IBM Rational Test RealTime 提升開發人員測試品質

級別： 初級
IBM,

軟體專案越來越複雜，由於在開發人員對模組測試不充分，導致在整合測試和系統測試階段耗費大量的時間和人力，甚至導致專案進度的重大延誤。因此，爲了保證專案品質和進度的可預見性，就要求開發團隊對自己開發的程式碼進行充分測試。但在不借助工具的情況下，開發人員對程式碼進行完善的測試需要花費50%左右的時間，而開發人員的主要職責是開發程式碼，在面對進度壓力時，開發人員進行的測試往往是留於形式，不能得以切實執行，留下了大量的品質隱患。IBM Rational Test RealTime 幫助開發人員建立測試腳本、執行測試用例和生成測試報告，並提供對被測程式碼進行靜態分析和運行時分析功能。利用該工具，開發人員可以大大提高測試的效率。本文通過舉例介紹如何利用 IBM Rational Test RealTime 進行開發人員測試的過程。

1. 引言

軟體專案越來越複雜，由於在開發人員對模組測試不充分，導致在整合測試和系統測試階段耗費大量的時間和人力，甚至導致專案進度的重大延誤。因此，爲了保證專案品質和進度的可預見性，就要求開發團隊對自己開發的程式碼進行充分測試。但在不借助工具的情況下，開發人員對程式碼進行完善的測試需要花費50%左右的時間，而開發人員的主要職責是開發程式碼，在面對進度壓力時，開發人員進行的測試往往是留於形式，不能得以切實執行，留下了大量的品質隱患。IBM Rational Test RealTime 幫助開發人員建立測試腳本、執行測試用例和生成測試報告，並提供對被測程式碼進行靜態分析和運行時分析功能。利用該工具，開發人員可以大大提高測試的效率。本文通過舉例介紹如何利用 IBM Rational Test RealTime 進行開發人員測試的過程。

2. IBM Rational Test RealTime 概述

Test RealTime 是 IBM Rational 提供的程式碼級測試工具。該工具包含如下特點：

1. 程式碼靜態分析，功能測試和運行時分析相整合。
2. 程式碼編輯、測試和調試相整合。
3. Test RealTime 通過分析源程式碼，自動生成測試驅動（Test Driver）和樁（Test Stub）模版。開發人員只需要在該測試腳本的基礎上指定測試輸入資料、期望輸出資料以及打樁函數的邏輯。

4 測試執行後自動生成測試報告和各種運行時候報告。測試報告展示通過或失敗的測試用例，而運行時分析報告包括程式碼覆蓋分析報告，記憶體分析報告、性能分析報告和執行追蹤報告。

5.通過 Target Deployment Port 技術同時支援開發機和目的機的測試。

3. 開發人員測試現狀分析

假設在 c:\rtrt\src 目錄下具有 UmtsCode.c 和 UmtsCode.h（通過 winzip 在 c:\ 目錄下展開 rtrt.zip 文件）。其中 UmtsCode.c 中包含了 code_int(int x, char *buffer) 函數的實現，該函數的設計規範如下：

1、 完成對整數 x 的編碼，並把編碼的輸出值返回到 buffer 中。

2、 編碼規則爲：

輸入值	輸出值
x=2, buffer = “”	Buffer “I12”, /*其中 I 表示整數編碼，1 爲整數串的長度，2 表示整數串*/
x=34, buffer = “”	Buffer “I243”, /*其中 I 表示整數編碼，2 爲整數串的長度，43 表示整數串，對進行倒序編碼*/
x=56, buffer = “I243”	Buffer “I243I265”

對 code_int(int x, char *buffer)進行測試的傳統過程：

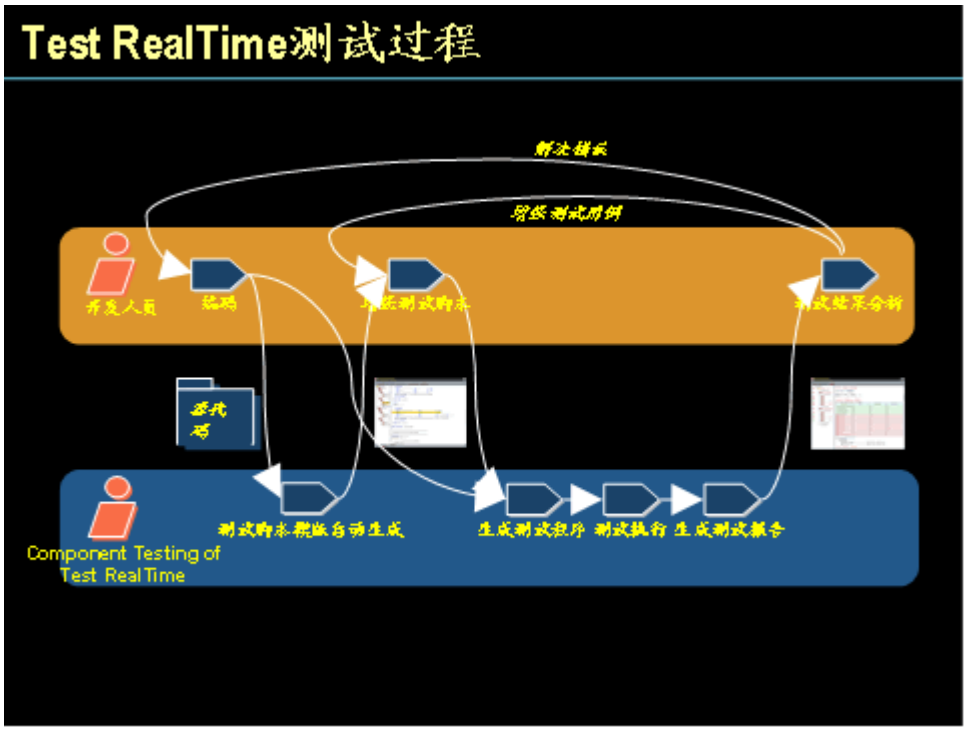
1. 利用 C 語言編寫測試驅動程式 test_code_int.c，該程式碼包含 main 函數，並 main 函數利用輸入值調用 code_int，然後檢查 code_int 的返回值和期望值是否匹配來判斷測試用例是否通過。
2. 分別編譯 code_int.c 和 test_code_int.c，然後連接執行 test_code_int.exe。
3. 根據 test_code_int.exe 的執行輸出，來整理測試報告。

該過程具有如下問題：

1. 利用 C 語言來編寫測試程式，編碼工作量大，而且易於出錯。測試人員的工作重心不是關注測試用例的設計，而是關注如何實現測試用例。
2. 不能對測試程式（test_code_int.c）進行有效的管理，測試執行不方便。
3. 包含測試用例成功與失敗的測試報告不能自動化生成，需要手工編寫。
4. 不能自動得到程式碼的覆蓋情況，測試完備性以及被測試單元的可靠性不能得到保證。

4. 利用 Test RealTime 對 code_int(int x, char *buffer)函數進行測試

4.1 Test RealTime 的開發人員測試過程



上圖是利用 Rational Test RealTime 的開發人員測試過程，步驟如下：

1、 編碼：開發人員在 Test RealTime 提供的 C/C++語言編輯器中進行程式碼編寫。

2、 測試腳本模版自動生成：在被測源程式碼編譯通過後，Test RealTime 將通過對源程式碼進行分析，形成測試腳本範本。

3、 增強測試腳本：開發人員根據設計的測試用例，在測試腳本範本的基礎上增加和修改測試用例。

4、 生成測試程式：Test RealTime 將根據測試腳本生成 C 語言測試程式。

5、 執行測試：Test Realtime 編譯測試程式、被測程式、連接並執行可執行程式。

6、 生成測試報告：Test RealTime 將根據測試執行產生的日誌檔生成測試報告。

7、 測試結果分析：開發人員根據測試報告判斷被測程式品質或測試完備性。

8、 解決錯誤：如果發現測試用例未通過，來定位錯誤位置，並修改錯誤。Test RealTime 可以和開發環境的調試器（如 Visual C 6.0 的 msdev.exe）整合，提高錯誤定位速度。

9、 增強測試用例：增強測試用例來覆蓋前次測試執行沒覆蓋的程式碼分支。

4.2 安裝配置測試環境

4.2.1 建立相關目錄

由於在編碼和單元測試階段中引入 Test RealTime，因此需要對新增加的檔類型，如測試腳本檔、測試報告檔進行有序的管理。建議參考如下目錄結構：

- scr：被測源程式碼，包括.c 文件和.h 文件

- scripts: 存儲測試腳本
- reports: 存儲 Test RealTime 格式的測試報告
- html: 存儲 HTML 格式的測試報告

4.2.2 安裝配置 Microsoft Visual C++ 6.0

安裝 Microsoft Visual C++ 6.0 後，需要配置 PATH 環境變數，從而保證在命令行下能執行 VC 的相關命令。可以通過在命令行下執行 cl.exe 命令進行驗證。

4.2.3 安裝配置 IBM Rational Test RealTime

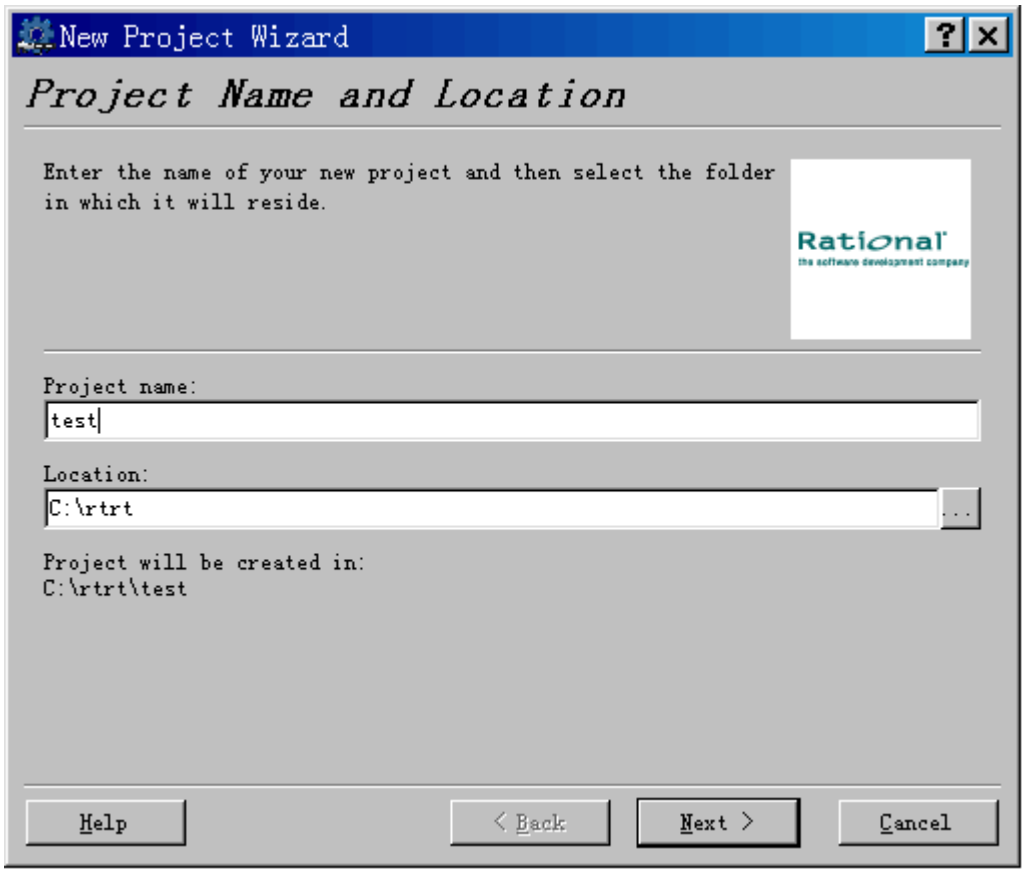
Test RealTime 的安裝軟體需要聯繫 IBM 當地的銷售代表獲得。詳細安裝步驟參見《Rational? Test RealTime Installation Guide》文檔。

安裝完成後，需設置環境變數 ATTOLSTUDIO_VERBOSE=1，這樣 Test RealTime 將顯示詳細的 build 資訊。

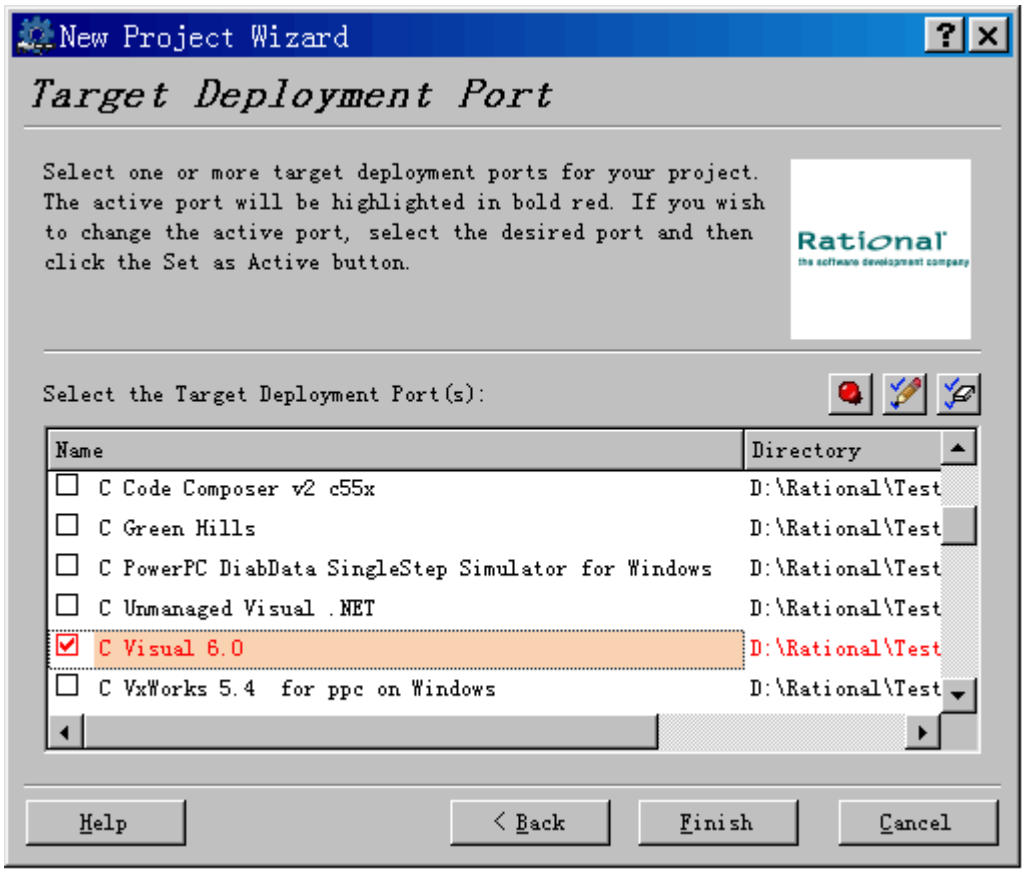
4.2.4 建立 Test RealTime Project

一個 Test RealTime Project 類似於 Visual C++ 6.0 的 Project，包含了被測試程式碼，測試腳本等相關資訊以及 C/C++語言編譯、連接等選項。

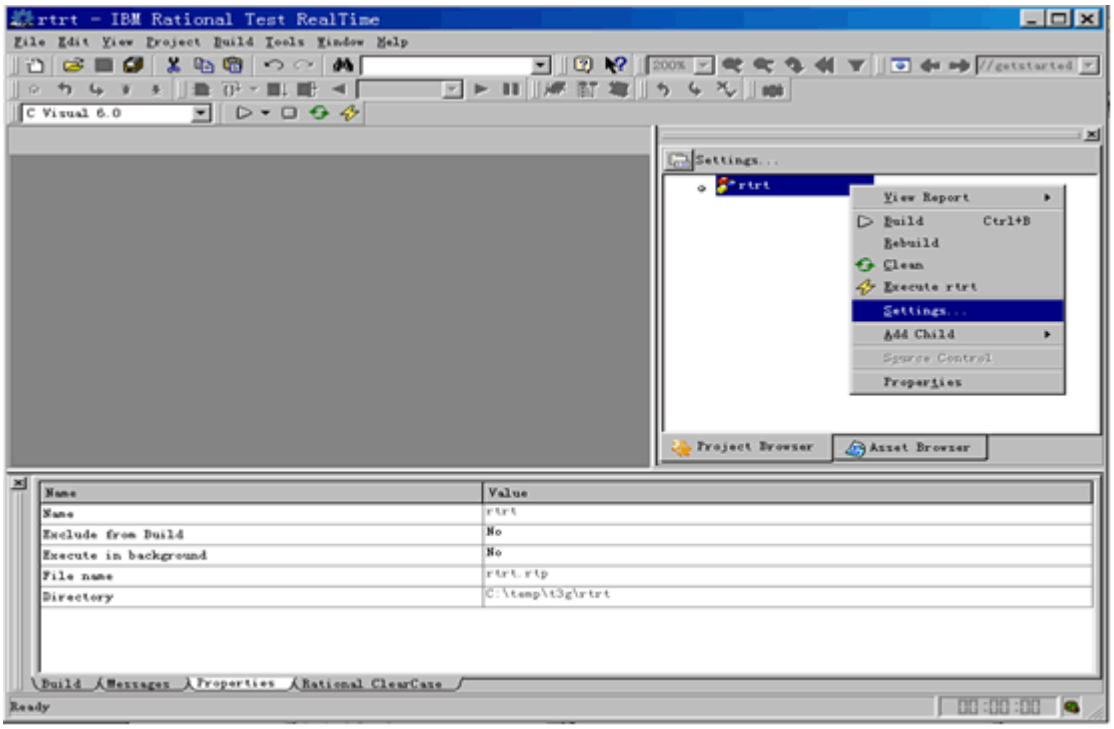
通過 File > New > Project…功能表進入如下 Project 建立介面：



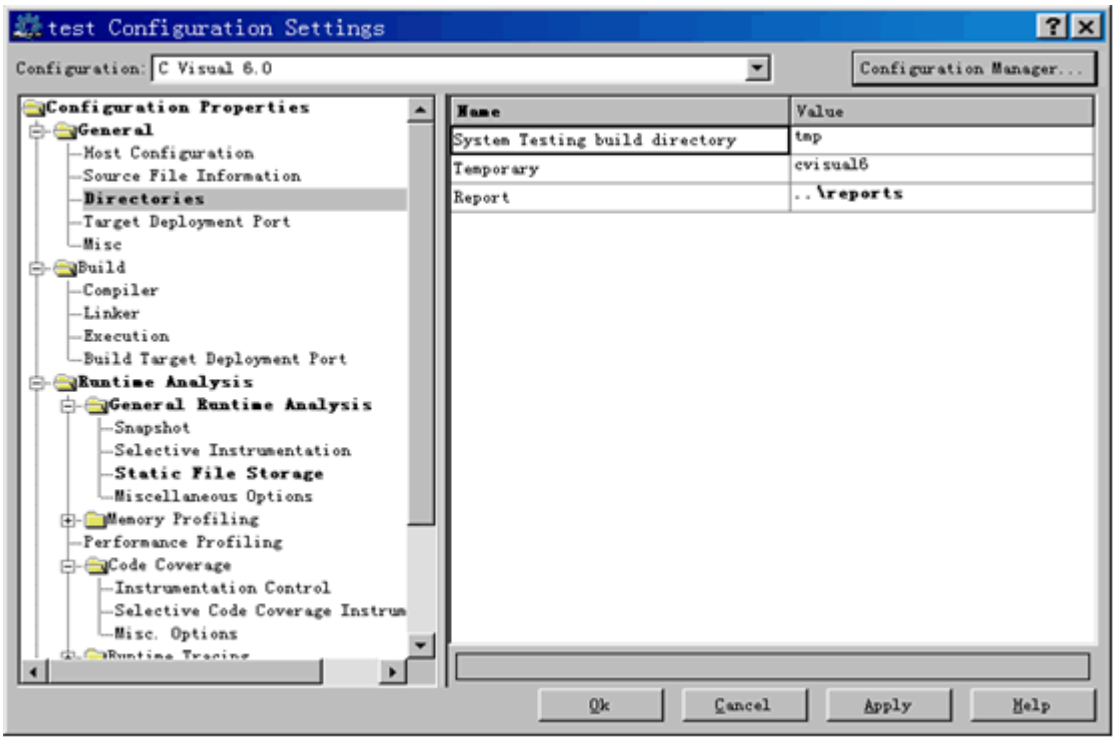
指定專案的名字和所處位置。Test RealTime 將自動在項目所處位置下以專案名建立一個目錄，本例爲 c:\rtrt\test，而且該目錄也是專案的當前目錄。選擇“Next >”進入如下介面：

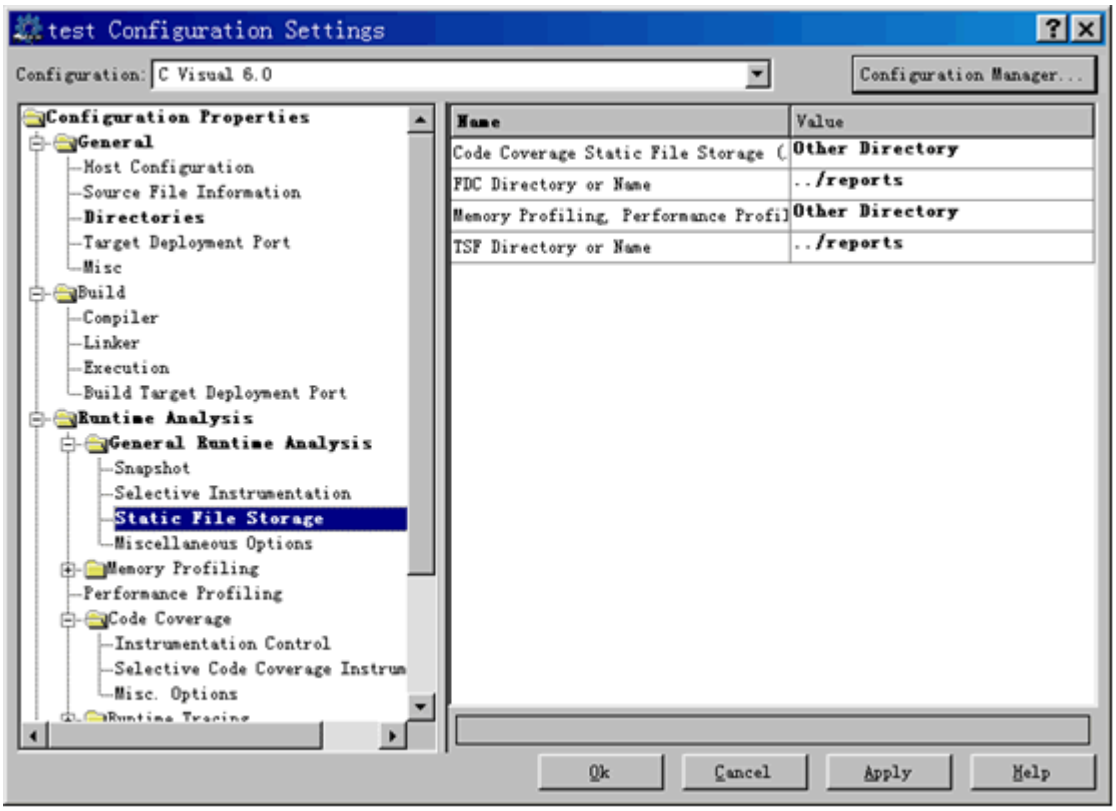


不同的開發環境對應不同的 Target Deployment Port，由於被測程式碼 UtmsCode.c 是 C 語言，因此選擇 C Visual 6.0。選擇 Finish 將建立一個 Project. 和 VC 類似，需要設置相關專案級的相關參數。通過如下介面進入 Configuration Settings 介面。



就本例而言，不需修改缺省的 C 語言編譯、連接等選項，而只需要通過如下兩個介面設置 report 檔所處的目錄爲 c:\rtrt\report（由於當前目錄爲 c:\rtrt\test，因此目錄值爲..\reports）。



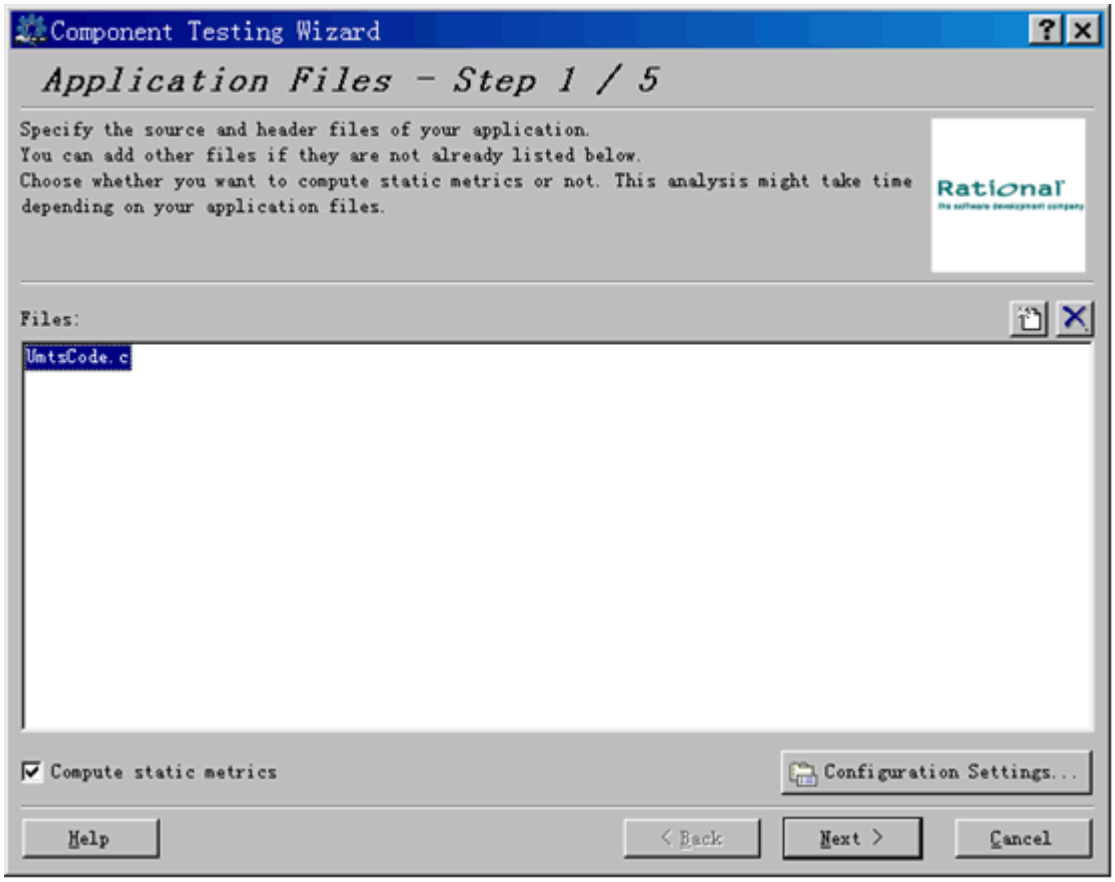


4.3 對函數 code_int(int x, char *buffer) 進行測試

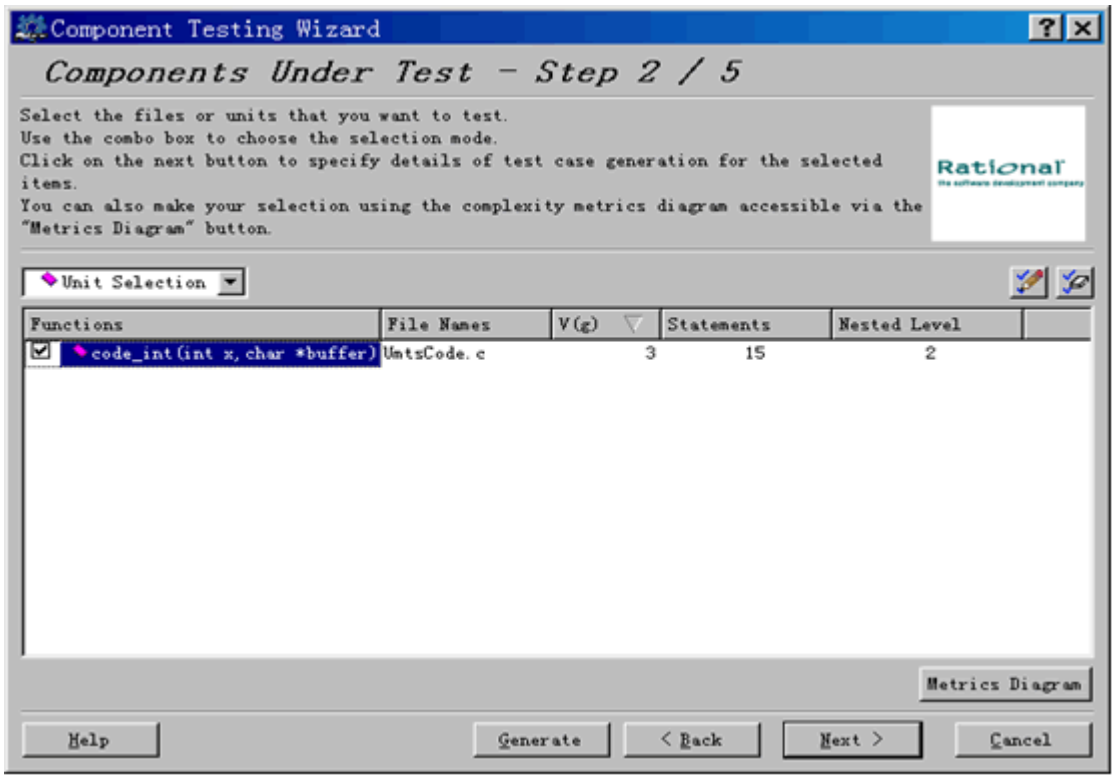
下面以 code_int 的測試爲例詳細介紹如何利用 Test RealTime 進行測試的過程。

4.3.1 根據源程式碼自動生成測試腳本模版

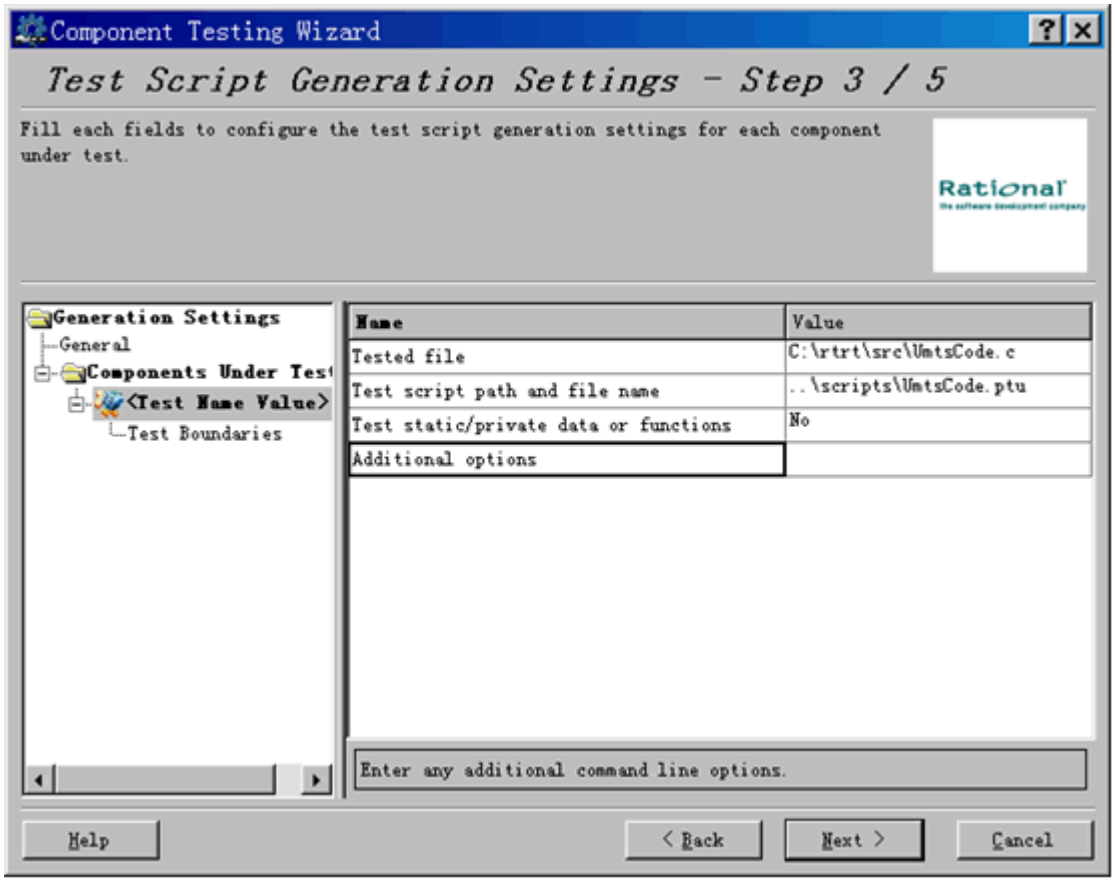
選擇 File > New > New Activity > Component Testing 功能表，進入 Component Testing Wizard 介面，通過 按鈕增加被測檔“UtmsCode.c”，並選中“compute static metrics”對被測程式碼進行靜態分析。如下圖顯示：



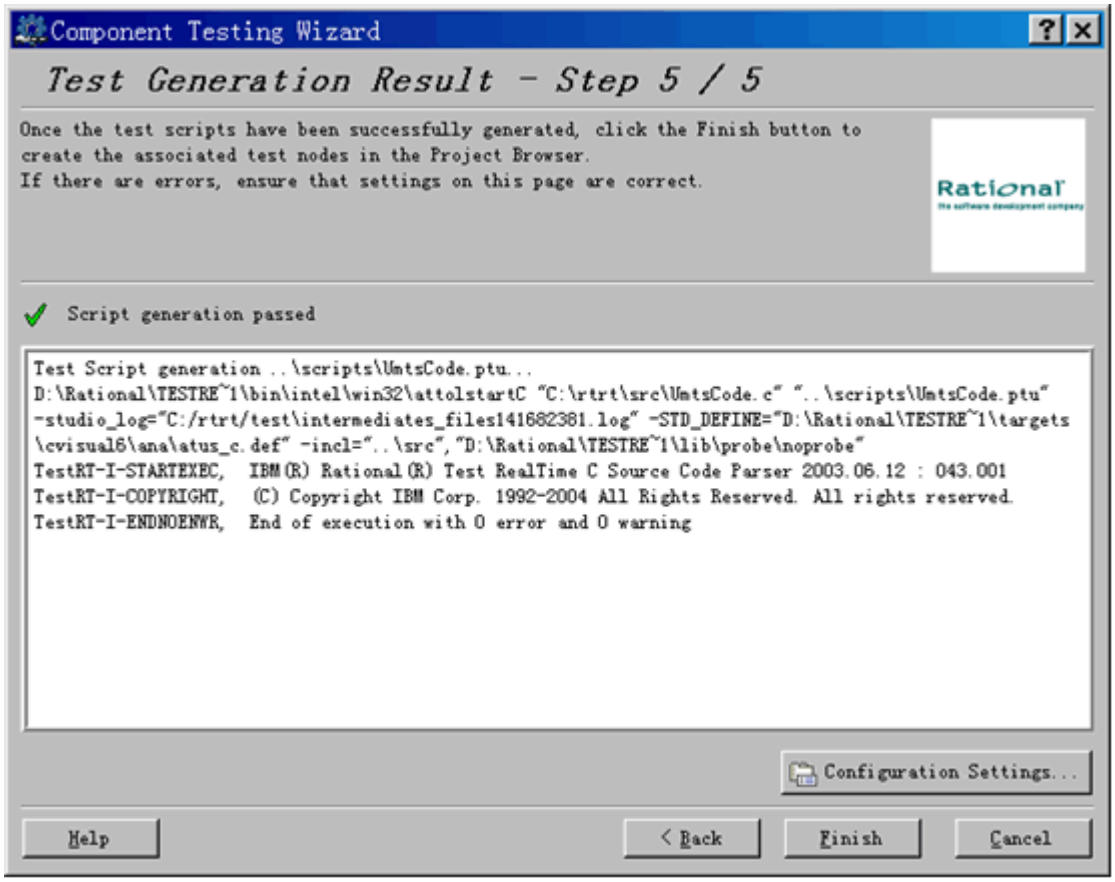
選擇“Next >”按鈕進入如下“Component Under Test”介面選擇被測函數：



對於 code_int 函數，v(g)表示與測試難度相關的函數複雜度度量。如 v(g)=1，表示該函數沒有分支。爲了控制軟體的可測試性，建議一個函數的複雜度不超過 10。關於 v(g)的詳細解釋，參見 Test RealTime 幫助。爲了對 code_int 進行測試，選中 code_int 旁邊的 checkbox，點擊“Next >”進入“Test Script Generation Settings”介面。爲了讓生成的測試腳本位於 scripts 目錄，如下圖修改“Test script path and file name”參數值爲“..\scripts\UmtsCode.ptu”。



選擇“Next >”按鈕,然後“Finish”按鈕，進入如下介面：



上述過程實際是通過圖形化介面設置命令 attolstartC 的相關參數。attolstartC 通過分析指定的 C 程式碼，形成測試腳本模版，詳細資訊參考 Test RealTime reference manual.

4.3.2 基於測試腳本模版，根據函數的設計規範，編寫測試用例

Test RealTime 生成的測試腳本範本中包含一個測試用例，該測試用例的相關輸入、輸出值設置爲 0 或“”。

```
SERVICE code_int
SERVICE_TYPE extern
   -- Tested service parameters declarations
   #int x;
   #char  buffer[200];
   ENVIRONMENT ENV_code_int
   VAR x,          init = 0,          ev = init
   VAR buffer,           init = "",         ev = init
   END ENVIRONMENT -- ENV_code_int
   USE ENV_code_int
   TEST 1
   FAMILY nominal
```

```
      ELEMENT
      #code_int(x, buffer);
      END ELEMENT
   END TEST -- TEST 1
END SERVICE -- code_int
```

其中VAR x, init = 0, ev = init語句表示x的初始值爲0，期望值等於初始值， VAR buffer, init = "", ev = init 表示 buffer 的初始值爲“”，期望值也等於初始值。

根據前面 code_int 函數的設計規範，形成如下三個測試用例如下：

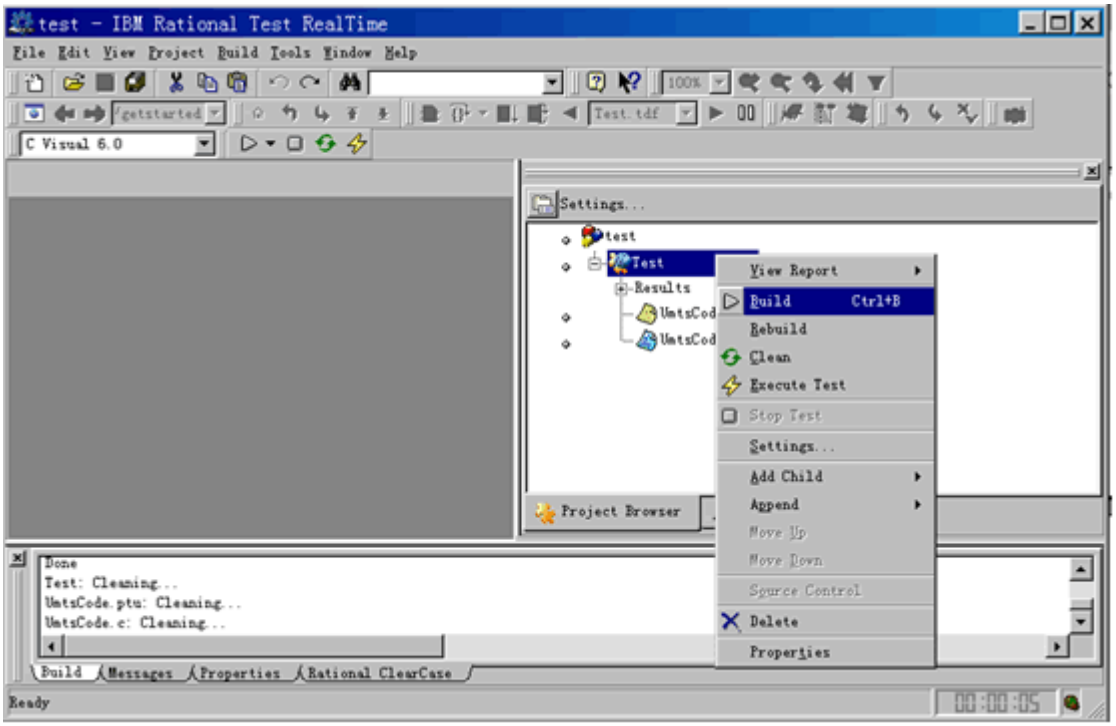
```
SERVICE code_int
SERVICE_TYPE extern
   -- Tested service parameters declarations
   #int x;
   #char  buffer[200];
   ENVIRONMENT ENV_code_int
   VAR x,          init = 0,          ev = init
   VAR buffer,           init = "",         ev = init
   END ENVIRONMENT -- ENV_code_int
   USE ENV_code_int
   TEST 1
   FAMILY nominal
      ELEMENT
            VAR x,       init = 2,          ev = init
            VAR buffer,        init = "",         ev = "I12"
      #code_int(x, buffer);
      END ELEMENT
   END TEST -- TEST 1
   TEST 2
   FAMILY nominal
      ELEMENT
            VAR x,       init = 34,         ev = init
            VAR buffer,        init = "",         ev = "I243"
      #code_int(x, buffer);
      END ELEMENT
   END TEST -- TEST 2
   TEST 3
   FAMILY nominal
```

```
      ELEMENT
            VAR x,       init = 56,          ev = init
            VAR buffer,        init = "I243",           ev = "I243I265"
      #code_int(x, buffer);
      END ELEMENT
   END TEST -- TEST 3
END SERVICE -- code_int
```

完整的測試腳本檔參見 c:\rtrt\scripts\ UtmsCode_new1.ptu。

4.3.3 執行測試

在修改 UtmsCode.put 測試腳本後，按如下圖選擇“Build”執行測試：



在 Build 過程中，將在“Output Window”中顯示 build 的詳細步驟：

1． 執行 attolpreproC 命令把把測試腳本 UtmsCode.put 編譯成 TTest.c：attolpreproC "C:\rtrt\scripts\UmtsCode.ptu" "cvisual6\TTest.c"

2． 對 TTest.c 進行預處理、編譯形成 TTest.obj。

3． 對 UtmsCode.c 進行預處理形成 UtmsCode.i：cl.exe -P "C:\rtrt\src\UmtsCode.c" "-I..\src

4． attolcc1 對 UmtsCode.i 進行插針形成 UmtsCode_aug.c：\attolcc1 "cvisual6\UmtsCode.i" "cvisual6\UmtsCode_aug.c" atct.def…

5． cl.exe 編譯 UmtsCode_aug.c 形成 UmtsCode.obj：cl.exe -ZI -Yd -GZ -GX -c "cvisual6\UmtsCode_aug.c" -Fo"cvisual6\UmtsCode.obj" "-I..\src"

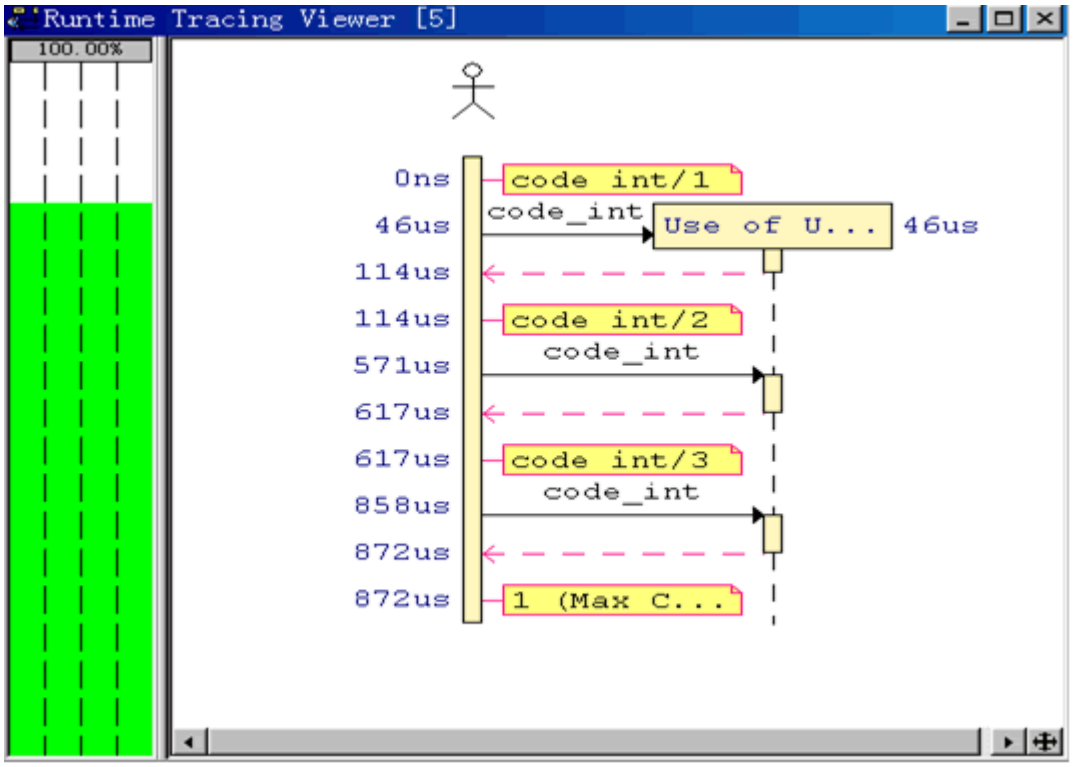
6． 計算被測程式碼 UmtsCode.c 的 Metric: attolstartC "C:\rtrt\src\UmtsCode.c" … -METRICS="../reports"。

7． 連接形成 Test.exe：link.exe /debug /subsystem:console /machine:I386 /pdb:none "C:\rtrt\test\cvisual6\TTest.obj" "C:\rtrt\test\cvisual6\UmtsCode.obj" "cvisual6\TP.obj" ws2_32.lib /out:".\cvisual6\Test.exe"。其中 TP.obj 是 Test RealTime 提供的庫檔。

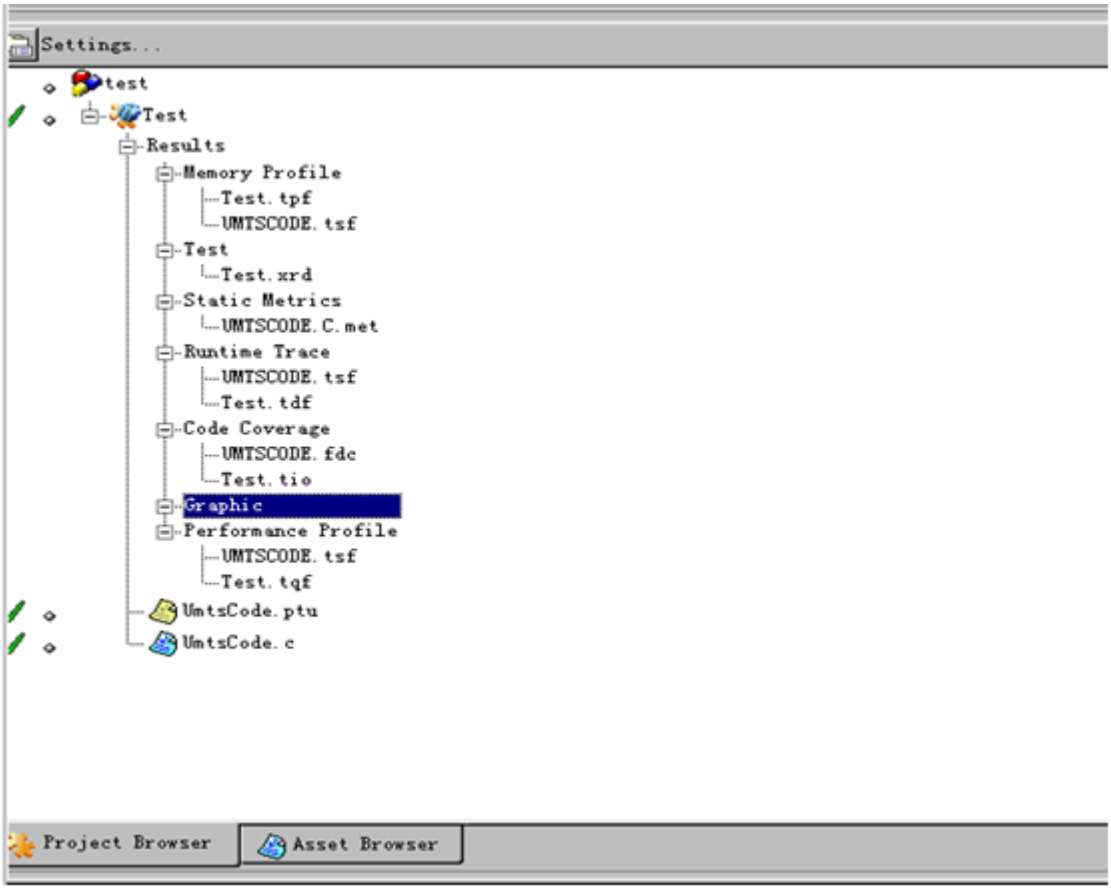
8． 執行 Test.exe。

9． 形成測試報告。

在執行過程中，Test RealTime 將以 UML Sequence Diagram 的形式顯示被測程式的調用關係。

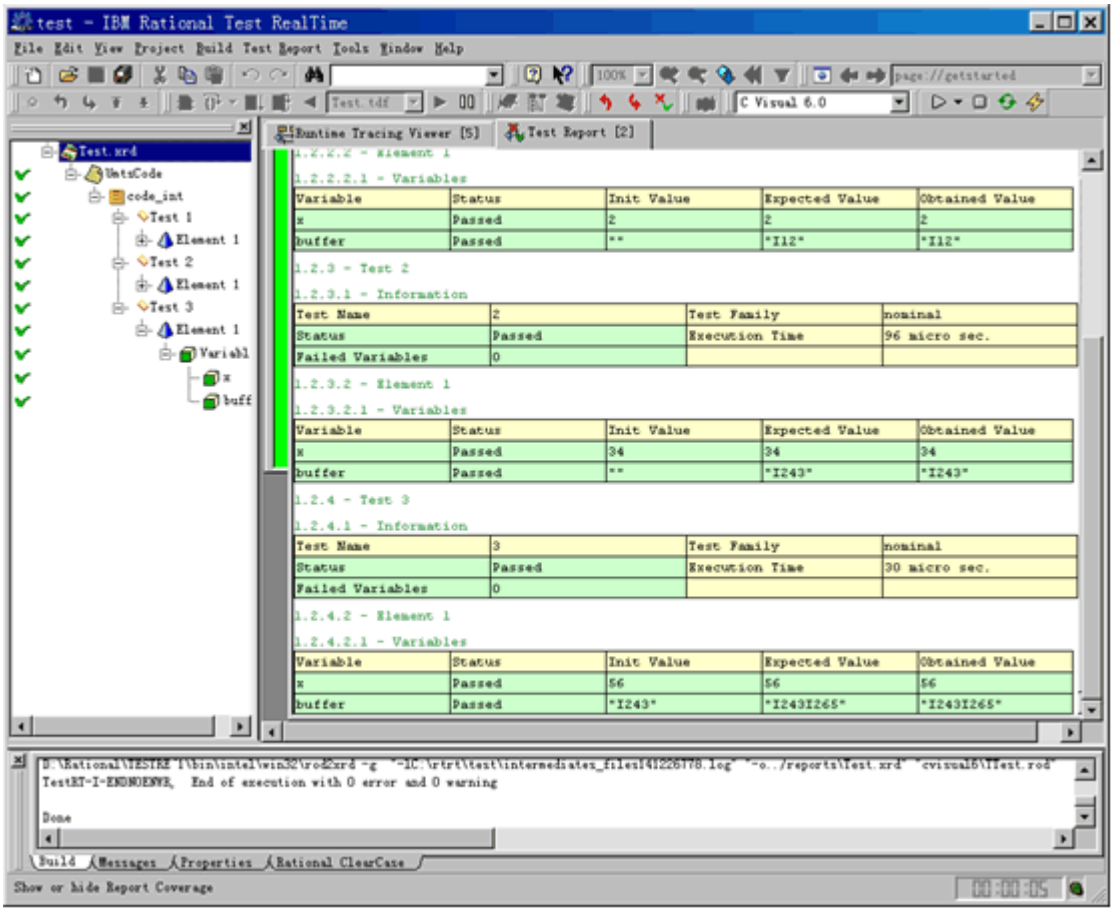


4.3.4 測試結果分析

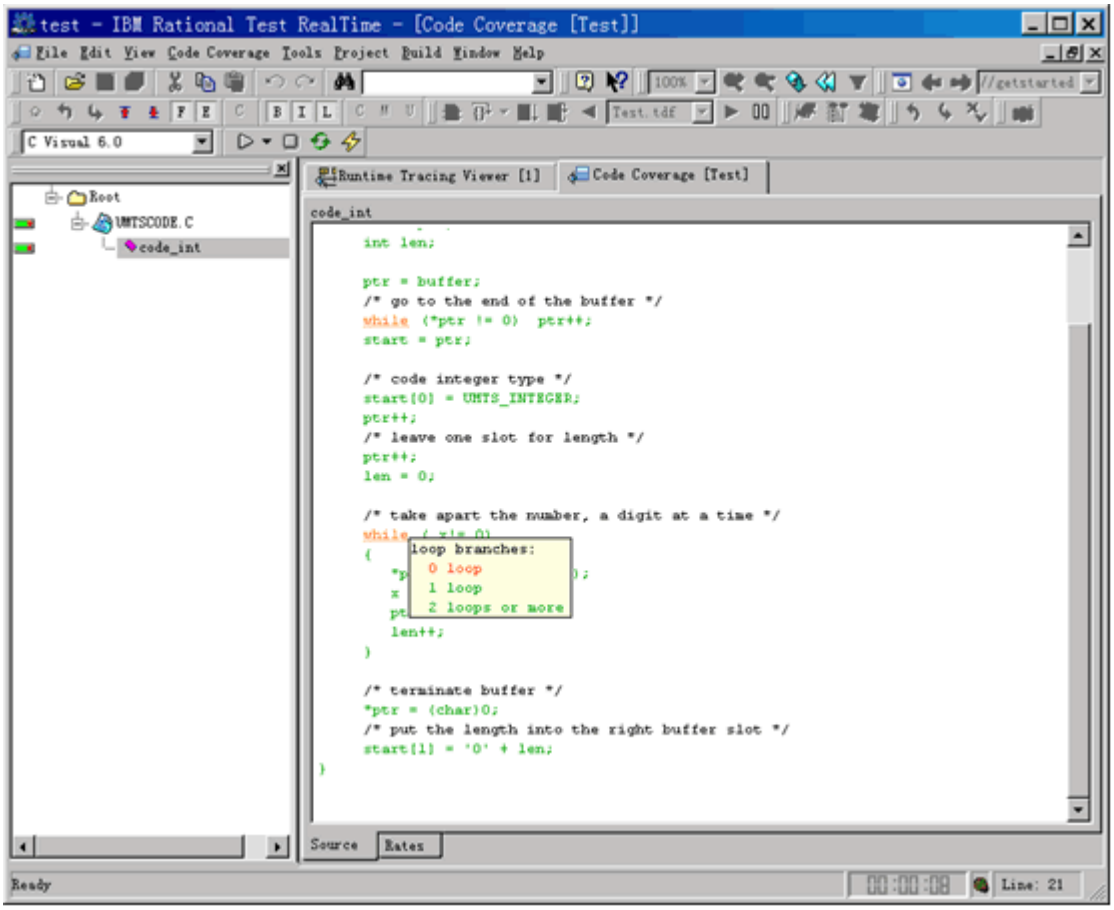


測試執行完成後，將在 Project Browser 中顯示所有的測試報告檔，這些檔均位於 c:\rtrt\reports 目錄。

滑鼠選中“Results”下的 Test, 點擊滑鼠右鍵，選擇“View Report”，進入測試如下測試報告，該報告將顯示測試用例通過和失敗的情況。



雖然 UtmsCode 函數的三個測試用例都通過，但需要通過程式碼覆蓋情況來分析測試的完備性，因爲沒有被測試的程式碼很有可能含有錯誤。滑鼠選中“Results”下的“Code Coverage”，點擊滑鼠右鍵，選擇“View Report”，進入測試如下程式碼覆蓋情況，如下圖：



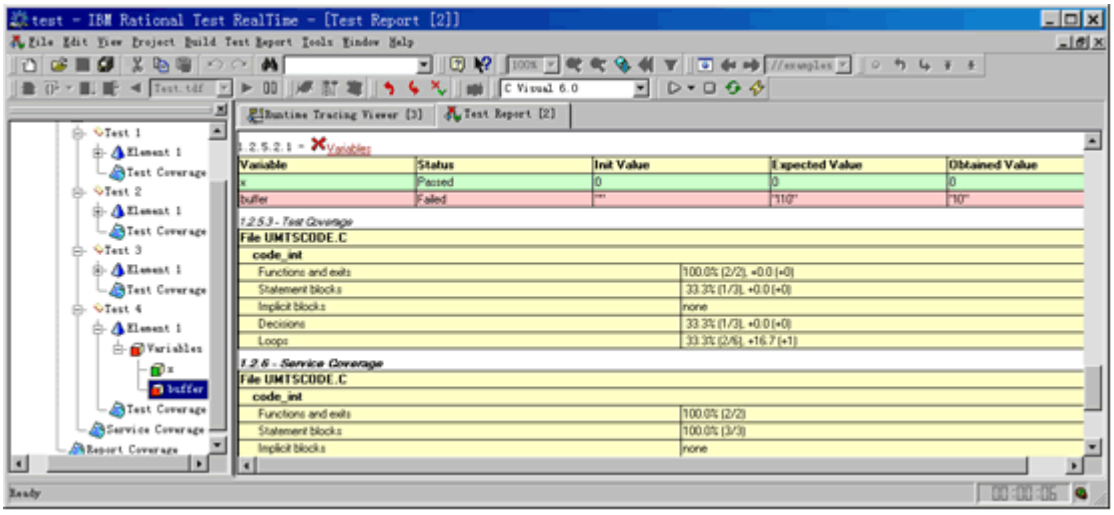
在程式碼覆蓋報告中，綠色的程式碼表示已經覆蓋， 橘紅的程式碼表示部分覆蓋，紅色的程式碼表示沒有覆蓋。通過對 UmtsCode.c 的程式碼覆蓋情況進行分析，發現 while (x!= 0) 語句只是部分覆蓋，該語句一次都不執行這種情況並沒執行，因此需要完善測試用例。關於程式碼覆蓋的詳細資訊參考線上幫助。

4.3.5 增強測試腳本 UtmsCode.ptu

通過對程式碼覆蓋情況進行分析，爲了覆蓋 while (x!= 0)的所有情況，需要增加如下測試用例：

```
TEST 4
FAMILY nominal
   ELEMENT
         VAR x,       init = 0,          ev = init
         VAR buffer,        init = "A",        ev = "AI10"
   #code_int(x, buffer);
   END ELEMENT
END TEST -- TEST 4
```

增強後的測試腳本參考 c:\rtrt\scripts\ UtmsCode_new2.ptu。再次執行測試。測試報告如下圖，發現 Test 4 不通過，表明 UtmsCode.c 中有錯誤。



4.3.6 修改測試腳本 UtmsCode.c

如附件 UtmsCode_new.c 的內容修改 UtmsCode.c，然後重新執行測試，將發現測試報告中的所有測試用例都通過，同時所有的程式碼均已被執行。

5. 小結

IBM Rational Test RealTime 除了支援 C 語言外，還支援 C++和 Java 語言，並支援基於消息的測試。開發人員利用該工具，實現了編碼、測試和調試的有機整合，使得邊開發邊測試或測試驅動的開發得以切實執行。

參考資料

- IBM Rational Test RealTime 手冊：ftp://ftp.software.ibm.com/software/rational/docs/documentation/manuals/testing.htm
- IBM Rational Test RealTime 技術支援：http://www-306.ibm.com/software/awdtools/test/realtime/support/

關於作者

IBM has authored this article