第10章 软件维护

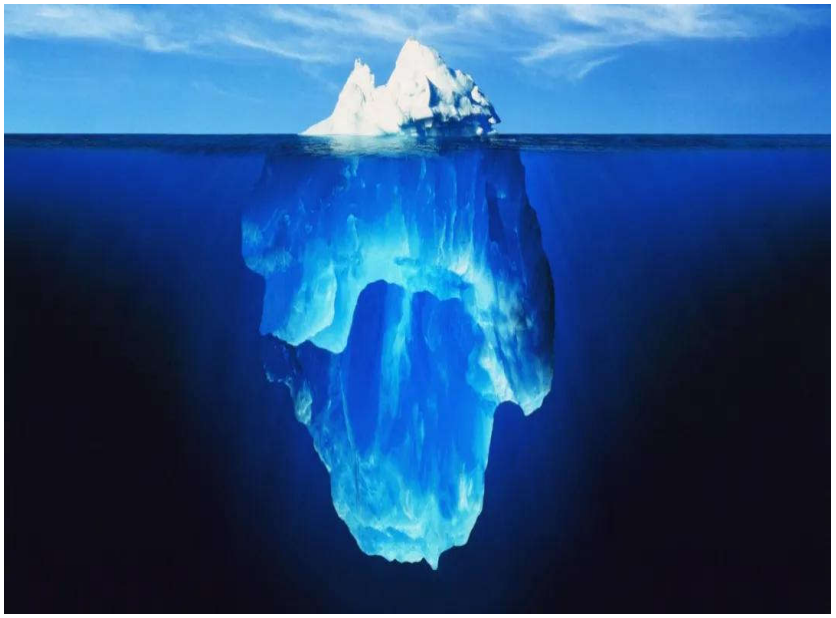
- 软件维护概述
- 软件维护过程
- 软件的可维护性

软件维护概述

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。

软件维护工作处于软件生命期的最后阶段，维护阶段是软件生存期中最长的一个阶段，其费用高达整个软件生命期花费的约60%-70%。

- 需求
- 设计
- 实现
- 测试
- 维护



软件维护概述

软件维护的任务

(1) 在软件系统运行过程中发现测试阶段未能发现的、潜在的软件错误和缺陷。

(2) 随着软硬件环境的改变，与系统交互的外部系统的改变，网络通信技术的发展，系统数据或文件格式、存储方式、读取步骤的变迁，要求软件系统适应这些变化。

(3) 根据实际情况的发展，用户操作、流程发生改变，需要改进软件设计，增强软件功能，提高软件性能。

(4) 不断扩大软件系统的应用范围。

软件维护的分类

按照不同的维护目的，维护工作可分成4类。

- 完善性维护（Perfective Maintenance）
- 纠错性维护（C…ve Maintenance）
- 适应性维
- 预防性维

扩充原有系统的功能，提高系统的性能，提高软件运行的效率，满足用户的实际需要而进行的维护活动。

当一个软件系统投入使用和成功地运行时，用户会根据业务发展的实际需要，提出增加新功能、修改已有功能以及性能的改进要求等。

软件维护的分类

按照不同的维护目的，维护工作可分成4类。

- 完善性维护（Perfective Maintenance）
- 纠错性维护（Corrective Maintenance）
- 适应性维护（Adapt…nce）
- 预防性维…

软件测试不可能找出一个软件系统中所有潜在的错误，所以当软件在特定情况下运行时，这些潜伏的错误可能会暴露出来。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，称为纠错性维护。

软件维护的分类

按照不同的维护目的，维护工作可分成4类。

- 完善性维护（Perfectiv
- 纠错性维护（Correctiv
- 适应性维护（Adaptive Maintenance）
- 预防性维护（Preventive Maintenance）

计算机的软、硬件环境，数据环境在不断的变化，使运行的软件能适应运行环境或者数据的变动而修改软件的过程称为适应性维护。

软件维护的分类

按照不同的维护目的，维护工作可分成4类。

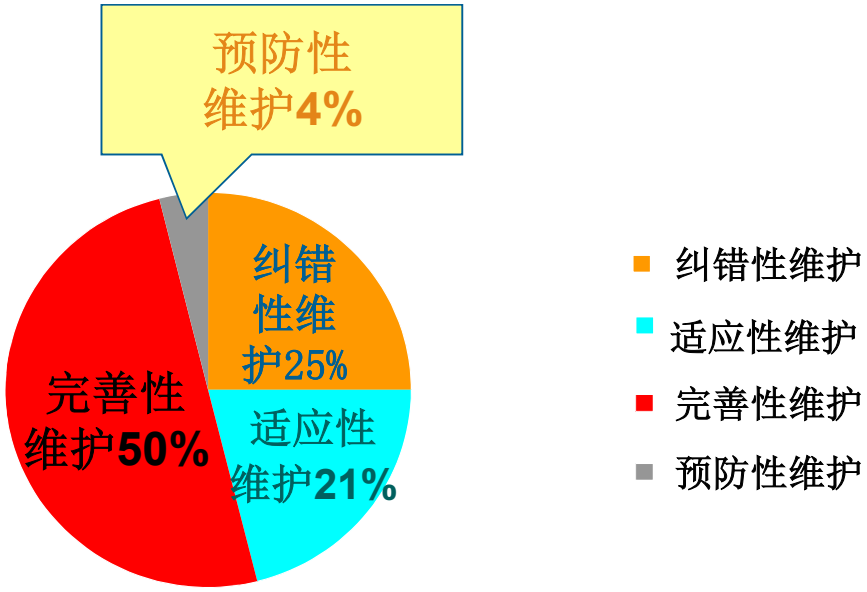
- 完善性维护（Pe
- 纠错性维护（Co
- 适应性维护（Adaptiv　　nance）
- 预防性维护（Preventive Maintenance）

为了进一步改善软件的可靠性和易维护性，或者为预见将来软件运行和维护打下更好的基础而对软件进行修改。

由于对于该类维护工作必须采用先进的软件工程方法，对需要修改的软件或部分进行设计、编码和测试。对该类维护工作的必要性有争议，它所占的比例较小。

软件维护的分类

针对不同类型的维护，可采取相应的维护策略，以提高维护效率，降低维护成本。



各类维护所占的比例

软件维护过程

软件维护管理的基本内容——概述

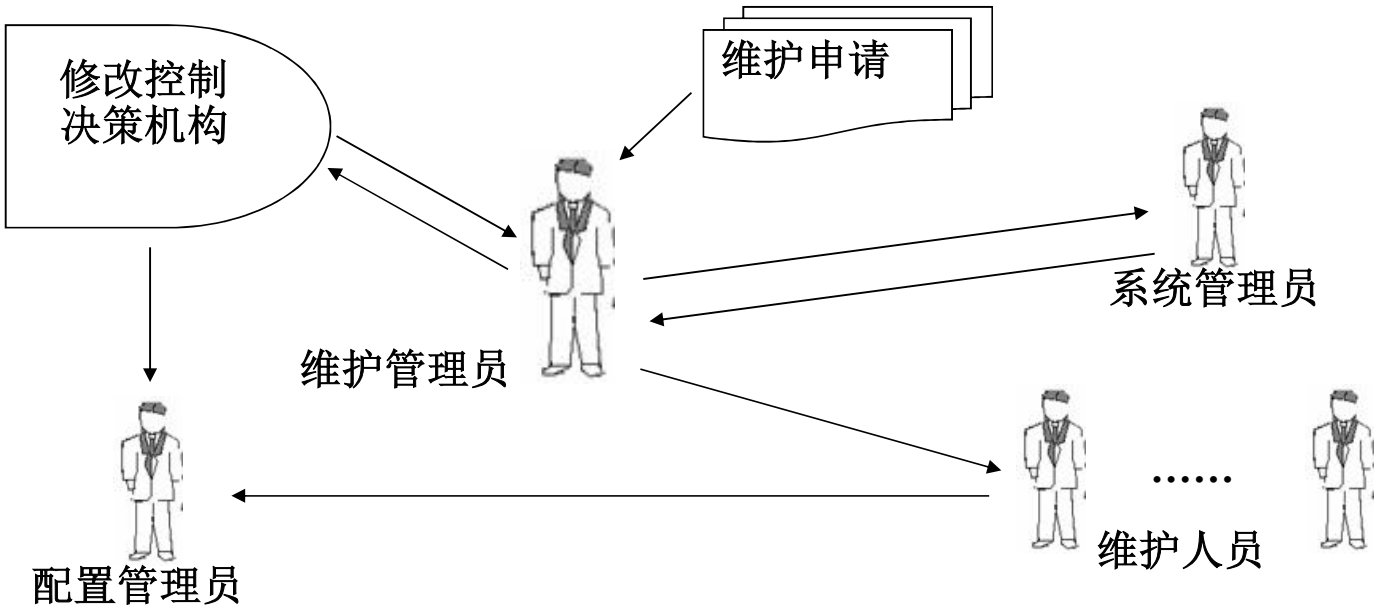
软件维护工作是一个完整的软件开发过程，因而必须纳入有效的管理中，才能保证维护工作的有效性和正确性，确保维护后系统的稳定性。

软件维护管理的内容主要涵盖以下几个方面：

- 维护的组织结构
- 维护的申请报告
- 维护工作流
- 维护成本
- 维护评审

软件维护过程

1. 软件维护管理的基本内容——维护的组织结构



2. 软件维护管理的基本内容——维护申请报告

所有的维护在开始前都要提交维护申请报告。正式申请报告通常会采用结构化的采用维护申请表（**Maintenance Request Form，MRF**）。当遇见系统问题时，完整地记录出现问题的详细描述，包括现场的输入/输出、文件、提示信息、问题表现情况等有关信息，并同时提交一个简短的修改说明书，提出希望完成的修改内容。

维护管理员和系统管理员就下列各项范围分析**MRF**，并就修改、请求对组织、现行系统和接口系统的影响做出评估。

⑴ 类型：例如：纠正、改进、预防或对新环境的适应；

⑵ 范围：例如：修改规模、涉及的费用、修改时机；

⑶ 关键性：例如，对性能、安全、保密的影响。

软件维护过程——软件维护申请报告

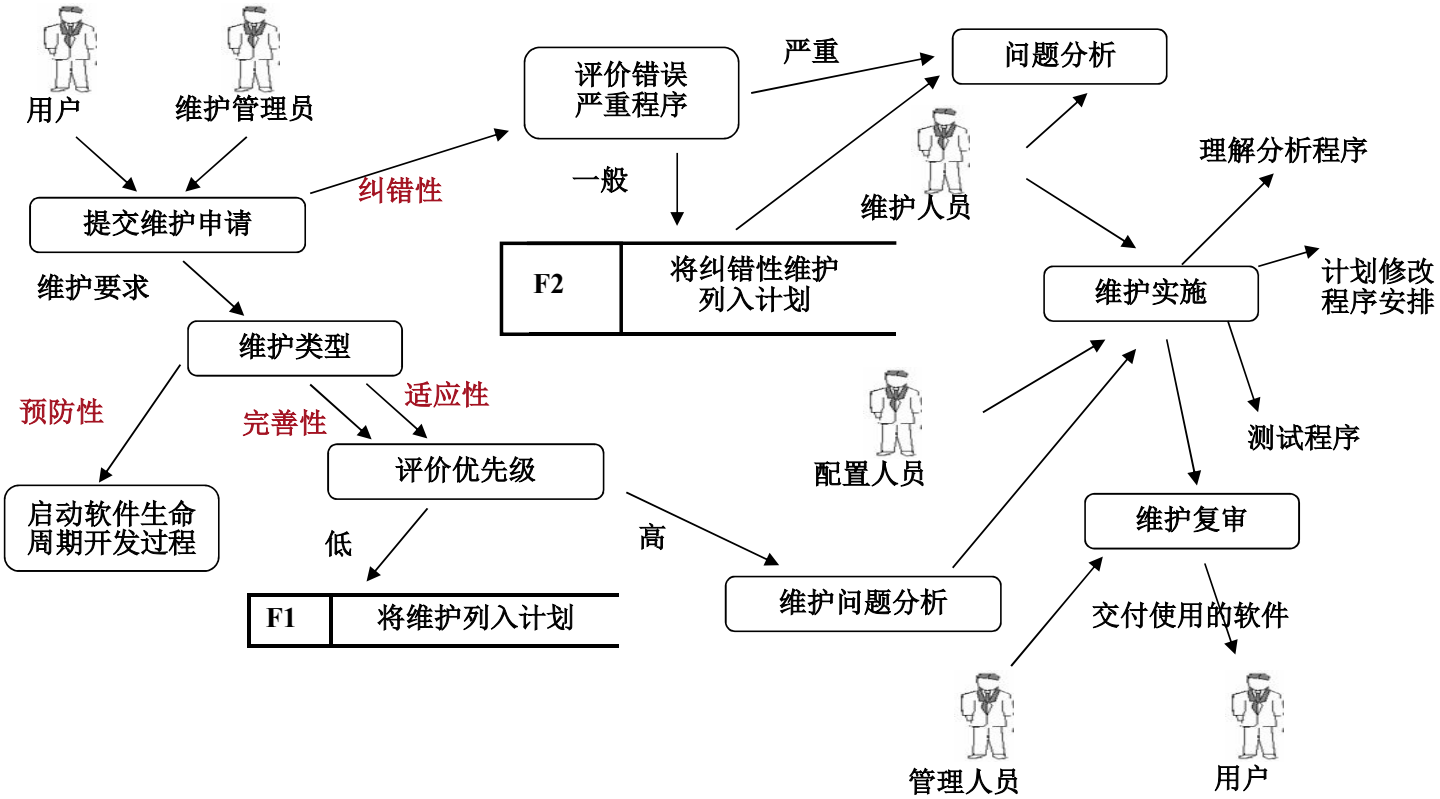
<table>
<tr><td>项目名称</td><td>网络测评系统</td><td>项目编号</td><td>××××××××××</td></tr>
<tr><td colspan="2" rowspan="3">问题说明：人员类型数据错误。
现象：不同类型的人员均可进行交叉测评。
实际需求：各类人员只进行自身类型的测评，如管理人员只能对管理人员进行评测，教师只能评测教师。</td><td colspan="2">预计评测结果：修正程序中的人员权限，使得每种类型的人员只能进行自身类型的评测。</td></tr>
<tr><td>维护安排</td><td>[　] 远程维护
[√] 现场维护</td></tr>
<tr><td>维护类型</td><td>软件：[√] 纠错性维护
[　] 适应性维护
[　] 完善性维护
硬件：[√] 系统设备
[　] 外部设备</td></tr>
<tr><td colspan="2" rowspan="2">维护要求及优先级：
在评测之前必须修正，否则会影响测评结果的正确性。</td><td>维护时间</td><td>××年××月××日～××年××月××日
共计：0.5人月</td></tr>
<tr><td>环境</td><td></td></tr>
<tr><td>申请人</td><td>×××</td><td colspan="2">[√] 批准　　[　] 拒绝</td></tr>
<tr><td colspan="2">申请评价结果：修正错误</td><td colspan="2">评价负责人：×××</td></tr>
</table>

软件维护过程——软件维护记录

<table>
<tr><td colspan="3">记录编号：eval_wh_012</td><td colspan="3">日期：2023年6月9日</td></tr>
<tr><td colspan="3">计划编号：eval_wh_012</td><td colspan="3">项目名称：网络测评系统</td></tr>
<tr><td colspan="6">初始状态描述：不同类型的人员可以进行交叉测评。按需求：各类人员只进行自身类型的测评，如管理人员只能对管理人员进行测评，教师只能评测教师。</td></tr>
<tr><td colspan="3">模块名称：测评控制管理
源程序函数：Evaluating
编程语言：C++
失效次数：3</td><td colspan="3">编号：evalobject_01
机器指令长度：25 Kb
程序安装日期：××年××月××日
程序运行时间：</td></tr>
<tr><td>日期</td><td>维护内容</td><td>增/删/改</td><td>工作量</td><td colspan="2">维护人员</td></tr>
<tr><td>××年××月××日</td><td>查错，定位错误的位置</td><td>修改部分源程序</td><td>0.5人月</td><td colspan="2">×××</td></tr>
<tr><td>……</td><td></td><td colspan="3"></td></tr>
<tr><td colspan="6">维护结束：经过对需求的进一步确认，对制定编号的模板进行了修改，纠正了源程序中出现的错误。
维护人员：×××</td></tr>
</table>

软件维护过程

3. 软件维护管理的基本内容——维护工作流



4. 软件维护管理的基本内容——维护成本

软件维护的成本主要由非技术因素和技术因素两大类构成。

非技术因素主要包括：

(1) 开发经验

(2) 人员稳定性

(3) 应用时间

(4) 外部支撑环境

(5) 用户需求变化

技术性因素主要包括：

(1) 软件复杂程度

(2) 软件维护员的能力

(3) 软件配置管理能力

(4) 软件编程规范

5. 软件维护管理的基本内容——维护评审

软件维护的每阶段工作完成，都应进行正式软件维护评审，或非正式的软件维护评价。

强调软件维护记录的管理，将记录的一些特性量化，作为维护评价的参考指标。这些量化特性包括：

(1) 记录维护申请报告的平均处理时间；

(2) 每次维护活动的总工作量；

(3) 每次程序运行时的平均出错次数；

(4) 统计每段程序、每种语言、各种维护类型的程序平均修改次数；

(5) 统计在维护中，增加、删除每个源程序语句所花费的平均工作量；

(6) 统计所有语言及用于每种语言的平均工作量；

(7) 计算各类维护申请的比例。

软件维护过程

1. 维护中存在的问题——可维护性问题

软件维护中出现的问题分为：一是较少考虑软件的可维护性问题，二是软件维护过程中带来的副作用；三是维护过程的非结构化。

在维护过程中，较少考虑软件系统的可维护性问题：

(1) 代码没有注释；

(2) 缺乏软件配置；

(3) 开发人员的流动性；

(4) 缺少软件维护理念；

(5) 缺乏对软件维护工作的认识。

2. 维护中存在的问题——副作用

在软件维护中还存在软件维护自身带来的副作用。软件维护副作用是指在维护软件过程中引发的其它不希望发生的情况。

1、修改代码的副作用

在修改源代码时，可能引起的错误。

2、修改数据的副作用

在修改数据结构时，有可能造成软件设计与数据结构不匹配，因而导致软件出错。数据副作用就是修改软件信息结构导致的结果。

3、修改文档的副作用

对软件的数据流、软件结构、模块逻辑等进行修改时，必须对相关技术文档进行相应修改。但修改文档过程会产生新的错误，导致文档与程序功能不匹配，缺省条件改变等错误，产生文档的副作用。

软件维护过程

3. 维护中存在的问题——非结构化维护

由于软件维护工作通常并不由软件的设计和开发人员来完成，维护人员首先要对软件各阶段的文档和代码进行分析、理解。因而出现了理解别人的程序困难、文档不齐等问题，尤其是对大型、复杂系统的维护，更加困难和复杂，甚至是不可能的！

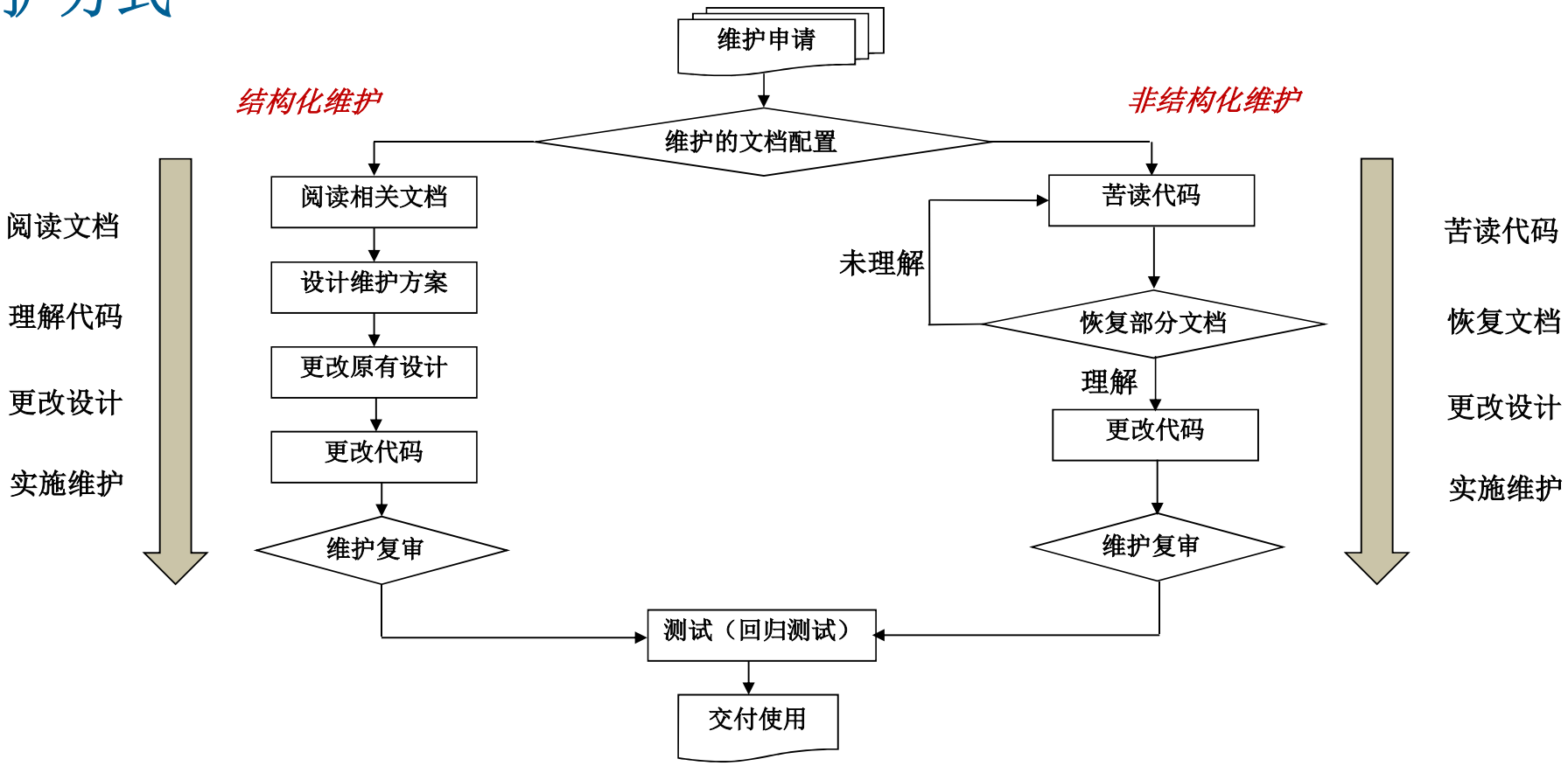
结构化维护与非结构化维护

非结构化维护 — 缺乏必要的文档说明，文档缺少或者不一致，难于确定数据结构、系统接口等特性，这样的维护工作令人生畏，事倍功半。

结构化维护 — 指软件开发过程是按照软件工程方法进行的，开发各阶段的文档齐全，软件的维护过程，有一整套完整的方案、技术、审定过程及文档。

软件维护过程

软件维护方式



软件的可维护性

软件的可维护性因素

软件可维护性是评价软件产品质量的一项重要指标，它是指当对软件实施各类型的维护而进行修改时，软件产品可被修改的能力。

决定软件可维护性的因素主要有以下**5**个要素：

(1) 可理解性：文档。源码及技术文档。

(2) 可测试性：文档。测试方案文档。

(3) 可修改性：设计方案的开闭原则。

(4) 可移植性：系统与外部交互的能力。

(5) 可重用性：重用设计模式与重用代码，提升系统稳定性、可靠性，确保系统质量。

软件的可维护性

1. 提高软件的可维护性——软件文档配置

软件系统文档在维护过程中分为用户文档、系统文档和历史文档三类。

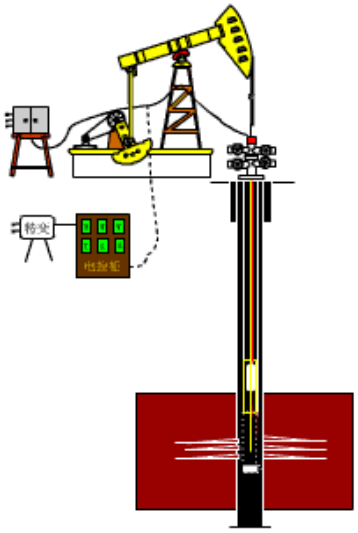
(1) 用户文档（非技术）：面向用户、描述系统功能和操作过程的文档。

(2) 系统文档（工程相关）：在软件生命周期中的工程、数据和管理文档。

(3) 历史文档（版本管理）：在软件生命周期中的SCI、临时文档。

软件的可维护性

2.提高软件的可维护性——使用能提高维护效率的开发技术和工具



采用主流的、成熟的软件开发技术，结合结构化程序设计和面向对象程序设计中的特性与指导性规则，使用相关的软件开发工具，以及DevOps过程管理工具，都将不同程度的改善软件维护特性。

(1) 模块化设计

(2) 结构化程序设计

(3) 面向对象程序设计

软件的可维护性

3. 提高软件的可维护性——可维护性复审

可维护性复审是指在为软件工程各阶段进行技术审查和管理复审的同时，考虑有哪些可维护性因素，并努力提高软件的可维护性。

⑴ 开发各阶段的可维护性因素；

⑵ 软件维护不仅仅是面对源程序代码，更重要的是整个软件文档配置；

⑶ 编程的风格和代码规范，对软件维护起着重要作用。