
中国石油大学(华东)
CHINA UNIVERSITY OF PETROLEUM
软件工程

主要内容

第一章 软件工程学概述
第二章 可行性研究
第三章 需求分析
第四章 总体设计
第五章 详细设计
第六章 编码与测试
第七章 软件维护
第八章 面向对象方法学
第九章 面向对象分析设计与实现
第十章 软件项目管理

第七章　软件维护

第一节　软件维护的类型

第二节　软件维护的特点

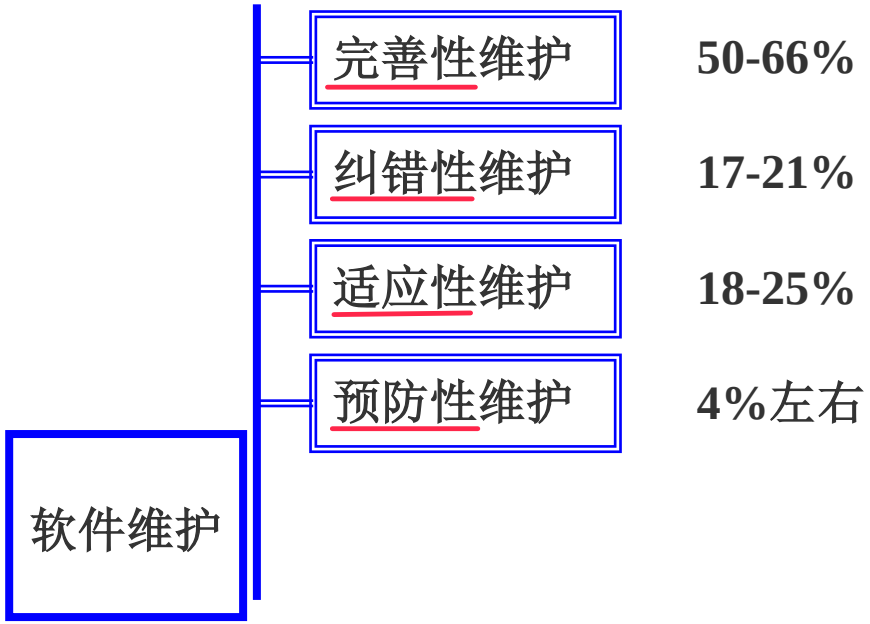
第三节　软件维护过程

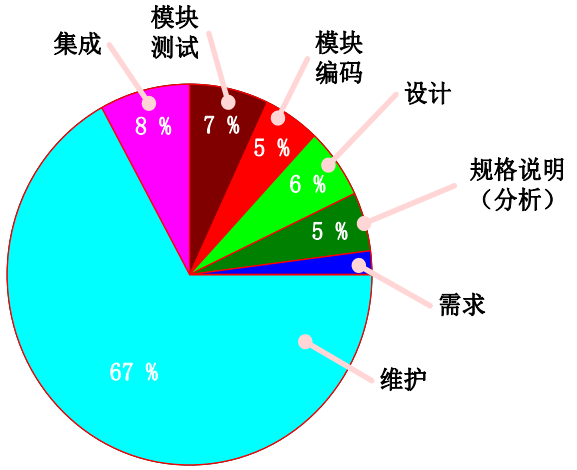
第四节　软件的可维护性

第五节　软件再工程

第七章　软件维护

- 软件维护是指软件系统交付使用以后，为了改正错误或满足新的需求而修改软件的过程。
- 按照不同的维护目的，维护工作可分成4类，如下图。
- 按国外统计数字，各类维护占全部维护活动的比例如图中所示。





软件生命周期各个阶段的相对近似花费

第七章　软件维护

第一节　软件维护的类型

1. 完善性维护（Perfective Maintenance）

扩充原有系统的功能，提高原有系统的性能，满足用户的实际需要。

2. 纠错性维护（Corrective Maintenance）

对在测试阶段未能发现的，在软件投入使用后才逐渐暴露出来的错误的测试、诊断、定位、纠错以及验证、修改的回归测试过程。

3. 适应性维护（Adaptive Maintenance）

要使运行的软件能适应运行环境的变动而修改软件的过程。

4. 预防性维护（Preventive Maintenance）

- 为了进一步改善软件的可靠性和易维护性，或者为将来的维护奠定更好的基础而对软件进行修改。
- 预防性维护对象：
  - ✓ 预计若干年内将继续使用的程序
  - ✓ 当今正成功使用的程序
  - ✓ 最近的将来要进行大修改和完善的程序

注：① 一般维护的工作量占生存周期70%左右，维护成本约为开发成本的4倍；

② 文档维护与代码维护同样重要。

第二节　软件维护的特点

1. 结构化维护与非结构化维护差别大

- 结构化维护 — 指软件开发过程是按照软件工程方法，软件的维护过程，有一整套完整的方案、技术、审定过程。
- 非结构化维护 — 缺乏必要的文档说明，难于确定数据结构、系统接口等特性。维护工作令人生畏，事倍功半。

2. 软件维护的代价高昂

- 维护费用高，而且逐年上涨；维护中还可能引入新的潜在错误。

3. 软件维护的问题多

- 理解别人的程序很困难
- 文档与代码不一致
- 开发人员往往不参加维护
- 大多数软件在设计时没有考虑将来的修改
- 维护工作不是一项吸引人的工作

软件工程的思想至少部分地解决了与维护有关的每一个问题。

第三节　软件维护过程

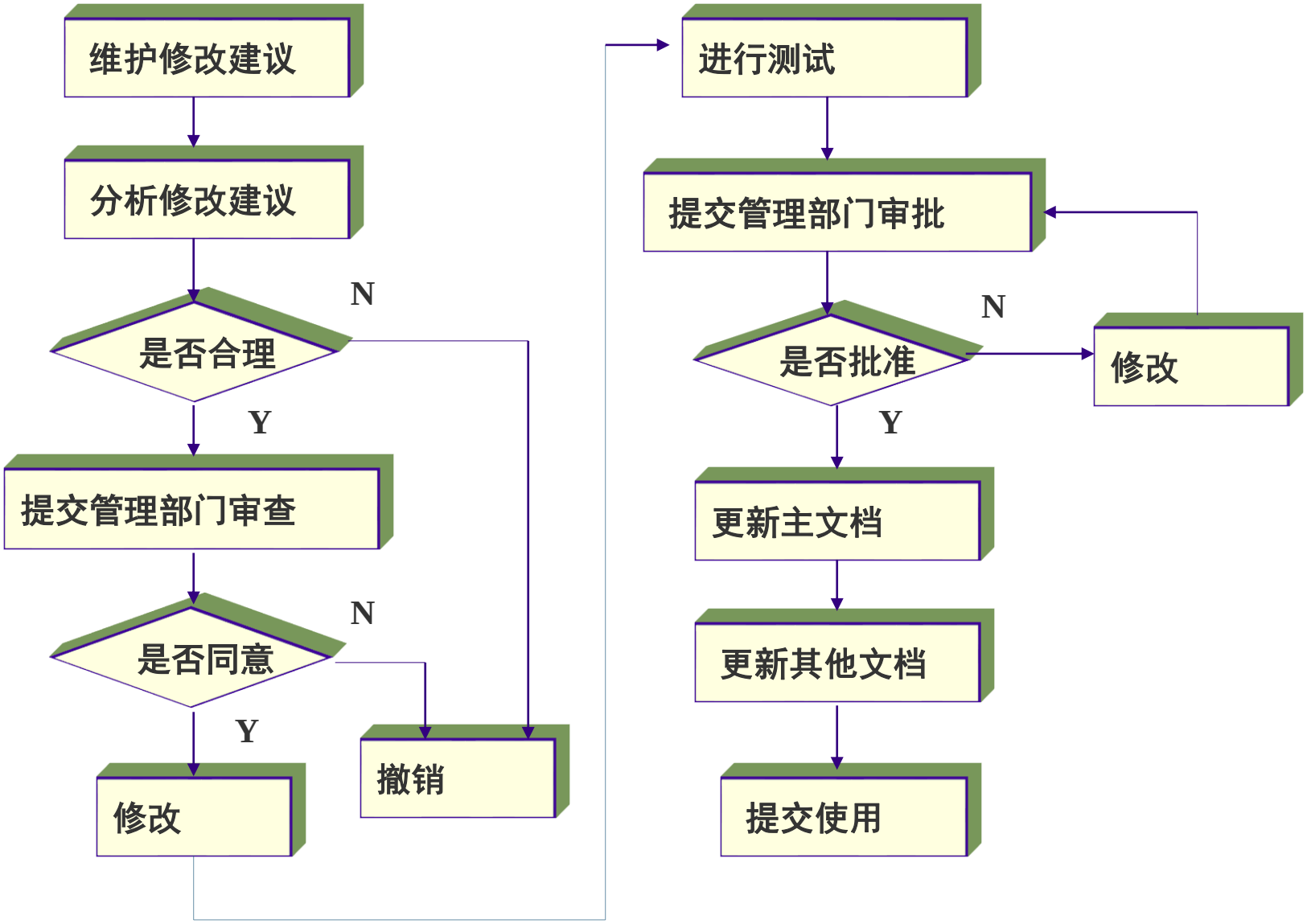
软件维护过程本质上是修改和压缩了的软件定义和开发过程。

1. 软件维护组织

软件维护工作不仅是技术性的，它还需要大量的管理工作与之相配合，才能保证维护工作的质量。管理部门应对提交的修改方案进行分析和审查，并对修改带来的影响作充分的估计，对于不妥的修改予以撤销。需修改主文档时，管理部门更应仔细审查。

软件维护的管理流程如图所示：

第七章　软件维护


维护修改建议
分析修改建议
是否合理
N
Y
提交管理部门审查
是否同意
N
Y
修改
撤销
进行测试
提交管理部门审批
是否批准
N
Y
修改
更新主文档
更新其他文档
提交使用

2. 维护报告

(1) 维护申请报告

由用户填写的外部文件，提供错误情况说明或修改说明书等。

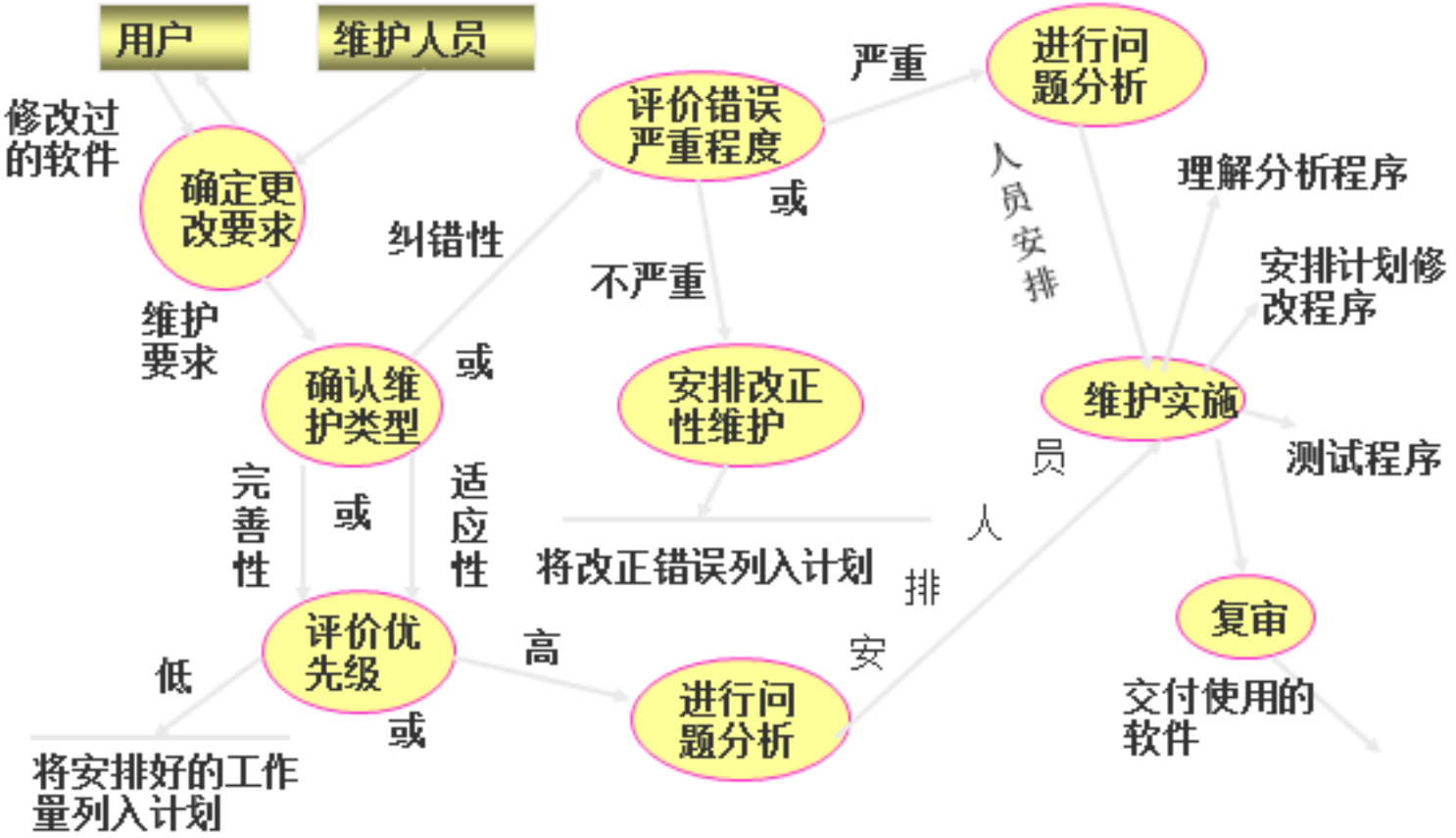
(2) 软件修改报告

与维护申请报告相应的内部文件，要求说明：

- 所需修改变动的性质；
- 申请修改的优先级；
- 为满足某个维护申请报告，所需的工作量；
- 预计修改后的状况。

3. 软件维护工作流程

4. 保存维护记录



第四节　软件的可维护性

软件的可维护性是指维护人员理解、改正、改动或改进这个软件的难易程度。

一、决定软件可维护性的因素

(1) 可理解性

- 是指由文档代码理解功能运行的容易程度。
- 好程序的特征：模块化、结构化、代码与设计风格一致。

(2) 可测试性

- 是指诊断和测试的容易程度，可用程序复杂性度量。
- 好程序的特征：可理解、可靠、简单。

(3) 可修改性

- 是指程序容易修改的程度。
- 好程序的特征：可理解、简单、通用。

(4) 可移植性

- 是指程序被移到一个新环境的容易程度。
- 好程序的特征：结构好，不特别依赖于某一具体的计算机或操作系统。

(5) 可重用性

- 重用是指同一事物不做修改或稍加改动就在不同环境中多次使用。软件采用可重用的软件构件越多，则可靠性高、可移植性越好。

二、文档

文档是影响可维护性的决定因素，对于大型软件系统在使用过程中必然会经受多次修改，所以文档比代码更重要。

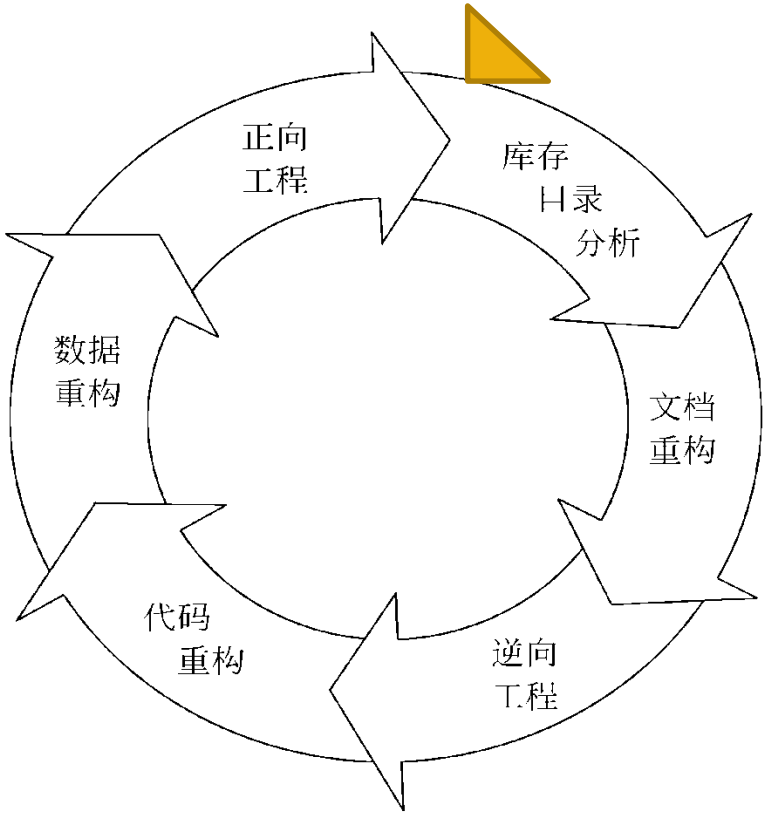
软件系统的文档可分为用户文档和系统文档两类：

- 用户文档主要描述系统功能和使用方法，并不关心这些功能是怎样实现的。
- 系统文档是指从问题定义、需求说明到验收测试计划，这样一系列与系统实现有关的文档。描述系统设计、实现和测试等方面的文档对于理解程序和维护程序来说是很重要的。

三、提高可维护性的方法

- 建立明确的软件质量目标和优先级
- 使用提高软件质量的技术和工具
- 进行明确的质量保证审查
- 选择可维护的程序设计语言
- 改进程序的文档

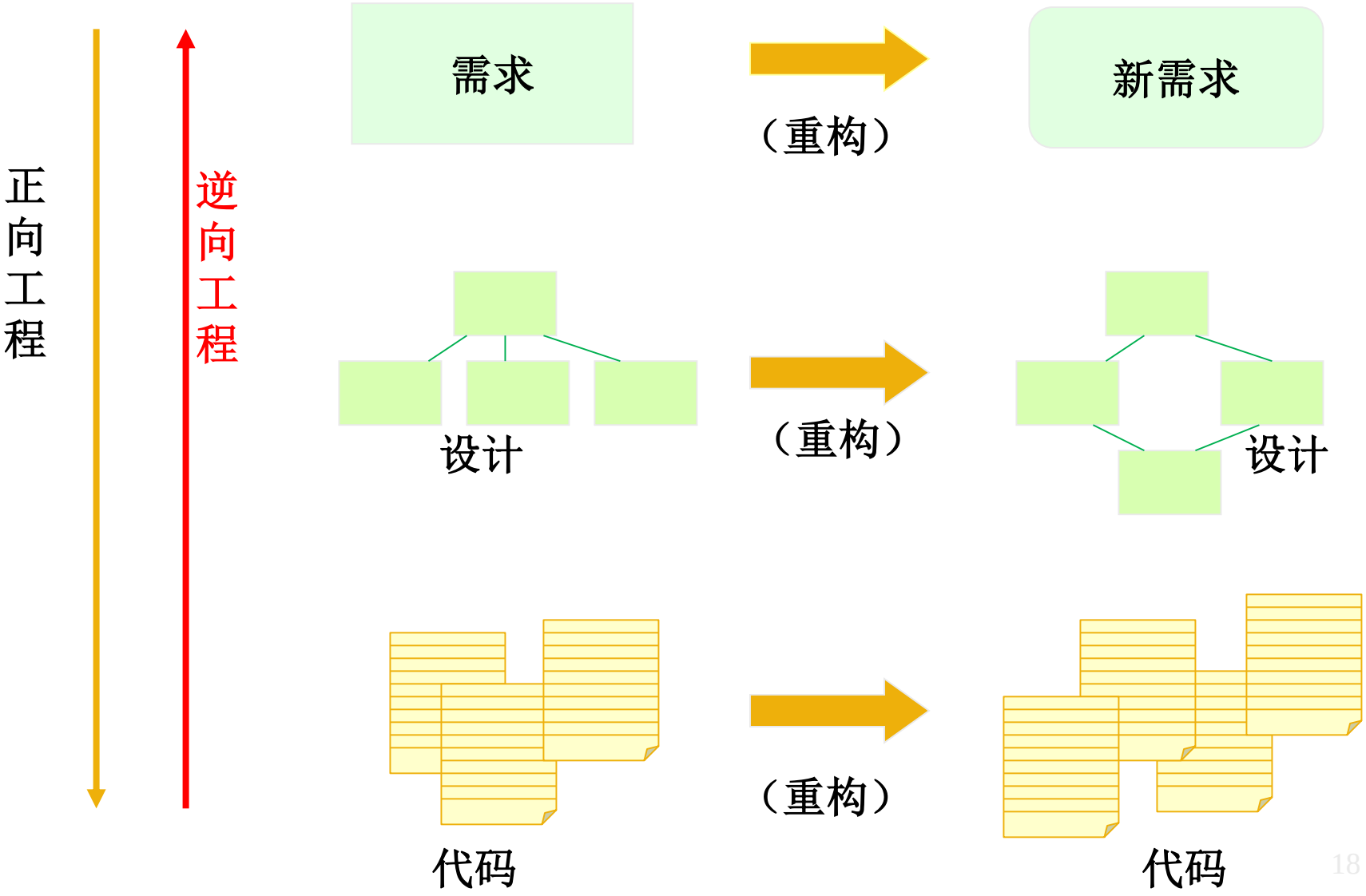
第五节　软件再工程



软件再工程过程模型

软件再工程是一类软件工程活动，是一个工程过程，它将逆向工程、重构和正向工程组合起来，将现存系统重新构造为新的形式。

软件再工程过程示意图



软件再工程过程

- 典型的软件再工程过程模型如上图所示，该模型定义了6类活动。
- 再工程范型是一个循环模型。这意味着作为该范型的组成部分的每个活动都可能被重复，而且对于任意一个特定的循环来说，过程可以在完成任意一个活动之后终止。

7.5 软件再工程过程

1. 库存目录分析

- 每个软件组织都应该保存其拥有的所有应用系统的库存目录。该目录包含关于每个应用系统的基本信息。
- 对库中每个程序都做逆向工程或再工程是不现实的。下述3类程序有可能成为预防性维护的对象：

（1）预定将使用多年的程序；

（2）当前正在成功地使用着的程序；

（3）在最近的将来可能要做重大修改或增强的程序。

7.5 软件再工程过程

- 应该仔细分析库存目录，按照业务重要程度、寿命、当前可维护性、预期的修改次数等标准，把库中的应用系统排序，从中选出再工程的候选者，然后明智地分配再工程所需要的资源。

7.5 软件再工程过程

2. 文档重构

- 老程序固有的特点是缺乏文档。具体情况不同，处理这个问题的方法也不同：

➢ 软件稳定或者生命周期临近结束，保持现状

➢ 首先建立起正在修改部分的文档

➢ 如果某应用系统是完成业务工作的关键，而且必须重构全部文档，则仍然应该设法把文档工作减少到必需的最小量。

7.5 软件再工程过程

3. 逆向工程

- 软件的逆向工程是分析程序以便在比源代码更高的抽象层次上创建出程序的某种表示的过程。
- 逆向工程是一个恢复设计结果的过程，逆向工程工具从现存的程序代码中抽取有关数据、体系结构和处理过程的设计信息。

7.5 软件再工程过程

4. 代码重构

- 重构难于理解、测试和维护的模块的代码。
- 为了完成代码重构活动，首先用重构工具分析源代码，标注出和结构化程序设计概念相违背的部分。然后重构有问题的代码。最后，复审和测试生成的重构代码并更新代码文档。
- 代码重构并不修改整体的程序体系结构，它仅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。
- 如果重构扩展到模块边界之外并涉及软件体系结构，则重构变成了正向工程。

7.5 软件再工程过程

5. 数据重构

- 数据重构是一种全范围的再工程活动。
- 数据重构始于逆向工程活动，分解当前使用的数据体系结构，必要时定义数据模型，标识数据对象和属性，并从软件质量的角度复审现存的数据结构。
- 当数据结构较差时，应该对数据进行再工程。
- 由于数据体系结构对程序体系结构及程序中的算法有很大影响，对数据的修改必然会导致体系结构或代码层的改变。

7.5 软件再工程过程

6. 正向工程

- 从现有程序中恢复设计信息，而且使用该信息去改变或重构现有系统，以提高其整体质量。
- 正向工程过程应用软件工程的原理、概念、技术和方法来重新开发某个现有的应用系统。在大多数情况下，被再工程的软件不仅重新实现现有系统的功能，而且加入了新功能和提高了整体性能。


Thank
You