Code Metrics Values

Visual Studio 2010

Code metrics is a set of software measures that provide developers better insight into the code they are developing. By taking advantage of code metrics, developers can understand which types and/or methods should be reworked or more thoroughly tested. Development teams can identify potential risks, understand the current state of a project, and track progress during software development.

Software Measurements

The following list shows the code metrics results that Visual Studio calculates:

- **Maintainability Index** – Calculates an index value between 0 and 100 that represents the relative ease of maintaining the code. A high value means better maintainability. Color coded ratings can be used to quickly identify trouble spots in your code. A green rating is between 20 and 100 and indicates that the code has good maintainability. A yellow rating is between 10 and 19 and indicates that the code is moderately maintainable. A red rating is a rating between 0 and 9 and indicates low maintainability.
- **Cyclomatic Complexity** – Measures the structural complexity of the code. It is created by calculating the number of different code paths in the flow of the program. A program that has complex control flow will require more tests to achieve good code coverage and will be less maintainable.

  > **Note**
  >
  > In some cases, the calculation of the cyclomatic complexity for a method in Visual Studio 2010 differs from earlier versions. For more information, see the "Changes in Visual Studio 2010 code complexity calculations section" of Troubleshooting Code Metrics Issues.

- **Depth of Inheritance** – Indicates the number of class definitions that extend to the root of the class hierarchy. The deeper the hierarchy the more difficult it might be to understand where particular methods and fields are defined or/and redefined.
- **Class Coupling** – Measures the coupling to unique classes through parameters, local variables, return types, method calls, generic or template instantiations, base classes, interface implementations, fields defined on external types, and attribute decoration. Good software design dictates that types and methods should have high cohesion and low coupling. High coupling indicates a design that is difficult to reuse and maintain because of its many interdependencies on other types.
- **Lines of Code** – Indicates the approximate number of lines in the code. The count is based on the IL code and is therefore not the exact number of lines in the source code file. A very high count might indicate that a type or method is trying to do too much work and should be split up. It might also indicate that the type or method might be hard to maintain.

Anonymous Methods

An *anonymous method* is just a method that has no name. Anonymous methods are most frequently used to pass a code block as a delegate parameter. Metrics results for an anonymous method that is declared in a member, such as a method or accessor, are associated with the member that declares the method. They are not associated with the member that calls the method.

For more information about how Code Metrics treats anonymous methods, see Anonymous

Methods and Code Analysis.

Generated Code

Some software tools and compilers generate code that is added to a project and that the project developer either does not see or should not change. Mostly, Code Metrics ignores generated code when it calculates the metrics values. This enables the metrics values to reflect what the developer can see and change.

Code generated for Windows forms is not ignored, because it is code that the developer can see and change.

See Also

Other Resources

Measuring Complexity and Maintainability of Managed Code

Community Content