## 贯彻执行

确保每个人听从、理解并实现结构师的决策。让每个结构师的小组保持系统概念上的完整性。

1. 文档化的规格说明——手册

手册：产品的外部规格说明，描述了用户所见的每一个细节；是结构师主要的工作产物。

规格说明不断被重复准备和修改。修改应是阶段化的。

描述外部可见的一切；避免描述用户看不见的事物，不能限制实现人员。

风格清晰、完整、明确。

1. 形式化定义

形式化定义缺点是不易理解，但它是精确的。需要记叙性语言的辅助。规格说明包括形式化和记叙性定义两种方式。

设计实现可以作为一种形式化定义的方法。

使用实现作为形式化定义易引起混淆，特别是在程序化的仿真中。当实现充当标准时，必须防止对实现的任何修改。

1. 直接整合

是设计被传递参数和共享存储器的声明。对于建立模块间接口语法而非语义时特别有用。

1. 会议和大会

两个级别，周例会和年度大会。周例会由所有结构师加硬件和软件实现人员代表和市场计划人员参加，由首席系统结构师主持。

为解决在周例会中未解决的堆积问题，举办年度大会

1. 多重实现

存在多重实现时，遵从手册更新机器造成的延迟和成本消耗，比根据机器调整手册要低。

1. 电话日志

结构理解和解释方面的问题，需要文字澄清和解释。实现人员打电话与结构师沟通。

1. 产品测试

独立的产品测试机构/小组，根据规格说明检查机器和程序。每个开发机构需要一个独立的技术监督部门，来保证其公正性。

## 为什么巴比伦塔会失败

1. 巴比伦塔的管理教训

缺乏交流和组织，无法合作。

1. 大型编程项目中的交流
2. 非正式途径
3. 会议
4. 工作手册
5. 项目工作手册
6. 是什么：项目工作手册是对项目必须产出的一系列文档进行组织的一种结构。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。
7. 为什么：查看相关用户手册，可以发现不仅思路，还能追溯到最早备忘录的文字和章节。

第二个原因是控制信息发布，以确保信息能到达所有需要它的人。

1. 处理机制：随着项目规模扩大，技术备忘录的问题以非线性趋势增长。

工作手册的实时更新是非常关键的。对于大型项目，建议把微缩胶片作为文字工作手册的补充。

1. 现在如何入手：采用可以直接访问的文件。在文件中，记录修订日期记录和标记变更标识条。每个用户可以从一个显示终端（打印机太慢了）来查阅。卡内基－梅隆大学的D.L.Parnas提出了更彻底的解决方法。他认为，编程人员仅了解自己负责的部分，而不是整个系统的开发细节时，工作效率最高。这种方法的先决条件是精确和完整地定义所有接口。
2. 大型编程项目的组织架构

如果项目有n个工作人员，则有（n^2 - n）/ 2个相互交流的接口，有将近2^n个必须合作的潜在团队。团队组织的目的是减少不必要交流和合作的数量，因此良好的团队组织是解决上述交流问题的关键措施。减少交流的方法是人力划分（division of labor）和限定职责范围（specialization of function）。

树状编程队伍，每棵子树所必须具备的基本要素：

1. 任务
2. 产品负责人：组建团队，划分工作及制订进度表。主要的工作是与团队外部，向上和水平地沟通。
3. 技术主管和结构师：对设计进行构思，识别系统的子部分，指明从外部看上去的样子，勾画它的内部结构。他提供整个设计的一致性和概念完整性；他控制系统的复杂程度。当某个技术问题出现时，他提供问题的解决方案，或者根据需要调整系统设计。
4. 进度
5. 人力的划分
6. 各部分接口之间的接口定义

## 胸有成竹

工作量是规模的幂函数：工作量=(常数)\*（指令的数量）^1.5



1. Portman的数据

项目估算对每个人年的技术工作时间数量做出了不现实的假设。

1. Aron的数据

非常少的交互 10,000指令每人年

少量的交互 5,000

较多的交互 1,500

1. Harr的数据

John Harr，Bell电话实验室电子交换系统领域的编程经理，在1969年春季联合计算机会议8的论文中，汇报了他和其他人的经验。这些数据如图8.2、8.3和8.4所示。

这些图中，图8.2是最数据详细和最有用的。头两个任务是基本的控制程序，后两个是基本的语言翻译。生产率以经调试的指令/人年来表达。它包括了编程、构件测试和系统测试。没有包括计划、硬件机器支持、文书工作等类似活动的工作量。







1. OS/360的数据

Aron、Harr和OS/360的数据都证实，生产率会根据任务本身复杂度和困难程度表现出显著差异。在复杂程度估计这片“沼泽”上的指导原则是：编译器的复杂度是批处理程序的三倍，操作系统复杂度是编译器的三倍。

1. Corbato的数据

Harr和OS/360的数据都是关于汇编语言编程的，好像使用高级语言系统编程的数据公布得很少。Corbato的MIT项目MAC报告表示在MULTICS系统上，平均生产率是1200行经调试的PL/I语句（大约在1和2百万指令之间）/人年。

两个重要结论：

1. 对常用编程语句而言。生产率似乎是固定的。这个固定的生产率包括了编程中需要注释，并可能存在错误的情况.
2. 使用适当的高级语言，编程的生产率可以提高5倍。

## 削足适履

1. 作为成本的程序空间

由于规模是软件系统产品用户成本中如此大的一个组成部分，开发人员必须设置规模的目标，控制规模，考虑减小规模的方法，就像硬件开发人员会设立元器件数量目标，控制元器件的数量，想出一些减少零件的方法。同任何开销一样，规模本身不是坏事，但不必要的规模是不可取的。

1. 规模控制

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。

三个道理：

1. 和制订驻留空间预算一样，应该制订总体规模的预算；和制订规模预算一样，应该制订后台存储访问的预算。
2. 在指明模块有多大的同时，确切定义模块的功能。
3. 在整个实现的过程期间，系统结构师必须保持持续的警觉，确保连贯的系统完整性。在这种监督机制之外，是实现人员自身的态度问题。培养开发人员从系统整体出发、面向用户的态度是软件编程管理人员最重要的职能。
4. 空间技能
5. 在速度保持不变的情况下，更多的功能意味着需要更多的空间。所以，其中的一个技巧是用功能交换尺寸。
6. 空间－时间的折衷。项目经理可以做两件事来帮助他的团队取得良好的空间－时间折衷。一是确保他们在编程技能上得到培训，而不仅仅是依赖他们自己掌握的知识和先前的经验。另外一种方法是认识到编程需要技术积累，需要开发很多公共单元构件。
7. 数据的表现形式是编程的根本

技艺改进的结果往往是战略上的突破，而不仅仅是技巧上的提高。这种战略上突破有时是一种新的算法.

更普遍的是，战略上突破常来自数据或表的重新表达——这是程序的核心所在。

## 提纲挈领

1. 计算机产品的文档
2. 目标
3. 技术说明
4. 进度，时间表
5. 预算
6. 组织机构图
7. 工作空间的分配
8. 报价、预测、价格
9. 软件项目的文档
10. 做什么：目标。定义了待完成的目标、迫切需要的资源、约束和优先级。
11. 做什么：产品技术说明。以建议书开始，以用户手册和内部文档结束。速度和空间说明是关键的部分。
12. 时间：进度表
13. 资金：预算
14. 地点：工作空间分配
15. 人员：组织图
16. 为什么要有正式的文档
17. 只有记录下来，分歧才会明朗，矛盾才会突出。书写这项活动需要上百次的细小决定，正是由于它们的存在，人们才能从令人迷惑的现象中得到清晰、确定的策略。
18. 文档能够作为同其他人的沟通渠道。项目经理常常会不断发现，许多理应被普遍认同的策略，完全不为团队的一些成员所知。正因为项目经理的基本职责是使每个人都向着相同的方向前进，所以他的主要工作是沟通，而不是做出决定。这些文档能极大地减轻他的负担。
19. 项目经理的文档可以作为数据基础和检查列表。通过周期性的回顾，他能清楚项目所处的状态，以及哪些需要重点进行更改和调整。