UNIVERSITETET I AGDER

KANDIDAT

8803

PRØVE

IS-402 1 Systemutviklingsprosesser og metoder

Emnekode	IS-402
Vurderingsform	Skriftlig eksamen
Starttid	01.12.2016 09:00
Sluttid	01.12.2016 12:00
Sensurfrist	22.12.2016 01:00
PDF opprettet	05.09.2018 13:18
Opprettet av	Digital Eksamen

1

IS-402 Exam Høst 2016

Course code: IS-402
Course name: Systems Development Process and Methods I

Date: 01. December 2016
Duration: 3 hours

Resources allowed: none

Notes: Please answer all exercises in prose. You may, however, use bullet point lists. Please also feel free to make drawings, diagrams, etc. and refer to them. To support your argumentation and for purposes of illustration, you may also make examples. Please note that some of the exercises require an elaborate answer. Nonetheless, grading will be done based on precision and quality of argumentation and *not* on textual length. As a rough hint as to how much time you should spend on each of the exercises (and how long answers in relation should be), their share on the exam's overall grade is given as a percent value.

Sometimes professors ask for exam answers that can be used for teaching purposes, but in order for this to take place, the university needs your consent.

Do you grant the University of Agder permission to use your exam answer for teaching purposes?

- (•) Yes ✓
- ( ) No

Riktig. 0 av 0 poeng.

2

Exercise 1

Exercise 1: Planning (5%)

For the purpose of development project planning, in the lecture *milestone planning*, *Gantt-charts*, and *precedence diagrams* were proposed. Explain *one* of the three methods!

Fill in your answer here

When planning a project there are many factors that must be taken into account. One of them is planning the development effort, and I have chosen to explain the use of milestone planning.

Milestone planning is about planning the project based on well defined milestones. You put them in a diagram, and assign different resources (personell, costs, the need of software/hardware to do the job) to each, as well as a timeframe. In the diagram you do not see the assigned resources, but they should be fully documented next to it.

Milestones

Opening						----
Testing			----------	---------	---------	
Implementation				---------		
Design		----------	--------			
Requirements	----------					
	Dec	Jan	Feb	Mar	Apr	May

Timeframe

When putting the different milestones in a diagram like this, it is easy to see what should be done when and the duration of the activity. By using this you can easily see where you are heading, that the project is moving forward.

Besvart.

3 Exercise 2

Exercise 2: Analysis and Definition (25%)

In a development project, you have been appointed requirements engineer and sit together with the customer. It is your job to identify requirements with her, and to communicate them to your colleagues in system design and implementation. Explain to the customer

1. why requirements engineering is a very important activity,
2. what basic kinds of requirements exist,
3. why documents are needed to write down requirements,
4. in which forms they can be written down, and
5. what user stories and use cases are.

Be sure to give examples for the different kinds of requirements, for a user story, and for a user case.

Fill in your answer here

1. Why requirements

Is it as easy as this; without requirements you dont know what to build! It is essential that you know what the system should do, how it should react and behave, as well as what it should look like in order to build a product. In general it is important that requirements are realistic, unambigous, comlete and consistent.

When compiling requirements communication is essential, and this should preferably happen face to face.

2. Different types of requirements

Requirements can be devided into different kinds, mainly functional and non-functional requirements. Functional requirements is about what the system should do, what functionalities it should have, how it should react to input, as well as what the expected outcome should be. An examle of a functional requirement is "when the camera-button is pushed the camera must open". Non-functional requirements is about the product's quality and performance, eg. usability, storage, robustness, responstime, maintainability. An example of a non-functional requirement is "the system should respond to input in less than 0.6 seconds".

3. Documentation

It is extremly important to document requirements. In this case, since only I am talking to the customer, it is very important that I get the requirements right and well documented so the rest of the team easily can understand what they are going to design/build. They should be written down to make sure everyone has the same intrepretation and understanding, and to minimize the risk that the customer ends up saying "this is not the product i wanted!".

4. Forms of documentation

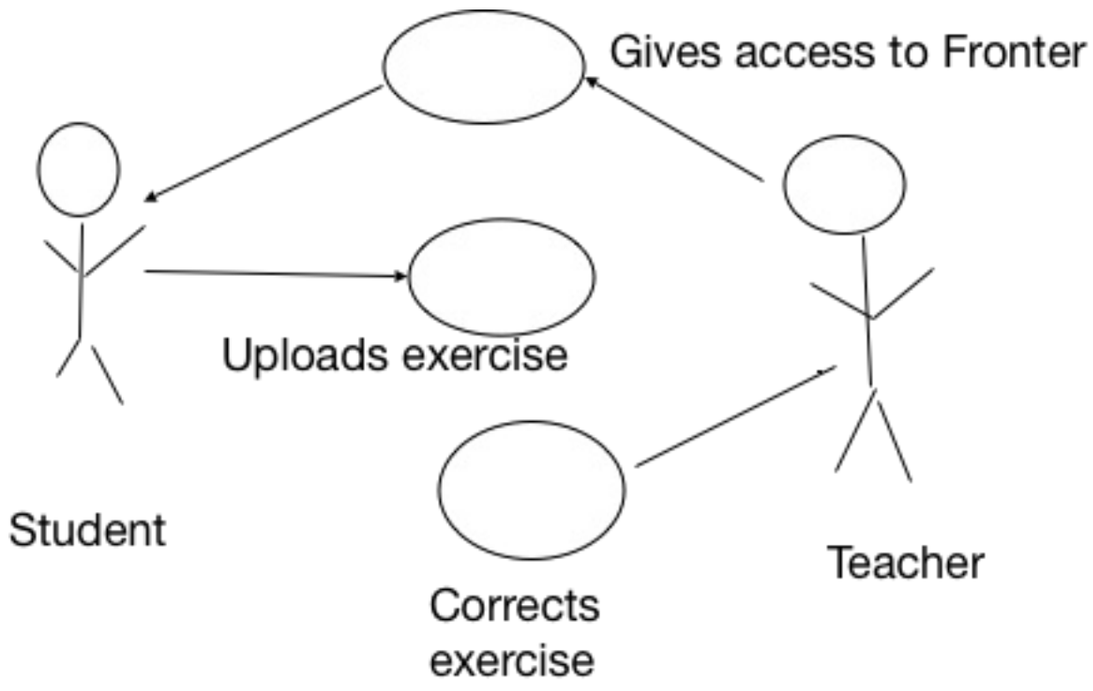
Requirements should be written down in a functional spesification document. This document must cover all the requirements, and it is important that it is comprehensive, unambigous, adequate, complete, consistent and assessable. The outline of the document is aims, product environment, product functionality, product data, product performance, product quality, test-cases, user-stories and use cases, miscellanious e.g.

5. User stories and use cases

User stories and use cases are other ways to compile requirements, and is also a good way to present them.

A user story is a short sentence describing a scenario: As a "role" I want "function", so that "purpose".
An example of this could be: "As a student I want access to Fronter so I can upload an assignment".

Use case diagrams are good to show the communication/interactions between actors ans the system. It is typical written in UML. Here is an example:



Besvart.

4 Exercise 3

Exercise 3: Design (30%)

You are responsible for developing *Kittender* – the Tinder for cat pictures. You have to work both on the graphical user interface (GUI) and on the module architecture.

1. Even though the app will be relatively simple, the users' capabilities need to be taken into account. Explain, what that means! Describe then why humans as users of software demand particular care in designing a user interface!
2. Since people will swipe through images quickly and most of them will only use the small screen of their smartphones, it would waste bandwidth to load high-quality pictures. However, people particularly interested in a kitten may decide to touch the picture to see a fully detailed version of it. The app should normally only load the low-quality version but on request also fetch the high-quality one. This setting asks for the usage of a design pattern. Which pattern do you propose? Explain your decision! Graphically sketch a possible way of usage of the pattern.

Fill in your answer here

1. Designing for humans

When designing a user interface there are several things to have in minds. First, always remember that you are making this for them, and not for yourself. Therefore you need to take into account their skills and previous experiences, and not take for grated that they know as much about software as you do. Every human has different ways of thinking, acting and understanding based on their culture, work, age, mantality e.g.

People are not very exited about learning new stuff, people tend to use things the way they always have. A new app should therefore have a simple design with the use of interface metaphores, so it is easy to understand the different functions and what they do. The design should be based on making main functionality very visible, and use constraints on the functionality that is not much needed.

It is also important to reuse design from similar apps/programs eg. menus, lists, swiping (Tinder).

People also have short term memory, and if it goes a long time between use the app, the user should not have to learn to use it again and again --> design simple so it is easy to remember!

Also it is important to think about "universell utforming", which means that standards such as WCAG should be considered. This is because people with hearing/reading/sight or other physical disabilities also should be considered during design. How can it be build so everyone can use it!?

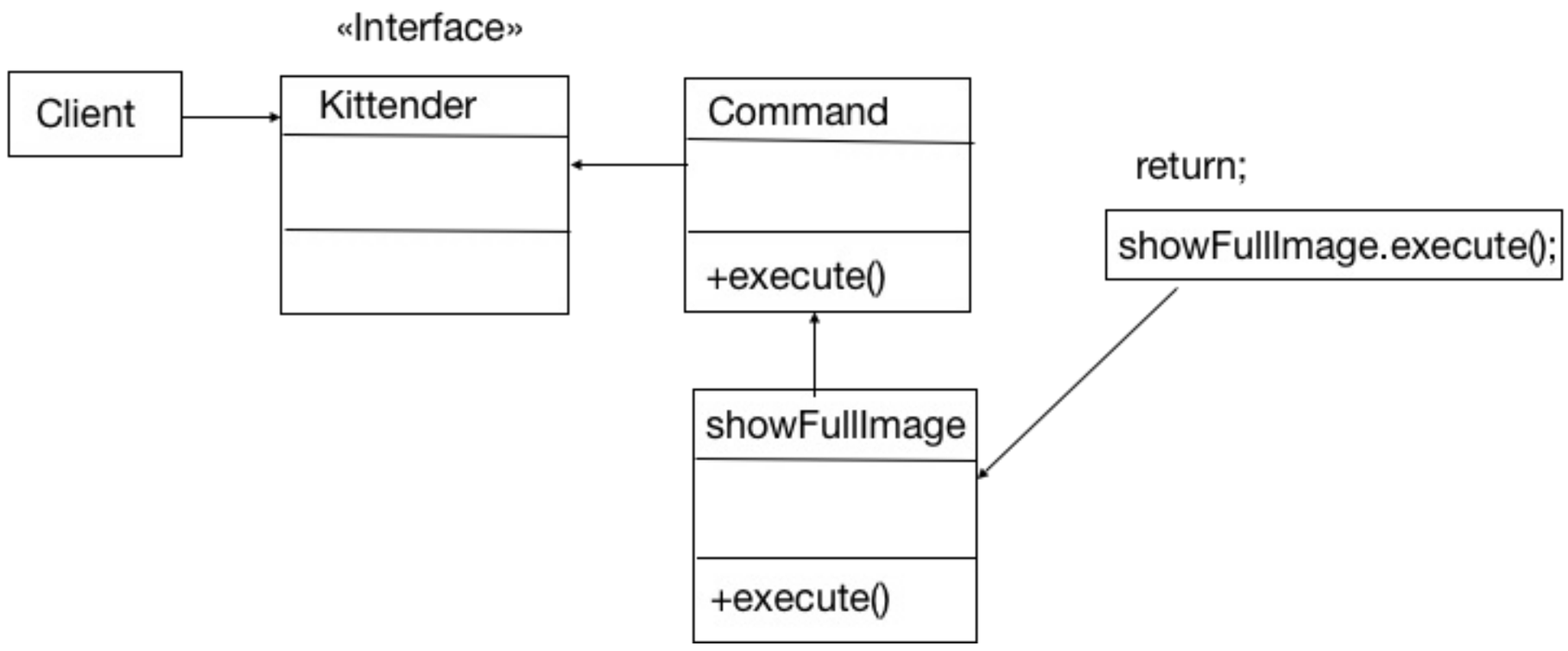
2. Patterns

Design patterns are best practices of how to solve design problems. They can be understood as abstract guidelines to solve repetable problems, and are applied in directly while programming the system.

In this case I would use the Command design pattern. The command pattern is a behavioral pattern, and is concerned about communication in the system. It's intet is to send requests encapsulated as parameters.

If the picture is being requested to show in full size, the system should send a request with the full size as parameter, execute the request, and send it back to the client.

Here is a brief scetch of the structure:



Besvart.

5 Exercise 4

Exercise 4: Implementation (5%)

Three possibilities for saving cost when implementing software have been discussed in the lecture: *software reuse*, *component-based development*, and the *inclusion of third party-libraries*. Explain *one* of these!

Fill in your answer here

When implementing software one of the main wishes is typically to spend as little resouces as possible. It is therefore important to evaluate different ways of how this can be done, and one of these is component-based development. This means that you buy some components that perfectly fits your system, and make the ones that you can not buy yourself, and "glue" these togheter.

This can typically mean that you puchase different components/modules from vendors (should be open source so they can be tailored to your needs). The ones you can not buy you make yourself (program code), and then you write interfaces for communication between them so they can work togheter.

Besvart.

6 Exercise 5

Exercise 5: Testing (10%)

Name and characterize the two general strategies for testing software! Is there also a compromise?

Fill in your answer here

Bottom-Up

There are two genreal strategies when it comes to testing software. These are bottum-up and top-down.

The bottom-up strategy means to start by testing small modules, and then adding more and more functionality as you go. This could for example be starting with a method, then a class - package - subsystem - system.

This can be good to make sure all components are working, and it can be done early in the phase as you don't need much functionality. A negative aspect by using this strategy is that design errors are found late in the process. It is typically used in unit-testing.

Top-down

Top down means to start with testing of large modules, and then adding more and more smaller ones.

In this strategy there is typically much use of mock-objects, which means that you trick the system to believe that the system is ready, but you have actually just put in empty methodes to make the system running.

A good thing about this strategy is that it finds design-errors early, and test the system as a "whole".

Top-down is typically used in component and system testing, as it requires a runnable system with many working modules.

Both strategies are incremental, meaning they add more and more functionality, the difference is buttom-up starts with small components, and top-down with bigger ones.

Compromise

Compromise is a combination of both. By using this strategy you can for example start with top-down, and then switch to bottum-up as you go.

Besvart.

7 Exercise 6

Exercise 6: Software Development Methodologies (25%)

Consider that you need to develop the *Kittender* app described in the design exercise. Imagine that it only needs to be realized as a smartphone app for either Android or iOS. Briefly describe how implementing it using the Waterfall model could look like! Explain then, how the same task would look like with either SCRUM or Extreme Programming! In both cases, be specific to the development project. Do not just describe the methodologies in general but actual steps, actions etc. for *Kittender*. In case you have too few details, make assumptions for preconditions and propose what should be done.

Considering the characteristics of the app, which approach should be favoured? Explain your decision based on a

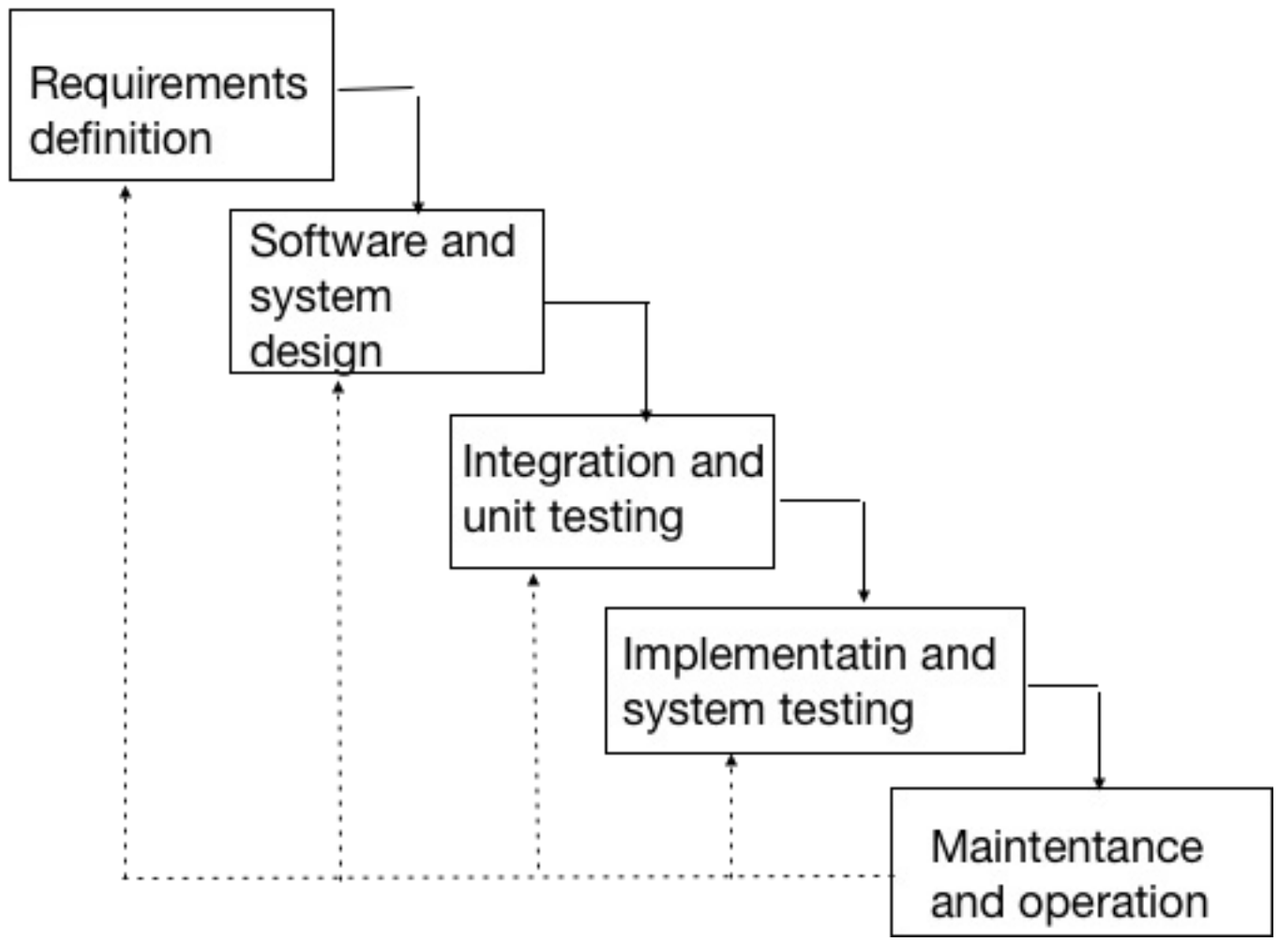
comparison!

Fill in your answer here

Implementation with Waterfall model

The Waterfall method is a plan-driven approach, meaning it is a strict, heavy documented method where activities are performed sequential. It is typically used in big projects in big companies, and in cases where requirements predictable as there is not much room for change when following this method. It is extremly important that all requirements are found well documented before start.

Here is what it looks like, just like a waterfall it is not possible to go upwards, after finishing a step and validating what you have done with the customer, you move on to the next:



If I were to develop Kittender by using the Waterfall method I would start by sitting down with the customer and gather requrements to make sure we are on the same page when it comes to what we are developing. I would get her to write a product requirement document so we have a rough scetch of the product including aims, application domain and target users, product functioanlity, data e.g.

I would then do a feasibility study, risk-analysis and develop a project plan covering scope, quality, time and costs. In this phase I would also set aim for the project, choose a path, consider resources, organize, and plan activities.Then I would write a functional spesification document. Documentation is essential because there is not much room to go back if she was to change her mind.

After planning is done, the project should then move to the system and software design phase. Here should desicions regarding the system be made, eg. architecture; what modules should be included, how should they communicate, what relations and responsibilities should they have. Also desicions regarding the user interface should be made and scetched based on the requirements.

In the next phase, integration and and unit testing, the system (functionalities) should be programmed. Here unit testing is central, and it is important that all the tests passes before moving on to the next phase.

In the implementation phase the system is being put togheter and tested as a whole.

The last phase is concerned about mantenance and operation. Here the system should be tested in a user-perspective, and get feedback from actual users. Also updating and upgrading errors is an important part here.

Scrum:

Scrum is an agile approach based on agile principles such as simple design, frequent delivery, customer collaboration, responding to change etc.

If I was to use this method for developing Kittender, we should first set up a team consisting of 6 +/- three people and three roles. I would name a Scrum master and give him the "coach" role, who would be responsible for that the project would move forwards. The development pepople shold have good programming skills, and also know well about human-computer interaction, design and testing. The last role is the product owner, which would be the lady with the Kittender idea. She would be an important part of the team as we actually are designing for her.

The project would start by data gathering. What does she actually want? She whould write a product requrement document so the team understands the aims, application domain and target users, product functionality, data, desired quality, performance eg. The team would then in collaboraton with the customer make user stroies and use cases, and compile all reqirements in a product backlog. By following an agile approach it is not nessecary to have everything figured out by start, but the team needs to have an idea of what is to be done to start building!

The project would go on in sprints lasting from 2-4 weeks, where different functionality is build. For each sprint some requirements from the product backlog would be moved to a sprint backlog, preferably starting with the most important (The "must-have's", based the MoSCow-prioritizing). Each sprint should result in a working piece of software (an increment). This is a nice way to show the customer progress.

As Scrum is an iterative approach, after each sprint the outcome is reviewd, and changes are done based on customer feedback. This may result in a new sprint, and so the cycle goes on and on until everyone is satisfied.

During the project the team would have daily Scrum meetings, which is very unformal stand-up meetings where progress, drawbacks and problems are discussed.

Comparison:

For this project I would deffinatly choose the method Scrum. This is due to the fact that it is a relatively small project, and because the mobile app-marked is not very stable. What people want is changing constantly, and I would say in this case it would be hard to follow a plan-driven approach as there would be a lot of factors it would be hard to predict. An iterative approach like Scrum is very suited in situations where customers might change their minds and requirements, as it favors customer collaboration and contuously feedback based on the developed increments. The product owner may come over a new app and for example like the design on that one, and thereby want to change the design on Kittender. An agile method like Scrum let's you do this, but a plan-driven method like Waterfall is not working well with change, so if the design was ready and the system-testing phase had begun, a change in the design would not be possible, at least not without serious drawbacks in the rest of the process.

Besvart.