Pelaksanaan Proyek

IF3250 Proyek Perangkat Lunak

Tim Dosen IF3250

Overview

Kelompok @ 5 orang.

- Setiap anggota mengambil peran yang sama: semua melakukan analisis, desain, coding, testing, termasuk saling mereview pekerjaan rekan satu tim.

Topik: sesuai usulan dan alokasi topik

Project berlangsung selama 10 minggu

Metodologi: Scrum

- 10 minggu dibagi ke dalam 5 sprint @2 minggu.
  - jika diperlukan, bisa menambah sprint 6 (kasus khusus)
- 1 sprint = 16 jam kerja per orang (2 mgg x 8 jam/mgg)
  - 16x5 = 80 jam kerja per kelompok
- Anggota kelompok bergiliran mengambil peran sebagai Scrum Master.
- **Ingat**: setiap sprint menghasilkan *working product* yang bisa didemokan

Tooling

- MS Teams Channel untuk komunikasi, diskusi, dan pengumpulan berbagai artifak dan link artifak
- GitLab (Version Control, Code Review, Kanban Board, Issue Tracking, CI/CD)
- Google Docs for Technical & Project Documentation
  - Seluruh diagram dibuat dalam UML dengan menggunakan label Bahasa Inggris
- UML Tool: StarUML

Minggu ke-3

Kickoff dengan *project owner* untuk mendapatkan gambaran sistem yang akan dibuat.

Tentukan *product backlog*

Buat release planning

Jadwal Sprint

Sprint 1: minggu ke-4 dan ke-5

Sprint 2: minggu ke-6 dan ke-7

Sprint 3: minggu ke-9 dan ke-10

Sprint 4: minggu ke-11 dan ke-12

Sprint 5: minggu ke-13 dan ke-14

Ceremonies @ each sprint…

Sprint planning:

- pilih sebagian *product backlog* sebagai *sprint backlog*
- pecahkan *sprint backlog* menjadi *tasks* yang **cukup kecil** dan memiliki estimasi waktu penyelesaian (jam)
- Setiap anggota mengambil *task* utk dikerjakan

Setiap "hari" (2 hari in real life atau 3× seminggu) lakukan ***stand up meeting***

Di akhir sprint, lakukan ***sprint review*** (melibatkan asisten dan project owner) dan **sprint retrospective**

→ Untuk setiap ceremony harus ada notulen tertulis

During last sprint...

- Testing aplikasi (*system testing & acceptance testing*)
- Preparasi lingkungan operasional
- *Deployment* dari aplikasi
- Penyusunan manual pengoperasian (*user manual*)
- Melengkapi dokumentasi (e.g., Scrum, Technical documents)
- *Training* pengoperasian aplikasi

After last sprint...

- Aplikasi *ter-deploy* di lingkungan pengguna (atau di *testbed* yang disediakan oleh asisten) dan siap dioperasikan.
- Telah menerapkan *automated testing and continuous integration* pada sedikitnya 1 *product backlog/user story* yang paling utama.
- Menuliskan semua proses pengerjaan proyek perangkat lunak dalam sejumlah dokumen (*Scrum Document* dan *Technical Document*)
  - Dibuat bertahap di setiap *sprint*

Deliverables @ each sprint

- *Working product as a Git tag*
- *Scrum document (evolving!)*
  - *Minutes of sprint planning, review, retrospective, and stand-up meeting*
  - *Burndown chart*
  - *Screenshot of Kanban board at each stand up meeting*
- *Software technical document (evolving*!)
  - *Diagrams must be in standard UML format*!

Working Product

Filetype:url, 1 URL untuk setiap sprint

URL ke Git tag pada GitLab

Scrum documents

(each sprint)

1. Burndown chart
2. Kanban Board
3. Scrum Document

Burndown Chart

Filetype:xlsx, 1 file untuk sepanjang proyek

Satu file untuk sepanjang proyek

Setiap sprint = satu sheet dalam file

Template ada di **Class Materials/Template – Burndown Chart.xlsx**

Upload ke Files di private channel kelompok, pin ke Tab

Using data from "estimate" vs "spent" in Gitlab's issues

Kanban Board

- Kanban Board is organized in the group of Gitlab (not each repository)
- Put the URL of the Kanban Board in the MS Team's Tab as "Website" --> Label "Board"
- Lane

  Open, To Do, Ready, WIP, Review, Testing, Closed

  Label (At least)

  - Product Backlog - label for Gitlab's issues that are considered as product backlog
  - User Story - label for Gitlab's issues that are considered as user story
  - Task - label for Gitlab's issues that are considered as sprint tasks (user story details, overhead, etc)

Kanban Board

Lane

- Open - any backlogs/user stories that you have identified
- To Do - product backlog, sprint backlog/task, user stories that need to be done on a sprint
- Ready - sprint backlogs/tasks that are ready/can be put in WIP
- Work-In-Progress - sprint backlogs/tasks that are been doing by team
- Review - sprint backlogs/tasks that are done and being review
- Testing - sprint backlogs/tasks that are been testing
- Closed - done

Scrum Document

Filetype:gdoc

Isi:

- Notulen sprint planning
- Notulen setiap stand-up meeting (6 × dalam 1 sprint)
  - Screenshot kanban board pada setiap meeting
- Notulen sprint review
- Notulen sprint retrospective

→ Lihat template

Scrum Document

Practices using gitlab.informatika.org

Sprint Goal --> milestone's description

Sprint backlog --> issues

Weight of a sprint backlog --> quick action "estimate" of a Gitlab's issue

Sprint #? --> "milestone" of a Gitlab's issue

Progress of doing a sprint backlog --> a quick action "spent" of a Gitlab's issue

Burndown chart using XLS's template using data from "estimate" vs "spent" in Gitlab's issues

Technical Document

Filetype:gdoc, 1 set of 5 files untuk sepanjang proyek (evolving!)

- Software Requirement Specification (SRS)
- Software Design Description (SDD)
  - Diagram: use case, class, sequence, deployment
- Implementation
- Testing
- Deployment Document (di sprint terakhir)
- User Manual (di sprint terakhir)

Technical Document – UML diagrams

Filetype:xmi, 1 file untuk setiap sprint

4 diagram UML dari StarUML, export jadi 1 XMI

Beri nama file: <nomorkelompok>uml.xmi, e.g. 03uml.xmi

Put working xxxx.mdj in MS Team

Softeware Requirement Specification

1. Product Backlog
2. Sprint Backlog
3. Use case diagram

Create a use case diagram for this system.

The use case diagram should present the main interactions between the system and the users and external systems.

The use case diagram should explicitly mention any external systems that are involved in this system.

Give a brief explanation in text of each use case.

Software Design Description

Stakeholders & Drivers

Describe the stakeholders of this system (describe at least 3, and at most 5 stakeholders)

For each stakeholder, describe one or more of his interest (‘stake’) in the system.

Architectural Drivers: describe 3 architectural drivers and motivate why they are drivers.

Software Design Description (2)

Identify a functional decomposition for this system

Each subsystem should represent some clearly separate, but coherent group of functionalities.

One way to check and recognize this is to describe the responsibility of each subsystem in at most one sentence. Typically, this responsibility is described in terms from the application domain.

For each of these subsystems/components, decide what type of role-stereotype this subsystem/ component has.

Hand in the list of subsystems with their role-stereotypes via the online form.

Software Design Description (3)

Structural View

- *Design a model that represents the structural view for this system.*
  - *This model should present the main components/subsystems of the system and the relations between them.*
- *In your diagram, represent subsystems as classes.*
- *For each class, show the most important functionality that they realize by listing appropriately named methods.*
- ***In your document, give an explanation of your design and explain the most important design decisions and any assumptions that you make.***

Software Design Description *(4)*

Dynamic View: Sequence Diagrams

The dynamic view consists of a collection of sequence diagrams.

The sequence diagrams describe key scenarios of the system.

- *These scenarios describe orders in which the components (as identified in the task ③) interact with each other.*
- *Interaction comprises various forms, such as the sending of messages, call-return interaction, or other styles of interaction.*

Typically, each important use-case is described by a (separate) sequence diagram. ***For this assignment, you may limit your hand-in to at most 4 sequence diagrams.***

Software Design Description *(5)*

Deployment Diagram

Create one or more deployment diagram(s) that represent the distribution of the components of the system on different machines (servers/ CUPs) and indicate how these machines are connected to each other.

See examples in the guest lecture slides.

Teknis penulisan dokumen

- Semua dokumen ditulis dalam bentuk Google Doc.
- Diagram UML dibuat menggunakan StarUML, install ekstensi “XMI”
  - Versi gambar (File ▸ Export Diagram As ▸ PNG...) “ditempel” dalam Docs
  - Versi XMI (File ▸ Export ▸ XMI Export...) disubmit juga secara terpisah

→ Lihat template

Teknis pengumpulan deliverables

- URL ke Git tag
- URL ke Google Doc (Scrum Document)
- URL ke Google Doc (Technical Document)
- File XMI (UML Diagrams)
- Di-upload di channel MS Teams masing-masing kelompok.