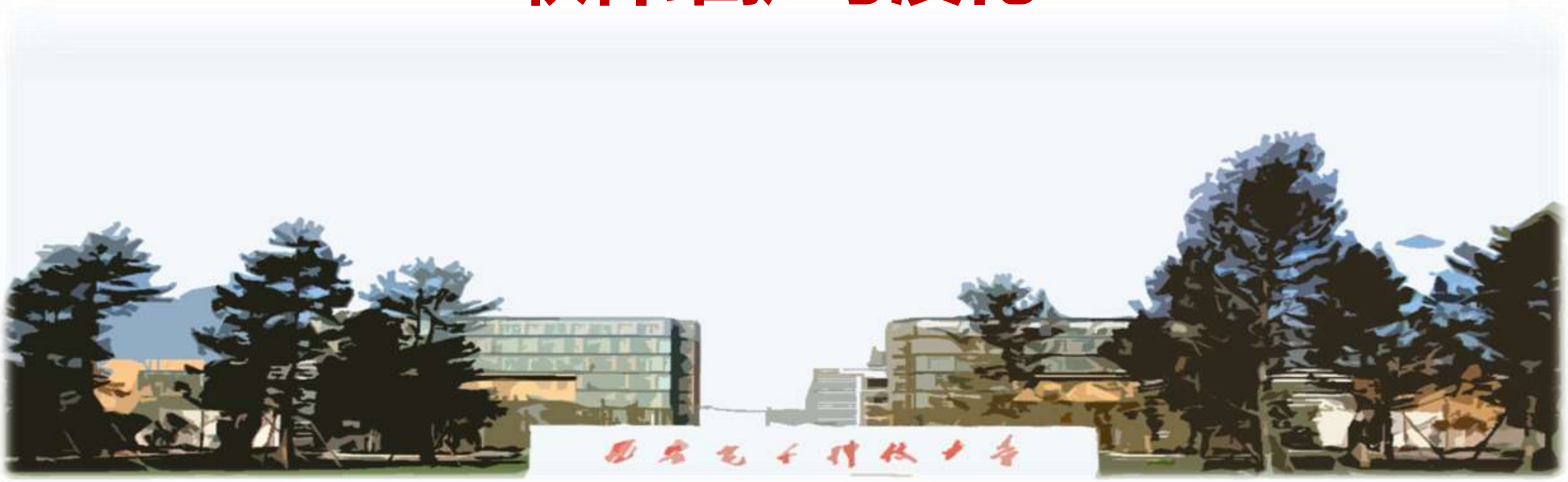
西安电子科技大学
XIDIAN UNIVERSITY

软件维护与演化



目录

物理设备的维护问题

- 物理设备在使用过程中会出现故障，需要进行维护

软件是否也会出故障？为什么？出了故障怎么办？

1.1 软件也需要进行维护

- **出故障，不可正常工作**
  - ✓潜在的缺陷产生软件错误
  - ✓需要对这些缺陷进行纠正
- **服务变化，需要升级**
  - ✓软件需求发生了变化
  - ✓需要增强软件的功能和服务
- **运行环境变化，需要适应**
  - ✓需要改变软件以在新的环境中运行

1.2 何为软件维护

- 软件在**交付使用**后，由于应用需求和环境变化以及自身问题，对软件系统进行**改造和调整**的过程
  - ✓软件自身问题，发现了一些新的缺陷
  - ✓运行环境变化，从Windows ➔ Linux
  - ✓软件需求变化，需要增加一些新的需求

小米便签软件的维护

- **设置便签访问密码**
- **解除便签访问密码**
- **根据密码访问便签**
- **改善程序代码质量**

1.3 软件维护的形式

- 纠正性维护
- 完善性维护
- 适应性维护
- 预防性维护

纠正性维护

- **何为纠正性维护**
  - ✓纠正软件中的**缺陷和错误**
- **起因**
  - ✓用户在使用软件过程中一旦发现缺陷，他们会向开发人员提出纠正性维护的请求
- **目的**
  - ✓诊断和改正软件系统中潜藏的缺陷

为什么软件在使用时还有缺陷存在?

适应性维护

❑何为适应性维护?

✓对软件进行改造以便适应新的**运行环境和平台**

❑起因

✓软件运行于一定的环境(硬件、OS、网络等)之上，运行环境发展很快，出现了变化

❑目的

✓适应环境变化和发展而对软件进行维护

为什么软件需要适应环境的变化?

改善性维护

❑何为改善性维护？

✓对软件进行改造以**增加新的功能、修改已有的功能**

❑起因

✓在软件系统运行期间，用户可能要求增加新的功能、建议修改已有功能或提出其他改进意见

❑目的

✓满足用户日益增长的各种需求而对软件系统进行的改善和补充

小米便签维护中是否涉及完善性维护？做了哪些完善性工作

预防性维护

❑何为预防性维护？

✓对软件结构进行改造以便提高软件的**可靠性和可维护性**等

❑起因

✓为进一步改善软件系统的可维护性和可靠性，为以后的软件改进奠定基础的维护活动

❑目的

✓获取软件结构，重新改善软件结构

小米便签需要做预防性维护吗？

1.4 软件维护的特点

❑同步性

- ✓软件维护需要与软件使用同步进行

❑周期长

- ✓软件维护周期会更长，一些软件会服役十几年甚至几十年的时间

❑费用高

- ✓维护成本高达总成本80%以上，维护费用是开发费用的3倍以上

❑难度大

- ✓充分理解待维护软件的架构、设计和代码，这极困难。尤其是在软件设计文档缺失的情况下，这一问题更为突出
- ✓50%-90%的时间被消耗在理解程序上

1.5 软件维护工程师

- **负责完成软件维护的各项工作，具备多方面的能力和素质**
  - ✓**阅读理解能力**，阅读待维护软件的相关文档和代码，以此来掌握软件的架构和相关设计
  - ✓需要有非常强的**掌握新技术能力**，以此来指导软件维护
  - ✓**洞察和分析能力**，根据软件缺陷的症状快速定位缺陷所在代码的可能位置，能够针对增强的软件需求明确要对软件的哪些部分进行重设计
  - ✓**沟通能力**，与客户、用户和软件工程师进行沟通的能力

1.6 软件演化及其特点

- **软件演化**是指针对软件的大规模功能增强和结构调整，以实现变化的软件需求，或者提高软件系统的质量
  - ✓**功能增强粒度大**，软件演化针对的是粗粒度需求变化及功能增强
  - ✓**主动应对变更**，基于对用户需求及其变化的理解，综合考虑软件各项功能实现的时间投入及开发成本，规划软件系统的整体演化，并以此开展功能增强等维护活动
  - ✓**持续性**，预先规划好软件演化的路线图，完成当前软件演化工作后，软件维护团队随后将连续性的进入到另一项软件演化工作
  - ✓**引发版本变化**，每一次演化结束后通常会产生一个新的软件版本

软件演化与软件维护

概念	功能增强粒度	应对变化方式	持续性或间隔性	版本变化
软件演化	粗粒度	主动	持续性	是
软件维护	细粒度	被动	间隔性	不一定

1.7 软件演化法则（1/2）

- **大型闭源软件的演化特点和规律**
- **持续变化法则**
  - ✓除非系统持续不断地被修改以满足用户的需求，否则系统将变得越来越不实用
- **增加复杂性法则**
  - ✓除非有额外的工作来明显降低软件系统的复杂性，否则软件系统会变得越来越复杂
- **自我调节法则**
  - ✓在软件演化过程中，软件产品和过程的测量遵循正态分布，演化过程是自我可调节的
- **组织稳定性守恒法则**
  - ✓在整个生命周期中，产生一个新版本所需的平均额外工作量几乎是相同的

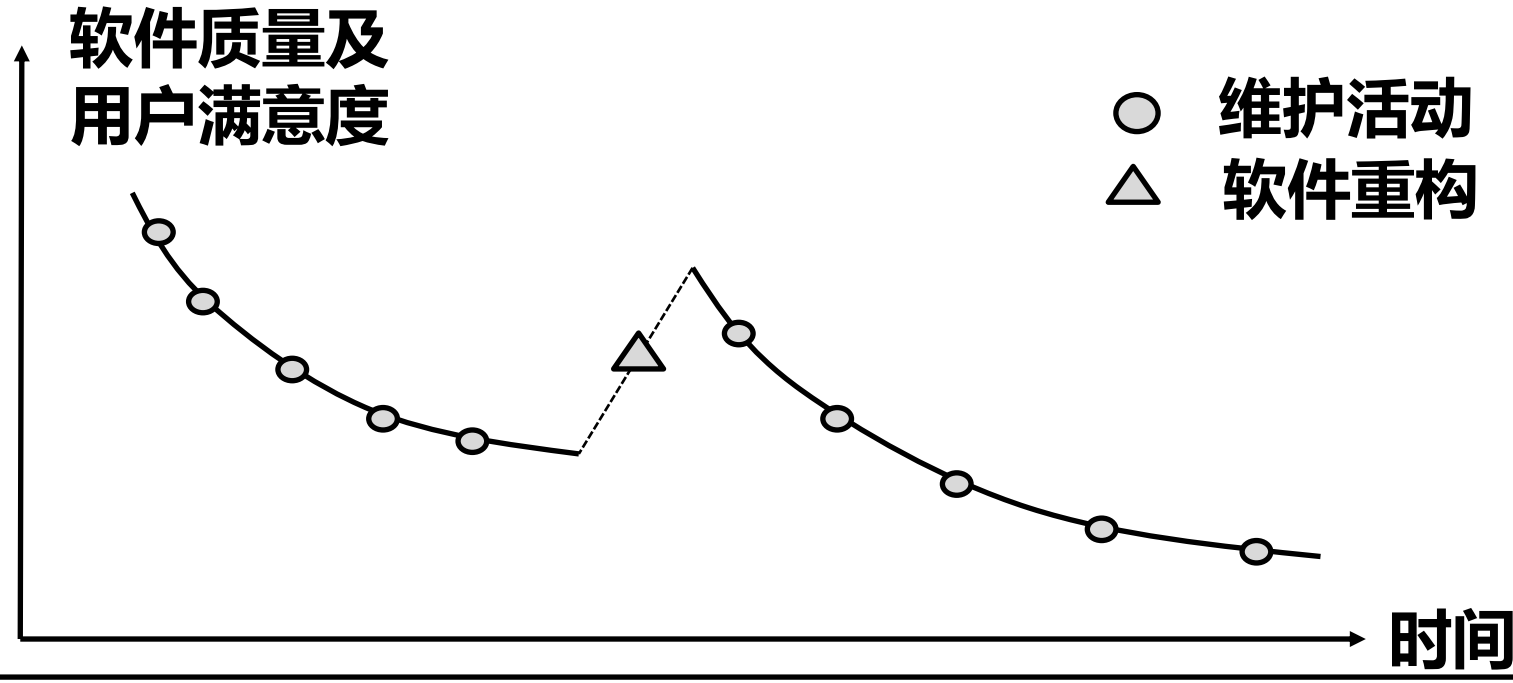
软件演化法则（2/2）

- **熟悉度守恒法则**
  - ✓在一个不断演化的系统中，平均增量增长几乎相同
- **功能持续增长法则**
  - ✓在软件的生命周期中，软件功能必须持续增加，否则用户的满意度会降低
- **质量衰减法则**
  - ✓软件的质量会随代码的不断变更而呈现出整体逐渐下降的趋势。如果没有严格的维护和适应性调整，使得软件适应运行环境的变化，软件的质量必然会随着软件演化而逐渐下降
- **反馈系统法则**
  - ✓系统的演化过程包括多回路的活动和多层次的反馈。软件工程师必须识别这些复杂的交互，以持续演化现有系统，提供更多的功能和更高的质量

目录

2.1 何为软件逻辑老化

- 软件在维护和演化的过程中出现的**用户满意度降低、质量逐渐下降、变更成本不断上升**这样一种现象
  - ✓这些现象发生在逻辑层面，而非发生在物理层面



软件会出现物理老化现象吗?

2.2 软件逻辑老化的现象（1/2）

❑质量下降

- ✓在对软件进行完善性维护的同时，尽管增加了新的功能，但也会**带来对软件架构的破坏**，从而**引入新的软件问题**，使得整个软件不易于维护，软件架构变得脆弱

❑变更成本增加

- ✓随着软件规模的不断增大、软件质量的下降，对软件进行变更的成本也会随之不断增加
- ✓**需要阅读更多的文档和代码**才能理解待维护的软件系统
- ✓软件系统架构的脆弱性意味着软件维护工程师不得不对软件进行**更多的“缝缝补补”**才能实现新的软件功能

软件逻辑老化的现象（2/2）

- **用户满意度降低**
  - ✓随着对软件认识不断深入，用户会逐步发现软件中存在的缺陷
  - ✓用户对软件的满意度会逐步降低
- **软件逻辑老化是一种必然的现象，不可避免**
- **如果对软件逻辑老化的现象置之不理，必然会导致软件“不可救药”，最终走向死亡**
- **解决软件逻辑老化的有效方法之一就是对软件进行重构，重构意味着给软件注入“强心针”，使得软件在一定程度上“返老还童”**

2.3 软件逻辑老化的原因

❑缺乏变更

- ✓当外部环境发生变化时，软件也应随之发生变化，进行必要的变更，否则软件就会进入老化
- ✓如果软件运行的操作系统已经发生了变化，软件就需要进行适应性的维护，否则就会被运行环境抛弃
- ✓**如果软件不能适应外部环境变化，缺乏必要变更，就会加速老化**

❑负面变更

- ✓软件变更不总是积极和正面的，有时候它会**带来负面和消极影响**
- ✓通过变更引入了新的、更为严重的缺陷；破坏了软件结构，使得软件架构更为脆弱
- ✓**负面变更会破坏软件的结构和质量，进而增加维护的成本和难度**

2.4 软件逻辑老化的表现

- **设计恶化是指软件维护过程中由于设计变更而导致的软件可变性显著降低的现象。**
  - ✓**设计僵化**，软件不易于变更，模块之间存在连带效应
  - ✓**设计脆弱**，一些“小规模”的软件变更会带来“大范围”的软件变更，甚至会破坏软件系统的整体架构
  - ✓**模块间紧耦合**，软件内部的多个模块之间关系过于密切，软件铁板一块，难以对其中的模块进行变更
  - ✓**晦涩的软件设计**，软件设计方案不易于理解
- **设计恶化会导致软件出错率上升，软件变更成本增高**

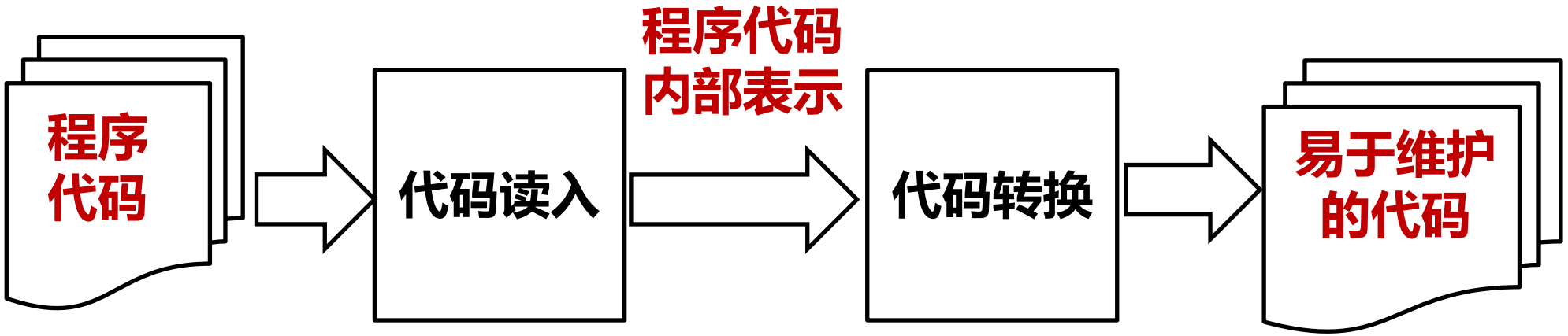
解决软件逻辑老化

- 维护：价值低，维护性好，采用积极的方式对软件进行有限的维护；
- 抛弃：如果价值低、维护性差，可逐步抛弃软件维护；
- 再工程：如果价值高，维护性弱，采取再工程的策略，对软件系统进行重构，提高软件的整体质量；
- 演化：价值高，维护性好，积极主动的演化，增强系统的功能；

目录

3.1 代码重组

- 在不改变软件功能的前提下，对程序**代码**进行**重新组织**，使得重组后的代码具有更好的**可维护性**，能够有效支持对代码的变更



3.2 逆向工程

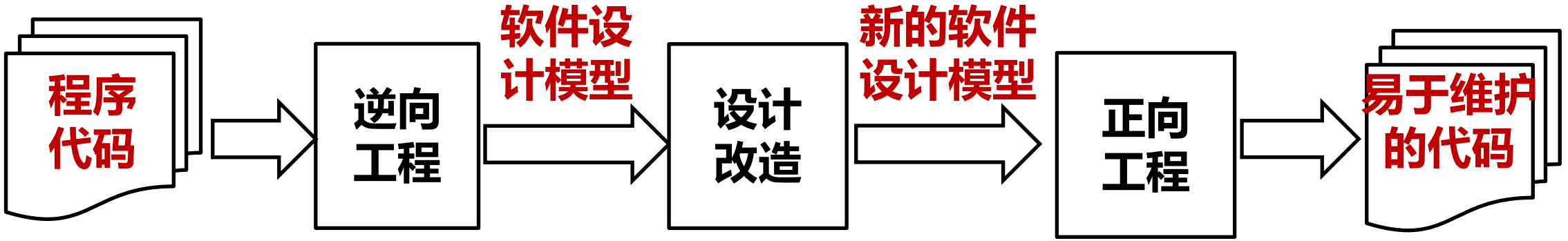
- **基于低抽象层次软件制品，通过对其进行理解和分析，产生高抽象层次的软件制品**
  - ✓通过对程序代码进行逆向的分析，产生与代码相一致的设计模型和文档
  - ✓基于对程序代码和设计模型的理解，逆向分析出软件系统的需求模型和文档
- **典型应用场景**
  - ✓分析已有程序，寻求比源代码更高层次的抽象形式（如设计甚至需求）

3.3 设计重构

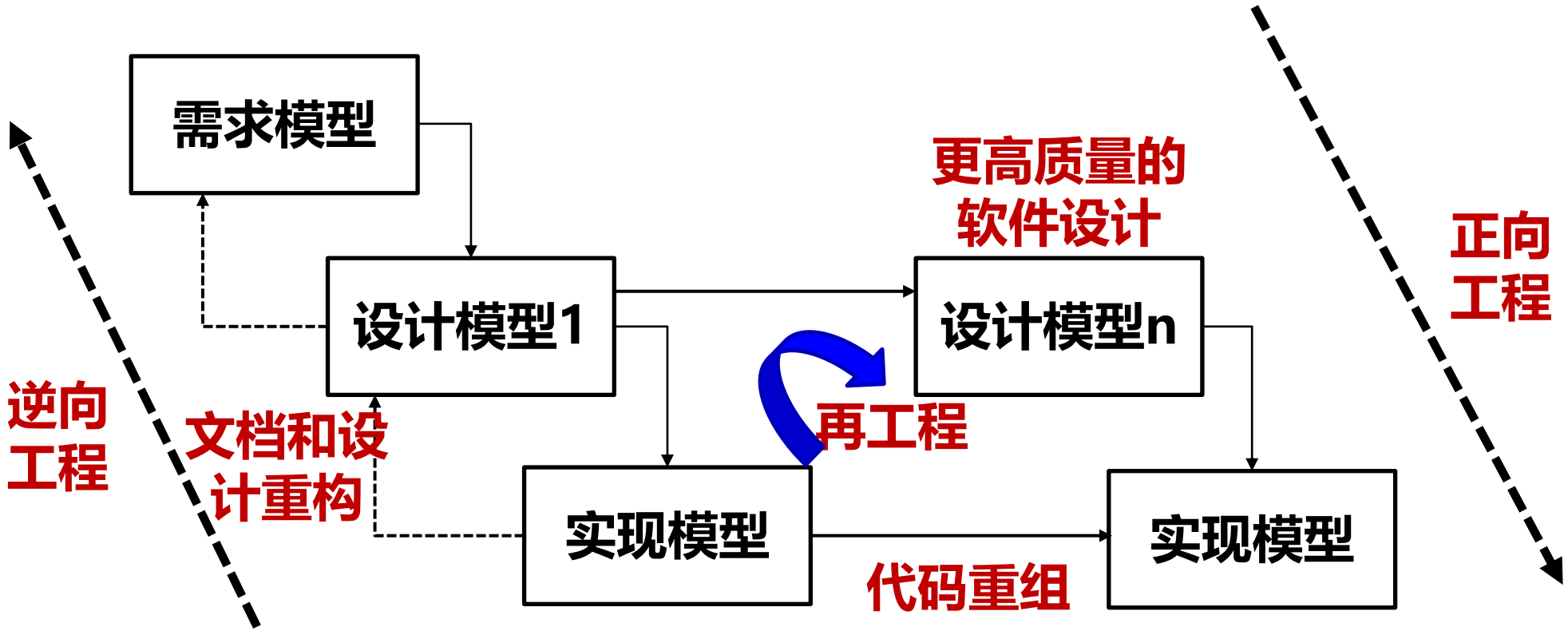
- 如果一个软件的设计文档缺失，软件文档与程序代码不一致、或者软件设计的内容不详实，那么软件维护工程师可以采用**设计重构的手段来获得软件设计方面的文档信息**
- 通过读入程序代码，理解和和分析代码中的变量使用、模块内部的封装、模块之间的调用或消息传递、程序的控制路径等方面的信息，**产生用自然语言或图形化信息所描述的软件设计文档**
- 是逆向工程的一种具体表现形式

3.4 再工程

- 通过分析和变更软件的架构，实现更高质量的软件系统的过程
- 再工程既包括逆向工程也包括正向工程

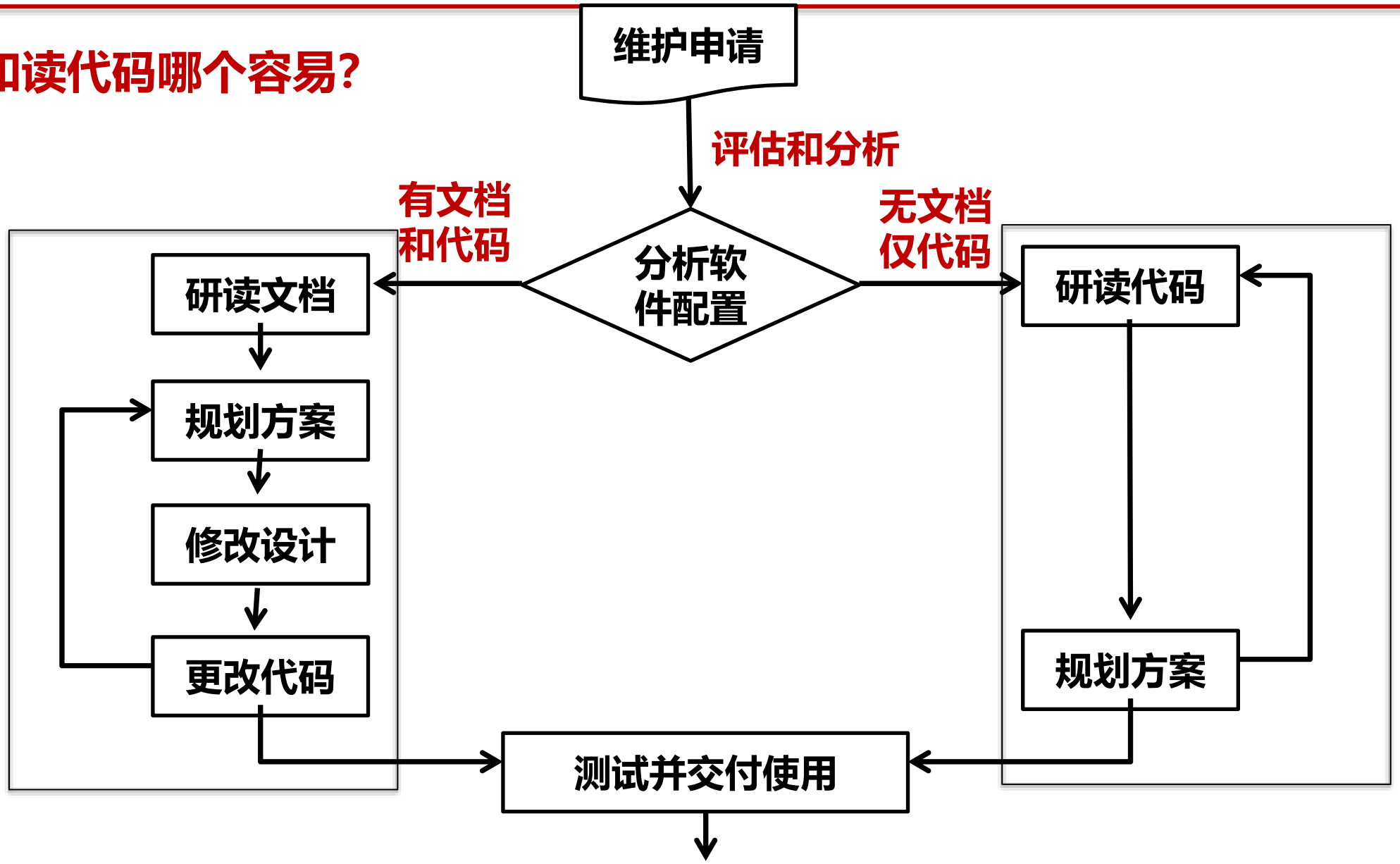


逆向工程、重组、重构和再工程示意图



3.5 软件维护过程

读文档和读代码哪个容易?



3.6 软件维护成本

- **维护成本不断增加**
  - ✓70年代： 35% - 40%
  - ✓80年代： 60%
  - ✓90年代： 75%
  - ✓如今：数据更高，80%

软件维护工作量 $M = P + K * e^{(c-d)}$

- ✓ **P=生产性工作量**
- ✓ **K=经验常数**
- ✓ **C=复杂度(设计好坏和文档完整程度)**
- ✓ **D=对欲维护软件的熟悉程度**

- **软件维护工作量涉及二方面**
  - ✓助动性：用于理解代码功能，结构特征以及性能约束
  - ✓生产性：用于分析和评价、修改设计和代码
- **模型表明**
  - ✓如果没有好的软件开发方法或者软件开发人员不能参与维护，那么软件维护工作量会指数上升

3.7 软件维护需要解决的问题（1/3）

❑人员的问题

- ✓软件维护工程师认为软件维护**缺乏成就感**，从而影响他们的工作激情和投入
- ✓软件维护工程师**得不到足够的关注和重视**，从而影响对他们的支持和帮助
- ✓软件开发工程师**流动大**，软件维护工程师无法得到软件开发工程师的帮助
- ✓软件开发工程师**不愿意帮助**软件维护工程师

如何解决软件维护的人员问题?

软件维护需要解决的问题（2/3）

❑软件制品的问题

- ✓待维护的软件不能提供软件文档
- ✓待维护的软件不能提供源程序代码
- ✓待维护软件的源代码可读性和可理解性差，如缺乏必要的注释等
- ✓待维护软件的文档可读性和可理解性差，如啰嗦、语言不简练等
- ✓待维护软件的文档不完整、不详实，漏掉重要内容，缺少细节
- ✓待维护软件的文档与其代码不一致，影响对软件的理解和维护
- ✓要读懂待维护软件的文档和代码非常困难
- ✓软件制品的版本混乱，无法获得合适版本的软件制品

为了有效支持维护，软件开发工程中应注意哪些事项?

软件维护需要解决的问题（3/3）

❑维护副作用的问题

- ✓**代码副作用**，如修改或者删除程序、修改或者删除语句标号、修改逻辑符号等等。为此，软件维护工程师在变更代码时要非常慎重，切忌随意的修改代码
- ✓**数据副作用**，因修改信息结构而带来的不良后果，如局部和全局数据的再定义，记录或者文件格式的再定义等
- ✓**文档和模型副作用**，软件维护工程师在对程序代码进行修改的同时，必须同步修改相关的模型和文档，以确保模型与代码之间、文档与模型之间、文档与代码之间相互一致

维护副作用问题会带来什么样的后果?

3.8 软件的可维护性

- **在软件开发时就要考虑到将来的维护问题**
  - ✓要有前瞻性，预测到变化和问题
- **设计和实现出具有可维护性的软件**
  - ✓软件被理解、改正、调整和改进的程度
- **软件的质量直接决定了软件是否易于维护**

在开发软件时就需要考虑到未来的软件维护问题

影响软件可维护性的因素

- 软件开发方法(结构化、OO)
- 软件文档是否齐全
- 文档结构是否标准化
- 软件是否易于扩展
- 软件结构是否清晰易于理解
- 是否采用标准的程序设计语言
- 程序代码是否易于理解

必须在设计时考虑软件可维护性问题

3.9 保证软件可维护性的复审

- **需求分析的复审**
  - ✓对将来可能修改和改进的部分加注释，对软件的可移植性加以讨论，并考虑可能影响软件维护的系统界面
- **设计阶段的复审**
  - ✓从易于维护和提高设计总体质量的角度全面评审数据设计、总体结构设计、过程设计和人机界面设计
- **编码阶段的复审**
  - ✓强调编码风格和内部文档
- **阶段性测试**
  - ✓必要的预防性维护
- **软件维护活动完成之际也要进行复审**

在开发软件早期阶段的质量保证中考虑到可维护性问题

小结

- **多样化的软件维护**
  - ✓纠正性、完善性、适应性、预防性维护
- **持续性的软件演化**
  - ✓对软件的大规模功能增强和结构调整，以实现变化的软件需求，或者提高软件系统的质量
- **软件逻辑老化**
  - ✓解决软件逻辑老化的有效方法之一就是对软件进行重构
- **软件维护技术**
  - ✓逆向工程、再工程、代码重组、设计重构
  - ✓软件维护会带来新问题，同样需要关注质量因素，进行必要质量保证