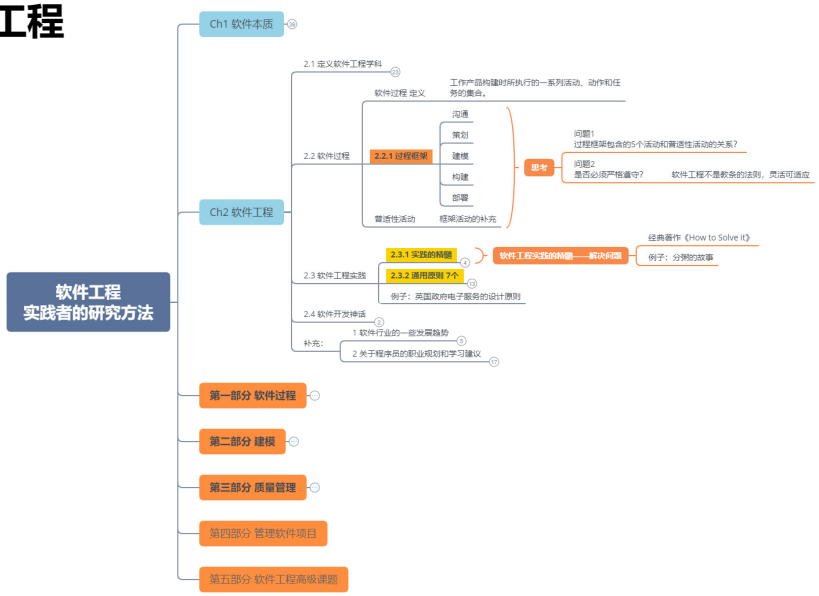
梦想，是用来实现的。

第02章 软件工程



College of Computer Science, Chongqing University

Software Engineering

A Practitioner's Approach Seventh Edition

2 软件工程

zmqmail@cqu.edu.cn 13708390417

重庆大学 计算机学院 COLLEGE OF COMPUTER SCIENCE

Think 您是否愿意与伙伴一起开发软件?

请说明原因?

Think 软件开发过程?

一盘宫保鸡丁?

Think **您愿意扮演哪个角色?**

家家有本难念的经!

吃饭 **不要/要** **茄子** **时间**

原始需求 需求变更 奇葩 成本

本质：达成共识!

如何协调?

软件工程是个面包机

https://mp.weixin.qq.com/s/cBjORghO2rXtH8i6NC-XBA

英国有个叫 Thomas Thwaites 的艺术家在几年前，花 9 个月时间做了一个面包机。

面包机	明确要实现的产品
把面包机拆成 400 个零件	分析已有产品，了解详尽的细节/架构
裁减功能到 5 种材料	制定自己的开发计划，并按照发布要求删减目标功能
垃圾桶炼钢炉	着手开发，发现预先设计的方案无法实施
油井不让拿走一桶石油	在开发过程中难免遇到无法逾越的困难，临时切换方案
用微波炉融化铁矿石	在寻找解决方案时，突然找到好用的工具
没有开关	几经周折，开发出一个 1.0 版本的面包机，丑且功能不完善
不绝缘	几乎无法作为可靠产品来使用， 但总归是一个能工作的产品
5 秒融化	第一次使用就挂了

Think 软件开发工作挺不靠谱是不是?

但实际上，大多数时候就是这样：

90% 的研发项目都在走这个流程

90% 的项目以失败告终

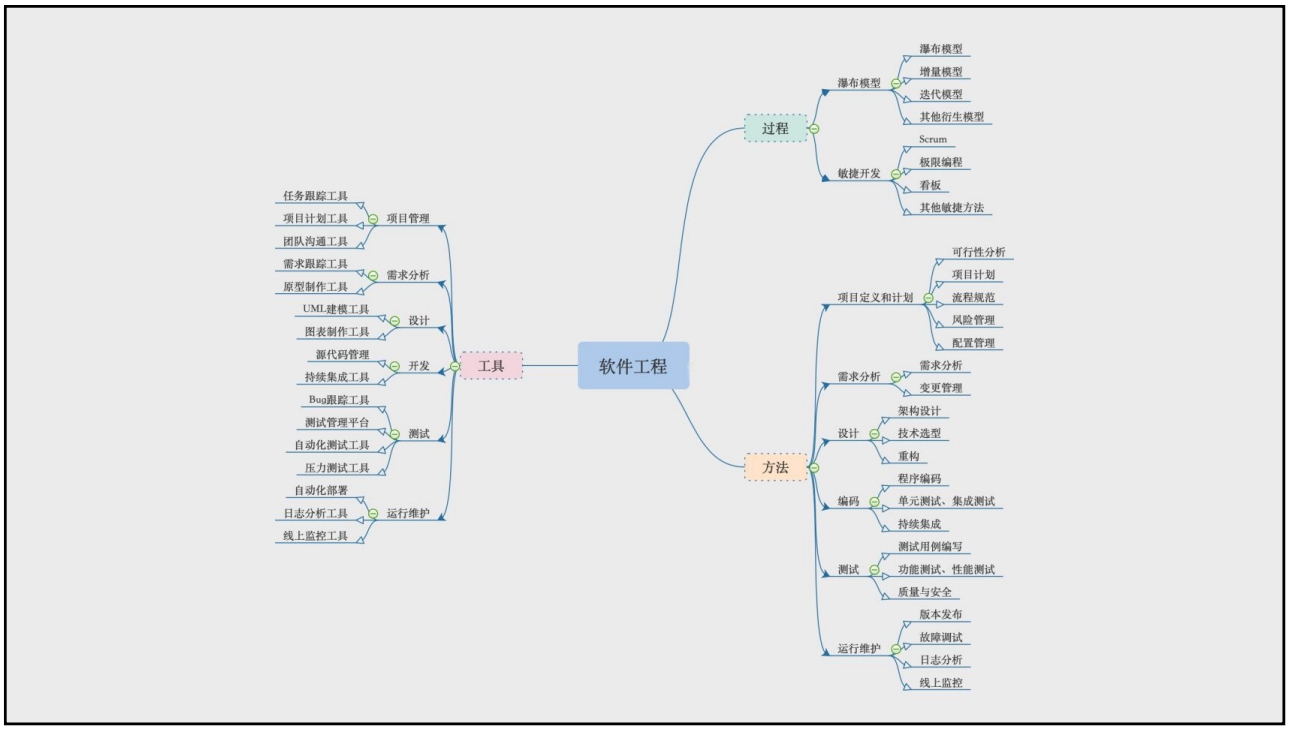
90% 的代码生存时间不超过 3 年

总结与思考

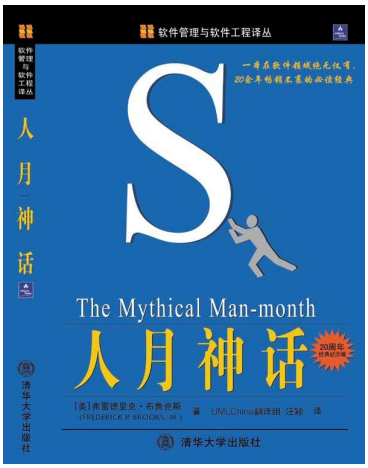
- **问题并不在于执行这个工作的人。**
  - 从 Thomas 的实施过程可以看出，他是个聪明的小伙子，翻阅资料、想各种办法，但仍然耗费了长达 9 个月时间，还只是做出了这么一个几乎是废品的产品。
- **为什么一个非常出色的工程师有时也无法做出好产品?**
- **这是上述故事里一个苛刻前提决定的——“从零开始”。**

一个完善的工具体系

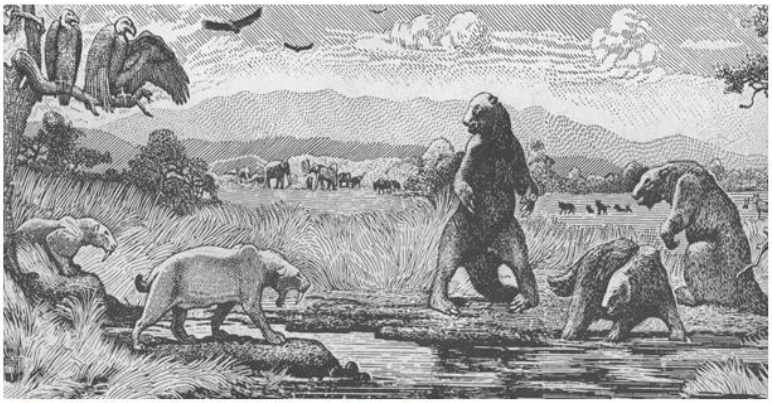
聪明的工程师无法做出高质量产品，原因是他必须依赖一个完善的工具体系。



《人月神话》



- 《人月神话》
  - 内容源于作者Brooks在IBM公司任System计算机系列以及其庞大的软件系统OS项目经理时的实践经验。
  - 作者:布鲁克斯(FrederickP.Brooks.Jr.)
  - **图灵奖得主**软件项目实战得失
  - 软件工程领域35年畅销不衰的经典
  - 软件从业人员不可不读的宝书



C. R. 奈特的《拉布雷阿的焦油坑壁画》(*Mural of La Brea Tar Pits*)

注：拉布雷阿焦油坑是著名的旅游景点，位于美国洛杉矶化石博物馆。

资料来源：The George C. Page Museum of La Brea Discoveries, The Natural History Museum of Los Angeles County

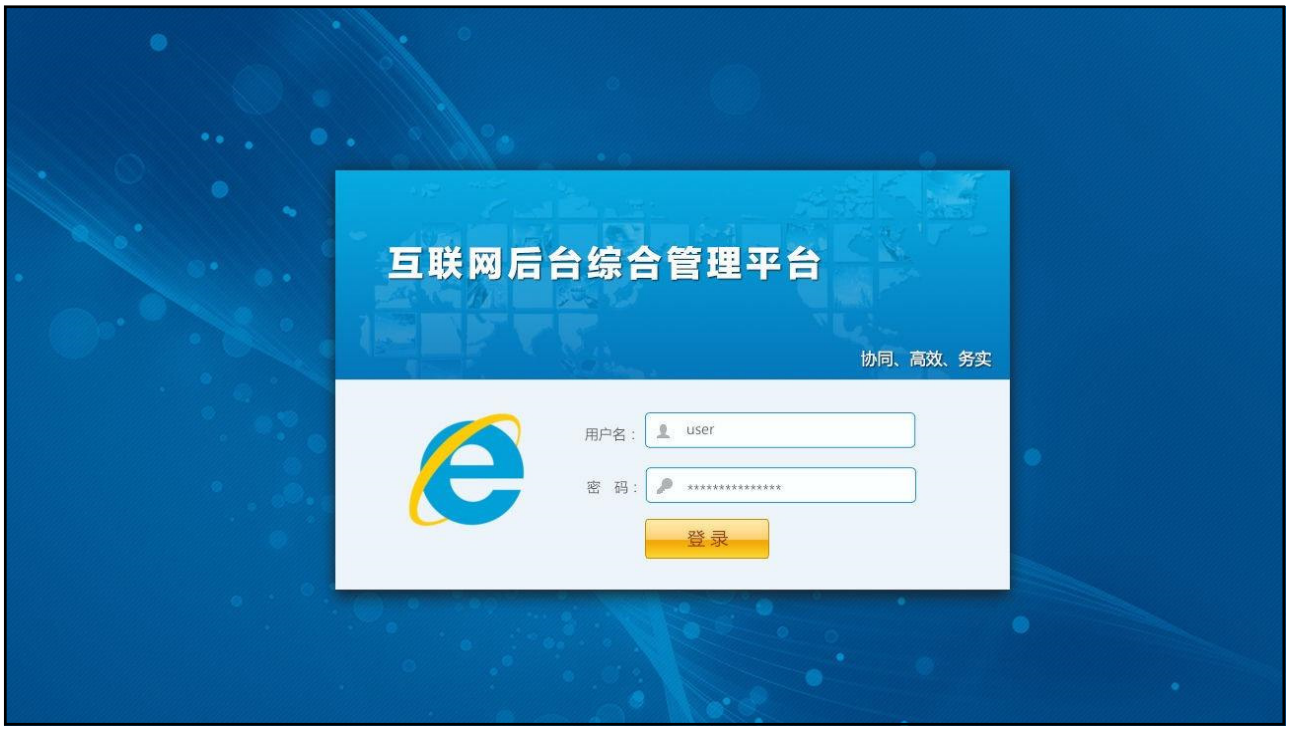
《人月神话(40周年中文纪念版)》

史前史中，没有别的场景比巨兽们在焦油坑中垂死挣扎的场面更令人震撼。上帝见证着恐龙、猛犸象、剑齿虎在焦油中挣扎。它们挣扎得越猛烈，焦油纠缠得就越紧，没有哪种猛兽足够强壮或具有足够的技巧，能够挣脱束缚，它们最后都沉到了坑底。

英国银行TSB 系统

测试不是很简单?

运行? 账号和密码正确?



Think 软件工程究竟是什么?

2.1 软件工程学科

- **一些事实(前提):**(或者说有求追求的目标)
  - A concerted effort should be made to **understand the problem** before a software solution is developed
    在定制软件之前，必须尽力理解需求。
  - **Design** becomes a pivotal(关键) activity
    设计已经成为关键活动。
  - Software should exhibit **high quality**
    软件必须保证高质量。
  - Software should be **maintainable**
    软件需具备可维护性。

有追求和目标

软件需要工程化

各种形式、各个应用领域的软件都需要工程化!

然而，对于某些开发队伍来说可能是好事，有些则是负担!

1.3 定义软件工程学科

- The IEEE **definition**:
  - Software Engineering:
  - (1) The application of a **systematic**(系统化), **disciplined**(规范化), **quantifiable**(可量化) approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

    将系统化的、规范化、可量化的方法应用于软件的开发、运行和维护，即将工程化的方法应用与软件。
  - (2) The **study** of approaches as in (1).

    在(1)中所述方法的研究。

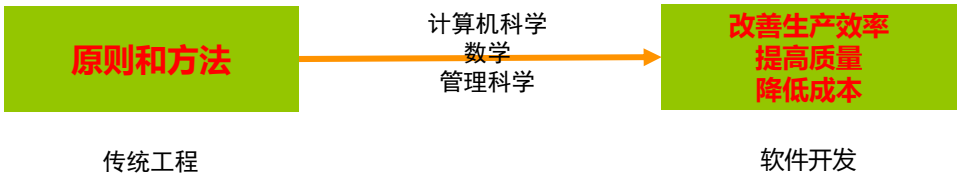
很多种定义……

- 软件工程
  - 是指导软件开发和维护的工程类学科，它以计算机科学理论及其他相关学科的理论为指导，采用工程化的概念、原理、方法和技术，进行软件的开发和维护，并与经过时间证明正确的管理方法与措施相结合，以较少的代价获取高质量的软件。
- 软件工程
  - 是软件工程是对需求、计算机技术、人员及其技能、时间、成本和其它资源的管理，并籍此在一个满足开发者需求的过程中形成一个满足客户需求的软件产品

课本为准！

软件工程的目标

- 目标：能够成功地完成软件开发项目。
  - 一个成功的项目目标是生产一个可接受的**产品**。
  - 应用计算机科学、数学及管理科学等原理，借鉴传统工程的原则和方法创建软件，以改善生产效率、提高质量、降低成本。



2.1 软件工程

- The seminal **definition**:
  - Software engineering is the establishment and use of sound **Engineering Principles** (工程原则) in order to obtain **economically** software that is reliable and works efficiently on real machines.

    软件工程是建立和使用一套**合理的工程原则**，以便经济地获得可靠的，可以实际机器上高效运行的软件。

工程原则

建立和使用一套合理的工程原则，以便**经济地**获得可靠的，可以实际机器上高效运行的软件

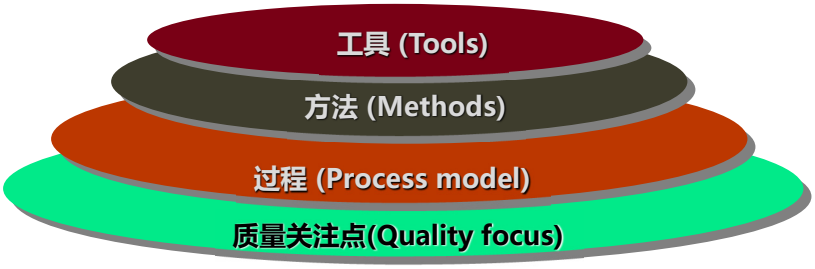
Think **根基?**

这样做的目标？目的？

根基在于质量关注点

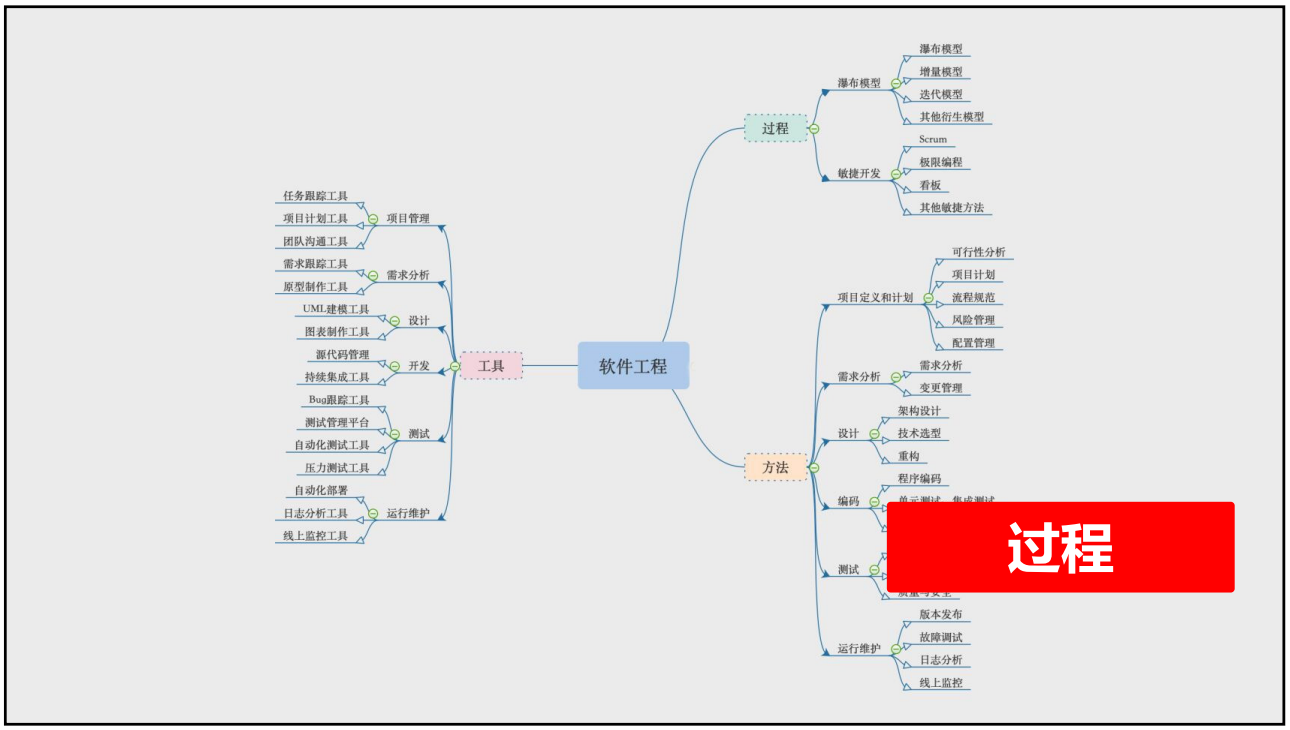
软件工程的根基在于质量关注点（Quality Focus）

A Layered Technology



软件工程层次图

Key Point 质量关注点(Quality focus)！！！



Think **您会怎么组织软件开发?**

阶段? 任务怎么划分? 顺序怎么组织?

软件过程

软件过程是工作产品构建时所执行的一系列**活动、动作和任务的集合**。

2.2 软件过程

- **软件过程**是工作产品构建时所执行的一系列**活动**、**动作**和**任务**的**集合**。
  - 活动(Activity)：沟通
  - 动作(Action)：体系结构设计
  - 任务(Task)：单元测试

2.2.1 过程框架-Framework Activities

框架活动

Framework Activities

- **沟通**(Communication)
- **策划**(Planning)
- **建模**(Modeling)
  - Analysis of requirements
  - Design
- **构建**(Construction)
  - Code generation
  - Testing
- **部署**(Deployment)

Communication 沟通　**Planning** 策划　**Modeling** 建模　**Construction** 构建　**Deployment** 部署

框架活动可以**迭代应用**。

2.2.2 普适性活动 (Umbrella Activities)

- 普适性活动贯穿软件项目的始终
  - 项目管理 (Software project management)
  - 技术审查 (Formal technical reviews)
  - 质量保证(Software quality assurance)
  - Software configuration management
  - Work product preparation and production
  - Reusability(复用) management
  - Measurement(测量)
  - Risk management(风险管理)

软件生存周期

也可以理解为国内版?

软件生存周期

SLC

(Software Life Cycle)

软件工程中重要概念

- 一个软件产品通常是从模糊的概念开始，逐步建立起产品的需求，并对需求进行说明，然后进行设计、实现和测试。如果客户是满意的，那么就可安装产品，并且开始运行和维护它。如果产品到达了其有用生命的尽头就会退役、报废或停止使用。
- 这一系列过程称为**软件的生命周期**。

软件生存周期的各个阶段

- 软件的生命周期归结为以下几个主要阶段：
  - **软件计划、需求分析、软件设计、编码、测试、维护与运行、退役等。**
  - 互相区别而又有联系。
  - 实际上，每个软件的生命周期有所不同，如有的软件可能在需求阶段花费几年的时间，有的软件在设计和实现阶段只需几个月时间，有的软件则在维护阶段可能长达十几年。

第一个阶段：软件计划(Planning)

- 确定要解决的“问题是什么”及“解决问题的可行方案”
  - 即确定要开发软件系统的总目标，给出它的功能、性能、可靠性以及接口等方面的概要性要求；
  - 从技术方面、经济方面、法律方面探讨解决问题的可能方案，对可利用的资源(如计算机硬件、软件、人力等)、成本、可取得的经济效益、开发的进度做出估计；
  - 制定出完成开发任务的实施计划等，提交管理机构评审。

问题+可行方案

第二个阶段：需求分析和规格说明(Requirement Analysis and Specification)

- 确定目标系统要“做什么”。
  - 即对软件计划阶段的要求进一步细化和求精，加强并集中软件的需求分析和规格说明，强调软件分析人员与用户、软件分析人员与软件开发人员的交互，充分理解软件的作用域、所需功能、性能及接口、安全与保密、人机工程与人机界面、数据定义及数据库、安装及验收等需求，落实用户所需文档、用户操作和运行需求、用户维护需求；
  - 写出软件需求规格说明书，提交管理机构评审。

需求+报告(规格说明)

第三个阶段：设计(Software Design)

- 确定目标系统要“怎么做”。
  - 软件设计是将需求转换成为软件的表示，包括数据结构、软件结构、接口表示和过程细节。
  - 通常将前三者划为软件的初步(概要)设计，后者则归为软件的详细设计。
    - 例如，可将需求转化为层次化的软件模块结构、模块所用的数据结构或数据库文件表示、模块之间接口描述、模块应完成的功能等，以及每个模块完成相应功能的过程细节如局部变量、内部数据结构、算法等。
  - 这些软件表示应该按照规定的标准形式加以描述，形成软件设计规格说明书，提交管理机构评审。

软件的表示(UML 原型等)

第四个阶段：编码(Coding)

- 编码体现了目标系统的“具体实现”。
  - 编码是将设计转换成计算机可以接受的语言代码——**源程序**。
  - 如果设计给出的描述很详细，那么编码几乎可以机械地完成。
  - 编码必须与设计表示一致、具有结构简单、清晰易读等良好的编码风格。

Java、Python等

第五个阶段：软件测试(Software Testing)

- 软件测试是保证软件质量的重要手段，其主要任务是检查该软件是否符合要求，其目的是发现软件存在的错误。
  - 软件测试应该是有计划地进行。
  - 软件测试依据软件规格说明设计测试用例，并施加在已经编码的程序上进行执行，经过对预期结果和实际执行结果的比较、分析，来发现程序的错误，最后形成测试报告。
  - 测试的主要过程有单元、集成、系统、验收和安装测试等。

测试用例！

第六个阶段：运行/维护(Running/Maintenance)

- 该阶段体现软件是否能够持久满足用户的需求。
  - 已交付的软件投入正式使用后，便进入运行和维护阶段。
  - 这个阶段可能持续若干年甚至几十年，因此占整个软件费用的比例最大，是相当重要的一个阶段。
  - 软件维护的实质是对软件继续进行查错、纠错、修改和确认的过程。
  - 无论是应用软件或系统软件，都要在使用期间不断改善和加强产品的功能和性能、适应运行环境的改变、纠正在开发期间未能发现的遗留错误。

根据实际情况

第七个阶段：报废/退役(Retirement)

- 当软件经过一段时期运行和服务后，便可能报废或退役。
- 主要的原因有：
  - 为满足用户的需求所做的维护费用太高，可能比新开发一个软件所花费的代价更高。
  - 维护的少量变化对于依赖性很强的软件的整体功能而言，有极大的危险。
  - 环境的变更(如硬件或操作系统)导致软件的更换。
  - 用户不再需要这个软件。

经济或发展等因素

软件生存周期是软件工程中一个重要概念

- 把软件的整个生存周期划分为较小的阶段，给每个阶段赋予确定而有限的任务，就能简化每一步的工作，使软件开发过程易于控制和管理。
  - 采用这种划分，使得每一个阶段的工作相对独立，有利于简化整个问题的解决，且便于**不同人员分工协作**。
  - 严格的科学的评审制度提高了软件的质量，从而大大提高了软件开发的生产率和成功率。

“量”化!

软件生存周期各阶段的关键问题(总结)

阶段	关键问题	结束标准(产品)
问题定义与可行性研究	问题是什么？有可行解吗？	可行性研究报告
需求分析	软件必须做什么？	需求分析报告
软件概要/总体设计	怎样概括地解决该问题？	概要设计报告
详细设计	怎样具体地解决该问题？	详细设计报告
编码	如何编码并最终实现该系统？	源程序清单
测试	寻找软件错误并使其符合要求？	测试报告
运行与维护	用户发现新问题？有新要求？ 如何解决新问题？满足新要求？	运行日志和维护记录

小组作业
仅涉及一部分

Think **过程必须严格遵守？**

灵活可适应

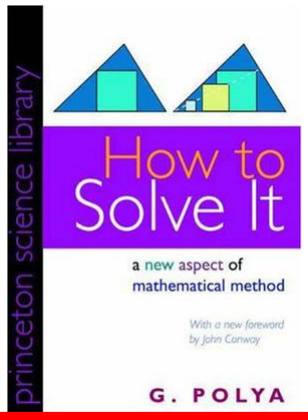
软件工程不是教条的法则，灵活可适应

软件工程实践的精髓——解决问题

解决问题的本质，也是软件工程实践的精髓！

2.3.1 实践的精髓 (The Essence of Practice)

- Polya suggests:
  - 1. Understand the problem (**Communication And Analysis**).
    - 理解问题(沟通和分析)
  - 2. Plan a solution (**Modeling And Software Design**).
    - 计划解决问题(建模和软件设计)
  - 3. Carry out the plan (**Code Generation**).
    - 实施计划(代码生成)
  - 4. Examine the result for accuracy (**Testing And Quality Assurance**).
    - 检查结果的正确性(测试和质量保证)



实践的精髓

2.3.2 通用原则 ——7个关注软件工程整体实践的原则

- 1: The Reason It All Exists(存在价值)
- 2: KISS (Keep It Simple, Stupid!)
- 3: Maintain the Vision(保持愿景)
- 4: What You Produce, Others Will Consume(关注使用者)
- 5: Be Open to the Future(面向未来)
- 6: Plan Ahead for Reuse(计划复用)
- 7: Think!

重要

英国政府电子服务的设计原则

- 1. 从用户的需求着手。
- 2. 少做。多考虑可复用性和共享，做政府部门应该做的事。
- 3. 通过数据来做设计决策。
- 4. 尽最大的努力让事情变简单。
- 5. 迭代，再迭代。
- 6. 为所有人设计。
- 7. 深刻的理解用户的使用场景。
- 8. 创建电子服务，而不是网站。
- 9. 保持一致，但不是死板的一成不变。
- 10. 保持开放，与全世界分享我们的代码、设计、想法，这样才会越来越好。

Think 学习系统或软件应如何考虑?

在完成小组作业时的思考：
结合课程的案例分析？相关的利益者？功能？

万能的？绝招？

希望一劳永逸！

2.4 软件神话

- 软件神话-关于软件及其开发过程被人**盲目相信**的一些说法。
  - Software Myths—**Erroneous beliefs** about software and the process

- Affect managers, customers (and other non-technical stakeholders) and practitioners
  - Are believable because they often have elements of truth, *but …*
  - Invariably lead to bad decisions, *therefore …*
  - Insist on reality as you navigate your way through software engineering

99%的程序员认不全的软件开发定律

https://mp.weixin.qq.com/s/bBAHYLxq9JI6Ew6VRqnlOA

资料.99%的程序员认不全的软件开发定律.pdf

Question?

Q&A?

College of Computer Science, Chongqing University

Software Engineering

A Practitioner's Approach Seventh Edition

2 软件工程

zmqmail@cqu.edu.cn 13708390417

重庆大学 计算机学院 COLLEGE OF COMPUTER SCIENCE