软件工程专业导论

复习

1. 一般而言，设计和实现一个计算系统，需要设计和实现_____？

- A.基本动作、控制基本动作的指令和一个程序执行机构
- B.基本动作和程序
- C.基本动作和控制基本动作的指令
- D.基本动作、控制基本动作的指令和程序

复习

1. 一般而言，设计和实现一个计算系统，需要设计和实现_____？

- [√] A.基本动作、控制基本动作的指令和一个程序执行机构
- [ ] B.基本动作和程序
- [ ] C.基本动作和控制基本动作的指令
- [ ] D.基本动作、控制基本动作的指令和程序

复习

2. 已知一个新运算被定义为(define (newCalc x y) (* (+ x 1) (* y 2)))，问newCalc可以完成的计算功能为_____？

- A. (x+1)*2y
- B. (x+1)+2y
- C. (x+1) +(y+2)
- D. (x+1)*(y+2)

复习

2. 已知一个新运算被定义为(define (newCalc x y) (* (+ x 1) (* y 2)))，问newCalc可以完成的计算功能为_____？

- [√] A. (x+1)*2y
- [ ] B. (x+1)+2y
- [ ] C. (x+1) +(y+2)
- [ ] D. (x+1)*(y+2)

复习

3. 关于递归定义的函数，下列说法正确的是_____？

- A.递归定义的函数一定是“递归计算”的
- B.有些递归定义的函数可以“迭代计算”，有些递归定义的函数则必须“递归计算”
- C.凡是可以“迭代计算”的函数，一定可以“递归计算”，凡是可以“递归计算”的函数，也一定可以“迭代计算”
- D.递归定义的函数一定是“迭代计算”的

复习

3. 关于递归定义的函数，下列说法正确的是_____？

- [ ] A.递归定义的函数一定是“递归计算”的
- [√] B.有些递归定义的函数可以“迭代计算”，有些递归定义的函数则必须“递归计算”
- [ ] C.凡是可以“迭代计算”的函数，一定可以“递归计算”，凡是可以“递归计算”的函数，也一定可以“迭代计算”
- [ ] D.递归定义的函数一定是“迭代计算”的

复习

4. 阿克曼函数如下所示，计算机结果正确的是_____？

$$A(m,n)=\begin{cases} n+1 & \text{若} m=0 \\ A((m-1),1) & \text{若} n=0 \\ A(m-1,A(m,n-1)) & \text{若} m,n>0 \end{cases}$$

A(1,2) = A(0,A(1,1)) = A(0, A(0,A(1,0))) = A(0, A(0,A(0,1)))=A(0,A(0,2))=A(0,3)=4

- A. A(2, 0) = 2
- B. A(1, 8) = 9
- C. A(2, 1) = 4
- D. A(1, n) = n + 2

复习

4. 阿克曼函数如下所示，计算机结果正确的是_____？

$$A(m,n)=\begin{cases} n+1 & \text{若} m=0 \\ A((m-1),1) & \text{若} n=0 \\ A(m-1,A(m,n-1)) & \text{若} m,n>0 \end{cases}$$

A(1,2) = A(0,A(1,1)) = A(0, A(0,A(1,0))) = A(0, A(0,A(0,1)))=A(0,A(0,2))=A(0,3)=4

- ☐ A. A(2, 0) = 2
- ☐ B. A(1, 8) = 9
- ☐ C. A(2, 1) = 4
- ☑ D. A(1, n) = n + 2

复习

5.已知一个运算被定义为(define (firstCalc x) (* x x))，在其基础上进一步定义新运算为

(define (secondCalc x) (firstCalc (firstCalc (firstCalc x))))

问secondCalc表达的运算功能为_____？

- A. $((x^2)^2)^2$
- B. $x^2+x^2+x^2$
- C. x^4
- D. $x*x*x$

复习

5.已知一个运算被定义为(define (firstCalc x) (* x x))，在其基础上进一步定义新运算为

(define (secondCalc x) (firstCalc (firstCalc (firstCalc x))))

问secondCalc表达的运算功能为_____？

- [√] A. $((x^2)^2)^2$
- [ ] B. $x^2+x^2+x^2$
- [ ] C. x^4
- [ ] D. $x*x*x$

复习

6.关于TSP问题的遍历算法和贪心算法，下列说法正确的是_____？

- A.对TSP问题而言，遍历算法和贪心算法求得的解是一样的，所不同的是贪心算法更快一些，而遍历算法更慢一些
- B.对TSP问题而言，遍历算法和贪心算法求得的解是不一样的，贪心算法是求近似解，执行更快一些，而遍历算法是求精确解，执行更慢一些
- C.对TSP问题而言，遍历算法和贪心算法求得的解是一样的，所不同的是遍历算法更快一些，而贪心算法更慢一些
- D.对TSP问题而言，遍历算法和贪心算法求得的解是不一样的，贪心算法是求精确解，执行更快一些，而遍历算法是求近似解，执行更慢一些

复习

6.关于TSP问题的遍历算法和贪心算法，下列说法正确的是_____？

☐ A.对TSP问题而言，遍历算法和贪心算法求得的解是一样的，所不同的是贪心算法更快一些，而遍历算法更慢一些

√ B.对TSP问题而言，遍历算法和贪心算法求得的解是不一样的，贪心算法是求近似解，执行更快一些，而遍历算法是求精确解，执行更慢一些

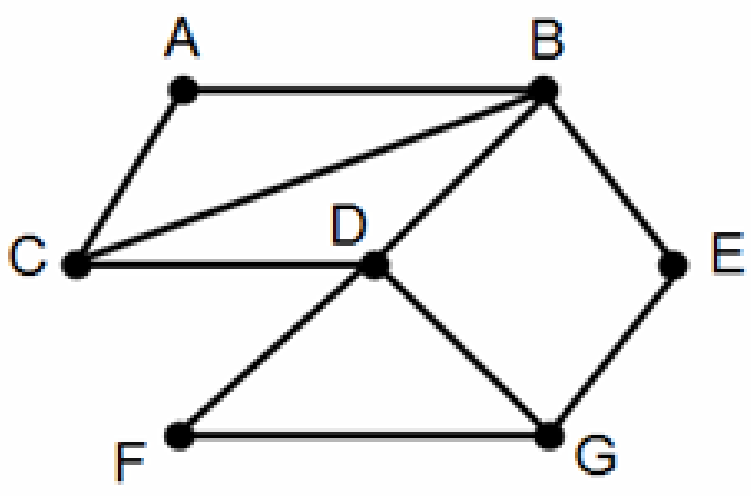
☐ C.对TSP问题而言，遍历算法和贪心算法求得的解是一样的，所不同的是遍历算法更快一些，而贪心算法更慢一些

☐ D.对TSP问题而言，遍历算法和贪心算法求得的解是不一样的，贪心算法是求精确解，执行更快一些，而遍历算法是求近似解，执行更慢一些

复习

7.哥尼斯堡七桥问题，增加哪些边，使得能够找到走遍每一座桥，且每座桥仅走过一次、最后又回到原出发点的路径？

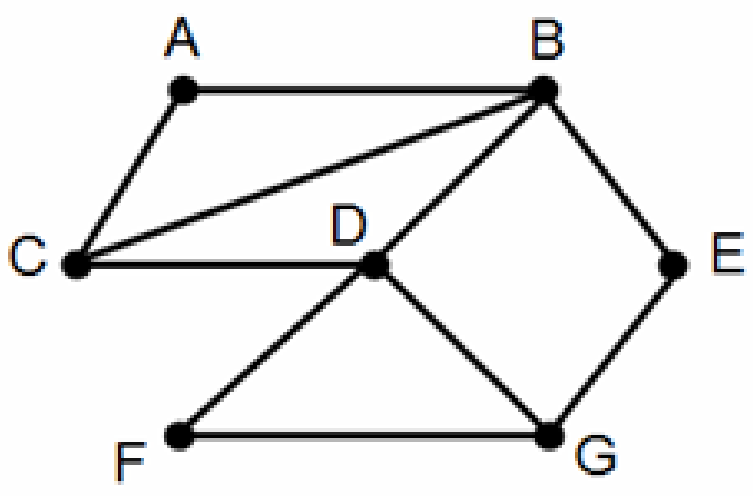
- A. BG边
- B. AG边
- C. AD边
- D. CG边



复习

7.哥尼斯堡七桥问题，增加哪些边，使得能够找到走遍每一座桥，且每座桥仅走过一次、最后又回到原出发点的路径？

- ☐ A. BG边
- ☐ B. AG边
- ☐ C. AD边
- ☑ D. CG边



复习

8.算法的时间复杂性，可以表达为关于问题规模n的一个函数T(n)，T(n)可以用大O表示法来处理。问T(n)=O(f(n))是什么意思？正确的是_____?

- A. T(n)是关于f(n)的一个函数
- B. T(n)是将函数f(n)代入O(x)中所形成的新函数
- C. T(n)是与f(n)同数量级的函数
- D. T(n)是依据f(n)计算出来的

复习

8.算法的时间复杂性，可以表达为关于问题规模n的一个函数T(n)，T(n)可以用大O表示法来处理。问T(n)=O(f(n))是什么意思？正确的是_____?

- [ ] A. T(n)是关于f(n)的一个函数
- [ ] B. T(n)是将函数f(n)代入O(x)中所形成的新函数
- [√] C. T(n)是与f(n)同数量级的函数
- [ ] D. T(n)是依据f(n)计算出来的

复习

9. 关于下列流程图，说法正确的是_____？

- A. 都错
- B. (a)正确
- C. (b)正确
- D. (c)正确

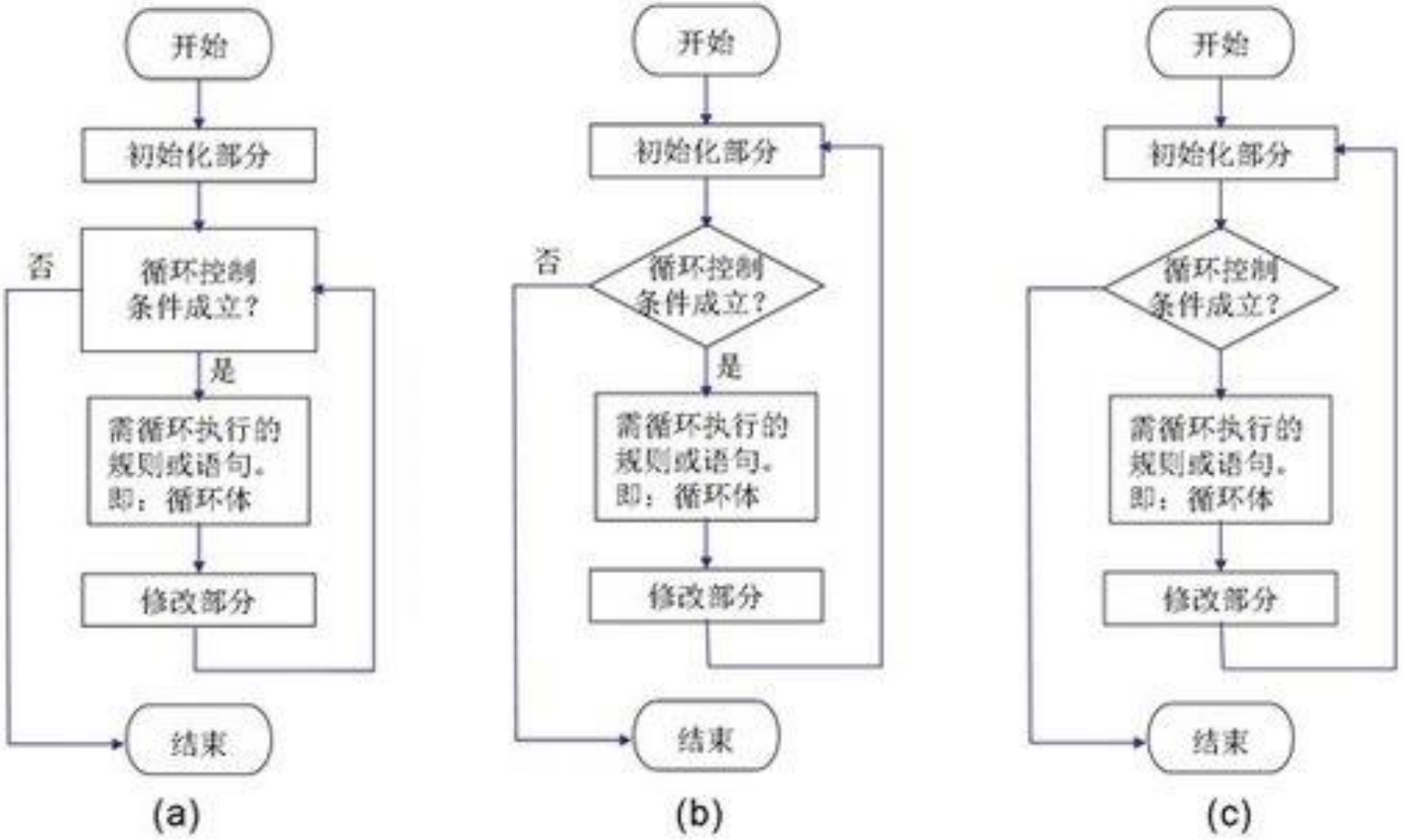


图 I.

复习

9. 关于下列流程图，说法正确的是_____？

- [√] A. 都错
- [ ] B. (a)正确
- [ ] C. (b)正确
- [ ] D. (c)正确

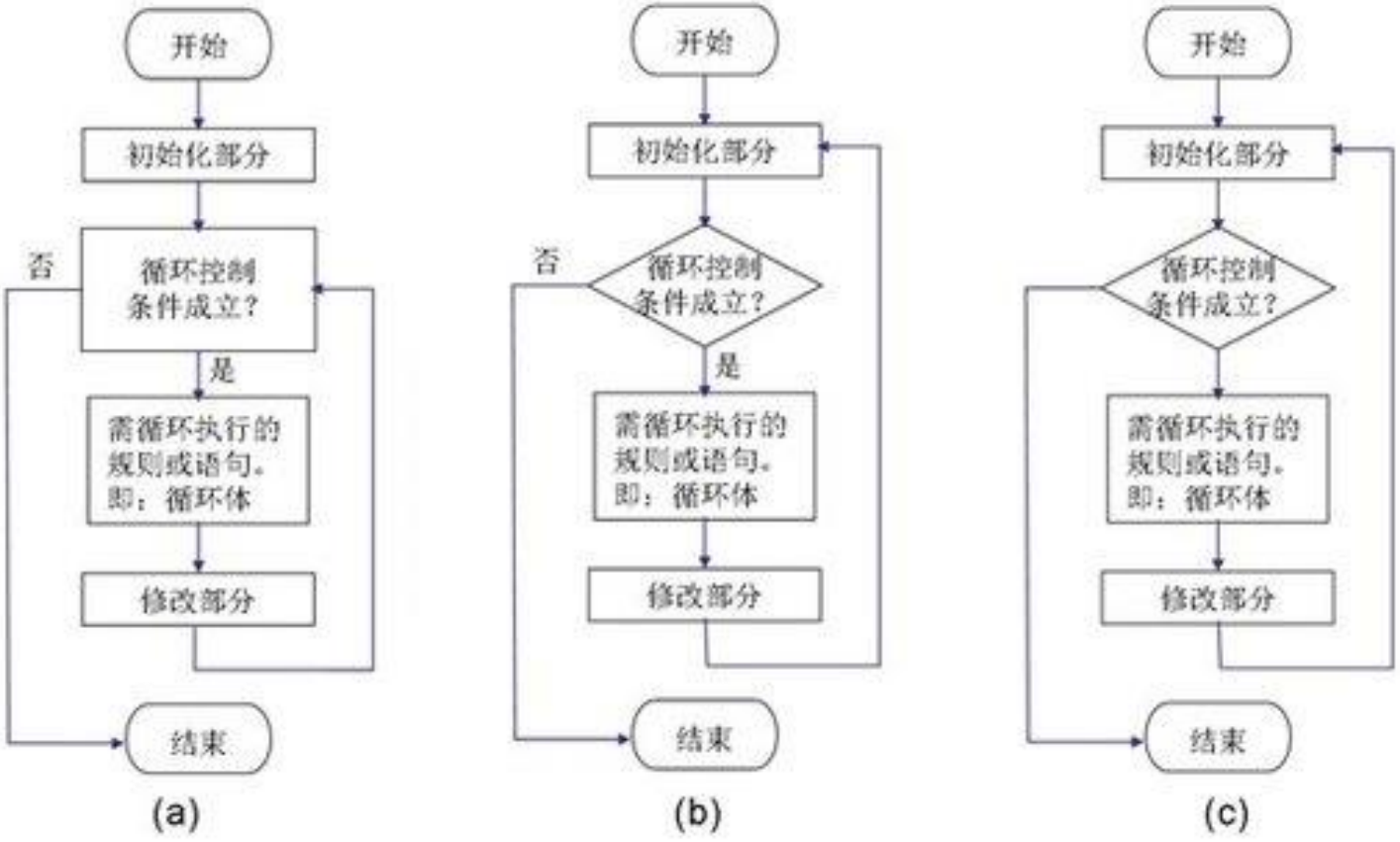


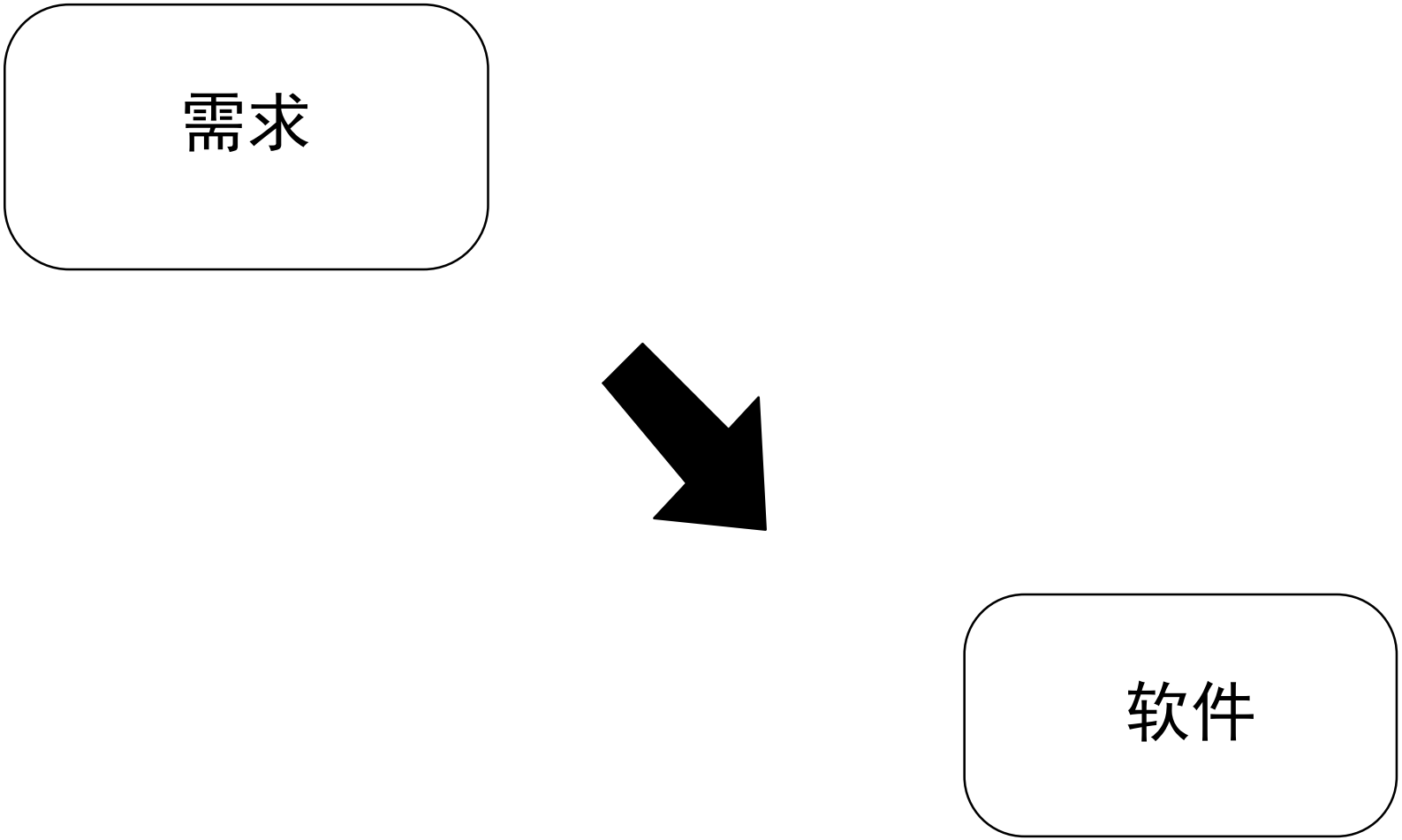
图 I.

软件生命周期模型

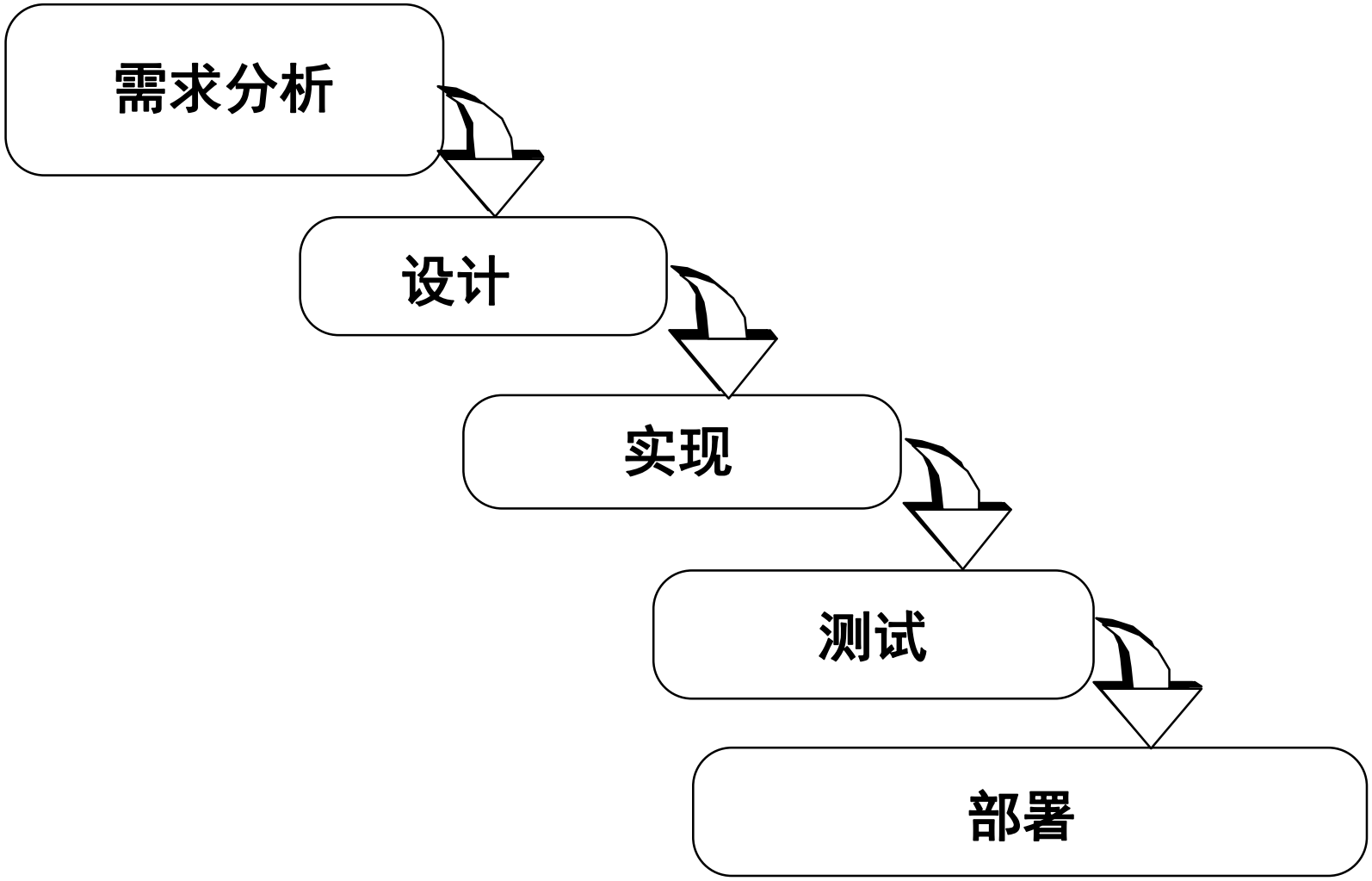
学习目标：

- 了解软件的复杂性
- 了解软件的生命周期
- 了解软件的生命周期模型

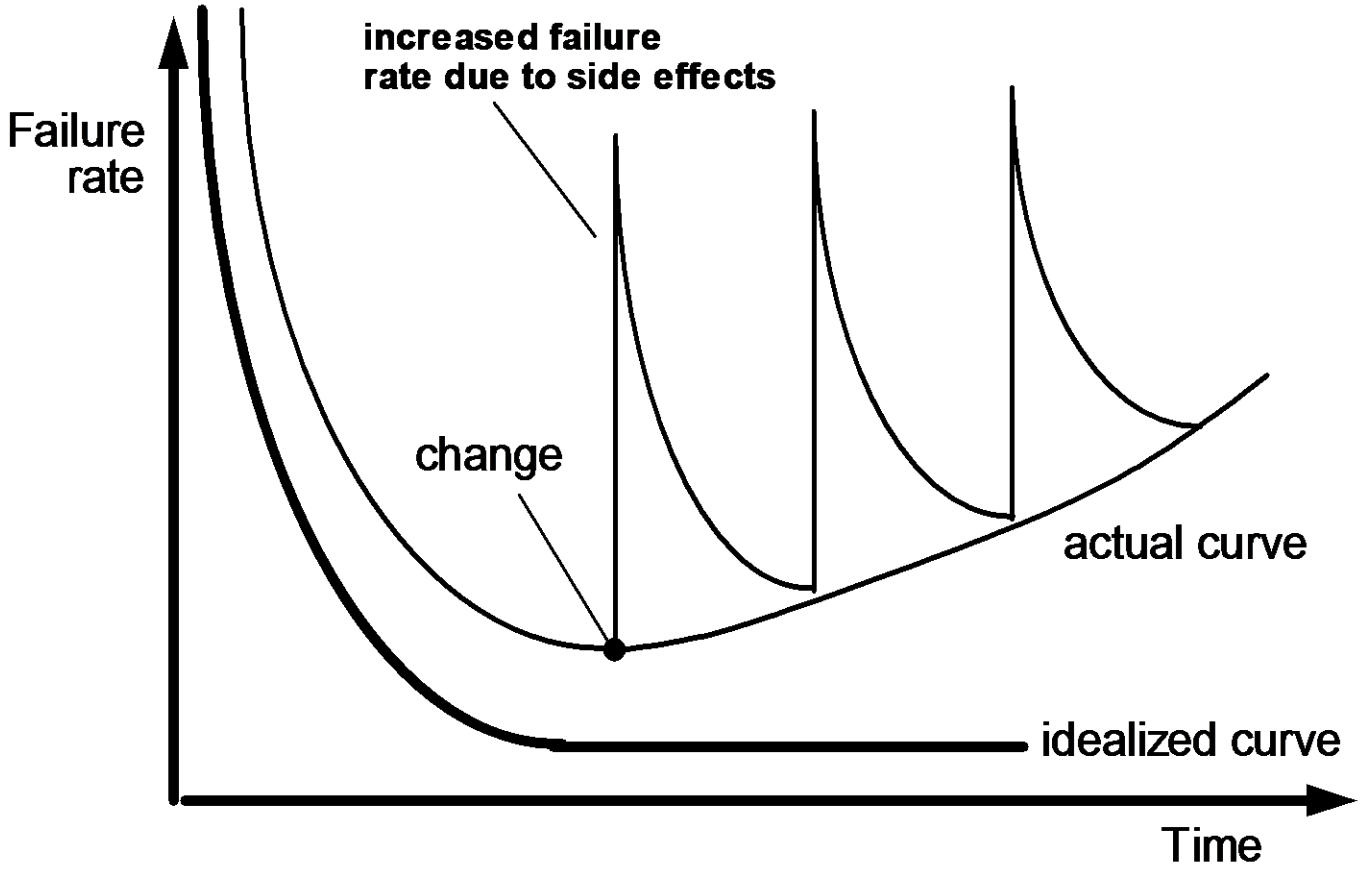
1. 软件的复杂性



1. 软件的复杂性



1. 软件的复杂性


increased failure
rate due to side effects
Failure
rate
change
actual curve
idealized curve
Time

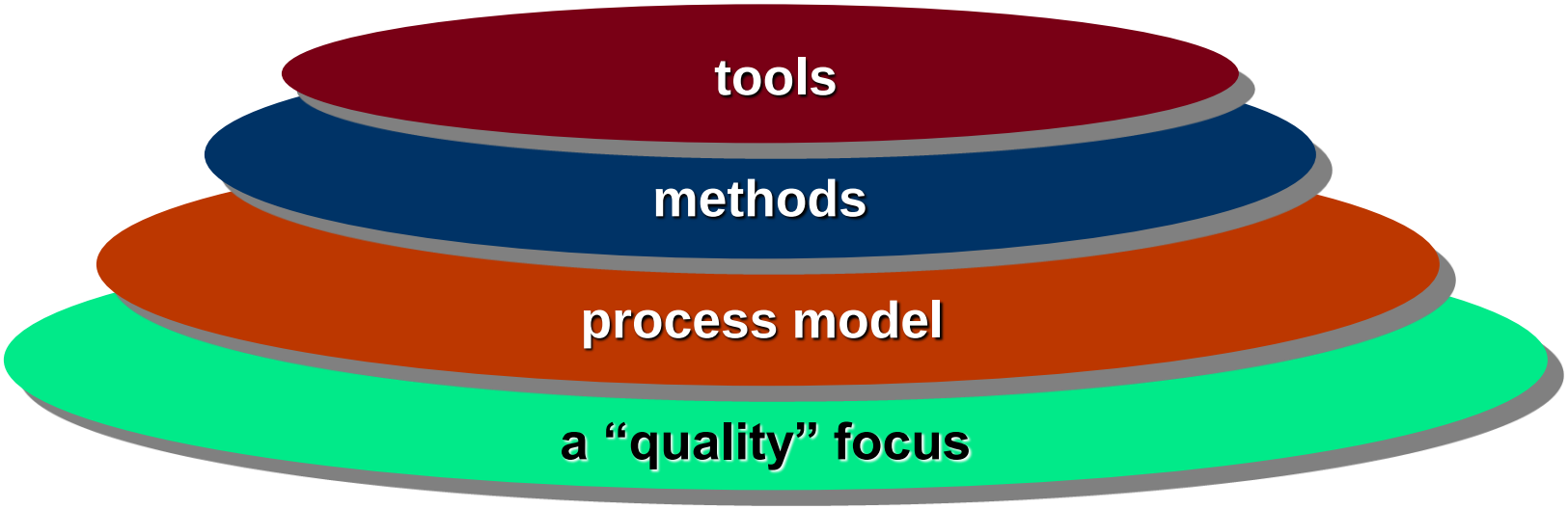
1. 软件的复杂性

- 为什么会变化？
  - 变化：以满足新计算环境或技术的需求；
  - 变化：以实现新的业务需求
  - 变化：以与更现代的系统或数据库互操作
  - 变化：以使其在网络环境中运行
  - ……

1. 软件的复杂性

- 软件工程（Software Engineering）：研究或应用工程化方法来设计、创造、构建和维护有效、实用和高质量软件的一门学科
- IEEE CS定义：软件工程是①将系统化的、严格约束的、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对于上述方法的研究

1. 软件的复杂性



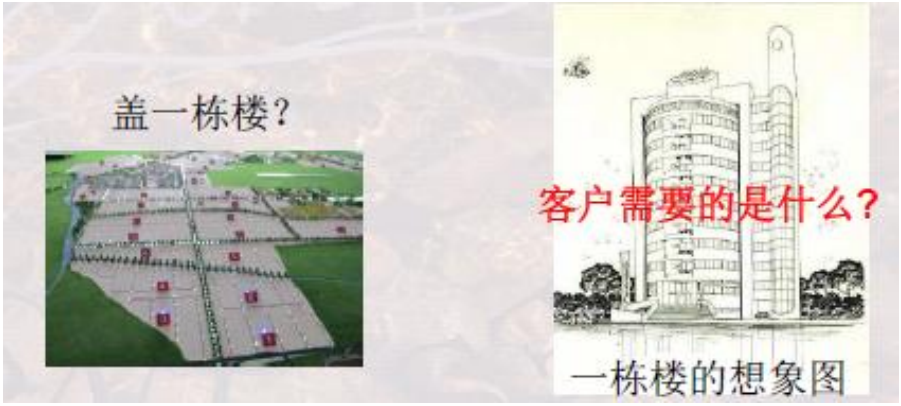
Software Engineering

2. 软件开发的各个阶段

需求分析	问题是什么?	问题域
系统设计	解决方案是什么?	
程序设计	实现解决方案的机制是什么?	实现域
程序实现	如何构建解决方案?	
测试	问题是否已解决?	
交付	用户是否可以使用?	
维护	需要改进和增加功能吗?	

2. 软件开发的各个阶段

- 需求分析是指对解决现实世界某个问题的软件产品的描述，及对软件产品约束的描述
  - 软件需求“表达了施加在要解决真实世界问题的软件产品上的要求和约束”---from SWEBOK
  - 说明将要做的软件是什么(What to do?, to what degree?)



如何使不同人员(如业务人员、设计人员、编程人员等)准确理解客户的需求

课程-“需求工程” “系统分析与设计”

2. 软件开发的各个阶段

- 软件设计是指要表达出怎样做出规定的软件，即将需求转换为可以实现的技术方案
  - 怎样做软件?怎样达到规定的程度(How to do? How to reach the degree?)
  - 设计是软件工程最核心的内容。设计既是“过程”，也是这个过程的“结果”
  - 编制软件设计说明书

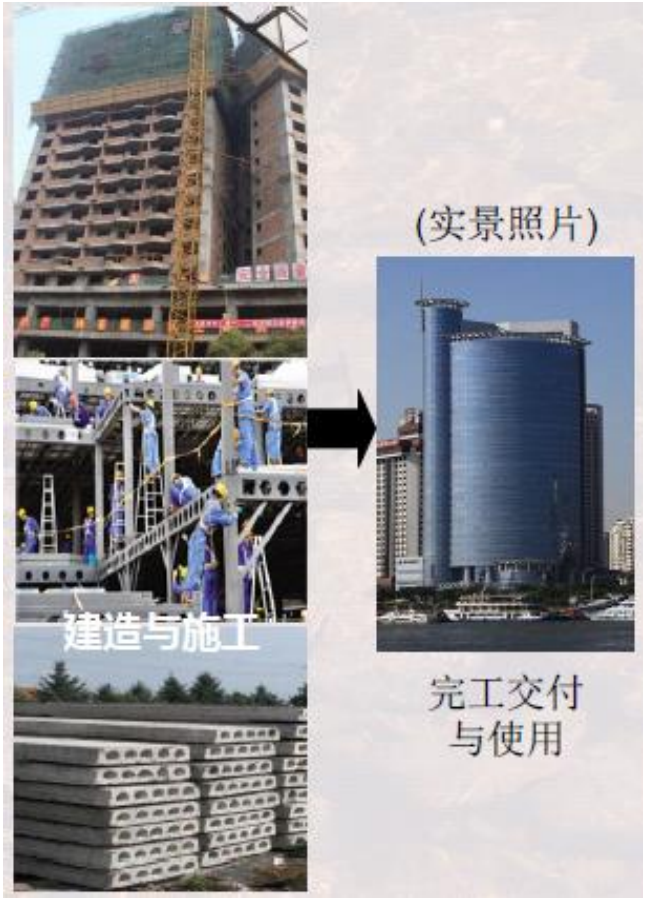
2. 软件开发的各个阶段

- 软件设计是指要表达出怎样做出规定的软件，即将需求转换为可以实现的技术方案
  - 总体设计：也称概要设计，把各项需求转换成软件的体系结构。结构中每一组成部分都是意义明确的模块，每个模块都和某些需求相对应
  - 详细设计：对每个模块要完成的工作进行具体描述，为源程序编写打下基础

课程-“系统分析与设计”“面向对象的分析与设计”“软件体系结构”“软件设计模式”“软件工程”

2. 软件开发的各个阶段

- 软件编码，把软件设计转换成结构良好、清晰易读的，且与设计相一致的计算机可以接受的程序代码即写成以某一种特定程序设计语言表示的“源程序清单”
  - 提交“源程序”与“可执行程序模块



课程- “面向对象的程序设计” “软件设计模式”
“J2EE程序开发” “.Net程序开发”

2. 软件开发的各个阶段

- 软件测试：测试软件是否满足规定的功能和性能，是否存在问题
  - 测试是软件生命周期的重要组成部分。测试的目的是及早发现软件所存在的问题，避免出现缺陷导致事故
  - 产生软件评测报告
  - 单元测试：检测各模块在功能和结构上存在的问题并加以纠正
  - 组装测试：将已测试过的模块按一定顺序组装起来，按规定的各项需求，逐项进行有效性测试，决定已开发的软件是否合格，能否交付用户使用

课程-“软件测试技术”

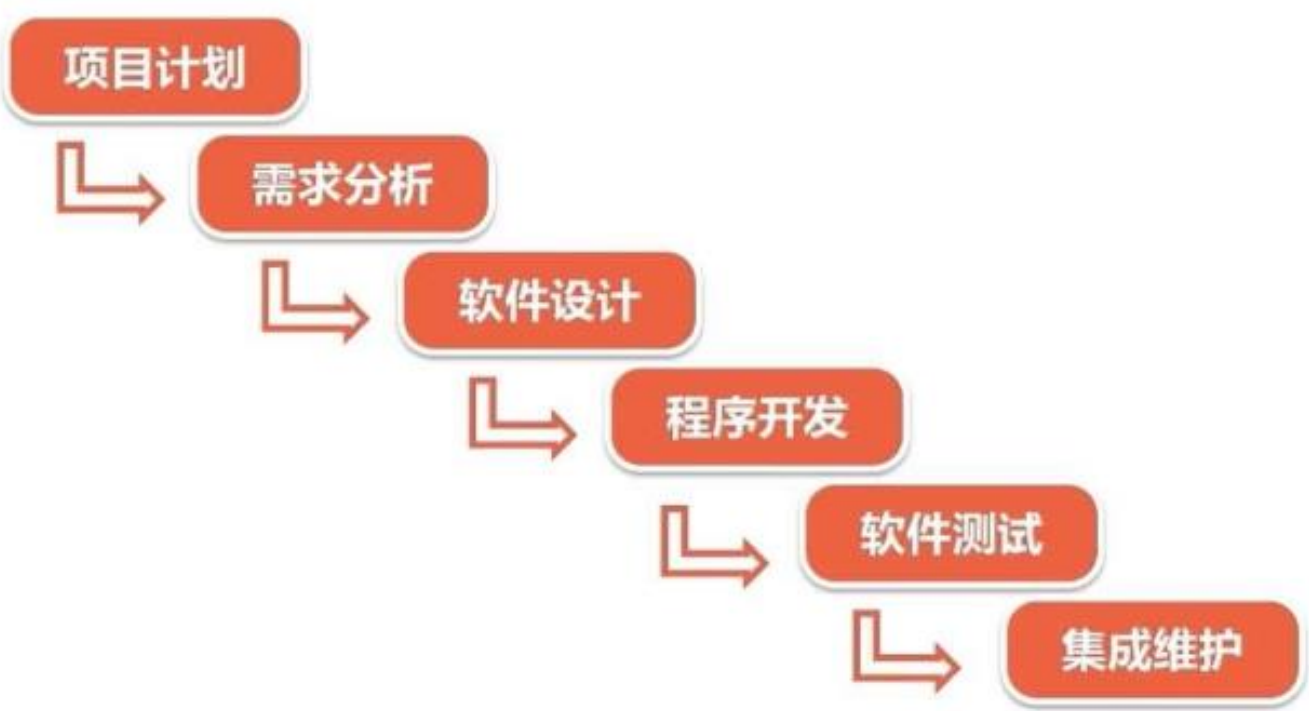
2. 软件开发的各个阶段

- 软件运行与维护：如何改正软件运行过程中发现的缺陷、如何提高软件性能或其他属性、如何使软件产品适应新的环境
  - 改正性维护：运行中发现了软件中的错误需要修正
  - 适应性维护：为了适应变化了的软件工作环境，需做适当变更
  - 完善性维护：为了增强软件的功能需做变更

课程-“软件维护与软件演化技术”

3. 软件的生命周期模型

- 瀑布模型（Waterfall Model）

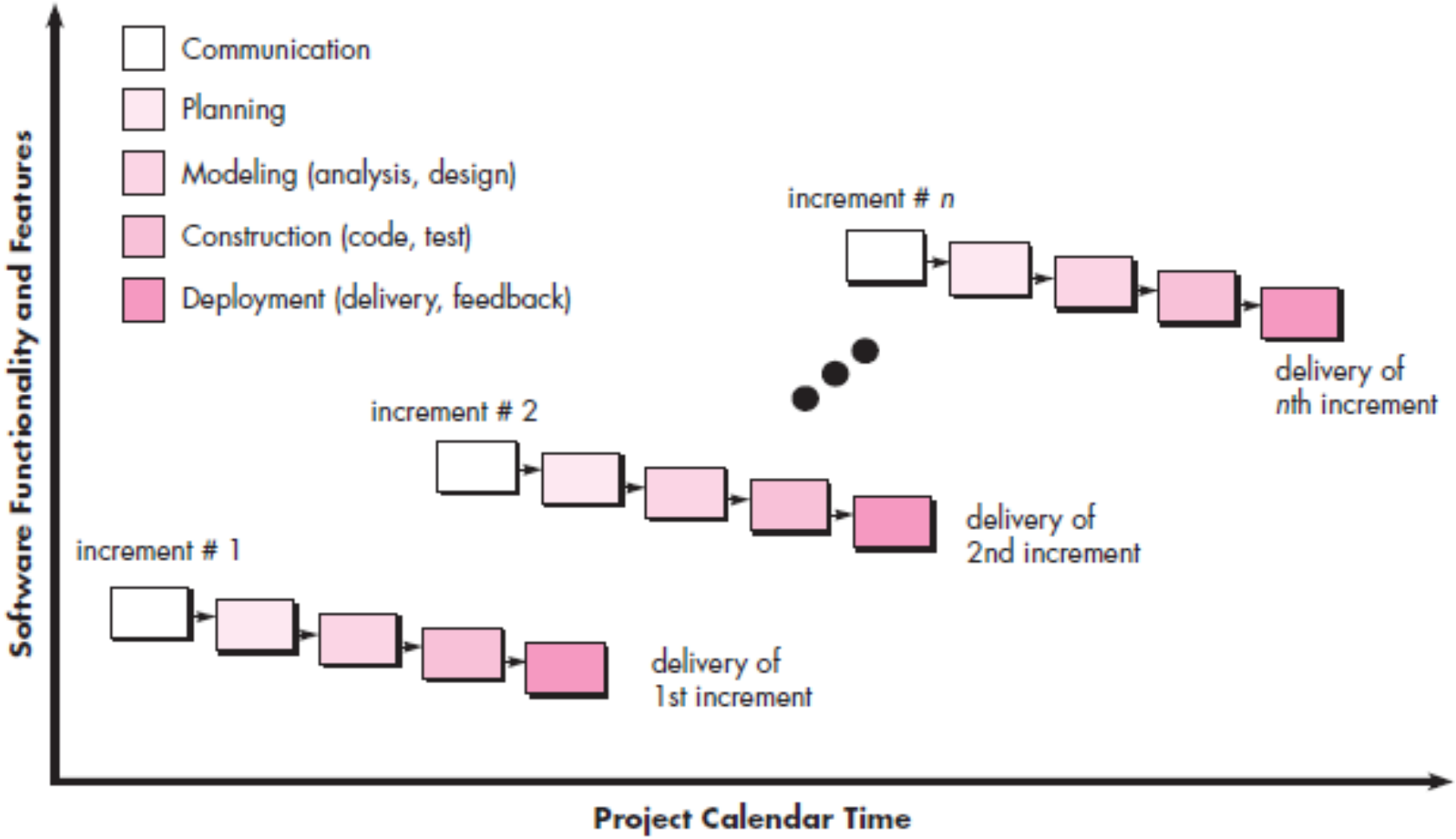


3. 软件的生命周期模型

- 瀑布模型（Waterfall Model）存在的问题
  - 真实项目很少遵循模型提出的顺序流程
  - 用户通常很难明确所有需求，瀑布模型难以适应许多项目开始时存在的不确定性
  - 如果重大的错误在程序交付前才被发现，可能是灾难性的

3. 软件的生命周期模型

- 增量模型（Incremental Process Model）

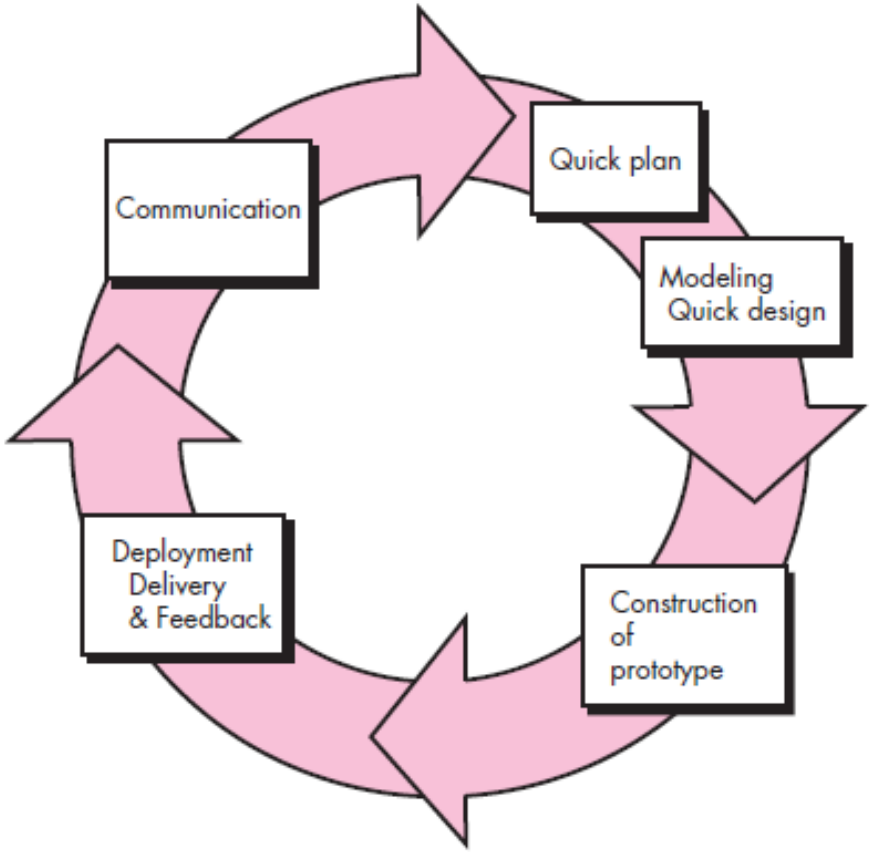


3. 软件的生命周期模型

- 增量模型（Incremental Process Model）
  - 例如，文字处理软件
    - 第一版：基本的文件管理，编辑和文档生成功能
    - 第二版：更复杂的编辑和文档制作功能
    - 第三版：拼写和语法检查
    - 第四版：高级页面布局功能
    - ……
  - 侧重于每个增量交付可运行产品
  - 早期增量是最终产品的精简版，但确实为用户提供了服务，并提供了一个用户评估的平台

3. 软件的生命周期模型

- 进化模型（Evolutionary Process Model）：原型（Prototyping）

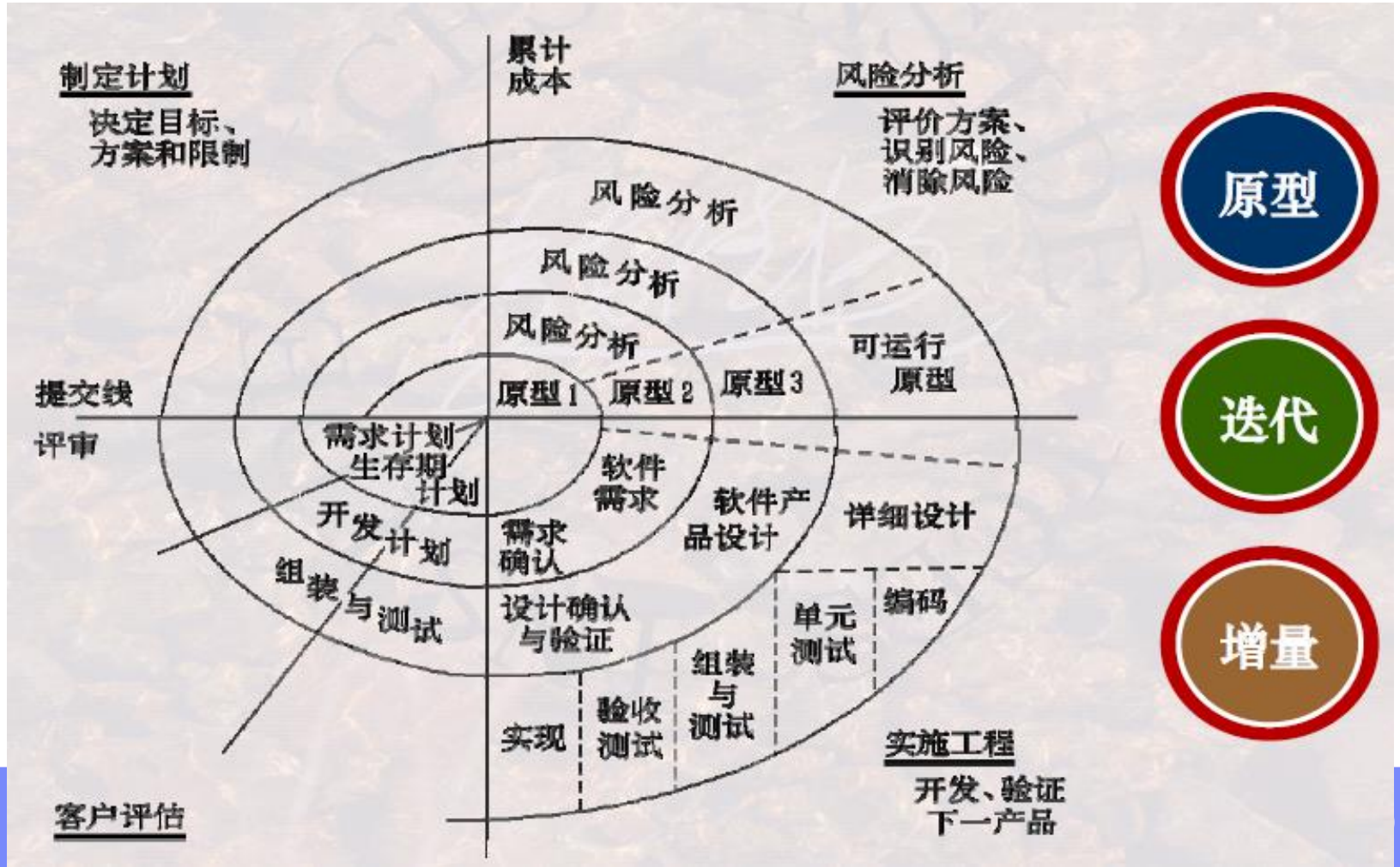


3. 软件的生命周期模型

- 进化模型（Evolutionary Process Model）：原型（Prototyping）
  - 用户为软件定义了一组目标，但没有确定功能的详细需求
  - 在某些情况下，开发人员可能不能确定算法的效率，操作系统的适用性，或人机交互应该采用的形式等

3. 软件的生命周期模型

- 进化模型：螺旋模型（Spiral Model）

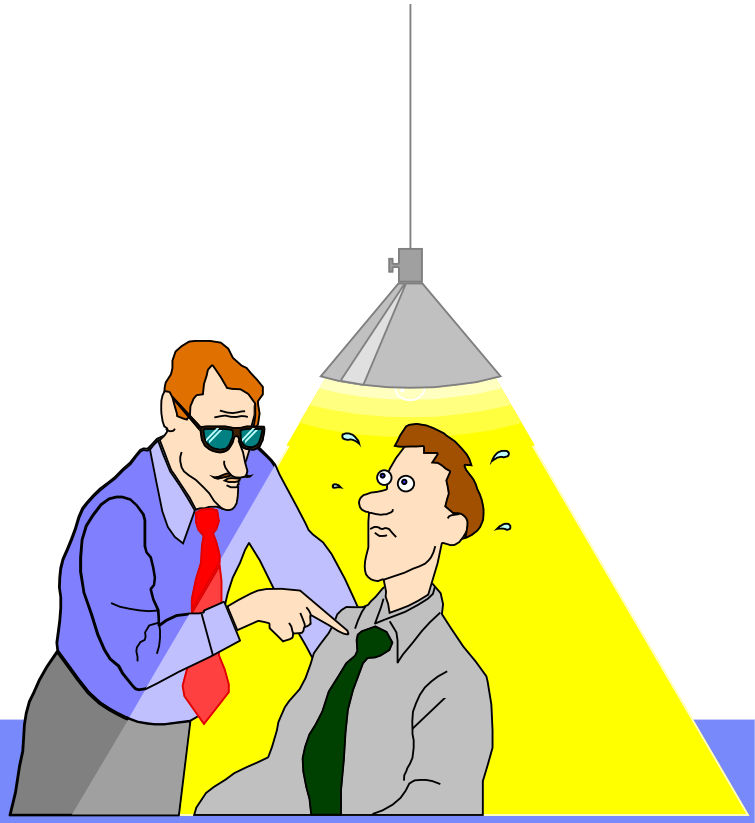


3. 软件的生命周期模型

- 进化模型：螺旋模型（Spiral Model）
  - 适用于软件的整个生命周期
  - 适合开发大规模系统和软件
  - 使用原型作为降低风险的机制
  - 结合瀑布模型和增量模型
  - 不是灵丹妙药，很难说服客户进化方法可控；需要大量的风险评估专业知识

4. 理解需求

- 开发人员的梦魇：“I know you think you understand what I said, but what you don’t understand is what I said is not what I meant.”



4. 理解需求

- 软件项目的成败 — 取决于对用户需求的理解

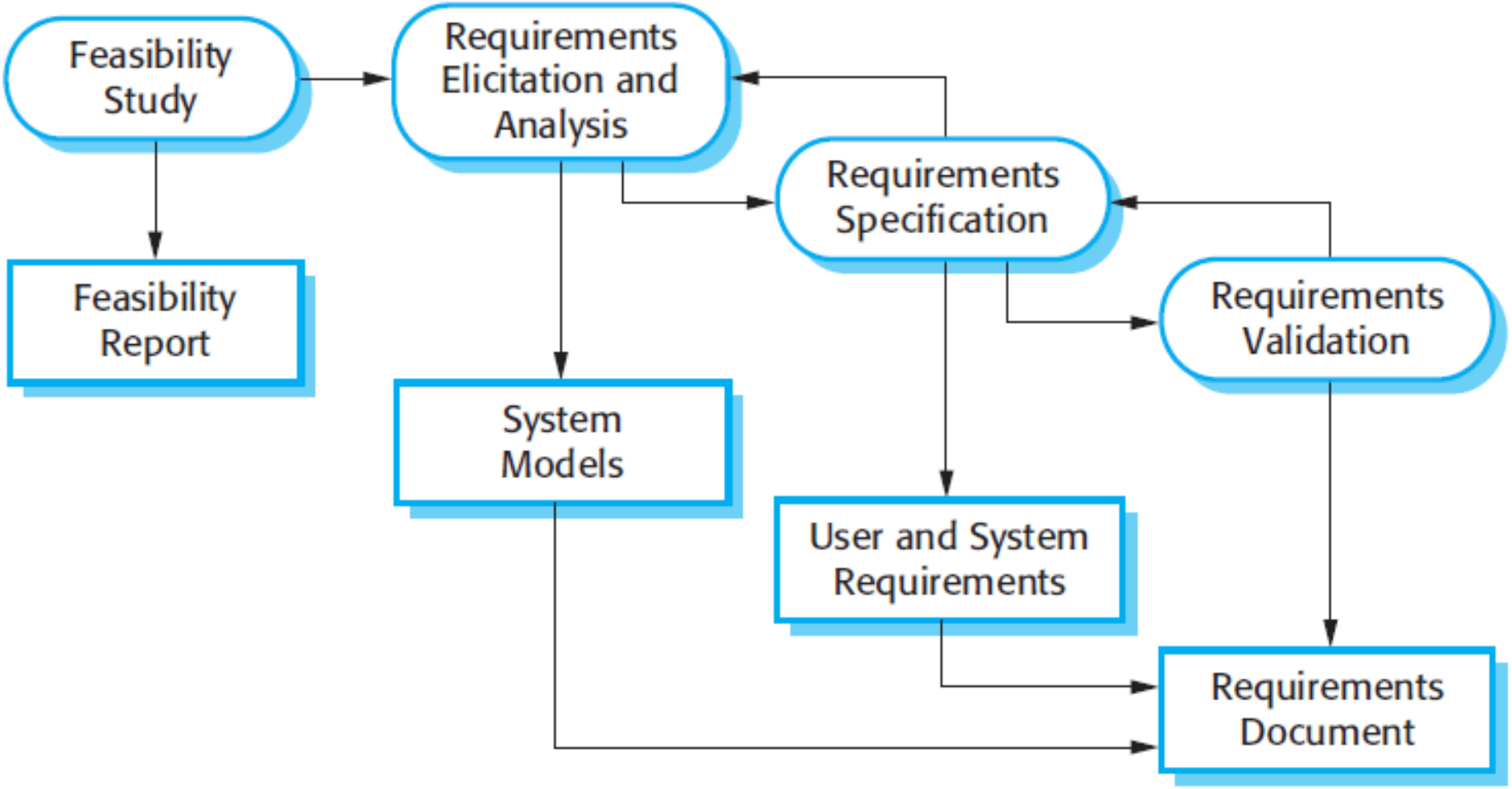
4. 理解需求

- 软件需求：被定义为软件用于解决真实世界问题而必须展示的特性，指用户对目标软件系统在功能、行为、性能、设计约束等方面的期望
- 需求包含系统特点的描述、系统如何运作的规格和限制，一般来说，**需求是在描述系统应该做什么(What)，而不是系统如何做到这些功能(How)**

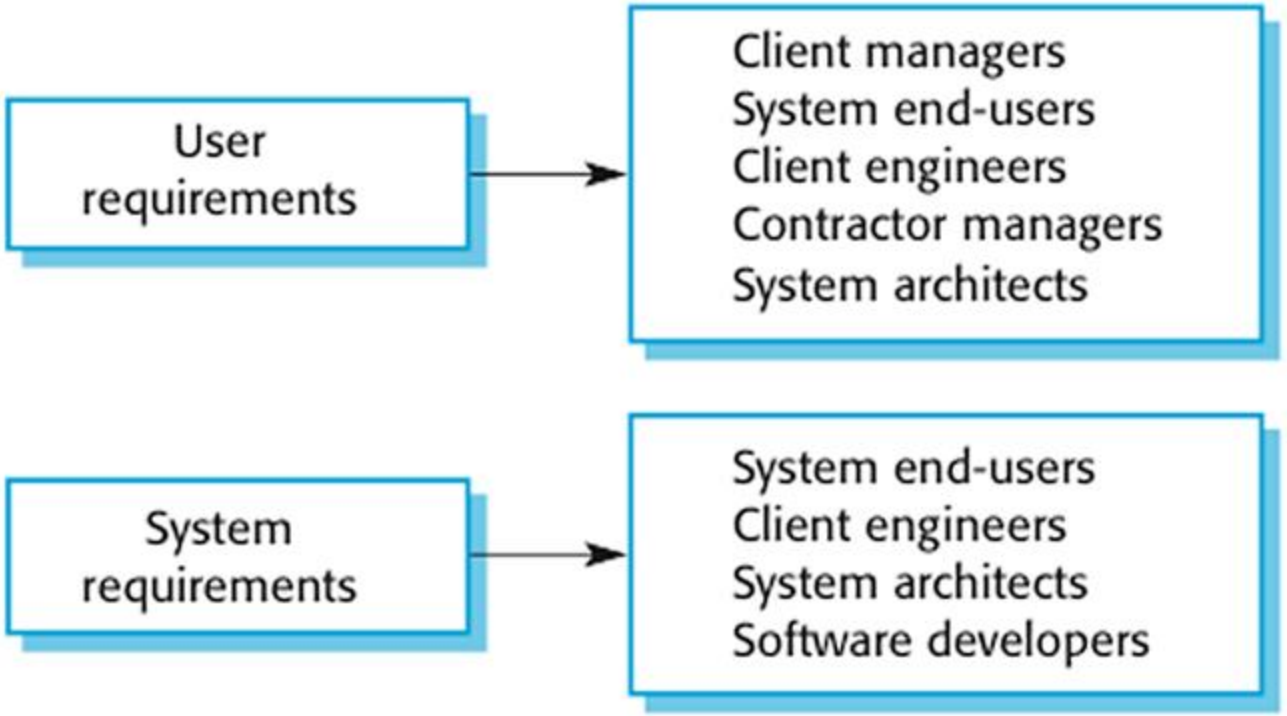
4. 理解需求

- 软件需求的特性

5. 需求工程（Requirements Engineering）



5. 需求工程（Requirements Engineering）



5. 需求工程（Requirements Engineering）

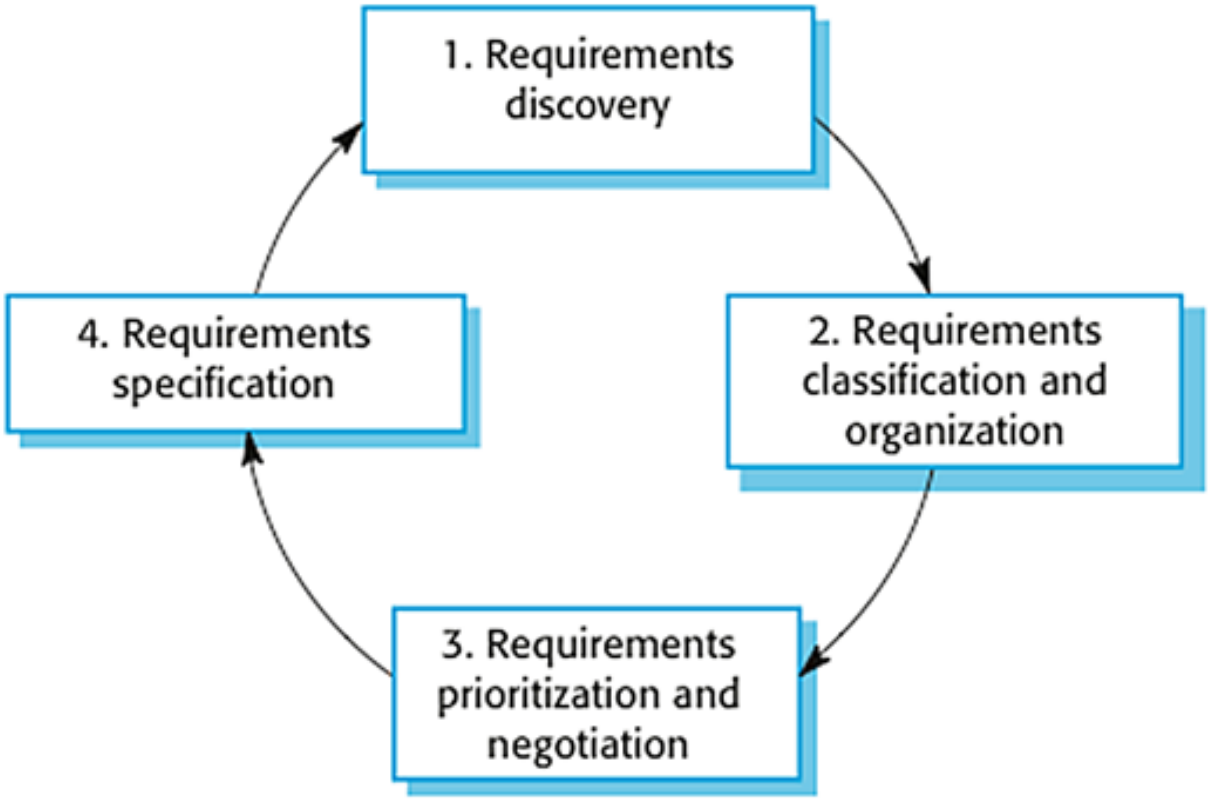
- 软件需求的分类
  - 功能需求：描述系统预期提供的功能或服务
    - 对系统应提供的服务
    - 如何对输入做出反应
    - 系统在特定条件下的行为
  - 非功能需求：主要与系统的总体特征相关的使用环境要求
    - 性能要求
    - 可靠性要求
    - 安全性要求
    - 可用性要求
    - 移植性要求
  - 领域需求：源于系统的应用领域需求

5. 需求工程（Requirements Engineering）

- 启发与分析（Elicitation & Analysis）
  - 与客户和最终用户交互，获得领域需求（系统应提供哪些服务，有哪些约束）
  - 可能还包括不同类型的利益相关者（stakeholder ）：管理者、系统工程师、测试工程师、维护工程师等

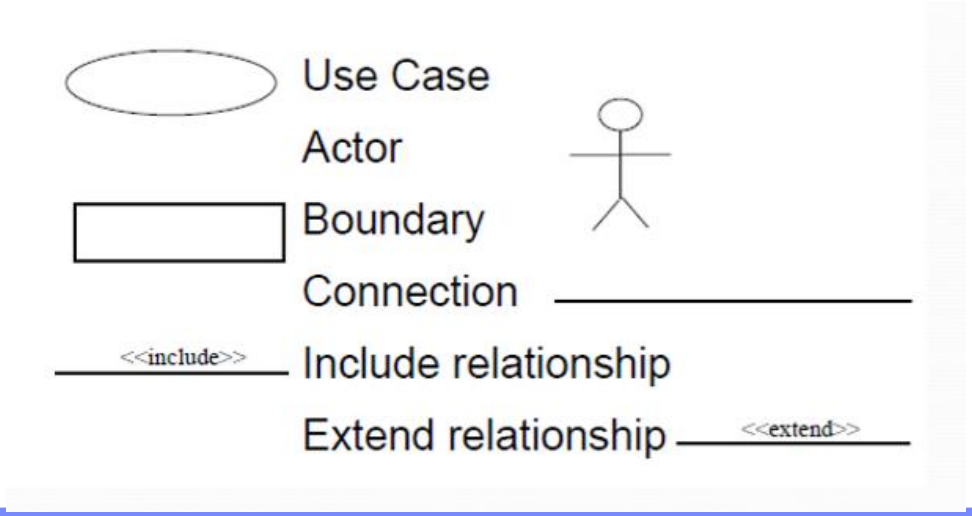
5. 需求工程（Requirements Engineering）

- 启发与分析（Elicitation & Analysis）



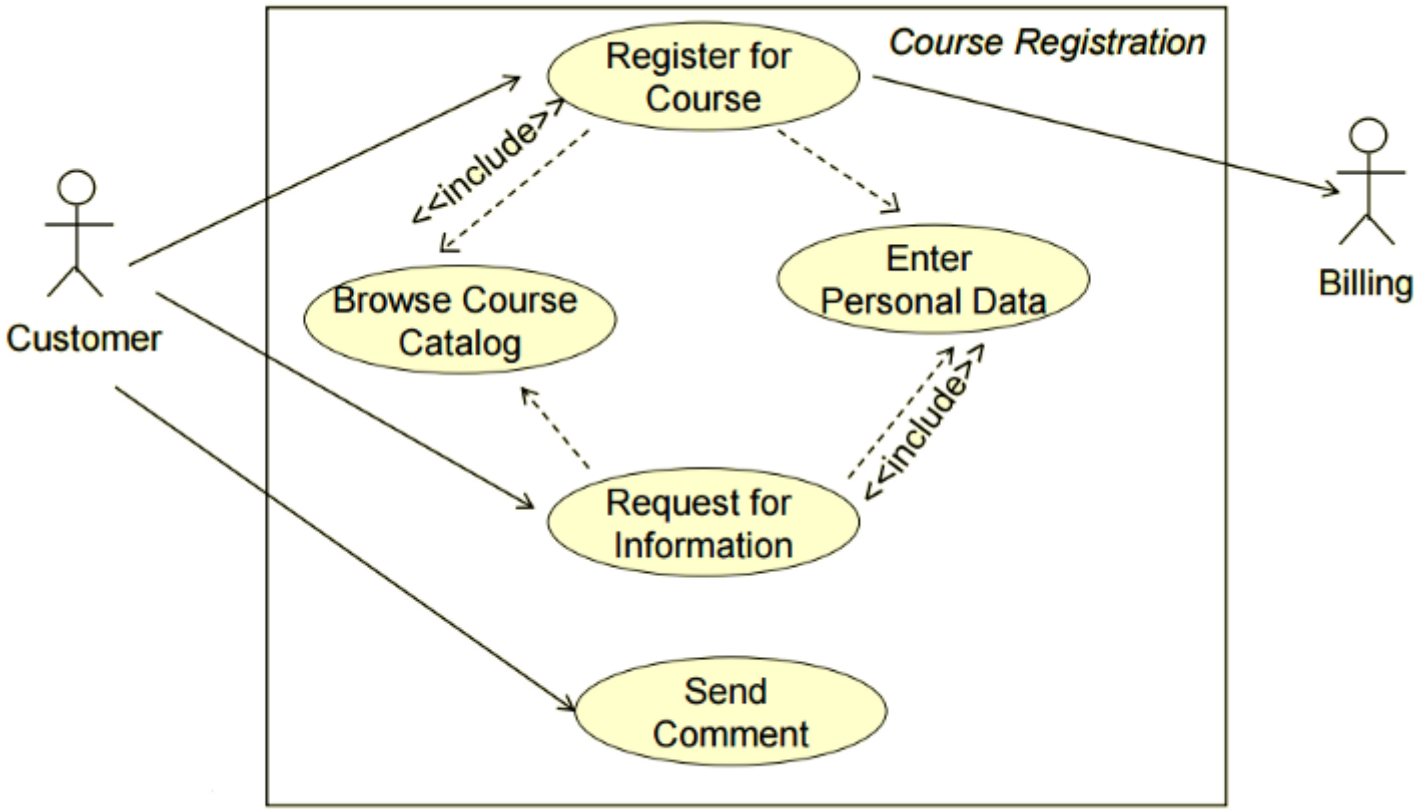
5. 需求工程（Requirements Engineering）

- 启发与分析（Elicitation & Analysis）
  - 需求发现（Requirements Discovery）
    - 用例和场景（Use Cases & Scenarios）：用例识别系统与其用户的交互，甚至是与其他外部系统的交互；场景是对这些交互的文字性描述



5. 需求工程（Requirements Engineering）

- 启发与分析（Elicitation & Analysis）
  - 需求发现（Requirements Discovery）



5. 需求工程（Requirements Engineering）

- 启发与分析（Elicitation & Analysis）
  - 需求发现（Requirements Discovery）

Name	Course Registration
Actors	Student and University System
Description	It shows how a student can register for a course and view personal info
Pre-condition	The student is logged in
Post-condition	The student registered his/her course list for the semester.
Actions(Main Scenario)	1. Student will press on "Course Registration" from Home Page. 2. Select desired courses for the next semester. 3. Enter personal info. 4. Press on "Register". 5. Confirmation message upon success. 6. Student can view his/her personal info.
Exceptions	#3 User entered invalid input, thus, an error message will be displayed

5. 需求工程（Requirements Engineering）

- 启发与分析（Elicitation & Analysis）
  - 需求分类和组织（Requirements Classification & Organization）：将相关的需求组织到一起，定义不同类需求之间的关系
  - 需求优先级和协商（Requirements Prioritization & Negotiation）：发现需求的优先级并通过协商解决可能有冲突的需求，达到不同利益相关者的妥协

5. 需求工程（Requirements Engineering）

- 需求说明（Requirements Specification）
  - 将用户需求和系统需求编写成文档，应清晰、易于理解、完整、一致

5. 需求工程（Requirements Engineering）

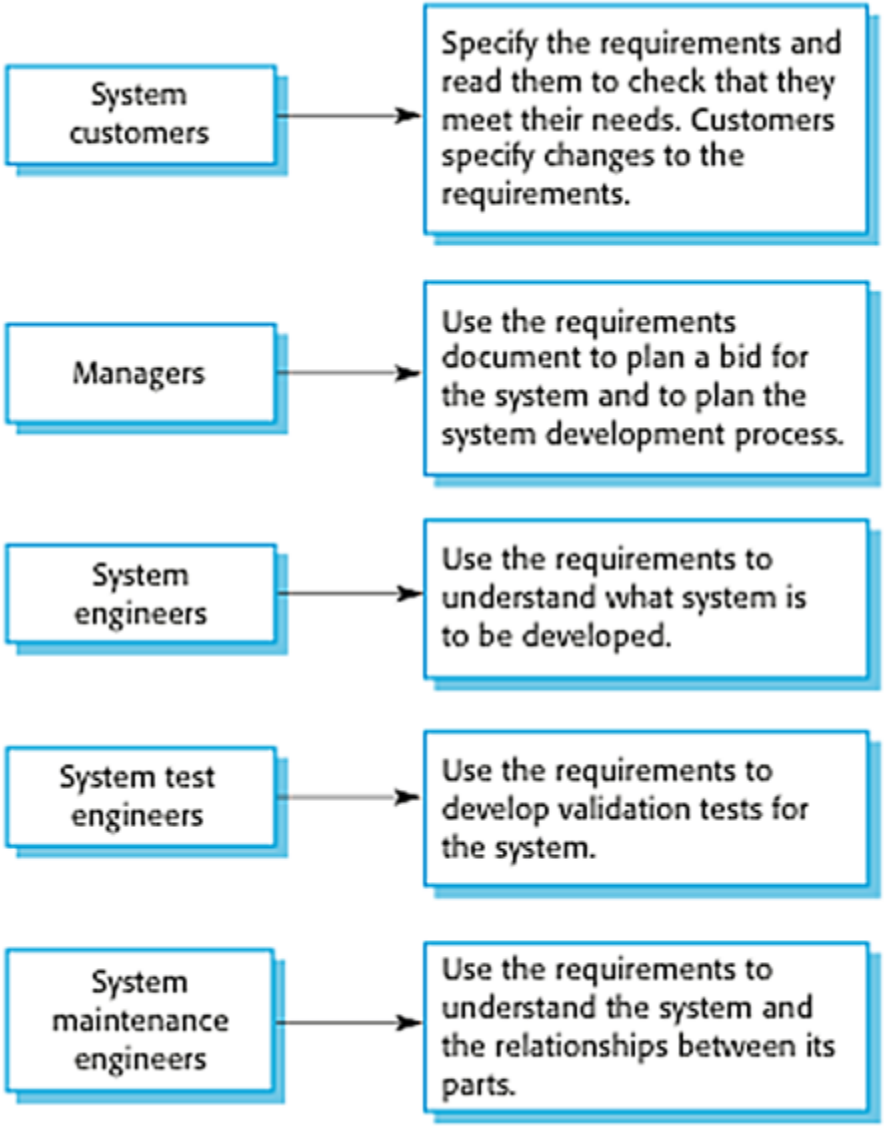
- 需求说明（Requirements Specification）
  - 软件需求文档（Software Requirements Document）或软件需求说明（software requirements specification or SRS）是软件实现何种功能的正式文档，也可作为系统购买者和软件开发者之间的合同

5. 需求工程（Requirements Engineering）

1 引言
　1.1 编写目的
　1.2 项目背景（单位和与其他系统的关系）
　1.3 定义（专门术语和缩写词）
2 任务概述
　2.1 目标
　2.2 运行环境
　2.3 条件限制
3 数据描述
　3.1 静态数据
　3.2 动态数据
　3.3 数据库描述
　3.4 数据字典
　3.5 数据采集

4 功能需求
4.1 功能划分
　4.2 功能描述
5 性能需求
　5.1 数据精确度
　5.2 时间特性
　5.3 适应性
6 运行需求
　5.1 用户界面
　5.2 硬件接口
　5.3 软件接口
　5.4 故障处理
7 其他需求
（检测或验收标准、可用性、可维护性、可移植性、安全保密性）

5. 需求工程（Requirements Engineering）



5. 需求工程（Requirements Engineering）

- 需求验证（Requirements Validation）
  - 确保声明的需求满足用户的要求，发现需求中存在的问题
  - 有效性、一致性、完整性、可行性、可确认性检查

5. 需求工程（Requirements Engineering）

- 需求验证（Requirements Validation）
  - 需求审查（Requirements Reviews）
    - 组织相关利益者阅读并审查需求，发现问题并协商
  - 原型（Prototyping）
    - 向用户展示原型系统并确认是否达到用户需求
  - 生成测试用例（Test-case Generation）
    - 将测试作为验证需求的一部分将有助于发现存在的问题

小结

- 软件的生命周期
- 软件的生命周期模型
- 需求分析是成功软件的第一步