《软件工程基础》

• 哈尔滨工业大学 •

电气工程及自动化学院

主　　讲：刘晓胜 博士

联系方法：0451-86402387

E m a i l：Liuxsh2004@126.com

第1章 软件工程概述

- 1.1 软件危机
- 1.2 软件工程基本原理
- 1.3 程序设计一般方法
- 1.4 软件开发模型
- 1.5 软件开发方法

第1章 软件工程概述

- 软件的概念、特点和分类：实例讲评1

名称	大小	类型	修改日期
串口调试助手V2.1	272 KB	应用程序	2001-6-26 7:02

第1章 软件工程概述

- 软件的概念、特点和分类：实例讲评2

第1章 软件工程概述

- 软件的概念、特点和分类：实例讲评2

课程各类实例\实验室自编小软件实例

名称	大小	类型	修改日期
COMTool	1 KB	配置设置	2006-8-9 10:35
PLCDebugger	1,133 KB	应用程序	2006-4-29 14:33
readme	1 KB	文本文档	2006-8-9 10:39

readme - 记事本

```
本软件采用Delphi7.0编写,
版本号说明:
版本号:V1.0.01 推出第一版,基本型;
版本号:V1.0.02 第一版改进型;
版本号:V1.1.01 第二版,汉字传输版,长度限制为10个汉字;
版本号:V1.1.02 第二版,改进型,长度限则改为100个汉字.
```

1.1 软件危机

- 软件的概念、特点和分类
  - 软件的概念：软件是计算机系统中与硬件相互依存的另一部分，它是包括程序、数据及其相关文档的完整集合。
    - 程序是按事先设计的功能和性能要求编写的指令序列；程序是完成指定功能的*一段特定语言代码*。
      - C,C++,Visual C++,Delphi, ASM, Matlab, Foxpro, PL/M... .
    - 数据是使程序能正常操纵信息的数据结构；
    - 文档是与程序开发、维护和使用有关的图文材料。

1.1 软件危机

- “软件”的定义为：计算机程序、方法、规则、相关的文档资料以及在计事机上运行时所必需的数据。
- 软件的特点
  - 软件是一种逻辑实体。
  - 软件的开发，是人的智力的高度发挥，而不是传统意义上的硬件制造。

1.1 软件危机

- 软件的特点
  - 软件维护与硬件的维修有着本质的差别。
  - 软件的开发至今尚未完全摆脱手工艺的开发方式，使软件的开发效率受到很大限制。
  - 软件的开发是一个复杂的过程。
  - 软件的成本非常高昂。

1.1 软件危机

- 软件的分类
  - 基于软件功能的划分
    - 系统软件:Windows/Units/Linux...
    - 应用软件:串口调试助手/MIS系统/...
    - 支撑软件: Visual C++/Matlab/Delphi/...

1.1 软件危机

- 软件的分类
  - 基于软件工作方式的划分
    - 实时处理软件：各种实时在线监控系统
    - 分时软件：各种定时软件/操作系统/...
    - 交互式软件：大部分软件都是交互式软件
    - 批处理软件：特定软件。

1.1 软件危机

- 软件工程的背景和历史
  - 1968年由NATO (北大西洋公约组织)在德国Garmish召开的学术会议上，Feitz Bauer首先提出了“软件工程”概念。
- 软件工程与编写程序(编程)
  - 前者是一门学科，一种科学理论来指导软件系统开发的标准化、自动化的过程。
  - 后者是单纯的代码编写。

1.1 软件危机

- 前者考虑如何分解一个系统，以便各人分工开发；考虑如何说明每个部分的规格要求；怎样才能易于维护。
- 后者是软件工程发展的前身。
- 后者是软件工程中占据很少时间和空间的一部分。

特别提示：南辕北辙！

- 容易关注编程，忽略软件工程的作用！
- 更容易关注技能，忽略理论指导！

1.1 软件危机

- 计算机学科的发展

计算机科学 (CS) ➡ 计算学科 (Computing discipline)

- 计算机科学 (CS)
- 计算机工程 (CE)
- 软件工程 (SE)
- 信息系统 (IS)

1.1 软件危机

- 计算机系统的发展历程
  - 第一代(20世纪60年代中期以前)：程序设计阶段。
  - 第二代(从20世纪60年代中期到70年代中期)：程序系统阶段——“软件工程”学科诞生。
  - 第三代(从20世纪70年代中期到80年代中期)：软件工程阶段。

1.1 软件危机

- 第四代(从20世纪80年代中期至今)：软件产业在世界经济中已经占有举足轻重的地位。
- 60年代以来
  - 工厂管理
  - 病人监护
  - 工资统发
  - 图书馆管理
  - 机票预定….

1.1 软件危机

- 软件发展
  - 早期(1960)
    - 面向批处理
    - 有限的分布
    - 自定义软件

简单的设计和实时控制

  - 第二阶段(1960-1975)
    - 多用户
    - 实时
    - 数据库
    - 软件产品

较复杂的设计、实时控制和集中管理

1.1 软件危机

- 第三阶段(1975-1988)
  - 分布式系统
  - 嵌入“智能”
  - 低成本硬件
  - 消费者的影响

复杂设计和在线实时控制

- 第四阶段(1988-)
  - 强大的桌面系统
  - 面向对象技术
  - 专家系统
  - 人工神经网络
  - 并行计算
  - 网络计算机

超大规模的设计、超高速实时控制和海量存储与运算等

1.1 软件危机

- 软件工程为什么发展如此之快：
  - 不准确的时间和金钱的估算；
  - 软件质量的低下；
  - 相对硬件产品开发软件开发费用的增加；
  - 维护、增强软件系统的必要性；
  - 硬件价格大幅度下降。

根本原因在于需求！软件危机的存在！

1.1 软件危机

- 软件危机概念
  - 软件开发和维护中存在的一系列严重问题，称“软件危机”。
  - 1968年，北大西洋公约组织的计算机科学家在德国召开的国际会议上，为解决“软件危机”问题，提出了“软件工程”的概念。

1.1 软件危机

- 软件危机表现
  - [1]成本和进度难掌握；
  - [2]对用户需求了解模糊，用户对软件不满意：“闭门造车”；
  - [3]质量不可靠：可靠性和质量保证/审查、复审和测试；
  - [4]不可维护：“可重用软件”；

1.1 软件危机

  - [5]没有文档：对管理者、开发者、维护者都是至关重要的；
  - [6]软件成本逐年上升；
  - [7]跟不上硬件发展。
- 实例讲评3：
  - 软件没有适当的文档资料导致软件废弃：石英摆片测试软件。

1.1 软件危机

- 实例讲评4：
  - 软件维护性实例：Microsoft word发展历程；
    - Microsoft word5.0　Microsoft word6.0
    - Microsoft word7.0　Microsoft word95
    - Microsoft word97　Microsoft word2000
    - Microsoft word2003 …
- 实例讲评5：
  - 硬盘刻录机软件：软件bugs无法出来。

1.1 软件危机

- 软件危机原因
  - (1)与软件特点有关：管理和控制软件开发过程相当困难！
  - (2)开发和维护方法不正确：
    - 个体化特点
    - 忽视需求分析
    - 忽视评审
    - 忽视测试
    - 忽视维护等。

1.1 软件危机

- 软件危机面临的问题：进展难衡量；质量难评估；管理难控制。
  - 如何开发软件，以满足对软件日益增长的需求；
  - 如何维护数量不断膨胀的已有软件；
  - 规模、复杂性、生产率。

1.1 软件危机

- 实例讲评6：
  - Windows95有1000万行代码
  - Windows2000有5000万行代码
  - Exchange2000和 Windows2000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

1.1 软件危机

- 软件危机的主要特征
  - 软件开发周期大大超过规定日期；
  - 软件开发成本严重超标；
  - 软件质量难于保证。
- 解决途径：
  - 技术措施(方法和工具)+管理措施=软件工程支撑环境；

1.2 软件工程

- 软件工程的定义
  - 软件工程是指研究软件生产的一门学科，也就是将完善的工程原理应用于经济地生产既可靠又能在实际机器上有效运行的软件。
  - 1983年美国《IEEE软件工程标准术语》【IEEE83】对软件工程下的定义为：软件工程是开发、运行、维护和修复软件的系统方法。

1.2 软件工程

- IEEE【IEE93】给出了一个更加综合的定义：“将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程，即将工程化应用于软件中。”
- Fritz Bauer在NATO会议上给出的定义：“软件工程是为了经济地获得可靠的和能在实际机器上高效运行的软件而确立和使用的健全的工程原理（方法）。”
- 软件工程强调的是过程！

1.2 软件工程

- 对比：“软件”的定义：计算机程序、方法、规则、相关的文档资料以及在计事机上运行时所必需的数据。（“软件”强调的是结果）
- 软件工程方法学
  - 通常把在软件生命周期全过程中使用的一整套技术的集合，称为软件工程方法学。软件工程方法学包括三个要素：方法、工具和过程。

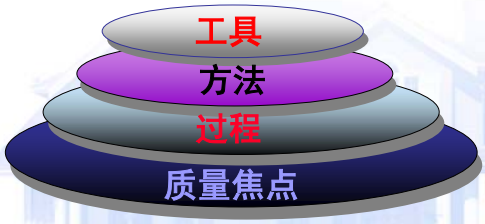
1.2 软件工程

- 软件工程方法是完成软件开发的各项任务的技术方法，为软件开发提供了“如何做”的技术。
  - 可行性分析、需求分析、评审与复审、 …
- 软件工程工具为软件工程方法提供了自动的或半自动的软件支撑环境。
  - 广义的工具
  - 系统流程图、数据流图、数据字典、…
  - PAD图、判断树、判断表、测试用例、…

1.2 软件工程

- 软件工程过程则是将软件工程的方法和工具综合起来以达到合理、及时地进行计算机软件开发的目的。
- 软件工程三个要素关系：方法、工具、过程
- 传统方法学和面向对象方法学是目前使用得最广泛的两种软件工程方法学。

1.2 软件工程



Software Engineering Layers

软件工程三个要素：方法、工具、过程

1.2 软件工程

- 软件工程的主要研究内容
  - 软件开发技术
    - 软件开发方法学
    - 软件开发过程
    - 软件工具和软件工程环境

1.2 软件工程

- 软件工程管理（本课程不讲）
  - 软件管理学
  - 软件经济学
  - 软件心理学

特别提示：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，应用发展的眼光看待它。

1.2 软件工程

- 软件工程的基本原理
  - B.W.boehm七条基本原理：
    - 用分阶段的生命周期计划严格管理；
    - 坚持进行阶段评审；
    - 实行严格的产品控制；
    - 采用现代程序设计技术；
    - 结果应能够清楚地审查；
    - 开发小组的人员应少而精；
    - 承认不断改进软件工程实践的必要性；

1.2 软件工程

- 项目概要计划
- 里程碑计划
- 项目控制计划
- 产品控制计划
- 验证计划
- 运行维护计划

特别提示:不能受用户和上级的影响!

1.2 软件工程

- 软件工程与一般工程的差异
  - 软件是逻辑产品而不是实物产品；
  - 软件的功能依赖于硬件和软件的运行环境以及人们对它的操作；
  - 软件设计的复杂性；
  - 软件特征：功能的多样性、实现的多样性、能见度低、软件结构合理性差；
  - 智力密集及知识产权保护。

1.2 软件工程

- 软件工程知识结构
  - 2001年5月ISO/IEC JTC 1（ISO和IEC的第一联合技术委员会）发布了《 SWEBOK指南 V0.95(试用版)》
  - SWEBOK把软件工程学科的主体知识分为10个知识领域。

1.2 软件工程

- 软件质量
- 软件工程工具和方法
- 软件工程过程
- 软件工程管理
- 软件配置管理
- 软件维护
- 软件测试
- 软件构造
- 软件设计
- 软件需求

1.2 软件工程

- 软件开发总体要求

软件产品的标准化

↓

软件开发过程的标准化

1.2 软件工程

- 软件的工业化生产过程应具备的特点：
  - 明确的工作步骤；
  - 详细具体的规范化文档；
  - 明确的质量评价标准。

1.2 软件工程

- 软件工程技术的两个特点
  - 强调规范化；
  - 强调文档化。

特别提示：国家有相关的国标要求！

1.3 软件生存期和软件开发模型

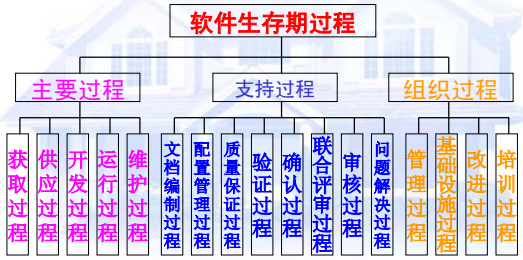
- 软件生存期(Software Life Cycle)
  - 如同任何其他事物一样，软件也有一个孕育、诞生、成长、成熟、衰亡的生存过程，一般称之为计算机软件的生存期。
  - 软件产品或软件系统从设计、投入使用到被淘汰的全过程。
  - 一般说来，软件生命期由软件定义、软件开发和软件维护三个时期组成，每个时期又可进一步划分成若干个阶段。

1.3 软件生存期和软件开发模型

- 软件生存期的阶段划分
  - (1) 可行性研究与计划 } 怀孕期（定义/计划期）
  - (2) 需求分析
  - (3) 总体设计
  - (4) 详细设计
  - (5) 实现(编码)
  - (6) 集成测试
  - (7) 确认测试 } 成长期（开发期）
  - (8) 使用和维护 } 成年期（运行/维护期）

1.3 软件生存期和软件开发模型

- 新的国际标准定义的软件生存过程（1995 ISO/IEC 12207）



1.3 软件生存期和软件开发模型

- 软件定义时期
  - 问题定义：这是软件生存期的第一个阶段，主要任务是弄清用户要计算机解决的问题是什么。
  - 实例讲评7：求一元二次方程的解的“问题定义”。

问题定义：对于给定的一元二次方程 $ax^2+bx+c=0$ 求解其根的情况。

1.3 软件生存期和软件开发模型

- 可行性研究：任务是为前一阶段提出的问题寻求一种至数种在技术上可行、且在经济上有较高效益的解决方案。
- 实例讲评7：求一元二次方程的解的“可行性研究”。

可行性研究：此方程可以根据判别式b^2-4ac的值来确定方程根的情况，即： b^2-4ac>0 时有两个不等实根； b^2-4ac=0时有相等的两个实根； b^2-4ac< 0 时有共轭复数根。

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

1.3 软件生存期和软件开发模型

- 软件开发时期
  - 需求分析：弄清用户对软件系统的全部需求，主要是确定目标系统必须具备哪些功能。
  - 实例讲评8：求一元二次方程的解的“需求分析”。

需求分析：用户任意输入一元二次方程的系数a,b,c，软件自动计算判别式，并给出该方程的相关解。

- a=b=c=0情况或a=b=0情况；
- a=0或a≠0情况；
- b^2-4ac>0或b^2-4ac=0或b^2-4ac< 0情况；

1.3 软件生存期和软件开发模型

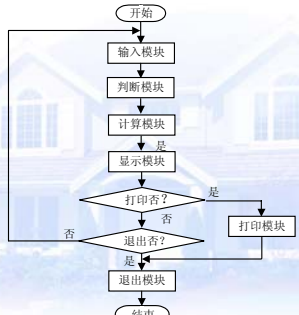
- 软件开发时期
  - 总体设计：设计软件的结构，即确定程序由哪些模块组成以及模块间的关系。
  - 实例讲评9：求一元二次方程的解的“总体设计”。

总体设计：

- ❖软件将包括：输入数据模块、判断模块、计算模块和显示输出模块以及退出模块。
- ❖模块之间关系如图所示。

1.3 软件生存期和软件开发模型

系统流程图



1.3 软件生存期和软件开发模型

- 软件开发时期
  - 详细设计：针对单个模块的设计。
  - 编码：按照选定的语言，把模块的过程性描述翻译为源程序。
  - 实例讲评10：编码。

1.3 软件生存期和软件开发模型

实例10：源程序

```
# include “math.h”
main()
{  float a,b,c,disc,x1,x2,p,q;
   scanf(“a=%f,b=%f,c=%f”,&a,&b,&c);
   disc=b*b-4*a*c;
   p=-b/(2*a);
   q=sqrt(disc)/(2*a);
   x1=p+q;
   x2=p-q;
   printf(\n\nx1=%5.2f\nx2=%5.2f\n”,x1,x2);
}
```

引自：谭浩强之《C程序设计》(1991.7)p45例3.12

1.3 软件生存期和软件开发模型

实例10：源程序

```
# include “math.h”
main()
{  float a,b,c,disc,x1,x2,p,q;
   int quit;
   quit=0;
   do
   {    scanf(“a=%f,b=%f,c=%f”,&a,&b,&c);
        if((abs(a)<=0.00001)&& (abs(b)<=0.00001)&&
                (abs(c)<=0.00001)
            printf(“方程系数不可全部为零”);
        else
```


1.3 软件生存期和软件开发模型

```
        if((abs(a)<=0.00001)&& (abs(b)<=0.00001)&&
                (abs(c)>=0.00001)
            printf(“方程系数不正确,请重新输入!”);
        else
            {
                disc=b*b-4*a*c;
                p=-b/(2*a);
                q=sqrt(disc)/(2*a);
                x1=p+q;
                x2=p-q;
            printf(\n\nx1=%5.2f\nx2=%5.2f\n”,x1,x2);
            printf(“退出吗?退出,请输入1,否则继续求解”)
```

1.3 软件生存期和软件开发模型

```
            scanf("quit=%d",&quit);
            }
        }
    }
    while (quit==0)
}
```


1.3 软件生存期和软件开发模型

- 测试：通过各种类型的测试(及相应的调试)使软件达到预定的要求。
  - 设计测试方案和测试用例
- 实例讲评11：测试.
  - 本例可采用白盒测试。
  - 测试用例: [1]0,0,0; [2]0,0,3; [3]1,3,1; [4]1,2,1; [5]1,1,1; [6] 'v',3,6。

1.3 软件生存期和软件开发模型

- 软件运行时期
  - 软件生存周期的最后一个时期。软件人员在这一时期的工作，主要是做好软件维护。维护的目的，是使软件在整个生存周期内保证满足用户的需求和延长软件的使用寿命。

1.3 软件生存期和软件开发模型

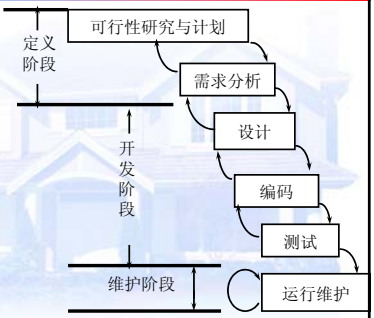
- 软件开发模型是软件开发全部过程、活动和任务的结构框架。它能直观表达软件开发全过程，明确规定要完成的主要活动、任务和开发策略。软件开发模型常称为：
  - 软件过程模型
  - 软件生存周期模型
  - 软件工程模型

1.3 软件生存期和软件开发模型

- 软件开发模型
  - 瀑布模型
  - 螺旋模型
  - 喷泉模型

1.3 软件生存期和软件开发模型

- 瀑布模型：将软件生存周期的各项活动规定为依照固定顺序连接的若干阶段工作，形如瀑布流水，最终得到软件产品。

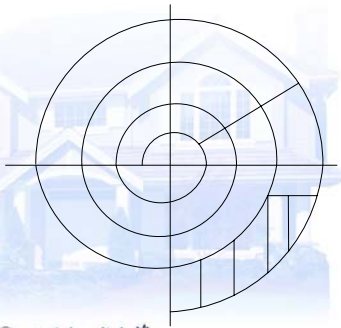


1.3 软件生存期和软件开发模型

- 按照传统瀑布模型开发软件的特点
  - 1.阶段间具有顺序性和依赖性。
  - 2.推迟实现的观点。
  - 3.每个阶段必须完成规定的文档；每个阶段结束前完成文档审查，及早改正错误。

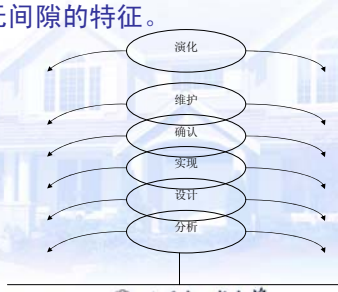
1.3 软件生存期和软件开发模型

- 螺旋模型：为了克服瀑布模型的不足，螺旋模型于1988年提出。该模型中加入了风险分析，通常用来指导大型软件项目的开发。



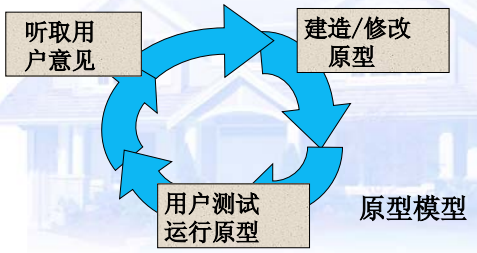
1.3 软件生存期和软件开发模型

- 喷泉模型：体现了软件创建所固有的迭代和无间隙的特征。



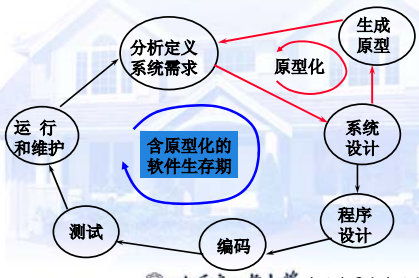
1.3 软件生存期和软件开发模型

- 原型模型（快速原型模型）



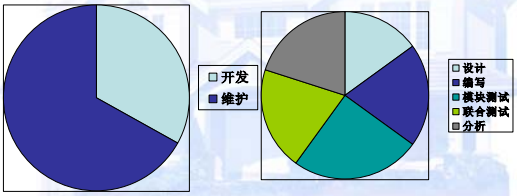
1.3 软件生存期和软件开发模型

- 采用原型模型的软件生存周期



1.4 软件质量的评价

- 开发软件不仅仅是编程



1.4 软件质量的评价

- 成功的标准：
  - 用户在用；
  - 用户可很容易做完要做的事；
- 失败的根本原因：
  - 开发人员写出的东西达不到用户要求（人的问题、技术问题）

1.4 软件质量的评价

- 质量与生产率
  - 质量是软件需求方最关心的问题，用户即使不图物美价廉，也要求个货真价实；
  - 质量与生产率之间有着内在的联系，高生产率必须以质量合格为前提；
  - 质量与生产率的提高就指望程序员与程序经理；
  - 非得在质量与生产率之间分个主次不可，那么应该是质量第一，生产率第二。

1.4 软件质量的评价

- 质量直接体现在软件的每段程序中，高质量自然是开发人员的技术追求，也是职业道德的要求。
- 高质量对所有的用户都有价值，而高生产率只对开发方有意义。
- 如果一开始就追求高生产率，容易使人急功近利，留下隐患。

1.4 软件质量的评价

- “没问题”的标准：
  - “运行正确”的程序就是高质量的程序吗？
  - 也许运行速度很低并且浪费内存；也许代码写得一塌糊涂！

1.4 软件质量的评价

- 软件的质量因素
  - 软件的质量因素很多，如正确性、精确性、可靠性、容错性、性能、效率、易用性、可理解性、简洁性、可复用性、可扩充性、兼容性等等。一般说来倾向于可维护性、可靠性、可理解性和效率

1.4 软件质量的评价

- 正确性与精确性
  - 机器不会主动欺骗人，软件运行不正确或者不精确一般都是人造成的。
  - 需求分析错了，那么对客户而言这个软件也存在错误。
  - 如果软件没有100% 地按需求规格执行，那么这个软件也存在错误。
  - 程序员要为“正确”、“精确”四个字竭尽全力。

1.4 软件质量的评价

- 性能与效率
  - 用户都希望软件的运行速度高些（高性能），并且占用资源少些（高效率）。
  - 通过优化算法、数据结构和代码组织来提高软件系统的性能与效率优化的关键。
  - 工作是找出限制性能与效率的“瓶颈”。

1.4 软件质量的评价

- 易用性
  - 导致软件易用性差的根本原因是开发人员犯了“错位”的毛病：他以为只要自己用起来方便，用户也一定会满意！
  - 当用户真的感到软件很好用时，一股温暖的感觉油然而生，于是就用“友好”来评价易用性！

1.4 软件质量的评价

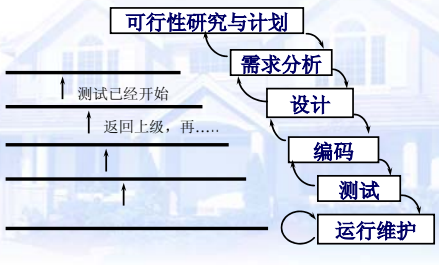
- 可理解性与简洁性
  - 开发人员只有在自己思路清晰时才可能写出让别人能理解的程序。
  - 编程时还要注意不可滥用技巧，应该用自然的方式编程。
  - 简洁是一种美。
  - 如果把学术文章写得很简洁，让人很容易理解，它往往中不了。

1.4 软件质量的评价

- 可复用性与可扩充性
  - 一种方式是原封不动地使用现成的软件构件。
  - 一种方式是对现成的软构件进行必要的扩充后再使用。
  - 可复用性好的程序一般也具有良好的可扩充性。

1.4 软件质量的评价

- 瀑布模型的质量保障体系



1.4 软件质量的评价

- 软件的高质量主要是设计出来的。
- 不是“管”出来的。
- 更不能依赖质量检查。

1.5 软件开发过程控制

- 开发进度：
  - 实例讲评12：国家标准：开发进度月报
- 项目开发计划：
  - 实例讲评13：国家标准：项目开发计划

本章结束