Report for . . .

authors

date
version

1 Introduction

Background explaining why this application is needed (besides mandatory in course). What's the problem addressed (use imagination)? What will it do? Who will benefit/use from this application? In what situation will the application be used? Define the application. General characteristics of application.

1.1 Definitions, acronyms, and abbreviations

Create word list to avoid confusion.

2 Requirements

2.1 User Stories

Use the template from the course website and list all user stories here. It is fine to have them in an spreadsheet (or other application) at first, but they must end up here as well.

These user stories should describe what the user will be able to do. Write a the user stories in language of the customer, and give the a unique ID. List the user stories in priority order.

2.2 User interface

Sketches, drawings and explanations of the application user interface (possible navigation).

3 Domain model

Give a high level view overview of the application using a UML diagram.

3.1 Class responsibilities

Explanation of responsibilities of classes in diagram.

4 System architecture

The most overall, top level description of the system. Which (how many) machines are involved? What are the system components, and what are they responsible for? Show the dependency between the different system components. If there are more computing entities (machines) involved: show dow they communicate. Describe the high level overall flow of some use stories. Describe how to start and stop the system.

Any general principles in application? Flow, creations, ...

4.1 Subsystem decomposition

Describe in this section each identified system component (that you have implemented).

4.2 'first component'

What is this component responsible for and what does it do.

Divide the component into subsystems (packages) and describe their responsibilities. Draw an UML package diagram for the top level. Describe the interface and dependencies between the packages. Try to identify abstraction layers. Think about concurrency issues.

If your application is a standalone then:

- Describe how MVC is implemented
- Describe your design model (which should be in one package and build on the domain model)
- Give a class diagram for the design model.

otherwise:

- MVC and domain model described at System Architecture

Diagrams

- Dependencies (STAN or similar)

- UML sequence diagrams for flow.
- Describe which design patterns you have applied.

Quality

- List of tests (or description where to find the test)
- Quality tool reports, like PMD (known issues listed here)

NOTE: Each Java, XML, etc. file should have a header comment: Author, responsibility, used by ..., uses ..., etc.

4.3 'next component'

As above, and continue for all components.

5 Persistent data management

How does the application store data (handle resources, icons, images, audio, ...). When? How? URLs, pathes, ... data formats... naming..

6 Access control and security

Different roles using the application (admin, user, ...)? How is this handled?

7 Peer review

A part of the examination is the peer review (kollegial granskning) of other's code and design. You will be assigned to review a particular group's code and design in the final week of the course (läsvecka 8). You perform the review with the entire group and will write a short report about it, with a maximum of two pages (A4). You should be critical and spot inadequacies, but also give appraisal where it is due.

Here are a number of aspects you should consider:

- Do the design and implementation follow design principles?
  - Does the project use a consistent coding style?
  - Is the code reusable?
  - Is it easy to maintain?
  - Can we easily add/remove functionality?
  - Are design patterns used?

- Is the code documented?
- Are proper names used?
- Is the design modular?
- Does the code use proper abstraktions?
- Is the code well tested?
- Are there any security problems, are there any performance issues?
- Is the code easy to understand? Does it have an MVC structure?
- Can the design or code be improved? Are there better solutions?

8 References