IBM Software Group

IBM Rational嵌入式实时软件开发测试全面解决方案

Rational. software

IBM 软件部　李卫锋

ON DEMAND BUSINESS

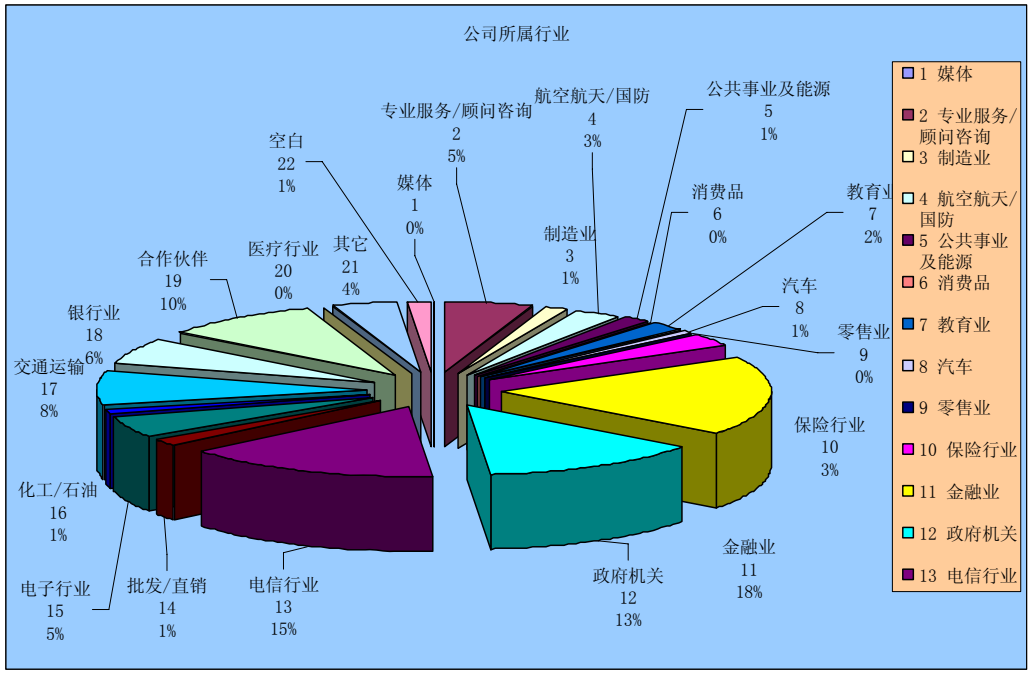
© 2007 IBM Corporation

标题

- 议题一
  - 嵌入式系统开发面临的挑战
  - 如何应对这些挑战?
  - IBM Rational针对嵌入式软件开发开发的测试解决方案
  - 案例介绍
  - 嵌入式系统开发的测试解决方案ROI
  - 总结及Q/A
- 议题二
  - Test RealTime实现针对嵌入式软件开发的运行时刻分析
    - TDP
    - 运行时刻分析
    - DEMO
  - Q/A

嵌入式实时软件开发数据统计

- 超过 50% 的嵌入式开发项目进度延期
- 约25%的嵌入式开发项目失败
- 只有44%的设计满足了20%的需求
- 超过50% 的开发工作量用于测试

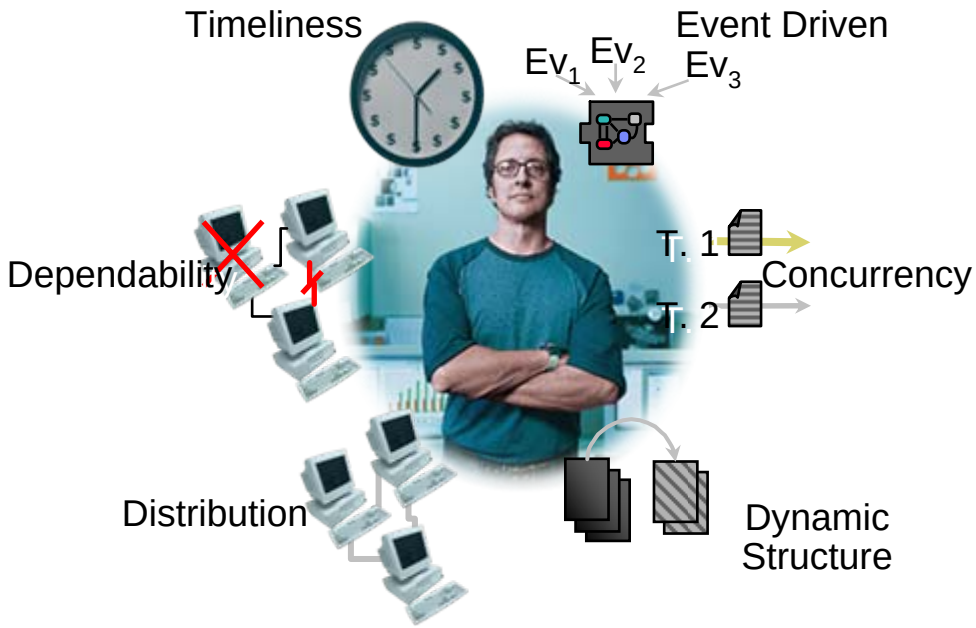


数据来自2006年RSDC

嵌入式实时软件开发的特征

- **应用的复杂性**
  - 时序配合要求高
  - 低内存占用
  - 并发/分布/网络
- **环境的复杂性**
  - RTOS/IDE/Chips厂商的多样性
  - 主机和目标平台的连接能力有限
  - 内建的排错能力较弱
- **流程的复杂性**
  - 需求
  - 设计、转换的错误
  - 难于维护
  - 低性能

嵌入式实时系统与生俱来的复杂性



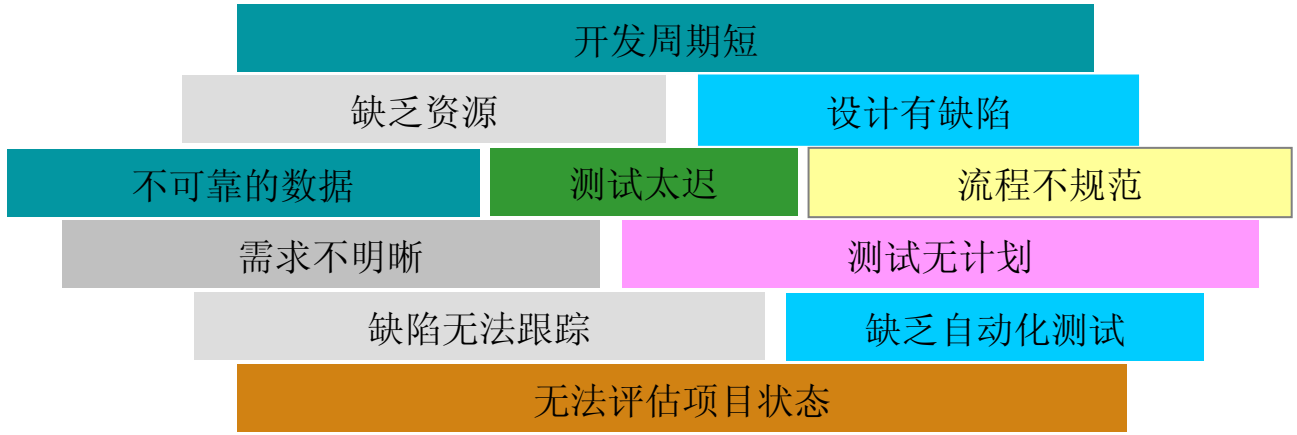
嵌入式实时软件开发的面临的挑战

业务驱动的特征日益明显

- **广泛的应用领域**
  - 国防科技领域：空间、陆地、海洋、空中、通信……
  - 民用领域：工业生产、电子消费品、智能家电、民用通信……
- **竞争激烈的生存环境**
  - 技术的进化的效率将直接影响在某个领域的竞争力
  - 新产品推出的速度将直接影响市场占有率：例如手机市场
- **相对落后的开发手段**
  - 基于代码的开发手段：影响开发能力，降低产品进化能力
  - 相对落后的测试手段使得开发的效率和成本居高不下
  - 开发过程需要得到有效的改进
    - 基于资产的开发替代传统方法
    - 需要迅速提高开发管理能力

嵌入实时软件开发面临的挑战

建立一个高效的测试框架非常困难

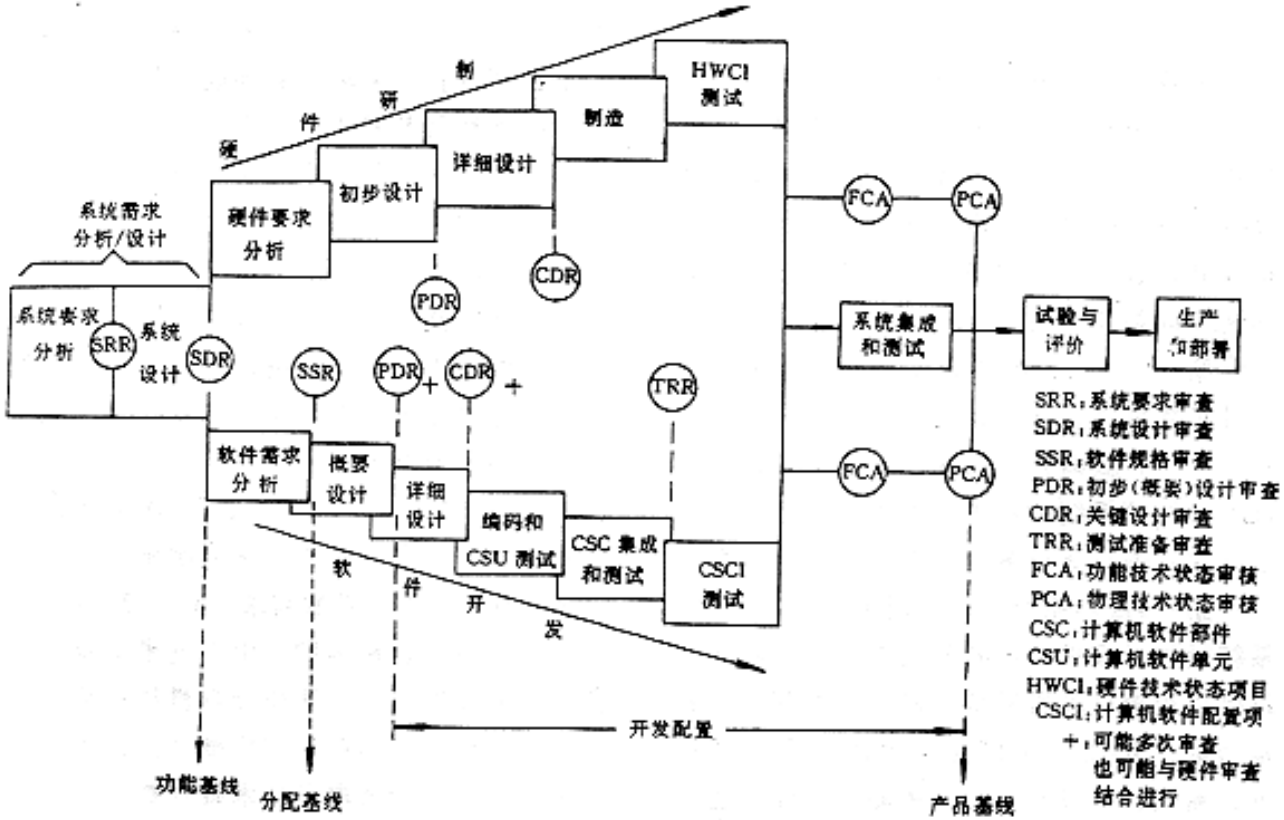


- 进行完整的手工测试非常困难
  - 开销大量的工作量?
  - 测试难于被重用?
- 测试的创建和执行都较为困难
  - 需要你付出太多的精力，甚至对于紧张的进度计划产生影响?
  - 要在源代码层面和系统层面都进行有效的质量保证需要开销太多的工作量?
  - 编写测试驱动工作量大
  - 通过手工插帧（在代码中加入调试指令）来获得被测系统内部运行行为，效率低
  - 测试手段和开发环境集成性不好，使用不方便
  - 执行和报告生成依赖于手工

嵌入实时软件开发面临的挑战

建立一个高效的测试框架非常困难

嵌入式开发把软件系统作为特定硬件系统的神经系统来对待，并予以整体综合考虑。把软件研制过程分作五部分，按时间顺序进行开发，反映了软件研制的生命周期演进的观点，即软件工程学的瀑布模型法。



标题

- 第一天上午
  - 嵌入式系统的开发面临的挑战
  - 如何应对这些挑战?
  - IBM Rational针对嵌入式软件开发开发的测试解决方案
  - 案例介绍
  - 嵌入式系统开发的测试解决方案ROI
  - 总结及Q/A
- 第一天下午
  - Test RealTime实现针对嵌入式软件开发的运行时刻分析
    - TDP
    - 运行时刻分析
    - DEMO
  - Q/A

如何应对这些挑战？

如何提高产品的质量？

- 对于各种复杂度水平的组件进行自动化的测试
  - 对于从最简单的功能到分布式的系统都能够进行完全自动化的测试脚本、测试桩、测试驱动生成和部署
  - 通过运行时分析能力加速应用排错
  - 内存和性能的概要分析，代码覆盖率分析和运行时跟踪的特性加速质量改进
- 让测试成为开发工作的主要部分
  - 建立被测代码、测试结果和可视化的模型之间的动态连接
- 测试优先级排定
  - 代码复杂度和静态指标的计算
- 完全可重复的测试
  - 完全自动化的回归测试

如何应对这些挑战？

这些能力能否帮助我们解决在真正的目标平台所需要进行的验证工作？

- 一个集成化的、灵活的框架完全足够提供对你所选择的编译器、链接器、排错工具和目标体系架构的支持
  - 在实际操作的条件下产生可执行的测试套件
  - 在目标环境发生变化时无需变更测试场景
- 在目标平台上能够进行自动化测试的下载和测试执行和结果的上载
  - 将开发人员从管理目标平台的繁琐中解放出来
  - 帮助开发团队将主要的精力集中在测试的设计上
- 细小的最小化的验证插针和测试驱动
  - 解决小内存和有效的CPU资源的问题
  - 鼓励进行脱离开目标平台约束的模拟条件验证

如何应对这些挑战？

以模型为中心的开发流程？

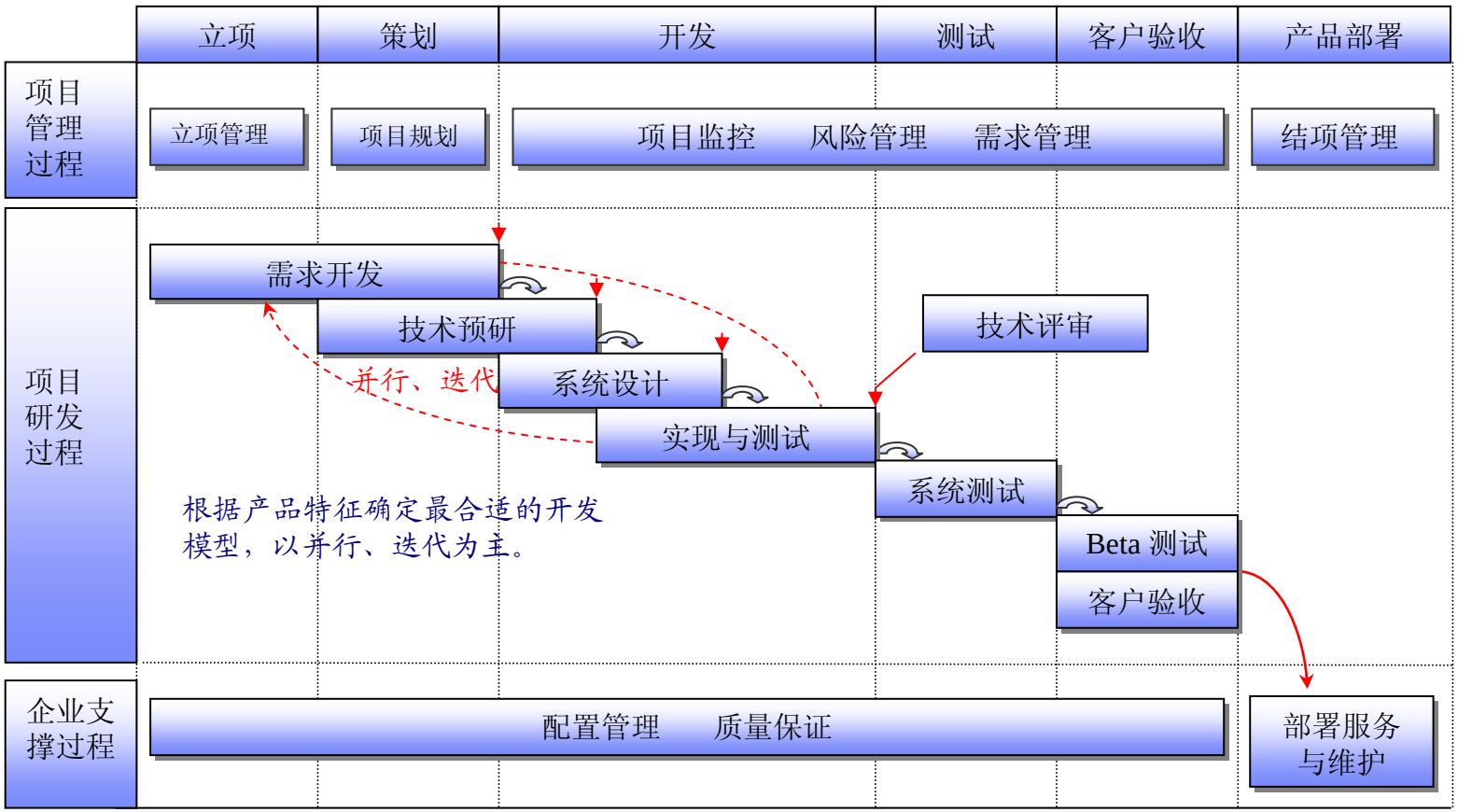
- 建立模型驱动开发流程和相关的开发平台
- 在模型的层面进行开发
- 在模型层面进行应用的测试和排错
  - 图形可视化那些测试中达到的或者没有达到的应用状态和转换
  - 在模型环境中的对应用的代码部分跟踪内存错误和性能
  - 采用UML近似的图形格式跟踪应用的执行流

标题

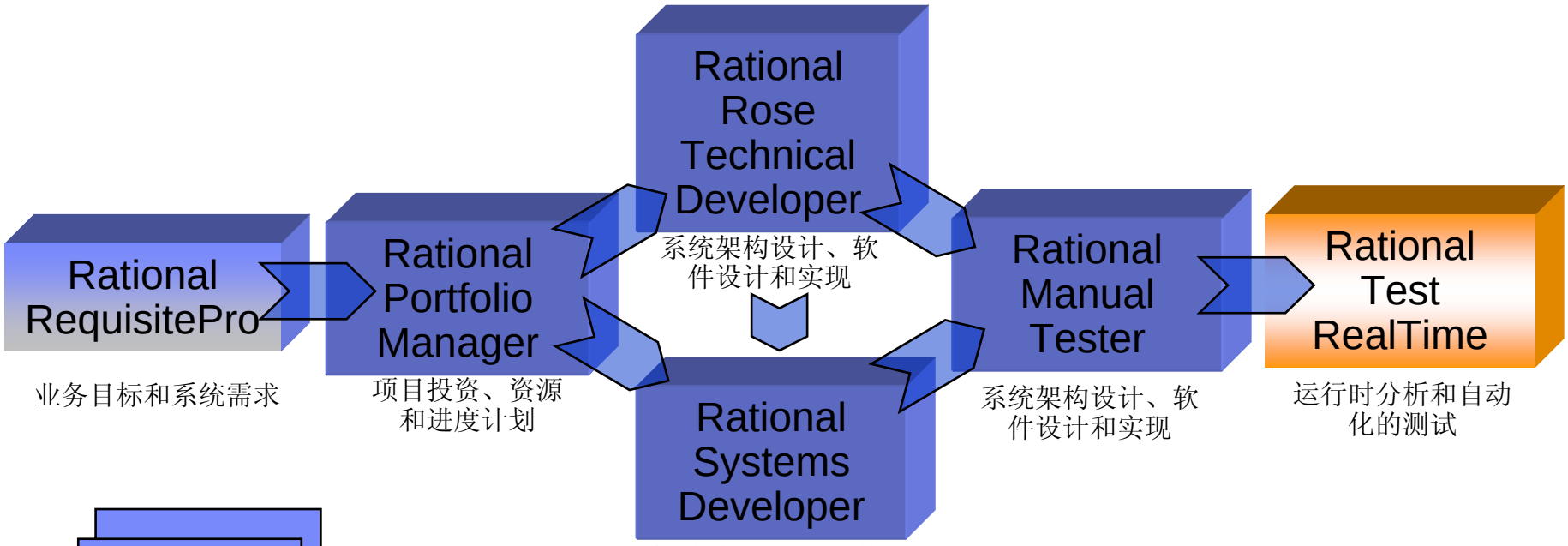
- 第一天上午
  - 嵌入式系统的开发面临的挑战
  - 如何应对这些挑战?
  - IBM Rational针对嵌入式软件开发开发的测试解决方案
  - 案例介绍
  - 嵌入式系统开发的测试解决方案ROI
  - 总结及Q/A
- 第一天下午
  - Test RealTime实现针对嵌入式软件开发的运行时刻分析
    - TDP
    - 运行时刻分析
    - DEMO
  - Q/A

嵌入式实时软件系统开发的生命周期

迭代化开发　以架构为中心　持续地质量验证　管理软件资产和变更

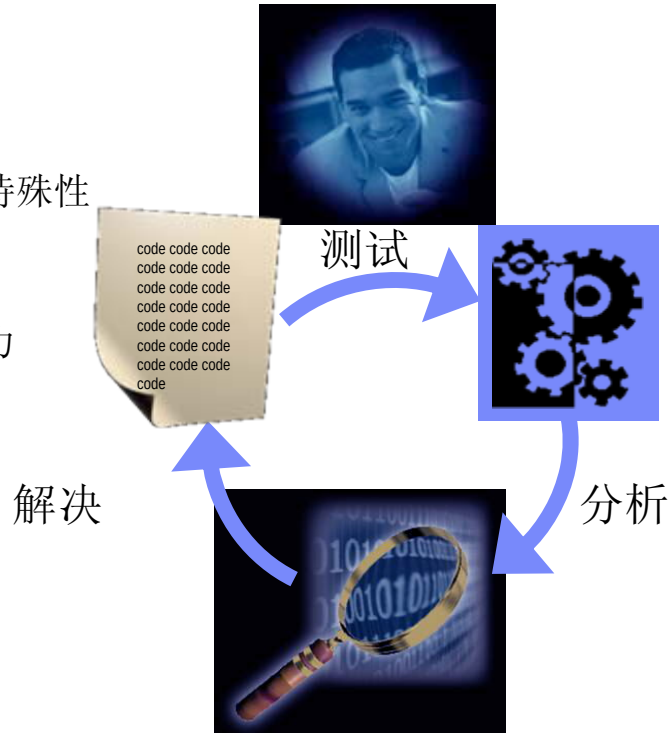


嵌入式实时软件系统开发的生命周期



IBM Rational Test RealTime，在出问题之前解决它

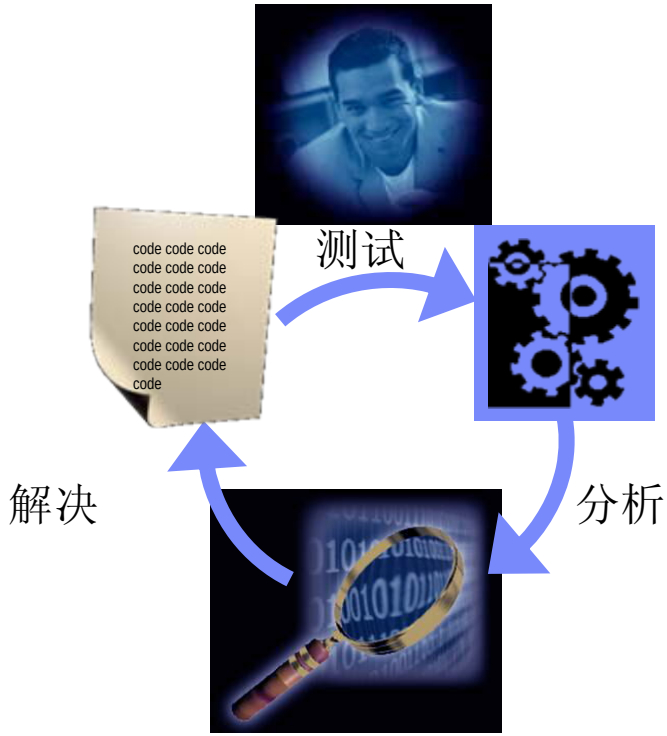
- Rational Test RealTime是一个针对嵌入实时软件产品的代码编写人员设计的交叉平台测试解决方案
  - 在开发生命周期的早期阶段就能够自动化地测试、分析、解决产品的错误
  - 通过特殊技术来支持独立于目标平台的测试
  - 利用可定制的TDP来屏蔽不同嵌入式软件开发时语言特殊性和开发工具的差异性
  - 通过提供运行时分析的能力扩展了模型驱动的开发能力



IBM Rational Test RealTime

自动化软件测试过程

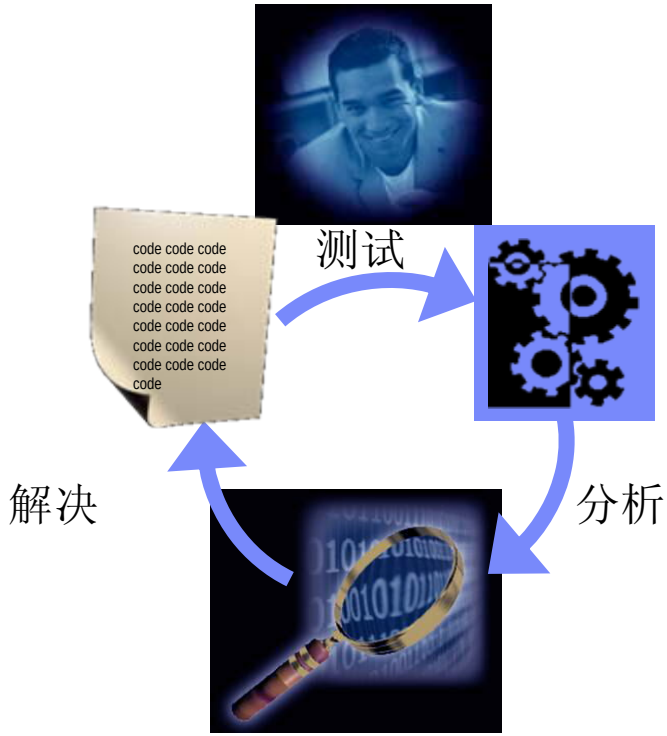
- 全面的自动化测试
  - 自动生成测试脚本模版和测试数据
  - 黑盒测试和白盒测试相集成
  - 多层次测试：从函数到分布式系统
  - 静态分析功能：
    - 明确测试优先级
    - 代码复杂度计算
  - 全面的回归测试



IBM Rational Test RealTime

自动化软件测试过程

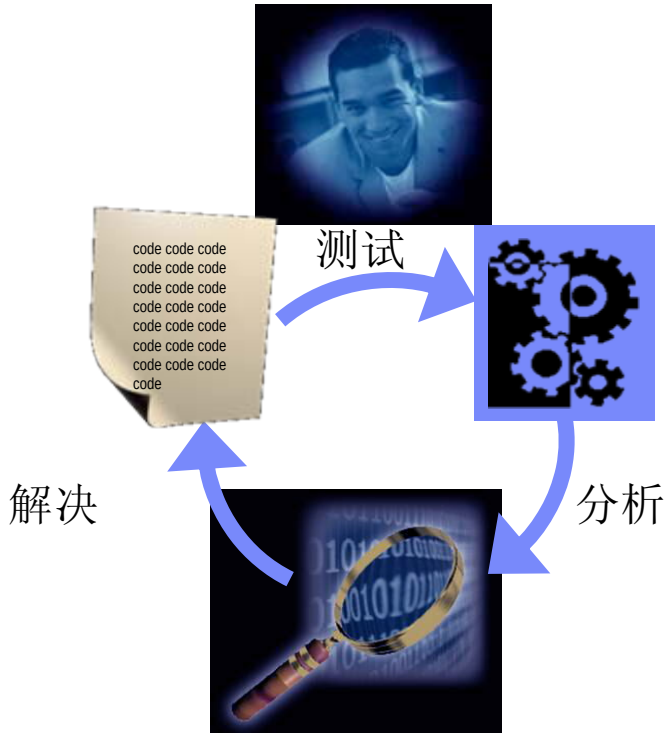
- 全面的自动化测试
- 全面的运行时分析
  - 代码覆盖率分析
  - 内存分析
  - 性能分析
  - 运行时分析
  - 线程分析



IBM Rational Test RealTime

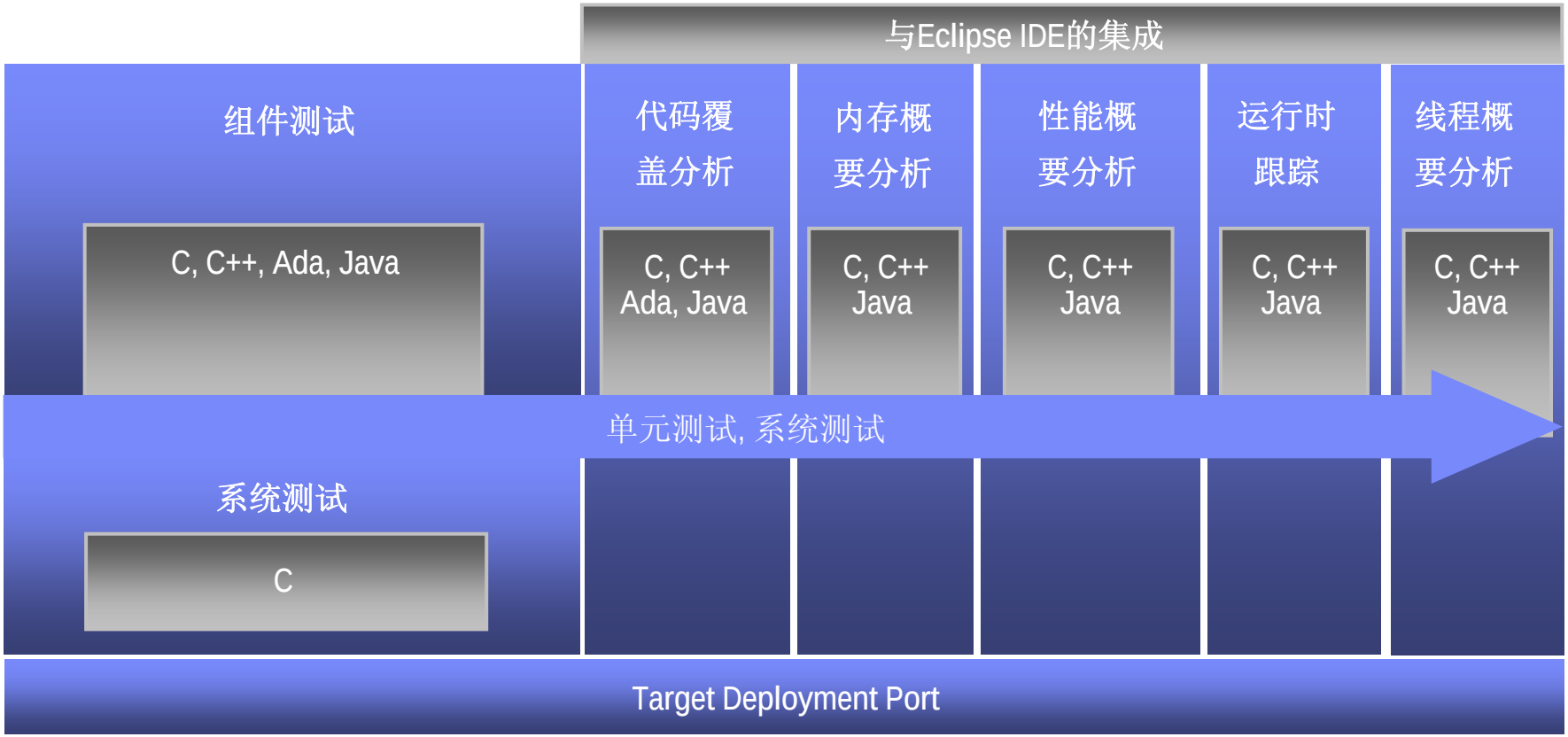
自动化软件测试过程

- 全面的自动化测试
- 全面的运行时分析
- 直观的测试结果
  - 测试执行和调试器集成
  - 统一详细的测试报告
  - 运行时分析数据和测试数据相关联



在开发过程中测试、分析并解决错误

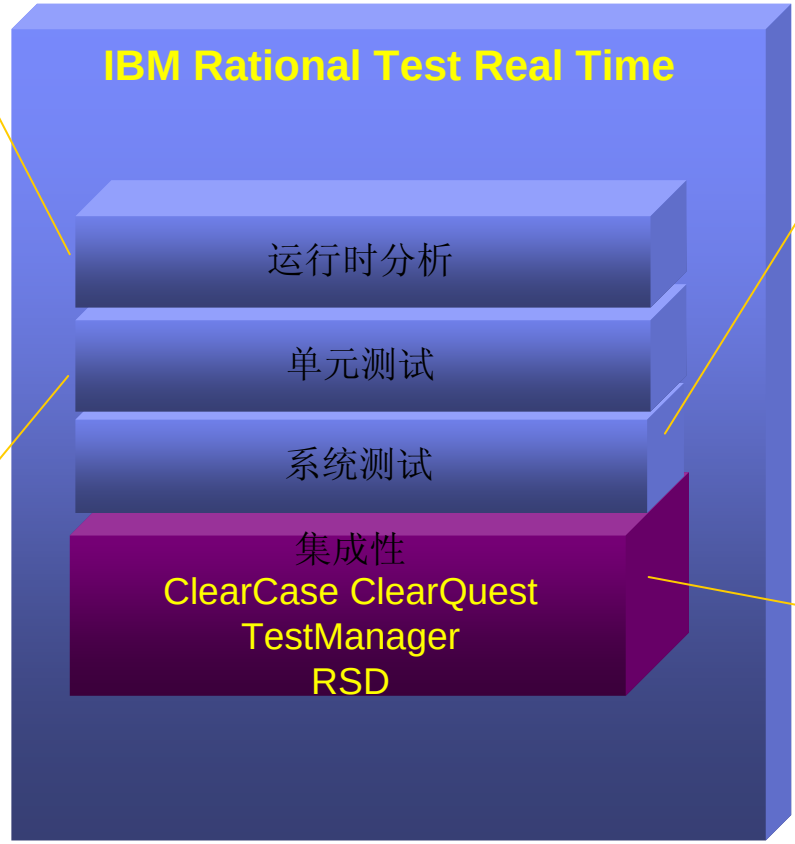
同时支持开发机和目标的全面测试方案



- 对于标准的遵从
  - DO-178B (航空电子), MISRA (汽车), DEF STAN 00-55 (国防),CMMI,GJB5000

IBM Rational Test RealTime(RTRT)

获取被测系统运行时的动态时序图、内存错误、代码覆盖率、系统调用图及性能信息



通过模拟其他系统（线程、进程或应用程序）与被测系统的通信，完成对被测系统的集成功能测试

生成被测单元的驱动函数、桩函数，以及测试脚本，用户定义被测单元的输入/输出数据，完成对该单元的功能测试。

RTRT能够很好的和ClearCase, ClearQuest, TestManager, RSD集成。

IBM Rational Test RealTime

广泛的平台支持

4-Bit to 64-Bit Cross-Development Environments		
WindRiver	*Montavista*	*Tasking*
GreenHills	*TI*	*CAD-UL*
ARM	*NEC*	*Cosmic*
Enea	*Hitachi*	*Hiware*
Windows CE	*Apex*	*Hitex*
LynuxWorks	*Sun*	*Symbian*
Lauterbach	*Microtec*	*……*

Languages

- C
- C++
- Ada
- J2ME/J2SE

Platforms

- *Windows*
- *Solaris*
- *RedHat*
- *HP-UX*
- *AIX*

标题

- 第一天上午
  - 嵌入式系统的开发面临的挑战
  - 如何应对这些挑战?
  - IBM Rational针对嵌入式软件开发开发的测试解决方案
  - 案例介绍
  - 嵌入式系统开发的测试解决方案ROI
  - 总结及Q/A
- 第一天下午
  - Test RealTime实现针对嵌入式软件开发的运行时刻分析
    - TDP
    - 运行时刻分析
    - DEMO
  - Q/A

案例背景介绍

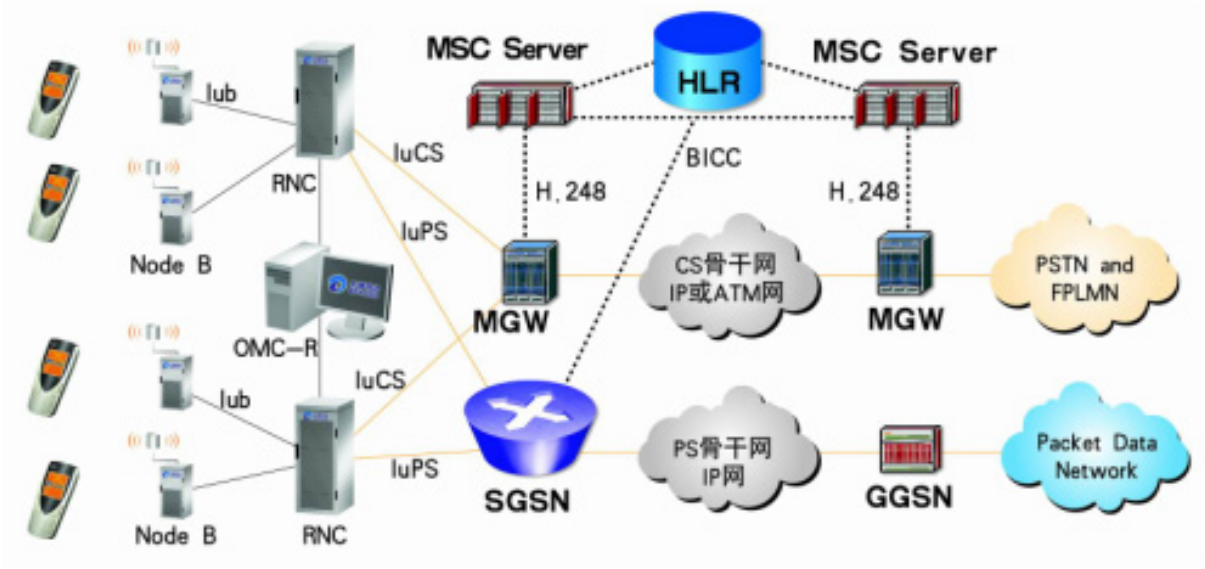
- A公司是XX科技产业集团的核心企业之一，并形成以北京为总部，西安、上海分设分、子公司的组织架构.
- 主要开发产品包括：
  - 开发第三代移动通信系统CDMA交换机、无线网络控制器（RNC）设备
  - 从事ATM交换机的开发
  - 承担 “第三代移动通信系统” 863计划项目
- 主要业务方向：
  - 3G核心网的预研和集成
  - 致力于3G和3G后的无线传输解决方案，提供CDMA全套系统产品及服务.
  - 公网、专网的客户应用服务和为客户提供全面解决方案

基于CDMA的3G产品架构

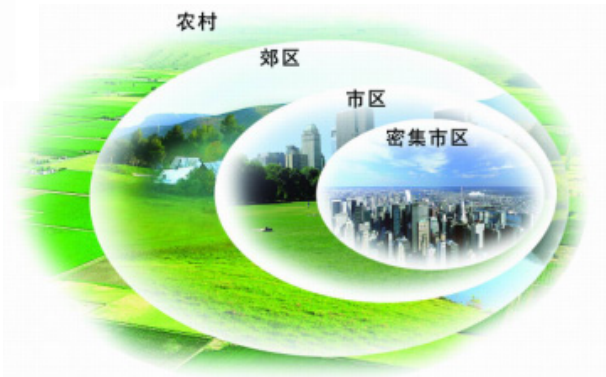
- 3G网络组网：

  依据CDMA系统具有的强大独立组网能力，提供经济高效的组网解决方案和平台.

- 产品架构：



整个基于CDMA的系统包括3G交换机（MSC）、基站控制器（RNC）、基站子系统（Node B）和移动终端等部分组成。



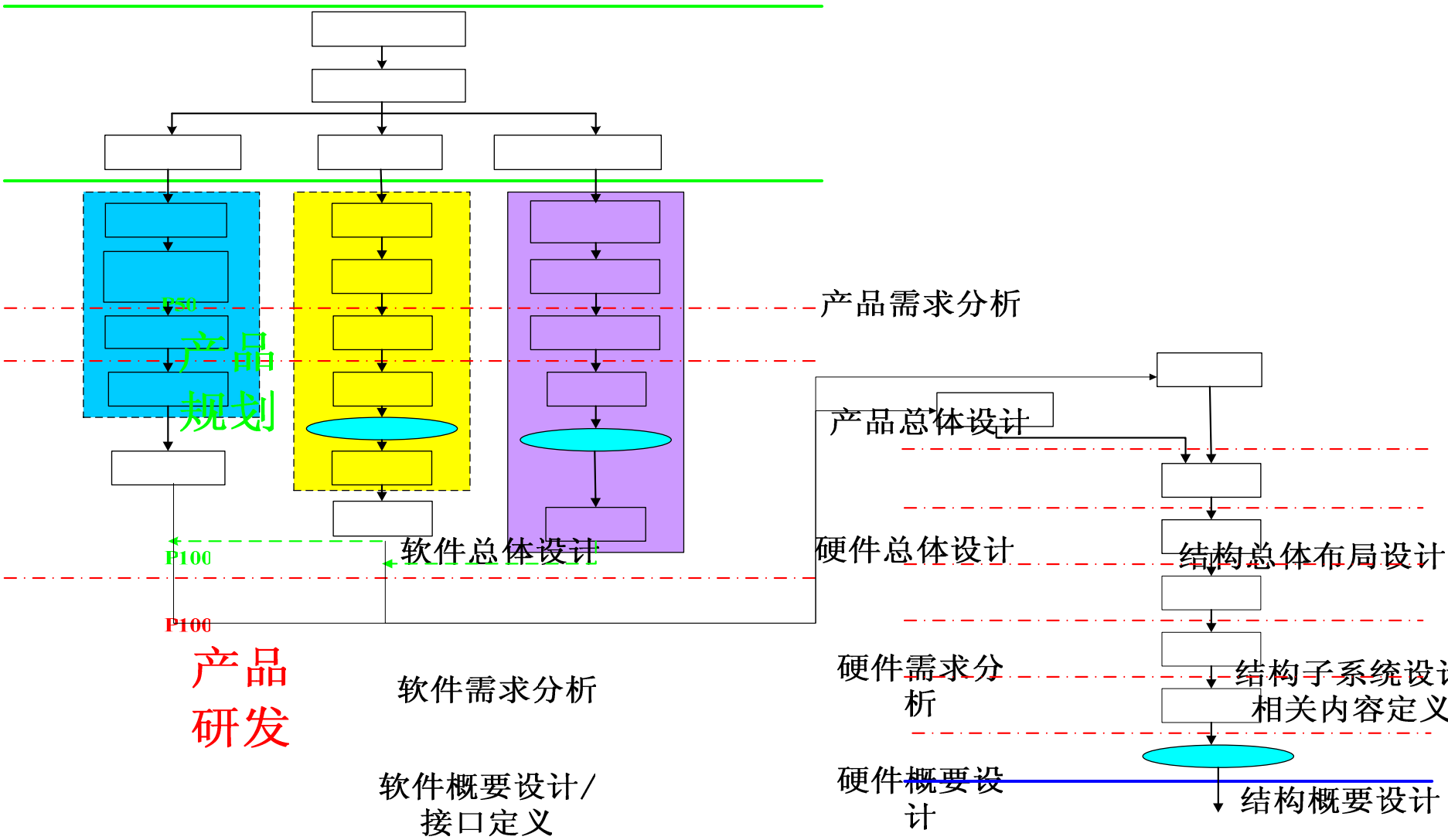
A公司面临的主要挑战

- 3G牌照即将发放，使公司的产品在外场测试方面要应对很大的压力,如何保障嵌入式软件产品版本的正确性并与硬件产品保持一致性,以及如何提高产品整体质量是首要问题.
- 引入了IPD的理念，并吸纳了ISO9000、CMM以及企业标准化等思想. 但这些只是定义了一些标准，标准中的很多目标都需要工具的支持,但客户整个生命周期的工具平台较为欠缺.而且以前使用的工具主要是OpenSources的,无法适应公司的业务发展需要.
- 市场对于产品的新特性的要求增长迅速，需要准备去应对更为大型化、复杂化的产品的研发，需要有效管理更为大型的项目和更为复杂化的产品的研发。

案例介绍

- 案例项目名称：HSPS软件子系统(RNC的核心软件,以下简称HSPS)
- 案例属性：
  - HSPS是组成UMTS网元RNC的软件组件之一--信令处理子系统。HSPS子系统实现：小区资源管理、寻呼消息处理、系统信息消息广播、无线链路的管理、RRC连接的管理、RB的管理、RAB的管理、NAS消息直传处理、标准消息的编解码、接口/资源复位、Iu连接的管理、安全模式控制、终端切换管理、小区更新、用户面配置、测量控制、协议参数的配置等。

HSPS项目开发总流程



生命周期阶段

6.4 设计实现阶段

- 阶段名称：设计实现
- 里程碑标记：*P330*
- 活动说明：

  软件开发项目的设计实现阶段主要是指对各模块或单元、子系统以及产品的具体实现，在此阶段生成原代码、可执行程序、数据库（如果要用的话）还有相应的编程标准。此外在该阶段还要进行必要的单元测试，也包括后续测试和验证目标的确定，测试方法和测试用例的选取等等。

- 输入：

  软件详细设计报告、模块间接口手册

- 活动：
  - 根据要实现的模块的详细设计或系统设计或系统功能/需求规格说明定义代码结构；
  - 逐步编写合格的代码，并进行模块单元测试/调试。
- 输出：

序号	文档名称	文档说明	标记
1	单元测试报告	单元测试结果说明	
2	软件代码、子系统预集成测试脚本		

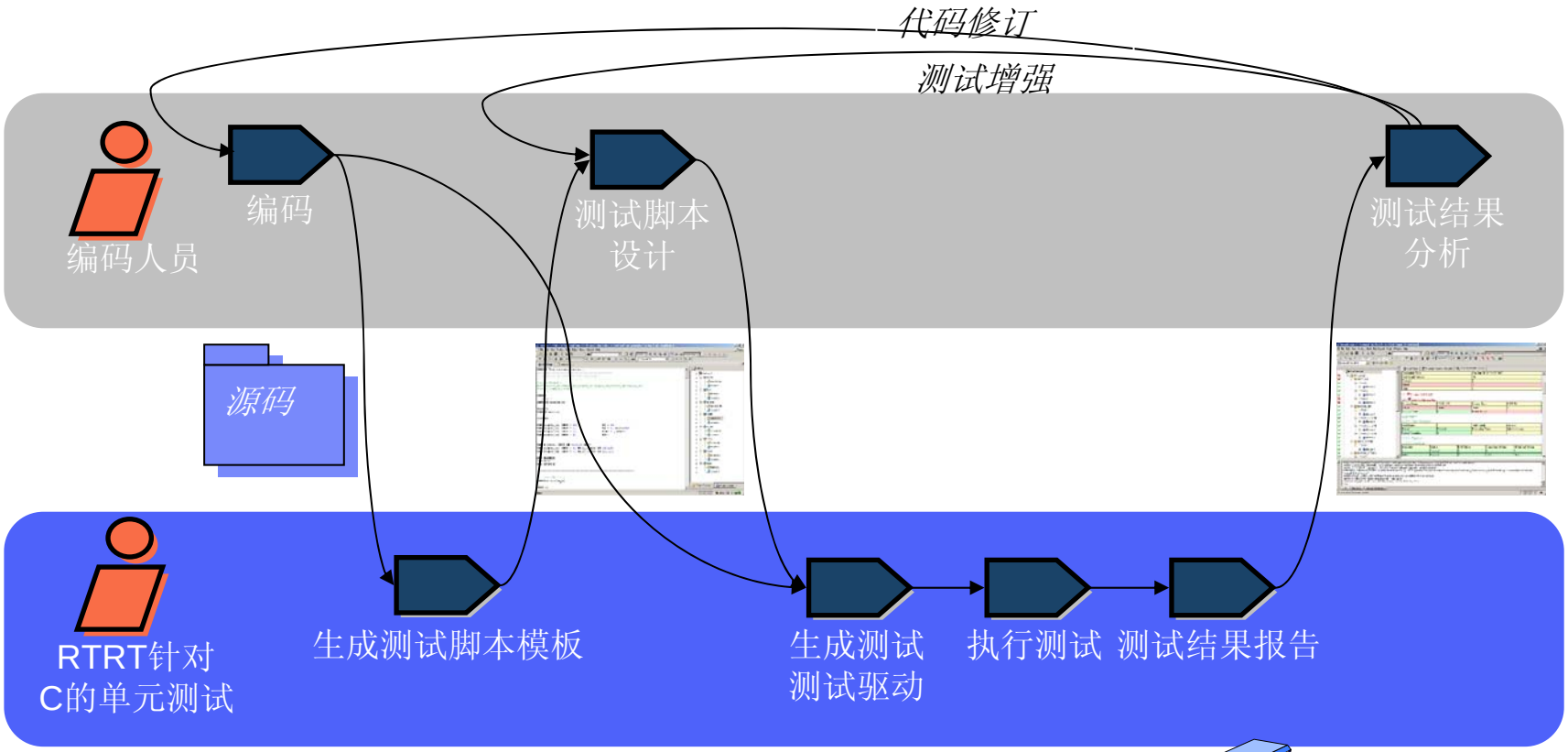
- 结束条件：

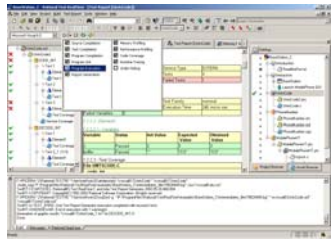
  源代码没有违反编程规范，模块单元测试结束，模块软件提交子系统预集成测试。

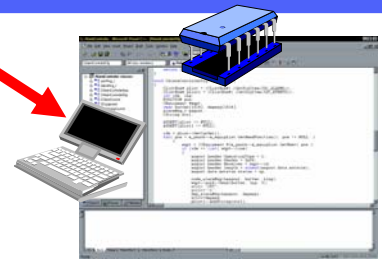
Test RealTime: *单元测试*

- 自动化
  - 生成桩模块和驱动模块
  - 生成测试模板
  - 生成测试用例组
  - 运行测试
  - 代码插桩，用于运行分析
  - 生成测试报告
  - 回归测试
- 详细的报表功能
- 使用静态测试划分测试优先级
  - 软件复杂级别
- 使用Test RealTime Runtime Analysis features
  - 内存 和性能分析
  - 覆盖率和运行时跟踪

单元测试流程

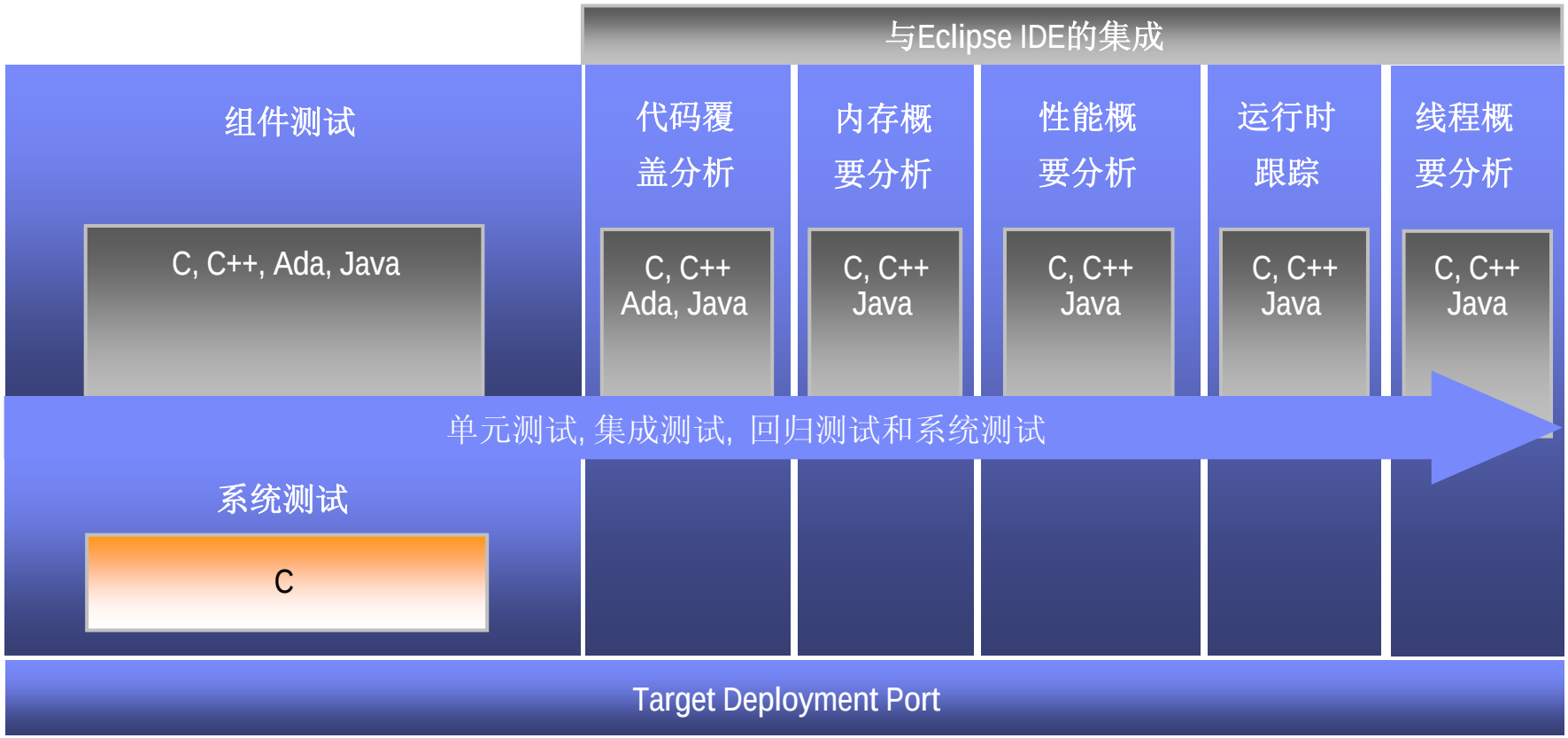






编译 /
连接 /
调试
模拟 /
仿针 /
目标

Test RealTime:系统测试



Test RealTime:系统测试类型

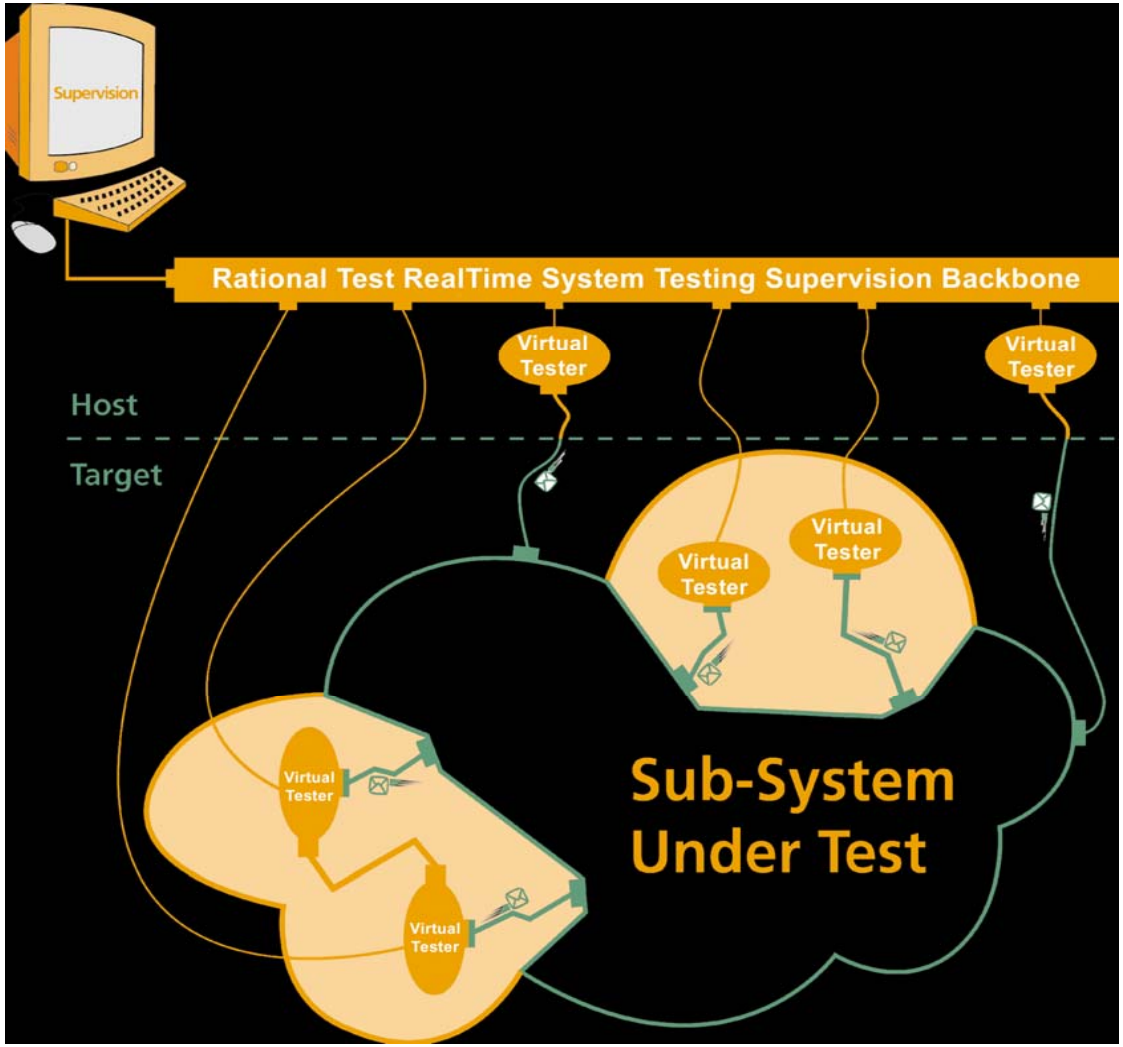
- **系统白箱测试**
  - 系统白箱测试(集成测试)是对系统集成构建后进行的测试,主要目的是测试系统间接口，使用白箱方法，关注一个系统(或子系统或模块组)内部过程，主要对其各项特性功能进行测试，确定是否满足设计的功能与特性，是否出现设计中不允许的缺陷。
- **系统黑箱测试**
  - 系统黑箱测试是对整个系统产品进行集成后进行的测试,主要目的是测试系统是否能够满足需求分析阶段确定的需求，面向用户的交付物是否完备，是否与系统实际实现一致，是否出现需求中未定义的缺陷。

系统测试-C: *基于消息的分布式系统测试*

- 对以下待测对象进行集成测试和确认性测试:
  - 单个线程 …
  - 单个或多个任务 …
  - 单个或多个物理节点…
  - 直至大型网络系统
- 通过消息传递函数，进行功能测试、压力测试和性能测试
- 脚本语言
- 详细的测试报告
- 回归测试
- 可以和运行时分析功能结合使用
  - 内存 和性能分析
  - 覆盖率和运行时跟踪

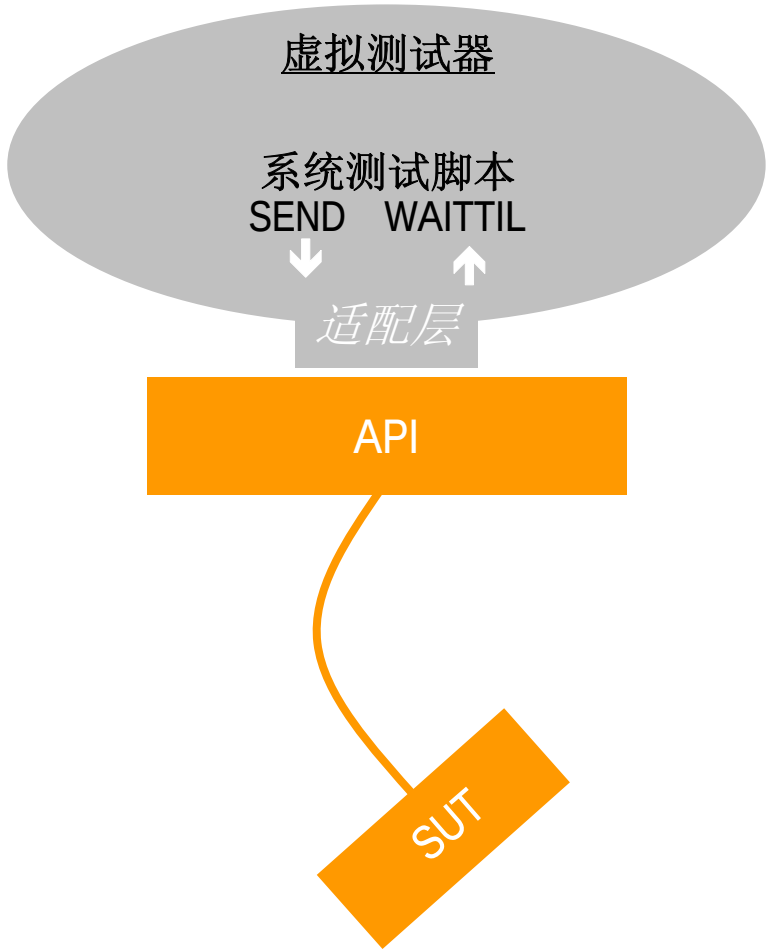
测试环境体系架构和功能

1. **虚拟测试器**
   - 模拟外部系统
   - 虚拟内部的桩模块
2. **每个虚拟测试器**
   - 发送事件到待测系统
   - 控制事件流
   - 验证事件数据和时间
   - 可再现的压力测试
3. **系统测试监控器**
   - 监控服务程序，用于虚拟测试器事件分发、通讯、同步



系统测试-C: *消息适配层*

- 支持各类通讯界面
  1. 适配层
     - 用C程序创建发送和接受事件
     - 事件的符号管理
     - 使测试脚本独立于消息API
  2. 消息API
     - 待测系统的一部分
     - 由一个通讯模块实现
     - 定义如何发送和接受消息



HSPS软件子系统集成测试流程

6.5 软件预集成测试阶段

- 阶段名称：软件预集成测试
- 里程碑标记：*P350*
- 活动说明：

  软件开发项目的预集成测试阶段包括各子系统预集成测试、子系统间的联调测试和冒烟测试三部分，在此阶段主要是针对软件模块、子系统的联合调测，保证基本功能能够初步运行，为下一步系统集成测试、系统测试创造条件。因此该阶段要进行工作有：根据既定的测试目标、测试方法，进行测试用例的选取、执行，记录软件运行状态、跟踪缺陷解决等等

- 输入：

  软件详细设计报告、模块间接口手册、单元测试报告

- 活动：
  - 子系统集成（预集成）测试、：在实体（如整个 NodeB 或 RNC）完全集成前的子系统的集成测试，包括软件和软、硬件间的集成。
  - 子系统间软硬件联调（联调测试和冒烟测试）
- 输出：

序号	文档名称	文档说明	文档标记
1	子系统预集成测试报告	子系统预集成测试结果说明	
2	软件联调测试报告	软件联调测试结果说明	
3	冒烟测试报告	冒烟测试结果说明	
4	子系统测试脚本和/或测试代码 子系统配置手册 子系统软件代码 单板配置手册		

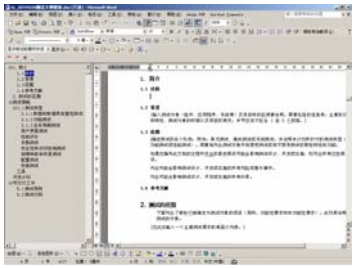
- 结束条件：

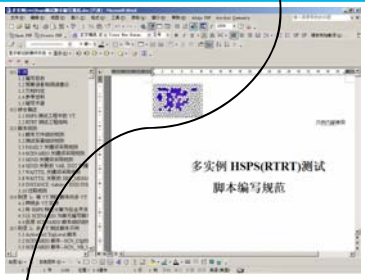
HSPS软件子系统在RTRT中的集成测试活动

测试经理

制定测试大纲

制定测试规范



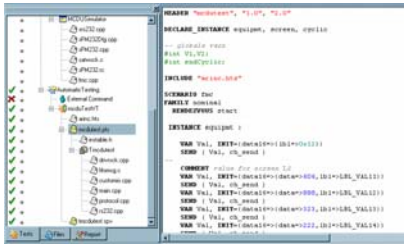


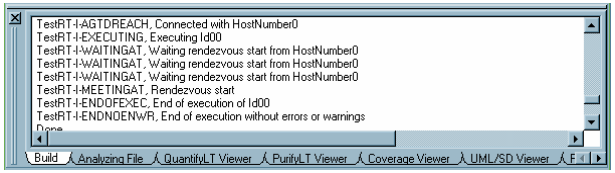
编码/测试人员

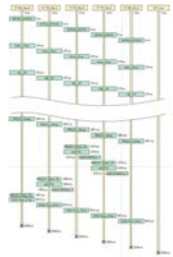
编码

制定测试脚本

分析测试结果







测试人员

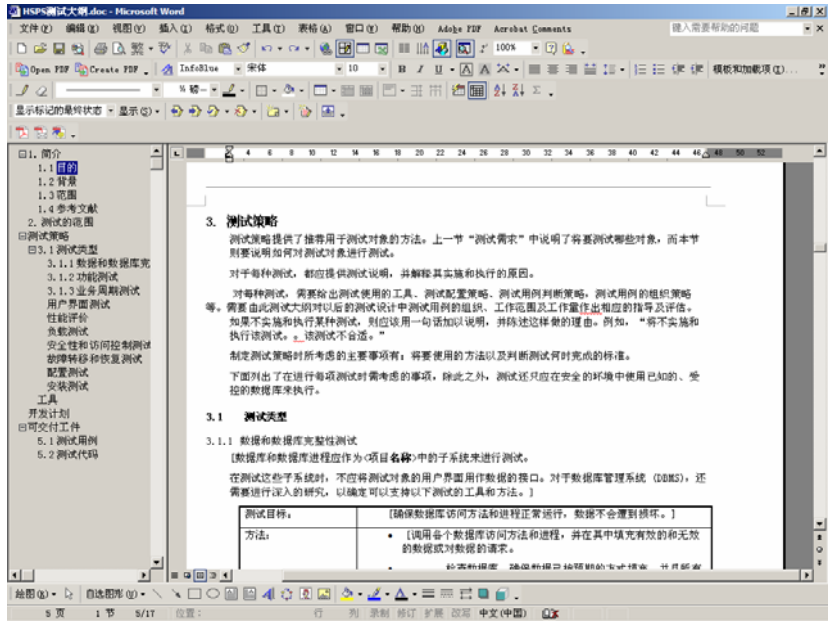
生成VT

执行VT

生成测试报告

HSPS软件子系统在RTRT中的集成测试策略

- HSPS软件子系统集成测试是针对HSPS软件子系统组织的一系列测试活动的重要组成部分。
- 基于Rational Test Realtime的HSPS集成测试脚本和工程的目的是在模拟的环境中验证HSPS子系统的功能（信令处理能力，可配置能力，故障检测能力），并借助Rational Test Realtime工具成熟、稳定的测试报告能力、执行路径跟踪能力、内部接口消息跟踪能力、测试环境配置能力等进一步提高缺陷定位和诊断的效率，为整个RNC系统的集成提供高质量的HSPS子系统代码。



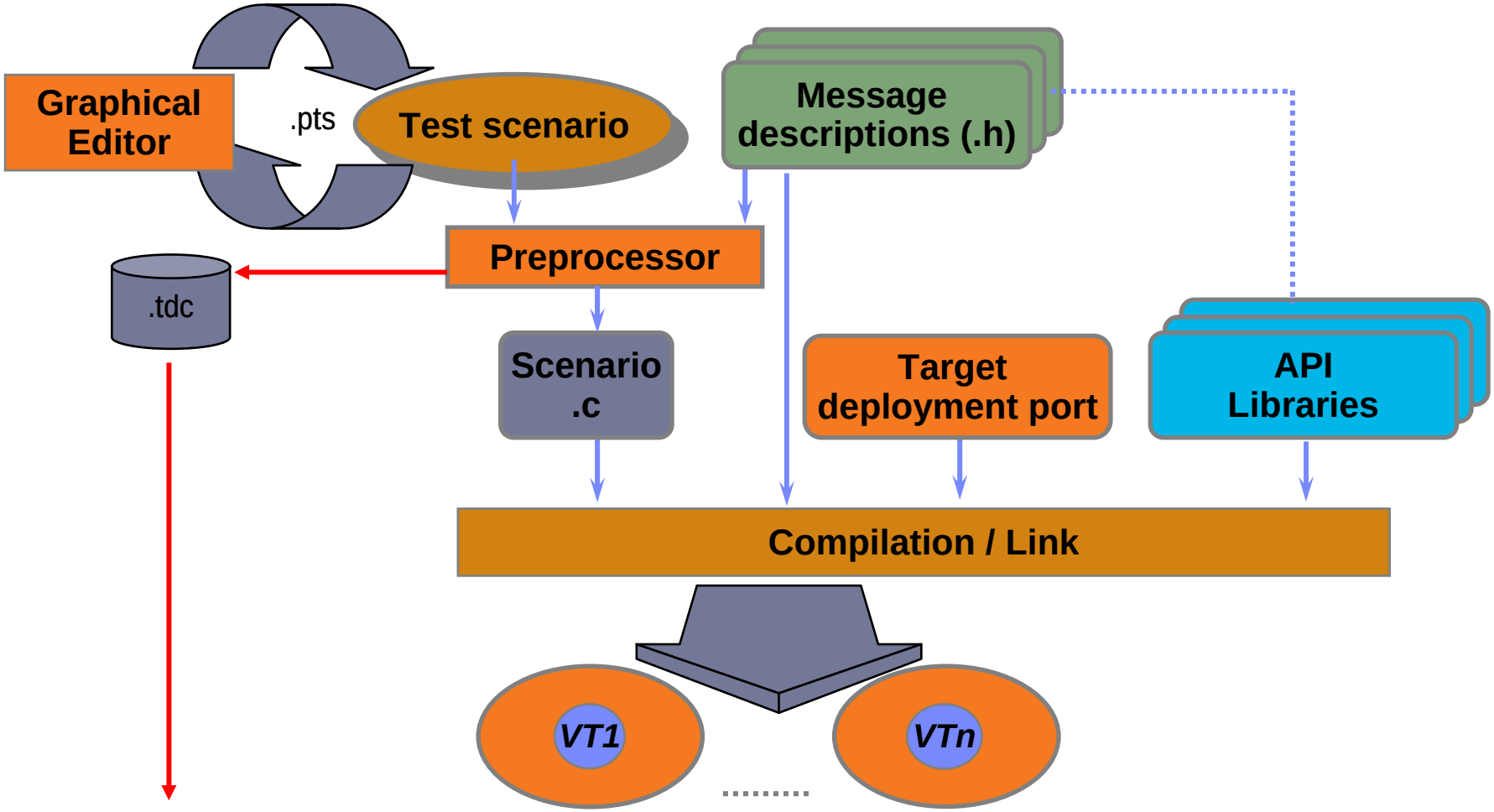
HSPS软件子系统在RTRT中的集成测试规范

- 规范定义HSPS测试中RTRT测试脚本需要遵照的约定，包括脚本结构、标识符命名等。
- 以期对顺利、灵活配置虚拟测试仪(VT)接口消息收发行为/虚拟测试仪的部署位置/虚拟测试仪(VT)数量提供必要的技术保证。

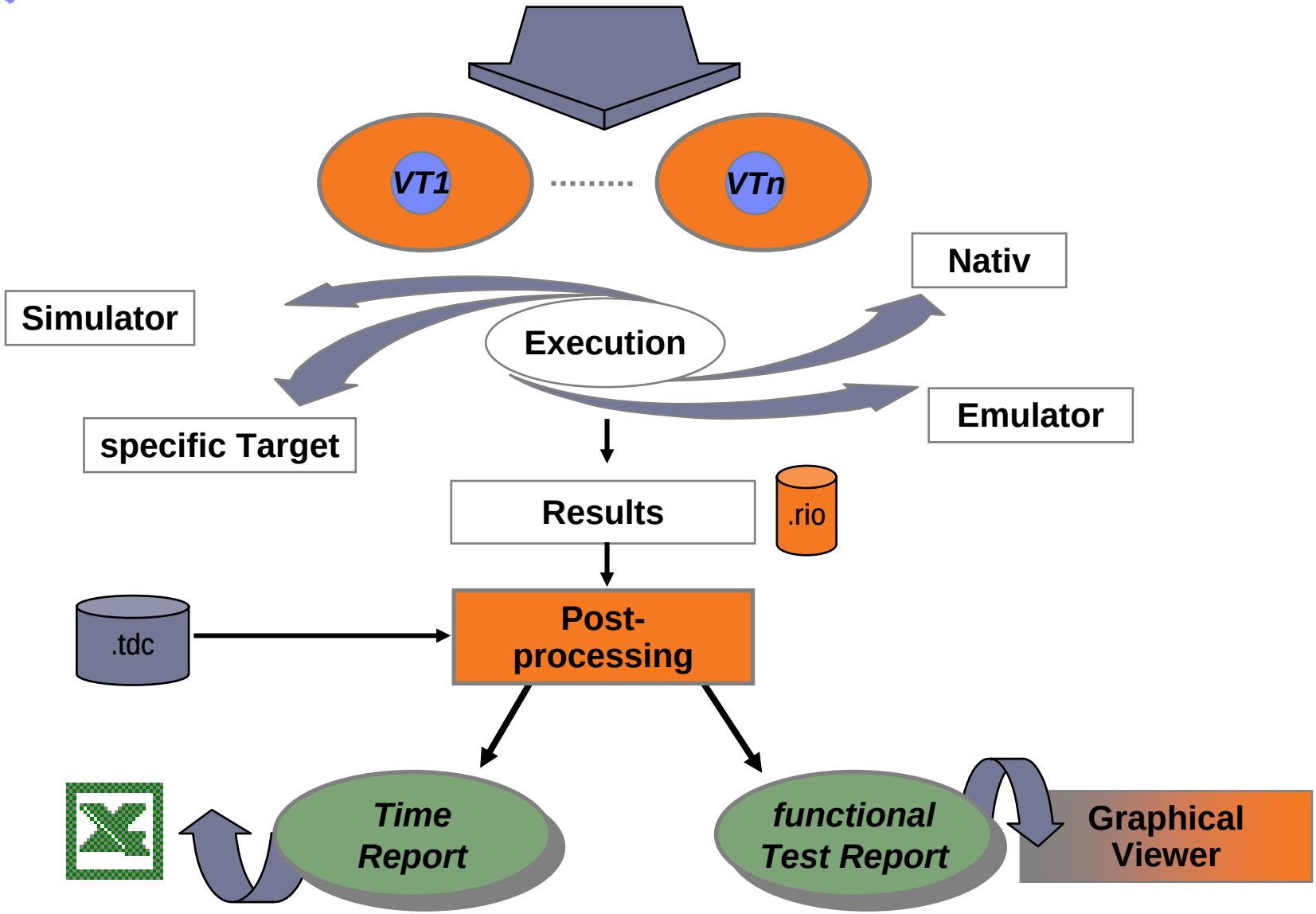
HSPS软件子系统在RTRT中的集成测试运行与支持环境

- RTRT集成测试脚本/工程的运行和支持环境为运行Windows 2000 Pro的PC/Workstation -LAN。测试支持环境的PC/Workstation数量可根据测试的阶段要求灵活配置（功能验证1~5；性能验证6~30）。
- PC/Workstation（用于运行测试工具或SUT）的最低配置要求为：
- CPU: P4 1.7 （推荐P4 2.8）；
- RAM: 256M （推荐512M）；
- HD: 20G;
- NIC: 10/100M with RJ45 port;
- OS: Windows 2000 pro;
- Utility App: Rational Test Realtime; Visual Studio 6; Office 2000

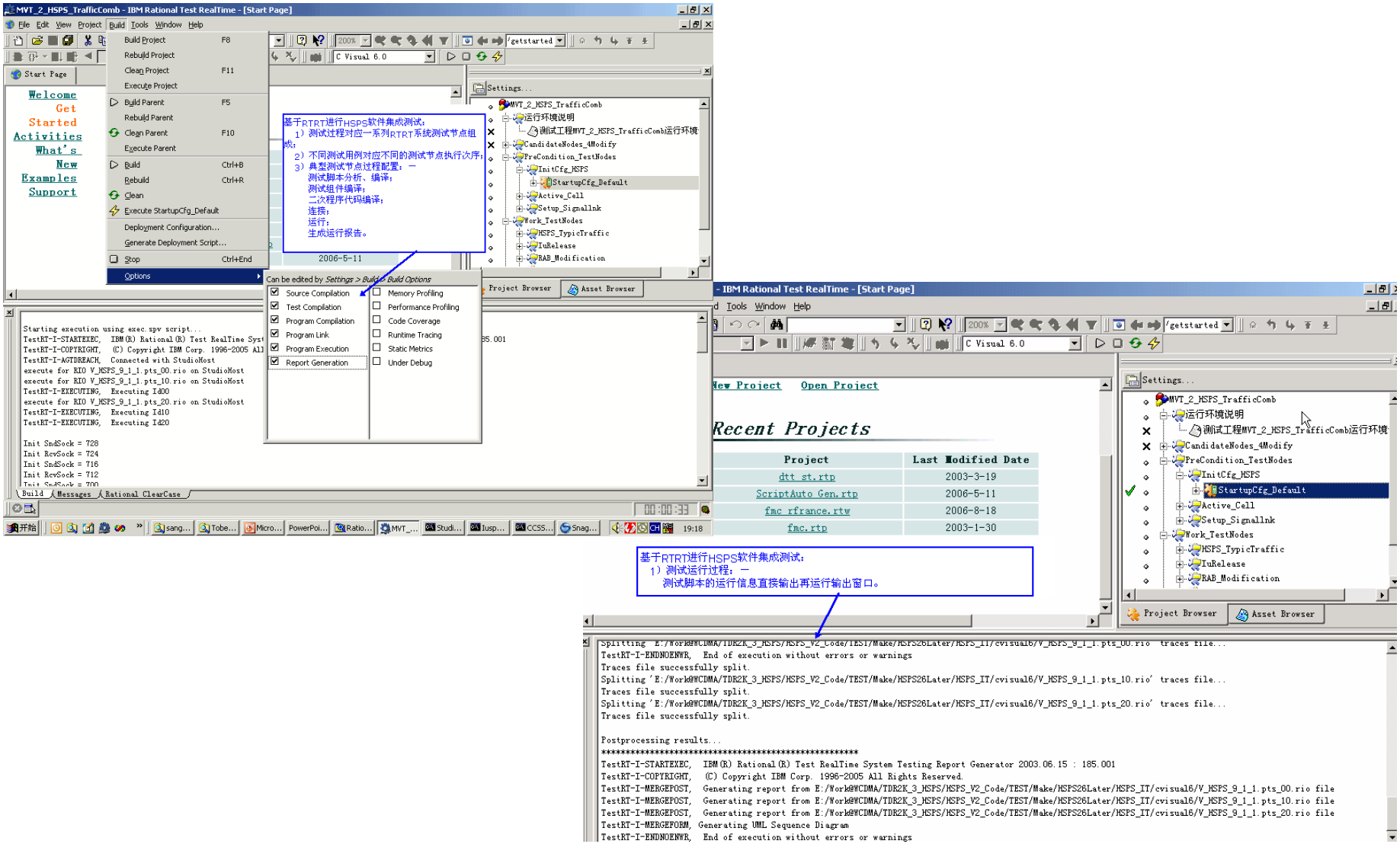
生成 Virtual Testers



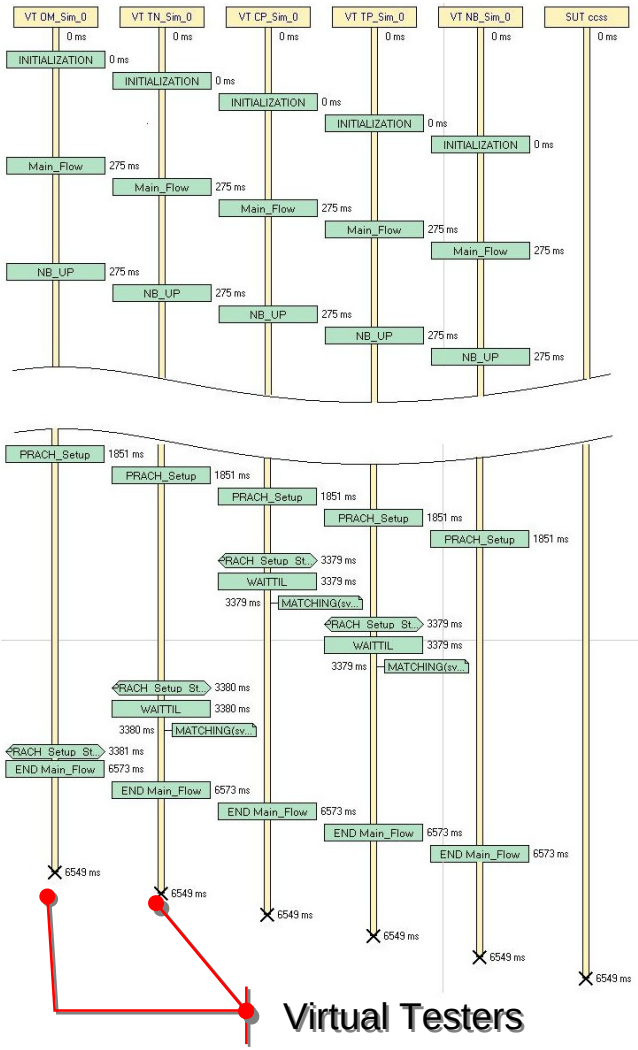
执行Virtual Testers

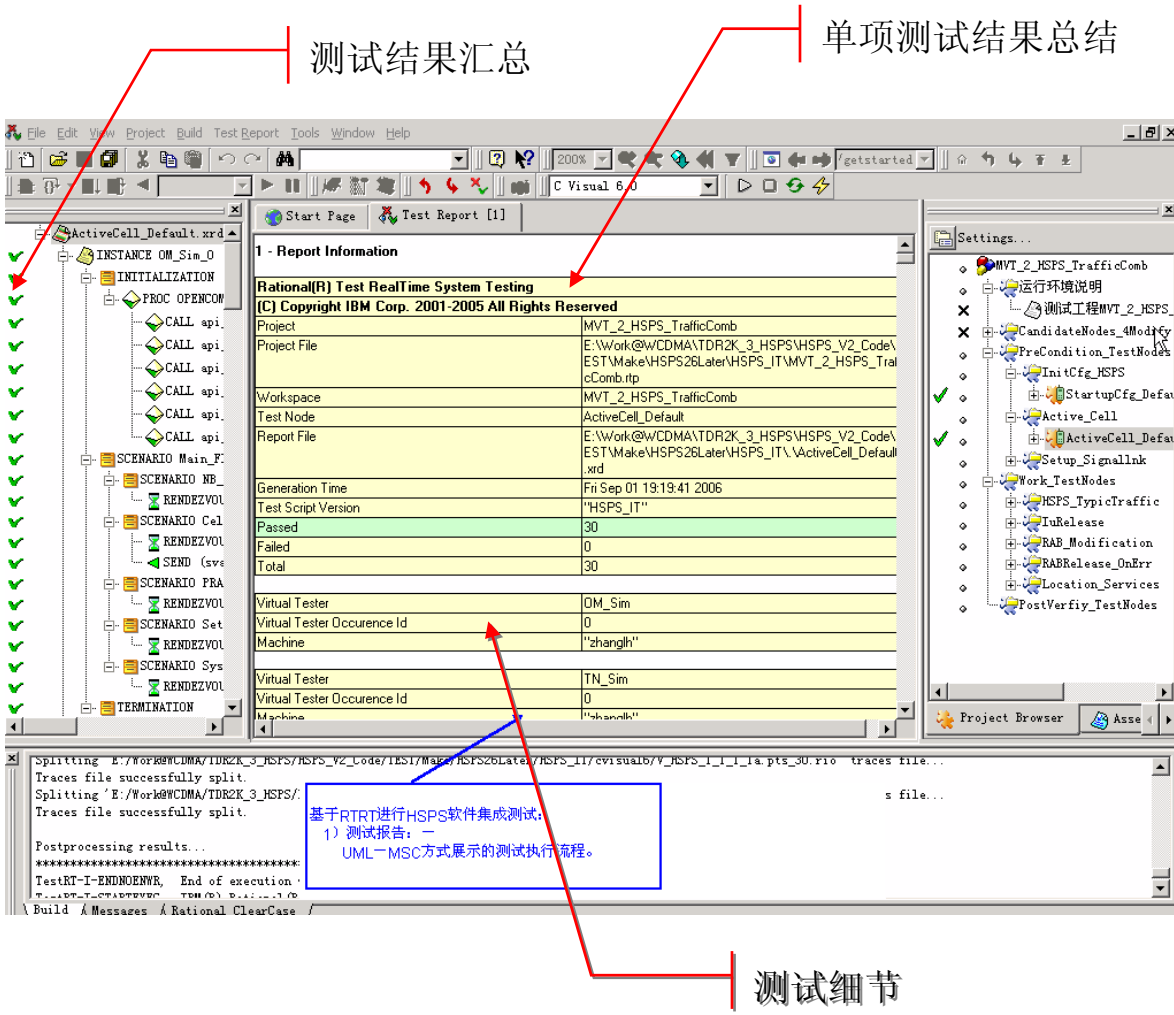


执行HSPS软件子系统集成测试

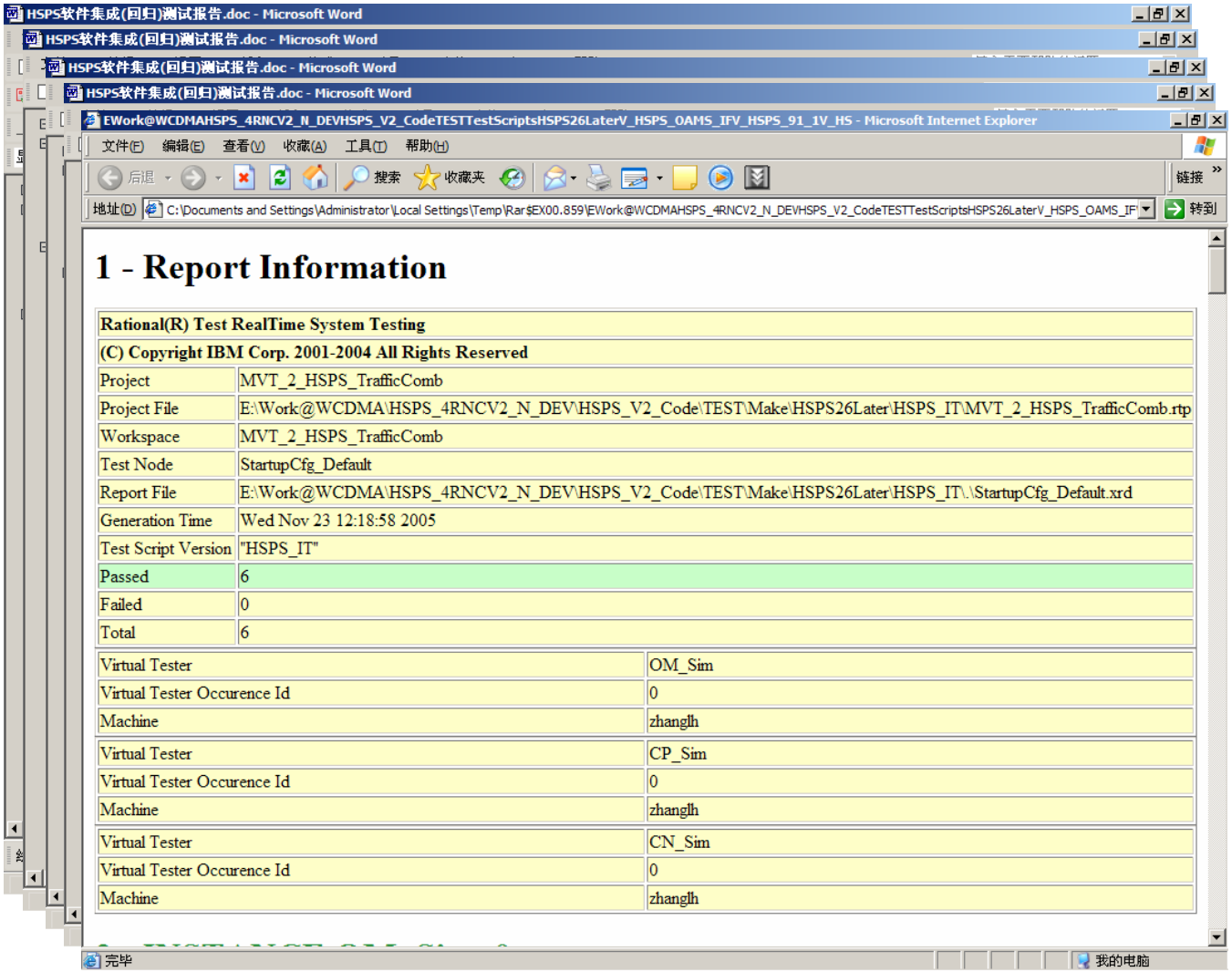


系统测试报告

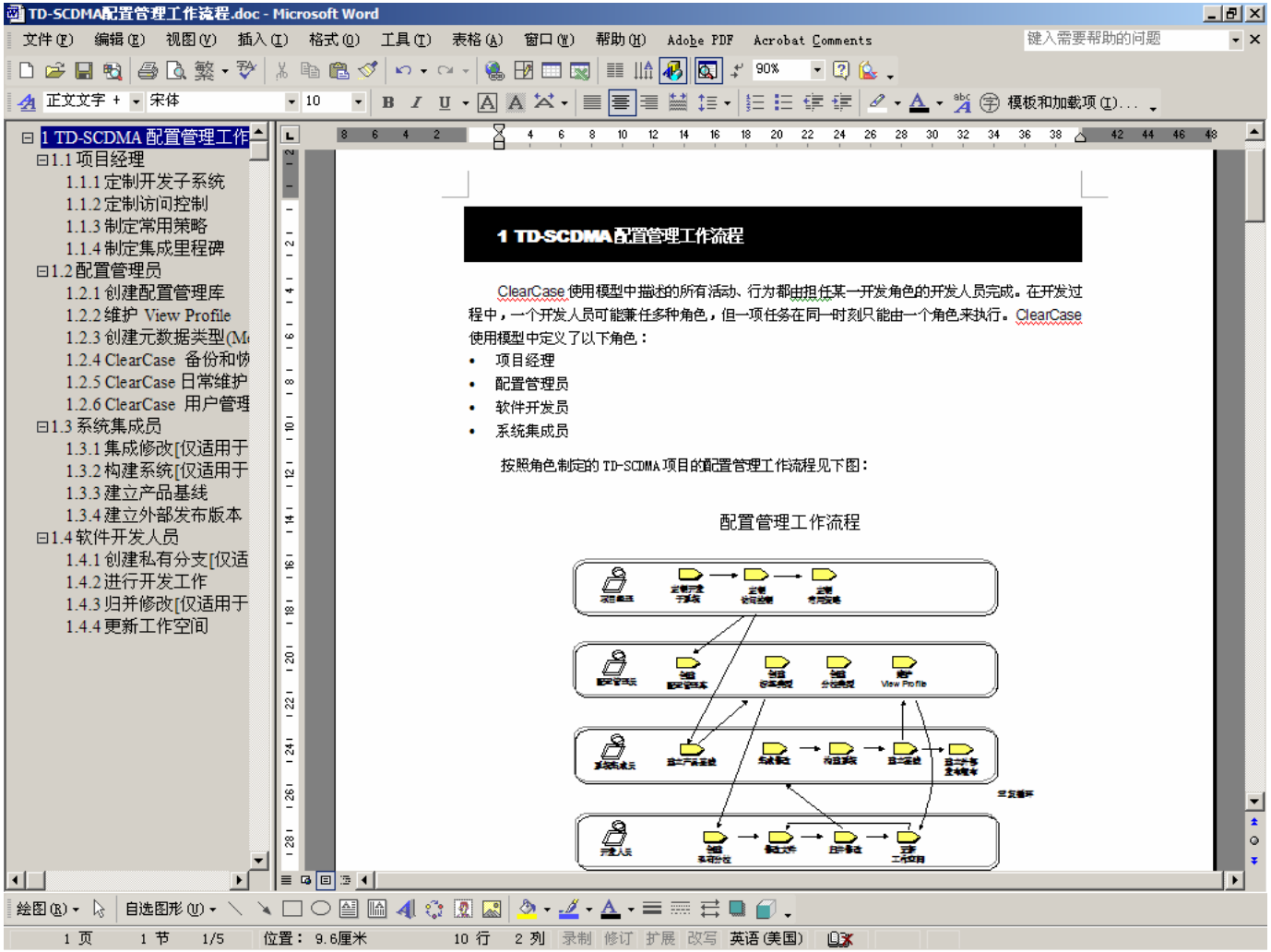

VT OM_Sim_0
VT TN_Sim_0
VT CP_Sim_0
VT TP_Sim_0
VT NB_Sim_0
SUT ccss
INITIALIZATION
Main_Flow
NB_UP
PRACH_Setup
RACH_Setup_St..
WAITTIL
MATCHING(sv...
END Main_Flow
6549 ms
Virtual Testers


测试结果汇总
单项测试结果总结
File Edit View Project Build Test Report Tools Window Help
Start Page
Test Report [1]
1 - Report Information
Rational(R) Test RealTime System Testing
(C) Copyright IBM Corp. 2001-2005 All Rights Reserved
Project
MVT_2_HSPS_TrafficComb
Project File
Workspace
Test Node
ActiveCell_Default
Report File
Generation Time
Fri Sep 01 19:19:41 2006
Test Script Version
"HSPS_IT"
Passed
30
Failed
0
Total
30
Virtual Tester
OM_Sim
Virtual Tester Occurence Id
0
Machine
"zhanglh"
TN_Sim
基于RTRT进行HSPS软件集成测试：
1）测试报告：一
UML一MSC方式展示的测试执行流程。
测试细节

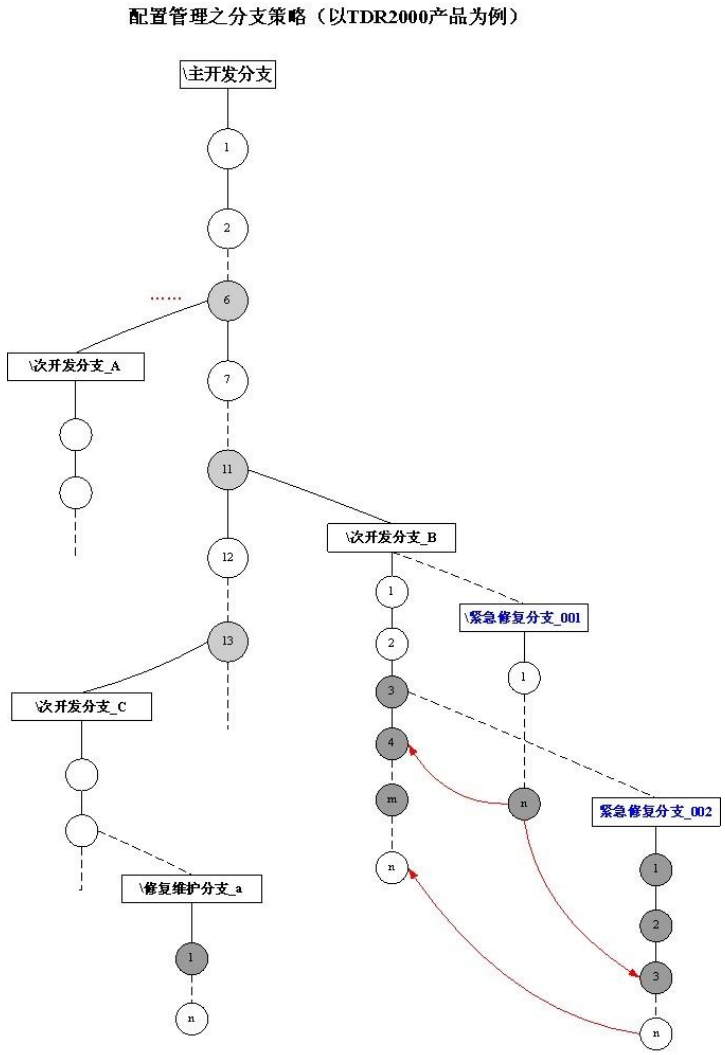
System Testing for C: *Reporting*



嵌入式软件开发的配置与变更管理流程

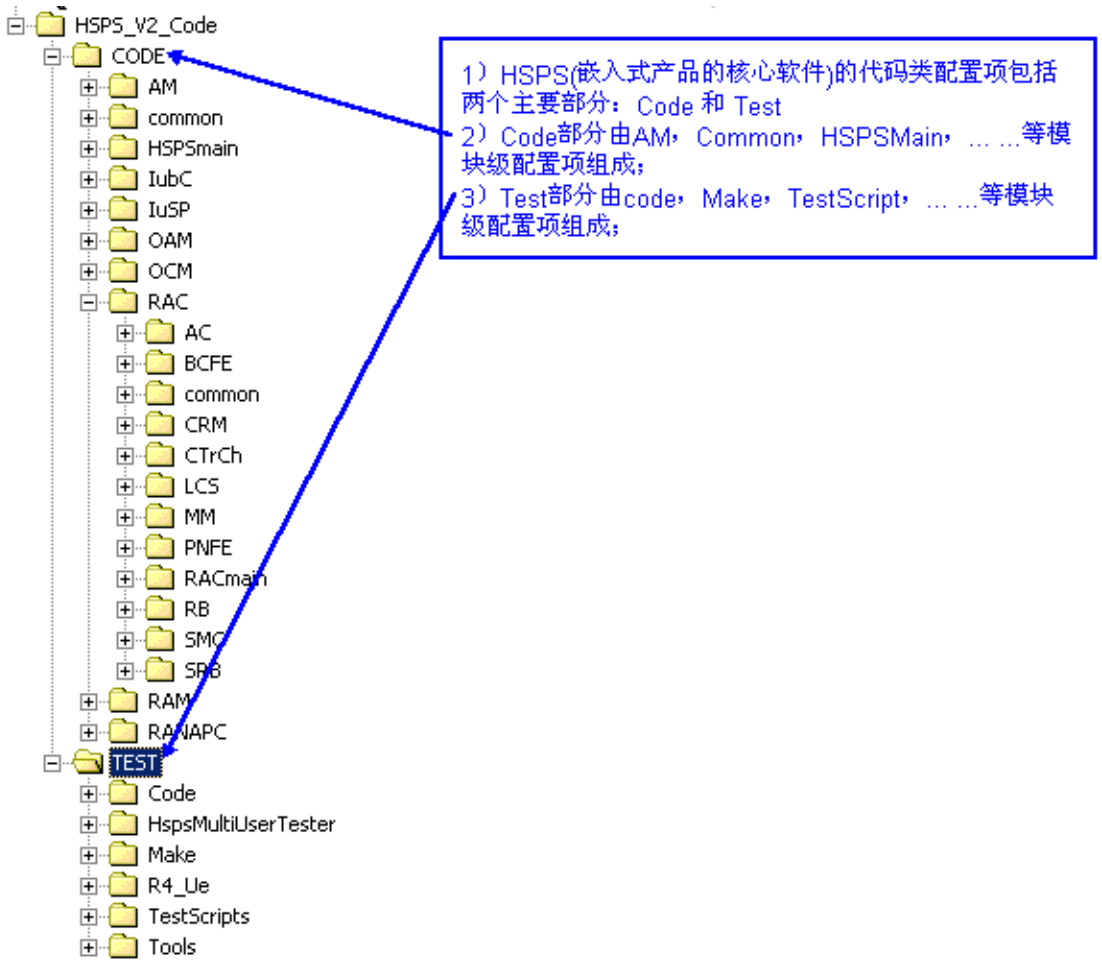


嵌入式软件开发的配置与变更管理分支策略



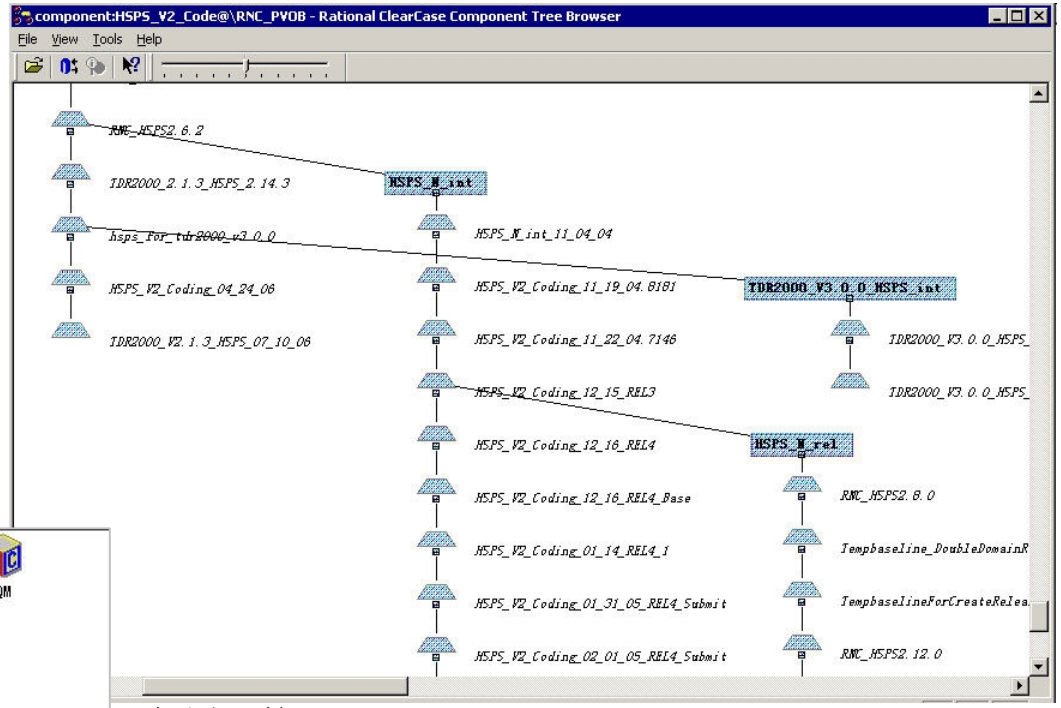
嵌入式软件开发的配置与变更管理库结构

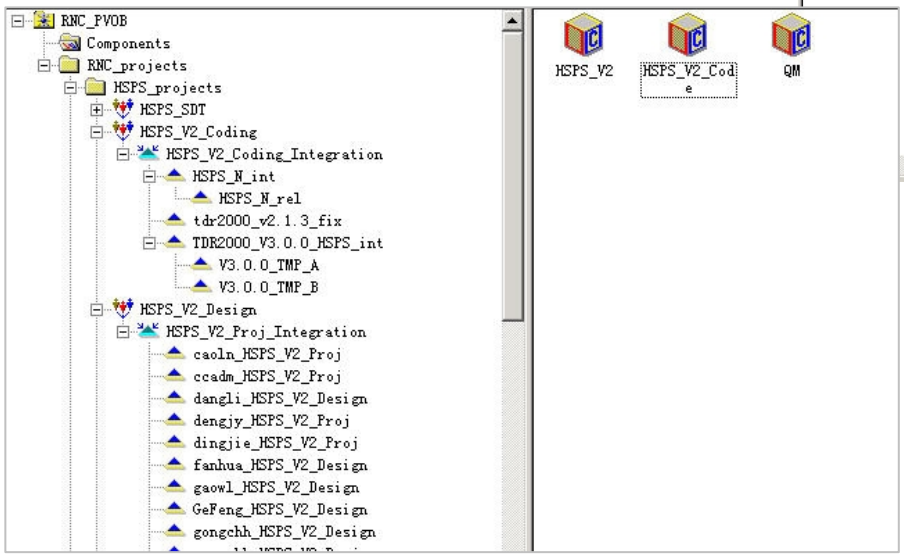
- HSPS(嵌入式产品的核心软件)的代码类配置项包括两个主要部分：Code 和 Test
  - Code部分由
    - AM
    - Common
    - HSPSMain
    - ...
    - 等模块级配置项组成；
  - Test部分由
    - Code
    - Make
    - TestScript
    - ...
    - 等模块级配置项组成；



嵌入式软件开发的组件和复用

- 组件名：HSPS_V2_Code\RNC_PVOB
- 复用组件的嵌入式软件开发项目
  - TDR2000_V2.1.3_int：V2.1.3版本
  - HSPS_N_int：N频点版本特性开发
  - HSPS_N_rel：N频点版本Bug修复及发布
  - TDR2000_V3.0.0_int：V3.0.0版本





- 复用目的

依开发计划，TDR2000 产品开发过程引入了不同的里程碑。在不同里程碑点到达时，发布具备一组既定特性的产品版本，满足多设备集成测试、网络试验的需要。

HSPS_N_int,HSPS_N_rel,TDR2000_V2.1.3_int,TDR2000_V3.0.0_int是分别对应四个开发历程碑的提交版本。开发任务管理时，必须及早对不同历程碑的开发任务及时规划、安排。后期提交的开发分支复用前期开发分支下的模块。

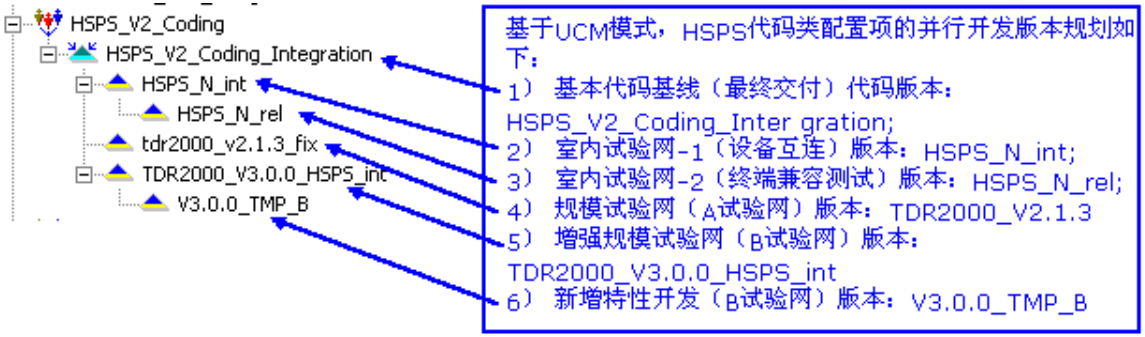
多嵌入式项目的并行开发

基于UCM模式，HSPS代码类配置项的并行开发版本规划如下：

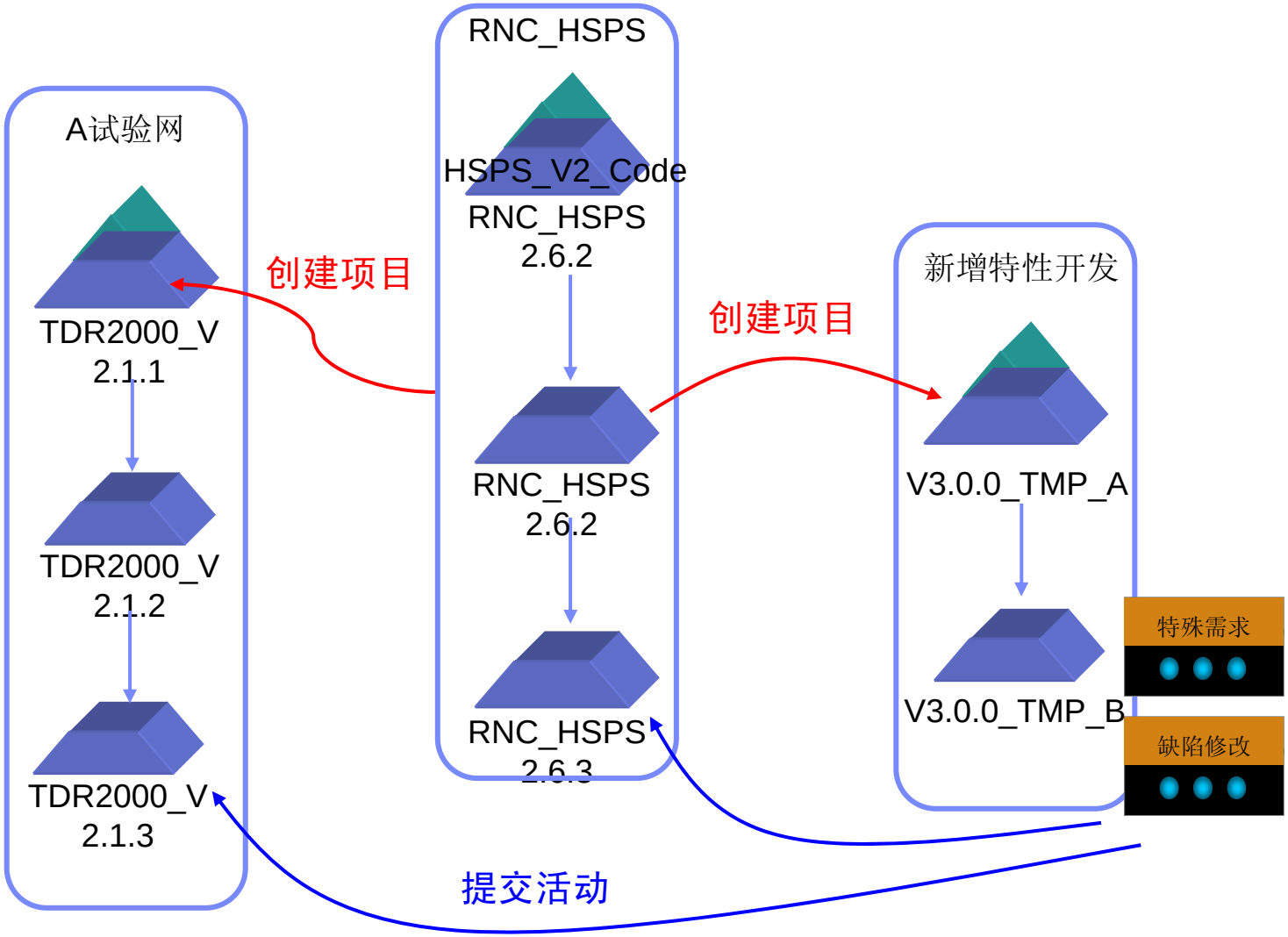
- 基本代码基线（最终交付）　　代码版本：HSPS_V2_Coding_Inter gration;
- 室内试验网-1（设备互连）　　版本：HSPS_N_int;
- 室内试验网-2（终端兼容测试）版本：HSPS_N_rel;
- 规模试验网（A试验网）　　版本：TDR2000_V2.1.3
- 增强规模试验网（B试验网）　　版本：TDR2000_V3.0.0_HSPS_int
- 新增特性开发（B试验网）　　版本：V3.0.0_TMP_B

多项目开发的目的

▪由于存在多个设备集成测试配置环境和网络试验环境，同时维护了HSPS_N_int（N频点特性开发版）和HSPS_N_rel（N频点特性Bug修复开发版），对应不同测试环境要求。

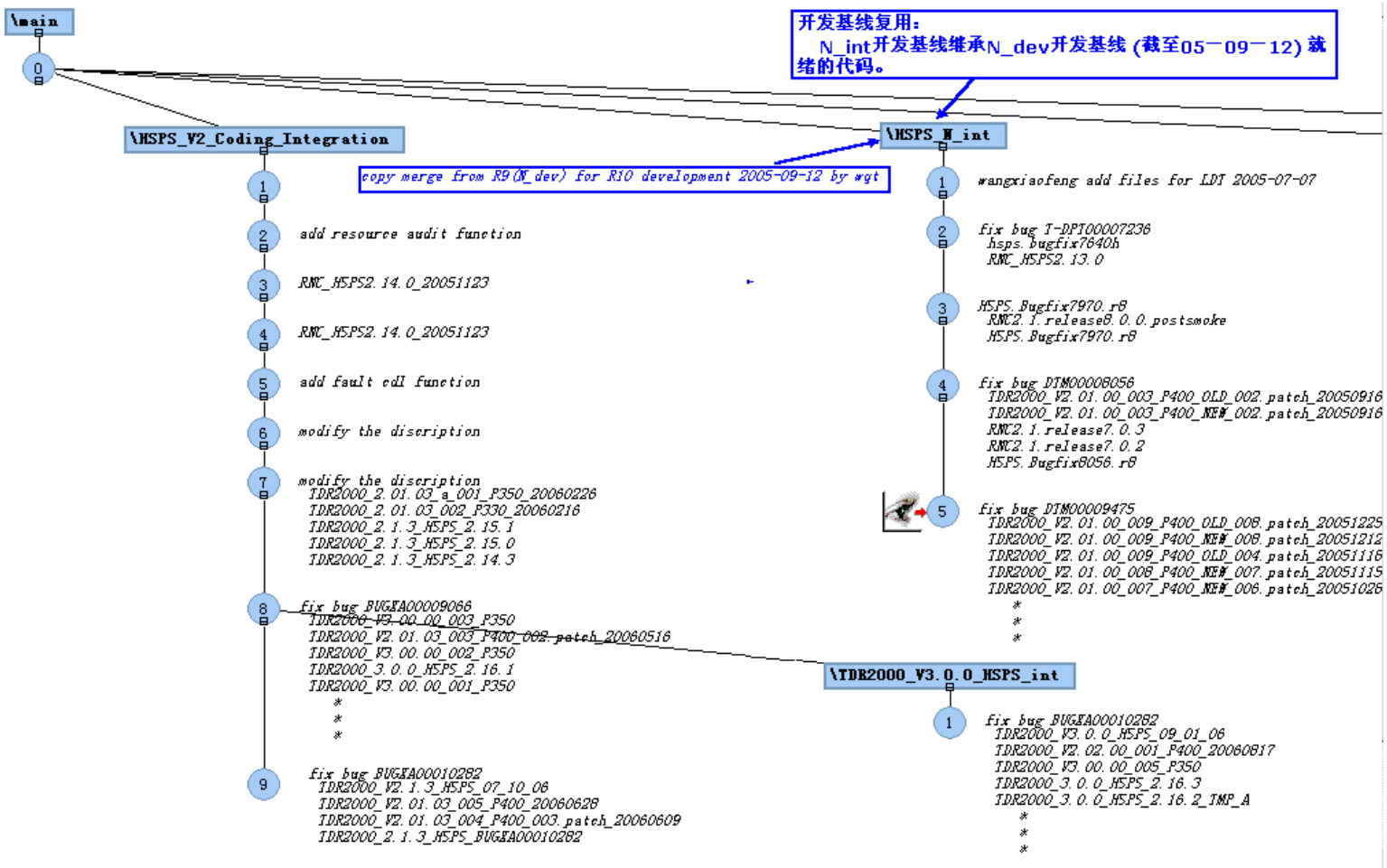


多嵌入式项目的并行开发



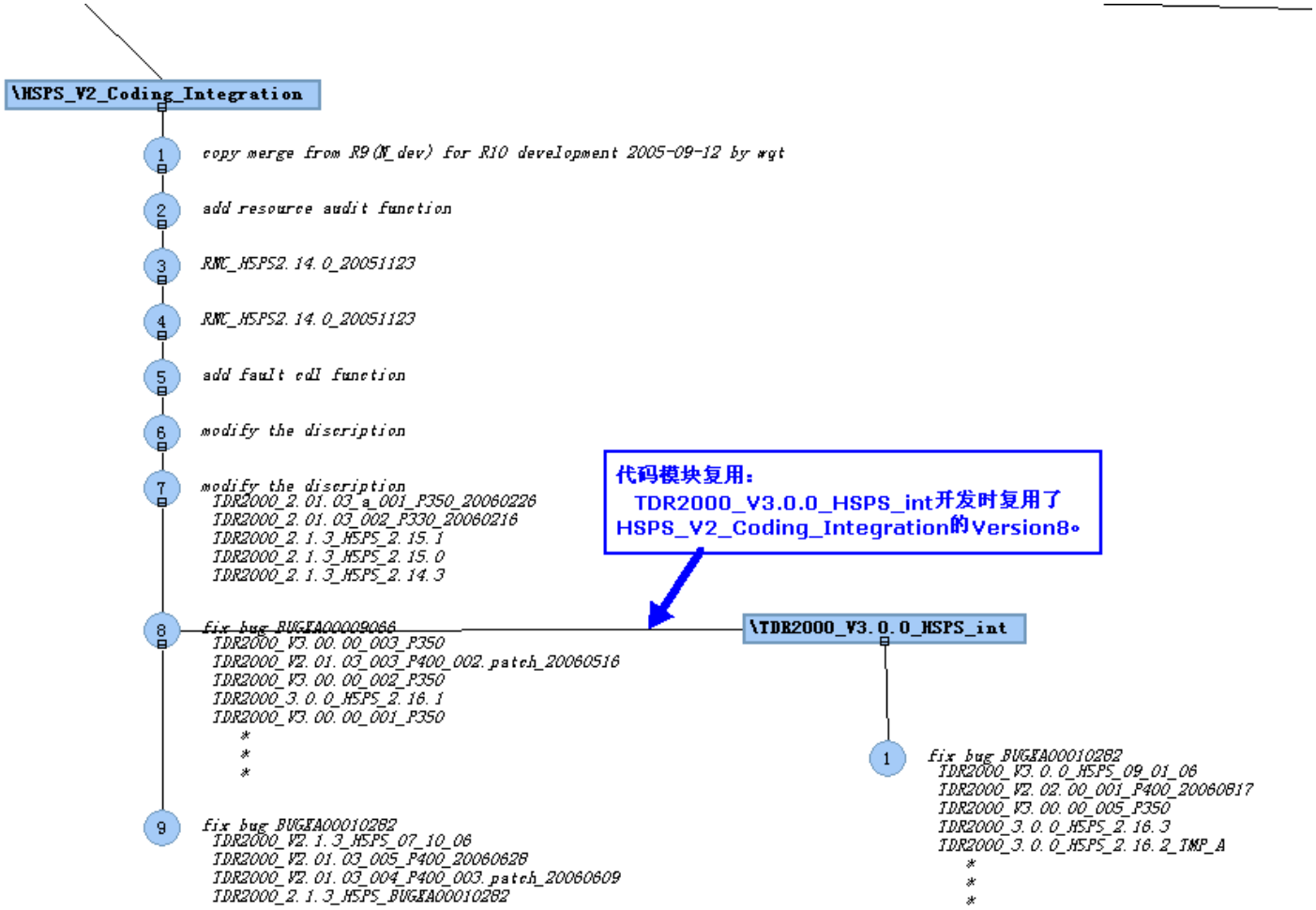
嵌入式软件开发的开发基线复用

N_int开发基线继承N_dev开发基线 就绪的代码。

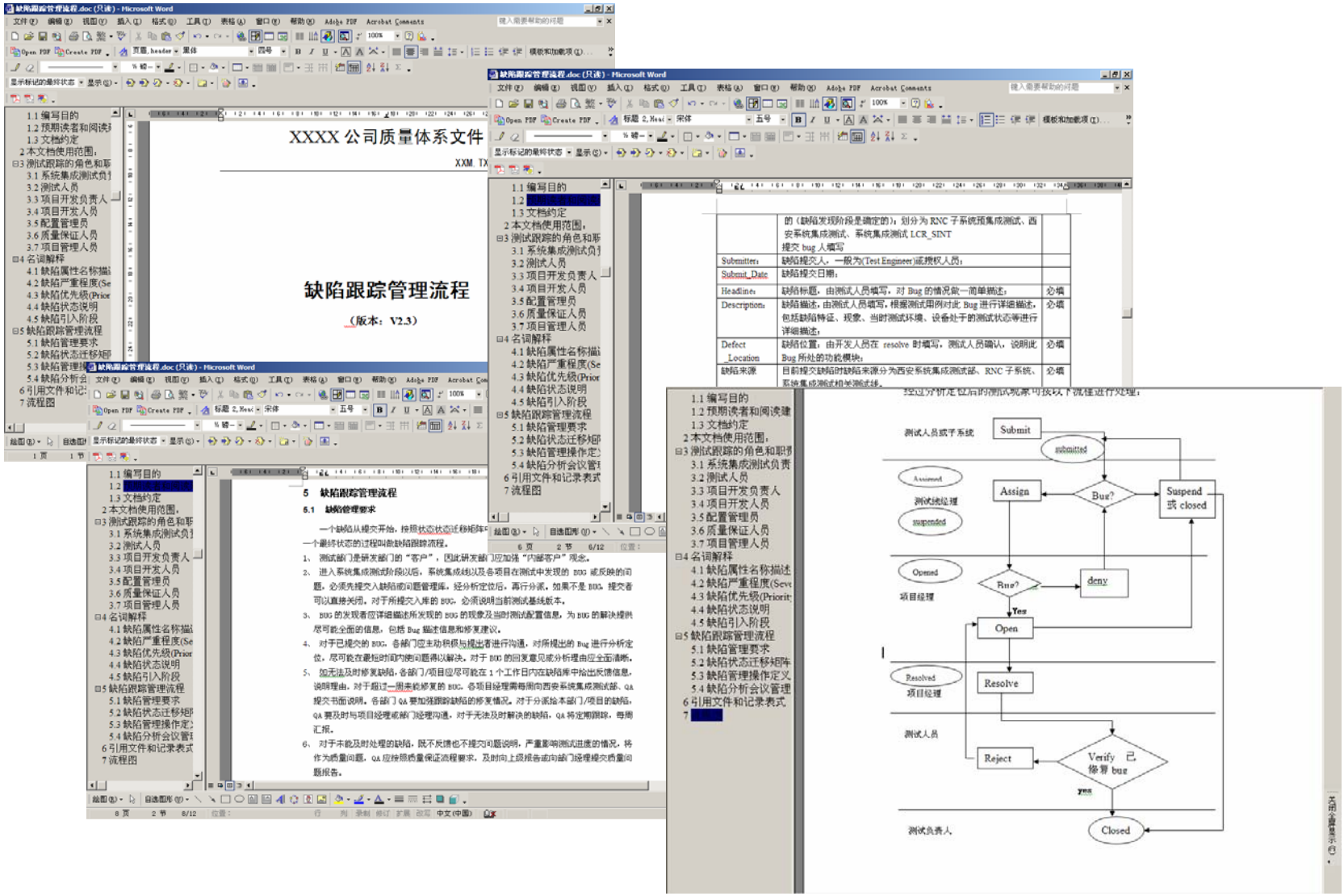


嵌入式软件开发的代码模块复用

TDR2000_V3.0.0_HSPS_int开发时复用了HSPS_V2_Coding_Integration的Version8



嵌入式软件开发的缺陷跟踪与管理



嵌入式软件开发的缺陷跟踪与管理：缺陷提交

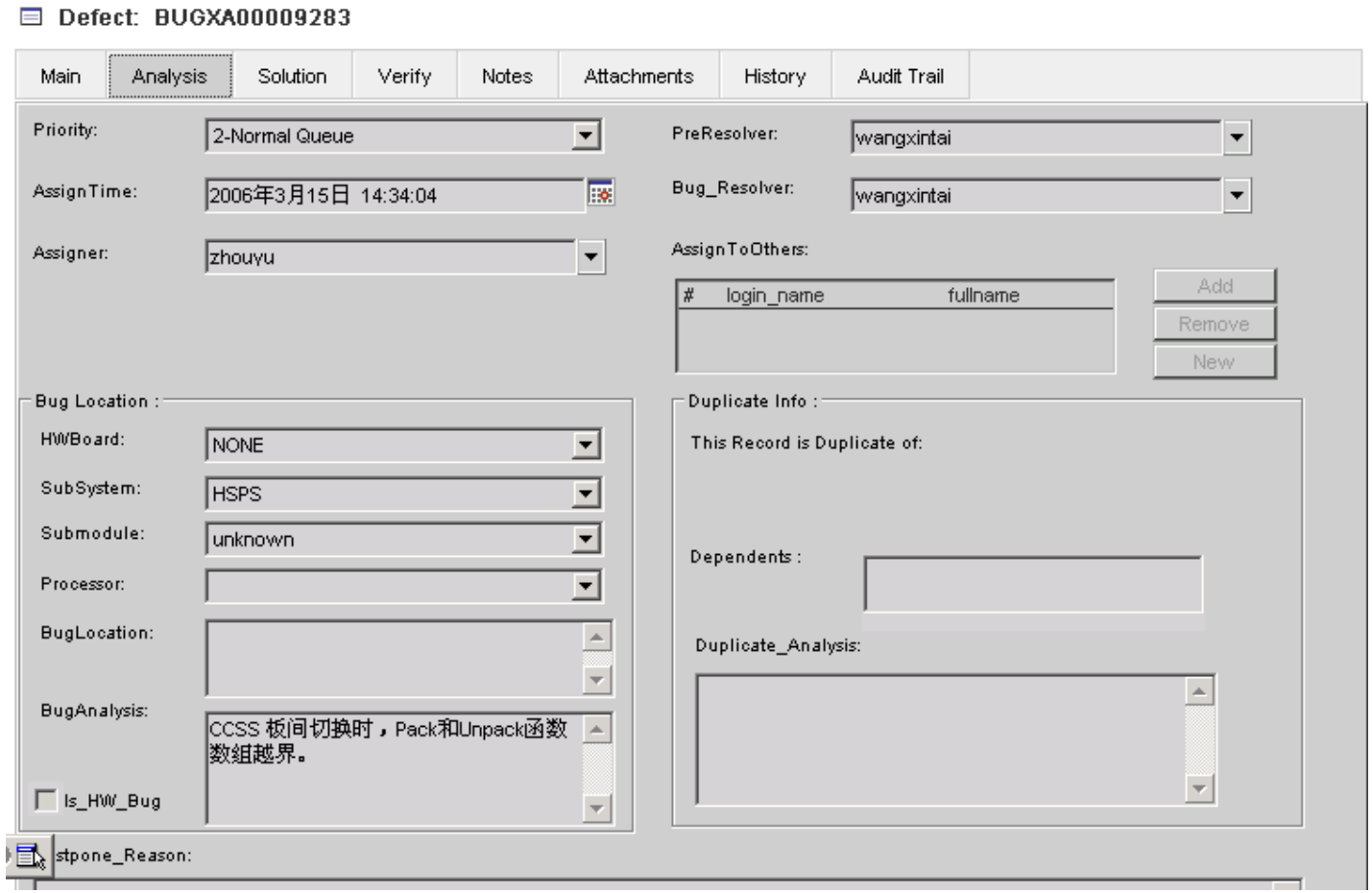
基于CQ跟踪管理缺陷——缺陷提交：

- 描述测试环境；
- 描述缺陷现象。

嵌入式软件开发的缺陷跟踪与管理：缺陷分析

基于CQ跟踪管理缺陷——缺陷分析：

- 记录缺陷分析



嵌入式软件开发的缺陷跟踪与管理：缺陷解决

基于CQ跟踪管理缺陷——缺陷解决：

- 缺陷解决方案和解决后的提交工件；

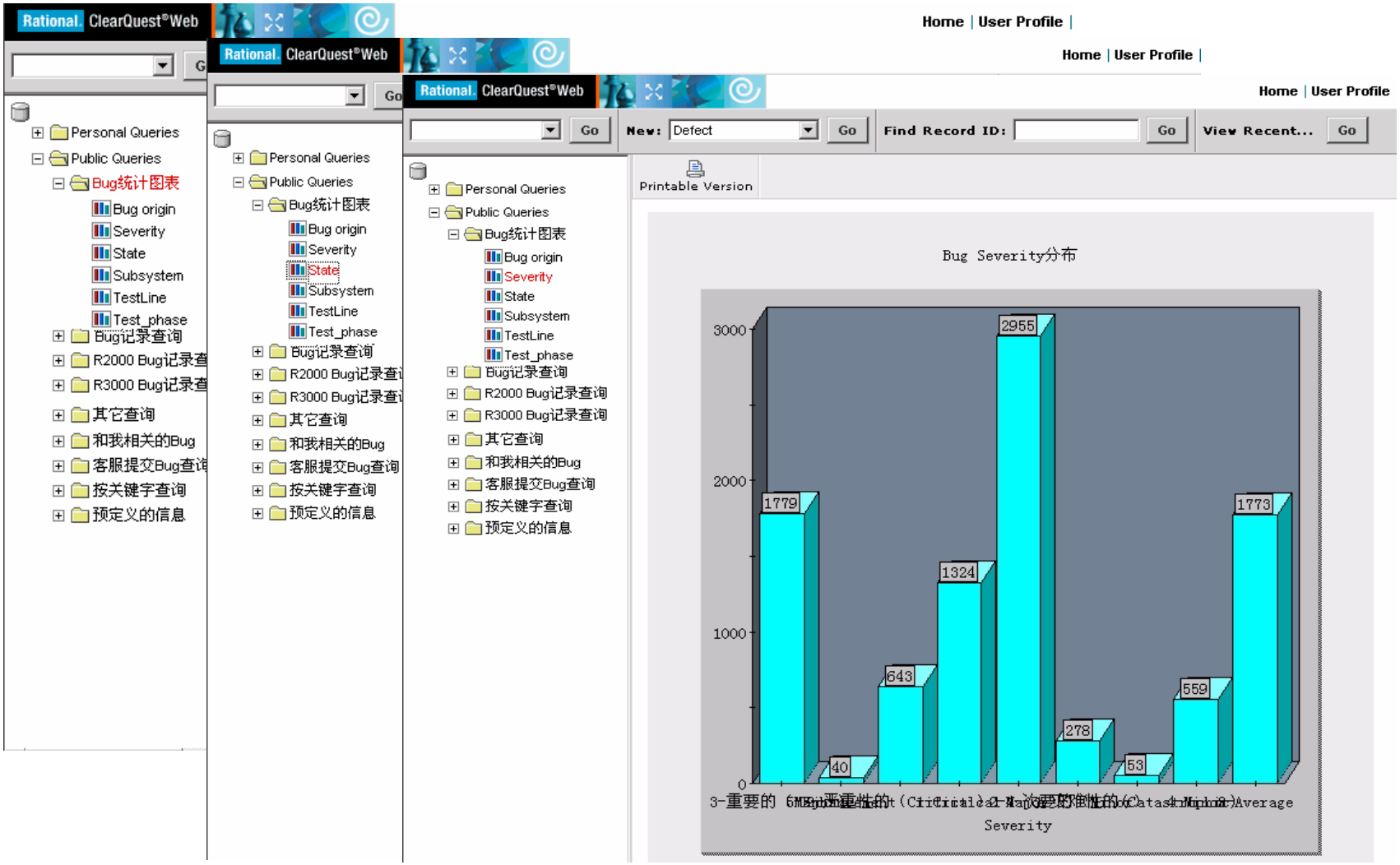
嵌入式软件开发的缺陷跟踪与管理：缺陷再验证和解决记录

基于CQ跟踪管理缺陷——缺陷再验证和解决记录：

- 缺陷定位、解决、处理记录。

嵌入式软件开发的缺陷跟踪与管理：缺陷度量和统计分析

- 缺陷的类型、分布、属性等的统计分析和度量。



嵌入式软件开发的同行评审

基于CQ管理测试工件评审：

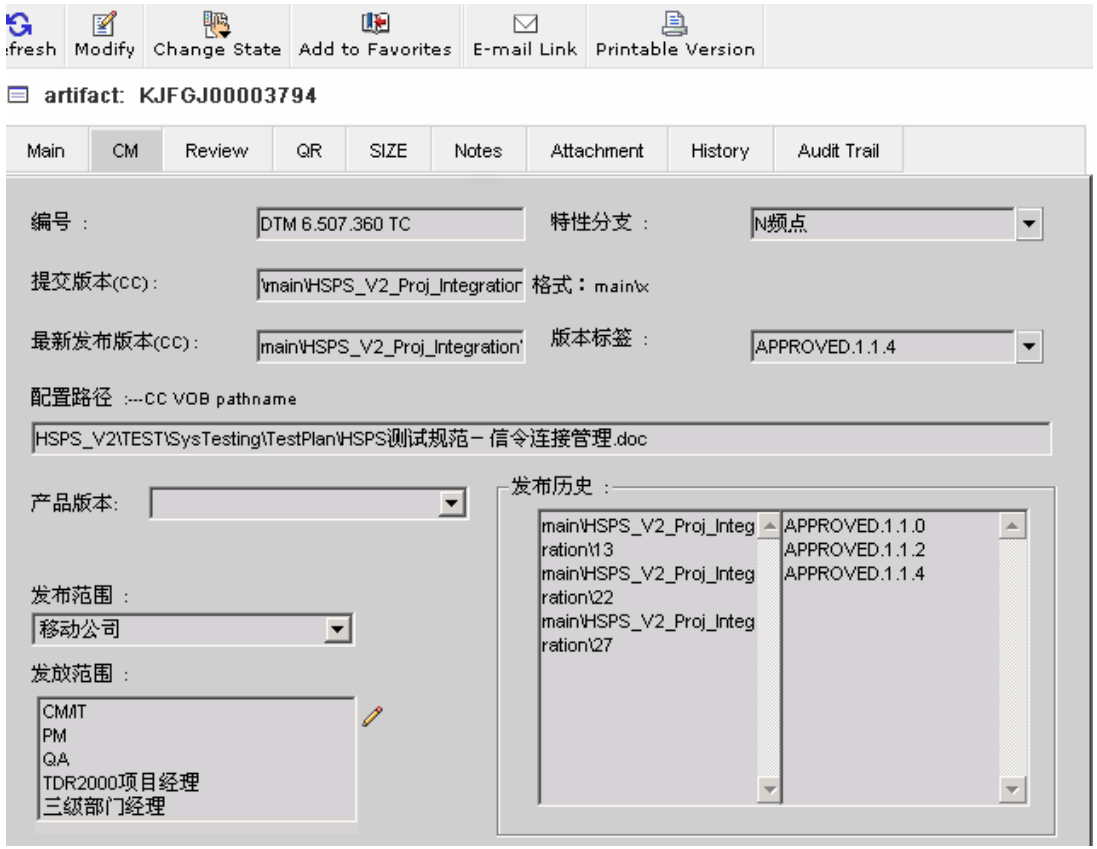
- 测试分析人员执行测试规范工件提交后，CQ通过邮件触发工件评审负责人(评审组长)启动评审活动；
- 评审员在阅读测试规范工件后提交意见，CQ记录并跟踪每条评审意见的处理结果；
- 测试开发人员可以通过CQ查阅评审工件的版本信息，发布范围等信息。
- CQ基于内建规则执行评审不同的活动。如：

工件责任人提交（工件）-->CQ通知评审组长；

评审组长启动评审-->CQ通知评审员评审安排及任务要求；

嵌入式软件开发的配置审计

- 为审计作好充分准备。
- 评估软件基线的完整性。
- 审查配置管理库系统的结构和设施。
- 证实软件基线库内容的完备性和正确性。
- 证实与适用的SCM标准和规程的符合性。
- 向项目软件经理报告审计结果。



标题

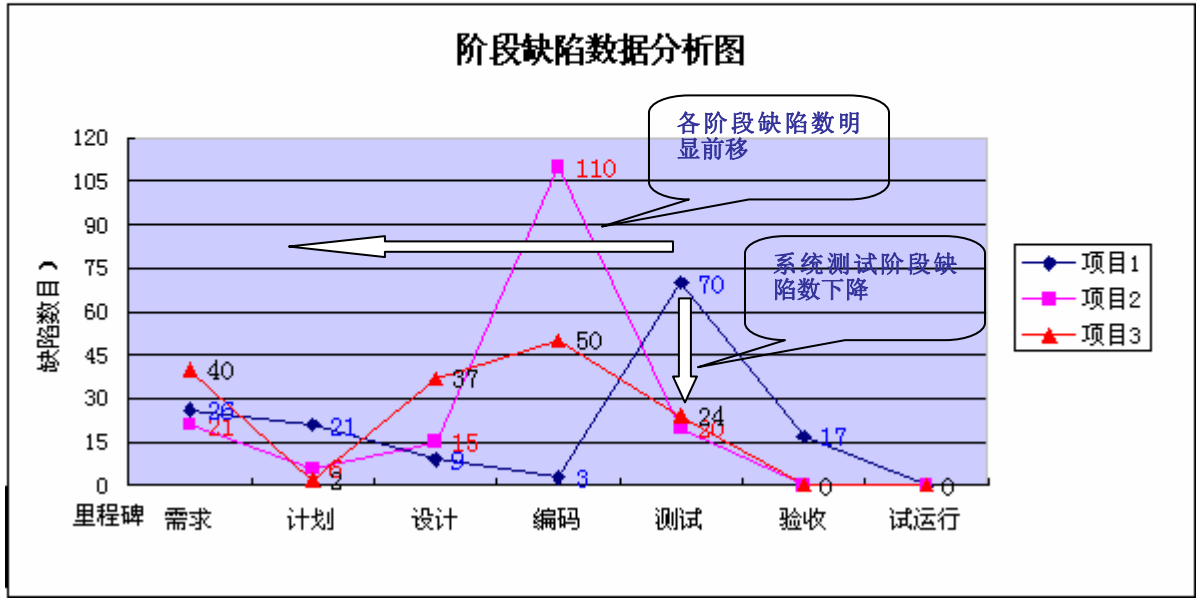
- 第一天上午
  - 嵌入式系统的开发面临的挑战
  - 如何应对这些挑战?
  - IBM Rational针对嵌入式软件开发开发的测试解决方案
  - 案例介绍
  - 嵌入式系统开发的测试解决方案ROI
  - 总结及Q/A
- 第一天下午
  - Test RealTime实现针对嵌入式软件开发的运行时刻分析
    - TDP
    - 运行时刻分析
    - DEMO
  - Q/A

Rational嵌入式系统开发解决方案ROI

案例分析中的原始数据均由客户提供。IBM仅是在此基础将其数据统计分析后加以展示，以说明Rational嵌入式系统开发解决方案给客户带来的益处。

- 项目1没有使用RTRT；
- 项目2和3使用RTRT进行单元测试和集成测试。

	需求	计划	设计	编码	测试	验收	试运行
项目1	26	21	9	3	70	17	0
项目2	21	6	15	110	20	0	0
项目3	40	2	37	50	24	0	0



Rational嵌入式系统开发解决方案成功案例

引进嵌入式软件开发解决方案后,由于的项目缺陷大部分在测试前就已识别出来，故在测试过程发现的缺陷数有了明显的下降。同时，提交给客户使用后的缺陷数也随之降低。由此，也为项目减少维护的成本及工作量。

引进IBM Rational嵌入式实时软件开发解决方案前的产品质量：

测试前缺陷数	测试缺陷数	提交后缺陷数	合计
40	34	5	79

引进IBM Rational嵌入式实时软件开发解决方案后的产品质量

测试前缺陷数	测试缺陷数	提交后缺陷数	合计
85	7	0	92

标题

- 第一天上午
  - 嵌入式系统的开发面临的挑战
  - 如何应对这些挑战?
  - IBM Rational针对嵌入式软件开发开发的测试解决方案
  - 案例介绍
  - 嵌入式系统开发的测试解决方案ROI
  - 总结及Q/A
- 第一天下午
  - Test RealTime实现针对嵌入式软件开发的运行时刻分析
    - TDP
    - 运行时刻分析
    - DEMO
  - Q/A

总结：Rational嵌入式系统开发的测试解决方案

- 可以使用统一的工具，完成在不同语言、平台上开发出来的应用系统的各种等级的白盒测试；
- RTRT自动生成驱动函数、桩函数和测试脚本，提高开发效率；
- 测试脚本无需随着目标平台的改变而变，降低开发成本；
- 支持从单元级到系统级所有级别的白盒测试，能够支持各种开发平台，将不同开发项目的白盒测试集中到统一的工具中；
- 丰富的测试结果报告：作单元测试、系统测试时能同时获得运行时分析数据；
- 应用范围广：在语言和平台的丰富性上，RTRT目前没有竞争对手；


Questions


Thank
You

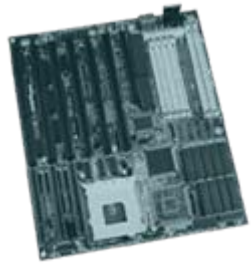
标题

- 第一天上午
  - 嵌入式系统的开发面临的挑战
  - 如何应对这些挑战?
  - IBM Rational针对嵌入式软件开发开发的测试解决方案
  - 案例介绍
  - 嵌入式系统开发的测试解决方案ROI
  - 总结及Q/A
- 第一天下午
  - Test RealTime实现针对嵌入式软件开发的运行时刻分析
    - TDP
    - 运行时刻分析
    - DEMO
  - Q/A

在主机和目标平台测试和排错
Target Deployment Port 技术使之成为可能

- 完全自动化的通用的特性
  - 测试套件完成构建并下载到目标平台上
  - 测试的执行和监控
  - 测试结果从目标平台上载
- 这个特性利用下列基础作为支持：
  - 你所选择使用的编译器/连接器/排错器
  - 你所采用的主机/目标机连接 (串行连接, 以太网连接, JTAG 探针…)

主机构建过程

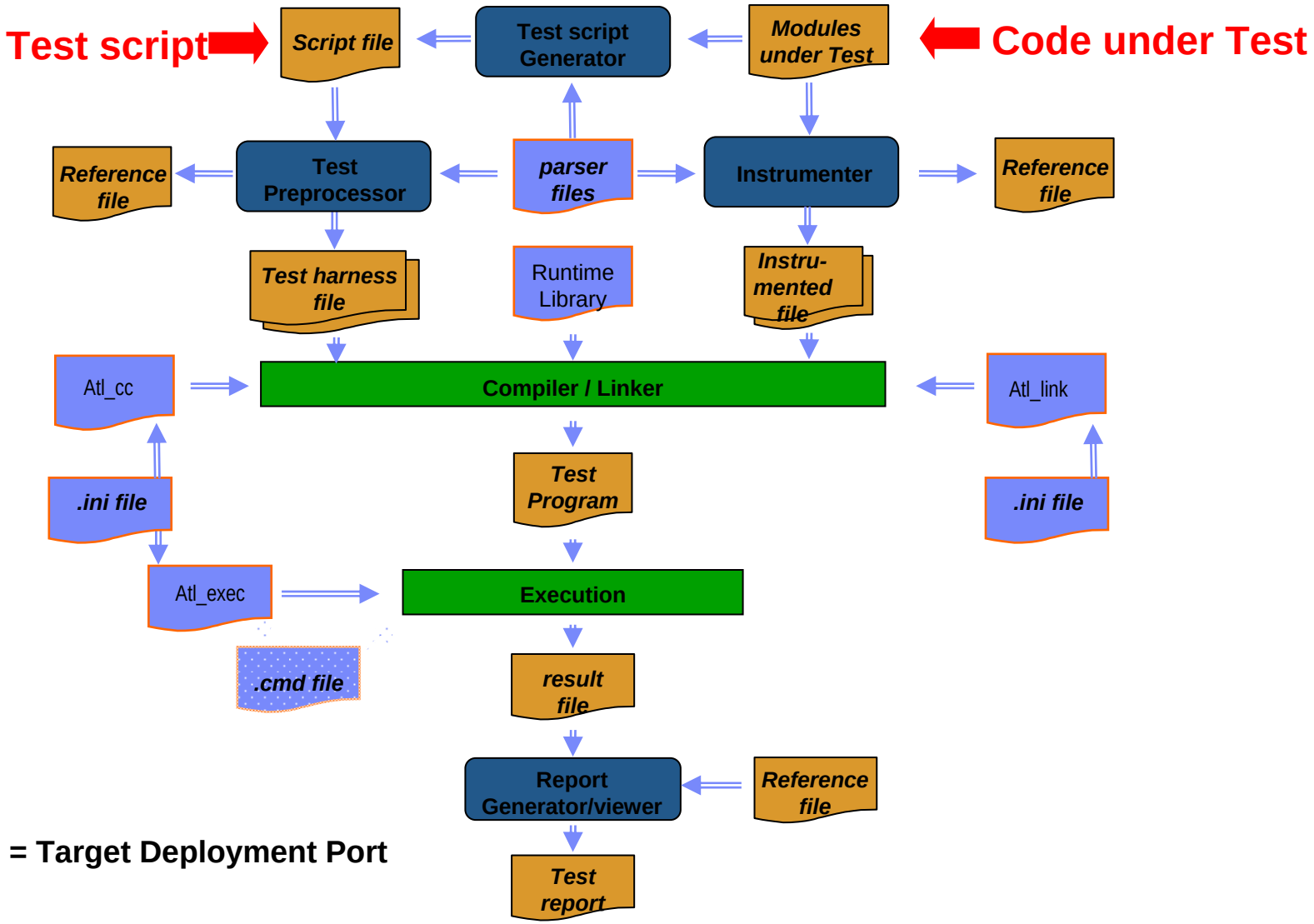


目标平台上的执行和监控

Target Deployment Port Definition

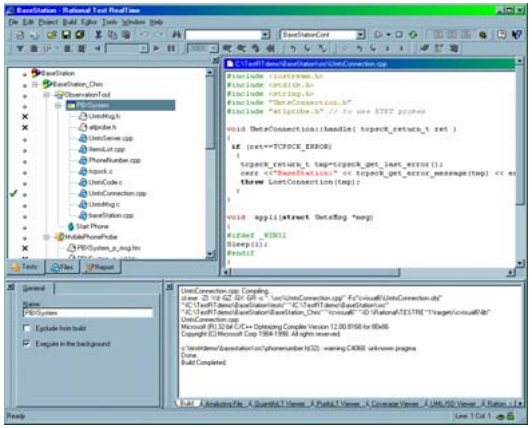
- Setting up a Target Deployment Port (TDP):
  - Set of files and procedures that enable the execution of generated test programs or instrumented applications directly on your target or host
  - Data retrieval of test results the target or host
- Optimized:
  - Early-design of TestRT with the embedded development constraints
  - Target specificities concentrated into the TDP settings
  - Scalable for the market: c166, TMS, 68k, PowerPc, 8051, ColdFire and many others (32, 16, 8 bits architectures) target or host
- Deeply linked with a specific development environment:
  - Host Machine/Target Environment
  - Compiler/Linker

Target Deployment Port: *Process Overview*



Target Deployment Port as Target Interface

Studio call scripts which uses the TDP to access targets



Generic Scripts

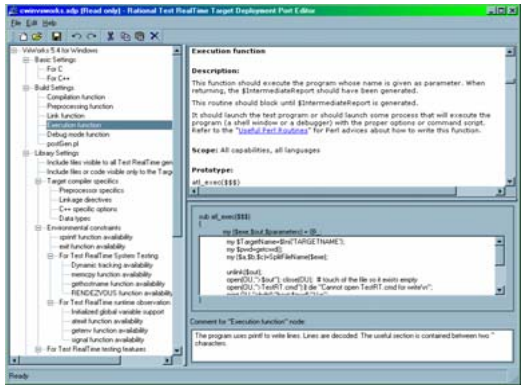
TP lib
atl_asm
atl_cc
atl_lib
atl_link
atl_exe
.def & .ini

Cross compiler

Lib
CC
Link
Simul_exe
Emul_exe
……_exe

1.Download
2.Run
3.upload
4.quit

.xdp



TDP editor extract TDP files from .xdp

Target Deployment Port Configuration

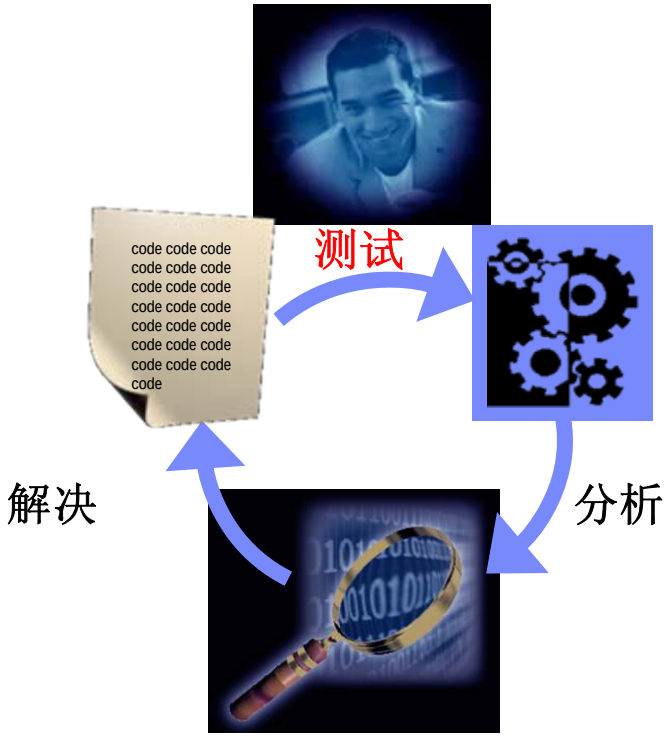
- TDP Configuration is composed of 4 components:
  - **Basic Settings:** Used to specify default file extensions, default flags, environment variables and custom variables required for your target architecture. The resulting files are stored in the top level TDP directory.
  - **Build Settings:** Used to configure the functions required for the integrated build process. It defines assembly, compilation, libraries build, link and execution (Perl) scripts, plus any user-defined scripts. These files are stored in the TDP **/cmd** subdirectory.
  - **Library Settings:** A set of source code files, and a dedicated customization file (custom.h) used to adapt the code library to the target requirements. These files are generated in the TDP **/lib** subdirectory.
  - **Parser Settings:** Used to modify the default behavior of the parser in order to address, for example, non-ANSI extensions. The resulting files are stored in the TDP **/ana** subdirectory.

标题

- 第一天上午
  - 嵌入式系统的开发面临的挑战
  - 如何应对这些挑战?
  - IBM Rational针对嵌入式软件开发开发的测试解决方案
  - 案例介绍
  - 嵌入式系统开发的测试解决方案ROI
  - 总结及Q/A
- 第一天下午
  - Test RealTime实现针对嵌入式软件开发的运行时刻分析
    - TDP
    - 运行时刻分析
    - DEMO
  - Q/A

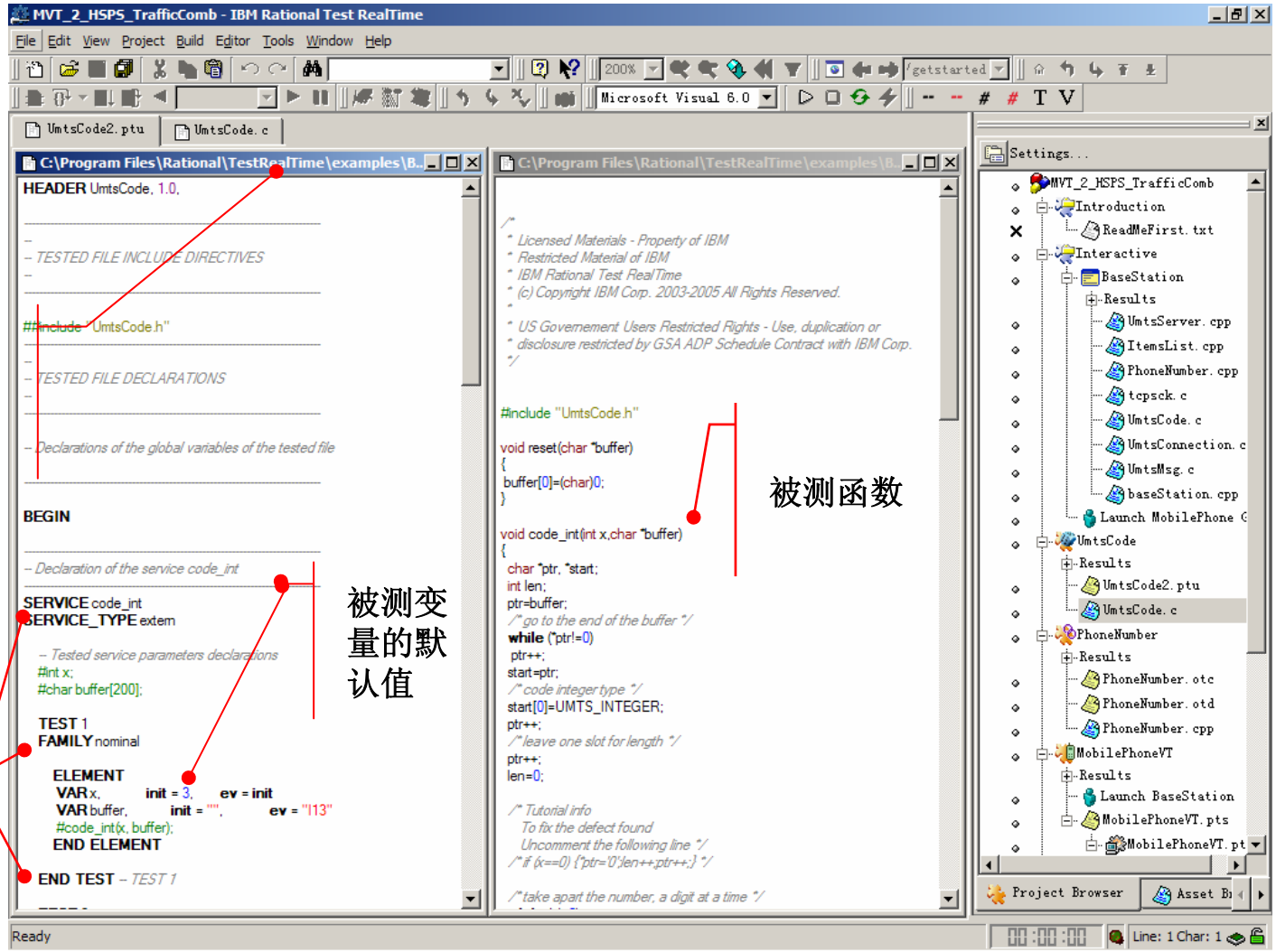
通过单元测试在开发过程中测试、分析并解决错误

- 在编码过程中测试
  - 自动化的组件测试模板和测试数据生成
  - 黑箱与白箱测试
  - 在各种复杂度级别：从测试一个简单的功能到测试分布的系统
  - 静态指标计算有助于
    - 排定测试优先级
    - 估算复杂度
  - 完整的回归测试



HSPS系统单元测试: 测试脚本

RTRT自动生成的测试脚本模板

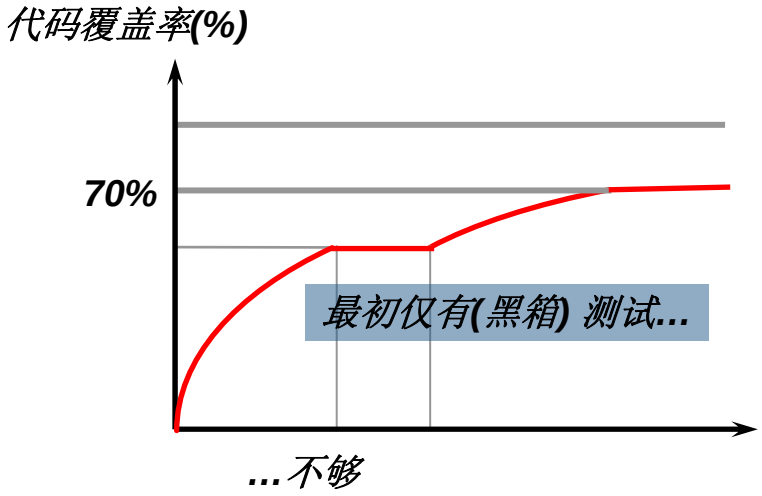


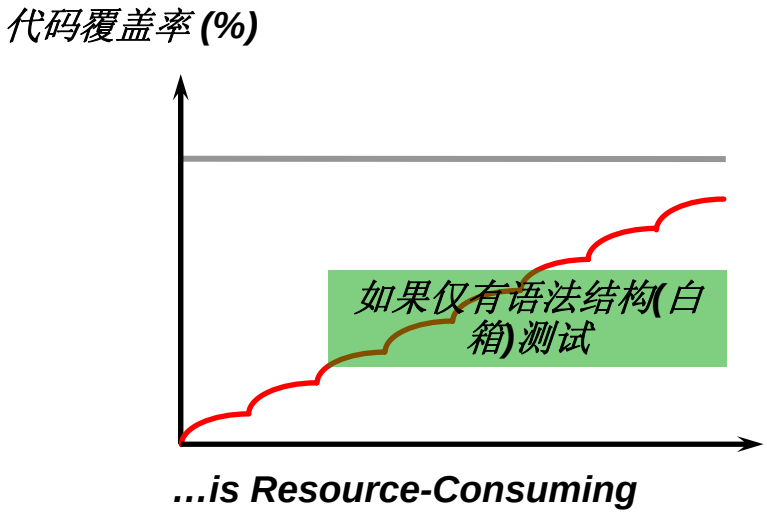
被测变量的默认值

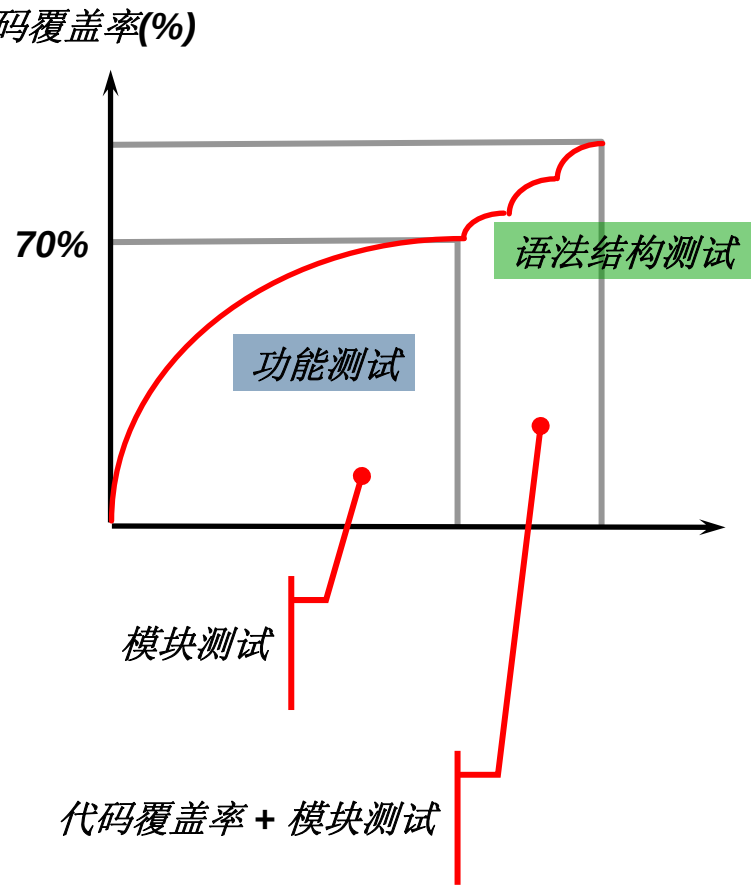
被测函数

默认的测试结构

HSPS系统单元测试中的黑箱与白箱测试

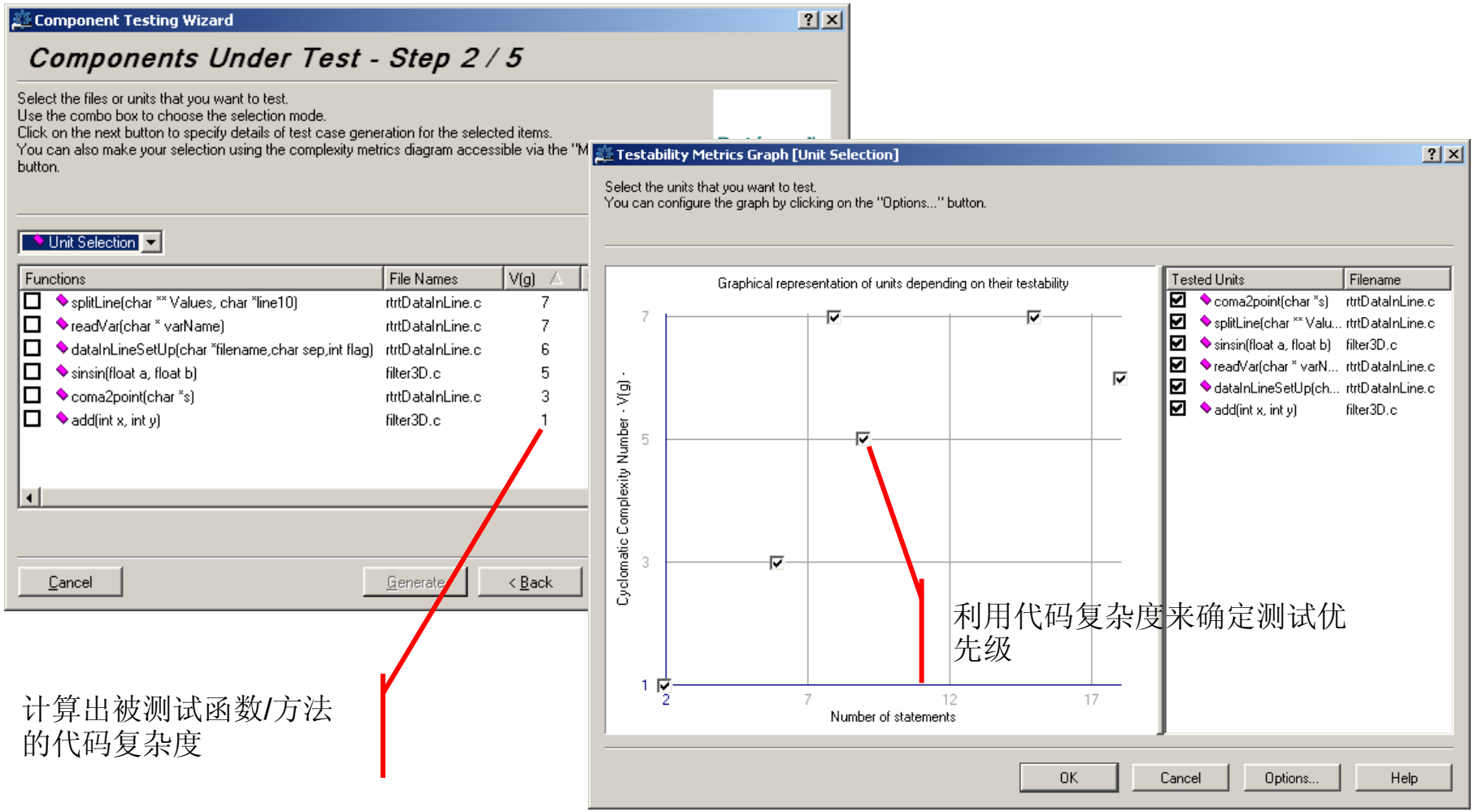




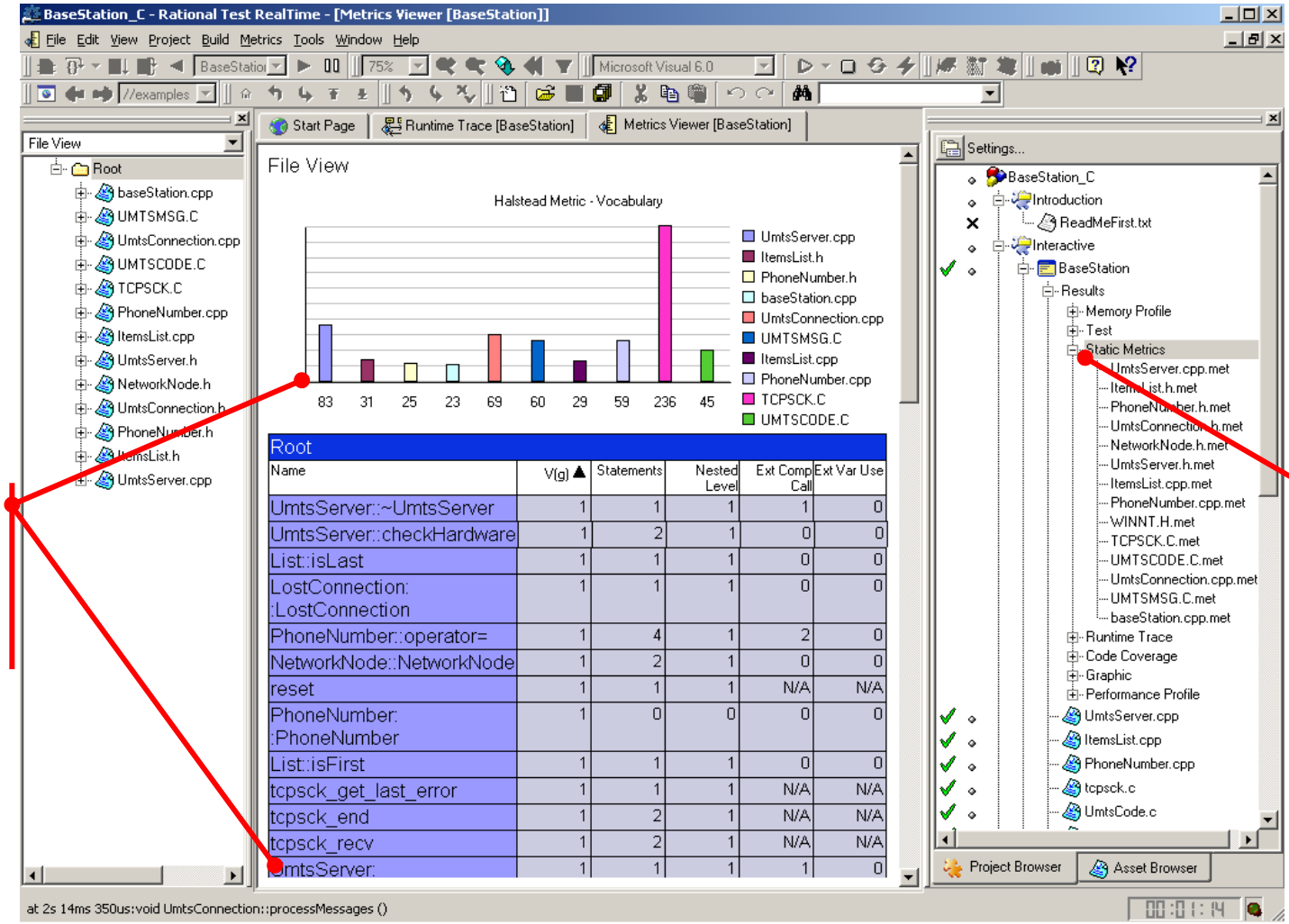


这是我们需要建议客户采取的最佳方式!

HSPS系统单元测试: *静态分析 - 1*



HSPS系统单元测试:*静态分析 - 2*

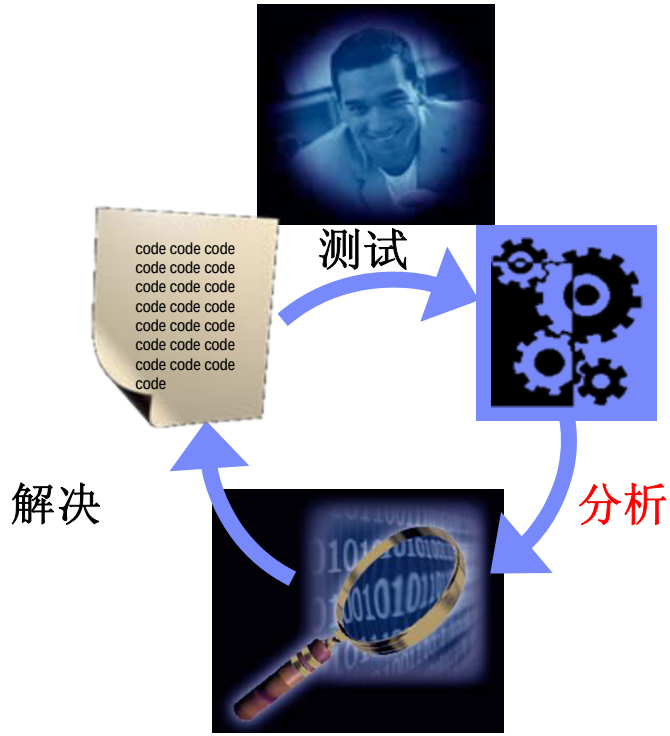


获得被插帧文件的代码度复杂信息

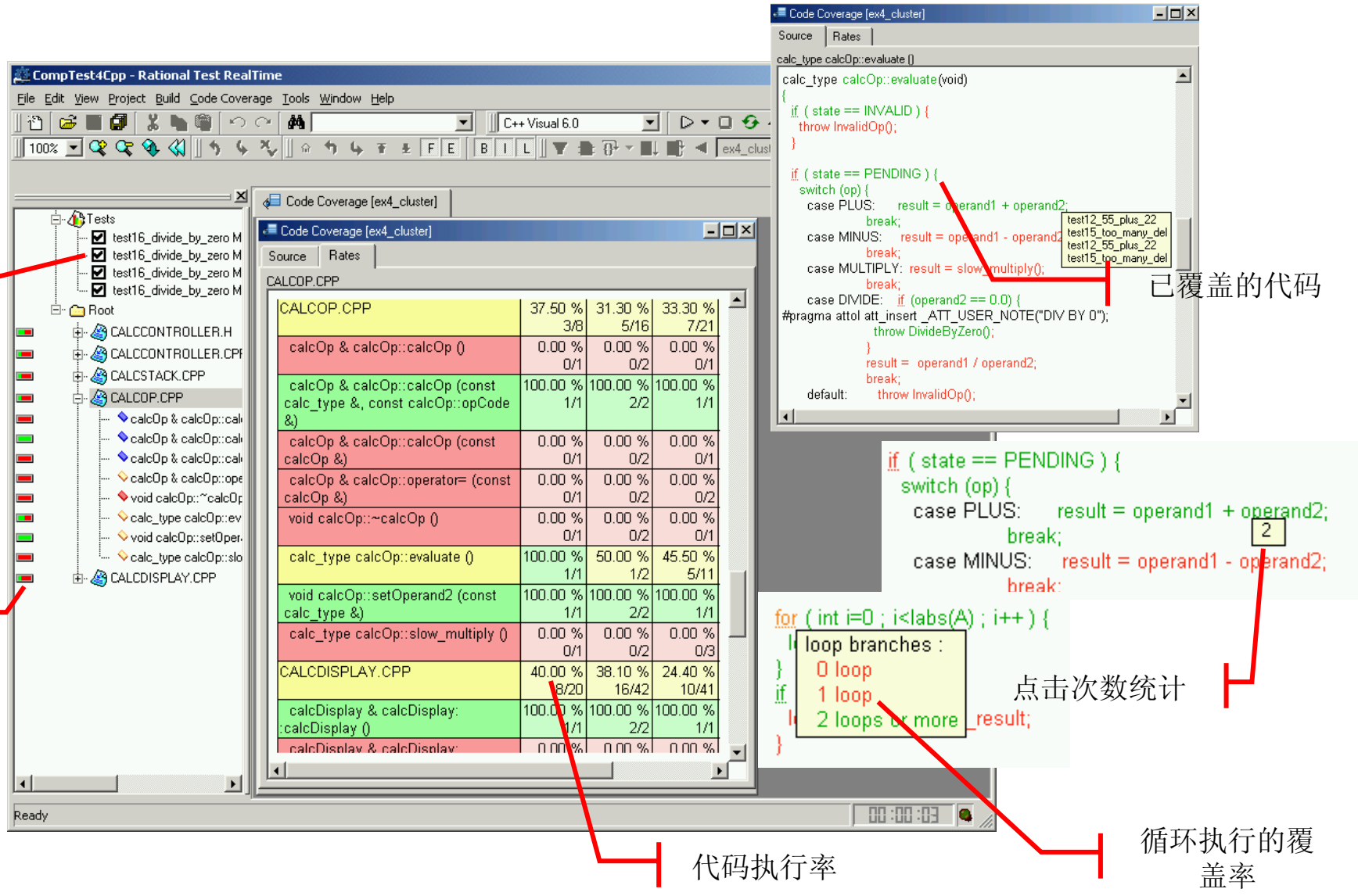
可随时访问静态分析数据

在开发过程中测试、分析并解决错误

- 在测试过程中分析
  - 代码覆盖率分析
  - 内存的概要分析
  - 性能的概要分析
  - 运行时追踪
  - 线程的概要分析

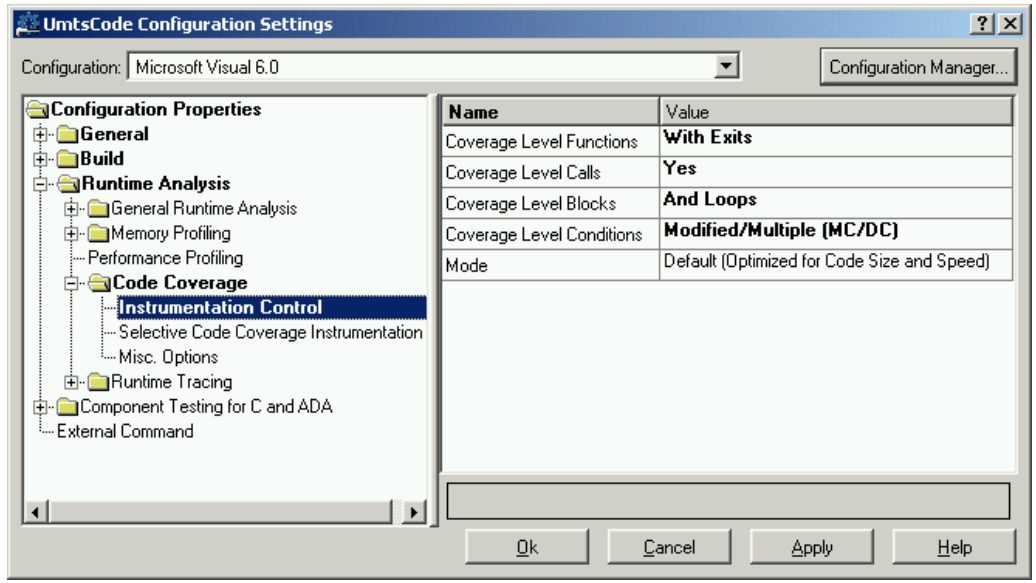


HSPS系统单元测试:代码覆盖率结果

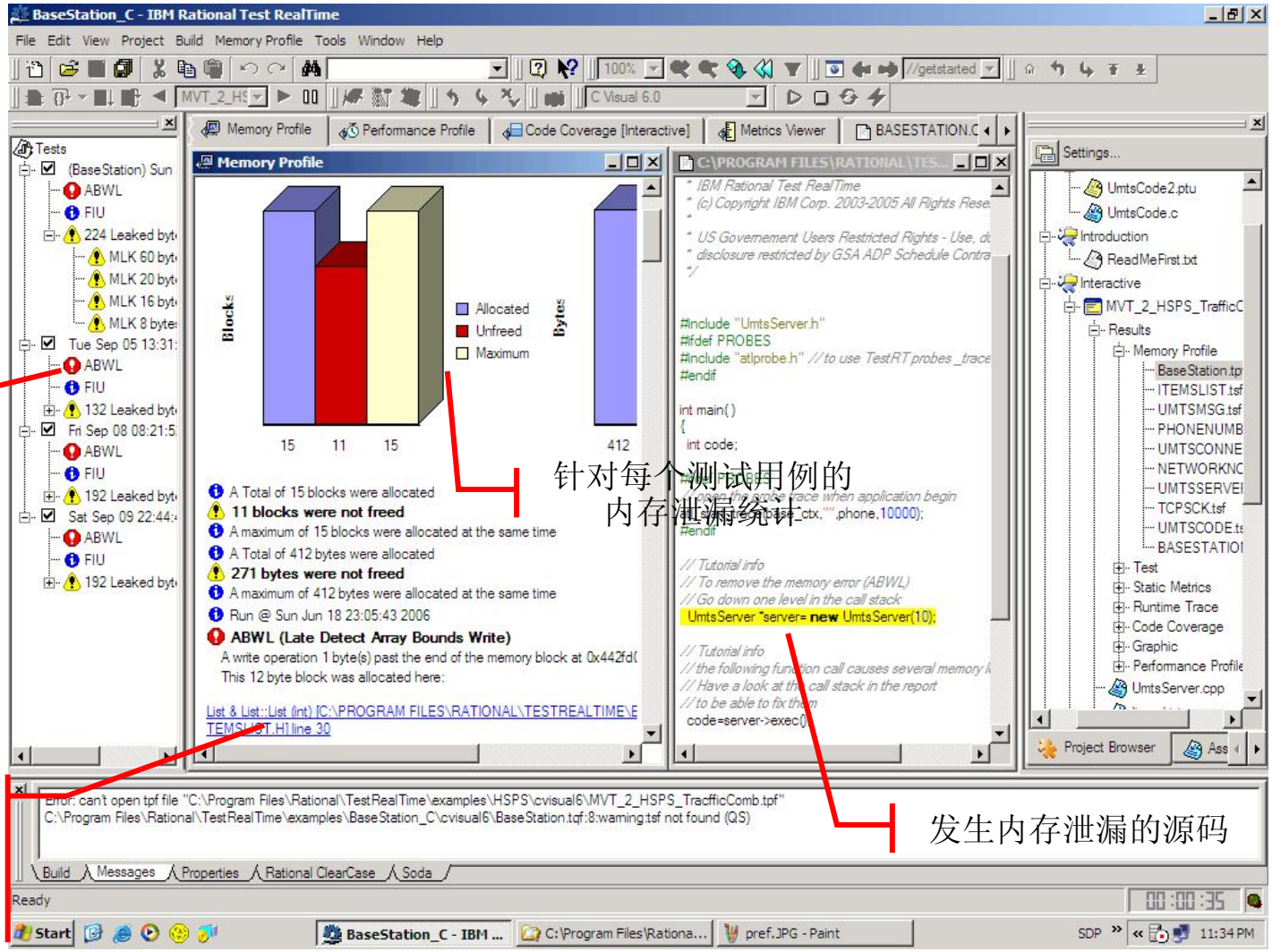


代码覆盖率:9个覆盖级别

- 覆盖级别:
  1. 函数入参覆盖Function Entry Coverage
  2. 函数体和出参覆盖
  3. 调用覆盖
  4. 语句块覆盖
  5. 冗余块
  6. 逻辑块覆盖 (loops)
  7. 基本条件
  8. 混合条件
  9. 分支条件
- 模式:
  - 默认模式 (1 byte per branch)
  - 紧凑模式 (1 bit per branch)
  - 计数模式 (1 integer per branch)



Runtime Analysis Features: *Memory Profiling*



内存错误类型

针对每个测试用例的内存泄漏统计

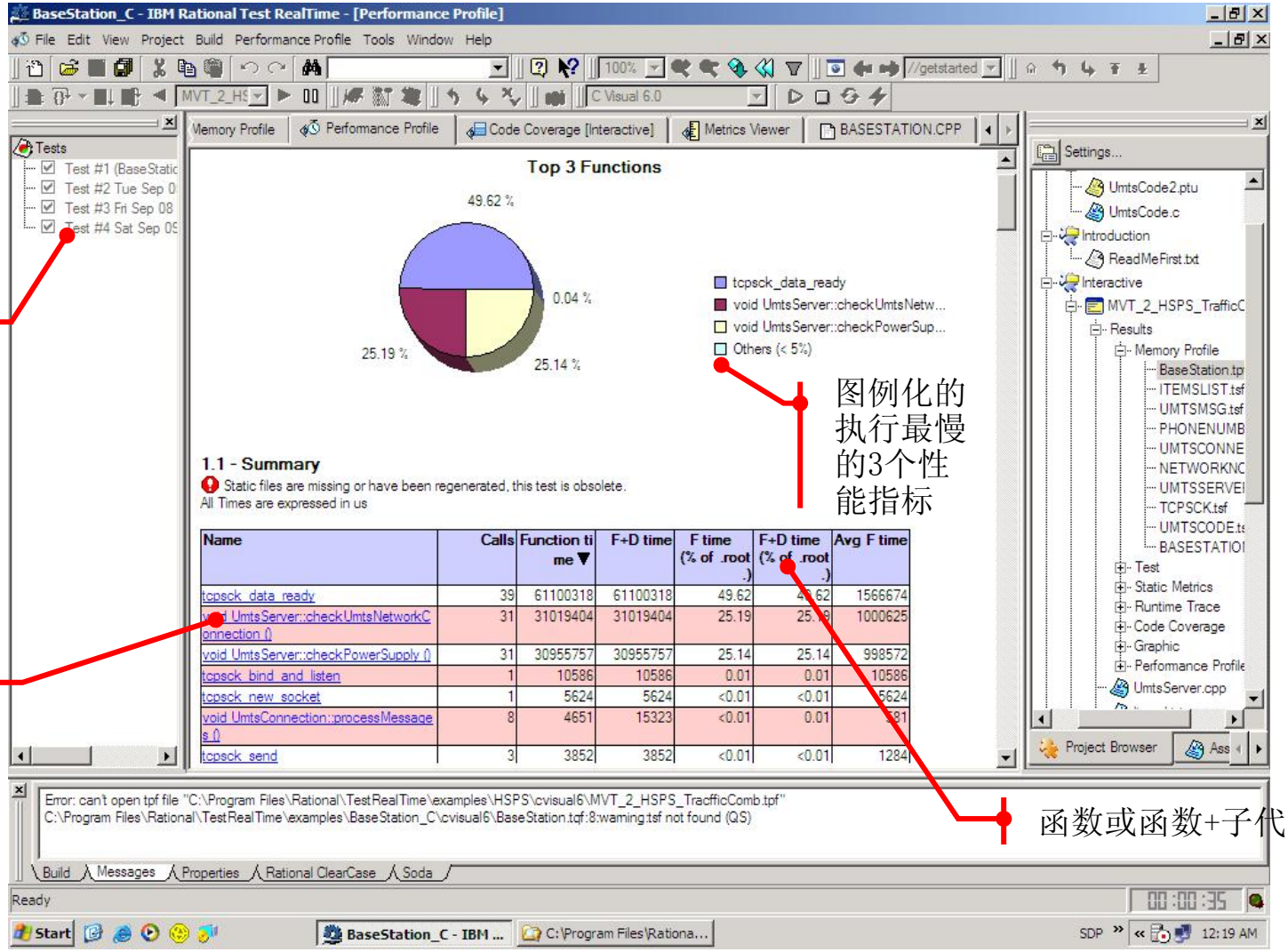
联接到发生内存泄漏的源码

发生内存泄漏的源码

Runtime Analysis Features: *Performance Profiling*

针对一或多个测试用例的性能的概要分析

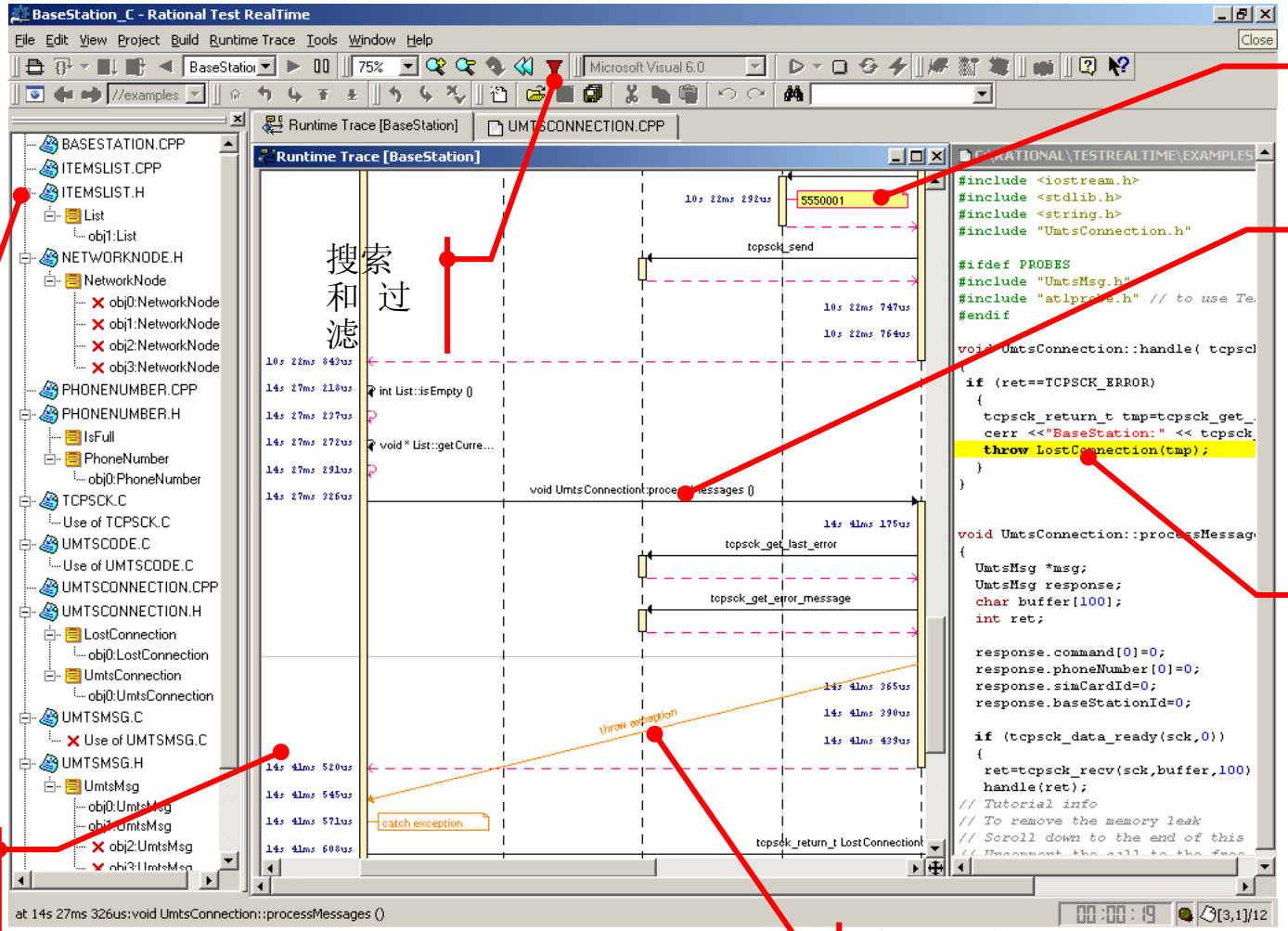
函数与方法列表



图例化的执行最慢的3个性能指标

函数或函数+子代时间度量

Runtime Analysis Features: *Runtime Tracing*



基于UML时序图生成的模块树图

搜索和过滤

跟踪数据值

On-the-fly 或者 post-mortem UML 时序图

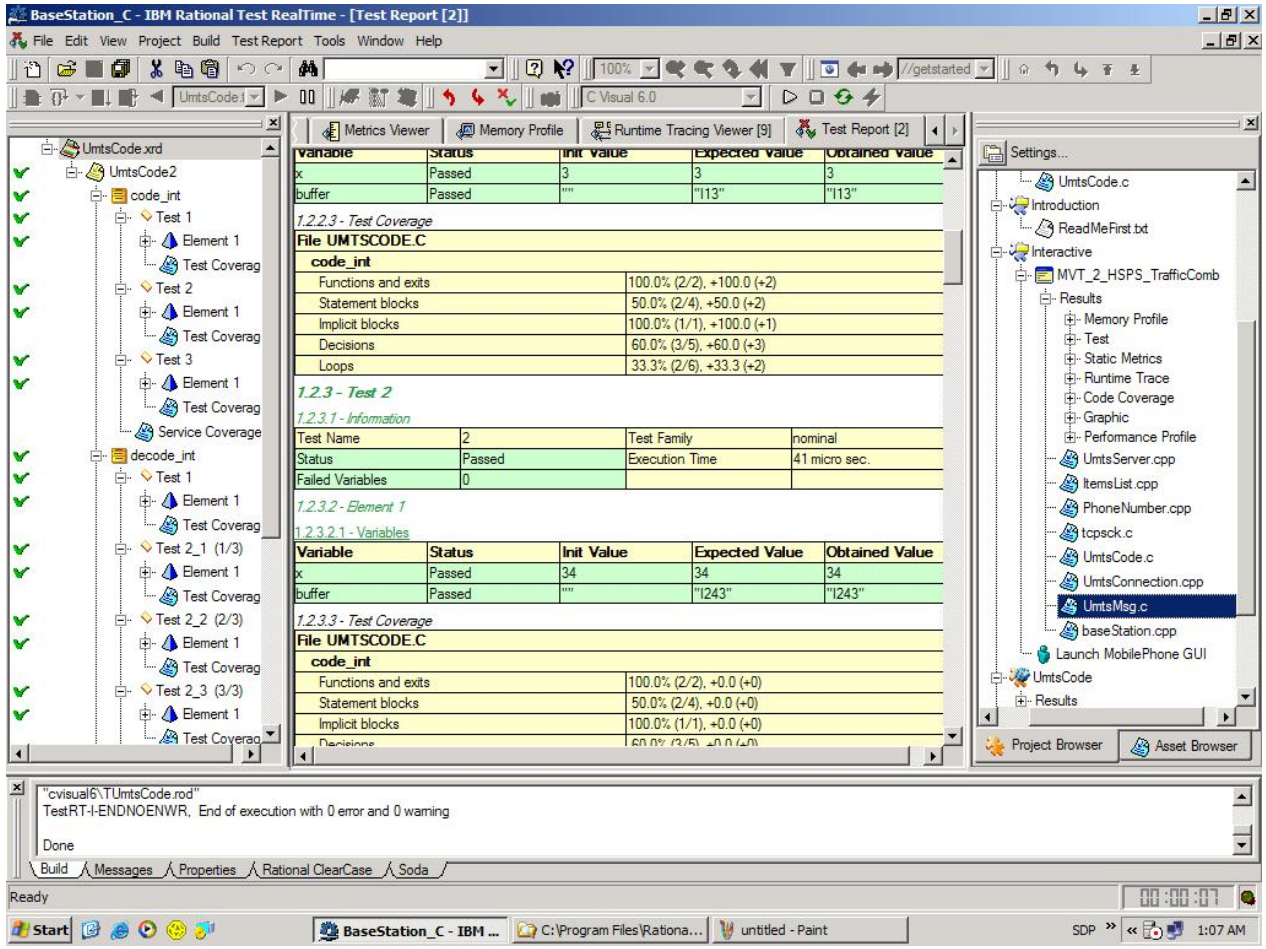
在源码中高亮正被跟踪的 event

计时信息

捕获异常抛出

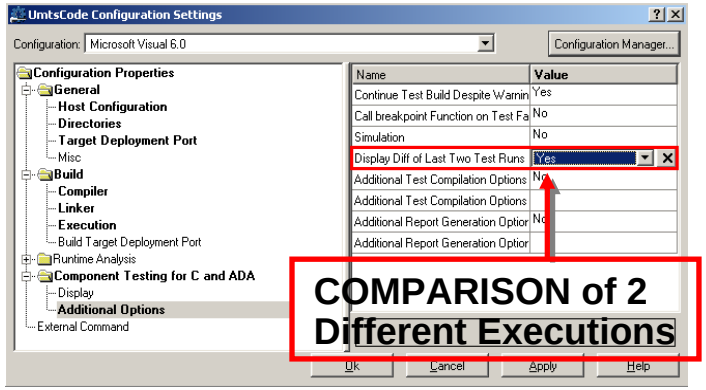
IBM Rational Test RealTime: *测试报告*

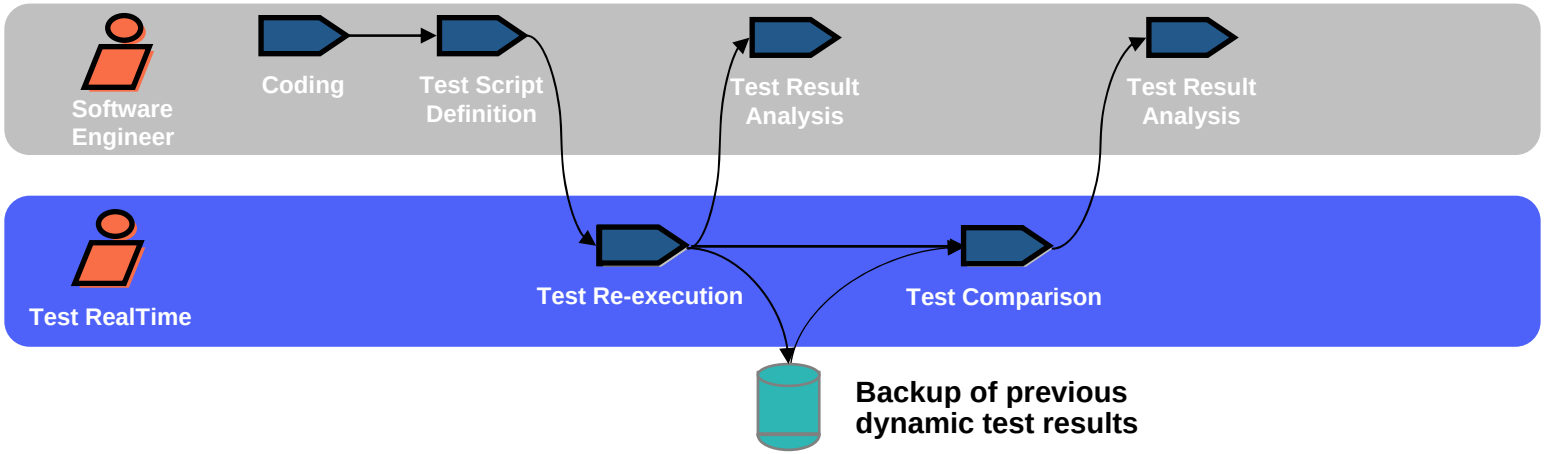
- 生成测试结果报表
  - Passed and failed test cases at a glance
  - Initial, expected, obtained values for all managed variables and stubs
  - Source code coverage information from Rational Test RealTime Code Coverage feature
- 导出为**HTML**格式
  - Distributed development
  - Test subcontracting



测试结果比较

- **Between:**
  - 2 iterations of same software component
  - 2 different development environments
  - Instrumented vs. non-instrumented code
  - Generated code vs. manual code

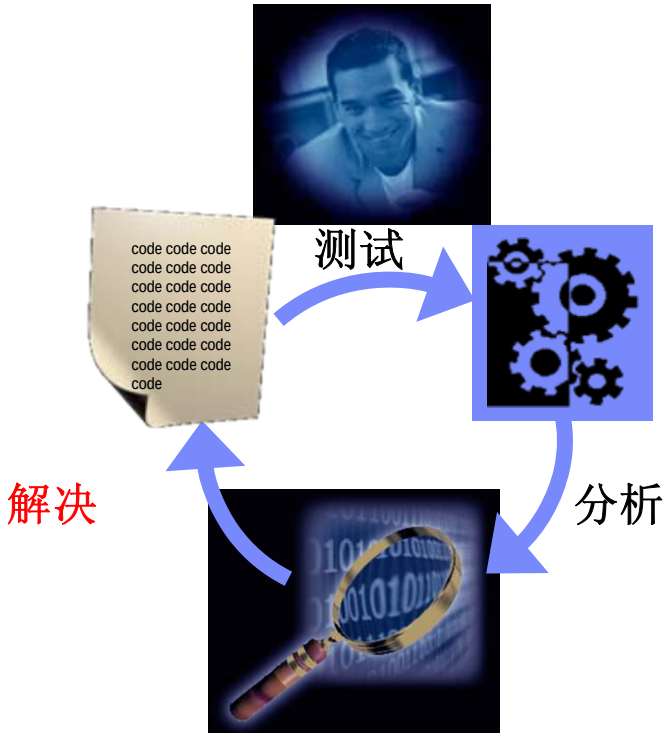




Variable	Status	Init Value	Expected Value	Obtained Value	Obtained value Comparison
x1	Failed	9	9	10	9
y1	Failed	9	9	10	9

在开发过程中测试、分析并解决错误

- 解决那些没有解决的问题
  - 测试的执行与排错器集成
  - 经过整理的、详细的、条目化的测试报告
  - 测试数据与运行时分析的结果和代码之间建立超链接



标题

- 第一天上午
  - 嵌入式系统的开发面临的挑战
  - 如何应对这些挑战?
  - IBM Rational针对嵌入式软件开发开发的测试解决方案
  - 案例介绍
  - 嵌入式系统开发的测试解决方案ROI
  - 总结及Q/A
- 第一天下午
  - Test RealTime实现针对嵌入式软件开发的运行时刻分析
    - TDP
    - 运行时刻分析
    - DEMO
  - Q/A


DEMO


Questions


Thank
You