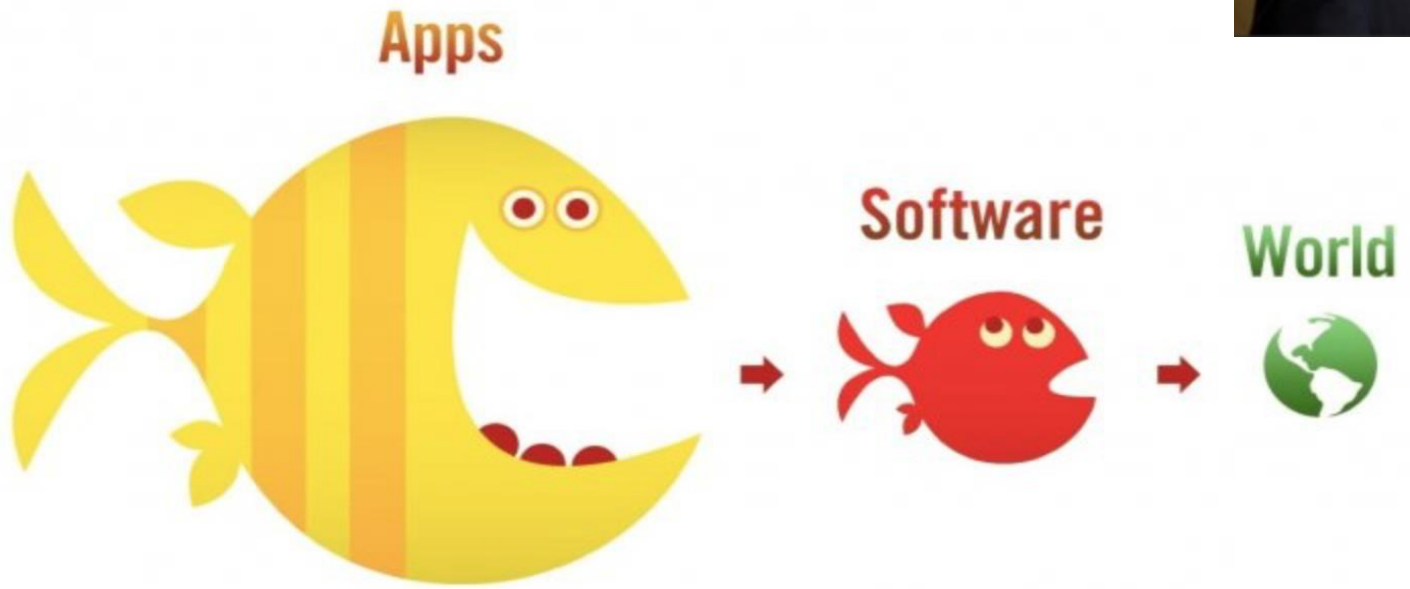
CS2001
软件工程

1. 软件工程概述

Software is Eating the World

Marc Andreessen

Netscape创始人之一，Facebook、Linkedin、HP等公司董事，硅谷风投公司Andreessen-Horowitz合伙人



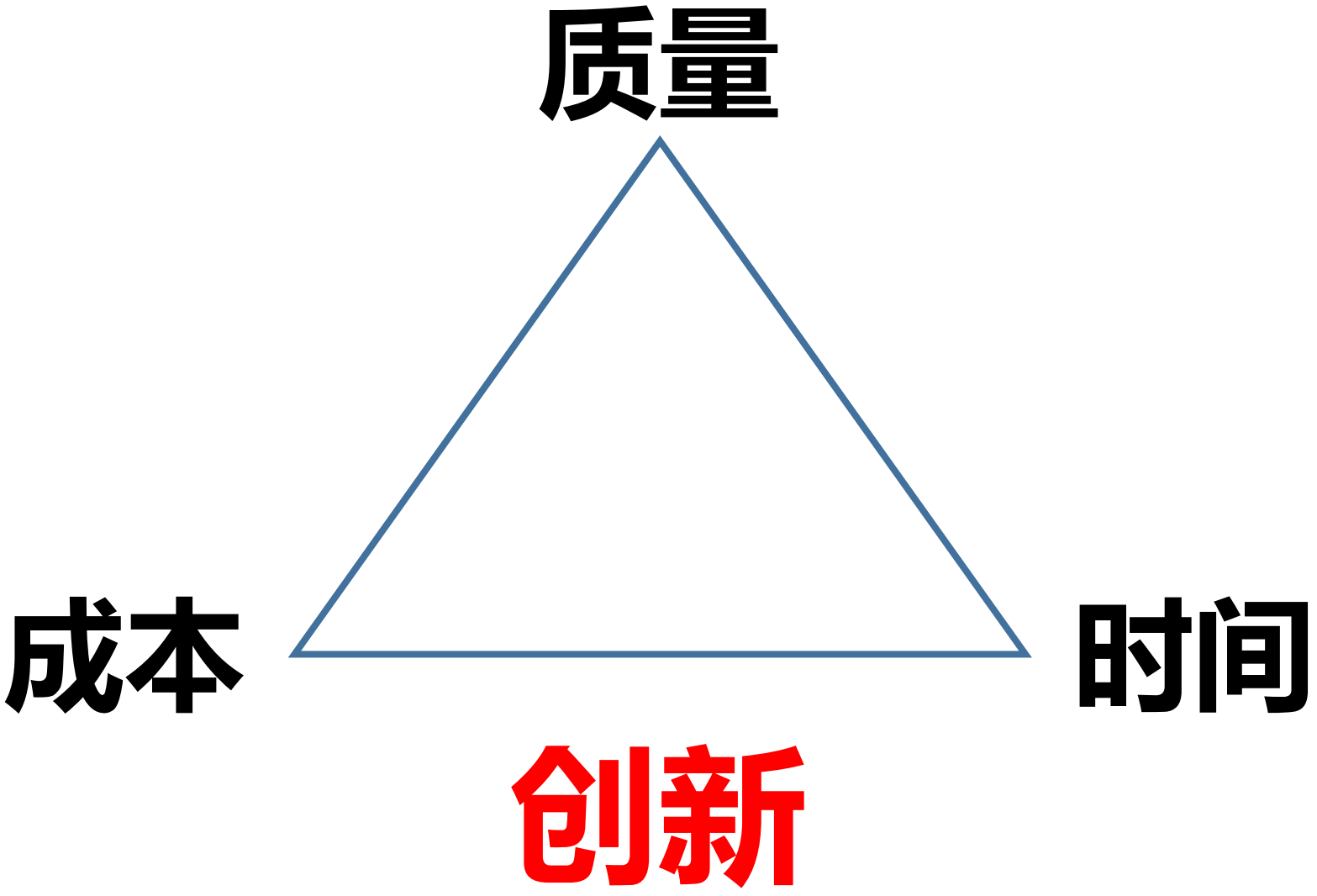
In the future every company will become a software company.

Marc Andreessen. Why Software Is Eating the World. Wall Street Journal, 2011.

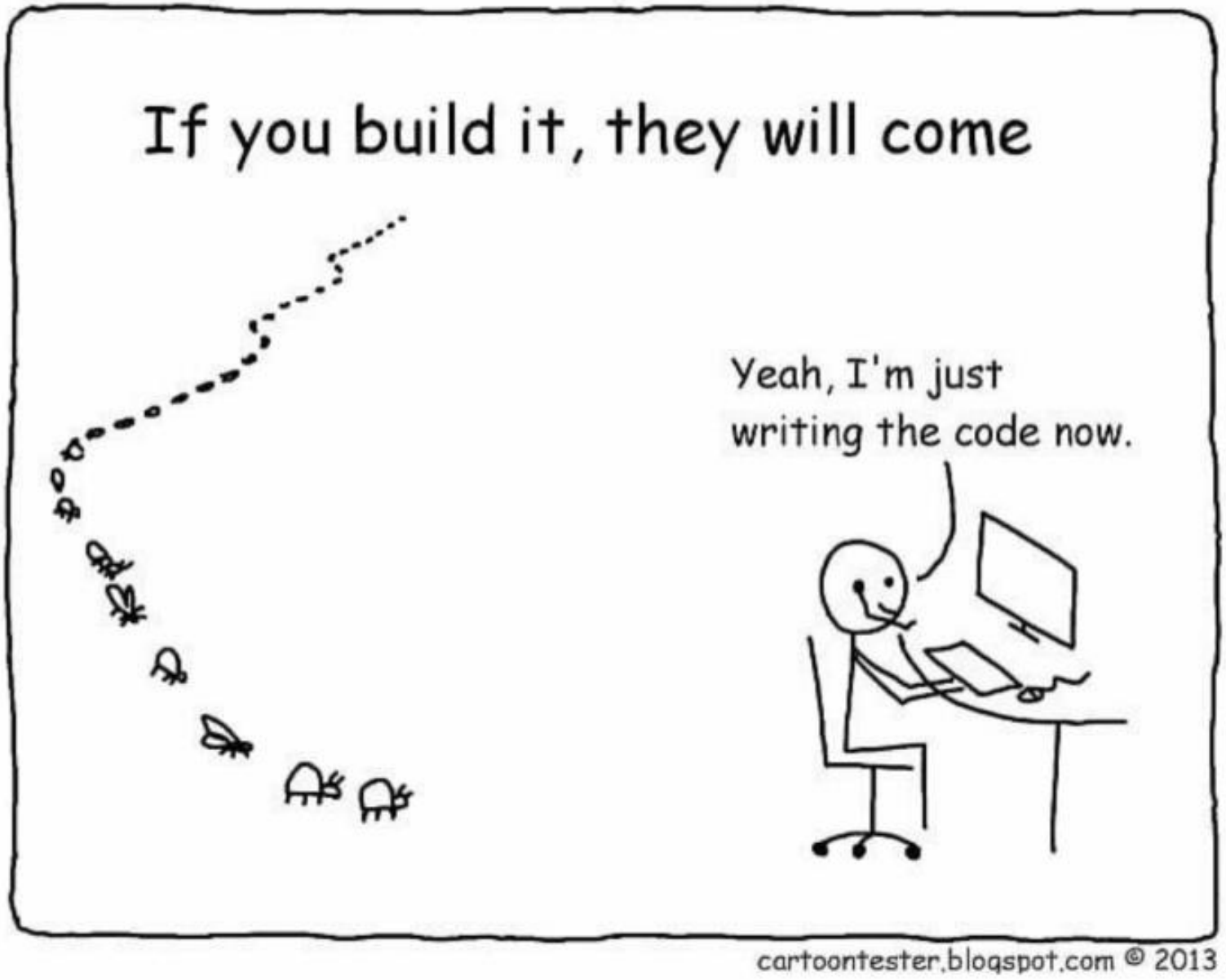
软件产品

- 通用产品
  - ✓ 在市场上向所有顾客销售的独立软件系统
  - ✓ 例如Office软件、通用软件工具、游戏等
  - ✓ 新的分发方式：软件即服务、应用商店
- 定制化产品
  - ✓ 接受特定客户委托、为满足客户需要而开发的软件
  - ✓ 例如嵌入式汽车电子系统、各种定制化开发的信息系统等

软件开发的基本关注点

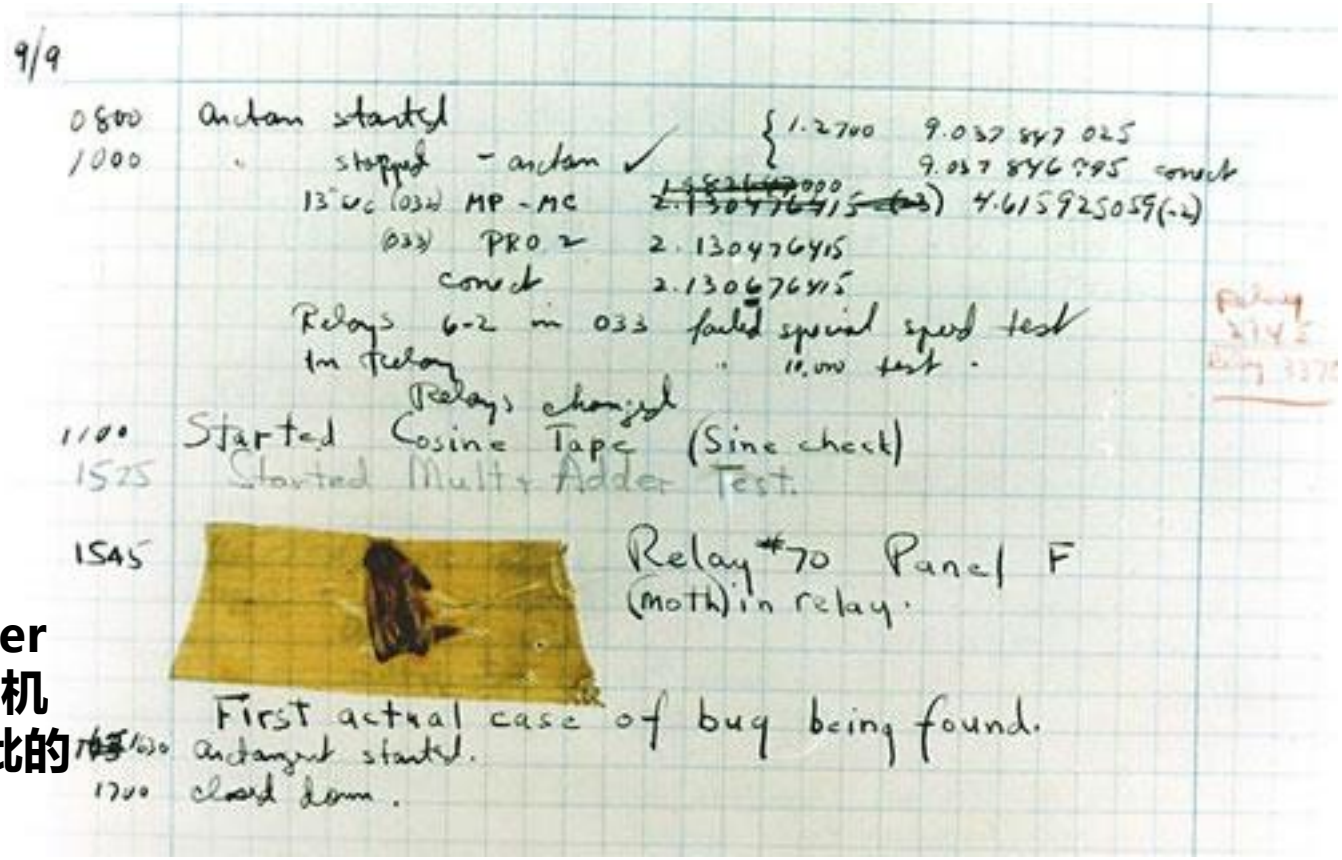


挥之不去的bug


If you build it, they will come
Yeah, I'm just writing the code now.
cartoontester.blogspot.com © 2013

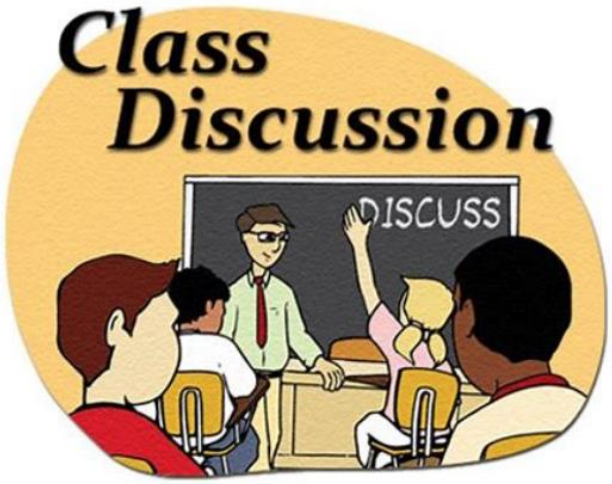
关于Bug

Grace Murray Hopper
COBOL之母，美国计算机科学家，世界上最早一批的程序员



1947年9月9日下午3点45分，Grace Murray Hopper在她的记录本上记下了史上第一个计算机Bug——在Harvard Mark II计算机里找到的一只飞蛾，她把飞蛾贴在日记本上，并写道“First actual case of bug being found”。

课堂讨论：什么是bug



课堂讨论：请结合自己的亲身经历说明下有哪些类型的bug

软件行为与用户期望不一致

崩溃、报错、丢失数据
无法实现所声明的功能
难以使用、性能差、不稳定

Bug or Feature?

取决于用户期望！

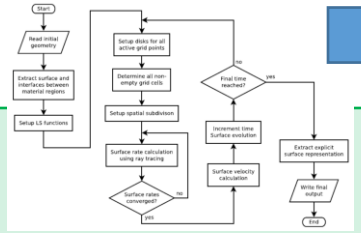
用户期望

- 产品质量：均经过出厂检验
  - ✓ 基本驾驶功能：动力、转向、刹车
  - ✓ 安全保障：刹车灵明度、部件耐磨性
  - ✓ 非功能性质量：舒适度、油耗、加速性能
- 个性化：每款车都有愿意购买的顾客
  - ✓ 价格便宜，省油
  - ✓ 舒适，价格适中，大气有面子
  - ✓ 越野性能好，坚固

开发视角与用户视角

软件开发

抽象问题描述



问题理解

程序员（Programmer）

编写程序

程序

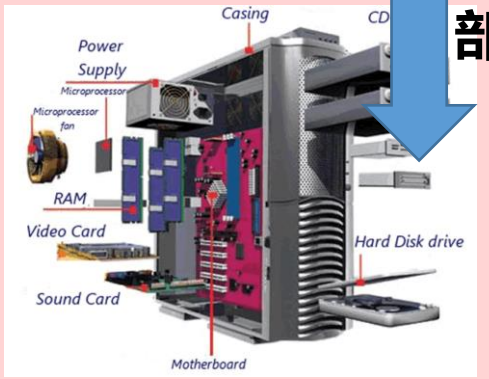
```
double dMax(double *dArray, int iLenOfArray)
{
    double max = dArray[0];
    int iCount;

    for (iCount = 0; iCount < iLenOfArray; ++iCount)
    {
        if (dArray[iCount] > max)
        {
            max = dArray[iCount];
        }
    }

    return max;
}
```

问题抽象

部署运行



现实世界问题，如计算工资

现实世界

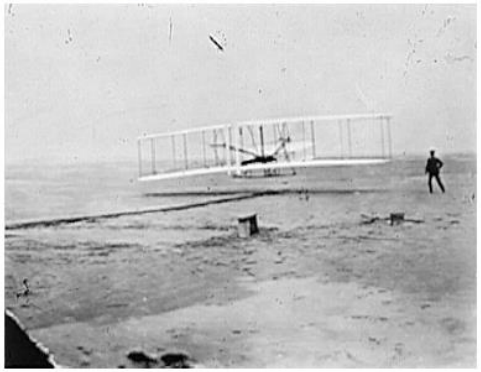
计算设备，如服务器、PC机

计算机世界

飞行器的发展历程

纸飞机
（玩具）

热气球
（尝试）



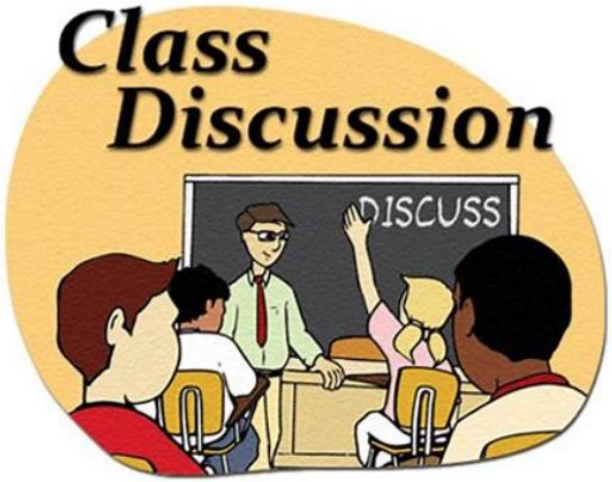
莱特兄弟的飞机
（原型）

波音/空客飞机
（产品）

如果软件出了问题

- 玩具阶段：随便敲的代码不工作了
  - ✓ 不要了，换个东东写写，乐在其中
- 尝试阶段：写的小程序不工作了
  - ✓ 花点时间慢慢调试一下
  - ✓ 调出来有成就感，实在调不出来就算了
- 原型阶段：创业的软件失败了
  - ✓ 总结教训，换个思路重新再来
- 产品阶段：专业软件出问题了
  - ✓ 用户投诉，耗费人力处理
  - ✓ 声誉下降，影响市场占有率
  - ✓ 重大经济财产损失甚至人身伤亡

课堂讨论：工程化产品



课堂讨论：请结合飞行器发展历程思考工程化产品的特征

工程化产品

持续迭代改进

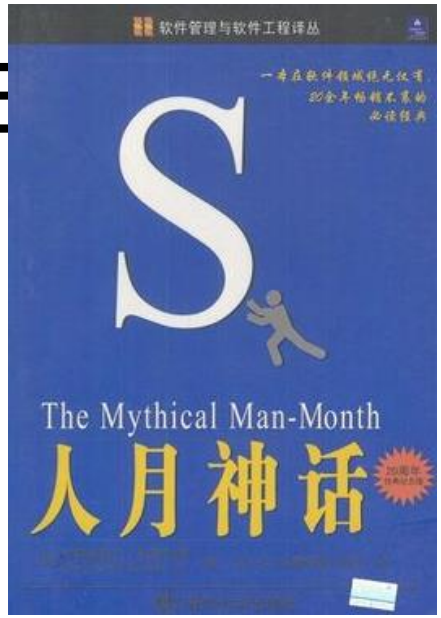
功能完善（提供客户和用户所需的功能）
质量可靠（稳定性、安全性等）
用户友好（使用方便、舒适、操作方便）
具备经济性（购买和维护成本可接受）

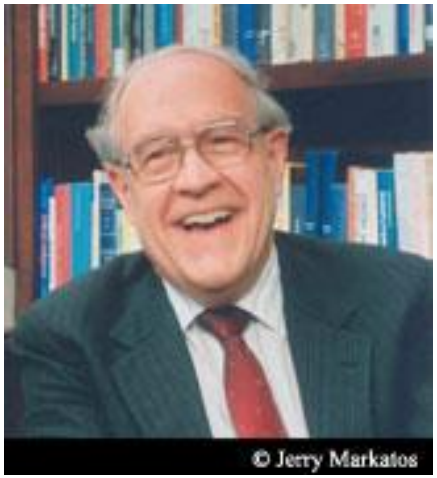
软件危机

- 60多年前，计算机科学从电子学中脱颖而出
- 随后出现了所谓的软件危机（Software Crisis）
  - ✓ 软件开发进度难以预测
  - ✓ 软件开发成本难以控制
  - ✓ 用户对产品功能难以满足
  - ✓ 软件产品质量无法保证
  - ✓ 软件产品难以维护

IBM360系统的例子

- 开发时间：1963-1966年
- 投入人力：5000人年
- 代码量：100万行
- 耗资数亿美元
- 结局
  - ✓ 发布推迟，成本超支好几倍
  - ✓ 需要的内存比计划多很多
  - ✓ 第一次发布时还不能很好运行，
  - ✓ 直到发布了好几次以后（每次都会修订上千个错误）





Frederick P. Brooks, Jr.

Brooks的总结

……正像一只逃亡的野兽落到泥潭中做垂死的挣扎，越是挣扎，陷得越深，最后无法逃脱灭顶的灾难。……程序设计工作正像这样一个泥潭，……一批批程序员被迫在泥潭中拼命挣扎，……谁也没有料到问题竟会陷入这样的困境……

软件工程的诞生

1968年NATO会议上第一次提出了软件工程的概念
从此，软件工程从计算机学科体系中脱颖而出

Doug McIlroy on Software Components, 1968

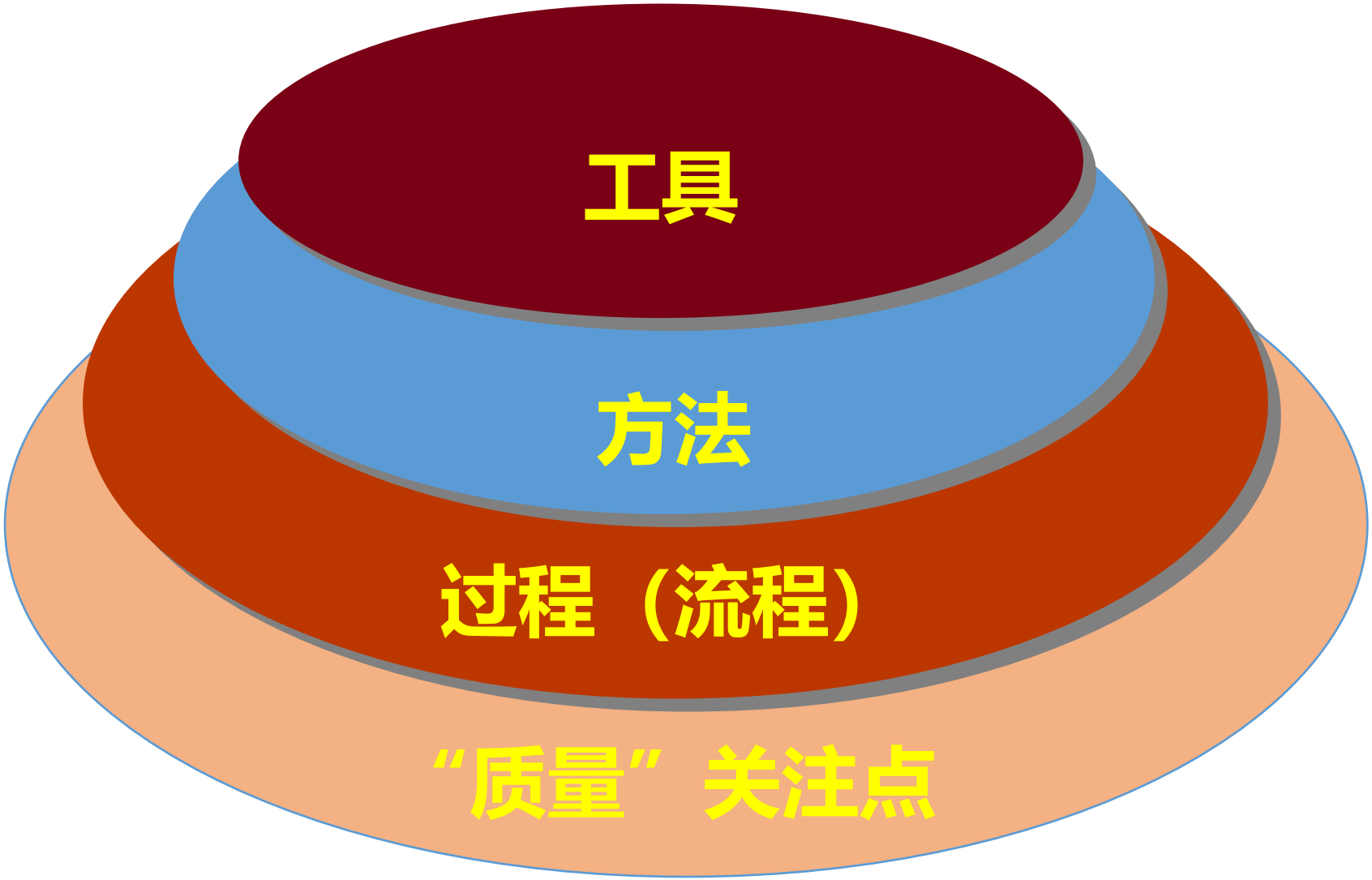
Fritz Bauer, NATO, 1968

The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

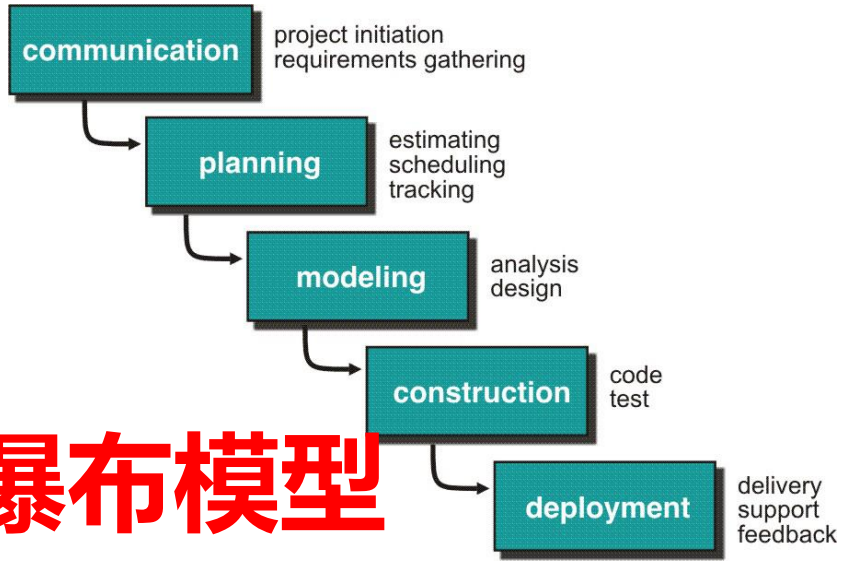
工程（Engineering）

- 质量及成功在很大程度上能够得到保障
- 可以通过标准等手段进行明确总结和定义
- 所涉及的原则有相应的理论和实践支撑
- 通常包含一系列定义良好的过程，包括活动、相关的工具、角色和工作指南
- 可重复(可复制成功)、可学习、可培训、可遵循
- 稳定、可控
- 遵循实用性原则，不追求完美

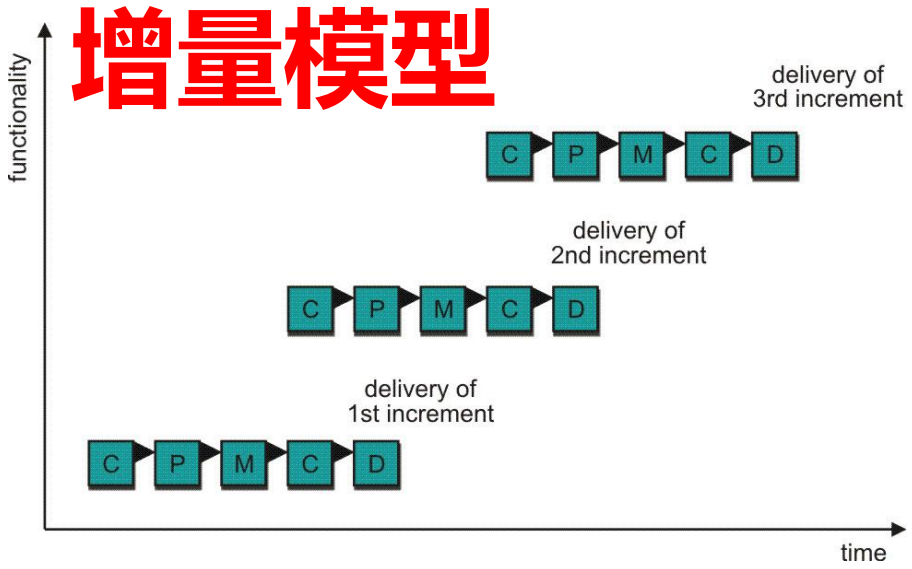
工程化的几个层次



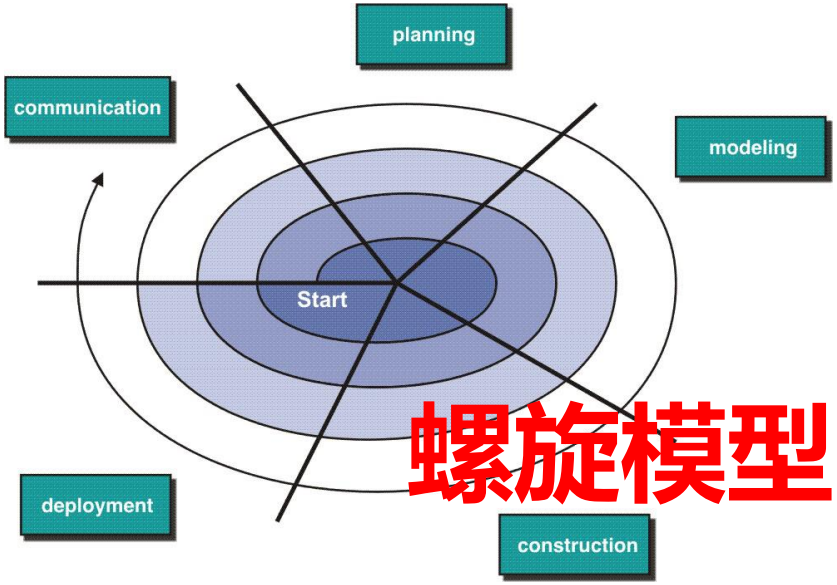
软件开发过程



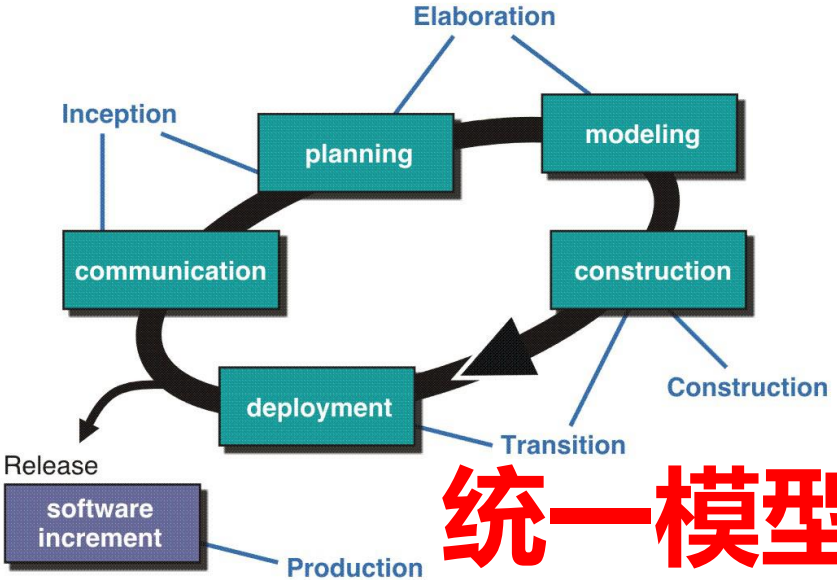
瀑布模型



增量模型



螺旋模型



统一模型

对软件工程的理解

- 面向质量和经济目标的软件开发工程化
- 包括技术和管理
- 对软件整个生存周期的理解
- 软件演化和软件维护
- 针对不同类型软件的工程化开发
- 涉及多个层面：软件产业、软件组织、软件项目、软件开发者

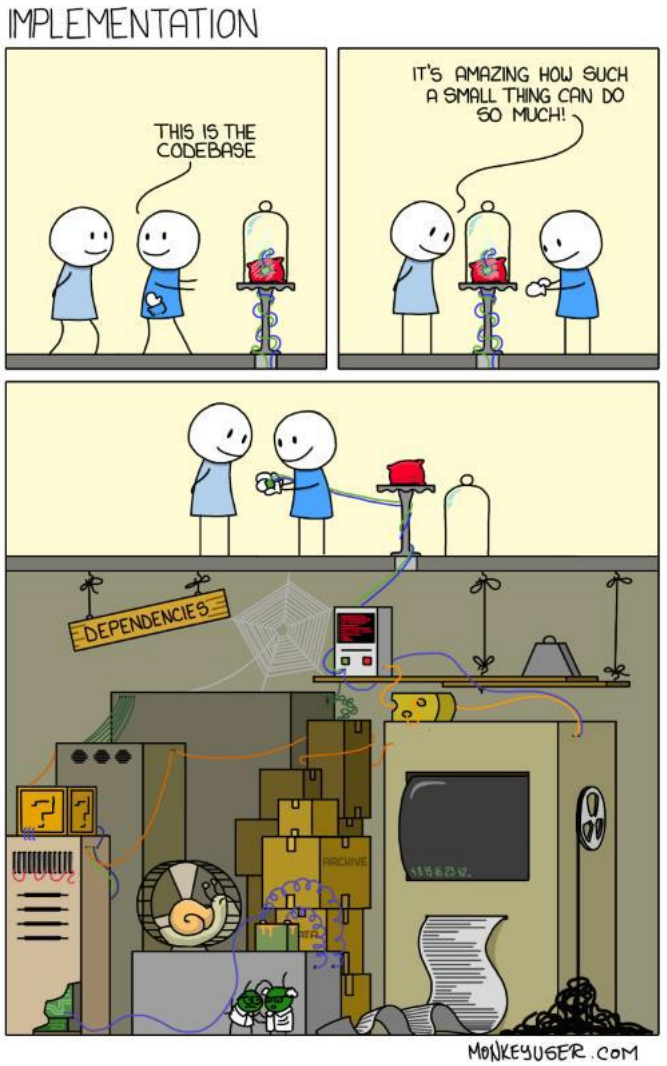
软件工程的根本性困难

- 软件开发就是将问题空间(所面临的特定问题域)转化为解空间(代码、文档)的过程
- 根本性困难
  - ✓ 不可见的逻辑产品（对比：物理学和化学对于制造业、建筑业等成熟领域的工程技术支撑)
  - ✓ 问题空间和解空间之间的巨大鸿沟
  - ✓ 面临的领域纷繁复杂、分布干泛，不同软件系统之间的差异性很大
  - ✓ 软件系统的复杂度越来越高（远高于计算机硬件)

隐藏的复杂性

隐藏的复杂性

软件维护的冰山



隐藏的依赖

对软件工程的两种理解

适应变化
快速迭代
个体与交互
可运行的代码

敏捷

计划

遵循计划
按部就班
过程与工具
详尽的文档

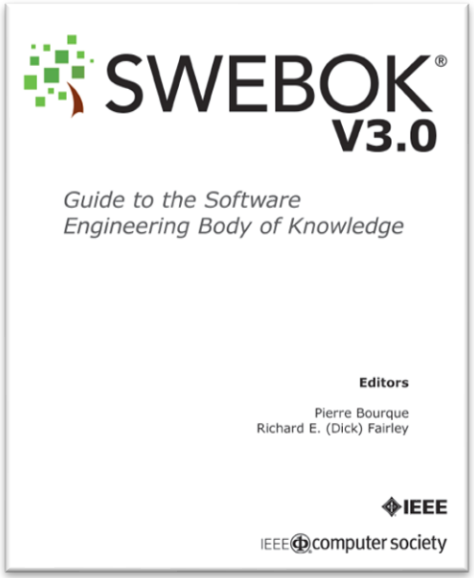
充分考虑软件的特点，灵活、便于调整

传统工程化思维稳定、便于协调

The ways of working out our differences took a sudden dangerous turn.

软件工程知识体系

- 任何一个职业都是建立在相应的知识体系基础上
  - ✓ 制定职业培训计划
  - ✓ 专家资格认证
  - ✓ 职业许可认证



软件工程知识体系

Software Engineering Body of Knowledge (SWEBOK)

SWEBOK总体结构（V3.0）

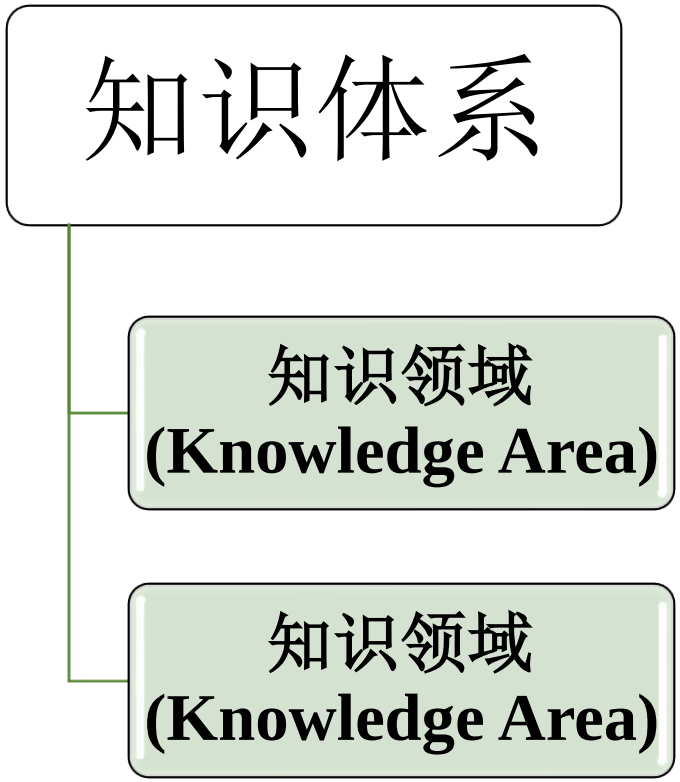


Table I.1. The 15 SWEBOK KAs
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Models and Methods
Software Quality
Software Engineering Professional Practice
Software Engineering Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations

软件工程师的社会责任

现实世界

国家需求
社会需要

伦理道德
社会影响

理解

应用

职业道德
工匠精神

软件定义一切

可靠可信
自主可控

交付

软件工程师

软件系统

软件工程师的职业道德

- 软件的开发及其应用必须符合法律规定
  - ✓ 不能利用软件技术作恶（例如开发软件进行恶意攻击或其他非法活动）
  - ✓ 不能在软件开发中侵犯他人知识产权
- 软件的开发及其应用必须满足道德规范和职业精神
  - ✓ 保守商业和技术秘密
  - ✓ 尊重雇主和客户利益
  - ✓ 保护用户隐私
- 软件工程师必须有责任感
  - ✓ 有着高度的质量意识、追求卓越
  - ✓ 有社会责任感、追求社会公平与正义

阅读建议

- 《软件工程》第1章“概述”
  - ✓ 支持视频：http://software-engineeringbook.com/videos/software-engineering/
  - ✓ 案例描述：http://software-engineeringbook.com/case-studies/
- 《构建之法》第1章“概述”

快速阅读后整理问题
在QQ群中提出并讨论

CS2001
软件工程

End

1. 软件工程概述