



IBM Software Group

如何保证软件质量

IBM Rational质量保证方案

李剑波

IBM Rational

lijianbo@cn.ibm.com



@business on demand.

© 2004 IBM Corporation

议程

- Rational质量解决方案概览
- 代码测试
 - ▶ 如何保证C/C++传统软件代码质量: Rational PurifyPlus
 - ▶ 如何保证C/C++嵌入式软件代码质量: Rational Test RealTime
 - ▶ 如何保证Java代码质量: Rational Application Developer
- 系统测试
 - ▶ 功能测试自动化: Rational Functional Tester
 - ▶ 性能测试: Rational Performance Tester

议程

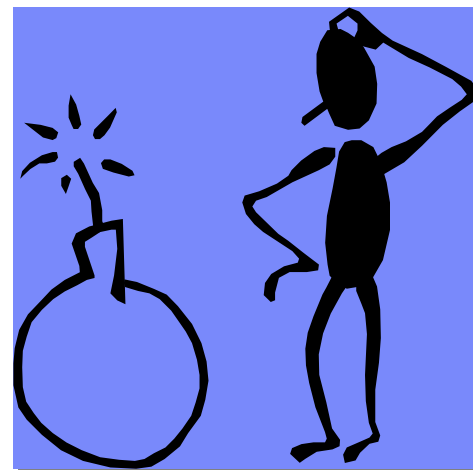
- 代码测试
 - ▶ 如何保证C/C++传统软件代码质量: Rational PurifyPlus
 - ▶ 如何保证C/C++嵌入式软件代码质量: Rational Test RealTime
 - ▶ 如何保证Java代码质量: Rational Application Developer
- 系统测试
 - ▶ 功能测试自动化: Rational Functional Tester
 - ▶ 性能测试: Rational Performance Tester

软件开发中的质量问题

在传统的软件开发中，软件功能的开发与软件的质量基本是分离的，也就是软件所具有的功能开发基本完成之后再进行全面测试。

潜在问题

延迟的软件发布时间
潜在的软件质量差距
潜在的软件修复代价



项目经理：我该怎么办？



项目失败的最主要的原因

主要矛盾：质量和速度



软件测试方面的具体原因如下：

- 缺乏统一的过程
- 缺乏有效的团队沟通
- 软件自动化的质量评测方法和工具
 - ▶ 功能
 - ▶ 性能
 - ▶ 可靠性



成功一定有方法

成功的质量保证方法 =
完善的过程 +
良好的团队沟通 +
正确的测试工具



IBM质量保证解决方案为您提供:

提供完善的测试管理流程

- ▶ 覆盖整个的软件测试生命周期

提供完善的测试管理平台, 保证团队的沟通:

- ▶ 覆盖整个的软件测试生命周期

全方位的自动化测试

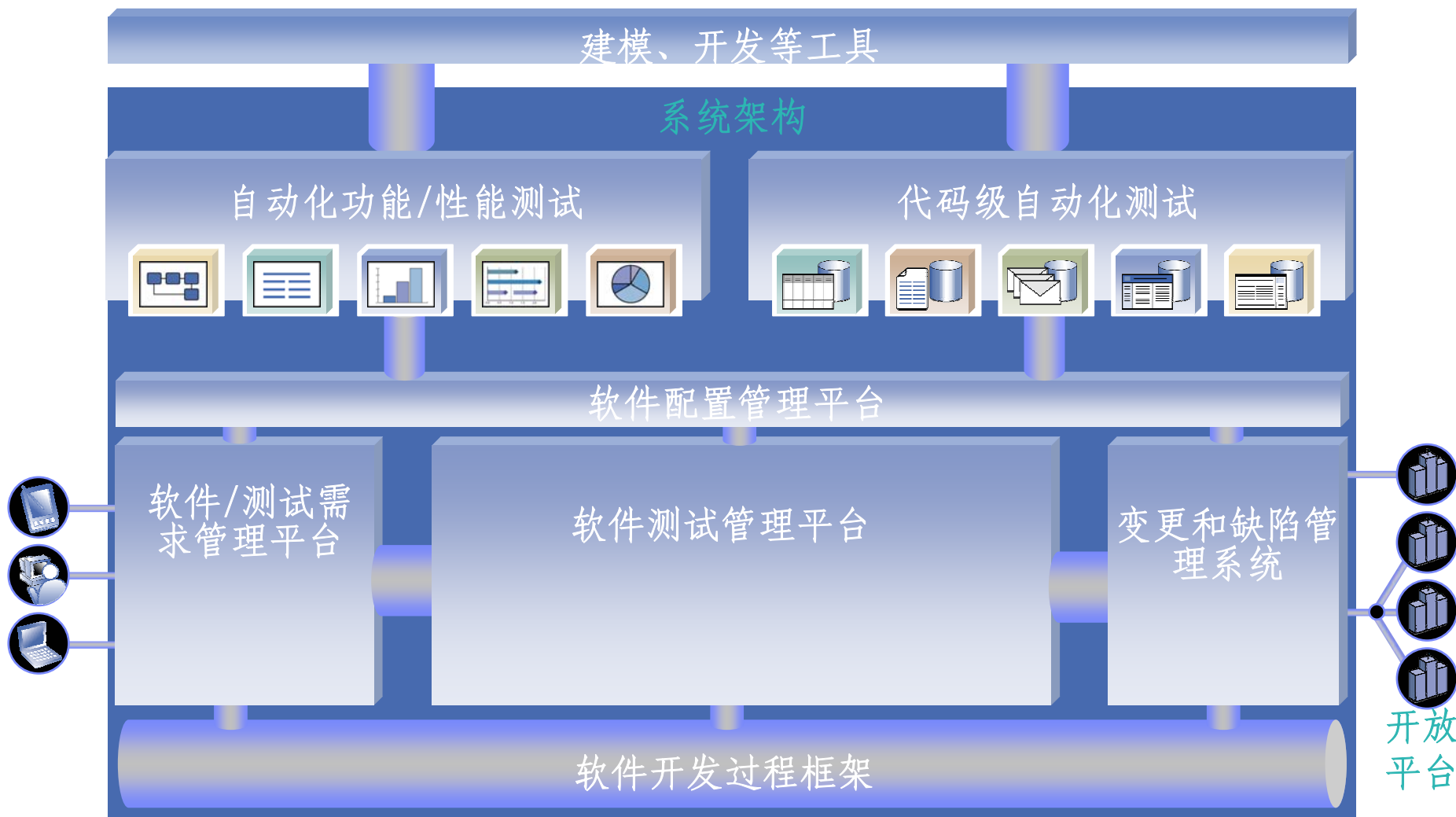
■ 完备的代码级测试:

- ▶ 代码覆盖率, 内存泄漏检查, 以及性能分析
- ▶ 方便的进行黑盒加白盒测试

■ 全面的回归测试

- ▶ 涵盖功能测试和性能测试
- ▶ 解决大批量数据驱动问题

IBM Rational质量解决方案系统架构



议程

- Rational质量解决方案概览
- 代码级测试
 - ▶ 如何保证C/C++嵌入式软件代码质量: Rational Test RealTime
 - ▶ 如何保证Java代码质量: Rational Application Developer
- 自动化/性能测试
 - ▶ 功能测试自动化: Rational Functional Tester
 - ▶ 性能测试: Rational Performance Tester

Rational PurifyPlus

- 传统C/C++应用的运行时分析工具
 - ▶ 内存分析
 - ▶ 性能分析
 - ▶ 代码覆盖
- 二进制代码插针
 - ▶ 对编译后形成的目标代码进行自动插针，然后再连接形成可执行文件。
- 其支持能力与编译器相关
 - ▶ Visual Studio 6.0, .NET 2003/2005
 - ▶ UNIX/LINUX平台下的gcc
 - ▶ Xlc(AIX)、C workshop(SUN Solairs)、acc(HP-UX)

```
# 连接成可执行文件
CC=cc
hello.pure: hello.o
                purify $(CC) -o hello.pure hello.o
# comiple成目标文件
hello.obj: hello.c
                $(CC) -g -c hello.c
```

Rational PurifyPlus/内存分析

指出数组越界读错误。

指出错误发生的位置。

指出错误引入的位置。

内存访问错误内存位置。

指出内存分配的代码。

是否支持Tuxedo/CICS中间件服务?

是否支持Oracle Pro*C程序?

充分利用IBM Rational技术支持网站:
<http://www.ibm.com/software/rational/support>

```

Purify: a.out
File View Actions Options Help
Finished a.out ( 1 error, 12 leaked bytes)
Purify instrumented a.out (pid 8701 at Fri Aug 8 16:22:57 2003)
ABR: Array bounds read
This is occurring while in:
_doprint [libc.so.1.9]
printf [libc.so.1.9]
main [hello_world.c:22]
#include <st
#include <ma
static char *helloWorld = "Hello, World";
main()
{
char *mystr = malloc(strlen(helloWorld));
strcpy(mystr, helloWorld, 12);
printf("%s\n", mystr);
}
Address 0x4423c is 1 byte past end of a malloc'd block at 0x44230 of 12 bytes
This block was allocated from:
malloc [rtlib.o]
main [hello_world.c:19]
start [crt0.o]
Current file descriptors in use: 5
Memory leaked: 12 bytes (100%); potentially leaked: 0 bytes (0%)
Program exited with status code 1.

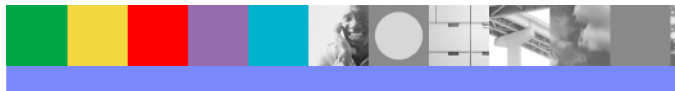
```

Rational PurifyPlus/性能分析

- 得到准确、可靠的性能数据，能统计出每个函数、每行代码的执行时间、执行次数以及耗时比重。
- 能比较代码调整前后的性能差异。
- 突出显示潜在的性能瓶颈。

The screenshot displays the Rational PurifyPlus performance analysis tool interface, showing several windows:

- CONTROL PANEL:** Shows performance and memory statistics for the process 'hello_world (pid 20352)'. It includes buttons for 'Function List', 'Call Graph', 'Help', and 'Exit'.
- ANNOTATED SOURCE:** Displays the annotated source code for 'hello_world.c'. It shows the execution time and distribution to callers for various functions, such as 'World' and 'main'.
- FUNCTION LIST:** Lists all 37 functions that match the search criteria, sorted by function time in microseconds. The 'main' function is highlighted as the most time-consuming.
- FUNCTION DETAIL:** Provides a detailed view of the selected function, 'main'. It shows the function name, file name, and various performance metrics, including function time, function-descendants time, and maximum function time.
- CALL GRAPH:** Visualizes the call graph for the selected function, showing the flow of control between different subroutines. It includes a search bar and buttons for 'Focus on Subtree', 'Previous Focus', 'Show Annotated Source', and 'Show Function Detail'.



Rational PurifyPlus/代码覆盖分析

- 目的：帮助开发人员不断丰富测试用例，提高测试完备性
- 指出未被测试的代码
- 实现多次执行的代码覆盖结果合并
- 自动生成各种代码覆盖统计报告

代码运行次数

源代码

Line	D	I	T	Hits	Annotated Source
16					void display_message();
17					
18					main(argc, argv)
19					int argc;
20					char** argv;
21					{
22				1	if (argc == 1)
23				1	display_hello_world();
24					else
25				0	display_message(argv[1]);
26				1	exit(0);
27				1	}
28					
29					void
30					display_hello_world()
31					{
32				1	printf("Hello, World\n");
33				1	}
34					
35					void
36					display_message(s)
37					char *s;
38					{
39				0	printf("%s, World\n", s);
40				0	}

代码行

未被测试代码

未被测试代码

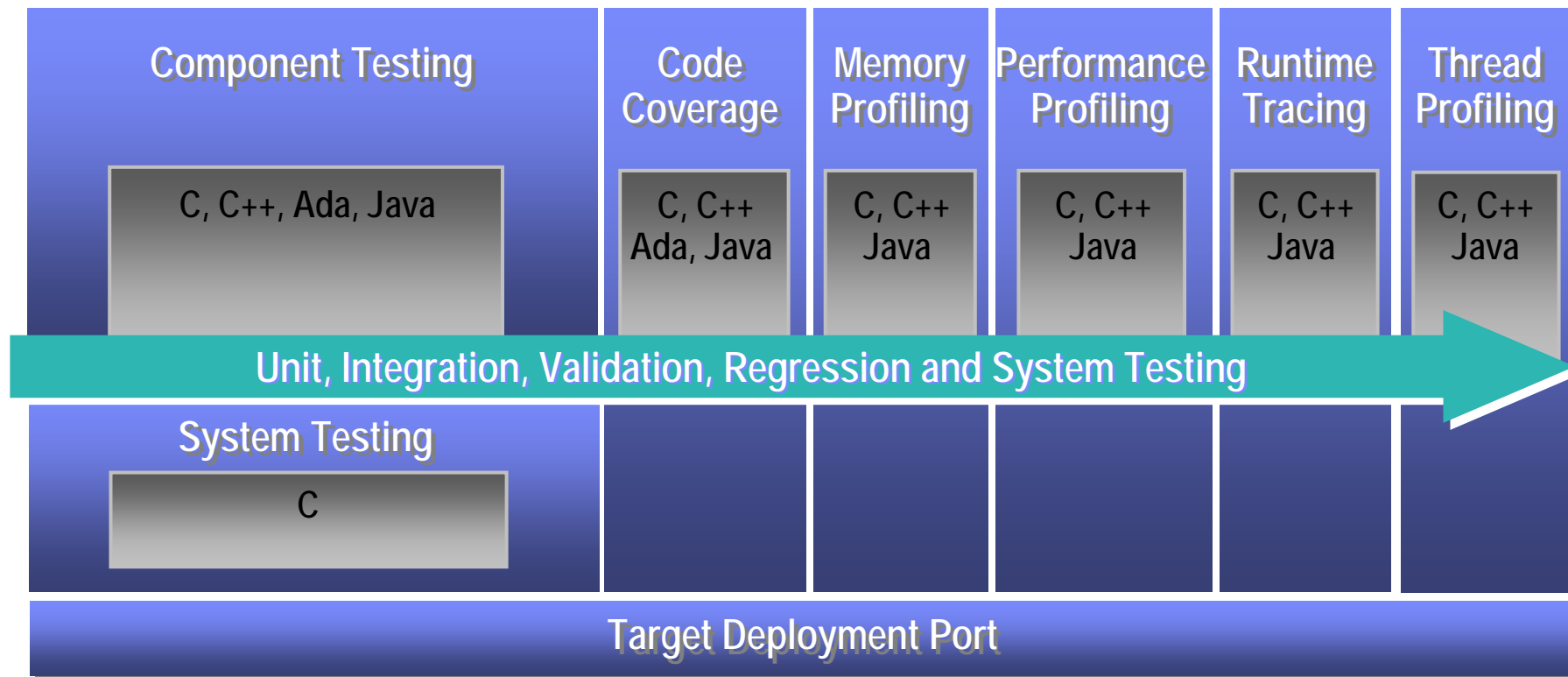


议程

- Rational质量解决方案概览
- 代码级测试
 - ▶ 如何保证C/C++传统软件代码质量: Rational PurifyPlus
 - ▶ 如何保证Java代码质量: Rational Application Developer
- 自动化/性能测试
 - ▶ 功能测试自动化: Rational Functional Tester
 - ▶ 性能测试: Rational Performance Tester

IBM Rational Test RealTime

同时支持开发机和目标的全面测试方案



- 基于源代码插针技术
- 利用可定制的TDP来屏蔽不同嵌入式软件开发时C/C++语言特殊性和开发工具的差异性

IBM Rational Test RealTime

广泛的平台支持

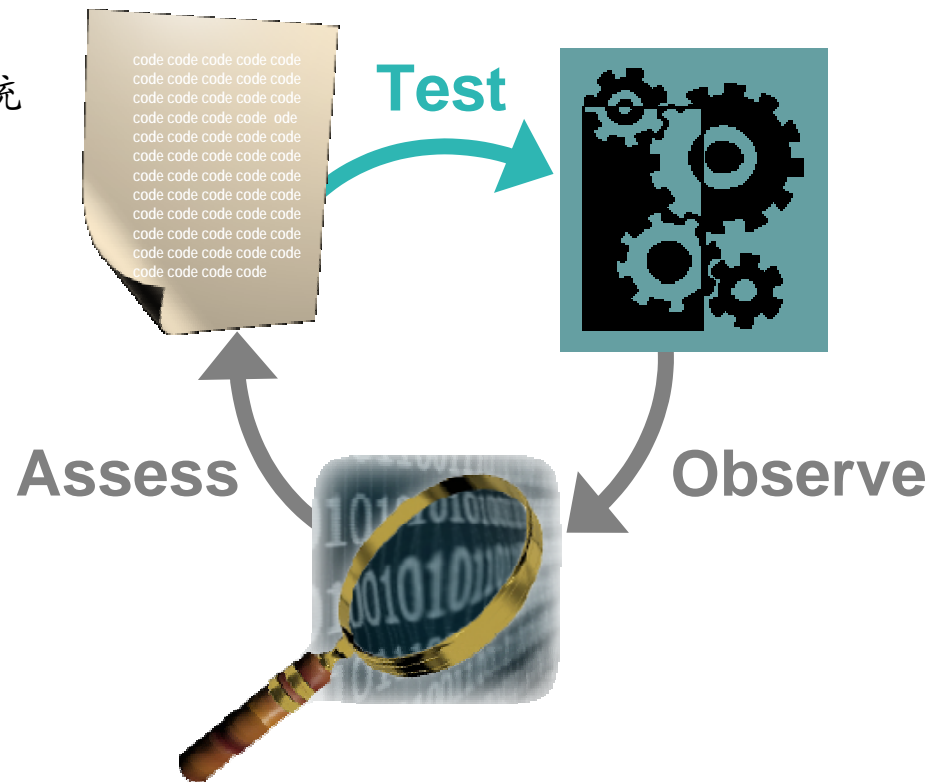
4-Bit to 64-Bit Cross-Development Environments			Languages
▪ <i>WindRiver</i>	▪ <i>Montavista</i>	▪ <i>Tasking</i>	▪ C
▪ <i>GreenHills</i>	▪ <i>TI</i>	▪ <i>CAD-UL</i>	▪ C++
▪ <i>ARM</i>	▪ <i>NEC</i>	▪ <i>Cosmic</i>	▪ Ada
▪ <i>Enea</i>	▪ <i>Hitachi</i>	▪ <i>Hiware</i>	▪ J2ME/J2SE
▪ <i>Windows CE</i>	▪ <i>Apex</i>	▪ <i>Hitex</i>	
▪ <i>LynuxWorks</i>	▪ <i>Sun</i>	▪ <i>Symbian</i>	
▪ <i>Lauterbach</i>	▪ <i>Microtec</i>	▪ <i>.....</i>	
			Platforms
			▪ <i>Windows</i>
			▪ <i>Solaris</i>
			▪ <i>RedHat</i>
			▪ <i>HP-UX</i>
			▪ <i>AIX</i>



IBM Rational Test RealTime

自动化软件测试过程

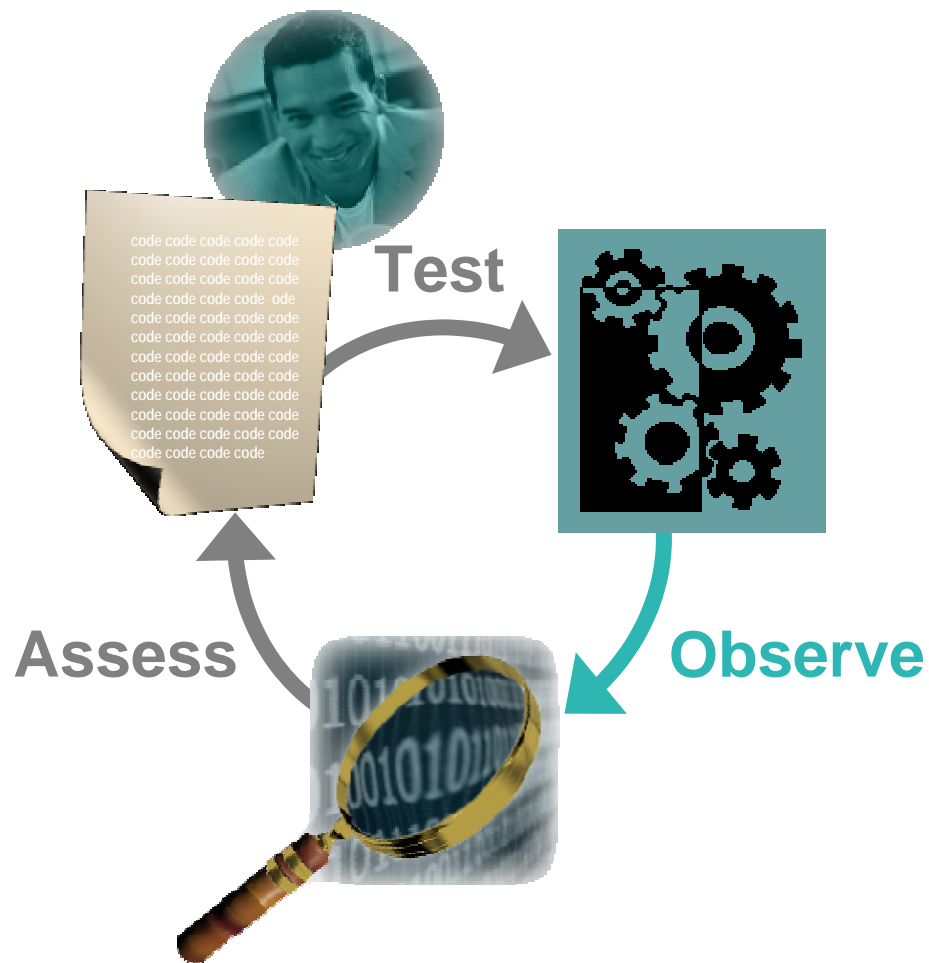
- 全面的自动化测试
 - ▶ 自动生成测试脚本模版和测试数据
 - ▶ 黑盒测试和白盒测试相集成
 - ▶ 多层次测试：从函数到分布式系统
 - ▶ 静态分析功能：
 - 明确测试优先级
 - 代码复杂度计算
 - ▶ 全面的回归测试



IBM Rational Test RealTime

自动化软件测试过程

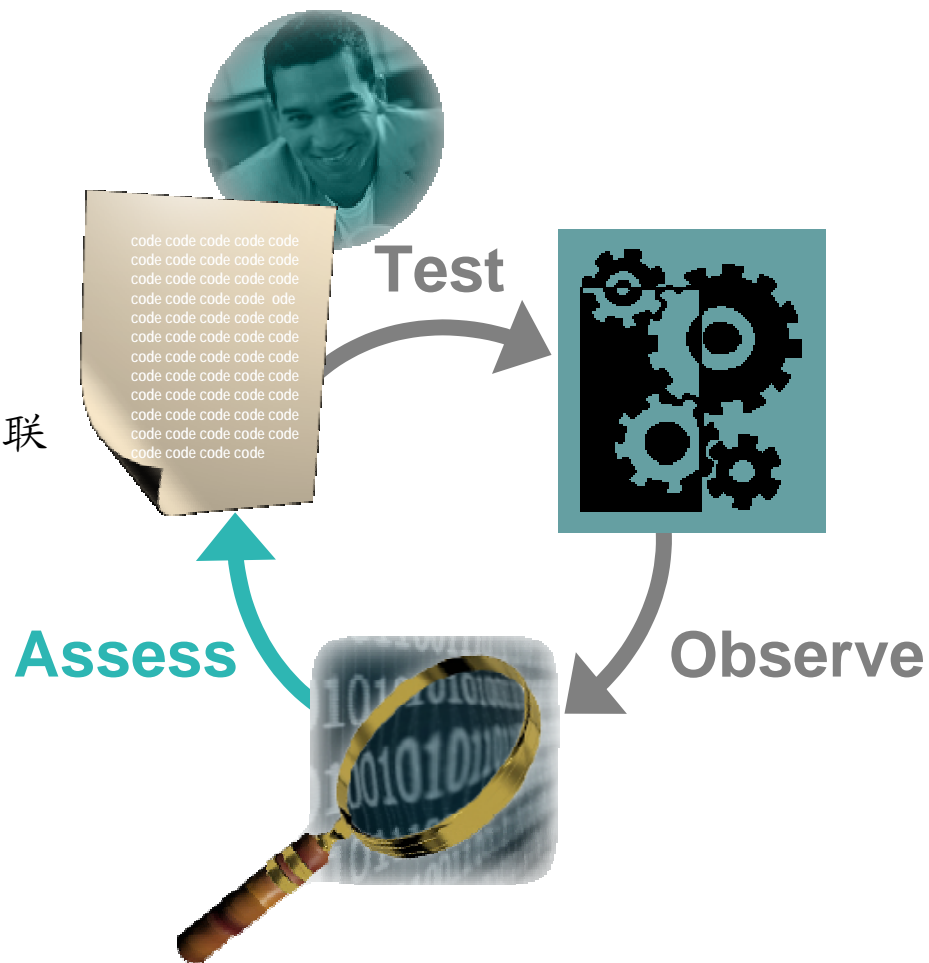
- 全面的自动化测试
- 全面的运行时分析
 - ▶ 代码覆盖率分析
 - ▶ 内存分析
 - ▶ 性能分析
 - ▶ 运行时分析
 - ▶ 线程分析



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



IBM Rational Test RealTime: 测试脚本

The screenshot displays the IBM Rational Test RealTime interface. The main window shows a test script for 'simple.ptu' with the following content:

```
HEADER Test_runtime_simple, ,
-----
-- variable declaration for simple.ptu
-----
#int coverage ;
#int simple_aa,simple_ab,simple_ac,simple_ad,simple_af,simple_ag;
#int s_link1,s_link2 ;

BEGIN

SERVICE simple_s1

TEST t1
FAMILY nominal

ELEMENT

VAR simple_aa, INIT = -5,   EV = -5
VAR simple_ab, INIT = 5,   EV = 6, delta=40*
VAR simple_ac, INIT = 5,   MIN= 5 , MAX=5
VAR simple_ad, INIT = 5,   EV==

VAR s_link1, INIT IN {1,2,3},EV==
VAR simple_af, INIT = 5, EV(s_link1) IN {5,5,5}
VAR simple_ag, INIT = 5, EV(s_link1) IN {5,5,5}

END ELEMENT
END TEST
END SERVICE

-----
-- init IN {...}
SERVICE simple_s2

TEST t1
FAMILY nominal
ELEMENT
VAR simple_aa, INIT IN {5,5,5},           EV = 5
END ELEMENT
END TEST

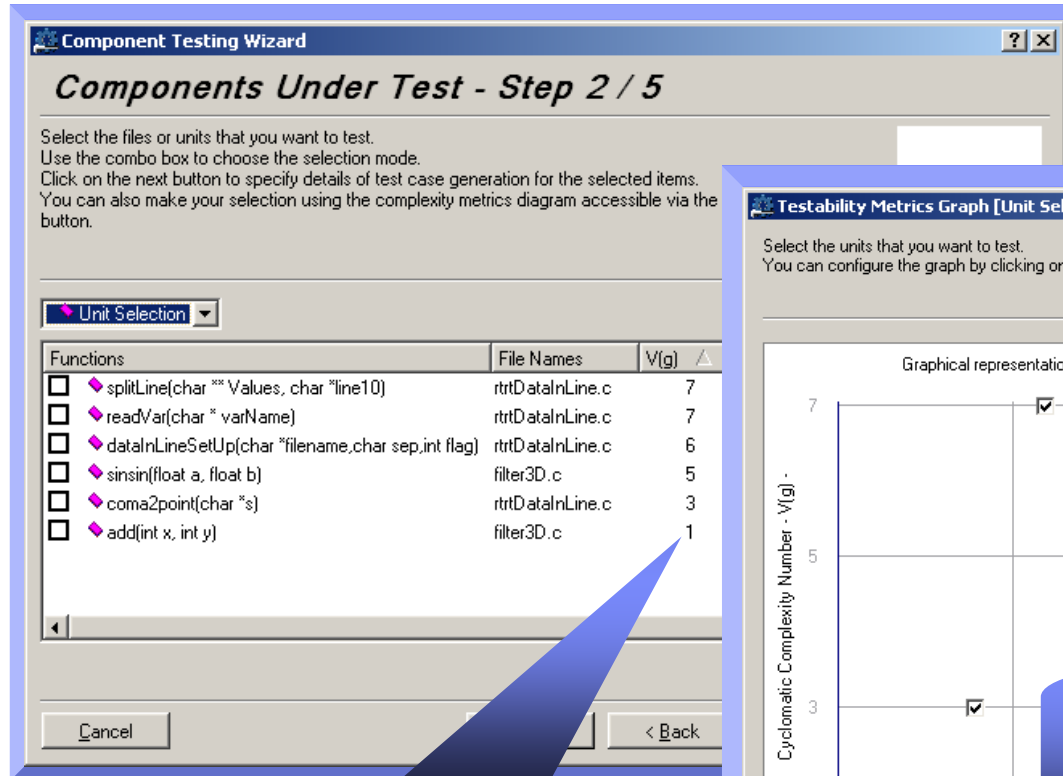
TEST t2
```

The right-hand pane shows a project browser for 'Testing_C' with a tree structure:

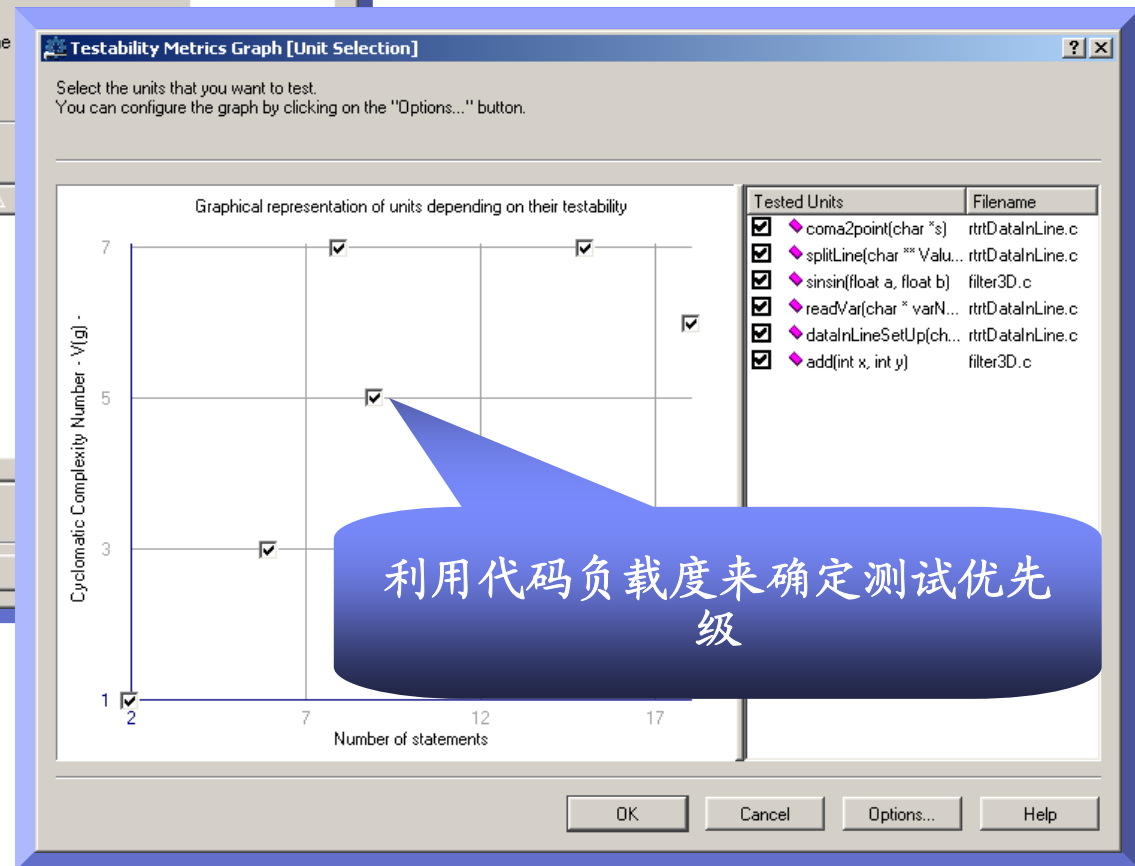
- Testing_C
 - double
 - Results
 - double.ptu
 - source.c
 - float
 - Results
 - float.ptu
 - source.c
 - general
 - Results
 - general.ptu
 - source.c
 - simple
 - Results
 - simple.ptu
 - source.c
 - str_nat
 - Results
 - str_nat.ptu
 - source.c
 - string
 - Results
 - string.ptu
 - source.c
 - struct
 - Results
 - struct.ptu
 - source.c
 - table

The status bar at the bottom indicates 'Ready' and 'Line: 31 Col: 1'.

IBM Rational Test RealTime: 静态分析 - 1



计算出被测试函数/方法的代码负载度



利用代码负载度来确定测试优先级

IBM Rational Test RealTime: 静态分析 - 2

The screenshot displays the 'Metrics Viewer' window for a project named 'BaseStation_C'. It features a 'File View' on the left showing a project tree with files like 'baseStation.cpp', 'UMTSMMSG.C', and 'UmtsConnection.cpp'. The main area shows a 'Halstead Metric - Vocabulary' bar chart and a table below it. The table lists various code elements and their metrics.

Name	V(g)	Statements	Nested Level	Ext Comp Call	Ext Var Use
Root					
UmtsServer::~UmtsServer	1	1	1	1	0
UmtsServer::checkHardware	1	2	1	0	0
List::isLast	1	1	1	0	0
LostConnection::LostConnection	1	1	1	0	0
PhoneNumber::operator=	1	4	1	2	0
NetworkNode::NetworkNode	1	2	1	0	0
reset	1	1	1	N/A	N/A
PhoneNumber::PhoneNumber	1	0	0	0	0
List::isFirst	1	1	1	0	0
tcpsck_get_last_error	1	1	1	N/A	N/A
tcpsck_end	1	2	1	N/A	N/A
tcpsck_recv	1	2	1	N/A	N/A
UmtsServer::	1	1	1	1	0

at 2s 14ms 350us: void UmtsConnection::processMessages ()

获得被插针文件的
代码负载度信息

可随时访问静态分析
数据

IBM Rational Test RealTime: 代码覆盖

The screenshot displays the IBM Rational Test RealTime interface for a C program named 'BaseStation_C'. The main window shows the 'Code Coverage [BaseStation]' view, which includes a file explorer on the left, a central code editor, a settings pane on the right, and a console window at the bottom.

File Explorer (Left): Shows a tree view of the project structure. The 'Tests' folder is expanded, showing two test runs: 'Test #1 Sat Apr 19 22:1' and 'Test #2 Sat Apr 19 23:1'. The 'Root' folder is also expanded, showing various source files like 'PHONENUMBER.H', 'UMTSCONNECTION.C', 'NETWORKNODE.H', 'UMTSSERVER.CPP', 'TCPSCK.C', and 'UMTSCODE.C'. The 'TCPSCK.C' file is selected, and its contents are displayed in the central editor.

Code Editor (Center): Displays the source code for the 'tcpsck_set_addr' function. The code is annotated with green checkmarks indicating code coverage. The function signature is 'tcpsck_return_t tcpsck_set_addr (tcpsck_sock_addr_t *addr, char *hostname, int portnum)'. The code includes a struct definition for 'hostent', a check for 'tcpsck_is_init', and logic for setting up a socket and host information. The console window shows the execution output, including the message 'BaseStation: A connection was forcibly closed by a peer'.

Settings Pane (Right): Shows the 'Settings...' dialog for the 'BaseStation_C' project. The 'BaseStation' folder is selected, and its contents are listed, including 'Results', 'UmtsServer.cpp', 'ItemsList.cpp', 'PhoneNumber.cpp', 'tcpsck.c', 'UmtsCode.c', 'UmtsConnection.cpp', 'UmtsMsg.c', 'baseStation.cpp', and 'Launch MobilePhone GUI'. The 'UmtsCode' folder is also expanded, showing 'Results', 'UmtsCode2.ptu', and 'UmtsCode.c'. The 'PhoneNumber' folder is also expanded, showing 'Results', 'PhoneNumber.etc', and 'PhoneNumber.otd'.

Console Window (Bottom): Shows the execution output for the program. The output includes the following text:


```

Executing \cvisual6\BBaseStation.exe ...
\cvisual6\BBaseStation.exe
Split unneeded
BaseStation: A connection was forcibly closed by a peer
    
```

The status bar at the bottom of the window shows 'Ready', '00:00:45', and 'Line: 130'.

IBM Rational Test RealTime: 内存分析

The screenshot displays the 'Memory Profile [BaseStation]' window. The main area shows two bar charts: 'Blocks Summary' and 'Bytes Summary'. The 'Blocks Summary' chart shows 12 allocated blocks, 10 unfreed blocks, and a maximum of 12 blocks. The 'Bytes Summary' chart shows 279 allocated bytes, 211 unfreed bytes, and a maximum of 279 bytes. Below the charts, a list of statistics is provided:

- A Total of 12 blocks were allocated
- 10 blocks were not freed**
- A maximum of 12 blocks were allocated at the same time
- A Total of 279 bytes were allocated
- 211 bytes were not freed**
- A maximum of 279 bytes were allocated at the same time
- Run @ Sat Apr 19 22:03:11 2003
- ABWL (Late Detect Array Bounds Write)**
A write operation 1 byte(s) past the end of the memory block at 0x431660

The left pane shows a tree view of tests, including '164 Leaked bytes in 5 blocks' and 'MLK 60 bytes (x2)'. The right pane shows a project browser with files like 'UmtsServer.cpp', 'ItemsList.cpp', and 'PhoneNumber.cpp'. The bottom status bar shows 'Ready' and a timer at '00:00:45'.

IBM Rational Test RealTime: 性能分析

Top 3 Functions

45.45 %
27.27 %
27.21 %
0.08 %
Others (< 5%)

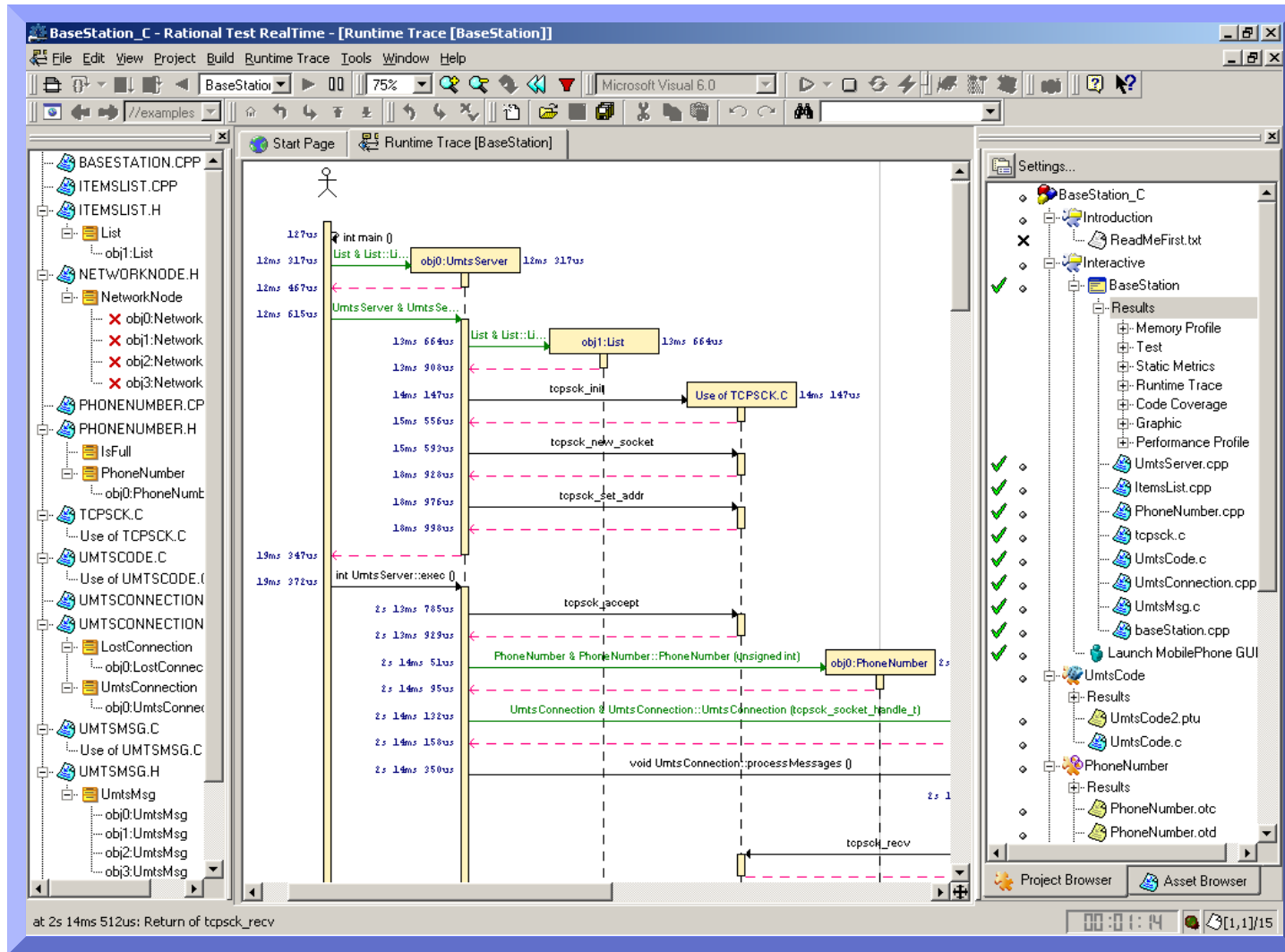
1.1 -Summary
All Times are expressed in us

Name	Calls	Function time	F+D time	F time (% of .root.)	F+D time (% of .root.)	Avg F time
tcpsck_data_ready	10	10014261	10014261	45.45	45.45	1001426
void UmtsServer::checkUmtsNetworkConnection ()	6	6007983	6007983	27.27	27.27	1001330
void UmtsServer::checkPowerSupply ()	6	5994633	5994633	27.21	27.21	999105
tcpsck_new_socket	1	6949	6949	0.03	0.03	6949
tcpsck_init	1	1394	1394	0.01	0.01	1394
tcpsck_send	2	1378	1378	0.01	0.01	689
lint main ()	1	92222034733	92222034733	<0.01	100.00	922

Settings...

- BaseStation_C
 - Introduction
 - ReadMeFirst.txt
 - Interactive
 - BaseStation
 - Results
 - UmtsServer.cpp
 - ItemsList.cpp
 - PhoneNumber.cpp
 - tcpsck.c
 - UmtsCode.c
 - UmtsConnection.cpp
 - UmtsMsg.c
 - baseStation.cpp
 - Launch MobilePhone GUI
 - UmtsCode
 - Results
 - UmtsCode2.ptu
 - UmtsCode.c
 - PhoneNumber
 - Results
 - PhoneNumber.otc

IBM Rational Test RealTime: 运行追踪图



IBM Rational Test RealTime: 测试报告

UmtsCode.wrd

- UmtsCode2
 - code_int
 - Test 1
 - Element 1
 - Test Coverage
 - Test 2
 - Element 1
 - Test Coverage
 - Test 3
 - Element 1
 - Test Coverage
 - Service Coverage
 - decode_int
 - Test 1
 - Element 1
 - Test Coverage
 - Test 2_1 (1/3)
 - Element 1
 - Test Coverage
 - Test 2_2 (2/3)
 - Element 1
 - Test Coverage
 - Test 2_3 (3/3)
 - Element 1
 - Test Coverage

1.2.3.2 - Element 1

1.2.3.2.1 - Variables

| Variable | Status | Init Value | Expected Value | Obtained Value |
|----------|--------|------------|----------------|----------------|
| x | Passed | 34 | 34 | 34 |
| buffer | Passed | "" | "1243" | "1243" |

1.2.3.3 - Test Coverage

File UMTSCODE.C

| code_int | Functions and exits | 100.0% (2/2), +0.0 (+0) |
|----------|---------------------|-------------------------|
| | Statement blocks | 66.7% (2/3), +0.0 (+0) |
| | Implicit blocks | none |
| | Decisions | 66.7% (2/3), +0.0 (+0) |
| | Loops | 33.3% (2/6), +16.7 (+1) |

1.2.4 - Test 3

1.2.4.1 - Information

| | | | |
|------------------|--------|----------------|---------------|
| Test Name | 3 | Test Family | nominal |
| Status | Failed | Execution Time | 29 micro sec. |
| Failed Variables | 1 | | |

1.2.4.2 - Element 1

1.2.4.2.1 - Variables

| Variable | Status | Init Value | Expected Value | Obtained Value |
|----------|--------|------------|----------------|----------------|
| x | Passed | 0 | 0 | 0 |
| buffer | Failed | "" | "110" | "10" |

1.2.4.3 - Test Coverage

File UMTSCODE.C

| code_int | Functions and exits | 100.0% (2/2), +0.0 (+0) |
|----------|---------------------|-------------------------|
| | | |

Settings...

 - BaseStation_C
 - Introduction
 - ReadMeFirst.txt
 - Interactive
 - BaseStation
 - Results
 - UmtsServer.cpp
 - ItemsList.cpp
 - PhoneNumber.cpp
 - tpscck.c
 - UmtsCode.c
 - UmtsConnection.cpp
 - UmtsMsg.c
 - baseStation.cpp
 - Launch MobilePhone GUI
 - UmtsCode
 - Results
 - UmtsCode2.ptu
 - UmtsCode.c
 - PhoneNumber
 - Results
 - PhoneNumber.etc

Build Messages / Properties / Rational ClearCase

```
-cio="cvisual6\atu.cio" -VA=EVAL
TestRT+-STARTEXEC, Rational(R) Test RealTime C and Ada Test Report Generator 2003.06.00.000.004
TestRT+-COPYRIGHT, Copyright(C) 1992-2002 Rational Software Corporation. All rights reserved.
C:\Rational\TESTRE~3\bin\intel\win32\rod2rd -g "I:\Rational\TestRealTime\examples\B\BaseStation_C\intermediates_files79024449.log" "-ocvisual6\UmtsCode.wrd" "cvisual6\TUmtsCode.rod"
TestRT+-TEST_ERRD, Unit Test Report Generator execution completed with incorrect tests
TestRT+-ENDNOEWAR, End of execution with 1 warning(s)
(rod2rd) Generation of graphic results "cvisual6\TUmtsCode_1.rtx" for decode_int/2.
```

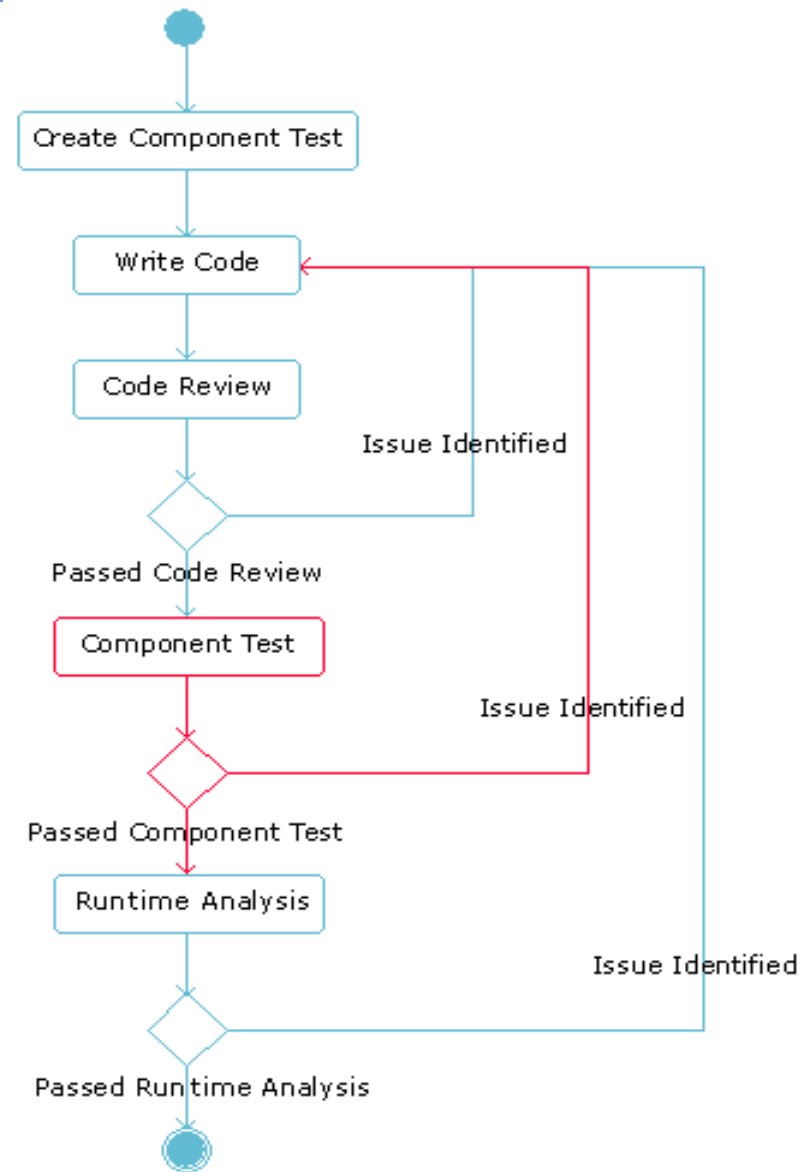
Ready 00:00:06

议程

- Rational质量解决方案概览
- 代码级测试
 - ▶ 如何保证C/C++传统软件代码质量: Rational PurifyPlus
 - ▶ 如何保证C/C++嵌入式软件代码质量: Rational Test RealTime
- 自动化/性能测试
 - ▶ 功能测试自动化: Rational Functional Tester
 - ▶ 性能测试: Rational Performance Tester

Rational Application Developer

- Java扩展集成开发环境
 - ▶ UML建模: 类图和序列图
 - ▶ 集成开发环境
 - ▶ 开发人员测试手段
 - 静态分析
 - 组件测试
 - 运行分析功能



Rational Application Developer/静态分析

- 利用UML图对代码中的类和依赖以及方法的内部行为进行可视化

The screenshot displays the IBM Rational Application Developer environment. The top window shows the source code for the `cashCheck` method in `PiggyBankControllerBean.java`. The code includes validation logic for a check against a CityBank account and a credit transfer to another account.

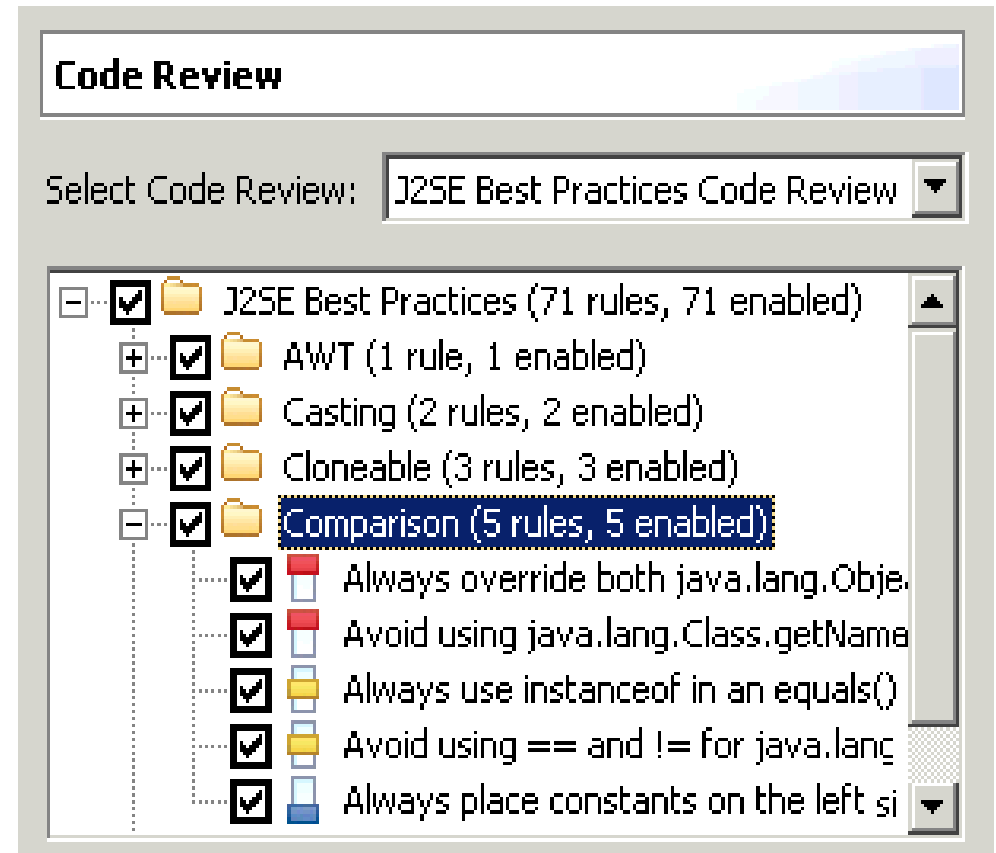
```
public void cashCheck(String checkReference, Integer checkAmount,
String accountToCredit) throws ServiceException
{
    // validate the check against CityBank
    boolean checkValid = false;
    try {
        CityBankDataAccessObject dao = new CityBankDataAccessObject();
        checkValid = dao.checkCityBankAccount(checkReference, checkAmount.intValue());
    } catch (Exception e) { // DataAccessException
        log.error(e.getMessage());
        throw new ServiceException(e.getMessage(), "error.connect.citybank", e);
    }
    if (!checkValid) {
        throw new ServiceException("Check " + checkReference + " with amount " + checkAmount + " is not valid", "error.check.invalid");
    }
    // credit account
    try {
        AccountLocalHome accountHome = (AccountLocalHome) getHome(ACCOUNT_EJB_REF);
        AccountLocal creditAccount = accountHome.findByPrimaryKey(new AccountKey(accountToCredit));
        creditAccount.setBalance(new Integer(creditAccount.getData().getBalance().intValue() + checkAmount.intValue()));
    }
}
```

The bottom window shows a Static Method Sequence Diagram for the `cashCheck` method. The diagram illustrates the sequence of method calls within the method's execution. It shows the creation of a `CityBankDataAccessObject` instance, followed by a call to its `intValue` method, and a return of `intValue`. The diagram is annotated with `distruptable` and `[try]` blocks.

Static Method Sequence Diagram: 理解特定方法的内部行为

Rational Application Developer/静态分析

- 代码评审：依据一系列规则来自动发现代码中的问题
 - ▶ 提高代码评审的效率，而且保证所有代码的一致性
 - ▶ 发现代码中的缺陷
 - ▶ 检查代码是否遵循编码规范和经验
 - ▶ 提供每个缺陷的详细说明以及解决方案
 - ▶ 对某些问题进行自动修复。
 - ▶ 可自定义规则来确保代码遵循特定的编码标准



Rational Application Developer/静态分析

- 利用Component Testing来获得代码的度量

Create Java Component Test

Select the components under test
Use the calculated metrics to help you choose the components to test. Numbers that are above average for the column are highlighted.

Components:

| Name | Architecture | | | | Component Complexity | | | | |
|--|--------------|--------|---------|----------|----------------------|---------|------------|---------------|------|
| | Level | Fan In | Fan Out | Ext User | Attributes | Methods | Statements | Nesting Level | V(g) |
| <input checked="" type="checkbox"/> CityBankServiceLocator | 0 | 14 | 0 | 2 | 0 | 14 | 78 | 2 | 4 |
| <input type="checkbox"/> CityBankSoapBindingStub | 0 | 2 | 0 | 0 | 0 | 2 | 69 | 1 | 4 |
| <input type="checkbox"/> CityBankServiceInformation | 0 | 4 | 0 | 0 | 0 | 4 | 41 | 1 | 2 |
| <input type="checkbox"/> LogHelper | 0 | 11 | 0 | 3 | 0 | 11 | 24 | 1 | 1 |
| <input type="checkbox"/> TransferServlet | 0 | 2 | 0 | 0 | 0 | 2 | 22 | 4 | 5 |
| <input type="checkbox"/> DataAccessException | 0 | 8 | 0 | 1 | 1 | 7 | 17 | 1 | 1 |
| <input type="checkbox"/> ServiceException | 0 | 8 | 0 | 3 | 1 | 7 | 17 | 1 | 1 |
| <input type="checkbox"/> MainMenuServlet | 0 | 2 | 0 | 0 | 0 | 2 | 12 | 2 | 4 |
| <input type="checkbox"/> DisplayAccountsServlet | 0 | 2 | 0 | 0 | 0 | 2 | 11 | 1 | 2 |
| <input type="checkbox"/> CityBankSoapBindingImpl | 0 | 1 | 0 | 0 | 0 | 1 | 6 | 1 | 2 |
| <input type="checkbox"/> PiggyBankCustomerAccountCreation | 1 | 3 | 2 | 0 | 1 | 2 | 163 | 3 | 7 |
| <input type="checkbox"/> PiggyBankTeller | 1 | 3 | 2 | 0 | 0 | 3 | 159 | 3 | 7 |
| <input type="checkbox"/> CityBankProxy | 1 | 6 | 2 | 0 | 0 | 6 | 39 | 2 | 8 |
| <input type="checkbox"/> HomeFactory | 1 | 3 | 8 | 1 | 0 | 3 | 33 | 3 | 6 |
| <input type="checkbox"/> CityBankDataAccessObject | 1 | 3 | 5 | 0 | 0 | 3 | 21 | 1 | 2 |
| <input type="checkbox"/> ApplicationInitServlet | 1 | 2 | 4 | 0 | 0 | 2 | 19 | 2 | 4 |
| <input type="checkbox"/> PiggyBankEJBDelegateImpl | 2 | 6 | 7 | 1 | 0 | 6 | 49 | 1 | 3 |
| <input type="checkbox"/> PiggyBankEJBDelegateFactory | 3 | 2 | 1 | 0 | 0 | 2 | 7 | 1 | 2 |

Test name and location

Use defaults

Name: CityBankServiceLocatorTest

Package: test

Rational Application Developer/组件测试

- 基于JUnit测试框架
- 自动生成测试驱动代码
- 测试代码和测试数据分离
- 直观的测试报告

The screenshot displays the Rational Application Developer IDE. The top window shows the source code for `ZipcodeTest.java` in the `test` package, which extends `TestCase` and contains two test methods: `test_isZipcode()` and `test_getCity()`. The bottom window, titled "Test Data Comparator", shows the execution results for the `test_isZipcode()` method. The table below summarizes the data shown in the comparator.

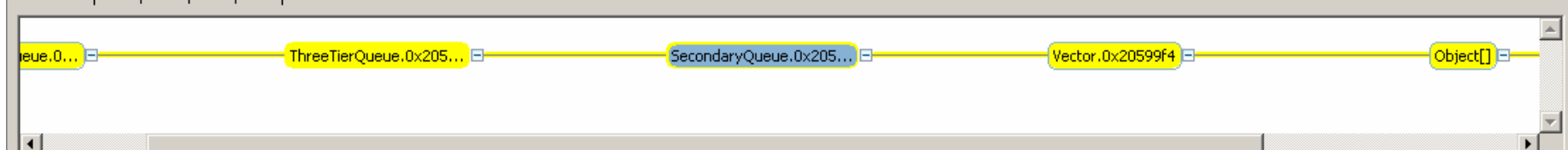
| Action | Type | New Test Data 2 | | |
|---|-------------------|-----------------|-----------------------------------|-----------------------------------|
| | | In | Expected | Actual |
| <code>objZipcode = new Zipcode()</code> | XY | | | |
| <code>objZipcode</code> | com.acme.commo... | | | |
| <code><expected exception></code> | Throwable | | <code><no exception></code> | <code><no exception></code> |
| <code>retValue = objZipcode.isZipcode(zipcode)</code> | XY | | | |
| <code>zipcode</code> | java.lang.String | 123456-789 | | |
| <code>retValue</code> | boolean | | true | false |
| <code><expected exception></code> | Throwable | | <code><no exception></code> | <code><no exception></code> |

Test Data Comparator loaded.

Rational Application Developer/运行时分析: 内存泄漏

对象引用图 [堆: 2] - ibm-myang 上的 com.queues.TestThreeTierQueue [PID : 3048]

缩放: [Slider] 突出显示: GC 根



可视: 6/7 突出显示: 4/4 SecondaryQueue.0x205983f

对象详细信息 [堆: 2] - ibm-myang 上的 com.queues.TestThreeTierQueue [PID : 3048]

对象: SecondaryQueue.0x205983f 突出显示: GC 根

- *LC* String
- *LC* Vector.0x20599f4
- *LC* TestThreeTierQueue.0x2057283
- *LC* Object[]

| 作出引用的对象 | 大小 | 计数 | 引用 | 作出引用... | 传播内存 |
|--------------------------|----|----|----|---------|------|
| ThreeTierQueue.0x20570a6 | 12 | 1 | 1 | 1 | 8 |

| 被引用对象 | 大小 | 计数 | 引用 | 被引用对象 | 传播内存 |
|------------------|----|----|----|-------|------|
| Vector.0x20599f4 | 16 | 1 | 1 | 1 | 16 |

疑似泄漏对象 [堆: 2] - ibm-myang 上的 com.queues.TestThreeTierQueue [PID : 3048]

堆转储: 标识: 1, 名称: optHeap.20051101.175556.3048.trcopt
堆转储: 标识: 2, 名称: optHeap.20051101.175631.3048.trcopt

| <可能性 | 泄漏源 | 容器类型 | 正在泄漏的对象 | 泄漏数 | 泄漏的字节数 | 泄漏的对象数 |
|------|---------------------------|------------------|---------|--------|------------|---------|
| 100 | TestThreeTierQueue.0x2... | Vector.0x20599f4 | String | 86,000 | 15,824,000 | 172,000 |

Rational Application Developer/运行时分析：性能分析

执行统计信息 - ibm-myang 上的 Sort [PID : 8704] (过滤器：没有过滤器)

| >包 | 额定时间 (秒) | 平均额定时间 (秒) | 累积时间 (秒) | 调用次数 |
|-------------------------------|----------|------------|----------|------|
| Bsort(int[]) void | 0.000063 | 0.000063 | 1.267116 | 1 |
| bsort(int[], int, int) void | 1.267053 | 1.267053 | | 1 |
| computemed(int[]) void | 0.027202 | 0.013601 | | 2 |
| main(java.lang.String[]) void | 0.011638 | 0.011638 | | 1 |
| Qsort(int[]) void | 0.000125 | 0.000125 | 5.886789 | 1 |
| quick(int[], int, int) int[] | 5.886664 | 0.001174 | 5.886664 | 5015 |
| java.lang | 0.000000 | 0.000000 | 0.000000 | 0 |

方法的执行时间、测试

性能调用图 - Java Profiling Agent - 进程：8704

缩放： [Slider] 突出显示：到根的最大路径

调用图，显示性能瓶颈

线程：main
线程：Reference Handler
线程：Finalizer
线程：Thread-0

Sort.Qsort
Sort.computemed
Sort
RandNum.RandNum

可视：13/13 突出显示：5/5 Sort.quick

方法详细信息 - Java Profiling Agent - 进程：8704

突出显示：到根的最大路径
Sort.quick

分析特定方法的详细性能数据

| 调用者 | 调用者 | 主机 | 进程 | >百分比 (秒) | 调用 | 传播时间 |
|------------|------------|-----------|------|----------|-------|------|
| Sort.Qsort | Sort.quick | ibm-myang | 8704 | 100.00 | 1 | 5.89 |
| Sort.quick | Sort.quick | ibm-myang | 8704 | 99.02 | 5,014 | 5.83 |

子节点

子节点 主机 进程 >百分比 (秒) 调用 传播时间

Rational Application Developer/运行时分析: 代码覆盖

The screenshot displays the Rational Application Developer interface with two main panes:

- Source Editor (Left):** Shows the source code for `Sort.java`. The code is annotated with green checkmarks (✓) for covered lines and red dashes (—) for uncovered lines. The code includes a swap function, a recursive call, and a quick sort demonstration algorithm.
- Coverage Navigator (Right):** Shows a tree view of the project structure. The `Sort` class is expanded, showing the following methods with green circles indicating coverage: `main(java.lang.String[])`, `Bsort(int[]) void`, `bSort(int[], int, int) void`, `Sort()`, `Qsort(int[]) void`, `quick(int[], int, int) int[]`, `quickFast(int[], int, int) v`, and `computedmed(int[]) void`. The `RandNum` class is also expanded, showing `RandNum(int)` and `getRandNum() int[]` with green circles.

```
Sort.java
/* Swap elements a[lo] and a[hi] */
✓ if (lo < hi) {
✓     int T = numbers[lo];
✓     numbers[lo] = numbers[hi];
✓     numbers[hi] = T;
}

/* Put the median in the "center" of the list */
✓ numbers[hi0] = numbers[hi];
✓ numbers[hi] = pivot;

/* Recursive calls */
✓ int outNum[] = quick(numbers, lo0, lo - 1);
✓ System.arraycopy(outNum, lo0, numbers, lo0, lo-lo0);
✓ outNum = quick(numbers, hi + 1, hi0);
✓ System.arraycopy(outNum, hi+1, numbers, hi+1, hi0-hi);
✓ return numbers;
}

/* Quick Sort demonstration algorithm */
static void quickFast(int numbers[], int lo0, int hi0) {
-     int lo = lo0;
-     int hi = hi0;

-     if (lo >= hi) {
-         return;
-     } else if (lo == hi - 1) {
-         if (numbers[lo] > numbers[hi]) {
-             int T = numbers[lo];
-             numbers[lo] = numbers[hi];
-             numbers[hi] = T;
-         }
-         return;
-     }
}
```

议程

- Rational质量解决方案概览
- 代码级测试
 - ▶ 如何保证C/C++传统软件代码质量: Rational PurifyPlus
 - ▶ 如何保证C/C++嵌入式软件代码质量: Rational Test RealTime
 - ▶ 如何保证Java代码质量: Rational Application Developer
- 自动化/性能测试
 - ▶ 性能测试: Rational Performance Tester

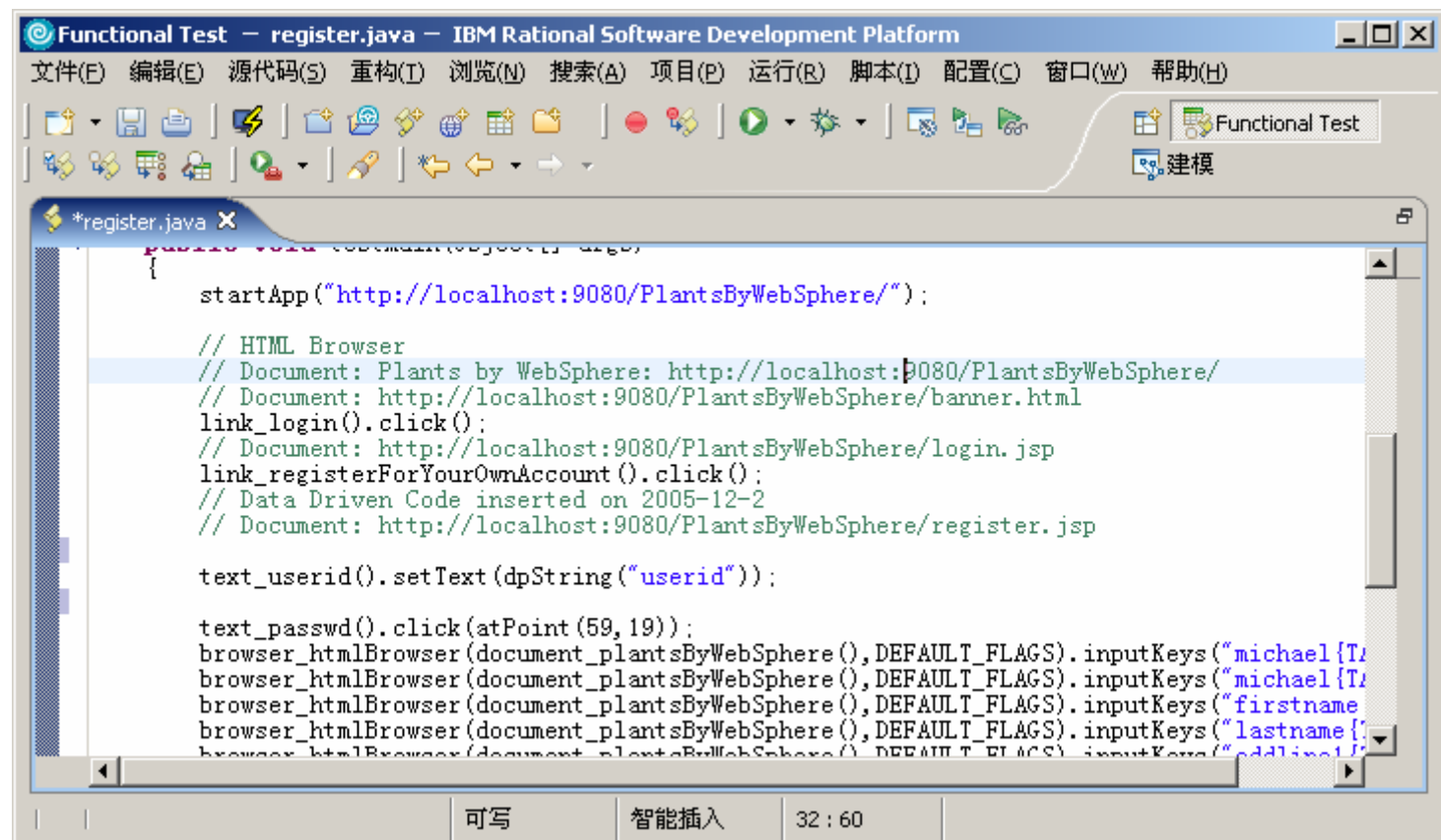
GUI功能测试自动化工具: Rational Functional Tester

- 支持的应用类型:
 - ▶ Visual Studio .NET 2003
 - ▶ Java Application
 - ▶ Browser
 - ▶ Terminal-Based(3270, 5250, VT100)
- 测试脚本语言
 - ▶ Java
 - ▶ VB .NET
- 脚本环境
 - ▶ Eclipse 3.0
 - ▶ Visual Studio .NET



基于Java或VB .NET的测试脚本

- 无须学习特有的测试脚本语法和API
- 充分利用开发环境的强大的编辑和调试功能
- 开发人员也可以用RFT进行自动化测试



The screenshot shows the IBM Rational Software Development Platform interface. The main window displays a Java test script for a registration page. The script includes comments for each step and the corresponding document URL. The script is as follows:

```
startApp("http://localhost:9080/PlantsByWebSphere/");

// HTML Browser
// Document: Plants by WebSphere: http://localhost:9080/PlantsByWebSphere/
// Document: http://localhost:9080/PlantsByWebSphere/banner.html
link_login().click();
// Document: http://localhost:9080/PlantsByWebSphere/login.jsp
link_registerForYourOwnAccount().click();
// Data Driven Code inserted on 2005-12-2
// Document: http://localhost:9080/PlantsByWebSphere/register.jsp

text_userid().setText(dpString("userid"));

text_passwd().click(atPoint(59, 19));
browser_htmlBrowser(document_plantsByWebSphere(), DEFAULT_FLAGS).inputKeys("michael{T
browser_htmlBrowser(document_plantsByWebSphere(), DEFAULT_FLAGS).inputKeys("michael{T
browser_htmlBrowser(document_plantsByWebSphere(), DEFAULT_FLAGS).inputKeys("firstname
browser_htmlBrowser(document_plantsByWebSphere(), DEFAULT_FLAGS).inputKeys("lastname{
browser_htmlBrowser(document_plantsByWebSphere(), DEFAULT_FLAGS).inputKeys("address{
```

无需编程，快速实现数据驱动测试

- 实现代码和测试数据分离，提高脚本可维护性

The screenshot displays the IBM Rational Software Development Platform interface. The main window is titled "Functional Test - data_driven.java - IBM Rational Software Development Platform". The menu bar includes options like "文件(E)", "编辑(E)", "源代码(S)", "重构(I)", "浏览(N)", "搜索(A)", "项目(P)", "运行(R)", "脚本(I)", "配置(C)", "窗口(W)", and "帮助(H)". The toolbar contains various icons for file operations and testing. The code editor shows the following Java code:

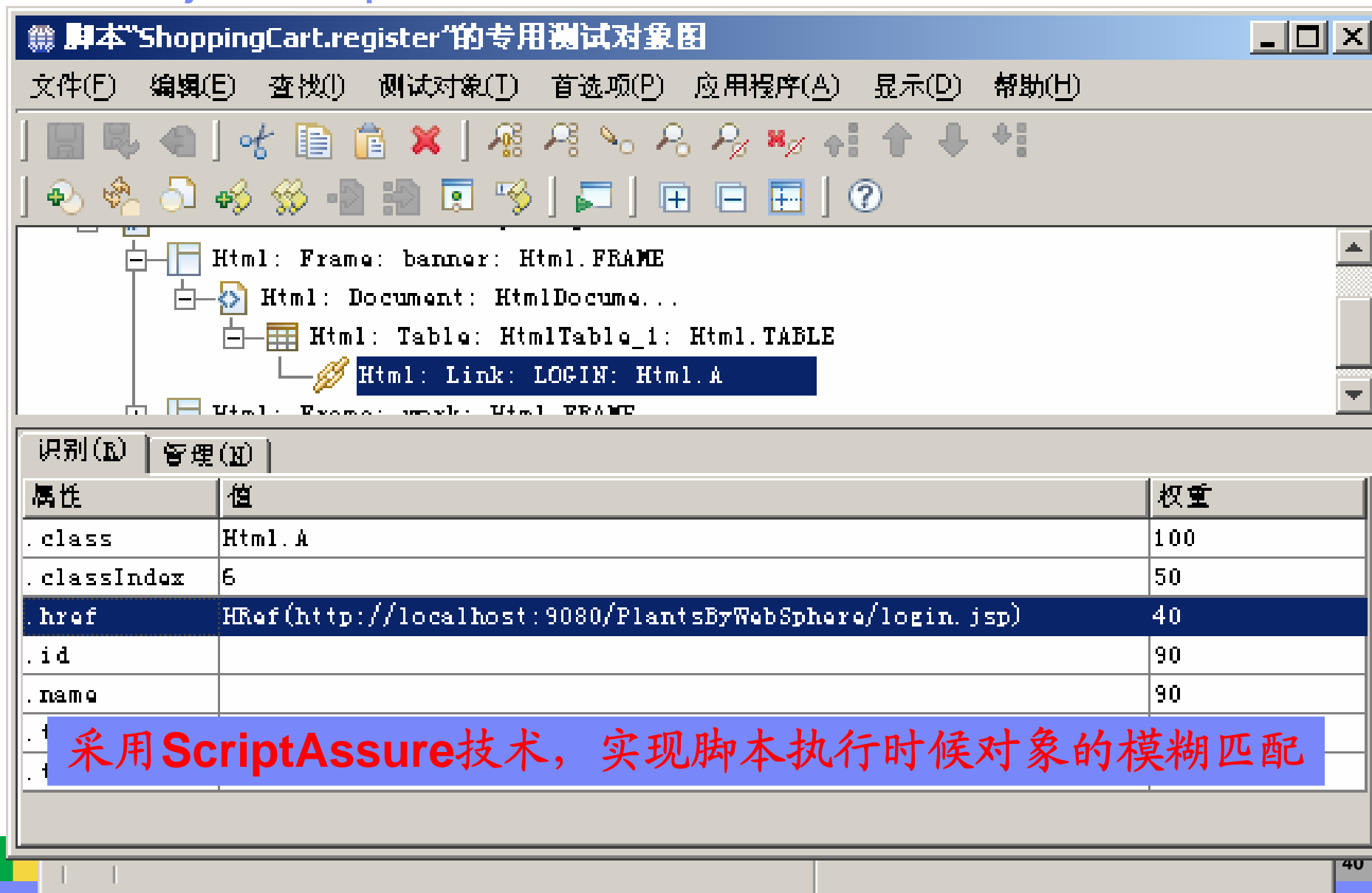
```
// Data Driven Code inserted on 2005-12-2  
  
// HTML Browser  
// Document: Plants by WebSphere: http://localhost:9080/PlantsByWebSphere/  
// Document: http://localhost:9080/PlantsByWebSphere/register.jsp  
text_userid().setText(dpString("userid"));  
}  
}
```

Below the code editor, there is a "测试数据池" (Test Data Pool) section. It includes tabs for "任务" (Tasks) and "问题" (Issues). The "测试数据池" is currently selected, showing a table of test data:

| | userid::java.lang.String | |
|---|--------------------------|--|
| 0 | rft001@cn.ibm.com | |
| 1 | rft002@cn.ibm.com | |
| 2 | rft003@cn.ibm.com | |

The status bar at the bottom indicates "可写" (Writable), "智能插入" (Smart Insert), and "34 : 38".

利用 Object Map 降低脚本维护成本



脚本“ShoppingCart.register”的专用测试对象图

文件(F) 编辑(E) 查找(I) 测试对象(T) 首选项(P) 应用程序(A) 显示(D) 帮助(H)

Html: Frame: banner: Html.FRAME
Html: Document: HtmlDocume...
Html: Table: HtmlTable_1: Html.TABLE
Html: Link: LOGIN: Html.A
Html: Frame: ...: Html.FRAME

| 属性 | 值 | 权重 |
|-------------|---|-----|
| .class | Html.A | 100 |
| .classIndex | 6 | 50 |
| .href | HRef(http://localhost:9080/PlantsByWebSphere/login.jsp) | 40 |
| .id | | 90 |
| .name | | 90 |

采用 ScriptAssure 技术，实现脚本执行时候对象的模糊匹配

40

采用模式匹配快速实现结果验证

The screenshot shows the ClassicsCD.com website with a confirmation message: "Your order has been placed. For future reference, your order ID is 230. Thank you for shopping at ClassicsCD.com!". A blue box highlights the text "Order ID 是动态数据" (Order ID is dynamic data). A diagram with green arrows points from this text to several green boxes containing "###", representing the pattern matching process for the order ID.

- 动态数据模式匹配
 - ▶ 如对“Order ID 230”, 验证点中的模式串为“Order ID ###”或“Order ID 2##”

基于HTML格式的报告

回放"ShoppingCart.comparator"的日志 - Microsoft Internet Explorer

验证点比较器 - 0000.ShoppingCart

文件(F) 编辑(E) 差别(D) 测试对象(T) 首选项(P) 帮助(H)

测试对象(T)

新建: Html: Browser: htmlBro
新建: Html: Document: Pla

识别(R) 管理(N) 元数据(M)

| 属性(P) | 值(V) |
|----------|------------|
| .caption | |
| .class | Html.TABLE |

期望的值 <--> 实际值

| ITEM # | ITEM # |
|--------|--------|
| F0001 | F0001 |

Order Subtotal: \$600.00

Order Subtotal: \$500.00

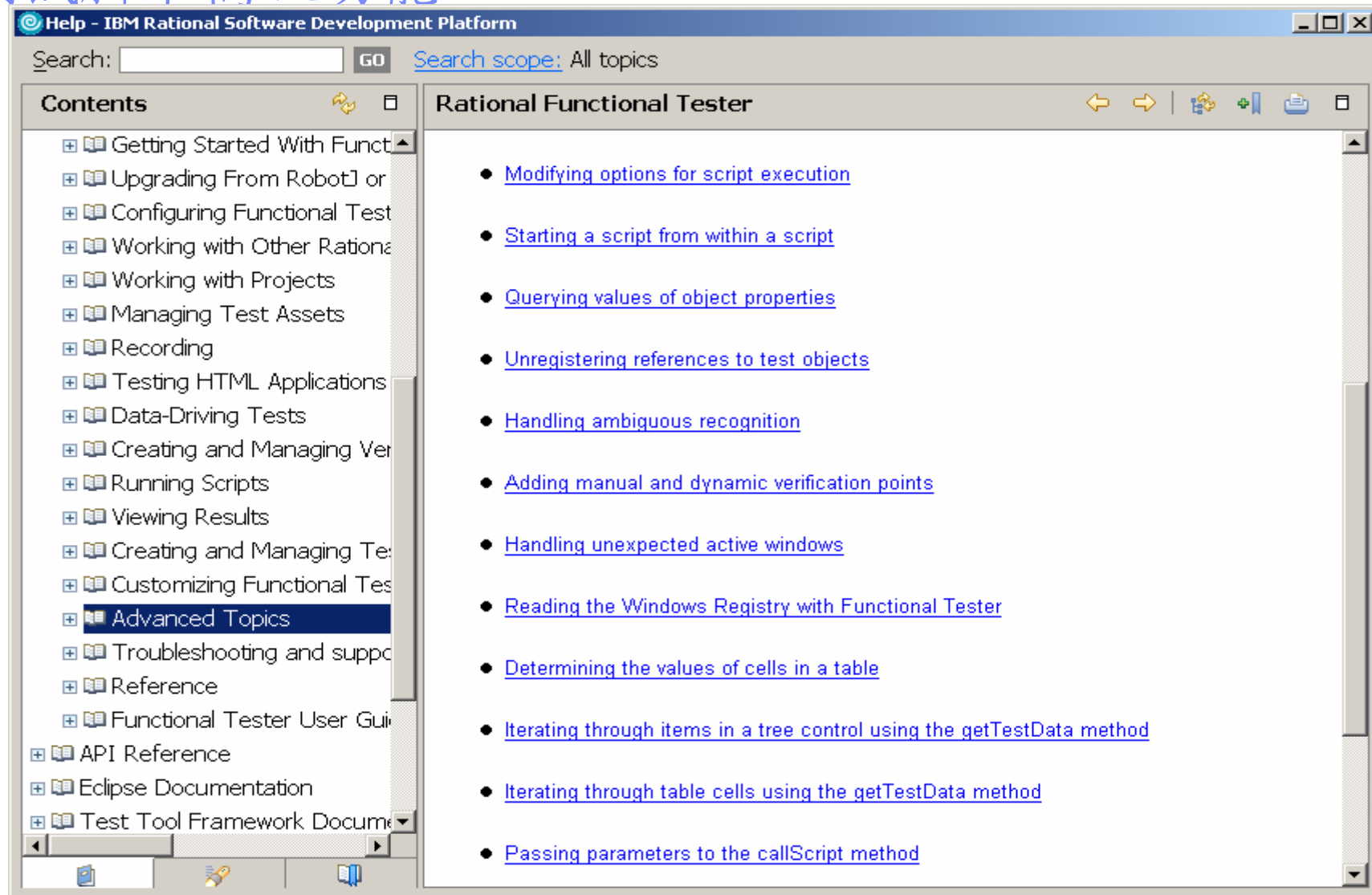
com.rational.test.ft.vp.impl.TestDataTe

小应用程序 ComparatorApplet started

My Computer

42

测试脚本高级功能



Help - IBM Rational Software Development Platform

Search: GO Search scope: All topics

Contents

- Getting Started With Functional Tester
- Upgrading From Robot[?] or Selenium
- Configuring Functional Tester
- Working with Other Rational Tools
- Working with Projects
- Managing Test Assets
- Recording
- Testing HTML Applications
- Data-Driving Tests
- Creating and Managing Verification Points
- Running Scripts
- Viewing Results
- Creating and Managing Test Cases
- Customizing Functional Tester
- Advanced Topics**
- Troubleshooting and support
- Reference
- Functional Tester User Guide
- API Reference
- Eclipse Documentation
- Test Tool Framework Documentation

Rational Functional Tester

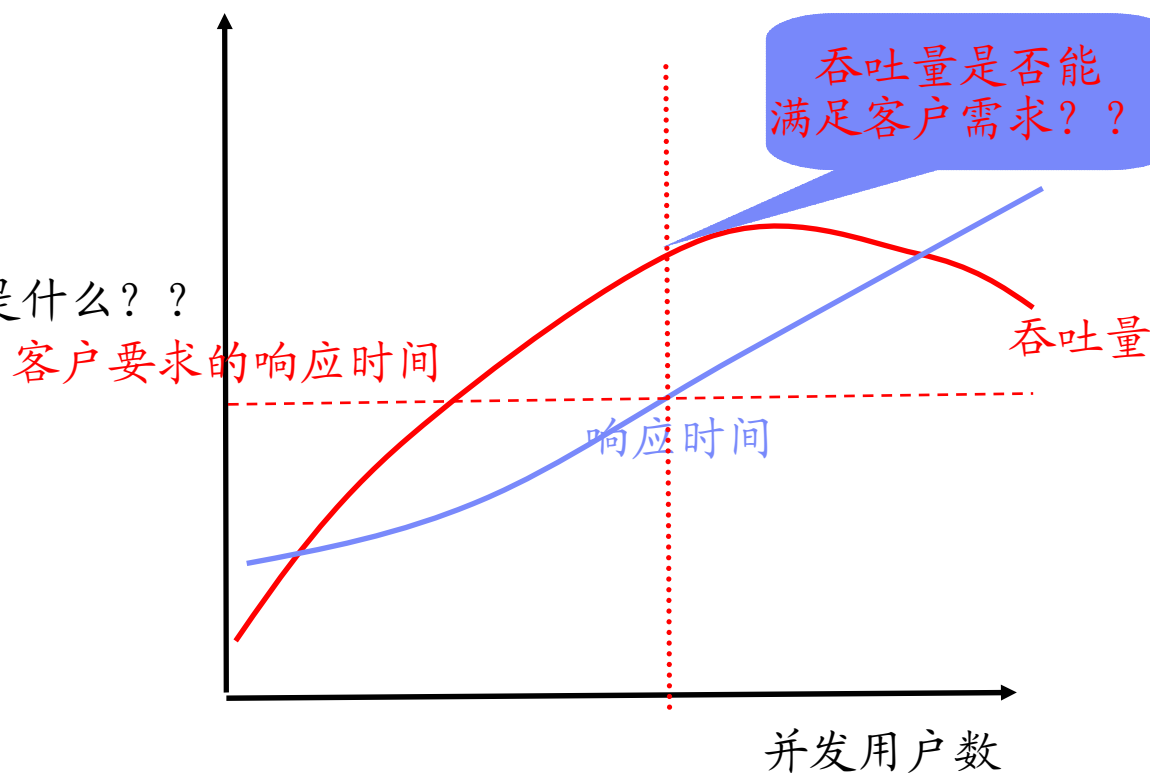
- [Modifying options for script execution](#)
- [Starting a script from within a script](#)
- [Querying values of object properties](#)
- [Unregistering references to test objects](#)
- [Handling ambiguous recognition](#)
- [Adding manual and dynamic verification points](#)
- [Handling unexpected active windows](#)
- [Reading the Windows Registry with Functional Tester](#)
- [Determining the values of cells in a table](#)
- [Iterating through items in a tree control using the `getTestData` method](#)
- [Iterating through table cells using the `getTestData` method](#)
- [Passing parameters to the `callScript` method](#)

议程

- Rational质量解决方案概览
- 代码级测试
 - ▶ 如何保证C/C++传统软件代码质量: Rational PurifyPlus
 - ▶ 如何保证C/C++嵌入式软件代码质量: Rational Test RealTime
 - ▶ 如何保证Java代码质量: Rational Application Developer
- 自动化/性能测试
 - ▶ 功能测试自动化: Rational Functional Tester

基于Rational Performance Tester的性能测试

- 基于Eclipse的性能测试工具
- 支持协议:
 - ▶ HTTP
 - ▶ Siebel
 - ▶ SAP
- Web应用的性能测试策略是什么??
- 基于RPT的性能测试过程
 - ▶ 录制测试
 - ▶ 修改测试
 - ▶ 创建负载模型
 - ▶ 执行测试
 - ▶ 分析结果



录制测试过程

The screenshot displays the IBM Rational Software Development Platform interface for a performance test named "Register".

- Test Contents:** A tree view on the left shows the test structure. The selected element is "register_submit", which includes a "Connection" and a "Response: 200 - OK".
- Test Element Details:** The right pane shows details for "PlantsByWebSphere/register_submit". It includes:
 - Page title: register_submit
 - Primary request: localhost:9080/PlantsByWebSphere/servlet/AccountServlet?action=register
 - Think time: 46006 milliseconds
 - Test Data:** A table listing data points for the request.
- Test Data Table:**

| Name | Value | Substituted with | Owner |
|----------|----------|------------------|------------|
| zip | 100027 | | localho... |
| y | 24 | | localho... |
| x | 42 | | localho... |
| phone | 65391188 | | localho... |
| city | beijing | | localho... |
| state | beijing | | localho... |
| addr1 | line1 | | localho... |
| addr2 | line2 | | localho... |
| fname | michael | | localho... |
| password | password | | localho... |
- Page Title Verification Point:** A checkbox labeled "Enable verification point" is present, with a field for "Expected page title".
- Browser Window:** The bottom pane shows the "Response Content" tab with two items:
 - × Gardens of Summer. They all start with the right colors, and we've got them all.
 - × Image of a colorful garden

Red callout boxes highlight key features:

- 包含录制过程的HTTP请求和响应 (Contains recorded HTTP requests and responses)
- 详细的请求数据 (Detailed request data)
- 收到的响应数据 (Received response data)

录制测试

Performance Test - Register

Test Contents
This section shows the test contents

- Register
 - homepage
 - login_form
 - register_form
 - register_submit
 - localhost:9080/PlantsByWebSphere/servlet/AccountServlet?action=register**
 - Connection
 - Response: 200 - OK
 - Response Size Verification Point

Test Element Details
localhost:9080/PlantsByWebSphere/servlet/AccountServlet?action=register

Request Attributes
Version: 1.1 Method: POST
Host: localhost Port: 9080
URL: /PlantsByWebSphere/servlet/AccountServlet?action=register

Data:
userid=pt0001@cn.ibm.com&passwd=password&vpasswd=password&fname=michael&lname=yang&addr1=line1&addr2=line2&city=beijing&state=beijing&zip=100027&phone=5391188&x=42&y=24

Request Headers

| Header Name | Value |
|-------------|---|
| Accept | image/gif, image/x-bitmap, image/jpeg, image... |
| Referer | http://localhost:9080/PlantsByWebSphere/regi... |

- Primary request for page Delay: 0 milliseconds
Character set: GB2312
Connection localhost:9080 Response 200 - OK

Common properties
 Enable application monitoring

HTTP请求和响应列表

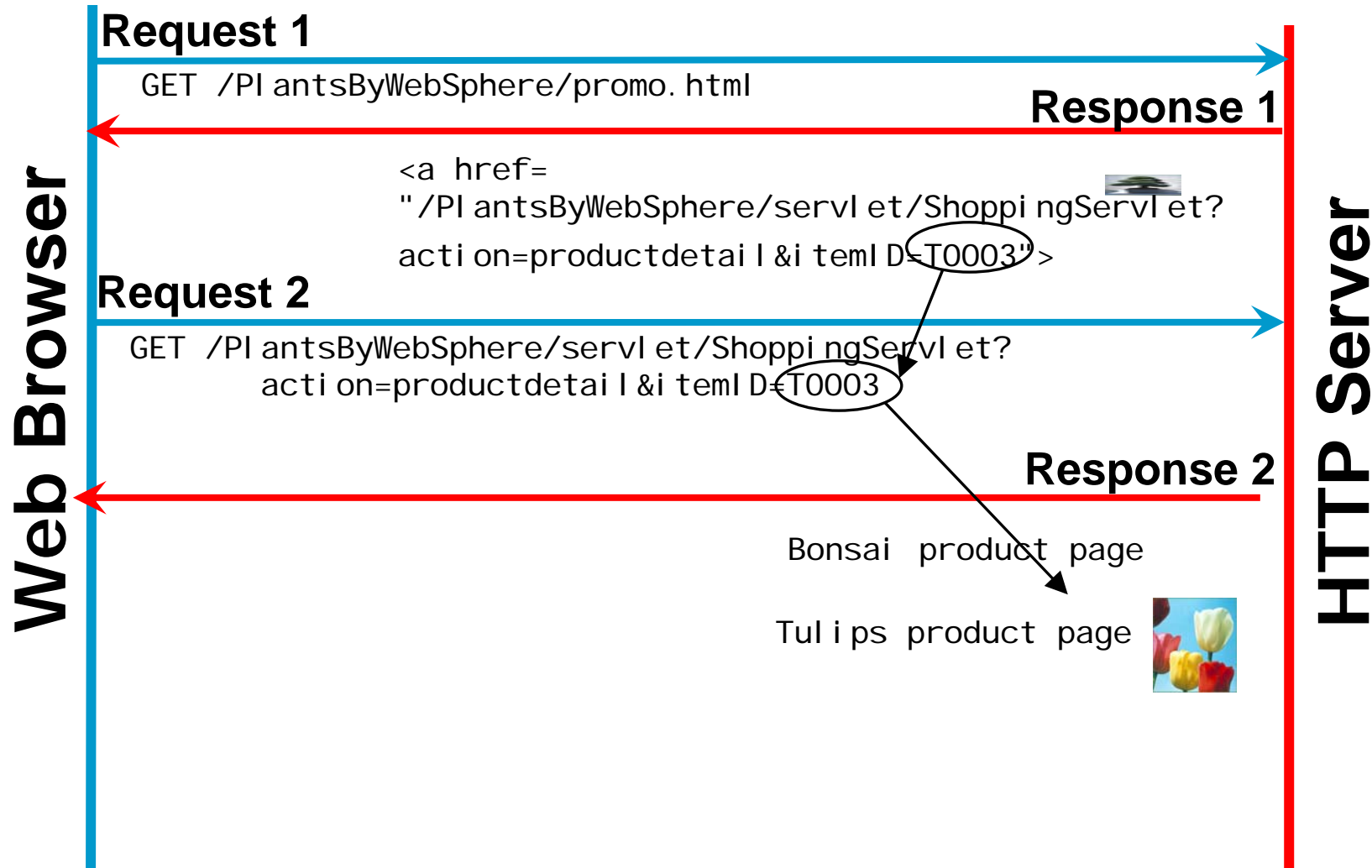
HTTP请求数据

HTTP响应数据

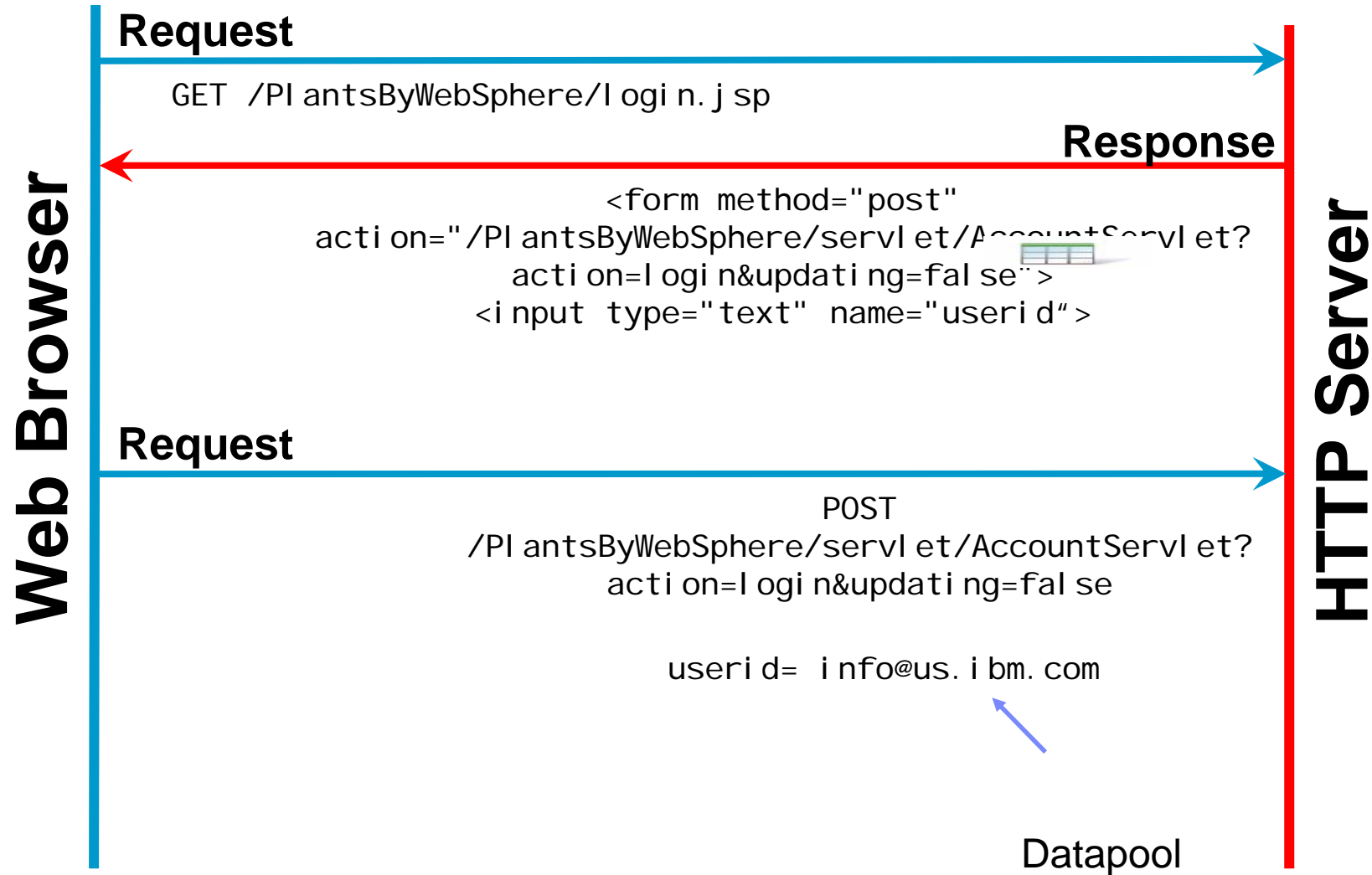
Request: Response Headers: Response Content: Browser

- Gardens of Summer: They all start with the right colors, and we've got them all.
- Image of a colorful garden

修改测试：数据关联



数据参数化



快速设置验证点 (Verification Point)

Performance Test - Register

Test Contents

This section shows the test contents

Test Element Details

localhost:9080/PlantsByWebSphere/servlet/AccountServlet?action=register

Request Attributes

Version: 1.1 Method: POST

Host: localhost Port: 9080

URL: localhost:9080/PlantsByWebSphere/servlet/AccountServlet?action=register

Data:

```
userid=rpt0001@cn.ibm.com&passwd=password&vpasswd=password&fname=
name=yang&addr1=line1&addr2=line2&city=beijing&state=beij
00027&phone=65391188&x=42&y=24
```

引用了变量

使用了Datapool

Datapool备选

- Content VP: 验证返回的内容, 支持模式匹配
- Response Code VP: 验证HTTP返回码
- Response Size VP: 验证返回内容的大小

| Value |
|---|
| image/gif, image/x-bitmap, image/jpeg, image... |
| http://localhost:9080/PlantsByWebSphere/regi... |

Add Edit Remove

Request for page Delay: 0 milliseconds

GB2312 GB2312

localhost:9080 Response 200 - OK

定义负载模型:Schedule

The screenshot shows the 'Performance Schedule - register_sch' configuration window. The left pane shows a tree view with 'register_sch' containing 'User Group 1 (100%)', 'Loop (50 iterations)', and 'Register'. The right pane has three sections: 'Schedule Element Details', 'Think Time', and 'Execution History'. Red callouts point to specific settings: 'Number of users: 5' (并发用户数, 执行时可增加用户数), 'Think Time' (请求延迟时间), and 'Execution history log level: Full' (日志控制).

Schedule Contents
This section shows the schedule contents

- register_sch
 - User Group 1 (100%)
 - Loop (50 iterations)
 - Register

Schedule Element Details
register_sch

Number of users: 5

Add a delay between starting each user

Delay: 0 milliseconds

Stop running the schedule after an elapsed time

Stop after: 10 minutes

Think Time
Modify the duration of think time delays.

Use the recorded think time

Limit think times to a maximum value

Maximum think time: 0 seconds

Execution History

Execution history log level: Full

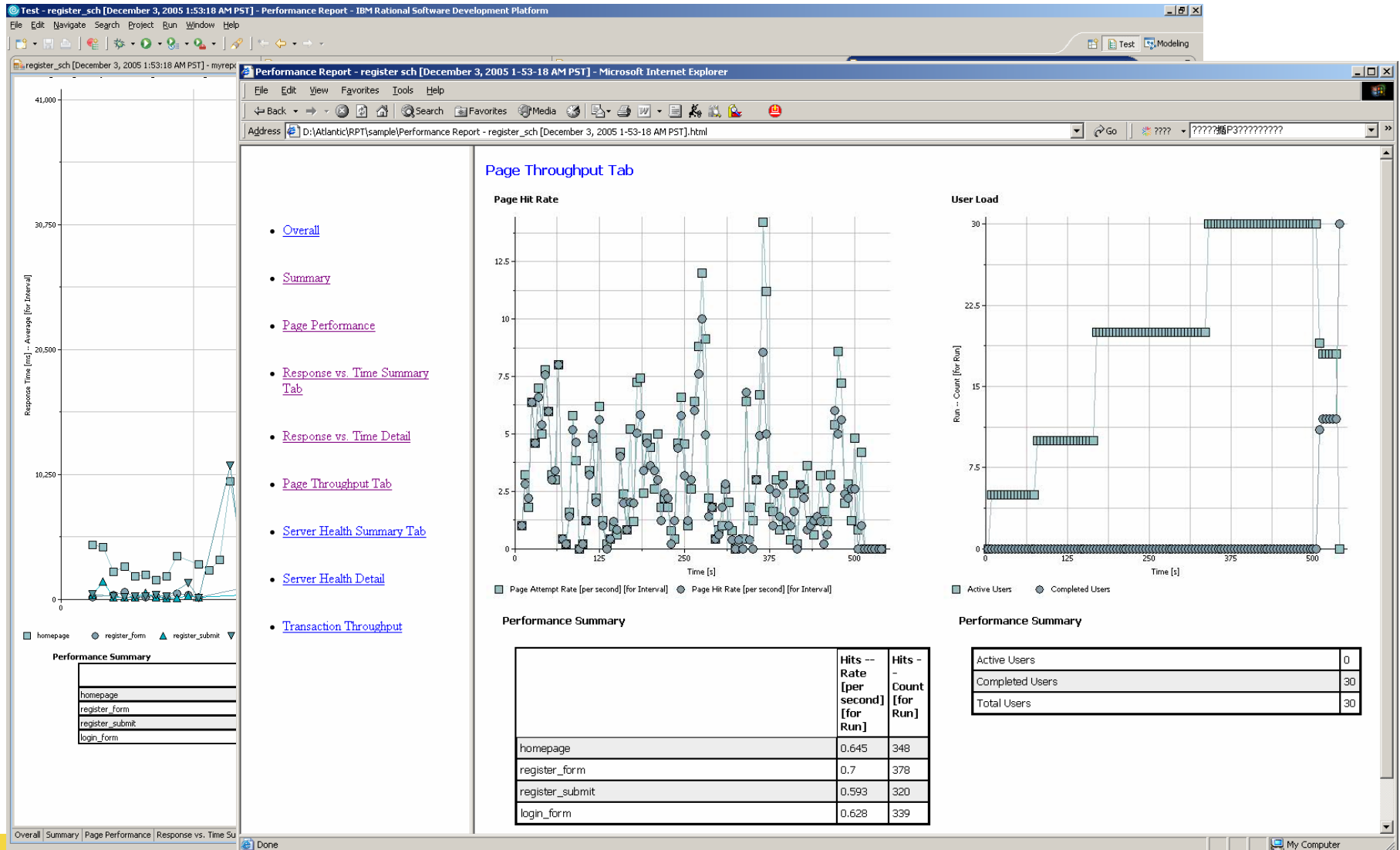
Only sample information from a subset of users

并发用户数, 执行时可增加用户数

请求延迟时间

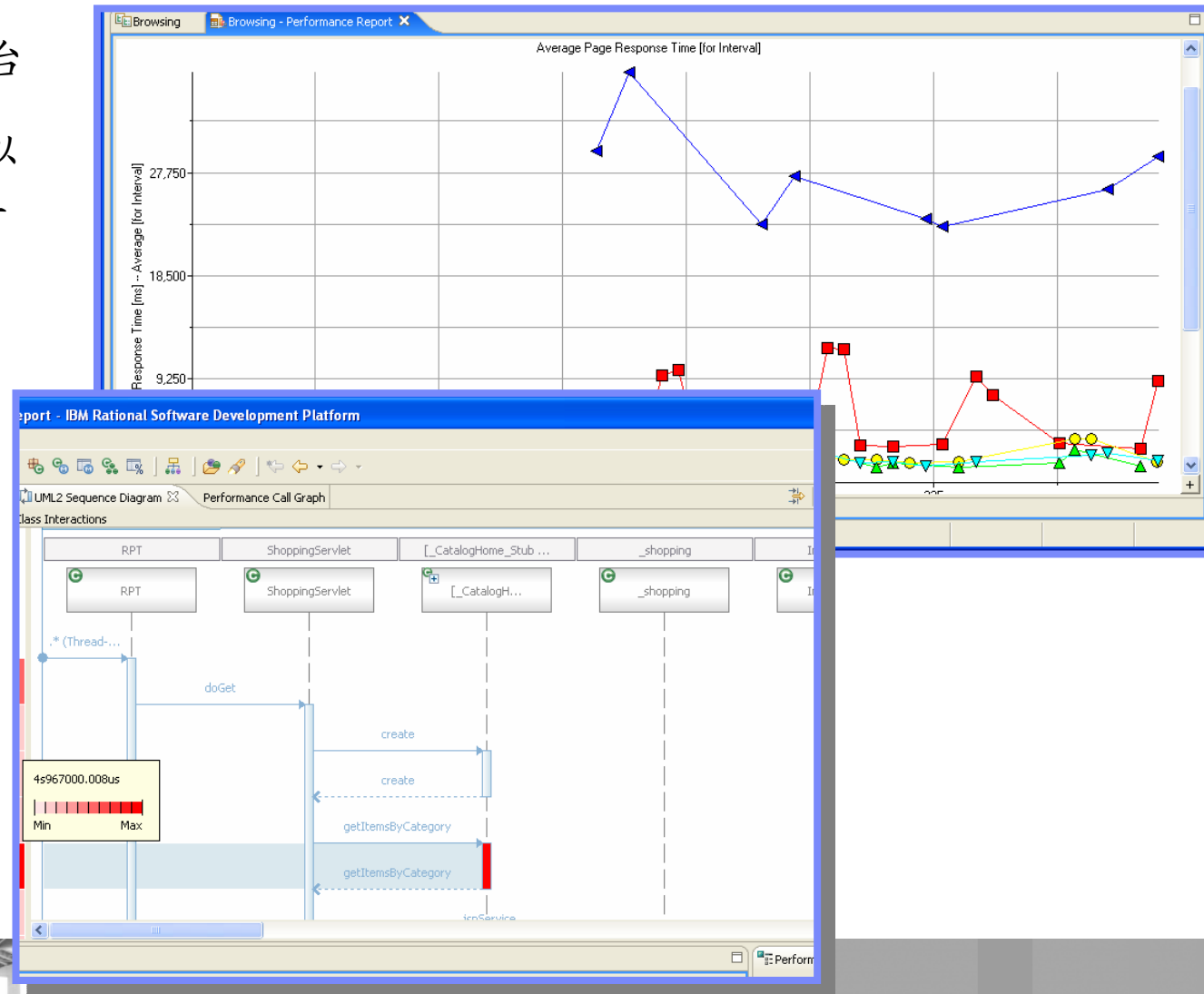
日志控制

执行中和执行后的性能测报告



IBM Performance Optimization Toolkit: 帮助发现应用的性能瓶颈

- 支持多种操作系统平台
- 支持WebSphere 5.x以上和WebLogic 7.x以上



IBM Rational软件测试工具

功能

- 运行时分析（内存、性能、代码覆盖）
- 代码自动评审、组件测试、系统功能测试和性能测试
- 覆盖系统测试周期的测试管理和分析



收益

- 确保软件的功能、性能和可靠性
- 加快测试周期
- 适合不同技术水平的测试人员
- 全员质量观

| 产品 | 业务人员 | 测试人员 | 开发人员 |
|------------------------------------|------|------|------|
| IBM Rational Manual Tester | ✓ | ✓ | ✓ |
| IBM Rational Functional Tester | | ✓ | ✓ |
| IBM Rational Performance Tester | | ✓ | ✓ |
| IBM Rational Robot | | ✓ | |
| IBM Rational PurifyPlus | | ✓ | ✓ |
| IBM Rational Test RealTime | | | ✓ |
| IBM Rational Application Developer | | ✓ | ✓ |

Questions

THANK YOU