Méthodes de Test
TP 2 — End-to-End Testing

Stefano Zacchiroli
zack@pps.univ-paris-diderot.fr

Laboratoire PPS, Université Paris Diderot

2013–2014

URL http://upsilon.cc/zack/teaching/1314/methtest/

Some project ideas

HTTP server

Implementation of an HTTP server, capable of serving static files and interacting with CGI scripts. References:

- `https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol`
- `https://tools.ietf.org/html/rfc2616` (full specification)

Spreadsheet with GUI

Implement a minimal spreadsheet (*a-la* LibreOffice Calc), which uses a GUI to interact with users.

Some project ideas (cont.)

SaaS calculator

Implement a simple calculator (arithmetic operations, trigonometry, base conversions, registries, etc.) that is used ~~in the cloud~~ as "Software as a Service": there is no user interface, but the calculator has an API (e.g. in REST style) with methods that exchange data in some machine-readable format (e.g. JSON, XML).

Batch source code formatter

Program that walk through the source code of a project and format all contained source files follow the project style guides.

- easy start: indentation, fixed language
- then: variable naming, directory structure, multi-language support, language auto-detection, etc.

Your goal: get to the walking skeleton

These are all complex projects — they are no "2 hour hacks."

Your assignment is to:

1. pick a project idea
   - feel free to propose other ideas you like, but validate with me before proceeding
2. kick-start TDD on it
   - understand the problem
   - broad-brush design...
3. ...create the walking skeleton
   - automate: build, ~~deployment~~,[1] end-to-end test
   - choose technologies as you go
4. complete iteration 0: make the first acceptance test pass

[1] you can ignore deployment automation for now, but *think* about how you would do it!

End-to-end testing

Remember that the first acceptance test should be end-to-end, i.e. it should exercise the system via its natural interfaces, and not via internal objects (as you did for unit testing).

To achieve that you should ask yourself the question *what are the system natural interfaces?*, and exercise *those*.

GUI end-to-end testing

Testing GUIs is a delicate topic. Depending on the language you have several toolkits, based on different approaches to GUI testing (e.g. structure introspection vs screenshots)

For Java, here are some tools you might want to start from:

- Swing introspection:
  - `http://www.uispec4j.org/`
  - `https://code.google.com/p/windowlicker/`
  - `https://java.net/projects/jemmy/`
- Screenshot based:
  - `http://www.sikuli.org/`