User Guide for FireEye

1 Overview

FireEye is a combinatorial testing tool that can be used to generate *t*-way test sets. Combinatorial testing can effectively detect faults that are caused by unexpected interactions among different contributing factors. In this section, we provide an overview of the major features of FireEye.

1.1 T-Way Test Set Generation

This is the core feature of FireEye. A system (configuration) is specified by a set of parameters, each of which takes a set of values. A test set is a *t*-way test set if it satisfies the following property: Given any *t* parameters (out of all the parameters in a system), every combination of the values of those *t* parameters are covered in at least one test in the test set.

Currently, FireEye supports *t*-way test set generation for $2 \leq t \leq 6$. (Empirical results show that t = 6 is sufficient in practice.) FireEye also supports several test generation algorithms developed by the ACTS group, namely IPOG, IPOG-D, IPOG-F, IPOG-F2, and PaintBall. In general, IPOG, IPOG-F, and IPOG-F2 work best for systems of moderate size (less than 20 parameters and 10 values per parameter on average), while IPOG-D and PaintBall are preferred for large systems.

FireEye also supports two test generation modes, namely, *scratch* and *extend*. The former allows a test set to be built from scratch, whereas the latter allows a test set to be built by extending an existing test set. In the *extend* mode, an existing test set can be a test set that is generated by FireEye, but is incomplete because of some newly added parameters and values, or a test set that is supplied by the user and imported into FireEye. Extending an existing test set can help save some earlier effort that has already been spent in the testing process.

1.2 Mixed Strength

This feature allows different parameter groups to be created and covered with different strengths. For example, consider a system consisting of 10 parameters, P1, P2, …, and P10. A default relation can be created that consists of all the parameters with strength 2. Then, additional relations can be created if some parameters are found to have a higher degree of interaction, based on the user's domain knowledge. For instance, a relation could be created that includes P2, P4, P5, P7 with strength 4 if the four parameters could potentially interact with each other, and their 4-way interaction may trigger certain software faults.

FireEye allows arbitrary parameter relations to be created, where different relations may overlap or subsume each other. In the latter case, relations that are subsumed by other relations will be automatically ignored by the test generation engine.

1.3 Constraints Support

Some combinations are not valid from the domain semantics, and must be excluded from the resulting test set. For example, in a mortgage application management system, the household income of an applicant may have to be equal to or greater than a certain threshold value to be eligible for a certain type of mortgage loan. A test including an invalid combination will be rejected by the system (if adequate input validation is performed) or may cause the system to fail. In either case, the test will not be executed properly, which may compromise test coverage, if some (valid) combinations are only covered by this test.

FireEye allows the user to specify constraints that combinations must satisfy to be valid. The specified constraints will be taken into account during test generation so that the resulting test set will cover, and only cover, those combinations that satisfy those constraints.

1.4 Coverage Verification

This feature verifies whether a test set satisfies *t*-way coverage, i.e. whether it covers all *t*-way combinations. A test set to be verified can be a test set generated by FireEye or a test set supplied by the user.

2 General Layout of GUI

Figs. 1 and 2 show the general layout of the FireEye GUI. The System View component is a tree structure that shows the configurations of the systems that are currently open in the system. In the tree structure, each system is shown as a three-level hierarchy. That is, each system (top level) consists of a set of parameters (second level), each of which has a set of values (leaf level). If a system has relations and constraints, the relations and constraints will be shown in the second level, i.e. the same level as the parameters.

Right to the System View is a tabbed pane consisting of two tabs, namely, Test Result, which is shown in Fig. 1, and Statistics, which is shown in Fig. 2. The Test Result shows a test set of the currently selected system, where each row represents a test, and each column represents a parameter. The Statistics tab displays some relevant statistical information about the test set. In particular, it includes a graph that plots the growth rate of the test coverage with respect to the tests in the test set displayed in the Test Result tab. Note that drawing the graph may involve expensive computations, and thus the graph is shown only on demand, i.e. when the Graph button is clicked.

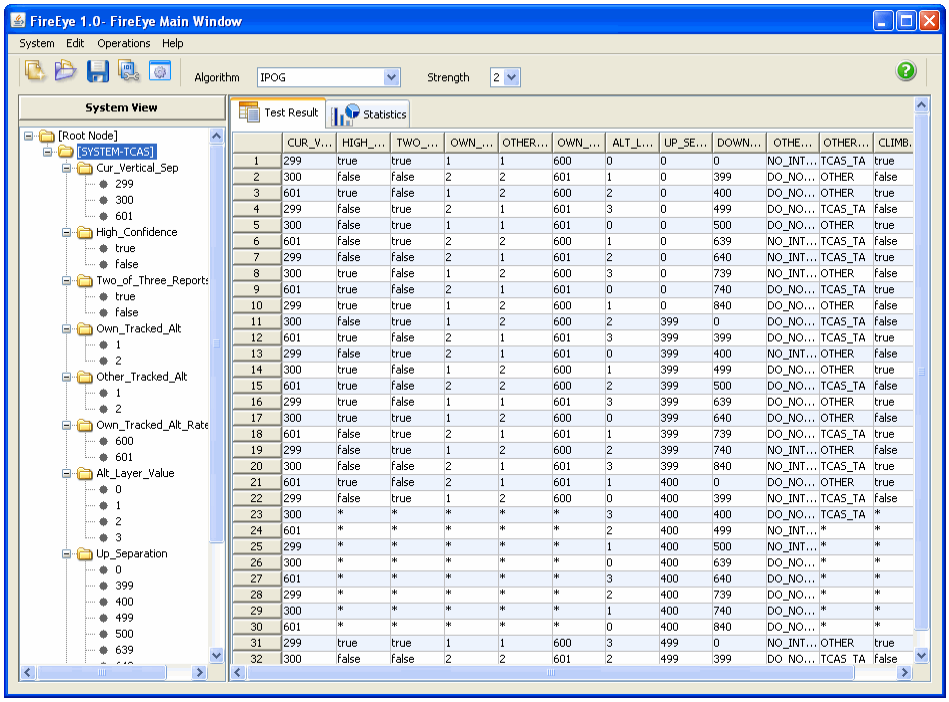

Figure 1. The Main Window – Test Result Tab

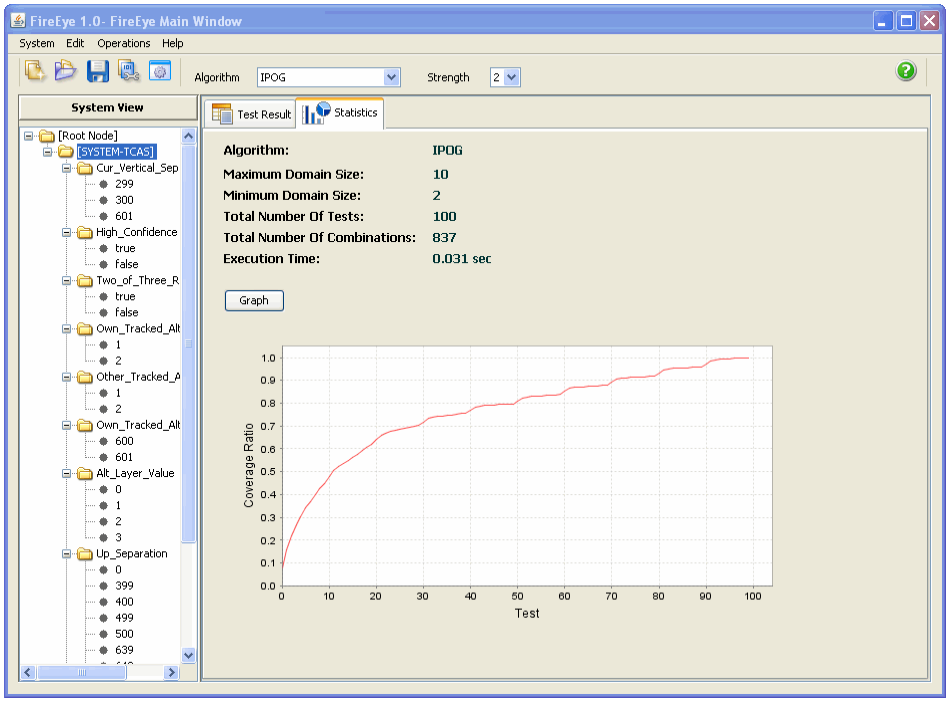

Figure 2. The Main Window - Statistics Tab

3 Major Operations

3.1 Create New System

To create a new system, select menu System → New, or the first icon in the toolbar, to open the New System window. The New System window contains a tabbed pane of three tabs, namely, Parameters, Relations, and Constraints. The three tabs are shown in Figs. 3, 4, and 5, respectively.

The Parameters tab (Fig. 3) allows the user to specify the parameters, as well as the values of those parameters, in the new system. Currently, four parameter types are supported, Boolean, Enum, Number, and Range. For both Number and Range, only integers are supported. Note that Range is a convenience feature that allows multiple, consecutive integers to be input quickly.

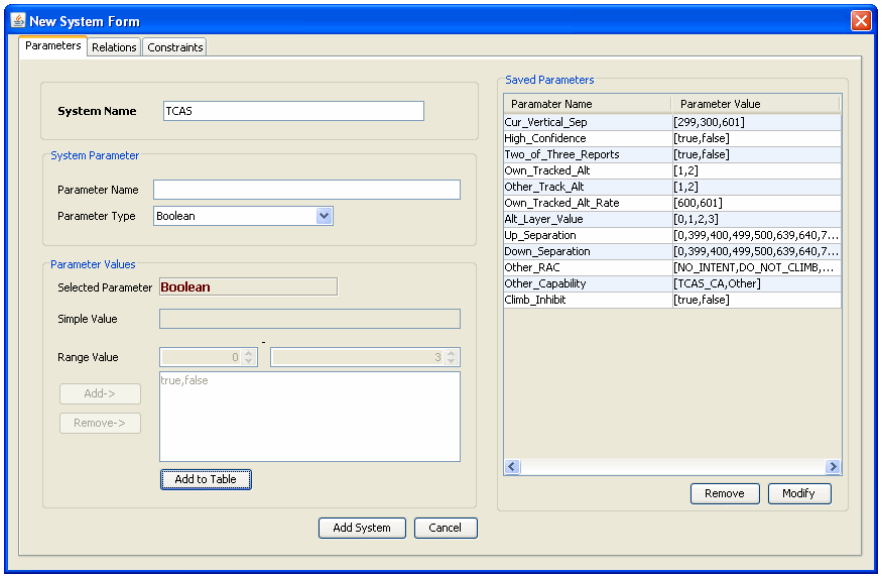

Figure 3. New System Window – Parameters

The Relations tab (Fig. 4) allows the user to create parameter groups with different strengths. The Parameters list (on the left side) displays all the parameters that have been specified in the Parameters tab, the Strength field allows a strength to be specified, and the table on the right side lists all the relations that have already been created. To define a new relation, the user can select one or more parameters in the Parameters list, specifies a strength, and then click the Add button in the middle. Note that multiple selections can be made in the Parameters list by pressing Ctrl (for non-consecutive selections) or Shift (for consecutive selections) during selection.

Note that a default relation is automatically created that consists of all the parameters that have been specified in the Parameters tab with the default strength (which is specified in the Options window, see Section 3.2). This default relation is provided as a convenience feature (so that the user does not need to do anything in this tab if the user does not want to specify any relation), and can be removed like other user-defined relations.

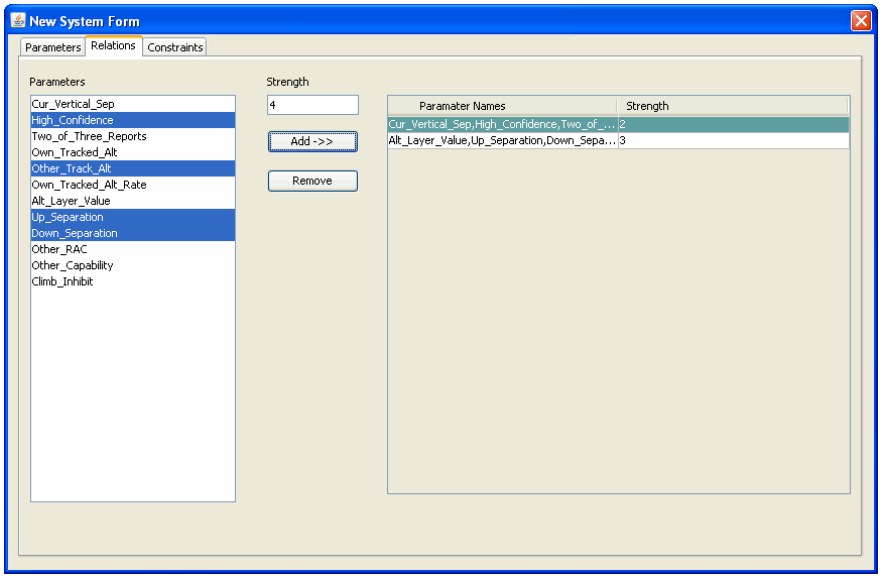

Figure 4. New System Window – Relations

The Constraints tab (Fig. 5) allows the user to specify constraints to exclude invalid combinations from the resulting test set. The constraints are, in general, in the form of a predicate logic formula, where each component can be a relational expression that are built using parameters, parameter values, and some arithmetic operators. The Palette group contains a set of buttons, which, from left to right, can be used to add, respectively, parameters, parameter values, left parenthesis, right parenthesis, relational operators, arithmetic operators, and digits, into a constraint expression. The Expression Editor group contains a text box that displays the current constraint expression being built. Note that "backspace" can be typed directly from the keyboard to erase part of the current expression. The Added Constraints group contains a list that displays all the constraints that have been added into the system.

Note that currently we only support binary constraints, i.e. constraints that involve at most two parameters. The user can add constraints involving three or more parameters, but the system may or may not produce the correct result. (In theory, any constraint involving three or more parameters can be transformed to binary constraints.) Full support for constraints involving more than three parameters will be made available in the next release of FireEye.

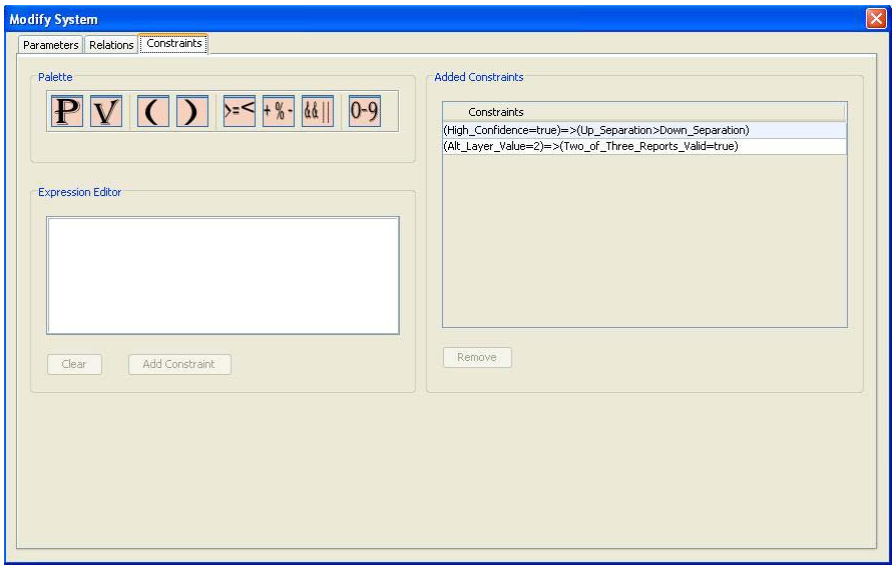

Figure 5. New System Window - Constraints

3.2 Build Test Set

To build a test set for a system that is currently open, select the system in the System View, and then select menu Operations → Build. The latter selection brings up the Options window, as shown in Fig. 6, which allows the following options to be specified for the build operation:

- *Algorithm*: This option decides which algorithm to be used for test generation. As mentioned in Section 0, IPOG, IPOG-F, and IPOG-F2 work best for systems of moderate size, while IPOG-D and PaintBall are preferred for large systems. Also note that relations and constraints are currently only supported for algorithm IPOG.
- *Max Tries*: This option is used by algorithm PaintBall, and specifies the number of candidates to be generated randomly at each step.
- *Randomize Don't Care Values*: If this option is checked, then all the *don't care* values (i.e. "*") in the resulting test set will be replaced with a random value.
- *Strength*: This option specifies the default strength of the test set.
- *Mode*: This option can be Scratch or Extend. The former specifies that a test set should be built from scratch; the latter specifies that a test set should be built by extending the existing test set (i.e. the one shown in the Test Result tab).
- *Progress*: If this option is turned on, progress information will be displayed in the console.

After the build operation is completed, the resulting test set will be displayed in the Test Result tab of the Main window.

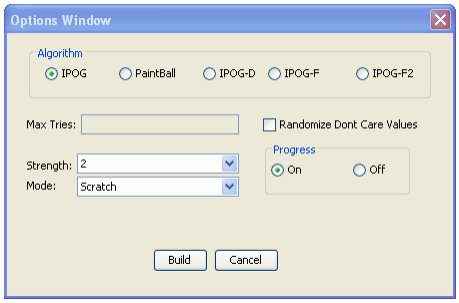

Figure 6. Build Options Window

3.3 Modify System

To modify an existing system, select the system in the tree view, and then select menu Edit → Modify. The Modify System window is the same as the New System window except that the name of the system cannot be changed. Note that a parameter cannot be removed if it is involved in a relation or constraint. In this case, the relation or constraint must be removed first before the parameter is removed.

Note that a system can also be modified through the tree view. For example, a parameter, or value, or relation, or constraint can be removed by first selecting the parameter, or value, or relation, or constraint, and then selecting menu Edit → Delete.

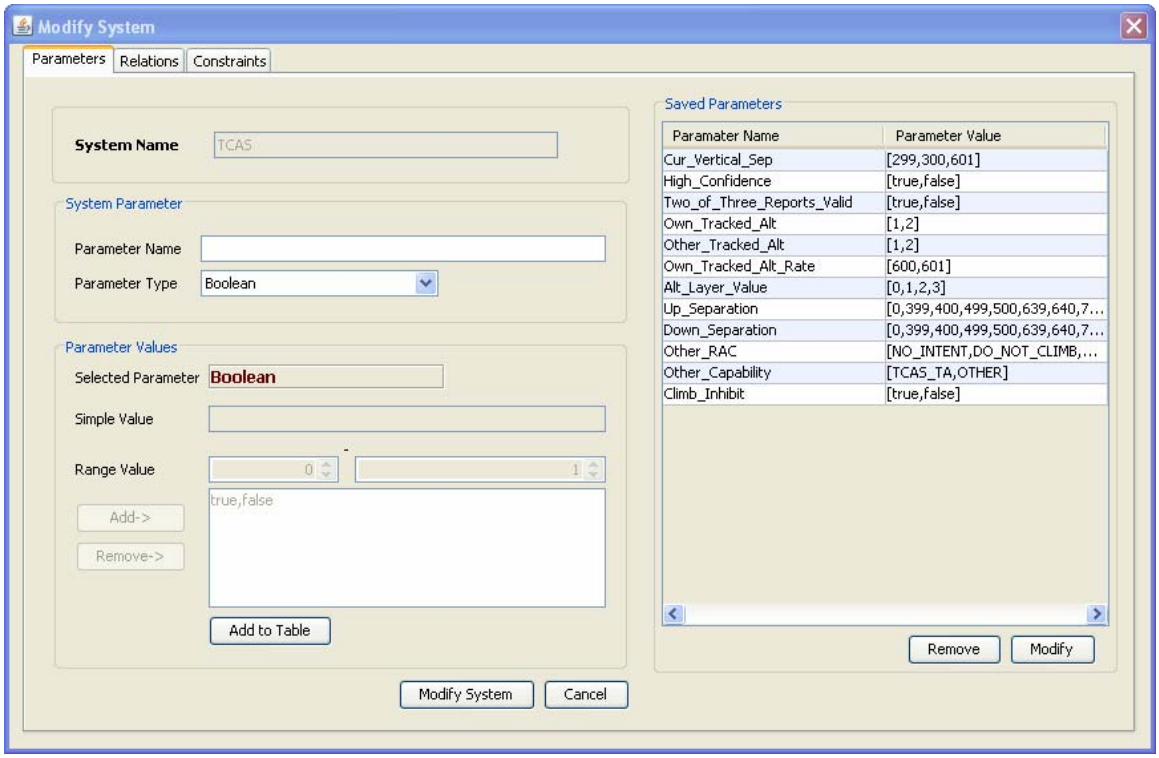

Figure 7. Modify System Window

3.4 Save/Save As/Open System

To save an existing system, select the system in the tree view, and then select menu System → Save or Save As. When Save As is selected, a standard file dialog will be brought up, where the user can specify the name of the file to be saved, as shown in Fig. 8.

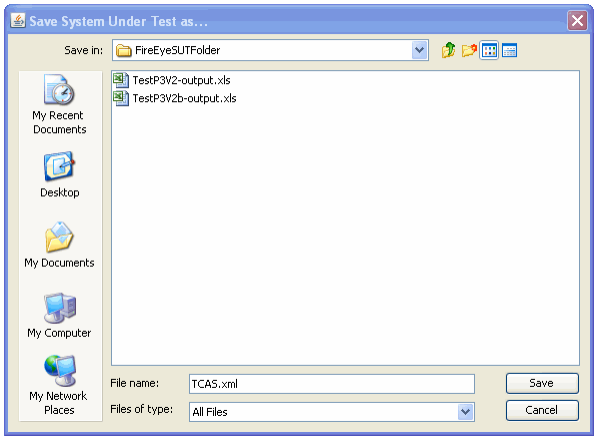

Figure 8. Save As Window

3.5 Import/Export Test Set

To import a test set of a system, first create the system in the GUI as described in 3.1. Then, select menu Operations → Import, which will bring up the window as shown in Fig. 9. After the file that contains the test set is selected, the test set will be imported and displayed in the Test Result tab of the Main window. Note that in the test set file, the test set must be formatted such that each row represents a test, and each column represents a parameter, without an explicit row or column header. The parameter values in each row must be a valid value of the corresponding parameters and must be separated by "," or ";".

To export a test set that exists in the GUI, first select the corresponding system so that the test set is displayed in the Test Result tab of the Main window, and then select Operations -> Export. Currently, two formats are supported, namely, NIST Format, and Numeric Format. A snippet of an exported test set in the NIST format is shown below:

Degree of interaction coverage: 2
Number of parameters: 12
Maximum number of values per parameter: 10
Number of configurations: 100

Configuration #1:

1 = Cur_Vertical_Sep=299

2 = High_Confidence=true
3 = Two_of_Three_Reports_Valid=true
4 = Own_Tracked_Alt=1
5 = Other_Tracked_Alt=1
6 = Own_Tracked_Alt_Rate=600
7 = Alt_Layer_Value=0
8 = Up_Separation=0
9 = Down_Separation=0
10 = Other_RAC=NO_INTENT
11 = Other_Capability=TCAS_TA
12 = Climb_Inhibit=true

Configuration #2:

….

Degree of interaction coverage: 2
Number of parameters: 12
Number of tests: 100

---

0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0 1 1 1 1
2 0 1 0 1 0 2 0 2 2 1 0
0 1 0 1 0 1 3 0 3 1 0 1
1 1 0 0 0 1 0 0 4 2 1 0
2 1 0 1 1 0 1 0 5 0 0 1
0 1 1 1 0 1 2 0 6 0 0 0
1 0 1 0 1 0 3 0 7 0 1 1
2 0 1 1 0 1 0 0 8 1 0 0
…

A snippet of the Numeric format is shown below:

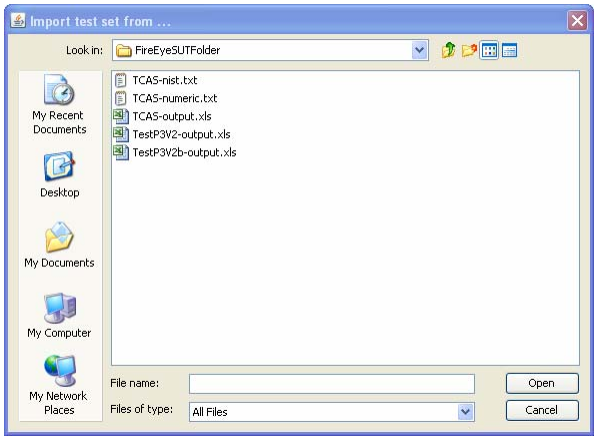

Figure 9. Import Test Set Window

3.6 Verify T-Way Coverage

To verify the *t*-way coverage of a test set, first import the test set as described in Section 3.5. Select menu Operations → Options, and specify a desired strength in the Options window. Then, select menu Operations → Verify. If the test set achieves the coverage for the specified strength, the message window in Fig. 10 will be displayed; otherwise, the message window in Fig. 11 will be displayed.

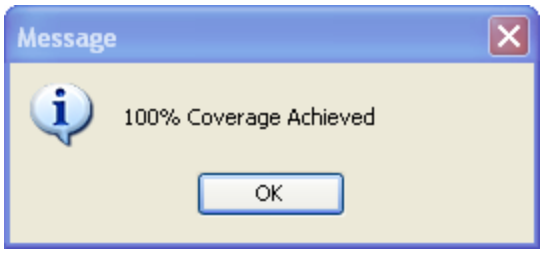

Figure 10. Coverage Achieved Window

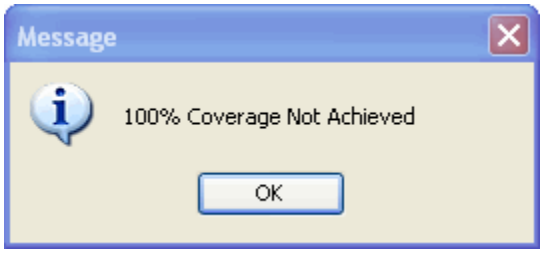

Figure 11. Coverage Not Achieved Window