VECTOR

User's Guide
Quick Start Guide

VectorCAST 2020

New editions of this guide incorporate all material added or changed since the previous edition. Update packages may be used between editions. The manual printing date changes when a new edition is printed. The contents and format of this manual are subject to change without notice.

Generated: 11/4/2020, 8:48 PM

Rev: 257ebc0

Part Number: VectorCAST Quick Start Guide for VectorCAST 2020

TABLE OF CONTENTS

Introduction

VectorCAST Overview

VectorCAST® is a suite of test automation tools:

- VectorCAST/C++™ and VectorCAST/Ada™ automate the unit and integration testing of C, C++, and Ada code, allowing you to easily test any subset of files (or packages for Ada) in isolation from the rest of the application.
- VectorCAST/QA™ automates the system testing of any application, and includes code coverage analysis for C, C++, and Ada source code.
- Enterprise Testing is a test automation platform that allows your team to collaborate on testing. It provides a single point of control for test creation, execution, and reporting for all test configurations.
- VectorCAST/Analytics collects source code and test metrics and publishes that data via a web-based dashboard, enabling you to identify trends in a single codebase or compare metrics between multiple codebases.

Before you begin:
This *Quick Start Guide* is intended to get you started quickly with the basic features of VectorCAST. Use it for quick reference.

For more detailed information about VectorCAST product features, please refer to the *Interactive Tutorials* and to the *User Guides* for VectorCAST/C++, VectorCAST/Ada, Enterprise Testing, VectorCAST/QA and VectorCAST/Analytics.

Starting VectorCAST

Before you start: Ensure that VectorCAST is installed and that the environment variable **`VECTORCAST_DIR`** is set to the installation directory. Refer to the *Interactive Tutorials* for detailed installation instructions.

UNIX

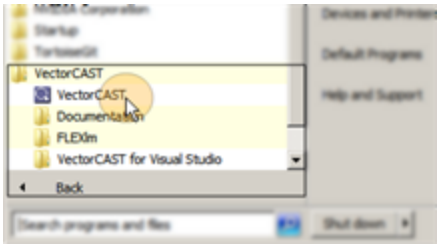
In UNIX or Cygwin, start VectorCAST by entering:

```
$VECTORCAST_DIR/vcastqt
```

Windows

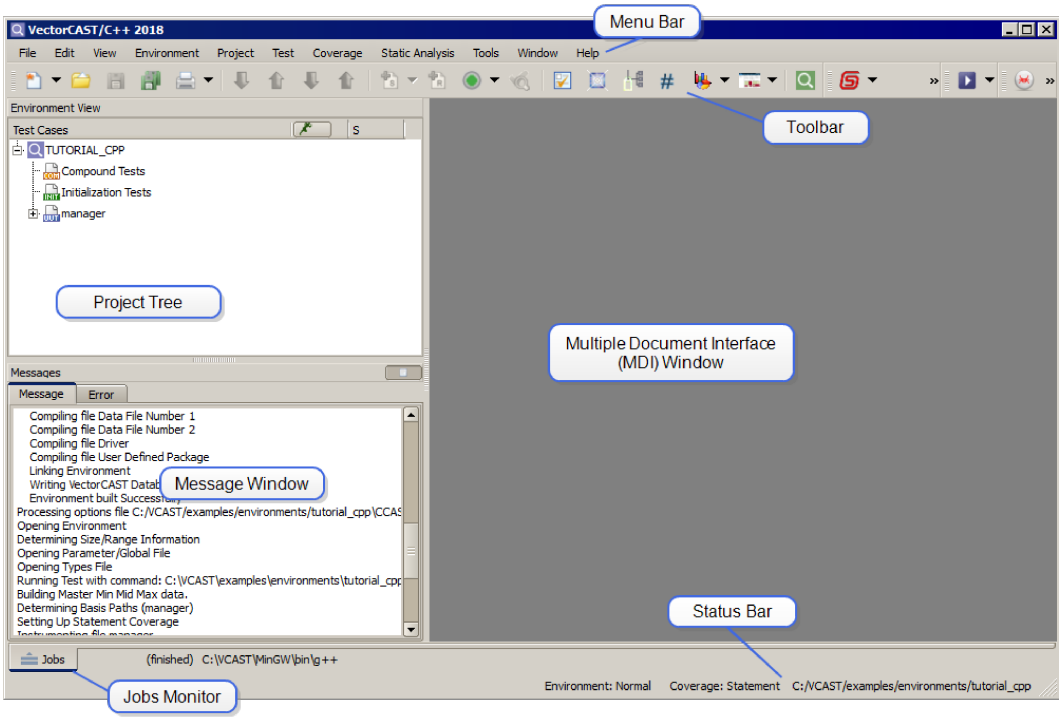
In Windows, from the Windows Start menu, select 

=> All Programs => VectorCAST => VectorCAST.



VectorCAST Interface

Below we discuss the default controls of the VectorCAST GUI. Note that you can return to this default arrangement at any time by using: **View =>Default Layout** from the menu bar.



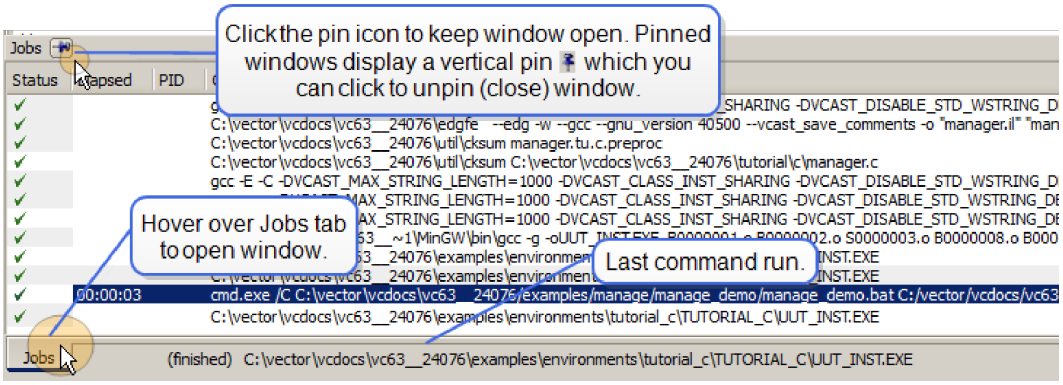
The VectorCAST main window is divided into four panes:

- **The Project Tree** is located on the left-hand side of the main window. It provides a high level view of the project structure.
- **The Message Window** is located along the bottom left of the main window. It contains tabs for informational messages and for error messages.
- **The Multiple Document Interface (MDI) Window** is located to the right of the Project Tree. It displays a variety of windows, including Test Case editors, Coverage Viewers, Report Viewers and Text Editors. Windows are collected into groups. See the *VectorCAST User Guides* for more information on MDI Window Groups.

> **The Jobs Monitor** is located on the bottom of the main window. It displays the status of jobs as they execute and exposes the associated back-end commands. See the *VectorCAST User Guides* for more information on the Jobs Monitor.

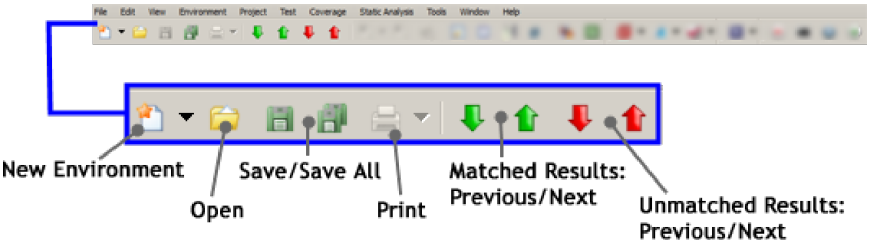
The **Project Tree** and the **Message Window** are docking windows and can be moved to any location on your desktop. See the *VectorCAST User Guides* for more information on docking windows.

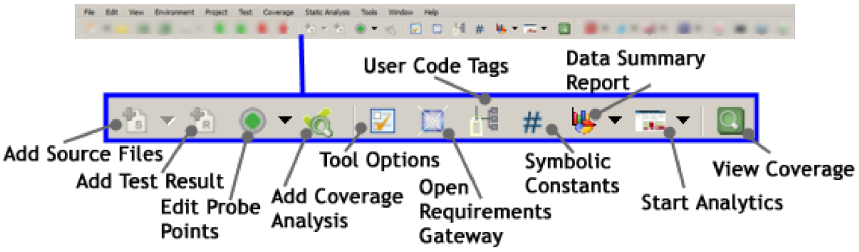
The **Jobs Monitor** is hidden by default when VectorCAST is first opened. Hover over the Jobs tab to open the Jobs Monitor window and move the mouse away from the tab to close the window. To keep the window open, click the Pin icon to the right of the Jobs tab.

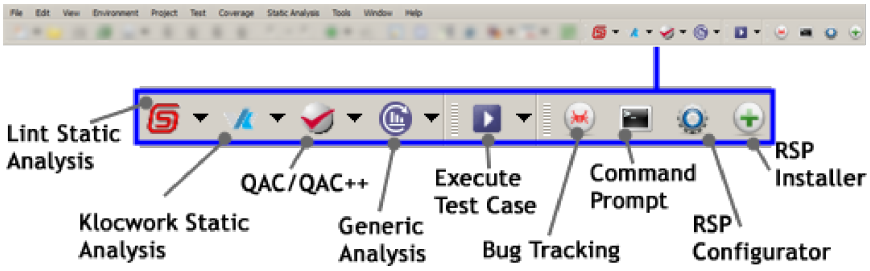


The **Status Bar** on the lower right will display the path to the working directory, the status of the environment, and the type of Coverage initialized.

The VectorCAST tools and functions are available from the **Menu Bar** and the **Toolbar** located at the top of the main window. The toolbar includes icons for third-party tools integrated with VectorCAST and are functional only if you are licensed for those features. You can become familiar with the most widely used of these tools and functions by using the *Interactive Tutorials*.

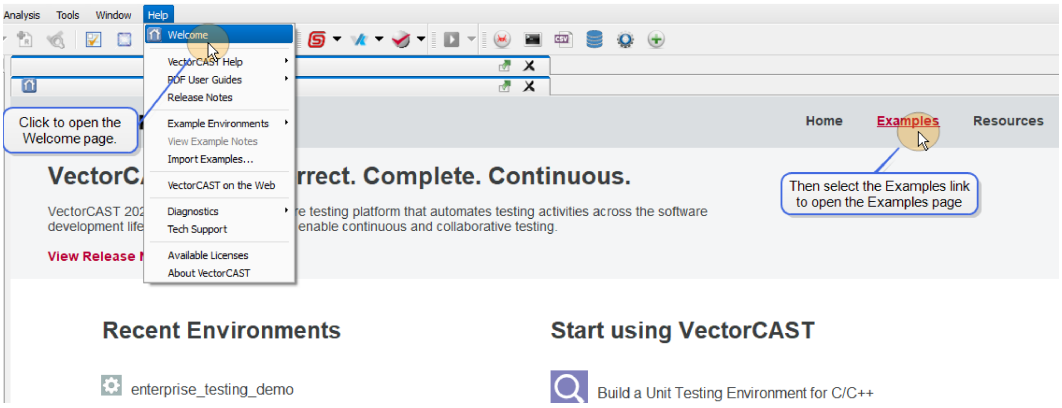




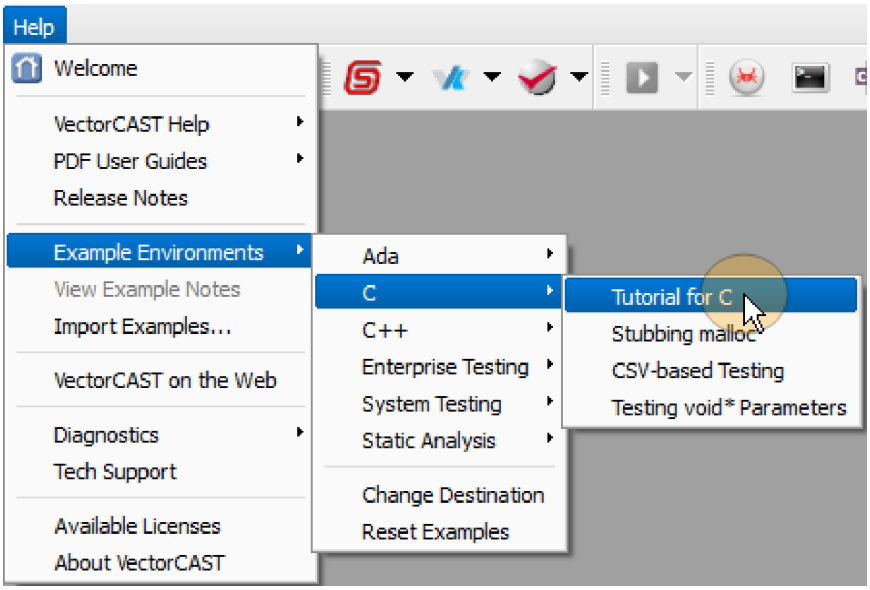


Using the VectorCAST Examples

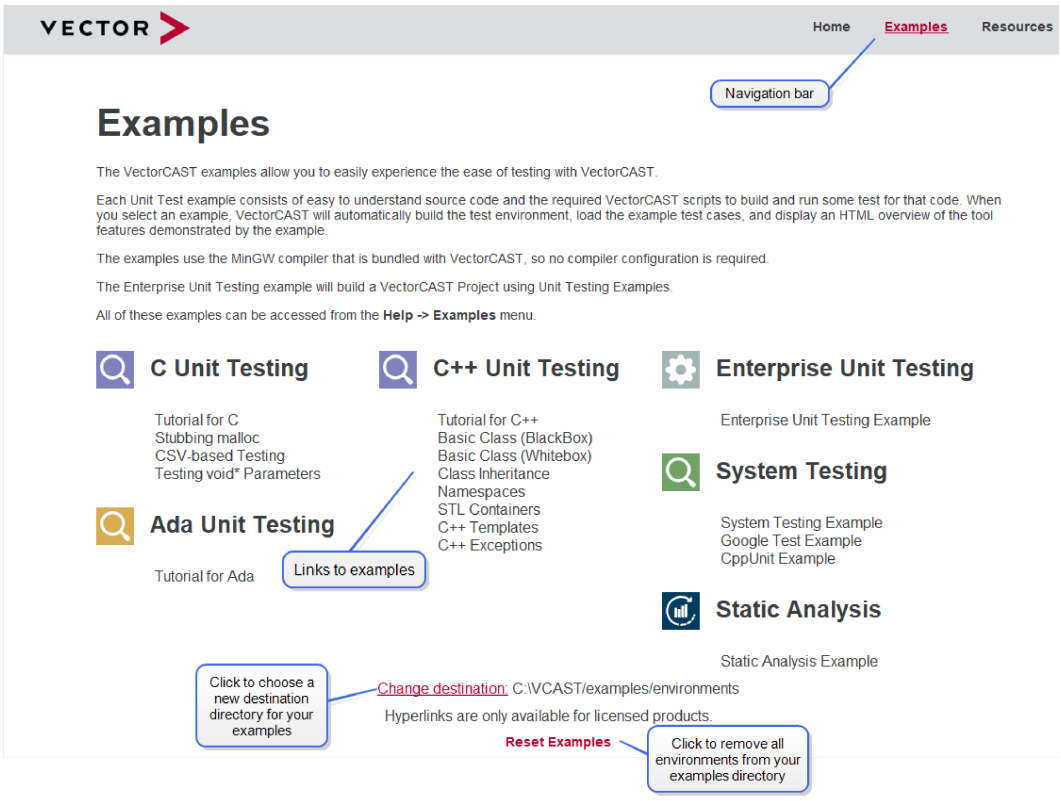
VectorCAST has a variety of built-in examples to help you quickly learn how to use the tools. Examples can be accessed from the navigation bar on the Welcome page by selecting the Examples link. To navigate to the Welcome page from anywhere within VectorCAST, select **Help =>Welcome** from the Menu Bar. Then select the Examples link from the navigation bar to open the VectorCAST Examples page.



You can also access the examples from anywhere within VectorCAST by selecting **Help => Example Environments** from the Menu Bar.



The VectorCAST Examples page containing links to all the examples is displayed:



At the bottom of the Examples page, a **Change destination** link is provided which allows you to choose a different location for your examples build directory. Alternatively, this link is available from the Menu Bar by selecting **Help => Example Environments => Change Destination**. The default location for the directory is **`%VECTORCAST_DIR%\examples\environments`**. Note that changing

the directory will not delete examples from the current directory.

A **Reset Examples** link is also available at the bottom of the Examples page. Alternatively, this link is available from the Menu Bar by selecting **Help => Example Environments => Reset Examples** from the Menu Bar. Selecting this link removes *all* of the example environments from your build directory.

In the remainder of this guide, you will do the following:

1. Create a VectorCAST/C++ unit testing environment and run some tests.
2. Create a VectorCAST/QA system testing environment and capture code coverage.
3. Create a VectorCAST project using Enterprise Testing.
4. Launch the VectorCAST/Analytics dashboard and view project metrics.

Create a Unit Testing Environment

Create a Unit Testing Environment

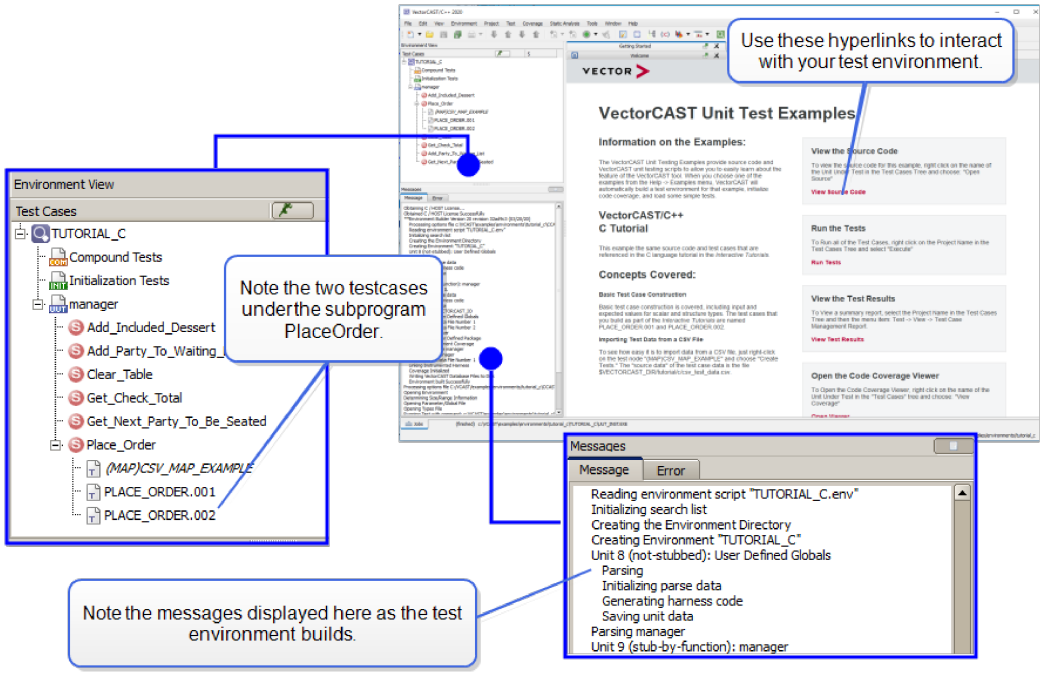
VectorCAST/C++ is a structural testing tool that allows you to easily perform Application Programming Interface (API) testing for portions of an application in isolation from the complete application.

The VectorCAST Unit Testing Examples provide source code and test scripts to allow you to quickly learn the features of VectorCAST/C++.

To create a Unit Testing Environment, navigate to the VectorCAST Examples page and under the C Unit Testing column, select **Tutorial for C**. Alternatively, from the Menu Bar you can select **Help =>Example Environments =>C=>Tutorial for C**. VectorCAST will automatically build the test environment.

Note: For the purposes of this Quick Start Guide, the discussion will be based on the Tutorial for C example. To create a C++ or an Ada Testing Environment, select **Tutorial for C++** or **Tutorial for Ada** from the **Example Environments** menu. The menu flow and concepts discussed will be the same for all three programming languages.

Once the environment completes building, a summary page is provided for you in the MDI Window. Refer to this page to learn more about basic test case construction concepts, and use the provided hyperlinks on the right to interact with your test environment.

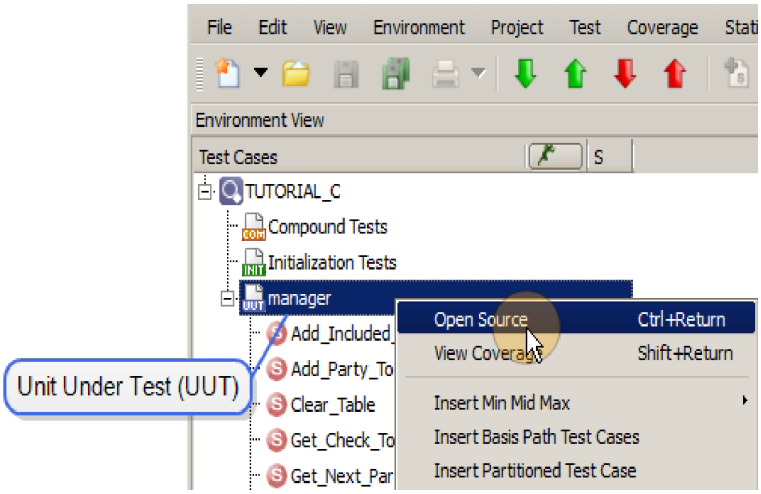


Next, you will view the source code.

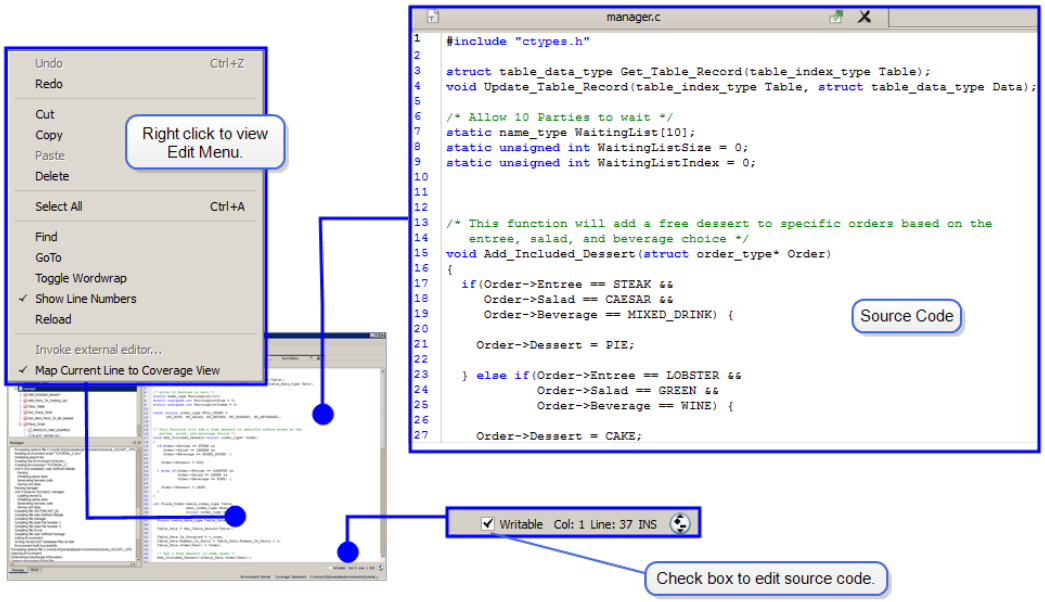
View Source Code

To view the source code for this example, right-click on the name of the Unit Under Test (UUT) located

in the Project Tree and select **Open Source**.

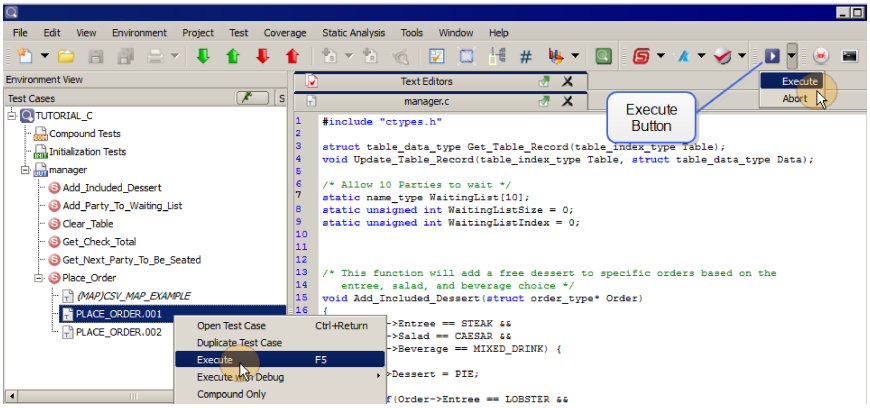


The Text Editor opens up in the MDI Window and you will see the source code for the selected file displayed. The initial code display is read-only. To edit the source code, check the Writable box in the lower right. Right-click with your mouse within the MDI Window to view a pop-up menu of editing commands.

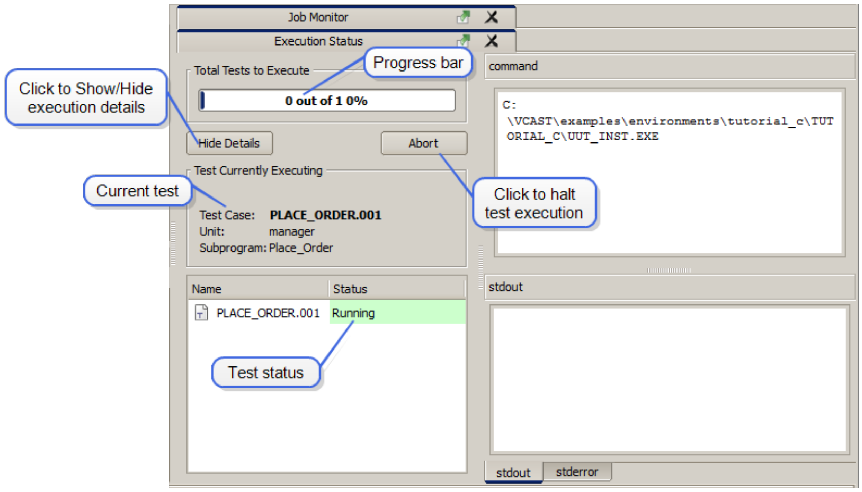


Run the Tests

The example environment has some pre-built test cases. To run a single test case, right-click on **`PLACE_ORDER.001`** in the Project Tree and select **Execute**. You can also execute the test case by selecting the **Execute** button from the Toolbar.

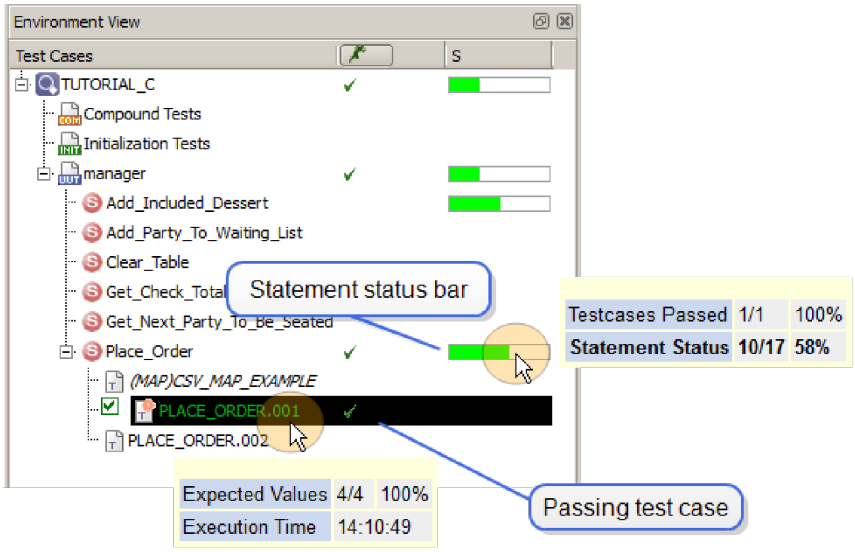


The Execution Status viewer opens in the MDI Window. The Execution Status viewer opens automatically when a test case is executed and provides detailed real time test case execution information. Refer to your *VectorCAST User Guide* to learn more about the Execution Status Monitor and Jobs Monitor windows.



Note that after the test has executed the Project Tree updates to display the test case status. Passing test cases are displayed in green. Failing test cases are displayed in red. Hover over the test case to see the execution time and the percentage of expected values reached.

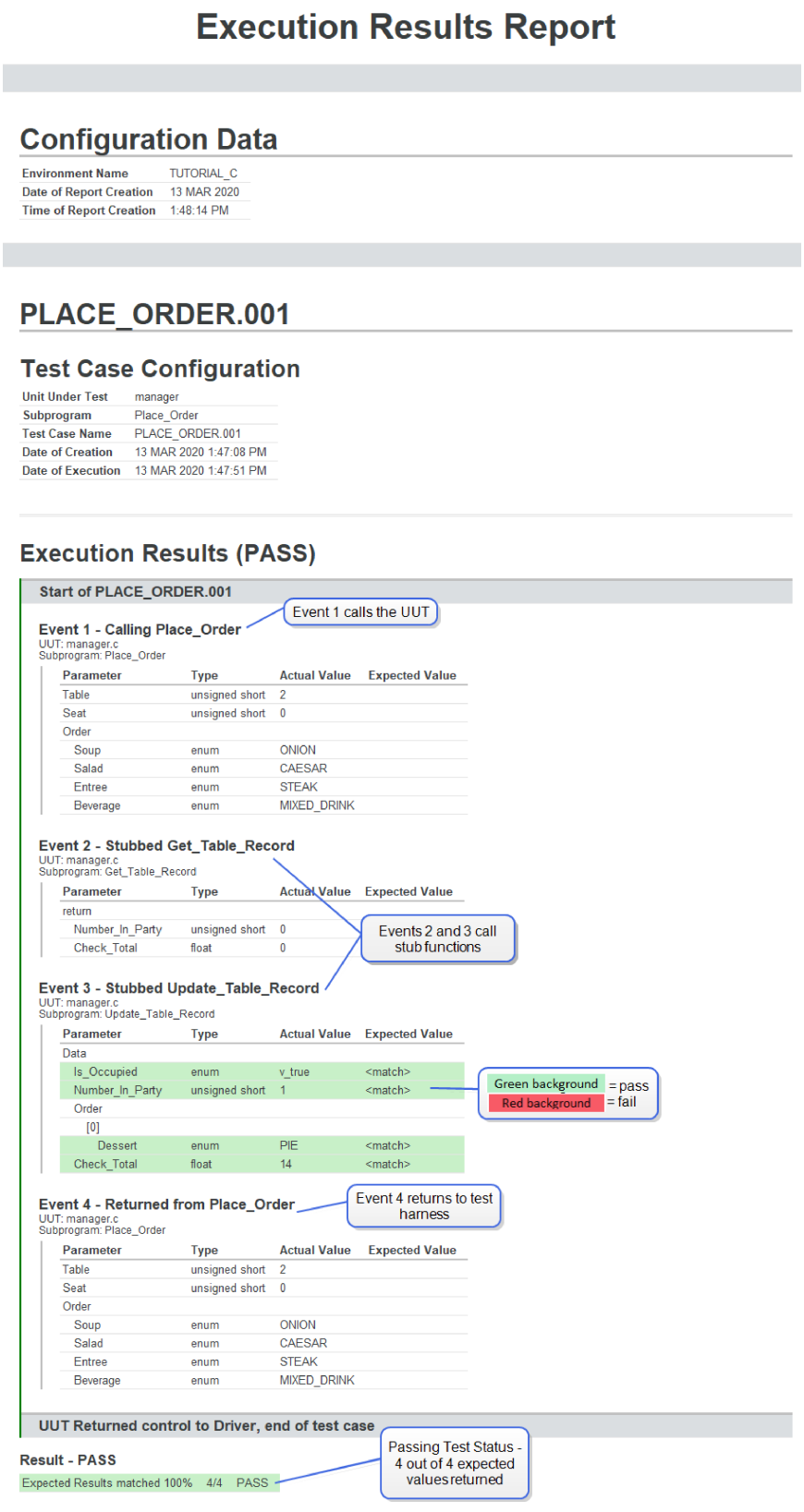
A new column is displayed in the Project Tree indicating statement coverage status. Hover over the status bar to see that running test case **`PLACE_ORDER.001`** achieves 58% statement code coverage for the subprogram Place_Order, covering 10 out of 17 statements.



View the Test Results

View the test results for the test case by highlighting the test case in the Project Tree and selecting **Test =>View =>Execution Results** from the Menu Bar.

The Execution Results Report is displayed in the Report Viewer. The Report includes the configuration of the test case, the execution results with the expected and actual values, and the test status. Refer to your *VectorCAST User Guide* for more information on viewing test execution results.



Execution Results Report

Configuration Data

Environment Name	TUTORIAL_C
Date of Report Creation	13 MAR 2020
Time of Report Creation	1:48:14 PM

PLACE_ORDER.001

Test Case Configuration

Unit Under Test	manager
Subprogram	Place_Order
Test Case Name	PLACE_ORDER.001
Date of Creation	13 MAR 2020 1:47:08 PM
Date of Execution	13 MAR 2020 1:47:51 PM

Execution Results (PASS)

Start of PLACE_ORDER.001

Event 1 calls the UUT

Event 1 - Calling Place_Order
UUT: manager.c
Subprogram: Place_Order

Parameter	Type	Actual Value	Expected Value
Table	unsigned short	2	
Seat	unsigned short	0	
Order			
Soup	enum	ONION	
Salad	enum	CAESAR	
Entree	enum	STEAK	
Beverage	enum	MIXED_DRINK	

Event 2 - Stubbed Get_Table_Record
UUT: manager.c
Subprogram: Get_Table_Record

Parameter	Type	Actual Value	Expected Value
return			
Number_In_Party	unsigned short	0	
Check_Total	float	0	

Events 2 and 3 call stub functions

Event 3 - Stubbed Update_Table_Record
UUT: manager.c
Subprogram: Update_Table_Record

Parameter	Type	Actual Value	Expected Value
Data			
Is_Occupied	enum	v_true	<match>
Number_In_Party	unsigned short	1	<match>
Order			
[0]			
Dessert	enum	PIE	<match>
Check_Total	float	14	<match>

Green background = pass
Red background = fail

Event 4 - Returned from Place_Order
UUT: manager.c
Subprogram: Place_Order

Event 4 returns to test harness

Parameter	Type	Actual Value	Expected Value
Table	unsigned short	2	
Seat	unsigned short	0	
Order			
Soup	enum	ONION	
Salad	enum	CAESAR	
Entree	enum	STEAK	
Beverage	enum	MIXED_DRINK	

UUT Returned control to Driver, end of test case

Result - PASS

Expected Results matched 100% 4/4 PASS

Passing Test Status - 4 out of 4 expected values returned

Depending on the items selected in the Project Tree, you can view the Execution Results for a single

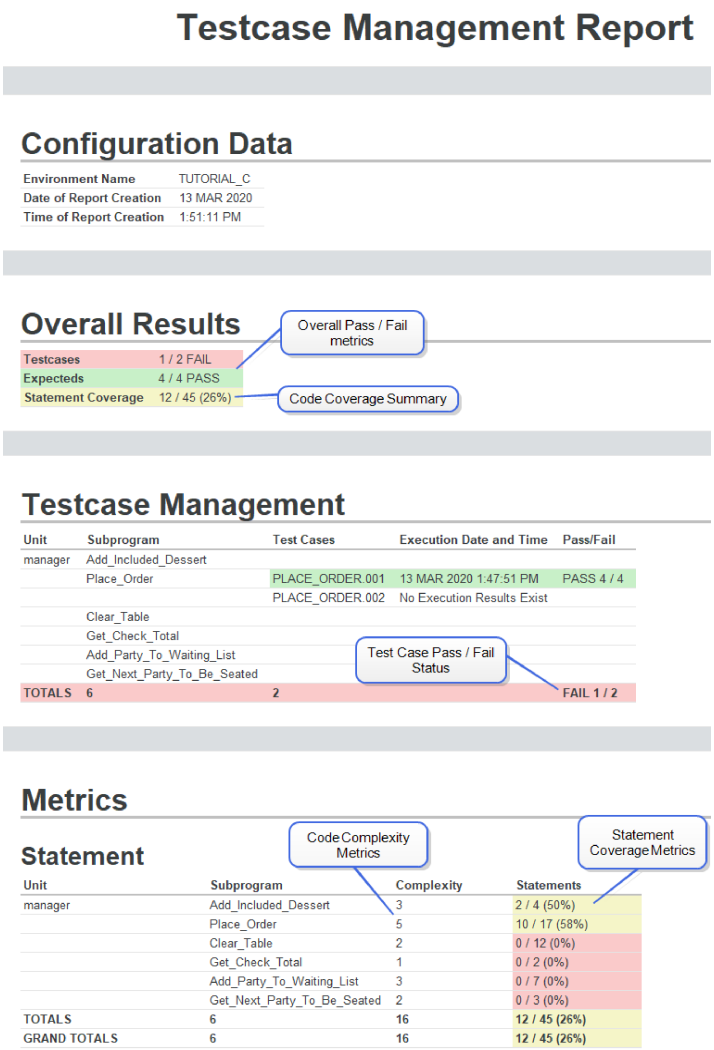
test case, a combination of test cases, a subprogram or an environment by selecting **Test =>View =>Execution Results** and viewing the Execution Results Report.

The Test Case Management Report is a summary of the testing status for a particular environment. First, execute all the Test Cases. Right click on the Project Name in the Project Tree and select **Execute** or select the Execute button from the Toolbar.

As the Test Cases execute you will see the Execution Status Monitor window open, allowing you to see the output and status of each test as it is running in real time.

Once the Test Cases have executed, you can view the Test Case Management Report by selecting **Test =>View =>Test Case Management Report** from the Menu Bar.

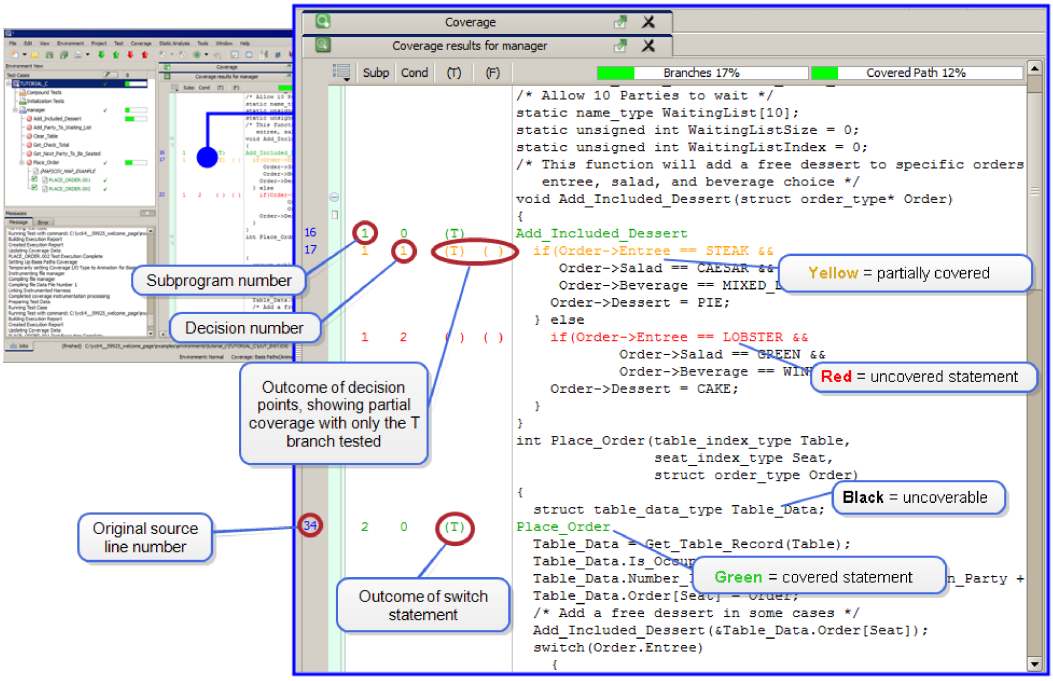
The contents of the report reflect the items selected in the Project Tree. Scroll through the report and note the information provided.



Open Code Coverage Viewer

Open the Code Coverage Viewer by selecting the View Coverage button from the Tool Bar. You can also access the Code Coverage Viewer by right-clicking on the name of the Unit Under Test in the Project Tree and selecting **View Coverage**.

An annotated version of the **`manager`** source code is displayed in the Coverage Viewer in the MDI Window. For display purposes here the example environment has been assigned Basis Paths coverage type.



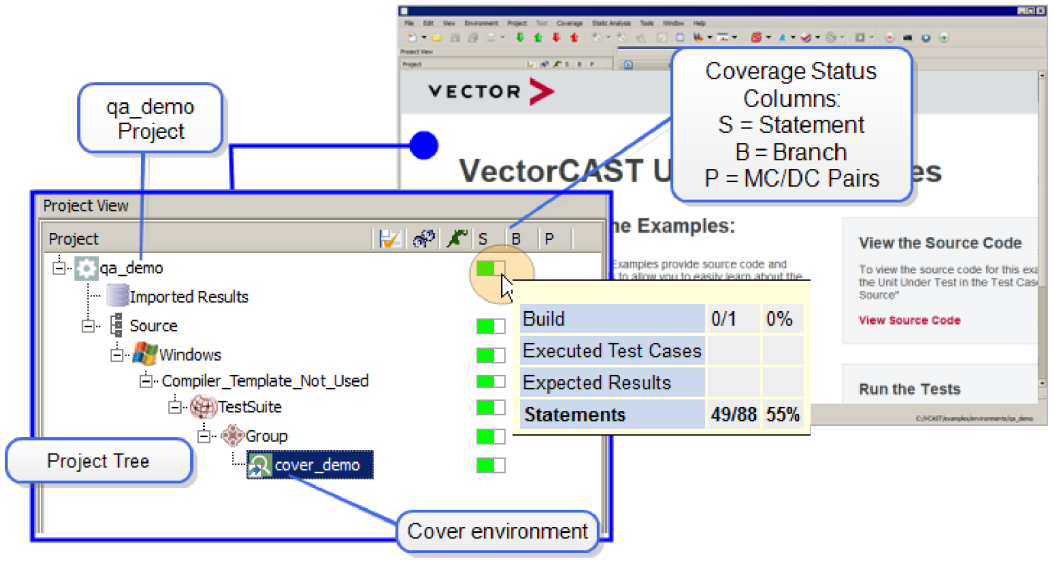
Create a System Testing Environment

Create a System Testing Environment

VectorCAST/QA is the integration of VectorCAST/Cover and Enterprise Testing. It automates the system testing of any application, and includes code coverage analysis for C, C++, and Ada source code. Code coverage analysis is the best way to measure the completeness of testing, and VectorCAST/QA makes this simple, regardless of your testing methodology.

To build the System Testing example, navigate to the VectorCAST Examples page and under the System Testing column, select **System Testing Example**. Alternatively, from the Menu Bar you can select **Help =>Example Environments =>System Testing=>System Testing Example**. VectorCAST will automatically build the System Testing example.

In the Project Tree, note that the qa_demo project is displayed. Expand the nodes to show the cover_ demo environment. There are also three columns on the right to display coverage status for Statement, Branch and MC/DC Pairs. Use your mouse to hover over the coverage bar for the qa_demo project and observe that it has 55% Statement coverage.

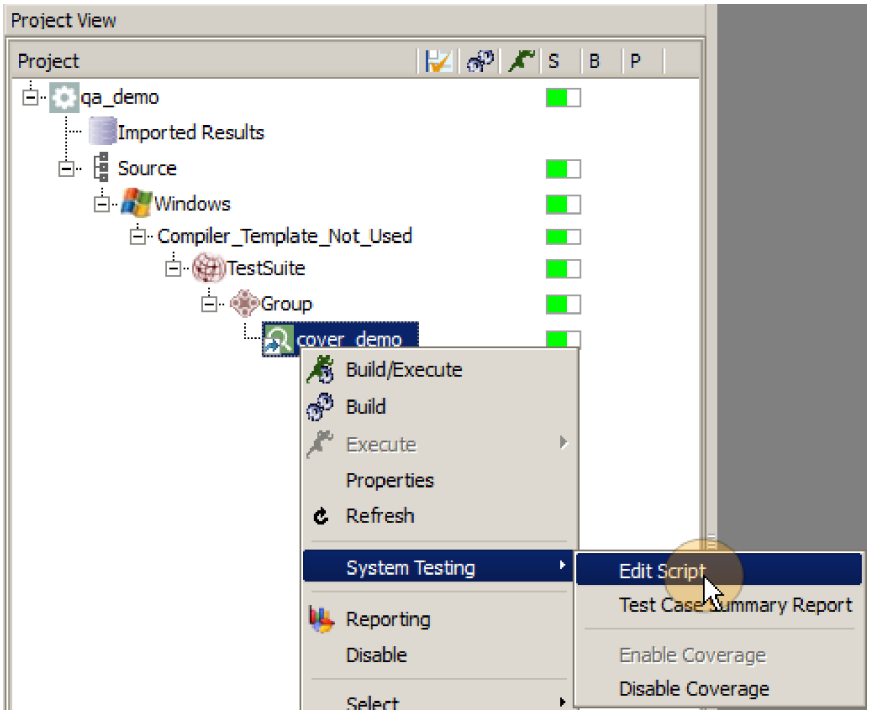


Next, you will view the Python Configuration Script, **`system_tests.py`**.

View System Test Script

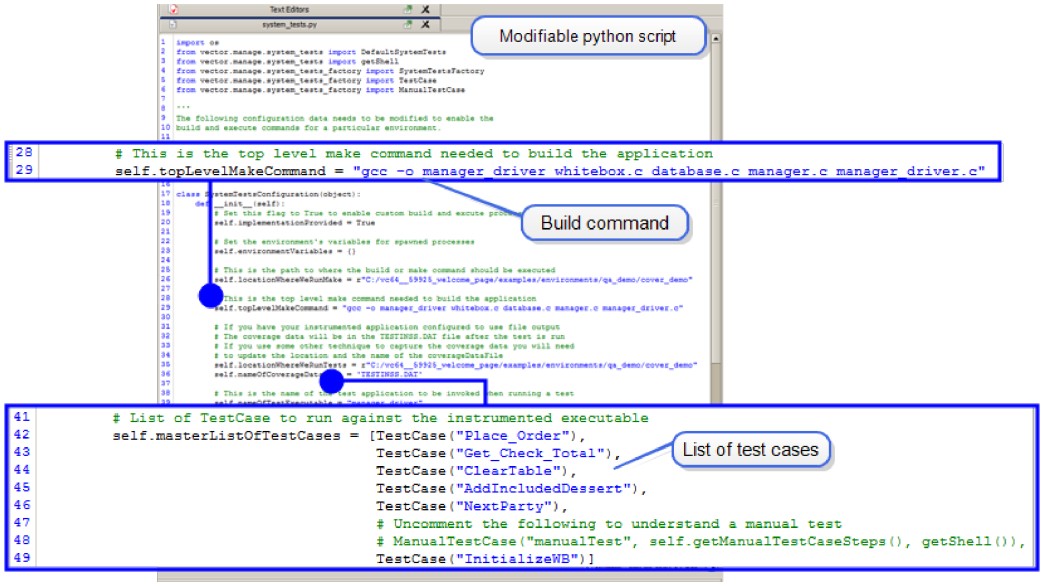
The Python Configuration Script contains functions that are invoked by VectorCAST to provide custom build and execute commands based on the name of a VectorCAST/Cover environment in a VectorCAST project. The configuration script is fully customizable and allows the user to define the application build commands, list of test cases, and test execution commands for an application.

To open and edit the Configuration Script, right-click on the cover_demo Environment node and select **System Testing => Edit Script** from the context menu.



The Python Configuration Script, **`system_tests.py`**, opens in the Text Editor.

Note that the script contains the build command for the application. Also note the list of 6 system test cases: **`Place_Order, Get_Check_Total, ClearTable, AddIncludedDessert, NextParty,`** and the manual test case **`InitializeWB`**.

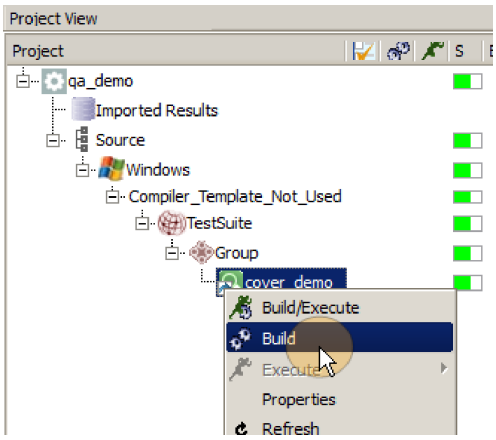


The script can be modified as required and the changes saved. We will use the existing script for our example and simply close the Text Editor.

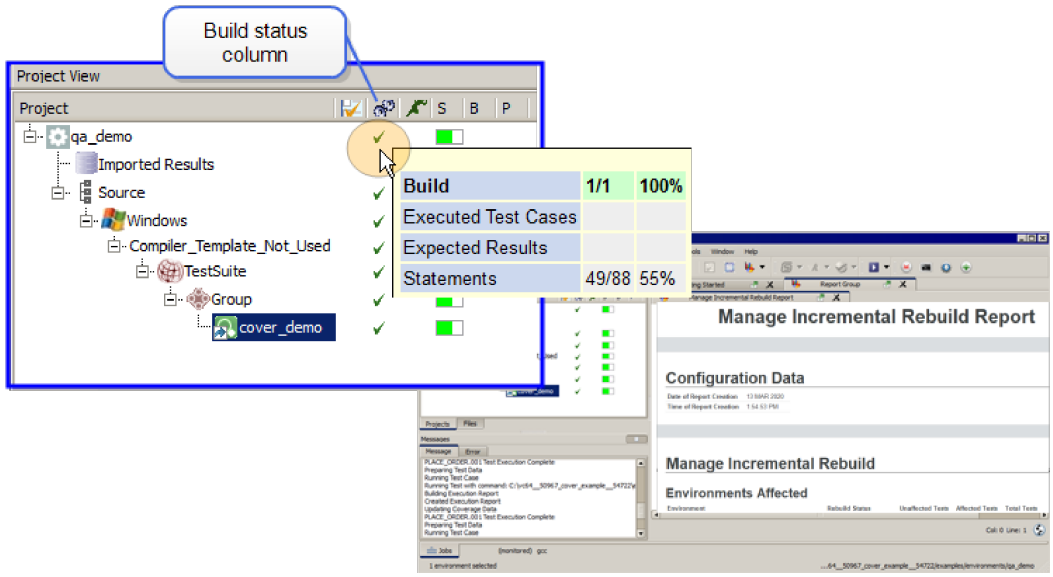
Next, you will build the instrumented executable using the **`system_tests.py`** script.

Build Instrumented Executable

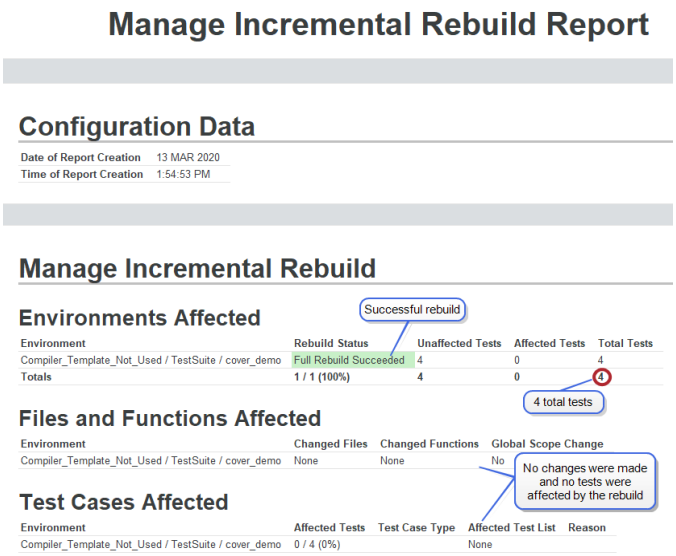
To build the instrumented executable, right-click on the cover_demo environment node and select **Build** from the context menu. Once the Python configuration has been set up, the incremental build functionality performs only the work that needs to be done, based on the current state of the project.



The Build Status column in the Project Tree now shows green check marks indicating successful build status. Hover over the check marks to see popups with the build details.



The Manage Incremental Rebuild Report opens when the build is complete. Note that we made no modifications to files or test cases, and the report shows a successful rebuild with no files, functions or test cases affected.

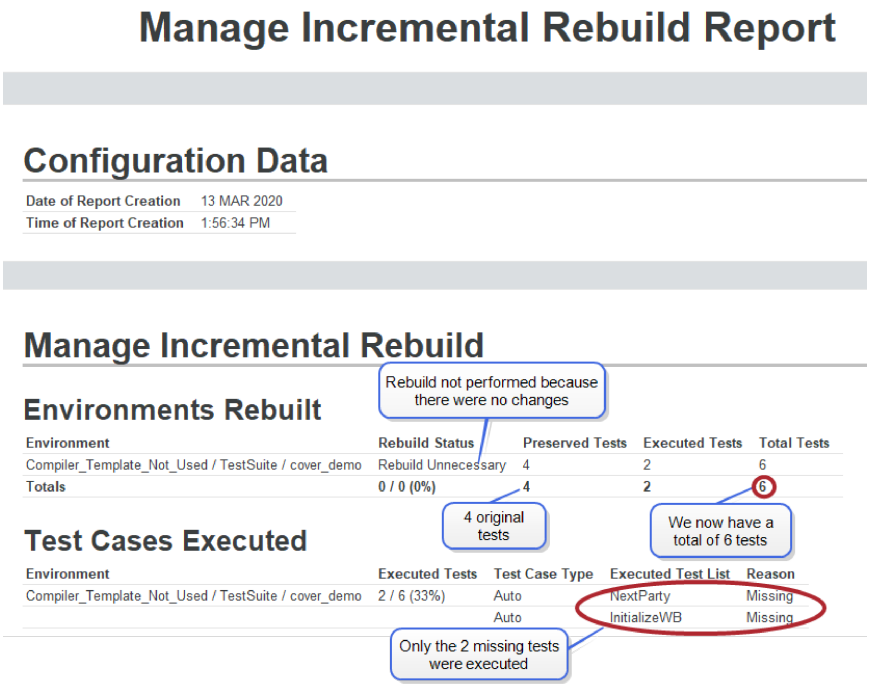


Also note that there are a total of 4 tests reported. Recall that we noted 6 tests are listed in the Python Configuration Script. Two of the tests are missing here.

Next, we will run the missing test cases in the cover_demo environment.

Execute Test Cases

To run the test cases in the cover_demo environment, right-click on the environment in the Project Tree and select **Execute => Full** from the context menu. Once again, the incremental build and execute functionality performs only the work that needs to be done based on the current state of the project.



In our example, only the two missing tests cases, **`NextParty`** and **`InitializeWB`**, were executed.

To learn more about System Testing environments, see the *"VectorCAST/QA - System Test Automation"* section in the *Enterprise Testing with VectorCAST User Guide*.

Create a VectorCAST Project

Create a VectorCAST Project

Enterprise Testing is a Test Automation Framework that sits on top of VectorCAST/C++ or VectorCAST/Ada test environments and allows test design, execution, and reporting to be distributed across the enterprise. The VectorCAST project supports a variety of work flows allowing for team collaboration, testing of multiple configurations, change-based testing, and massively parallel testing.

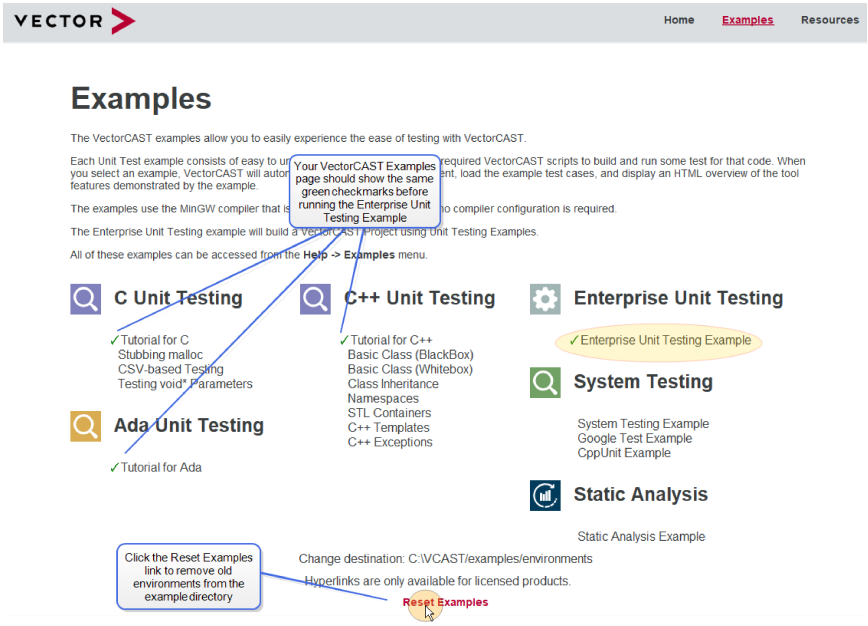
Enterprise Testing can import existing VectorCAST/C++ and VectorCAST/Ada test environments, or be used to create new environments. In this section, you will take existing VectorCAST environments and import them into a VectorCAST Project.

Before you start: The basic Enterprise Unit Testing example is built using any Unit Testing Environments you created previously. If there are no existing environments, the example will create a new unit test environment using the Tutorial for C.

For our example, we will first reset the examples on the Examples page and then import the C, C++, and Ada Unit Testing environments.

To set up our VectorCAST project, go to the VectorCAST Examples page and select the **Reset Examples** link located at the bottom of the page.This removes all of the example environments from your build directory, and gives us a clean directory.

Next we will create our unit environments. Click to run the example Tutorial for C located under the C Unit Testing column. A green check mark is displayed next to it when the build is complete. Return to the VectorCAST Examples page and click on the Tutorial for Ada and the Tutorial for C++ examples so that each also displays a green check mark:

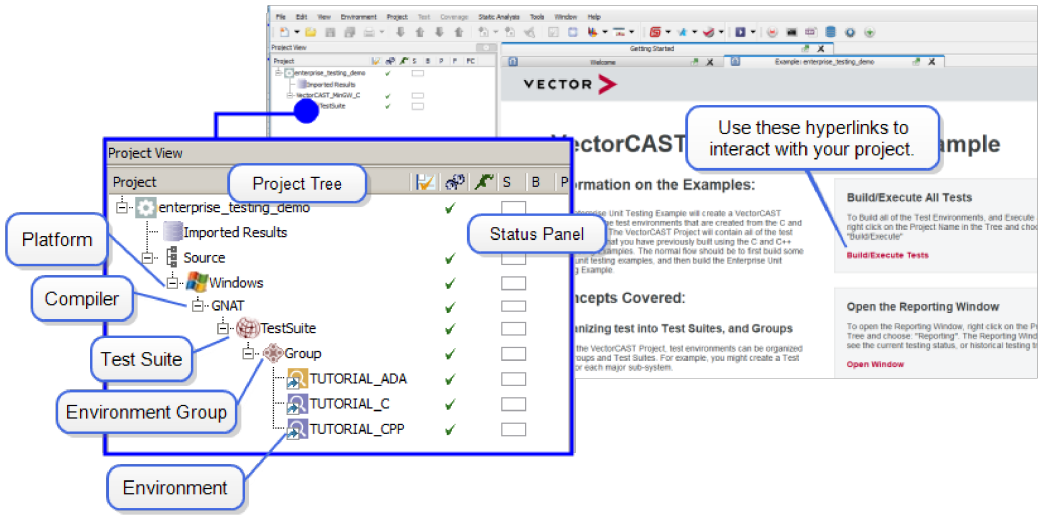


Finally, we create our VectorCAST Project. Navigate to the VectorCAST Examples page and under the Enterprise Unit Testing column, select **Enterprise Unit Testing Example**. Alternatively, from the Menu Bar, you can select **Help =>Example Environments =>Enterprise Testing =>Enterprise Unit Testing Example**. VectorCAST will automatically build the VectorCAST Project using the Unit

Testing environments you created previously.

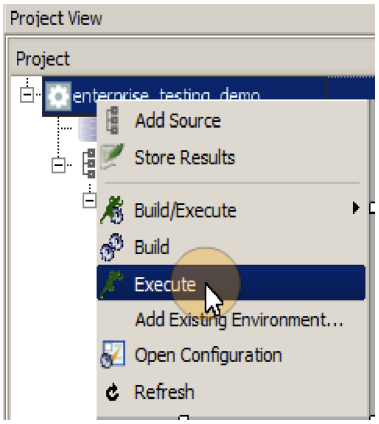
Once the project completes building, a VectorCAST Enterprise Testing Example summary page is provided for you in the MDI Window. Refer to this page to learn more about basic VectorCAST project concepts, and use the provided hyperlinks on the right to interact with your project.

In the Project Tree you will now see the enterprise_testing_demo project displayed. Note that a Test Suite has been created which contains each of the environments. Expand the Project Tree to see the individual Environments. Next, you will execute all of the tests.



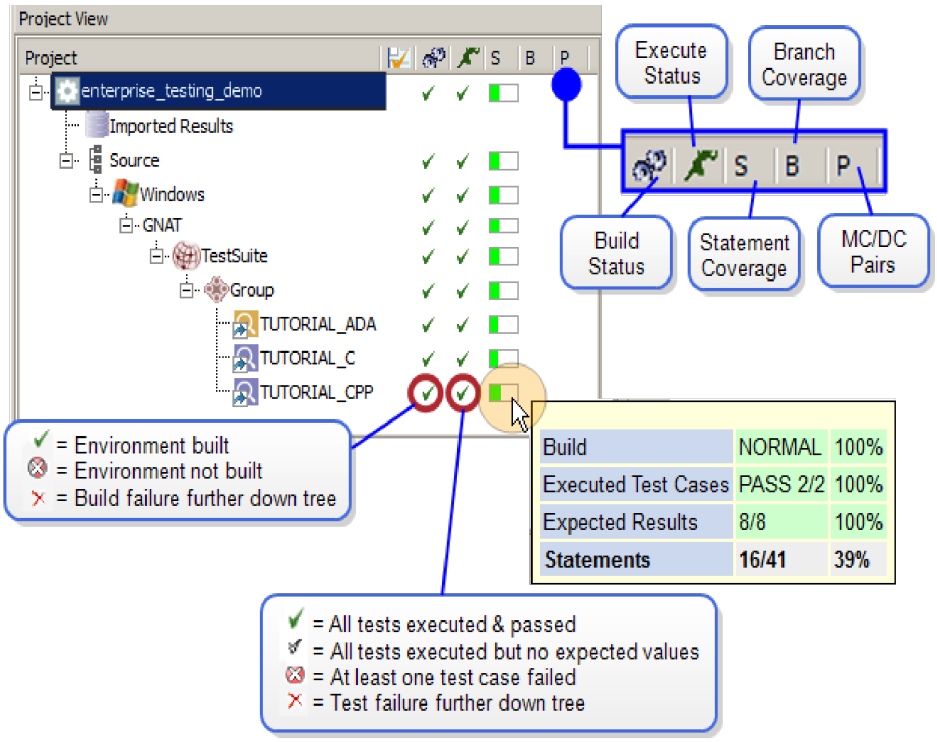
Execute All Tests

To execute all of the tests in the enterprise_testing_demo project, right-click on the Project Name (enterprise_testing_demo) and select **Execute** from the context menu.



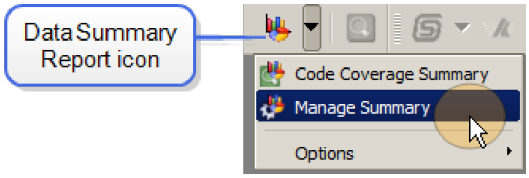
You can follow the execute process in the Manage Status viewer which opens in the MDI Window. As test cases are executed data is stored in a SQL database and used to generate reports showing testing status and trends, making it easy to analyze regression trends.

The Status Panel updates to display testing status. On the status panel you will see status for the Environment Build, Test Execution and Statement Coverage. Hover over the Statement Coverage bar to see a pop-up of the Build and Coverage details.



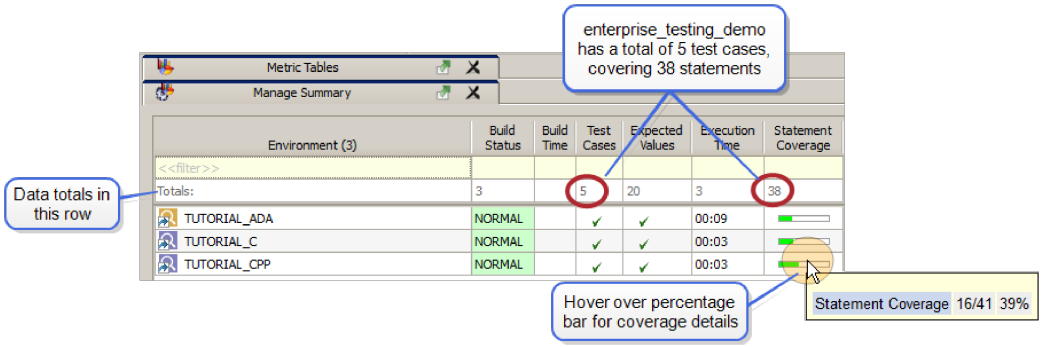
View Manage Summary

The Manage Summary table allows you to easily view environment execution results and metrics for all the environments of a VectorCAST project. To open the summary from the Toolbar, select the enterprise_testing_demo node in the Project Tree and then select **Manage Summary** from the Data Summary Report icon drop-down menu.



Alternatively, from the Menu Bar, select **Project => Manage Summary**.

The Manage Summary opens and displays data for all three environments in our enterprise_testing_ demo project.

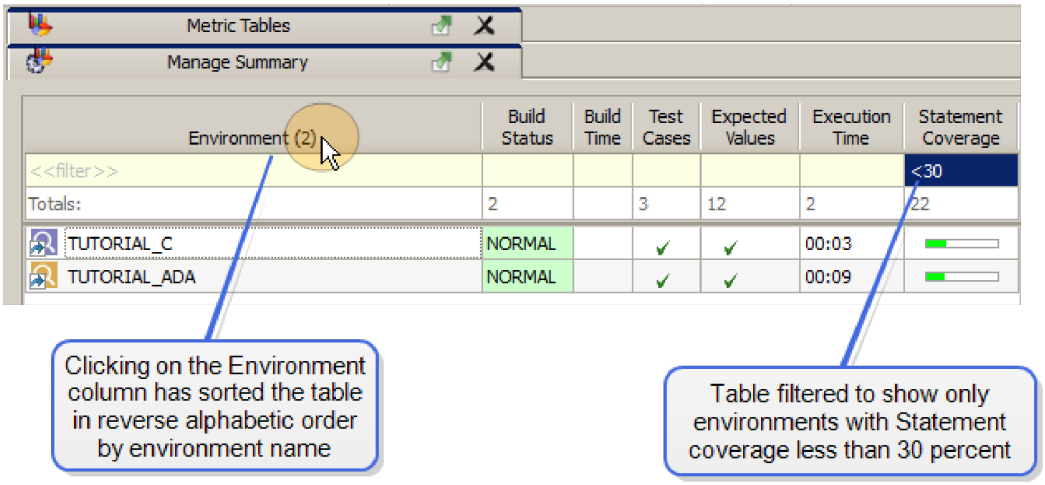


The Manage Summary is dynamic and tracks the current selection in the Project Tree. Try selecting different environment nodes and note that as units are selected and deselected in the Project Tree, the Manage Summary table updates in real time reflecting the selections.

The data displayed in the Manage Data Summary includes the Environment name, Build Status, Build Time, number of Test Cases, number of Expected Values, Execution Time and achieved coverage for each coverage type.

The Totals row at the top of the table displays the totals for each data column. In the example above, note that in the enterprise_testing_demo project we have a total of 5 test cases, covering 38 statements.

Sorting and filtering is available to locate data of interest. Sort by clicking on any column heading. Access the filter by typing into the top row of any column.

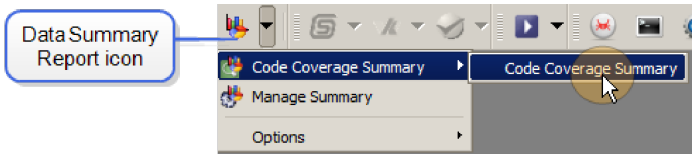


See the *Enterprise Testing with VectorCAST User Guide* for more detailed information on the Manage Summary table.

View Code Coverage Summary

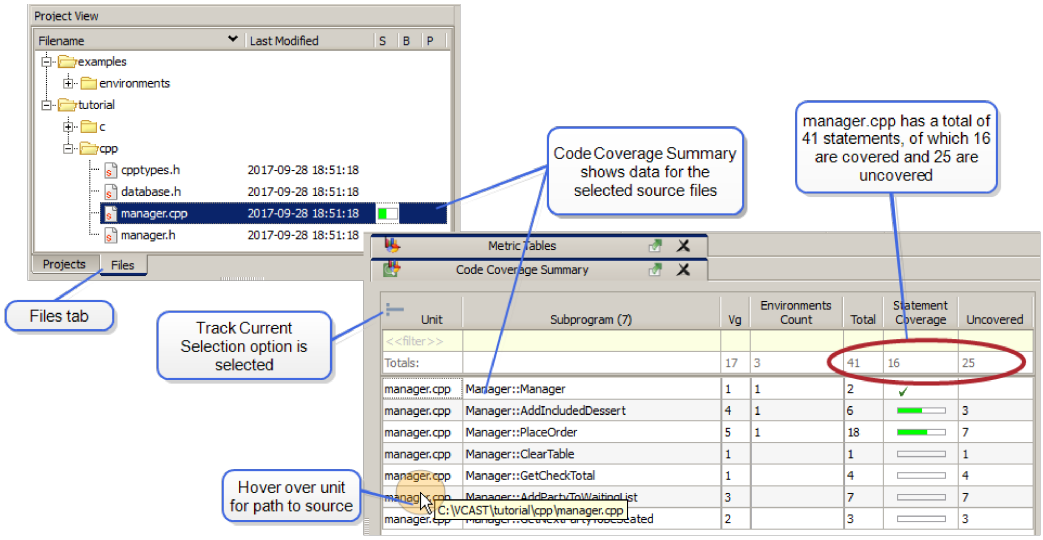
The Code Coverage Summary table allows you to easily view coverage information for all the environments of a VectorCAST project. To open the summary from the Toolbar, select **Code Coverage Summary => Code Coverage Summary** from the Data Summary Report icon drop-down

menu.



Alternatively, from the Menu Bar, select **Coverage => Code Coverage Summary => Code Coverage Summary**.

On initial open, the Code Coverage Summary table displays coverage data for all environments in the VectorCAST project. By default, the contents of the Code Coverage Summary reflect the source files selected in the Files tab. Use the Files tab to select data of interest to be displayed in the summary. In our example, we have selected the manager.cpp source file in the Files tab. A tracking icon is displayed at the top of the Unit column indicating that the Code Coverage Summary is currently tracking selected source files.



The Code Coverage Summary table is dynamic. When the **Track Current Selection** option is enabled, as units are selected and deselected in the Files tab, the Code Coverage Summary table updates in real time reflecting the selections. Try selecting different source files and note that as files are selected and deselected in the Files tab, the Code Coverage Summary updates in real time reflecting the selections.

To override tracking of selected source files, open the drop-down menu for the Data Summary Report and select **Options => Track Current Selection**. Remove the check next to the option. The tracking icon on the Summary table will change to gray to indicate that the summary is now tracking all units in the Files tab.

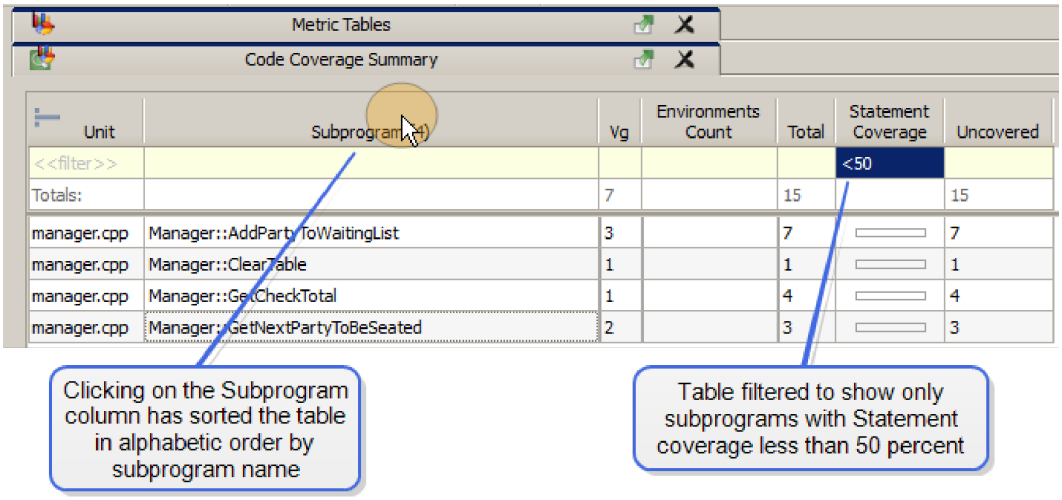
The data displayed in the Code Coverage Summary includes the Unit name, Subprogram name, Cyclomatic Complexity (Vg), the Environments Count (showing the number of environments contributing coverage to a function) and the achieved coverage for each coverage type.

The Totals row at the top of the table displays the totals for each data column. In the example above,

note that in the unit manager.cpp we have a total of 41 statements, of which 16 statements are covered and 25 are uncovered.

The Summary table updates whenever coverage data is updated. For example, the table refreshes when coverage is initialized, or following test execution.

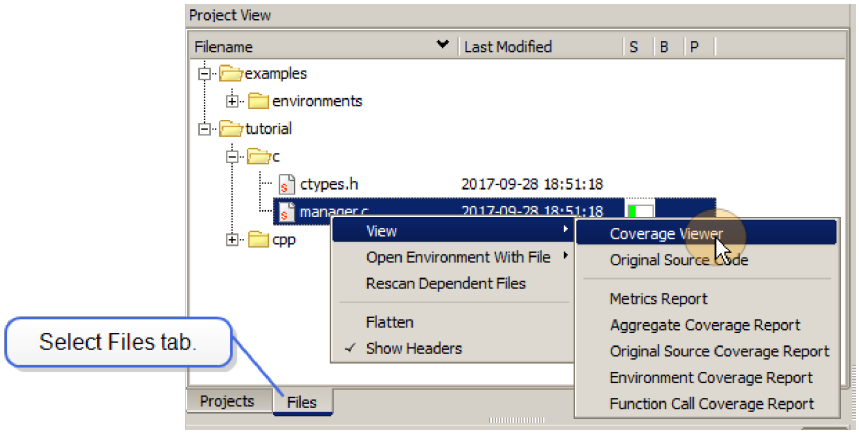
Sorting and filtering is available to locate data of interest. Sort by clicking on any column heading. Access the filter by typing into the top row of any column.



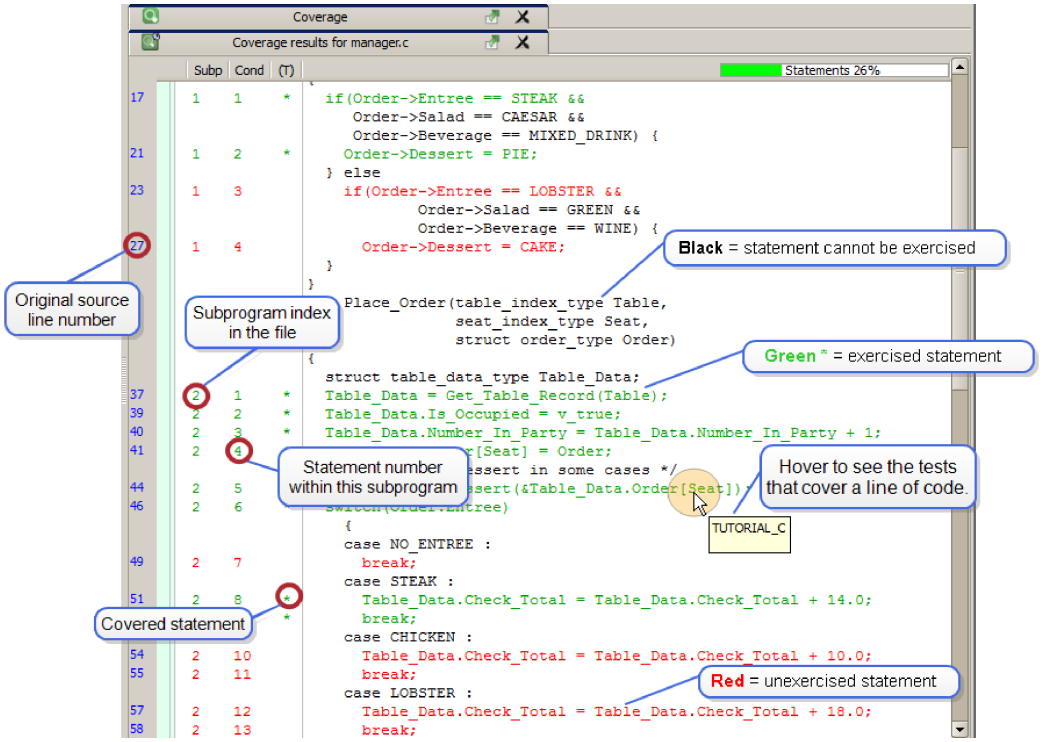
See the *Enterprise Testing with VectorCAST User Guide* for more detailed information on the Code Coverage Summary table.

View File Based Coverage

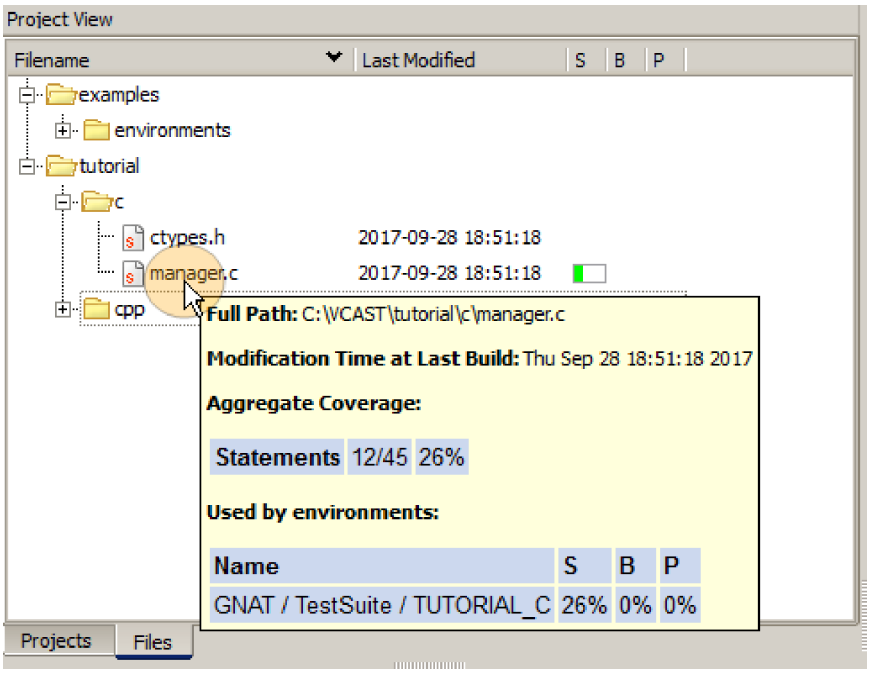
To view the File Based Coverage Results, select the Files tab in the Project Tree. Right-click on the filename and select **View =>Coverage Viewer**.



The coverage results for the selected file will open in the MDI Window. Scroll through the results for the selected file. Note the bar in the upper right indicating that manager.c has 26% Statement coverage.



For a quick view of the details of a specific file hover over the filename listed in the Files pane of the Project Tree. The at-a-glance information includes the full path for the file, date and time of the modification at the last build, an aggregate of coverage and a list of which environments use the file.



To learn more about using Enterprise Testing Projects in your organization, refer to the *Enterprise Testing with VectorCAST User Guide*. There you will find examples of common work flow scenarios

and in-depth discussion of features and functionality.

Publish Project Metrics

VectorCAST/Analytics Project Metrics

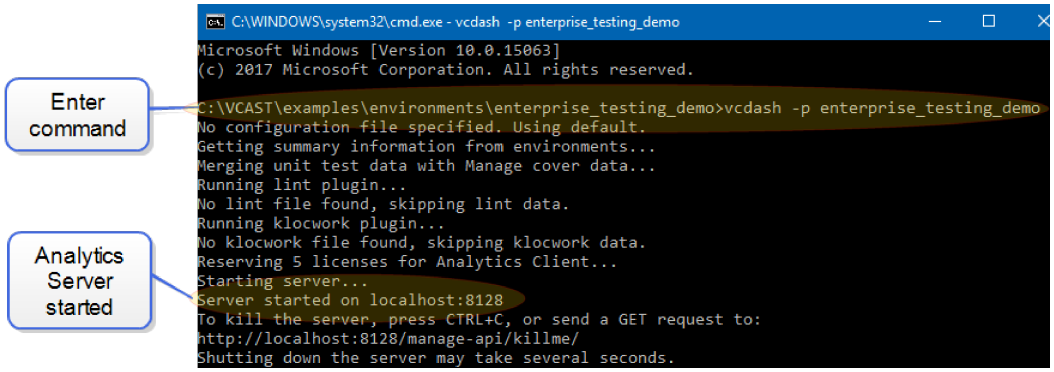
VectorCAST/Analytics provides a web-based dashboard view of software code quality and testing completeness metrics, making it easy to understand the current state of quality and testing completeness for a software project. This critical intelligence allows all stakeholders to make decisions about release readiness and process improvement.

For more detailed information on using, configuring and customizing Analytics metrics, see the *VectorCAST/Analytics User's Guide*.

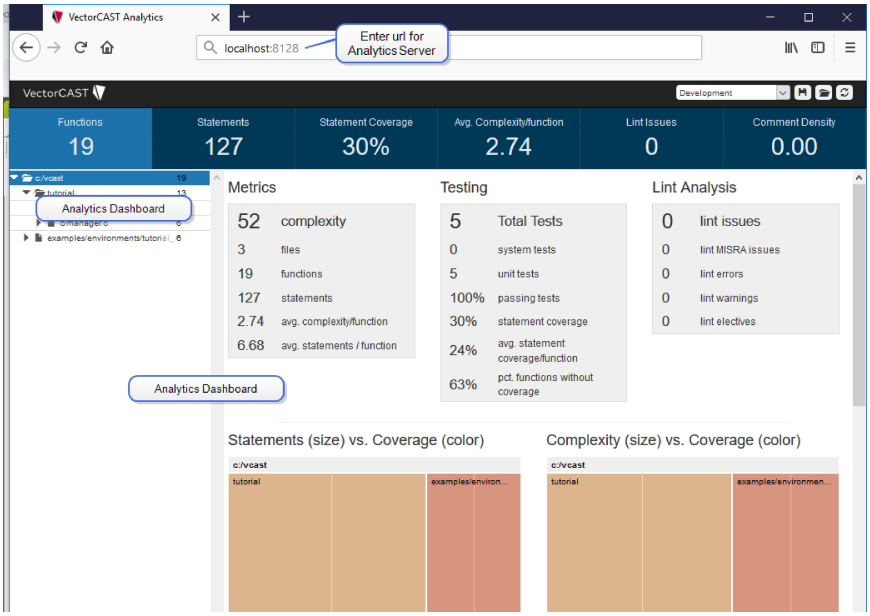
Start Analytics

To launch the Analytics Server, open a DOS command prompt by selecting the Command Prompt icon  from the Toolbar. From the command line, enter:

```
%VECTORCAST_DIR%/vcdash -p enterprise_testing_demo
```



To open the Analytics Dashboard, first open a web browser and enter the web address: **localhost:8128**. The dashboard opens in the web browser.

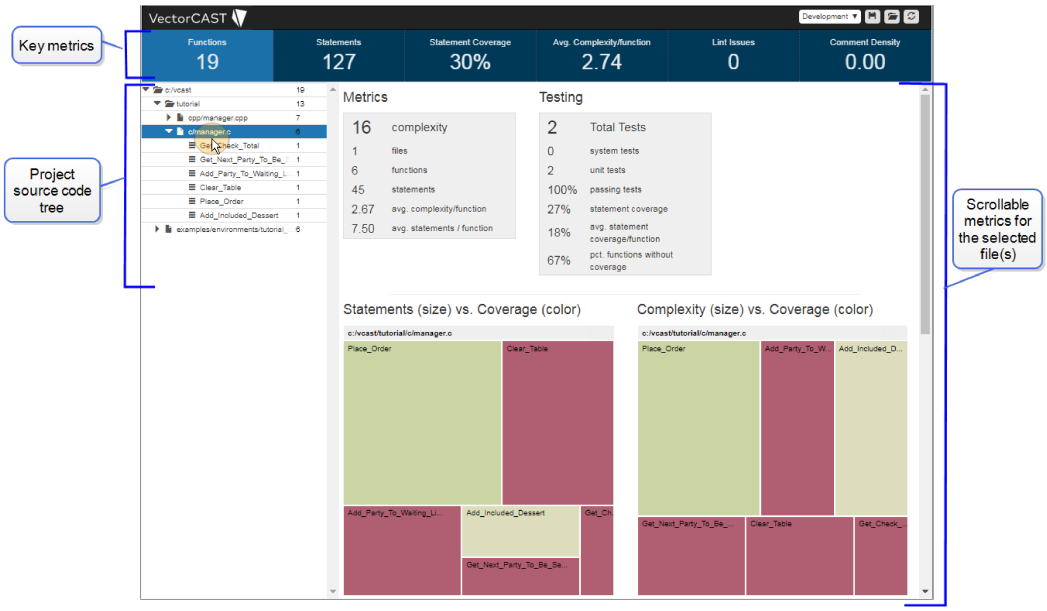


Understanding the Analytics Dashboard

For the purposes of our discussion, we have selected to view the metrics for the source file **`c/manager.c`** by clicking on the link in the Project Source Code Tree.

The Analytics Dashboard is composed of three main areas:

- Key Metrics
- Project Source Code Tree
- Metrics for selected source file(s)

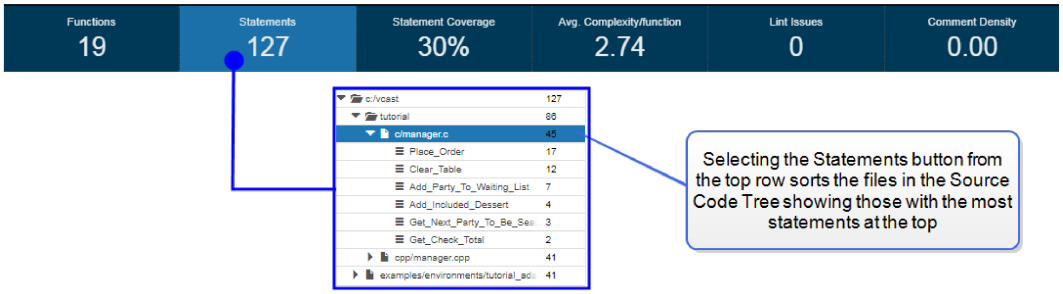


Key Metrics

Key Metrics are project-wide metrics displayed in the top bar of the dashboard, providing an at-a-glance view of the size, complexity and testing completeness of the project. Metrics include:

- Functions - total number of functions in the project
- Statements - total number of statements in the project
- Statement Coverage - percentage of statements covered in the project
- Avg. Complexity/Function - the average code complexity, or V(g), per function
- Static Analysis Issues - total number of static analysis issues in the project
- Comment Density - the percentage of comments to effective lines of code

The Key Metrics are functional buttons which control the sorting and display of the project's Source Code Tree. In our example, the **Statements** button is selected, and the source code files are listed in the Tree showing those having the greatest number of statements at the top.

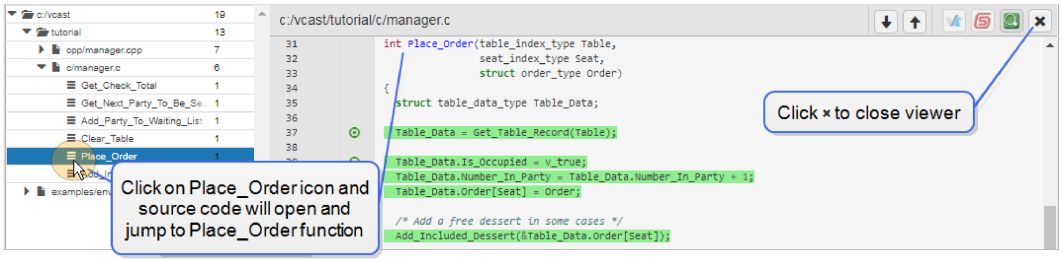


Source Code Tree

The Source Code Tree lists the source files and functions associated with the project. The selections made in the Source Code Tree control the set of functions used to calculate the metrics displayed in the dashboard. Metrics can be displayed for the entire project, or all the way down to metrics for an individual function.

The data displayed in the right column of the Source Code Tree is associated with the selected Key Metric. Select the **Statement Coverage** button, and note that the right column then shows the percentage of statement coverage achieved, listed from lowest to highest percentage.

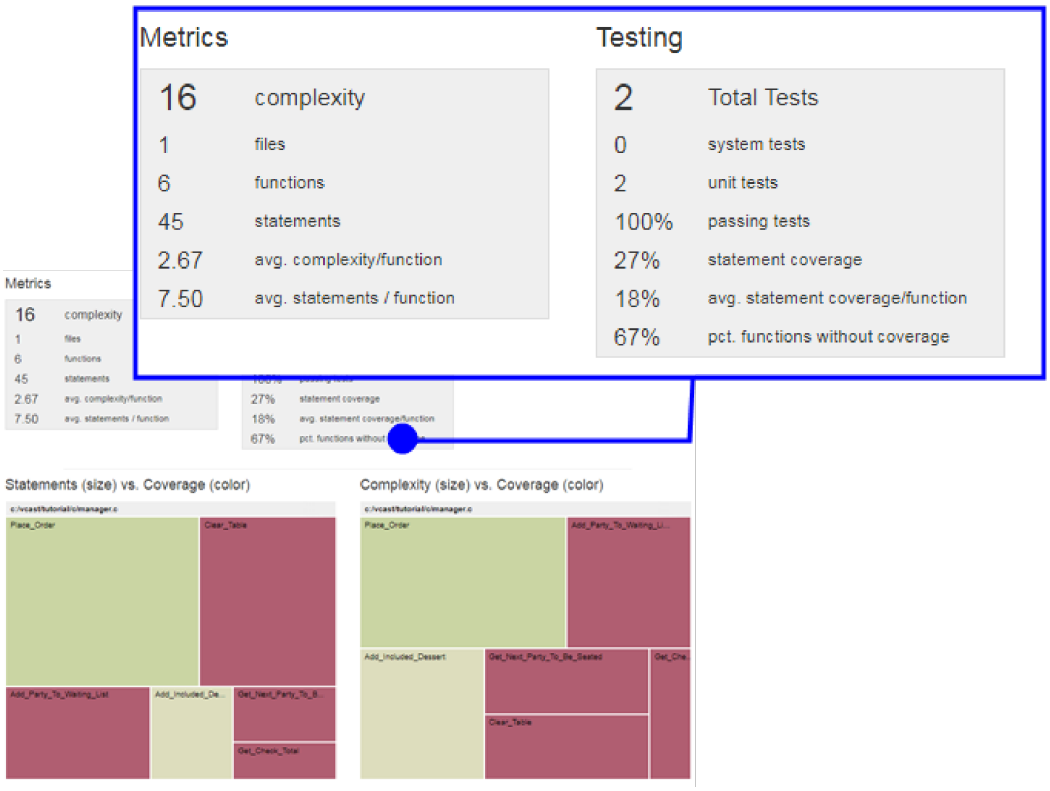
Note that the Source Code Tree uses an icon beside a file name and an icon beside a function name. Clicking on a file name icon will open the file's source code in a Viewer. Clicking on a function name icon will open the file's source code in a Viewer and will jump to the function's location in the code.



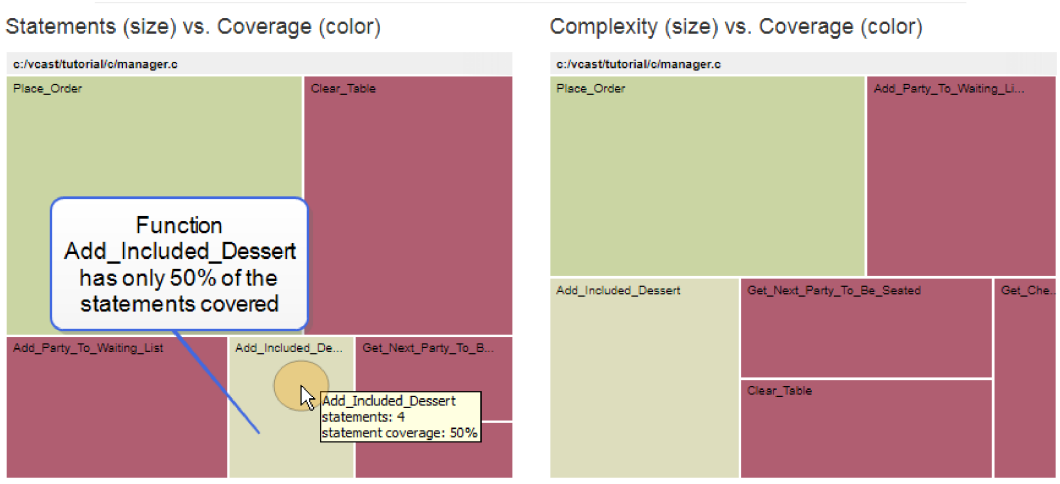
Metrics Display

Metrics are displayed in the right of the browser for the file or files selected in the Source Code Tree. In our example, we have selected the single file **`manager.c`**, and the dashboard displays the metrics associated with that file.

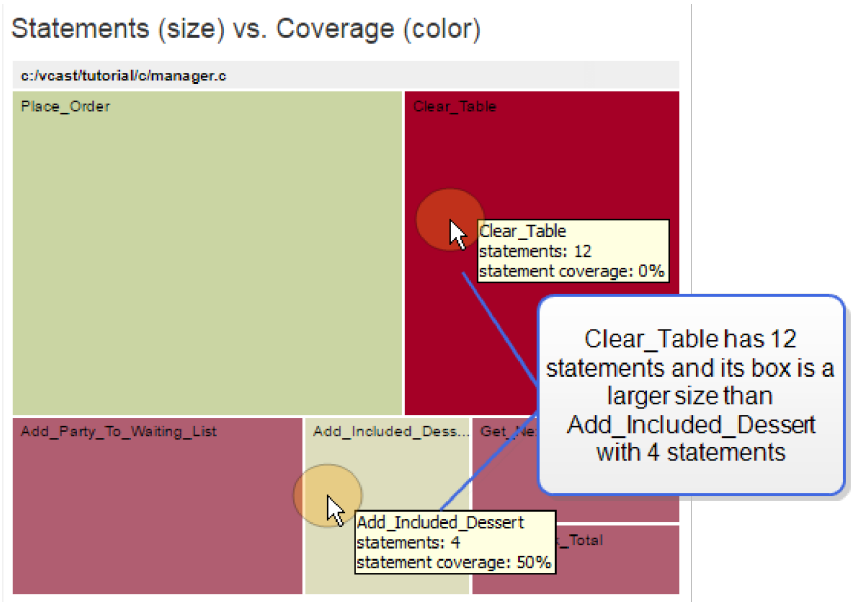
The top boxes provide the Metrics and Testing data for the selected file, **`manager.c`**. When Static Analysis data is available, a third box is provided.



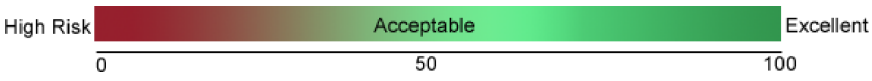
Below the boxes two treemaps are displayed for the file **`manager.c`**, one for Statements vs. Coverage and one for Complexity vs. Coverage. Each function maps to a box in the treemap. Hovering over a box displays the underlying data.



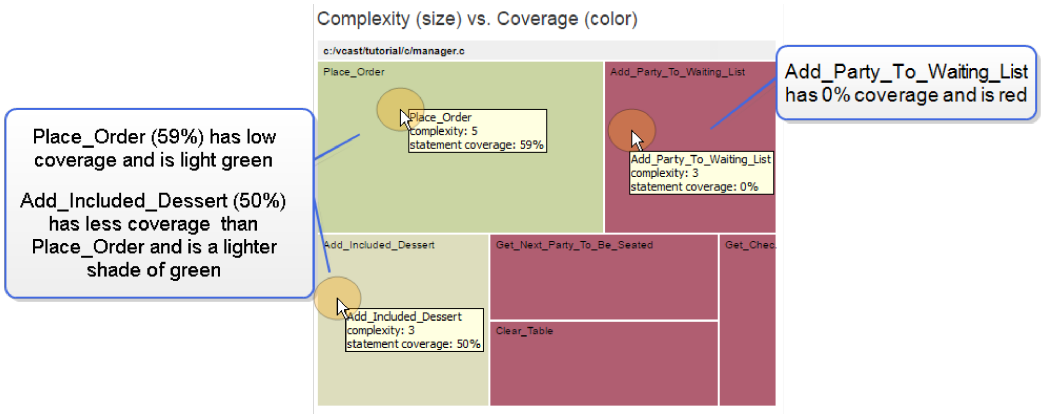
The size of the boxes within the treemaps reflects the number of statements or level of complexity of the functions. Functions with a large number of statements or high complexity will be larger in size.



The color of the boxes within the treemap indicates the level of coverage of the functions.

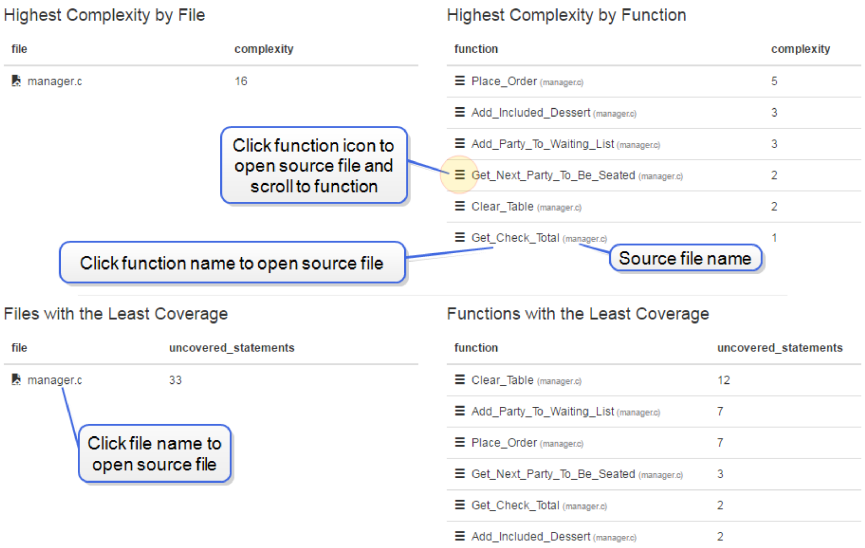


- Green indicates a high level of coverage.
- Red indicates a low level of coverage.
- Gradient shades reflect partial coverage, with less coverage and higher risk as the values approach red.



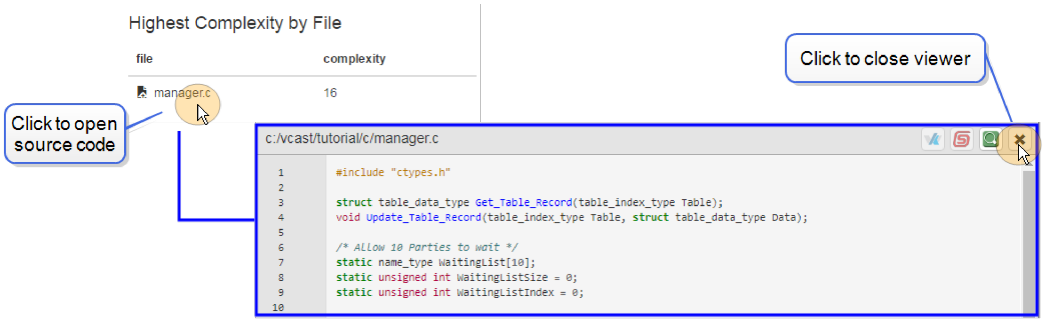
Using the example above, we can easily identify that the function Add_Party_To_Waiting_List is a hot spot which is highly complex and poorly tested. This information is critical in deciding how to best allocate testing and refactoring resources on a project.

Tabular data is located by scrolling down the browser window. A set of four tables is provided showing the Highest Complexity listed by file and by function, and the Least Coverage listed by file and by function for the selected files. In this example, we have selected to view the metrics for the manager.c unit. Note that the name of the source file is provided in parentheses to the right of the function name.



Source Code Viewer

Clicking on any of the listed source file names or the icon in either the Source Code Tree or the Metrics Tables will open the source code in a viewer. Clicking on the icon of a listed function will open the source file and scroll to the function. Use the button in the Title Bar to close the viewer.

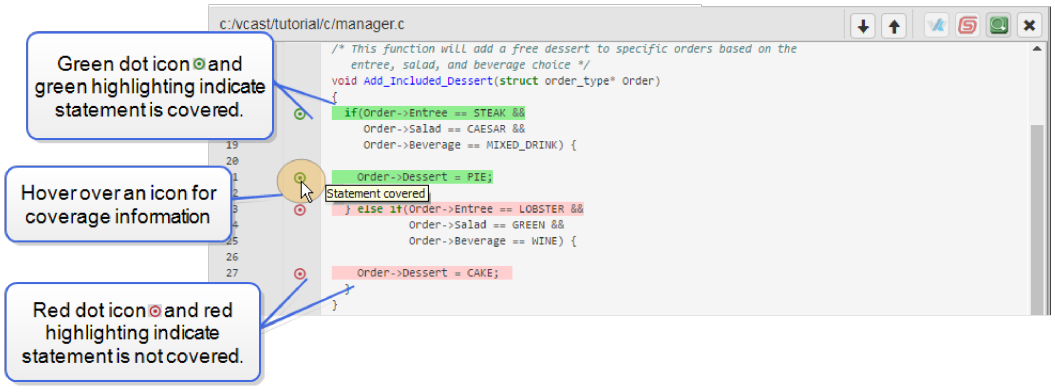


Coverage Viewer

Select one of the following buttons from the upper right of the Source Code Viewer's Title Bar to view coverage for the source file:

- Opens the Coverage Viewer. This button is only available when the selected file has covered branches, pairs or statements.
- Opens Lint Static Analysis results. This button is only available when the selected file has Lint Static Analysis results.
- Opens Klocwork Analysis results. This button is only available when the selected file has Klockwork Analysis results.
- Closes the Viewer and returns to the Analytics dashboard.
- Jumps to the next uncovered or partially covered line.
- Jumps to the previous uncovered or partially covered line.

The Coverage Viewer provides an annotated version of the source file, colorized to indicate the coverage level achieved. Green highlighted code indicates the line is covered. Red highlighted code indicates the line is not covered. Yellow highlighted code indicates partial coverage for the line. In the example below, the file **`manager.c`** shows Statement coverage:



Icons in the column on the left give additional information regarding coverage. Hover over an icon for more information. The following icons (in combination with the red, green and yellow line highlighting) are used to annotate the coverage level:

For Statement coverage:

 - Statement covered

 - Statement not covered

For Branch coverage:

 - True covered (No false branch)

 - True and False covered

 - True covered, False not covered

 - True not covered, False covered

 - Neither True nor False covered

For Merged coverage data:

 - All covered

 - Not covered

 - Partial coverage

Close the Analytics Server

To close the Analytics Server, return to the DOS command prompt, and from the command line enter **Ctrl +C**.

Index