# 软件工程

李京

2020 春季学期

# 课程基本信息（1）

- 课程名称：软件工程 Software Engineering
- 课程属性：专业基础课
- 课程对象：本科三年级
- 上课时间：1~14周，周二（3、4、5）
- 上课地点：西区教三楼3C104
- 学时：40+20
- 教师：李京　　lj@ustc.edu.cn
- 助教：孙诗伦、梁润秋、王皓辰、董建亮、邱浩宸
- 参考教材：软件工程：实践者的研究方法（原书第8版 ） [美] 罗杰 S. 普莱斯曼 等著，郑人杰，马素霞等译

# 课程基本信息（2）

- 课程目的：软件工程是一门指导软件开发与维护的综合性课程，包括软件工程的基本概念、软件过程和生命周期建模、项目计划和管理、需求工程、体系结构和模块设计、编码与测试、软件维护、评估与改进等。通过本课程的学习，使学生掌握系统的软件开发理论、技术、方法和工具，基本具备应用恰当软件工程方法开发低成本、高可靠性和高效软件系统的能力。

# 课程基本信息（3）

- 课程主要内容：
  - 学习和研究的对象：软件和软件工程的基本概念
  - 软件开发过程
  - 软件的建模
  - 软件的质量管理
  - 软件项目的管理

# 授课方式

- 按照教材的章节授课，适当扩充裁剪内容。
- 实践课部分，本次课程特地安排了“计蒜客”来负责此部分内容。

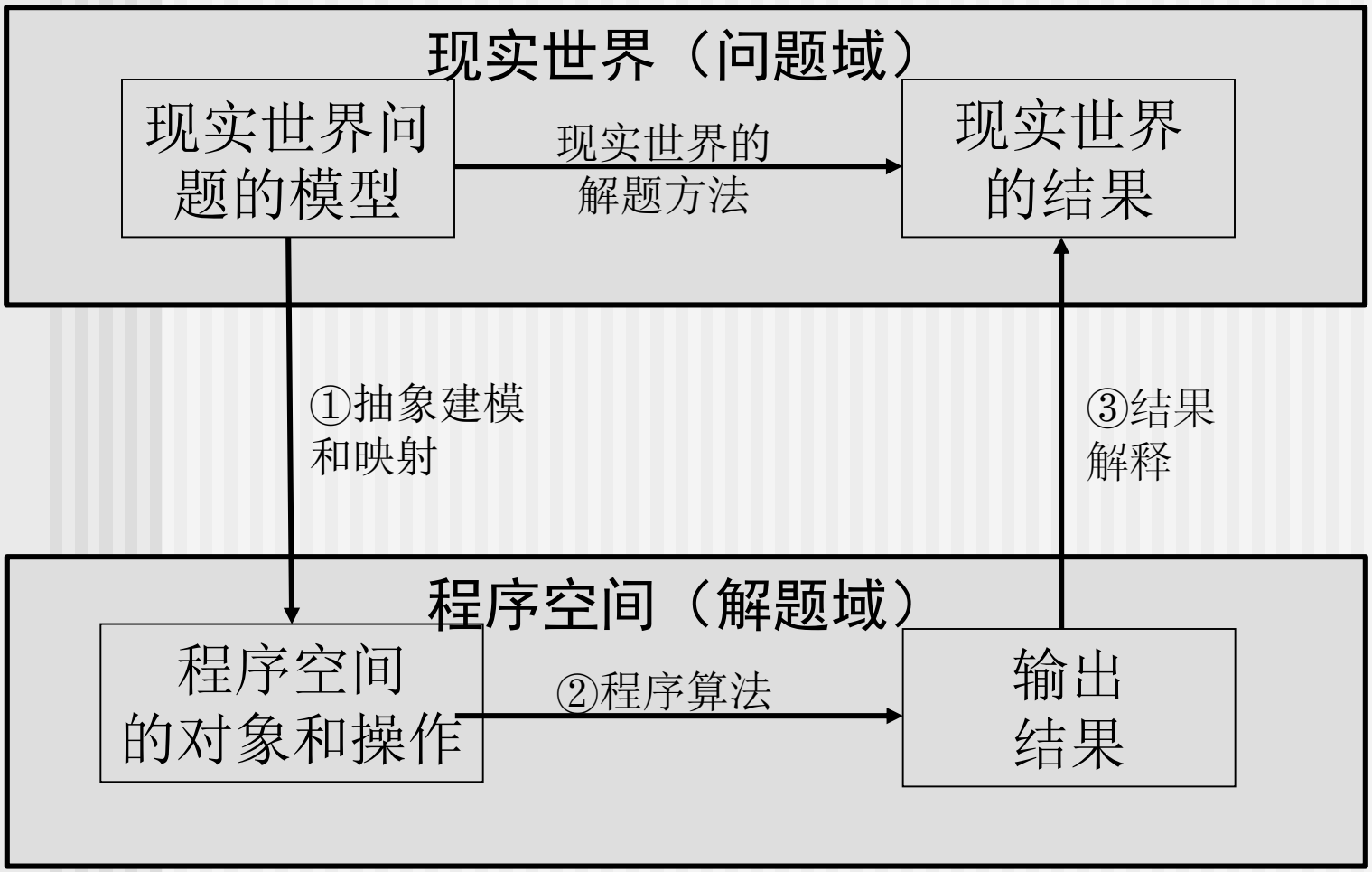
# 考核方式

- 课程考核方式：
  - 实践课成绩占70分
  - 平时成绩占30分，包括作业和随堂测验

# 软件工程简述

# 软件开发的基本目标

人们试图用抽象的方法将现实世界的问题转化成程序空间的解题程序，通过程序的执行，控制计算机获得问题的结果



# 软件发展的历程

**早期**

- 面向批处理
- 有限的分布
- 自定义软件

**第二阶段**

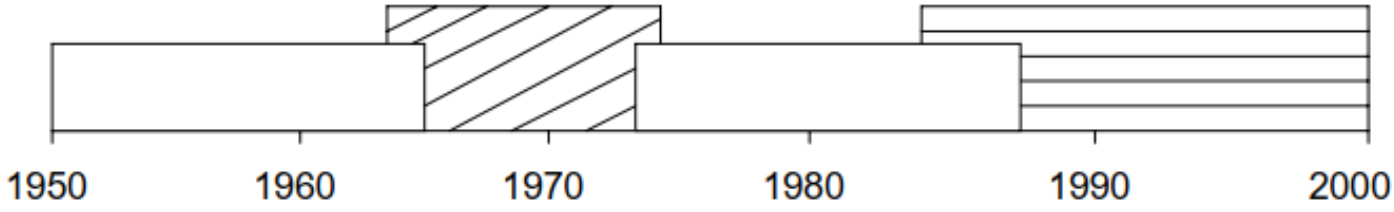
- 多用户
- 实时
- 数据库
- 软件产品

**第三阶段**

- 分布式系统
- 嵌入“智能”
- 低成本硬件
- 消费者的影响

**第四阶段**

- 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



# 软件危机

- 软件项目的规模越来越大，例如windows2000团队包含了几千人，一个航天软件的源代码可以带到几千万行。
- 软件开发成本和进度的估计不准，常常失控
- 软件质量不能让客户满意，BUG和补丁太多
- 软件的可维护度太低
- 软件通常缺乏文档

总之，软件开发的生产率的提高远远落后于硬件的发展和人们需求的增长。

# 软件工程

- 1968年秋季，NATO（北约）的科技委员会召集了近50名一流的编程人员、计算机科学家和工业界巨头，讨论和制定摆脱“软件危机”的对策。在那次会议上第一次提出了软件工程（software engineering）这个概念。
- 软件工程是研究和应用如何以系统性的、规范化的、可定量的过程化方法去开发和维护软件，以及如何把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来的学科。

# 软件工程要素

- 方法 methodology
  - 为软件开发提供“如何开发”的原理和技术，不同的方法会导致不同的软件过程
- 工具 tools
  - 为软件工程方法提供自动或半自动的支撑，是方法的实在体现，各种软件工具和软件运行基础设施构成了软件工程环境
- 过程 process
  - 软件工程方法和工具的综合运用以科学地进行软件的开发和实施

# 软件工程的目标

- 研究科学的软件工程原理和方法，并开发与之相适应的软件工具，从技术和管理上保证软件工程项目实施的成功，用有限的投资在规定的工程期限内完成高质量的软件

# 第0部分 基本概念

- 软件的本质
- 软件工程

# 第1章　软件的本质

# 什么是软件?

软件具有产品和产品交付载体的双重作用。

软件是:

(1)指令的集合(计算机程序),通过执行这些指令来满足预期的特征、功能和性能需求;

(2)数据结构,使得程序可以合理的利用信息;

(3)软件描述信息,以硬拷贝和虚拟形式存在,用来描述程序操作和使用。

**可以认为:软件=程序+数据+文档**

# 什么是软件？

软件在我们生活中无所不在，并且日益深入到商业、文化和日常生活的各个方面。

- 软件是设计开发的，而不是传统意义上的生产制造的。
- 软件不会“磨损”。
- 虽然整个工业向着基于构件的构造模式发展，然而大多数软件仍是根据实际的顾客需求定制的。

# 磨损vs.退化

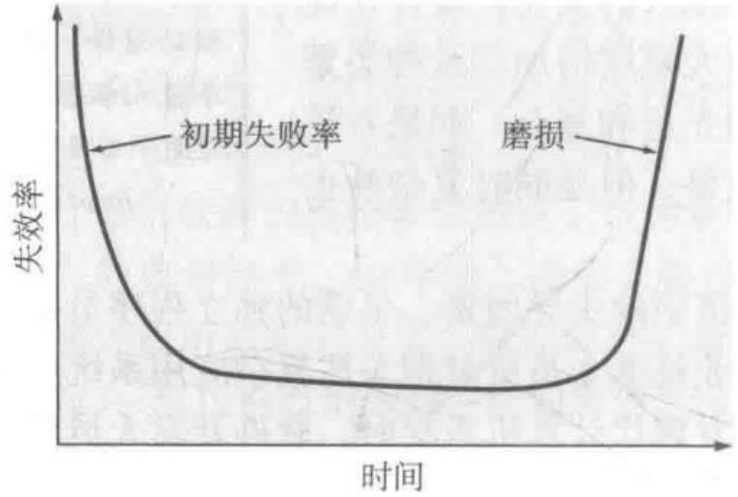



图 1-1　硬件失效曲线图

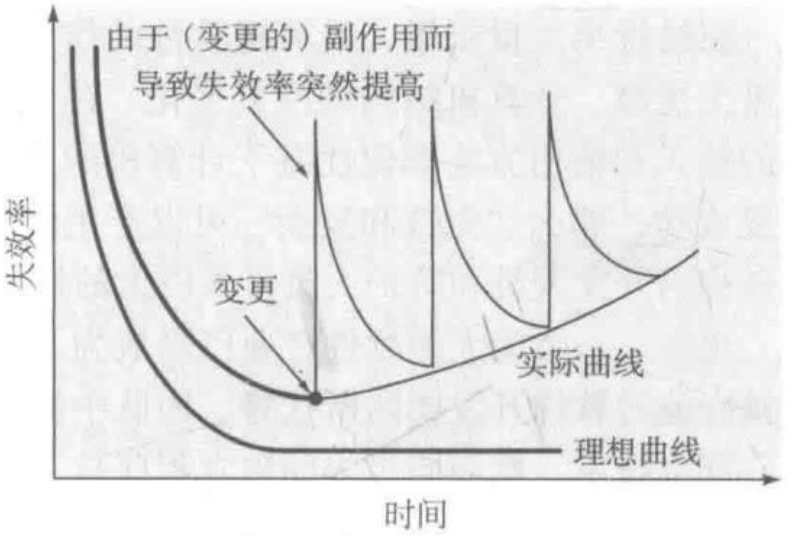



图 1-2　软件失效曲线图

# 软件应用领域

- 系统软件：服务于其他程序的程序，如操作系统
- 应用软件：解决特定业务需要的独立应用程序
- 工程/科学软件：数值计算类程序
- 嵌入式软件：存在于某个产品或系统内
- 产品线软件：为多个不同用户的使用提供特定功能
- Web应用软件：以互联网为中心，基于浏览器
- 移动应用软件：安装在移动设备上的软件
- 人工智能软件：具有推理和学习能力的程序，如专家系统、机器人、定理证明、深度学习、人工神经网络等。

# 遗留软件

遗留软件(legacy software)是从现有的技术之前的语言平台和技术中继承下来的那些旧的系统。遗留软件通常支持核心的商业功能，是业务必不可少的支撑。

**遗留软件的特性：**

- 生命周期长
- 业务关键性
- 质量差

# 遗留软件

**遗留软件最好不修改，但是仍经常需要演化，导致演化的变更为什么往往是必须的呢？**

- 软件必须进行适应性调整，以满足新的计算环境和技术的需求。
- 软件必须升级以实现新的商业需求。
- 软件必须扩展使之具有与更多现代系统和数据库的互操作能力。
- 软件必须进行改建使之能适应多样化的网络环境。

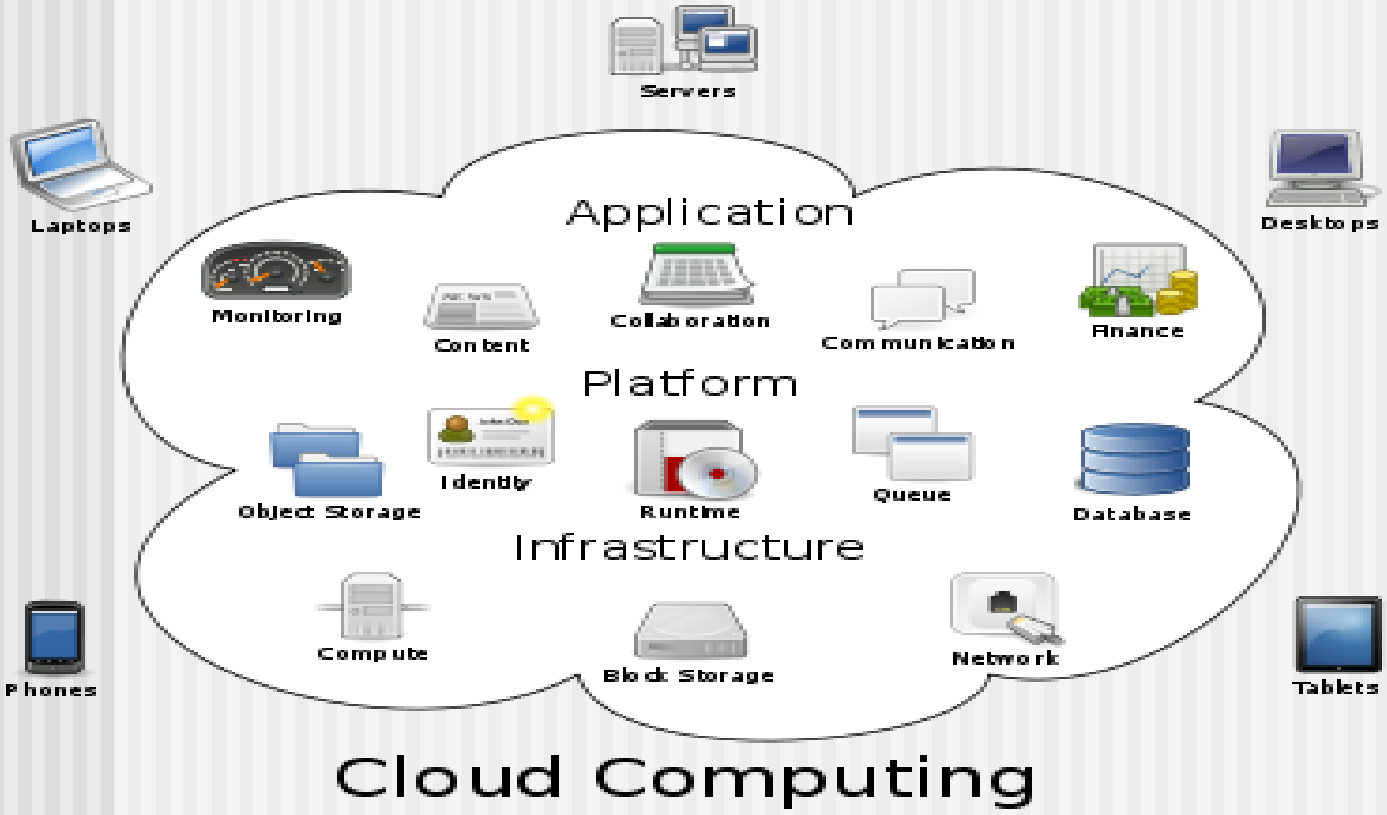
# WebApp

- 最初的Web站点仅仅是由链接在一起的超文本文件，是静态的。现代WebApp远远不止于少量图片的超文本文件。
- 一些开发工具（例如，XML、Java）扩展了WebAPP的能力，使得Web工程师在向客户提供信息的同时也能提供计算能力。
- WebApps不仅可以为最终用户提供独立的功能，而且已经同公司数据库和业务应用系统集成在一起了。
- 语义Web技术（通常指Web 3.0）已经演化为成熟的企业和消费者应用系统，包括提供新功能的语义数据库，这些新功能需要Web链接、灵活的数据表示以及外部访问APIs。
- 应用内容的精美程度仍是决定WebAPP质量的重要因素。

# 移动App

- 术语app已经演化为在移动平台（例如，iOS、Android或Windows Mobile）上专门设计的软件。
- 移动应用系统包括用户接口，用户接口利用移动平台所提供的独特的交互机制。
- 基于Web资源的互操作性提供与app相关的大量信息的访问，并具有本地处理能力。
- 提供了在平台中的持久存储能力。
- 移动Web应用系统允许移动设备通过针对移动平台的优点和弱点专门设计的浏览器获取基于Web内容的访问。
- 移动app可以直接访问设备的硬件特性（例如，加速器或者GPS的位置），然后提供前面所述的本地处理和存储能力。
- 随着时间的推移，移动Web应用系统与移动apps之间的区别已变得模糊。

# 云计算


Servers
Laptops
Desktops
Application
Monitoring
Content
Collaboration
Communication
Finance
Platform
Identity
Object Storage
Runtime
Queue
Database
Infrastructure
Compute
Block Storage
Network
Phones
Tablets
Cloud Computing

# 云计算

- 云计算提供分布式数据存储和处理功能，它能使得任何用户无论在任何地点都可以使用计算设备来共享广泛的计算资源。
- 计算设备位于云的外部，可以访问云内的各种资源。
- 云计算的实现需要开发包含前端和后端服务的体系结构。
- 前端包括客户（用户）设备和应用软件（如浏览器）用于访问后端。
- 后端包括服务器和相关的计算资源、数据存储系统（如数据库）、服务器驻留应用程序和管理服务器。
- 可以对云体系结构进行分段，提供不同级别的访问。

# 云的分类

| 云计算类型 | IT云计算 | 互联网云计算 |
| --- | --- | --- |
| 资源整合方式 | 分裂资源：将计算和存储能力较强的资源“分裂”为更小的“计算和存储”单元，实现对更小粒度资源的充分利用 | 聚合资源：通过将普通的资源能力“聚合”起来，提供强大的存储和计算能力 |
| 技术 | 以虚拟化技术的应用为主 | 以分布式技术的应用为主，采用分布式架构 |
| 服务器类型 | 主要支持x86硬件架构，但每个节点的处理能力、软件系统都可以不同；同时也适用于整合不同的硬件架构 | 同构的通用服务器（包括硬件和软件的配置） |
| 存储架构 | 一般利用共享的高性能存储（例如SAN） | 利用每个节点的存储空间构成分布式存储（例如GFS、HDFS） |
| 计算模型 | 几乎不改变传统的计算架构和模型，遗留应用和系统几乎无需改动或少量改动即可运行在虚拟化架构上，并且由于引入虚拟化而带来更快的应用部署速度和更高的应用可靠性 | 新型的计算架构和模型，需要基于新的编程模式来开发新的应用（如MapReduce） |
| 提供商 | 以IT企业为主，例如IBM，HP，Oracle，微软等 | 以互联网企业为主，例如Google、Yahoo、Facebook、Salesforce |
| 主要应用场景 | 用于对IT资源整合，提高资源利用率 | 大数据的存储和处理 |

# 软件产品线

- 软件产品线是一系列软件密集型系统，可以共享一组公共的可管理的特性，这些特性可以满足特定市场或任务的特定需求。
- 软件产品线都使用相同的底层应用软件和数据体系结构来开发，并使用可在整个产品线中进行复用的一组软件构件来实现。
- 软件产品线共享一组资源，包括需求、体系结构、设计模式、可重用构件、测试用例及其他软件工程工作产品。
- 软件产品线在对这些产品进行工程设计时，利用了产品线中所有产品的公共性。

# WebApp的特性

- **网络密集型：**WebApp 驻留在网络上，服务于不同客户群体的需求。
- **并发性：**大量用户可能同时访问WebApp 。
- **无法预知的负载量：**WebApp的用户数量每天都可能有数量级的变化。
- **性能：** 如果一位WebApp用户必须等待很长时间（访问、服务器端处理、客户端格式化显示），该用户就可能转向其他地方。
- **可用性：**尽管期望百分百的可用性是不切实际的，但是对于热门的WebApp，用户通常要求能够24/7/365（全天候）访问。

# WebApp的特性

- **数据驱动：**许多WebApp的主要功能是使用超媒体向最终用户提供文本、图片、音频及视频内容。
- **内容敏感性：**内容的质量和艺术性仍然很大程度上决定了WebApp的质量。
- **持续演化：**传统的应用软件是随一系列规划好的时间间隔发布而演化的，而Web应用则持续地演化。
- **即时性：**尽管即时性—也就是将软件尽快推向市场的迫切需求—是很多应用领域的特点，然而将WebApp投入市场可能只是几天或几周的事。
- **安全性：**由于WebApp是通过网络访问来使用的，因此要限制访问的最终用户，即使可能也非常困难。
- **美观性：**不可否认，WebApp的用户界面和外观很有吸引力。

# 第2章　软件工程

# 软件工程

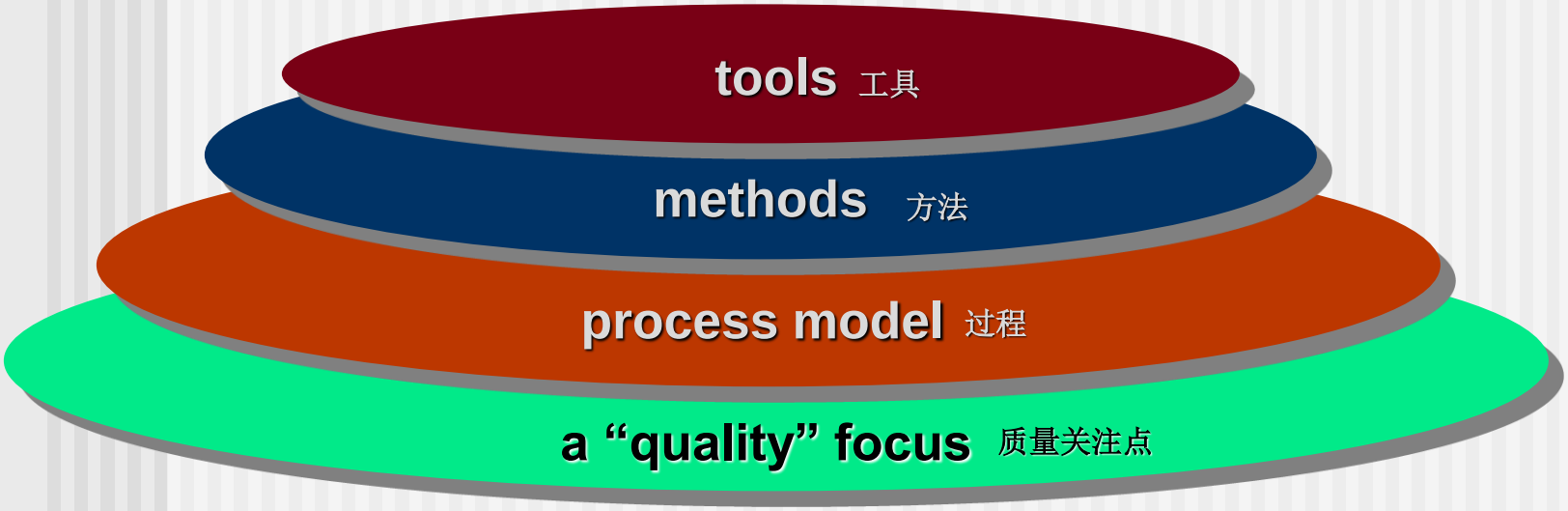
- 要构建能够适应挑战的软件产品，要认识到以下几个事实:
  - 在制定软件解决方案之前，必须共同努力来理解问题
  - 设计已成为关键活动
  - 软件必须保证高质量
  - 软件需要具备可维护性
- 结论：工程化
  - 建立和使用一套合理的工程原则，以便经济地获得可靠的、可以在实际机器上高效运行的软件。

# 软件工程

- IEEE 定义：

  软件工程是：（1）将系统化的、规范的、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(1)中所述方法的研究。

# 层次化技术




***Software Engineering***
软件工程

支持软件工程的根基在于质量关注点

# 层次化技术

- 支持软件工程的根基在于质量关注点
- 基础是过程（process），其将各个技术层次结合在一起，构成了软件项目管理控制的基础，提供了应用各种方法、技术和工具的工作环境
- 方法（methodology）为构建软件提供技术上的解决方法（如何做），如结构化方法、面向对象方法等
- 工具（tool）为过程和方法提供自动化或半自动化的支持，如软件开发环境（有时也称计算机辅助软件工程CASE），自动测试工具等

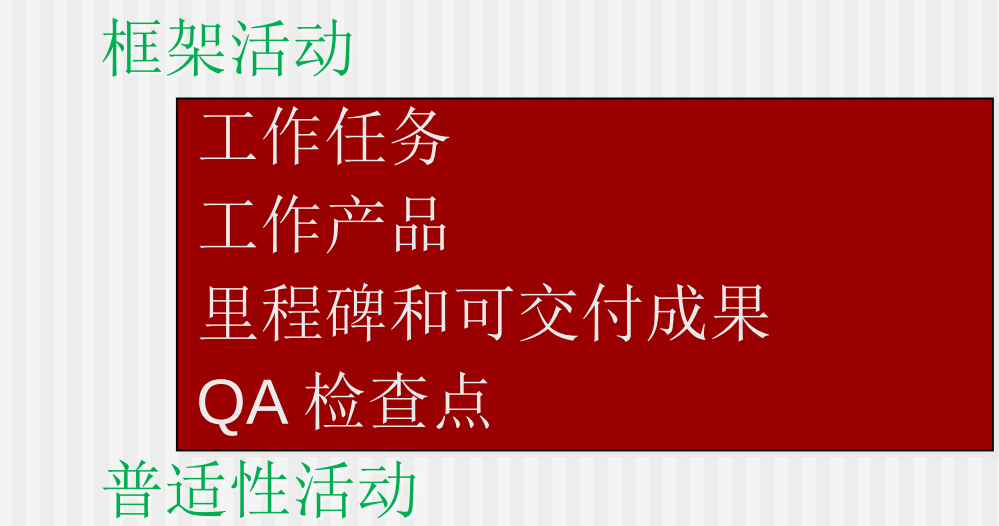
# 软件过程

- 软件过程是构建软件产品时所执行的一系列活动、动作和任务的集合。
  - 活动（activity）主要实现宽泛的目标，如需求分析
  - 动作（action）包含了主要工作产品生产过程中的一系列任务，如动作体系结构设计要执行体系结构设计模型所决定的构建软件体系结构需要的一系列任务。
  - 任务（task）关注小而明确的目标，能够产生实际产品，如构建一个单元测试

过程通常具有弹性的

# 过程框架

过程框架(process framework)定义了若干框架活动(framework activity)，为实现完整的软件工程过程建立基础，还包括一些全局影响的普适性活动，如：




过程框架可以理解为过程模型的一个可操作实现

# 框架活动

- 沟通（需求了解）
- 策划（软件项目计划）
- 建模
  - 需求分析
  - 设计
- 构建
  - 代码生成
  - 测试
- 部署

# 普适性活动

- 软件项目跟踪和控制
- 风险管理
- 软件质量保证
- 技术评审
- 测量
- 软件配置管理
- 可复用管理
- 工作产品的准备和生产

# 过程的弹性（适应性调整）

- 活动、动作和任务的总体流程，以及它们之间相互依赖关系
- 在每一个框架活动中，动作和任务细化的程度
- 工作产品的定义和要求的程度
- 质量保证活动应用的方式
- 项目跟踪和控制活动应用的方式
- 过程描述的详细程度和严谨程度
- 客户和利益相关者对项目参与的程度
- 软件团队所赋予的自主权
- 队伍组织和角色明确程度

可以将软件工程师使用的过程框架类比为厨师的菜谱

# 软件工程的实践

**软件工程的实践该如何融入软件过程框架呢？**

- 实践的精髓，Polya 给出了解决问题的精髓：

1.理解问题（沟通分析）。
2.策划解决方案（建模和软件设计）。
3.实施计划 （代码生成）。
4.检查结果的正确性（测试和质量保证）。

# 理解问题

- 谁将从问题的解决中获益？也就是说，谁是利益相关者？
- 什么是未知的？哪些数据、功能、特征和行为是解决问题必需的？
- 问题可以划分吗？是否可以描述为更小、更容易理解的问题？
- 问题可以图形化描述吗？可以建立分析模型吗？

# 策划解决方案

- 以前曾经见过类似问题吗？ 在潜在的解决方案中，是否可以识别一些模式？是否已经存在有软件实现了所需要数据、功能、特征和行为？
- 类似问题是否解决过？如果是，解决方案所包含元素是否可以复用？
- 可以定义子问题吗?如果可以，子问题是否已有解决方案？
- 能用一种可以很快实现的方式来表述解决方案吗？能够建出设计模型吗？

# 实施计划

- 解决方案和计划一致吗？源码是否可追溯到设计模型？
- 解决方案的每个组成部分是否可以证明正确？设计和代码是否经过评审？或者更好的算法是否经过正确性证明？

# 检查结果

- 能否测试解决方案的每个部分？是否实现了合理的测试策略？
- 解决方案是否产生了与所要求的数据、功能、特征和行为一直的结果？是否按照项目共同利益者的需求进行了确认？

# Hooker的一般原则

Hooker原则是关注软件工程整体实践的原则

- 1: 存在价值（为用户带来价值，从用户角度出发）
- 2: 保持简洁（简洁非简化。可理解易维护）
- 3: 保持愿景（清晰的愿景，保持逻辑性）
- 4:关注使用者（从使用者角度出发，所有的产品）
- 5: 面向未来（可扩展性，适应变更的能力）
- 6: 计划复用（构件式软件开发，组合软件开发）
- 7: 认真思考（深思熟虑，多动脑子）

# 软件开发神话

- 影响管理者，客户(和其他非技术性的利益相关者)和从业人员
- 被认为是可信的, 因为它们有时包含真实的部分

例如开发宝典、追加工程师、外包等

但是 …

- 不可避免的导致错误的决策

因此 …

- 按照正确理解软件工程的方式从实际出发解决问题

# 软件项目是如何开始的

- 每个软件项目都来自业务需求
  - 对现有应用程序的纠错；
  - 改变遗留系统以适应变化的业务环境；
  - 扩展现有应用程序功能和特性；
  - 开发一种新的产品、服务或系统。

SafeHome项目是教材贯穿始终的一个案例