Parasoft Findings for Jenkins 10.3

In this section:

Introduction

The Parasoft Findings Plugin for Jenkins allows you to visualize static analysis and test results in Jenkins. It converts XML reports generated by Parasoft products into trend graphs and enables you to conveniently view the details or easily navigate to rule documentation. The plugin can be used with Freestyle, Maven, and Pipeline jobs.

The plugin can consume the following report types:

- Functional test reports generated by Parasoft SOAtest 9.x.
- Static analysis and unit test reports generated by C/C++test 10.x Desktop and DTP Engines 10.x (C/C++test, Jtest, and dotTEST).

Requirements

- Jenkins 1.625.1 or later is required to use the plugin.
- Java 7 JDK and Maven 3 are required to build the plugin

Installation

The simplest installation method is to use the Jenkins plugin UI:

1. Choose **Manage Jenkins> Manage Plugins> Available.**
2. Locate and enable **Parasoft Findings Plugin for Jenkins.**
3. Click **Download now and install after restart**.

Alternatively, you can download the plugin from Parasoft's GitHub repository and follow the instructions to build the plugin as an HPI file (plugin format for Jenkins): https://github.com/jenkinsci/parasoft-findings-plugin. After creating the HPI file, you can deploy the plugin through the Jenkins web UI or through the command line.

Web UI

1. Choose **Manage Jenkins> Manage Plugins** from the Jenkins menu and click the **Advanced** tab.
2. In the Upload Plugin section, click **Choose File** and choose parasoft-findings.hpi.
3. Click **Upload**.
4. Restart Jenkins.

Command Line

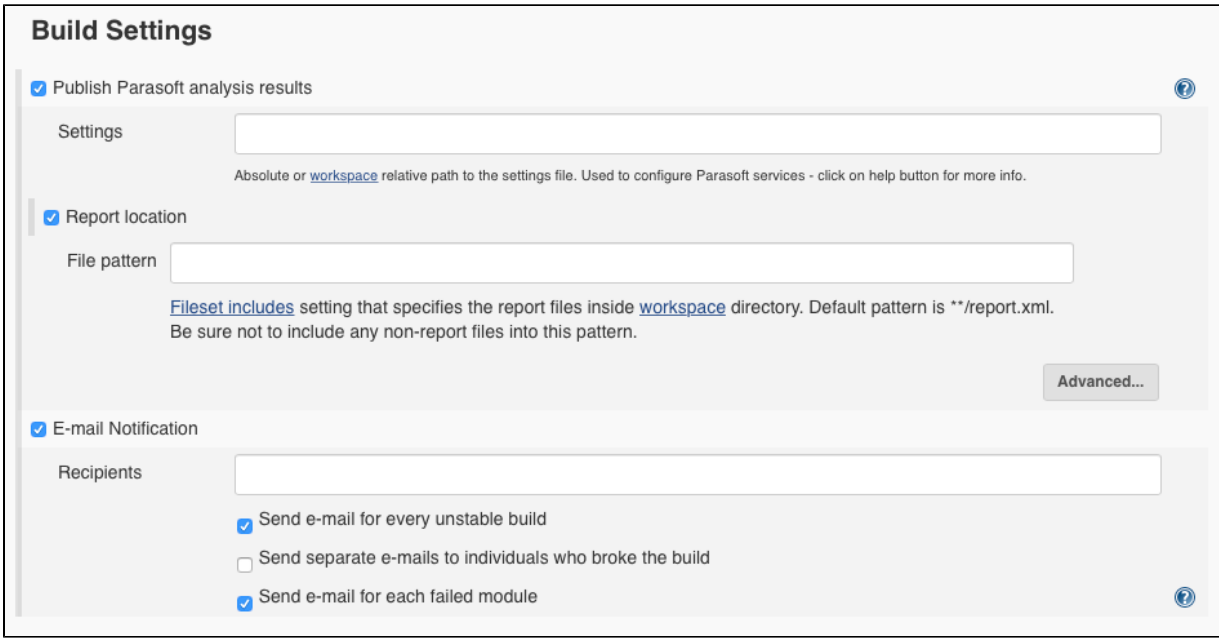
See the Jenkins documentation for command line installation. Restart Jenkins after installing the plugin.

Configuration

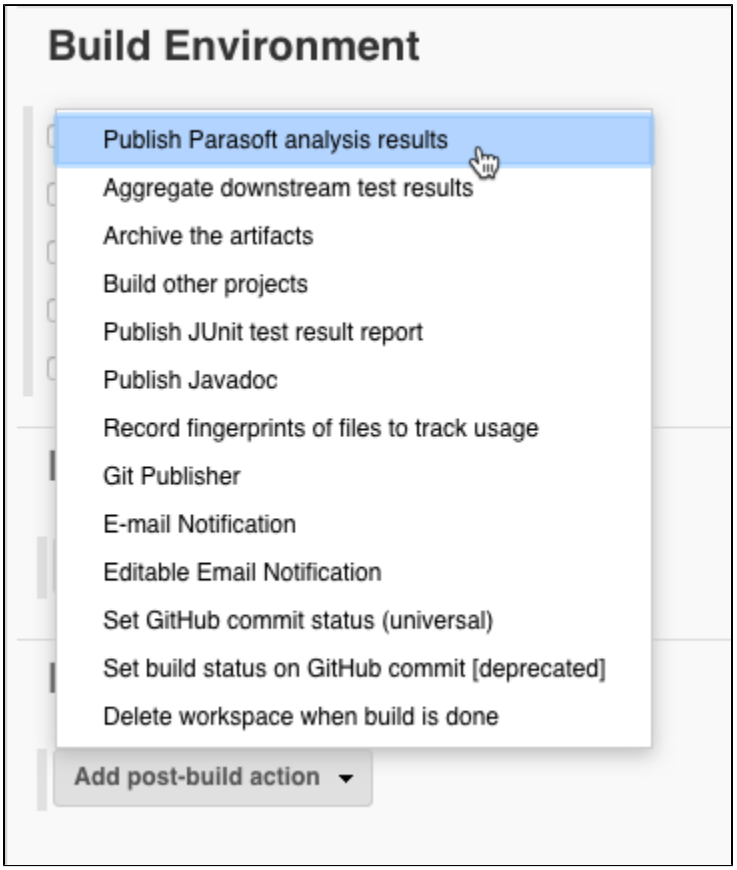
You can use the plugin with new and existing Jenkins jobs.

Publishing Static Analysis Results

1. Enable the **Publish Parasoft analysis results** option.
    - For Maven jobs, choose **Configure> Build Settings** and enable the **Publish Parasoft analysis results** option.



- For Freestyle jobs:
    1. Choose **Configure> Post-build Actions> Add post-build action**
    2. Choose **Publish Parasoft analysis results** from the drop-down menu.



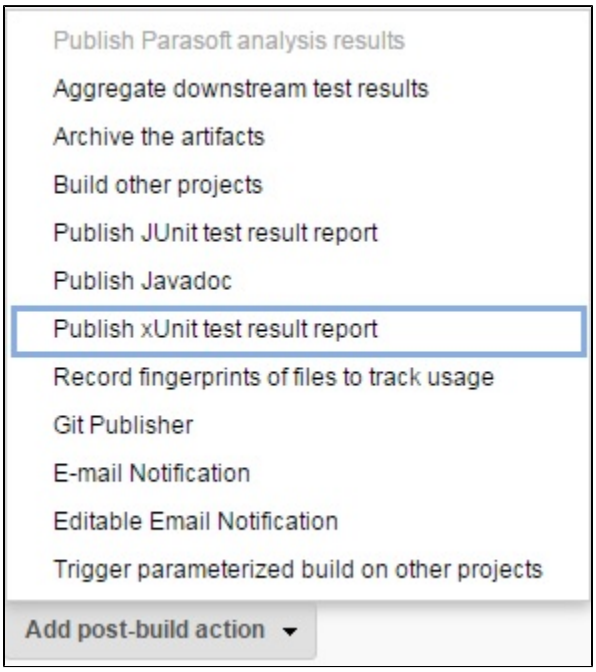
2. Specify an absolute or workspace-relative path to the settings file used by your Parasoft Analyzer in the Settings field. The settings file should include the `report.location` property, which specifies where the report.xml file is saved. See the documentation for your analyzer for additional information about configuring the settings file.
3. You can also enable the **Report location** option and specify a workspace-relative path to the report.xml file generated by Parasoft Analyzer in the File pattern field. This setting overrides the `report.location` configuration from the previous step and is useful when Jenkins and the directory for the report.xml file are both on the local machine.

> **C++test 10.x Desktop Report Settings**
>
> C++test 10.x Desktop reports for static analysis must be generated with the **Add absolute file paths to XML data** option enabled. You can enable this option on the command line by setting the `report.location_details=true` property in the settings file, which allows you to navigate to the source code if it is not stored in the Jenkins workspace.

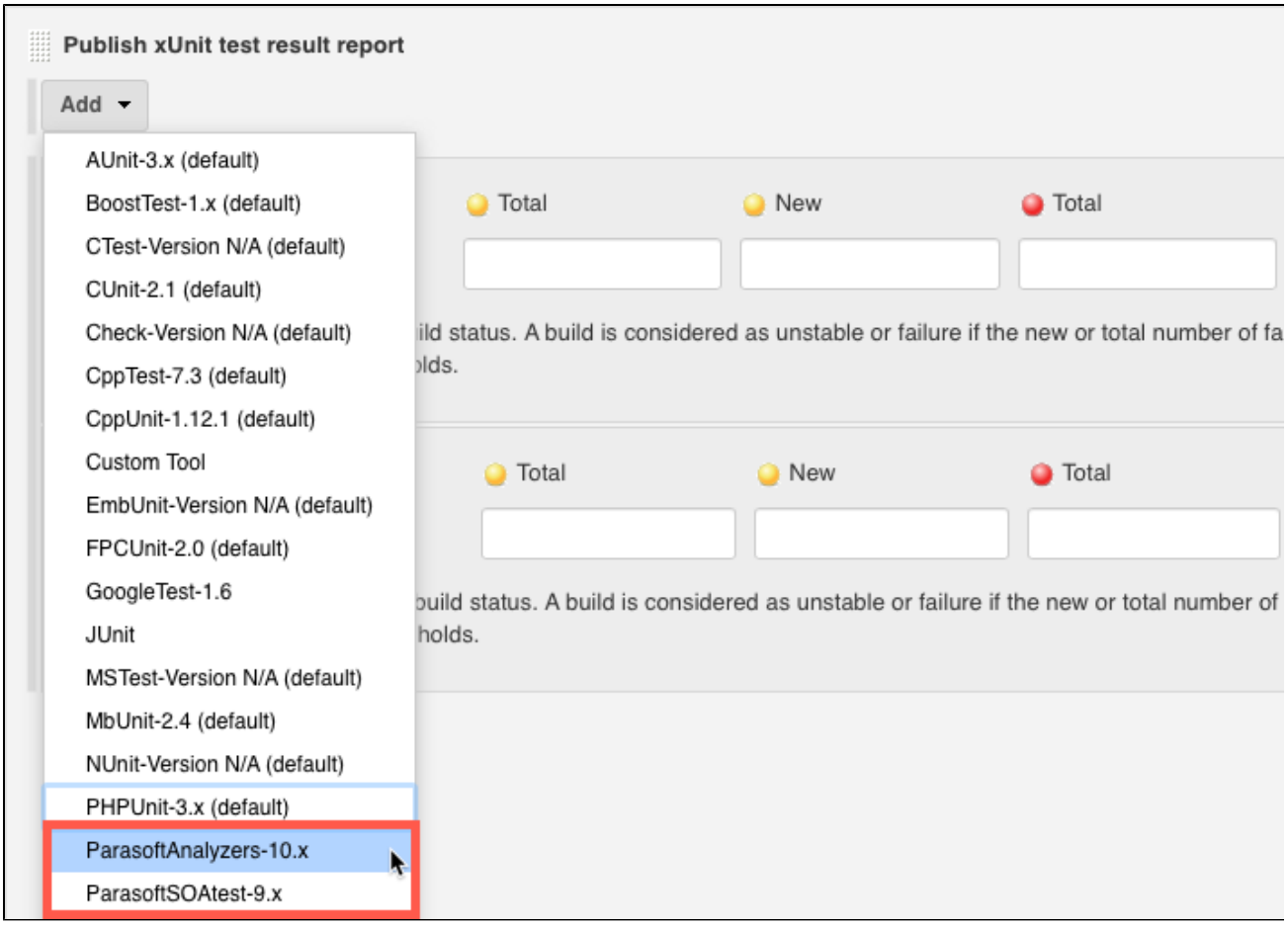
4. Configure access to the rule documentation in your settings file.
    1. Connect to DTP to access online documentation by setting the following properties:
        - `dtp.server`
        - `dtp.port`
        - `dtp.user`
        - `dtp.password`
    2. For local documentation, set the `report.rules` property to the directory that contains the analyzer documentation. For example:
    `report.rules=<engine-location>/rules/doc`

Publishing Test Execution Results for Freestyle and Maven Jobs

1. Choose **Configure> Post-build Actions> Add post-build action**.
2. Choose **Publish xUnit test result** from the drop-down menu.



3. Click **Add.**
4. Choose **ParasoftAnalyzers-10.x** or **ParasoftSOAtest-9.x.**



5. Specify a workspace-relative path to the report.xml file generated by Parasoft Analyzer in the Pattern field.

> **C++test 10.x Desktop Unit Test Results**
>
> C++test 10.x Desktop reports for unit test results must be generated with the **Overview of checked files and executed tests** option enabled. You can enable this option on the command line by setting the `report.contexts_details=true` property in the settings file.

Publishing Findings for Pipeline Jobs

Jenkins Pipeline is a suite of plugins that support implementing and integrating continuous delivery pipelines into Jenkins. See the Jenkins documentation for additional information about pipelines.

Static Analysis

To publish static analysis results using a pipeline job, add a step to the pipeline script to call the Parasoft Findings publisher class or use the symbolic name:

```
step([$class: 'ParasoftPublisher', useReportPattern: true, reportPattern: '*.xml', settings: ''])
```

or

```
parasoftFindings useReportPattern: true, reportPattern: '*.xml', settings: ''
```

The parameters passed correspond to the options following available in plugin configuration:

useReportPattern	Report location option
reportPattern	File pattern field

settings	Settings field

Unit and Functional Tests

To publish unit and functional test results using a pipeline job, add a step to call the xUnit publisher class and specify the appropriate type of reports to publish:

ParasoftAnalyzers-10.x

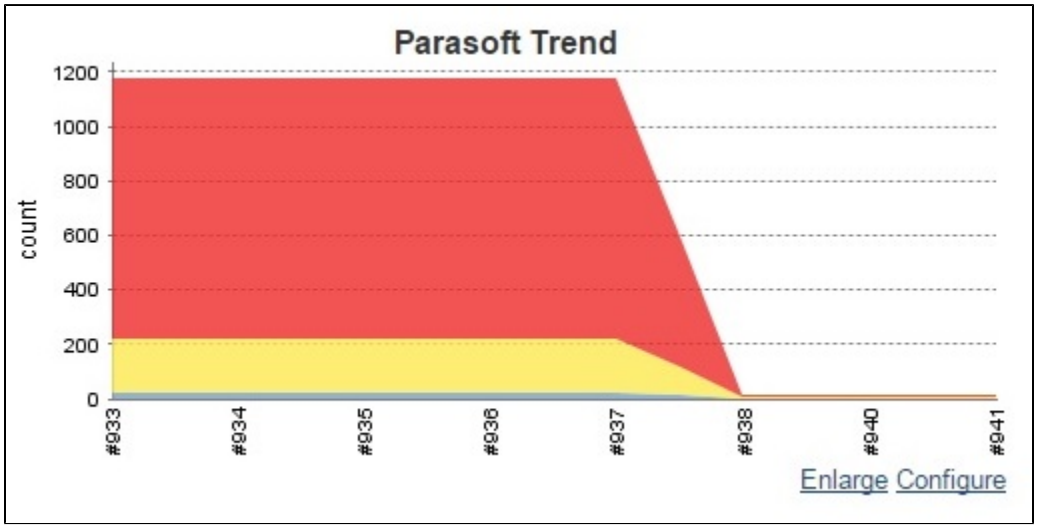
```
step([$class: 'XUnitBuilder', tools: [[$class: 'ParasoftType', pattern: '']]])
```

ParasoftSOAtest-9.x

```
step([$class: 'XUnitBuilder', tools: [[$class: 'ParasoftSOAtest9xType', pattern: '']]])
```

Viewing Static Analysis Findings

Static analysis trend graphs display results organized by the module, rule category or severity.



You can review the findings in the source code and navigate to rule documentation.

Parasoft Warnings - High Priority

Details

Modules | Packages | Files | Categories | Types | Warnings | Details

Module	Total	Distribution
com.parasoft.xtest.common:com.parasoft.xtest.analyzers.checkstyle	5	
com.parasoft.xtest.common:com.parasoft.xtest.analyzers.findbugs	2	
com.parasoft.xtest.common:com.parasoft.xtest.analyzers.simulator	13	
com.parasoft.xtest.common:com.parasoft.xtest.authorship	6	
com.parasoft.xtest.common:com.parasoft.xtest.chart.api	3	
com.parasoft.xtest.common:com.parasoft.xtest.common	106	
com.parasoft.xtest.common:com.parasoft.xtest.common.api	1	
com.parasoft.xtest.common:com.parasoft.xtest.common.swing	4	
com.parasoft.xtest.common:com.parasoft.xtest.common.ui	80	
com.parasoft.xtest.common:com.parasoft.xtest.communicator	2	

Parasoft Warnings - High Priority

Details

Modules | Packages | Files | Categories | Types | Warnings | Details

CheckstyleRulesImporter.java:85, BD.PB.CC, Priority: High , Author: dario, Revision: 246859

Condition "installDir != null" always evaluates to true

Avoid conditions that always evaluate to the same value

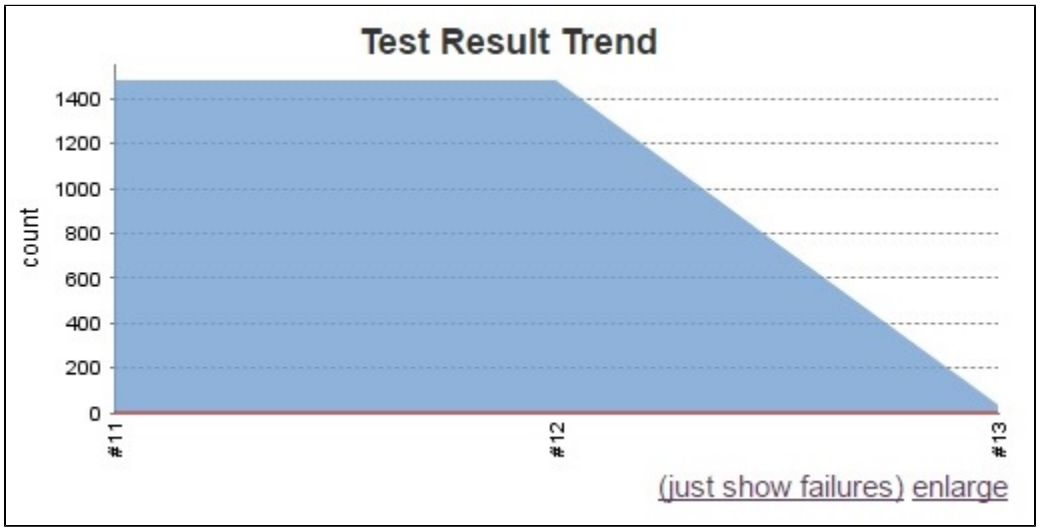
- .C CheckstyleRulesImporter.java (81): `doImport(File outputDir)` ♦ **Path start point**
- . CheckstyleRulesImporter.java (83): `File checkstyleConfig = null;`
- ! CheckstyleRulesImporter.java (84): `File installDir = _settings.getCheckstyleInstallDir();` ♦ **Contains critical data flow**
  - . CheckstyleSettings.java (53): {
  - . CheckstyleSettings.java (54): `if (_installDir == null) {`
  - ! CheckstyleSettings.java (61): `return _installDir;` ♦ **Critical data flow**
  - . CheckstyleSettings.java (53): {
- .P CheckstyleRulesImporter.java (85): `if (installDir != null) {` ♦ **Point where Senseless Condition is Used**

```
37      @Override
38      protected void drawItem(Graphics2D g2, int section,
39              Rectangle2D dataArea, PiePlotState state, int currentPass)
40      {
41          super.drawItem(g2, section, dataArea, state, currentPass);
42          if (currentPass == 1 && section == 0) {
43              Font centerTextFont = new Font(_centerTextFont, Font.PLAIN, _centerTextHeight);
44              g2.setFont(centerTextFont);
45              g2.setPaint(Color.BLACK);
46              TextUtilities.drawAlignedString(_centerText, g2,
47                      (float) dataArea.getCenterX(),
48                      (float) dataArea.getCenterY(),
49                      TextAnchor.CENTER);
50          }
51      }
52  }
```

"double" type cast to lower precision "float" typeDo not cast primitive data types to lower precision

Viewing Test Execution Results

The test execution trend graphs shows test status, execution time, and stack trace for test failures.



Test Result

2 failures (±0)

33 tests (±0)

Took 0.8 sec.

All Failed Tests

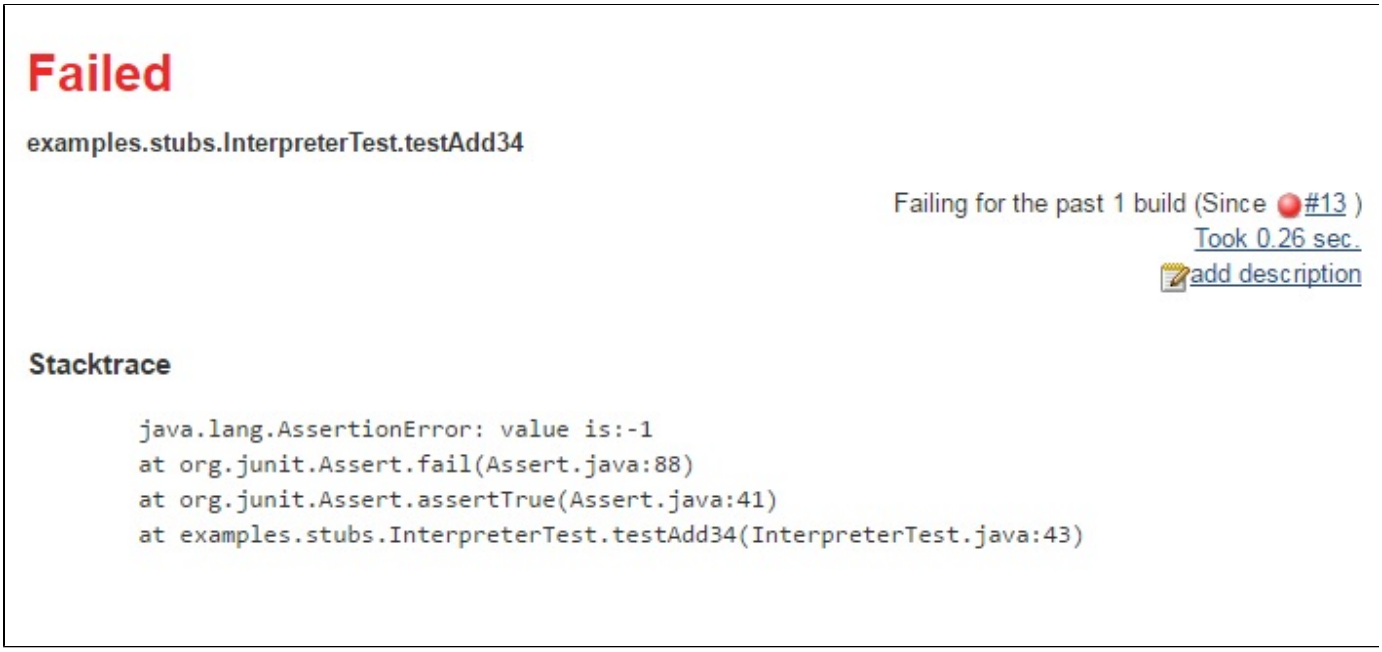
Test Name	Duration	Age
examples.stubs.InterpreterTest.testAdd34	0.26 sec	4
examples.nbank.AccountTest.testApply3	0 ms	4

All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
examples.junit4	25 ms	0		0		23		23	
examples.nbank	1 ms	1		0		2		3	
examples.servlets	0.22 sec	0		0		4		4	
examples.stubs	0.56 sec	1		0		2		3	

All Failed Tests

Test Name	Duration	Age
examples.stubs.InterpreterTest.testAdd34		
Stack Trace		
java.lang.AssertionError: value is:-1 at org.junit.Assert.fail(Assert.java:88) at org.junit.Assert.assertTrue(Assert.java:41) at examples.stubs.InterpreterTest.testAdd34(InterpreterTest.java:43)	0.26 sec	1
examples.nbank.AccountTest.testApply3		
Stack Trace		
junit.framework.AssertionFailedError: expected:<0> but was:<-1> at junit.framework.Assert.fail(Assert.java:57) at junit.framework.Assert.failNotEquals(Assert.java:329) at junit.framework.Assert.assertEquals(Assert.java:78) at junit.framework.Assert.assertEquals(Assert.java:234) at junit.framework.Assert.assertEquals(Assert.java:241) at junit.framework.TestCase.assertEquals(TestCase.java:409) at examples.nbank.AccountTest.testApply3(AccountTest.java:39)	0 ms	1



Change Log

10.3.6	Fixed an issue related to resolving environment variables when static analysis results are imported.
10.3.5	Added support for static analysis and unit test results for C++test 10.x Desktop. Added support for pipelines.
10.3.4	Fixed wrong number of test results reported in some cases. Fixed handling error messages and stack traces for tests having multiple failures. Fixed missing "time" attributes not handled properly. Fixed not using correct "time" attribute in some cases.
10.3.3	Updated Parasoft Services to 10.3.3.
10.3.2	Fixed the issue with Parasoft Findings Plugin not working with analysis-core version 1.82. Updated Parasoft Services to 10.3.2.
10.3.0	Added publishing functional test results from Parasoft SOAtest 9.x reports. Updated Parasoft Services to 10.3.0.
10.2.3.1	Fixed an issue with visualization of parameterized test results. Updated Parasoft Services to 10.2.3.
10.2.2	Initial release based on Parasoft Services 10.2.2.