事指挥和控制系统(WWMCCS)的计算机网络报告说，苏联向美国发射了一枚导弹。实际的情况是，这是一场模拟进攻，而计算机却解释成真实事情，就像约5年后电影“WarGames”中叙述的一样。尽管美国国防部因众所周知的原因，没有详细说明将测试数据解释成实际数据的精确机制，但我们可以合理地将其归结为软件故障。要么系统作为一个整体，并且没有设计成可以区分模拟和现实；要么用户接口没有包含必要的检查，以保证系统的用户能区分出虚拟和现实。换句话说，一个软件故障(如果这个问题确实是由软件故障引起的)可以使我们当今的文明痛苦地、突然地结束(可参见“1-I你想知道的进一步的信息”中描述的其他由软件故障引起的灾难)。

1-I 你想知道的进一步的信息

在WWMCCS网络引发的事件中，灾难只持续了几分钟就停止了。然而，其他软件故障的结局有时很悲惨。在1985年到1987年之间，至少有两个病人是死于Therac-25医疗线性加速器的过量辐射[Leveson and Turner, 1993]。其原因是控制软件中的一个故障。

在1991年海湾战争期间，一枚飞毛腿导弹刺入爱国者反导弹的外壳中，打中了位于沙特阿拉伯的一座军营，造成28名美国军人死亡，98人受伤。这是因为爱国者导弹

的软件包含有一个累加计时故障。爱国者导弹在设计时规定运行几小时，时钟将复位。该故障从未产生重大的影响，因而没有被发现。然而在海湾战争那枚导弹的电池连续运行了100多个小时，这使累计时间差变得太大，导致系统错误。

在海湾战争期间，美国将爱国者导弹运到以色列，以防卫飞毛腿导弹。以色列军队8小时就发现了这个定时问题，并立即向美国的导弹生产商报告。后者以最快速度修正了错误。但新软件在军营被飞毛腿导弹击中后一天才到达。

不论我们是做处理帐单软件还是防空软件，会出现交付期推迟、预算超出或软件充满残余故障。软件工程是解决这些问题的尝试。换句话说，软件工程是一门旨在生产无故障的、及时交付的、在预算之内的和满足用户需要的软件的学科。更进一步地，软件必须能按用户的需要，很容易做修改。为达到这些目标，一名软件工程师应努力获取技术上和管理上的范围广泛的技巧。这些技巧不仅要用在编程阶段，更要用在从需求分析到系统维护的软件生产周期中的每一个阶段。

软件工程范围极为广泛。软件工程的某些方面属于数学或计算机科学，其他方面可归入经济学、管理学或心理学中。为展示软件工程这一范围广泛的领域，我们将从5个不同的方面加以考察。

1.1 历史方面

电动机会出故障，但发生的频率远远小于工资单软件产品；桥墩有时会塌，但出现的次数远远小于操作系统崩溃的次数。

由于认识到软件的设计、实现、维护和传统的工程规则有相同的基础，于是NATO研究组于1967年提出了“软件工程(Software engineering)”这一术语。关于编制软件与其他工程任务类似的提法，得到了1968年在德国Garmisch召开的NATO软件工程会议的认可[Naur,Randell,and Buxton,1976]。这种认可不太令人惊讶，因为该会议的名称就已反映出将软件生产作为一种类似于工程行动的思想。委员会的结论是，软件工程应使用已有的工程规则的理论和模式，来解决所谓的“软件危机(software crisis)”，即解决软件的整体质量较低，以及最后期限和费用没有得到满足等问题。

软件危机在25年以后仍然困扰着我们。这告诉了我们两件事情。第一，软件生产过程在许多方面和传统的工程相似，但也有独特的属性和问题。第二，鉴于危机的长期性和症状不明显，软件危机应更名为“软件萧条(software depression)”。

桥墩崩塌的次数远远小于操作系统崩溃的次数，这一点确信无疑。那么为什么架桥技术不能用于创建操作系统呢？NATO委员会所忽略的一点是，桥墩和操作系统的区别，正如鸟和写字桌的区别一样。

桥墩和操作系统之间的主要区别在于，土木工程领域和软件工程领域对崩溃事件的态度不同。当桥墩崩塌时(例如，1940年Tacoma Narrow桥墩崩塌时)人们几乎总是要对桥墩进行重新设计和重新建造。因为桥墩的崩塌说明该桥的最初设计有问题，这将威胁人类的安全，所以必须对设计作大幅度的改动。此外，桥墩崩塌后，几乎桥的所有结构被毁掉，所以唯一合理的做法是将桥墩残留下的部分全部拆除再重新建造。更进一步地，其他所有具有相同设计的桥都必须仔细考察，在最坏的情况下，要拆除重新建造。

相比之下，一个操作系统的崩溃很少被认真考虑，人们很少立即对它的设计进行考察。当出现操作系统崩溃时，人们很可能只会重新启动系统，希望引起崩溃的环境设置不再重现。这