Maintainability

可维护性

- During the maintenance phase, the software system, is corrected, adapted, perfected and enhanced perhaps following the planning, execution and control as well as review and evaluation guidelines recommended for the Maintenance phase in ISO/IEC 12207.
- To avoid maintenance cost overruns, it is imperative that the software is maintainable.
- Software maintainability as a software quality attribute is defined as the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, adapt to a changed environment or accommodate new requirements.
- Maintainability of a software is assessed in terms of Analyzability, Modularity, Reusability, Modifiability and Testability.

- 在维护阶段，软件系统会得到修正、适应、完善和增强，可能还需遵循ISO/IEC 12207中为维护阶段推荐的计划、执行与控制以及评审和评估指南。
- 为了避免维护成本超支，软件必须具备可维护性，这是至关重要的。
- 作为软件质量属性的软件可维护性，是指软件系统或组件能够被修改以纠正故障、提高性能或其他属性、适应变化的环境或满足新需求的难易程度。
- 软件的可维护性通过可分析性、模块化、可重用性、可修改性和可测试性来评估。

Analyzability

- ease of effort required to understand, decompose and/or modify a particular source code
- While consistency is the key, following efforts help improve analyzabiliy:
  - Enforcing established coding conventions for names, comments and format of the software.
  - Utilizing well known coding patterns, design patterns and reference architectures as the foundation of the software.
  - Maintaining mandated documentation and any accompanying change logs, and making sure that these are accessible to all stakeholders.
  - Establishing a set criteria and format for error reporting and logs.
  - Conducting code reviews on a regular basis or integrating static code analysis tools such as lint in the build process to monitor that the above conventions are being followed.
  - Collecting metrics e.g. how long it took to fix the bug from the time it was reported so that the above conventions could be improved upon.

可分析性

- 理解、分解和/或修改特定源代码
- 虽然一致性是关键，但以下措施有助于提高可分析性：
  - 强制执行既定的编码规范，包括命名、注释和软件格式。
  - 采用广为人知的编码模式、设计模式和参考架构作为软件的基础。
  - 维护规定的文档以及任何相关的变更日志，并确保所有相关方都能访问这些文档。
  - 建立错误报告和日志的固定标准和格式。
  - 定期进行代码审查，或在构建过程中集成静态代码分析工具（如 lint），以监控上述规范是否被遵守。
  - 收集指标，例如从报告缺陷到修复完成所花费的时间，以便改进上述规范。

Modularity

- Modularity is the extent of partitioning of the source code and its refactoring into separate functions, classes, packages and components. Modularization shall aim at:
  - Avoiding code duplication
  - Advancing cohesion but minimizing coupling among functions, classes, packages and components so that a change in one of these sections of the software does not affect other sections
  - Ensuring that modules could be tested independently

模块化

- 模块化是指对源代码进行划分，并将其重构为独立的函数、类、包和组件的程度。模块化应旨在：
  - 避免代码重复
  - 提高内聚性，同时尽量减少函数、类、包和组件之间的耦合，使得软件某一部分的更改不会影响其他部分
  - 确保模块可以独立测试

Reusability

- Reusability is the level of molarity in the source code. In other words, it is reflection of how well each of the individual modules hide information e.g. implementation details and variables etc. to other modules.
- As a rule of thumb, if a piece of code is not usable outside of the original system then it is not reusable at all.
- Following OO paradigm religiously and taking advantage of its properties e.g. abstraction, encapsulation, inheritance and encapsulation inherently improve reusability, as an example.

可重用性

- 可重用性是指源代码中模块化的程度。换句话说，它反映了各个独立模块在多大程度上向其他模块隐藏了信息，例如实现细节和变量等。
- 根据经验法则，如果一段代码无法在原始系统之外使用，则它完全不具备可重用性。
- 例如，严格遵循面向对象的范式，并利用其特性（如抽象、封装、继承和多态），本身就可提高可重用性。

Modifiability

- Modifiability is the ease of effort required to change a section of the software and thereafter test all other affected pieces.
- Modifiability improves with decoupling among functions, classes, packages and components.
- Presence of structural constructs identified with OO Paradigm such as interfaces and abstract classes naturally implies extendibility and modifiability.
- How much a module interfaces with other modules will impact its modifiability.
- Modifiability of pieces of software can be deduced in terms of cyclomatic complexity.
- Impact on the regression testing following a change in the code is another measure of modifiability.

可修改性

- 可修改性是指更改软件某一部分所需的努力程度以及随后测试所有其他受影响部分的难易程度。
- 函数、类、包和组件之间的解耦程度越高，可修改性就越好。
- 面向对象范式中所识别出的结构构造（如接口和抽象类）的存在自然意味着可扩展性和可修改性。
- 一个模块与其他模块接口的程度将影响其可修改性。
- 软件各部分的可修改性可以通过圈复杂度来推断。
- 代码更改后对回归测试的影响是可修改性的另一个度量标准。

Testability

It is the ease of effort required to determine if each of the software requirements had been satisfied. Whether these tests were written before or after the code was written would influence the testability of the test suite.

From another perspectives, it is also measured in terms of how many unit tests have been written and how much test coverage these unit tests provide.

可测试性

它是指判断软件各项需求是否已被满足所需付出的努力程度。这些测试是在代码编写之前还是之后编写的，将会影响测试套件的可测试性。

从另一个角度来看，它还可以通过编写了多少个单元测试以及这些单元测试提供了多少测试覆盖率来衡量。

Maintainability Measures

- Given two implementations of an application, it should be possible to quantitatively decide which implementation is more maintainable than the other.
- Since the seminal work by Halstead several maintainability measures have been defined to measure maintainability. The MI (Maintainability Index) presented below was originally defined by Paul Oman and Jack Hagemeister.
  - MI = 171-5.2ln(HV)-0.23(CC)-16.2ln(LoC)+50.0sin*sqrt(2.46*COM).
- MI increases with COM (comments) but decreases with HV (Halstead Volume), CC (Cyclomatic Complexity) or LOC (Lines of Code).

○ Cyclomatic Complexity is a quantitative measure of the complexity of the code [Thomas J. McCabe in 1976]. It is determined by counting number of linearly independent paths through a program's source code.

○ Calculation of Halstead Volume requires determination of number of distinct operators (e.g. mathematical or Boolean operators etc.), number of distinct operands (e.g. variables and constants etc.), number of operator instances in the code and number of operand instances in the code [Maurice Halstead]

可维护性度量

- 对于一个应用程序的两种实现，应能够定量地判断哪一种实现更具可维护性。
- 自霍尔斯特德（Halstead）的开创性工作以来，已定义了多种可维护性度量方法来衡量可维护性。下文介绍的MI（可维护性指数）最初由保罗·奥曼（Paul Oman）和杰克·哈格梅斯特（Jack Hagemeister）提出。
  - MI = 171-5.2ln(HV)-0.23(CC)-16.2ln(LoC)+50.0sin*sqrt(2.46*COM).
- MI 随 COM（注释）增加而增大，但随 HV（Halstead Volume）、CC（圈复杂度）或 LOC 减小。（代码行数）。

○ 圈复杂度是一种 代码复杂性的定量度量 [托马斯・J・麦克凯布（Thomas J. McCabe）于 1976]。它通过计算程序源代码中线性无关路径的数量来确定。

○ 计算Halstead Volume需要确定代码中不同操作符（例如数学或布尔操作符等）的数量、不同操作数（例如变量和常量等）的数量、操作符在代码中的出现次数以及操作数在代码中的出现次数 [Maurice Halstead]

Other metrics ?

- how equitably the class functionality is distributed among its methods;
- how uniformly the instance variables are being referenced in the instance methods;
- how coupled is the class to other classes ;
- how long is the hierarchy of super-classes;
- how deep is the hierarchy of sub-classes;
- how big is the class in terms of its instance variables and methods total etc.
- how big is the package
- how many interfaces and abstract classes in the package
- coupling among the classes in the package

其他度量标准?

- 类的功能在其方法之间分配的公平程度;
- 实例变量在实例方法中被引用的均匀程度;
- 该类与其他类之间的耦合程度;
- 超类层次结构的长度;
- 子类层次结构的深度;
- 该类在实例变量和方法总数等方面的规模有多大
- 该包的规模有多大
- 该包中有多少接口和抽象类
- 包中类之间的耦合程度

- Maintainability Analysis Tools:
  - Metrics Reloaded (Android)
  - Radon (Python)

- 可维护性分析工具：
  - Metrics Reloaded（Android）
  - Radon（Python）