软件工程概述

软件工程与实践课程组

电子科技大学信息与软件工程学院

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

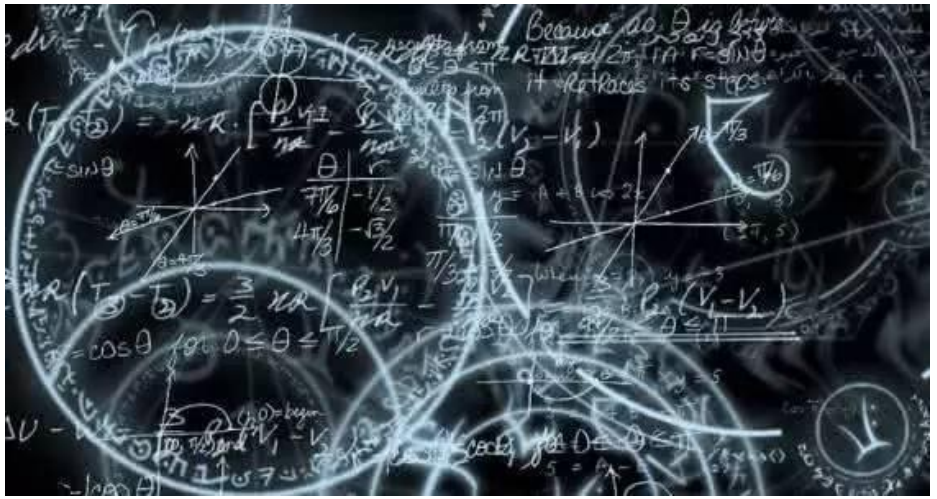
✓思想、要素、目标和原则

3. 软件工程发展历程

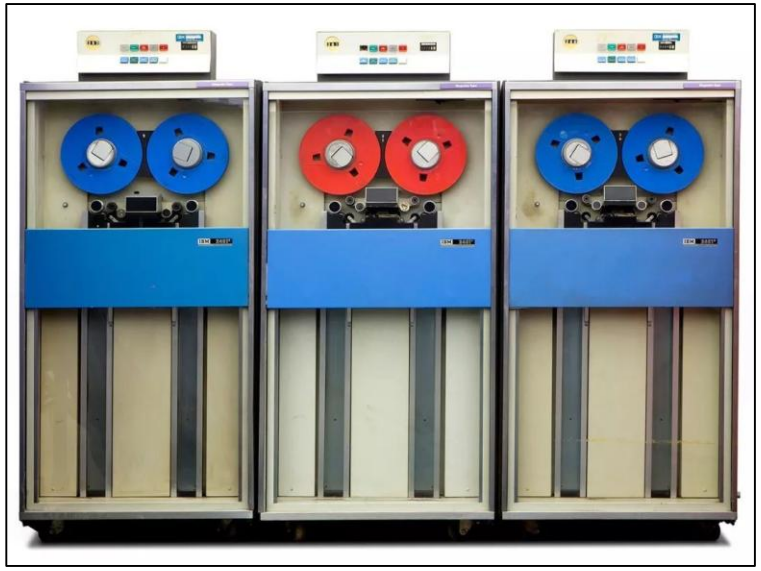
✓不同发展阶段的成果及特点

1.1 1950s-1960s的计算机软件应用背景

- **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

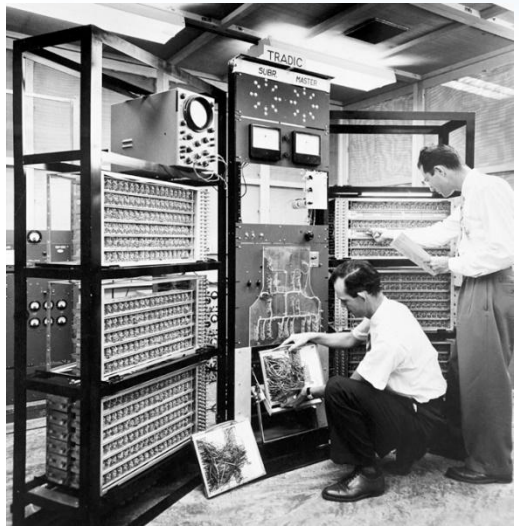


IBM 360 OS软件开发实践及其面临的挑战



- **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的软件开发特点：个体作坊式软件开发



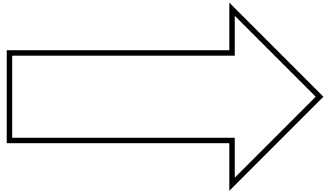
作坊式的个人创作

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

1.2 个体作坊式创作面临的问题和挑战

作坊式的个体编程开发



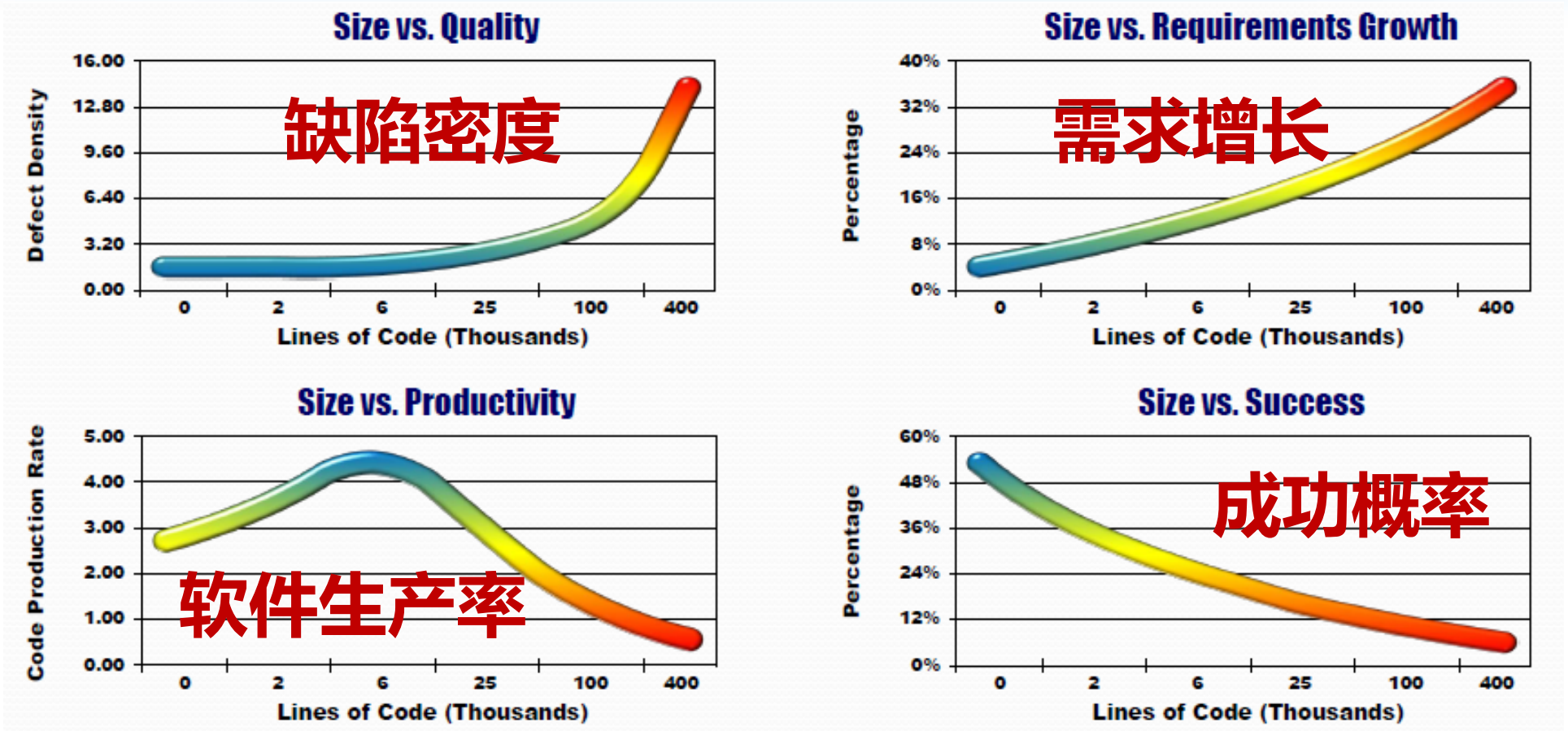
大批量和大规模软件系统的开发

- 作坊式和手工式软件开发存在哪些问题?
- 作坊式软件开发能否应对大规模软件开发要求?

作坊式软件开发需要解决的问题

- **开发过程**：按照什么样的步骤来开发软件系统
- **指导方法**：采用什么样的方法来指导软件开发活动和步骤
- **项目管理**：如何组织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

1.3 软件危机的出现

开发手段

作坊式和手工式的个体编程和创作

实际需求

大规模和复杂软件开发要求

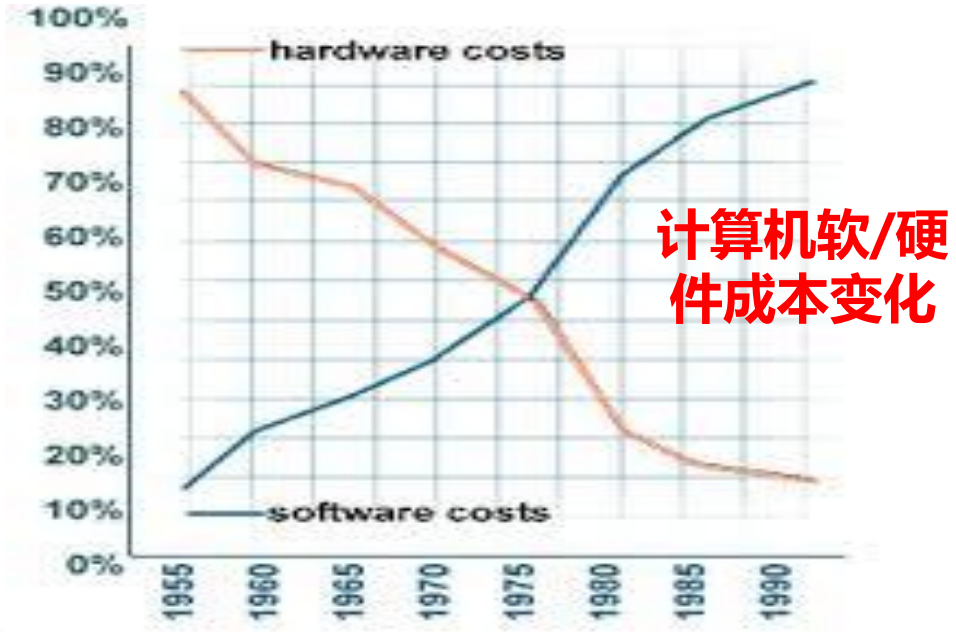
软件危机

- 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

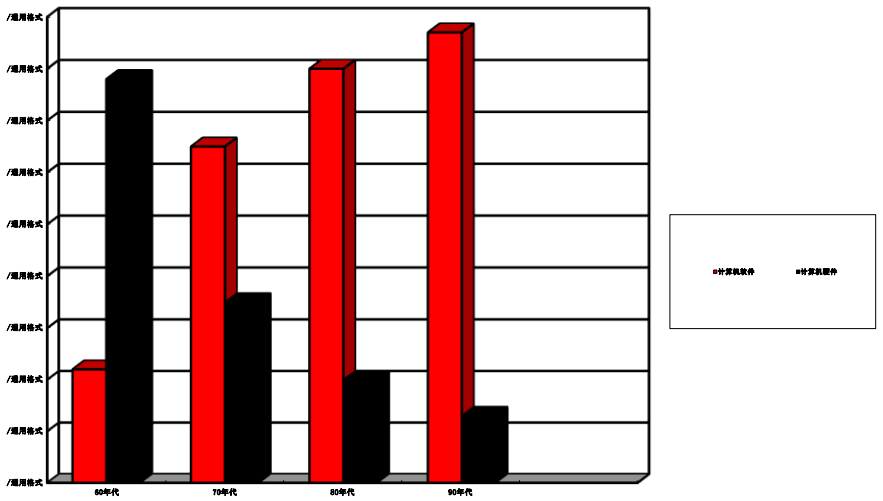
1.3.1 开发成本高

❑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

- 软件项目延期比比皆是
- 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

- 美国银行的信托软件系统原计划于1984年底前完成，但是及至1987年3月该软件系统仍未能交付给用户使用
- IBM尽管在OS/360系统中投入了多达2000多名软件工程师，耗资5000多个人年的工作量，但是该系统还是未能按期完成交付使用。

充分说明了人们对软件开发艰巨性认识不够、对开发工作量估算不准，软件开发效率低下

1.3.3 质量难以保证

- **软件存在诸多的错误和缺陷**
  - ✓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
  - ✓没有正确实现功能
  - ✓……
- **有些软件错误可能是致命的**

低质量的软件系统好似“定时炸弹”

软件质量低下带来的问题和后果

- 波音737-Max和教练机上的软件缺陷导致**机毁人亡**
- 美国银行的信托软件系统尽管最后交付使用，但由于软件系统运行不稳定，用户最终不得不**放弃该软件系统**
- IBM OS/360交付后仍有2000个以上的问题，**影响软件的使用**
- 亚丽安娜火箭中的软件缺陷导致火箭发射后就发生了爆炸，**经济损失严重**

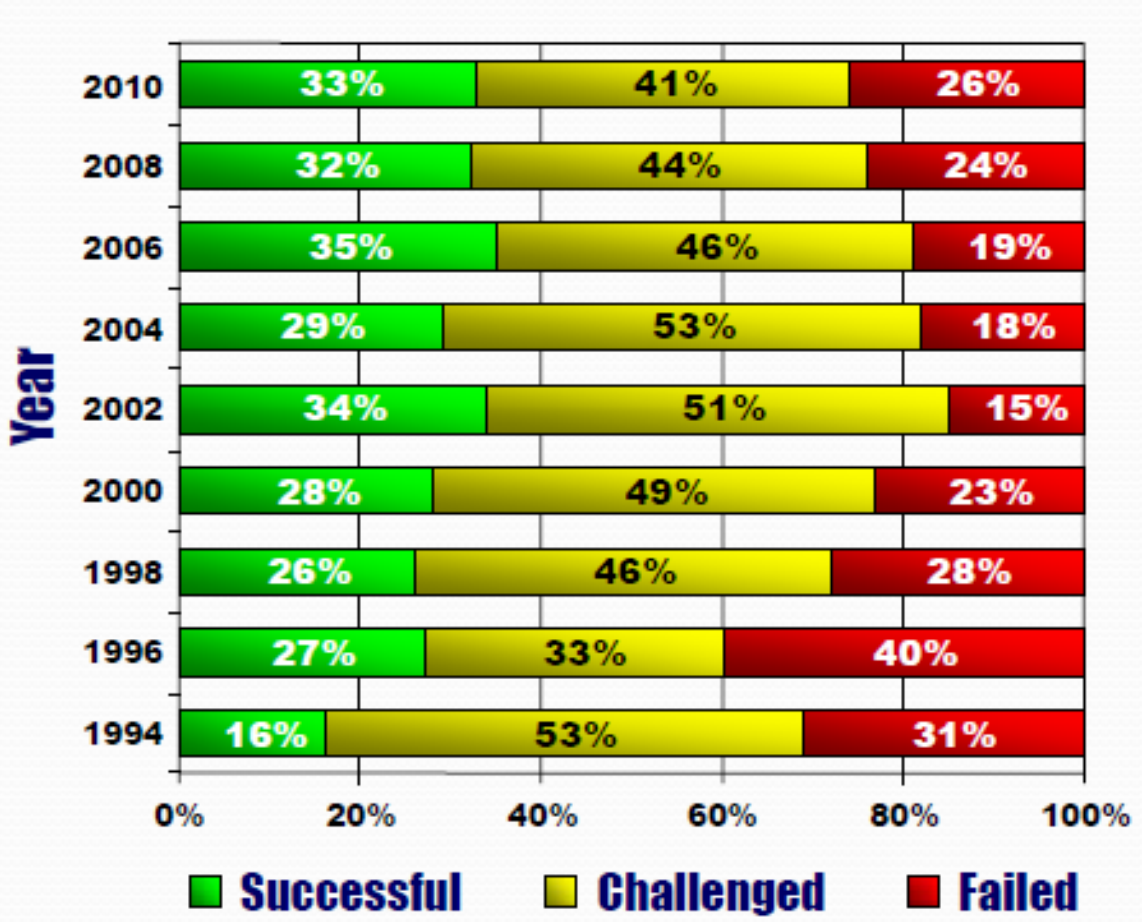
原软件用于教练飞机某参数屏显范围±100，重用于新战斗机，该参数屏显范围应该为±300！

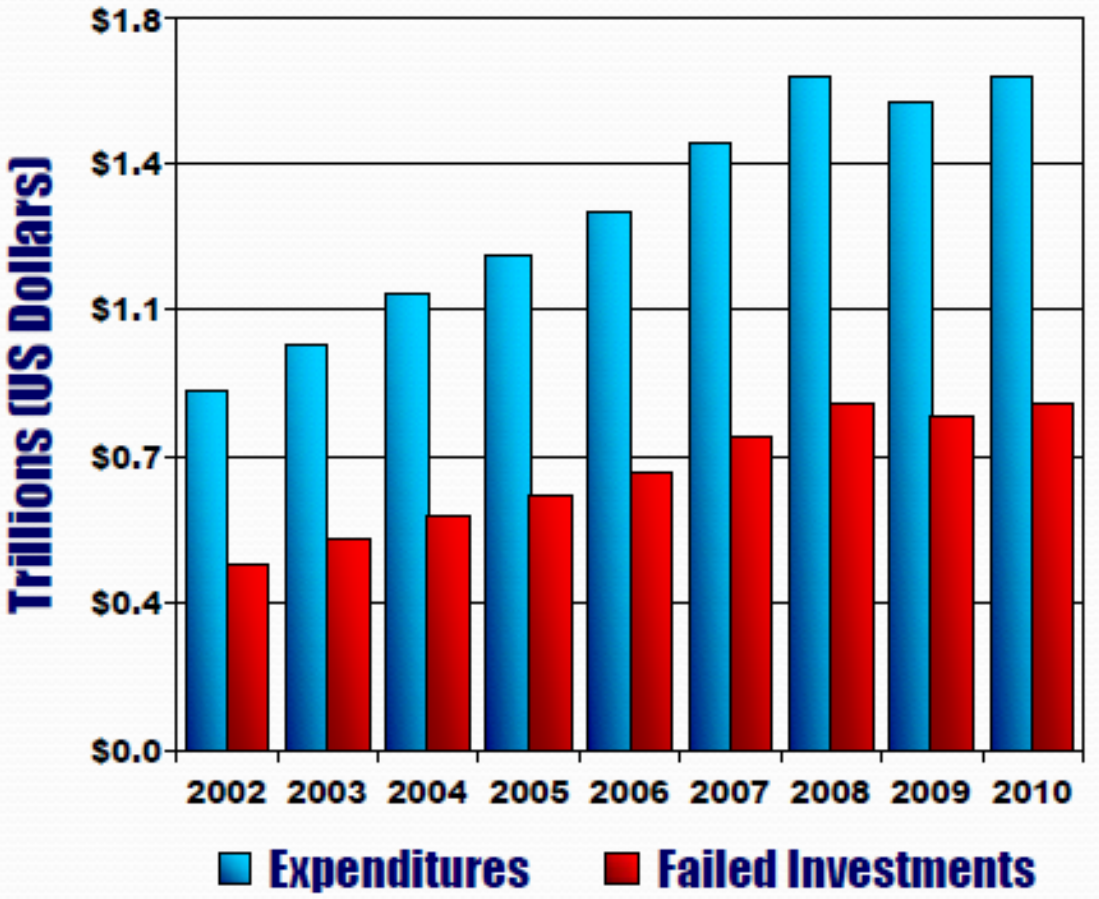
1.3.4 软件维护困难

- **难以理解**
  - ✓读懂程序比较困难，尤其是他人程序
- **不易修改**
  - ✓程序非常脆弱，牵一发而动全身
- **容易出错**
  - ✓改了以后易引入错误，产生新的问题
- **不易发现**
  - ✓有了错误后难以发现，缺陷非常隐蔽

```
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 软件项目失败风险很大





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 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！

如何解决软件危机?

- 需要寻求**新颖有效的方法**
  - ✓策略、方法、理论、技术等

- **多方共同关注的现实问题**
  - ✓用户（如美国军方）- 软件大户
  - ✓工业界（如IBM）- 软件开发商
  - ✓学术界（如研究学者）

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

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

软件工程

解决软件危机
促进软件开发

- 更快速
- 更高效
- 低成本
- 高质量

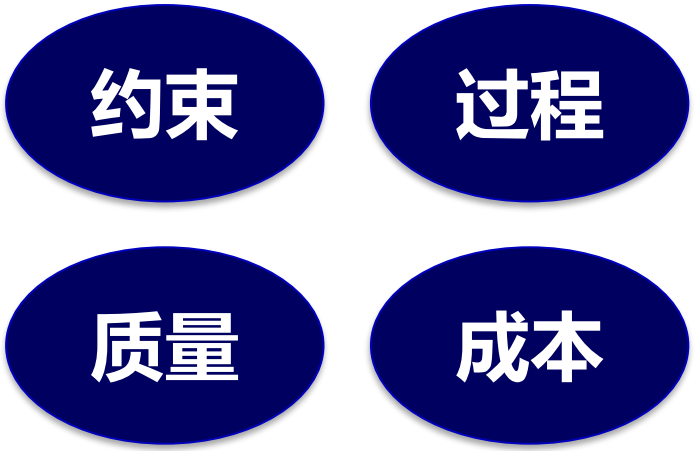
软件系统
开发

2.2 何为软件工程?

- **将系统化、规范化、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**：规范化软件系统的开发，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**：为软件开发和管理提供量化的支持，如工作量、成本、进度、质量等要素的量化

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（Engineering）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……

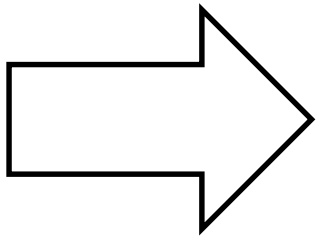


软件开发思想和方式的改变

❑基于个体的作坊式创作 ==》基于团队的协同开发

- 单枪匹马
- 独立创作
- 基于编程
- 手工方式
- 忽视质量
- ……

基于个体的作坊式创作

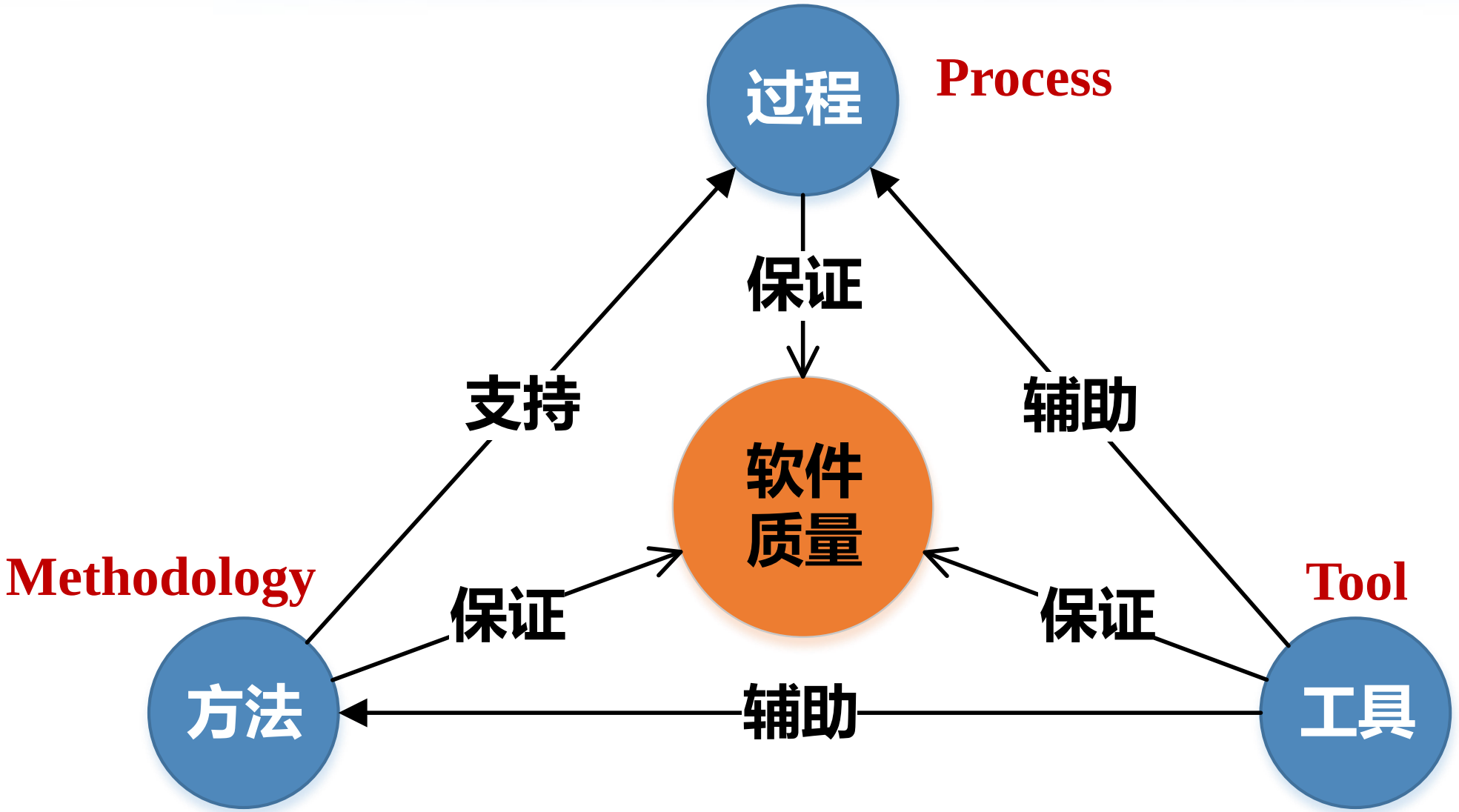


基于团队的协同开发

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

对软件开发认识上的变化

2.3 软件工程的三要素



- 过程
- 方法
- 工具

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**
  - ✓如何一步一步地进行软件系统的开发
  - ✓每一个步骤要完成什么样的工作、产生怎样的软件制品，不同步骤间存在什么样的先后次序和逻辑关系

- **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

为软件开发的开展提供过程和步骤及其管理手段

2.3.2 方法(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

为软件开发提供方法指导和技术支持

2.3.3 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

“工欲善其事必先利其器”

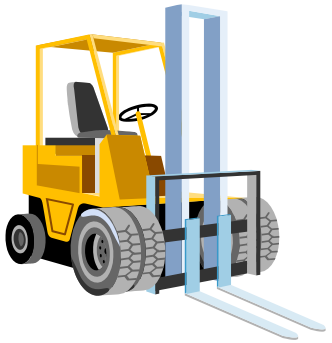
2.4 计算机辅助软件工程

❑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，**提升效率**
- ✓人的因素决定易出错，**提高质量**

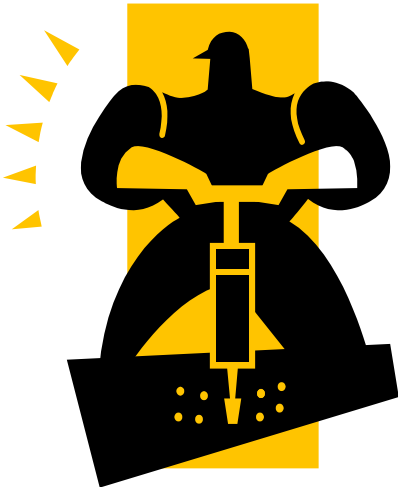


工欲善其事必先利其器

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
- ✓具有单一性特点



❑CASE环境

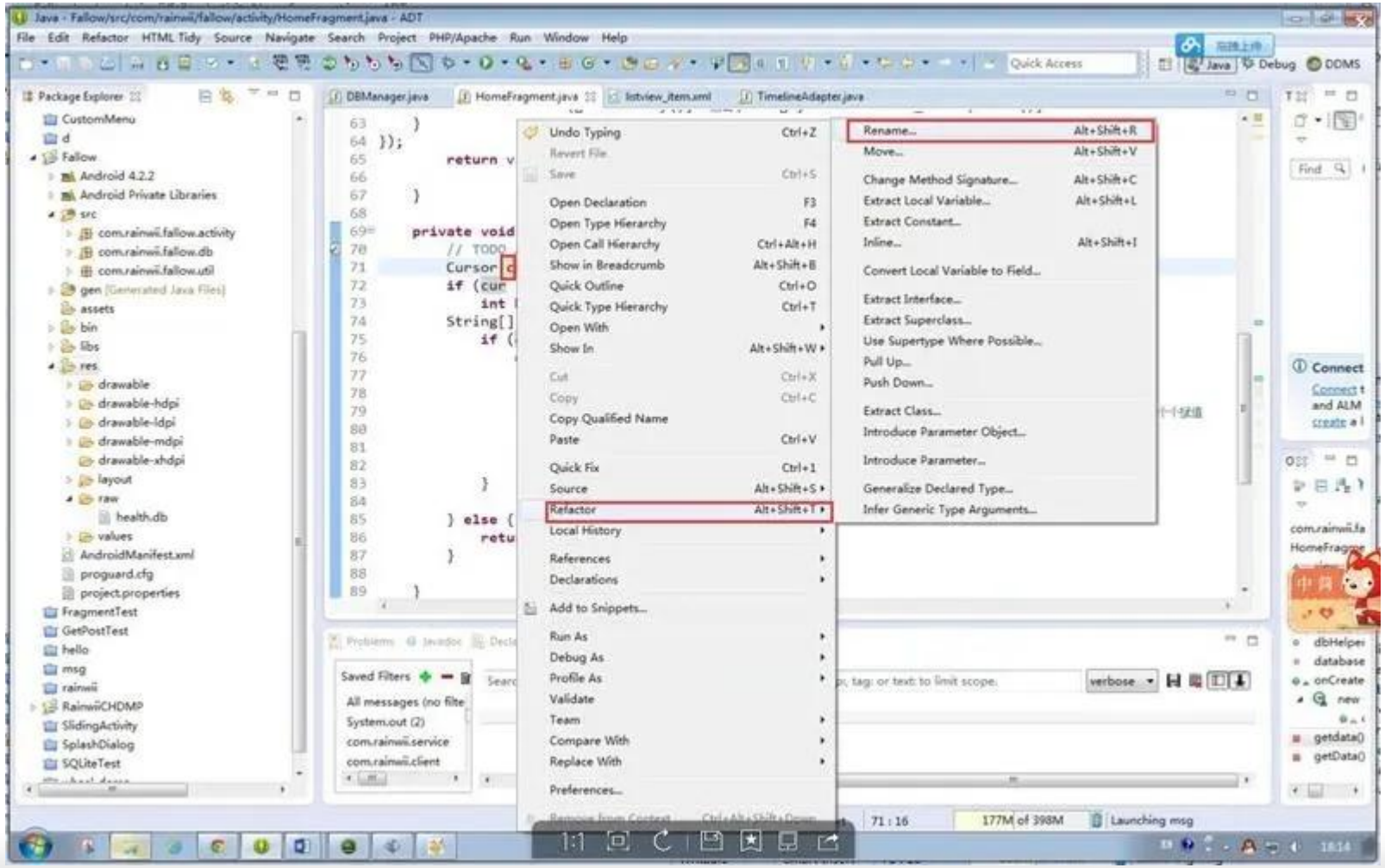
- ✓将CASE工具按统一标准和接口组装起来，使工具间、人员间、各个过程间能方便交互的集成环境
- ✓如Visual Studio将编辑、编译、调试、界面设计、安装程序生成等等集成在一起
- ✓具有集成性特点

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

计算机辅助软件工程环境

- Eclipse、Visual Studio、Copilot、Cursor等



集成编辑器、编译器、连接器、调试器、代码生成工具等一系列的工具，并且不同工具之间可以交互信息

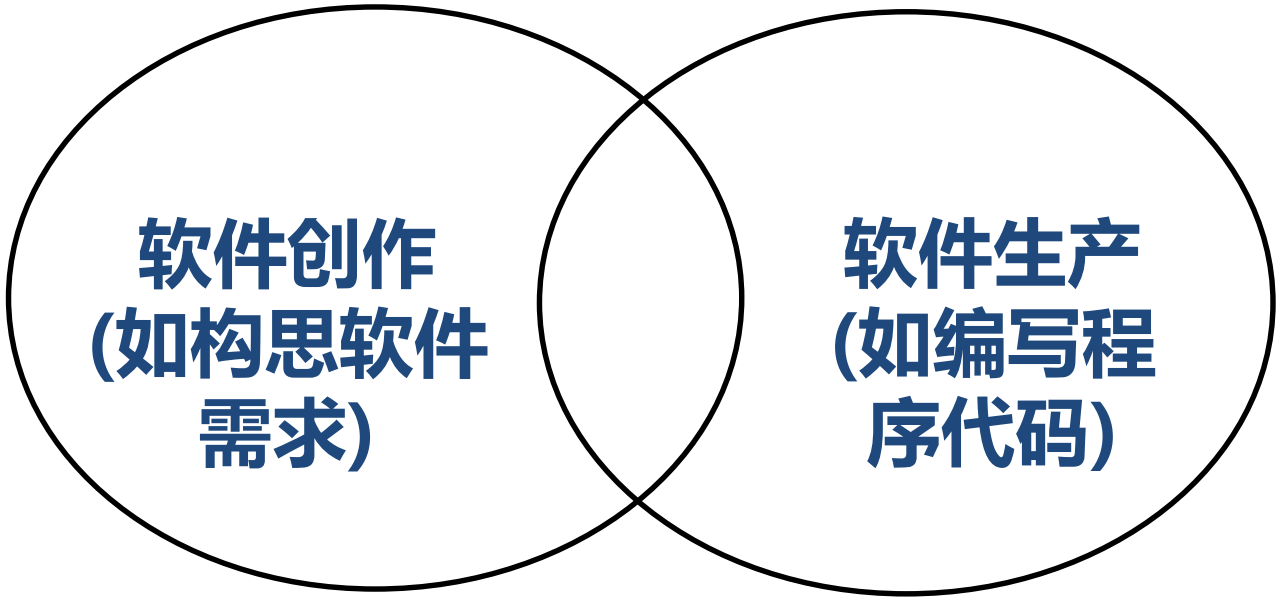
CASE工具和环境的变化

- **基于互联网的在线服务**
  - ✓部署和运行在互联网平台上，为互联网上的开发者提供软件在线的开发和运维服务，如odeArts、GitHub、SonarQube等
- **基于大数据的智能化服务**
  - ✓为软件开发工作以及软件运维提供智能化服务，如代码自动生成、代码片段推荐，典型例子如Copilot等
- **基于共享的集成化服务**
  - ✓不同CASE工具及环境间的数据共享

2.5 软件开发的本质

软件开发 = 软件创作 + 软件生产

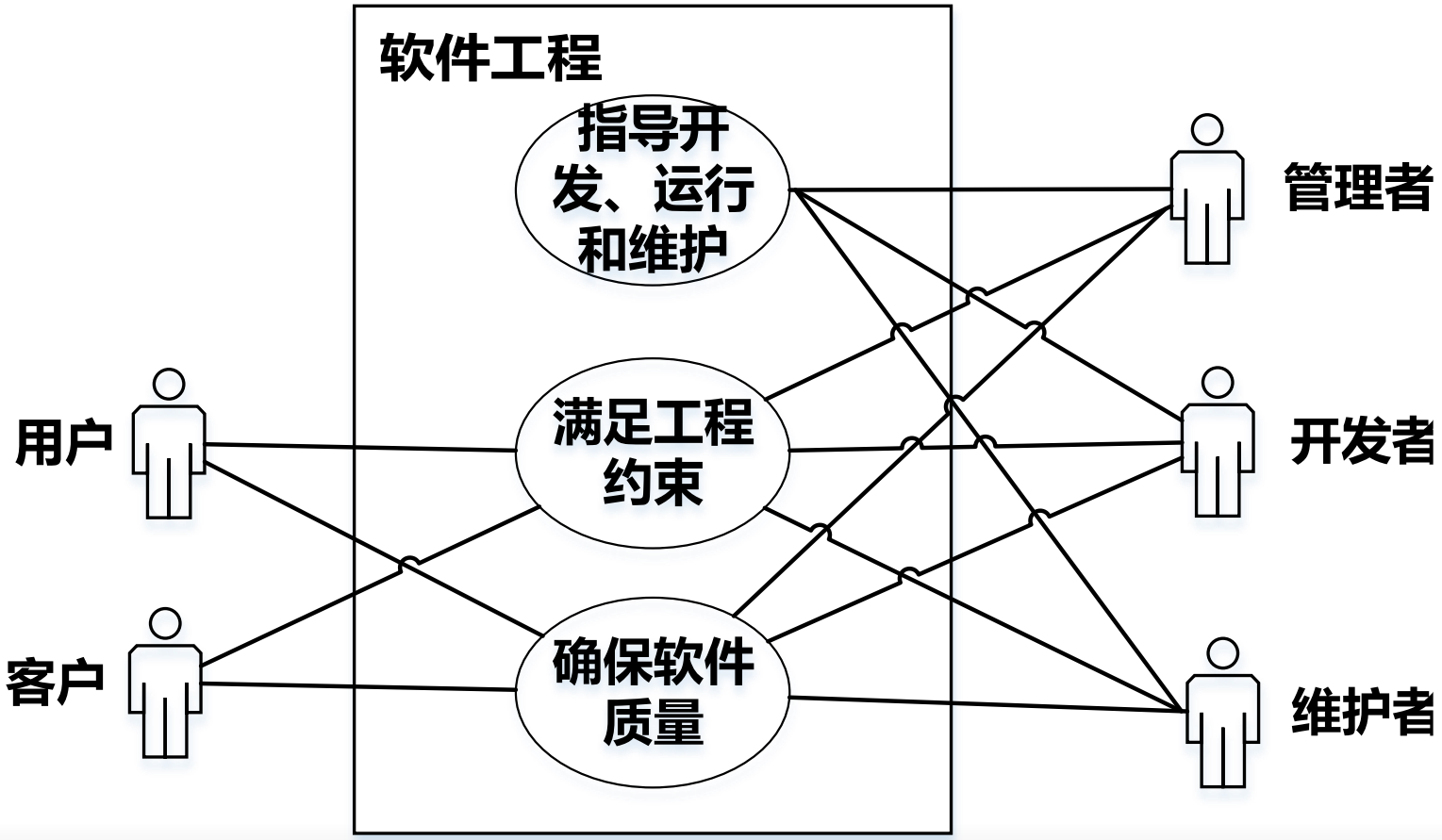
基于软件开发者的经验和技能，借助于智慧，进行自由创新，如软件设计、编码实现等



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
  - ✓满足约束和要求
  - ✓指导软件开发
  - ✓确保软件质量

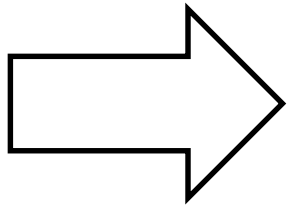


2.7 软件工程原则

软件开发要遵循软件工程原则

- 抽象与建模
- 模块化
- 软件重用
- 信息隐藏
- 关注点分离
- 分而治之
- ……

软件工程原则



软件工程目标

- 满足约束
- 满足需求
- 高质量
- 高效率
- ……

- 软件工程原则有助于促进软件工程目标的实现
- 软件工程原则是在长期实践中总结出来、行之有效

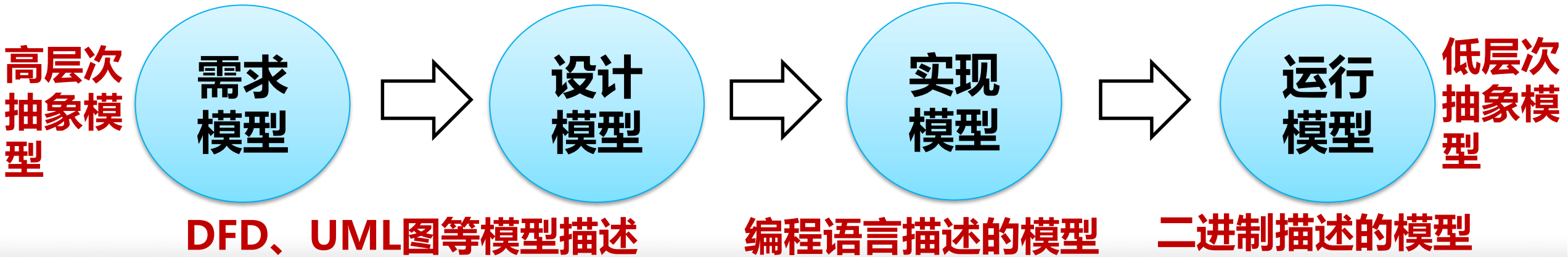
原则1：抽象和建模

❑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

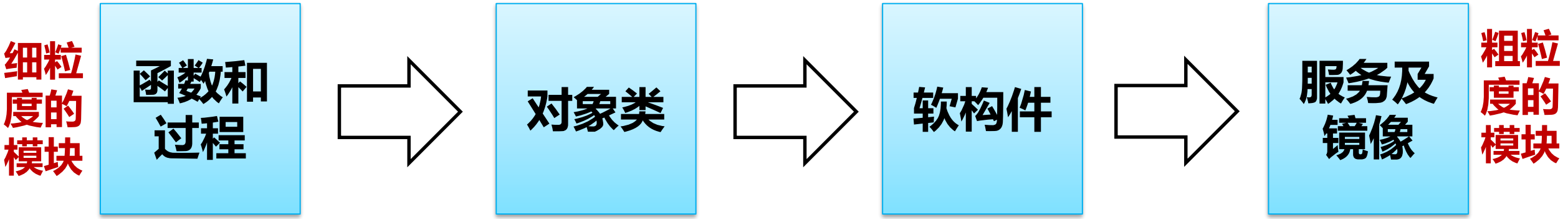
❑建模

✓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



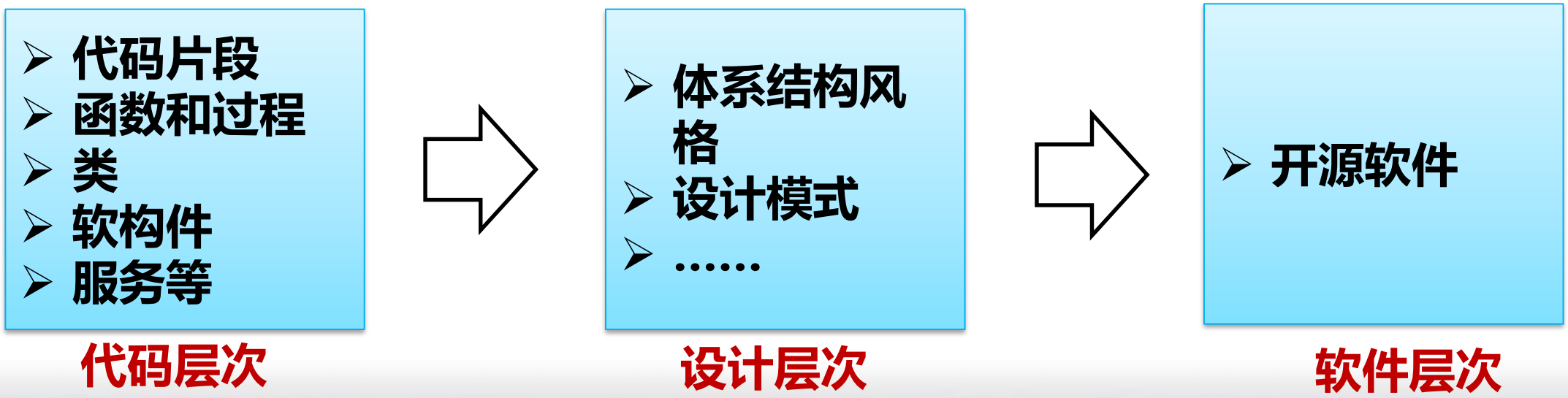
原则2：模块化

- 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**



原则3：软件重用

- 开发过程中尽可能**利用已有软件资源和资产**（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
- 开发出**可被再次重用**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**



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);
        }

```

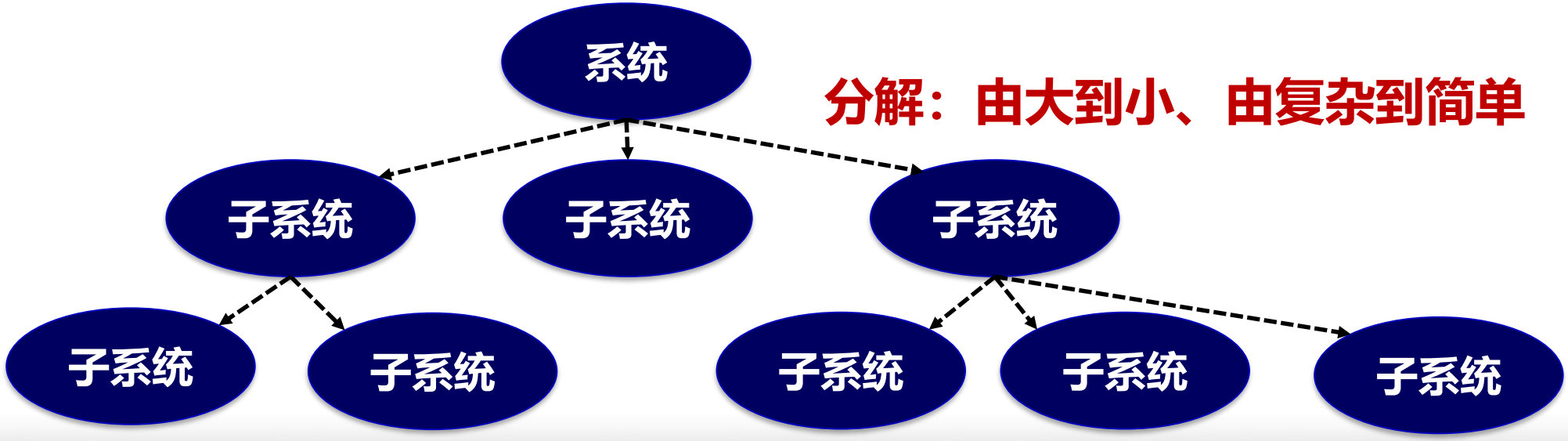
原则5：关注点分离

- 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

关注点分离的目的是要做到“做事不要分心”

原则6：分而治之

- **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，还可以继续分解，及至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**



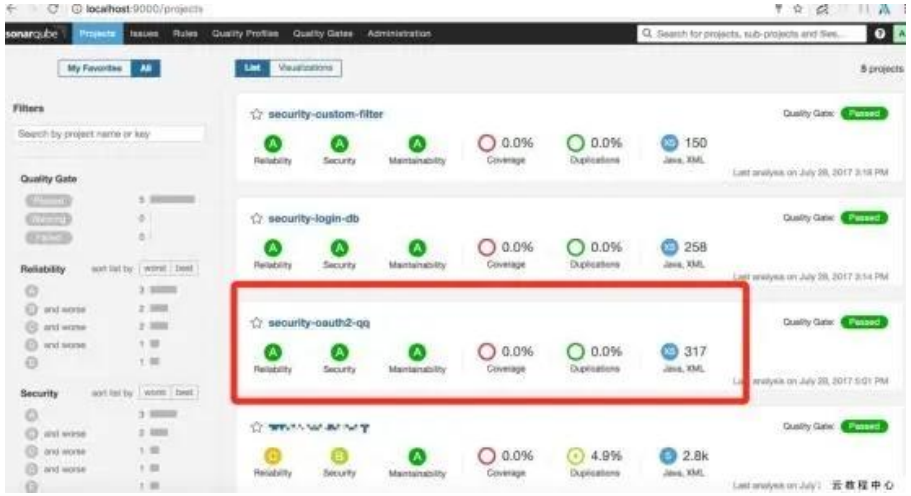
原则7：双向追踪原则

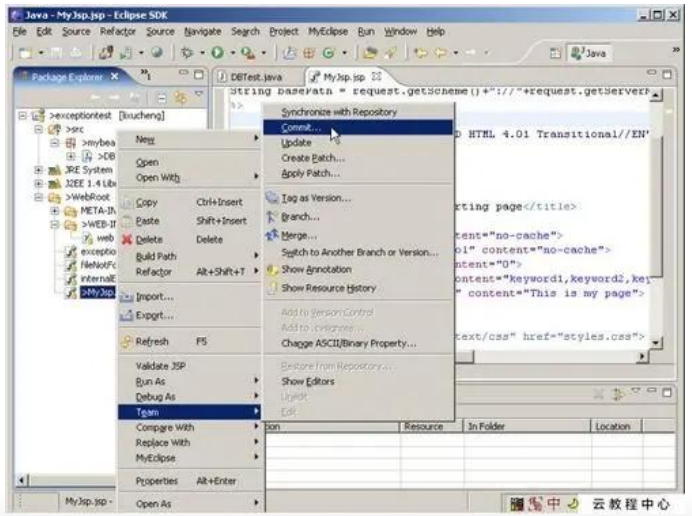
- **正向和反向追踪**
  - ✓当软件制品**发生变化**时，要追踪这种**变化会对那些软件制品产生影响**，进而指导相关的开发和维护工作，此为正向追踪
  - ✓**追踪产生这种变化的来源**，或者说是什么因素导致该软件制品的变化，明确软件制品发生变化的原因及其合理性，此为反向追踪

- **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**

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

思考和讨论：编程过程中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**？

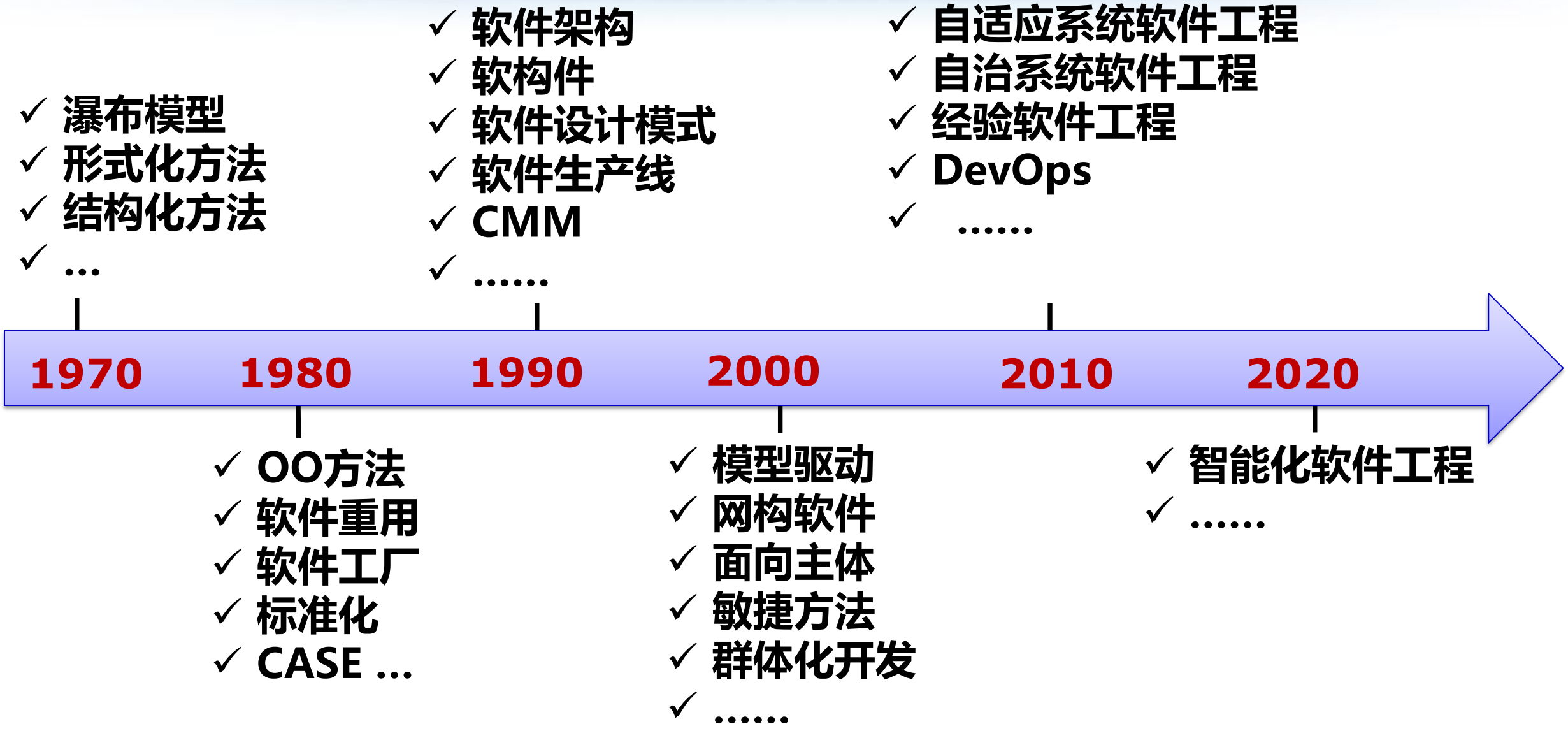
- 抽象和建模
- 模块化
- 软件重用
- 信息隐藏
- 关注点分离
- 分而治之
- 双向追踪
- 工具辅助

```
/*
 * 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);
        }

```

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

3.1 软件工程的发展历程



1950s-1960s

❑软件的特点

- ✓软件功能较为简单，计算机软件与硬件结合的非常紧密

❑软件开发的实践

- ✓ “**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使用
- ✓设计了若干**高级程序设计语言**，如Fortran、COBOL、LISP等
- ✓产生了软件工程
- ✓这一时期的特点：**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**

1970s

❑软件特点

- ✓软件部署在**主机**上，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**

❑软件工程的研究与实践

- ✓**程序设计语言和程序设计方法学**成为研究热点，出现了PASCAL、C、Prolog、ML等高级语言
- ✓产生了**软件工程新技术**，如**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）
- ✓研制了支持结构化软件开发方法学等的**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

1980s

❑软件的特点

- ✓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

❑软件工程的研究与实践

- ✓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

1990s

❑软件特点

- ✓软件部署在局域计算环境上运行，互联网应用开始出现

❑软件工程的研究与实践

- ✓**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，如UML
- ✓**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

2000-2010

❑软件特点

- ✓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**

❑软件工程的研究与实践

- ✓**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

2010以来

❑软件特点

- ✓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

❑软件工程的研究与实践

- ✓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，如大模型技术、Copilot

2022至今

❑基于大模型的智能化软件工程

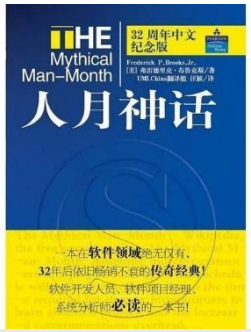
- ✓基于通用生成式大模型，辅助软件的智能化开发工作
- ✓采用**自然语言**进行交互
- ✓支持**代码生成、代码适配、测试用例生成、测试代码生成**等一系列的工作
- ✓极大提高了软件开发的效率和质量

GPT 3 (2020.05) ⇨ GPT 4 (2023.03) ⇨ GPT 4O (2024.05) ⇨ GPT O1 (2024.09)

GPT：Generative Pre-Trained Transformer

软件工程发展的特点

- 与**不同时代软件的特点**息息相关，根本问题是要推动软件系统的开发和运维
- 由**实践驱动**的学科，在实践中不断探索，在探索中提出新技术新方法，实践先行、“**摸着石头过河**”的研究策略
- 几乎**每隔十年**就有一次飞跃，软件抽象层次越来越高，软件重用粒度越来越大
- “**软件开发没有银弹**” -- 布鲁克斯（Brooks）在《人月神话》中预言，大模型和智能化开发是银弹吗？



软件开发理念的变化

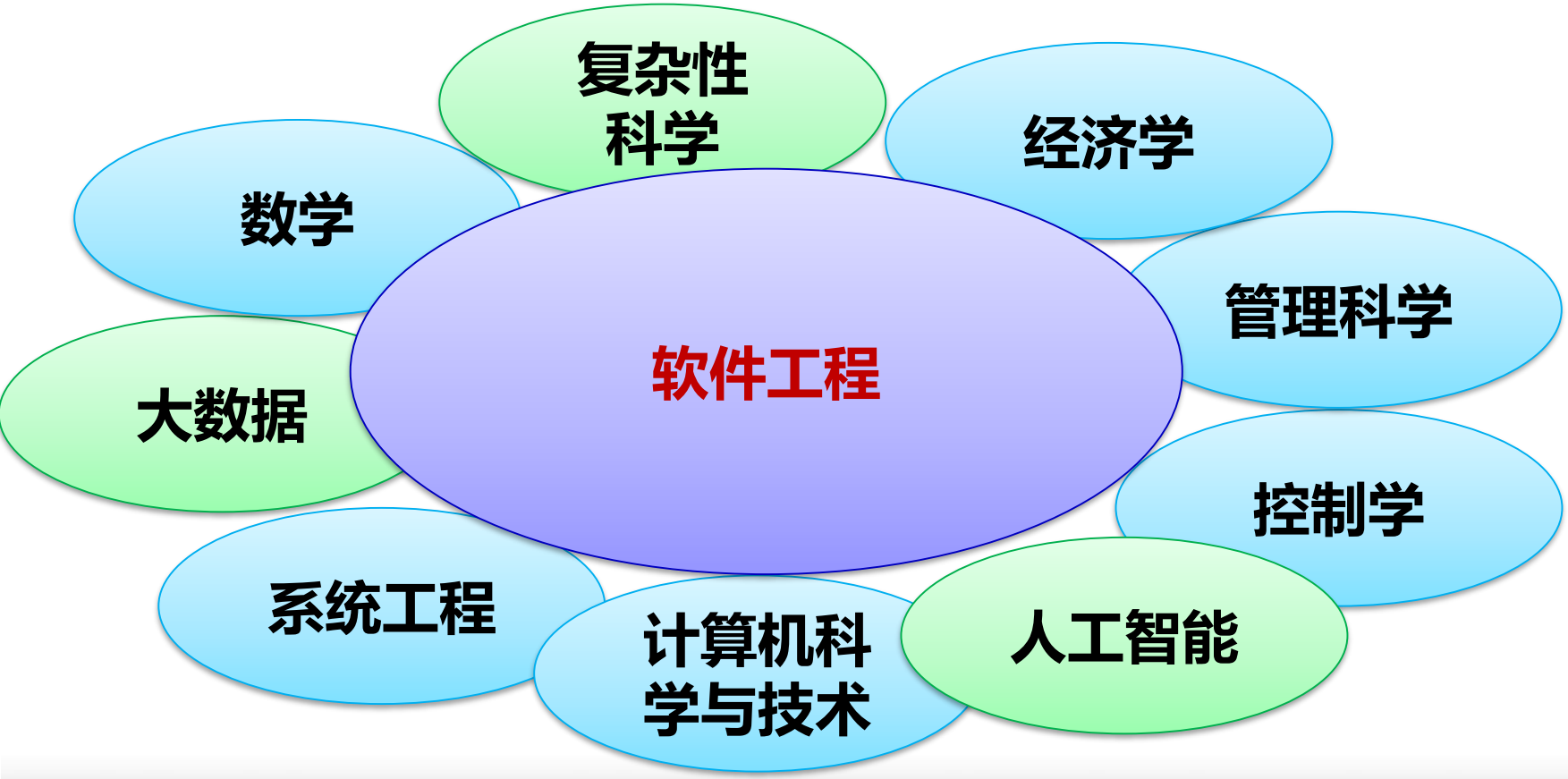
- 以**文档**为中心与以**代码**为中心
  - ✓从重型方法到敏捷方法
- 从**个体、团队到群体**的开发组织
  - ✓从团队开发到群体化开发
- 方法的**相悖性**
  - ✓如重型和敏捷等

软件工程技术进步的特点

- **软件抽象的层次越来越高**
  - ✓体现在编程语言和开发方法
- **软件重用的粒度越来越大**
  - ✓体现在可重用的软件对象
- **软件开发智能化程度越来越高**
  - ✓体现在支持代码生成、需求获取等方面

软件工程的多学科交叉

- 软件工程越来越多地与**相关学科进行交叉**，以推动**更多领域软件、更为复杂软件系统**的研究和实践



- ✓ 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

- 问题和目标
  - ✓ 软件危机
- 基本原则
  - ✓ 软件重用
  - ✓ 模块化
  - ✓ 问题分解
  - ✓ 分而治之
  - ✓ 关注点分离
  - ✓ ....

不变

- 软件复杂性不断增加
- 对软件系统理解和认识
  - ✓ 社会技术视点
- 学科交叉
  - ✓ 交叉更多的学科
- 采用的方法和手段
  - ✓ 敏捷和DevOps方法
  - ✓ 群体化开发
  - ✓ 智能化开发
  - ✓ ....

变

基于“不变”要素，寻求“变化”的方法和技术，解决规模和复杂性不断增长的软件系统

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
- 1990s，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、智能化软件开发

习题

2-7、2-8、2-9、2-10、2-18