KANDIDAT

8808

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 13:19
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

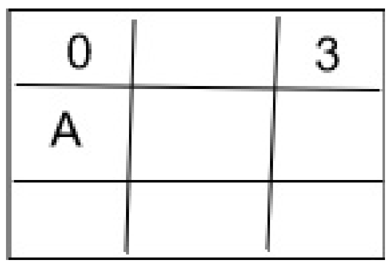
Exercise 1

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

Of the three mention project planning tools i will explain **precedence diagrams:**

Before we can use this diagram we need to have som data beforhand. We need to know all the acticities in the project, how long they are going to last, and the dependency between them. When we know this we can set each activity up in diagrams how looks like this:



Here you can see that we have activity A, that last from 0 to 3 days. When we do this for all the activites we can draw lines to the activities that has dependeency. After that you can calculate around the diagram to find the starting and ending point for the hole project, and the earlyes and latest start for each activity. You will also find witch activites that are on the **critical path**. If a activity is on this path, it means that if you delay that activity whit for example 3 days, the hole project wil be delayed with the same amout of days. Som activities wil also have **slack**. For example if one activity has one week slack, you can delay that activity by up to one week without effecting the project time.

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. The stating point for all procjekt is to find out what we are supost to do. And one of these activites is to document requirements from the customer. This phase is critical for projcet since its the requirements that identify what you are going to do. If you skip this part you can slomst say that you are working in the blind. The developers will not know what they are developing. An extreme example is that the customer wants you to develop a plain, but you are developing a car. The requierments are the guidelinse for the produckt we are going to develop. Requirements engineering is important in all procjet, bout in plan-driven and agile methodology. You have to have som kinde of requirements even in a ad-hoc project.

2. We have three basic kinds of requirements; Functional, Non-functional, and spesial.

- **Functional requirements:** That is requirements about the functions for the system. The requirements have an intended input and extended output. We can say that they are "answers to events". An example on a functional requirements can be: "When the page "employee list" is clickt, it must show a list of all the employees.
- **Non-functioan requirements:** That is requirements about the part of the system that not is functional. We can divide this category into two sub-categorys: Perfromance and quality. The **performance requirements** is about how the perfromance of the system should be, and an example on this can be: "The system must not take longer than 0,4 sec to respond when clicking on a button" The **quality requirements** is about the quality in a very broad way. An example on this can be: The user interface has to be easy to use" or "The colour on the interface has to be apeeling"
- **Spesial requirements:** Its about what we need to develop the system, for testing the system and to deliver the system to the customer. Example on this can be what hardware, software etc. we need need to make the system work.

3. Its important to document the requirements and not just talk about them beacuse they can be misunderstod.

To document the requirements in a Function Spesefication document will prevent this sice it has some critera you have to go truoght. These critera we can have in mind when we are documenting the requiermets so they dont get misunderstod:

- Comprehensibility: Does everybody understand the document, even without skills in IT?
- Unambigousness: Does the requirements hva different meening?
- Adequancy: Does it coer all that the customer wants?
- Completness: Does it cover all we need?
- Consistensy: Does any of the requirements speaks against eachother?
- Assesibility: Can we use the requierments to validate the finished produckt?

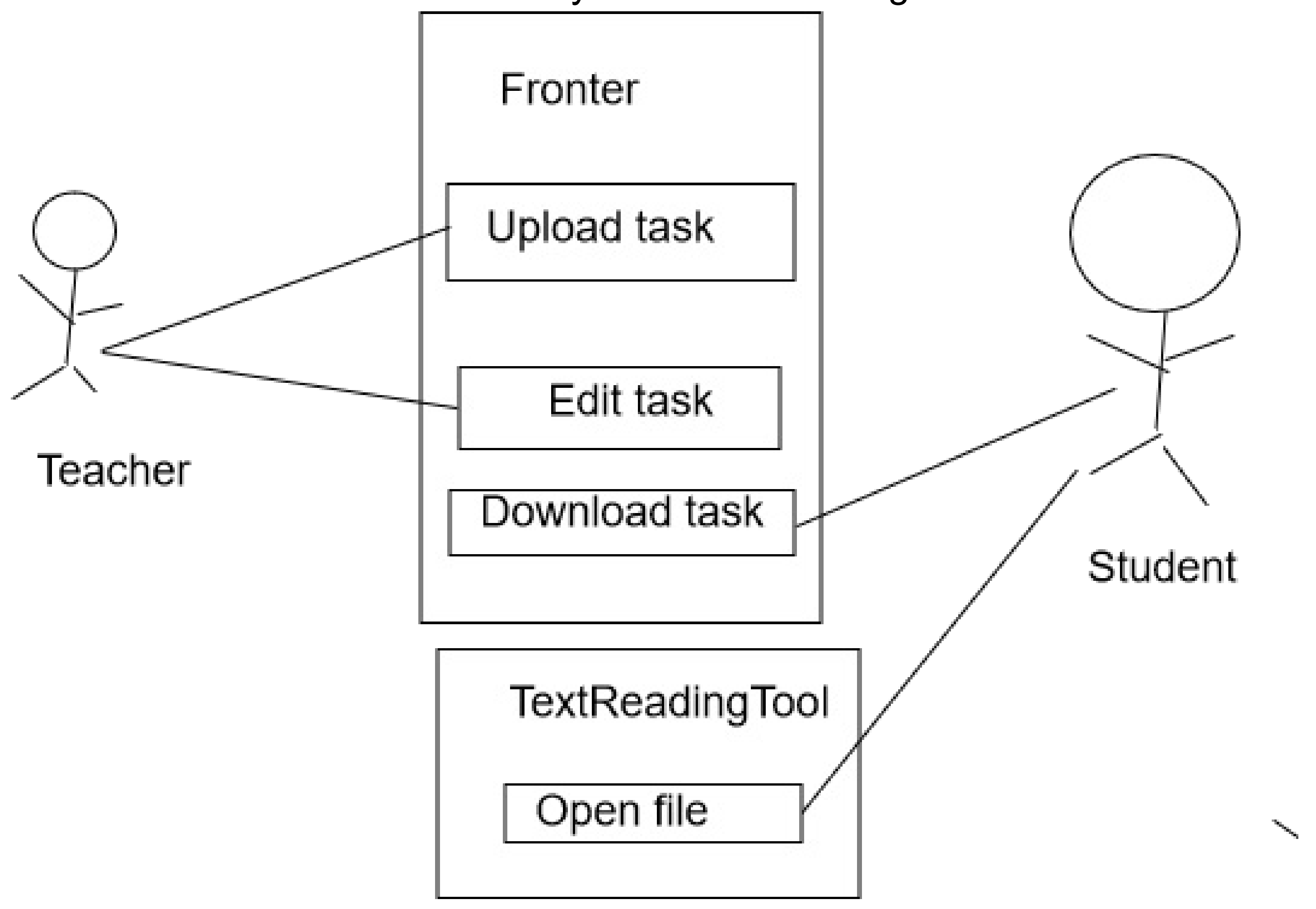
If we have thees in mind, it will be much clearer what we are going to develop. And the wil be no misunderstanding that can leeds to the wrong produckt is beeing developt, like the extreme example in the first part.

4. Before you can start to write the requierments down, you have to identify them. This can be done in several ways like interviews, querrys, doucments from the customer, oberservation, working in groups, and even acting.

When you have identified them, you can start to write them down. This you can do by using natrual structural speak, a graphic notation, or a plain text notation. The most usaly is to combind all these. And some tools you can use to document them i will talk about in the next part (**5**).

5.

- **User stories:** This is a natural way of writing down and document the requierments. We write user stories like this: As a **<role>** i want **<desire/goal>** so that **<benefit>.** The last "so that" part is optional when you are documenting requirements. But i will recomend that we aleays include that part, sice its cover the effect goal for the identified requirement. An example on a user story can be: As a **student** i want **to get access to Fronter** so that **i can find my exercises.** When documenting whit user stories its inmportant to have in mind that user stories does not cover big and complx requirements.
- **Use case:** With use cases you can document the requirements graphically (UML use case). This wil illustrate in a good way the fuctional requirements. The actor of the system is drawn as a stickman (dont have to be a human). And the system/sub system/comonents/modules can be illustrated as boxes, and the actions as sircles. A user story can look somthing like this:



The linse in the UML use case can have an include or exted to give more innformation on what is happening. Also by useing use cases its inmportant to know that it does not coer complex requirements. Do handle this its importhat to have a text with the use case to explain what is happining in words. That way wil prevend misunderstanding.

When working with requierments is important to involve stakeholder and have good communcation with them.

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. First of, degisn is desitions about how you are going to structure the system, how you are going to aviod problems, and how you are going to meet the requirements. A good design wil help the coders to develop the right produckt for the customer, and it wil increase flexibility and maintainence. One of the thing you have to design is the uder intercase. Here you design how the system is going to interact whit the users. By doing this you have to have the human as a software user in you mind all the time. The users can have different **capabilites** that you have to think about. Some of thees i wil list up below:

- The users can have different capabilites when it comes to IT-skils, some can have used IT their hole life, and other ave bearly lookt at a screen before.
- The users can hca different capabilities when it comes to (norsk: funksjonshemming), For example sombody needs biger writings because of their sight. Here they demand particular care in designing a user interface.
- Even if the users have skils in IT, **people make mistakes!** It is important to have in mind that everything that can go wrong wil go wrong (Murphie's law).

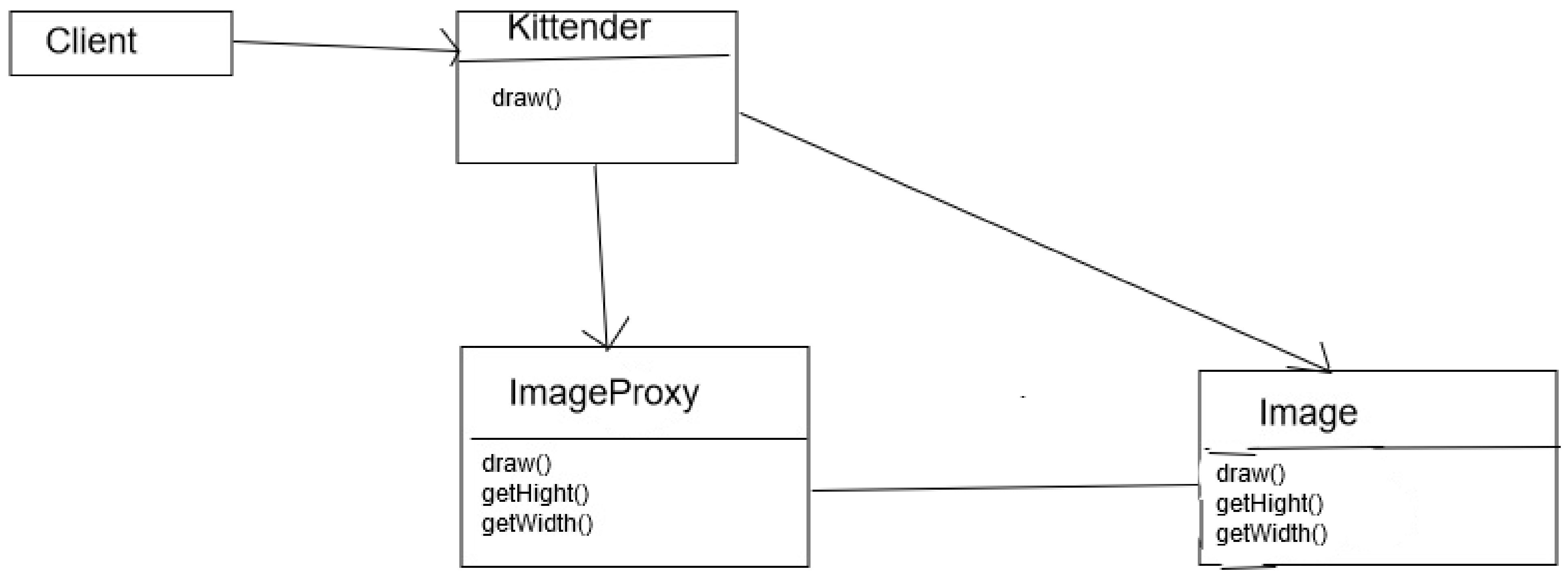
It is many othe reasons for why you need to think about the humand as a software user, and factors that you have to conisder when you are making the user interface. When you are designing the UI you can test it on the customer in different ways to get feedback on that they want:

- Sketching: Here you can use pen and paiper to show the customer how you are planing on designing the system.
- Mockups: Alomst looks like the finished produckt, and can give the users a picture of that the system wil look like.
- Prototyping: Looks and feels like the finished system, that can give the users a better understanding of the user interface.

When using mockups and prototyping its important to have in mind that its very expensive and takes time to develop. It can even give the customer a wrong impresion of how long you have come to developing the system.

2. In system developing projects in genral, its important to use design patters. We can say that design patterns is a collection og "best practises" for component designing. Design patters have been made to prevendt developer to stumble on to problems that happes over and over again in development projects. The design patters can also be directly applyd to the code in the implementation phase and wil give better flexibility and maintainence. We have som main categories when it comes to design patterns, for example; creational patterns, structured patterns and behaviornal patterns.

When im going to choose a design patter for this case, it was som factors in the case that made my desistion. These factors was: small screen, waste to load high quailty images, but must be able to see the full version, and it should normally only load the low quality images, but also fech the high-quality one. So in this case i would recomend to use a proxy pattern. This way we can deside the how big the image wil be despalyd in the app, and to fetch the orginal pictures then its clickt. The reason why this can be a good ide to use, is beacuse in apps like that, people can uplod many pictures in any form. With a proxy we can scale the picture down, so the app wont use to much time to respond. A graphically sketch of a possible way of using the pattern can look somthing like this:



Here the proxy wil take the original picture that was uploaded to th app, and make a new hight and width before it sendt the picture to be delplayd in the app. And of a user clicks on the picture, the full verson wil show.

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

Third pary-libraries:
When we want a new system, or think about developing a new system, it can be important to find out whats in the makred today. Many systems, sub system, libraries etc. is out for sale. And these products are often in a higher quality and the onece you can develop yourself. If you are going to develop a new system, i would consider to by a thirt part library. That will save you for a lot of time, and can even save costs. But you also have to think about that these third pary-libraries can have a licence fees. But if you do the calculation, and find out you can save money on it, its not a effective way to go and reevent the weel. A negative side my a third party can be that you maby have to be depended on the provider, you have to take into account how critcal the part is, and what are going to happen if the provider doesent exsist anymore.

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

The main goal in all testing is to try to make the system fail. By testing the system wil have much higher quality and bigger value for the developers and the customer. Testing can be expensive, but if you skip that part som critcal things can happen that will make it more expensive then the testing. For example if the customer finds many bugs after you have integrated the system, they can demand changes. You also skip some soscial norms by not testing the product, and in worst case scenario you can end up with a law sute. There are many different strategy and technicques when it comes to testing, and for this exercise i will will write about the strategyes: **Buttom-up** and **Top-down.**

Buttom-up is a testing strategy where you start whit the smaller parts of the system, and your your way up. An example on this in object orientaded development we first start by testing the classes, then moving up to packeges, then to sub-system. And when all of this is testes, we then test the system as a whole. The positive side whit this way of testing is that we dont need a lot of mockups. So that can save us som money. The downside is that if you find a bug further up in the system, it can effect things further down in the system. Example: if you fint a bug in the sub-system, you have to change the code in every part of the system that is below the sub system.

Top-down is the oposit of buttom-up. Here you start whit the system, and work your way down. If we use the same example as in buttom-up for object orentated development, we first start whit the system, then sub-systen, packages, and at last we ends up with testing the classes. This way of testing can require lots of mocups and prototyping, somthing that can cost a lot of money. But when you do it this was, you can get a little more control of the testing and find bugs that are critical for the hole system early in the testing phase.

These two strategies also have a **compromise,** where you can combine both strategies. An example on this can be to first start with a top-down to test the whole system first, and after that you change to buttom-up to test the classes and work your way up to the sub-systems.

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

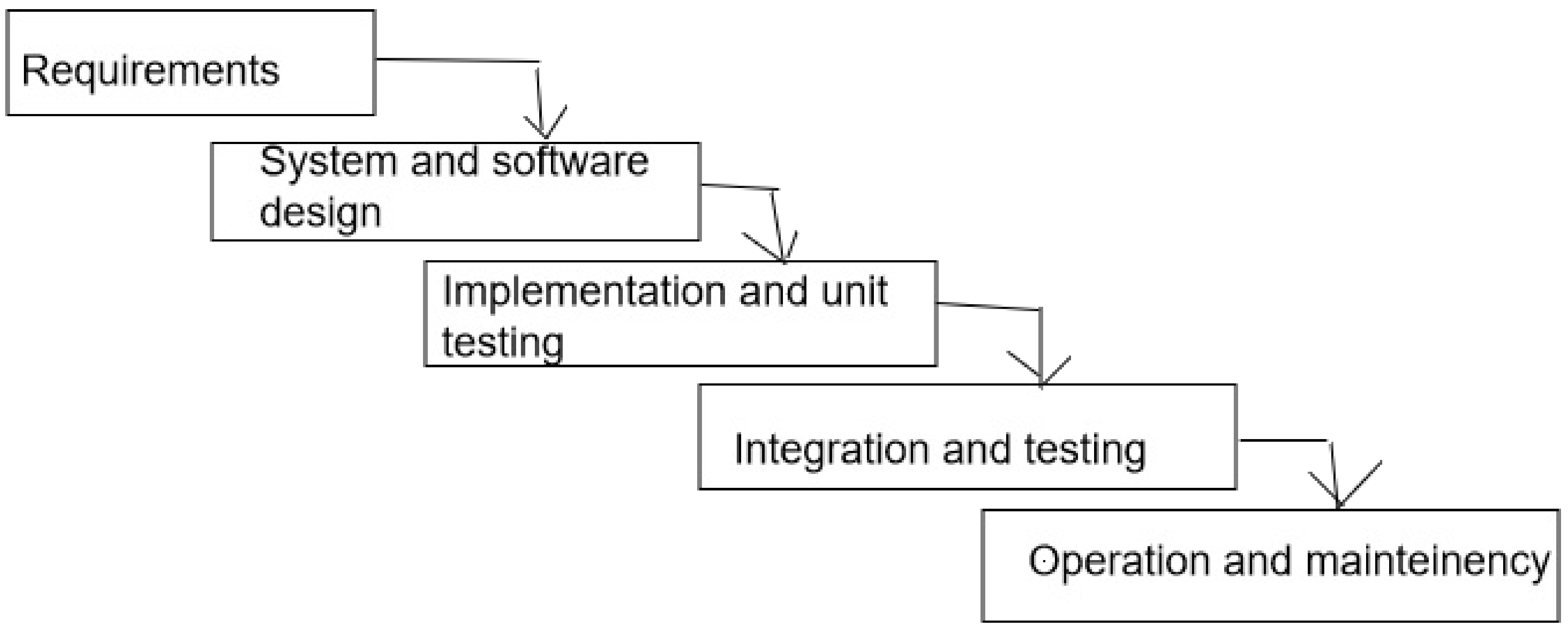
Implementing the Kittender app by using the Waterfall model:

The Waterfall is a model that falls under the category plan-driven methodolgoy. The aims for this model, and in plan-driven development in general is to preticipate the thing we have to do to reach the gole. The model focuses on getting predactibility in three diffrent ways:

- Systematic approtch: Doing the projects
- Difing processes: Document evert process, and follow them
- Strict quality contol: Verify and validate the requierments

The waterfall model has 5 steps you go troth in a developing project. The reason it has the name waterfall is beacuse you go thru one and one stap systemaicly, and when you are finished with one step, you talk to the customer and get hes approval to move to the next. Once you have moved to the next, you can not go back one step, so you can look at it like a waterfall, where you only can go down.

Further i will descripe brifely how developing the Kittender app would be whit a Waterfall model. The Waterfall model lookse like this:


Requirements
System and software design
Implementation and unit testing
Integration and testing
Operation and mainteinency

1. Requirements: Fist of, the Waterfall is plan-driven, so the wirst part will be to make a good plan for the implementation of the APP. The first thing we can do, is to start planning i 5 phases, does phases can be:

1. Set Aims: what do we want with this APP
2. Chose a path: How are we going to make that aims happend?
3. Resorces: What do we have, and what do we need?
4. Orginize: Who is going to be responsible for what?
5. Activity planning: Who is going to do what, and when?

Among with this we have to identify and document the requierments for the APP we are making. Like i mention in an easlyer exerice, this is an important part.

2. System and software design: When we get to this phase we have to make som design desitions about the APP we are making. We need to find out:

- What modules do we need?
- How are the modules going to interact?
- How should the user intercase be
- What comonent will we need?
- What programing language and platform are we going to use
- etc.

3. Implementation and unit testing: In this phase we have to allways have some factors in mind:

- Dont go over the cost
- Meet the requierments
- Develop it whit a minimum amount of bugs
- Develop it so its easy to maintain

For this APP we can use Psudocode to document a basic idea of what the code should look like. An example can me:

```
IF imageIsClicket
showBigImage()
```

This makes it easy for the developer to get a undersatanding of that they have to program, even if they use different programming language.
In this phase we also can make som test cases for our APP.

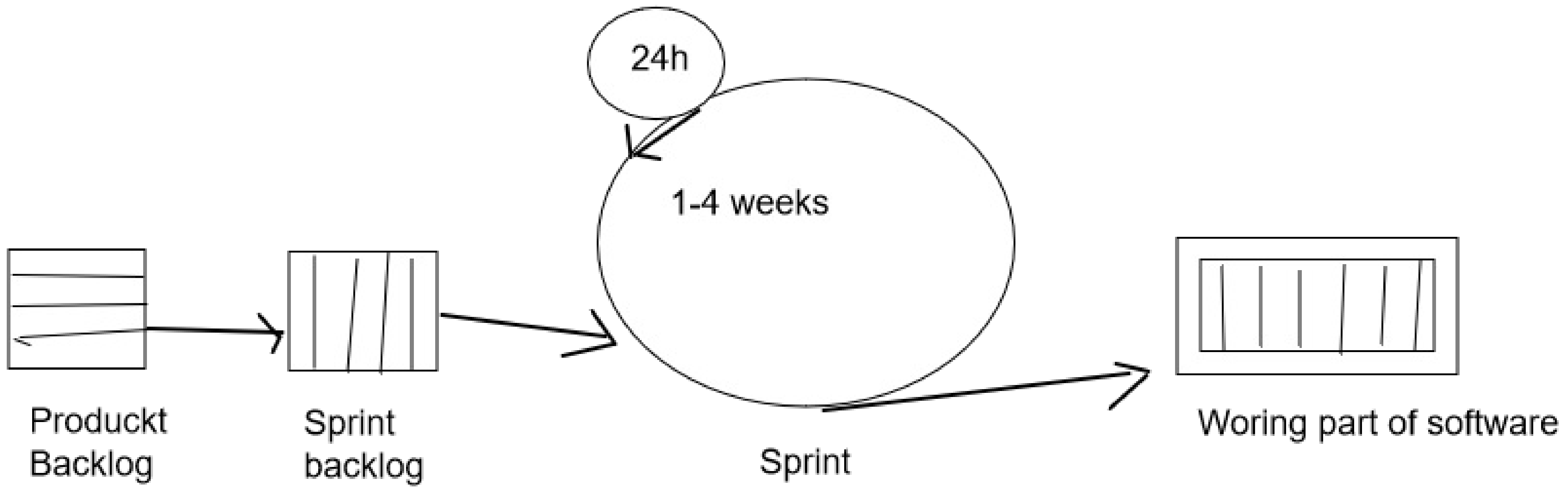
4. Integraion and testing:
Here we have to test and integrate the system. For our APP a performance test is recomended, since users dont like APPs that are slow. Here we can do a load- and a stress test to find out how the system behaves when i gets many requests. The goal for our APP is that is shold respond even with a big amount of stress. Maby security also should be tested, since the APP will gather user information. This we can do by simulate atacks on the APP.

5. Operation and maintainence:
In this phase we need to maintain the APP, like having (norsk:kontinuerlige) oppdates. This phase is all about reviewing and maintain the product. And if we are not happy with the result we have to move all the way back to the first phase and work the way down the steps again.

In all phases its important to document the work that is beeing done!

Implementing the Kittender app by using Scrum:

Scrum falls under the category agile methodology and consist of several sprints. A sprint is small working cycles that usally last from 1-4. The Scrum model lookse somthing like this:



When we follow this model in our developing of the APP, we aslo in agile methods need to requierments. Those requierments we wind we will but in our **produckt backlog.** And its the customer (project owner) of the APP who is the boss of this part. After that the scrum team pick the requierments that they want to do for the given sprint in **sprint backlog.** This activity the prduckt owner, sprint master and team works togheter to find that out. After that the **sprint** starts, where the team is left alone to develop the given rquierments. Here they work on parts of the APP, and pick requirements by post-it notes. Every day wthe team will have a **daily scrum** meeting to talk about what they have done on the APP, the problems and what they are going to do today. After the sprint, the team must have a finshed part of the APP that they can show to the customer. In this phase the customer can have chages, and these changes will be put in the produckt packlog. And then we go thru this steps over and over again until the Kittender app is finished.

To make an APP like Kittender, it is more likely that it is a small project. And since its just one APP we dont need a big team or many people involved in the development. So I would recoment to use a agile approch, and in this case Scrum. This is beacuse in agile methodology the valu priority is a little different from plan driven:

- **Individual and iteraction** over process and tools
- **Working software** over complx planning
- **Customer cooperation** over contract negotiation
- **Responding to change** over following a plan

The factors on the right side will also give value for the APP, but i think those on the left vil give more value.

When we developing an APP we can get many changes, and good comunication whit the customer is important so we develo the APP he wants. Thats way a agile metholdogy will work better for this project. Scrum have a team that consist of 2-9 people, and i dont think we need more. The team consist of experts that can structure themself. A god idea for the implementation part of this APP can be to use pair programing and test driven development. These methods are often used in agilde methods and can give the APP better quality. But it is also important to know that non of the models will give a silver bullet for the project. So a good idea is to consider bout, and ask the developers, customer, and other stakeholders what they think is the best.

Besvart.