Object oriented analysis and modeling
Testable code

Robert Lukoťka
lukotka@dcs.fmph.uniba.sk
www.dcs.fmph.uniba.sk/~lukotka

M-255

Testing

Testing pyramid

- We need a lot of tests.
- It is good idea to automate most of the testing from the start of the project.
- We need a lot of test at the lowest level

OO design and object dependence

OO design is based on objects that interact.

Depencency types:

- creates, destroys
- calls method
- modifies
- ...

Unit test

Unit test

- Solitary - at lowest level we always test just one object
- Sociable - we may test an object together with a related object

Even if we use sociable unit tests, we want to severely limit the number of classes tested at once.

OO design and object dependence

So how do the dependencies affect our ability to write tests?

- creates, destroys
- calls method
- modifies
- ...

All these dependency types are just bad!!!

Dependency injection and dependency inversion

- An object should not create other object that is not closely related, it should get its collaborators via dependency injection.
- The object should not depend on collaborator's implementation, there should be an interface depending on the essence of the collaboration.

OO design and object dependence

So what to do?

- creates, destroys - dependency injection
- calls method - interface
- modifies - interface
- ...

When we want to run tests we still need the collaborating objects.

Test doubles

Collaborations are behind interfaces. So we make an implementation of these interfaces for the purpose of testing - **test doubles**.

M. Fowler: Test doubles

When to write tests?

- After writing implementation.

When to write tests?

- After writing implementation.
- After the object is designed but before writing implementation.
    - We may use test during the implementation.

When to write tests?

- After writing implementation.
- After the object is designed but before writing implementation.
  - We may use test during the implementation.
- Before design.
  - Guarantees testable design.
- During specification by example.
  - Specification is unambiguous.
  - We avoid the step specification $\rightarrow$ test cases.
  - Harder to achieve readability for all stakeholders, but many tools emerge.

Executable specifications

- Example: Power
- Cucumber
- jnario
- nahodny link

Executable specifications

More readable:

- Easier to read for non-technical stakeholders
- Defines the object interface in a less readable way
- Requires non-trivial translation
- Guarantees testability of the object

Code:

- Hard to read for non-technical stakeholders
- Defines the object interface very clearly
- No or easy translation
- Guarantees testability of the object