Code Analysis Through Unit Testing

Paul Mitchum
Mile23 on drupal.org
@PaulMitchum

☐ MidCamp 2015
☑ SANDcamp 2015

Community Shout-Outs

- SeaDUG: https://groups.drupal.org/seattle
  - Organizing PNWDrupalSummit
- SeattlePHP: http://www.meetup.com/seaphp/
  - Organizing PNWPHP

Remember To Ask The Audience:

- Who has participated in a d.o project?
- Core issue for D8?
- Core developer for D8?
- Reviewed a patch for D8?

Things You'll Understand After This Talk

- A structured process for code quality analysis and improving maintainability.

- Metrics for understanding the effectiveness of the process.

Why Are We Analyzing Code?

- Scenario for the purposes of this talk:
  - We've been assigned to maintain legacy code.
    - We have no familiarity with the code's structure or purpose.
- We have to analyze this code to understand it and do our job.

What Is ‘Legacy Code?’

- All code is legacy code.
- Usually, people think of legacy code as old, smelly, dusty code that needs a bath. Unmaintainable, unmaintained…
- But actually: The code you wrote yesterday is legacy code to a new person today.
- The destiny of all code is to become legacy code.

What Is ‘Maintainability?’

- The ease with which code can be understood.

- The ease with which changes can be evaluated.

Maintainability User Stories

- The Newbie: How do you find your feet? Relies on documentation, first-pass analysis.

- The Maintainer: How hard is it to evaluate a patch or PR? Relies on CI, relationship/reputation.

- The Genius: You can understand the complexity, but you can't explain it to others effectively. AKA Hit By A Bus Scenario. Relies on no one. Does not exist.

Maintainability
Rules Of Thumb

- Establish a metric for risk of change.

- If you can read code, you can maintain it. (Simple code.)

- If you can analyze code quality, you have tools available to help you maintain the code without reading it. (Tests, coverage.)

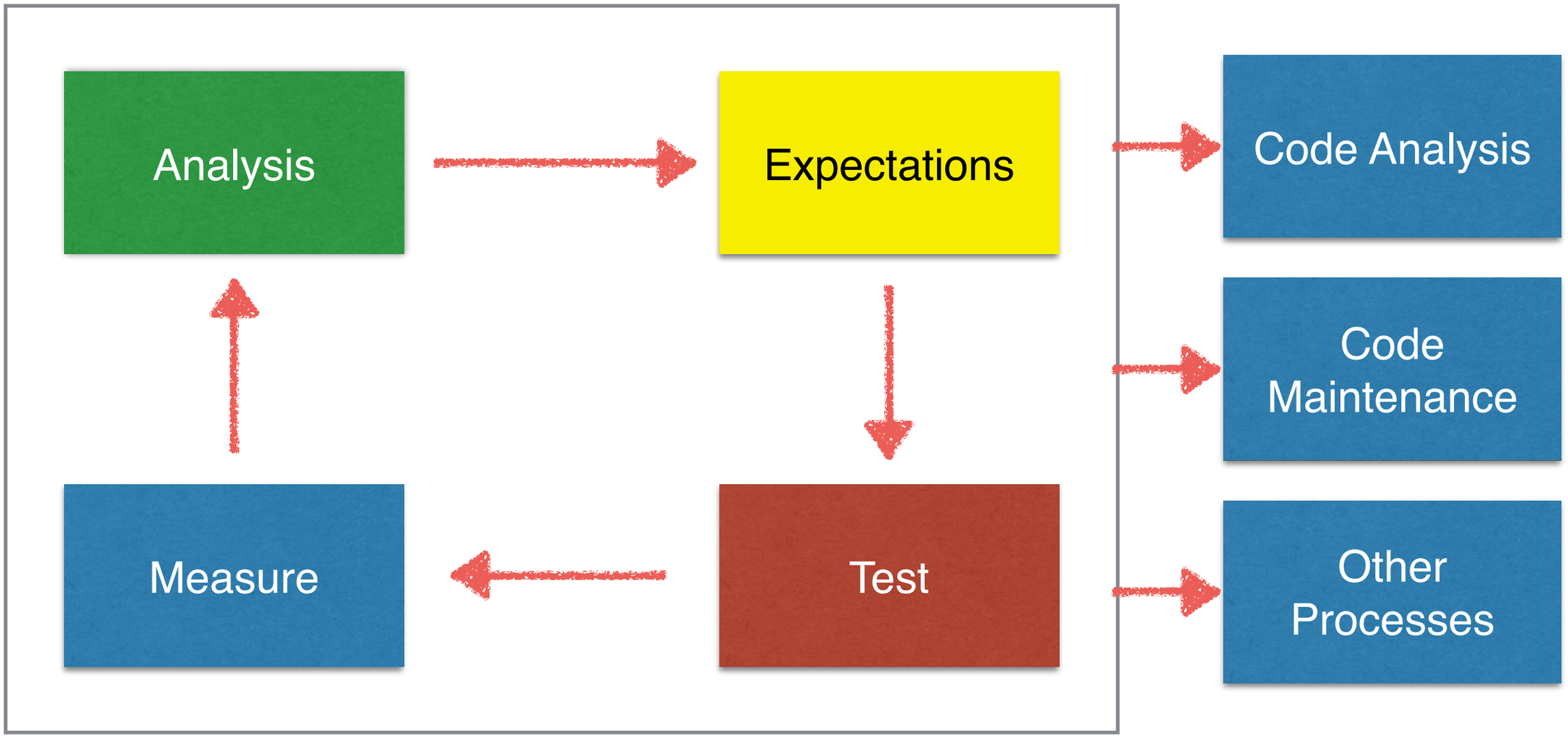
- If code is complex, you need the tools to help you understand the code. (Complex code, tests, coverage.)

Mile23's Big Idea

- Analysis leads to expectations, right or wrong.
- Expectations can be written as a test.
- Running the test shows the difference between your expectations and implementation behavior.
- If you write tests you will know things, if you don't write tests, you won't. Untested expectations can only be called 'assumptions.'

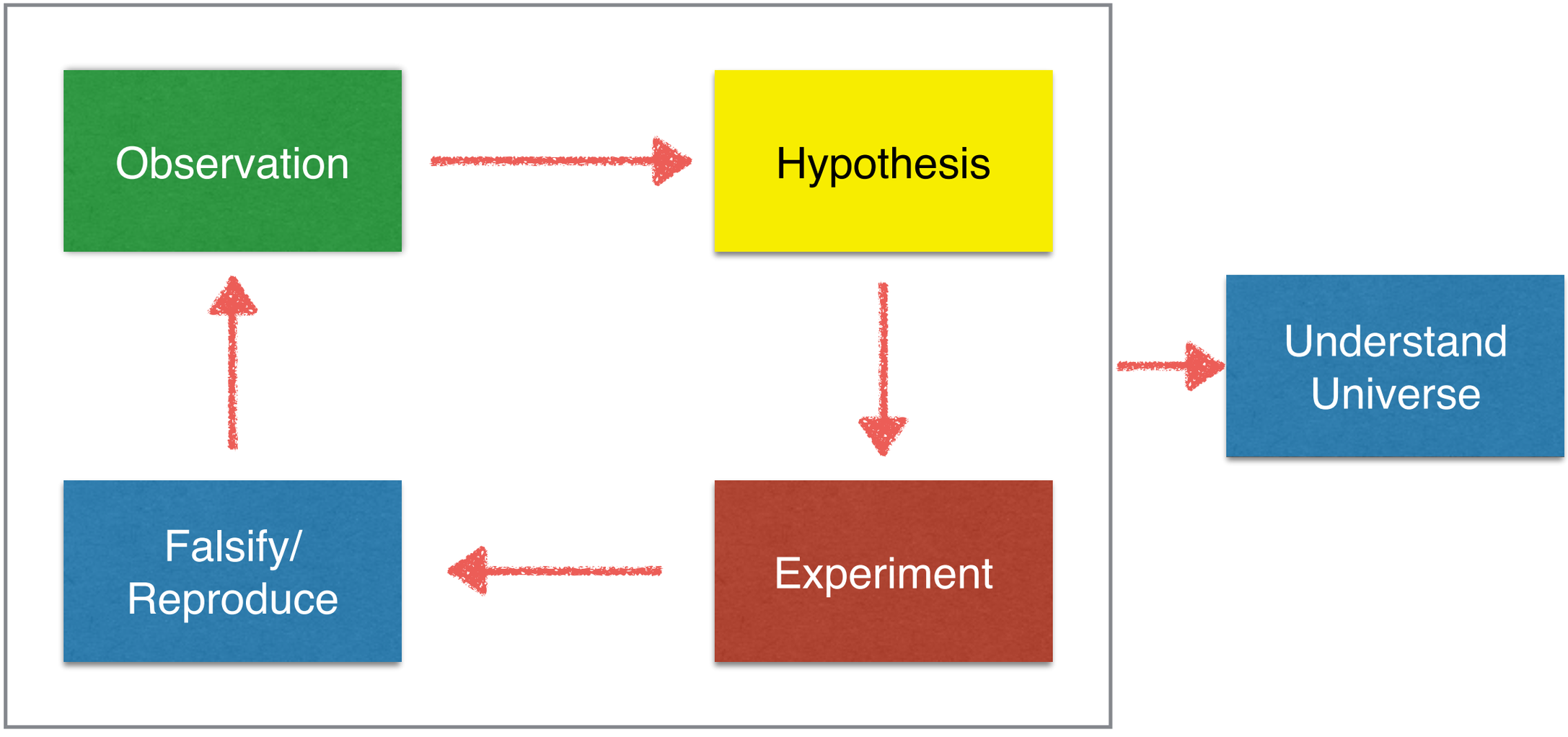
Mile23's Big Idea Picture Version

(circa 2015)



Scientific Method

(c. end of the dark ages)



What Tools Do We Have?

- PHPUnit:
    - Validation and sanity-check while we're writing code.
    - Regression checks.
- PHPMD, PHPUnit coverage + CRAP
    - Measure code quality in aggregate.

Mission: Gettext

- You have inherited Gettext as legacy code.
- Assignment 1: Determine project risks.
- Assignment 2: Mitigate those risks.

What Is Gettext?

- `t("Translate this, @name!", array('@name' => $account->name));`

- Read string translation from .po file.

- Many implementations, including a native one in Drupal.

- Drupal's Gettext suffers from lack of love:

  - https://www.drupal.org/node/1637662 (2012)

Mission: Gettext

- Obvious question:
  - Should we just replace it?
- Answer:
  - We have no way of evaluating whether we should.
- NOTE: Not advocating actually replacing Drupal Gettext. :-)

Mission: Gettext

- [insert live demo here]

Mission: Gettext

- Revisit those questions:
  - Assignment 1: What's the main project risk? PoStreamReader::readLine().
  - Assignment 2: How do we mitigate the risk? Write tests with an eye towards coverage.
  - Should we replace it? We're more able to evaluate that.

Mission: Gettext

- If we decide to replace it, we know why we're doing that.

- If we decide to keep it, we have tests.

- If we want to make it more performant, we have a clue as to where to start because we performed analysis.