　　《人月神话 (The Mythical Man-Month)》是软件行业中的一部里程碑式的名著。作者Brooks曾任IBM System/360项目经理，被称为"S/360之父"。在书中，他详细讨论了包括工期规划、团队组成、文档、排错等软件项目进行全程中的方方面面。可以说是软件从业者的必读书目。

  不再说了，最后，说一说没有银弹，不从技术层面考虑，想想软件工程做的事情的覆盖面就知道真的没有银弹，人们常常将软件开发同汽车制造进行类比，个人觉得 这个类比的不太恰当，现在的软件开发设计到各个行业，生活中的各个方面，就像实体制造业一样，这样的类比就如同将实体制造业同车载系统的开发进行类比。 汽车的生产就相当于一款软件的复制，实际效果就是这样，只是汽车的生产有实物的消耗，软件的复制只消耗电量而已。  神话的作者在20年前断言没有任何技术或管理上的进展，能够独立地许诺十年内使生产率、可靠性或简洁性获得数量级上的进步。在之后出现了一些技术比如面 向对象，基于构建的开发，确实使软件开发的过程得到简化，但仍没有人没期望的那么快，人工智能是一个可能的解决方案，但是从七十年代至今，仍没有出现真正 意义上的人工智能，我们只能期待出现一种新的技术或者管理方法使得软件开发的效率得到提高，但这种提高绝对不是像银弹之于人狼一样有效。我觉得，如果未来 的某一天真的有银弹的话，那就是人工智能程序员。但是这个需要做的还有很多，首先是需要机器有像人一样的理解能力，能够理解所要解决的问题的本质，然后是 让其掌握一门语言（机器语言除外），懂得算法的思想，再有项目的经验，这对于一个天生就有人工智能的人来说都不容易，何况是机器。

 Chap.3 外科手术队伍

程序员在生产率上甚至可以达到数量级的差异；专业的、分工良好的小规模团队的生产率更高，在这种架构下，决策的集中性也保证了概念的一致性。

对于小型队伍，最好是选取少数能干和有开发经验的程序员而不是更多的能力一般的程 序员，但是对于大型项目，小型精干队伍的开发速度又显得太慢了，若采用一拥而上的办法，仍然不可行。对此，神话的作者拿出了Mills的建议，即大型项目 的每一个部分由一个团队解决，但是该队伍以类似外科手术的方式组建，而并非一拥而上。并且，神话中将软件开发团队成员及其职责类比到外科手术队伍，并给出 了运作和团队扩建的意见。

Chap.10 提纲挈领

软件项目经理应该编写一系列关键的文档，它们反映了对目标、技术特征、进度、预算和组织结构的管理；它们不仅仅是检查列表和控制手段，它也是沟通渠道和汇报材料。

在提纲掣领中，神话的作者又谈到了文档的重要性，既是实在现在Scrum开发过程中，依然强调文档是非常重要的，虽然不是最重要的。并给出 计算机产品的文档、计算机科学系的文档和软件项目文档应该包含的内容。之后讨论了为什么要有文档，首先，书面记录决策是必要的；第二，文档能够作为同其他 人的沟通渠道；最后，项目经理的文档可以作为数据基础和检查列表。

Chap.15 另外一面

用户需要文档来帮助他们使用（目的、环境、IO范围、功能算法、指令、选项、运行时间和精度检验）、验证和修改程序；流程图的使用与维护都很麻烦，已经过时；自文档化技术是一种非常有效的解决方案。

Chap.4 贵族专制、民主政治和系统设计

概念的完整性最应该被重视，它带来产品的简洁和易懂；因此，系统的体系结构设计应由少数人来完成，即系统设计师专制；但系统设计师应该严守底线，避免干涉外部特征之外的实现细节。

神话的作者十分强调软件设计过程中概念的一致性，提出为了获得概念完整性，设计必须由一个人或者具有共识的小型团队来完成，这样 就可以确保概念的一致性和完整性，作者似乎没有过多的考虑概念的优化和改进，而是尽量避免这些，即宁可省略一些不规则的特性和改进，也不提倡独立和无法整 合的系统，哪怕它们其实包含着许多很好的设计，

Chap.8 胸有成竹

本章将对生产率的估计，工作量 = 常数×指令数目1.5，本章还总结了一些软件工程中一些对生产率进行估计的技术。

在胸有成竹一章中，神话的作者讨论了软件开发工作量估计的问题，并且列出了一些人的实验和结论。在规模控制中，神话的作者认为规模控制既是管理工作的一 部分又是技术工作的一部分，仅对核心程序设定规模目标是不够的，必须把所有的方面都编入预算。

Chap.11 未雨绸缪

第一次开发的系统应该准备好要抛弃，因为变化不可避免，要对此计划好系统与组织架构的变化；缺陷修复会引入新的BUG，而且到了最后必然会不能再进行改进。

未雨绸缪中，神话的作者认为第一个开发的测试系统往往是不能用的，在开发第一个系统的同 时，就需要着手第二个系统的设计，程序维护中的一个基本问题就是缺陷修复总会以20%到50%的概率引入新的bug，所以现实的系统不可能永远可用，基于 原有系统的重新设计是完全有必要的，这就是未雨绸缪。

Chap.14 祸起萧墙

里程碑应该明确定义以防止产生隐瞒；使用PERT图指示对进取的需要；老板不应与一线经理产生角色冲突，而应使用能了解状态真相的评审机制，计划和控制小组在这一过程中很有价值。

Chap.7 为什么巴别塔会失败

巴别塔有着清晰的目标、充足的人力、材料、技术和时间，但是巴别塔的建造人员缺乏沟通；大型软件工程应有不同层次的沟通，并用项目工作手册规定定义接口与划分以减少交流所需的工作量；软件工程的组织结构应采取树形结构，以保证有效的沟通。

对于大型项目的完成，管理的作用绝对是最大的，在这一章中神话的作者给出的大型 项目中组织和交流的意见，对于交流，是需要靠会议和工作手册来实现，而项目工作手册不是独立的一篇文档，它是对项目必须产出的一系列文档进行组织的一种结 构，项目所有的文档都必须是该结构的一部分，这包括目的、外部规格说明、接口说明、技术标准、内部说明和管理备忘录。而团队组织的目标是为了减少必要的交 流和协作量，具体方法是人力划分和限定职责范围。对于大型团队，技术主管和产品负责人不能是同一个人，要明确各自的职责，技术主管负责技术，产品负责人负 责对外的交流和沟通。

Chap.5 画蛇添足

在第一版设计中，设计师往往能保证精炼简洁；但在第二版设计时，画蛇添足是常见的问题，设计师容易被诱惑着开发过多的功能，这是应该被避免的。

第二个系统是人们所设计的最危 险的系统，通常的倾向是过分地进行设计，

Chap.9 削足适履

程序空间也是与时间一样的成本，程序规模应该为预算所控制；空间与时间可以进行一定的折中，但好的数据处理方式是可以在两方面同时进行优化的。

Chap.6 贯彻执行

对产品的文档化规格说明是必要的，它应该用清楚的形式化定义表达；开发人员之间应该定期有不同层次（大会、组例会、电话）的交流，并对交流进行记录、整理和思考；开发人员应坚守手册，尤其在有多重实现时；大型项目中的测试小组对确保用户体验十分重要。

Chap1. 焦油坑

编程系统产品的成本数倍于编程本身；编程这样一个行业给人以创造、动手与控制的乐趣，但是，对沟通的依赖、对完美的追求也是编程所令人苦恼的一面。

Chap.16 没有银弹

软件开发所不可规避的困难在于复杂度、一致性、可变性和不可见性；尽管高级语言、分时系统、IDE在编码过程中帮了程序员的大忙，但他们无助于解决内在困难；高级编程语言、面向对象技术、人工智能、专家系统、自动编程、图形化编程和更快的工作站都无助于问题的根本部分；有希望缓解根本问题的是：购买软件、快速原型技术、增量开发和卓越的设计人员。

Chap.2 人月神话

“一切都将运转良好”在软件工程中是不适用的；完成工作的人数与时间是不能进行简单的互换的，因为沟通需要额外的成本；Brook法则：向进度落后的项目中添加人手，只会使进度更加落后。

Chap.12 干将莫邪

工具的安排：软件将最终在之上运行的目标机和开发辅助机，目标机应该被规划使用，辅助机上应该运行可靠的仿真器、解释器和程序库文档。

Chap.13 整体部分

概念完整性与产品精确定义是关键的；结构化编程是必要的；构件应该进行单独的测试，之后再进行集成测试；修改时应控制变更规模。

 最后，总结一下，《人月神话》强调组件一个像外科手术队伍一样的开发团队，强调文档的重要性，瀑布模型是没有用的，增量和迭代开发才是重要的。《人月神 话》就像是一个方法论，即使现在编程工具和开发环境如何变化，大型项目开发过程中的基本原理是不会变的，就像几千年前古人写的富含哲理的文章，现在读起来 依然觉得很有道理，虽然现在生活方式和那个时代的已经是天差地别。《人月神话》的作者在书中说了很多的IBM 的OS/360是一个失败的系统，但是神话中的大部分内容是基于这个失败的系统总结过来的，在我们平时的小项目中，我们是不是也能把失败的原因抽象出来 呢？