Projecte d'Enginyeria del Software: Documentació i Comunicació

X. Franch, M.J. Casany, S. Martínez, J. Piguillem

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Outline

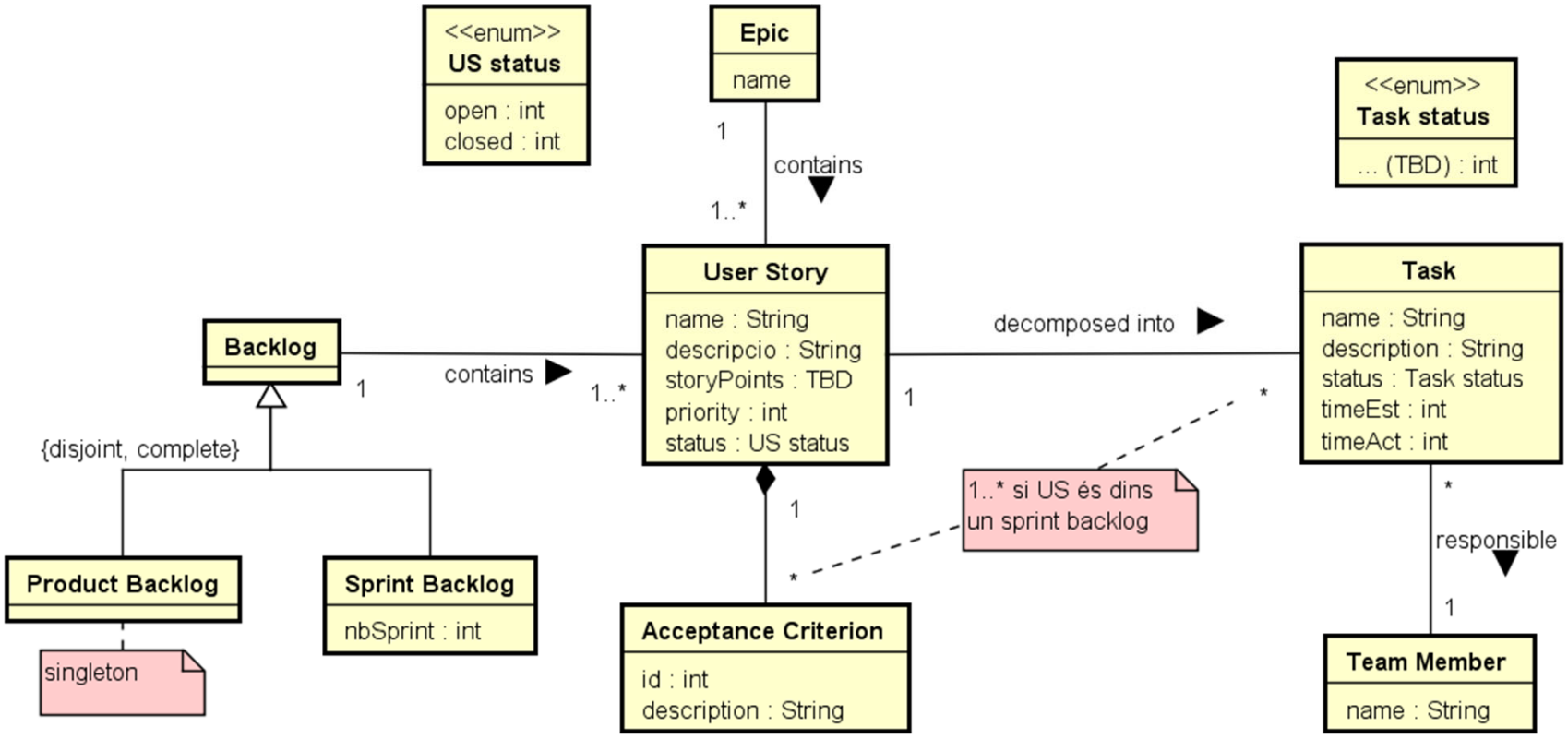
- Agile in PES
- Team management
- Methodology
- Models
- Third-party services
- Technical aspects
- Documentation and evaluation
  - General
  - Inception
  - Development sprints
  - Final documentation
- Final demo guidelines

AGILE IN PES

Agile in PES

- Two phases held: inception and development
  - Development phase with three sprints
- Only one release
  - We do not have release backlog, but only product backlog and sprint backlogs
- User stories
  - Grouped into epics
  - Decomposed into tasks
  - Measured with story points
  - Prioritised
- Agile ceremony
  - Planning, retrospective and review meetings
  - Daily scrum meetings: at least twice a week (at class)

Agile in PES


<<enum>>
US status
open : int
closed : int
Epic
name
1
contains
1..*
<<enum>>
Task status
... (TBD) : int
User Story
name : String
descripcio : String
storyPoints : TBD
priority : int
status : US status
Backlog
1
contains
1..*
decomposed into
1
Task
name : String
description : String
status : Task status
timeEst : int
timeAct : int
{disjoint, complete}
1
*
1..* si US és dins un sprint backlog
*
responsible
1
Product Backlog
singleton
Sprint Backlog
nbSprint : int
Acceptance Criterion
id : int
description : String
Team Member
name : String

TEAM MANAGEMENT

Roles

- Product Owner
  - your beloved teachers
- Each team member shall take one responsibility
  - Member 1: inception (the two parts)
  - Member 2-4: one development sprint each
  - Member 5: management of services produced to/consumed from other teams
  - Member 6: preparation of final demo and closing documentation
  - Groups with 7 members: split inception into two
  - Groups with 8 members: assign one particular task (e.g., continuous integration)
- Every responsible needs to make sure that his/her duties are well fulfilled
  - E.g., documentation uploaded in Racó at due time in correct format

Project Record Track

Date	Activity	Member	Duration	Comments

- activity: everything!! E.g., attending the weekly lessons, meeting held, infrastructure tested, …
  - include user story and task when applicable
- only one member per row → duplicate rows if needed
- separate sheet with summary, including statistics per member/iteration

Somehow redundant with other artifacts and software (e.g., Git, Taiga), but still requested for evaluation

Updated regularly from the first day! Sprint master verifies this.

Use google drive

Every single deliverable will include the link to the updated project record track (not in the pdf)

Be consistent in descriptions!!!

Communication

- every email sent to the teacher about PES, should have as prefix: "**[PES_*nom-projecte*]"** (square brackets included)
- kindly provide some mechanism to your teacher so that s/he can send an email and all the members of the team receive this email
- every document or repository should have as prefix **PES_*nom-projecte*** and then a self-explanatory name (e.g., *ProjectRecordTrack, Inception-2,* etc.)
- inform your teachers about any team problem you may experience **as soon as possible**

METHODOLOGY

Methodology

- It is necessary to set up a particular way of working
  - Set-up at the end of the inception phase
  - Revised and improved at the end of each development sprint (retrospective meeting)
- Several aspects to be covered
  - Backlog management, testing, ... (see next slide)
  - Tool-supported
  - A holistic vision is needed
    - A number of tools may be integrated
  - Recommendation: describe the relevant aspects that take place at every significant event and consequences on the different methodology dimensions:
    - Start project, Start sprint, Start user story, Initiate task, Manage bug, Finalize task, etc.

© PES@FIB (UPC)

Methodology dimensions

Mission	Example	Comments
Agile project management	Taiga (required)	Details given in a separate document
Code repository	GitHub (required)	Details in a separate document
IDEs	Tons of tools: IntelliJ Idea, Xamarin, …	There's a great variety; explain the reason for your choice
Software quality analysis	Sonarqube/SonarCloud Altres: jslint, CheckStyle…	Programming conventions also to be included here
Testing environment	Jenkins (automation) jUnit, … (unit tests) Selenium, … (interfaces)	It is important that you describe well the testing methodology, besides the tool
Issue tracker	Mantis, JIRA	You may also use git's or Taiga's
Communication	Slack, Discord	Preferred over Whatsapp
CI/CD	Jenkins (build) Docker (packaging)	Highly recommended, at least CI

Other methodological aspects

- Definition of done: user story granularity → life cycle
  - E.g.: specified → implemented → tested → documented → diagrams updated
- Conventions
  - E.g. vocabulary for commits, backlog elements, etc.
  - Cross-references among dimensions and tools

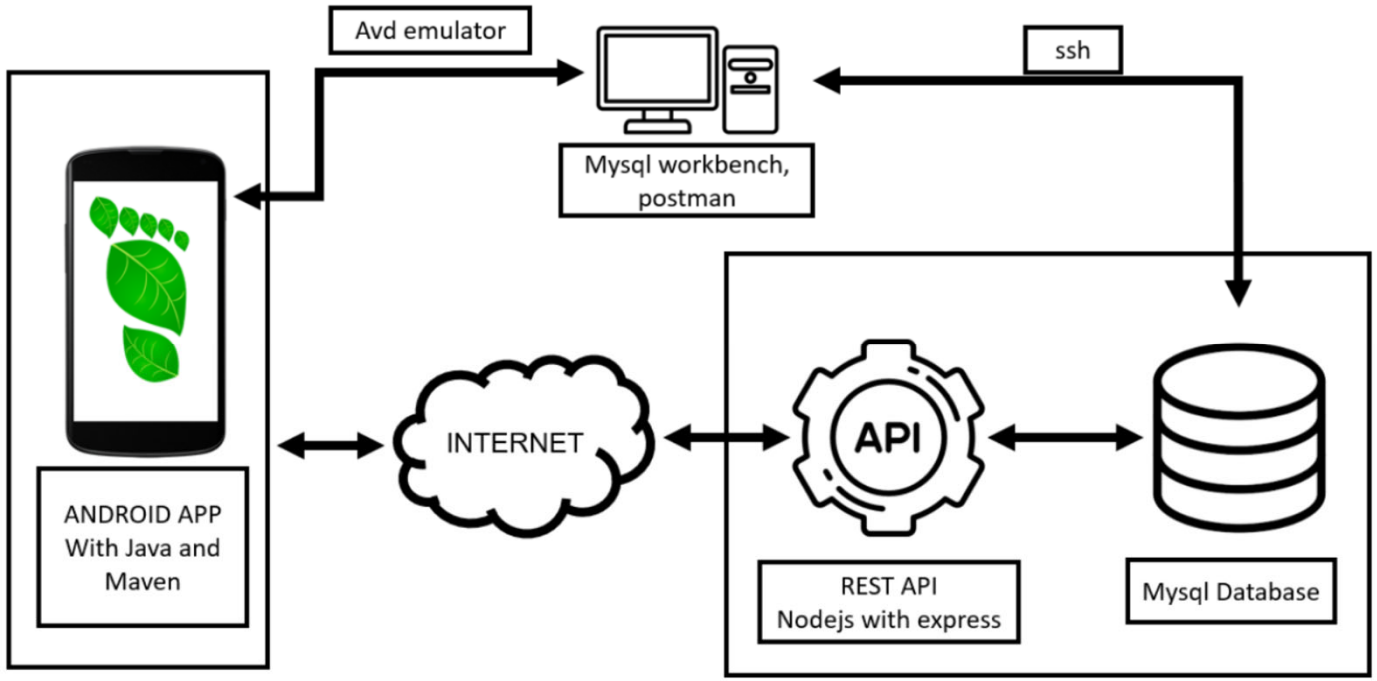
MODELS

Models

- Agile is not much model-oriented
  - But still we think that some models provide value
  - We request them as part of the documentation
- Types of models
  - Conceptual class diagram (UML) to describe the problem
  - Architectural models
    - Domain layer class diagram (UML) -- i.e., after pattern application
    - Component diagram (UML)
    - Physical diagram (informal)
    - (optional) Some sequence diagram when considered worthy
  - Data models:
    - Class diagram describing the data base schema
    - (optionally) the data base tables

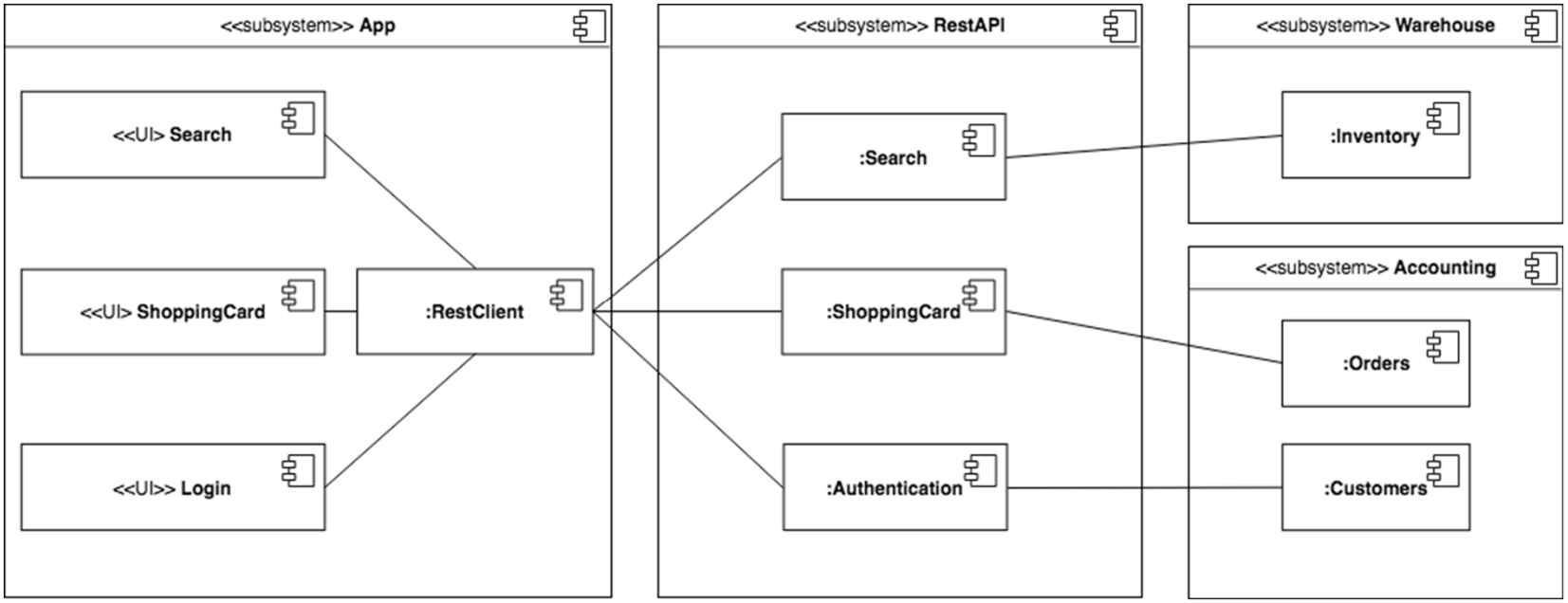
Physical diagram

- Describe the nodes and subsystems that participate in the system
- To be refined progressively



Component diagram

Simplified Component Diagram in UML to provide an overview the whole system and show relationships between different components.

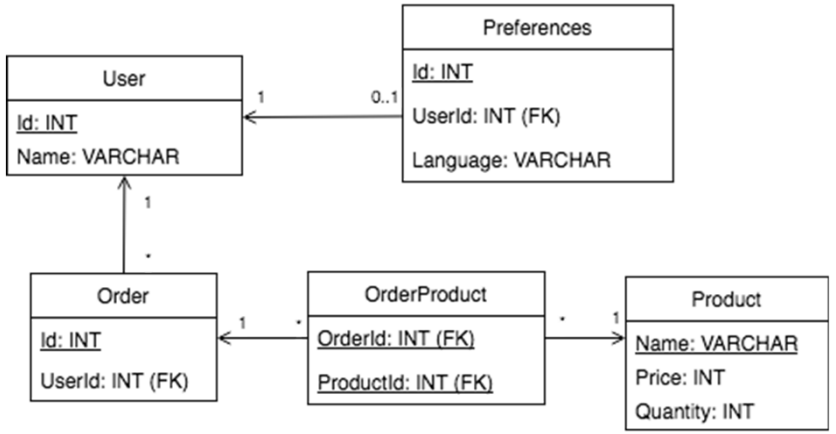


https://www.ibm.com/developerworks/rational/library/dec04/bell/index.html

Data base diagram

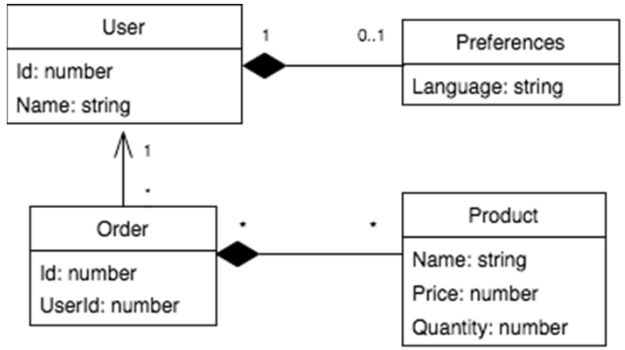
Relational Database

- PK underlined
- FK in parentheses



NoSQL Database

- Use of Composition
- Use of Association



THIRD-PARTY SERVICES

Third-party services

- Objective: to face the challenge of working in a service-oriented world
  - as a producer of a service to be integrated by others
  - as a consumer of a service delivered by others
- In the course: given the three teams A, B, C that usually compose a group in PES:
  - A produces a service for B and consumes a service from C
  - B produces a service for C and consumes a service from A
  - C produces a service for A and consumes a service from B
- The services will be produced along the three development sprints
  - Who is A, B and C? Determined in the inception

First iteration: define the service

- Goal: to agree on the services that each team provides to each other
  - it must be a negotiation
- What can be the service about:
  - A well-defined, independent functionality
  - General-enough as to be of interest for both involved teams
  - E.g., related to transversal functionalities
- Other aspects may need clarification:
  - E.g., where the service will be deployed

Second iteration: provide the service

- Goal: each team finishes the implementation of the service they provide
- This comprises:
  - Coding and testing of the service
  - Offering an endpoint for the service
  - Appropriate documentation for the consumer team

Third iteration: consume the service

- Goal: each team integrates the service provided by the other partnering team

TECHNICAL ASPECTS

Cloud Services

- Virtech
  - Unlimited access from campus
  - VPN access from home
- Heroku
  - Limited hours every month
  - Automatic sleeps
- AWS
  - Automatic shut downs
- Other services

Servers configuration

- Configure servers to fulfill your requirements
- Document the entire procedure
  - Creating machines and characteristics
  - Installing and configuring services and tools
    - User and passwords
    - Open ports
    - Certificates
    - Versions
    - And any other relevant parameter
  - How to start and stop the service

Database schema changes

- Incremental development implies lots of refactoring and changes. Database schema included.
- You need to track and manage all changes and evolution
  - Just SQL files containing create and alter statements
  - Using a specialized framework
    - Liquibase for Java
    - Django migrations
    - Laravel migrations
    - Mysql-migrations for nodejs
    - … explore and find which fits your needs

DOCUMENTATION AND EVALUATION – GENERAL

General rules

- Main report: a single pdf file with all the information
  - to be uploaded in the Racó
  - cover page with team information and links (see next slide)
  - check good quality of images!
- Artifacts need to be in repositories
  - always updated
    - shortly report artifacts' updates in every deliverable
- Be polished
  - formatting, typos, style, …
- Deadlines are hard
  - deliverables arriving late will be at least penalised, in the worst case just discarded

Cover page

In all deliverables, a cover page clearly stating:

- name of the team/project
- which deliverable (inception or sprint, and which one)
- links to resources:
  - project record track (Gdrive)
  - Taiga space
  - GitHub repository
  - All other tools your beloved teachers need to access (SonarCloud, …)
- team members' description:

Cognom	Nom	Responsible	UPC e-mail	Gdrive email	Taiga account	GitHub account
Fuertes	Dolores	Dev-1	...	...	...	...
Lotas	Felipe	Service	...	...	...	...
...						

Documentation

- Updates from previous deliverables must be self-contained.
  - Each document needs to be readable without looking at the previous ones
  - It should be clear what has changed in every sprint
- When a document section is updated you must include:
  - The original text/images from previous sprint.
  - The new updated text/images clearly distinguished, e.g. in a different color, or with an explanation of what is new.

General evaluation criteria

- **Quality of documentation**
  - *Text*: writing, orthography, figures, tables, …
  - *Structure*: front page, numbering, …
  - *Layout*: page breaks, margins, …

- **Project record track**
  - *Completeness*: does it appear to be complete?
  - *Correctness*: the information included follows the given guidelines?
  - *Legibility*: is it easy to read? E.g., the activities are clear, the style is uniform, …

DOCUMENTATION AND EVALUATION – INCEPTION

Inception: documentation, 1st phase

1. sprint master report
2. general conception of the project (a couple of paragraphs)
3. product box (can be link to external resource)
4. market analysis (what similar apps are available)
   - you need to include a short individual description AND a summary (e.g. a table classifying the different apps with respect to some criteria)
   - this analysis needs to demonstrate a business opportunity for your app
   - provide evidence of search method
5. "NOT list" – always keep it updated

Vision to be summarised in an elevator pitch

https://agilewarrior.wordpress.com/2010/11/06/the-agile-inception-deck

Inception: documentation, 2nd phase

1. sprint master report
2. updated "NOT" list -- clearly mark the changes (always)
3. list of stakeholders
   - only key stakeholders needed, fully-fledged descriptions not needed
4. table of epics and user stories (only names)
   - for each epic, list of its user stories (e.g., structure as a table)
5. initial product backlog (including story points) → at Taiga
6. critical mock-ups
7. architectural physical diagram (first overall sketch)
8. UML class diagram (of the envisaged final solution)
9. description of working methodology and technology
   - to be updated in every upcoming iteration
10. consumer and producer of services: identify the teams

https://agilewarrior.wordpress.com/2010/11/06/the-agile-inception-deck

Inception: evaluation criteria (I)

- **Product box**
  - *Vision*: does the box communicates clearly what the product does?
  - *Features*: are the 3/4 selected feature the most representative ones?
  - *Design*: is the box aesthetically appealing?
- **Not-List**
  - *Cohesion*: gives the idea of a cohesive product
  - *Coherence*: are there contradictions among the 3 parts of the list?
- **Market analysis**
  - *Scope*: to what extent the market search appears to be systematic and exhaustive?
  - *Quality*: is the analysis of the alternatives found useful and makes clear the contributions of the envisaged product?
- **Elevator pitch**
  - *Message*: has it clearly communicated the vision of the product?
  - *Attraction*: has it been an attractive talk?

Inception: evaluation criteria (II)

- **Presentation to customer**
  - *Message*: has the presentation given a clear message?
  - *Convincing*: has the talk demonstrated that the product has a business value?
  - *Attraction*: has it been an attractive talk?
  - *Coherence*: has the presentation been aligned with the documentation?
- **Product backlog**
  - *Contents*: is the set of user stories reasonable as to implement the envisaged product? Are well ordered in a coherent manner (from a priority point of view)?
  - *Quality*: are the user stories well described? Do they include story points? Are there the most important acceptance criteria (even if not at a great level of detail)?
  - *Coherence*: is the backlog coherent with other artifacts (e.g., NOT-list)?
- **Miscellaneous**
  - *Stakeholders*: the main ones are included and described well enough?
  - *Epics*: is there a complete and well-explained list?
  - *Architecture*: is there a clear, high-level physical architecture of the solution?
  - *Mock-ups*: have the most critical mock-ups been selected and shown?

DOCUMENTATION AND EVALUATION – DEVELOPMENT SPRINTS

Sprint: planning

The same day of the planning meeting, send by email to your teacher a pdf containing (in addition to the sprint number and team name):

- The overall goal of the sprint
- The list of user stories planned for the sprint:
  - Name
  - Epic which it belongs
  - List of tasks identified for the user story
    - Each task with its estimation of hours

The sprint backlog should be created according to this information

Sprint: documentation (I)

1. Introduction
   - **1.1.** a kind of executive summary of the sprint (description of what has been done)
   - **1.2.** sprint master report (summary of the vision of the sprint master)
   - **1.3.** individual statements of work (each member briefly describes her work)
   - **1.4.** team mates evaluation (by default, all happy)
2. Requirements
   - **2.1.** "NOT" list – updated
   - **2.2.** summary of product backlog in the document (user stories' names, organized by epic)
     - → clearly indicate new/deleted/changed items in the product backlog from last document
     - → clearly indicate the status of the US (in which sprint has been/is being implemented)
   - **2.3.** non-functional requirements using Volere template (ref. ER course)
   - **2.4.** complete contents in Taiga
     - → product and sprint backlog
     - → non-functional requirements should be somehow represented
   - **2.5.** treatment of transversal aspects
     - → clearly indicate final target and current state
   - **2.6.** third-party services
     - → describe both the service offered, and the service to be integrated, as mentioned in previous slides
     - → describe how are you communicating with the other teams

Sprint: documentation (II)

3. Agile ceremony
   - **3.1.** report on the sprint planning, review & retrospective meetings
     - → E.g.: what to keep, what to improve, what to avoid
     - → teacher may be present in review (and planning) meetings, not in retrospective
     - → in sprint *k*, report refers to review of sprint *k*-1 and planning&retrospective of sprint *k*
   - **3.2.** updated release&iteration burndown charts and velocity chart
     - → all three from the first sprint (even velocity!)
     - → not just the diagrams, but some analysis of what the diagrams show
4. Methodology
   - **4.1.** general explanation
   - **4.2.** individual description of all methodological dimensions
     - → at an adequate level of detail
     - → use significant events (new sprint, close tasks, …) to articulate the description

Sprint: documentation (III)

5. Technical description
   - **5.1.** overall conception of the architecture
     - → clearly explain the main architectural pattern(s) applied (MVC, …)
     - → include the domain class diagram (UML)
   - **5.2.** other architectural diagrams
     - → refined physical layer (informal); component diagram (UML)
   - **5.3.** design patterns applied
     - → justification of need
     - → explanation of application (eventually including some code for illustration purposes)
   - **5.4.** data models (UML)
     - → UML class diagram describing the problem (keep it always updated)
     - → UML class diagram describing the data base, only of the part implemented so far
   - **5.5.** other technological aspects
     - → check slides Chapter 1, and feel free to add more information
     - → include technical descriptions of third-party services (both offered and consumed)

Very important: 1) highlight all changes from the previous sprint

2) check always the slides of Chapter 1: transversal criteria Section 2.5, methodological aspects for Section 4, and technological aspects for Section 5.5

Sprints: evaluation criteria (I)

- **Architecture and data models**
  - General: architecture as a whole (architectural styles and patterns)
  - Design patterns applied (which ones and how)
  - Diagrams: architecture diagrams (component diagram in UML, physical diagram updated)
  - Conceptual model as class diagram, DB schema
- **Product and iteration backlogs**
  - Contents: do the set of user stories describe the functionalities and qualities of the app (product backlog) and the iteration (iteration backlog)?
  - Quality: are the user stories well described? How about acceptance criteria? And NFRs?
  - Tasks: have the tasks being well chosen (balanced, meaningful, …)?
  - Metadata: story points; in the case of tasks, hours (estimated and actual). More info is welcome if it is used afterwards

Sprints: evaluation criteria (II)

- **Sprint meetings**:
  - *Planning*: how have been decided the user stories for the sprint? How has the work been planned? How well described and complete was the initial planning sent to the teacher right after the planning meeting?
  - *Review and retrospective meetings*: how have been analysed the outcomes of the sprint and what conclusions are drawn?
- **Charts**
  - Iteration burndown, velocity and product burndown as explained in the slides

DOCUMENTATION AND EVALUATION – FINAL DOCUMENTATION

Project closing documentation

Besides the documentation required by iterations, you need to deliver at the end of the project:

- **managerial report** (for evaluation or course improvement)
  - an executive summary of the work done
  - report on most significant deviations found
  - self-evaluation of the project
    - main challenges, barriers, opportunities, …
    - what have your learned, what would you do differently, …
    - what concepts have you used from which courses
  - final statement on involvement/work of all team members
- **final releasing information**
  - detailed information on how to execute the app
  - any convenient material (multi-modal, documentation, …)
- **final presentation slides**

This documentation is also evaluated!

FINAL DEMO GUIDELINES

Final demo guidelines

- It is advisable to do a live demo. If possible try to show the demo on the classroom projector.
- It is advisable that one or two people present the demo (for example if a there are functionalities that require the presence of 2 people, for example, a chat).
- You must rehearse the presentation and avoid improvisation. It is important to decide what to show and what not.

Final demo guidelines

- It is not necessary to show all the App functionalities. You should avoid showing incomplete or untested functionalities. Instead of showing uncompleted or untested functionalities you can explain them.
- If you do not have enough time to show all the functionalities you want, it is better to concentrate in the core functionalities and the ones that add value to your App. You can omit showing very basic or simple functionalities in this case.

Final demo guidelines

- It is advisable that you have a data set to do the demo instead of presenting the App unpopulated or empty. Obviously, if you decide so, during the demo you can add, update or delete data.

Projecte d'Enginyeria del Software: Documentació i Comunicació

Xavier Franch

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona