Testing Function main() Through Cantata++

Version No.: 1.0

Date: 15-Apr-2011

Project Name: DMAU ARINC 615A

Project Code: C102176

Document Revision History

Version No.	Date	Prepared by / Modified by	Significant Changes
1.0	15-Apr-2011	Saurabh Rahul, Kiran Devalla	Initial Version

Table of Contents

1 Problem Statement

Cantata++ has its own main() function irrespective of the code that it is going to test. So, if the testable code contains a main() function as well, then it treats the main() function from the testable code redundant over its own main() function. This renders main() function from the source code un-testable.

2 Goals and Objectives

The objective is to create an environment for main() function, of source code under test, such that it is testable through Cantata++

3 Strategy

Understanding the knitty-gritty of Cantata++, brainstorming and research forms the strategy for solving this problem.

4 Technical Comparison and Cost Benefit Analysis

Technical comparison of Cantata++ with other relevant testing tools is not applicable here as tool selection process is already completed by client itself and use of Cantata++ for Unit Testing is mandated by client.

5 Success Criteria

Success Criteria:

The approach decided should be

1. Compatible to embedded software testing fundamentals
2. Compatible to DO-178B guidelines
3. Compatible to client needs

6 Resource Requirement

In-project resources may be used to complete this activity. Nevertheless, a generic list of resources is presented here for the purpose of providing a better insight:

1. Hardware Resource:
   a. Personal Computer with minimum 512 KB RAM

2. Software/Tool Resource:
    a. Cantata++/ V 5.3.2
    b. GNU-X Tools/ V 4.2
    c. Microsoft Windows/ XP
    d. Microsoft Office/ 2003
3. Network Resource:

    Network resource is not a constraint for carrying out this activity. However, default network resources are available as part of organization's and department's IT policy.
4. Human Resource:

    Human resource is again part of the project team itself. Basic requirement may be of only one person with knowledge of Cantata++. However, other project members/ resources may team-up to provide better insight and expertise. Knowledge of Cantata++ may not be mandatory for support team but for the core team working on the solution a good knowledge of Cantata++ is mandatory.

7 Evaluation Requirement

Success of the approach is evaluated based on the successful evaluation of following documents containing the implementation details:

1. Test Procedure
2. Test Case/ Script
3. Test Result

8 Results Validation

If the test procedure, test case/ scripts and test result are prepared correctly as per Cantata++ features and test result successfully shows that main() has been successfully tested, as expected, through Cantata++, then it will be considered validated.

9 Piloting Requirements

There is no separate piloting requirement for this. However, once successful the finalized approach will be implemented in DMAU ARINC 615A Project during its Unit Testing Phase.

10 Market Survey

Not Applicable

11 Project Management Plan for In-house Development

No separate project management plan is required for this activity as the effort required is not more than a week.

12 Implementation Plan

The solution found and implemented successfully is listed below in steps:

1. Create the test script file, say test.c, as per testing requirements
2. Create a .c file, say replace.c
3. Copy all the contents of the Include Section of file test.c to the Include Section of file replace.c
4. Include an undefined command in replace.c for main() function which is there in the testable source code.

   ```
   #undef main
   ```
5. In dummy.c define alternate name for main by replacing 'main' with 'umain'

   ```
   #define main() umain()
   ```
6. Include the source code file (say, source.c), where main() function is defined, in replace.c

   ```
   #include "source.c"
   ```
7. Create a function, say pp_test_main(), in replace.c to invoke the function under test, umain().
   a. Case I: If main returns some value (not void), say of data-type int

   ```
   int pp_test_main()
   {
           int ret;
           ret = umain();
           return ret;
   }
   ```

   b. Case II: If return type of main is void

   ```
   void pp_test_main()
   {
       umain();
   ```

```
        return;
    }
```

8. Declare main() function and pp_test_main() function as extern in test.c file

   extern <<datatype>> main();

   extern <<datatype>> pp_test_main();
9. In test script, test.c, invoke pp_test_main() in each test case

Through this step-by-step approach function main() is renamed as umain() to avoid it clashing with main() function of Cantata++ and this strategy creates an environment through which functionality of function main () from source file under test is tested.

Both structural and functional coverage is achieved through this approach.

This approach is compatible with embedded software testing fundamentals, DO-178B guidelines and also with client's requirements.