中山大學
SUN YAT-SEN UNIVERSITY

Chapter 3
Software Static Testing

Software Testing: Approaches & Technologies
School of Data & Computer Science, Sun Yat-sen University

Outline

- 3.1 软件静态测试概述
- 3.2 阶段评审
- 3.3 代码检查
- 3.4 软件复杂性分析
- 3.5 软件质量控制
  - 软件质量度量
  - 软件质量度量元
  - 软件质量的定量评价
- 3.6 软件静态分析工具
  - IBM Rational Logiscope
  - HP FortifySCA
  - PRQA QAC/QAC++
  - Parasoft C/C++ Test

3.5 软件质量控制

- **软件质量度量概述**
  - 软件产品度量
    - ✧ 实体的某个特定属性被赋予一个数值，称为对该特性的一个度量。
    - ✧ 软件产品度量包含费用、工作量、生产率、性能、可靠性和质量等方面的度量，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，软件产品度量实质上就是软件质量的度量。
    - ✧ 软件的质量度量与软件的质量周期密切相关。
    - ✧ 没有软件产品质量的定量描述就无法对软件质量进行评价。

3.5 软件质量控制

- **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进行质量控制。
      - ◌ 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。

3.5 软件质量控制

- **IEEE Std 1061-1998 软件质量度量方法学**
  - 概述
    - ✧ IEEE Std 1061-1998 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存周期，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。
  - 过程

    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。

3.5 软件质量控制

- **IEEE Std 1061-1998 软件质量度量方法学**
  - 过程

2. 准备度量

✧ 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，应根据应用环境，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。

3. 实现软件质量度量

✧ 收集度量元。

✧ 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

3.5 软件质量控制

- **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。
    - ✧ 并非所有的质量特征都可用定量指标进行描述，有时需要采用一定的定性指标。
    - ✧ 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。

3.5 软件质量控制

- **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考虑利弊，全面权衡，根据质量需求，适当合理地选择/设计质量特性进行评价。

3.5 软件质量控制

- **软件质量度量元**
  - 软件质量度量元举例

All supported languages	Object-oriented languages
✧ Cyclomatic number V(G) ✧ Comment frequency ✧ Number of nesting levels ✧ Number of execution paths ✧ Number of macros ✧ Number of function parameters ✧ Number of instructions ✧ Number of GOTO ✧ Number of RETURN ✧ Fan in/out ✧ etc.	✧ 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.

3.5 软件质量控制

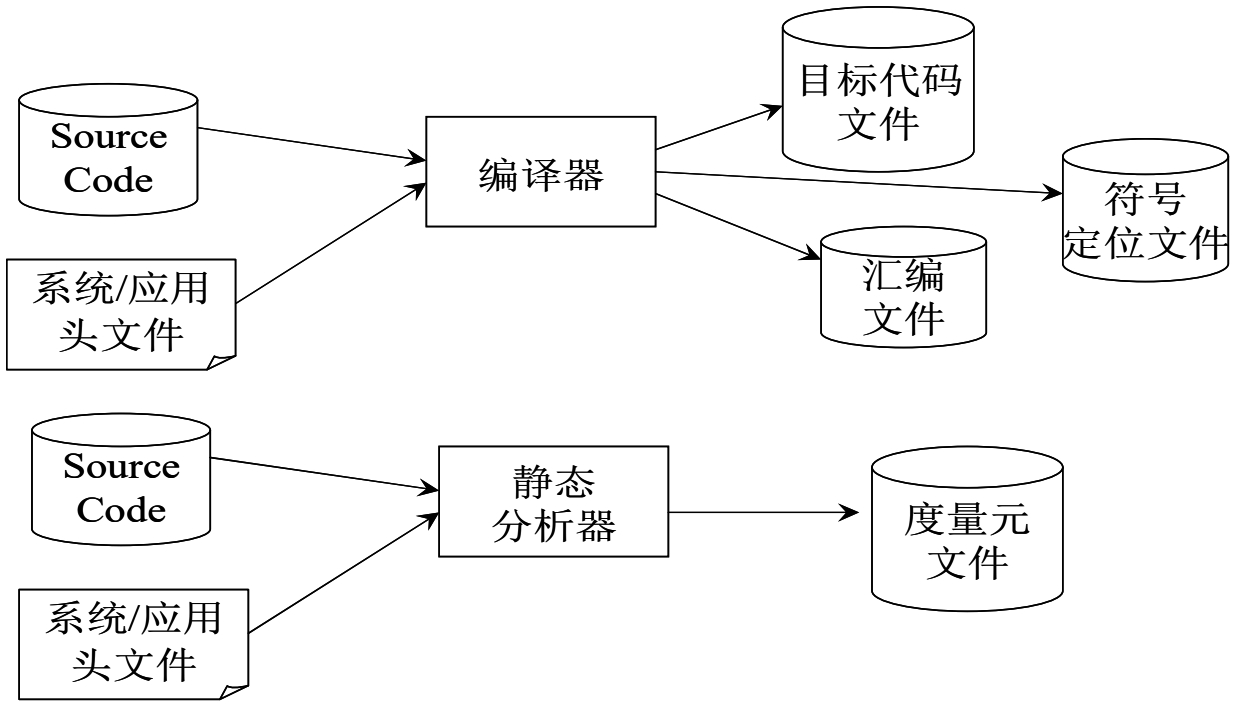
- **软件质量定量评价公式**
  - 软件质量的定量评价公式举例
    - ✧ 结构性：0.2 × 编码语句的最大嵌套层次 + 0.2 × 修改全局数据 + 0.2 × 使用 Goto 语句 + 0.2 × 数据习惯用法 + 0.2 × 无条件循环语句所占比例
    - ✧ 简洁性：0.4 × 实体的习惯用法 + 0.4 × 局部调用 + 0.2 × 被调用
    - ✧ 自描述性：0.2 × B_comment + 0.3 × 全部注释行所占的比例 + 0.5 × 注释实体所占比例
    - ✧ 独立性：0.5 × 异常比例 + 0.5 × 用户定义类型
    - ✧ 完整性：(if 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)

3.5 软件质量控制

- **软件质量定量评价公式**
  - 软件质量的定量评价公式举例
    - ✧ 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

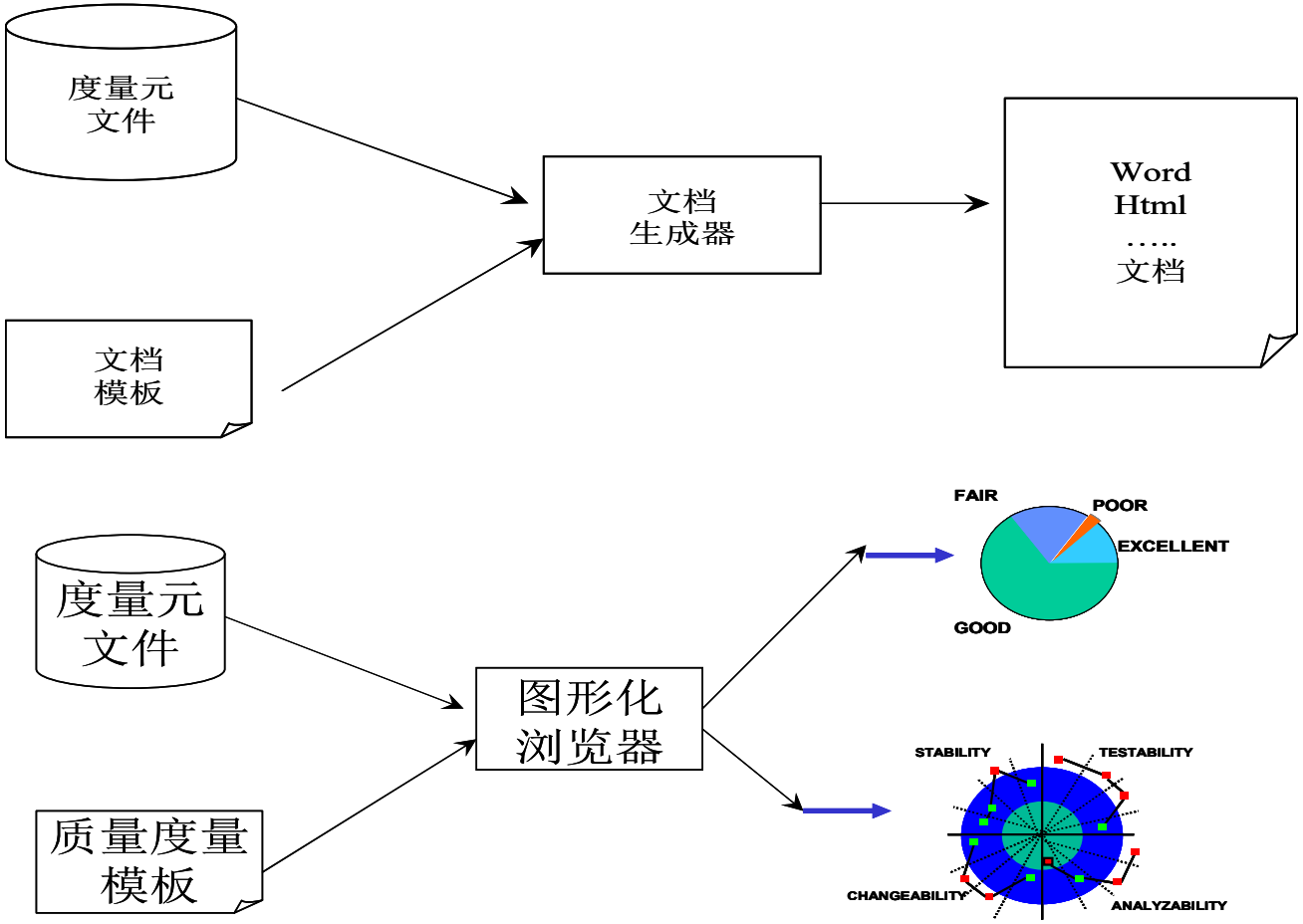
3.5 软件质量控制

- **源代码的度量过程**



3.5 软件质量控制

- **源代码的度量过程**



3.6 软件静态分析工具

- **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。

3.6 软件静态分析工具

- **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。

3.6 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

3.6 软件静态分析工具

- **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 *Halstend* 、*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。

3.6 软件静态分析工具

- **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。

3.6 软件静态分析工具

- **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。

3.6 软件静态分析工具

- **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，并实时在线地显示。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。

3.6 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。

3.6 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

3.6 软件静态分析工具

- **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。

3.6 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。

3.6 软件静态分析工具

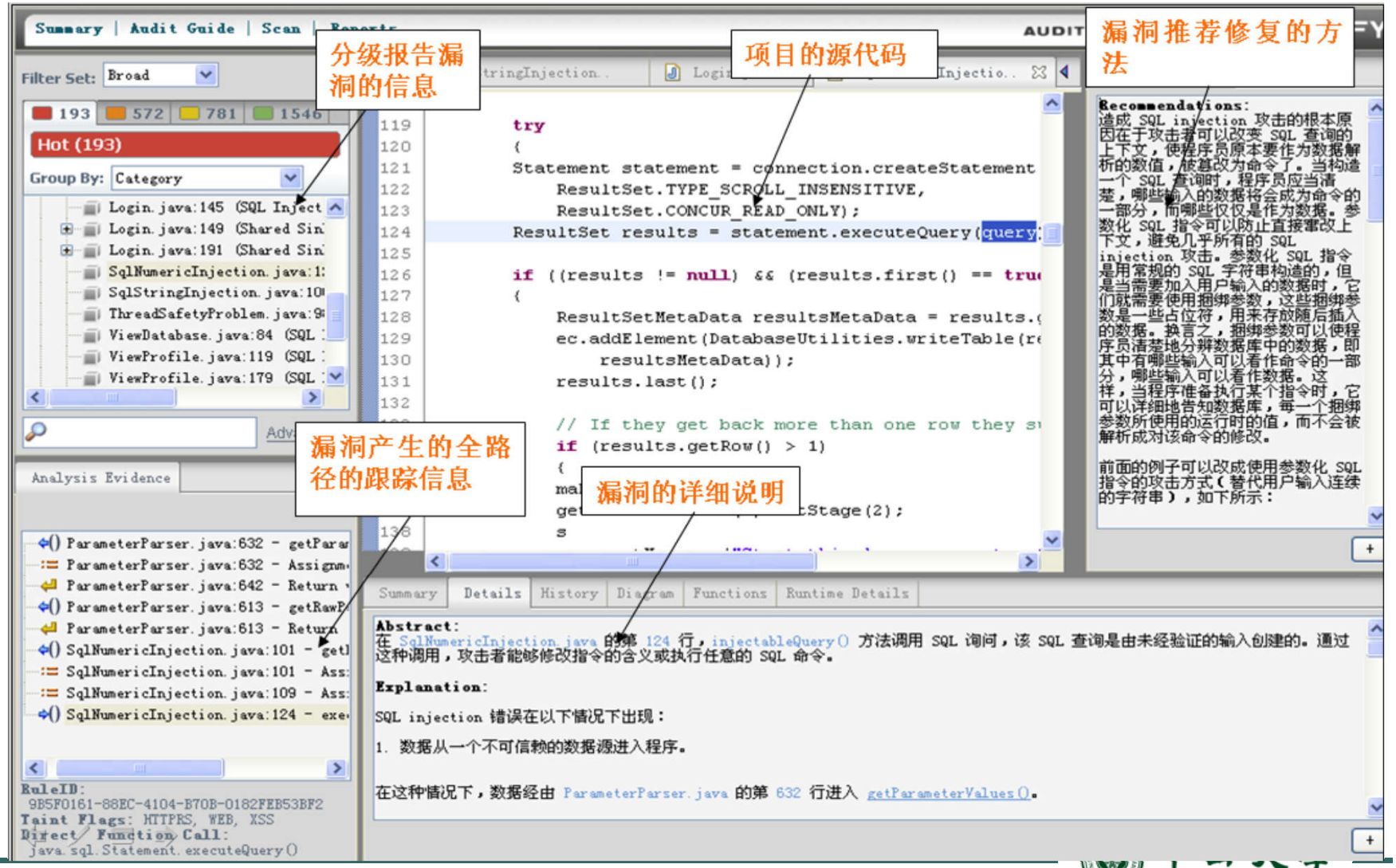
- **HP FortifySCA 静态分析工具**
  - 产品组件
    - ✧ 规则自定义向导/编辑器
      - ◌ HP FortifySCA 的规则支持自定义功能，方便用户扩展 SCA 对漏洞的分析能力。为此 SCA 提供了一个用户自定义规则的向导和编辑器。
    - ✧ IDE 插件
      - ◌ 为了方便用户使用 SCA 对程序源代码进行安全扫描，产品提供了多种 IDE 工具的插件。
        - 面向 IDE 如 Eclipse, Visual Studio, WSAD (IBM Websphere Application Developer), RAD (IBM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

3.6 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。

3.6 软件静态分析工具

- **HP FortifySCA 静态分析工具**



3.6 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的时间，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++ 的问题，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。

3.6 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。

Thank you!