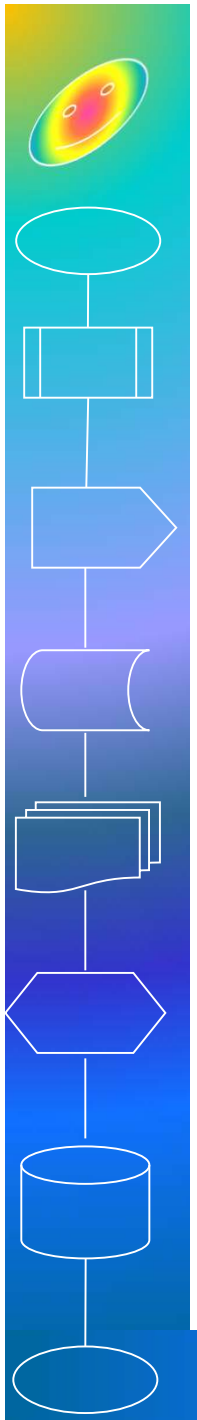


第4章 软件质量

软件质量—观点与模型

目录

4.1软件质量定义

- 4.1.1 程序的类型划分
- 4.1.2 质量定义

4.1.1 程序的类型划分

- Lehman和Belady从1972年开始注意并研究计算机程序进化的动力学规律，把程序分为三种类型：
  - **S-型程序**是指能够用形式化方法进行说明的程序。
  - **P-型程序**不能够完全说明清楚，需要通过多次迭代开发过程才可以发现其应当如何工作。
  - **E-型程序**是嵌入现实世界中，或者是其一部分，因此系统的改变必然要求该程序与其环境一起进化。
- 只有**S-型程序**是能够满足先前预定义的规格说明。程序的正确性具有唯一仲裁条件。

- **P-型程序**是那些用来解决板上钉钉的问题的程序，其需求是固定的，例如，一元二次方程的求根程序的算法是固定的。
- **E-型程序**解决和实现真实世界中的应用问题。执行的结果，例如信息，必须转换为人类可读的形式。程序特征归纳为附加的或可控制的要求，并且一起决定了程序的可接受性、价值和满意程度----这些就是评价的E-型程序的质量准则。

- **P-型程序介于S和E-型程序之间，但没有明确的界限。决定S和E类型是程序能否被形式化的描述。**

Yasuda的观点

- Yasuda等人提出的，依据软件产业在全球竞争条件下的质量要求来划分软件类型：
  - **ZD类型的软件：**
  - **CS类型的软件：**
  - **SV类型的软件：**
- **ZD类型的软件：**
  - 以高可靠性和高可信性为目标，主要用于军事和关系到国民经济和公共安全的系统中的软件。这种软件的质量必须是充分可度可信赖的。
  - 例如，航天、航空、高铁、医疗设备等中的软件。

Yasuda的观点

- **CS类型的软件：**
  - 以客户满意为目标。主要是公司或企业开发出的系统和产品，其质量是以客户满意地使用为准则的。例如企业用的信息管理系统，办公自动化系统等。
  - 这些面向客户定制的系统必须满足客户需要。并伴随着客户业务和个性化需求的变更，而不断地进化软件。
- **SV类型的软件：**
  - 以提高服务竞争力和市场份额。这类产品是面向普通的、大众消费的软件产品。衡量其质量的标准往往以用户量的多少为基准。
  - 例如，微软的Office、SQLSever。近年来谷歌的Android手机操作系统以其开源和免费的策略，短期内大跨了一些也许是传统质量很好的产品，逼着类同的产品退出市场，人们就会认为这类程序的质量是好的。
  - 微软的“同步-稳定”开发方法适合于这类软件开发。

质量特征分解

软件用途 / 问题明确程度	ZD(高可信和安全)	CS(客户满意)	SV(大众消费市场竞争)
S(特征唯一，可证明其正确性)	质量特征分解		
P(问题明确，可以判断软件正确)			
E(人为判断可接受性)			

4.1.2 质量定义

- 对于**S-型程序**，只需讨论程序的正确性证明，不用讨论软件的工程质量，因为用计算机可以判断出程序的正确与否。

- 软件质量的主要议题是针对**E-和P-型程序**的讨论，由于用户的要求不能用严格的数学语言描述清楚，因此必须评价软件能否满足对用户服务的要求，并通过这种手段评价和提高软件产品和系统的质量。

- 质量(quality)定义为：一个产品和服务满足所说明的和隐含的需求能力的特征和特性的总和(ISO1986）。

- （GB/T-11457-2006）将软件质量定义为:

  “软件产品中能满足给定需要的性质和特性的总体；软件具有所期望的各种属性的组合程度；顾客或用户觉得软件满足其综合期望的程度；软件在使用中满足顾客预期要求的程度。”

4.2 质量观点

- 4.2.1 Garvin的质量分类
- 4.2.2 Brra的质量侧面
- 4.2.3 项目经理们对质量看法

4.2.1 Garvin的质量分类

- 质量“是一个复杂的和多侧面的概念”。从五个不同视角对质量进行解释:
  - 4.2.1.1 先验质量观点
  - 4.2.1.2 基于产品的质量观点
  - 4.2.1.3 基于用户的质量观点
  - 4.2.1.4 基于制造的质量观点
  - 4.2.1.5 基于经济的质量观点

4.2.1.1 先验质量观点

- 质量是“先天卓越”的同义词。
  - 这种观点的支持者认为质量是不可定义的，仅仅可以通过经验获得一些简单的、不可分析的特征。
- 正如“漂亮”只是帕拉图式的含义，无法用术语准确定义出来一样。
- 这种观点过于悲观，对于工业化的产品和生产没有太大的帮助。

4.2.1.2 基于产品的质量观点

- 产品的质量是一个可精确和可测量的变量。质量的差异反应了某些成分或属性上的量化差异。
  - 例如，高质量的冰淇林就是指乳脂含量高，高级地毯每平方厘米的结数一定比一般地毯的多
- 对这种模型的理解有两种结果：
  - 首先，高质量必然需要高成本，由于质量反应了产品属性的量化，每个属性的提高会增加成本，越是高质量的产品就越昂贵。
  - 其次，质量是产品的内在特征，而不是产品本身，质量反应了产品是否存在可测量属性，而这些属性可以被客观地评价。

4.2.1.3 基于用户的质量观点

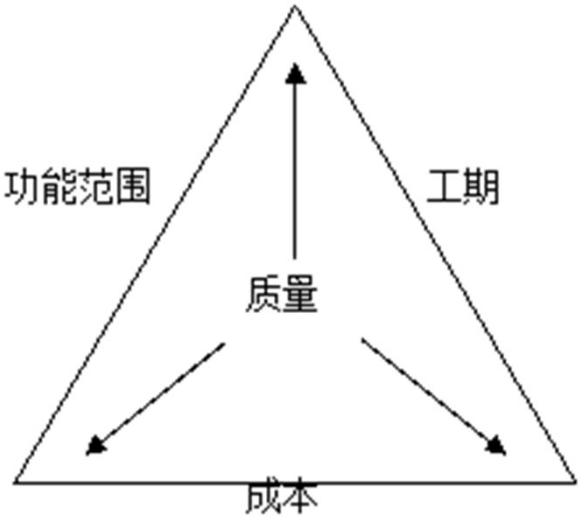
- 质量是：“用户眼里所看到的”。
- “最大化地满足不同客户需要”的质量方程。
- 可能会有不同的解释：
  - 一种解释是满足的客户数量大，
  - 另一种解释是满足客户独特的要求。
  - 例如，耐用性(durability)是质量的重要元素。
    - 长期耐用的产品要比容易用坏的产品质量高。
    - 这种观点在西方社会从19世纪后期开始发生转变，社会的风气是“耐用意味着贫穷和落后”。

4.2.1.4 基于制造的质量观点

- 认为“质量是产品与需求的符合程度”，而设计决定了需求。
  - 一旦设计或产品规格说明定义下来，产品的偏差就意味着产品规格说明有没有得到全部满足。
  - 良好的做法是“第一次就作对”，避免工程和制造的返工。
  - 从设计方面看，需要强调可靠性工程（参见第2章和第10章）。
  - 从制造方面看，强调基于统计的质量控制（参见第2章和16章）。

4.2.1.5 基于经济的质量观点

- 质量是成本和价格的反应。因此合格的产品是在可接受的价格内的性能，或者是可接受的成本内的符合性。
  - “物美价廉”！
- 软件质量不仅受到成本费用的约束，也受到软件交付时间的约束，以及对软件需求的约束。
- 客户期望的是成本受限，交付时间受限，功能(范围)尽可能全面，质量(包括性能等)尽可能好。



4.2.2 Brra的质量侧面

- Braa和Ogrim针对ISO的质量保证过程(基于制造的质量观点)。认为应当从技术、使用、美学、品牌和组织等5个侧面进行分析：
  - 4.2.2.1 技术质量侧面
  - 4.2.2.2 使用质量侧面
  - 4.2.2.3 美学质量侧面
  - 4.2.2.4 品牌质量侧面
  - 4.2.2.5 组织质量侧面

4.2.2.1 技术质量侧面

- 单纯地从技术角度看，技术质量表示了系统的结构和性能。
- 基于计算机系统的质量是以功能性为基础的，即，计算机必须执行所期望的操作。
- 因此除了计算机系统具有良好的技术之外，更重要是适合于其工作任务。
  - 应用软件层面的

4.2.2.2 使用质量侧面

- 从使用的侧面看，质量反应了用户的经验。
- “使用质量”很难说清楚，需要通过实验设计出未来使用情景和模型，例如原型方法，表达出系统未来的需求和使用场景。
- 对系统的使用、学习和研究是获取需求规格说明和进行设计的重要因素。

4.2.2.3 美学质量侧面

- 在汽车、建筑等工业中，从美学(aesthetic)的角度评价质量。
- 美学观点也可以用于数学定理的表达和证明中。在软件开发中，我们往往会忽略美学。
  - 例如，“结构化程序”就要比“非结构化程序”美—参看第12章
- 把美学质量的“优雅(elegance)”作为一个质量评估准则。
  - 为了把“优雅”表达的更直观，可以通过提问的方式进行评价，例如系统设计是否过度复杂？系统从美学角度是否让人感到很舒服？等等。

4.2.2.4 品牌质量侧面

- 计算机系统不仅仅是产品，而且也是组织和企业的品牌。
- 许多企业把信息管理系统作为一种品牌，即，一个良好的企业是否能完整地使用信息。
  - 例如，政府部门的网站，

4.2.2.5 组织质量侧面

- 一个组织部署和使用了计算机系统，是否就意味着组织质量的提高了哪？
  - 不同的用户群体会有不同的回答，并且可能是相互矛盾的。
  - 例如，人员资源管理系统和福利管理系统，对于人力资源管理部门很好的系统，但是普通员工们并不会认为这是一个好系统，可能会引起员工们对系统的反感，甚至会降低企业的工作效率。

4.2.3 项目经理们对质量看法

- 开发方的项目经理们：
  - 认为软件质量是“软件能很好地服务用户，完成所希望的功能，并在需要时候可以使用”。
  - 面临着财政和时间的约束，这些约束条件极大地影响着软件质量目标，并可能由此牺牲测试（的等级、范围、详细程度等）、代码审查、以及文档编写等工作。这些工作会影响软件质量。
- 采购方的经理们：
  - 往往面临着公司业务的压力和高层管理者的压力。
  - 由于软件生产过程的可见性差和难于预测，高层管理者往往会给出理想的进度和成本目标。与软件可靠性、可用性等指标相比，进度和成本指标是最直接可见的。
  - 因此，开发方的管理者往往会迫使采购方项目经理们的压力，不断地追赶进度和压低成本，牺牲那些可见性差和难以度量的质量指标。

4.3 软件质量模型的归纳

- Wong把软件质量的研究归结为两个学派：
  - 一个是产品质量学派，认为要清楚地定义、测量和改进质量，就必须测量影响软件质量每个特性。
    - 例如，数据库管理系统(DBMS)的测试集：性能测试集和功能测试集合—标准测试集，评价Oracle, DB2，SQL-Server等质量。
  - 另一个是过程质量学派，认为最终产品质量取决于生产过程的质量，因此，软件质量来源于软件工程过程的质量。
    - 例如，给定任意一个应用软件，就没有标准的测试集合。那么，你如何评价这个应用软件的质量？
    - 通过评价其生产(开发)过程，判断最终提交的应用软件是否好？

4.3 软件质量模型的归纳

- 产品模型：
  - Boehm模型、COAUAMO模型、McCall模型、Marine的SQM模型、ISO9126、SQUID和Dormer模型属于产品质量模型。最典型是ISO-9126标准，将在4.4和4.5节讨论。
- 过程模型：
  - 过程模型包括ISO9000-3、CMM和CMMI、SPICE(ISO/IEC15504标准等，我们将在第20章讨论CMM/CMMI。
  - 后续的“软件项目管理”、“软件质量保证”、“软件过程改进”基于的就是这种观点。

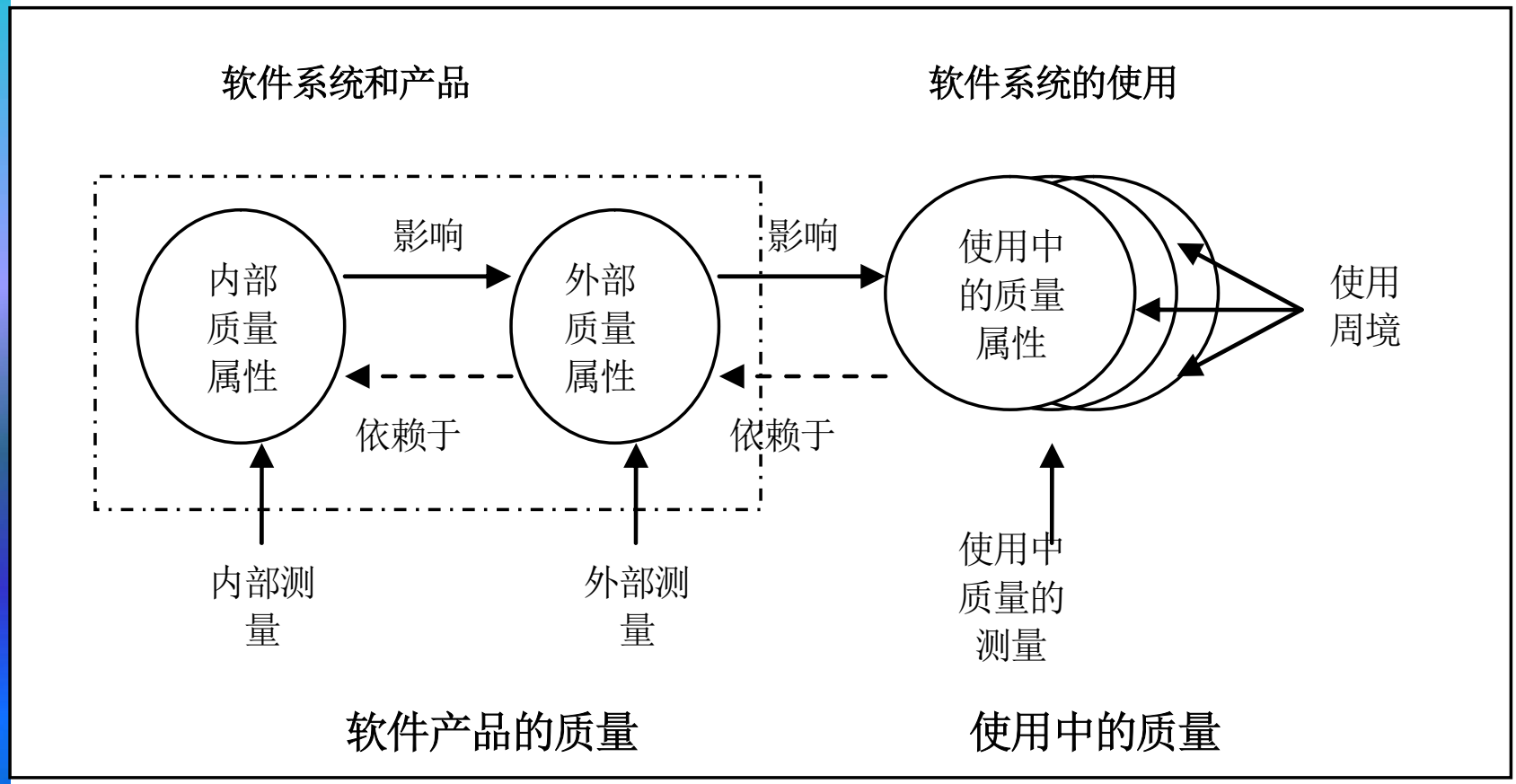
4.4 ISO9126的质量定义

- 4.4.1软件产品质量模型
- 4.4.2产品质量属性分解
  - 4.4.2.1 功能性
  - 4.4.2.2 可靠性
  - 4.4.2.3 易用性
  - 4.4.2.4 效率
  - 4.4.2.5 维护性
  - 4.4.2.6可移植性
- 4.4.3部件和服务项的质量评价

4.4.1软件产品质量模型

- 从产品质量观点出发，产品质量模型将产品质量分解为许多质量因素(特征、准则或特性)，最の层的是可以测量的度量元，或指示器。
- 从用户的角度，使用者最为关注的是系统的使用和运行是否会给其带来效益。
  - 质量是用户对软件系统的最基本要求。
- 软件系统的内部质量、外部质量、以及使用中的质量，三者之间具有密切的关联关系。

软件使用、外部和内部的质量关系



- 用户的质量要求包括在指定的使用周境(contexts of use)对使用质量的需求。
  - 使用周境是指使用软件产品和系统的用户、任务、设备（硬件，软件和材料）及物理和社会环境。
- 软件使用中的系统质量主要反应系统在运行期间的质量，这也是用户和运维人员最关心的质量特征。
  - 例如，功能性(functionality)、密安性(security)、可使用性(Availability)、易用性(usability)和互操作性(interoperability)等。

- 软件系统的使用中质量依赖于软件的外在所表现出来的质量。这些质量是软件设计人员所关心。
  - 例如，是否好用(usability)，系统是否容易与其他系统集成--可集成性(Integrability)，能否被充分测试---可测试性(Testability)，能否适应于多种机器平台---可移植性(Portability)等。

- 更进一步，软件系统的外部质量取决于内部质量，这是程序员和测试人员所关心的质量。
  - 例如，代码是否符合标准要求---代码一致性，代码是否有安全漏洞---代码安全，代码是否会有除零错误、浮点和定点溢出等等。

4.4.2产品质量属性分解

ISO9126

一级特性	功能性	可靠性	易用性	效率	维护性	可移植性
二级子特性	适合性	成熟性	易理解性	时间特性	易分析性	易安装性
	准确性	容错性	易学性	资源特性	易更改性	共存性
	互操作性	易恢复性	吸引性		稳定性	易替换性
	密安性				易测试性	

$$TQ = A1 \times 功能 + A2 \times 可靠性 + A3 \times 易用性 + A4 \times 效率 + A5 \times 维护性 + A6 \times 可移植性$$

4.4.2.1 功能性

- 1）适合性：软件产品为指定的任务和用户要求目标提供一组合适的功能的能力。例如，面向任务的由子功能构成的功能组合是否合适？数据库表的容量是否合适？等等。
- 2）准确性：软件产品提供具有所需精度的正确或相符的结果或效果的能力。
- 3）互操作性：软件产品与一个或更多的规定系统进行交互的能力。
- 4）密安性：许多时候人们称为信息安全性，而本书中称为密安性，以区别于安全性(safety)。

4.4.2.2 可靠性

- 1）成熟性：软件产品为避免由软件中故障而导致失效的能力。即，评价软件产品是否成熟，例如一项软件产品的用户数越多，总的运行时间越长，遍历的路径情况就会越全面，软件越成熟。

- 2）容错性：在软件出现故障或者违反其指定接口的情况下，软件产品维持规定的性能级别的能力。

- 3）易恢复性：在发生故障的情况下，软件产品重建规定的能力级别并恢复受直接影响的数据的能力。
  - 一般来讲，在发生故障后会软件产品会在某些时间停机，修复后会恢复运行。可以用故障发生到恢复的这段时间的长度来度量和评价易恢复程度。

4.4.2.3易用性

- 1）易理解性：用户能否很好地理解软件是否合适以及如何能将使用软件完成特定的任务和使用条件的能力。
  - 软件的使用手册、在线帮助、界面等，以及用户对软件的第一印象在很大程度上决定着软件是否容易理解。
- 2）易学性：软件产品使用户能学习其应用的能力。
- 3）易操作性：用户操作和控制软件的能力。
- 4）吸引性：软件产品吸引用户的能力。
  - 这涉及到软件旨在使自身对用户更具吸引力的属性，例如颜色的使用和图形化设计的特征

4.4.2.4效率

- 1）时间特性：在规定条件下，软件产品执行其功能时的响应和处理时间，以及数据吞吐能力。
  - 例如，一个网站支持的并发用户数量和对每个用户的平均响应时间，单位时间内传输的数据量等。
- 2）资源利用性：在规定条件下，软件产品执行其功能时，使用合适数量和类别的资源的能力。
  - 例如，软件完成其规定的任务所使用的最大内存和平均内存，最大和平均占用CPU时间。在完成同样功能和性能的情况下，一个软件任务占用的内存和CPU时间越小越好。

4.4.2.5维护性

- 1）易分析性：在运行中，容易能够诊断软件中的缺陷和失效原因，以及识别出待修改部分的可能性或能力。
- 2）易修改性：软件产品容易被修改和实现其要求的程度。例如，代码、设计和文档的容易修改程度。一段很乱的代码的可修改性要比结构清晰的代码更难。
- 3）稳定性：避免由于软件修改而造成意外错误的情况。因此可以解释为软件版本的稳定性。一个软件修改的越多，表明该软件越不稳定(参见第2章图2-3)。由于每次修改均可能带来新的错误，因此软件版本的稳定是一项主要的质量指标。
- 4）易测试性：评价一个软件，特别是在其被修改后，是否容易被测试和确认。容易测试的软件，结构上一定比较清晰和简单。

4.4.2.6 可移植性

- 1）易安装性：
  - 衡量软件是否在各种环境下易于安装。
- 2）共存性：
  - 软件是否容易与其它系统在同一个平台上运行，而不发生冲突。
- 3）易替换性：
  - 软件是否容易彻底的卸载，并被其它软件或本软件的更高版本替换。

4.4.3 部件和服务项的质量评价

- 从工程角度上看，软件产业需要造出许多软件部件(OTS)。部件的质量决定整机的质量。
- 对于软件部件，也需要定义其质量要求。同样，需要定义单个服务的质量，给出各个部件和服务环境的质量要求。
- 由于并不知道部件或单独的服务项将来会嵌入在哪样的使用周境中，因此很难确定部件的质量等级。
- 因此，每个部件和服务的质量要求可能是一个范围要求。

4.5 软件过程质量

- 在传统工业中的基于过程质量观点，也同样适用于软件开发，即，软件开发过程中的质量决定了最终软件产品的质量。
  - 以瀑布式模型为例，软件需求分析、设计、编码、测试等活动的质量决定了最终产品质量。
- 反对这种观点的很容易举出反例，难道说：不关注开发过程，直接编程的黑客们就做不出好软件吗？
- 对此，Humphrey于1988年首先陈述了软件过程对产品质量的决定关系，认为有效的过程管理是获得高质量产品的关键因素，Humphrey陈述到：“如果对开发过程和产品质量进行统计，这种关系是存在的；如果离开了统计，这种规律就会被破坏”。
  - 参考第20章。进一步需要学习“软件过程管理和改进”相关的课程！

4.5 总结

- 质量归结起来主要是基于产品和基于过程质量观点。
  - 产品质量观点会定义出产品的质量度量指标，并分解到产品的外部和内部特征要求。
  - 基于过程的质量观点，希望把把这些指标分解到软件开发过程的每个阶段的每个活动中，并期望通过对过程管理落实质量要求。
  - 质量贯穿于整个软件工程过程的活动中，即实施全面质量管理(TQM—Total Quality Management)。

Homework

- 建立一个质量评价模型，比较Linux和Windows的质量。
  - 可以用不同的观点，例如，用户观点、产品观点等