Testing

Goal

- Show you what type of tests we have in ORO codebase

Q: why do we need tests?

Why do we need tests?

- Reliability
- Better code quality
- Coverage of business use cases
- Continuous integration

Types of tests in Oro application

- Unit tests
- Functional tests
- Behat tests

Unit tests

- Tests for separate PHP classes
- No interaction with storages
- Mocks
- Minimum environment setup
- Do not require installed application
- Validate behaviour on class level
- Very fast (milliseconds)
- <bundle>/Tests/Unit

Functional tests

- Tests for backend functionality
- Interactions with storages (DB, search index)
- Minimum number of mocks
- Require application installed in test mode
- Validate behaviour on bundle and component levels
- Average speed (seconds)
- <bundle>/Tests/Functional

Behat tests

- Test for whole application
- Real application
- Real business use cases
- Require application installed in prod mode
    - Recommendation: separate application
- Validate behavior on the application level
- Slow (seconds-minutes)
- <bundle>/Tests/Behat

[Documentation]

How to run tests

- PHPUnit
- Run from CLI
- Run from PhpStorm
- Integration with PhpStorm

Demonstration >>>

Good tests

- Isolated, do not depend on each other
- Validate initial state
- Validate result
- Validate state after operation
- Covers positive and negative cases
- One operation at a time
- Describe what they are testing

[Example]

Bad tests

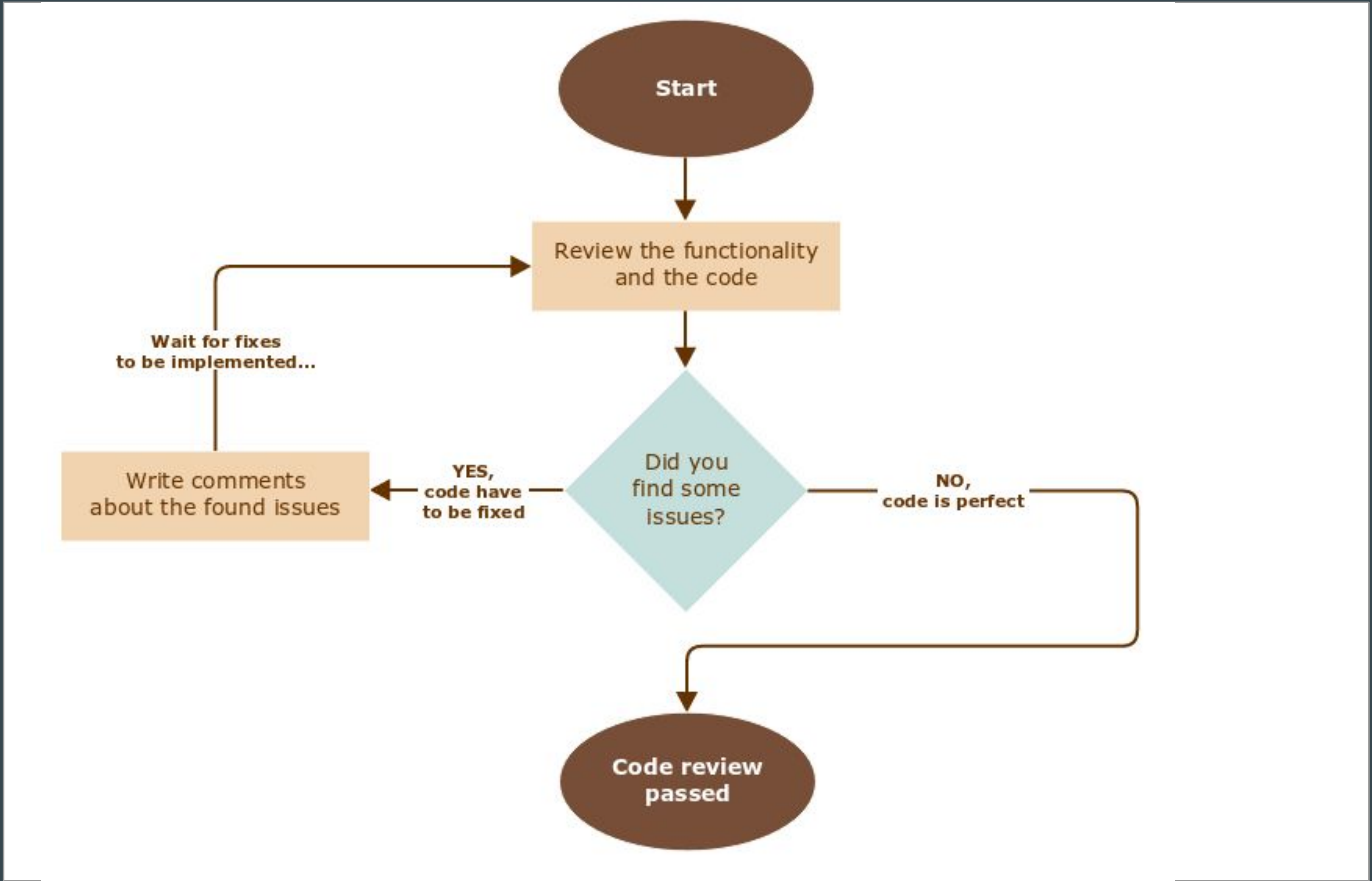
- Not isolated
- Don’t validate state or result
- Lots operations in one test
- Hard to read
- Hard to change
- Coupled

Code review

- Ensures quality of the code
- Indicates common issues
- Improves qualification of developers
- Experience sharing
- “Live review”
- Separate application for code review

[Best practices]
[Tips and FAQ]

Code review workflow


Start
Review the functionality and the code
Did you find some issues?
YES, code have to be fixed
Write comments about the found issues
Wait for fixes to be implemented...
NO, code is perfect
Code review passed

Code review checklist

- Functional review
- Architectural review
- Implementation review
- Automated tests
- Documentation

[Full checklist]