Project

Capita Selecta Software Engineering

Maja D'Hondt

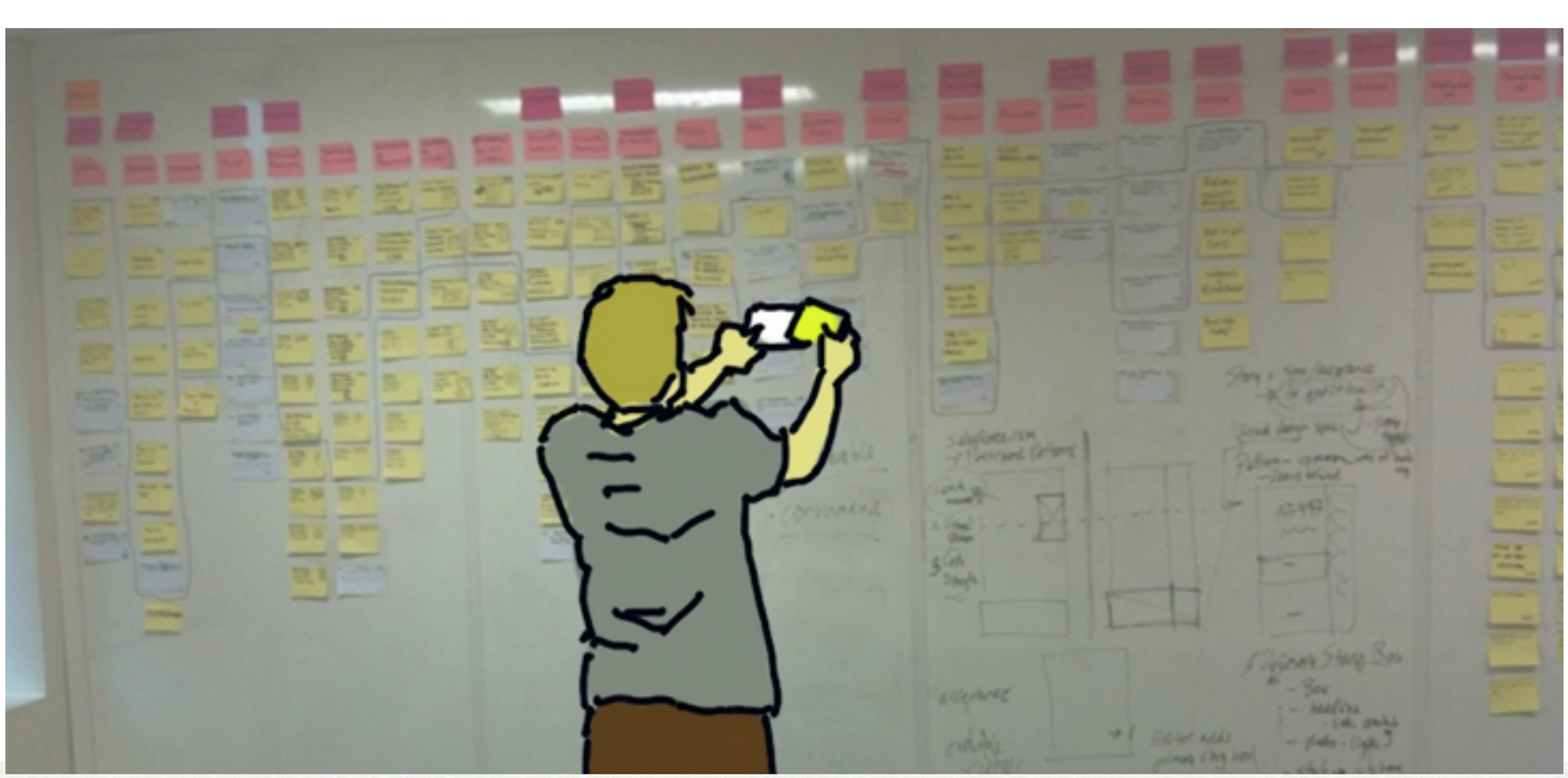
2015

Project Goal

- improve an existing program in a high-level programming language (e.g. C, C++, C#, Java, Java for Android, Objective-C, Scheme, ...) by means of automated unit testing and refactoring
- you can select a program that you developed in the past or an open source program
- I have to approve the program before you can use it for the project
- you can do the project alone or with a team of 2 people
- the time you have to spend on the project is 80-100 hours per person, i.e. 2 to 2,5 work weeks
- write a short description of the software of your project (1/2 page or more) with optionally screen shots and other supporting material (=deliverable)
  - what does it do, on what hardware does it run, what is the programming language, what are the main components or parts (including relevant third-party components), did you implement (parts of) it, ...?

Project steps: 1. story map

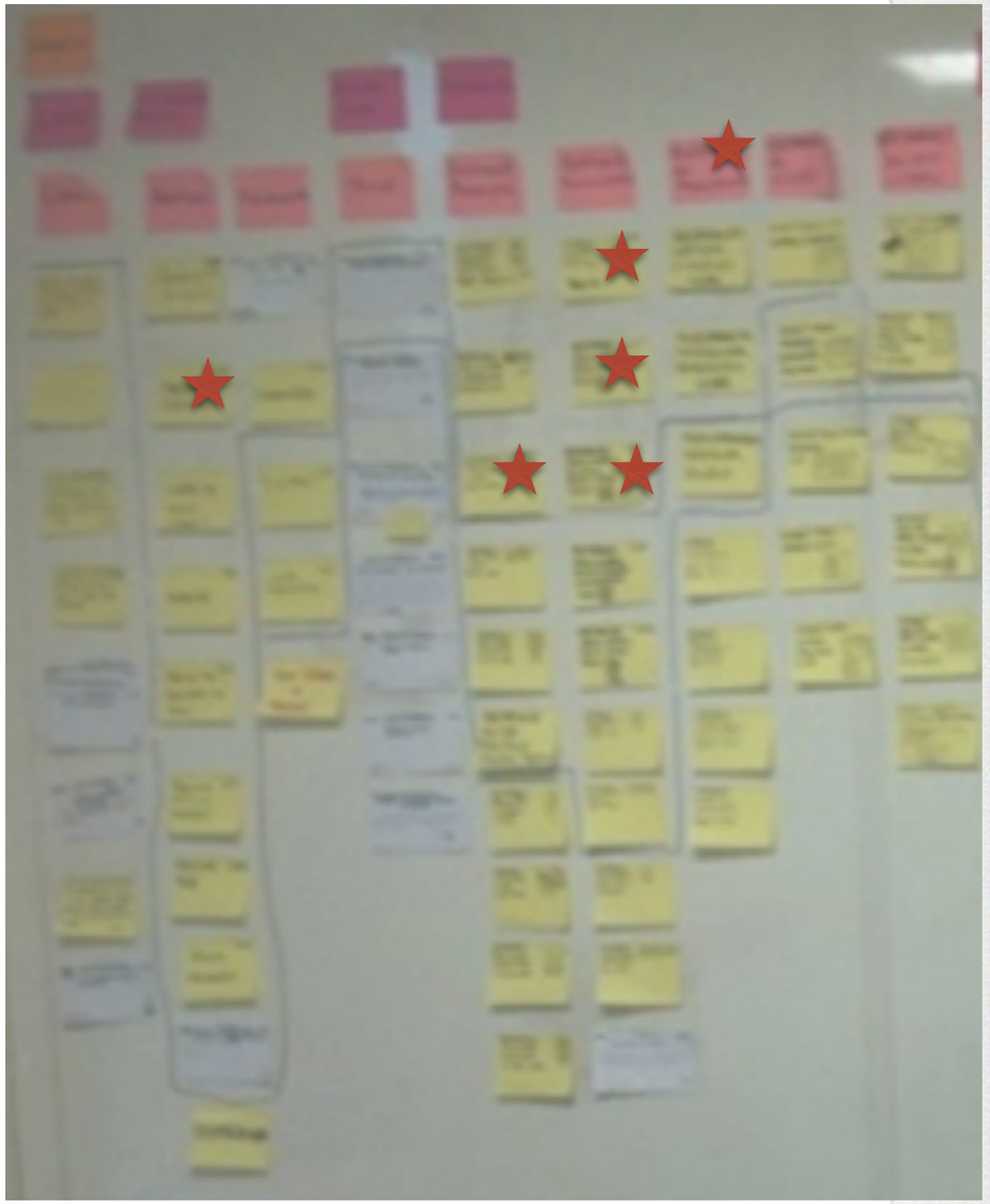
- create a story map of the software of your project (=deliverable)
- note that the story map is probably constructed retroactively if the software already exists, whereas a story map is usually a tool to map the functionality that has to be developed in a software project
- make the story map detailed enough: use the guideline that a user story (i.e. detailed card belonging to an activity step, or yellow cards in picture) should correspond to an activity that is typically executed without interruption; since this is not a clear definition, you will have to validate the level of detail of your story map in subsequent project steps and iterate back to this step if further refinement is needed



Project steps: 2. identifying risks

- identify high-risk user stories (or entire activities or an activity step) and mark them (see example with a red star)
- what is a high-risk story, activity or activity step? it is a story etc. which is important (high business value) and/or in which an error in the intended use would make it impossible to use the software
- so this is about risks in the software itself, not about “project risks” such as “the risk of running out of time or money”; so high risk could be that data from smartphone is not correctly stored in backend, low risk could be that the layout of a screen is not correct in certain cases - it’s up to you
- remember that you are the product owner and have to decide business value

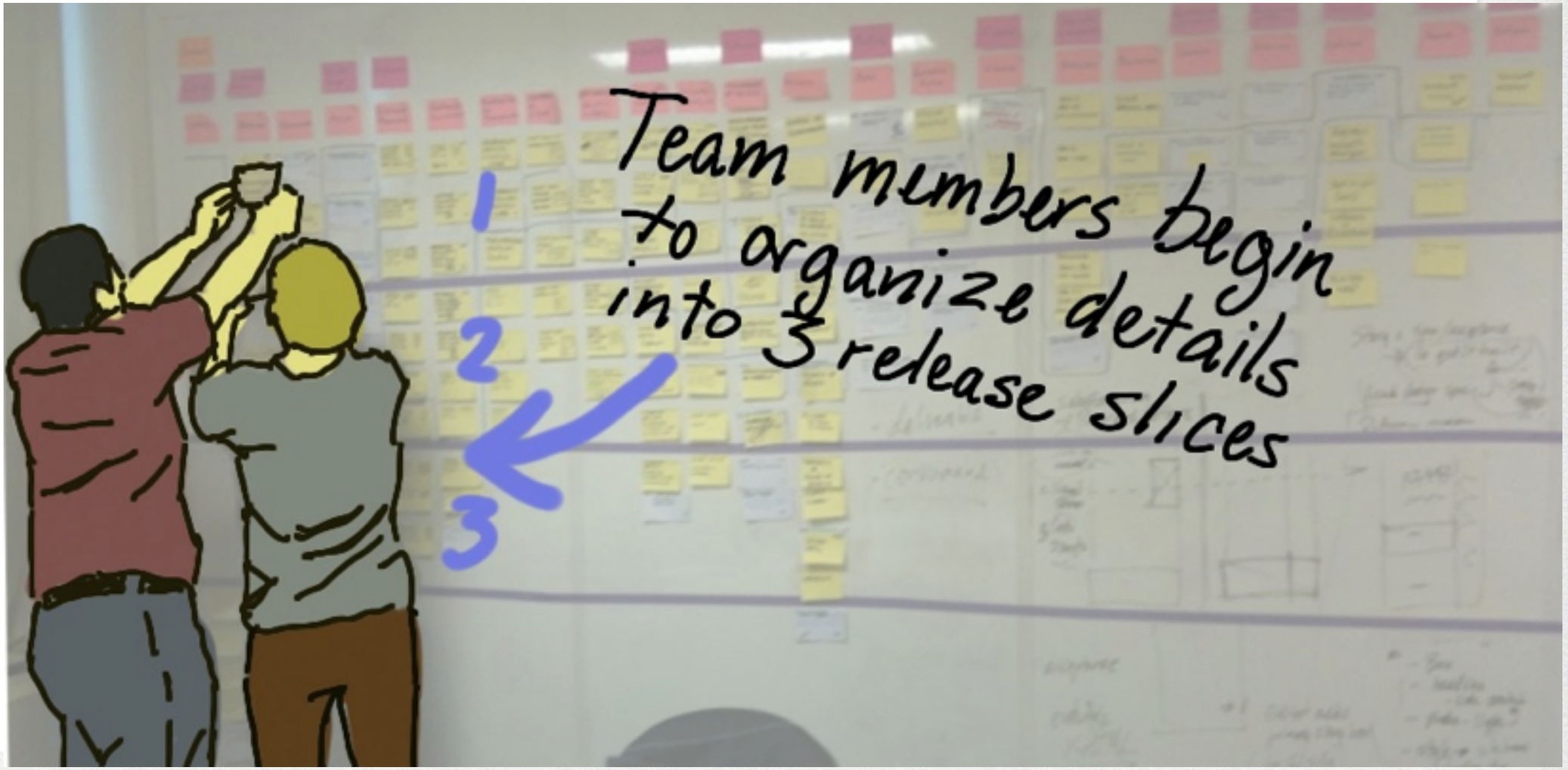
- practical: make a photo of your story map here because it is going to be rearranged in the next step (=deliverable)
- practical: if you iterate back to step 1 or 2 later on because you have to refine the story map and/or you identify new risks, then make that change in the sliced story map of step 3; in other words: a photo suffices, you don’t have to keep this version of the story map



Project steps: 3. slice high-risk stories

- slice the story map according to the identified risks
- make at least three slices corresponding to three levels of risks (e.g. high, medium, low)

- practical: for version management, take photo's of your story map's different versions (=deliverable)



Project steps: 4. make test strategy

- in this step I want you to think about the overall approach you're going to take in order to mitigate those risks
- risk mitigation in this project = writing automated tests
- so you will have to think about the "testability" of the software
  - e.g. the modularity of the software: does it have "units" that can be unit tested? or do you need to refactor? how are you going to refactor if you don't have tests?
  - e.g. the ease in which dummy input and output values can be created for testing
  - e.g. the ease in which a part can be decoupled from another part in order to test it, in other words how much test setup you have to develop?
  - ...
- and you will have to think about the test coverage of the software (see slides)
  - e.g. what are your guidelines wrt test cases in order to cover a part of the software under test?
  - …
  - this can vary from user story to user story

- take an agile approach to this test strategy:
  - think long enough about the overall test strategy and write it down in a document (=deliverable)
  - then dive into the highest risk and analyse it in more detail, then the next highest risk, etc.
  - note that you don't have to do more detailed analysis than what you will be able to develop in this project (that does not mean you cannot have more identified risks than what you can

Project steps: 5. make overall planning

- first part is to make an overview of you own time allocation to this project until the deadline
- second part is to consider the slices you made, which correspond to “releases” of this project
- third part is to map the first and the second part onto each other and come up with a roadmap as in agile software development, which has several milestones
  - note that you can iterate back to the slices and make more (3 is only an indication)
  - you will have one less milestone than you have slices (the 3rd slice is the backlog which will still exist, i.e. be undone, by the end of this project)
  - I think 2 to 3 milestones is a good target (last milestone is deadline of project)
  - note that in the context of this project the roadmap and milestones are at the level of releases or sprints; it is hard to do sprints since you will probably not work full-time for this project, so it’s more of a release roadmap
- the planning (=deliverable) will consist of the dates of the milestones, the available time you have between two milestones (= release/sprint), the work you plan to do in the releases/sprints without going into detail of each “user story”

Project steps: 6. make detailed planning

- this step details the work you plan to do in the upcoming release/sprint
- this means that you will have to make a detailed planning at the start of each release, i.e. 2 or 3
- the first planning describes in detail the work you plan to do, i.e. the detailed test strategy for each planned high-risk “user story”
- this includes estimates, this means breaking down the work in tasks of each max 1 day (8hours) of work

- for subsequent detailed plannings, review the previous release (=deliverable):
  - what is done with actual effort in addition to the estimated effort
  - what is not done and why, which problems or blocks did you encounter (e.g. underestimated, something unexpected came up, …)
  - note that it is up to you to decide if unfinished work of one release is carried over to the one you are currently planning

Project deliverables

- The following documents or source code are deliverables for the project. They have to be handed in at certain moments (see timing later on). These deliverables are prerequisites for participating in the exam.
  - Project description
  - Story map + risks
  - Story map sliced
  - Test strategy
  - Overall planning
  - Detailed planning v1, v2 and optionally v3
  - Reviews of detailed planning v1, v2 and optionally v3
  - Source code versions: the version of the source code of the program from which you start for this project, the intermediate versions and the final version
  - Final report = revision of last detailed planning

Source code versions

- Commit all versions of the source code of the program for your project in the version management system GitHub or a variant thereof.
- Give me access to this repository.
- v1 is the first version of the source code, i.e. the starting point for this project
- v2 etc are the versions of the next releases
- use vx.y for each high-risk user story
  - make minor versions, e.g. v2.1.3, e.g. after every unit is fully tested
- commit at each end of your work day (even if you did not work the full day)
- comment each commit with a reference to the user story for traceability from your planning

Timing

- Send all deliverables in electronic format (word or pdf or rtf or txt) to mjdhondt@vub.ac.be on the dates given below, unless indicated otherwise in an email from me.
  - **before** March 2, 2015 (can be combined in 1 document)
    - Project description
    - Story map + risks
    - Story map sliced
    - Test strategy
    - Overall planning
    - Detailed planning v1
  - date of your first milestone
    - Review of detailed planning v1
    - Detailed planning v2
    - source code (implicit deliverable because source code repository should always be up to date)
    - any changes to Project description, Story map + risks, Story map sliced, Test strategy, Overall planning
  - date of your second milestone (if you have it)
    - Review of detailed planning v2
    - Detailed planning v3
    - source code (implicit deliverable because source code repository should always be up to date)
    - any changes to Project description, Story map + risks, Story map sliced, Test strategy, Overall planning
  - deadline = **before** date exam minus 2 working days
    - Review of detailed planning v2 or v3
    - source code (implicit deliverable because source code repository should always be up to date)