Dealing with legacy

Stop writing code without unit tests.

Add the Right Tests

Create tests with all new features

Do not write new features without unit testing. You are just making the code base worse and getting further away from introducing tests into your system.

Sprout Method

When adding new functionality, write the code in a new method and with TDD and then call this method from the old code. So even if you can’t test the code where your method is being called, at least the new code has tests. This technique (from Michael Feathers again) is called Sprout Method.

Wrap Method

1. Extract all the code in exiting function or method into a private method & call it from the same method.
2. Create a new method with the new features & add the new call before or after as per requirement.

All new features will need to have tests created to prove that the feature works as expected. If you’re covering the new aspects of your application, then at least things aren’t getting worse.

Create tests as you fix bugs

Add tests to prove that your specific fix is working and keep them running to show that this does not break again.

This benefits from being somewhat targeted – you are creating tests in your weakest areas first. The weaker it is (i.e. more bugs) the faster you will build up tests.

Create tests as you add to old features

Create tests in identified risk areas

Turn on Code Coverage

1. Start running code coverage against all your code
2. Get a baseline
3. Determine areas that you want to exclude.
4. Determine coverage goals.
5. Work-out steps to improve your coverage
6. Determine your pass/fail criteria
7. Run Code Coverage constantly

Run your Tests on a Scheduled Basis

1. Run them on every check-in
2. Run them on every build
3. Run them on every deploy
4. Run them every X days/hours/minutes