软件工程

Software Engineering

——实践者的研究方法

课程教学目标

- 掌握系统的软件开发理论、技术和方法，学会使用正确的工程化方法开发出成本低、可靠性好、能高效运行的软件。
  - 转变对软件的认识：程序 —上升→ 系统
  - 转变思维定式：程序员 —上升→ 系统工程师（系统分析员）
  - 系统分析员的地位：

  - 工程化训练。

与软件编程课的区别

- 其他编程类课程：具体某种编程语言（如C/C++、JAVA等）、专业领域（如网络、嵌入式等）。
- 《软件工程》课程：
  1. 立足于系统的整体。
  2. 讲授获取需求、系统分析、系统设计、测试及维护，以及项目管理的理论和方法。
  3. 构筑一个软件系统，实践软件开发全过程。
  4. 较之技术，更偏重于管理。
- 软件开发知识的半衰期为3年，但软件工程的知识可在整个职业生涯内提供服务。

课程内容

1. 概述：软件和软件工程
2. 软件过程
3. 需求工程：需求分析与建模
4. 设计：体系结构、构件、用户界面等
5. 软件测试策略和测试技术
6. 项目管理：估算与度量、风险管理等
7. 质量管理：评审技术、质量保证、配置管理等

考核方式

- 期末卷面考试
- 作业
- 工程化训练：
  - 组建团队：5~6人一组，拥有组名和组长
  - 项目选题：
    - 订餐系统
    - 闲置书籍流通管理系统
    - 航班订票系统
    - 自拟题目

能成功吗？

软件项目成功的三个主要因素：

- 用户的参与
- 主管层的支持
- 需求的清晰表述

软件人成功的三个主要因素：

- 需求的把握与控制能力
- 良好的沟通能力
- 责任心与承担责任的能力

本周任务

1. 确定分组名单
2. 召开项目启动会议，确定如下内容：
   - 组名
   - 组长人选
   - 做什么项目？
3. 提交：组名、组长、名单(学号&姓名)、项目题目

- 备注：
  - 应于两周内确定，此后不可更改。
  - 自拟题目应给出一份简要的项目说明。
  - 团队成员上课时请坐在一起，方便讨论。

第一章
软件和软件工程

本章内容

- 软件的定义、特性、分类。
- 软件危机！
- 软件工程的含义。
- 容易误导人们的那些软件神话.....

一个笑话

- 比尔·盖茨在一次展览会的演讲中谈到：假如GM的技术能像计算机技术那样发展，我们现在应该能用25美元买到一辆1加仑油跑1000英里的汽车。
- GM反唇相讥：如果GM的技术像微软那样发展，我们现在开的汽车会有以下的特点：
  1. 你的汽车可能毫无道理的每天抛两次锚。
  2. 每次公路上重新画线时，你都得买辆新车。
  3. 有时候你的车在高速公路上莫名其妙的熄火，你必须accept，然后restart。

4. 当你买了“轿车95”或者“轿车NT”后，每次车上只能坐一个人，你要给其他人再买椅子。
5. 油量、水温和其他警告牌等将由一个“general car failure”的警告灯所代替。
6. 新坐椅要求大家屁股的尺寸相同。
7. 气囊系统弹出前将询问“are you sure?”，要求你加以确认。
8. 有时汽车会锁死车门使你无法进入汽车，你得不停地提门把手，拿钥匙捅，晃天线，直到打开车门。

软件？问题多多！

软件缺陷造成的损失

Therac-25 放射线治疗仪

- **1985-87** 年
  - 过量辐射 (6 个死亡 / 截肢)!

Mars Pathfinder（火星探路者）

- **1997** 年
  - 周期性重新启动 (在火星上)!

汽车导航控制系统

- **1986** [Grady Booch]
  - 点火后加速 (撞车)!

软件缺陷造成的损失(续)

航空器控制系统

- 功能错误 (飞机撞毁)!

核电站控制系统

- 堆芯熔化 (大灾难)!

移动电话

- 2000 年以后 -…
  - 死机 (烦死人)!

第一节
软件概述

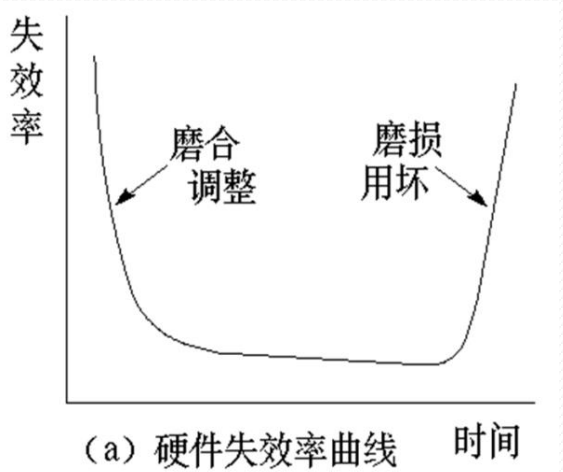
软件的定义

- 软件是计算机系统中与硬件相互依存的另一部分，它包括：

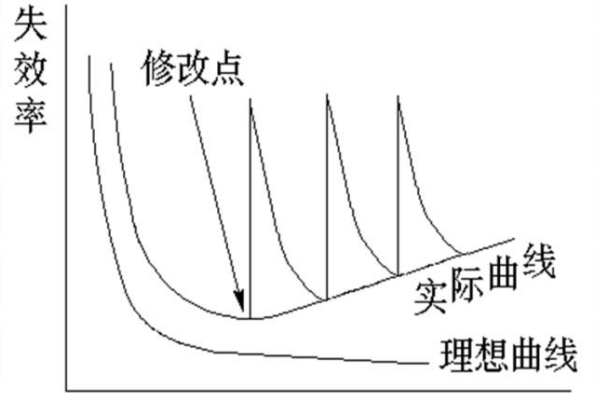
1. 指令的集合(计算机程序)：通过执行这些指令，可以满足预期的特性、功能和性能需求。
2. 数据结构：使得程序可以合理利用信息。
3. 软件描述信息(文档)：以硬拷贝和虚拟形式存在，用来描述程序的开发、维护、操作和使用。

软件的特性

1. 软件是逻辑产品，没有明显的物理形态，它是设计开发的，而不是传统意义上生产制造的。
2. 软件在运行和使用期间，没有硬件那样的机械磨损和老化问题，但存在软件退化问题 。
3. 虽然软件产业正在向基于构件的组装前进，但大多数软件仍是定制式的手工生产方式。

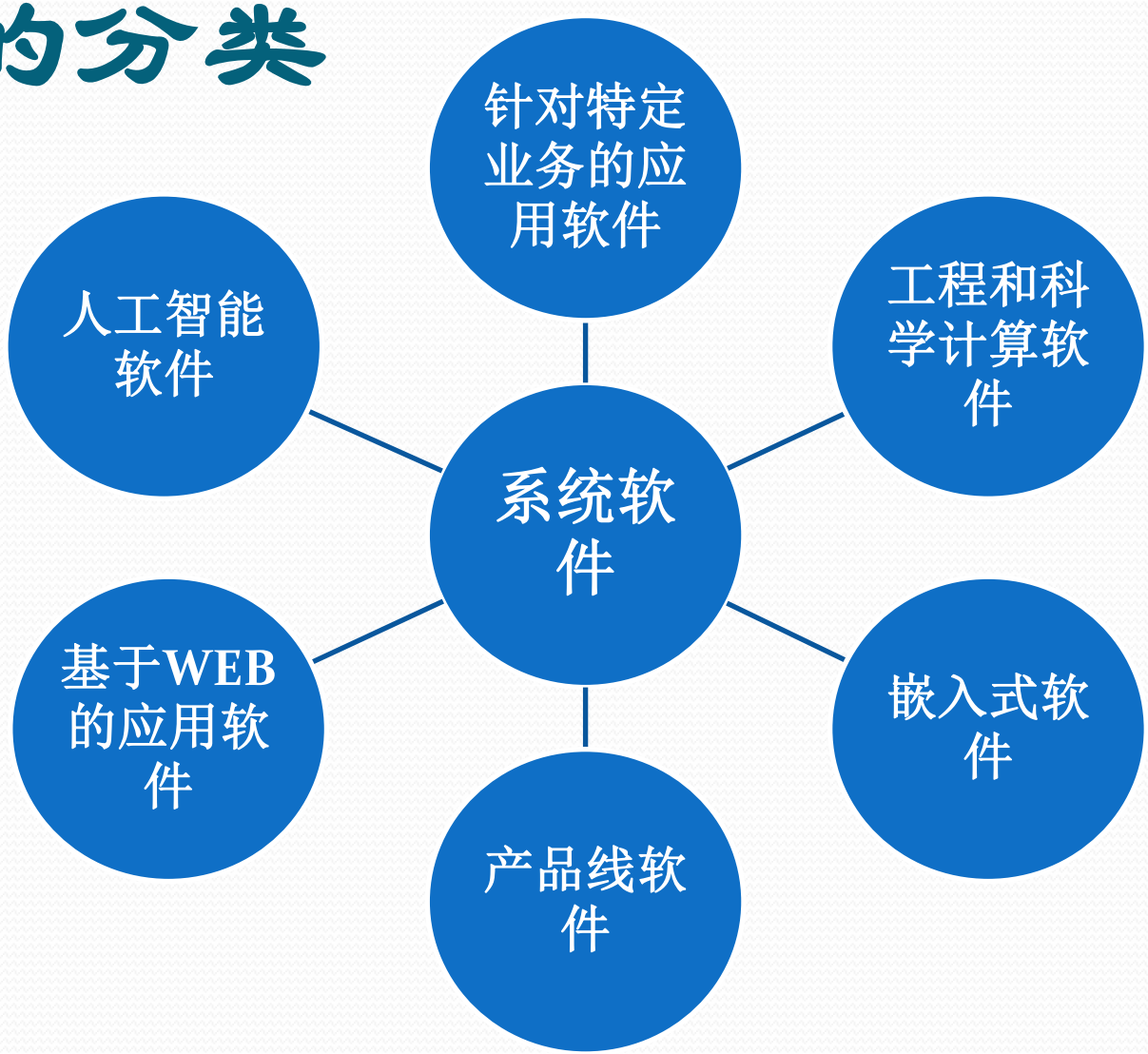


（a）硬件失效率曲线



（b）软件失效率曲线

软件的分类


针对特定业务的应用软件
人工智能软件
工程和科学计算软件
系统软件
基于WEB的应用软件
嵌入式软件
产品线软件

软件的规模

软件规模类别	参加人员数	开发期限	产品规模(源代码行数)
微型	1	1～4周	500行
小型	1	1～6月	1000～2000行
中型	2～5	1～2年	5000～5万行
大型	5～20	2～3年	5万～10万行
甚大型	100～1000	4～5年	10万～100万行
极大型	2000～5000	5～10年	100万～1000万行

几个实例

系统名称	规模
水星计划系统(1963年)	200万条指令
双子星座计划系统(1967年)	400万条指令
阿波罗计划系统(1973年)	1000万条指令
哥伦比亚航天飞机系统(1979年)	4000万条指令
Windows95	1000万行代码
Windows2000	5000万行代码

开发人员结构	Exchange2000	Windows2000
项目经理	25人	约250人
开发人员	140人	约1700人
测试人员	350人	约3200人

软件的发展历程

阶段 / 特点	程序设计 1950 至 60 年代	程序系统 60至70年代	（现代）软件工程 70年代以后
软件所指	程序	程序及说明书	程序、文档和数据
程序设计语言	汇编及机器语言	高级语言	软件语言
软件工作范围	程序编写	包括设计和测试	软件生存期
需求者	程序设计本人	少数用户	市场用户
开发软件的组织	个人	开发小组	开发小组及大中型软件开发机构
软件规模	小型	中小型	大中小型
决定质量的因素	个人程序技术	小组技术水平	管理水平
开发技术和手段	子程序/程序库	结构化程序设计	数据库、开发工具、开发环境、工程化开发方法、标准和规范、网络及分布式开发、面向对象技术、软件复用
维护责任者	程序设计者	开发小组	专职维护人员
硬件特征	价格高/存储容量小 工作可靠性差	降价、速度、容量及工作可靠性明显提高	快速向超高速、大容量、微型化及网络化发展
软件特征	完全不受重视	软件技术的发展不能满足需求，出现软件危机	开发技术有进步，但未获突破性进展，价高，未完全摆脱软件危机

软件危机

- 软件工程的概念源自软件危机(Software Crisis)——60年代后期，在软件的开发和维护过程中存在一系列严重问题，如：
  - 成本难以估计：开发成本常严重超标；
  - 无法制定合理的进度计划：开发周期常大大超过规定日期；
  - 用户对完成的软件常不满意：错误多，质量差，不能保证可用性和可靠性；
  - 系统无法增加新功能，难于维护和扩充。
  - 有些系统彻底失败。

一个实例

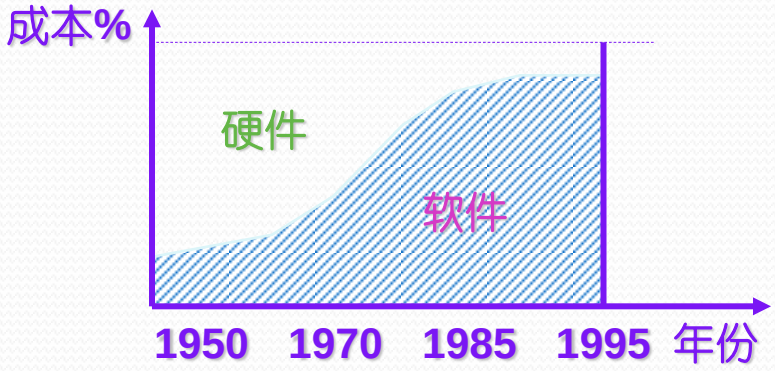
- 60年代，IBM OS/360系统，由4000多个模块组成，约100万条指令，人工为5000人年，耗费数亿美元。
- 该系统投入运行后便发现了2000多个错误，修改后发布新版本，前后共发布了19个版本，每个版本都是修改前一个版本的上千个错误产生的，系统开发陷入了僵局。
- OS/360系统的负责人F. D. Brooks曾这样形象地描述了开发过程中的困难和混乱：“……像一头巨兽在泥潭中作垂死挣扎，挣扎得越猛，泥浆就沾得越多，最终没有一个野兽能逃脱淹没在泥潭中的命运……程序设计就像是这样一个泥潭……一批批程序员在泥潭中挣扎……没人料到问题竟会这样棘手……”。

一份统计数据

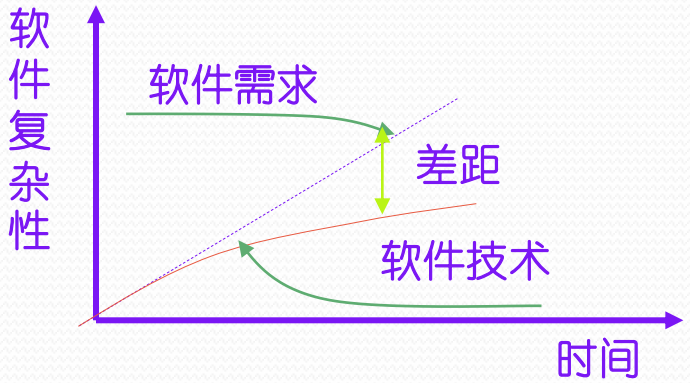
- 1995 年，美国斯坦迪申咨询公司对美国 365 位信息技术高层经理人员所管理的 8380 个项目进行了调查研究，得到如下结论：
  - 信息技术项目正处于一个混沌的状态
  - 平均成功率为16％
  - 50％的项目需要补救
  - 34％的项目彻底失败
  - 平均超出时间为 222%
  - 实际成本是估计成本的 189%
  - 性能与功能只达到要求的61％

软件危机出现的原因

1. 硬件的发展速度一直高于软件；
2. 使用计算机的应用领域和行业不断增加，软件的规模扩大，功能繁多，逻辑复杂，开发者没能充分获取需求信息；
3. 遗留软件缺乏相应文档，不易维护。



硬、软件成本比例的变化



软件技术的发展落后于需求

遗留软件

- 早期的编程是在无序的、崇尚个人技巧的状态中完成的。
  1. 编程作为一门技艺，程序员的素质决定软件质量；
  2. 忽视软件需求分析；
  3. 认为软件开发就是写程序并使之运行，几乎不写相关文档，不重视软件开发管理，也缺少有效方法与软件工具的支持；
  4. 轻视软件维护，软件的维护工作很难进行。
- 遗留软件需要演化的原因：
  1. 修改，以满足新的计算环境或技术需求；
  2. 升级，以实现新的商业需求；
  3. 扩展，使其具有与更多新系统和数据库的互操作能力；
  4. 改建架构，使之能适应多样化的网络环境。

开发者面临的问题

1. 如何快速开发以满足人们对软件日益增长的需求？
2. 如何维护数量不断膨胀的遗留软件？

- 为什么软件开发需要如此长的时间？
- 为什么软件开发成本居高不下？
- 为什么在软件交付顾客前，无法找到所有的错误？
- 为什么软件开发和维护的过程难以度量？
- 为什么维护程序要花费高昂的时间和人力代价？
- 什么样的软件称做好软件？好软件具有什么特点？

软件危机的解决方案

- 统计数据表明，多数软件开发项目的失败，并不是开发技术方面的原因，而是由于不适当的管理造成的。
- 针对软件危机中的问题，人们借鉴在其它领域的经验和知识，认为可采用如下的解决方案：
  1. 推广使用在实践中总结出来的开发软件的成功技术和方法，并研究探索更有效的技术和方法；
  2. 开发和使用更好的软件工具；
  3. 良好的组织管理措施。
- 人们认识到：“摆脱软件危机的出路在于软件开发的标准化和工程化”，即“软件工程”的概念。

第二节
软件工程概述

工程化的方法

- 在近代技术发展历史上，工程学科的进步一直是产业发展的巨大动力。

传统工程 → 水利工程　建筑工程　机械工程　……

新兴工程 → 气象工程　生物工程　软件工程

- 软件工程：应用工程化的方法和技术，研究软件开发与维护的方法、工具和管理。
- 软件工程的目的：经济地开发出高质量的软件并有效地维护它。

软件工程的定义

- 1968年，Fritz Bauer在NATO会议上：建立并使用一套合理的工程原则，以便经济地获得可靠的、可以在实际机器上高效运行的软件。
- 1993年IEEE：软件工程是①将*系统化的、规范的、可量化的方法*应用于软件的开发、运行和维护，即将工程化方法应用于软件；②研究①中提到的方法。
- Fairly：研究在成本限额内按时完成和修改软件产品所需要的系统生产和维护技术，以及管理科学。
- Boehm：运用现代科学技术知识来设计并构造计算机程序及为开发、运行和维护这些程序所必需的相关文件资料。

软件工程项目的基本目标

- 付出较低的开发成本；
- 达到要求的软件功能；
- 取得较好的软件性能；
- 开发的软件易于移植；
- 需要较低的维护费用；
- 能按时完成开发工作，及时交付使用。

软件工程与其它工程的比较

- 每个软件项目都是新的，即新的需求和功能，新的技术实现手段和新的部署方式。
- 每个项目都是在不断变化的，项目进行中会有预想不到的问题出现。
- 软件项目一般胜算不大，或者超预算开支，或者项目延期，或者夭折，或者永远完不成。
- 软件项目通常团队活动，但人多不一定力量大。
- 软件工程还没有工程化、系统化的稳定的方法，还不是真正的工程。
- 软件工程兼有科学和艺术的特点。

软件工程层次图

为过程和方法提供自动化或半自动化的支持。工具可集成，成为计算机辅助软件工程CASE。

为构建软件提供技术上的解决方法（如何做）。

软件工程三要素

基础！能有效实施软件工程技术的流程。

根基！任何工程方法都必须构建在质量承诺的基础上。

软件工程实践的精髓

当我们遇到一个问题，**How to solve it?**

通用的解决步骤：

- 理解问题
- 计划解决方案
- 实施计划
- 检查结果的正确性

软件工程的做法：

- 沟通和分析
- 建模和软件设计
- 代码生成
- 测试和质量保证

软件工程实践的一般原则

1. 存在价值：应能为用户提供价值。
2. 保持简洁：更易于理解和维护。
3. 保持愿景：应保证系统实现始终与愿景一致。
4. 关注使用者：让你的工作产品使用者的工作更简单，包括其他设计人员、程序员、测试员、最终用户等。
5. 面向未来：需求多变，应考虑周全，尽量构建可以解决通用问题的系统。
6. 计划复用
7. 认真思考

第三节
软件神话

管理神话1

- 神话1：我们已经有了一本写满软件开发标准和规程的宝典，它能提供我们所需要了解的所有信息。
- 事实：好的参考书无疑能指导我们的工作。充分利用书中的方法、技术和技巧，可以有效解决软件开发中大量的常见问题。但实践者不能依赖书籍，因为：
  a. 现实的工作中，由于条件千差万别，即使是相当成熟的软件工程规范，常常也无法套用。
  b. 软件技术日新月异，没有哪一种软件标准能长盛不衰。祖传秘方在某些领域很是吃香，而在软件领域则意味着落后。

管理神话2

- 神话2：如果未能按时完成计划，可以通过增加程序员而赶上进度。
- 事实：软件开发并非如机器制造般的机械过程，人多不见得力量大。
  - 新手会产生很多新的错误，使项目变得混乱。
  - 增加新人后，原有的开发者必须牺牲他本来的开发时间去培训新人，从而减少了本应用于高效开发的时间。
- 问题：难道永不能增加人手吗？
  - 不，人手可以增加，但只能是在计划周密、协调良好的情况下，千万不可为了赶进度而增加人手。

管理神话3

- 神话3：如果决定将软件外包给第三方公司，就可以放手不管，完全交给第三方公司开发了。
- 事实：
  - 如果第三方公司的开发团队不了解如何在内部管理和控制软件项目，则该外包项目将总处于挣扎的境地。
  - 如果本公司(发包方)不了解该软件的开发现状、进度等信息，则将影响到本公司与该软件相关的产品研发的总体进度，甚至导致项目失败。

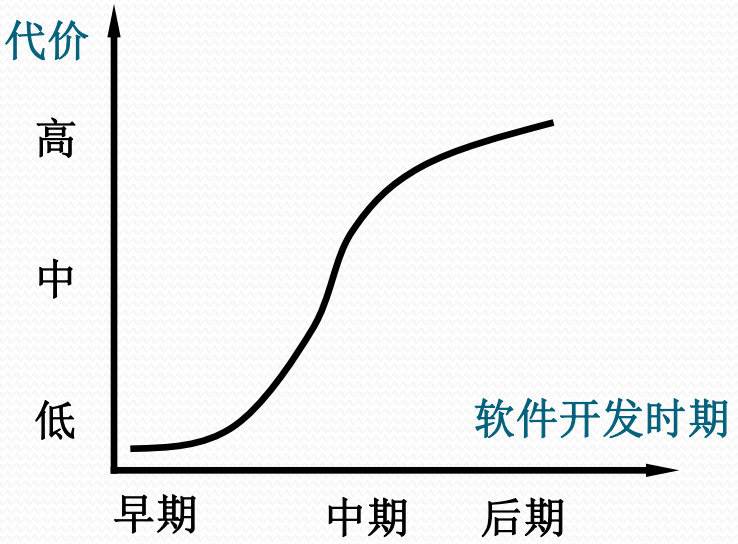
客户神话1

- 神话1：有了对项目目标的大概了解，便足以开始编写程序，可在之后的项目开发过程中逐步充实细节。
- 事实：
  - 虽然通常很难得到综合全面且稳定不变的需求描述，但是对项目目标模糊不清的描述将为项目实施带来灾难。
  - 要得到清晰的需求描述（经常是逐步变得清晰的），只能通过客户和开发人员之间保持持续有效的沟通。
  - 得到需求后，开发人员也不可能立即着手编程，还需要进行设计工作。

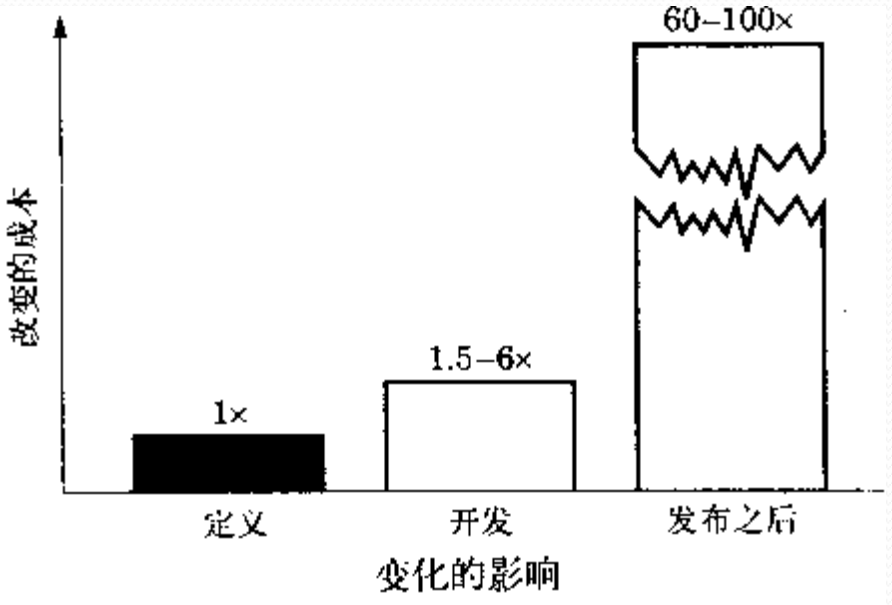
客户神话2

- 神话2：虽然软件需求不断变更，但是因为软件是弹性的，因此可以很容易地适应变更。
- 事实：对需求把握得越准确，软件的修补就越少。有些需求在一开始时很难确定，有些需求的确在随时发生变化，但随变更引入的时机不同，变更所造成的影响也不同。
  - 若需求变更提出得较早（如设计阶段），则该修改的成本较小。
  - 若设计框架已建立，或代码已开发，变更则可能会引起巨变。

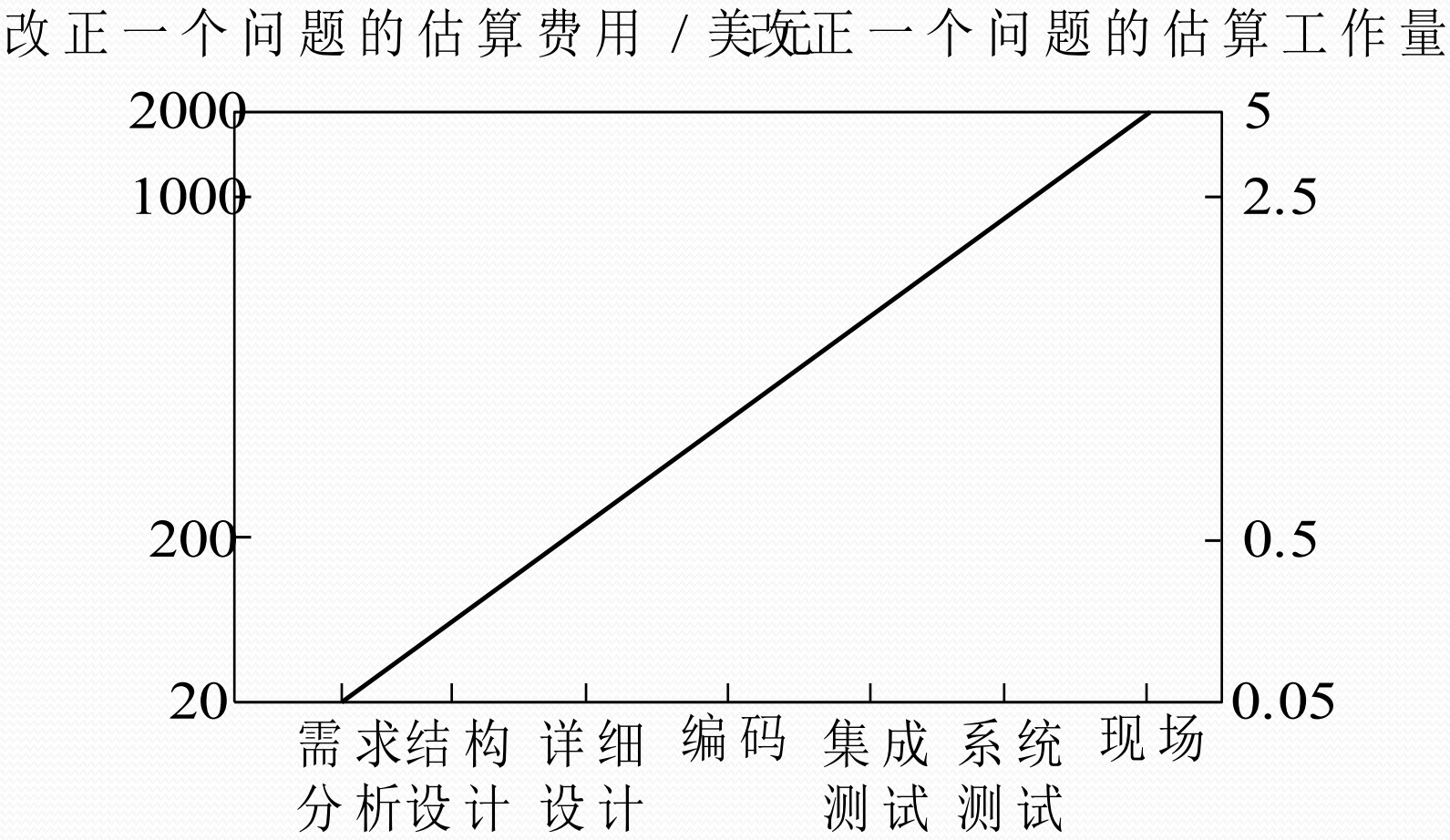
修改的代价



引入同一修改的代价随时间变化的趋势



- 美国贝尔实验室统计了在不同时期修改某个变更所付出的代价，得出的定量结果如下：



从业者神话1

- 神话1：当我们完成程序并将其交付使用后，我们的任务就完成了。
- 事实：
  - 业界的数据显示，60%~80%的工作耗费在软件首次交付顾客使用之后。
  - 通常，顾客在使用软件后会发现或多或少的错误，或者不符合他的需求，或者会有新的需求。

从业者神话2

- 神话2：直到程序开始运行，才能评估其质量。
- 事实：
  - 技术评审，最有效的软件质量保证机制之一，可以从项目启动就开始实行。
  - 软件评审作为“质量过滤器”，已经证明可以比软件测试更为有效地发现多种类型的软件缺陷。

从业者神话3

- 神话3：对于一个成功的软件项目，可执行程序是唯一可交付的工作成果。
- 事实：除了可执行程序，模型、文档、计划等其他工作产品也是必须的。
  - 面向开发者：在不同阶段产生的工作产品可为下一阶段的生产服务。
  - 面向最终的用户：相关文档可提供一定的技术支持。

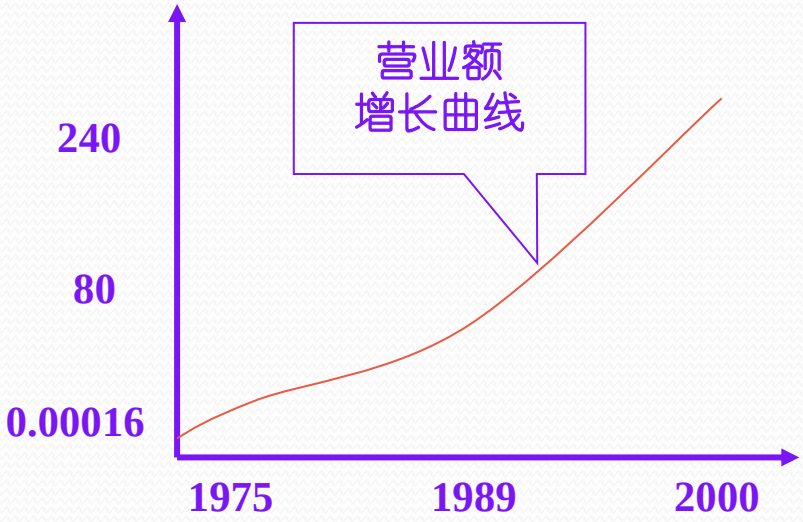
从业者神话4

- 神话4：软件工程将导致我们产生大量的无用文档，并因此降低工作效率。
- 事实：
  - 软件工程并非以创建文档为目的，而是为了保证软件产品的开发质量。
  - 好的质量可以减少返工，从而加快交付时间。

第四节
小结

软件危机解决了吗？

- 看看实行现代软件开发的微软公司：
  - 1975年，3名员工，营业额仅有16000美元；
  - 1989年，8000名，80亿美元；
  - 2000年，35000名，240亿美元（利润150亿）。



如此看来，
软件危机
解决了吗？

没有银弹

- 40多年来，“软件危机”一直困扰着人们，严重阻碍着IT项目和软件产业的发展。
- 人们在苦苦地寻找“银弹”（即解决软件存在问题的有效办法），希望有一天能够降服这些“人狼”。
- Brooks在其著名的文章《No Silver Bullet（没有银弹）》一文中指出：软件的特殊性使寻找能解决所有软件产品问题的方法将成为完全不可能的事，所以不可能存在“银弹”。

新的挑战

- 开放计算：无线网络和各种终端设备使得普适计算、分布式计算、云计算成为可能。
- 网络资源：web2.0让万维网发展成一个计算引擎和内容提供平台。
- 开源软件。
- 软件即服务。

- 意外效应法则

讨论

1. 编程时是否应该多使用技巧？
2. 如果拥有最好的开发工具、最好的计算机，就一定能做出优秀的软件吗？
3. 有最好的软件工程方法，最好的编程语言吗？

本章重点

- 软件的定义和特性。
- 计算机软件的七大分类。
- 遗留系统发生系统演化的原因。
- 软件工程的含义和目标。
- 软件工程的层次图。
- 软件工程三要素。
- 软件神话：管理者，用户，从业者

The End.