

嵌入式系统

An Introduction to Embedded System

嵌入式系统测试

课程大纲

 嵌入式系统测试概述

 嵌入式软件测试技术概述

 嵌入式软件测试工具简介

History's Worst Software Bugs (1/3)

- 1962年7月，携带着金星探测器水手1号的大力神火箭从美国卡那维拉尔角空军基地发射升空。起飞后5分钟，火箭偏离预定轨道。在离星箭分离只有6秒钟时发出了自毁指令，在大西洋上空将整个火箭摧毁。
- 这次事故使美国损失了**1850万美元**，导致美国太空探险史上首次太阳系飞行计划化为泡影。
- 原因：遗漏了对火箭飞行速度取平均值。

History's Worst Software Bugs

Simson Garfinkel  11.08.05

Last month automaker Toyota announced a recall of 160,000 of its Prius hybrid vehicles following reports of vehicle warning lights illuminating for no reason, and cars' gasoline engines stalling unexpectedly. But unlike the large-scale auto recalls of years past, the root of the Prius issue wasn't a hardware problem -- it was a programming error in the smart car's embedded code. The Prius had a software bug.

TOP TEN MOST INFAMOUS SOFTWARE BUGS OF ALL TIME

Looking through some of my favorite articles of all time, I came across this jewel from 2005 - [Wired News's 10 Worst Bugs in History](#). I remember at the time I felt like their list was incomplete, and it has always bugged me a little bit (yes, pun intended). So I decided to do something about it. After taking a look at 20 or so of the worst software failures in history, I have compiled my own, updated, top ten list...

History's Worst Software Bugs (2/3)

- 1991年2月，爱国者导弹由于未能成功拦截飞向沙特宰赫兰兵营的飞毛腿导弹，致使美军**28名**士兵丧生，**98名**士兵受伤。
- 原因：跟踪软件每10秒误差0.000000095秒，出现0.36秒的累积时间误差，导致拦截飞毛腿导弹时偏离687米。



History's Worst Software Bugs (3/3)

- ❑ 1996年6月，欧洲航空航天局耗资**67亿**美元，历时10年研制的Ariane501火箭带着四颗卫星在法属圭亚那库鲁航天中心第三发射场首次升空。
- ❑ 起飞后37秒，火箭发生倾斜，大幅度偏离轨道，箭体结构断裂，随后火箭安全系统自动炸毁。
- ❑ 原因：惯性制导系统中的水平偏差量，由**64位**浮点数转换为**16位**带符号整数运算中发生溢出。



嵌入式系统测试的重要性

□ 一汽大众 召回6速自动变速箱（DSG）汽车2760辆

- ✓ 油液温度传感器上的插头处导线未卡紧，传感器可能发出错误温度信息，严重时会导致控制单元启动变速箱保护模式，暂时中断动力输出

□ 宝马 召回气门电控马达问题汽车5470辆

- ✓ 因软件错误，发动机气缸的气门电控马达可能出现不同步现象，将导致发动机工作不稳定，有时会熄火

□ 现代 召回悬架自动稳定控制缺陷问题车辆2017辆

- ✓ 在部分装备了3.3L发动机和电子稳定控制系统（ESC）车辆上，ESC可能设置得对过渡转向过于敏感，会对外侧前轮施加不必要

如何使嵌入式系统运行稳定、可靠是重要研究课题，而测试及调试技术是质量保障的基本手段！

测试 VS 调试

□ 测试与调试的关系

- ✓ 测试是检查软件发现问题，调试是分析系统解决问题
- ✓ 测试贯穿于整个系统研制周期，调试主要是在系统开发过程中
- ✓ 调试具有随意性、不明确性，而测试是明确的、可重复的
- ✓ 测试是全方位的，而调试是代码级的

□ 测试与调试不能相互替代，相互支持，相辅相成

嵌入式系统测试的若干原则

- 完全彻底的测试是不可能的
- Good-enough原则：权衡投入 / 产出比，既不要不充分，也不要过分
- Pareto法则：分析、设计、试验、复审等阶段能够发现和避免80%bug，系统测试能够找出其余80%bug
- 应尽早地开展软件测试：问题发现得越早，解决问题的代价就越小
- 排除测试的随意性，严格执行测试计划
- 测试是需要维护的，妥善保存测试计划、测试用例、出错统计和测试分析报告，为维护提供方便

关于测试设计

- 测试是需要设计的，因此测试是一项具有很大创造性的工作，其工作量一点也不比系统设计小。
- 测试与开发相比，并不低人一等。
- 测试的创造性主要表现在
 - 测试方案选择
 - 测试计划制定
 - 测试用例设计
 - 测试结果的分析
 - 测试过程的管理



关于测试用例

- 测试用例是一份关于具体测试步骤的文档，用于指导测试的实际操作。
- 测试用例可以是说明文档，也可以是脚本语言或高级语言编写的代码。
- 测试用例应包括
 - 测试用例名称及标识
 - 测试目的
 - 测试条件及设置
 - 输入数据要求
 - 预期的输出结果
 - 操作步骤

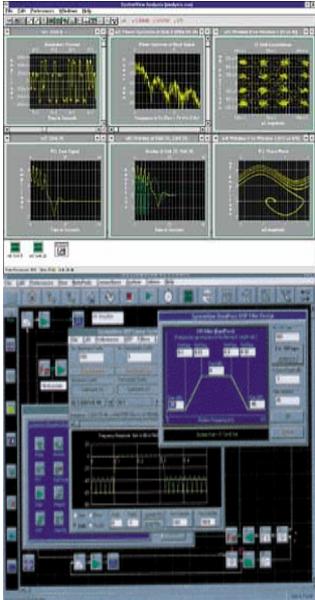
嵌入式系统测试方法

- 真实环境测试
- 交叉测试
- 数字化模拟测试
- 形式化验证



嵌入式系统测试方法——真实环境测试

采集/存储/分析



被测目标机



VITRA shown here with both debug and trace connections to ARM™ Integrator/CM-966E-S development board.

高速采集

外部激励及反馈

传感器、敏感器
A/D、D/A
1553、CAN总线
网络、RF
执行机构

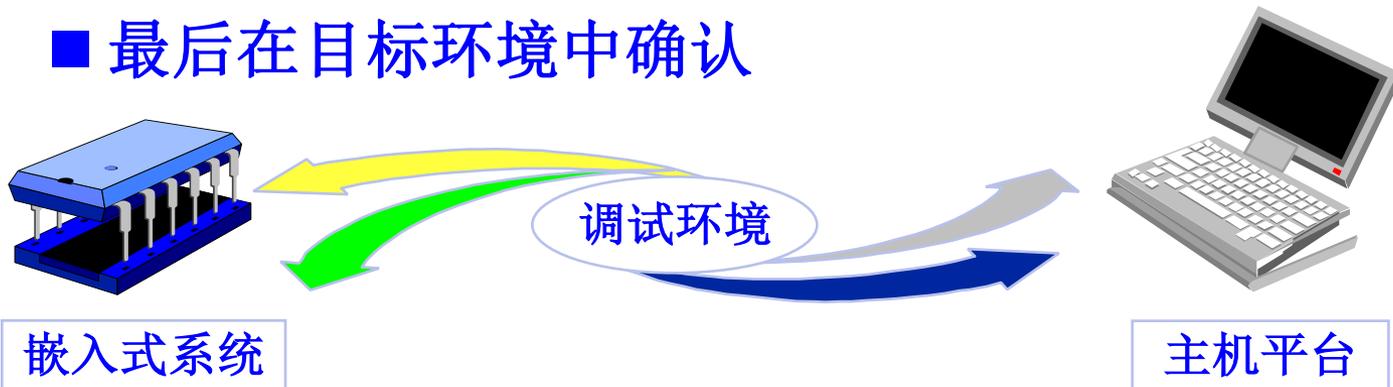


时序激励

嵌入式系统测试方法——交叉测试（1/3）

□ 交叉测试（Host/Target测试）

- 与目标环境无关的部分在PC机上完成
- 充分利用高级语言的可移植性
- 借鉴常规的软件测试方法
- 与硬件密切相关的部分在Target上完成
- 需要调试环境支持
- 测试工具需要支持目标环境
- 最后在目标环境中确认



嵌入式系统测试方法——交叉测试（2/3）

□ 交叉测试特点

- 将大部分工作转移到PC平台上，在硬件环境未建好或调试工具缺乏时就可以开展
- 适用于高级语言，如C、C++、Java
- 操作方便，测试成本较低
- 实时性受调试环境的制约
- 目标环境中测试时要占用一定的目标资源
- 注意目标环境和主机环境的差异

嵌入式系统测试方法——交叉测试（3/3）

□ 如何开展交叉测试

- 选用带有目标支持包的软件测试工具
- 确定哪些模块与硬件无关，哪些与硬件相关
- 配置相应的调试环境和目标环境
- 分别进行Host和Target测试
- Host：源代码+测试用例→编译连接→执行
→测试结果
- Target：源代码+测试用例+目标包→编译连接
→下载→执行→反馈测试结果

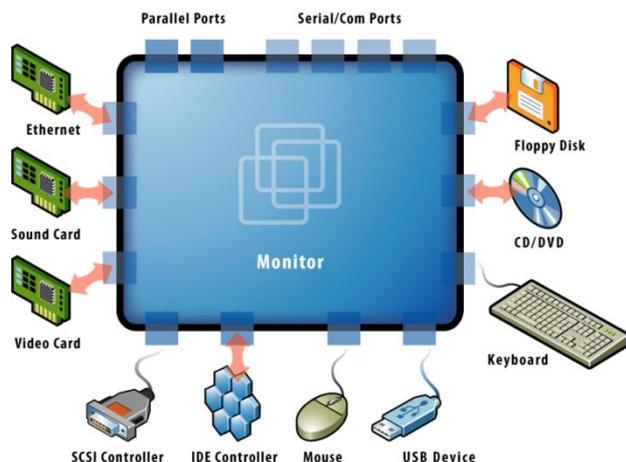
嵌入式系统测试方法——数字化测试（1/2）

- 数字化测试是指在宿主机上创建一个虚拟的目标机环境，实现对嵌入式系统的测试。
- 数字化测试可分为
 - 全数字化测试（全部器件数字化）
 - 半物理化测试（部分器件数字化）
- 数字化测试特点
 - 与嵌入式硬件平台脱钩
 - 操作简单，可以借鉴常规的软件测试方法
 - 易于开展故障注入测试
 - 适用于功能测试

嵌入式系统测试方法——数字化测试（2/2）

□ 数字化测试的局限性

- 实时性与准确性难以反映嵌入式系统的真实情况，测试出与时序有关的故障价值不大
- 理顺时序关系、维护统一精确的系统时钟较为困难，难以对并发事件、同步关系进行准确测试
- 主要作为嵌入式系统测试的辅助手段



嵌入式系统测试方法——形式化验证

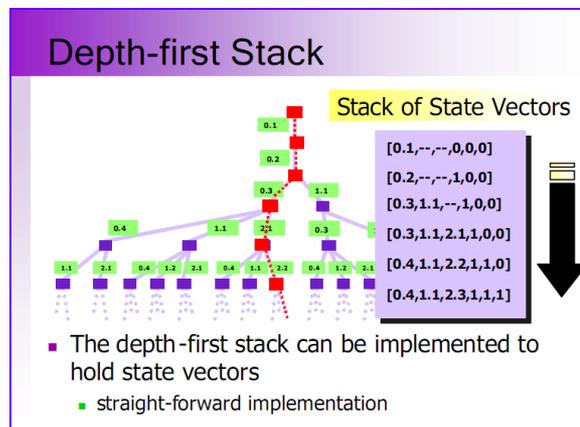
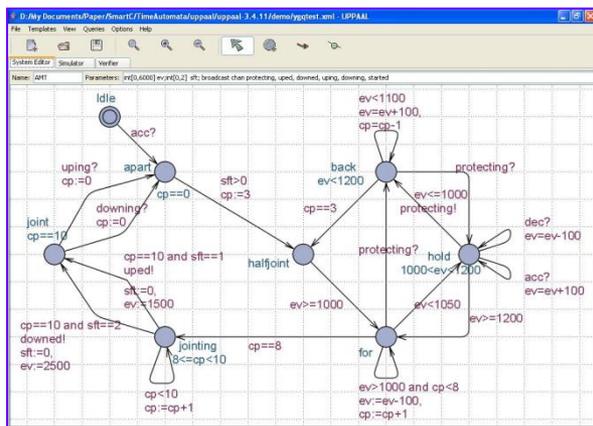
□ 形式化验证是指采用数学证明或逻辑推导的方式，验证嵌入式系统的正确性。

□ 例如

■ 任务可调度性分析：RM算法中任务可调度性分析的一个充分条件

$$\sum_{i=1}^n \frac{C_i}{T_i} \leq n \times (2^{\frac{1}{n}} - 1)$$

■ 系统状态可达性分析：基于模型检测的死锁分析



课程大纲

 嵌入式系统测试概述

 嵌入式软件测试技术概述

 嵌入式软件测试工具简介

嵌入式软件系统面临的严峻挑战

- 软件在嵌入式系统中所占有的比例越来越高，由于软件错误直接造成系统失效的比率持续递增。
 - 据1986年的统计数据表明，系统失效事件中诱因是软件错误的比率约占25%，而到2000年，该比率已超过40%。
- 嵌入式软件规模越来越大，功能越来越复杂。据统计，每1000行程序中，潜在bug数量约1~10个。
 - 国际空间站 ——软件系统规模大于1,000,000行
 - NASA太空飞船——船载代码量500,000行，地面3,500,000行
- 美国每年由于软件bugs导致600亿美元的经济损失，占GDP的0.6%。

可测试软件的特征

- 可操作性：运行得越好，被测试的效率越高
- 可观察性：每个输入有唯一输出、系统状态和变量可见、所有影响输出的因素都可见
- 可控制性：软件的控制越好，测试越能被自动化
- 可分解性：通过控制测试范围，能够更快地分解问题，执行灵活的再测试
- 稳定性：改变越少，对测试的破坏越小
- 易理解性：得到的信息越多，进行的测试越灵巧

软件测试方法分类（1/5）

□ 按照软件测试的动静态特性分类

- 静态测试（评审）

- 动态测试（测试用例/驱动程序/桩模块/测试监视代码）

□ 按照软件管理层面分类

- 技术评审

- 管理评审

软件测试方法分类（2/5）

□ 按照软件开发过程分类

- 单元测试（语句覆盖率100%，分支覆盖率85%）
- 集成测试
- 系统测试
- 验收测试

□ 按照软件产品类别测试分类

- 通用产品测试：功能测试、性能测试、接口测试、GUI测试、安装性测试、Alpha测试、Beta测试
- 专用产品测试：可靠性测试、标准符合性测试、安全性测试、强度测试、易用性测试、Benchmark测试

软件测试方法分类（3/5）

□ 按照测试用例所依据的信息来源分类

- 以软件结构为基础的测试（白盒测试）
- 以需求规约和需求描述为基础的测试（黑盒测试）
- 软件和需求相结合的测试（灰盒测试）
- 以接口为基础的测试（软件与运行环境接口）

软件测试方法分类（4/5）

□ 按照判断测试的充分性分类

- 功能测试
- 结构性测试
- 排错性测试
- 分域测试

□ 按照软件测试的完整性分类

- 图路径测试
- 程序路径测试
- 穷举测试

软件测试方法分类（5/5）

□ 白盒测试技术

- 语句覆盖率
- 分支覆盖率
- 条件覆盖率
- 分支/条件覆盖率
- 条件组合覆盖率

□ 黑盒测试技术

- 等价分类法
- 边界值分析法
- 因果图法
- 错误推测法

软件测试策略分析（1/6）

□ 静态分析很重要

- 动态测试比静态测试更重要，但事实并非如此
- 80%的软件错误归咎于对于编写语言的错误使用，而这些错误往往无法由功能测试完全解决

□ 静态分析的重要内容——代码规则检查

- 实施简单、方便
- 无需执行程序，与嵌入式环境无关
- 早期介入，代价小，见效快
- 有利于降低动态测试的难度



软件测试策略分析（2/6）

- 动态测试是验证软件功能最直接、最有效的手段
 - 通过运行被测程序验证其功能、性能，检查代码的执行情况
 - 与静态分析相辅相成
 - 需要事先设计详细、完备的测试用例
 - 可用白盒、黑盒相结合的方法开展动态测试
- 动态测试的主要内容
 - 功能、性能验证：是否符合需求规格定义
 - 代码覆盖：哪些代码执行，哪些没有执行，比例如何

软件测试策略分析（3/6）

□ 白盒测试、黑盒测试相辅成

- 白盒测试侧重于代码运行的过程，从其逻辑结构入手，对路径、控制结构、数据流等进行测试，通过插装检查程序的状态，确定是否与预期的状态一致
- 黑盒测试则不需要考虑程序内部逻辑结构，只依据需求定义，检查运行结果，多用于功能测试和性能分析

□ 常用黑盒测试方法——边界值分析

- 边界值分析是一种黑盒测试方法
- 经验表明：大量的错误发生在输入或输出的边界上

软件测试策略分析（4/6）

□ 单元测试、集成测试两步走

- 单元测试将被测软件分解为单元，逐个从程序的内部结构和功能出发设计测试用例，多个模块可以平行地独立进行单元测试
- 集成测试将所有模块按照设计要求组装起来测试，主要测试内容包括：接口间参数传递、集成的功能实现、模块间的影响等

软件测试策略分析（5/6）

□ 先静态，后动态

- 从代码规则检查做起
- 测试开展得越早，付出的代价就越小
- 为动态测试打下良好基础

□ 先单元，后集成

- 单元测试是集成测试的基础
- 单元测试得越好，集成测试的工作量就越小

□ 先黑盒，后白盒

- 先验证软件功能是否满足需求
- 后验证程序覆盖率

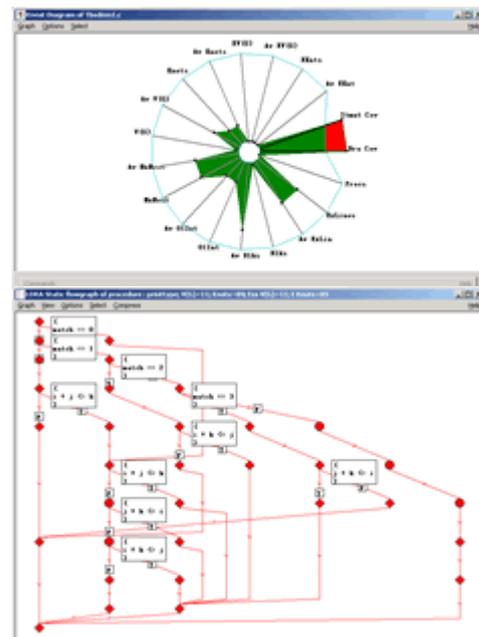
软件测试策略分析（6/6）

□ 圈复杂度和逻辑结构能客观的反映软件质量

- 逻辑越“复杂”，就越容易出错
- 结构越“良好”，代码就越可靠

□ 代码质量分析的好方法——结构化测试

- 从结构入手分析代码的复杂程度
- 逻辑复杂度定量化
- 复杂度与代码出错的关联性非常强
- 指导测试的执行
- 指出代码质量改进的方向



课程大纲

 嵌入式系统测试概述

 嵌入式软件测试技术概述

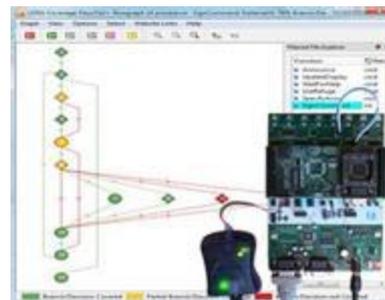
 嵌入式软件测试工具简介

嵌入式软件测试工具简介（1/8）

□ LDRA——软件自动化测试、验证

- 英国LDRA公司出品
- 支持整个开发周期的软件验证，LDRA套件包括 Testbed、Tbrun、TBeXtreme、TObject Box、Tbreq、Tbsafe、Tbpublish、Tbaudit、TBevolve等
- 提供需求跟踪验证、代码评审、设计评审、质量评审、单元测试、目标机测试、测试验证、测试管理

LDRA



嵌入式软件测试工具简介（3/8）

▣ McCabe IQ——结构化测试/质量分析

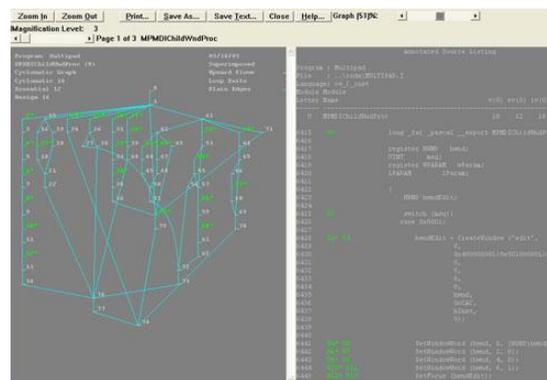
- 美国McCabe & Associates公司出品

- 适用于软件测试的不同阶段，主要构成：

 - ✓ McCabe QA（质量分析）

 - ✓ McCabe Test（追踪与测试）

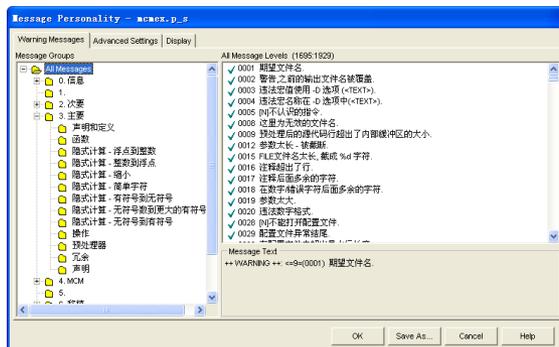
 - ✓ McCabe Reengineering（再工程）



嵌入式软件测试工具简介（4/8）

□ PRQA、QAC/C++——代码规则检查

- 英国Programming Research公司出品
- 代码规则检查支持C/C++、FORTRAN、JAVA、MISRA C/C++等国际标准及用户自定义的编程规范
- 提供多达44种业界广泛接受的度量
- 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、头文件关联、统计度量分析



嵌入式软件测试工具简介（5/8）

□ Tessy——自动单元测试

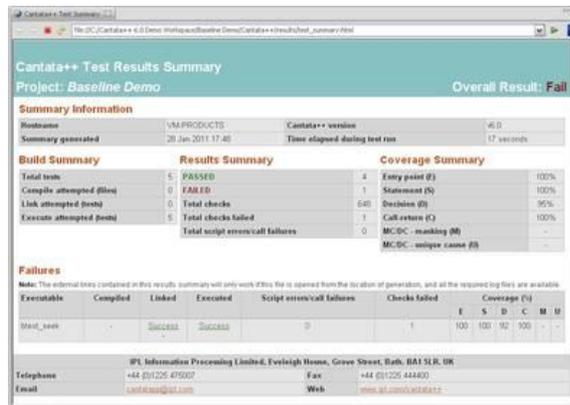
- 德国Hitex/Razorcat公司出品
- 支持C/C++语言目标系统的单元测试（动态测试）
- 分析源代码识别变量和类型，提供输入和输出的接口
- 自动生成基于Master/Slave结构的测试驱动
- 管理测试用例，允许无限多的用例，支持回归测试
- 自动生成标准的测试报告
- 各种数据类型智能分析，包括指针的支持
- 批处理功能可以在GUI和命令行下完成

嵌入式软件测试工具简介（6/8）

□ Cantata++——单元测试/集成测试

- 英国IPL公司出品
- 基于自动脚本技术的动态测试工具
- 支持动态/静态/覆盖测试，可自定义通过/失败标准
- 自动生成驱动程序和桩模块
- Wrapping技术，控制程序接口，实现逆向测试

IPL



Cantata++ Test Results Summary
Project: Baseline Demo Overall Result: Fail

Summary Information

Hostname	VM PRODUCTS	Cantata++ version	46.0
Summary generated	20 Jan 2011 17:48	Time elapsed during test run	17 seconds

Build Summary Results Summary Coverage Summary

Total tests	5	PASSED	4	Entry point (E)	100%
Compile attempted (#log)	0	FAILED	1	Statement (S)	100%
Link attempted (#obj)	0	Total checks	640	Decision (D)	95%
Execute attempted (#obj)	6	Total checks failed	1	Call return (C)	100%
		Total script errors/call failures	0	MC/DC - ranking (M)	-
				MC/DC - unique cases (U)	-

Failures

Note: The external links contained in this results summary will only work if this file is opened from the location of generation, and all the required log files are available.

Executable	Compiled	Linked	Executed	Script errors/call failures	Checks failed	Coverage (%)	E	S	D	C	M	U
Test_000	-	Success	Success	0	1	100	100	92	500	-	-	-

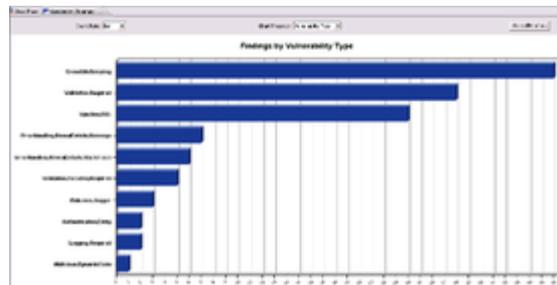
IPL Information Processing Limited, Eveleigh House, Grove Street, Bath, BA1 1LR, UK
Telephone +44 (0)1225 475007 Fax +44 (0)1225 444800
Email info@ipl.com Web www.ipl.com/cantata++

嵌入式软件测试工具简介（7/8）

□ Klocwork——软件缺陷检查

- 加拿大Klocwork公司出品
- 能够自动分析C、C++、C#和Java代码
- 能够发现的软件缺陷种类：软件质量缺陷、安全漏洞方面的缺陷、软件架构、编程缺陷的违反情况
- 对软件进行可视化的架构分析、优化
- 对软件进行各种度量分析

Klocwork



嵌入式软件测试工具简介 (8/8)

CasePlayer

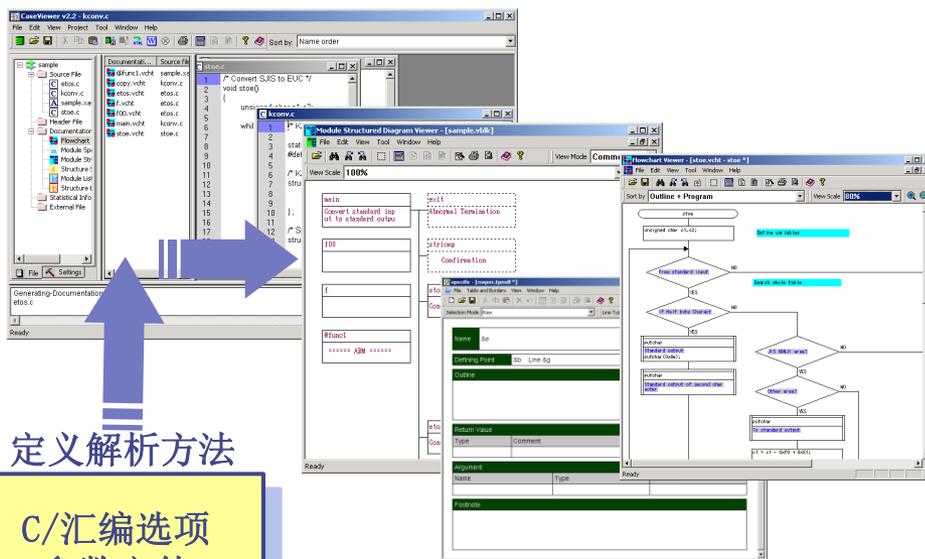
- 面向嵌入式开发的程序规格说明书制作工具
- CasePlayer2是一种可以解析汇编、C语言源代码，并且制作流程图以及模块规格说明书的制作工具 打印

嵌入式
C源程序
和头文件

各类MPU
汇编源程序

定义解析方法

C/汇编选项
参数文件



组合各类文档、源文件的专用阅读器



谢谢!

