SOFTWARE PROJECT

LECTURE 1: INTRODUCTION

Coordinator: Dr Kuldar Taveter
Lecturer: Dr Marinos Georgiadis
Mentors: Dr Marinos Georgiadis, Rafał Włodarski
LECTURE OUTLINE

1. Course main points
2. Guidance and expected effort
3. Deadlines
4. Agile Development
5. Infrastructure
6. Iterations - Grading
7. What to do now/next?
Goal
- To develop a real-life software project. You will deliver software to a real customer with real requirements.

Objectives
- Understand and apply software engineering methods to agile development for a real project.
  - E.g.: User stories, UML modeling
- Use agile software development practices, including rapid iterative development, version control, collaborative planning, issue tracking, test automation and continuous integration.
- Teamwork
- Communication
- Your task is to detail the requirements, create a plan and deliver accordingly.

You will work in teams of four students.

Evaluation
- Your deliverables will be verified and validated by both the course organisers and the customer who proposed the project.

https://courses.cs.ut.ee/2021/tvp/
GUIDANCE & EFFORT

- Lectures
  - Introduction
  - Requirements Engineering – Methods of elicitation, User stories, Use case modeling, etc.
  - UML – Use cases, sequence diagrams, activity diagrams, class diagrams, etc.
  - UX Design

- Lots of mentoring – weekly meetings with mentors starting on the 8th of September
  - Mentors will clarify what you need to achieve and by when, they will help you resolve conflicts and mediate problematic situations, and will assess your deliverables

- Lots of teamwork (about 150 hours per member, 600 per team)

- Lots of online communication
  - Please make sure you are registered in the Slack workspace https://tvp2021.slack.com
  - Preferably, use Slack channels to communicate within your team

- The schedule of the mentoring meetings will be posted in the Teams section of the course Web site
DEADLINES

- **Project deadlines**
  - 3 September: teams formed
  - 20 September: 1st iteration (two weeks)
  - 11 October: 2nd iteration (three weeks)
  - 8 November: 3rd iteration (four weeks)
  - 6 December: 4th iteration (four weeks)

- **Other deadlines**
  - 20 October: Intermediate demo day
  - 5 November: You will receive a message with information about peer-review
  - 12 November: Select a project to peer-review
  - 22 November: Peer reviews due
  - 6 December: Peer-review responses due
  - 15 December: Final Demo Day

- If you want a deadline extension, you need to explicitly ask it from your mentor. The mentor will apply relevant penalties.
1. **Plan** and prioritize new features/requirements by creating **milestones** and **issues**. Use **Wiki** to write down requirements.

2. Create the features by committing **code** and pushing it to GitLab

3. Create a build and run automated **tests** to verify the coded features

4. Continuous Delivery

5. Transition to the Planning stage by improving existing features or creating new ones
Every team must set up a development/collaboration infrastructure

- Min requirements:
  - Version Control System (VCS), e.g., Git
  - Issue tracker
  - Wiki
  - Continuous Integration

- GitLab is recommended, Github or BitBucket are alternatives, but they need to be combined with a CI platform (e.g., Github Actions, Travis, Shippable)
  - It’s your choice, you set it up and then grant access to your mentor

- For CI of mobile apps, we recommend using a free NeverCode.io account
We can provide:
- A Linux Ubuntu or CentOS bare-bones virtual server with 2 cores, 2 GB RAM, 20GB Disk in University of Tartu’s cloud (available with VPN access only)
- If you need more capacity (CPUs, RAM), ask, but not sure we can provide it
- If you need an external IP address (to show to your customer), please ask upfront
- Send us an email to make your virtual server request

But you can also use another deployment infrastructure, e.g.
- Engine Yard for PHP/Ruby/Node.js
- Heroku for Java/Ruby
- DigitalOcean – you can get $100 credits by applying for a GitHub education pack
- Docker for CI is allowed
  - Normally, your CI process should end in a deployed/executable application
  - However, producing a docker image in Docker Hub and “manually” pulling it into your deployment server on UT's cloud is OK
The project consists of 4 iterations

Deliverables, by the end of each iteration

Follow the deadlines

Grading* and feedback by the end of each iteration

- 1st iteration - 10pt
- 2nd iteration - 15pt
- Intermediate demo - 5pt
- 3rd iteration - 25pt
- Peer review - 5pt
- 4th iteration - 35pt
- Final demo - 5pt

*Bonus and differentiated grading may apply.
• Project vision
• Definition of Functional and Non-Functional Requirements at high level – User Stories
• Project plan
  o List of all project tasks (backlog)
  o Define in detail the tasks of the second iteration and their deliverables
• At least two of the core functional requirements are detailed in the selected format (use case specification, detailed user stories).
• Links between detailed requirements (e.g., use case descriptions)/functional requirements/non-functional requirements and tasks in issue tracker
<table>
<thead>
<tr>
<th>#</th>
<th>Title</th>
<th>Created</th>
<th>Updated</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Send first email to client</td>
<td>11 months ago</td>
<td>11 months ago</td>
<td>Initial Meeting with Client (Sep 3, 2020)</td>
</tr>
<tr>
<td>#2</td>
<td>Inform Yar and Alexandru of our preference for meeting between 12:00</td>
<td>11 months ago</td>
<td>11 months ago</td>
<td>Initial Meeting with Team Mentor (Sep 3, 2020)</td>
</tr>
<tr>
<td>#3</td>
<td>Research the integration of WordPress and GitLab</td>
<td>11 months ago</td>
<td>11 months ago</td>
<td>First Iteration (Sep 8, 2020)</td>
</tr>
<tr>
<td>#4</td>
<td>Analyse project PDF, write down questions and potential users &amp;</td>
<td>11 months ago</td>
<td>11 months ago</td>
<td>Initial Meeting with Client (Sep 8, 2020)</td>
</tr>
<tr>
<td>#5</td>
<td>Research WordPress and understand how it works</td>
<td>11 months ago</td>
<td>11 months ago</td>
<td>Initial Meeting with Client (Sep 8, 2020)</td>
</tr>
<tr>
<td>#6</td>
<td>Create (empty) Wiki entries for each of the deliverables for the</td>
<td>11 months ago</td>
<td>11 months ago</td>
<td>First Iteration</td>
</tr>
<tr>
<td>#7</td>
<td>Set up a central database for storing events, registrations, and</td>
<td>10 months ago</td>
<td>10 months ago</td>
<td>Second Iteration (Oct 2, 2020)</td>
</tr>
<tr>
<td>#8</td>
<td>Create a reusable subform for selecting events which have</td>
<td>10 months ago</td>
<td>10 months ago</td>
<td>Second Iteration (Oct 9, 2020)</td>
</tr>
<tr>
<td>#9</td>
<td>Create a special form for creating and modifying workshops for</td>
<td>10 months ago</td>
<td>10 months ago</td>
<td>Second Iteration (Oct 9, 2020)</td>
</tr>
</tbody>
</table>

**Categories:**
- dev-category: general
- dev-category: general-build
- dev-category: general-event-management
- requirement: FR 1.2
As an event manager I want to...

- create events
  - FR-E-1: be able to create an event, so that people could find and register to my event. UC-2
  - FR-E-2: be able to insert the date, time and place of my event, so that people would know when and where the event is going to take place. UC-2
  - FR-E-3: be able to choose the category (vahetund, ainutund, vörgustik, kohalik omavalitsus) of my event, so that people could filter events by categories relevant for them. UC-2
  - FR-E-4: be able to add a description of my event, so that people would know what my event is about. UC-2
  - FR-E-5: be able to set the start and end time of registration, so that people would know when they can register to an event. UC-2
  - FR-E-6: be able to limit the number of available places at an event, so that the number of participants would be in accordance with the limitations to space, nature of the event, etc. UC-2
  - FR-E-7: have the participants be able to register to only one event with the same content in different locations, so that participants would not keep unnecessary places vacant. UC-2
  - FR-E-8: be able to add separate workshops to events, so that the participants know about and can register to workshops for an event. UC-16

- modify existing events
  - FR-E-9: edit information about my events, so that I could update the information when there have been significant changes. UC-3
Use cases

Use Case UC-1: Adding a new category of events

(CORE) Use Case UC-2: Creating events

(CORE) Use Case UC-3: Modifying existing events

(CORE) Use Case UC-4: Deleting events

(CORE) Use Case UC-5: Sending automatic registration confirmation email

Use Case UC-7: Viewing the information about participants on all events

Use Case UC-9: Canceling the registration of a participant on any event

(CORE) Use Case UC-11: Exporting participant data about all events

Use Case UC-12: Searching for events

(CORE) Use Case UC-13: Registering for an event

(CORE) Use Case UC-14: Canceling registration for an event

Use Case UC-15: Registering to workshops

Use case UC-16: Creating workshops
Other possible deliverables

- a Use Case diagram, including at least the two use cases for which their detailed description will be specified,
- a sequence diagram and an activity diagram for each of the aforementioned use cases
- a class diagram and database design may be parts of this iteration.
- However, apart from the Use Case diagram, the rest of the diagrams and database design may be planned for the next iteration or may be refined during that if started in the first iteration.
- For creating UML diagrams
  - You may use an external UML tool and add links in your Wiki to these diagrams or just upload them to your Wiki.
  - You may use PlantUML or a similar tool
  - If possible, make the necessary links in the issue tracker

Collaboration infrastructure

- VCS, Wiki, Issue Tracker
• Why the system will be built. What problem it will solve or what opportunity is it meant to exploit?

• Who is going to use the system?

• What type of system it will be (e.g. a Web API, a Web frontend, a mobile app, a desktop app?) and what will the system allow its users to achieve?

• Note: The "project vision" can be written as a preamble to the list of functional requirements or as a separate "Project Vision" entry in the Wiki.
ITERATION 1 — PROJECT PLAN

First version of the project plan is present in the Wiki/Issue tracker. It contains at minimum:

- **Roles in project.** You have understood each team member roles and assigned responsibilities.
  - E.g.: Product owner, requirements engineer, programmer
First version of the project plan is present in the Wiki/Issue tracker. It contains at minimum:

- **Communication means.** You have agreed both among yourselves and with the customer how and when and via which channels you are going to communicate.
The nature of the information to be conveyed:

- What is the extent and complexity of the information to be conveyed?
  - A phone conversation if message is simple
  - Is it easy to understand? Is the context well known to both the sender and the recipient?
  - Two way communication
  - Where the communication is personally sensitive
    - Face-to-face contacts

- At different stages of a project – different communication categories will be preferred

<table>
<thead>
<tr>
<th>Same place</th>
<th>Different places</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same time</td>
<td>Meetings, Interviews</td>
</tr>
<tr>
<td>Different times</td>
<td>Noticeboards, pigeon-boards</td>
</tr>
</tbody>
</table>
COMMUNICATION CATEGORIES

- **Early stages – meeting(s)**
  - Team members need to build up their trust and confidence in their co-workers
  - Decision making

- **Intermediate stages (design) – teleconferencing**
  - Activities executed in parallel
  - Some points needs to be clarified

- **Implementation stages - emails**
  - Everyone knows their role, work can progress
  - Face to face meetings – helps coordination and maintain motivation
You have understood how you are going to work together and have explained this to your mentor. At least the following has to be clear:

- How and using what materials the customer is going to understand what you are going to build (e.g. with wireframes)
- How do you determine that the customer is accepting your solution proposal
- How you are internally going to build the accepted solution (who assigns the tasks, who is going to implement it, will the tests be written, will code be reviewed, who is going to verify, who is doing the validation, etc.)
- When do you consider something ready to be delivered to the customer for review
- How do you gather feedback from the customer and/or end users.
- What is the definition of DONE on a task
List of all tasks you currently foresee need to be delivered in order to complete the project.
- Tasks are given priorities, have assignees, and they contain initial estimation about the amount of work involved. The descriptions of the tasks in the project plan is understandable.

- The tasks/tasks deliverables of the second iteration (at least) must be detailed in the project plan. Later iterations might not be clear enough to have their tasks broken down according to your current knowledge about requirements.
Application functionality

- The functionality in the application covers at least one of the core use cases.

Continuous Integration (CI)

- Automated process in place from source code to a running application.
- A trace that the CI process has been running for at least a week.
- When a new version of the code appears in the VCS, the new code is checked out, the application is built, and it is deployed on a server or it is made available as an executable file.
ITERATION 2 (CONT.)

- Detailed requirements.
  - At least 75% of the requirements are detailed in selected form (for example, in use cases).
  - Detailed requirements are in individual wiki pages/documents. Detailed requirements are accessible from the overall requirements list.
  - For creating UML diagrams
    - You may use an external UML tool and add links in your Wiki to these diagrams or just upload them to your Wiki.
    - You may use PlantUML or a similar tool
    - If possible, make the necessary links in the issue tracker

- UI wireframes/mockups
  - If applicable, the application UI has been prototyped using wireframes or mockups and there are links from the detailed requirements to the corresponding UI screens.
ITERATION 2 (CONT.)

- **Scope.**
  - The remaining tasks are estimated, and the estimates are discussed with the customer.
  - The project plan indicates what tasks are delivered in the subsequent iterations.
  - There is a clear picture of what functionality will be ready after iteration 3 and after iteration 4.

- **Release notes.**
  - Release notes for this iteration are present.
  - Overview of what features of the system are supported by the released version of the system
  - Input for the acceptance testing by the customer and the course coordinator.
  - If the delivered functionality contains known bugs, those are highlighted in release notes.
Iteration 3

- Application.
  - Most important deliverable, awarded with the most points.
  - The functionality in the application covers all core use cases.
  - The provided functionality is tested and works.
  - User can access the delivered functionality and test its behaviour with minimal additional assistance.
  - The provided functionality is either bug-free or known bugs (which must not block the testing of the functionality) are registered in the issue tracker.
  - The functionality provided by the system reflects what is indicated in the release notes.

- Automated tests
  - There are automated tests present.
  - The tests verify the application's core functionality (system testing, not only unit testing).
  - The tests must cover at least two core use cases.
  - The tests are created most likely in a form of a script.
  - The tests can run without human intervention.

Feedback from the customer will count to approximately 30% of the grade in this iteration
Continuous Integration (CI).

- A trace that the CI processes have been running throughout the iteration.
- The environment monitors the VCS in use, checks out the new code, builds the application, runs the automated tests and deploys the application on a server (if applicable) or makes the executable file available somewhere where customer and course coordinator can access it.
- In case of the build errors, or in case the app is not passing some of the automated tests, these errors are reported to the team.
- As part of the CI process, there should be an automated build process in place, which the mentors can use to build your application from the source code with minimal effort (even outside your CI environment).

Requirements.

- All requirements are identified.
- Detailed requirements are in individual wiki pages/documents. They are accessible from the overall requirements list.
- Requirements dealing with UI interactions are complemented with UI prototype.
ITERATION 3 (CONT.)

- **Scope.**
  - There is a clear picture of what will be ready after the course finishes.
  - The tasks for the last iteration are planned and assigned in the issue tracker.

- **Release notes.**
  - Release notes for the iteration are present.
  - It is clear what has been added or modified to the application since the last release and what bugs are fixed.
  - If delivered functionality contains known bugs, those issues are highlighted in release notes.
ITERATION 4

- Everything must be tested and run
- Additional testing including internal acceptance testing
- Non-functional requirements verification
- Response to peer review

Feedback from the customer will count to approximately 40% of the grade in this iteration
1. **Transparency**
   - Anything that is not in your dev and collaboration infrastructure does not exist
   - Any server-side application that is not running on a server does not exist.

2. **Traceability**
   - Every commit must be linked to an issue
   - Where applicable, every issue must be linked to a requirement documented in Wiki

3. **Reproducibility**
   - A person having ordinary skill in the art (PHOSITA) should be able to build and test your app from the source code with reasonable effort. **Your mentor is a PHOSITA.**

4. **Continuous Integration (CI)**
   - In the end, every commit/push should lead to a new tested & deployed version of the system
The GitLab Issue Board is used to plan, organize, and visualize a workflow for a feature or product release.

Visualize all the work in a given iteration.
- To Do, In Progress, and Done

During the iteration planning meeting, the team moves items from the product backlog into the iteration/sprint backlog.

Issue tracking functionality and labels.
- Issues appear as cards in vertical lists, organized by their assigned labels, milestones, or assignees.

Examples of Labels: To-Do, In-Progress, Done, Feature <Name>, Requirement::ID (scoped label)

Milestones → Iterations
WHAT TO DO NOW/NEXT?

- Choose your project & contact your customer
  - Choose your project and form the team by Friday 3rd September, 8 AM
  - We will send you the contact details by Friday at 12:00 (should be available, as a rule)

- Agree on a first meeting with customer

- Prepare for your meeting – read the project description, prepare your meeting, do some searches online to get info, check for possible approaches, think about possible requirements, prepare questions

- Read the grading criteria (at least for iteration 1)

- Set up your dev and collab infrastructure

- If possible: discuss how to use the collab infrastructure – who monitors/oversees issue tracker, who writes to Wiki what and when, what is your Git workflow (e.g., branching and merging policy)

- Reminder: Iteration 1 deadline is the 20th of September