Mocking - Unit Testing - TDD

Cel Pynenborg

AXXES

Cel Pynenborg

Full Stack Java Developer @ Axxes

4 Years

Projects:

- *HealthConnect*
- *TomTom*
- *ST Engineering*
- *Telenet*

AXXES_

EForms Core/HealthLink

HealthConnect - Corilus

1Y3M

Communication between caregivers – organizations

Digital forms

Version 2

Persistence => Forms became Dossiers

AXXES_

TT – Open Maps

TomTom

1Y

Added value data

Indexing all the oneway streets

Merging open-source with closed-source data

Big data project

AXXES_

ST-E – FnP

ST Engineering

Fault and Performance

Telemetry and metrics

Satellite networks

Telenet – Digital Acceleration and Transformation

Telenet

Full Stack (Angular - Java)

DevOps (CI/CD - Kuberentes)

Security

Accelerating B2B teams with applications

Providing one application for Large Enterprise customers to manage their Telenet Products

AXXES_

Content

- Software Testing
  - What? Why? How?
  - Testing Levels
- Testing Plain Old Java Code
  - JUnit5
  - Mockito
  - AssertJ
- Test Driven Development
  - What? Why? How?
  - TDD and Testing in modern software development
- Integration Testing
  - @SpringBootTest
  - TestContainers / RestAssured / DBRider-DBUnit

AXXES_

Software Testing

–

A byte sized explanation

What is Software Testing

- Supplementary (non-business) code / software
- Verify the behavior of business logic
- Objective and independent view into the software

AXXES_

Why should I test my software

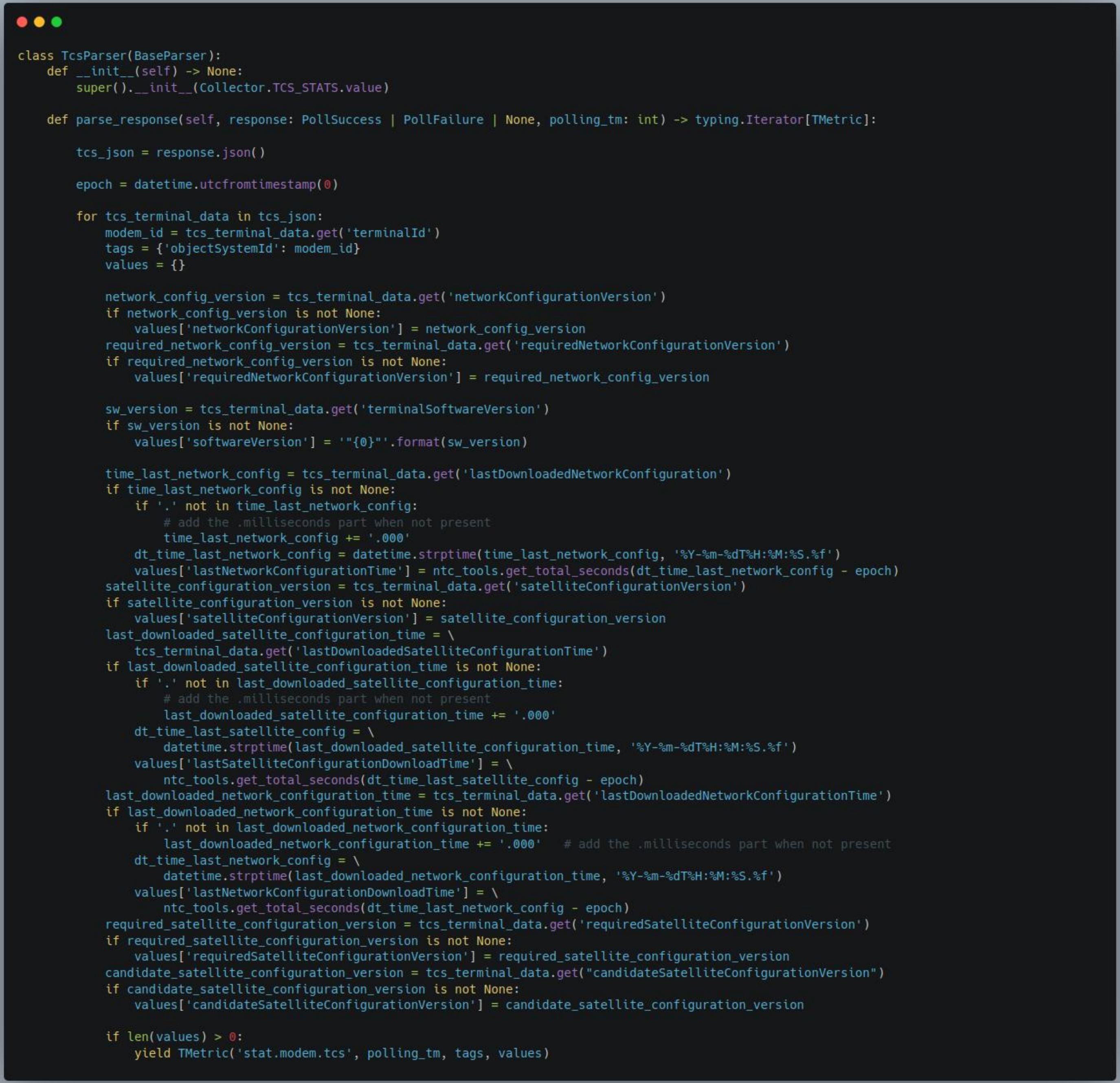
- Confidence in the source code
- Know the strengths and weaknesses
- Possibilities are infinite
- Sometimes there's more testing code than business logic
- Self-documenting
  - updateAccount_asAdmin_hasNotVerifiedAccount
  - BDD (Behaviour Driven Development)

```
new Time().equals(new Money())
```

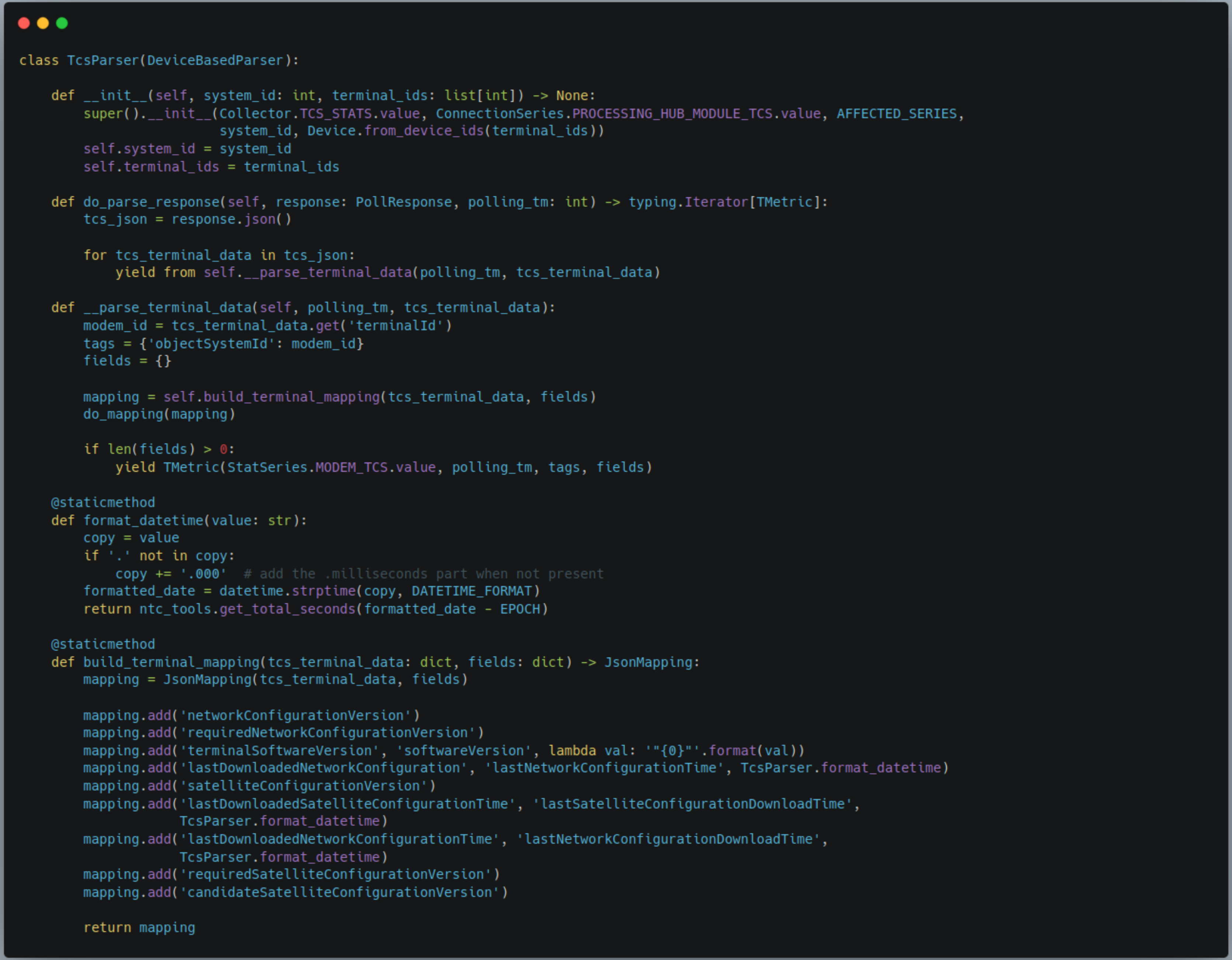
AXXES_

Things to consider

- Confidence in the test code
- “Good code” with bad tests cannot be considered reliable
- Bad code with good tests is a solvable issue
- Bad code with bad tests is a nightmare



```python
class TcsParser(BaseParser):
    def __init__(self) -> None:
        super().__init__(Collector.TCS_STATS.value)

    def parse_response(self, response: PollSuccess | PollFailure | None, polling_tm: int) -> typing.Iterator[TMetric]:

        tcs_json = response.json()

        epoch = datetime.utcfromtimestamp(0)

        for tcs_terminal_data in tcs_json:
            modem_id = tcs_terminal_data.get('terminalId')
            tags = {'objectSystemId': modem_id}
            values = {}

            network_config_version = tcs_terminal_data.get('networkConfigurationVersion')
            if network_config_version is not None:
                values['networkConfigurationVersion'] = network_config_version
            required_network_config_version = tcs_terminal_data.get('requiredNetworkConfigurationVersion')
            if required_network_config_version is not None:
                values['requiredNetworkConfigurationVersion'] = required_network_config_version

            sw_version = tcs_terminal_data.get('terminalSoftwareVersion')
            if sw_version is not None:
                values['softwareVersion'] = '"{0}"'.format(sw_version)

            time_last_network_config = tcs_terminal_data.get('lastDownloadedNetworkConfiguration')
            if time_last_network_config is not None:
                if '.' not in time_last_network_config:
                    # add the .milliseconds part when not present
                    time_last_network_config += '.000'
                dt_time_last_network_config = datetime.strptime(time_last_network_config, '%Y-%m-%dT%H:%M:%S.%f')
                values['lastNetworkConfigurationTime'] = ntc_tools.get_total_seconds(dt_time_last_network_config - epoch)
            satellite_configuration_version = tcs_terminal_data.get('satelliteConfigurationVersion')
            if satellite_configuration_version is not None:
                values['satelliteConfigurationVersion'] = satellite_configuration_version
            last_downloaded_satellite_configuration_time = \
                tcs_terminal_data.get('lastDownloadedSatelliteConfigurationTime')
            if last_downloaded_satellite_configuration_time is not None:
                if '.' not in last_downloaded_satellite_configuration_time:
                    # add the .milliseconds part when not present
                    last_downloaded_satellite_configuration_time += '.000'
                dt_time_last_satellite_config = \
                    datetime.strptime(last_downloaded_satellite_configuration_time, '%Y-%m-%dT%H:%M:%S.%f')
                values['lastSatelliteConfigurationDownloadTime'] = \
                    ntc_tools.get_total_seconds(dt_time_last_satellite_config - epoch)
            last_downloaded_network_configuration_time = tcs_terminal_data.get('lastDownloadedNetworkConfigurationTime')
            if last_downloaded_network_configuration_time is not None:
                if '.' not in last_downloaded_network_configuration_time:
                    last_downloaded_network_configuration_time += '.000'   # add the .milliseconds part when not present
                dt_time_last_network_config = \
                    datetime.strptime(last_downloaded_network_configuration_time, '%Y-%m-%dT%H:%M:%S.%f')
                values['lastNetworkConfigurationDownloadTime'] = \
                    ntc_tools.get_total_seconds(dt_time_last_network_config - epoch)
            required_satellite_configuration_version = tcs_terminal_data.get('requiredSatelliteConfigurationVersion')
            if required_satellite_configuration_version is not None:
                values['requiredSatelliteConfigurationVersion'] = required_satellite_configuration_version
            candidate_satellite_configuration_version = tcs_terminal_data.get("candidateSatelliteConfigurationVersion")
            if candidate_satellite_configuration_version is not None:
                values['candidateSatelliteConfigurationVersion'] = candidate_satellite_configuration_version

            if len(values) > 0:
                yield TMetric('stat.modem.tcs', polling_tm, tags, values)
```



```python
class TcsParser(DeviceBasedParser):

    def __init__(self, system_id: int, terminal_ids: list[int]) -> None:
        super().__init__(Collector.TCS_STATS.value, ConnectionSeries.PROCESSING_HUB_MODULE_TCS.value, AFFECTED_SERIES,
                         system_id, Device.from_device_ids(terminal_ids))
        self.system_id = system_id
        self.terminal_ids = terminal_ids

    def do_parse_response(self, response: PollResponse, polling_tm: int) -> typing.Iterator[TMetric]:
        tcs_json = response.json()

        for tcs_terminal_data in tcs_json:
            yield from self.__parse_terminal_data(polling_tm, tcs_terminal_data)

    def __parse_terminal_data(self, polling_tm, tcs_terminal_data):
        modem_id = tcs_terminal_data.get('terminalId')
        tags = {'objectSystemId': modem_id}
        fields = {}

        mapping = self.build_terminal_mapping(tcs_terminal_data, fields)
        do_mapping(mapping)

        if len(fields) > 0:
            yield TMetric(StatSeries.MODEM_TCS.value, polling_tm, tags, fields)

    @staticmethod
    def format_datetime(value: str):
        copy = value
        if '.' not in copy:
            copy += '.000'  # add the .milliseconds part when not present
        formatted_date = datetime.strptime(copy, DATETIME_FORMAT)
        return ntc_tools.get_total_seconds(formatted_date - EPOCH)

    @staticmethod
    def build_terminal_mapping(tcs_terminal_data: dict, fields: dict) -> JsonMapping:
        mapping = JsonMapping(tcs_terminal_data, fields)

        mapping.add('networkConfigurationVersion')
        mapping.add('requiredNetworkConfigurationVersion')
        mapping.add('terminalSoftwareVersion', 'softwareVersion', lambda val: '"{0}"'.format(val))
        mapping.add('lastDownloadedNetworkConfiguration', 'lastNetworkConfigurationTime', TcsParser.format_datetime)
        mapping.add('satelliteConfigurationVersion')
        mapping.add('lastDownloadedSatelliteConfigurationTime', 'lastSatelliteConfigurationDownloadTime',
                    TcsParser.format_datetime)
        mapping.add('lastDownloadedNetworkConfigurationTime', 'lastNetworkConfigurationDownloadTime',
                    TcsParser.format_datetime)
        mapping.add('requiredSatelliteConfigurationVersion')
        mapping.add('candidateSatelliteConfigurationVersion')

        return mapping
```

Signs that you’re doing it wrong

- Convoluted setup
- Duplicate tests
- Slow tests
- Flaking tests
    - Race conditions
    - Reliance on the order in which tests are run

These factors result in “loss of confidence” in the code base!

AXXES_

How to do it right

- KISS (Keep-it-simple)
- Isolated test scenarios
- Run tests often
- Treat test code with the same care as core logic

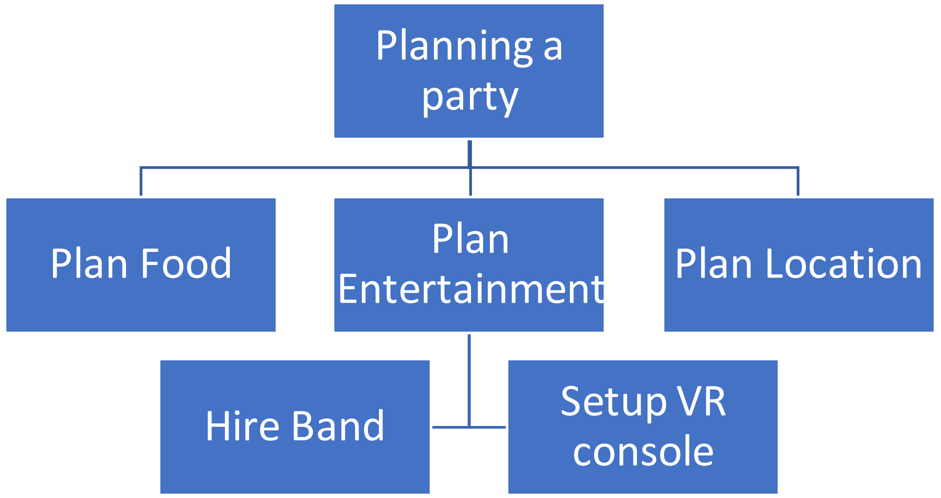
Testing Levels: Unit Tests

–

We’re supposed to be a unit!

What’s a unit test?

- Automated test
- Fully Controls all the pieces it is testing
- Are isolated
- Are independent of each other
- Runs in memory
- Is consistent
- Fast
- Tests a single concept/class
- Readable
- Maintainable



```
public class AuthenticationService {

          void authenticate(String token) {
              …
          }
}
```

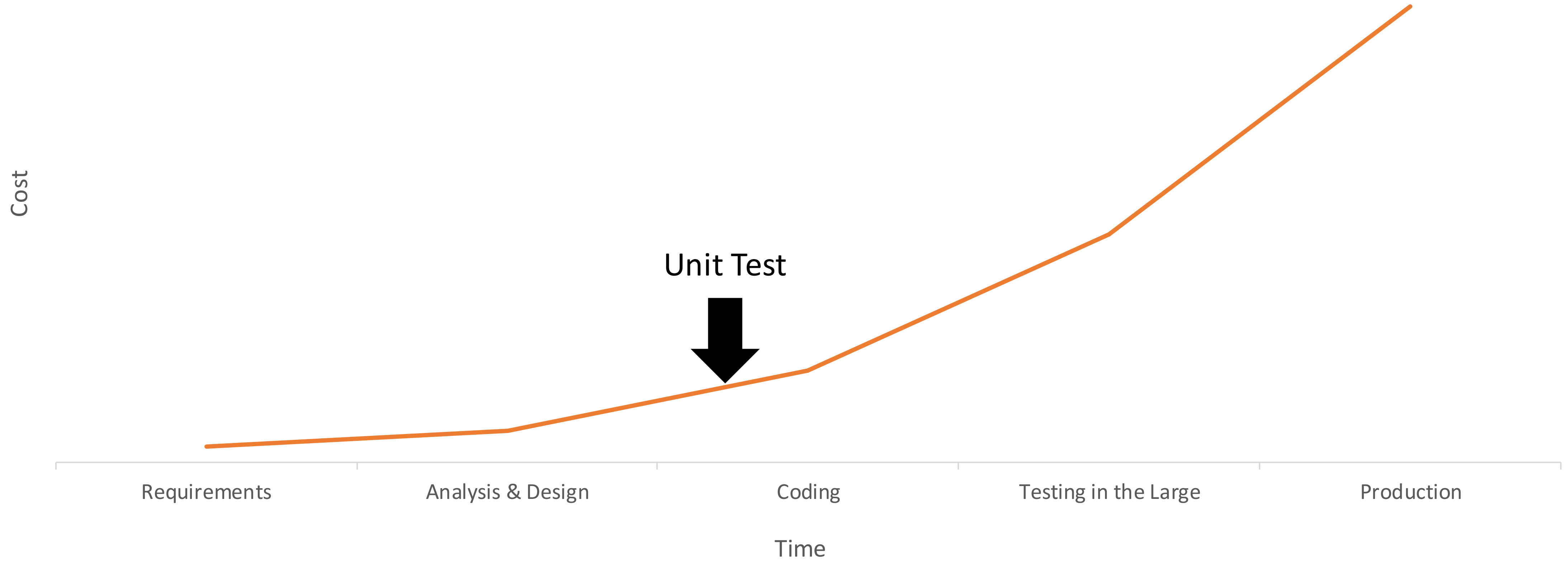
AXXES_

Why write unit tests?

- Short term loss, long term gain
- Are owned by the team and are everyone’s responsibility
- Super handy during development/bug fixing/refactoring
- Enables frequent integration
- Small units == less likely to make mistakes

AXXES_

COST OF CHANGE GRAPH



Testing Levels: Integration Tests

–

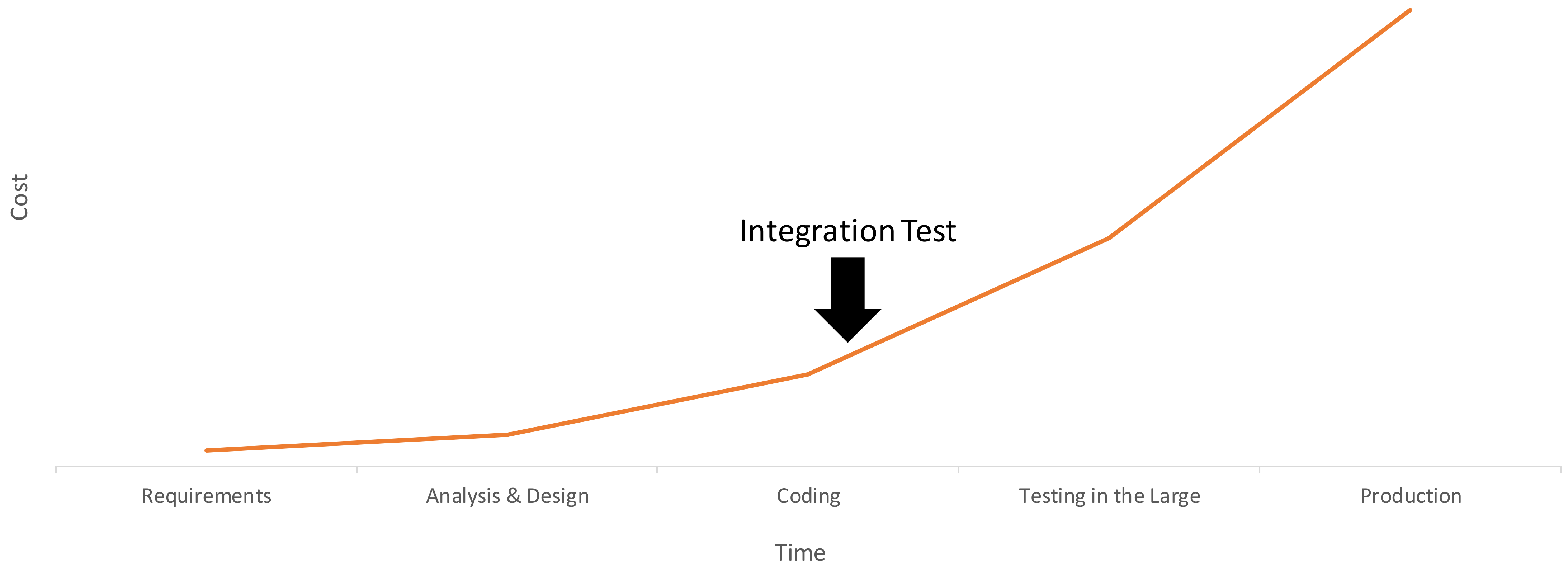
Verifying the interaction between components

What’s an integration test?

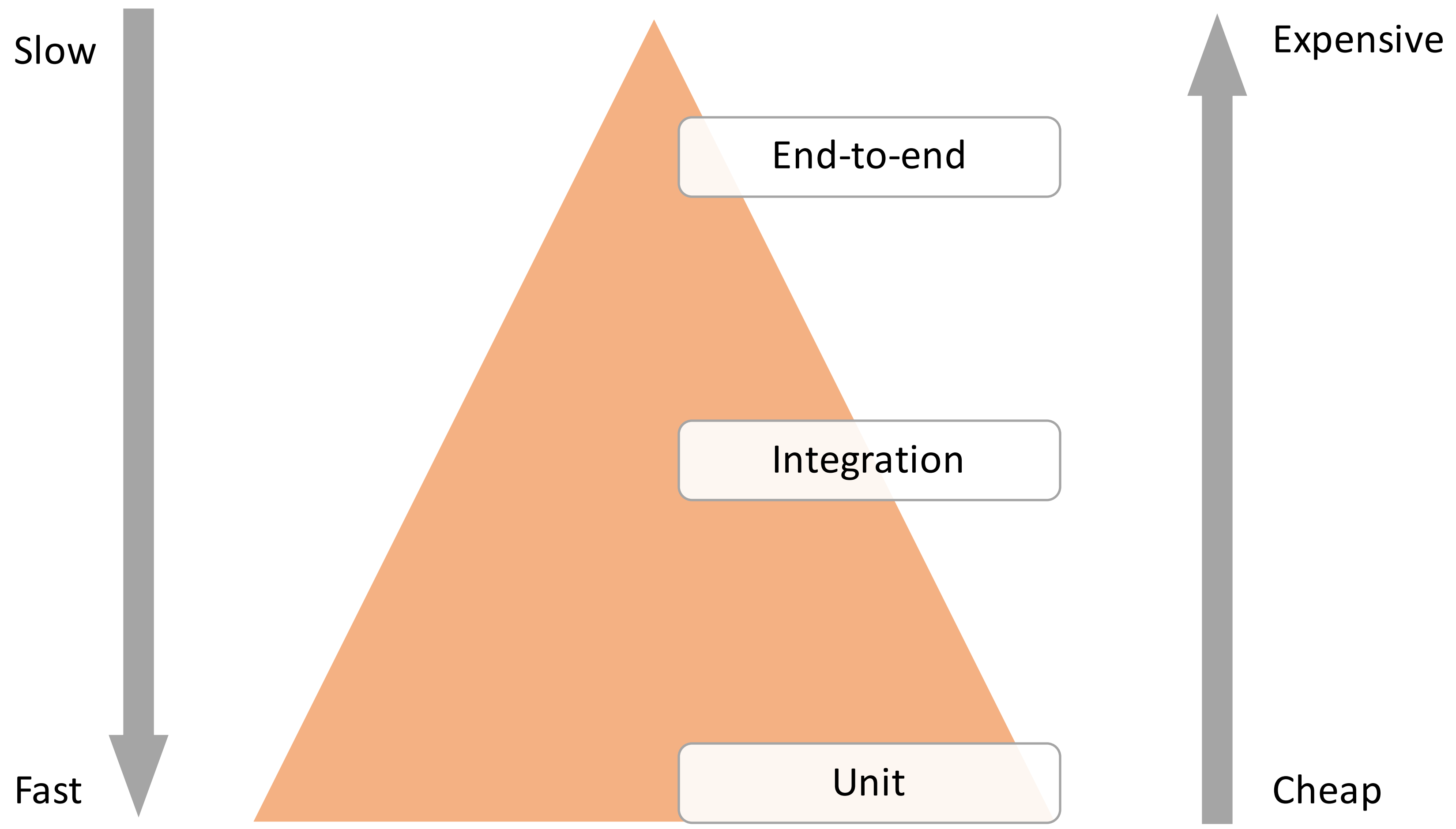
- Automated test
- Integration of multiple units/components
- No mocking, or at least as little as possible
    - As a result: less control over individual components => input/output validation
- Slower than unit tests
- More complex than unit tests
- Easily mappable to user stories
- Regression detection
- Harder to maintain
- Documentation

AXXES_

COST OF CHANGE GRAPH



Test Automation Pyramid



Testing Plain-old Java Code

–

Arrange – Act – Assert

Testing Plain Old java code

- Using testing frameworks and tools
  - JUnit
    - Practically the standard when it comes to testing frameworks for Java
  - Mockito
    - Library that allows the creation of mock objects which allow for easy control of components in the unit test
  - AssertJ
    - Fluent-style assertion library
    - Boasts wide adoption
      - Specific implementations for specific cases (e.g. XMLAssert)

DEMO & EXERCISE

–

DIY

Test Driven Development

–

It’s a never-ending cycle

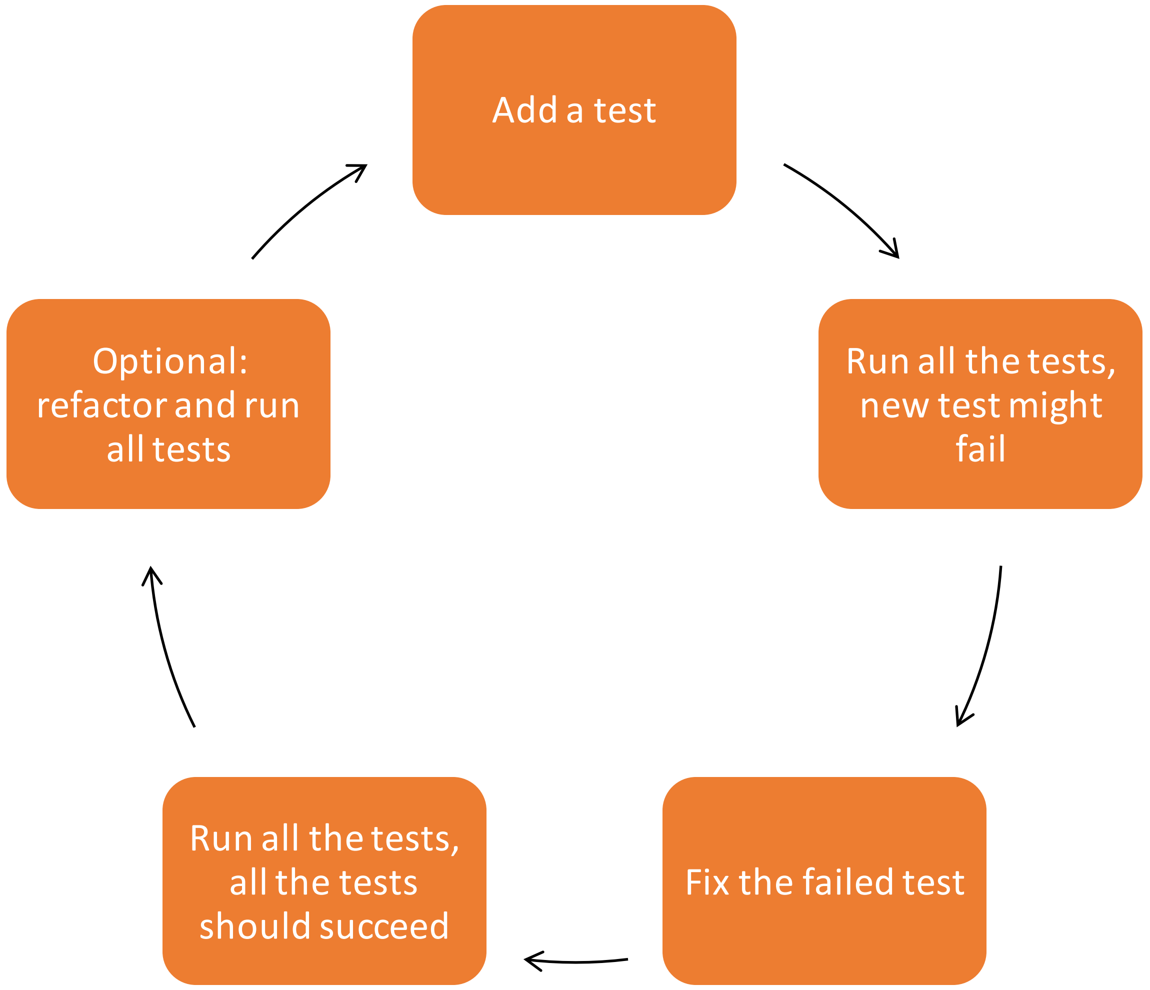
“Test-driven development (TDD) is a software development process relying on software requirements being converted to test cases before software is fully developed and tracking all software development by repeatedly testing the software against all test cases.”

WIKIPEDIA

In other words

- Convert requirements into test cases
- Write tests while writing business logic
  - Added benefit of having tests of old and new functionalities
  - Increase development speed
  - Track the status of the project by running all the tests
- Continuous cycle of the same steps

TDD Cycle

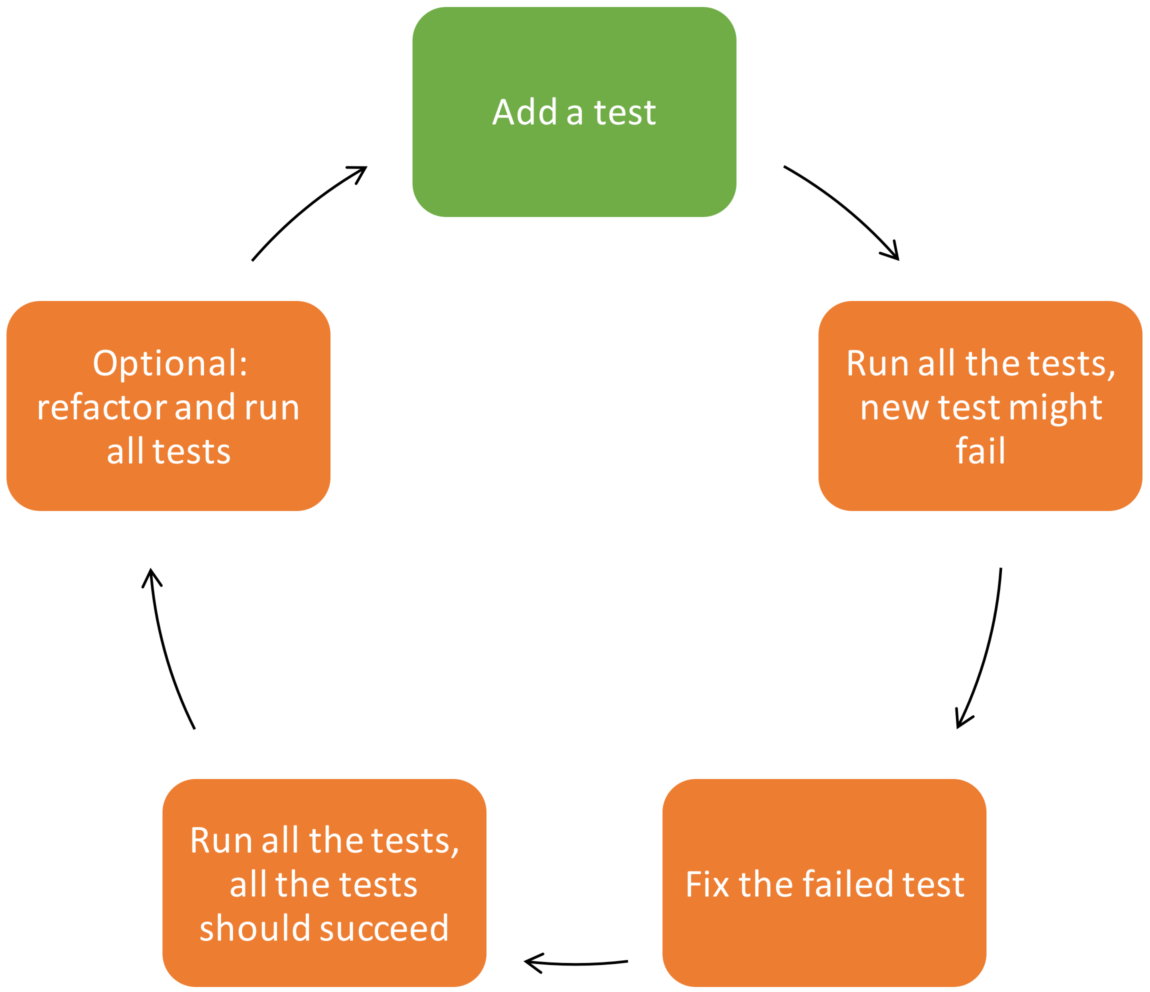

Add a test
Run all the tests, new test might fail
Fix the failed test
Run all the tests, all the tests should succeed
Optional: refactor and run all tests

Why the cycle?

- Increase coverage to near 100%
- Small steps
- Detect faults in the design/architecture
- Find bugs during development
  - Defensive programming

If all the requirements are converted into test cases and all the tests succeed,
all the requirements are (correctly) implemented

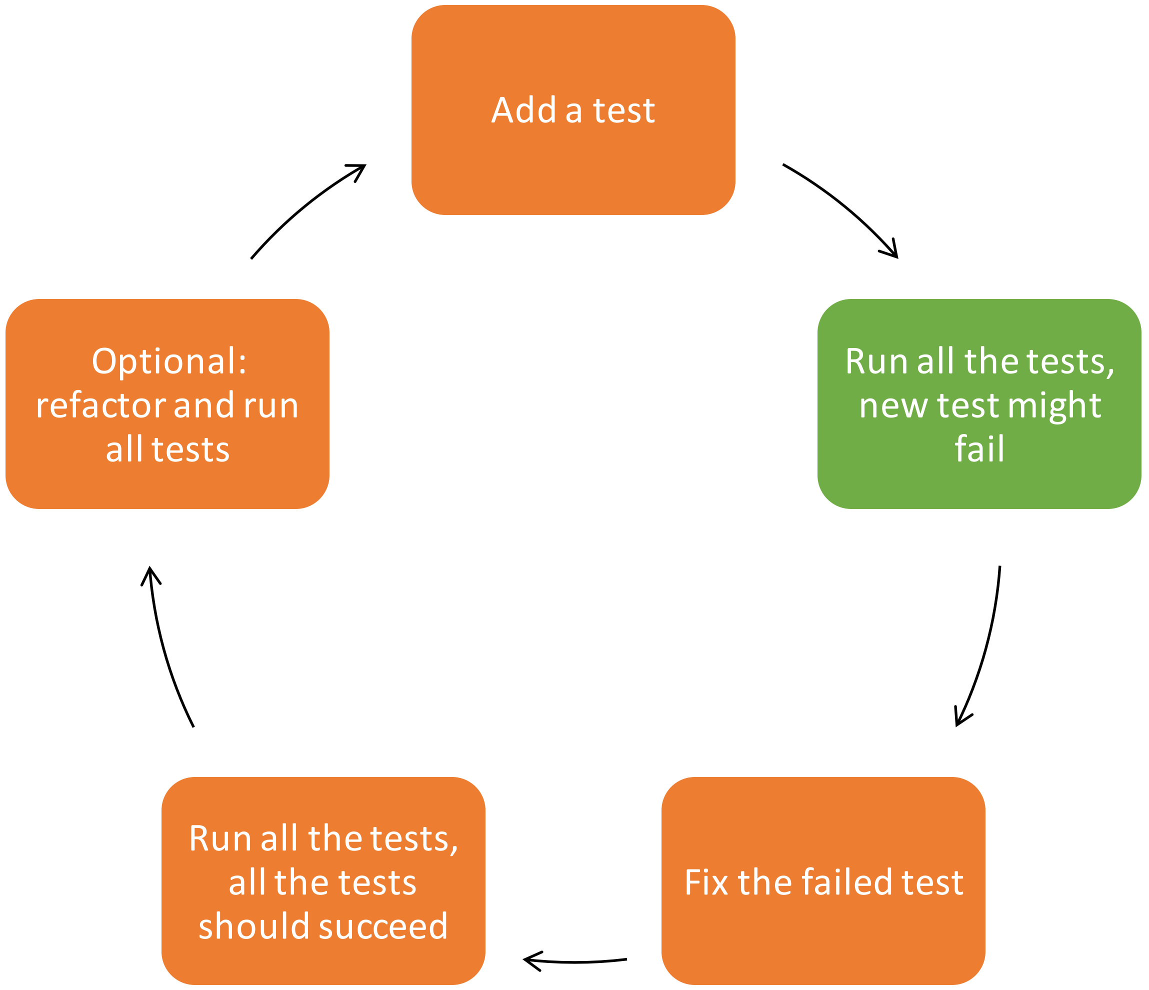
TDD Cycle


Add a test
Run all the tests, new test might fail
Fix the failed test
Run all the tests, all the tests should succeed
Optional: refactor and run all tests

Add a test

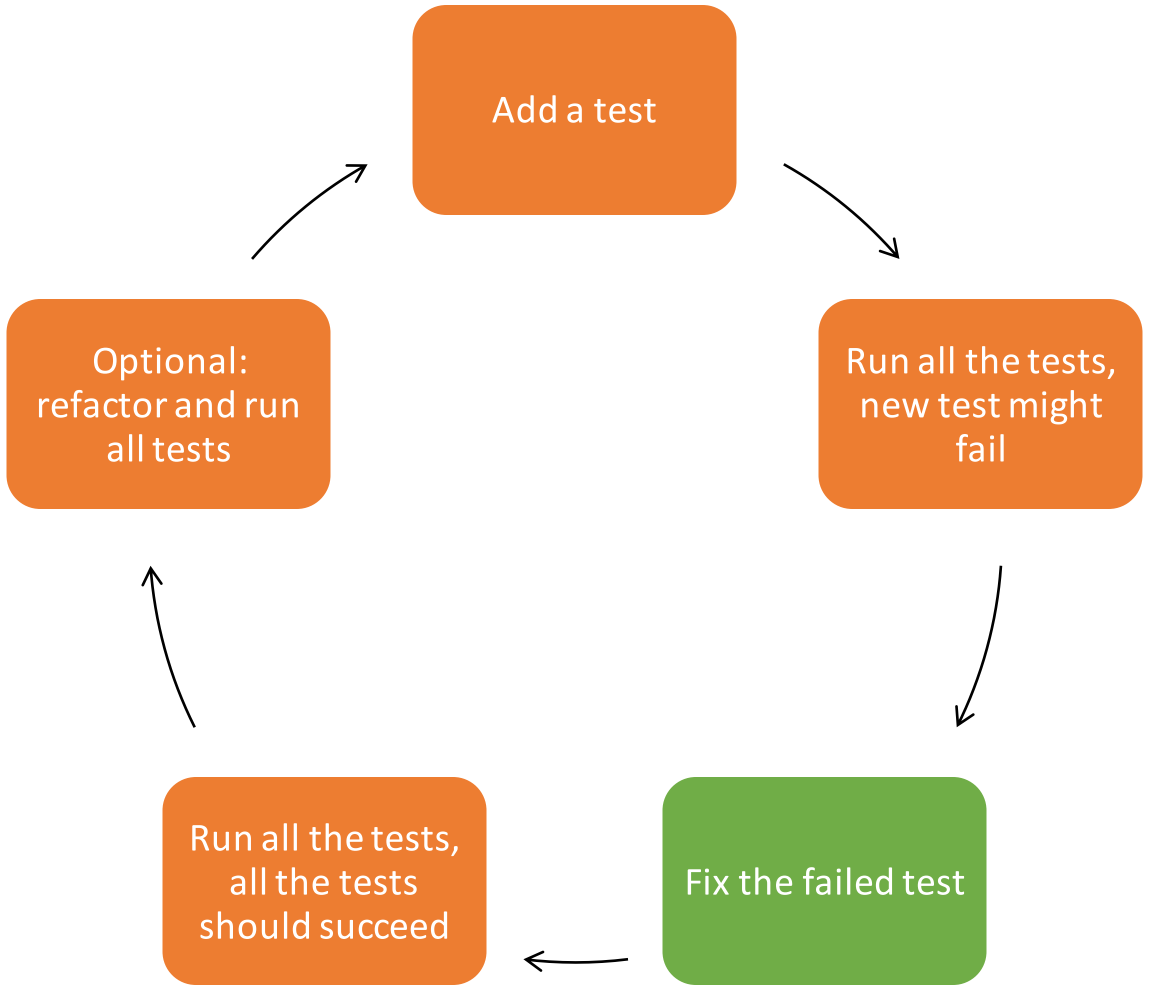
- Unit test
- Small increment
- One failing test at a time

TDD Cycle



Add a test
Run all the tests, new test might fail
Fix the failed test
Run all the tests, all the tests should succeed
Optional: refactor and run all tests

TDD Cycle


Add a test
Run all the tests, new test might fail
Fix the failed test
Run all the tests, all the tests should succeed
Optional: refactor and run all tests

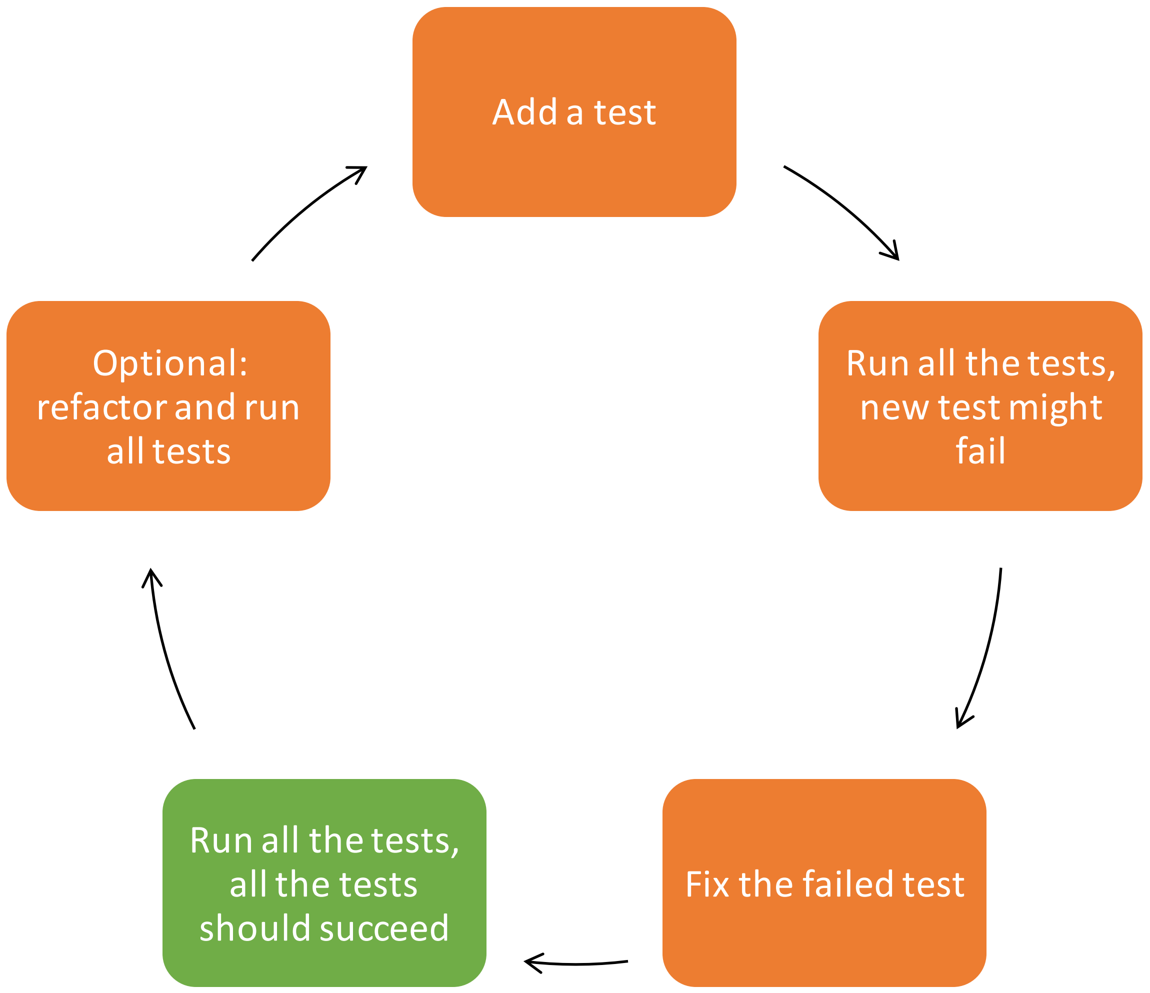
Implementing a test

- Strategies:
  - Fake it till you make it (Mocking)
  - Obvious implementation
  - Triangulation
    - Do not limit yourself to the happy path
    - Verification by elimination of non-happy paths

Triangulation in TDD

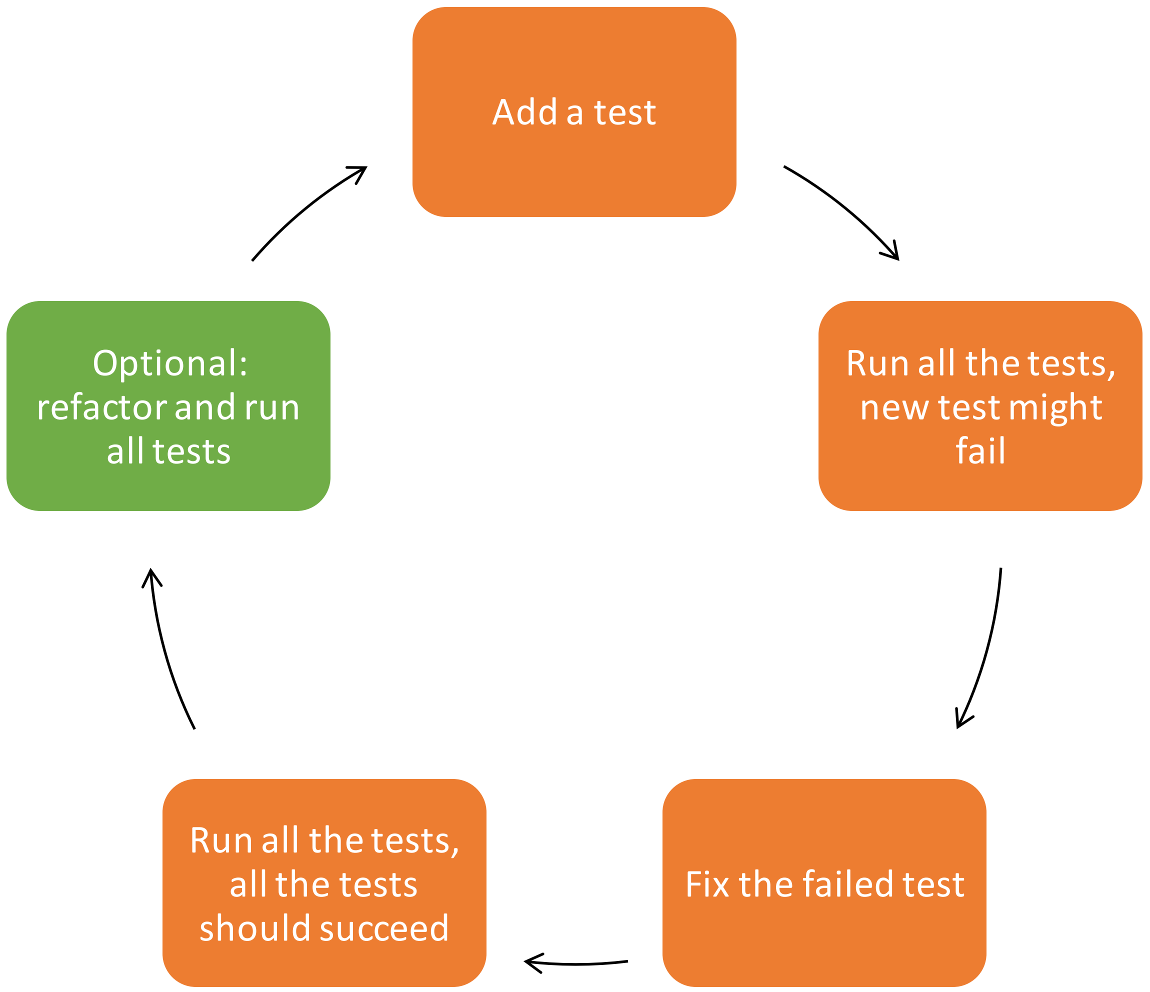
- Case: Testing a method which accepts parameter between 1 and 10
    - Write a test with parameter == 5
    - Write a test with parameter == 0 or 11
    - Write a test with parameter == null
    - Write a test with parameter == “Five”

TDD Cycle


Add a test
Run all the tests, new test might fail
Fix the failed test
Run all the tests, all the tests should succeed
Optional: refactor and run all tests

AXXES_

TDD Cycle


Add a test
Run all the tests, new test might fail
Fix the failed test
Run all the tests, all the tests should succeed
Optional: refactor and run all tests

Refactoring

- Rewrite existing code by:
  - Simplifying
  - Abstractions
  - Other small changes
- Alter the internal structure of a code block, w/o changing the external behavior
- Validate nothing broke by running all the tests again
- Arguably the most important step in the cycle

Clean code == happy team

AXXES_

Practical TDD

–

Implementing the cycle

Requirement

- Write a method `String greet(String name);` that does String interpolation with a simple greeting.

- Input: `name = "Bob"`
- Output: `"Hello, Bob"`

AXXES_

Requirement

- Handle `nulls` by introducing a stand-in.

- Input: `name = null`
- Output: `"Hello, there."`

AXXES_

Requirement

- Handle shouting. When `name` is all uppercase, then the method should shout back to the user.

- Input: `name = "JEFF"`
- Output: `"HELLO JEFF!"`

Requirement

- Handle two names of input. When `names` is an array of two names, then both names should be returned. (You can change the signature to `String greet(String... names);` to retain backwards compatibility)

- Input: `names = ["Jill", "Jane"]`
- Output: `"Hello, Jill and Jane."`

Requirement

- Handle three names of input. When name is an array of three names, then all names should be returned.

- Input: `names = ["Amy", "Brian", "Charlotte"]`
- Output: `"Hello, Amy, Brian, and Charlotte."`

AXXES_

Requirement

- Allow mixing of normal and shouted names by separating the response into two greetings.

- Input: `names = ["Amy", "BRIAN", "Charlotte"]`
- Output: `"Hello, Amy and Charlotte. AND HELLO BRIAN!"`

Requirement

- If any entries in `name` are a string containing a comma, split it as its own input.

- Input: `names = ["Bob", "Charlie, Dianne"]`
- Output: `"Hello, Bob, Charlie, and Dianne."`

Going a step further

–

Integration Testing

@SpringBootTest

- Annotation on a JUnit test to allow Spring functionalities
- Starts up the Spring Application Context
  - Allows you to autowire dependencies to write integration tests
  - Using the actual dependencies your service/components need
  - Minimal mocks, keep mocked beans to a minimum
    - Reduce the effectiveness of the integration test
  - Less control over the components
    - Good thing in this case
    - Control Input/Output using database assert tools/email clients/rest assertion tools

TestContainers

- Library to allow starting up docker containers for tests
  - Deep integration tests with other services
    - Only if your organization owns the other service
  - Easy setup for integration tests
    - Database
    - ElasticSearch
    - …
- Running on your machine
  - Allows you to really dig into I/O

RestAssured

- Replaces MockMVC for Rest Layer
  - Tests ran against a running service
    - Docker in CI!
    - Staging environments
- Performs actual HTTP requests
  - Request/response validation
  - Easily setup a blackbox testing framework for integrating projects
- Hamcrest matchers and JSONPath for validating response body

DEMO & EXERCISE

–

DIY

“The best day to start writing tests was yesterday”

- “The second best is today”

Thank you!

Questions?
cel.pynenborg@axxes.com

Entrepotkaai 10A,
2000 Antwerpen

Leonardo Da Vincilaan 9,
1930 Zaventem

Ottergemsesteenweg Zuid 808
bus 300 , 9000 Gent

T +32 3 23499.58
info@axxes.com

www.axxes.com