Analysis Types 1

In this section:

Pattern-based Analysis

Pattern-based analysis detects constructs in the source code that are known to result in software defects based on programming standards, such as CWE and OWASP. Pattern-based static analysis helps ensure that developers are following coding best practices, unit testing best practices, as well as the organization's development policy.

This and all the following analysis types are performed with a built-in or user-defined test configuration; see Configuring Test Configurations.

Code Duplication Analysis

C/C++testcan check for duplicate code to help you improve application design and decrease maintenance costs. During code duplication analysis, the code is parsed into smaller language elements (tokens). The tokens are analyzed according to a set of rules that specify what should be considered duplicate code. There are two types of rules for analyzing tokens:

- Simple rules for finding single token duplicates, e.g., string literals
- Complex rules for finding multiple token duplicates, e.g., duplicate methods or statements

Run the Find Duplicated Code built-in test configuration during analysis to execute code duplicates detection rules:

```
builtin://Find Duplicated Code
```

Metrics Analysis

C/C++testcan compute several code metrics, such as code complexity, coupling between objects, or lack of cohesion, which can help you understand potential weak points in the code. Run the Metrics test configuration during analysis to execute metrics analysis rules:

```
builtin://Metrics
```

Metrics analysis results are added to the HTML and XML report files; see Viewing Reports

Setting Metrics Thresholds

You can set upper and lower boundaries so that a static analysis finding is reported if a metric is calculated outside the specified value range. For example, if you want to restrict the number of logical lines in a project, you could configure the Metrics test configuration so that a finding is reported if the Number of Logical Lines metric exceeds the limit.

The built-in Metrics test configuration includes default threshold values. There are some rules, such as Number of Files (METRIC.NOF), for which thresholds cannot be set.

Metric thresholds can be specified using the following methods:

- By using the test configuration interface in DTP (see "Report Center> Test Configurations> Editing Test Configurations> Metrics Tab" in the DTP user manual for details).
- By editing the Metrics test configuration using the interface in an IDE (see Creating Custom Test Configurations).
- By manually editing the test configuration file:

  1. Duplicate the built-in Metrics test configuration (`[INSTALL_DIR]/configs/builtin`) to the user configurations directory (`[INSTALL_DIR]/configs/user`).

  2. Open the duplicate configuration in an editor and set the `[METRIC.ID].ThresholdEnabled` property to `true`.

  3. Configure the lower and upper boundaries in the `[METRIC.ID].Threshold` property according to the following format: `[METRIC.ID].Threshold=l [lower boundary value] g [upper boundary value]`

  4. Save the test configuration and run the analysis using the custom metrics test configuration.

Flow Analysis

Flow Analysis is a type of static analysis technology that uses several analysis techniques, including simulation of application execution paths, to identify paths that could trigger runtime defects. Defects detected include use of uninitialized memory, null pointer dereferencing, division by zero, memory and resource leaks.

Since this analysis involves identifying and tracing complex paths, it exposes bugs that typically evade static code analysis and unit testing, and would be difficult to find through manual testing or inspection.

Flow Analysis' ability to expose bugs without executing code is especially valuable for users with legacy code bases and embedded code (where runtime detection of such errors is not effective or possible).

Run one of the Flow Analysis test configurations during analysis to execute flow analysis rules:

```
builtin://Flow Analysis Fast
builtin://Flow Analysis Standard
builtin://Flow Analysis Aggressive
```

Configuring Flow Analysis

See Configuring Flow Analysis for information how to configure the Flow Analysis options.