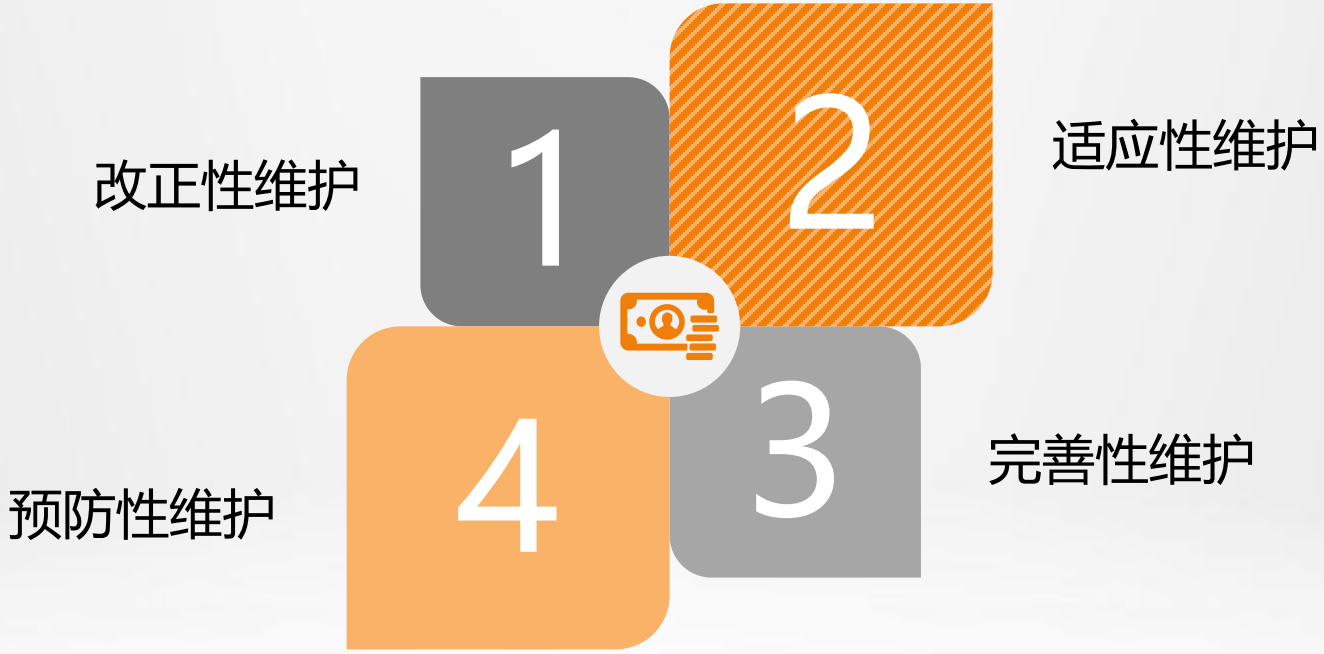
第 7 章　软件工程概述

本章内容

7.1.1 软件维护的种类

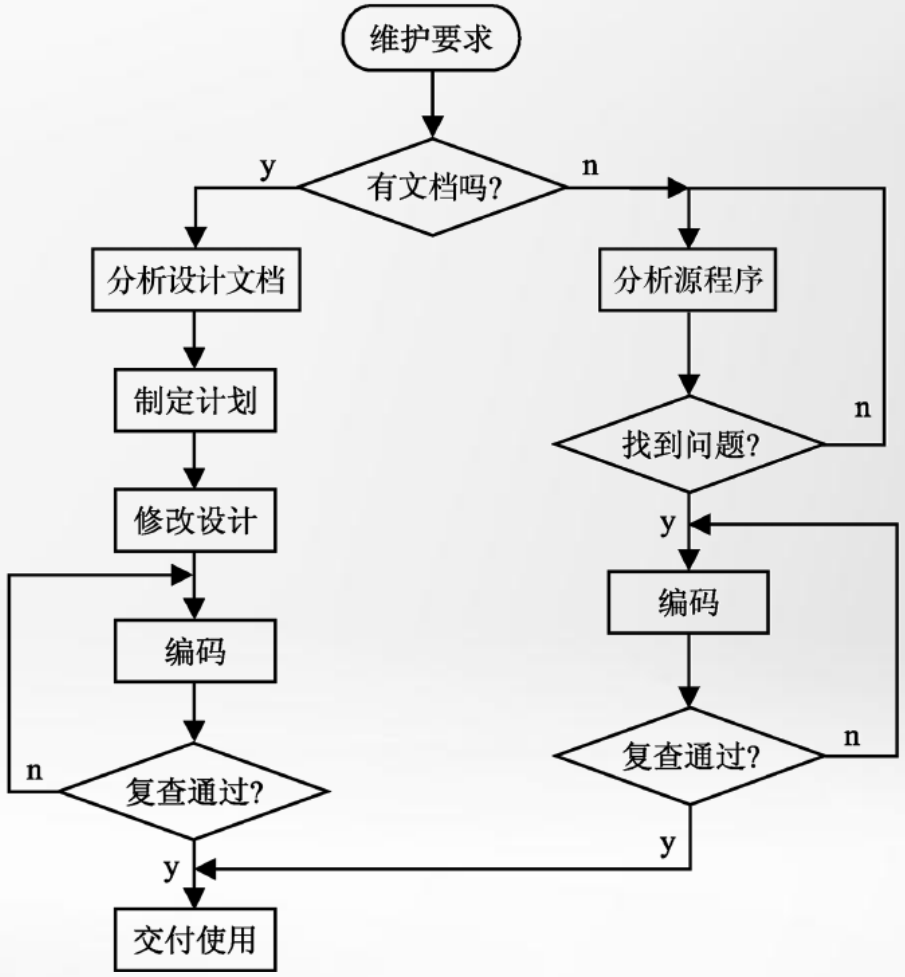
软件维护分为改正性维护、适应性维护、完善性维护和预防性维护4 种。



7.1.2 软件维护的困难

01 OPTION 非结构化维护与结构化维护

不采用软件工程方法开发的软件，只有程序没有文档，维护工作很难进行，称为非结构化维护。采用软件工程方法开发的软件，每个阶段都有文档，容易进行各种维护，称为结构化维护。



7.1.2 软件维护的困难

02
OPTION

软件过程中不考虑维护问题造成软件维护困难

理解他人编写的程序往往是非常困难的

软件开发人员经常流动，因而当需要维护时，往往无法依赖开发人员本人来对软件进行解释和说明

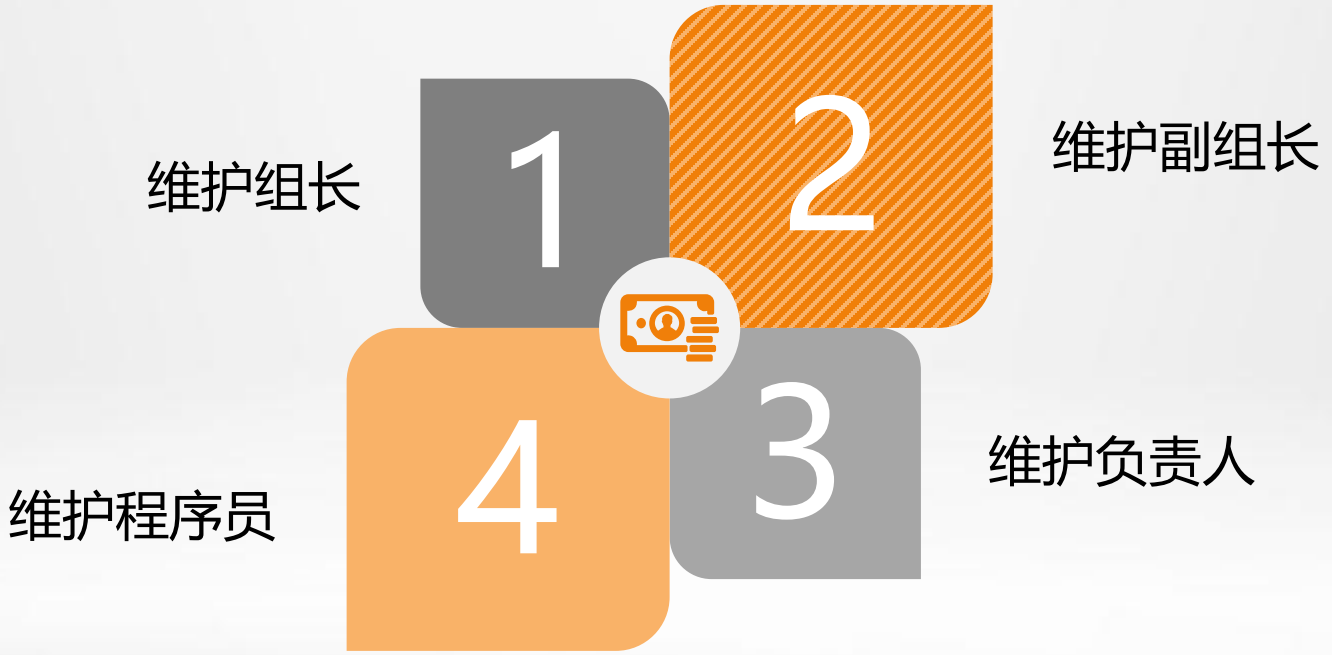
需要维护的软件往往没有足够的、合格的文档

绝大多数软件在设计时并不会充分考虑到以后修改的便利问题

7.1.3 软件维护的实施

01 OPTION 维护组织

维护组织通常以维护小组的形式出现，维护小组分为非长期维护小组和长期维护小组。长期维护小组由以下人员组成。



7.1.3 软件维护的实施

02 OPTION 维护文档

1 维护要求表

- 软件维护人员应当向用户提供空白的维护要求表，由要求维护的用户填写。
- 该表应能完整描述软件产生错误的情况（包括输入数据、输出数据及其他有关信息）

2 软件修改报告

- 按维护要求表进行维护所需要的工作量。
- 维护要求的性质。
- 该项要求与其他维护要求相比的优先程度。
- 预计修改后的状况

7.1.3 软件维护的实施

03 OPTION 维护的流程

确定维护的类型

- 维护工作首先要根据维护要求表确定维护属于哪种类型。如果属于改正性维护，则需评价其出错的严重性。
- 如果错误严重，就进一步指定人员，在系统管理员的指导配合下，分析错误的原因，进行维护。
- 对不太严重的错误，则该项改正性维护可与其他软件开发的任务一起统筹安排。
- 如果属于完善性或适应性维护，则先确定各个维护要求的优先次序，并且安排所需工作时间。
- 从其意图和目标来看，属于开发工作，因此可将其视同开发任务。
- 如果某项维护要求的优先级特别高，可立即开始维护工作。

7.1.3 软件维护的实施

维护记录的保存

- 维护要求表的标识、维护类型。
- 程序名称。
- 所用的编程语言。
- 程序语句数或机器指令条数。
- 程序开始使用的日期。
- 已运行次数、故障处理次数。
- 程序改变的级别及名称。
- 修改程序所增加的源语句数、所删除的源语句数。
- 各次修改耗费的人数× 时数。
- 软件工程师的姓名。
- 维护开始和结束的日期。
- 累计用于维护的人数× 时数。
- 维护工作的净收益。

7.1.3 软件维护的实施

维护的复审

- 设计、编码、测试工作的完成情况。
- 维护资源的使用情况。
- 维护的主要障碍和次要障碍。

7.1.4 软件维护的实施

1 编码副作用

使用程序设计语言修改源程序时可能引入错误

2 数据副作用

修改数据结构时可能造成软件设计与数据结构不匹配，因而导致软件错误

3 文档副作用

对数据流、软件结构、模块逻辑或任何其他特性进行修改时，必须对相关的文档进行相应修改，否则会导致文档与程序功能不匹配，文档不能反映软件当前的状态

本章内容

7.1 软件维护过程

7.2 软件的可维护性

7.2.1 影响可维护性的因素

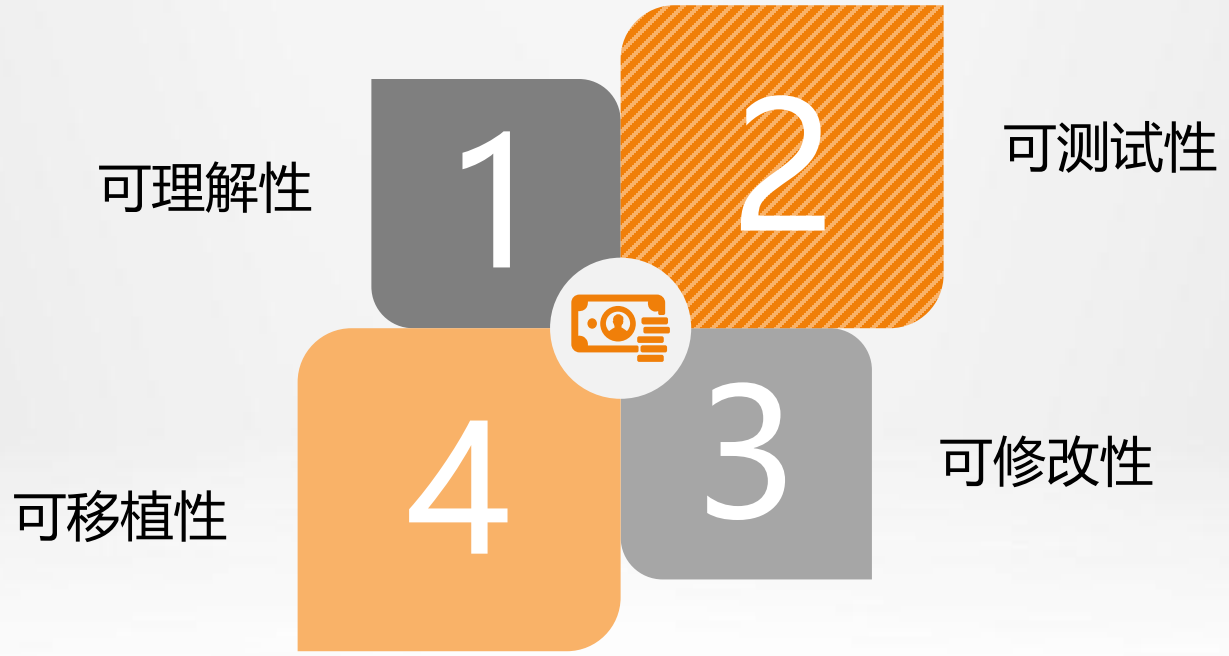
- 是否拥有一组训练有素的软件开发人员。
- 系统结构是否可理解，是否合理。
- 文档结构是否标准化。
- 测试用例是否合适。
- 是否已有嵌入系统的调试工具。
- 所选用的程序设计语言是否合适。
- 所选用的操作系统等是否合适。

7.2.2 可维护性的度量

- 识别问题的时间。
- 修改需求规格说明书的时间。
- 分析、诊断问题的时间。
- 选择维护工具的时间。
- 纠错或修改软件的时间。
- 测试软件的时间。
- 维护复审的时间。
- 软件恢复运行的时间。

7.2.2 可维护性的度量

软件的可维护性主要表现在以下方面。



7.2.3 提高软件的可维护性

可移植性就是指软件不加改动地从一种运行环境转移到另一种运行环境下的运行能力，也即程序在不同计算机环境下能够有效地运行的程度。