SOFTWARE TESTING

Software Testing

Quality Metrics and Tools

School of Computer Science & Engineering
Sun Yat-sen University

Instructor: Guoyang Cai
email: isscgy@mail.sysu.edu.cn

Approaches & Technologies

中山大學
SUN YAT-SEN UNIVERSITY

OUTLINE

- 3.1 软件静态测试概述
- 3.2 软件代码检查
- 3.3 软件复杂性分析
- 3.4 软件质量度量
  - 软件质量度量概述
  - IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - 软件质量评价指标
- 3.5 软件静态分析工具
  - IBM Rational Logiscope
  - HP FortifySCA
  - PRQA QAC/QAC++
  - Parasoft C/C++ Test

软件质量度量概述

- 软件产品度量
  - 实体特性度量：实体的某个特定属性被赋予一个数值，称为对该实体特性的一个度量。
  - 软件产品度量：对软件产品开发的费用、工作量、生产率以及功能、性能、可靠性和其它质量方面的度量，其目的在于对软件产品进行评价，并在此基础上优化产品的设计、制造和服务。
    - 软件产品度量主要针对软件产品的质量进行，软件产品度量实质上就是软件质量的度量。
    - 软件质量的度量与软件的质量周期密切相关。

- 软件产品的质量特性参见 Chap.1.4 Software Quality Criteria.

软件质量度量概述

- 软件质量度量
  - 软件质量度量从整体上对软件质量进行测评。
    - 用于软件开发过程中对软件的质量控制；
    - 用于软件产品最终的评价和验收。
  - 软件质量度量应根据软件质量要求做到：

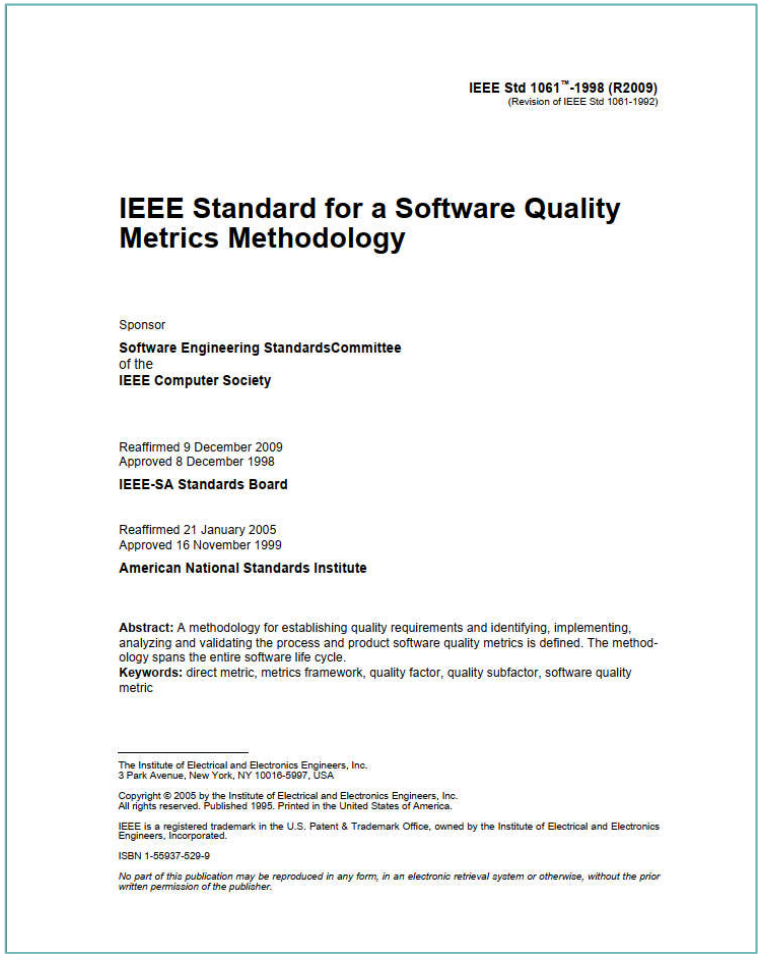
    (1) 确定各个质量特性要求的级别 (评定等级)。

    (2) 标识每个质量子特性所要求的度量元和度量方法。
      - 软件质量特性和子特性描述的软件度量需求很难直接测量，需要进一步确定相关的度量元，并将它们与质量子特性、质量特性以及质量模型联系起来。

    (3) 确定软件产品质量的定量定级水平。

IEEE Std 1061-1998(R2009) 软件质量度量方法学

IEEE Standard for a Software Quality Metrics Methodology



IEEE Std 1061™-1998 (R2009)
(Revision of IEEE Std 1061-1992)

IEEE Standard for a Software Quality Metrics Methodology

Sponsor

Software Engineering StandardsCommittee
of the
IEEE Computer Society

Reaffirmed 9 December 2009
Approved 8 December 1998

IEEE-SA Standards Board

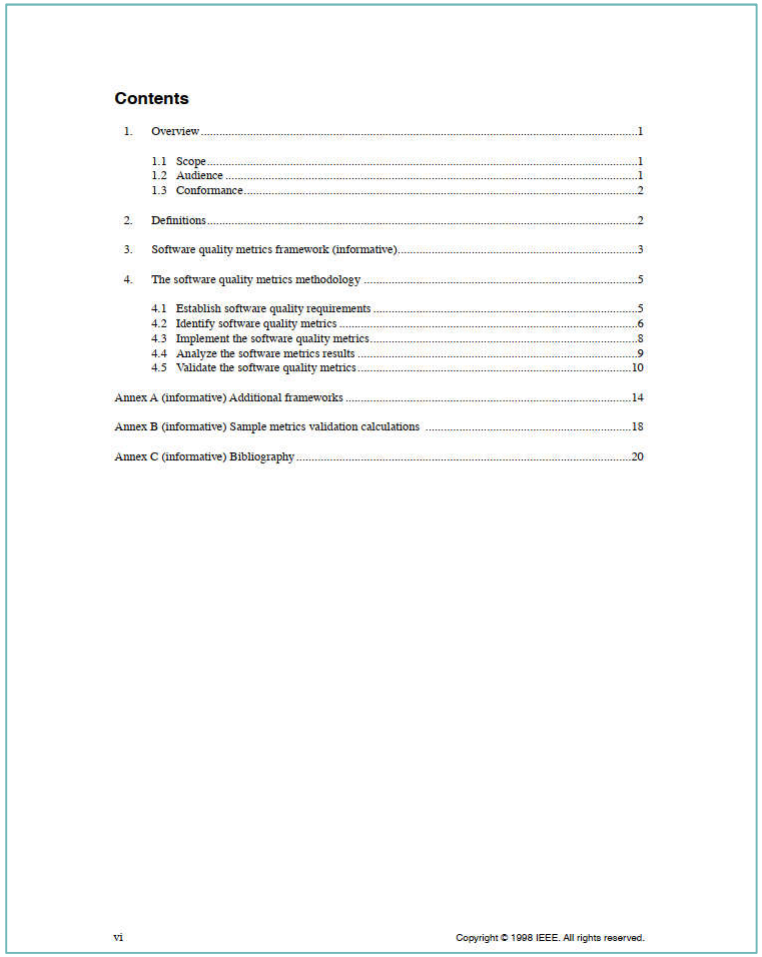
Reaffirmed 21 January 2005
Approved 16 November 1999

American National Standards Institute

Abstract: A methodology for establishing quality requirements and identifying, implementing, analyzing and validating the process and product software quality metrics is defined. The methodology spans the entire software life cycle.
Keywords: direct metric, metrics framework, quality factor, quality subfactor, software quality metric

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

ISBN 1-55937-529-9



Contents

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - 概述
    - IEEE Std 1061-1998 (IEEE Standard for a Software Quality Metrics Methodology, Reaffirmed 2004, 2009) 提供了系统地进行软件质量度量的途径。该方法学跨越整个软件生命周期，包括建立软件系统的质量需求、标识/准备、实现、分析并确认该软件的质量度量等5个过程步骤。标准共由4章及3个附录组成：第1章给出标准的适用范围，第2章给出一组有关的定义，第3章描述一个软件质量度量的框架，第4章对软件质量度量方法学作了比较详细的陈述。附录给出了使用该标准的指南和实例。
    - History
      - Published Date:1994-11-30
      - Reaffirmed:2009-12-09
      - Withdrawn Date:2020-03-05

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **Overview of IEEE Std 1061**
    - *Software quality* is the degree to which software possesses a desired combination of quality attributes.
    - *The purpose of software metrics* is to make assessments throughout the software life cycle as to whether the software quality requirements are being met.
    - *The use of software metrics*
      - reduces subjectivity (主观性) in the assessment and control of software quality by providing a quantitative basis for making decisions about software quality.
      - does not eliminate the need for human judgment in software assessments.
      - is expected to have a beneficial effect by making software quality more visible (可见/直观).

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **Overview of IEEE Std 1061**
    - The use of this standard's methodology for measuring quality enables an organization to
      - Assess achievement of quality goals;
      - Establish quality requirements for a system at its outset;
      - Establish acceptance criteria and standards;
      - Evaluate the level of quality achieved against the established (quality) requirements;
      - Detect anomalies or point to potential problems in the system;
      - Predict the level of quality that will be achieved in the future;
      - Monitor changes in quality when software is modified;
      - Assess the ease of change to the system during product evolution;
      - Validate a metrics set.

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **Overview of IEEE Std 1061**
    - 使用本标准中的方法来度量软件质量，可以使一个组织机构能够：
      - 评估质量目标的达到程度；
      - 从一开始就建立系统的质量需求；
      - 建立验收准则和标准；
      - 对照已经建立的 (质量) 需求 , 评价质量达到的级别；
      - 检测系统中的异常情况或指出系统中的潜在问题；
      - 预测未来系统将达到的质量级别；
      - 在修改软件时监督质量的变化；
      - 在产品的进化期间评估系统变更的容易程度；
      - 对度量集进行确认。

IEEE Std 1061-1998(R2009) 软件质量度量方法学

Software Quality Metrics Framework

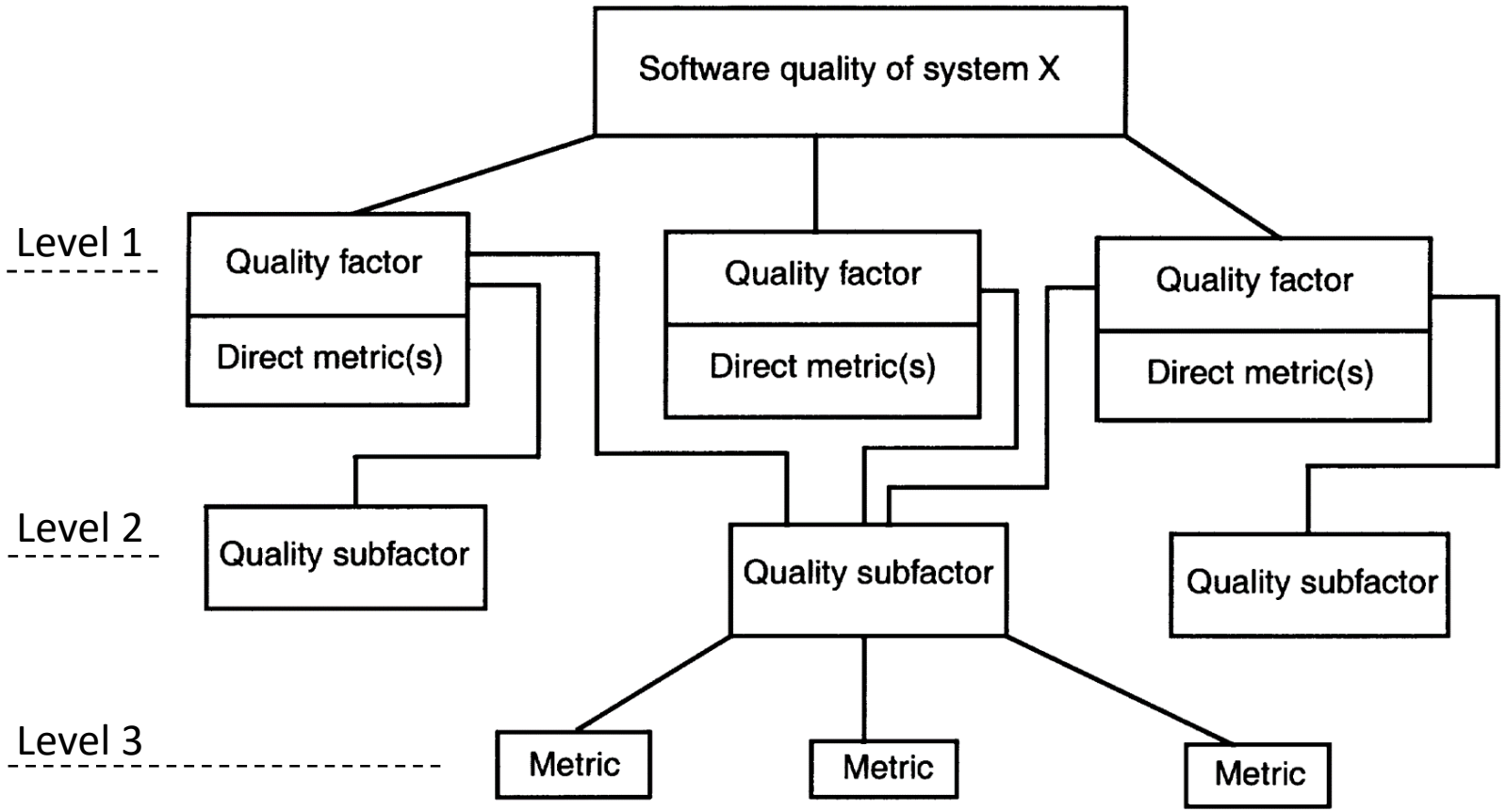


Figure 1—Software quality metrics framework

IEEE Std 1061-1998(R2009) 软件质量度量方法学

Software Quality Metrics Framework

- The framework is designed to be flexible. It permits additions, deletions, and modifications of quality factors (质量要素), quality subfactors (质量子要素), and metrics (度量). Each level may be expanded to several sublevels. The framework can thus be applied to all systems and can be adapted as appropriate without changing the basic concept.

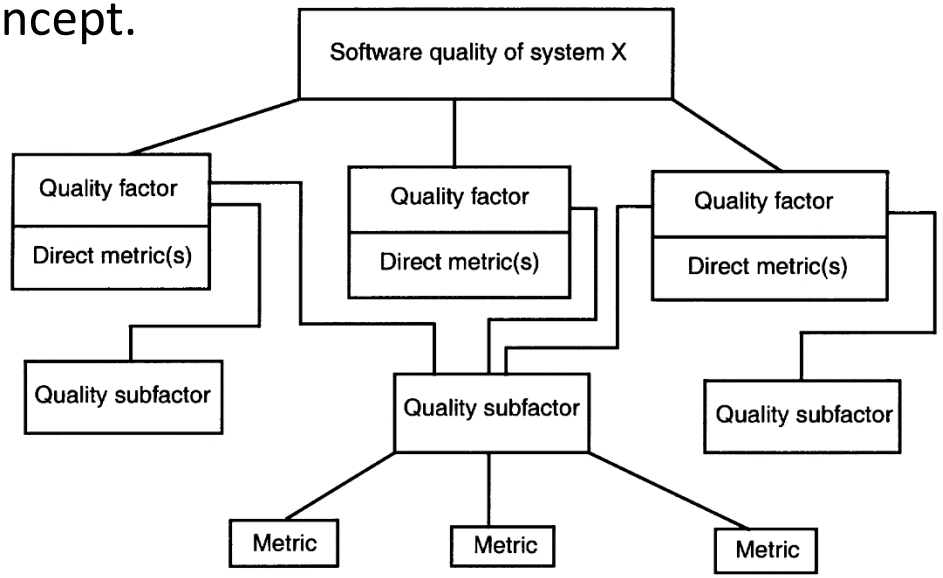


Figure 1—Software quality metrics framework

IEEE Std 1061-1998(R2009) 软件质量度量方法学

- **Software Quality Metrics Framework**
  - *The first level* of the software quality metrics framework hierarchy begins with the establishment of quality requirements by the assignment of various *quality attributes*, which are used to describe the quality of the entity system X. All attributes defining the quality requirements *are agreed upon* by the project team, and then the definitions are established. Quality factors that represent *management and user-oriented views* are then assigned to the attributes. If necessary, quality subfactors are then assigned to each quality factor. Associated with each quality factor is a direct metric that serves as a quantitative representation of a quality factor. For example, a direct metric for the factor reliability could be *mean time to failure* (MTTF).
    - Identify one or more direct metrics and target values to associate with each factor, such as an execution time of 1 hour, that is set by project management. Otherwise, there is no way to determine whether the factor has been achieved.

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **Software Quality Metrics Framework**
    - At *the second level* of the hierarchy are the quality subfactors that represent *software-oriented attributes* that indicate quality. These can be obtained by decomposing each quality factor into measurable *software attributes*. Quality subfactors are independent attributes of software, and therefore may correspond to more than one quality factor. The quality subfactors are concrete attributes of software that are more meaningful than quality factors to technical personnel, such as analysts, designers, programmers, testers, and maintainers.
      - The decomposition of quality factors into quality subfactors facilitates objective communication between the manager and the technical personnel regarding the quality objectives.

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **Software Quality Metrics Framework**
    - At *the third level* of the hierarchy the quality subfactors are decomposed into metrics used to measure system products and processes during the development life cycle.
      - Direct metric values are typically unavailable or expensive to collect early in the software life cycle.
    - From top to bottom the framework facilitates
      - Establishment of quality requirements, in terms of quality factors, by managers early in a systems life cycle;
      - Communication of the established quality factors, in terms of quality subfactors, to the technical personnel;
      - Identication of metrics that are related to the established quality factors and quality subfactors.
    - From bottom to top the framework enables the managerial and technical personnel to obtain feedback by
      - Evaluating the software products and processes at the metrics level;
      - Analyzing the metric values to estimate and assess the quality factors.

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **Software Quality Metrics Framework**
    - 第一级：质量要素。分配用于描述实体系统 X 质量的各种质量属性，从而建立质量需求。所有定义质量需求的质量属性应先由项目组共同商定，再建立这些属性的定义，然后再把体现面向管理和用户观点的质量要素赋予这些属性。必要的话，为每个质量要素分配若干质量子要素。与一个质量要素相关联的是用于定量表示一个质量要素的一个直接度量 (如可靠性要素的直接度量可以是平均失效时间 MTTF)。由项目管理部门为每个质量要素设置一个或者多个相关的直接度量及其目标值 (如可靠性要素的直接度量为 MTTF，目标值为1小时)，否则将无法确定该质量要素是否达到要求。

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **Software Quality Metrics Framework**
    - 第二级：质量子要素。这些子要素体现了面向软件的质量属性。质量子要素可以通过将质量要素分解成为可测量的软件属性得到。质量子要素是独立的软件属性，因此可以由多个质量子要素对应于一个质量要素。对于诸如分析员、设计人员、程序员、测试人员及维护人员之类的技术人员来说，质量子要素是比质量要素更有意义的具体化的软件属性。把质量要素分解为质量子要素， 有利于管理人员和技术人员之间客观地交流有关软件的质量目标。

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **Software Quality Metrics Framework**
    - 第三级：软件子要素被分解成为软件度量，用于在软件开发生命周期测定系统的产品和过程。在软件生命周期的早期阶段，直接度量 (质量要素) 的值通常不易得到或者费用昂贵。因此，为了估算软件生存周期早期阶段的质量要素值，通常不采用直接度量，而采用已经得到确认的第三级度量。

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **Software Quality Metrics Framework**
    - IEEE 1061 软件质量度量框架的优点
      - 自顶向下有助于
        - 在系统生命周期的早期阶段，管理人员根据质量要素建立质量需求；
        - 通过质量子要素的概念与技术人员进行关于已经建立的质量要素的沟通；
        - 识别与已经建立的质量要素和子要素相关的度量。
      - 自底向上便于管理和技术人员得到反馈
        - 在基本度量级评价软件产品和过程；
        - 分析度量值以预计和评估质量要素。

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **The Software Quality Metrics Methodology**
    - The software quality metrics methodology is a systematic approach to establishing quality requirements and identifying, implementing, analyzing, and validating the process and product software quality metrics for a software system. It comprises five steps. These steps are intended to be applied iteratively because insights gained from applying a step may show the need for further evaluation of the results of prior steps. Each step states the activities needed to accomplish the indicated results.
    - **Step 1. Establish software quality requirements**
      - The result of this step is *a list of the quality requirements*.
      - The activities to accomplish this result are as
        - Identify a list of possible quality requirements;
        - Determine the list of quality requirements;
        - Quantify each quality factor.

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **The Software Quality Metrics Methodology**
    - **Step 2. Identify software quality metrics**
      - The result of this step is *an approved metrics set*.
      - The activities to accomplish this result are as
        - Apply the software quality metrics framework;
        - Perform a cost-benefit analysis:
          (1) Identify the costs of implementing the metrics;
          (2) Identify the benefits of applying the metrics;
          (3) Adjust the metrics set.
        - Gain commitment to the metrics set.

IEEE Std 1061-1998(R2009) 软件质量度量方法学

The Software Quality Metrics Methodology

Table 1 - Metrics Set in Step 2 (1)

Item	Description
Name 名称	Name given to the metric. 度量的名称
Costs 代价	Costs of using the metric. 使用该度量的成本代价
Benefits 收益	Benefits of using the metric. 使用该度量的收益
Impact 影响	Indication of whether a metric can be used to alter or halt the project (ask, “Can the metric be used to indicate deficient/有缺陷 software quality ?”). 表明一个度量能否用来决定项目的变更甚至停止
Target value 目标值	Numerical value of the metric that is to be achieved in order to meet quality requirements. Include the critical value and the range of the metric. 该度量须达到的数值以满足质量需求，包括其关键值和范围。

IEEE Std 1061-1998(R2009) 软件质量度量方法学

The Software Quality Metrics Methodology

Table 1 - Metrics Set in Step 2 (2)

Item	Description
Quality factors 质量要素	Quality factors that are related to this metric. 与该度量有关的质量要素
Tools 工具	Software or hardware tools that are used to gather and store data, compute the metric, and analyze the results. 软件或硬件工具，用于收集和存储数据、计算度量值，以及分析结果
Application 应用	Description of how the metric is used and what its area of application is. 该度量的使用方法和应用领域的描述
Data items 数据项	Input values that are necessary for computing the metric values. 计算该度量值所必要的输入值
Computation 计算	Explanation of the steps involved in the metrics computation. 该度量的计算过程所包含的步骤的解释
Interpretation 解释	Interpretation of the results of the metrics computation. 该度量的计算结果的解释

IEEE Std 1061-1998(R2009) 软件质量度量方法学

The Software Quality Metrics Methodology

Table 1 - Metrics Set in Step 2 (3)

Item	Description
Considerations 考虑事项	Considerations of the appropriateness of the metric (e.g., Can data be collected for this metric? Is the metric appropriate for this application?). 该度量的适当性的考虑事项
Training required 需要的培训	Training required to implement or use the metric. 实现或使用该度量所需要的培训
Example 例子	An example of applying the metric. 应用该度量的一个例子
Validation history 确认历史	Names of projects that have used the metric, and the validity criteria the metric has satisfied. 曾经使用过该度量的项目名称，以及该度量满足的确认准则。
References 参考材料	References, such as a list of projects and project details, giving further details on understanding or implementing the metric. 为更为细致地理解和实现该度量所提供的参考材料，例如项目列表和项目细节。

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **The Software Quality Metrics Methodology**
    - **Step 3. Implement the software quality metrics**
      - The outputs of this step are *a description of the data items*, *a traceability matrix, and a training plan and schedule*.
      - The activities to accomplish this result are as
        - Define the data collection procedures;
        - Prototype the measurement process;
        - Collect the data and compute the metric values.

IEEE Std 1061-1998(R2009) 软件质量度量方法学

The Software Quality Metrics Methodology

Table 2 - Data Item Description in Step 3 (1)

Item	Description
Name 名称	Name given to the data item. 数据项的命名
Metrics 度量	Metrics that are associated with the data item. 与该数据项关联的度量
Definition 定义	Straightforward description of the data item. 该数据项的直接描述
Source 来源	Location of where the data item originates. 该数据项的来源
Collector 收集者	Entity responsible for collecting the data. 负责收集该项数据的实体
Timing 时间	Time(s) in life cycle at which the data item is to be collected. (Some data items are collected more than once.) 生命周期中该数据项被收集的时间

IEEE Std 1061-1998(R2009) 软件质量度量方法学

The Software Quality Metrics Methodology

Table 2 - Data Item Description in Step 3 (2)

Item	Description
Procedures 过程	Methodology (e.g., automated or manual) used to collect the data. 用于搜集该项数据的方法论 (例如，自动或人工)
Storage 存储	Location of where the data are stored. 该项数据的存储位置
Representation 表示	Manner in which the data are represented, e.g., precision and format (Boolean, dimensionless 无量纲, etc.). 该项数据表示的方式，例如精度和格式。
Sample 采样	Method used to select the data to be collected and the percentage of the available data that is to be collected. 选择收集该项数据的方法，以及收集数据的百分比。
Verification 验证	Manner in which the collected data are to be checked for errors. 对收集到的数据进行错误检查的方式

IEEE Std 1061-1998(R2009) 软件质量度量方法学

The Software Quality Metrics Methodology

Table 2 - Data Item Description in Step 3 (3)

Item	Description
Alternatives 候选方法	Methods that may be used to collect the data other than the preferred method. 与首选方法不同的用于搜集该项数据的其它方法
Integrity 完整性	Person(s) or organization(s) authorized to alter the data item and under what conditions. 获得授权更改该数据项的个人或组织，以及约束条件。

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **The Software Quality Metrics Methodology**
    - **Step 4. Analyze the software metrics results**
      - The results of this step are changes to the organization and development process, which are indicated from interpreting and using the measurement data.
      - The activities to accomplish this result are as
        - Interpret the results;
        - Identify software quality;
        - Make software quality predictions;
        - Ensure compliance with requirements.

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - **The Software Quality Metrics Methodology**
    - **Step 5. Validate the software quality metrics**
      - The result of this step is *a set of validated metrics* for making predictions of quality factor values.
      - The activities to accomplish this result are as
        - Apply the validation methodology;
        - Apply validity criteria;
        - Validation procedure:
          (1) Identify the quality factors sample;
          (2) Identify the metrics sample;
          (3) Perform a statistical analysis;
          (4) Document the results;
          (5) Revalidate the metrics;
          (6) Evaluate the stability of the environment.

IEEE Std 1061-1998(R2009) 软件质量度量方法学

- IEEE Std 1061 软件质量度量方法学的5个步骤

(1) 建立软件质量需求

- 质量需求表达了在具体应用的特定环境下对软件产品质量的定量要求，应该在软件开发前或初期进行定义。它是有效构造软件质量、客观评价质量的前提。
- 质量需求规格说明可定量定义对所需质量特性的直接度量及其直接度量目标值。
- 质量特性的度量用来验证最终产品是否达到了质量需求。

(2) 标识软件质量度量

- 由软件质量特性和子特性描述的软件质量需求很难直接测量，需要进一步确定相关的度量元。
- 在度量的准备阶段，根据应用环境，为软件开发的各个阶段和其最终产品分别确定适当的度量元，建立度量元、质量子特性、质量特性的映射模型，确定合理的评估准则。

- IEEE Std 1061-1998(R2009) 软件质量度量方法学
  - IEEE Std 1061 软件质量度量方法学的5个步骤 (续)

    (3) 实现软件质量度量
    - 在软件生存周期的每一阶段，采购或者开发必要工具，收集
    - 有关数据并基于质量模型进行质量度量。

    (4) 分析质量度量结果
    - 分析并报告度量结果；做出度量和评估的结论。
    - 进行度量元的确认。

    (5) 确认软件质量度量
    - 把预测的度量结果与直接度量结果进行比较，以确定预测的度量是否准确地测定了它 们的相关质量要素。

- 软件质量评价指标 (评估准则)
  - 软件质量评价指标可分为定性指标和定量指标
    - 定量指标可以更为客观地反映软件的质量特征；
    - 有的质量特征需要采用一定的定性指标进行描述；
    - 选择合适的指标体系并使其量化是软件测试与评估的关键。
  - 评价指标的选择原则
    - 针对性
      - 能够反映评估对象软件的质量特征，具体表现为其功能性与可靠性。
    - 可测量性
      - 可以通过数学计算、平台测试、经验统计等方法得到具体数据，定量描述软件的质量。
    - 简明性
      - 选择的指标易于被各方理解和接受。

- 软件质量评价指标 (评估准则)
  - 评价指标的选择原则 (续)
    - 完备性
      - 选择的指标应覆盖分析对象所涉及的所有范围。
    - 客观性
      - 选择的指标应客观反映软件本质特征，不能因人而异。
    - 数量适中
      - 选择评价指标的关键在于指标在评估中所起的作用，而不是指标的数量。评估时指标太多，会增加结果的复杂性，甚至还会影响评估的客观性。
    - 均衡性
      - 软件质量特性/子特性之间存在冲突，在设计软件质量评价时必须全面权衡，根据质量需求进行适当合理的选择。

- 软件质量评价指标 (评估准则)
  - 软件质量度量元举例
    - All supported languages
      - Cyclomatic number V(G) 环路复杂度
      - Comment frequency 注释频度
      - Number of nesting levels 嵌套层数目
      - Number of execution paths 执行路径数目
      - Number of macros 宏定义数目
      - Number of function parameters 函数参数数，目
      - Number of instructions 指令条数
      - Number of GOTO GOTO 语句数目
      - Number of RETURN RETURN 语句数目
      - Fan in/out 扇入/扇出数目
      - etc.

软件质量度量

- 软件质量评价指标 (评估准则)
  - 软件质量度量元举例
    - Object-oriented languages
      - Weighted Methods per Class
      - Class testability
      - Number of Children
      - Number of Ancestors
      - Depth of inheritance Tree
      - Coupling between objects
      - Multiple inheritance indicator
      - Fan in/out of a class
      - Class comment frequency
      - Class Coupling
      - Number of redefined methods
      - etc.

- 软件质量评价指标 (评估准则)
  - 软件质量的定量评价公式举例
    - 结构性：0.2 × 编码语句的最大嵌套层次 + 0.2 × 修改全局数据 + 0.2 × 使用 Goto 语句 + 0.2 × 数据习惯用法 + 0.2 × 无条件循环语句所占比例
    - 简洁性：0.4 × 实体的习惯用法 + 0.4 × 局部调用 + 0.2 × 被调用
    - 自描述性：0.2 × B_comment + 0.3 × 全部注释行所占的比例 + 0.5 × 注释实体所占比例
    - 独立性：0.5 × 异常比例 + 0.5 × 用户定义类型
    - 完整性：(if 语句 + case 语句 + 初始化对象) / 3
    - 模块复杂性：(环路复杂度 + 模块设计复杂度 + 设计复杂度 + 集成复杂度) / (圈复杂度临界值 + 模块设计复杂度临界值 + 设计复杂度临界值 + 集成复杂度临界值)

- 软件质量评价指标 (评估准则)
  - 软件质量的定量评价公式举例 (续)
    - 可测试性：0.5 × 结构性 + 0.5 × McCabe 复杂度
    - 可理解性：0.25 × 结构性 + 0.25 × McCabe 复杂度 + 0.25 × 简洁性 + 0.25 × 自描述性
    - 可维护性：0.5 × 可测试性 + 0.5 × 可理解性
    - 可移植性：0.5 × 独立性 + 0.5 × 完整性
    - 模块性：0.5 × 编码行数 + 0.5 × 结构性
    - 可靠性：0.33 × 完整性 + 0.33 × 模块性 + 0.34 × 可测试性

软件质量评价指标 (评估准则)

软件项目质量度量指标举例

基本度量项	
持续时间偏差 (%)	((实际持续时间-计划持续时间)/计划持续时间)*100， 持续时间不包含非工作日
进度偏差 (%)	((实际结束时间-计划结束时间)/计划持续时间)*100
工作量偏差 (%)	((实际工作量-计划工作量)/计划工作量)*100
规模偏差 (%)	((实际规模-计划规模)/计划规模)*100
分配需求稳定性指数 (%)	(1- (修改、增加或删除的分配需求数/初始的分配需求数))*100
软件需求稳定性指数 (%)	(1- (修改、增加或删除的软件需求数/初始的软件需求数))*100
发布前缺陷发现密度(个/KLOC)	发布前缺陷发现总数/千行代码规模 这里的发布指开发部门向测试部门发布
遗留缺陷密度 (个/KLOC)	测试部门发现缺陷总数/千行代码规模 遗留缺陷指由测试部门发现的缺陷
生产率 (LOC/人天)	软件代码规模(LOC)/总工作量(人天)

软件质量评价指标 (评估准则)

软件项目质量度量指标举例

控制活动缺陷发现密度 度量目的：建立基线，为评估评审、测试的充分性提供参考	
SRS评审缺陷发现密度 (个/页)	SRS评审发现的缺陷数/SRS文档页数
STP评审缺陷发现密度 (个/用例)	STP评审发现的缺陷数/ST用例数
HLD评审缺陷发现密度 (个/页)	HLD评审发现的缺陷数/HLD文档页数
ITP评审缺陷发现密度 (个/用例)	ITP评审发现的缺陷数/IT用例数
LLD评审缺陷发现密度 (个/页)	LLD评审发现的缺陷数/LLD文档页数
UTP评审缺陷发现密度 (个/用例)	UTP评审发现的缺陷数/UT用例数
Code评审缺陷发现密度 (个/KLOC)	Code评审发现缺陷数/编码阶段代码规模
UT缺陷发现密度 (个/KLOC)	UT发现缺陷数/UT阶段代码规模
IT缺陷发现密度 (个/KLOC)	IT发现缺陷数/IT阶段代码规模
ST缺陷发现密度 (个/KLOC)	ST发现缺陷数/ST阶段代码规模

软件质量评价指标 (评估准则)

软件项目质量度量指标举例

缺陷类型引入密度 度量目的：建立基线，为分析能力水平薄弱环节及交付件质量提供参考	
SRS缺陷引入密度 (个/页)	SRS类型缺陷数/SRS文档页数
HLD缺陷引入密度 (个/页)	HLD类型缺陷数/HLD文档页数
LLD缺陷引入密度 (个/页)	LLD类型缺陷数/LLD文档页数
Code缺陷引入密度 (个/KLOC)	Code类缺陷数/代码规模
评审活动的有效性 度量目的：建立基线，对相关评审的充分性提供参考	
SRS评审有效性 (%)	SRS评审发现的SRS类缺陷数/SRS类缺陷总数
HLD评审有效性 (%)	HLD评审发现的HLD类缺陷数/HLD类缺陷总数
LLD评审有效性 (%)	LLD评审发现的LLD类缺陷数/LLD类缺陷总数
Code评审有效性 (%)	Code评审发现的Code类缺陷数/Code类缺陷总数

软件质量评价指标 (评估准则)

软件项目质量度量指标举例

每千行代码的文档规模 度量目的：建立基线，为评估交付件的质量从设计充分性、粒度合理性角度提供参考	
每千行代码SRS文档规模 (页/KLOC)	SRS文档页数/代码规模
每千行代码HLD文档规模 (页/KLOC)	HLD文档页数/代码规模
每千行代码LLD文档规模 (页/KLOC)	LLD文档页数/代码规模
质量成本	
质量成本 (%)	((评审工作量+返工工作量+缺陷修改工作量+测试计划准备工作量+ 测试执行工作量＋培训工作量＋质量保证工作量)/实际总工作量)*100
返工成本指数 (%)	((返工工作量+缺陷修改工作量)/实际总工作量)*100

软件质量评价指标 (评估准则)

软件项目质量度量指标举例

交付件生产率	
SRS文档生产率 (页/人天)	SRS文档页数/(SRS文档准备工作量+SRS评审工作量+SRS修改工作量)
HLD生产率 (页/人天)	HLD文档页数/(HLD文档准备工作量+HLD评审工作量+HLD修改工作量)
STP用例生产率 (用例/人天)	ST用例数/(STP准备工作量+STP评审工作量+STP修改工作量)
ITP用例生产率 (用例/人天)	ITP用例数/(ITP准备工作量+ITP评审工作量+ITP修改工作量)
UTP用例生产率 (用例/人天)	UTP用例数/(UTP准备工作量+UTP评审工作量+UTP修改工作量)
编码阶段代码生产率 (LOC/人天)	编码阶段实际代码规模/(编码工作量+代码评审工作量+代码修改工作量)

软件质量评价指标 (评估准则)

软件项目质量度量指标举例

测试执行效率	
UT用例执行效率 (用例/人天)	UT用例数/(UT准备工作量+UT用例执行工作量+UT缺陷修改工作量)
IT用例执行效率 (用例/人天)	IT用例数/(IT准备工作量+IT用例执行工作量+IT缺陷修改工作量)
ST用例执行效率 (用例/人天)	ST用例数/(ST准备工作量+ST用例执行工作量+ST缺陷修改工作量)
每千行代码测试用例规模 度量目的：建立基线，为评估交付件的质量从设计充分性、粒度合理性角度提供参考	
每千行代码ST用例规模 (用例/KLOC)	ST用例数/代码规模
每千行代码IT用例规模 (用例/KLOC)	IT用例数/代码规模
每千行代码UT用例规模 (用例/KLOC)	UT用例数/代码规模

软件质量评价指标 (评估准则)

- 软件项目质量度量指标举例

实测规模缺陷发现密度 度量目的：建立基线，为评估测试用例的质量提供参考	
UT实测规模缺陷发现密度 (个/KLOC)	UT发现的缺陷数/UT活动实际测试代码规模
IT实测规模缺陷发现密度 (个/KLOC)	IT发现的缺陷数/UT活动实际测试代码规模
ST实测规模缺陷发现密度 (个/KLOC)	ST发现的缺陷数/UT活动实际测试代码规模

- IBM Rational Logiscope
  - 概述
    - Logiscope 是一组嵌入式软件测试工具集，贯穿于软件开发、代码评审、单元/集成测试、系统测试以及软件维护阶段，面向源代码进行工作。
    - Logiscope 针对编码、测试和维护，重点是为代码评审 (Review) 和动态覆盖测试 (Testing) 提供帮助。
    - Logiscope 对软件的分析采用基于通用的度量方法 (*Halstead*、*McCabe* 等) 的质量模型，以及从多家公司收集的编程规则集，可以从软件的编程规则、静态特征和动态测试覆盖等多个方面，量化地定义质量模型，并检查、评估软件质量。

- IBM Rational Logiscope
  - Logiscope 的主要功能
    - Logiscope 以三个独立工具的形式分别提供三项主要功能：
      - Audit：软件质量分析工具，用于静态检查程序代码的各种指标，如健壮性、可维护性等。
      - RuleChecker：代码规范性检测工具，用于静态检查程序代码是否符合相应的编码规范。
      - TestChecker：测试 (边) 覆盖率统计工具。边覆盖率是执行的测试用例对程序流程图中的边的覆盖情况。
    - Audit 和 Rulechecker 提供了对软件进行静态分析的功能，TestChecker 提供了测试覆盖率统计的功能。

软件静态分析工具

- IBM Rational Logiscope
  - Logiscope 的支持环境
    - 支持的主机平台：
      - UNIX-Like：Sun OS/Solaris, HP 700 HP-UX, RS6000 AIX, Power PC, DEC UNIX
      - IBM Mainframe MVS 环境
      - Microsoft Windows/NT
    - 支持的语言：
      - C, C++, Ada, Java
    - 支持的嵌入式目标机环境：
      - 嵌入式实时操作系统 VxWorks, PSOS, VRTX

- IBM Rational Logiscope
  - Audit
    - Audit 以 ISO 9126 模型作为质量评价模型的基础。质量评价模型描述了从 Halstead 、McCabe 的度量方法学和 VERILOG 引入的质量方法学中的质量因素 (可维护性、可重用性等) 和质量准则 (可测试性、可读性等)。
    - Audit 将被评价的软件与规定的质量模型进行比较，对度量元素和质量模型不一致的地方作出解释并提出纠正的方法，用图形形式显示软件质量的级别。

- IBM Rational Logiscope
  - RuleChecker
    - RuleChecker 预定义了50 个编程规则：名称约定 (如：局部变量用小写)；表示约定 (如：每行一条指令)； 限制 (如：不能用 GOTO 语句，不能修改循环体的循环计数器等)。用户可以从这些规则中选择，也可以用 Tcl 、脚本和编程语言定义新的规则。此外，还提供了50个面向安全-关键系统的编程规则。
    - RuleChecker 用所选定的规则对源代码进行验证，指出所有不符合编程规则的代码，并提出改进源代码的解释和建议。
    - RulrChecker 通过文本编辑器直接访问源代码并指出需要纠正的位置。

- IBM Rational Logiscope
  - TestChecker
    - TestChecker 推荐面向指令 (IB)、面向逻辑路径 (DDP) 和面向调用路径 (PPP) 的覆盖测试。此外对安全-关键软件还提供了 MC/DC 的覆盖测试。 (MC/DC : Modified Condition/Decision Coverage)
    - TestChecker 产生每个测试的测试覆盖信息和累计信息，用直方图显示覆盖比率，并根据测试运行情况实时在线更改，随时显示新的测试所反映的测试覆盖情况。
    - TestChecker 允许所有的测试运行依据其有效性进行管理。用户可以减少那些用于非回归测试的测试。
    - 执行测试期间，当测试策略改变时，TestChecker 可以综合运用检测关键因素以提高效率。将 TestChecker 与 Audit 配合使用能够帮助用户分析未测试的代码。

- IBM Rational Logiscope
  - 对嵌入式领域的支持
    - 嵌入式系统软件的开发采用交叉编译方式进行，在目标机上不可能有多余的空间记录测试的信息，必须实时地将测试信息通过通信接口回传到宿主机上，并实时在线显示。因此，对源代码的插装和目标机上的信息收集与回传成为解决问题的关键。
    - Logiscope 是嵌入式领域测试工具的领先者。它支持各种实时操作系统 (RTOS) 上的应用程序的测试以及逻辑系统的测试，提供 VxWorks 、 pSOS 、 VRTX 等实时操作系统的测试库。

软件静态分析工具

- IBM Rational Logiscope
  - 对航空/航天/国防/核电站领域的支持
    - 航空/航天领域的软件安全性至关重要。因此，欧美的航空/航天制造厂商和使用单位联合制定了 RTCA/DO-178B 标准
    - Logiscope 通过 “Reviews and Analysis of the Source Code” 和 “Structural Coverage Analysis” 能够使开发的软件达到 RTCA/DO-178B 标准的 A 、B 、C 三个系统级。
    - Logiscope 也是第一个提供 MC/DC (Modified Condition/Decision Coverage) 测试的工具。

软件静态分析工具

- IBM Rational Logiscope
  - 现状
    - On September 12, 2012, IBM announced the withdrawal from marketing of IBM Rational Logiscope product family.
    - Kalimetrix has acquired in 2012 the rights of IBM Rational Logiscope.
    - A first release is already available containing:
      - All functionalities included in the last IBM Release 6.6
      - Support for Windows 7 (32 & 64 bits), 2008 Server R2
      - Preliminary support of VB and C# languages
    - Product Roadmap for Logiscope 2013
      - Full support of Linux flavors Red Hat Enterprise Linux 5 and Ubuntu 12.04 LTS
      - Full implementation of Misra C++, JSF
      - New module Req Tracker
      - New language COBOL
    - Last release: v7.2-2016

- HP FortifySCA 静态分析工具
  - 基本功能
    - HP FortifySCA 产品套件由内置的扫描分析引擎、安全编码规则包、审计工作台、规则自定义编辑器和向导、IDE 插件五部分组件构成。五个组件配合完成对源代码安全漏洞的扫描、分析、查看、审计等工作。
    - 通过内置的数据流、语义、结构、控制流、配置等五大主要扫描分析引擎对应用软件的源代码进行静态分析。
    - 扫描分析的过程中与它特有的软件安全漏洞规则集进行全面匹配、查找，发现源代码中存在的安全漏洞，并给予整理报告。
    - 扫描分析的结果中不但包括详细的安全漏洞的信息，还包括相关的安全知识的说明，并提供了修复意见。

软件静态分析工具

- HP FortifySCA 静态分析工具
  - 产品组件
    - 扫描分析引擎
      - 内置五大扫描分析引擎与规则包配合工作，从五个侧面分析程序源代码中的安全漏洞。
    - 安全编码规则包
      - 数十万条软件安全漏洞特征的集合，能查找约350多种安全漏洞，内置在 SCA 中与分析引擎配合工作。
    - 审计工作台
      - 包含大量的丰富的软件漏洞的信息，如漏洞的分级、漏洞产生的全过程、漏洞所在的源代码定位、以及漏洞的解释说明和推荐的修复建议等。
      - 是为用户对 SCA 的漏洞分析结果的查看，审计等工作提供方便的综合平台。

- HP FortifySCA 静态分析工具
  - 产品组件
    - 规则自定义向导/编辑器
      - HP FortifySCA 的规则支持自定义功能，方便用户扩展 SCA 对漏洞的分析能力。为此 SCA 提供了一个用户自定义规则的向导和编辑器。
    - IDE 插件
      - 为了方便用户使用 SCA 对程序源代码进行安全扫描，产品提供了多种 IDE 工具的插件。
        - 面向 IDE 如 Eclipse, Visual Studio, WSAD (IBM Websphere Application Developer), RAD (IBM Rational Application Developer) 等。

软件静态分析工具

- HP FortifySCA 静态分析工具
  - 扫描分析引擎
    - 数据流引擎
      - 跟踪、记录并分析程序数据传递过程所产生的安全问题。
    - 语义引擎
      - 分析程序中不安全的函数，方法的使用的安全问题。
    - 结构引擎
      - 分析程序上下文环境，结构中的安全问题。
    - 控制流引擎
      - 分析程序特定时间，状态下执行操作指令的安全问题。
    - 配置引擎
      - 分析项目配置文件中的敏感信息和配置缺失的安全问题。

软件静态分析工具

HP FortifySCA 静态分析工具

软件静态分析工具

- PRQA QA·C/QA·C++
  - 概述
    - QA·C/QA·C++ 由 Programming Research Ltd (PRQA) 开发，用于 C 和 C++语言的静态编程规则检查，具有领域领先地位。
    - PRQA 参与了 MISRA-C 标准的起草和编写工作，其产品在汽车、通信、航天航空、军工等领域有很大影响。
  - 功能
    - 对 C/C++ 代码规则进行自动审查，报告所违反的编程标准和准则，从而减少代码审查所需时间，缩短后期动态测试的周期。
    - 提供全面的规范支持，能够发现1700多种 C 语言问题、1200多种 C++ 语言问题，支持所有编译器的扩展；支持多种编程标准 (ISO, MISRA C2/C3, JVF, EC++ 等)，支持多种其它行业编程规则；提供二次开发接口，允许添加其它自定义的编程规则。

软件静态分析工具

- PRQA QA·C/QA·C++
  - 功能 (续)
    - 提供另外两种静态分析的能力：软件结构分析和质量度量。
      - 软件结构分析包括：函数控制结构图、函数调用树、数据引用关系图，文件包含关系。
      - 软件质量度量包括：提供60多种 C 语言和20多种 C++ 语言的复杂度度量，包括环路复杂度、静态路径统计和 Myer‘s interval 等，还可以扩展定制复杂度度量。
      - 提供开发接口，可扩展执行特定的分析检查。
    - 提供多种可视化输出，包括函数结构图、函数调用树、外部参考、文件包含关系和统计的度量分析。
    - 支持语言包括 C、C++、Java、Fortran；支持平台包括 Microsoft Windows、Sun Solaris、HP-UX、Redhat Linux、Slackware Linux；可以和流行的开发环境集成。

Lecture 15. Quality Metrics and Tools

End of Lecture