Incremental Changes

From Design to Developer
Maintaining and Improvement of
Existing Code and Architecture

Massive Change

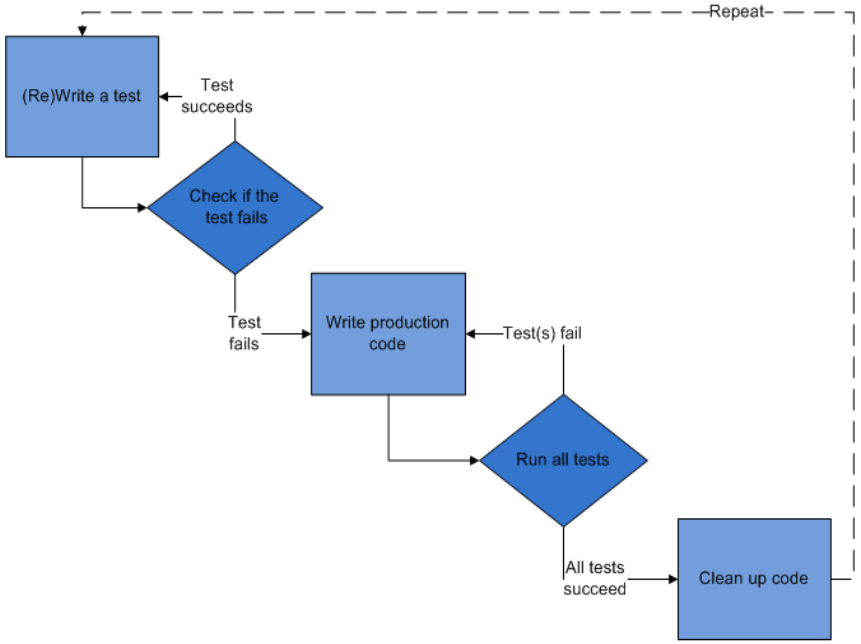
- can happen quickly by modifying code structure
- a costly way to develop software
- will probably happen if developers are not prudent

Massive changes in software are not inheritably bad changes. Software projects that require ongoing development will need a better approach.

Test-Driven Development

Verifies the system is running to specification at any point in time

Forces developers to make a large number of tiny changes in order to realize the improvement



From Wikipedia TDD page

Refactoring

- Create appropriate places to put different kinds of code
- Easy with modern IDE's
- Works well with TDD but not necessary
- Great way to increase cohesion

Working Around Bad Architecture

An architecture where coupling is high and cohesion is low will require “work-arounds”

First create a new module with your own code that uses (wa) the architecture to get a working product.

Second create small changes to re-factor the current software layout to include your code

Repeat until software has a stable design for change or no more improvements are required.

Good Idea

- Cleaning up code as you go
  - Removing code > adding code
- Creating, modifying, or specifying tests that verify the integrity of new changes as they happen
  - This can be unit testing or human testing.
  - Be sure to have a maintained document for any testing that requires a human.

Bad Idea

- Writing a “first draft” for your feature/ improvement on a project with pre-existing code
- Eliminating key classes that employ the concept of the program.