Metrics

Percent Test Cases Passed

Definition: Number of completed test cases / Number of proposed test cases * 100%

Process: Each week we look at the test cases in our testing suite and see which ones pass.

Analysis: This metric will be used to track the week to week progress of our project. If we have a consistently high number of passing tests each week then we know our project is doing well. If the percent is low, then we know that our project needs to be focused on more.

Why: The metric will give a very good indicator of the health of the project.

Lines of Code

Definition: The number of lines of code in the project not including comments and pure whitespace lines.

Process: Use Titanium's/Eclipse's search method to count the number of matches that match the regex (^[^/][^/][^\n]\S*).

Analysis: The line of code count will give an indicator of total progress on different parts of the project. This information will be used with other metrics to try to get a better picture of the development. For example, a large number of new lines of code and few passing tests could be an indicator of bad code.

Why: When paired with other metrics, the lines of code provides a better picture of development.

Fix Quality

Definition: The number of correct fixes that are used in our project / The number of fixes created for the week * 100%

Process: Every time a fix is created for a test case that fails we add to the denominator. If the fix makes the test case pass and does not cause other previously working test cases to fail, then we will add it to the numerator as well

Analysis: This metric will tell us our overall effectiveness with our fixes. If the metric is low, then we are wasting a lot of time creating wasted code. If the metric is high then we know we are correctly dealing with the test cases.

Why: This metric is a great indicator of our work effectiveness.

Depth of Callback functions

Definition: The average depth of the callback stack for any given calculated data.

Process: We will determine which functions have data that needs to be tracked. Then, using some form of tracing (either the debugger or by manual count) we will check to see what the depth of the function is. We will create an average of all the data and keep track of the maximum depth for possible reduction.

Analysis: This metric will give a good indicator of how our design is progressing. A low Depth means that we might have functions that do too much. In this case, we split them. However, if we have functions that have a very large depth, then we might look at it to see if any simplification can be made.

Why: This metric is an indicator of design quality.

Number of Methods

Definition: The number of methods per class/module

Process: Titanium provides an outline of each file, which lists each method within it.

Analysis: Similar to depth of callback functions, this metric will give a general guide to our design. Classes with few methods indicates that either the class is simple or the methods do too much. This information can be used for refactoring purposes.

Why: This metric is an indicator of design quality.