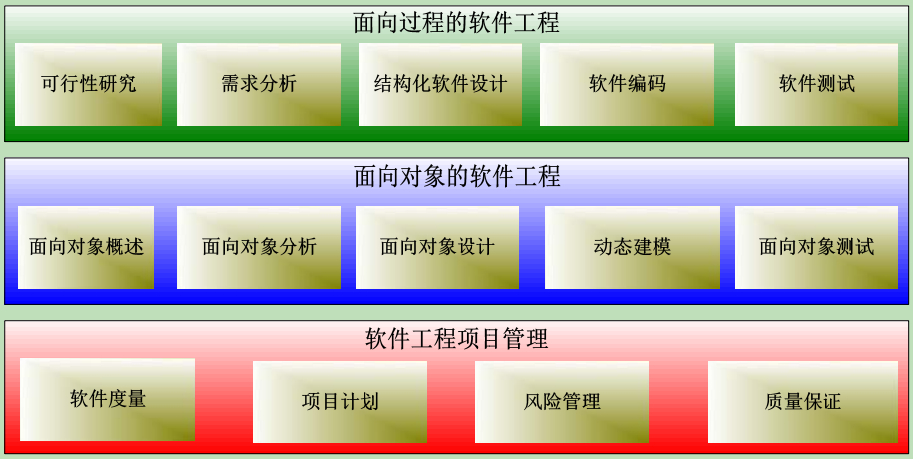
软件工程
第14讲 软件度量

贾西平
Email: jiaxp@126.com

课程主要内容



内容摘要

- 软件度量
- 面向规模的度量
- 面向功能的度量
- 软件质量的度量
- 集成度量数据

3

内容摘要

- 软件度量
- 面向规模的度量
- 面向功能的度量
- 软件质量的度量
- 集成度量数据

4

软件度量

- 目的
  评价软件产品质量，显化软件开发的生产率.
- 度量对象
  - 软件产品
  - 软件过程
  - 资源

5

度量方式

在软件工程中，度量的方式分为两种：

(1) 直接度量

- **对过程：度量投入的成本、完成的工作量等；**
- **对产品：度量产生的代码行数LOC、文档的页数、缺陷数/千代码行、软件执行速度等等。**

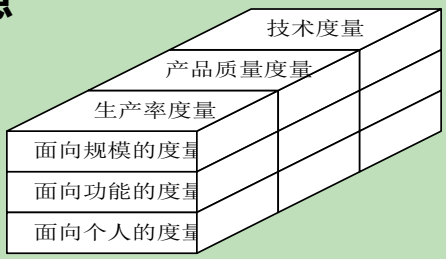
(2) 间接度量

- **通过对其他项目直接度量的结果进行分析，获取对本项目的间接度量结果。**
- **如：软件的正确性、效率、可靠性、可维护性、可用性等。**

6

软件度量分类1

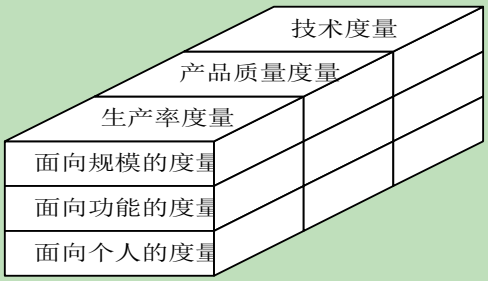
- **面向规模的度量**：收集与直接度量有关的软件工程输出信息和质量信息
- **面向功能的度量**：提供直接度量的尺度，关注程序的“功能性”和“实用性”
- **面向人的度量**：收集个人工作方式与效率方面的信息



7

软件度量分类2

- **软件生产率度量**：集中在软件工程过程的输出
- **产品质量度量**：反映产品满足用户需求的程度
- **技术度量**：集中于软件的一些技术特性(如复杂度、聚合度、耦合度等)



8

内容摘要

- 软件度量
- 面向规模的度量
- 面向功能的度量
- 软件质量的度量
- 集成度量数据

9

面向规模的度量

- 软件规模：指软件的大小(size)，一般用代码行度量
  - **优点：方便、直观**
  - **缺点：很大程度上取决于程序设计语言以及软件设计的质量**
- 测量出软件规模后可方便地度量其它软件属性，包括：

度量名	含义及表示
LOC或KLOC	代码行数或千行代码数
生产率P	P=LOC/E，E为开发的工作量(常用人月数表示)
每行代码平均成本C	C=S/LOC，S为总成本
文档代码比D	D=Pe/KLOC，其中Pe为文档页数
代码错误率EQR	EQR=N/KLOC，其中N为代码中错误数

10

表16.1　面向规模的度量数据样例

项目名称	代码行 (KLOC)	工作量 (人月)	成本 (千元)	文档 页数	错误 (发布前)	缺陷 (一年内)	人数
项目1	121	24	168	365	134	29	3
项目2	272	62	440	1224	321	86	5
项目3	202	43	314	1050	256	64	6

11

内容摘要

- 软件度量
- 面向规模的度量
- 面向功能的度量
- 软件质量的度量
- 集成度量数据

12

面向功能的度量

- 对软件和软件开发过程的一种间接度量
- 以未来软件应当满足的“功能性”、“实用性”作为度量的原始依据。
- “功能”不能直接度量，必须通过其他直接的度量来导出
- 实用性要求在度量过程中被用作计算权值

13

面向功能的度量

- 面向功能的度量基本单位是“功能点”(FP)。计算方法参见右图
- **用户输入数(EI)**：每个EI应当是面向不同应用的输入数据。输入数据与查询数据不同，应分别计数。
- **用户输出数(EO)**：各个EO应当是为用户提供的面向应用的输出数据。这里的输出是指报表、屏幕信息、错误提示等等，报表中的各个数据项不再分别计数。

测量参数	计数		加权因子 简单	平均	复杂	
用户输入数		*	3	4	6	=
用户输出数		*	4	5	7	=
用户查询数		*	3	4	6	=
内部文件数		*	7	10	15	=
外部接口数		*	5	7	10	=
总计数值						

14

面向功能的度量

- **用户查询(EQ)：EQ是一种联机输入，它引发软件以联机方式产生某种即时响应。每一种不同的查询都要计数。**
- **内部逻辑文件(ILF)：每一个逻辑主文件都应计数。**
  - **逻辑主文件，是指逻辑上的一组数据组合。可以是数据库的一部分，或者一个单独的文件。**
- **外部接口(EIF)：对所有用来将信息传送到另一个系统中或从另一系统接收数据的接口均应计数**

测量参数	计数		加权因子 简单	平均	复杂	
用户输入数		*	3	4	6	=
用户输出数		*	4	5	7	=
用户查询数		*	3	4	6	=
内部文件数		*	7	10	15	=
外部接口数		*	5	7	10	=
总计数值						

15

测量参数	计数		加权因子 简单	平均	复杂	
用户输入数		*	3	4	6	=
用户输出数		*	4	5	7	=
用户查询数		*	3	4	6	=
内部文件数		*	7	10	15	=
外部接口数		*	5	7	10	=
总计数值						

图16.2　功能点度量的计算

16

面向功能的度量

- **分析软件需求，搜集到上述五类数据之后，按照下式计算软件的功能点总数：**

  $FP = 总计数值 \times [\ 0.65+0.01 \times \sum F_i\ ]$

  - **“总计数值”是根据图16.2所计算出来的原始功能点数；**
  - **$\sum F_i$ 是按照表16.2计算出来的系统难度系数。i的取值从1～14。**
  - **∑是求和函数。**

17

表16.2　计算项目功能点数的难度校正系数值

F_i权值数据 难度因素F_i描述	没有影响	偶有影响	轻微影响	平均影响	较大影响	严重影响
1. 系统需要可靠的备份与复原吗	0	1	2	3	4	5
2. 需要数据通信吗	0	1	2	3	4	5
3. 有分布式处理功能吗	0	1	2	3	4	5
4. 性能很关键吗	0	1	2	3	4	5
5. 系统是否运行在既存的、高度实用化的操作系统环境中	0	1	2	3	4	5

18

表13.2　计算项目功能点数的难度校正系数值

6. 系统是否需要联机数据项	0	1	2	3	4	5
7. 联机数据项是否要多屏幕切换	0	1	2	3	4	5
8. 需要联机更新主文件吗	0	1	2	3	4	5
9. 输入/输出、文件、查询是否复杂	0	1	2	3	4	5
10. 内部处理复杂吗	0	1	2	3	4	5
11. 代码是否要设计成可复用的	0	1	2	3	4	5
12. 设计中需要包括转换和安装吗	0	1	2	3	4	5
13. 系统设计是否要支持多次安装	0	1	2	3	4	5
14. 应用设计是否要方便用户修改	0	1	2	3	4	5

19

面向功能的度量

- **求得FP值之后，通过简单的计算，结合一些历史数据，间接地度量出软件的生产率、质量和其他一些属性。**
  - **(1) 软件质量：每个功能点(FP)的缺陷数。**
  - **(2) 平均成本：每个功能点(FP)的成本。**
  - **(3) 文档规模：每个功能点(FP)的文档页数。**
  - **(4) 生产率：每个人月完成的功能点(FP)数。**

20

“特征点”度量方法

- 功能点度量方法充分考虑了“数据域”需求对功能点的影响，但是对“功能域”、“控制域”需求对功能点的影响没有考虑。
- “特征点”(FPs，Feature Points)度量方法就是一种扩充的功能点度量方法，适用于针对复杂计算软件进行功能度量。

21

“特征点”度量方法

特征点度量的计算增加了“算法数”这一度量参数；具体计算公式和功能点方法完全相同

测量参数	计数		加权因子		加权计数
用户输入数		*	4	=	
用户输出数		*	5	=	
用户查询数		*	4	=	
内部文件数		*	7	=	
外部接口数		*	7	=	
算法数		*	3	=	
总计数值				→	

图16.3 特征点度量的计算

22

其它方法

- Boeing提出另一个专门用来对实时系统和工程产品进行功能度量的功能点扩展方法。其特点是把数据域、功能域、行为域集成起来考虑，被称为3D功能点度量。
- LOC和FP都可以用作软件度量的基本单位，很多研究者试图将FP和LOC联系起来考虑。
  - 代码行和功能点度量之间的关系依赖于设计的质量和实现软件所用的程序设计语言。

23

在不同的程序设计语言中建造一个功能点平均所需要的代码行数的粗略估算：

表16.3　代码行-FP粗算

程序设计语言	LOC/FP 平均值	程序设计语言	LOC/FP 平均值
汇编语言	320	面向对象语言	30
C	128	第四代语言(4GLs)	20
Cobol	105	代码生成器	15
Fortran	105	电子表格	6
Pascal	90	图形语言(图标)	4
Ada	70		

24

内容摘要

- 软件度量
- 面向规模的度量
- 面向功能的度量
- 软件质量的度量
- 集成度量数据

25

软件质量

- 软件质量定义
  ISO/IEC 9126：与软件产品满足明确或隐含需求的能力有关的特征和特性的全体
- **软件质量包括**软件过程质量**(即过程能力)和**软件产品质量**两个范畴**
- 典型的软件质量模型：McCall模型、Boehm模型和ISO/IEC 9126质量模型

26

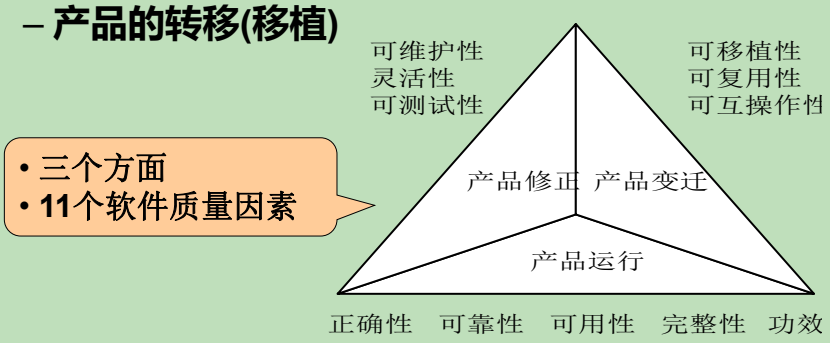
软件质量度量

- 产品交付之前：**程序复杂性度量、模块有效性度量，等等；**
- 软件交付之后：**软件中残存的差错数，系统的可维护性。**

27

McCall软件质量因素

- **可以从三个方面来评估软件质量**
  - **产品的运行(使用)**
  - **产品的修正(变更)**
  - **产品的转移(移植)**

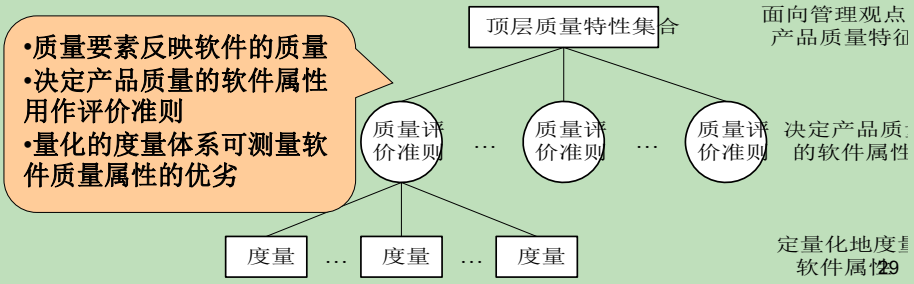


- 三个方面
- **11**个软件质量因素

28

McCall质量度量模型

- **通常采用分层的结构来定义**软件的质量模型**(如图)**
  - **顶层定义基本质量特征，如正确性、可靠性等;**
  - **质量特征由下一层的子特性来定义和度量；**
  - **子特性又可以由它的下级子特性描述和度量**



McCall软件质量要素定义1

- 正确性(correctness)：一个程序满足它的需求规约和实现客户任务目标的程度
- 可靠性(reliability)：一个程序期望以所需的精确度完成它的预期功能的程度
- 效率(efficiency)：一个程序完成其功能所需的计算资源和代码的数量
- 完整性(integrity)：对未授权人员访问软件或数据的可控制程度
- 可用性(usability)：学习、操作、准备输入和解释程序输出所需的工作量
- 可维护性(maintainability)：定位和修复程序中一个错误所需的工作量

30

McCall软件质量要素定义2

- 灵活性(flexibility)：修改一个运作的程序所需的工作量
- 可测试性(testability)：测试一个程序以确保它完成所期望的功能所需的工作量
- 可移植性(portability)：把一个程序从一个硬件和/或软件系统环境移植到另一个所需的工作量
- 可复用性(reusability)：一个程序(或一个程序的部分)可以在另外一个应用程序中复用的程度，与程序完成的功能的包装和范围相关
- 可互操作性(interoperability)：连接一个系统和另一个系统所需的工作量。

31

质量要素之间的关系

其中△表示正相关，▼表示负相关

关系要素 / 要素	正确性	可靠性	效率	完整性	可用性	可维护性	可测试性	灵活性	可移植性	可复用性	可互操作性
正确性											
可靠性	△										
效率											
完整性			▼								
可用性	△	△	▼	△							
可维护性	△	△	▼		△						
可测试性	△	△	▼		△	△					
灵活性	△	△	▼	▼	△	△	△				
可移植性			▼			△	△				
可复用性		▼	▼	▼		△	△	△	△		
可互操作性			▼	▼					△		

32

ISO软件质量模型

- 按照ISO/TC97/SC7/WG3/1985-1-30/N382，软件质量模型也由三层结构组成：
  - 高层(Top Level)：软件质量需求评价准则(SQRC)；
  - 中层(Mid Level)：软件质量设计评价准则(SQDC)；
  - 低层(Low Level)：软件质量度量评价准则(SQMC)。
- ISO组织认为，应对质量模型的高层建立国际标准，以便于在国际范围内推行软件质量管理(SQM)技术。中层的21项质量属性只作为推荐，不作为标准；而低层可由使用者自行确定。

33

ISO软件质量模型

- SQRC：功能性、可靠性、可维护性、效率、可使用性、可移植性
- SQDC：适合性、准确性、互用性、依从性、安全性、成熟性、容错性、可恢复性、可理解性、可学习性、可操作性、时间特性、资源特性、可分析性、可变更性、稳定性、可测试性、适应性、可安装性、一致性、可替换性。
- SQMC：软件开发组织根据自己的需要自行定义

34

质量度量方法

- 在许多软件质量度量方法中，使用最广泛的是事后度量或验收度量，包括对产品的正确性、可维护性、完整性和可用性的度量
- Cilb提出了这些质量指标的定义和度量方法

35

质量度量方法

- 正确性：软件是否能正确地执行所要求的功能。
  - 用缺陷数/KLOC来度量；
  - 产品交付后，根据标准的时间周期(一般为一年)，按照用户反馈报告中的缺陷数进行度量
- 可维护性：利用平均变更时间MTTC(Mean Time To Change)间接度量软件的可维护性

36

质量度量方法

- 完整性：这个属性反映软件系统抗拒针对它的安全攻击(事故或人为)的能力。
  - 完整性 = ∑(1 - 危险性×(1 - 安全性))
  - 危险性指一定时间内特定攻击发生的概率
  - 安全性是排除特定类型攻击的概率
  - 两者都可以从估算或历史数据中得出。
- 可用性：即“用户友好性”。从四个角度度量：
  - 学习应用软件所花费的代价；
  - 为达到有效使用软件所花费的时间；
  - 使用软件带来的生产率净增值；
  - 通过问卷调查得到的用户的主观评价

37

内容摘要

- 软件度量
- 面向规模的度量
- 面向功能的度量
- 软件质量的度量
- 集成度量数据

38

在软件过程中集成度量数据

- 为了持续发展，必须通过对能力、效率、质量的度量进行度量数据的收集、计算与分析，并且将历史度量数据、当前度量数据进行集成，建立工程过程的“度量基线”
- 利用基线来评估各类改进工作的效用，估算新项目的规模、工作量、预测成本、质量指标

39

度量基线

- 软件项目的管理者经常要考虑的问题：
  - “以前类似项目的估算结果是什么？”
  - “影响软件质量的主要问题在哪里？”
  - “本组织的实际生产效率究竟有多高？”
  - ……
- 搜集以前的度量数据，并进行集成整合，建立项目的“度量基线”数据库，解决上述问题提供参考信息.
- 度量数据的收集使一个软件开发组织能调整自身的软件工程过程，排除那些对软件开发有重大影响的错误产生的根源
- 度量基线由以往的软件开发项目收集到的度量数据构成。也称“数字化的经验”。

40

基线数据的属性

为了有助于估算、计划和质量控制，纳入基线的数据应当具有如下的属性：

(1) 数据必须精确、合理。

(2) 数据应来自尽可能多的项目。

(3) 对于所有称为数据搜集对象的项目，对“代码行”、“功能点”等基本度量单位的解释应当一致

(4) 在应用基线数据时，必须保证类型的匹配。

- **比如，来自批处理项目的基线数据就不能用来指导针对实时项目的估算。**

- **建立基线，实际上就是集成了已有的度量数据。**

41

度量数据的收集

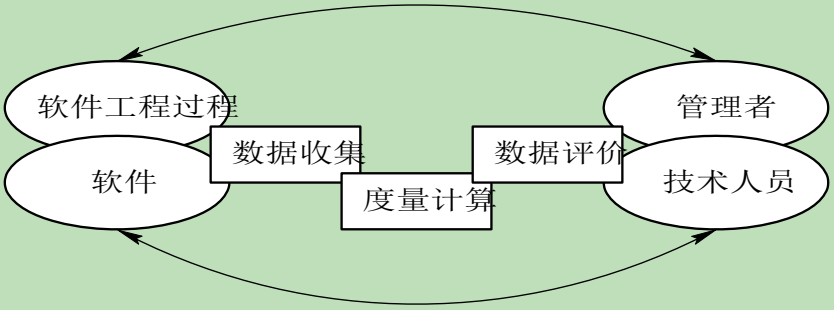


图16.6　度量收集的过程

42

表16.4 面向规模的度量——生产率和成本的度量数据项

数据项	单位	样本数据
项目名	字母数字	Proj_1
输出	KLOC/人月	0.905
全部维护代码的成本	元/KLOC	22 514
除去复用代码的成本	元/KLOC	24 028
经历时间	月/KLOC	1.0
文档	页/KLOC	30
文档	页/人月	10
文档	元/页	739

43

表16.5 面向规模的度量——质量的度量数据项

数据项	单 位	样本数据
错误数	错误数/KLOC	13.0
错误成本	元/错误	376
出错维护/总维护量	比率	0.36
修改维护/总维护量	比率	0.64
维护工作量/开发工作量	比率	1.15

44

表16.6　面向功能的度量

名 称	描　述	单　位	样本数据
功能点数计算	未校正的功能点		378
	总影响度		43
	复杂性校正值		1.08
	功能点		408
生产率和成本度量	项目名	字母数字	Proj#1
	输出	FP/人月	11.1
	成本	元/FP	700
	经历时间	月/FP	31.4
	文档	页/FP	0.9
质量度量	错误数	错误数/ FP	0.064
	出错维护工作量	人日/FP	0.817
	修改维护工作量	人日/FP	1.472
功能性	程序规模	FP/程序	408
	单位规模的功能	维护的FP/KLOC	32　45

敏捷项目管理

- 敏捷开发之Scrum
  https://www.bilibili.com/video/BV1Q741157ve?p=30
- 用户故事与估算
  https://www.bilibili.com/video/BV1Q741157ve?p=31
- Tower工具介绍（1）
  https://www.bilibili.com/video/BV1Q741157ve?p=32
- Tower工具介绍（2）
  https://www.bilibili.com/video/BV1Q741157ve?p=33
- 配置管理
  https://www.bilibili.com/video/BV1Q741157ve?p=34
- 配置管理工具Git
  https://www.bilibili.com/video/BV1Q741157ve?p=35

46

谢谢大家！
感谢清华大学刘强老师的视频资源！

08:43