Testing Overview

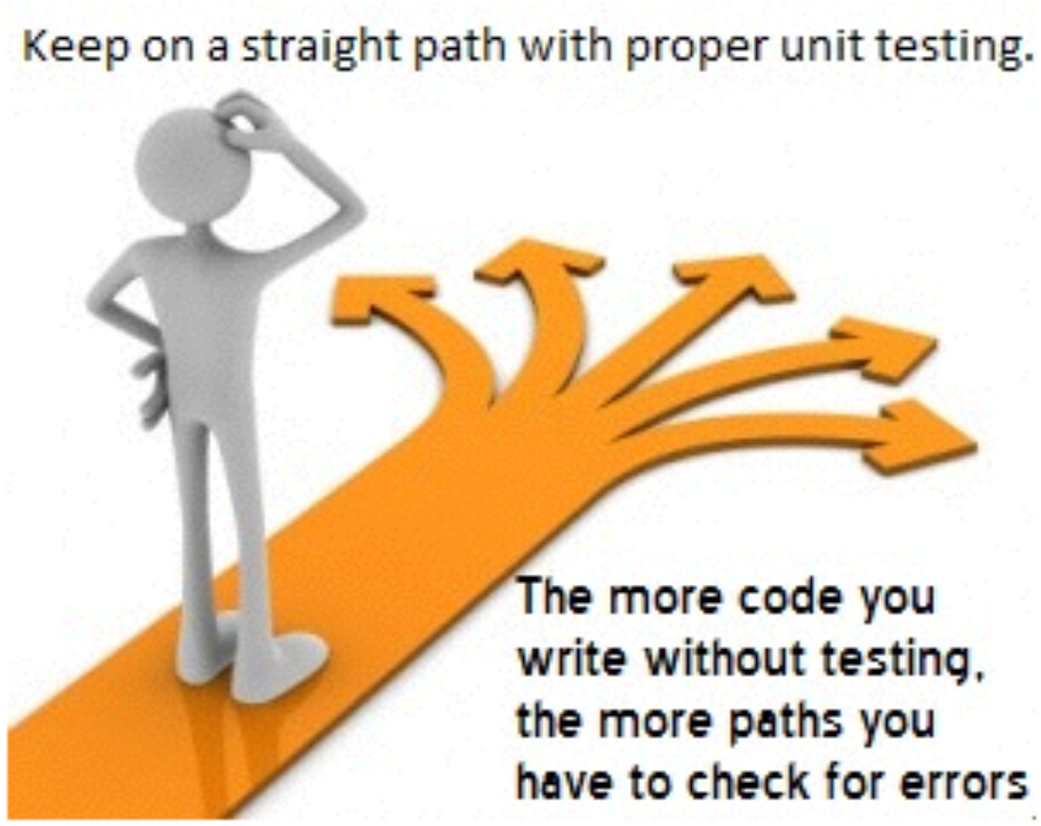
Overview

- Types of Testing
- Unit Testing Theory
- Test Driven Development (TDD)
- Unit Testing Exercises with PHPUnit and Mockery

Current Testing Strategy

1. Write some code, save
2. Open up a browser and see if it works
3. Repeat

This is a slow and manual process. Let a computer automatically do this for you.

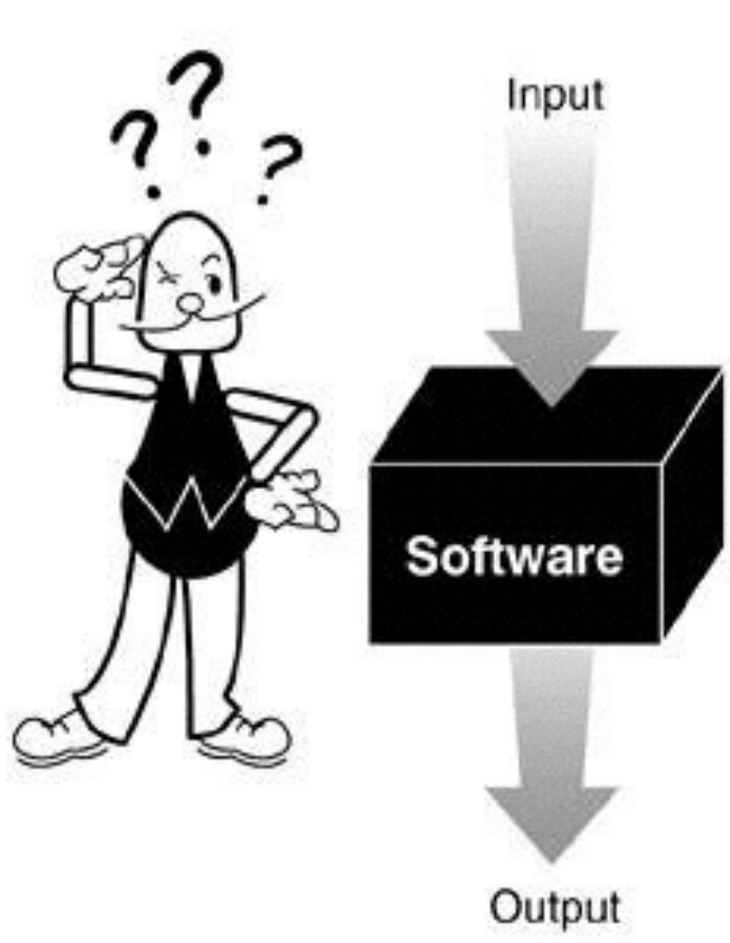


End to End / Acceptance / UI Tests

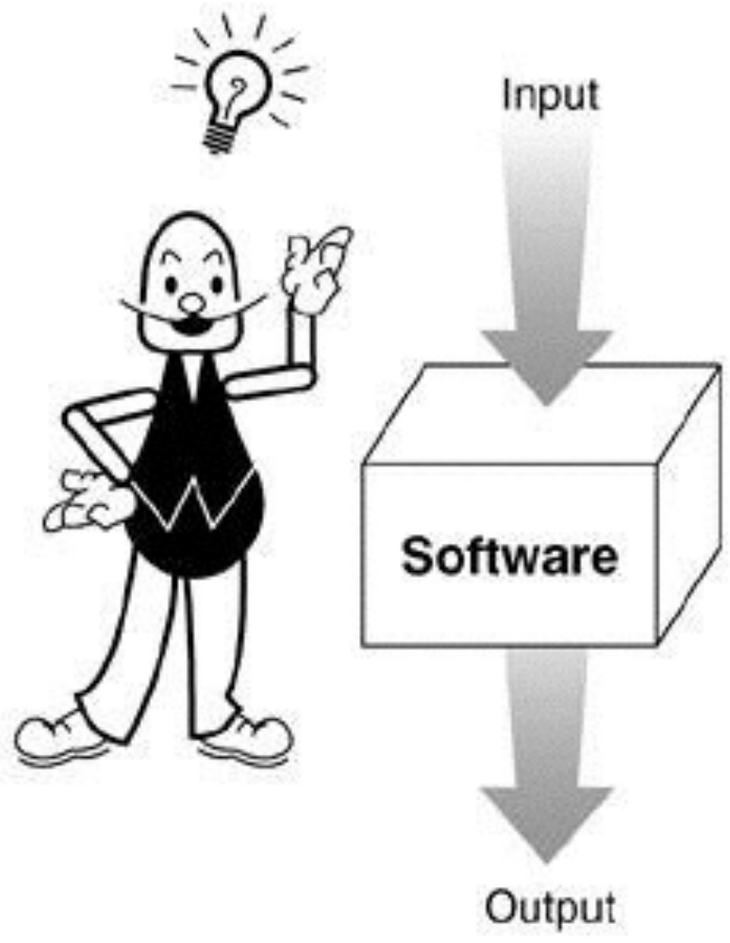
- Browser automation
- Verify the client's expectations
- Tests that all the pieces "fit together"

Unit Tests

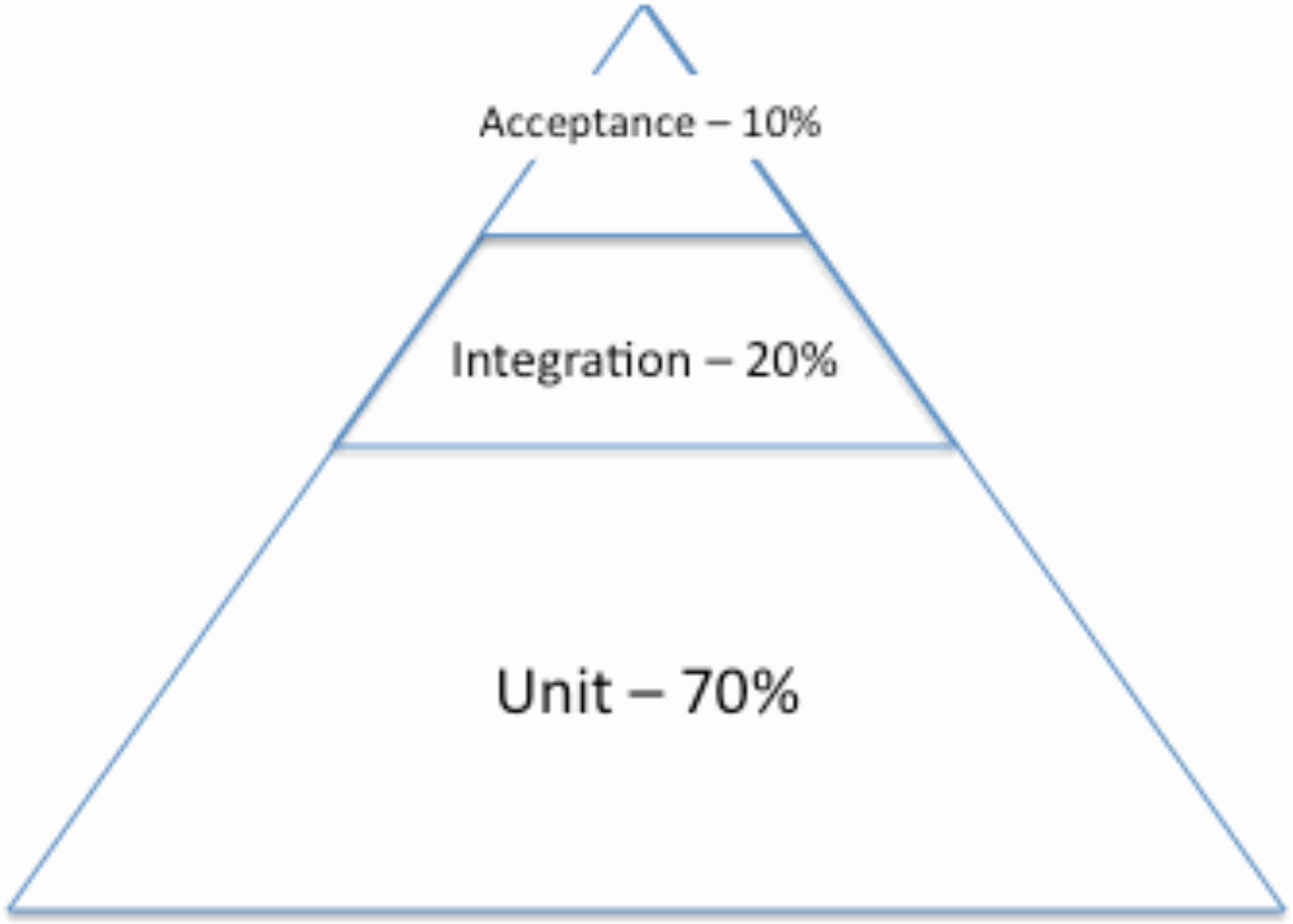
- Test small units of code in **isolation**
- Typically tests one aspect of a method/function
- Can be repeatedly run in seconds
- Follow the format
    1. Arrange
    2. Act
    3. Assert



Black-Box Testing



White-Box Testing


Acceptance – 10%
Integration – 20%
Unit – 70%

Unit Test Example

```
public function test_computes_bill_total()
{
    // Arrange
    $invoice = new Invoice([ /* data */ ]);

    // Act
    $total = $invoice->total();

    // Assert
    $this->assertEquals(10, $total);
}
```

What about TDD?

- TDD = Test Driven Development
- It is an approach to writing tests

1. Write a test, watch it fail
2. Write code to make it pass
3. Repeat

Why TDD?

1. Lots of test coverage from the start. Confidence in your code
2. Improved architecture
    - Design before writing code. Think about the public API
    - Help keep you from writing code that does too much stuff
3. Documentation
    - See how to use classes / functions by looking at the tests
4. **It's fun!**

Write automated tests to verify that your code behaves as expected

Demos

Hard to Test

```
class Database {
    public function __construct()
    {
        $this->conn = new MySQLConnection(…);
    }
}

$db = new Database();
```

Dependency Injection

```
class Database {
    public function __construct(MySQLConnection $connection)
    {
        $this->conn = $connection;
    }
}

$connection = new MySQLConnection(…);
$db = new Database($connection);
```

Hard to Test

```
class RottenTomatoes {
  public function search($dvd_title)
  {
    if (Cache::has($dvd_title)) {
      return json_decode(Cache::get($dvd_title));
    } else {
      $url = “http://api.rottentomatoes.com/api/public/
v1.0/movies.json?page=1&apikey=SOMEKEY&q=$dvd_title";

      $json_string = file_get_contents($url);
      Cache::put($dvd_title, $json_string, 60);
      return json_decode($json_string);
    }
  }
}
```