01Lab
Requirements, Acceptance Testing, and BDD/ATDD

Mirko Viroli
mirko.viroli@unibo.it

C.D.L. Magistrale in Ingegneria e Scienze Informatiche
Alma Mater Studiorum—Università di Bologna, Cesena

a.a. 2024/2025

Again on labs at ASMD, and exam

Lab

- each module has a lab activity
- in each lab we propose two class of exercises: "operational steps" and "R&D tasks"
  - operational steps are designed to be quick exercises completed during the lab
  - R&D tasks involve more in-depth "research and development" activities, similar to exam projects, and typically require additional work outside the lab.
- the R&D tasks are **not** to meant to be **all** completed
  - just pick one, or two, or more, depending on your interest and time :)
  - you can start in the lab, and continue at home
- students work alone, or in small groups (typically, in pairs)
- the teachers provide assistance at need
  - during the lab, or later (offline)
  - share with the teacher interesting results or completed tasks, as soon as you have them
  - send email with subject "`[ASMD24-LABXX-TASKYY]`" with authors/repo in the body

Exam

- assume $\sim$ 90/100 hours of work (additional to labs)
- discussion of tasks completed by students, e.g.:
  - one big task, producing a software artifact (it can later become a scientific paper)
  - various smaller tasks
- discussion of links to other parts of the course

One slide sum-up on Acceptance Testing

Requirements and specification

- a requirement is a stakeholder's expression of a need/wish with regard to the system
- a specification is a description of system behaviour required to fulfil a requirement

ATDD/BDD

- acceptance testing: testing on overall system to determine approval by stakeholders
- ATDD: determine/pick an acceptance test, TDD to make it pass
- BDD: essentially ATDD, using behaviour-oriented tests vs acceptance tests
- in ATDD/BDD, use tests expressing "features"

Gherkin/cucumber

- Gherkin: language to express features (mixing prose/keywords)
- Cucumber: binding Gherkin to code execution (of "steps")

Starting point and goals

References

- **Cucumber**: `https://cucumber.io`
- **Gherkin**: `https://cucumber.io/docs/gherkin/reference/`
- **01-repo-atdd**: `https://github.com/mviroli/asmd23-public-01-atdd`
- LLMs:
  - **ChatGPT**: `https://chatgpt.com/`
  - **Perplexity**: `https://www.perplexity.ai/`
  - **Claude**: `https://claude.ai`
  - **Gemini**: `https://gemini.google.com/app?hl=it`
- Slide 01b provide rather complete examples of Gherkin syntax

General goals

- be operative with Cucumber/JUnit/Java/IntelliJ
- exercise writing requirements with Gherkin
- exercise ATDD
- pre-check ability of LLMs to deal with Gherkin
  - this is still very preliminary, we will deepen this in the future

Operational Steps – Cucumber and IntelliJ

Step 1 – Get ready

- clone the repo `01-repo-atdd`: `https://github.com/mviroli/asmd23-public-01-atdd`
- open it in IntelliJ
- Install the Cucumber plugin in IntelliJ
  - Both for Scala `https://plugins.jetbrains.com/plugin/7460-cucumber-for-scala` and Java `https://plugins.jetbrains.com/plugin/7212-cucumber-for-java`
- run the tests
  - Both JUnit and Cucumber tests
  - For Cucumber, go to `src/test/resources` and run the feature files in the `features` directory

Step 2 – Play with Cucumber and ATDD

- Add operations (e.g., multiplication) to the calculator
- Write Gherkin specifications for the new operations
- Implement the new operations
- Repeat the process for other operations :))
  - Follow the ATDD process!!!
  - Does everything work as expected?

Tasks R&D

TOOLING

Experiment with installing/using Cucumber with Scala and/or in VSCode. Is VSCode better at all here? Does Cucumber play well with Scala 3?

REENGINEER

Take an existing implemented small app with GUI, e.g. an OOP exam. Write Gherkin specifications explaining what the system is expected to do, and make acceptance tests pass. Does the system need a refactor of implementation? What does it tell us about how an application has to be designed to be easily acceptance tested?
Search here: `https://bitbucket.org/mviroli/oop2023-esami` (2023, 2022,...)

REQUIRE

Write Gherkin specifications that completely capture specification of requirements (functional, non-functional) of a real application. Might use a previous project of yours, with requirements already written down, or use any existing/hypothetical (small) application. What are good/bad aspects of Gherkin?

ATDD-LLM

LLMs/ChatGPT can arguably help in write/improve/complete/implement/reverse-engineer a Gherkin specification. Experiment with this, based on the above tasks or in other cases. Are LLMs useful here? Among the LLMs, which one is better?