软件工程

微信扫码签到

内容

1. 软件工程产生背景

✓软件危机的表现及根源

2. 软件工程基本内涵

✓思想、要素、目标和原则

3. 软件工程发展历程

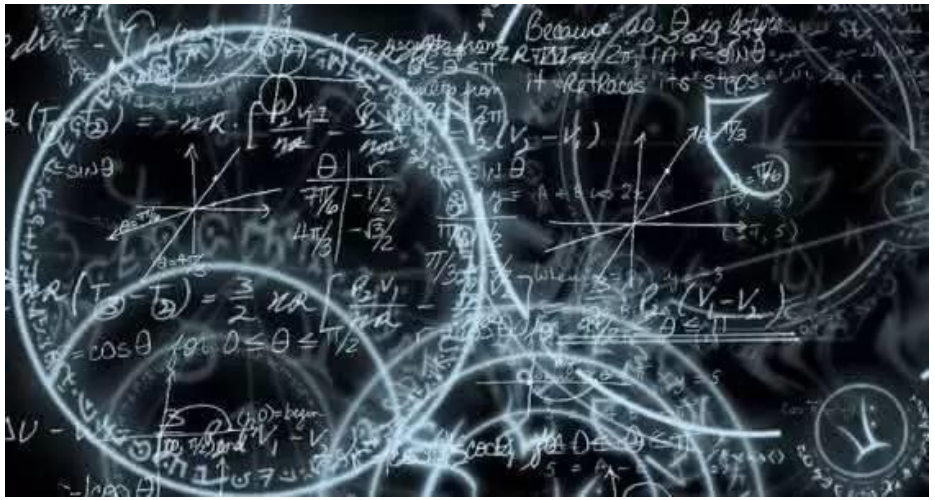
✓不同发展阶段的成果及特点

4. 软件工程教育特点

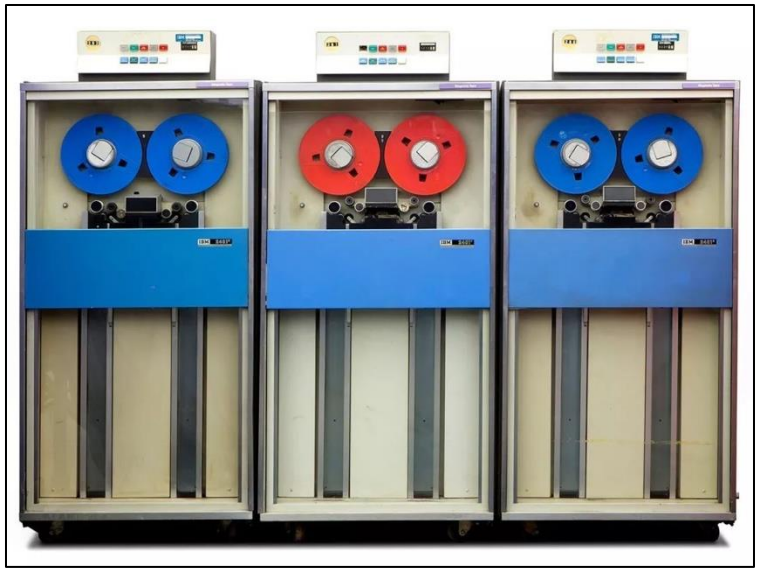
✓教育规范、知识体系和课程特点

1950s-1960s的计算机软件应用

- **应用领域变化**
  - ✓最早满足军方应用，如科学计算
  - ✓逐步走向商业应用等新领域，如**银行、航空**等领域的事务处理
- **应用数量增长**
  - ✓计算机软件的需求量不断上升
- **应用复杂性增加**
  - ✓多样化的用户
  - ✓多样化的需求

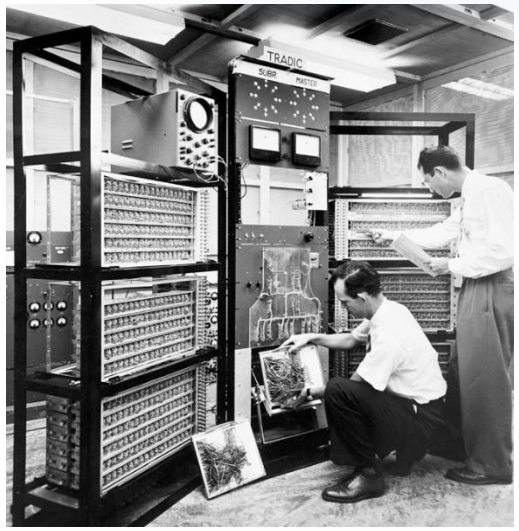


示例：IBM 360 OS软件开发



- **OS/360 超大型软件项目(1960s初)：**
  - ✓ 复杂软件：支持多道程序，最多可同时运行15道程序
  - ✓ 软件工程师超2000人，花费超5亿美元，工作量超5000人年
- **有史以来最可怕的软件开发泥潭**
  - ✓ Brooks，《人月神话》The Mythical Man-Month、图灵奖获得者

1960s的个体作坊式软件开发



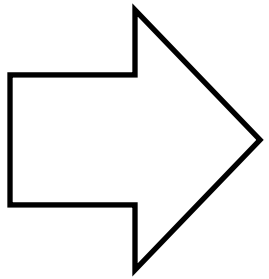
作坊式的个人创作

第二代晶体管计算机：
TRADIC（1954）
IBM 1401（1958）

- 依靠个人的能力
- 相互之间缺乏合作
- 关注计算存储时空利用，精雕细琢
- 程序规模小且功能单一
- 无系统性方法和标准流程

1.1 个体作坊式创作带来的问题

作坊式的个体编程开发



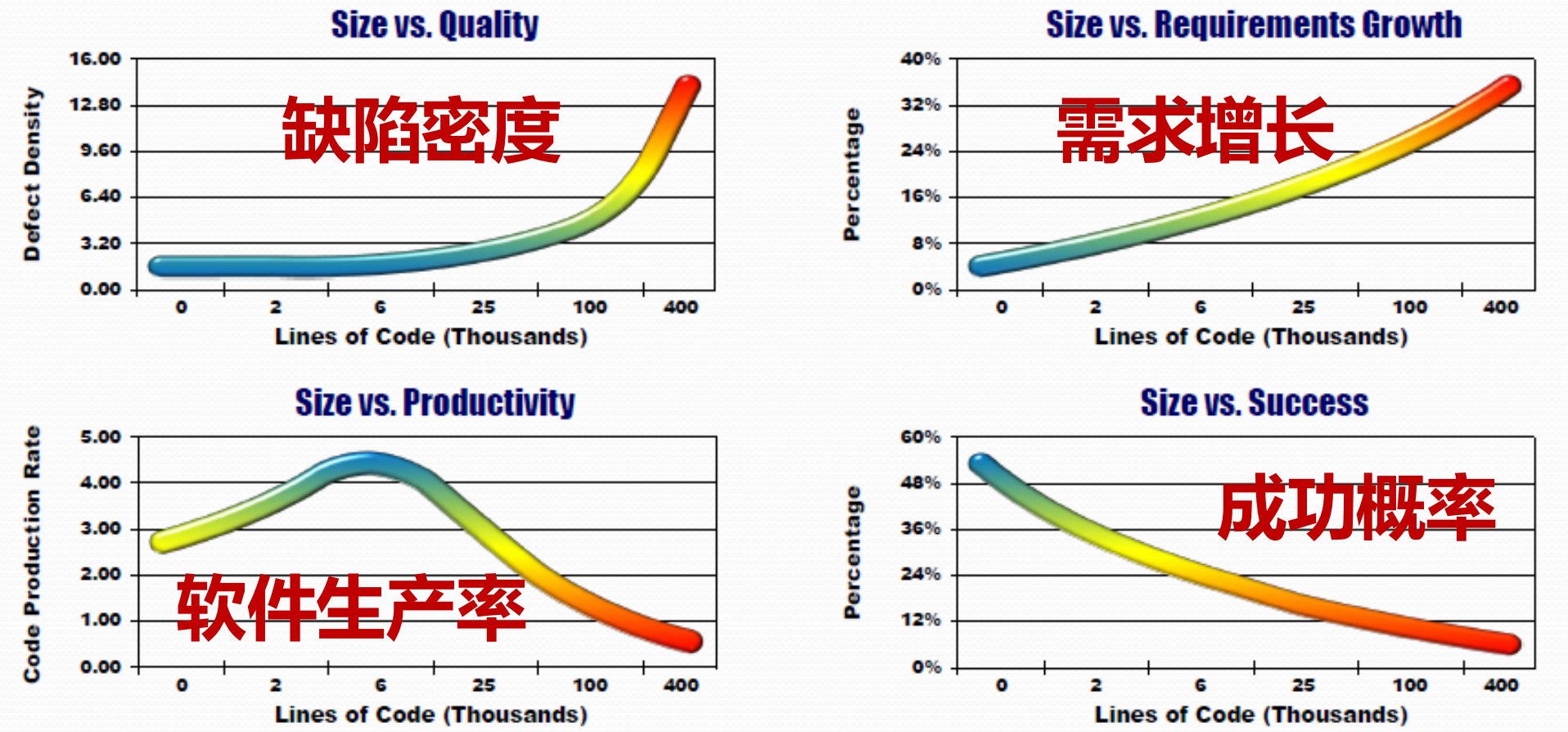
大批量和大规模软件系统的开发

个体作坊式方法会给软件开发带来什么问题?

1.2 软件开发需要解决的问题

- **开发过程**：基于什么样的步骤来开发软件系统
- **开发方法**：采用怎样的方法来指导各项软件开发活动
- **开发管理**：如何组织开发人员和管理软件产品
- **质量保证**：如何保证软件开发活动和制品的质量

软件开发面临的挑战日趋突出



代码规模增长对质量、生产率、成功开发带来的影响

大规模软件开发的案例

- **Windows系列软件代码量**
  - ✓Windows 95: 1500万行
  - ✓Windows 98: 1800万行
  - ✓Windows XP: 3500万行
  - ✓Windows Vista: 5000万行
  - ✓**Windows 7: 7000万行**
- **Windows 7 开发组织**
  - ✓核心团队的人大约有1000人、25个功能小组
  - ✓每个小组大约有40个人，每个小组包括三个部分的工作人员：程序经理，开发工程师，测试工程师

当软件规模越大，上述挑战就越突出，面临的困难也就越多

我们所面临的挑战

- **指挥信息系统中的软件**
  - ✓规模大、质量要求高
- **装备中嵌入式软件系统**
  - ✓规模大、质量要求高

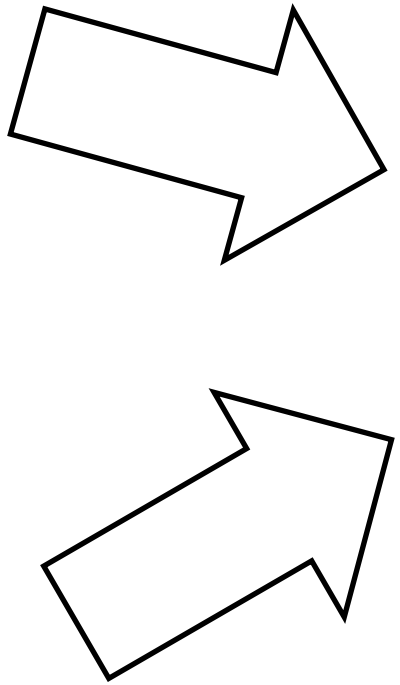
□ 信息化需要多样化和高质量的软件！使命伟大！责任重大！

□ 信息化建设任重道远！

1.3 软件危机的出现

作坊式的个体编程

大批量、大规模软件开发



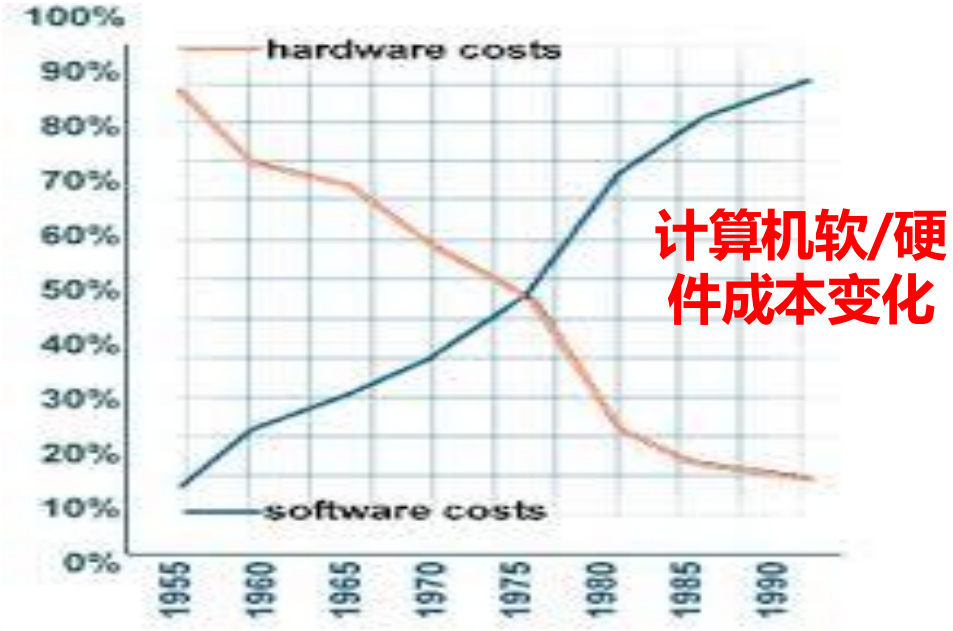
软件危机

- ✓ **进度**经常延迟
- ✓ **质量**无法保证
- ✓ **成本**超出预算
- ✓ 软件**维护**困难
- ✓ 失败**风险**很大

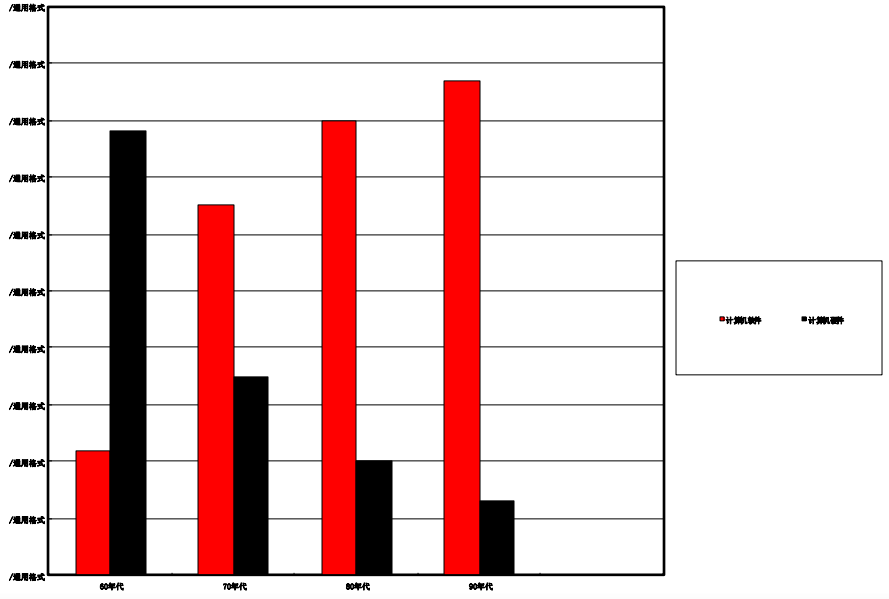
1.3.1 开发成本高

❑软件成本高，软硬件投资比发生急剧变化

- ✓美国空军：1955年软件占总费用(计算机系统)的18%，70年60%，85年达到85%
- ✓IBM 360 OS：5000+人年，耗时4年(1963-1966)，花费2亿多美元



计算机软/硬件成本变化



1.3.2 进度难以控制

- 项目延期比比皆是
- 由于进度问题而取消的软件项目较常见
- 只有一小部分的项目能够按期完成

1.3.3 质量难以保证

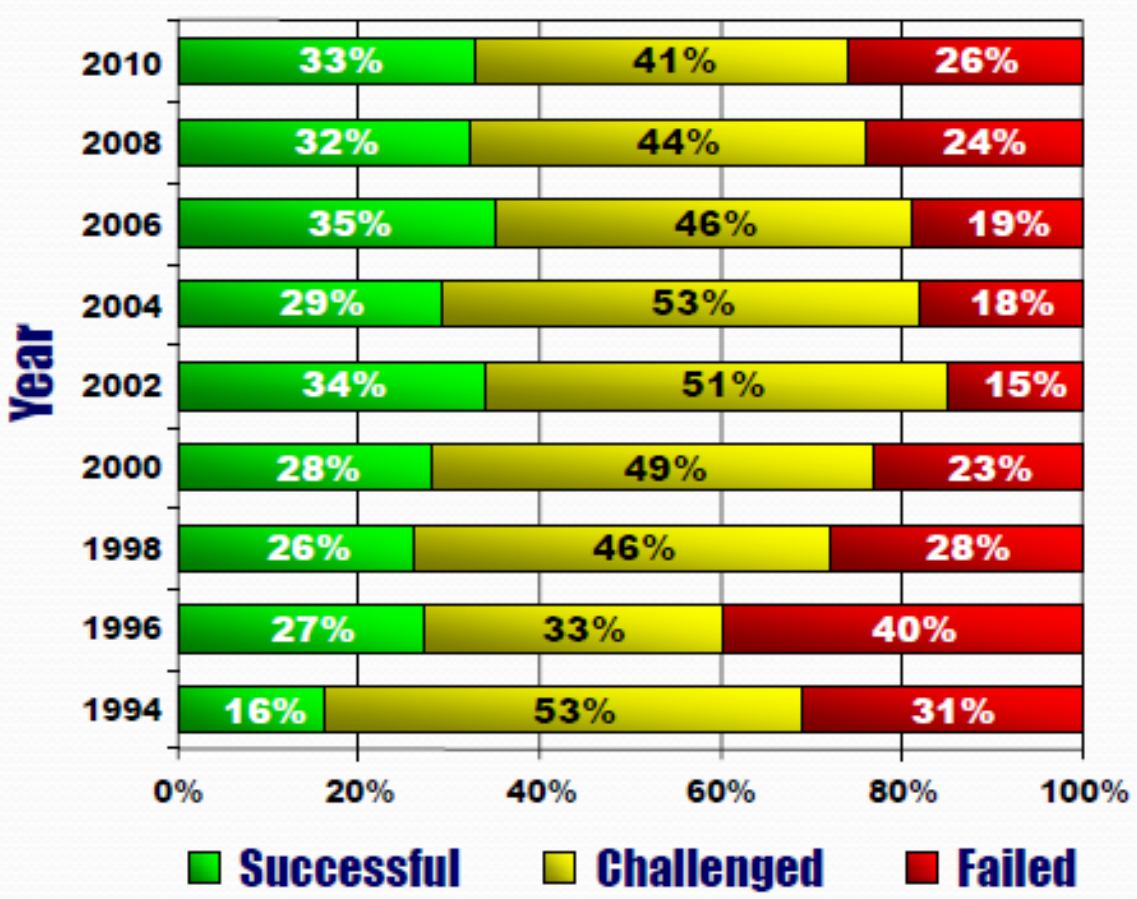
- 人总是会犯错误的
- 软件开发的错误表现为多种形式
  - ✓没有按照要求（需求）来开发
  - ✓编写的代码在功能上存在错误
  - ✓实现了功能但是性能达不到要求
  - ✓所开发的软件交互界面用户不喜欢
  - ✓.....
- 有些软件错误可能是致命的

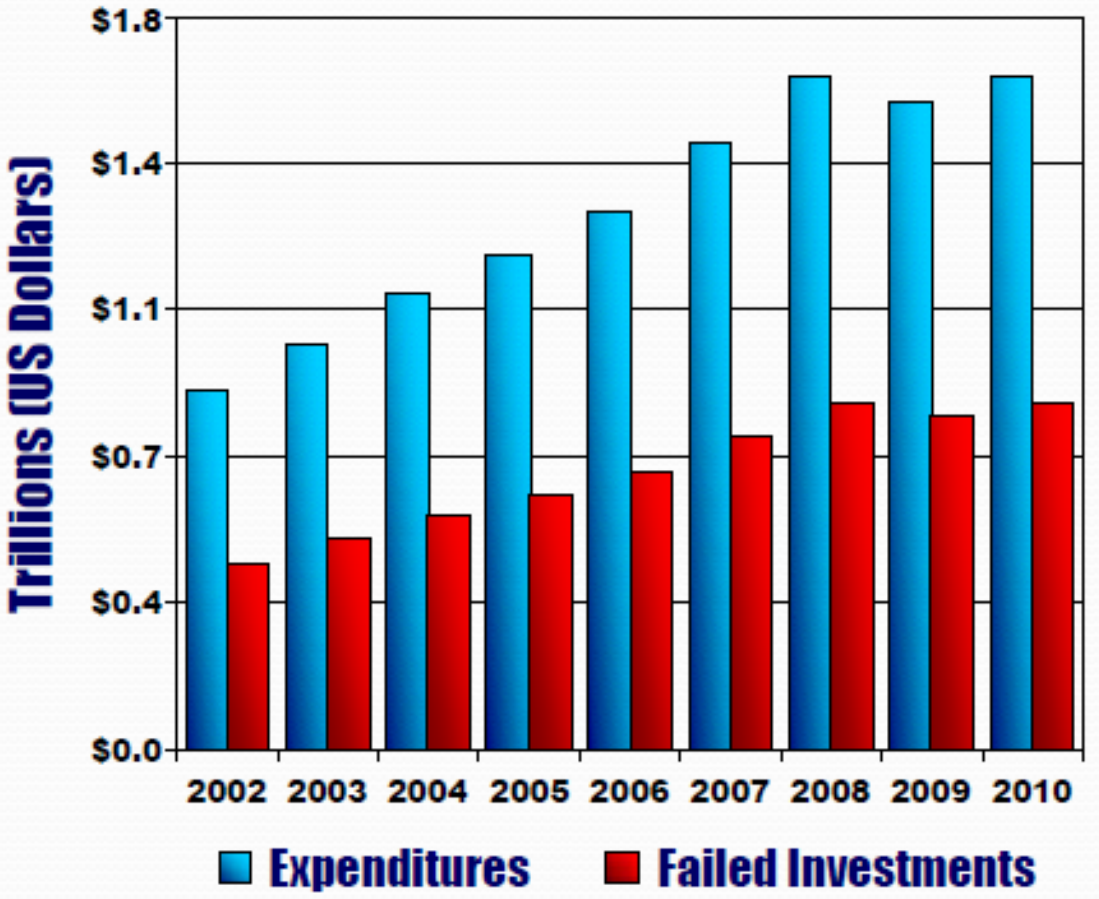
1.3.4 软件维护困难

- **理解**
  - ✓读懂程序比较困难，尤其是他人程序
- **修改**
  - ✓程序非常脆弱，牵一发而动全身
- **出错**
  - ✓改了以后易引入错误
- **发现**
  - ✓有了错误后难以发现

```
1  import jade.core.Agent;
10
11 public class MiningAgent extends Agent
12 {
13     private final static int EMPTY = 0;
14     private final static int GOLD = 1;
15     private final static int OBSTACLE = 2;
16     private final static int HOUSE = 3;
17     private final int DETECT = 4;
18     private final int EXPLORE = 5;
19     private final int PROVIDE = 6;
20     private final int SIZE = 15;
21
22     private MarUI ui = null;
23     private Gold gold = null;
24
25     public OneRoleAgent()
26     {
27         ui = MarUI.getUI();
28         gold = Gold.getGold();
29     }
30
31
32     public void setup()
33     {
34         ui.runInfo.setText(ui.runInfo.getText()+"加载采矿机器人. . . . "+"\n");
35         addBehaviour(new DetectGold());
36     }
37
38     public static Coordinate makeTarget(int[][] myMap ) {
39         int i = 0, tx =0, ty = 0;
40
41         for (; i < 1;) {
42             tx = (int) (Math.random() * 15);
```

1.3.5 失败风险很大





计算机软件开发的成功比例和失败投资

David F. Rico, PMP, ACP, CSM, Lean & Agile Software Methods, 2012.

1.4 如何来解决软件危机?

❑如何来解决软件危机?

- ✓策略、方法、理论、技术等

❑多方共同关注的问题

- ✓用户(如美国军方)
- ✓工业界(如IBM)
- ✓学术界（如研究学者)

软件危机的产生根源

- **对软件这样一类复杂和特殊系统的认识不清**
  - ✓软件是新生事物，对其特点、规律性和复杂性认识不够
- **没有找到支持软件系统开发的有效方法**
  - ✓基础理论、关键技术、开发过程、支撑工具等
- **缺乏成功软件开发实践以及相应的开发经验**
  - ✓系统总结、认真分析、充分借鉴、吸取教训

软件开发迫切需要理论和方法指导，软件工程应运而生！

内容

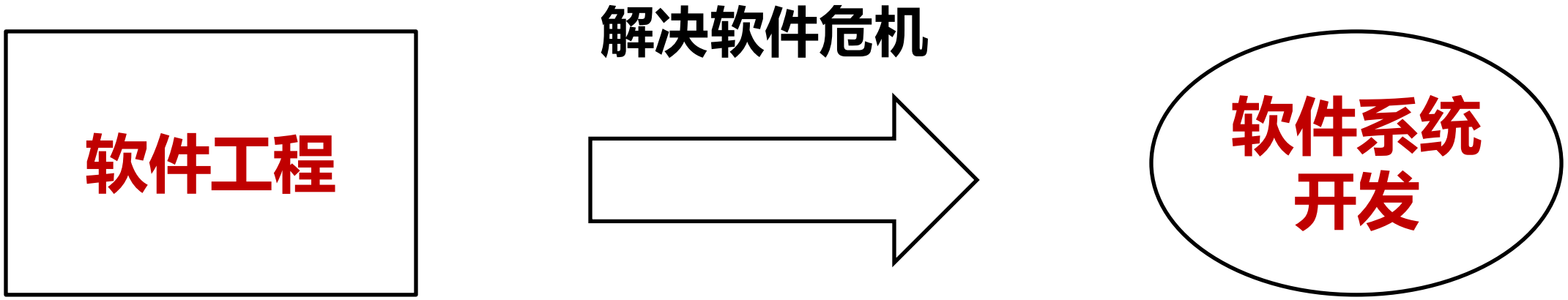
1. 软件工程产生背景
   - ✓软件危机的表现及根源
2. **软件工程基本内涵**
   - ✓思想、要素、目标和原则
3. **软件工程发展历程**
   - ✓不同发展阶段的成果及特点
4. **软件工程教育特点**
   - ✓教育规范、知识体系和课程特点

2.1 软件工程的诞生

- 时间：1968年
- 地点：西德南部小城
- 事件：NATO科技委出资召开的会议
- 人物：11 个国家 50 位代表参加
- 主题：如何解决软件危机
- 成果：提出了软件工程

提出了软件工程的概念及思想，标志着软件工程的诞生

软件工程产生的动机



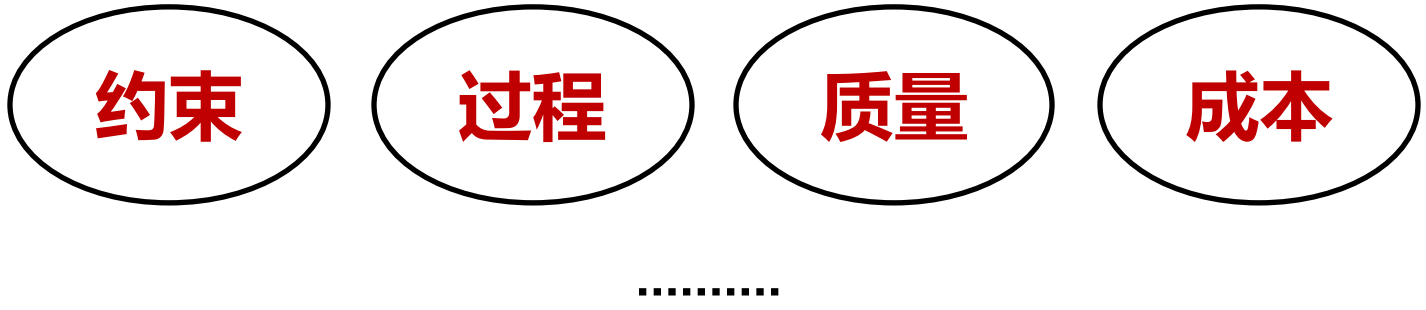
- 快速
- 高效
- 低成本
- 高质量

2.2 何为软件工程?

- **将系统的、规范的、可量化的方法应用于软件的开发、运行和维护的过程；以及上述方法的研究 --** [*IEEE 93*]
  - ✓**系统化**：提供完整和全面的解决方法，包括目标、原则、过程模型、开发活动、开发方法和技术等
  - ✓**规范化**：支持各类软件系统的开发，包括语言标准、质量标准、编程标准、方法标准、能力极其改进标准等
  - ✓**可量化**：工作量、成本、进度、质量等要素可以量化

软件工程对软件开发的新认识

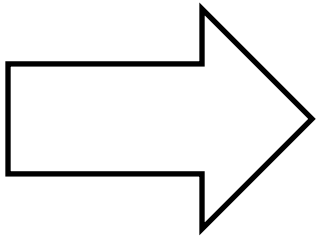
- **软件是产品(Product)**
  - ✓面向用户，存在质量、成本、利润等特征
- **软件开发是一项工程(Project)**
  - ✓存在约束，需要质量保证，进行组织管理，……
- **要按工程化方法来组织软件生产**
  - ✓分阶段分步骤来实施
  - ✓按计划开展开发活动
  - ✓进行各种形式质量保证
  - ✓采用行之有效的方法
  - ✓借助各种工具的支持……



软件开发方式的改变

- 从个体作坊式行为 ==》基于团队的协同开发方式

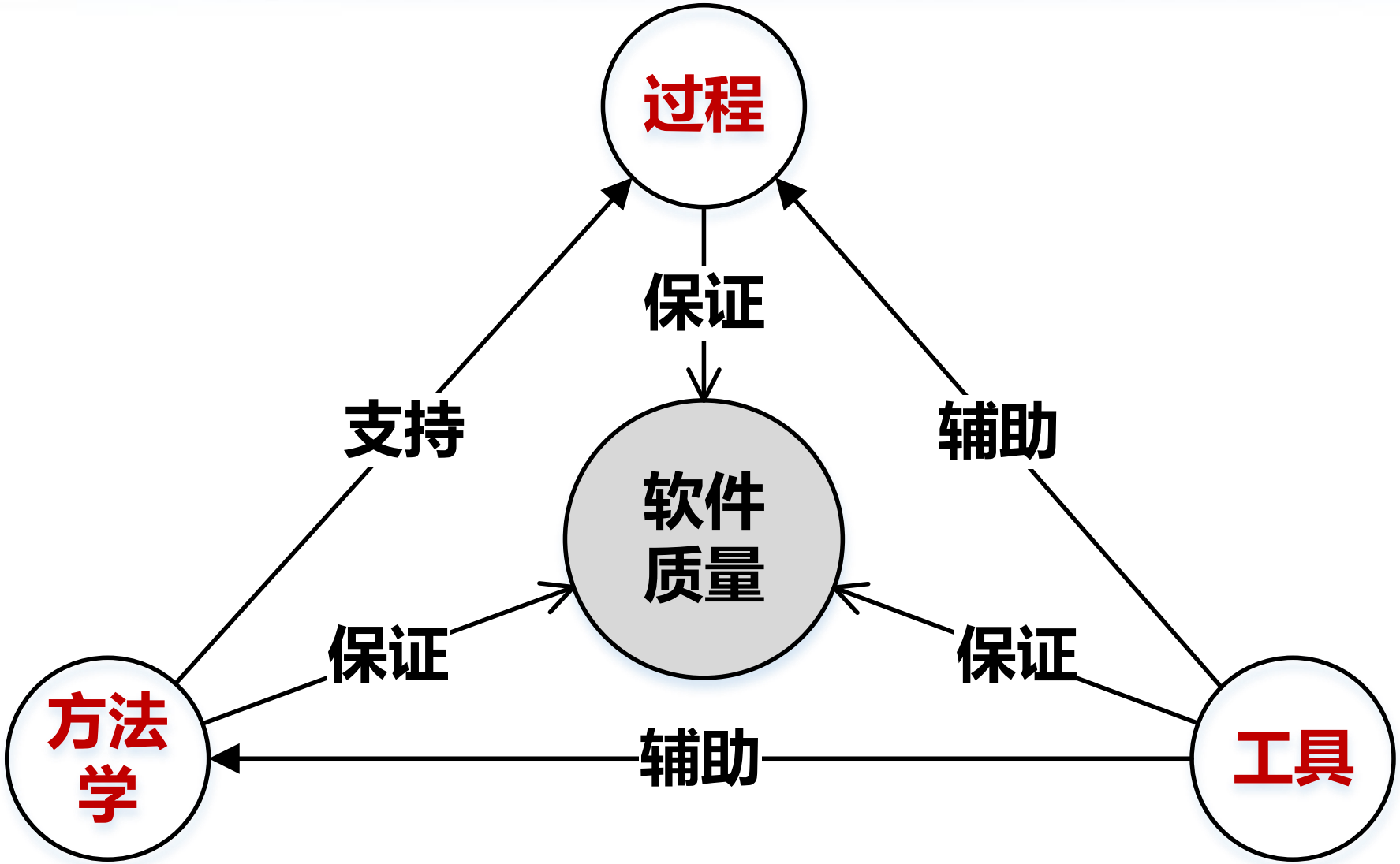
作坊式的个体编程创作



基于团队的协同开发

- ✓ 团队协作
- ✓ 分步实施
- ✓ 质量保证
- ✓ 开发技术
- ✓ 支持工具
- ✓ ……

2.3 软件工程的三要素



2.3.1 过程(Process)

- 从**管理**的视角，回答软件开发、运行和维护需要开展**哪些工作、按照什么样的步骤和次序**来开展工作
- 对软件开发过程所涉及的人、制品、质量、成本、计划等进行有效和可量化的管理
- 典型成果
  - ✓过程模型，如瀑布模型、增量模型、原型模型、迭代模型、螺旋模型等等
  - ✓方法，如敏捷开发方法、群体化开发方法、DevOps方法
  - ✓管理，如配置管理、质量管理、团队组织等

2.3.2 方法学(Methodology)

- 从**技术**的视角，回答软件开发、运行和维护**如何做**的问题
- 为软件开发过程中的各项开发和维护活动提供**系统性、规范性**的技术支持
  - ✓如何理解和认识软件模型是什么
  - ✓如何用不同抽象层次的模型来描述软件制品
  - ✓采用什么样的建模语言来描述软件模型等等
- **典型成果**
  - ✓结构化软件开发方法学
  - ✓面向对象软件开发方法学
  - ✓基于构件的软件开发方法学

2.3.2 工具(Tool)

- 从**工具辅助**的视角，主要回答如何借助工具来**辅助软件开发、运行和维护**的问题
- 帮助软件开发人员更为**高效地**运用软件开发方法学来完成软件开发过程中的各项工作，提高软件开发效率和质量，加快软件交付进度。
  - ✓如需求分析、软件设计、编码实现、软件测试、部署运行、软件维护、项目管理、质量保证等，简化软件开发任务，
- 典型成果
  - ✓SonarQube、Eclipse、Visual Studio等

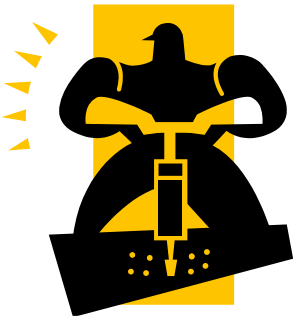
2.4 计算机辅助软件工程

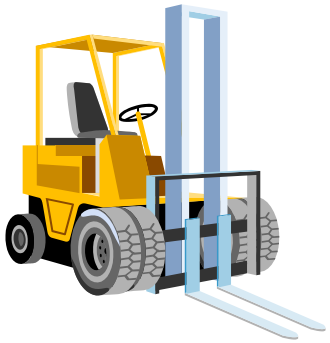
❑什么是计算机辅助软件工程(Computer-Aided Software Engineering, CASE)

- ✓在软件工程活动中，开发人员按照软件工程的方法和原则，借助于**计算机及其软件**的帮助来开发、维护和管理软件产品的过程

❑为什么需要计算机辅助软件工程

- ✓软件及其开发很复杂，简化开发
- ✓产品数量多难以管理，提高效率
- ✓人的因素决定易出错，提高质量

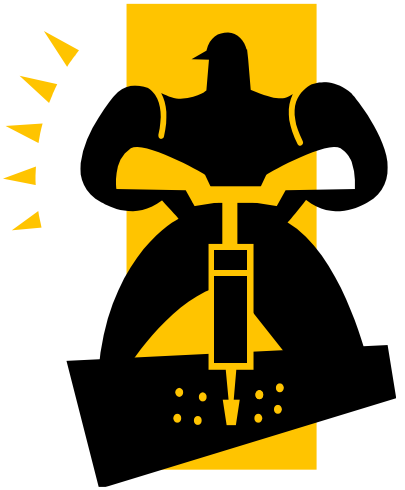




工欲善其事必先利其器

CASE工具和环境

❑CASE工具



- ✓支持CASE的软件工具
- ✓如编辑器、编译器等
- ✓具有单一性

❑CASE环境

- ✓将CASE工具按统一标准和接口组装起来，使工具间、人员间、各个过程间能方便交互的集成环境
- ✓如Visual Studio将编辑、编译、调试、界面设计、安装程序生成等等集成在一起

计算机辅助软件工程工具

- **代码编写**
  - ✓编辑、编译、分析、查找、代码生成等
- **项目管理**
  - ✓工作量和成本估算、制定和跟踪计划、配置和版本管理
- **软件建模**
  - ✓需求建模、UML建模、数据建模等
- **软件测试**
  - ✓测试用例自动生成、代码测试、缺陷报告等
- **软件运维**
  - ✓软件运行，管理和维护

讨论：你知道的CASE工具和环境

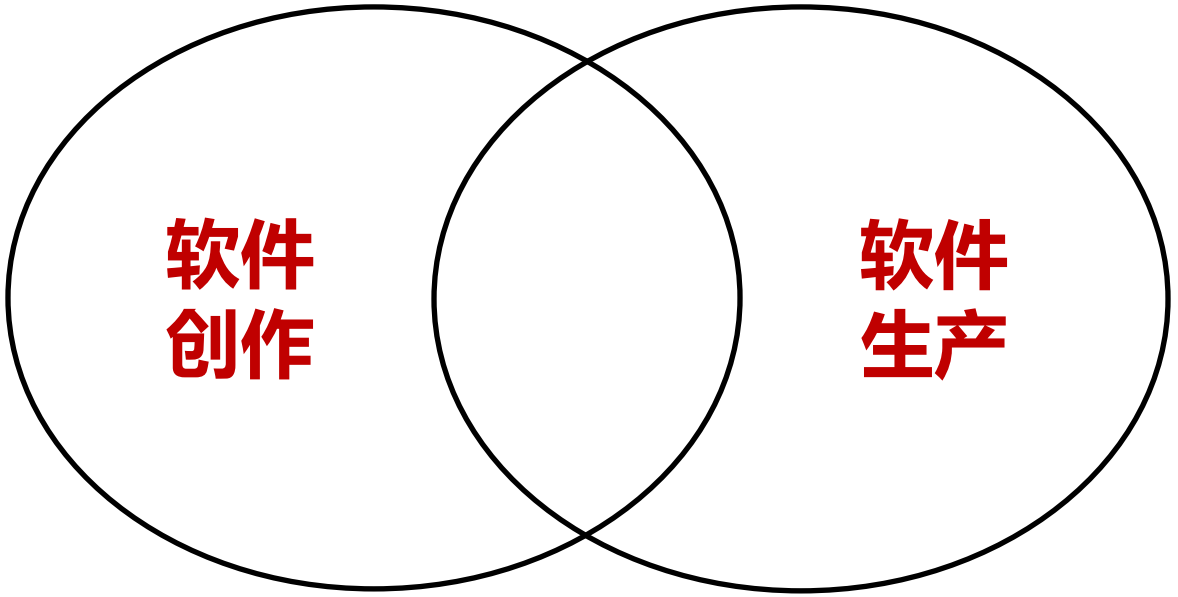
- 什么样的CASE工具和环境？提供了哪些功能和服务？发挥了什么样的作用？

- 没有CASE工具能够进行软件开发吗？

2.5 软件开发的本质

软件开发 = 软件创作 + 软件生产

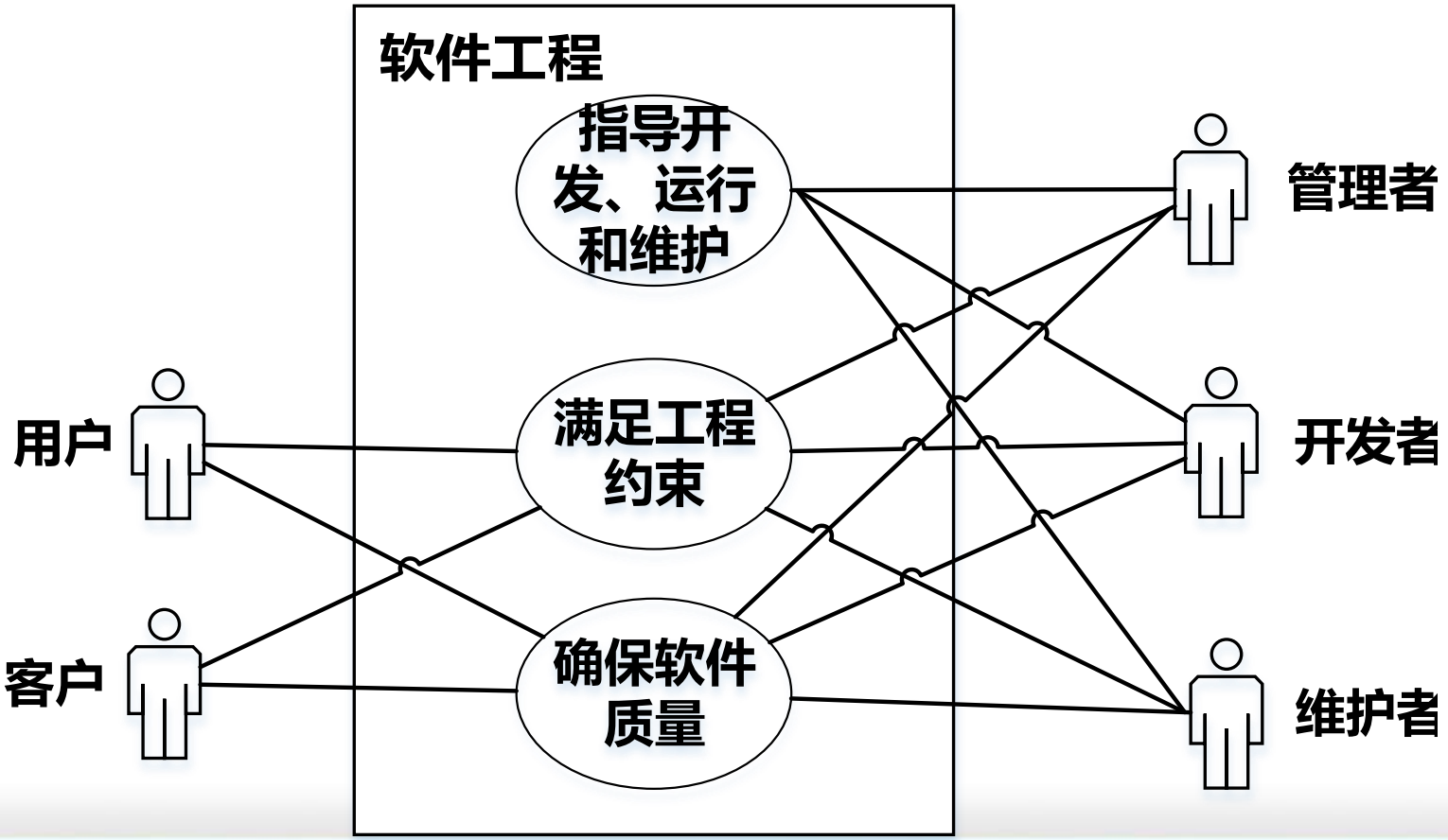
基于**软件开发者的经验和技能，借助于智慧**，进行自由创新，如软件设计、编码实现等



基于**工程化的手段，遵循约束和规范**，开展软件生产，如遵循过程、按照标准、质量保证等

2.6 软件工程的目标

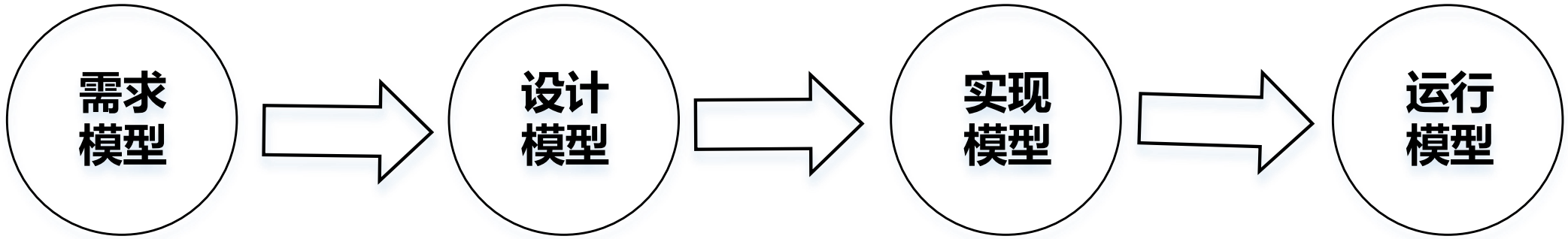
- 在成本、进度等**约束**下，指导软件开发和运维，开发出**满足用户要求**的**足够好**软件



2.7 软件工程原则（1）

❑抽象和建模

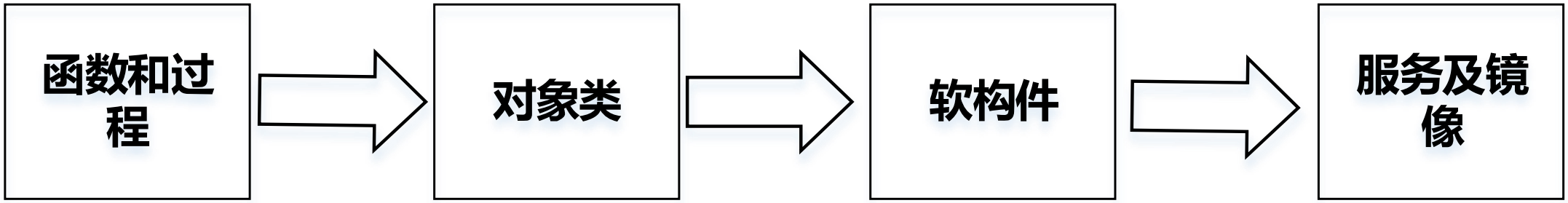
- ✓**抽象**：将与相关开发活动所关注的要素提取出来，不关心的要素扔掉，形成与该开发活动相关的软件要素
- ✓**建模**：基于特定的抽象，借助于**建模语言**（如数据流图、UML等），建立起基于这些抽象的**软件模型**，进而促进对软件系统的准确理解



软件工程原则（2）

❑模块化

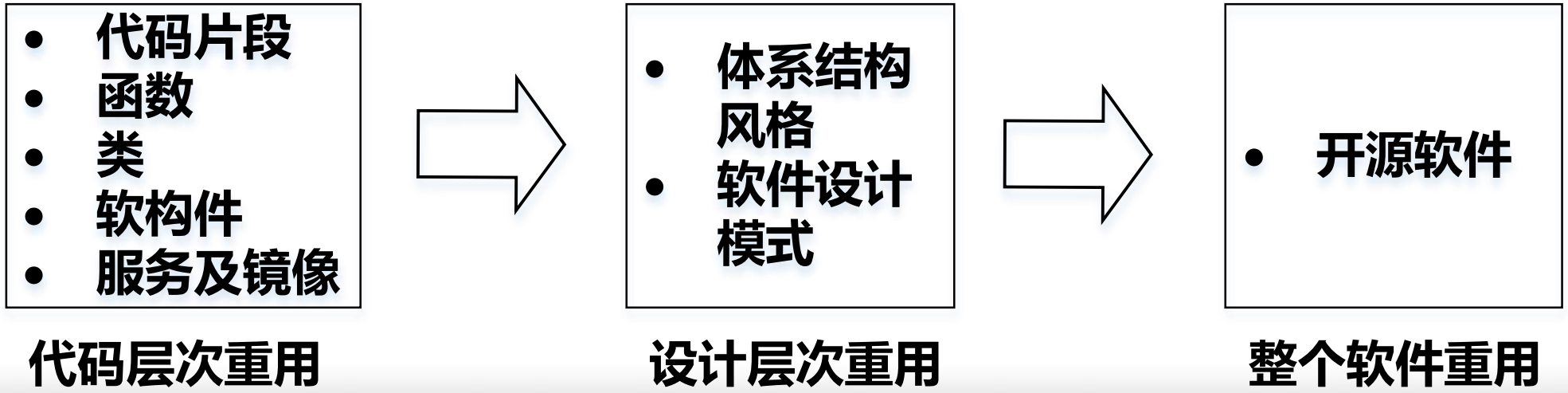
- ✓将软件系统的**功能分解**和实现为若干个模块，每个模块具有**独立的功能**，模块之间通过接口进行调用和访问。
- ✓**模块内部高内聚，模块间松耦合**



软件工程原则（3）

❑软件重用

- ✓在软件开发过程中尽可能**利用已有的软件资源和资产**（如函数库、类库、构件库、开源软件、代码片段等）来实现软件系统
- ✓努力开发出**可被再次重用**的软件资源（如函数、类、构件等）
- ✓有助于提高软件开发效率，降低软件开发成本，满足开发工程约束，得到高质量的软件产品



软件工程原则（4）

❑信息隐藏

- ✓模块内部信息（如内部的语句、变量等）对外**不可见或不可访问**，模块间仅仅交换那些为完成系统功能所必需交换的信息（如接口）
- ✓模块设计时**只对外提供可见的接口**，不提供内部实现细节。信息隐藏原则可提升模块的独立性，减少错误向外传播，支持模块的并行开发

```
/*
 * Copyright (c) 2010-2011, The MiCode Open Source Community (www.micode.net)
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *        http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package net.micode.notes.data;

import android.content.Context;

public class Contact {
    private static HashMap<String, String> sContactCache;
    private static final String TAG = "Contact";

    private static final String CALLER_ID_SELECTION = "PHONE_NUMBERS_EQUAL(" + Phone.NUMBER
    + ",?) AND " + Data.MIMETYPE + "='" + Phone.CONTENT_ITEM_TYPE + "'"
    + " AND " + Data.RAW_CONTACT_ID + " IN "
            + "(SELECT raw_contact_id "
            + " FROM phone_lookup"
            + " WHERE min_match = '+')";

    public static String getContact(Context context, String phoneNumber) {
        if(sContactCache == null) {
            sContactCache = new HashMap<String, String>();
        }

        if(sContactCache.containsKey(phoneNumber)) {
            return sContactCache.get(phoneNumber);
        }

```

软件工程原则（5）

❑关注点分离

- ✓在软件开发过程中，将若干性质不同的**关注点分离**开来，以便在不同的开发活动中针对不同的关注点，随后将这些关注点的开发结果整合起来，形成关于软件系统的完整视图
- ✓软件系统具有**多面性的特点**，既有结构特征，如软件的体系结构，也有行为特征，如软件要完成的动作及输出的结果；既有高层的需求模型，描述了软件需要做什么，也有低层的实现模型，描述了这些需求是如何实现的
- ✓使得开发者在**每一项开发活动中聚焦于某个关注点**，有助于简化开发任务；同时通过**整合多个不同视点**的开发结果，可获得关于软件系统的更为清晰、系统和深入地认识

软件工程原则（6）

❑分而治之

- ✓在软件开发和维护过程中，软件开发人员可**对复杂软件系统进行分解**，形成一组子系统
- ✓如果子系统仍很复杂，还可以继续进行分解，及至通过分解所得到的子系统易于处理；然后通过**整合**子系统的问题解决得到整个系统的问题解决
- ✓有助于简化复杂软件系统的开发，降低软件开发复杂性，从而提高软件开发效率，确保复杂软件系统的质量。

软件工程原则（7）

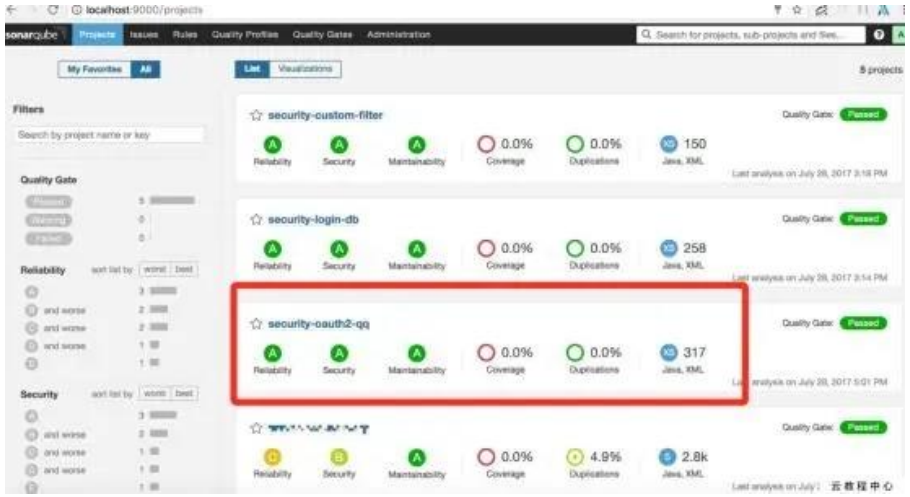
❑双向追踪原则

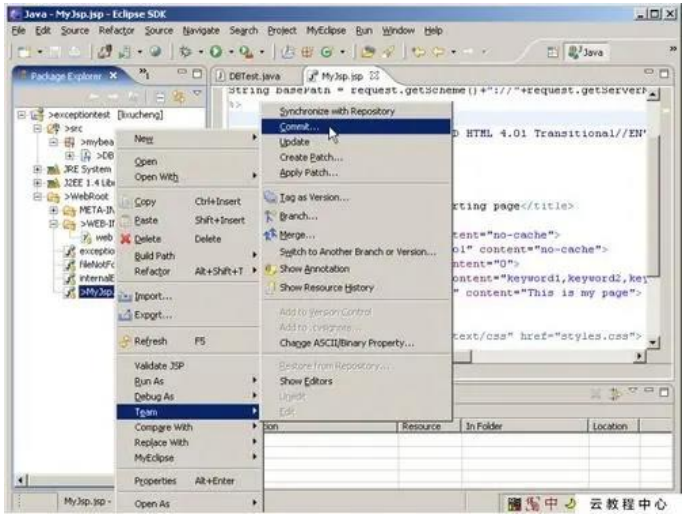
- ✓当某个软件制品**发生变化**时，一方面要追踪这种**变化会对那些软件制品产生影响**，进而指导相关的开发和维护工作，此为正向追踪；另一方面要**追踪产生这种变化的来源**，或者说是什么因素导致了该软件制品的变化，明确软件制品发生变化的原因及其合理性，此为反向追踪。
- ✓有助于确保软件制品间的一致性，发现无意义的变化，并基于变化指导软件的开发和维护，确保软件质量。

软件工程原则（8）

❑工具辅助

- ✓利用软件工具来辅助软件开发和维护工作是一项行之有效的方法
- ✓尽可能地借助计算机工具来辅助软件开发和维护，以降低开发者和维护者的工作负担，提高软件开发和维护效率，提升软件开发及软件制品的质量





工欲善其事必先利其器

讨论：编写程序用到的软件工程原则

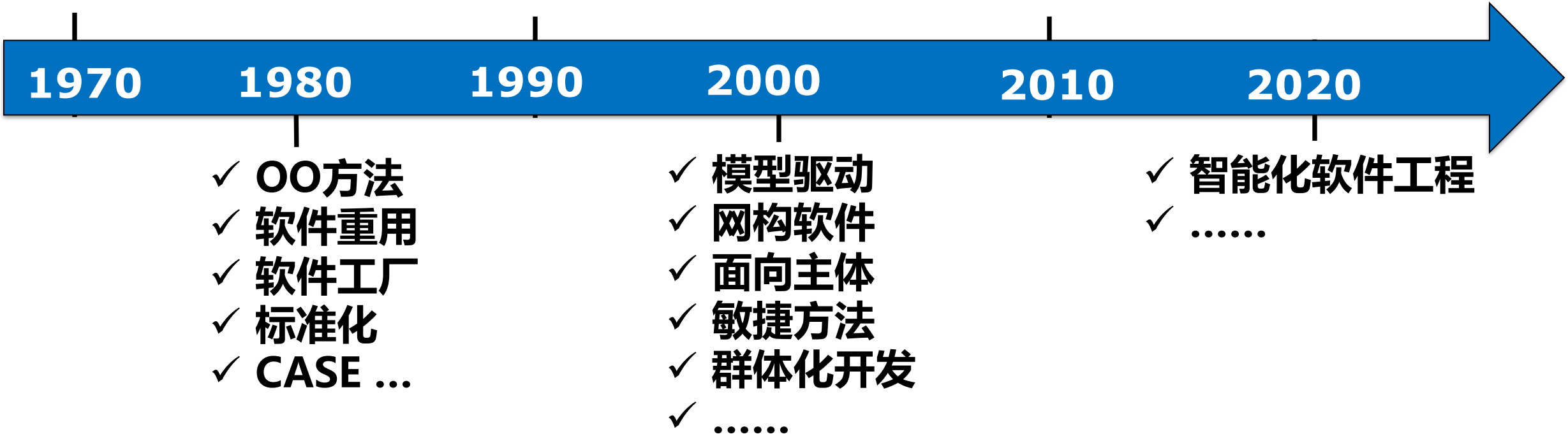
- 在编写程序代码的过程中，你用到了哪些软件工程原则？这些软件工程原则在编程中发挥了什么作用？

抽象和建模、模块化、软件重用、信息隐藏、关注点分离、分而治之、双向追踪、工具辅助

内容

1. 软件工程产生背景
   - ✓软件危机的表现及根源
2. 软件工程基本内涵
   - ✓思想、要素、目标和原则
3. **软件工程发展历程**
   - ✓不同发展阶段的成果及特点
4. **软件工程教育特点**
   - ✓教育规范、知识体系和课程特点

3.1 软件工程的发展历程



20世纪50-60年代

- 软件系统较为简单，计算机软件与硬件结合的非常紧密
- “精雕细琢”程序代码，以充分利用宝贵的计算资源
- 出现了黑客文化，倡导自由
- 成功的案例
  - ✓IBM OS/360软件系统的成功研制并投入商业使用
- 出现了高级程序设计语言
  - ✓如Fortran、COBOL、LISP、ALGOR等
- 这一时期的软件开发手段落后，开发效率低，质量无法保证，进而引发了软件危机

20世纪70年代

- 计算机主机的**计算能力**得到了很大提升
- 计算机软件朝着商业应用拓展，需要处理繁杂的事务流程
- **程序设计语言和程序设计方法学**成为研究热点
  - ✓出现了诸如PASCAL、C、Prolog、ML等高级语言
- 产生了软件工程新技术
  - ✓提出了**瀑布软件开发过程模型，结构化软件开发方法学**
  - ✓**形式化方法**（Formal Method）的研究非常活跃
- 研制了一些支持结构化软件开发方法学、形式化方法的CASE工具和环境
- 开始采用定量方法来指导软件开发、管理和质量保证

20世纪80年代

- 计算机软件的应用领域和范围不断扩大，软件数量、系统规模和复杂性不断增长
- 产生了**面向对象程序设计技术**
  - ✓如Smalltalk、C++等
- 提出了**SW-CMM**，即**软件能力成熟度模型**
- **软件重用**被视为是解决软件危机的一条现实可行途径
- **CASE工具和环境**的研制和使用成为热点
- **软件工程标准化**工作非常活跃，成果丰硕

20世纪90年代

- 局域计算环境开始流行，互联网应用开始出现
- OOP技术趋于成熟，面向对象分析和设计方法学的研究非常活跃，逐步形成系统化的面向对象软件工程
  - ✓制定面向对象建模语言UML，产生了统一软件开发过程RUP
- 软构件技术得到了快速发展，萌生了软件体系结构和软件设计模式的研究与实践
- 开源软件及技术开始出现
- 人机交互技术取得长足进步

21世纪前十年

- 互联网技术日趋成熟，信息技术快速发展，软件数量不断增长，越来越多的软件部署在互联网上运行并提供服务
- 产生了网构软件技术、自适应软件工程、可信软件技术、面向主体软件工程等
- 群体化软件开发技术在开源软件开发实践中广泛应用
- 面向服务软件工程的研究与实践
- 敏捷开发方法的在软件开发中的应用
- 模型驱动软件开发技术的研究与应用
- 软件可信技术的研究与实践非常活跃

近十年

- 移动互联网得到了快速发展，应用软件需求激增，信息系统的人机物融合趋势日趋突出
- 人类正进入到软件定义一切的时代，从而对软件开发和运维提出了严峻的挑战
- 越来越多的企业和个人参与开源软件实践，涌现形成了规模极为庞大的开源软件生态
- DevOps方法在软件产业界和软件开发实践中的广泛应用
- 智能化软件开发技术研究活跃

3.2 我国软件工程发展

- **起步于1980年前后，过去四十多年取得了长足的进步**
  - ✓1980s，软件自动化开发和形式验证
  - ✓1980s，软件开发方法学和CASE工具及集成环境
  - ✓21世纪初期，网构软件技术
  - ✓21世纪以来，可信软件技术
  - ✓近10多年来，开源软件研究与实践

3.3 软件工程发展的时代特点

- 与时代的计算技术和信息技术的发展有着紧密的联系
- 从20世纪60年代到80年代中后期，部署和运行在单机或主机上
- 从20世纪80中后期到90年代，个人计算机和局域网的出现
- 进入21世纪，随着互联网应用的不断普及，更多的软件系统部署和运行在动态、难控和不确定的互联网环境上
- 2010年以来，随着泛在计算环境和移动互联网应用的不断普及，信息系统的人机物融合特征日益突出

软件工程发展的技术特点

- **软件抽象的层次越来越高**
  - ✓二进制编码、结构化编程、结构化开发方法学、面向对象编程、面向对象开发方法学,……
- **软件重用的粒度越来越大**
  - ✓函数和过程，类，构件，开源软件，……
- **软件开发理念的不断变化**
  - ✓以文档为中心与以代码为中心
  - ✓从个体、团队到群体的开发组织
  - ✓从还原论到演化论

软件工程的多学科交叉

- ✓ 认识大型复杂软件系统
- ✓ 揭示和解释内在的机理
- ✓ 指导方法的研究与实践
- ✓ ……

软件工程的变与不变

- 目标
  - ✓ 软件危机
- 基本原则
  - ✓ 软件重用
  - ✓ 模块化
  - ✓ 问题分解
  - ✓ ….

不变

- 对软件的理解和认识
  - ✓ 社会技术视点
- 学科交叉
  - ✓ 交叉更多的学科
- 方法和手段
  - ✓ 敏捷方法
  - ✓ 群体化开发
  - ✓ ….

变

内容

1. 软件工程产生
   - ✓软件危机的表现及根源
2. 软件工程概念
   - ✓思想、要素、原则和目标
   - ✓软件开发 = 软件创作 + 软件生产
3. 软件工程发展
   - ✓发展历程及特点
4. **软件工程教育**
   - ✓教育规范和知识体系

4.1 软件工程师的培养

- 领域和需求分析工程师
- 软件设计工程师
- 程序员
- 软件测试工程师
- 软件运维工程师
- 软件项目管理人员

- 需要具备多方面的能力
  - ✓ 创新能力
  - ✓ 系统能力
  - ✓ 解决复杂工程问题能力
  - ✓ ……

4.2 软件工程的知识领域

- **IEEE SWEBOK V3.0 (SoftWare Engineering Body of Knowledge)**
  - ✓知识域(Knowledge Area: KA)

- ✓软件需求
- ✓软件设计
- ✓软件构造
- ✓软件测试
- ✓软件维护
- ✓软件配置管理
- ✓软件工程管理
- ✓软件工程过程
- ✓软件工程建模与方法
- ✓软件质量
- ✓软件工程专业实践
- ✓软件工程经济学
- ✓计算基础
- ✓数学基础
- ✓工程基础

“软件工程专业”开设的课程

- 计算机程序设计
- 计算机程序设计课程设计
- 软件工程
- 软件工程综合实践
- 软件项目管理
- 软件体系结构与设计
- 软件需求工程
- 软件测试与验证
- 人机交互
- 计算机原理
- 离散数学
- 编译原理
- 操作系统
- 计算机网络
- 人工智能导论
- 数据库原理与技术
- 数值分析

4.3 软件工程课程的特点

课程特点

- 内容“虚”
  - 针对软件复杂逻辑系统
  - 思想性和抽象教学内容
- 要求“实”
  - 掌握软件工程实践能力
  - 解决软件开发实际问题

教学难点

- 难听懂、不易学
  - 知识点抽象，难以讲透
  - 不易理解和掌握
- 知难做更难
  - 运用知识来开发软件难
  - 开发出高质量的软件难

教学重点

教学不能停留于知识讲授，不能纸上谈兵
实践教学是课程教学重点，也是上好这门课的关键

小结

- **软件工程产生的背景和目的**
  - ✓软件危机，持续存在，关注点不同
- **软件工程的本质**
  - ✓软件视为产品，软件开发视为工程、创作和生产相结合的过程
  - ✓三要素：过程、方法学和工具
  - ✓软件工程的基本原则
- **软件工程的发展**
  - ✓不同阶段和时期，不同的思想和技术
- **软件工程教育**
  - ✓软件工程知识体系及课程特点

综合实践1

- **任务：理解和分析开源软件的整体情况**
- **方法**
  - ✓运行开源软件，理解软件功能；泛读开源代码，分析代码的构成，绘制出软件系统的体系结构图；利用SonarQube工具分析开源代码的质量情况
- **要求**
  - ✓理解开源软件提供的功能和服务，掌握软件系统的模块构成，分析开源软件的质量水平
- **结果：**
  - ✓（1）软件需求文档；（2）软件体系结构图，描述开源软件的模块构成；（3）SonarQube的开源软件质量报告

综合实践2

- **任务：分析相关行业和领域的状况及问题。**
- **方法**
  - ✓选择你所感兴趣的行业和领域（如老人看护、防火救灾、医疗服务、出行安全、婴儿照看、机器人应用等），开展调查研究，分析这些行业和领域的当前状况和未来需求，包括典型的应用、采用的技术、存在的不足和未来的关注。
- **要求**
  - ✓调研要充分和深入，分析要有证据和说服力。
- **结果**
  - ✓行业和领域调研分析报告。

课后实践

- **构想软件需求，形成一页纸（不超1000字）描述**
  - ✓名字：概括软件系统
  - ✓问题：描述该软件欲解决的实际问题
  - ✓方法：描述如何通过软件及相关系统来解决问题
  - ✓举例：举一个使用该软件解决问题的应用案例和具体场景
  - ✓功能：软件大致有哪些功能?
  - ✓设备：软件需要与哪些设备进行集成
- **文档格式可参见模板**

问题和讨论