

# Viewing Reports

Open the report.html or report.pdf file saved to the working directory or location specified with the `-report` switch. Reports may contain different sections depending on the type of analysis, but the following sections are included in all static and flow analysis configurations.

## Header

# Parasoft dotTEST Report

dotTEST 10.4.2.114

**Session Summary**

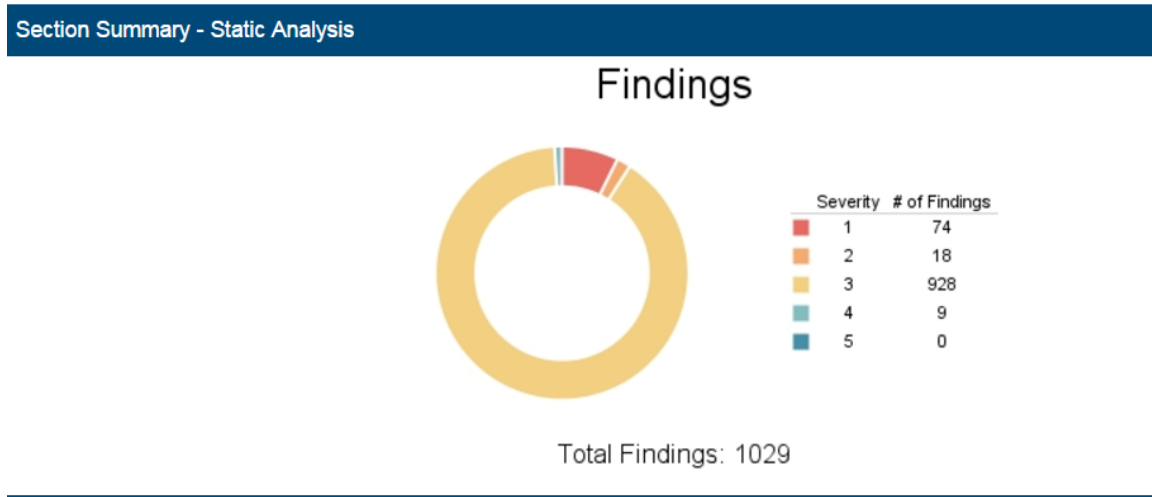
<b>Build ID:</b>	2019-03-20	<b>Static Analysis</b> Severity 1 Findings: 37 <b>Test Execution</b> Test Failures: 11/29
<b>Test Configuration:</b>	user://All analysis	
<b>Started:</b>	2019-03-20T08:54:46+01:00	
<b>Performed on:</b>	lilac by hanna	
<b>Session Tag:</b>	\${scontrol_branch}-win32_x86_64	
<b>Project:</b>		

The following information is included:

- Tool used for the analysis
- Build ID
- Test configuration
- Time stamp of the analysis
- Machine name and user name
- Session tag
- Project name
- Number of findings with the highest severity
- Number of failed tests

## Static Analysis

The first part of the report covers the Static Analysis findings and is divided into two main sections. The first section is a summary which shows an overview of findings displayed as a pie chart. The colors indicate different severity types and their corresponding number of findings detected during static analysis:



The second section shows the details of static analysis findings. It starts with a table which includes static analysis results:

# Static Analysis

Module	Findings			Files		Lines	
	suppressed	total	per 10,000 lines	checked	total	checked	total
com.parasoft.demo	1	1029	2525	70	70	4074	4074
<b>Total [0:00:14]</b>	<b>1</b>	<b>1029</b>	<b>2525</b>	<b>70</b>	<b>70</b>	<b>4074</b>	<b>4074</b>

The following information is included:

- Name of module
- Number of suppressed rules
- Total number of findings
- Average number of findings per 10,000 lines
- Number of analyzed files
- Total number of files in the module
- Number of code lines analyzed
- Total number of code lines in the module

## All Findings

The All Findings section displays the details of findings organized by category or severity. Click the **Severity** or **Category** link to toggle between views.

In the Category view, findings are reported by rule and grouped by category. A count of how many times each rule was violated in the scope of analysis is also shown.

### All Findings by Category

[Category](#) | [Severity](#)

- [12] **Possible Bugs** (BD.PB)
  - [2] Avoid use before explicit initialization (BD.PB.NOTEXPLINIT-1)
  - [1] Avoid use of fields before initialization in constructors and static initializers (BD.PB.NOTINITCTOR-1)
  - [1] Do not append null value to strings (BD.PB.STRNULL-1)
  - [1] Avoid division by zero (BD.PB.ZERO-1)
  - [7] Avoid conditions that always evaluate to the same value (BD.PB.CC-2)
- [5] **Optimization** (BD.OPT)
  - [3] Avoid inefficient removal of Collection elements (BD.OPT.INEFCOL-3)
  - [1] Avoid inefficient iteration over Map entries (BD.OPT.INEFMAP-3)
  - [1] Avoid inefficient removal of Map entries (BD.OPT.INEFMAPRM-3)

In the Severity view, findings are reported and grouped by severity. A count of findings per severity is also included.

### All Findings by Severity

[Category](#) | [Severity](#)

- [32] **Severity 1 - Highest**
  - [22] Constructors allowing for conversion should be made explicit (CODSTA-CPP-04-1)
  - [4] Never allow an exception to be thrown from a destructor, deallocation, and swap (EXCEPT-01-1)
  - [1] All member variables should be initialized in constructor (INIT-06-1)
  - [1] Declare a copy assignment operator for classes with dynamically allocated memory (MRM-37-1)
  - [1] Declare a copy constructor for classes with dynamically allocated memory (MRM-38-1)
  - [1] Be wary about using multiple inheritance of classes that are not abstract interfaces (OOP-07-1)
  - [1] Do not directly access global data from a constructor (OOP-08-1)
  - [1] Make destructors virtual in base classes (OOP-24-1)
- [156] **Severity 2 - High**
  - [13] Declare at least one constructor to prevent the compiler from doing so (CODSTA-CPP-19-2)
  - [12] The #include directive shall use the <filename.h> notation to include header files (JSF-033-2)
  - [7] Identifiers for constant and enumerator values shall be lowercase (JSF-052-2)
  - [3] The statements forming the body of an 'if', 'else if' or 'else' statement shall always be enclosed in braces (JSF-059\_b-2)

You can't toggle between these sections in the PDF versions of the report and they are published separately.

## Findings by Author

This section includes a table of authors associated with the analyzed code and a count of findings per each author. Findings are segmented into findings associated with suppressed rules and findings recommended for remediation. Click on an author link to view their finding details.

## Findings by Author

[Back to Top](#)

Author	Findings		
	suppressed	total	recommended
<a href="#">hanna</a>	0	26	26

[hanna](#) Total Findings : 26

[Back to Top](#)

**BankExample/Parasoft.Dottest.Examples.Bank/Money/Currency.cs**

49: Replace equality comparison of floating-point types to comparison which handles floating-point uncertainty.

PB.DNCF-2

**BankExample/Parasoft.Dottest.Examples.Bank/Money/CurrencyProvider.cs**

23: Provide 'default' for 'switch' statement

CS.PB.DEFSWITCH-2

38: Provide 'default' for 'switch' statement

CS.PB.DEFSWITCH-2

The details view includes the following information:

- File containing the finding and its location
- Violation message and rule
- Flow analysis reports also mark the cause of the violation (C), violation points (P), thrown exceptions (E), and important data flows (!)

## Findings by File

You can navigate the analyzed code to the reported findings in the Findings by File section. Each node begins with a value that indicates the total number of findings in the node. The value in brackets shows the number of suppressed rules in the node. You can click nodes marked with a plus sign (+) to expand them. PDF versions of the reports are already fully expanded.

## Findings by File

[Expand All](#) [Collapse All](#) [Back to Top](#)

+ 26 (0) Total (Suppressed)

+ 26 (0) Parasoft.Dottest.Examples.Bank

+ 3 (0) AccountNumber.cs

Findings: 3

54: Change Equals() implementation to match a standard pattern. Use: if (obj == null) { return false; }	<a href="#">hanna</a>	OPU.CPNEQ-1
54: Change Equals() implementation to match a standard pattern. Use: if (this.GetType() != obj.GetType()) { return false; }	<a href="#">hanna</a>	OPU.CPTEQ-2
57: Change bitwise operator to logical operator in method 'Equals'.	<a href="#">hanna</a>	CS.PB.BITBOOL-2

## Metrics Summary

If your test configuration includes metrics analysis, a metrics section will appear in the report. See [Metrics Analysis](#) for additional information.

## Metrics Summary

[Expand All](#) [Collapse All](#) [Back to Top](#)

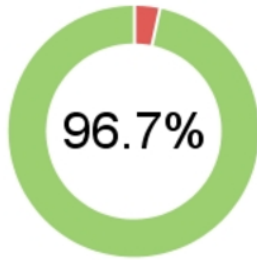
Metric name	Number of Items	Average	Std. Deviation	Maximum	Minimum
+ McCabe Cyclomatic Complexity (METRIC.CC)	359	1.507	1.122	10	1
demo	359	1.507	1.122	10	1
+ Nested Blocks Depth (METRIC.NBD)	359	0.301	0.711	8	0
demo	359	0.301	0.711	8	0
+ Number of Physical Lines in Files (METRIC.NOPLIF)	74	61.662	63.118	460	11
demo	74	61.662	63.118	460	11
+ Number of Source Lines in Methods (METRIC.NOSLIM)	359	7.036	5.443	40	1
demo	359	7.036	5.443	40	1

## Test Execution

The second part of the report covers the Test Execution results and is divided into two sections. The first section is a summary which shows an overview of test failures and coverage displayed as pie charts:

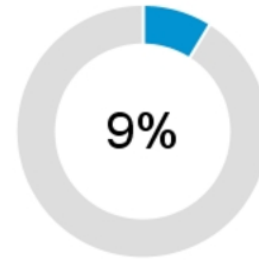
## Section Summary - Test Execution

### Tests



Test Failures: 1/30

### Coverage



Line Coverage: 9%

The second section shows the details of test execution. It starts with a table which includes test execution results and coverage information:

### Details - Test Execution

## Test Execution

[Back to Top](#)

Module	Findings			Executed Test Cases				Coverage (%)
	fix unit test problems	review exceptions	review assertion failures	passed	failed	incomplete	total	line
com.parasoft.demo	1	0	0	29	1	0	30	9
<b>Total</b> [0:00:00]	<b>1</b>	<b>0</b>	<b>0</b>	<b>29</b>	<b>1</b>	<b>0</b>	<b>30</b>	<b>9</b>

The following information is included:

- Module name
- Number of unit test problems which need to be fixed
- Number of exceptions which need to be reviewed
- Number of assertion failures which need to be reviewed
- Number of unit tests successfully executed
- Number of unit tests failures
- Number of incomplete unit tests
- Total number of unit tests
- Line coverage expressed as percentage

## All Findings

The All Findings section displays the details of all unit test problems detected during test execution:

## All Findings

[1] **Unit Test Problems**

[1] **Assertion Failures**

[1] java.lang.AssertionError

## Findings by Author

This section includes a table of authors associated with the analyzed code and shows the total number of findings for each author. Click on an author link to view their finding details.

## Findings by Author

[Back to Top](#)

Author	Findings	
	total	recommended
<a href="#">hanna</a>	5	5

[hanna](#) Total Findings : 5 [Back to Top](#)

**BankExample/Parasoft.Dottest.Examples.Bank/BankUser.cs [ 5 ]**

Test case: Parasoft.Dottest.Examples.Bank.Tests.BankUserTests.Test("foo.bar")

System.ArgumentException : parsing "[w\-\+\&!\*](?:\.[w\-\+\&!\*])\*(?:[w\-\+\&!\*])+[a-zA-Z]{2,7}\$" - Unrecognized escape sequence \\_ [failure]

System.Text.RegularExpressions.RegexParser.ScanCharEscape()

System.Text.RegularExpressions.RegexParser.ScanCharClass(Boolean caseInsensitive, Boolean scanOnly)

The details view includes the following information:

- Finding location
- Test name
- Failure message

## Executed Tests (Details)

You can view the findings in the Executed Tests (Details) section. The nodes where all the test passed are marked with "P" in square brackets. The nodes with test failures begin with a set of values in square brackets. The first value is a count of successfully passed tests and the second indicates the total number of tests executed in the node. The letter "F" indicates the final node where the test failed. You can click nodes marked with a plus sign (+) to expand them.

## Executed Tests (Details)

[Expand All](#) [Collapse All](#) [Back to Top](#)

+ **[8/13]** [0:00:00.214] **Passed / Total**

+ **[8/13]** [0:00:00.214] **Parasoft.Dottest.Examples.Bank.Tests**

+ **[P]** [0:00:00.104] **Parasoft.Dottest.Examples.Bank.Tests.AccountNumberTests.TestAccountNumberCreation**

+ **[P]** [0:00:00.010] **Parasoft.Dottest.Examples.Bank.Tests.AccountNumberTests.TestAccountNumberHashing**

+ **[P]** [0:00:00.010] **Parasoft.Dottest.Examples.Bank.Tests.AccountNumberTests.TestInvalidAccountNumberCreation**

+ **[F]** [0:00:00.056] **Parasoft.Dottest.Examples.Bank.Tests.BankUserTests.Test**

+ **[F]** [0:00:00.056] **Parasoft.Dottest.Examples.Bank.Tests.BankUserTests.Test**

+ **[F]** **Parasoft.Dottest.Examples.Bank.Tests.BankUserTests.Test("foo.bar")**

System.ArgumentException : parsing "[w\-\+\&!\*](?:\.[w\-\+\&!\*])\*(?:[w\-\+\&!\*])+[a-zA-Z]{2,7}\$" - Unrecognized escape sequence \\_ [failure] hanna

System.Text.RegularExpressions.RegexParser.ScanCharEscape()

System.Text.RegularExpressions.RegexParser.ScanCharClass(Boolean caseInsensitive, Boolean scanOnly)

System.Text.RegularExpressions.RegexParser.CountCaptures()

System.Text.RegularExpressions.RegexParser.Parse(String re, RegexOptions op)

## Coverage

This section shows the details of coverage collected during the test execution. Each node starts with a set of values. The first value shows coverage expressed as a percentage. The second value is a count of the number of lines in the node which were covered during the test execution. The third value indicates the total number of lines in the node. You can click nodes marked with a plus sign (+) to expand them.

## Coverage

[Expand All](#) [Collapse All](#) [Back to Top](#)

+ **Total** [9% 126/1423 executable lines]

+ **com.parasoft:demo** [9% 126/1423 executable lines]

+ **src/examples** [9% 126/1423 executable lines]

+ **eval** [0% 0/10 executable lines]

+ **flownalysis** [0% 0/489 executable lines]

+ **junit4** [88% 88/100 executable lines]

+ **Money.java** [90% 27/30 executable lines]

+ **Money** [90% 27/30 executable lines]

**Money( int, java.lang.String )** [100% 4/4 executable lines]

**add( examples.junit4.IMoney )** [100% 1/1 executable lines]

**addMoney( examples.junit4.Money )** [100% 3/3 executable lines]

**addMoneyBag( examples.junit4.MoneyBag )** [100% 1/1 executable lines]

## Test Parameters

The arguments specified during analysis are shown in the Test Parameters section

## Test Parameters

---

```
dottestcli -solution C:\dotTEST\Examples_10.2.3.137\VS2013\030_NUnit\BankExample\BankExample.sln -config user://Run NUnit Tests with coverage -report C:\dotTEST\Examples_10.2.3.137\VS2013\030_NUnit\020_Run_NUnit_Tests_With_Coverage_Report -out C:\dotTEST\Examples_10.2.3.137\VS2013\030_NUnit\020_Run_NUnit_Tests_With_Coverage_Output.txt
```