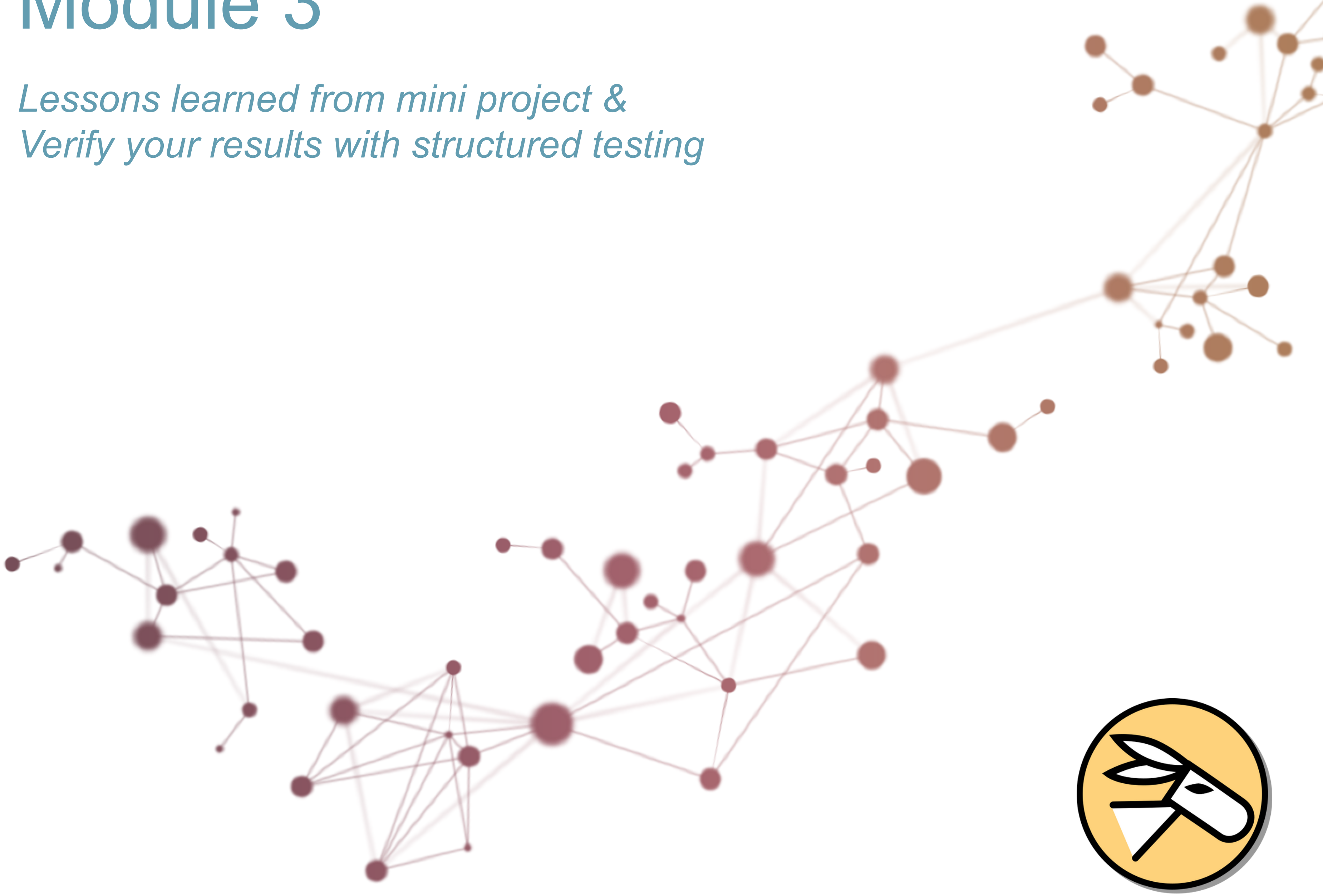
Module 3

Lessons learned from mini project &
Verify your results with structured testing



Lessons learned: Tops

Things you did great:

- Friendly contact with your client
- Ask context of system. This will help to retrieve better requirements
- Show stuff as soon as possible. E.g. mock ups. It will get the ball rolling
- Interactive contact with your client as soon as possible.
- Great mock up’s. Special mock up tools help
- Ask priorities (what to do first).
- Speak the language fluently (or arrange someone that can)
- Chase your client to respond/decide
- Using a structured method (e.g. SR2.0) to get requirements clear

Lessons learned : Tips

Things you can improve:

- You do not have to hesitate about asking about the price
- Obvious copy paste errors e.g. "We are happy to inform you that your building permit request has been denied.“
- Early clear & friendly push back on new requirements
- Say Yes to new requirements, but put them on backlog/new sprint
- Suggest new requirements, but for a price or used as bargaining.
- Have a signed contract/proposal from the start
- Discuss issues with your client first before going over his head
- Bring bad news as soon as possible.
- Make Profit ㅁ price high, cost low (but not too low ㅆ)
- Use flow diagramming properly (it is not just a way to draw pictures)
- <u>Interact with client synchronously (in person)</u>

Lessons learned : the interesting stuff

Funny, interesting moment to learn from:

- Humour may help in relationship: *“No costumes then?”*
- Eavesdropping
- *“Let’s talk about the price first”*
- Why it was sometimes hard to schedule something
- Asking what was explained just seconds ago
- *“You’re the only one that could reach this deadline”* ▭ *“Than that will be 10 million euro’s”*
- Keep lingering in Edwin’s room just to check something
- More requirements in shorter deadline changes the price

Lessons learned from mini project: Questions

- How much would Bill have accepted?
- How much would Bill have payed?
- Which deliverables do you need?
  - Client control: e.g. contract (what should that contain?)
    - What will be achieved
    - At what price
    - At what time
    - What if something changes?
  - Reqs (always) because this defines what to achieve
  - Design (in user interactivity always, optional when fully automated)
  - Realization (always)
  - Test (always)
  - Acceptance (always but can be part of contract)
  - Deployment/Delivery (always)
- How to handle fixed contracts with agile approach?
- ………….
- ………….

What is structured (functional) testing

Test if a process produces the agreed upon result by creating one or more test cases. Each test case has a:

a. **Start state:** the state the system need to be in before the test script(s) can be executed (usually some data is already in the system).
In many cases there is just one start state for all test cases.

b. **Script:** a set of non conditional steps to execute

c. **Predicted outcome:** the actual result that should be there after the script is executed.

Write this down. You need it later during the exercises.

Test design techniques

As described in http://en.wikipedia.org/wiki/Black-box_testing and described in this lecture:

- Decision table testing: e.g. If A then B else C (2 test cases)
- Boundary value analysis: e.g. If A < *boundary* then C else D (3 test cases)

Not described in this lecture

- All-pairs testing
- State transition Analysis
- Equivalence partitioning
- Cause–effect graph
- Error guessing

Most used test design techniques

Decision table testing:

1. E.g. If A then B else C (▭ 2 test cases)
2. E.g. If A1 or A2 then B else C (▭ Question: How many test cases?)
3. E.g. If A1 or A2 or A3 then B else C (▭ Question: How many test cases?)

Boundary value analysis:

4. E.g. If x < 10 then B else C (▭ 3 test cases)
5. E.g. if 5 < x and x < 10 then B else C (▭ Question: How many test cases?)

Brainstorm: What will require less testing effort?

Which version would cost more testing effort (blue or purple)?:

```
if A1 then B1
else if A2 then B2
else if … etc;
```

```
if A1 then B1;
if A2 then B2;
if … etc;
```

Class exercise: *Which test cases are needed for the following (part of the) specification (assume no start state)*:

{ If **<u>"Order contains deposit products"</u>**

then *'You can retrieve the deposits on the deposits products when you return the packaging.'* }

Context info:
An order has multiple order lines with chosen products. Some of those products are "deposit products" (in Dutch "producten met statiegeld").
This piece of specification defines a conditional result on the invoice.

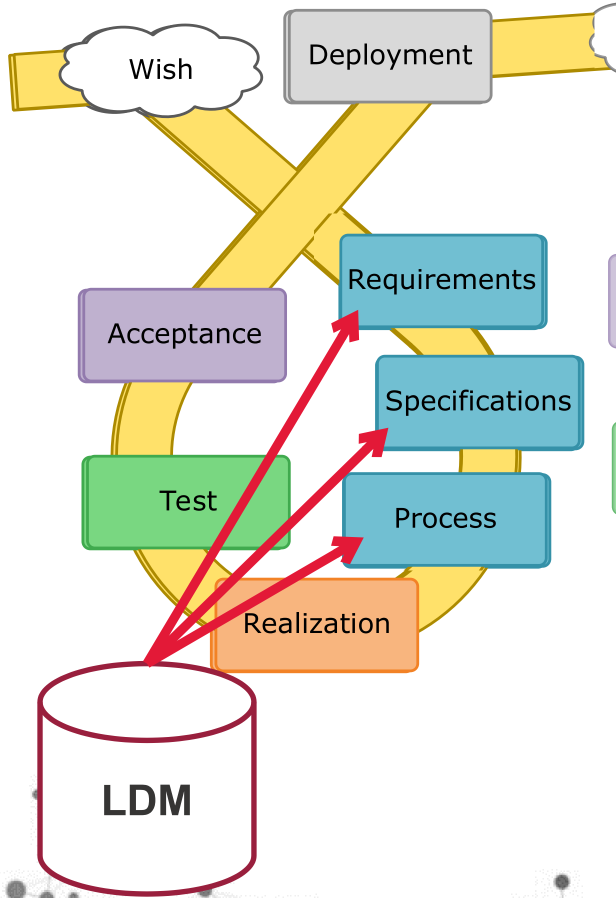
Answer (as Test cases)

1. *Test case: No deposit products delivered at all*
   - ***Start state: customer is logged in***
   - ***Script:***
     - *Create an order*
     - *Create two order lines but don't choose deposit products*
     - *Finish order (the system creates the invoice)*
   - ***Predicted outcome:*** *An invoice* ***without*** *the text "You can retrieve the deposits …"*

2. *Test case: Only deposit products*
   - ***Start state: customer is logged in***
   - ***Script:***
     - *Create an order*
     - *Create two order lines, both with deposit products*
     - *Finish order (the system creates the invoice)*
   - ***Predicted outcome:*** *An invoice* ***with*** *the text "You can retrieve the deposits …"*

3. *Test case: Some deposit products and some are not*
   - ***Start state: customer is logged in***
   - ***Script:***
     - *Create an order*
     - *Create one order line with deposit products and one order line with deposit free product*
     - *Finish order (the system creates the invoice)*
   - ***Predicted outcome:*** *An invoice* ***with*** *the text "You can retrieve the deposits …"*

Testing linked to Reqs, Specs, Process & LDM



- ***Do we create test cases for (functional) requirements?***
  - *No, but why?*
  - *Because the functional requirements always have specifications*
- ***What part of the test case can be created from the specs?***
  - *Predicted end result*
  - *Start state*
- ***Does one spec have one predicted end result?***
  - *No, it has one or more*
- ***What part of the test case can be created from the process?***
  - *Test scripts*
- ***Can you use the LDM and if so how?***
  - *Yes, to see how information is defined.*
  - *But for actual testing you need more. What?*
  - *You need to know how the information of the LDM is actually stored (e.g. in a table, file, message) and….*
  - *How to check the actual end result (e.g. database browser, system screen, file viewer, etc.)*

Module 3 Assignment

1. ***Create test cases:*** *Based on next sheet “Requirements to test”, create a set of test cases in a docx or pdf file. The test cases should be optimized for efficiency. For instance by creating test cases that test many things at once. Only include a maximum of 6 test cases. If you have to choose, choose the ones that test the most functionality and/or the ones you think Bill will find the most important.*

2. ***Perform those tests with (system of) other group:*** *Group 1 will test (system) of Group 2, Group 2 will test system of Group 3, …, and Group 10 will test system of Group 1. E.g. Group 1 will give the test case and Group 2 will tell or show what the actual outcome of their system would have been. Tip: you can make this a paper exercise but it is much more fun to do this live with the other group. Document the test result in the test cases document after the predicted outcome as “Actual outcome”.*

3. ***Solve ambiguities:*** *you may find ambiguities which you did not yet solve during you sessions with Bill. Solve the ambiguities you find by adding assumptions in a separate docx/pdf that make these ambiguities clear.*

4. ***Upload the 2 documents before 18th of March 23:59 on https://brightspace.ru.nl/ under same Mini-project assignment***

Requirements to test

Create a permit system that makes it possible for a CIVILIAN to enter a request for a building permit. He needs to enter:

- *The construction start date of the building and*
- *The height in meters (`integer`)*

The system will automatically add the following information:

- *The REQUESTOR (which is CIVILIAN who entered the request).*
- *The current date (with function `currentDate()`)*

Expand the permit system with an automatic approval if the height of the building is between 3 and 5 meters. If not an approval by an APPROVER.

Expand the permit system with an message send to the REQUESTOR informing him / her of the result (approved or not).