



# 软件工程

## 课程介绍



戚荣志

---

# Know your staff

- 主讲教师：许峰、戚荣志
  - 邮件： [rzqi@hhu.edu.cn](mailto:rzqi@hhu.edu.cn)
  - QQ: 32914221
  - 办公室：勤学楼4207

# CS2013    CSTransfer2017



## Body of Knowledge

<b>Algorithms and Complexity (AL) – 17 LOs</b>	<b>Architecture and Organization (AR) – 11 LOs</b>
<b>Computational Science (CN) – 3 LOs</b>	<b>Cybersecurity (CYB) – 25 LOs</b>
<b>Discrete Structures (DS) – 34 LOs</b>	<b>Graphics and Visualization (GV) – 5 LOs</b>
<b>Human-Computer Interaction (HCI) – 6 LOs</b>	<b>Information Management (IM) – 13 LOs</b>
<b>Networking and Communications (NC) – 8 LOs</b>	<b>Operating Systems (OS) – 13 LOs</b>
<b>Parallel and Distributed Computing (PD) – 5 LOs</b>	<b>Platform-based Development (PBD) – No LOs</b>
<b>Programming Languages (PL) – 10 LOs</b>	<b>Software Development Fundamentals (SDF) – 19 LOs</b>
<b>Software Engineering (SE) – 14 LOs</b>	<b>System Fundamentals (SF) – 9 LOs</b>
<b>Social Issues and Professional Practice (SP) – 22 LOs</b>	

# SWEBOK v3.0

- ❖ Guide to the Software Engineering Body of Knowledge
- ❖ 软件工程知识体系指南
- ❖ 美国IEEE协会和ACM于2014年联合公布



# SWEBOK 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

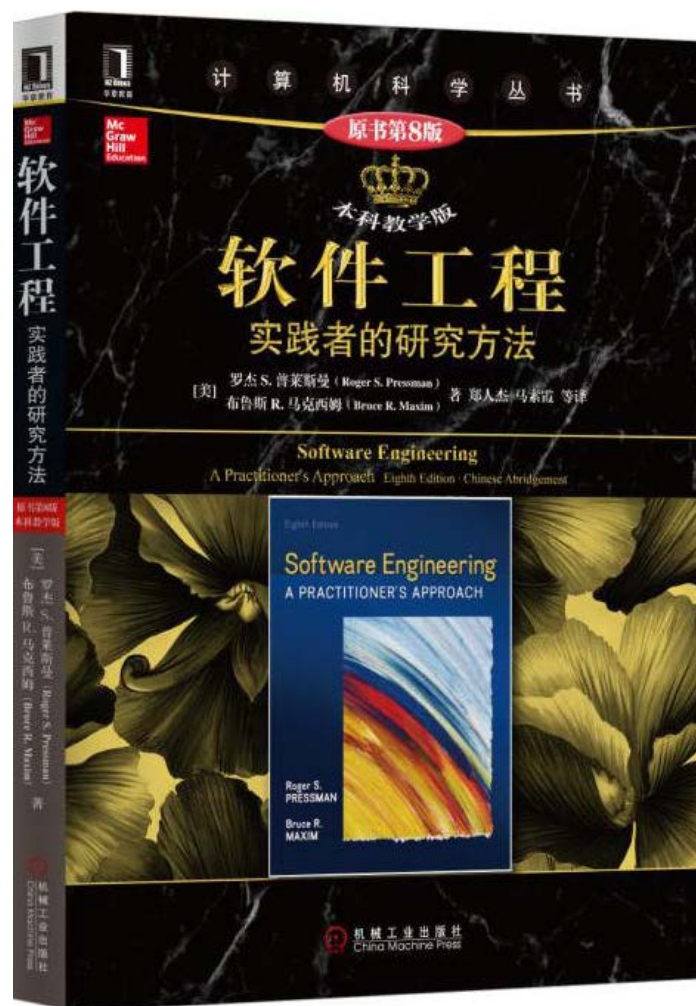
Engineering Foundations

# 教材

- 软件工程：实践者的研究方法（原书第8版，本科教学版）

Roger S. Pressman

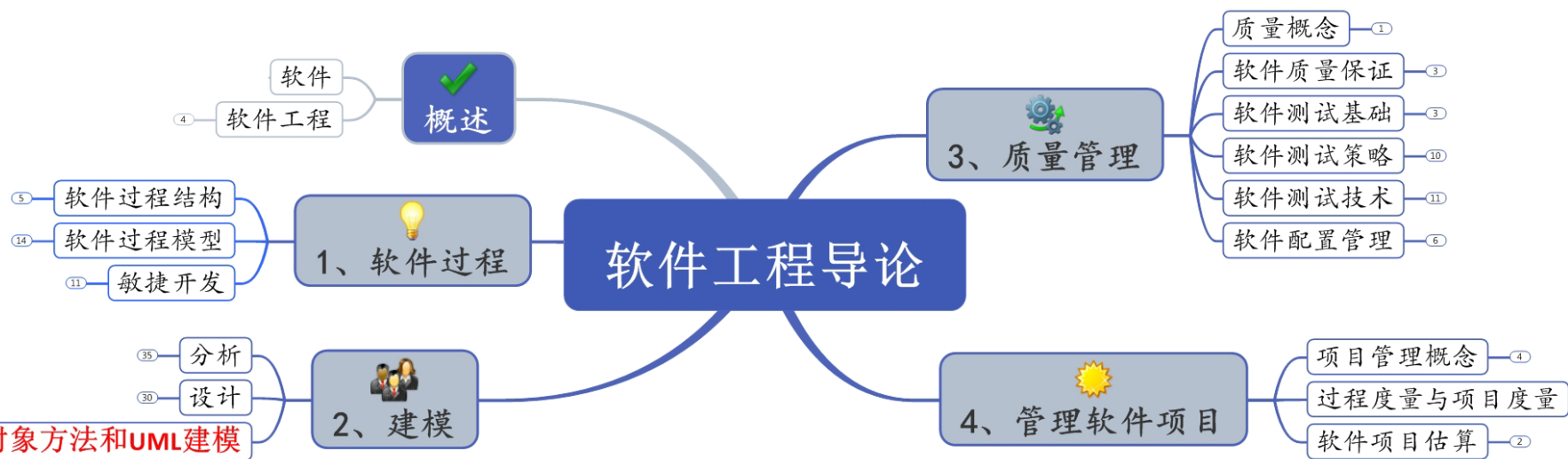
Bruce R. Maxim



# 参考书

- Software Engineering: A Practitioner's Approach (8th Edition), Roger S. Pressman.
- Software Engineering (10th Edition), Ian Sommerville
- The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition (2nd Edition), Frederick P. Brooks, Jr.
- 软件工程导论（第6版），张海藩，清华大学出版社

# 课程内容





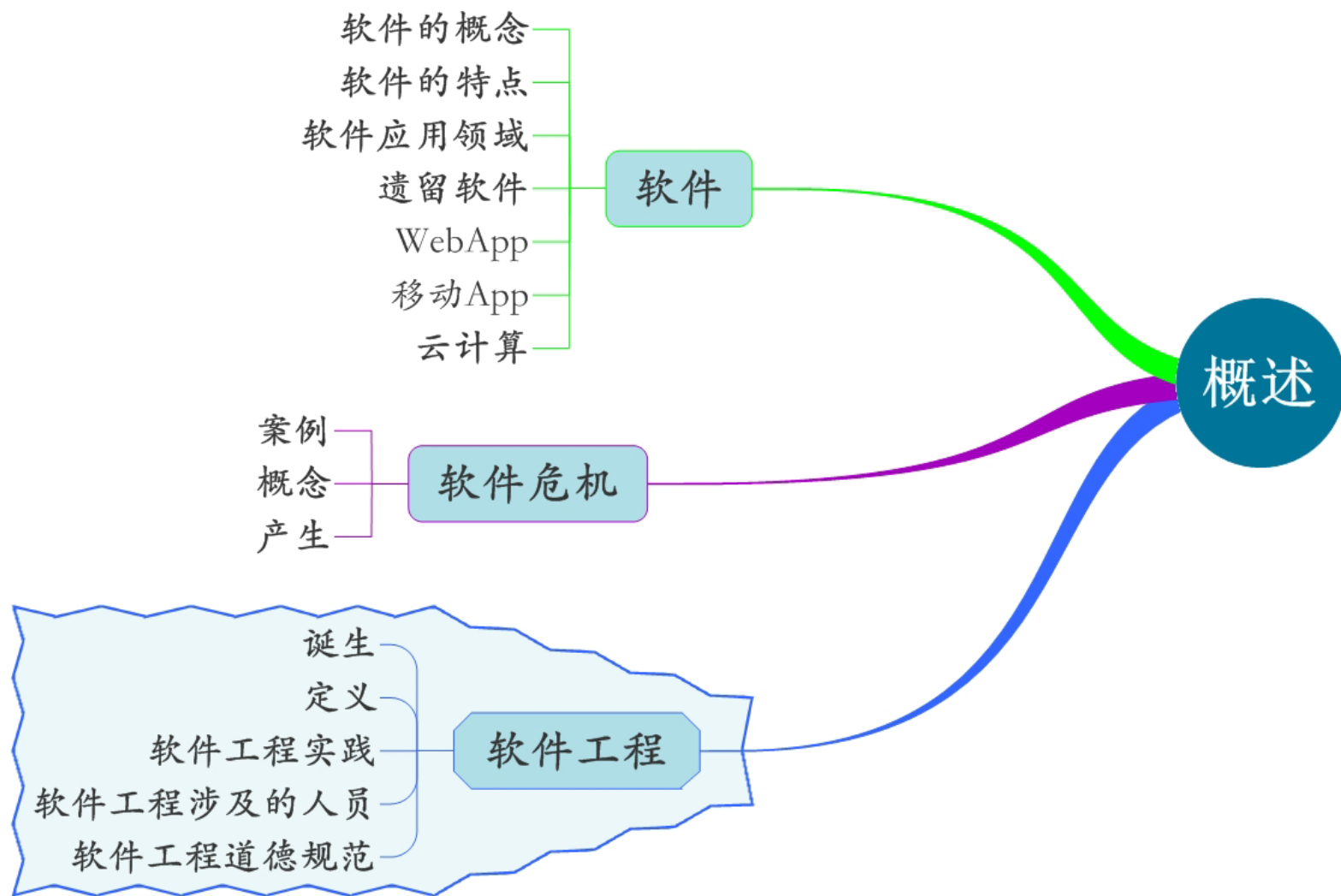
# 课程目标

- 在掌握软件工程基础理论知识的同时，真正学会运用软件工程的**思想**、工具和技术进行软件的需求分析、设计、开发实现和测试。

# 课程考核

- 平时 40%
  - 出勤+课堂互动+作业
- 期末 60%
  - 闭卷考试

# outline



# 软件的概念

**程序 = 算法+数据结构**

**软件 = 程序+数据+文档**

## ■ 软件是：

- **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征、功能和性能需求；
- **数据结构**，使得程序可以合理利用信息；
- **软件描述信息**（文档），它以硬拷贝和虚拟形式存在，用来描述程序操作和使用。

# 问题1

## ■ 有哪些类型的程序设计语言？

# The Evolution Of Computer Programming Languages

4D	54	68	64	00	00	00	06	00
72	6E	00	00	00	61	00	F0	0A
00	41	F7	00	D0	00	00	00	C0
5A	32	01	00	00	00	00	00	00
00	00	00	FF	51	03	06	8A	1B
6A	6F	86	00	90	43	40	2C	80
00	42	00	04	90	41	5E	2C	00
80	40	00	00	FF	2F	00	4D	54
B8	00	00	00	C8	04	00	FF	7
00	00	00	00	00	00	00	00	00
03	07	65	2D	70	89	61	6E	6F
08	00	01	4B	28	80	42	2A	00



Hex

```

ORG      ORG      20h
VARI     ORG      1
STATE    BIT      VARI.0
OUTPUT   BIT      PI.0

CORG     ORG      0h
         AAMP     START

         ORG      0h
         AAMP     INTERRUPT

START    MOV      R0, R2D2
         MOV      R0, #PI
         MOV      R0, #PCB
         MOV      R0, #DCB
         TRP

```



# Assembler

```
#include <stdio.h>
#include <io.h>
#include <dos.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```

```
main()
{
char ch,*txt;
int bit;
int data;
FILE *r;
struct
```



C

[illegible]

## Fortran

```
#include <graphics.h>

class pentape {
public:
    int x, y;
    void draw() { gotox(x,y,WHITE);
};

class cruptype: public pentape {
public:
    int radius;
    void draw() { circle(x,y,radius); }
};

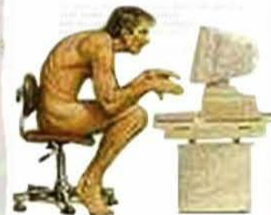
main()
{
    int driver=0;
    driver = getDriver();
    initgraph(&driver,"E:\\AP\\GPA");
    pentape p(100,100);
    cruptype c(100,100,50);
    c.draw();
}
```



C++

[illegible]

# Java

[illegible]

Ruby

## 问题2

- 在软件开发过程中会产生哪些文档？
  - 可行性研究报告
  - 需求规格说明书
  - 概要设计说明书
  - 详细设计说明书
  - 测试报告
  - 用户手册
  - .....

# 问题3

- 软件开发过程中为什么要编写文档？
  - 便于对软件开发的管理和维护
  - 便于各种人员的交流
- 有哪些文档标准
  - 国际标准ISO
  - 行业标准IEEE、CMMI
  - 国家标准GB
  - 企业标准

# 软件的特点

## ■ 复杂性

- ❑ 软件实体比任何由人类创造的其他实体更复杂
- ❑ 难以理解，使维护过程变得十分困难

## ■ 一致性

- ❑ 软件必须适应已有的技术和系统，随接口的不同而改变
- ❑ 复杂性来自保持与其他接口的一致



# 软件的特点

## ■ 可变性

- ❑ 软件经常会遭受到持续的变更压力
- ❑ 软件很容易修改
- ❑ 修改会带来副作用，造成故障率的升高

## ■ 不可见性

- ❑ 客观世界空间和计算机空间中的一种逻辑实体，不具有物理的形体特征。
- ❑ “需要做什么”

---

# 软件应用领域

- 系统软件
- 应用软件
- 工程 / 科学软件
- 嵌入式软件
- 产品线软件
- Web/移动应用软件
- 人工智能软件

# 软件的变更本质

- Web App
  - B/S
  - 分布式
  - Web服务 ( Web services )
- 移动App
- 云计算
  - 软件即服务 ( software as a service )
  - 软件运行在 “云端”
  - “云” 是巨大的计算机集群系统
  - 用户无需购买软件，需根据使用软件的情况付费


# 什么是好的软件？



正确性



可靠性



可用性



可维护性



可测试性



可移植性



可复用性



.....

# outline

- 软件
- 软件危机
- 软件工程

# 案例分析1： IBM360



- IBM360的操作系统：1963~1966年，5000人一年的工作量，近100万行源程序。
- 项目负责人F. D. Brooks：“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……”。

人月神话：“没有一种单纯的技术或管理上的进步，能够独立地承诺在10年内大幅度地提高软件的生产率、可靠性和简洁性”。

## 1、软件活动包含根本任务和次要任务

根本任务：打造构成抽象软件实体的复杂概念结构；

次要任务：使用编程语言表达这些抽象实体。

## 2、现有解决方案致力于解决次要任务

## 3、结论：没有银弹

无论这些方案多么完善，都不可能在根本上解决问题。



# 案例分析2: Ariane 5

- June 4, 1996
- 欧洲宇航局，阿丽亚娜5型火箭
- 首航飞行了大约40秒后开始偏离航向
- 火箭通过远程控制被销毁，5亿美元
- 嵌入式软件，惯性参照系统（SRI）
- 异常被监测到了，但并没有被适当处理
- numeric exception (integer overflow)





# 软件危机的产生



上世纪六十年代  
(1960年代),  
软件开发和维护  
遇到了一系列严  
重的问题

- 需求不明确
- 软件存在大量缺陷
- 软件开发成本和项目周期失控
- 软件难以维护
- .....

**软件  
危机**

# 软件危机的概念

- 软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。
- 这些问题绝不仅仅是不能正常运行的软件才具有的，实际上，几乎所有软件都不同程度地存在这些问题。

Key points:

- how to develop new software
- how to support old software

# outline

- 软件
- 软件危机
- 软件工程

# 软件工程的诞生

## SOFTWARE ENGINEERING

Report on a conference sponsored by the  
NATO SCIENCE COMMITTEE  
Garmisch, Germany, 7th to 11th October 1968

Chairman: Professor Dr. F. L. Bauer  
Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

January 1969



- 1968年秋，NATO（北约）的科技委员会在德国举行“软件工程”大会
- 商讨如何更好地开发软件
- 第一次提出了软件工程 software engineering 的概念
- “软件”有自己的生命周期 (life cycle)
- 软件工程学科诞生的标志

# 软件工程定义

## ■ 几个事实:

- 确定软件方案之前，需要共同努力来理解问题
- 设计已成为关键活动
- 软件应该具有高质量
- 软件需具备可维护性

## ■ 结论:

- 各种形式、各个应用领域的软件都需要工程化

# 软件工程定义

The IEEE Computer Society defines software engineering as

- (1) The application of a **systematic, disciplined, quantifiable** approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
- (2) The study of approaches as in (1).

# 软件工程定义

- CC2001:Software engineering is the discipline concerned with the application of **theory, knowledge, and practice** for effectively and efficiently building software systems that satisfy the requirements of users and customers.

# 软件工程定义

- ◆ 软件工程是一门交叉学科

- 软件开发技术：软件开发方法  
软件开发过程  
软件工具和软件工程环境
- 软件工程管理：软件管理学  
软件经济学

- ◆ 软件工程所包含的内容不是一成不变的，随着人们对软件系统的研制开发和生产的理解。应该用发展的眼光看待它。



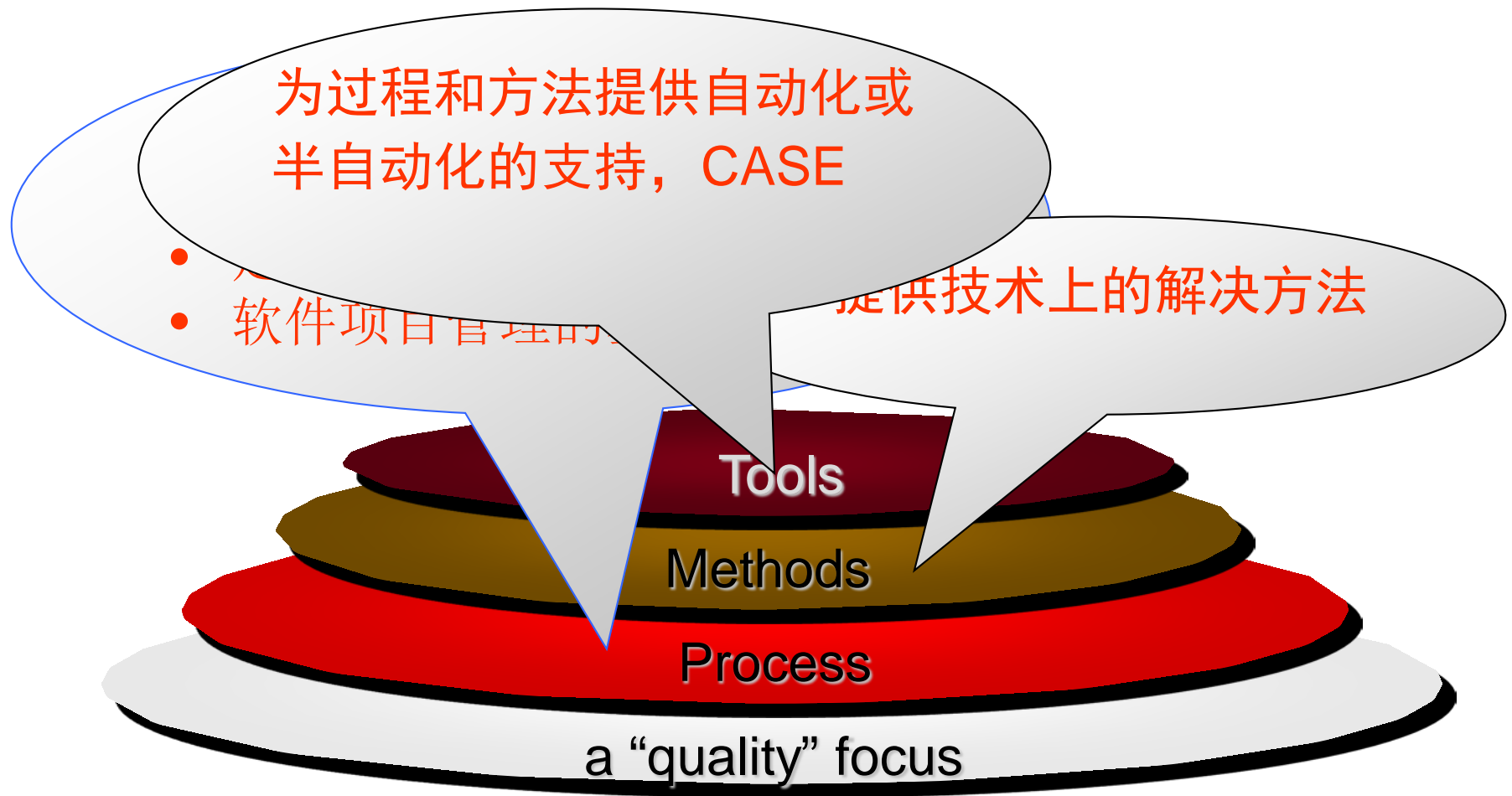
- 转变对软件开发的认识：



- 转变思维定式：



# A Layered Technology



# 软件工程实践的精髓

## ■ Polya给出的建议：

### □ 理解问题 (沟通和分析)

- 谁将从问题的解决中获益？即谁是利益相关者？
- 有哪些是未知的？哪些数据、功能和特征是解决问题必需的？
- 问题可以划分吗？是否可以描述为更小、更容易理解的问题？
- 问题可以图形化描述吗？可以建立分析模型吗？

# 软件工程实践的精髓

## □ 策划解决方案 (建模和软件设计)

- 以前曾经见过类似问题吗？在潜在的解决方案中，是否可以识别一些模式？是否已经有软件实现了所需要的数据、功能和特征？
- 类似问题是否解决过？如果是，解决方案所包含元素是否可以复用？
- 可以定义子问题吗？如果可以，子问题是否已有解决方案？
- 能用一种可以很快实现的方式来描述解决方案吗？能构建出设计模型吗？

# 软件工程实践的精髓

## □ 实施计划(代码生成)

- 解决方案和计划一致吗？源码是否可追溯到设计模型？
- 解决方案的每个组成部分是否可以证明正确？设计和代码是否经过评审？或者采用更好的方式，算法是否经过正确性证明？

## □ 检查结果的正确性 (测试和质量保证)

- 能够测试解决方案的每个部分？是否实现了合理的测试策略？
- 解决方案是否产生了与所需求的数据、功能和特征一致的结果？是否按照项目利益相关者的需求进行了确认？

# Hooker软件工程实践的原则

- 1: 存在价值
  - 为用户提供价值
- 2: 保持简洁
  - 设计尽可能简洁，不是过于简化
- 3: 保持愿景
  - 清晰的愿景是软件项目成功的基础
- 4: 关注使用者
  - 在需求说明、设计和实现中，牢记要让别人理解你所做的事情

# Hooker软件工程实践的原则

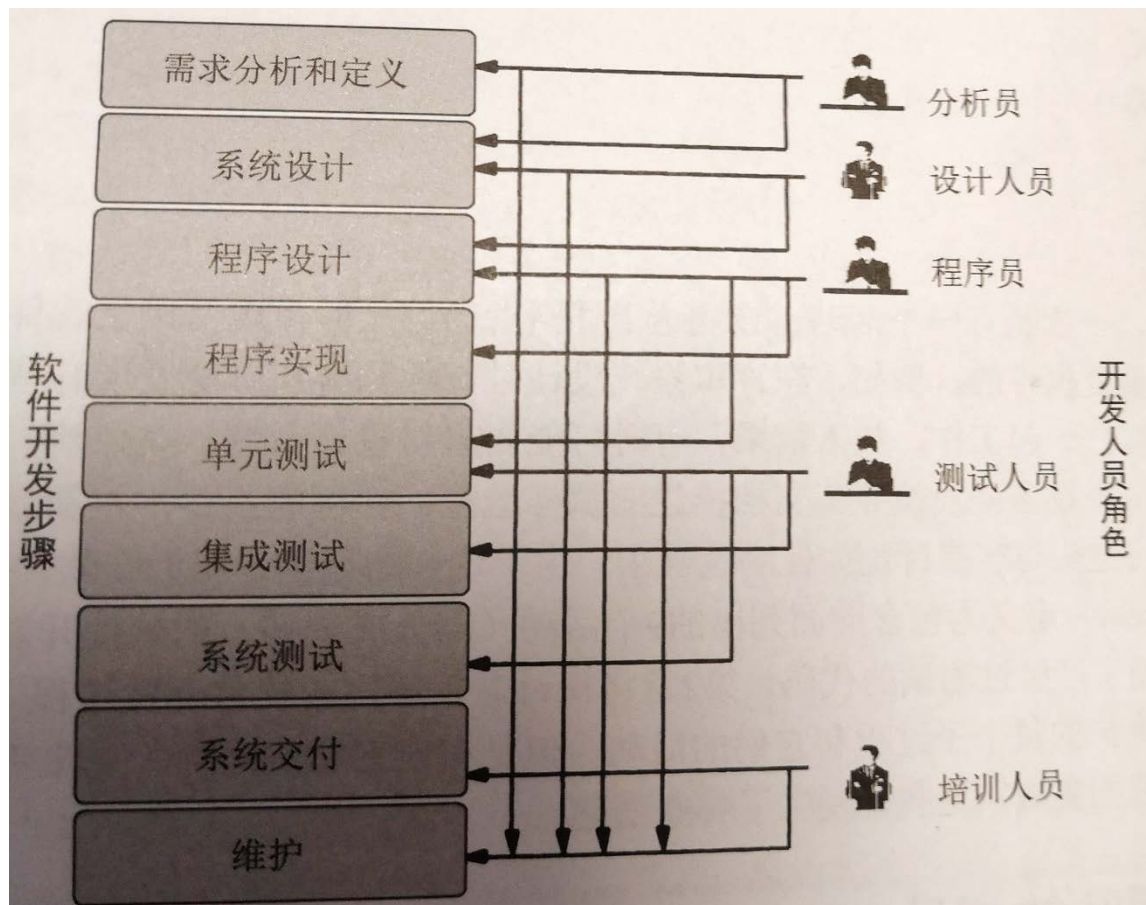
- 5: 面向未来
  - 系统能够适应各种变化，不要把自己的设计局限于一隅
- 6: 计划复用
  - 降低开发费用
- 7: 认真思考
  - 在行动之前清晰定位、完整思考通常能产生更好的结果

# 软件工程涉及的人员





# 软件工程涉及的人员



# 一切是如何开始的

- 每个软件工程项目都来自**业务需求**
  - 对现有应用程序缺陷的修正；
  - 改变遗留系统以适应新的业务环境；
  - 扩展现有应用程序功能和特性；
  - 开发某种新的产品、服务或系统。

# 软件工程道德规范

- IEEE与ACM联合制定
- 软件工程师应履行其实践承诺，使软件的需求分析、规格说明、设计、开发、测试和维护成为一项有益和受人尊敬的职业。
- 软件工程师不但需要熟练掌握软件工程的知识与技能，还必须要认识和遵循社会伦理和职业道德，具有强烈的职业责任感。
- 针对具体的事情，软件工程师应该根据自己的道义判断，最终做出合理的抉择。

# 软件工程道德规范

## ■ 八项原则

- ❑ **公众(public)**：软件工程师应当始终如一地以符合公众利益为目标；
- ❑ **客户和雇主(client and employer)**：在与公众利益保持一致的原则下，软件工程师应满足客户和雇主的最高利益；
- ❑ **产品(product)**：软件工程师应当确保他们的产品和相关的改进符合可能达到的最高专业标准；
- ❑ **判断(judgment)**：软件工程师在进行相关的专业判断时，应该坚持正直、诚实和独立的原则；

# 软件工程道德规范

## ■ 八项原则

- ❑ **管理(management)**：软件工程的管理和领导人员在软件开发和维护的过程中，应自觉遵守、应用并推动合乎道德规范的管理方法；
- ❑ **专业(profession)**：软件工程师应当自觉推动本行业所提倡的诚实、正直的道德规范，并自觉维护本行业的声誉，使软件行业更好的为公众利益所服务；
- ❑ **同行(colleagues)**：软件工程师对其同行应持平等互助和支持的态度；
- ❑ **自身(self)**：软件工程师应终生不断地学习和实践其专业知识，并在学习和实践的过程中不断提高自身的道德规范素养。

# 软件工程道德规范

## ■ 应遵守的规则

- ❑ 从不为了个人利益而窃取数据；
- ❑ 从不散布或售卖你所工作的软件项目的专利信息；
- ❑ 从不恶意地破坏或修改别人的程序、文件或数据；
- ❑ 从不侵犯别人、别的团队或组织的隐私；
- ❑ 从不为了某种利益而非法入侵别人的系统；
- ❑ 从不创建或传播计算机病毒。