目录

[1. 书中的关键点： 1](#_Toc11436083)

[1.1. 提倡外科手术式的团队组织 1](#_Toc11436084)

[1.2. 软件项目的核心概念要由很少的人来完成 1](#_Toc11436085)

[1.3. 软件开发过程中必要的沟通手段 1](#_Toc11436086)

[1.4. 如何保持适度的文档 1](#_Toc11436087)

[1.5. 在软件开发的过程中，只有适度改进，没有包治百病的银弹 2](#_Toc11436088)

[2. 每章内容的总结 2](#_Toc11436089)

[2.1. 焦油坑 2](#_Toc11436090)

[2.2. 人月神话 2](#_Toc11436091)

[2.3. 外科手术队伍、 2](#_Toc11436092)

[2.4. 元老制、民主制和系统设计 3](#_Toc11436093)

[2.5. 第二个系统效应 3](#_Toc11436094)

[2.6. 沟通顺畅 3](#_Toc11436095)

[2.7. 巴别塔为何失败 4](#_Toc11436096)

[2.8. 掌控之中 4](#_Toc11436097)

[2.9. 袖里乾坤 4](#_Toc11436098)

[2.10. 文档先行 4](#_Toc11436099)

[2.11. 准备抛弃 5](#_Toc11436100)

[2.12. 良工利器 5](#_Toc11436101)

[2.13. 整体和局部 5](#_Toc11436102)

[2.14. 潜伏的祸患 5](#_Toc11436103)

[2.15. 另一面 6](#_Toc11436104)

[2.16. 没有银弹――软件工程的必然和偶然 6](#_Toc11436105)

[2.17. 再议“没有银弹” 6](#_Toc11436106)

[2.18. 是耶非耶 7](#_Toc11436107)

[2.19. 人月神话二十年 7](#_Toc11436108)

# 书中的关键点：

## 提倡外科手术式的团队组织

在软件开发组织上的过份民主，往往带来的是没有效率和责任，参与其中的人想法太多，层面参差不齐。所以，软件开发的组织，应该借鉴外科手术式的团队方式，有一个主要的负责人，其他人都是分工协作的副手，这样效率最好，结果最好。

## 软件项目的核心概念要由很少的人来完成

少就是多，项目的定位需要和功能多少的权衡。太多的想法，使项目没有焦点，什么都要放进去，结果什么都做不象

## 软件开发过程中必要的沟通手段

软件开发中最大的风险往往不是技术的缺陷，而是缺少沟通

## **如何保持适度的文档**

在开发中，保持适度的文档。喜欢过度多的文档的人，忘记了文档不是最终的产品，不是用户需要的，最后以为文档好，就是好的开发，其实完全不是。

## 在软件开发的过程中，只有适度改进，没有包治百病的银弹

在软件开发的过程中，重要的不是采用了什么工具，而是不论用何种工具，都要达到项目本身的客户需求。任何方法论之前，先要探求问题的来源，否则，对各种方法论的依赖或滥用，有害无益。

# 每章内容的总结

## 焦油坑

史前时代的焦油坑吞噬了成千上万个力大无穷的巨兽，今天的大型软件项目则令无数庞大的开发团队陷入无从逃脱的窘境。软件程序按其规模和目标的不同，对开放过程的要求也有极大的不同，这给软件开放这一职业带来无穷乐趣，同时也是这一行业苦恼的根源。

## 人月神话

软件开发项目常以人月来衡量工作量，这种度量暗示着人手和时间是可以互换的。这种“人多力量大”的想法是一种一厢情愿的虚妄神话，布鲁克斯法则：向滞后的软件项目追加人手会使得进度更迟缓。自本书第一版以来，这一法则在软件业广为传诵。

## 外科手术队伍、

虽然优秀的程序员的工作效率往往数倍于平庸的程序员，但若是缺乏合理的配置，优秀的成员未必能构成优秀的团队。大型软件开发项目的团队需要和外科手术组一样妥善分工，各司其职协调配合。

## 元老制、民主制和系统设计

概念完整性是系统设计中最重要的因素，尤其对于大型软件系统，概念完整性是项目顺利完成的必要保障，为获得概念完整性，架构设计由精简的架构设计小组负责，具体实现则围绕核心概念展开，架构设计和具体实现既相分离，又相辅相成。以建筑工程为类比，概念完整性也是软件项目通往成功的保证。

## 第二个系统效应

人们在第一个系统成功完成后，往往会在开发后续的第二个系统时犯冒进的错误。第二个系统经常成为过度设计或画蛇添足的牺牲品。要避免这种错误，必须在第二个系统开发时审慎地考查技术环境的变化，广泛进行交流和沟通，聆听各方面的建议，确立严谨的估算和规划。

## 沟通顺畅

架构设计通常由核心设计小组完成，将设计概念传达到整个开发团队是贯彻概念完整性的必然要求。以System 360的开发经验为例，要贯彻概念完整性，需要在团队中保持良好顺畅的沟通和交流，采用形式化定义等技术来确保概念被精确地定义和传达。独立的测试小组是系统质量的良好保证。

## 巴别塔为何失败

如果缺乏良好有效的沟通和协作，成员间难以有效的配合，团队项目的目标就无法实现。清晰的工作文档，明确的组织结构，合理的职责分配，都是大型软件项目最终成功的保证。

## 掌控之中

对大型软件系统产品的开发所需的时间和资源进行准确的估测，能让我们在项目进度和前景胸有成竹。软件代码的开发效率和代码模块之间所需的交互相关。界面交互复杂的程序需要更多的测试和调试时间，单纯地增加人手并不能有助于开发效率的提高。

## 袖里乾坤

最大化资源利用率，减少不必要的资源占用，合理规划，使软件系统在资源有限的情况下依然保证了良好的性能，从而实现良好的可伸缩性和健壮性，这能体现软件开发人员精湛的设计技巧。巧妙的数据结构往往能大幅度地俭省资源耗费，提高系统运行的性能。

## 文档先行

在软件项目开发过程中，文档是不可或缺的，文档为整个团队规范了概念，以便团队中的沟通协作，以及进度校验。本章阐述了软件系统项目中至关重要的几类文档。这些关键文档应及时地更新，始终作为项目进展的有效指南。

## 准备抛弃

变化是永恒的，用户的需求和期望在变化，开发者对用户需求的理解在变化，适用的技术也在变化，故而最佳的解决策略也可随之变化。软件开发团队应灵活地配置人力和资源，适应开发过程中的种种问题。程序的复杂性、用户需求的不确定性、软硬件技术环境的发展等因素导致了软件维护工作并非总是能够百分之百地获得回报。

## 良工利器

软件开发项目所选择的技术和工具对保障项目能否令人满意地如期完成至关重要。合适的开发工具、评测技术能有事半功倍之效果，切于项目实用的工具和技术是项目团队的重要财富。本章提供了当年软件开发项目选择技术和工具的重要原则和建议。

## 整体和局部

大而无当的笼统见地并不能表现你真正地理解了一个软件系统，应该具体而系统地深入了解各个局部的技术。良好的自顶向下的设计，不仅能保证概念完整性，也能及早消灭许多隐患。及早在软件项目中引入测试， 错误发现得越早，修复错误的代价越小。

## 潜伏的祸患

项目进度的滞后经常来自不易察觉的点滴延误的累积。软件项目的经理应该尽量建立可以明确量化的阶段性目标，定期进行严谨而规范的项目阶段性验收，了解项目的进展状况，并及时进行计划、资源和人力的调整。关键路径图等技术有助于观察项目的进度。

## 另一面

虽然用户直接使用软件系统，但在许多应用领域中，用户不可能仅仅凭借与软件的直接交互就迅速掌握其所有功能。故而提供给用户的使用说明等文档是软件呈现给用户的另外一面，它也能直接影响用户对软件的满意度和可用性评价。文档的用途决定它的形式和内容。

## 没有银弹――软件工程的必然和偶然

本文最初发表于1985年的IFIP第十届世界计算机大会上，此时距《人月神话》初版发行已有十年，其间计算机技术领域的变化令人振奋，但作者在此提出，由于软件的复杂性，一致性，变化性和不可见性，解决软件危机的银弹并不存在。作者点评了二十世纪80年代前期为业界寄予厚望的一些新技术，讨论了它们在克服软件危机中所具备的优势和缺憾。作者预言在近十年内，没有任何单独的软件工程进展可以使软件生产率有数量级的提高。

## 再议“没有银弹”

相比《人月神话》初版而言，1986年发表的“没有银弹”（第十六章）发表后引发了热烈的争论，本章结合20世纪80年代后期到90年代前期之间软件复用、面向对象程序开发等等新技术的发展状况，回应了对《没有银弹》一文各种主要异议，认为由于《没有银弹》一文归纳的软件的几大特性，人们期待中的重大突破不可能在近期内到来。

## 是耶非耶

在撰写《人月神话》的回顾和更新过程中，作者发现初版中断言的观点甚少被软件工程研究和实践所抨击、证实或证伪，因此在本章中作者提炼了初版中十五个章节中的概要，结合近年来软件技术的发展状况，对这些观点进行强调、修正和反思。

## 人月神话二十年

在《人月神话》初版发布二十周年后，计算机技术领域已有很大变化，《人月神话》体现出深远的影响力，初版中的许多观点依然经常被人们谈论和引用，其中有些断言至今仍被软件开放人员奉为圭臬。作者结合当前软件工程领域的发展现状重新梳理了初版中的各核心观点，强调了概念完整性，重新评议了第二个系统效应，反省了瀑布模型的局限性，结合初版中的观点，作者评述了图形桌面系统、信息隐藏、面向对象高级语言等技术的发展，以及近年来软件工程领域的重要成果。