

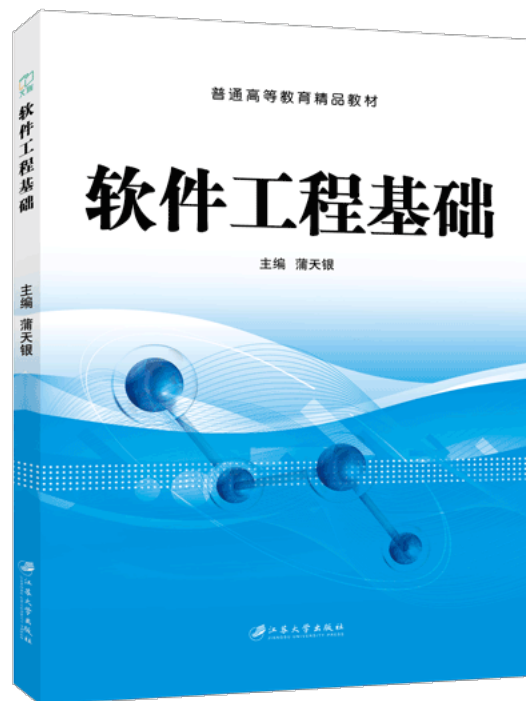
软件工程



教材和参考书

- 教材

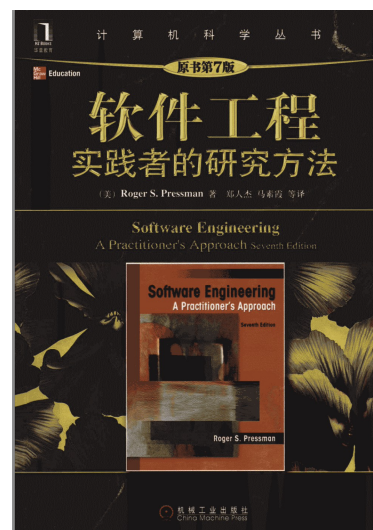
- 《软件工程基础》 蒲天银著，江苏大学出版社，**2023年10月**。



教材和参考书

• 参考书

- 《软件工程》著，杜文峰 袁琳 朱安民 叶聪等，清华大学出版社，**2023年2月**。
- 《软件工程：实践者的研究方法（第7版）》
Roger S.Pressman著，郑人杰，马素霞等译，机械工业出版社，**2011年5月**。



考核标准

- 平时成绩（参与性） **30%** —— 课堂签到 课堂表现
- 实验成绩（动手性） **30%** —— **6次** 实验报告
- 考核成绩（检查性） **40%** —— 期末考核成绩

第1章 软件与软件工程的观念

- 软件的概念、特性和分类
- 软件危机与软件工程
- 系统工程的目标
- 软件生存期
- 软件工程方法概述
- 软件工具概述
- 软件工程知识体系及知识域

1.1 软件的概念、特性和分类

- 软件的作用

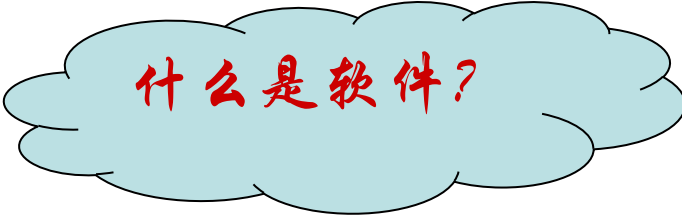
具有**产品**和**产品生产载体**的双重作用。

- (1) 作为产品，软件显示了由计算机硬件体现的计算能力，扮演着信息转换的角色：产生、管理、查询、修改、显示或者传递各种不同的信息。
- (2) 作为产品生产的载体，软件提供了计算机控制（操作系统）、信息通信（网络），以及应用程序开发和控制的基础平台（软件工具和环境）。

1.1 软件的概念、特性和分类

- 软件的概念

虽然软件对于现代的人并不陌生，但很多人对于软件的理解并不准确，“**软件就是程序，软件开发就是编程序**”的这种错误观点仍然存在。



什么是软件?

1.1 软件的概念、特性和分类

- **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执行的指令序列。
- **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 软件的概念、特性和分类

- 软件的特性

- (1) **形态特性**: 软件是无形的、不可见的逻辑实体。度量常规产品的几何尺寸、物理性质和化学成分对它却是毫无意义的。
- (2) **智能特性**: 软件是复杂的智力产品, 它的开发凝聚了人们的大量脑力劳动, 它本身也体现了知识实践经验和人类的智慧, 具有一定的智能。它可以帮助我们解决复杂的计算、分析、判断和决策问题。

1.1 软件的概念、特性和分类

- (3) **开发特性**: 尽管已经有了一些工具（也是软件）来辅助软件开发工作，但到目前为止尚未实现自动化。软件开发中仍然包含了相当份量的个体劳动，使得这一大规模知识型工作充满了个人行为和个人因素。
- (4) **质量特性**：目前还无法得到完全没有缺陷的软件产品。

1.1 软件的概念、特性和分类

- (5) **生产特性**:与硬件或传统的制造业产品的生产完全不同, 软件一旦设计开发出来, 如果需要提供多个用户, 它的复制十分简单, 其成本也极为有限。
- (6) **管理特性**: 由于上述的几个特点, 使得软件的开发管理显得更为重要, 也更为独特。

1.1 软件的概念、特性和分类

- (7) **环境特性**: 软件的开发和运行都离不开相关的计算机系统环境, 包括支持它的开发和运行的相关硬件和软件。软件对于计算机系统的环境有着不可摆脱的依赖性。
- (8) **维护特性**: 软件投入使用以后需要进行维护, 但这种维护与传统产业产品的维护概念有着很大差别。

1.1 软件的概念、特性和分类

- (9) **废弃特性**: 与硬件不同, 软件并不是由于被“用坏”而被废弃的。
- (10) **应用特性**: 软件的应用极为广泛, 如今它已渗入国民经济和国防的各个领域, 现已成为信息产业、先进制造业和现代服务业的核心, 占据了无可取代的地位。

1.1 软件的概念、特性和分类

- 软件的分类

- (1) 系统软件

- 操作系统
 - 数据库管理系统
 - 设备驱动程序
 - 通信和网络处理程序等

- (2) 支撑软件（工具软件）

- 纵向支撑软件：分析、设计、编码、测试工具等
 - 横向支撑软件：项目管理工具，配置管理工具等

1.1 软件的概念、特性和分类

- 软件分类

- (3) 应用软件

- 工程与科学计算软件
 - 商业数据处理软件
 - ERP软件
 - 计算机辅助设计 / 制造软件
 - 系统仿真软件
 - 智能产品嵌入软件
 - 事务管理、办公自动化软件

- (4) 可复用软件

- 标准函数库、类库、构件库等

1.2 软件危机与软件工程

- 软件危机
- 软件危机暴发于上个世纪六十年代末。
- 主要表现为：软件的发展速度远远滞后于硬件的发展速度，不能满足社会日益增长的软件需求。软件开发周期长、成本高、质量差、维护困难。

1.2 软件危机与软件工程

- 典型例子：美国IBM公司在1963年至1966年开发的IBM 360机的操作系统。
- 这个项目的负责人F.D.Brooks事后总结了他在组织开发过程中的沉痛教训时说：

.....正像一只逃亡的野兽落到泥潭中做垂死的挣扎，越是挣扎，陷得越深。最后无法逃脱灭顶的灾难，.....程序设计工作正像这样一个泥潭，.....一批批程序员被迫在泥潭中拼命挣扎，.....谁也没有料到竟会陷入这样的困境.....

1.2 软件危机与软件工程

具体来说，软件危机主要有以下一些典型表现：

- **对软件开发成本和进度的估计常常很不准确。**
- **用户对“已完成的”软件系统不满意的现象经常发生。**
- **软件产品的质量往往靠不住。**
- **软件常常是不可维护的。**
- **软件通常没有适当的文档资料。**
- **软件成本在计算机系统总成本中所占的比例逐年上升。**
- **软件开发生产率提高的速度，既跟不上硬件的发展速度，也远远跟不上计算机应用迅速普及深入的趋势。**

1.2 软件危机与软件工程

除了软件本身的特点，软件危机发生的主要原因有：

- (1) 缺乏软件开发的经验和有关软件开发数据的积累，使得开发工作的计划很难制定。
- (2) 软件人员与用户的交流存在障碍，使得获取的需求不充分或存在错误。
- (3) 软件开发过程不规范。如，没有真正了解用户的需求就开始编程序。
- (4) 随着软件规模的增大，其复杂性往往会呈指数级升高。需要很多人分工协作，不仅涉及技术问题，更重要的是必须有科学严格的管理。
- (5) 缺少有效的软件评测手段，提交用户的软件质量不能完全保证。

1.2 软件危机与软件工程

如何摆脱软件危机?

- 彻底消除“软件就是程序”的错误观念。
- 充分认识到软件开发应该是一种组织良好、管理严密、各类人员协同配合、共同完成的工程项目。
- 推广和使用在实践中总结出来的开发软件的成功技术、方法和工具。
- 按工程化的原则和方法组织软件开发工作。

1.2 软件危机与软件工程

- 软件工程概念的提出
- 为了克服软件危机，1968年10月在北大西洋公约组织（NATO）召开的计算机科学会议上，Fritz Bauer首次提出“软件工程”的概念，试图将工程化方法应用于软件开发。
- 在NATO会议上，Fritz Bauer对软件工程的定义是：
“软件工程就是为了经济地获得可靠的且能在实际机器上有效地运行的软件，而建立和使用完善的工程原理。”

1.2 软件危机与软件工程

- 什么是软件工程？

- 概括地说，软件工程是指导计算机软件开发和维护的工程学科。

采用工程的概念、原理、技术和方法来开发与维护软件，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，以经济地开发出高质量的软件并有效地维护它，这就是软件工程。

1.2 软件危机与软件工程

- 软件工程的若干定义

- **Boehm** : 运用现代科学技术知识来设计并构造计算机程序及为开发、运行和维护这些程序所必需的相关文件资料。
- **IEEE** : 软件工程是开发、运行、维护和修复软件的系统方法。
- **Fritz Bauer** : 建立并使用完善的工程化原则, 以较经济的手段获得能在实际机器上有效运行的可靠软件的一系列方法。

1.2 软件危机与软件工程

- **软件工程的基本原理**
 - **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**
 - **结果应能清楚地审查**
 - **开发小组的人员应该少而精**
 - **承认不断改进软件工程实践的必要性**

1.3 软件工程的目标

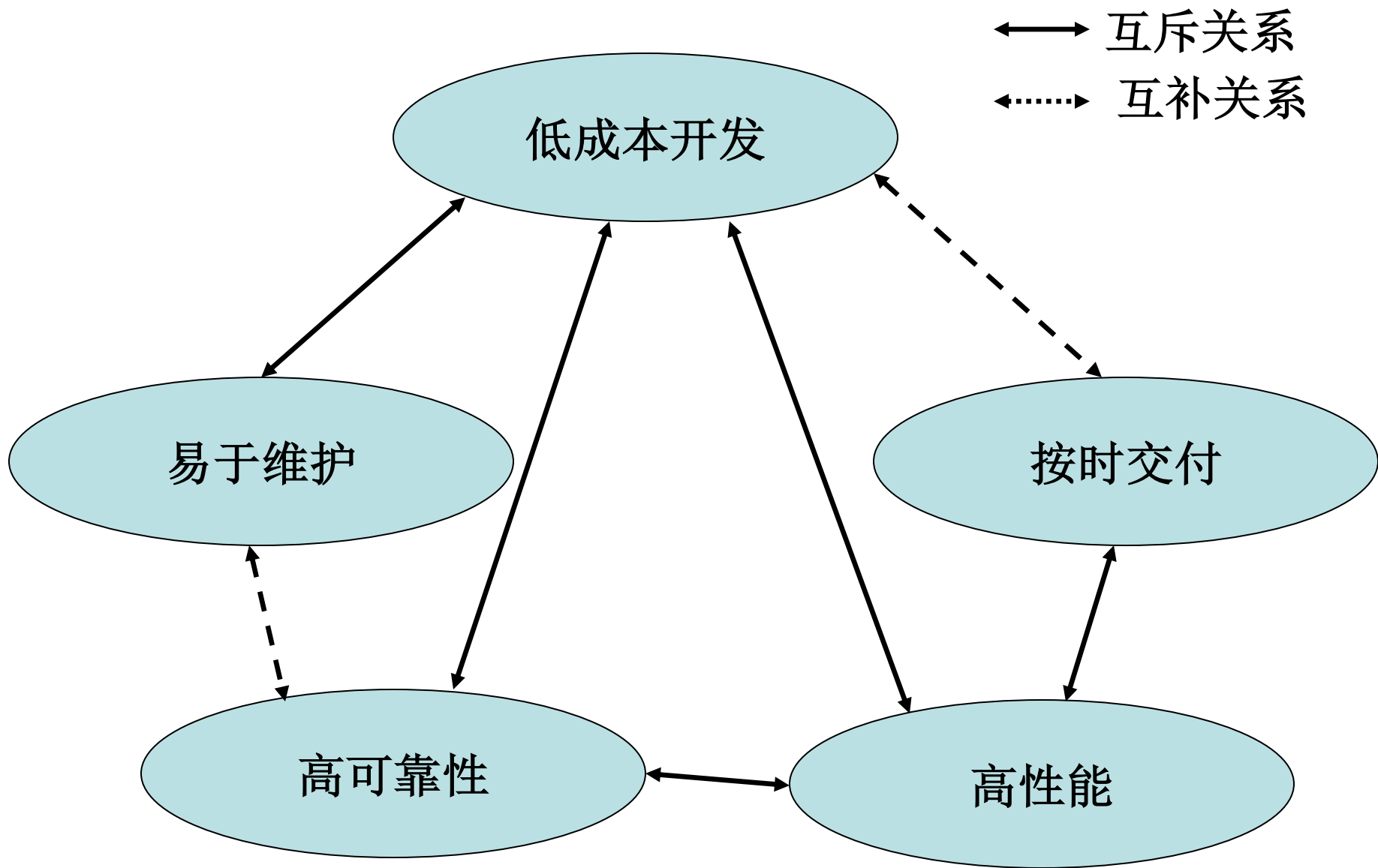
- 软件工程的目标是运用先进的软件开发技术和管理方法来提高软件的质量和**生产率**，也就是要以较短的周期、较低的成本生产出高质量的软件产品，并最终实现软件的工业化生产。

1.3 软件工程的目标

基本目标：

- 付出较低的开发成本
- 达到要求的软件功能
- 取得较好的软件性能
- 开发的软件易于移植
- 需要较低的维护费用
- 能按时完成开发工作，及时交付使用

1.3 软件工程的目标



1.3 软件工程的目标

软件的质量特性：功能性、可靠性、可使用性、效率、可维护性和可移植性。

- **功能性**是指软件所实现的功能达到它的设计规范和满足用户需求的程度；
- **可靠性**是指在规定的的时间和条件下，软件能够正常维持其工作的能力；
- **可使用性**是指为了使用该软件所需要的能力；
- **效率**是指在规定的条件下用软件实现某种功能所需要的计算机资源的有效性；
- **可维护性**是指当环境改变或软件运行发生故障时，为了使其恢复正常运行所做努力的程度；**可移植性**是指软件从某一环境转移到另一环境时所做努力的程度。

1.3 软件工程的目标

■ 学习软件工程，需要达到以下转变：

- 转变对软件的认识



- 转变思维定式



1.4 软件生存期

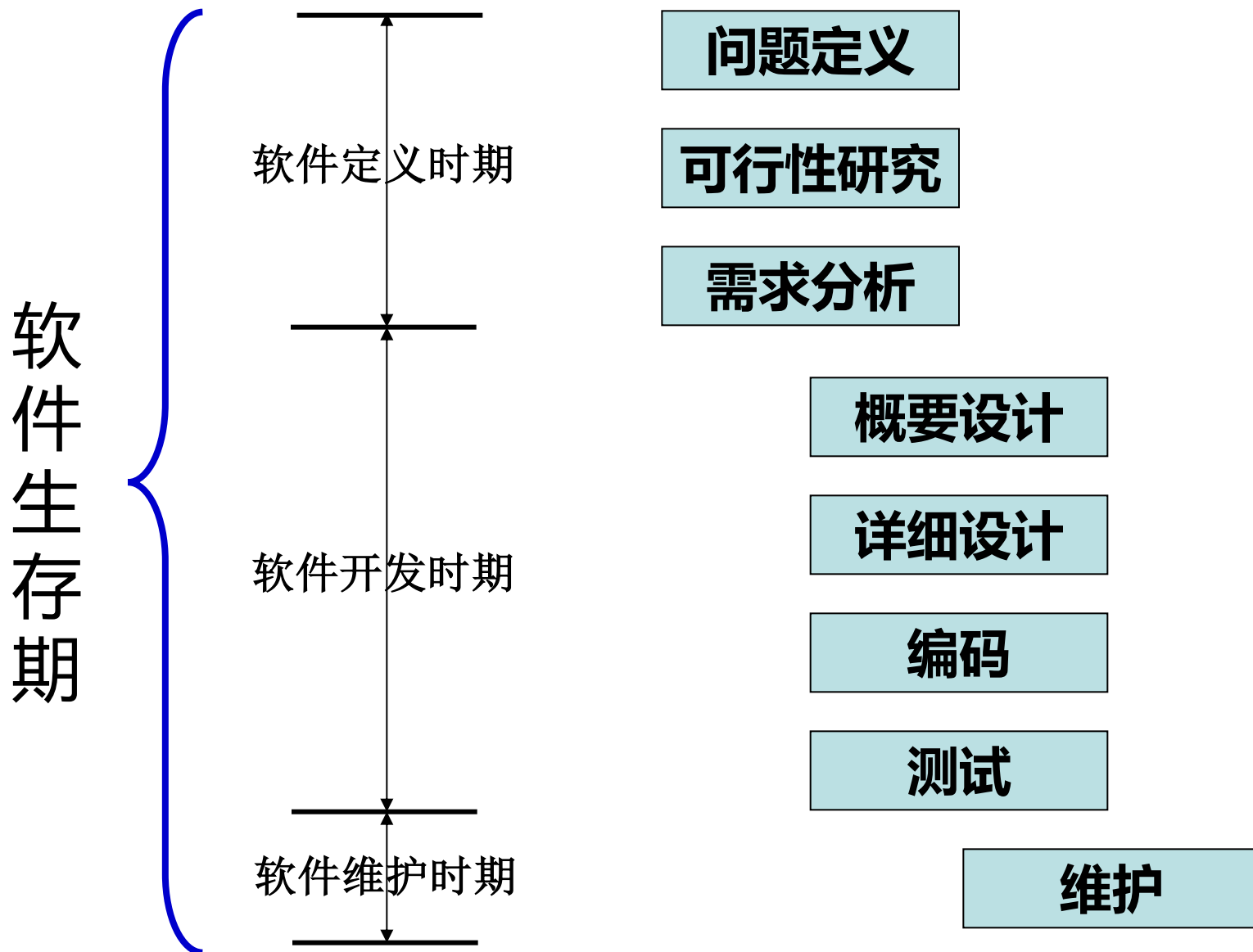
- 概念

软件也有一个孕育、诞生、成长、成熟和衰亡的生存过程，我们称这个过程为**软件生命周期**或**软件生存期**。

- 软件生存期分为三个时期

- 软件定义
- 软件开发
- 运行维护

1.4 软件生存期



1.4 软件生存期

- 软件定义时期
 - 确定总目标和可行性；
 - 导出策略和系统功能；
 - 估计资源和成本；
 - 制定工程进度表

分为三个阶段

- 问题定义
- 可行性研究
- 需求分析

1.4 软件生存期

- **问题定义**

关键问题是：“**要解决的问题是什么**”。

提交的内容为关于问题性质、工程目标和工程规模的书面报告。

- **可行性研究**

回答的关键问题是：“**上一个阶段所确定的问题是否有行得通的解决办法**”。

提交的内容为可行性研究报告，即从技术、经济和社会因素等方面研究各方案的可行性。

1.4 软件生存期

• 需求分析和定义

- ✓ 对用户提出的要求进行分析并给出详细的定义
- ✓ 准确地回答“**目标系统必须做什么**”这个问题。
也就是对目标系统提出**完整、准确、清晰、具体**的要求。
- ✓ 编写**软件需求说明书或系统功能说明书及初步的系统用户手册**
- ✓ 提交管理机构**评审**

1.4 软件生存期

- 软件开发时期

- **任务**：具体设计和实现前一个时期即软件定义时期定义的软件。
- **执行人**：系统设计员，高级程序员，程序员，测试工程师和辅助人员等
- **阶段划分**：分为概要设计、详细设计、编码和单元测试、集成测试和系统测试。其中前两个阶段又称为系统设计，后两个阶段又称为系统实现。

1.4 软件生存期

➤ 概要设计

- 概括地回答“怎样实现目标系统?”。
- 设计程序的体系结构，也就是确定程序由哪些模块组成以及模块间的关系。
- 提交的文档是概要设计说明书。

➤ 详细设计

- 回答“应该怎样具体地实现这个系统”。
- 详细地设计每个模块，确定实现模块功能所需要的算法和数据结构。
- 提交的文档是软件的详细设计说明书。

1.4 软件生存期

➤ 程序编码和单元测试

- 写出正确的容易理解、容易维护的程序模块。
- 提交的文档为源程序、详尽的程序说明和单元测试报告。

➤ 集成测试和系统测试

- 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。
- 提交的文档为测试计划、详细测试方案以及实际测试结果等。

1.4 软件生存期

- **软件运行维护时期**

主要任务是使软件持久地满足用户的需要，通常有4类维护活动：

- **改正性维护**，也就是诊断和改正在使用过程中发现的软件错误；
- **适应性维护**，即修改软件以适应环境的变化；
- **完善性维护**，即根据用户的要求改进或扩充软件，使它更完善；
- **预防性维护**，即修改软件为将来的维护活动预先做准备。

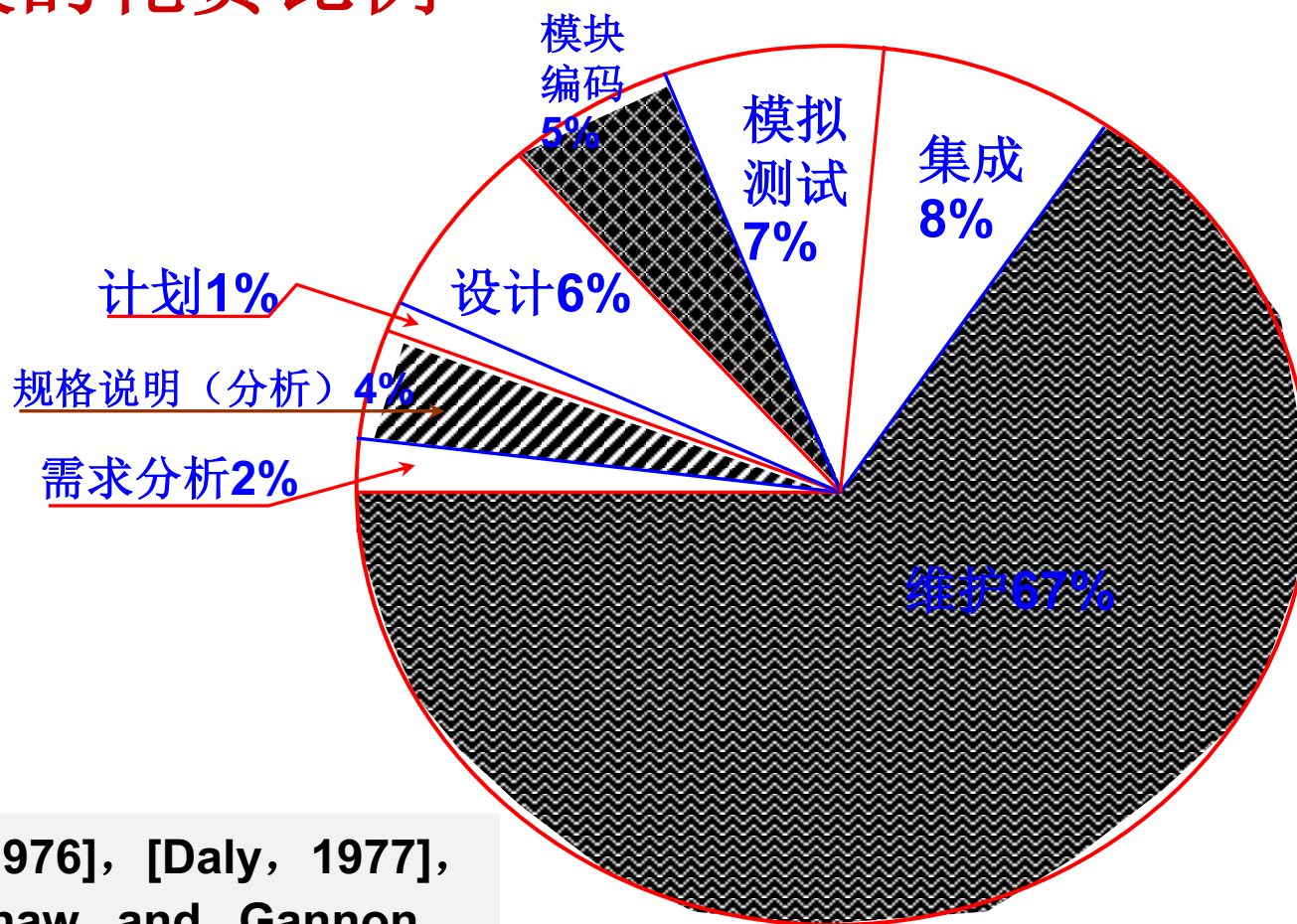
1.4 软件生存期

- 开发过程中的典型文档

- **软件需求规格说明书**：描述将要开发的软件做什么。
- **项目计划**：描述将要完成的任务及其顺序，并估计所需要的时间及工作量。
- **软件测试计划**：描述如何测试软件，使之确保软件应实现规定的功能，并达到预期的性能。
- **软件设计说明书**：描述软件的结构，包括概要设计及详细设计。
- **用户手册**：描述如何使用软件。

1.4 软件生存期

- 各阶段的花费比例



来自[Elshoff, 1976], [Daly, 1977], [Zelkowitz, shaw and Gannon, 1979]和[Boehm, 1981]的统计数据

1.5 软件工程方法概述

- 概念

- 软件工程包含**技术**和**管理**两方面的内容，是技术和管理紧密结合所形成的工程学科。
- 通常将软件开发全过程中使用的一整套技术方法的集合称为**方法学**(methodology)，也称为**范型**(paradigm)。
- 目前使用最广泛的软件工程方法学：**传统方法学**（**结构化方法学**），**面向对象方法学**。

1.5 软件工程方法概述

- 三要素：方法、工具和过程。
- 软件工程**方法**为软件开发提供了“如何做”的技术；
- 软件**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；
- **过程**是为了获得高质量的软件所需要完成的一系列任务框架，它规定了完成各项任务的工作步骤。

1.5 软件工程方法概述



软件工程层次图

1.5 软件工程方法概述

- 结构化方法学

也称为生命周期方法学或结构化范型。将软件生命周期的全过程依次划分为若干个阶段，采用结构化技术来完成每个阶段的任务。

特点：

- (1) 强调自顶向下顺序地完成软件开发的各阶段任务;**
- (2) 结构化方法要么面向行为，要么面向数据，缺乏使两者有机结合的机制。**

1.5 软件工程方法概述

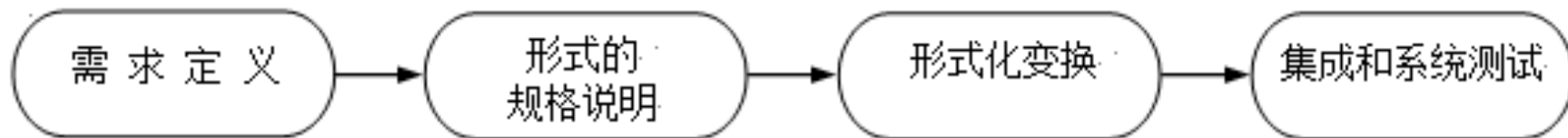
- 面向对象方法学

- 是将数据和对数据的操作紧密地结合起来的方法。
- 软件开发过程是多次反复迭代的演化过程。
- 面向对象方法在概念和表示方法上的一致性，保证了各项开发活动之间的平滑过渡。
- 对于大型、复杂及交互性比较强的系统，使用面向对象方法学更有优势。

1.5 软件工程方法概述

- 形式化方法

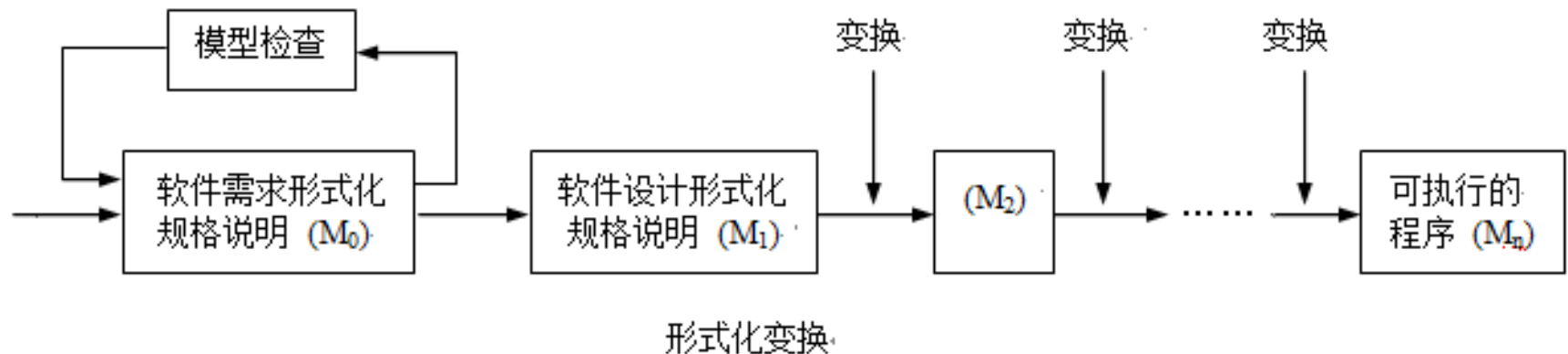
- 形式化方法是一种基于形式化数学变换的软件开发方法，它可将系统的规格说明转换为可执行的程序。



1.5 软件工程方法概述

● 形式化方法的主要特点

- 软件需求规格说明被细化为用数学记号表达的详细的形式化规格说明；
- 设计、实现和单元测试等开发过程由一个变换开发过程代替。通过一系列变换将形式的规格说明细化成为程序。



1.6 软件工具概述

- 软件工具的概念
- 软件工具的发展
- 软件工具的分类
- 常用软件工具介绍

1.6 软件工具概述

- 软件工具的概念

- 软件工具是指能支持软件生存周期中某一阶段（如系统定义、需求分析、设计、编码、测试或维护等）的需要而使用的软件工具。
- 早期的软件工具主要用来辅助程序员编程，如编辑程序、编译程序、排错程序等。
- 软件工具通常也称为CASE(计算机辅助软件工程，computer aided software engineering)工具。

1.6 软件工具概述

● 软件工具的发展

- 50年代末期出现了编辑程序、汇编程序和各种程序语言的编译程序或解释程序、连接程序、装配程序、排错程序等辅助软件编程活动的工具。
 - 60年代末提出软件工程的概念后，支持软件开发、维护、管理等过程的各种活动的工具也应运而生。
 - 80年代中期提出了软件过程的新概念，人们开始研制过程建模的工具、过程评价工具。
- 如今，软件工具已由单个工具向多个工具集成的方向发展，且注重工具间的平滑过渡和互操作性。

1.6 软件工具概述

● 软件工具的分类

- **支持软件开发过程的工具**：主要有需求分析工具、设计工具(通常还可以分为概要设计工具和详细设计工具)、编码工具、排错工具、测试工具等。
- **支持软件维护过程的工具**：主要有版本控制工具、文档分析工具、信息库开发工具、逆向工程工具、再工程工具等。
- **支持软件管理过程和支持过程的工具**：主要有项目管理工具、配置管理工具、软件评价工具等。

1.6 软件工具概述

- 常用软件工具

- **需求分析与设计工具**：目前使用的大多数工具既支持需求分析工作，也支持软件设计工作。

- IBM Rational Requirement Composer
- Enterprise Architect (EA)
- IBM Rational Software Architect
- Rational Rose
- Microsoft Office Visio
- PowerDesigner

1.6 软件工具概述

- **编码工具与排错工具**：现代软件开发使用集成开发环境IDE，一般包括代码编辑器、编译器、调试器和图形用户界面工具。
 - Turbo系列的集成开发环境（包括Turbo C, Turbo Pascal等）、C++ Builder、Delphi等
 - Microsoft Visual Studio（简称VS）系列
 - Jbuilder
 - Eclipse、MyEclipse
 - Netbeans

1.6 软件工具概述

- **测试工具**：分为程序单元测试工具、组装测试工具和系统测试工具。
- ✓ **单元测试工具**：目前最流行的单元测试工具是xUnit系列框架。
- ✓ **组装测试工具**：也称为集成测试或联合测试。包括WinRunner、IBM Rational Functional Tester、TestDirector等。
- ✓ **系统测试工具**：系统测试是对整个基于计算机的系统进行一系列不同考验的测试。典型工具包括IBM Rational Robot、IBM Rational Quality Manager、LoadRunner 等。

1.7 软件工程知识体系及知识域

- **软件工程教育(3个历史时期)**

(1) 1978年以前：软件工程教育以计算机专业的一门孤立的课程形式存在。

(2) 1978—1988年期间：早期的研究生学位教育，开始建立软件工程专业研究生学位教育项目。

(3) 1988年以后：快速发展的研究生学科教育，使软件工程的理论快速发展，其中，卡内基·梅隆大学软件工程研究所（SEI）的影响不可忽视。

1.7 软件工程知识体系及知识域

- 软件工程知识体

- 软件工程已从计算机科学与技术中脱离出来，逐渐形成了一门独立的学科。对其知识体系的研究从20世纪90年代初就开始了。
- 标志是美国Embry-Riddle航空大学计算与数学系Thomas B.Hilburn教授的“**软件工程知识体系指南**”（**Guide to Software Engineering Body of Knowledge , SWEBOK**）研究项目。

1.7 软件工程知识体系及知识域

- 软件工程知识体系指南的目标

- (1) 促使软件工程本体知识成为世界范围的共识。
- (2) 澄清软件工程与其他相关学科，如与计算机科学、项目管理、计算机工程以及计算机数学之间的关系，并且确定软件工程学科的范围。
- (3) 反映软件工程学科内容的特征。
- (4) 确定软件工程本体知识的各个专题。
- (5) 为相应的课程和职业资格认证材料的编写奠定基础。

1.7 软件工程知识体系及知识域

- 软件工程知识体系指南的内容
 - SWEBOK指南将软件工程知识体系划分为15个知识域（knowledge areas, KA），这些知识域又划分为三类：
 - **软件工程基础类**：数学基础、计算基础、工程基础、软件工程经济学。
 - **软件生存期过程类**：软件工程模型和方法、软件需求、软件设计、软件构造、软件测试、软件维护。
 - **软件工程管理类**：软件工程过程、软件工程管理、软件配置管理、软件质量、软件工程专业实践。



That's All!