软件工程与实践

授课教师：邱元杰　　电子邮箱：yuanjiq@126.com，微信电话: 13679081552

第7章 软件维护

- 软件维护的概念和分类
- 软件维护应注意的问题
- 软件维护技术

7.1 软件维护的概念和分类

- 基本概念
- 基本类型

7.1.1 软件维护的概念

定义

IEEE/EIA 12207[ISO/IEC2008]中对软件维护的定义是：软件维护是指由于软件产品出现问题或需要改进而对代码及相关文档的修改，其目的是对现有软件产品进行修改的同时保持其完整性。

7.1.2 软件维护的成本

- **软件维护阶段一般要消耗软件生命周期中经费开支的大部分。**

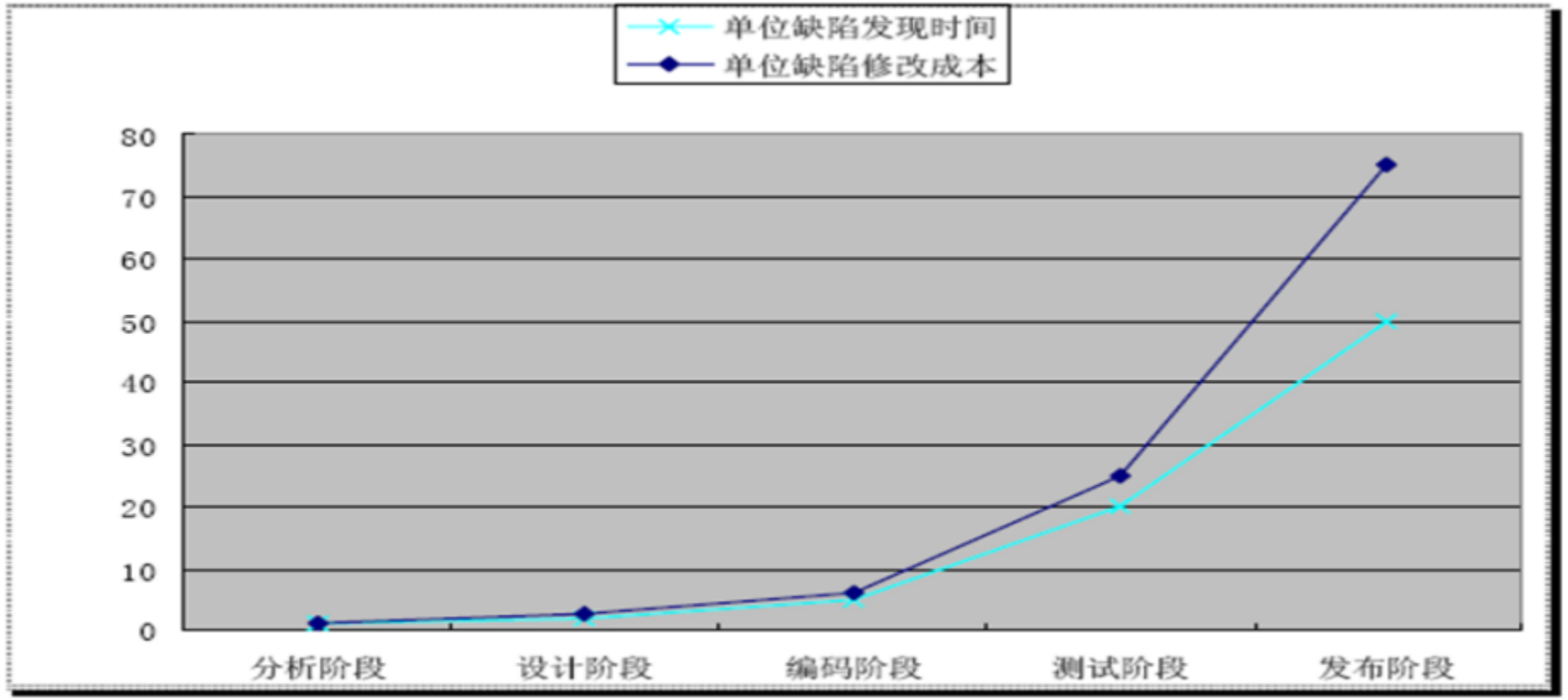
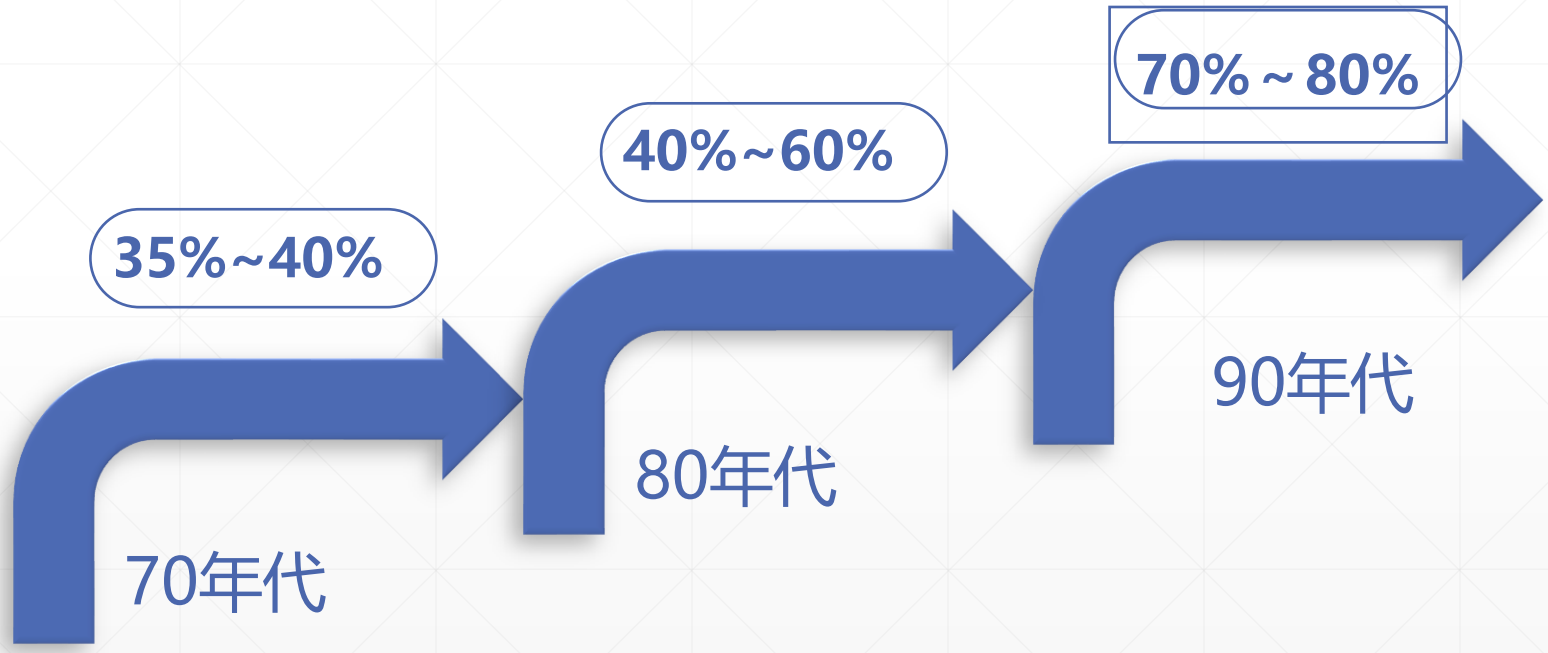


图 3、 缺陷代价曲线

不同年代用于维护已有软件的费用占软件总预算成本情况



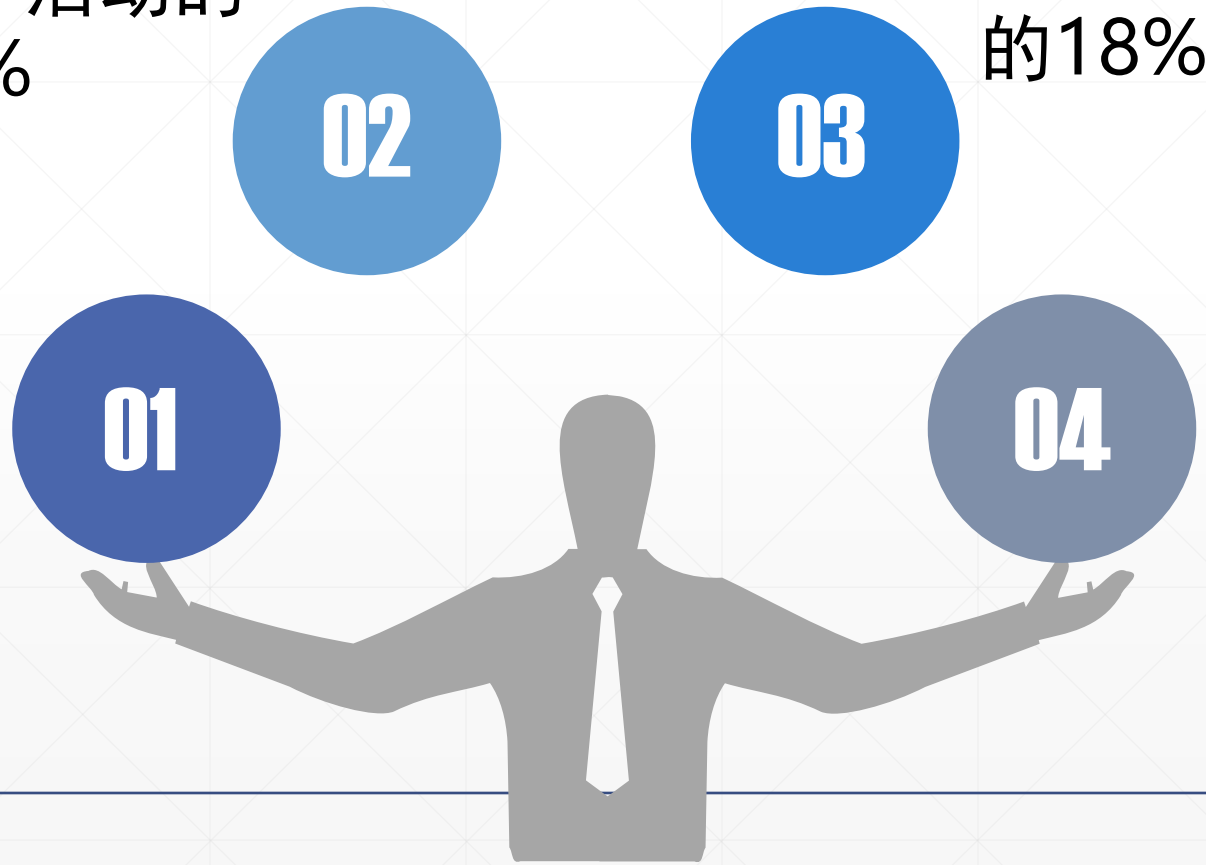
7.1.2 软件维护的基本类型

纠错性维护

占全部维护活动的17%～21%

适应性维护

占全部维护活动的18%～25%



完善性维护

占全部维护活动的50%～66%

预防性维护

只占全部维护活动的4%左右

1）完善性维护

- 在软件的使用过程中，用户往往会对软件提出新的功能与性能要求。
- 为了满足这些要求，需要修改或再开发软件，以扩充软件功能、增强软件性能、改进加工效率、提高软件的可维护性。
- 这种情况下进行的维护活动叫做完善性维护。

实践证明：

- 大部分维护工作是改变和加强软件，而不是纠错。
- 完善性维护不一定是救火式的紧急维修，而可以是有计划、有预谋的一种再开发活动。
- 来自用户要求扩充、加强软件功能、性能的维护活动约占整个维护工作的50%。

2）纠错性维护

- 在软件交付使用后，因开发时测试的不彻底、不完全，必然会有部分**隐藏的错误**遗留到运行阶段。
- 这些隐藏下来的错误在某些特定的使用环境下就会暴露出来。
- 为了识别和纠正软件错误、改正软件性能上的缺陷、排除实施中的误用，应当进行的诊断和改正错误的过程就叫做**纠错性维护**。

3）适应性维护

- 在使用过程中，
  - 外部环境（新的硬、软件配置）
  - 数据环境（数据库、数据格式、数据输入/输出方式、数据存储介质）

  可能发生变化。
- 为使软件适应这种变化，而去修改软件的过程就叫做适应性维护。

4）预防性维护

- 预防性维护是为了提高软件的可维护性、可靠性等，为以后进一步改进软件打下良好基础。
- 预防性维护定义为：采用先进的软件工程方法对需要维护的软件或软件中的某一部分（重新）进行设计、编制和测试。

7.2 软件维护应注意的问题

- 技术方面
- 管理方面
- 维护费用估算

7.2.1 维护的困难性（why）

配置管理工作不到位

人员变动造成的影响

许多软件的可读性差

任务紧、时间急的情况下处理维护请求

7.2.2 软件维护中应注意的问题：技术方面

- 程序的理解
- 测试
- 影响分析
- 可维护性

决定软件可维护性的主要因素

- (1) 可理解性
- (2) 可测试性
- (3) 可修改性
- (4) 可移植性
- (5) 可重用性

影响软件可维护性的维护环境的因素

- (1) 软件维护的文档
- (2) 软件的运行环境
- (3) 软件的维护组织
- (4) 软件维护质量

7.2.3 软件维护中应注意的问题：管理方面

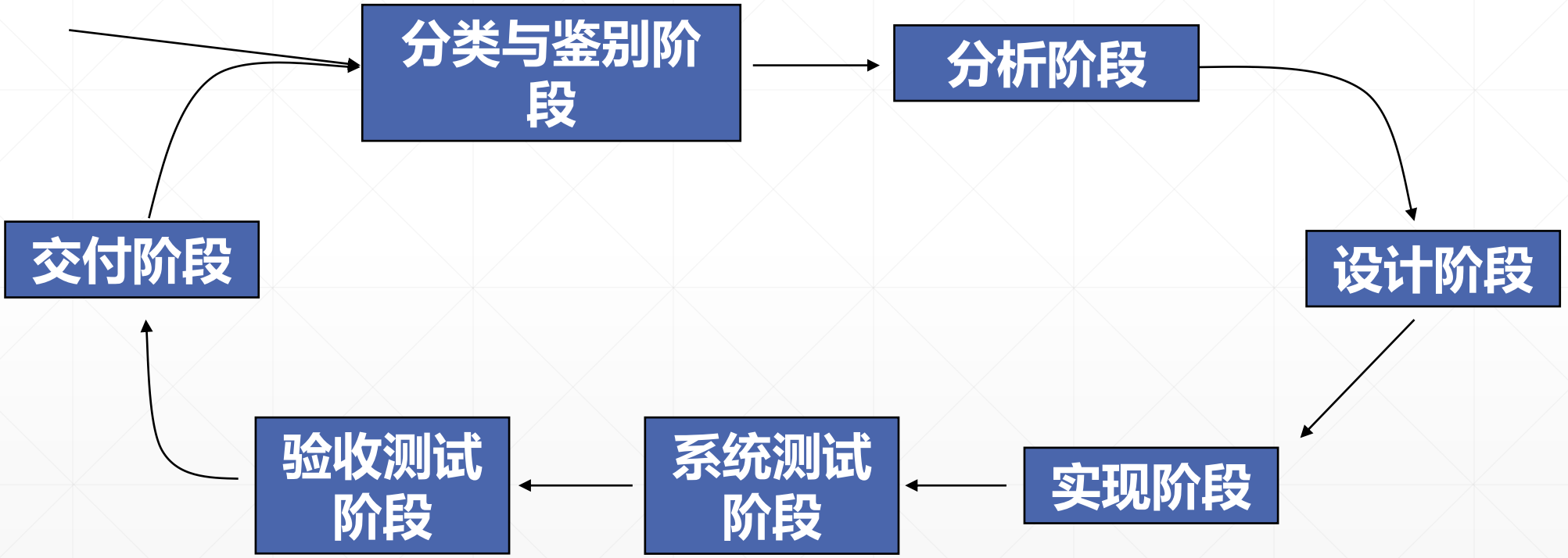
- 契合组织的目标
- 人力资源
- 过程
- 如何组织维护活动
- 外包

7.2.4 软件维护中应注意的问题：维护费用估算

- **参数模型**
  - $M = P + K \times \exp(c-d)$
  - M是维护用的总工作量，P是生产性工作量，K是经验常数，
  - exp是以e为底的指数函数
  - c是复杂程度，d是维护人员对软件的熟悉程度
- **基于经验**
  - 专家判断、类推、工作分解结构

7.2.5 软件维护过程模型

- IEEE维护模型图



7.3 软件维护技术

- 程序的理解
- 软件再工程
- 软件逆向工程

7.3.1 程序的理解

- 程序理解的任务：以软件维护、升级和再工程为目的，在不同的抽象级别上建立基本软件的概念模型，包括从代码本身的模型到基本应用领域的模型，即建立从问题/应用域到程序设计/实现域的映射集

7.3.1 程序理解的具体任务

- 通过检查单个的程序设计结构，程序被表示成抽象语法树、符号表或普通源文本
- 尽量做到程序隐含信息的显性表示及程序内部关系的可视化
- 从源代码中提取信息，并存放在通用的数据库中，然后通过查询语言对数据库进行查询
- 检查程序构造过程中的结构关系，明确表示程序组成部分之间的依赖关系。
- 识别程序的高层概念，如标准算法、数据结构、语法及语义匹配等。

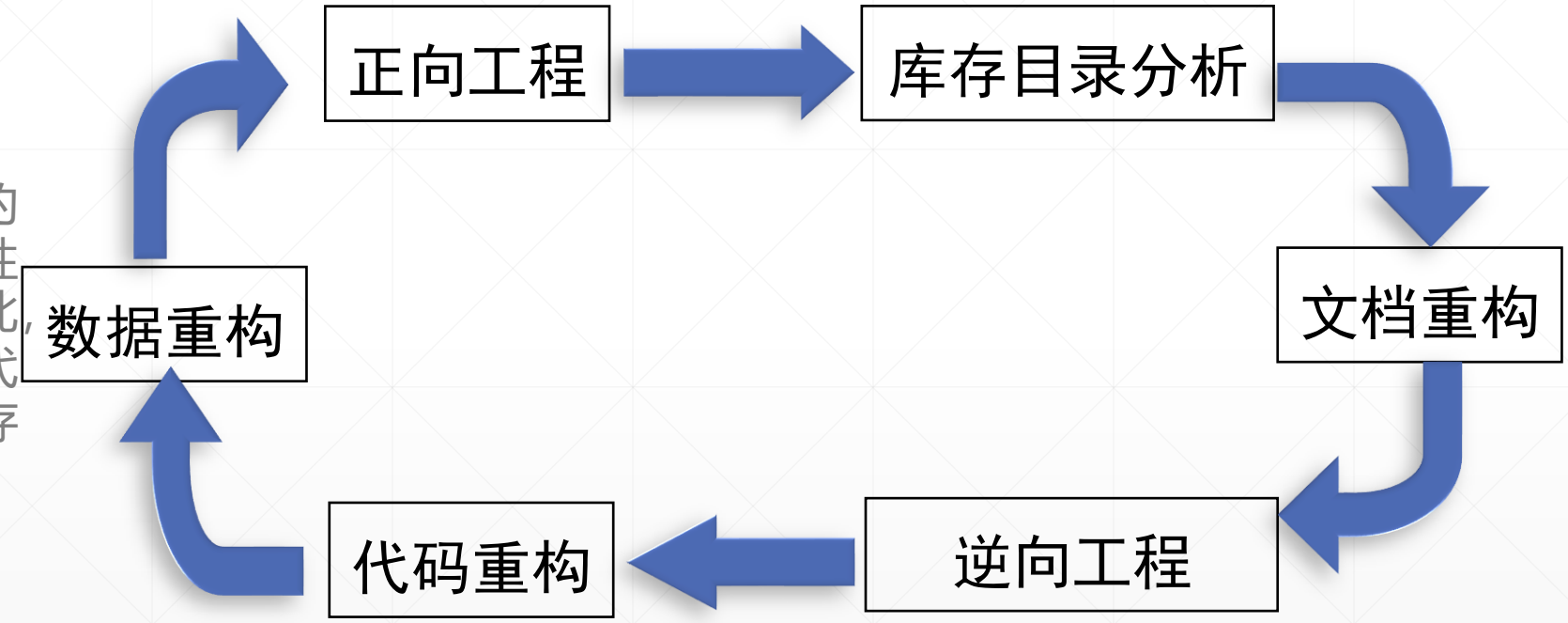
7.3.2 软件再工程

- 定义
- 软件再工程（Re-engineering）指对现有软件进行仔细审查和改造，对其进行重新构造，使之成为一个新的形式，同时包括随之产生的对新形式的实现。

7.3.2 软件再工程模型

正向工程也称为革新或改造。
正向工程过程应用现代软件工程的概念、原理、技术和方法，重新开发现有的某些应用系统。

仔细的、分析库存目录，按照业务重要程度、寿命、当前可维护性等标准，把库中的应用小排序，从中选出再工程的侯选者。然后合理地分配再工程所需要的资源。



对数据体系结构差的程序很难进行适应性和完善性维护，因此，数据体系结构比源代码对程序的长期生存力有更大的影响。

代码重构并不修改程序的体系结构，它只关注个体模块的设计细节以及在模块中定义的局部数据结构。

分析程序以便在比源程序更高的抽象层次上创建出程序的某种描述的过程，也就是说，逆向工程是一个恢复设计结果的过程。

7.3.3 软件逆向工程

- **软件逆向工程（Software Reverse Engineering）**
  - 是分析目标系统，识别系统的构件及其交互关系，并且通过高层抽象或其他形式来展现目标系统的过程。
  - 对逆向工程而言，抽象的层次、完备性、工具与分析人员协同工作的程度、过程的方向性等因素是需要考虑的。

7.3.3 逆向工程主要内容

为了理解过程抽象，需要在不同的抽象级别（系统级、程序级、构件级、模式级和语句级）分析代码

弄清几个问题：

- 界面必须处理的基本动作是什么?
- 系统对这些动作的行为反应的简要描述是什么?
- 有哪些界面的等价概念是相关的?

发生在不同的抽象层次：

- 内部数据结构的逆向工程
- 数据库结构的逆向工程

- **静态模型逆向工具**
  - Rational Rose
  - Rigi
  - JBPAS
- **动态模型逆向工具**
  - SCED
  - ISVis
  - Borland
  - Together

本章小结

感谢观看！

授课教师：邱元杰　　电子邮箱：yuanjiq@126.com