人月神话读书笔记

本书要点：

人月神话作为软件开发方面的经典著作之一，主要针对了软件开发中项目管理方面的内容，书中大部分内容涉及团队，人和沟通。在各章节内容中，作者为了论证自身观点，列举了许多实际的项目作为证据，对于观点的说服力和易理解程度有很大的加强。此外，作者充分表达了对文档的重要性的认可，对于传统的大型软件工程项目的强调人的重要性，以及肯定了开发工作是一种高智力的脑力工作。

精编书摘：

焦油坑

编程系统产品（Programming Systems Product）开发的工作量是供个人使用的、独立开发的构件程序的九倍。
编程的乐趣：

1. 创建事物的快乐
2. 开发对其他人有用的东西的乐趣
3. 将可以活动、相互啮合的零部件组装成类似迷宫的东西，这个过程所体现出令人 神魂颠倒的魅力
4. 面对不重复的任务，不间断学习的乐趣
5. 工作在如此易于驾驭的介质上的乐趣——纯粹的思维活动，其存在、移动和运转方式完全不同于实际物体。

编程行业内在固有的苦恼：

1. 将做事方式调整到追求完美，是学习编程的最困难部分。
2. 由其他人来设定目标，并且必须依靠自己无法控制的事物。
3. 真正的权威来自于每次任务的完成。
4. 任何创造性活动都伴随着枯燥艰苦的劳动，编程也不例外
5. 人们通常期望项目在接近结束时（bug、工作时间）能收敛得快一些，然而软件项目的情况却是越接近完成，收敛得越慢。
6. 产品在即将完成时总面临着陈旧过时的威胁。

人月神话

围绕成本核算的估计技术，混淆了工作量和项目进展。人月是危险和带有欺骗 性的神话，因为它暗示人员数量和时间是可以相互替换的。
关于软件任务的进度安排，作者的经验法则是：1/3 计划、1/6 编码、1/4 构件测

试以及 1/4 系统测试。
作为一门学科，软件工程缺乏数据估计。
Brooks 法则：“向进度落后的项目中增加人手，只会使进度更加落后。”
向软件项目中增派人手从三个方面增加了项目必要的总体工作量：1、任务重新分配本身和所造成的工作中断；2、培训新人员；3、额外的相互沟通。

外科手术队伍

同样有两年经验而且在受到同样的培训的情况下，优秀的专业程序员的工作效率是较差程序员的十倍。
小型、精干队伍是最好的。
两个人的团队，其中一个项目经理，常常是最佳的人员使用方法。
对于真正意义上的大型系统，小型精干的队伍太慢了。
实际上，绝大多数大型编程系统的经验显示出，一拥而上的开发方法是高成本、速度缓慢、不充分的，开发出的产品无法进行概念上的集成。
一位首席程序员、类似于外科手术队伍的团队架构提供了一种方法，既能获得由少数头脑产生的产品完整性，又能得到多位协助人员的总体生产率，还彻底地减少了沟通的工作量

贵族专制、民主政治和系统设计

概念完整性是系统设计中最重要的考虑因素。
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。
对于非常大型的项目，将设计方法、体系结构方面的工作与具体实现相分离是获得概念完整性的强有力方法。
纪律、规则对行业是有益的。外部的体系结构规定实际上是增强，而不是限制实现小组的创造性。
体系结构、设计实现、物理实现的许多工作可以并发进行。

画蛇添足

尽早交流和持续沟通能使结构师有较好的成本意识，以及使开发人员获得对设计的信心，并且不会混淆各自的责任分工。
结构师如何成功地影响实现：
牢记是开发人员承担创造性的实现责任；结构师只能提出建议。
听取开发人员在体系结构上改进的建议。
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。关于这一点也许是正确的，但是这是一个回避不了的问题，如果没有开发第二个系统经验的人，就不可能有开发第三个系统经验的人了。

贯彻执行

即使是大型的设计团队，设计结果也必须由一个或两个人来完成，以确保这些决定是一致的。
必须明确定义体系结构中与先前定义不同的地方，重新定义的详细程度应该与原先的说明一致。
出于精确性的考虑，我们需要形式化的设计定义，同样，我们需要记叙性定义来加深理解。
允许体系结构师对实现人员的询问做出电话应答解释是非常重要的，并且必须进行日志记录和整理发布。
项目经理最好的朋友就是他每天要面对的敌人——独立的产品测试机构/小组。

为什么巴比伦塔会失败？

巴比伦塔项目的失败是因为缺乏交流以及组织。
因为左手不知道右手在做什么，从而进度灾难、功能的不合理和系统缺陷纷纷出现。由于对其他人的各种假设，团队成员之间的理解开始出现偏差。
团队应该以尽可能多的方式进行相互之间的交流：非正式、常规项目会议，会上进行简要的技术陈述、共享的正式项目工作手册。

胸有成竹

仅仅通过对编码部分的估计，然后乘以任务其他部分的相对系数，是无法得出对整项工作的精确估计的。
构建独立小型程序的数据不适用于编程系统项目。
程序开发与程序规模成指数增长趋势。
当使用适当的高级语言时，程序编制的生产率可以提高 5 倍。

削足适履

这一章主要是要解决项目程序与计算机磁盘空间和内存之间的矛盾，然而对于如今计算机硬件的发展已经不是一个主要问题了。

提纲挈领

软件项目的文档要求：目标、用户手册、内部文档、进度、预算、组织机构图和工作空间分配。
即使是小型项目，项目经理也应该在项目早期规范化上述的一系列文档。
本章强调文档重要性，但并没有单纯的赘述文档的重要性和必要性的大道理，而是充分说明了文档在项目中起到的明确而关键的作用。

未雨绸缪

对于大多数项目，第一个开发的系统并不合用。它可能太慢、太大，而且难以使用，或者三者兼而有之。系统的丢弃和重新设计可以一步完成，也可以一块块地实现。这是个必须完成的步骤，如果将开发的第一个系统丢弃原型发布给用户，可以获得时间，但是它的代价很高。对于用户，使用极度痛苦；对于重新开发的人员，分散了精力；对于产品，影响了声誉，即使最好的再设计也难以挽回名声。
用户的实际需要和用户感觉会随着程序的构建、测试和使用而变化。
软件产品易于掌握的特性和不可见性，导致了它的构建人员面临着永恒的需求变更。
目标和开发策略上的一些正常变化无可避免，事先为它们做准备总比假设它们不会出现要好得多。
对于一个广泛使用的程序，其维护总成本通常是开发成本的 40％或更多。
维护成本受用户数目的严重影响。用户越多，所发现的错误也越多。
Campbell 指出了一个显示产品生命期中每月 bug 数的有趣曲线，它先是下降，然后攀升。
缺陷修复总会以（20－50）%的机率引入新的 bug。
在每次修复之后，必须重新运行先前所有的测试用例，从而确保系统不会以更隐蔽的方式被破坏。
同样，设计实现的人员越少、接口越少，产生的错误也就越少。
所有修改都倾向于破坏系统的架构，增加了系统的混乱程度。即使是最熟练的软件维护工作，也只是放缓了系统退化到不可修复混乱的进程

干将莫邪

项目经理应该制订一套策略，以及为通用工具的开发分配资源，与此同时，他还必须意识到专业工具的需求。

整体部分

系统的构件单元调试和集成调试

祸起萧墙

一天一天的进度落后比起重大灾难，更难以识别，更不容易防范和更加难以弥补。
根据一个严格的进度表来控制项目的第一个步骤是制订进度表，进度表由里程碑和日期组成。
里程碑必须是具体的、特定的、可度量的事件，能进行清晰能定义。
如果里程碑定义得非常明确，以致于无法自欺欺人时，很少有人会就里程碑的进展弄虚作假。
并不是每一天的滞后都等于灾难。

另外一面

对于软件编程产品来说，程序向用户所呈现的面貌与提供给机器识别的内容同样重要。
即使对于完全开发给自己使用的程序，描述性文字也是必须的，因为它们会被用户和作者所遗忘。
文档能在整个软件开发的生命周期对程序员克服懒惰和进度的压力起促进激励作用，但向编程人员成功地灌输对待文档的积极态度是一件困难的事情。
为了使文档易于维护，将它们合并至源程序是至关重要的，而不是作为独立文档进行保存。

没有银弹

作者将软件开发比作人狼，而将提高软件开发效率的方法比作银弹。作者预言未来十年，想要试图通过寻找一种有效地银弹将软件开发效率提高一个甚至几个数量级，这种银弹不可能出现。
文章中作者列举出了当时一些非常先进的技术或思想理念，例如 Ada 和其他高级编程语言、面向对象编程、人工智能、专家系统、“自动”编程、图形化编程、程序验证、环境和工具、工作站等。虽然这些先进技术在一定程度上提高了软件开发的效率，但是始终没有达到银弹的效果。
针对概念上根本问题的颇具前途的方法：1、购买和自行开发；2、需求精炼和快速原型；3、增量开发；4、卓越的设计人员。

再论没有银弹

Brooks 在此处继续解释了他的各种观点，以及再次讨论并分析了很多存在的潜在银弹。
现在，有可能，我们可以在软件生产率上取得逐步的进展，而不是等待不大可能到来的突破。

二十年后的人月神话

20 年后的人月神话有些结论得到验证，有些情况已经变化：

- 第二系统定律得到验证：开发第二个系统总是因为盲目的功能导致易用性、甚至是可用性的灾难。
- 图形界面的成功
- 瀑布模型被证明是错的了，因为没有构建舍弃原型。事实上增量开发与快速迭代才是理想的开发方式。
- 增量开发和快速原型，渐进地精华，让软件像生物进化那样逐渐演化成更为复杂的结构，演化出更多的功能。
- 关于信息隐藏，Parnas 是正确的，我是错误的。20 年前关于信息隐藏的两大

观念，其一是 Brooks 主张的，所有的程序员应该了解所有的材料。而 Parnas 则认为代码模块应该采用定义良好的接口来封装，这些模块内部结构应该是程序员的私有财产。Brooks 承认，Parnas 所主张的方案确实更符合实际。

- 对人月神话实际研究发现，向进度落后的项目中添加人手会增加项目的成本，但是不一定会使项目更加落后。如果在项目早期添加额外的人比在后期添加额外的人更安全些。
- 人就是一切。这一点可以从《人件：高生产率的项目和团队》可以见出。
- 放弃权利的力量——公司通过将权利下放到具体的团队，事实上使得组织机构变得更加“融洽和繁荣”。
- 最令人惊讶的新事物——数百万的计算机
- 使用塑料包装的成品软件包作为构建：成熟的模块和对象组合提升了软件复用的层次。

软件工程的未来

软件工程的焦油坑在将来很长一段时间内会继续地使人们举步维艰，无法自拔。软件系统可能是人类创造中最错综复杂的事物，只能期待人们在力所能及的或者刚刚超越力所能及的范围内进行探索和尝试。

由于软件系统多变性和复杂性，这个行业只能是一步一个阶梯的缓慢上升前进，而出现银弹的机会在我们当前的预见范围内是非常渺茫的，我们将长期与焦油坑作斗争。