软件工程

第9章软件演化和维护

©第9章.教材

思考

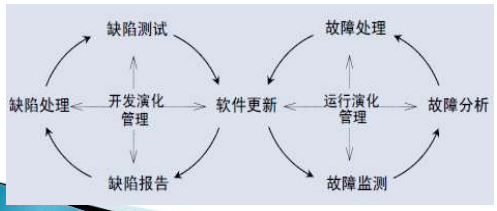
- 软件演化的规律是如何的？
- 软件维护就是改错吗？
- 软件维护和软件新产品开发哪个更具挑战？
- 软件维护的关键技术是什么？如何保证它的成功？

软件演化--软件的基本属性

- **什么是软件演化** *(Software Evolution)*
  - 是指对软件进行维护和更新的一种行为，它是软件生命周期中始终存在的变化活动。
- **按生命周期的不同阶段，软件演化可分为：**
  - **开发演化**
    - 创造一个新软件的过程，它强调要在一定的约束条件下从头开始实施，占软件演化的30%
  - **运行演化**
    - 又称软件维护（Software Maintenance），是软件系统交付使用以后，为了改正错误或满足新的需要而修改软件的过程，它强调必须在现有系统的限定和约束条件下实施，占软件演化的70%。

软件开发演化与运行演化

- 在传统环境下，运行演化在开发演化后发生
- 在网络环境下，开发演化与运行演化已呈现出交织协同、并生共长的态势，软件运行状态改变的同时，软件版本也不断升级



Lehman的8条软件演化法则

- (1974)**持续变更法则**。软件必须持续改进，否则就会变得越来越不令人满意。
- (1974)**复杂度递增法则**。软件的复杂性随着演化不断增加，除非采取措施使系统保持或降低复杂性。
- (1974)**自调节法则**。软件的演化过程可以自动调节产品分布和过程测量，以接近正常状态。
- (1978)**组织稳定性法则**。在软件的生命周期中，组织的平均开发效率是稳定的。

- (1978)**通晓法则**。随着软件的演化，所有相关人员（如开发人员、销售人员和用户）都必须始终掌握软件的内容和行为，以便达到满意的演化效果。
- (1991)功能持续增加法则。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。
- (1996)**质量衰减法则**。如果没有严格的维护和适应性调整使之适应运行环境的变化，软件的质量会逐渐衰减。
- (1996)**反馈系统法则**。软件演化过程是由多层次、多循环、多主体的反馈系统组成，而且要想在任何合理的基础上达到有意义的改进就必须这样进行处理。

软件维护“冰山”

- 有关调查结果表明，软件维护是软件工程中最消耗资源的活动，很多软件公司中**软件维护的成本**已经达到了整个软件生存周期资源的**40%到70%**，甚至达到了90%。
- 软件系统趋于大型化和复杂化，大多数软件在设计时没有考虑到将来的修改问题，常常还伴有开发人员**变动频繁、文档不够详细、维护周期过长**等等问题，这些问题导致维护活动的时间与花费不断增加。

维护类型

	纠错	增强
积极的	预防性维护	完善性维护
被动的	纠错性维护	适应性维护

- **纠错性维护**（Corrective maintenance）
  - 由于软件中的缺陷引起的修改
- **完善性维护**（Perfective maintenance）
  - 根据用户在使用过程中提出的一些建设性意见而进行的维护活动
- **适应性维护**（Adaptive maintenance）
  - 为适应环境的变化而修改软件的活动
- **预防性维护**（Preventive maintenance）
  - 为了进一步改善软件系统的可维护性和可靠性，并为以后的改进奠定基础

维护的技术问题

- **有限理解**（Limited understanding），对他人开发软件进行维护时，如何快速理解程序并找到需要修改或纠错的地方？
- 在软件维护过程中需要大量的回归测试，**耗时耗力**。
- 当软件非常关键以致**不能停机时**，如何进行在线维护而不影响软件的运行？
- **影响分析**，如何对现有软件的变更所进行的全面分析？
- 如何在开发中促进和遵循软件的**可维护性**？
  - **易分析性**（*Analyzability*）、**易改变性**（*Changeability*）、**稳定性**（*Stability*）和**易测试性**（*Testability*）

维护成本

- **维护的工作可划分成：**
  - **生产性**活动 如，分析评价、修改设计、编写程序代码等
  - **非生产性**活动 如，程序代码功能理解、数据结构解释、接口特点和性能界限分析等
- **维护工作量的模型**

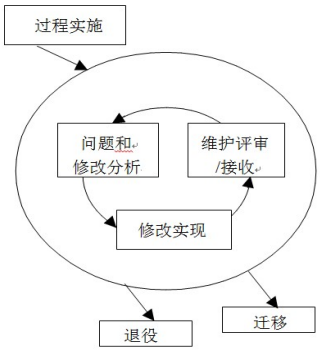
$$M = p + Ke^{c-d}$$

  - M：维护的总工作量；
  - P：生产性工作量；
  - K：经验常数；
  - c：软件的规模；
  - c：复杂程度；
  - d：维护人员对软件的熟悉程度

影响维护成本的因素

- **操作环境：硬件和软件**
  - 软件的规模越大、复杂性越高、年龄越大，硬件的能力越低，软件维护的成本和工作量就越大。
- **组织环境：策略、竞争力、过程、产品和个人**
  - 软件开发过程和维护过程**越规范**，采用的设计方法和编程语言模块化程度越高，工程师对软件的熟悉程度越高、能力越强，产品的可靠性和安全性要求越低，**软件维护的成本和工作量就越小**。

维护过程



维护活动

- **过程实施**（Process Implementation）。建立维护过程期间应执行的计划和规程，包括维护计划、维护规程、问题解决规程、用户反馈计划、移交计划、配置管理计划。
- **问题和修改分析**（Problem and Modification Analysis）。在修改软件前，要分析修改请求/问题报告，以确定其对组织、现行系统和接口系统的影响，提出可能的方案建议并形成文档，通过核准形成期望的解决方案。
- **修改实现**（Modification Implementation）。根据计划和方案更新相应的需求、设计和代码，并进行测试等软件验证工作。

- **维护评审/接收**（Maintenance Review/Acceptance）。对上述的维护进行评审，以确保对软件的修改是正确的，并且这些修改是使用正确的方法按批准的要求完成的。
- **迁移**（Migration）。在软件的生存周期期间，如果需要将它迁移到一个新环境，则应制订迁移计划、通告用户迁移、提供迁移培训、把软件迁移至新环境、通告迁移完成情况、评估新环境的影响、并进行旧软件和数据的归档。在迁移实施时，旧环境和新环境可以并行进行工作，以便平稳迁移到新环境。
- **退役**（Retirement）。软件一旦结束使用生存周期，必须退役。退役时，要制定退役计划、通知用户退役、提供退役培训、实施退役、通告退役完成情况、并进行旧软件和数据的归档。在制定退役计划时，要分析退役的成本和影响、决定是局部还是全部退役、是否用新软件来代替退役软件等。

软件维护技术

- **程序理解**
- **逆向工程**（*Reverse Engineering*）
- **再工程**（*Reengineering*）

程序理解

- 软件维护的**总工作量大约一半**被用在理解程序上。
- 程序理解通过提取并**分析程序**中各种实体之间的关系，形成系统的不同形式和层次的抽象表示，完成程序设计领域到应用领域的映射。
- 程序员在理解程序的过程中，经常通过反复三个活动——阅读关于程序的**文档**，阅读源**代码**，**运行**程序来捕捉信息。
- 程序理解工具
  - 基于程序结构的**可视化工具**，通过分析程序的结构，抽取其中各种实体，使用图形表示这些实体和它们之间的关系，可以直观地为维护者提供不同抽象层次上的信息。
  - 帮助维护者**导航浏览源代码**，为浏览工作提供着眼点，缩小需要浏览的代码范围。

逆向工程

- 逆向工程是分析软件，**识别出软件的组成成份及其相互的关系**，以更高的抽象层次来刻画软件的过程，它并不改变软件本身，也不产生新的软件。
- 逆向工程主要分为以下几类:
  - **重新文档化**（redocumentation）：分析软件，改进或提供软件新的文档。
  - **设计恢复**（design recovery）：从代码中抽象出设计信息；
  - **规约恢复**（specification recovery）：分析软件，导出需求规约信息；
  - **重构**（refactoring, restructuring）：在同一抽象级别上转换软件描述形式，而不改变原有软件的功能；
  - **数据逆向工程**（data reverse engineering）：从数据库物理模式中获取逻辑模式，如实体关系图。

再工程

- 再工程是在逆向工程所获信息的基础上修改或重构已有的软件，产生一个新版本的过程，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**。