SOFTWARE PROJECT MANAGEMENT (11)

Du Yugen

ygdu@sei.ecnu.edu.cn

Chapter software quality
软件质量管理

OBJECTIVES

When you have completed this chapter you will be able to:

Explain the importance of software quality to software users and developers;

Define the qualities of good software;

Design methods of measuring the required qualities of software;

Monitor the quality of processes in a software project;

Use external quality standards to ensure the quality of software acquired from an outside supplier;

Develop systems using procedures that will increase their quality.

软件质量的重要性

- **尽管所有的商品或服务都存在质量问题，但是软件的特殊性，特别是其复杂性和不可见性，使其更为重要**
  - **软件危险性的增加：企业越来越依赖软件**
  - **软件的无形性：很难知道项目中的特定任务是否完全满足**
  - **软件开发过程中错误积累**

软件质量

- **对软件系统可以从三方面描述：**
  - **描述系统如何工作的功能描述**
  - **功能如何提供的质量描述**
  - **花费在系统上的资源描述**
- **问题：学院工资系统准备选用商品化软件。请定义选择原则？**

软件质量

- **答案：**
  - **对用户需求进行调查，以明确不同用户的不同需求集合**
  - **将需求分成若干组，并定义质量和其它属性，如质量，可用性，效率，灵活性等**
  - **某些需求具有绝对特性。例如，应用程序必须能够记录职员的最大条数。这种需求必须满足。**
  - **某些需求具有相对特性。某些相对需求比其它需求更重要**
  - **市场上调查一系列待选的软件。**
  - **度量软件质量的方法需要确定**
  - **某些软件的某些方面有不足，但是其它方面可能给以补偿，因而定义某些方法来完策划能够选择。**

软件质量

- **软件质量可以包括：**
  - **运行质量**
    - **正确性**
    - **可靠性**
    - **集成性**
    - **可用性**
  - **修改质量**
    - **可维护性**
    - **可测试性**
    - **灵活性**
  - **转换质量**
    - **可移植性**
    - **可重用性**
    - **互操作性**

问题：请指出质量特性中哪些是无关的，互补的和冲突的？

无关的：如可用性和可重用性

互补的：如灵活性与可维护性

冲突的：由于考虑了某一类型的平台可能效率高，但是移植性差

软件质量

Table 12.1 *Software quality criteria*

Quality factor	*Software quality criteria*
Correctness	traceability, consistency, completeness
Reliability	error tolerance, consistency, accuracy, simplicity
Efficiency	execution efficiency, storage efficiency
Integrity	access control, access audit
Usability	operability, training, communicativeness, input/output volume, input/output rate
Maintainability	consistency, simplicity, conciseness, modularity, self-descriptiveness
Testability	simplicity, modularity, instrumentation, self-descriptiveness
Flexibility	modularity, generality, expandability, self-descriptiveness
Portability	modularity, self-descriptiveness, machine independence, software system independence
Reusability	generality, modularity, software system independence, machine independence, self-descriptiveness
Interoperability	modularity, communications commonality, data commonality

问题：同一准则出现在不同的质量要素中说明了什么？

说明这些质量要素是互补的。

质量度量

- 对于每一准则，必须定义一个和多个度量标准来完成评估工作
- 任何相对度量需要将度量的单元与环境中发生的最大可能性相联系，例如程序中最大的错误数就需要和程序的大小相联系。
- 在某些时候，我们可以直接度量质量，在另外一些时候，我们度量的是质量的表现。
- 软件的使用者关注的是**质量因素**，而开发者需要关注的是**质量准则**。

质量度量

- 为了度量，需要对每一质量规定：
  - 度量的单元
  - 测试的范围
  - 最差的可接受的值
  - 计划达到的值
  - 当前可达到的最佳的值
  - 目前的值
- 问题：针对字处理系统，举出一个质量度量的例子。

质量度量

- 质量：易学习性
- 定义：新手学会使用软件生成一份标准文档的时间
- 度量的单元：小时
- 测试：首先对新手进行调查以确定他们的字处理软件的使用经验，然后给他们一台机器，一套软件，训练手册和安装文档。然后测试他们学会生成一份文档的时间
- 最差：4小时
- 计划：2小时
- 最好：1小时
- 目前：4小时

质量度量

- **为了产生一份质量描述文档，经常需要将质量准则进一步细分。例如可用性下面的易理解性，可以分成：菜单结构的易理解性，特别是某项执行功能的命令要容易找到，其它方面还包括错误消息的提供，帮助信息的提供等**

ISO9126

- 目前，不少人提出了不同的软件质量特性表示方法。但是缺少一个公共的标准。例如可维护性可以指错误可以迅速确定并被修改，也可以指软件能够很容易地被修改。
- 制定于1991年的ISO9126标准就是处理软件质量问题的。这份13页的标准为制定进一步的标准奠定了基础。

ISO9126

- ISO9126规定软件质量可以从6个特性来评价：
  - Functionality:与一组功能及其指定的性质有关的一组属性
  - Reliability:在规定的时间和条件下，软件维持其性能水平的能力有关的一组属性
  - Usability:与一组规定或潜在用户为使用软件所需作的努力和对这样的使用所作的评价有关的一组属性
  - Efficiency:在规定的条件下，软件性能水平与所用的资源量之间关系相关的一组属性
  - Maintainability:与进行指定的修改所需的努力有关的一组属性
  - Portability:与软件可从某一环境转移到另一环境的能力有关的一组属性

ISO9126

- **ISO9126对每一特性规定了子特性**
  - **功能性：**

Characteristic	*Sub-characteristics*
Functionality	Suitability
	Accuracy
	Interoperability
	Compliance
	Security

- **Compliance(符合性)指的是软件符合应用标准或法律的程度。**
- **Interoperability（互操作性）指的是软件与其它系统交互的能力**

ISO9126

- 可靠性

Characteristic	*Sub-characteristics*
Reliability	Maturity Fault tolerance Recoverability

  - Maturity（成熟性）指的是由于软件产品的问题而出现故障的频率

- 可用性

Characteristic	*Sub-characteristics*
Usability	Understandability Learnability Operability

ISO9126

- 效率和可维护性

Characteristic	*Sub-characteristics*
Efficiency	Time behaviour Resource behaviour
Maintainability	Analysability Changeability Stability Testability

  - Stability(稳定性)不是指软件从不变化，而是指软件由于预想不到的原因而要修改的风险很低。
- 可移植性

Characteristic	*Sub-characteristic*
Portability	Adaptability Installability Conformance Replaceability

  - Conformance(一致性):与Compliance(符合性)不一样，它与可移植性有关。例如使用标准的编程语言就是一种遵循性。

ISO9126

- ISO9126也提供了使用这些质量特性的指南。
- 对于不同的产品，各种质量特性的重要程度是各不相同的。一旦软件产品的需求建立起来后，就要进行下列步骤：
  - 质量度量标准的选择：ISO9126没有给出具体的方法
  - 排序水平定义：度量的结果需要映射成等级以确定需求满足的程度。

response time (seconds)	*quality score*
< 2	5
2-3	4
4-5	3
6-7	2
8-9	1
>9	0

ISO9126

- 评价准则定义
  - 对每一个特性进行评价后，需要对整个产品有一个综合的质量评价
  - ISO9126没有给出具体的方法。
  - 推荐方法：首先根据产品的特征确定必须的特性，如果产品在这些特性方面没有满足，则不能采用，对于那些不是最为关键的特性可以采用下表的方法进行综合计算：

Table 12.2 *Quality rating scores*

product quality	*importance rating (a)*	*product A*		*product B*	
		quality score (b)	*weighted score (a × b)*	*quality score (c)*	*weighted score (a × c)*
usability	3	1	3	3	9
efficiency	4	2	8	2	8
maintainability	2	3	6	1	2
overall			17		19

软件质量度量方法

- 本节所讲的是某些质量特性的建议的度量方法，对于具体

Each day the system should be available from 18.00 – 8.00 hours, = 10 hours.
Over four weeks that should be 10 × 5 × 4 hours = 200 hours.
It was unavailable for one day, i.e. 10 hours.
It was unavailable until 10.00 on two other days, = 4 hours.
The hours available were therefore 200 – 10 – 4 = 186 hours.
Availability would therefore be 186/200 × 100 = 93%

Assuming that three failures are counted, *mean time between failures* would be 186/3 = 62 hours.

- **某一系统安装后，一般在星期一到星期五中8:00到6：00使用，四星期中，系统因为硬盘问题有一整天不可用。在接下来的另外两天由于头天晚上的批处理运行的问题每次直到早晨10点才能使用，请计算Availability和MTBF。**

软件质量度量方法

- 可维护性
  - 该特性与灵活性紧密关联。灵活性是系统修改的容易程度。可维护性可以用灵活性加上可诊断性来度量。诊断性可以用诊断一个错误的平均时间来度量。
- 可扩展性
  - 将新的特征加到现有系统中的效率占从头开始开发一个新系统的效率的比。
  - 某公司开发一个包含5000SLOC的系统花费了400个人日。对系统添加一个新功能增加了100SLOC花费20个人日，问扩展性为多少？
  - $40\% = \frac{100/20}{5000/400} * 100\%$

产品和过程质量管理

- 上述度量都是在产品出来后进行的，这种事后度量也许太晚了。
- 在过程的各个阶段可能都会引入错误，这种错误将传递到后续的阶段。因而需要在过程进行中尽心仔细的检查。
  - Entry Requirements: 在活动开始前，需要准备好的条件。如在开始前，要准备好测试数据和期待结果。
  - Implementation Requirements:过程如何进行。如在测试中，当发现一个错误并加以改进后，所有的测试必须重新进行。
  - Exit Requirements:在一个活动结束前必须满足的条件。例如测试阶段结束的条件是所有的错误都被更正并且不能够再发现任何错误。
- 问题：在什么情况下，前面一个活动的结束条件不是后面一个活动的进入条件。

产品和过程质量管理

- 答案：在某些场合，某一活动可以在它前面一个活动完全结束前开始。在这种情况下，后面活动的进入条件可以与前面一个活动的退出条件不一致。例如，某些软件模块的界面还没有最后调整好前，就可以先在硬件平台上来测试性能。
- 问题：请为学院工资系统的代码编写活动确定进入条件和退出条件?

提高软件质量的途径

- **Increasing visibility(增加可见性):例如“egoless programming”编程员相互浏览对方代码。**
- **Procedure Structure(过程结构)**
- **Checking Intermediate Stages(检查中间环节）：将错误消灭在萌芽状态**

提高软件质量的途径

- **Inspection(检查):**
  - **通过将完成的工作交付给多个合作者检查，然后召开会议进行讨论如何修改。该方法可以:**
    - **很容易地发现表面错误**
    - **激励编程人员编写出结构更好，更清晰地代码，因为他知道否则别的人将会批评他**
    - **提高团队精神**

提高软件质量的途径

- **IBM建立了一套更为正式和结构化的检查过程，称为Fangan检查**
  - **对所有主要的交付物都进行检查**
  - **所有的错误都需加以注意，而不仅是逻辑的和功能的错误**
  - **检查可以由在所有层次的人员（除了最上层的人员外）进行检查**
  - **检查采用预定义的步骤进行**
  - **检查会议不超过两小时**
  - **检查由一个经过训练的“moderator”来领导**
  - **其它参与者也有定义的角色，例如一个人员担任记录员，另一担任阅读者等**
  - **采用Checklist来帮助检查过程的实施**
  - **检查材料时采用100行一小时的速度**
  - **采用统计方法来对检查过程的有效性进行监控**

结构化编程和净室软件开发

- 在二十世纪60年代，软件变得越来越复杂而人记忆细节的能力是有限的。因此，不可能去对软件的所有部分进行完全测试。测试所能做的是证明错误的出现，而不是没有错误。Dijkstra建议保证软件代码正确性的唯一方法是对代码重新审视。
- 复杂系统可以分成子部件，为了使这种分解工作正常，每个部件必须自包含，同时只有一个进入点和一个退出点。
- 该观点进一步发展为IBM的净室软件开发，该方法将软件开发分为三个小组：
  - 分析小组
  - 开发小组
  - 校验小组
  - 系统采用增量式方式开发，每个团队的产出都必须满足用户的需要。开发小组不进行编译而代之以用数学方法来证明。校验小组不断的测试直到满足特定的统计水平。

形式化方法

- **数学方法采用的是对每一个过程定义前提条件和后置条件。**
- **前提条件定义了在处理前允许的状态，后置条件定义了处理后的状态。由于数学方法是精确的，因此可以保证其正确性。**

软件质量循环

- 尽管测试方法和Fagan检查能够帮助发现问题，但是相同类型的错误还是一犯再犯。
- 通过发现错误的来源，应该可以减少相同的错误。
- 因而，开发人员可以在质量循环中参与错误来源的查找，该过程称为软件质量圈（Software quality circles, SWQC).
- 质量圈由四个到十个自愿者构成，每个星期采用一定时间如一小时寻找、分析和处理他们工作中的问题。

软件质量圈

- 识别一系列问题
- 选择一个问题加以处理
- 将问题分析清楚
- 分析原因并加以评价
- 分析解决方案并加以评价
- 决定采用某一解决方案
- 开发一个实施计划
- 将计划交给管理部门
- 实施计划
- 监控计划
- 考虑解决方案更广泛的应用
- 重新选择问题
- 问题：软件质量循环与一般的检查过程有何区别
- 一个是面向所有过程，一个是面向某一产品

GQM方法

- 过程改善可以采用定量测量技术来实现。
- GQM(Goal/Question/Metric)方法：
  - 首先需要定义一个目标，该目标可能是评估是否一个新的编程语言能够提高开发者的效率。
  - 为了完成目标，可以提出许多问题，例如对于上述目标，问题可以为：
    - 目前开发者编程的速度有多快?
    - 开发者利用新的编程语言有多快?
    - 目前的软件质量如何?
    - 利用新语言软件质量如何?
  - 对每一个问题，可以定义度量方法，例如来度量开发速度可以根据开发每一功能点花费的时间来进行。