第四章 软件质量度量

- 1测量基础
  - —测量原理
  - —测量标准
  - —测量过程和原则
- 2软件开发生命周期的度量活动
- 3软件项目的质量度量
- 4软件产品的质量度量
- 5软件过程的质量度量
- 6软件质量度量中的统计分析

测量原理

3个基本概念

- *测量*

  是对产品过程的某个属性的范围、数量、维度、容量或大小提供一个定量的指示。

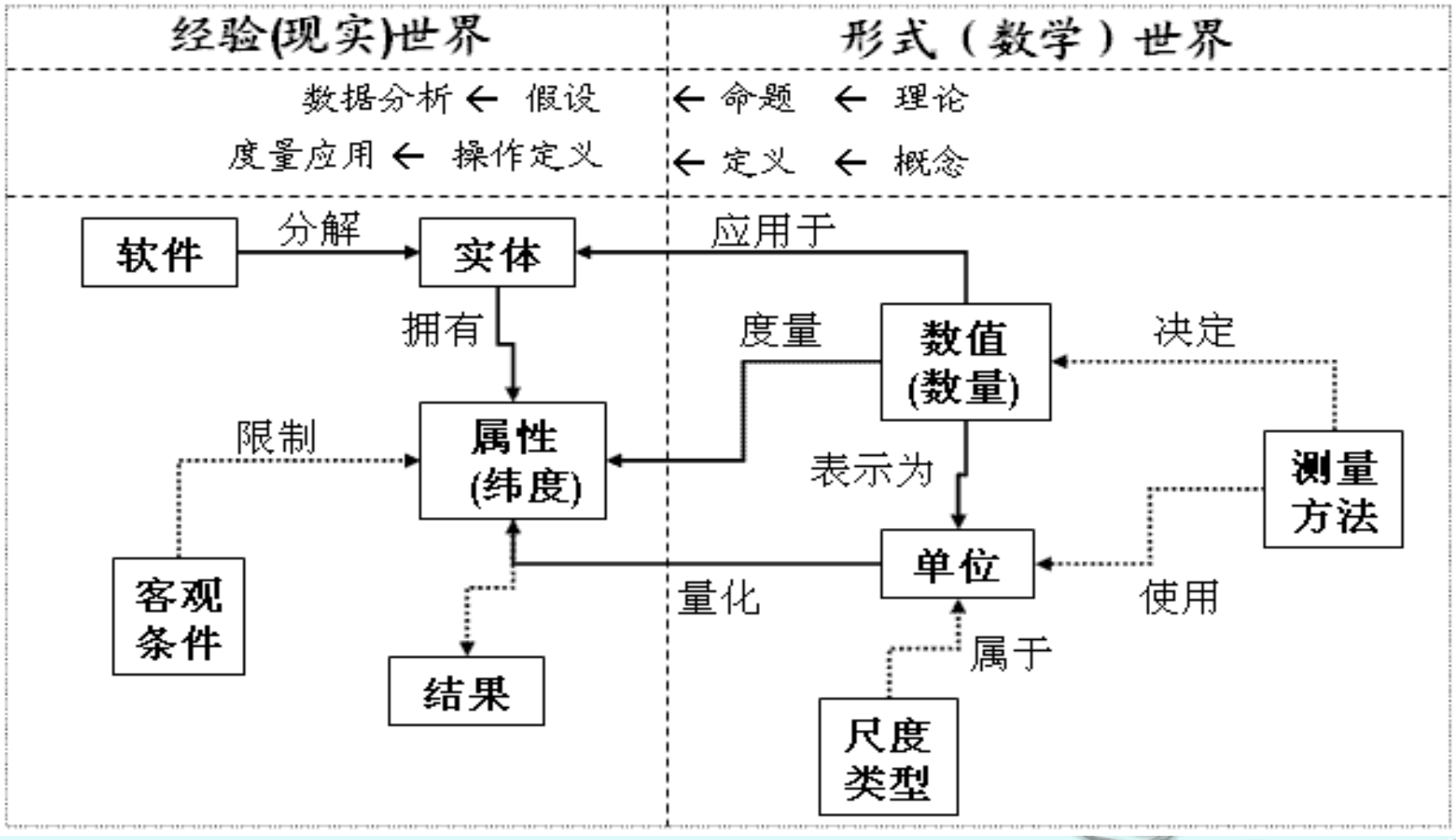
- *度量*

  是对软件产品进行范围广泛的测度，它给出一个系统、构件或过程的某个给定属性的度的定量测量。

- *指标*

  是一个度量或一组度量的组合，采用易于理解的形式，对软件过程、项目或产品质量提供更全面、深入的评价和了解，以利于过程和质量的分析。

测量原理



测量原理示意图

测量原理

度量尺度

- *分类尺度：*

  某个指标被分成一系列的类别。

  如产品质量属性有：功能性、适用性、性能、安全性、可靠性、可维护性等。

- *序列尺度：*

  分类的序列，即在分类的基础上，再加以排序。

  如用1、2、3表示用户的满意度，1度最低，3度最高。

  也可以用某中线为基准的相对百分比来表示程度。

- *间隔尺度：*

  通过数值来表示两个邻近测量点之间的差异，

  但没有绝对的“零”值。

- *比值尺度：*

  和间隔尺度相似，但有绝对的“零”值存在。

测量标准

有效性和可靠性是测量标准中最重要的指标

- ***有效性***

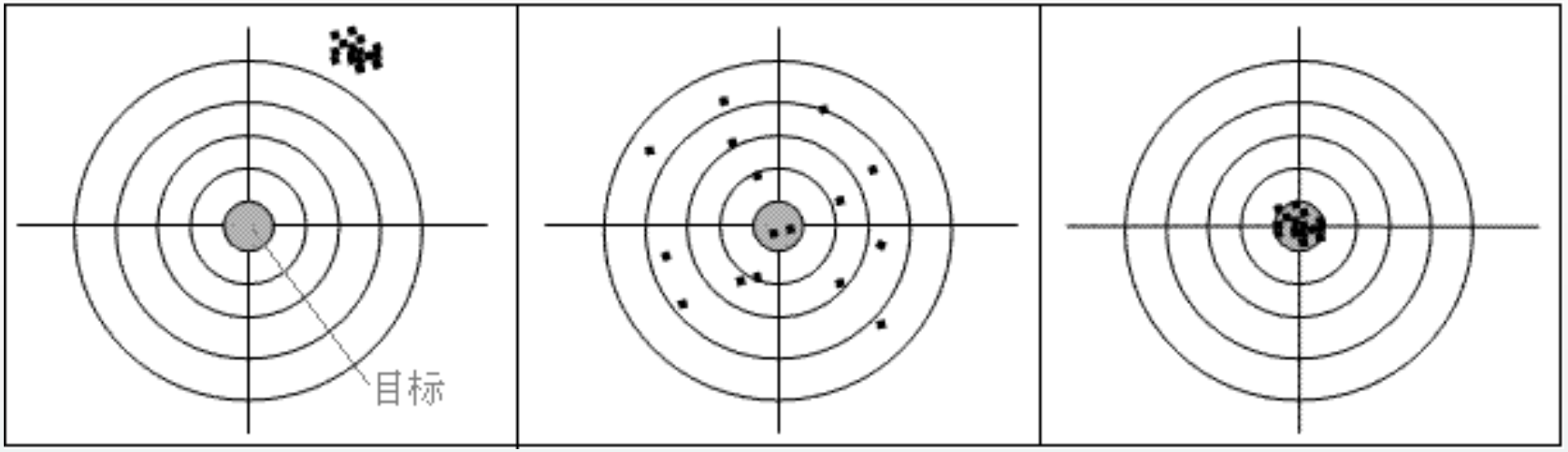
  指的是测量的结果正确反映了被测试对象的实际状况和程度、或合乎事务的发展、变化的规律。

- ***可靠性***

  指的是使用同样的测量方法对同样的事物进行多次测量，得到值的一致性。

  多次测量的值越接近，可靠性就越高；反之则可靠性越低。

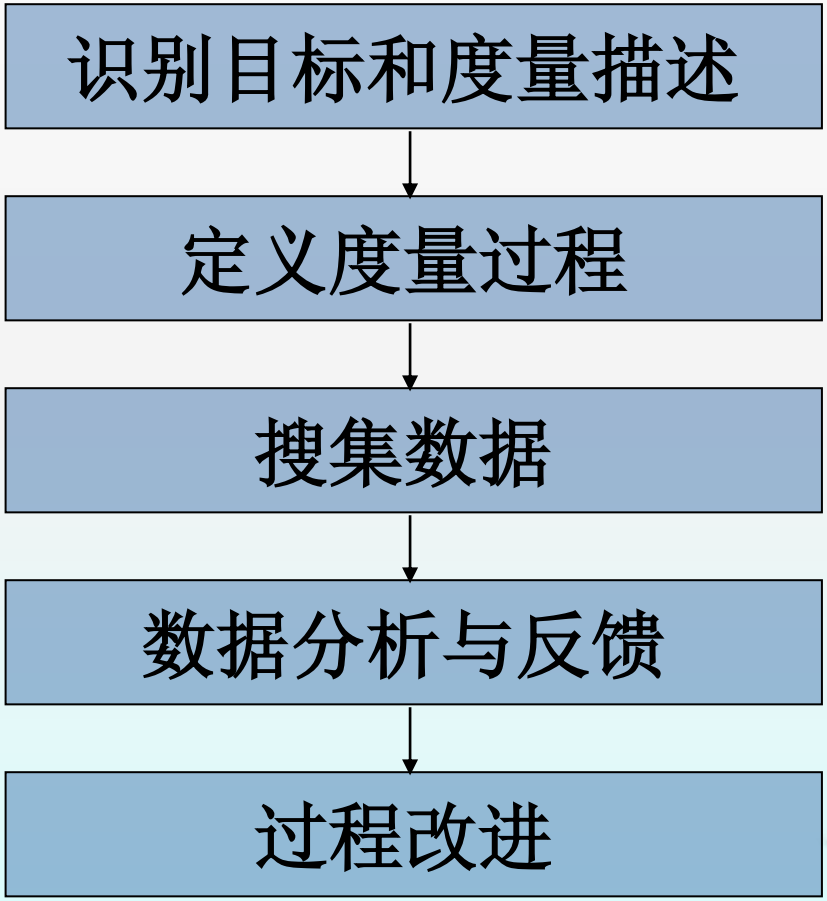
测量标准



可靠但不有效　　有效但不可靠　　可靠且有效

测量过程和原则

- *测量过程*



测量过程和原则

- ***基本的测量原则***
  - 测量应该基于该应用领域**正确的理论**之上，并在测量的定义中**确定测度的目标**；
  - 每一个技术测量的定义应该具有**一致性、客观性、无二义性**；
  - 测量在**经验和直觉**上也应该有说服力；
  - 测量的方法力求**简单**以及**可计算性**；

测量过程和原则

- *基本的测量原则*
  - 测量应该被剪裁以**最适应特定的产品和过程**，而且任何时候应尽可能使得收集和分析**自动化**；
  - 应该用正确的**统计技术**来**建立**内部产品属性和外部待测量特征的**关系**；
  - 测量结果应该是**可靠**的，不会因为一些技术问题导致测量结果很大的偏离。
  - 测量应该建立**反馈机制。**

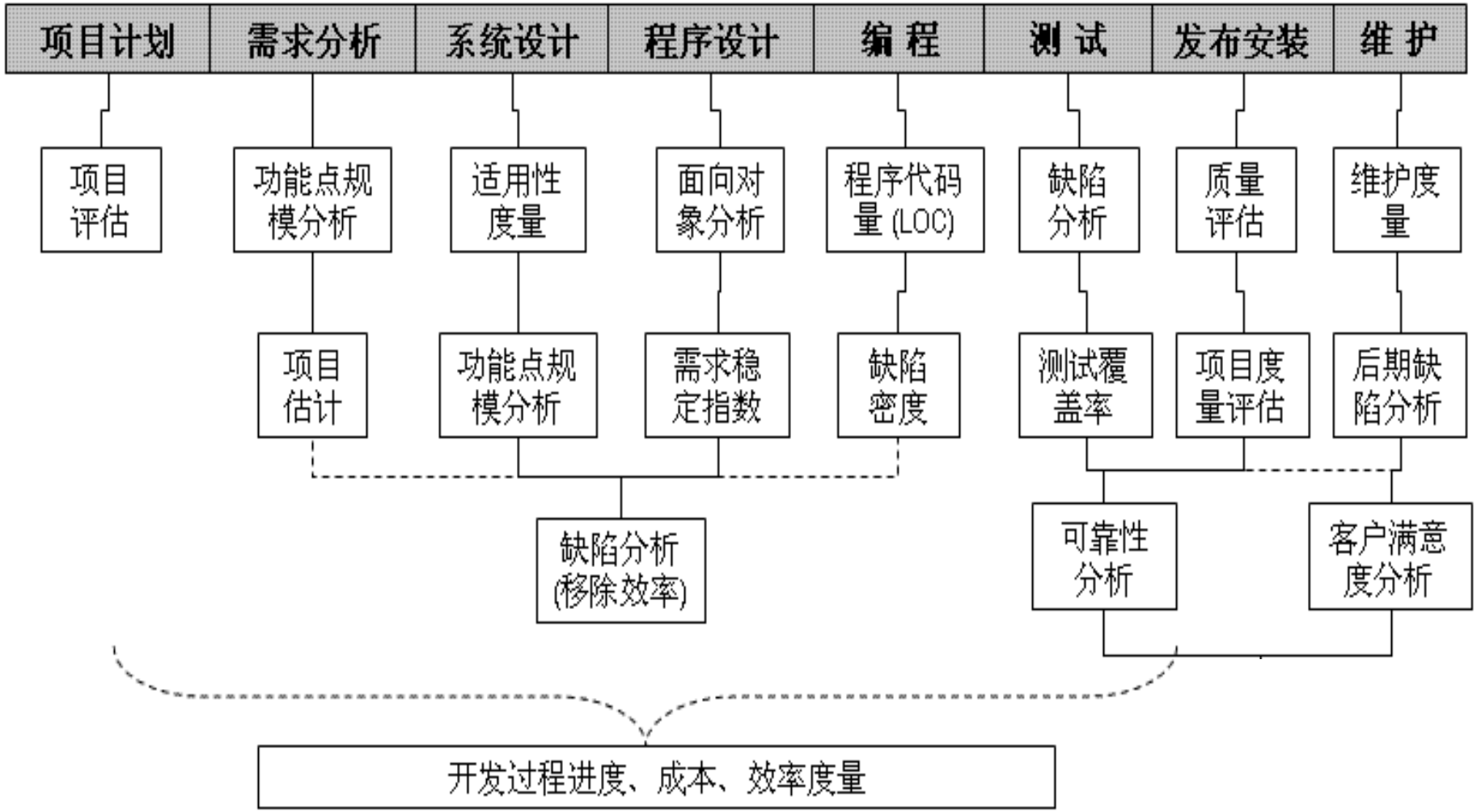
第四章 软件质量度量

- 1测量基础
- 2软件开发生命周期的度量活动
- 3软件项目的质量度量
- 4软件产品的质量度量
- 5软件过程的质量度量
- 6软件质量度量中的统计分析

软件开发生命周期的度量活动

软件开发生命周期中的度量活动

软件开发生命周期的度量活动



软件开发生命周期中的度量活动

软件开发生命周期的度量活动

软件质量度量按其研究对像可分为3类:

项目质量度量、产品质量度量、过程质量度量

- 软件项目质量度量
  - 用来描述项目的特性和执行状态。
  - 项目度量包括项目计划的有效性、项目资源使用效率、成本效益、项目风险、进度和生产力等。
  - 目的:
    - 评估项目开发过程的质量、预测项目进度、工作量等,
    - 辅助管理者进行质量控制和项目控制。

软件开发生命周期的度量活动

- 软件产品质量度量
  - 主要用来描述软件产品的特征，用于产品评估和决策。
  - 产品度量包括软件规模大小、产品复杂度、设计特征、性能以及质量水平。
- 软件过程质量度量
  - 用于软件开发、维护过程的优化和改进。
  - 过程度量包括开发过程中的缺陷移除效率、测试阶段中的缺陷到达模式以及缺陷修复过程的效率等。
  - 对于软件过程本身的度量，目的：
    - 形成适合软件组织应有的各种模型，作为对项目、产品的度量基础；
    - 对软件开发过程进行持续改进，提高软件生产力。

第四章 软件质量度量

- 1测量基础
- 2软件开发生命周期的度量活动
- 3软件项目的质量度量

  —度量分类

  —规模度量

  —顾客满意度度量
- 4软件产品的质量度量
- 5软件过程的质量度量
- 6软件质量度量中的统计分析

度量分类

- **进度**度量

  通过任务分解、工作量度量、有效资源分配等做出计划，然后将实际结果和计划值进行对比来度量。

- **风险**度量

  一般通过两个参数“风险发生的概率”和“风险发生后所带来的损失”来评估风险。

- **规模**度量

  以代码行数、功能点数、对象点或特征点等来衡量。软件规模度量是工作量度量、进度度量的基础，用于估算软件项目工作量、编制成本预算、策划项目进度的基础。

度量分类

- **复杂度**度量

确定程序控制流或软件系统结构的复杂程度指标。复杂度度量用于估计或预测软件产品的可测试性、可靠性和可维护性，以便选择最优化、最可靠的程序设计方法，来确定测试策略、维护策略等。

- **缺陷**度量

帮助确定产品缺陷分布的情况、缺陷变化的状态等，从而帮助分析修复缺陷所需的工作量、设计和编程中存在哪些弱点、预测产品发布时间、预测产品的遗留缺陷等。

度量分类

- **工作量**度量

  任务分解并结合人力资源水平来度量，合理地分配研发资源和人力，获得最高的效率比。工作量度量是在软件规模度量和生产率度量的基础上进行。

- **其他**的项目度量

  顾客满意度、需求稳定性、资源利用效率、文档复审水平、问题解决能力、代码动态增长等。

规模度量

- 德尔菲法

德尔菲法是一种专家评估技术，适用于在没有或没有足够的历史数据情况下，来评定软件采用不同的技术、新技术所带来的差异。

专家的水平及对项目的理解程度是工作中的关键点。

- 造性成本模型

造性成本模型是一种精确、易于使用的基于模型的成本估算方法。

- 代码行度量方法
- 功能点分析法
- 面向对象软件的对象点方法

顾客满意度度量

顾客满意度要素	顾客满意度度量内容
软件产品	功能性、可靠性、易用性、 效率性、可维护性、可移植性
开发文档	文档的构成、质量、外观、 图表以及索引、用语
项目进度以及交期	交期的根据、进度迟延情况下的应对、进展报告
技术水平	项目组的技术水平、项目组的提案能力、 项目组的问题解决能力
沟通能力	事件记录、格式确认、问题解答
运用维护	支持、问题发生时的应对速度、 问题解决能力

第四章 软件质量度量

- 1测量基础
- 2软件开发生命周期的度量活动
- 3软件项目的质量度量
- 4软件产品的质量度量
  - —度量分类
  - —软件缺陷的度量
  - —软件复杂性的度量
  - —顾客满意度度量
- 5软件过程的质量度量
- 6软件质量度量中的统计分析

度量分类

- 按度量方式和手段
  - 直接度量：
    - 软件缺陷数、缺陷密度、
    - 软件性能，所耗资源、投入成本等
  - 间接度量：
    - 复杂性、客户满意度、
    - 功能性、效率、可维护性等

软件缺陷的度量

- 缺陷密度--软件缺陷在规模上的分布

  例如每KLOC或每个功能点（或类似功能点的度量--对象点、数据点、特征点等）的缺陷数。

- 缺陷率--缺陷在时间上的分布

  例如对于应用软件的角度来说，90%以上的缺陷是在发布后两年内被发现出来。

- 整体缺陷清除率

  在软件开发过程中发现的所有缺陷数/发现的总缺陷数。

- 阶段性缺陷清除率

$$\frac{\text{开发阶段清除的缺陷数}}{\text{产品中潜伏的缺陷数}} \times 100\%$$

软件复杂性的度量

- 可以评估软件系统的可测试性、可靠性和可维护性；
- 可以提高工作量估计的有效性和精度；
- 可以在测试和维护过程中，帮助选择更有效的方法来提高软件系统的质量和可靠性；
- 可以对今后系统和程序设计进行改进。

软件复杂性的度量

复杂性的度量方法举例(一)

- McCabe环形复杂度

  对软件结构进行严格的算术分析得来的，实质上是对程序拓扑结构复杂性的度量，明确指出了任务复杂部分。

- McCabe复杂度的用途
  - 作为测试的辅助工具
  - 作为程序设计和管理指南
  - 作为网络复杂性度量的一种方法

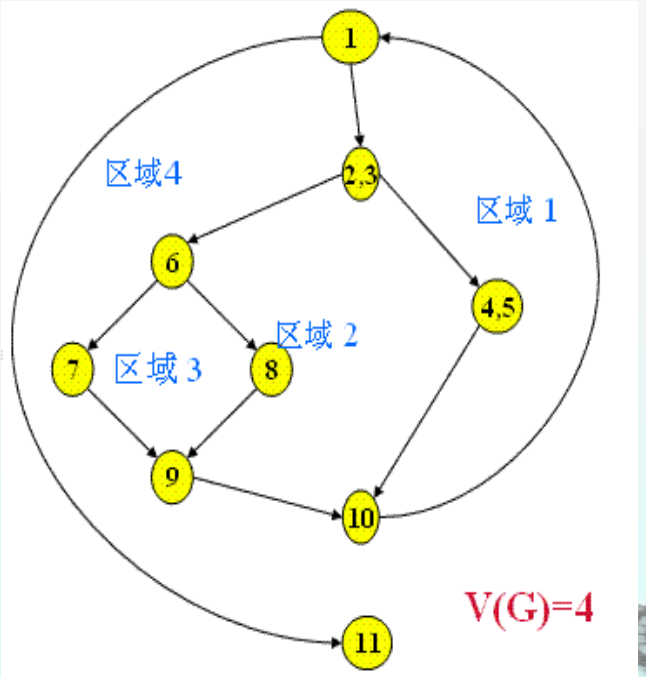
软件复杂性的度量

McCabe环形计算公式为：

① V(G) = D (D:区域数)

② V(G) = E – N + 2 (E:流图中边的条数; N:流图中结点数)

- 右图中圆表示结点，代表一条或者多条语句
  程序流程图中一个顺序的处理框序列和一个菱形判定框可以映射成流图中的一个结点。
- 右图中箭头线称为边，代表控制流
  一条边必须终止于一个结点，
  即使这个结点并不代表任何语句。
- 右图中边和结点围成的面积为区域
  计算区域数时应该包括
  图外部未被围起来的区域。



软件复杂性的度量

优点：

- 避免软件中的错误倾向
- 指出极复杂模块，这样的模块也许可以进一步细化
- 度量测试计划，确定测试重点
- 在开发过程中通过限制程序逻辑，指导测试过程
- 指出将要测试的区域
- 帮助测试人员确定测试和维护对象
- 与所用的高级程序设计语言类型无关

缺点：

- 局限于二元决策的指标，不能区分不同种类的控制流复杂性，例如Loop和if then else或case和if then else

软件复杂性的度量

应用：

- 圈复杂度指出为了确保软件质量应该检测的最少基本路径的数目。
- 在实际中，测试每一条路经是不现实的，测试难度随着路径的增加而增加。
- 但测试基本路径对衡量代码复杂度的合理性是很必要的。

建议：

- 建议圈复杂度到10，因为高的圈复杂度使测试变得更加复杂而且增大了软件错误产生的概率。
- 圈复杂度度量是测量在一个软件模块中的分支数目，在所有的开发周期中都要使用。

软件复杂性的度量

复杂性的度量方法举例(二)

- 语法构造方法

  基本思路是根据程序中可执行代码行的操作符和操作数的数量来计算程序的复杂性。

  操作符和操作数的量越大，程序结构就越复杂。

- 语法构造方法可以揭示程序中单独的语法构造和缺陷率之间的关系：

  缺陷率= 0.15 + 0.23 DO WHILE + 0.22 SELECT

  + 0.07 IF-THEN-ELSE

软件复杂性的度量

复杂性的度量方法举例(三)

- 结构度量方法
  - Henry给出的复杂性度量模型

    Cp = (扇入× 扇出)^2

    其中：
    - 扇入:调用外部模块的模块数
    - 扇出:被外部模块调用的次数
  - Card/Glass给出的复杂性度量模型

    Ct= St + Dt

    其中：
    - St:结构复杂性
    - Dt:数据复杂性

$$S_t = \sum f^2(i)/n$$

$$D_t = \sum \frac{D(i)}{n} = \sum (V_{(i)}/(f(i)+1))/n$$

n:系统的模块数
f(i):模块i的扇出
V(i):模块i的I/O变量数

顾客满意度度量

顾客满意度要素	顾客满意度度量内容

顾客满意度度量

顾客满意度要素	顾客满意度度量内容
技术解决方案	质量、可靠性、有效性、 易用性、价格、安装、新技术
支持与维护	灵活性、易达性、产品知识
市场营销	解决方案、接触点、信息
管理	购买流程、请求手续、 保证期限、注意事项
交付	准时、准确、交付后过程
企业形象	技术领导、财务稳定性、执行印象

第四章 软件质量度量

- 1测量基础
- 2软件开发生命周期的度量活动
- 3软件项目的质量度量
- 4软件产品的质量度量
- 5软件过程的质量度量

  —共性过程质量要素

  —个性过程质量要素
- 6软件质量度量中的统计分析

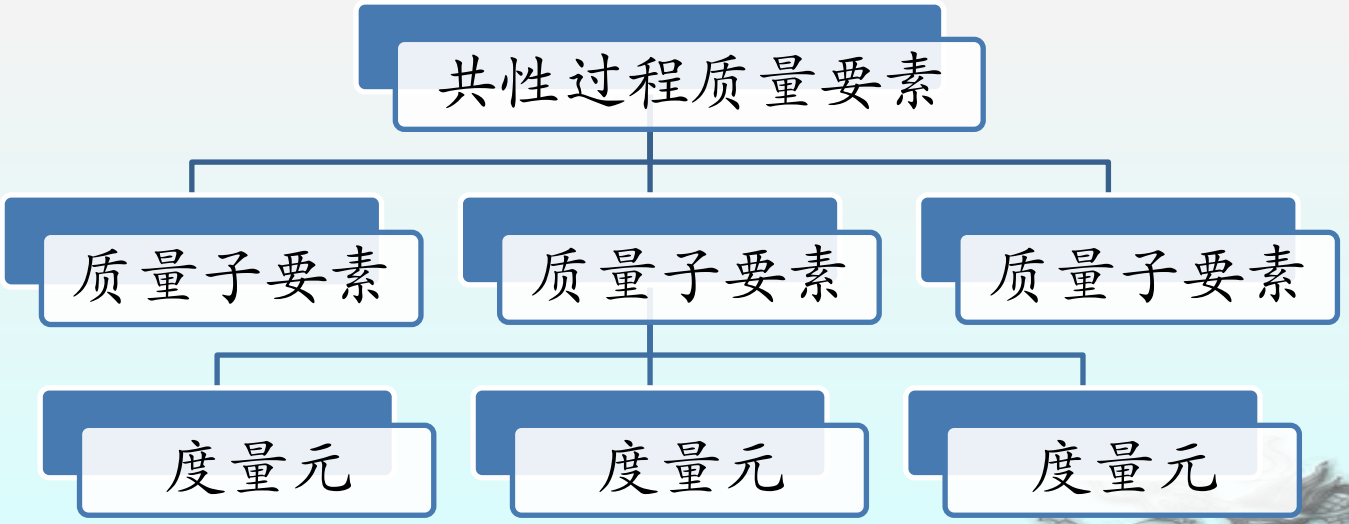
软件过程的质量度量

软件过程质量要素分为共性过程质量要素和个性过程质量要素两大类。

- 共性过程质量要素
  - 与软件开发过程中多个活动阶段都相关的质量要素，它在每个阶段中所需收集的数据、所用的度量方法、评价准则都是类似的。
- 个性过程质量要素
  - 指只和当前的过程活动相关的质量要素。

共性过程质量要素

- 对软件过程质量的改进围绕改善软件质量、提高生产效率和降低成本这三方面展开。
- 涉及到7个互相关联的共性过程质量要素：进度、资源和费用、评审、缺陷、开发性能、技术完备性、需求稳定性。
- 共性过程质量要素组成部分



共性过程质量要素

质量要素	质量子要素	度量元	计算公式（评分标准）

共性过程质量要素

质量要素	质量子要素	度量元	计算公式（评分标准）
进度	里程碑有效性ME	项目里程碑时间t0	ME=|t1-t0|/t1
		计划完成时间t1	
	迭代效率IE	每一次迭代增加的功能点数nf0	IE=nf0/nf1
		总功能点数nf1	
资源和费用	人员稳定性PSI	变动的人员数pc	PSI=(ps-pc)/ps
		计划人员数ps	
	成本性能指数CPI	进度预算pc	CPI=pc/pp
		实际成本pp	
	资源完备性RE	计划资源的数量np	RE=nf/np
		买际资源的数量nf	

共性过程质量要素

质量要素	质量子要素	度量元	计算公式（评分标准）

共性过程质量要素

质量要素	质量子要素	度量元	计算公式（评分标准）
开发性能	能力成熟度CMMI	过程所处的能力成熟度等级	CMMI1,…,CMMI5
	生产效率PDC	产品规模fx	PDC=fs/wl
		人员投入的工作量wl	
技术完备性	计算机资源利用率SC	当前利用的资源数量s1	SC=s1/s2
		CPU I/O内存存储系统的总资源数s2	
	技术性能TE	满足用户的功能需求和技术性能需求的功能点数nr,总功能点数ns	TE=nr/ns
	模块复用率ME	可重用部件数目nr	ME=nr/ns
		总模块数目ns	

共性过程质量要素

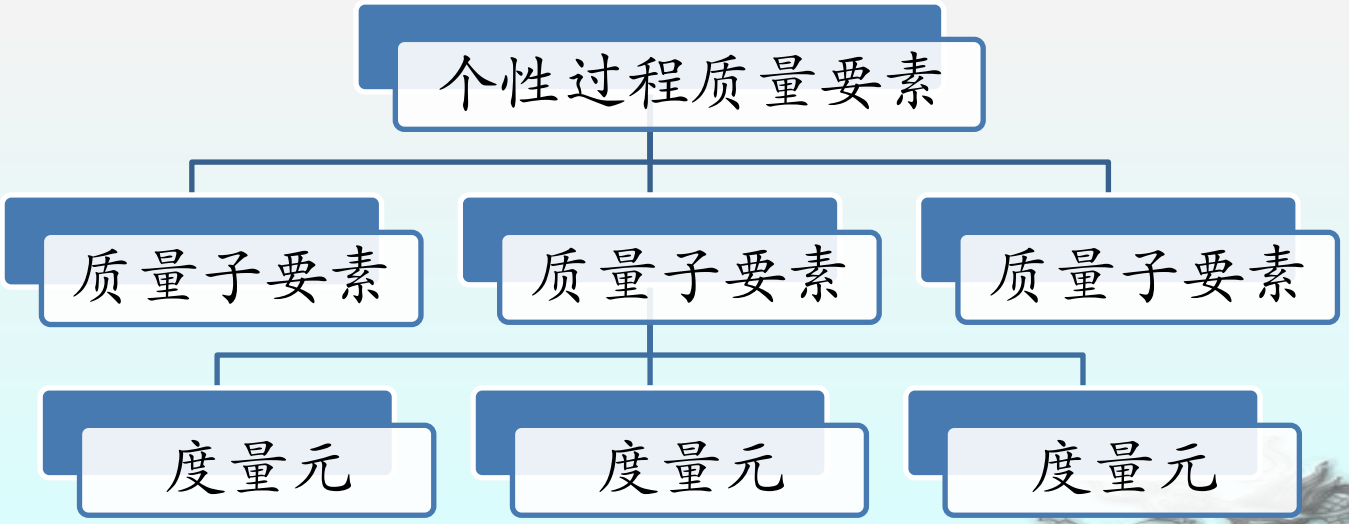
质量要素	质量子要素	度量元	计算公式（评分标准）

共性过程质量要素

质量要素	质量子要素	度量元	计算公式（评分标准）
评审有效性	评审效率CE	各阶段评审发现的问题数nc	CE=nc/nf
		各阶段总的缺陷数nf	
	评审人力率CF	各阶段的评审工作量w1	CF=w1/w2
		各阶段的总工作量w2	
缺陷	阶段缺陷清除率DRE	各阶段清除的缺陷数量nfi	DRE=nf1/(nf2+nf3)
		各阶段注入的缺陷数量nf2	
		各阶段入口处存在的缺陷数量nf3	
	缺陷密度	每千行代码缺陷数量ns	BD=ns/kloc
需求变更	需求稳定性RSI	累计的需求变化请求数nc	RSI=(n1+n2-nc)/(n1+n2)
		初始需求请求列数表n1	
		接受的需求变化请求数n2	

个性过程质量要素

- 软件开发分为需求、设计、编码、测试、运行和维护5个阶段，
- 因此个性过程质量要素也和5个开发阶段相对应，包括：需求规格说明书、设计有效性、代码审查、测试用例和环境、问题和缺陷修复。
- 个性过程质量要素组成部分



个性过程质量要素

质量要素	质量子要素	度量元	计算公式（评分标准）

个性过程质量要素

质量要素	质量子要素	度量元	计算公式（评分标准）
需求规格说明书	需求的确认度Q0	已经确认为正确的需求个数nc	Q0=nc/(nc+nnc)
		尚未被确认的需求个数nnc	
	需求的无二意性Q1	所有复审者都有相同解释的需求数目nui	Q1=nui/(nf+nnf)
		功能需求数nf	
		非功能需求数nnf	
设计有效性	设计的兼容性DE	兼容的模块个数nfc	DE=nfc/nm
		总的模块个数nm	
代码审查	审查效率WE	项目经理走查发现的缺陷数量nc	WE=nc/nn
		编码阶段总的缺陷数量nn	

个性过程质量要素

质量要素	质量子要素	度量元	计算公式（评分标准）

个性过程质量要素

质量要素	质量子要素	度量元	计算公式（评分标准）
测试用例和环境	测试用例的深度TCD	每KLOC的测试用例数nt	TCD=nt/KLOC
	测试用例的质量TCQ	测试用例发现的缺陷数tf	TCQ=tf/tt
		测试阶段总的缺陷数tt	
	测试环境的稳定性TSI	测试失效的次数nf	TSI=nl/nf
		测试时间长度nl	

个性过程质量要素

质量要素	质量子要素	度量元	计算公式（评分标准）

个性过程质量要素

质量要素	质量子要素	度量元	计算公式（评分标准）
问题和缺陷修复	修复响应时间RRT	所有问题从发现到解决的时间ti	RRT=($\sum_{i=1}^{n} ti$)/n
		发现问题的数目n	
	积压管理指标BMI	每月解决的问题数目n0	BMI=n0/n1
		每月报告的问题数目n1	
	拖欠修复百分数DRE	延迟的修复数目dn	DRE=dn/pn
		特定时间内完成的修复数pn	
	修复质量RD	一定时间间隔内有缺陷的修复df	RQ=df/(df+dd)
		一定时间间隔内无缺陷的修复dd	

第四章 软件质量度量

- 1测量基础
- 2软件开发生命周期的度量活动
- 3软件项目的质量度量
- 4软件产品的质量度量
- 5软件过程的质量度量
- 6软件质量度量中的统计分析

—统计分析步骤简介

—错误的根本原因来源

统计分析步骤简介

1. 收集和分类软件缺陷信息；
2. 找出导致每个缺陷的原因(例如，不符合规格说明书、设计错误、代码错误、数据处理不对、对客户需求误解、违背标准、界面不友好等)；
3. 使用Pareto规则(80％缺陷主要是由20％的主要因素造成的，20％缺陷是由另外80％的次要因素造成的)，要将这20％的主要因素分离出来。
4. 一旦标出少数的主要因素，就比较容易纠正引起缺陷的问题。

错误的根本原因来源

- 说明不完整或说明错误(IES)
- 与客户交流不够所产生的误解(MCC)
- 故意与说明偏离(IDS)
- 违反编程标准(VPS)
- 数据表示有错(EDR)
- 模块接口不一致(IMI)
- 设计逻辑有错(EDL)
- 不完整或错误的测试(IET)
- 不准确或不完整的文档(IID)
- 将设计翻译成程序设计语言中的错误(PLT)
- 不清晰或不一致的人机界面(HCI)
- 杂项(MIS)

小结

- 度量的目的是改进开发过程，提高产品质量，塑造度量文化。
- 软件度量只针对项目、产品和过程而开展。
- 从小规模的简单的度量项目开始
- 度量将在一定程度上增进对软件开发的理解、预测、评估、控制和改善。
- 根据项目实情加以具体实施。
- 通过数据共享增进信任，消除软件度量可能带来的误解。
- 度量应该简单易懂，从而降低沟通和实施成本。