软件质量度量——产品质量（上）

概述

- 产品质量度量
- 过程中质量度量
- 软件维护中质量度量

软件产品质量度量

- 产品本质质量度量
  - 平均失效时间
  - 缺陷密度
- 用户满意度度量
  - 用户问题
  - 用户满意度

1. 平均失效时间

- MTTF，Mean Time to Failure
- 软件产品在规定的环境下，从正常运行到发生下一次故障的平均时间。
- MTTF的计算十分困难
  - 缺陷与失效之间并非一一对应
  - 数据的采集非常困难
- MTTF常用于安全性要求较高的系统

2. 缺陷密度

- 用于测量与软件规模相关的缺陷率
- 缺陷率 = 缺陷数/一定时间范围内的软件规模
- 软件规模：
  - 代码行（Line of code, LOC）
  - 功能点（Function point，FP）

基于代码行的缺陷率

SSI（i）= SSI（i-1）+ CSI（i）- 删除的代码 - 修改的代码

- 代码行：基于指令语句（逻辑LOC），包括可执行代码和数据定义，不包括注释
- SSI（Shipped source instructions）：整个软件产品的代码行
- CSI（Changed source instructions）：新版本中增加和修改代码对应的代码行

基于代码行的缺陷率

- 总缺陷数 / KSSI：评价整个产品的代码质量
- 实际应用缺陷数 / KSSI：评价应用领域内的缺陷率
- 版本的缺陷数 / KCSI：评价当前的开发质量
- 版本的应用缺陷数 / KCSI：评价针对用户发现的缺陷的开发质量

基于代码行的缺陷率的目标

- 缺陷率 = 缺陷数 / 代码行
- 目标：
  - 更低的缺陷率？
  - 缺陷率 = 缺陷数 / 代码行
  - 更少的总缺陷数？
  - 缺陷率 = 缺陷数 / 代码行

基于代码行的缺陷率

- 从技术角度估算软件的规模
- 局限性
  - 实际的代码行与指令语句之间存在差别，导致程序大小的差异很难估算
  - 代码行的数量通常与设计效率成反比，高效的设计往往会减少代码行
  - 代码行的多少并不能反映除了编码之外的工作

基于功能点的缺陷率

- 功能：执行一定任务的可执行语句的集合
- 理想情况下，平均每个功能点的缺陷数越低，软件质量越好

代码行方法 vs 功能点方法

	代码行法	功能点法
使用场合	开发完成时	项目开始或项目需求基本明确时
估算的角度	以技术角度	以用户角度
技术相关性	与开发技术密切相关	与开发技术无关
准确度	较低	较高