Logbook - BDSA15

Dennis Thinh Tan Nguyen, William Diedrichsen Marstrand, Jacob Mullit Mniche, Thor Valentin Aakjr Olesen Nielsen

November 24, 2015

Contents

1 Log for date: 11/11-2015

Start: 14:00 — End: 16:00

Member attendence:

- Dennis Thinh Tan Nguyen attended.
- Jacob Mullit Mniche attended.
- Thor Valentin Aakjr Olesen Nielsen attended.
- William Diedrichsen Marstrand attended.

Meeting pins

- Set up Trello and Pomello
- Set up Log book
- Initialized C# program structure

Sprint Planning

- Sudy ConfigurationUI (Web Service Interface)
- Internal communication among subsystems
- Subsystems Classes
- Storage Interface
- Work Load
- Prviding Blue Team API
- Dependency Injections
- Using Adapter Pattern

2 Log for date: 11/12-2015

Start: 13:00 — End: 16:00

Member attendence:

- Dennis Thinh Tan Nguyen attended.
- Jacob Mullit Mniche attended.
- Thor Valentin Aakjr Olesen Nielsen attended.
- William Diedrichsen Marstrand attended.

Meeting pins:

- WebAPI (Fra Steven) - Adapter Pattern, Facade (Autosys Server)
- Initialized and refactored project structure - packages to projects
- UML Subsystem refactored
- UML Entity Object refactored
- Entities first

Branching Rules

- First create a local branch
- Then publish the branch so it resides on oring/master
- Required start name for naming convention for branches: implementation/, testing/, documentation/, bugfix/, refactor/

3 Log for date: 14/11-2015

Start: 12:30 — End: 15:10

Member attendence:

- Dennis Thinh Tan Nguyen attended.
- Jacob Mullit Mniche attended.
- Thor Valentin Aakjr Olesen Nielsen not attended.
- William Diedrichsen Marstrand attended.

Meeting pins:

- Creating overview of every subsystems
- Updated LucidChart of subsystems
- Defining classes and their responsibilites of every subsystems (Application logic and Storage)

Sprint Planning:

- Implement design decision for the new codeskeleton

4 Log for date: 11/16-2015

Start: 16 — End: 21

Member attendence:

- Dennis Thinh Tan Nguyen attended.
- Jacob Mullit Mniche not attended.
- Thor Valentin Aakjr Olesen Nielsen attended.
- William Diedrichsen Marstrand attended.

Meeting pins:

- Code Skeleton completed.
- Pull requests reviewed.
- Merged branches into master.
- Discussed about work processes and how we contact each other outside ITU. Overall conflict is that study related activities seem not to be separated from other activities thus causing a tense stressful environment for certain members. One proposal is to keep track of each members time board so that people know when they can contact each other or when they should not be disturbed. This will be done with a document showing everyones schedule and using status tags on Trello.

Sprint Planning:

- Need to plan future work and time for meetings to achieve a clear distinction between activities related to ITU and others.
-

5 Log for date: 11/17-2015

Start: 9:00 — End: 12:00

Member attendence:

- Dennis Thinh Tan Nguyen attended.
- Jacob Mullit Mniche not attended.
- Thor Valentin Aakjr Olesen Nielsen attended.
- William Diedrichsen Marstrand attended.

Meeting pins:

- Cleaned up code skeleton
- Added missing classes
- Implemented WebApi

Sprint Planning:

- Update SDD and RAD (UML subsystems, datafield definition, user definition, update entity object model)
- Create time schedule for all group members and dertermine meeting hours

Work planning:

- A Google Document has been made where the group members have written the time intervals where they are okay being contacted.
- In the document mentioned above working schemes have been made as well, indicating when group meetings are held.
- This is to separate study related activities from social activities thus avoid a stressful environment

6 Log for date: 11/24-2015

Start: 9:00 — End: 16:00

Member attendence:

- Dennis Thinh Tan Nguyen attended.
- Jacob Mullit Mniche attended.
- Thor Valentin Aakjr Olesen Nielsen attended.
- William Diedrichsen Marstrand attended.

Meeting pins:

- Implement code skeleton feedback
- Connect implementation tasks with functional requirements
- Fix UWP vs standard application issue

Sprint Planning:

-
-

Implementation priorities based on functional requirements subsystem dependencies:

Application logic

- UserManagement will be implemented first because WebApi and StudyManagement depend on it. We need to define how a User is to be represented so that SM can use WA.
- StudyManagement should be implemented secondly because its criteria and phase classes are used to define a study configuration. ProtocolManagement depends on a finalized study configuration.
- ExportManagement and PaperManagement can be implemented independent from rest of project.
- All subsystems depends on Repository with predefined CRUD operations.
- WebAPI depends on all subsystems based on their methods. We need to define what methods each package handler holds in order to develop our WebApi. The WebApi will be based on the deliveries from Steven.

Interface

- StudyConfigurationUI is independent from all subsystems in application logic. It is based on a web api that uses the server in the application logic. Should be able to retrieve teams stored in database and create a study define passed to the application logic.

User Validation The application logic will not handle user validation (e.g. manager or researcher), which is handled in the interface part of the system between yellow and blue part. The only thing application logic considers is whether a given user exists in the database. Roles are checked in Study Management based on an enum flagtype related to a user, e.g. a user can have a "Reviewer", "Manager" or "Validator" role determining which tasks are to be returned. A Reviewer will receive review tasks and a Validator will receive conflict tasks. Blue team will validate users based on whether they represent a Manager or Researcher.

Code Skeleton Design Changed to reduce coupling

If all packages have a facade interface used by other subsystems it allows single responsibility. Also allows team members to work on subsystems independently. We do not touch each other's code this way. A sub system should be able to be pulled out and replaced without causing trouble. By way of example, ExportManagement depends on the ProtocolManagement and its a predefined protocol. A using statement for Protocol is used in ExportManagement. Thus, we need to change the UML and use a dashed line between ExportManagement and ProtocolManagement. The dependency resides in where the object comes from and not how it is passed (e.g. from WebAPI).