Software Engineering –Theory and Practice (4th Edition)

- Instructor: HE Wei
- Contact Information
  - E-mail: hewei@sdu.edu.cn

- 编程相关的课程

  PL(Java、C++、Python……)

  ↓

  OOP/OOD

  ↓

  SE及相关课程

  ↓

  PM及相关课程

- 通过学习本课程，可以
  - 理解
    - **软件产品**：理解**软件质量**的含义、以及如何提高软件质量（建模方法、设计原则、编程规范、测试技术……）
    - **软件开发过程**：以工程的视角理解软件开发过程，俯视**软件开发活动**（分析、设计、实现、测试……）

了解一下“软件工程”

- 软件开发的一个普遍观点
  - 只要掌握一门技术，比如编程语言、开发工具，就足以开发一个可用的软件系统……
  - 很多学校、培训机构正是这样培养“IT人才”的
- 说法并没有错，但从工程的角度来说这种观点是危险的
- 因为（在特定情况下）
  - 软件解决方案有好坏之分
  - **好的软件**：能够很好的解决用户的问题；又能够轻松地适应需求（功能、可扩展等）的变化……
  - **坏的软件**：不能很好的满足用户需求；交付周期过长；不支持需求的变化……

- 软件工程试图研究的问题
  - **什么是好的软件/开发过程？**
  - **什么是坏的软件/开发过程？**
  - **如何生产出好的软件产品？**
  - ……

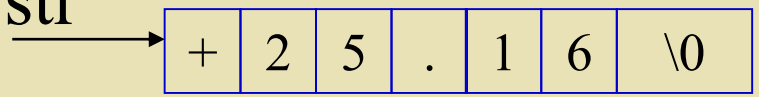
- 源程序文档化：符号名、注释、源代码布局……
- 编写清晰第一，效率第二
- 不要修补不好的程序，要重新编写
- 任何对效率无重要改善，且对程序的简单性、可读性、正确性不利的方法都是不可取的
- 注意浮点数运算的特点；不要单独进行浮点数的比较，通常结果会发生异常

Writing Clean Code

—— Microsoft Techniques for developing Bug-free C Programs

EXAMPLE 1:

使用标准C函数atou将一个字符串转成数字，需要省略前面的”+”号。



Method 1:

```
Return ((int)atou(str + (*str == '+') ) );
```

Method 2:

```
If ((*str != '+') || str++) /* 跳过可选择的’ +’号 */
    return ((int)atou(str)) ;
```

Method 3:

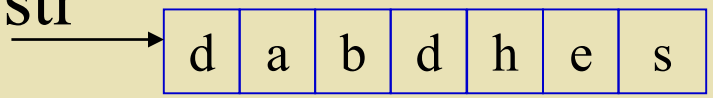
```
if(*str == '+') /* 跳过可选的 '+'号 */
          str++;
  return ((int)atou(str));
```

你认为哪种写法最好？为什么？

EXAMPLE 2:

在字符串(*pv, size)中搜索字符ch，返回ch地址，否则返回空指针：

```
void *memchr(void *pv, unsigned char ch, size_t size)
{
        unsigned char *pch = (unsigned char *)pv;
        while(size-- > 0)
        {
                if(*pcd == ch)
                   return(pch);
                pch++;
        }
        return(NULL);
}
```



优化：程序员热衷于 “如何使循环加快？”的游戏，放弃传统的编码方式并进行大胆尝试：

- 之所以需要循环内的检查仅仅是因为：当在存储器的头size个字节内没有找到要找的字符ch时，就要返回NULL。要删除该检查，只要简单地保证总可以找到ch字符就可以了。
- 实现：在被查找的存储区域后面的第一个字节上存放字符ch。

```
void* memchr(void *pv, unsigned char ch, size_t size)
{
        unsigned char *pch = (unsigned char *)pv;
        unsigned char *pchPlant;
        unsigned char chSave;
        /* pchPlant指向要被查寻的存储区域后面的第一个字节；将ch存
        储在pchPlant所指的字节内来保证memchr肯定能搜索到Ch */
        pchPlant = pch+size;
        chSave = *pchPlant;
        *pchPlant = ch;
        while(*pch != ch)
                pch++;
        *pchPlant = chSave;
        return((pch == pchPlant)?NULL : pch);
}
```

- 通过用ch覆盖pchPlant指向的字符，可以保证memchr总能找到ch，这样就可以删除范围检查，使循环的速度加倍
- 巧妙吗？
- 正确吗？
- 可靠（坚挺）吗？

- 大多数程序员玩弄的一种游戏是“我如何使得代码更快？”的游戏
- 这并不是坏游戏，但是如果过份地热衷于这种游戏，那就是坏事

- 在小说中：使用惊奇和悬念很重要、也很必要
- 但是，如果把它们放到代码中，那就糟糕了
- 当写代码时，“情节”应该直观，以便别的程序员能预先清楚地知道将要发生的一切
- 如果用代码表述：罪犯走近John并刺伤了他，那么写成“罪犯走近John并刺伤了他”最恰当了；而不能像写小说一样（古龙）
- 该代码简短、清楚、并讲述了所发生的一切

- Tips：
  - 直观的代码并不意味着是总是简单，直观的代码可以使你沿着一条明确无奇的路径从A点到达B点
  - 必要的时候直观的代码可能也很复杂

EXAMPLE 3:

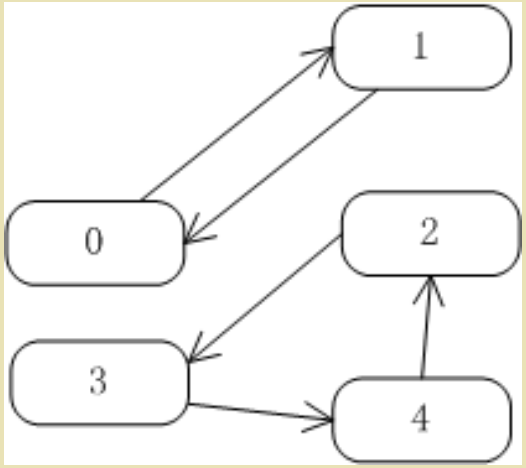
Return what the next checkbox state should be. This function h andles both two-state checkboxes and three-state checkboxes.

Method 1:

```
unsigned uCycleCheckBox(unsigned uCur)
{
    return ((uCur<=1)?(uCur?0:1) : (uCur == 4)?2 : (uCur +1));
  }
```

Method 2:

```
unsigned uCycleCheckBox(unsigned uCur)
{
   if (uCur == 0) return 1;
   if (uCur == 1) return 0;
   if (uCur ==4) return 2;
   return uCur + 1;
}
```



- 点评：
  - 如果你总是使用稀奇古怪的表达式，以便把代码尽量写在源代码的一行上，从而达到最好的瑜伽状态的话：
  - 你很可能患有可怕的“一行清”疾病（也称为程序设计语言综合症）
- 准则：
  - 写直观、简捷、清楚的代码，禁止使用具有技巧的、比较精炼的、异乎寻常的编码方法
  - 直观的代码并不意味着是简单的代码，直观的代码可以使你沿着一条明确无奇的路径从A点到达B点（必要的时候直观的代码可能也很复杂）。

EXAMPLE 4: 设计问题（设计C语言的内存分配API）

```
void* realloc(void* pv, size_t size);
```

realloc改变先前已分配的内存块的大小，该内存块的原有内容从该块的开始位置到新块和老块长度的最小长度之间得到保留。

- 如果**该内存块的新长度小于老长度**：realloc释放该块尾部不再想要的内存空间，返回的pv不变
- 如果**该内存块的新长度大于老长度**：扩大后的内存块有可能被分配到新的地址处，该块的原有内容被拷贝到新的位置。返回的指针指向扩大后的内存块，并且该块扩大部分的内容未经初始化
- 如果**满足不了扩大内存块的请求**：realloc返回NULL，当缩小内存块时，realloc总会成功
- 如果**pv为NULL**：那么realloc的作用相当于调用malloc(size)，并返回指向新分配内存块的指针，或者在该请求无法满足时返回NULL
- 如果**pv不是NULL，但新的块长为零**：那么realloc的作用相当于调用free(pv)并且总是返回NULL
- 如果**pv为NULL且当前的内存块长为零**：结果无定义

缺点：

- 首先，无法安全地使用
- 其次，传递给realloc的可能是无用信息，但是因为其定义如此通用，使它很难防范无效的参数

Tips：

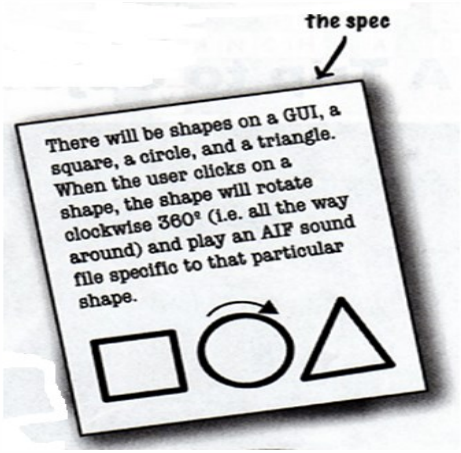
不要编写多种功能集于一身的函数（为了对参数进行更强的确认，要编写**功能单一**的函数）

解决：

- 把realloc分解为四个不同的函数：**扩大内存块**、**缩小内存块**、**分配内存块**和**释放内存块**，我们就能使错误检查的效果更好
- 例如，如果要缩小内存块，我们知道相应的指针必须指向一个有效的内存块，而且新的块长必须小于（也可以等于）当前的块长。除此之外任何东西都是错误的。利用单独的ShrinkMemory函数我们可以通过断言来验证这些参数。

EXAMPLE 5: 另一个设计问题

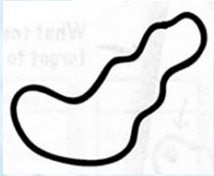
在一个GUI上有3种形状：正方形、圆形、三角形，点击其中某个形状时，使之旋转360度，同时播放一个各形状对应的AIF声音文件。



方法一：面向过程，首先设计出重要的过程框架（旋转和播放函数）

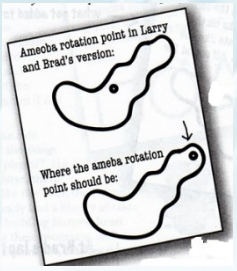
方法二：面向对象，首先为每种形状设计一个类

随着时间的推移……



变化1：加上阿米巴原虫形状：用户点击旋转并播放.hif文件

随着时间的推移……



变化2：阿米巴原虫不是这样转的！

可以预见，“变化”无处不在……

EXAMPLE 6: 一个测试问题

程序TRANGLE输入三个整数，表示一个三角形的三条边长，该程序产生一个结果指出三角形是等腰三角形、等边三角形还是不等边三角形。

怎样设计测试用例（构造哪些测试数据）？

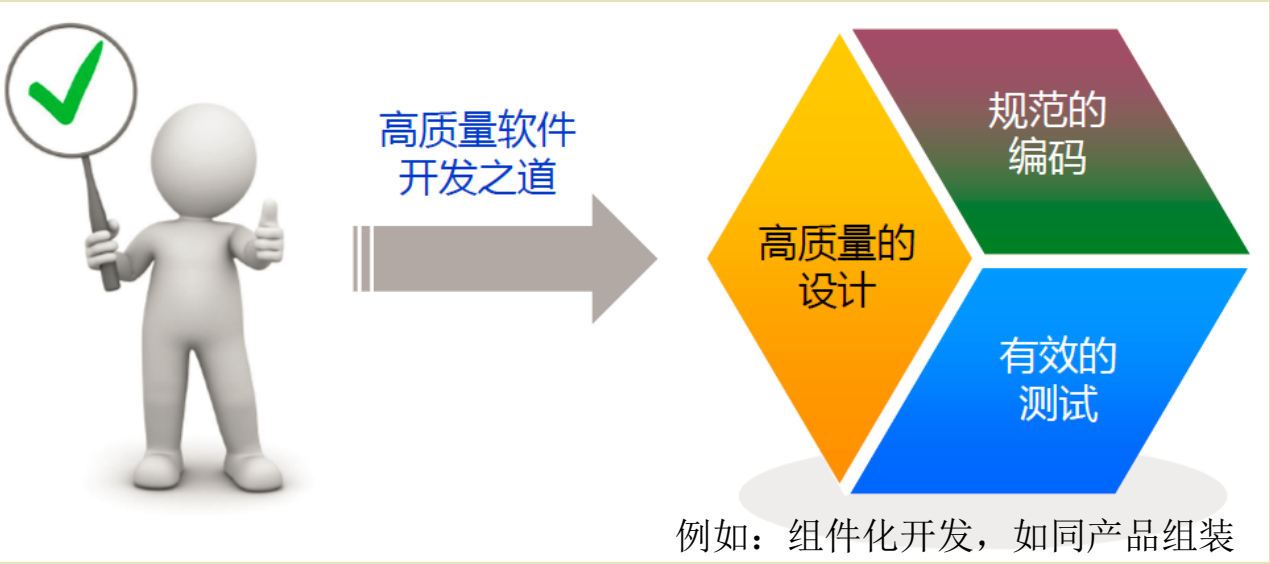
1. 合理的不等边三角形（任意两边之和大于第三边）
2. 合理的等边三角形 （任意两边之和大于第三边）
3. 合理的等腰三角形（任意两边之和大于第三边）
4. 等腰三角形的三种排列次序(3，3，4;3，4，3;4，3，3)
5. 三个正数，其中两个之和等于第三个
6. 上一种情况的三种排列次序
7. 三个正数，其中两个之和小于第三个
8. 上一种情况的三种排列次序
9. 输入数据含有零值
10. 输入数据含有负数
11. 输入数据含有非整数值
12. 三个数均为零
13. 输入数据不是三个数
14. 输入数据中有字符

上述例子的总结

- 1、任何软件的实现方法，都会有*n*种方法
  - 在这*n*种方法中，只有一种或两种是好的，其他的都很差
  - 即使你是用最差的方法实现，也很有可能正常运行
  - 但是：几十年软件发展实践表明
    - 情况并非如此简单，尤其做大型软件，或通用软件……
  - 因为：差的方法是危险的，成本越大后果越严重
    - 如同楼脆脆、烂尾楼、甚至很多已经交付的楼房（哪个更差？哪个更危险？）

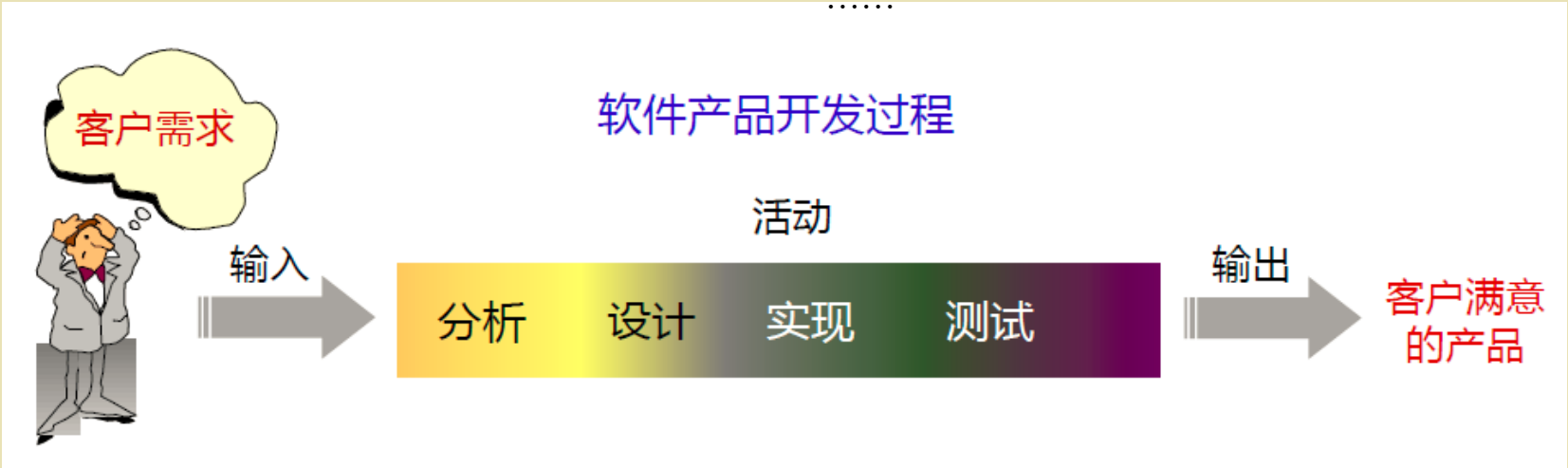
上述例子的总结

- 2、实现高质量软件的途径



例如：组件化开发，如同产品组装

……



- **针对某个问题，怎样选择出一种比较好的方法呢？**
  - 这是所有软件工程师的梦想
  - 软件工程的实践告诉我们：这个问题没有明确的答案
    - 因为：问题几乎涵盖所有领域；其解决方案也难以复制
  - 但是，人们一直没有放弃对正确方法的追寻
    - 思想决定行为
    - 有了正确的思想，就有更大的可能找到一种比较好的方法
- **只有正确的思想，没有正确的方法**
  - 正确的思想可以帮助选择或创造出相对正确的方法，至少避免选择差的方法
  - 不存在万能的方法，最主要的是思想、道理（而非备查的技术资料）
- **软件工程课程**
  - 讲述软件开发的道理（用正确的方法开发软件系统/软件产品）
  - 试图通过方法的讲述，使我们理解方法背后的思想

1. 软件工程做什么？

- 计算机发展初期（20世纪60年代）
  - 计算机　　昂贵而弱智
  - 程序员　　稀少而聪明
  - 软件（程序）　　智力与技能的产品，精巧而难懂
- 计算机快速发展
  - 软件　产品化、复杂化
  - 有成功，更多的是失败
  - 人们开始重视软件质量

- 视角的转变
  - 技术的视角：计算机需要处理的问题就是一个有着明确的定义的问题，通过泾渭分明的“0”和“1”的种种组合、运算，最终给出明确的答复
  - 工程的视角：如果在对技术的使用和构建过程中没有赋予足够的人文方面的重视，技术将毫无价值，甚至是危险的 (technology is worthless - even dangerous - if we don‘t pay attention to the human aspects of both its use and its construction ---《Are your lights on?》Donald.C.Gause, Gerald.M.Weinberg)

- 基本观点
  - 工程的基本观点是：要解决问题
  - 软件工程的基本观点是：用软件来解决用户的问题，软件只是一个手段
  - 解决方案：软件是其中一个组成部分，重要但不是全部
- 基本内容
  - 讲述软件开发的道理（用正确的方法开发软件系统/软件产品）
  - 其内容涉及软件开发的方方面面：人员、计划、成本、进度、风险、设计、技术路线、培训、维护……

软件开发的基本策略

软件复用

- 构造一个新的系统不必从零做起，直接复用已有的构件进行组装
- 构件是经过反复使用验证的，由其组成的新系统具有较高的质量

分而治之

- 将一个复杂的问题分解成若干个简单的问题，然后逐个解决
- 来源于人们生活与工作的经验，完全适合于技术领域

逐步演进

- 软件开发是自底向上逐步有序的生长过程
- 小步快跑：每走完一步再调整并为下一步确定方向，直到终点

组件化开发以及演进：比产品组装更复杂……

优化折中

- 优化：优化软件的各个质量特性，如运行速度、资源利用、用户体验
- 折中：通过协调各个质量特性，实现整体质量的最优

2. 软件工程教给我们什么？

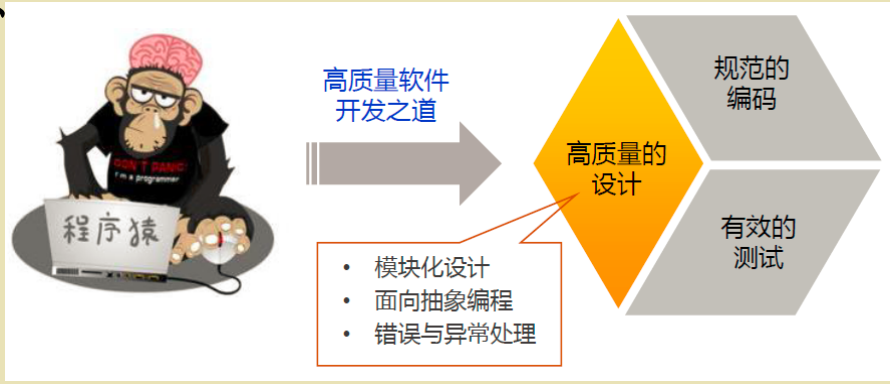
- 软件产品生产中的一些道理
  - 大多数人都希望学到一些招数、方法，能尽快在工作中用上，这并不错（学招式：主要靠自己，学习+实践）
  - 要想达到更高境界，就须明白背后的道理（学内功：好理论指导和个人的领悟、积累）
- 明白道理，才能知变通之道
  - 软件开发中，每个团队、每个项目都不是尽同的
  - 世界“虽变化万端，而理为一贯”。只有招数，不明道理，碰到变化的情况，或者照猫画虎，或者束手无策

- 对于同样一个问题
  - 初级水平程序员：首先考虑如何使用某种编程语言实现，VB、VC……，看起来效果不错。
  - 高级水平程序员：首先考虑问题的本质（分析），然后是具有最快效率、最稳定性能的解决问题的方法（思路），最后才会考虑选择合适的开发工具
  - 比如，数据结构、算法不会依赖于实现语言。

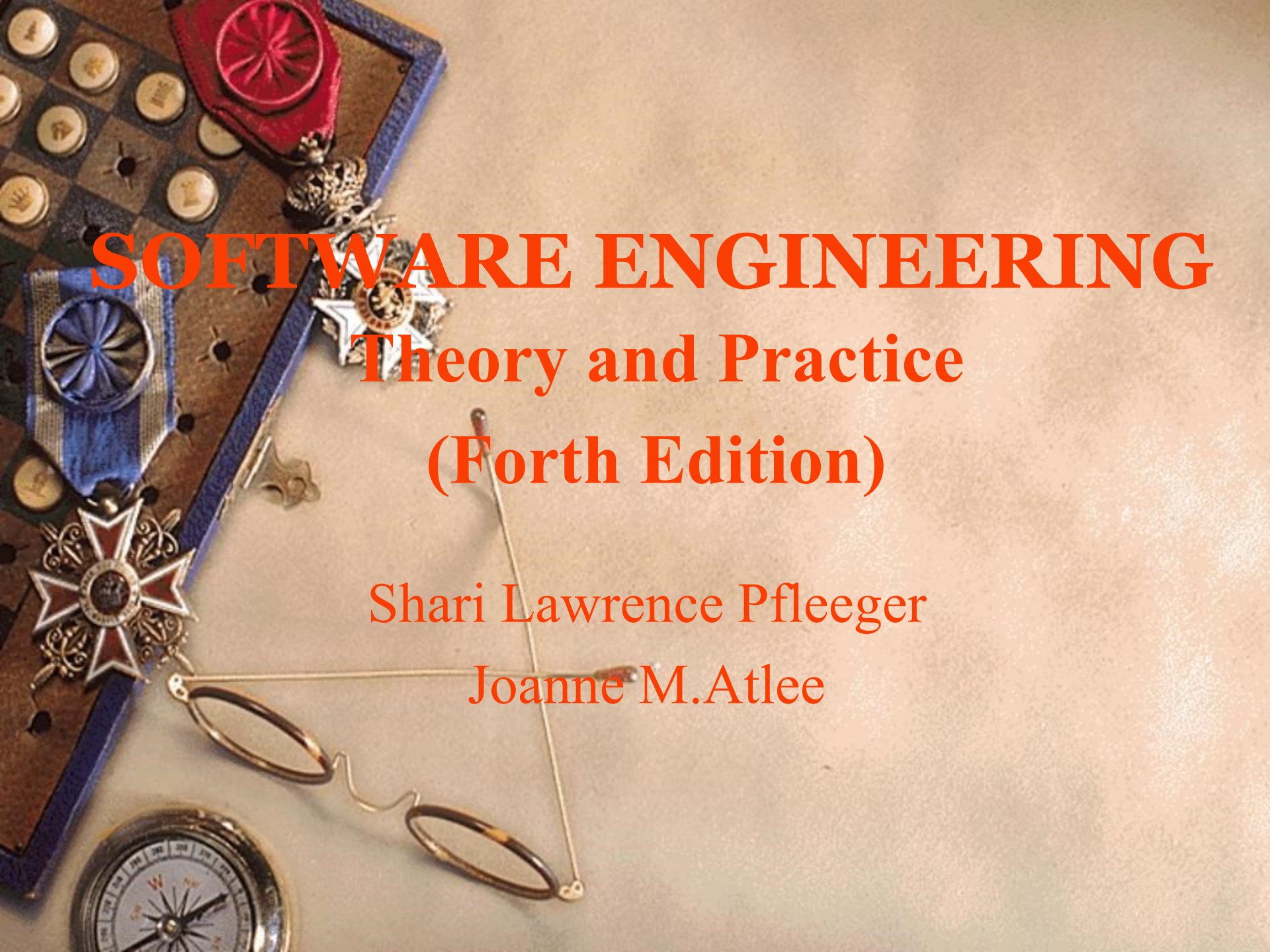
3. 如何成为一个好的软件工程师

- Foundational Competency of an IBMer
  - Drive to Achieve 自我驱策
  - Adaptability 适应能力
  - Client Focus 客户为尊
  - Communication 沟通能力
  - Creative Problem Solving 创意解题
  - Passion for the Business 工作热忱
  - Taking Ownership 勇于负责
  - Trustworthiness 值得信赖

- 团队精神：纪律、协作、责任
- 个人能力：编程能力、设计能力、分析能力、表达（交流）能力
  - 练习表达能力，无论是书面表达还是口头表达，能够清晰、自如、具有说服力地传达观点
  - 一个普通程序员与优秀程序员的区别，不在于懂得编程语言多少，也不在于用Python还是Java语言，而在于能否与他人交流思想，比如能说服其他人，能写出清晰的注释和技术规格说明书
  - 建议多选修“写作密集型”（writing intensive）课程，锻炼良好的写作能力
  - 动手写日记或者网志



- 态度：敬业（努力）、耐心、专一
  - 时刻跟随技术前进的步伐，时刻掌握最新的技术（保持最小的压力）
  - 保持正常的心态。正常的生活、正常的节奏、正常的思路，只有正常的东西才能长久
- 其他
  - 学好C/C++语言
  - 重视GPA成绩
  - 找一份好的暑期实习工作
    - 不要轻易接受太多与编程无关的事情，等到毕业的时候，简历上应该写满了一大堆与编程相关的经历



SOFTWARE ENGINEERING

Theory and Practice

(Forth Edition)

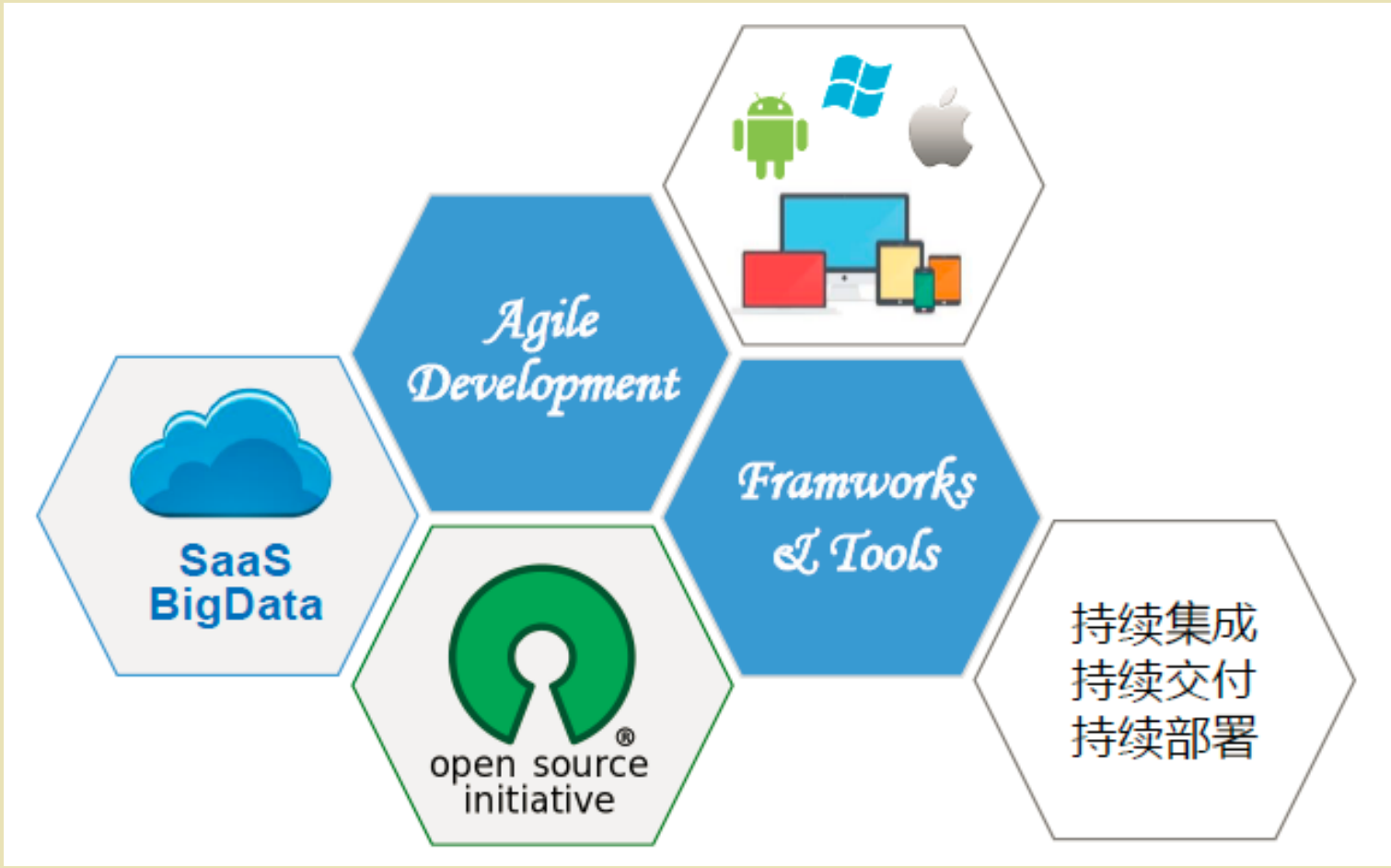
Shari Lawrence Pfleeger

Joanne M.Atlee

Two Aspects of Software Engineering

- 实践者
  - 构建用于解决问题的优质产品。
- 研究者
  - 研究改进产品质量和提高产品生产率的方法。
- 理论和实践之间的桥梁
  - 对问题本质进行抽象和建模
  - 设计解决方案

互联网时代的软件开发


Agile Development
Framworks & Tools
SaaS BigData
open source initiative
持续集成
持续交付
持续部署

CONTENTS AND ORGANIZATION

- **Part I Why software engineering is important and What;Process modeling;Project plan&management**
  - CH1:Why Software Engineering ?
  - CH2:Modeling the Process and Life Cycle
  - CH3:Planning and Managing the Project
- **Part II major steps of software development**
  - CH4:Capturing the Requirements
  - CH5:Designing Architecture
  - CH6:Designing Modules
  - CH7:Writing the Programs

CONTENTS AND ORGANIZATION

- **Part II (continue)**
  - CH8:Testing the Programs
  - CH9:Testing the System
  - CH10:Delivering the System
  - CH11:Maintaining the System
- **Part III**
  - CH12:Evaluating Products, Processes and Resources
  - Other chapters will be omitted

THE NATURE OF SE

- **Technical aspects, include :**
  - Concepts and descriptions for software structures
  - Concepts and descriptions for software development design and implementation
  - Software tests and verification
  - Software development tools
- **Management aspects, include :**
  - Concepts model about the software lifecycle
  - Concepts about software development process
  - Software quality and standards
- **It is the both, with Management emphasized in this book.**

课程考察方式

- 笔试考核　　70分
  - PPT、课堂内容
- 实验/展示成绩　　20分
  - 分组讲解展示，至少1次/小组成员
  - 实验安排：
- 平时成绩　　10分
  - 出勤情况&作业情况

PREFACE

一、Software=Program+Data+Documen

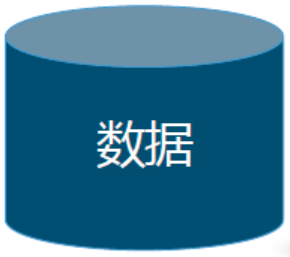
Program ： Executable （for machine）

Data ： Data Structure & Date

Document： About development,running, using,trainning, maintenance…（for human）

软件 = 程序 + 数据 + 文档

- 程序：计算机可以接受的一系列指令，运行时可以提供所要求的功能和性能。
- 数据：使得程序能够适当地操作信息的数据结构。
- 文档：描述程序的研制过程、方法和使用的图文资料。



二、The Developing of Software

Phase 1：Program Coding 程序编码阶段

Developer ： Personal

App Domain： science compute…

Time ： 20^{th}50s—20^{th}60s

With the decrease of hardware cost&price and the increase of software requirement on more and more app domain.

Phase 2：Program System 程序系统阶段

Developer ： Software Workshop

App Domain： more wide…

Time ： 20th60s—20th70s’end

Software Crisis: 软件危机

Serious problems in the process of software development.

Software quality; developing progress; cost…

1968,NATO,”Software Engineering”,

more than 100 principles.

Phase 3：Software Engineering 软件工程

Developer ： Software Company

App Domain： more wide…

Time ： 20th70s—20th80s middle

More and more complex of program domain;
e.g. Personal→T/H→C/S→C/S/S…
Market pressure…

Phase 4：Developing of Software Engineering

Developer :Software Company

App Domain:more wide…

Time :20th80s middle—

Content :Distributed System;

O-O theories & technologies;

Process improvement&CMM

三、Software Engineering Definition:

◆软件工程是将系统性的、规范化的、可定量的方法应用于软件的开发、运行和维护，即工程化应用到软件生产过程。

◆Using principles and methods of engineering, mathematical, scientific to develop&maintenance software, manage the software developing process.

◆ Software+engineering: By integrating new or exists components, software with high-quality are developed within time limit and cost budget.

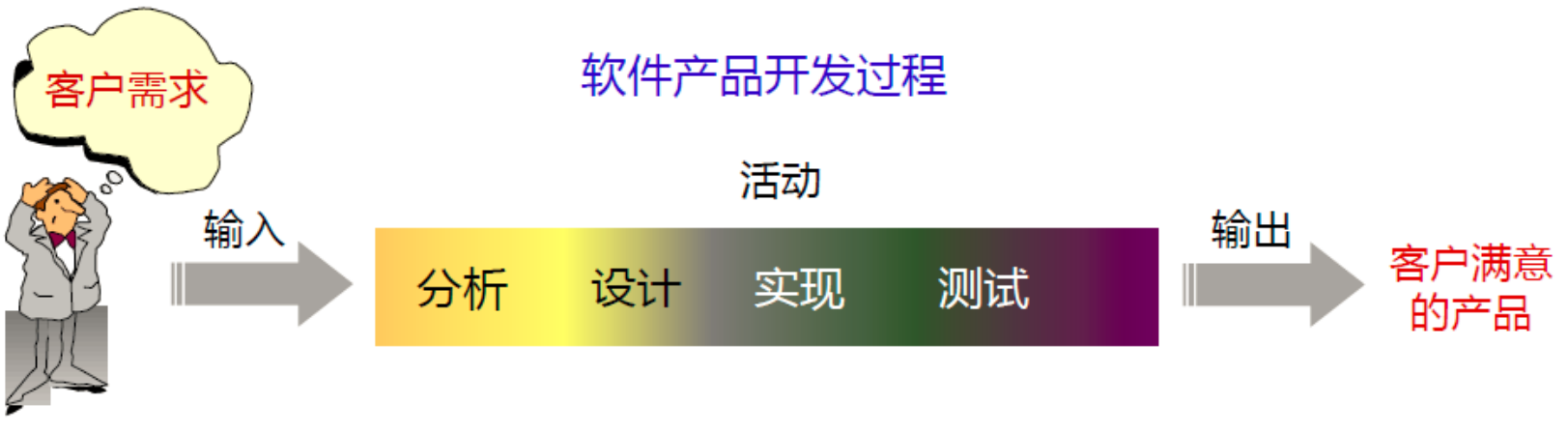
◆To solve customers' problem with computer technologies (technology+tool+procedure+paradigm).

Objective:

improve software quality;
improve software productivity.

Activities:

requirement elicit and analysis;
system design;
technical design;
coding;
program testing;
system testing;
deliver;



四、Goal of Software Engineering

- Improve software quality
- Raise software productivity
- Reduce cost of software development

Namely,

To develop high quality software within time limit and at budgetary cost.

五、Software Engineering Concept with UML class diagram(OMG,1998)