Modern Software Development

Serkan Anar serkanan@kth.se
Hassan Dia hdia@kth.se
Kenan Dizdarevic kenandi@kth.se
Mahdi Nazari mnazari@kth.se
Noah William Pettersson noahpe@kth.se
Abud David Zoughbi zoughbi@kth.se

Assignment 1 **3**
Format 1 3
Format 2 3
Format 3 4
Format 4 5
Requirement 1 5
Requirement 2 6
Requirement 3 7
Comparison 8
Detail of information 8
Level of coverage of the information communication the requirement 8
Level of documentation effort 9
Difficulty to write the requirements 9
Result 9
Requirements 9
Requirement items 9
Opportunity alpha 10
Assignment 2 **11**
Test for UC1 11
Test for UC2 11
Test for UC3 12
Estimation 13
Step 1- Agile 13
Step 2 - Traditional 13
Comparison 13
Results 14
Software System Alpha 14
Team Alpha 14
New Checks 15
Requirement 15
Opportunity alpha 15
Assignment 3 **16**
Understanding of the scenario 16
Impression of playing game within software engineering 16
Impression of software engineering with the alphas 17
Assignment 4 **18**
Implementation 18
Design 18
Test-first programming 18
Experience 18
Pair programming 18
Test first programming 19
Refactoring 19

Design before coding 19
Estimation of effort required 19
Assignment 5 **20**
Requirements Alpha 20
Team Alpha 20

Assignment 1

For assignment 1 **Hassan Dia** is the leader, and the customer is **Mahdi Nazari**.

The customer wants a mobile parking meter app. The app is supposed to be able to handle payments and keep track of the time. When the parking time is out the app is supposed to notify the user.

Format 1

Description of the requirement:	**Handle payments**	**GPS Location**	**Track time**
Rationale:	Parking spots cost almost everywhere today. The user experience will be seamless if the payment is automatically processed when the parking time is out.	When opening the app the user wants to start a parking. By implementing a GPS tracker the user will see his position and get the correct fee to pay.	Keeping track of the time for a specific parking is vital as it will be used for the calculation of the fee that the user will pay.
Reference:	Users should be able to pay directly in the app. There should be different payment methods such as card, swish, paypal, crypto. The user only needs to register a payment method once then the payment will automatically proceed when the parking time is out.	The user will in most cases only open the app when it wants to pay for parking. Therefore it would be neat if the user instantly got his own location. The app can then divide parking spots into zones which have different fees.	The user will need to have a timer embedded in the parking meter app. It will be used for specifically setting the time when the user wants to end the parking. The timer will keep track of when the user started the parking and notify when the session has ended.
Source:	Mahdi Nazari, KTH	Mahdi Nazari, KTH	Mahdi Nazari, KTH

Format 2

As a car owner I want to be able to park my car without the need of finding a parking meter so I can spend more time on my errands.

As a car owner I want to be able to get a fee for my current parking spot just by opening the app so I can pay the right fee.

As a car owner I want to be notified before my parking time is up so I can think about other stuff and not get a parking ticket.

Format 3

Use Case ID	UC1	UC2	UC3
Use Case Title	Payment handling	GPS	Timer
Actors	user, parking spot owner, payment gateway providers (such a bank), customer support, parking app	user, parking app, parking spot's owner	user, date and time, parking enforcement officers
Flow of Events	Payment details entered by the customers → At the end of the parking period the payment is sent to the payment processor using the payment details.	GPS receiver receiving signals from GPS satellites → Calculate its location → sends location data to the GPS software → display location on a map	User sets the timer at the beginning of a parking session → parking enforcement officers now have access to the details of the parking session → timer notifies the user when session has ended
Alternative Events	Take the payment beforehand and have them as credit in the users app.	Use codes to identify parking zones, for those that do not permit GPS tracking.	Add cameras to the parking zones. They are connected to the app, and the payment will proceed when the user leaves the zone.
Pre-condition	- Money available in the payment method of choice. - Payment processor is up and running State: Payment Pending	- GPS shows the correct location. State: User verifying	- App is connected to the correct time zone that the user will park its car at. State: Parking in progress
Post-condition	- Payment received by payment processor State: Payment Complete	- Correct location is found and verified. State: Location found	- App is connected to the correct time zone that the user will park its car at. State: Parking ended
Assumption	- Money available in	- No signal	- That date and time

	the payment method of choice and the payment info is already filled in.	obstructions -A correct GPS signal is already in use	corresponds to the correct time zone that the user is located at.
Priority	1	3	2

Format 4

Requirement 1

<u>General Requirement Description</u> **Requirement ID:** UC1 **Requirement Title:** Payment handling. **Requirement Des cription:** Process the payment when the parking time is out. **Requirement Type:** Functional Requirement. **Internal/external Req:** External **Rationale:** The user should not spend extra time on entering the details after every finished parking. **Event/Use case ID:** UC1.0 **Related to requirement(s):** UC2 & UC3. **Conflicting Req:** None. **Non-functional Requirements:** Make it available for iOS and Android. Deadline 2023-04-01. **Constrains:** Other actors which process the payments, API's and banks. **Intended users:** Vehicle drivers **Specific user who stated the req:** Mahdi Nazari **Customer Satisfaction:** High. **Customer Dissatisfaction:** Low, users can forget that the timer is on when it is done thus paying for a parking which is not active. **Reference Documents:** None. **<u>Requirements Evaluation Data</u>** **Business Value:** Cooperation with a company which has developed the secure payment system.	**<u>Other Description Data</u>** **System Data:** It will use safe payment systems which are already developed. **Adjacent/Interfacing Systems ID:** Cooperate with UC2 & UC3. **Environment:** Software in the parking app. **<u>Requirements Reporting Data</u>** **Problem Reporting Date:** 2023-03-21 **Originated By:** Mahdi Nazari **Reported By:** CEO Parking startup **Problem Owner:** Developer team **<u>Requirements Management Data</u>** **Preliminary Implementation Plan:** Develop the payment system for a parking app which works with GPS and timer. **Preliminary outline of activities:** Develop the app, perform testing on the implemented system. **Planned and actual activity(s):** ● **Activity Description:** Implement the payment handling system into the parking app with a good user interface. ● **Activity Start Date:** 2023-03-21 ● **Activity End Date:** 2023-04-01 ● **Actual Result:** Payment handling with good UI. ● **Activity Conducted By:** Developer team ● **Activity Approved By:** Developer & testing team ● **Effort Spent on Activity:** Two weeks

Other Value: Vehicle drivers will be the user group which we target, mainly car drivers. **Requirements Priority (Rank):** It is the most vital requirement of the app. The user needs to be able to pay for the parking otherwise the app is useless and the user can use a “p-skiva”. **Fit Criteria:** Pay the parking and save the payment information, send receipt. **Risk:** Saving information of payment details. If one payment method fails some users are not able to park their car.	● **Cost of Activity:** Developer salaries and payment system cost. **<u>Requirements Completion Data:</u>** **Actual Completion Date: -** **Planned Completion Date:** 2023-04-01 **Relation To Tests:** Test1: transaction was done successfully. **Released In:-** **Requirement Completion Approved By:-** **Estimated Total Effort:** two weeks **Actual Total Effort: -** **Estimated Total Cost: -** **Actual Total Cost: -**

Requirement 2

<u>General Requirement Description</u> **Requirement ID:** UC2 **Requirement Title:** GPS **Requirement Description:** Find the parking location using the gps signal. **Requirement Type:** Functional Requirement. **Internal/external Req:** External **Rationale:** The user should not have to search or enter a specific location instead the app should just find the correct location. **Event/Use case ID:** UC2.0 **Related to requirement(s):** UC1 & UC3. **Conflicting Req:** None **Non-functional Requirements:** Deadline **Constrains:** No constraints **Intended users:** Vehicle drivers **Specific user who stated the req:** Mahdi Nazari **Customer Satisfaction:** High **Customer Dissatisfaction:** Low, can potentially give an inaccurate location **Referens Documents:** None **<u>Requirements Evaluation Data</u>** **Business Value:** No cooperation with another company is made for this requirement.	**<u>Requirements Reporting Data</u>** **Problem Reporting Date:** 2023-03-21 **Originated By:** Mahdi Nazari **Reported By:** Mahdi Nazari **Problem Owner:** Developer team **<u>Requirements Management Data</u>** **Preliminary Implementation Plan:** Displays the correct location and receives the correct fee rate for the parking. **Preliminary outline of activities:** Develop the app, perform testing on the implemented system. **Planned and actual activity(s):** ● **Activity Description:** Implement the GPS so that we can correctly retrieve the fee rate for the parking. ● **Activity Start Date:** 2023-03-21 ● **Activity End Date:** 2023-04-01 ● **Actual Result:** Display location and correct fee rate ● **Activity Conducted By:** Developer team ● **Activity Approved By:** Developer & testing team ● **Effort Spent on Activity:** Two weeks

Other Value: **Requirements Priority(Rank):** This requirement will have the third priority rank of all the requirements. **Fit Criteria:** Find the correct parking area and display the correct pricing. **Risk:** GPS signal might be inaccurate and may display the wrong pricing at the place the user is standing at which can cause problems for both the user and the mother company. **Other Description Data** **System Data:** It will use the GPS system of the phone in order to identify the parking spot to specify the fee, and will be implemented with the timer and payment system. **Adjacent/Interfacing Systems ID:** Cooperate with UC1 & UC3. **Environment:** Software in the parking app.	● **Cost of Activity:** Developer salaries and payment system cost. **Requirements Completion Data:** **Actual Completion Date: -** **Planned Completion Date:** 2023-04-01 **Relation To Tests:** Test1: a location accuracy check was successfully done. **Released In:-** **Requirement Completion Approved By:-** **Estimated Total Effort:** two weeks **Actual Total Effort: -** **Estimated Total Cost: -** **Actual Total Cost: -**

Requirement 3

General Requirement Description **Requirement ID:** UC3 **Requirement Title:** Timer **Requirement Description:** Keep track of time from start to end of the parking. **Requirement Type:** Functional Requirement. **Internal/external Req:** External **Rationale:** The parking shall end when time is up, and a payment must be processed. **Event/Use case ID:** UC3.0 **Related to requirement(s):** UC1 & UC2 **Conflicting Req:** None **Non-functional Requirements:** Deadline **Constrains:** No constraints **Intended users:** Vehicle drivers **Specific user who stated the req:** Mahdi Nazari **Customer Satisfaction:** High **Customer Dissatisfaction:** Low, timer is unlikely to be inaccurate.	**Requirements Management Data** **Preliminary Implementation Plan:** Check the app that it can count an amount of time. **Preliminary outline of activities:** Develop the timer and test that the timer is working correctly. **Planned and actual activity(s):** ● **Activity Description:** Implement a timer that tracks the total time a vehicle is parked at a specific parking spot. ● **Activity Start Date:** 2023-03-21 ● **Activity End Date:** 2023-04-01 ● **Actual Result:** Display the correct time that the vehicle has been parked at the parking spot. ● **Activity Conducted By:** Developer team. ● **Activity Approved By:** Developer & testing team. ● **Effort Spent on Activity:** Two weeks

Referens Documents: None **<u>Requirements Evaluation Data</u>** **Business Value:** A day time server **Other Value:** None **Requirements Priority(Rank):** This requirement is next most important after payment handling, the app needs a timer to be able to charge the user by a right fee. **Fit Criteria:** A memory is needed to keep track of the timer. **Risk:** daytime server could be down. **<u>Other Description Data</u>** **System Data:** The requirement will perform independently and does not need any subsystem. **Adjacent/Interfacing Systems ID:** Cooperate with UC2 & UC3. **Environment:** Software in the parking app. **<u>Requirements Reporting Data</u>** **Problem Reporting Date:** 2023-03-21 **Originated By:** Mahdi Nazari **Reported By:** Mahdi Nazari **Problem Owner:** Developer team	● **Cost of Activity:** Developer salaries and payment system cost. **<u>Requirements Completion Data:</u>** **Actual Completion Date: -** **Planned Completion Date:** 2023-04-01 **Relation To Tests:** Test3: Timer check successful. **Released In: -** **Requirement Completion Approved By:-** **Estimated Total Effort:** two weeks **Actual Total Effort: -** **Estimated Total Cost: -** **Actual Total Cost: -**

Comparison

Detail of information

Format 1-3 all contain a general idea of the requirements. They are short and easily understood. Format 2 and 3 also contain some information on how the interaction between the user and the app should happen. Format 4 is a more formal format of the requirements that gives detail of the requirements, who it originated from, who implements it, tests it, approves it etc.

Level of coverage of the information communication the requirement

Format 4 is the one that covers the most information about the requirement. The documentation in format 4 lets us know how to implement what and who will test and approve the implementation. Format 2 does not cover much information about the requirement, it gives the reader a general picture of what is supposed to be implemented. Format 1 and 3 go a bit deeper into the requirement but not as deep as in format 4.

Level of documentation effort

Format 4 needs the most effort for the documentation whereas format 2 needs the least. Format 2 gives the reader a good overview but format 4 is more in depth and gives the reader a clear understanding of who is supposed to do what.

Difficulty to write the requirements

Format 2 is the easiest to write, while format 1 and 3 are somewhat more demanding in comparison to format 2. Format 4 is the most difficult and most demanding way of writing requirements, but gives a very detailed view of them.

Result

Requirements

- **Conceived**

- [x] Stakeholders agree the system is to be produced.
- [x] Users are identified.
- [x] Opportunity is clear.

- **Bounded**

- [x] Development stakeholders identified.
- [x] System purpose agreed.
- [x] System success is clear.
- [x] Requirements' format agreed.
- [x] Prioritisation scheme clear.
- [x] Assumptions are clearly stated.

- **Coherent**

- [x] Requirements shared.
- [x] Requirements' origin clear.
- [x] Rationale clear.
- [x] Prioritise cleared.
- [x] Key usage scenarios explained
- [x] The impact of implemented requirements are understood.
- [x] Team knows and agrees on what to deliver.

Requirement items

- [x] The items are identified.
- [x] They are described with different formats.

Opportunity alpha

- **<u>Identified</u>**

☑ Idea behind the opportunity is identified.
☑ At least one investing stakeholder is interested.

- **<u>Solution needed</u>**

☑ Solution identified
☑ Stakeholders' needs are identified.
☑ Problems and root causes are identified.
☑ Need for a solution confirmed.
☑ At least one solution proposed.

- **<u>Value Established</u>**

☑ Solution impact understood.
☑ System value understood.
☑ Outcomes clear and quantified.

Assignment 2

Test for UC1

Test case ID:	100
Requirements:	User able to connect payment methods.
Input:	Eg. credit card, paypal, crypto etc.
Output:	User connected.
Environmental needs:	Connection to safe payment system.
Inter-case dependencies:	None.
Special procedural need:	Insight into the payment system.

Test case ID:	101
Requirements:	Process payment.
Input:	Time parked.
Output:	Fee calculated and receipt sent.
Environmental needs:	Payment system and software for timer.
Inter-case dependencies:	UC3.
Special procedural need:	Insight into the payment system.

Test for UC2

Test case ID:	200
Requirements:	Request and successfully retrieving location address.
Input:	User geoinformation.
Output:	Location successfully returned

Environmental needs:	Device with GPS.
Inter-case dependencies:	None.
Special procedural need:	None.

Test case ID:	201
Requirements:	Get parking zone.
Input:	User geoinformation.
Output:	Current position of the device and the current zone displayed on the device.
Environmental needs:	Device with GPS.
Inter-case dependencies:	None.
Special procedural need:	Connection to e.g. google maps to display position.

Test for UC3

Test case ID:	300
Requirements:	Retrieve correct time.
Input:	Button click.
Output:	Current time.
Environmental needs:	Internet connection.
Inter-case dependencies:	None.
Special procedural need:	Connection to time specific server.

Test case ID:	301
Requirements:	Display correct time

Input:	User geoinformation.
Output:	Correct time based on users location.
Environmental needs:	Internet connection.
Inter-case dependencies:	None.
Special procedural need:	Time specific server working correct.

Estimation

Step 1- Agile

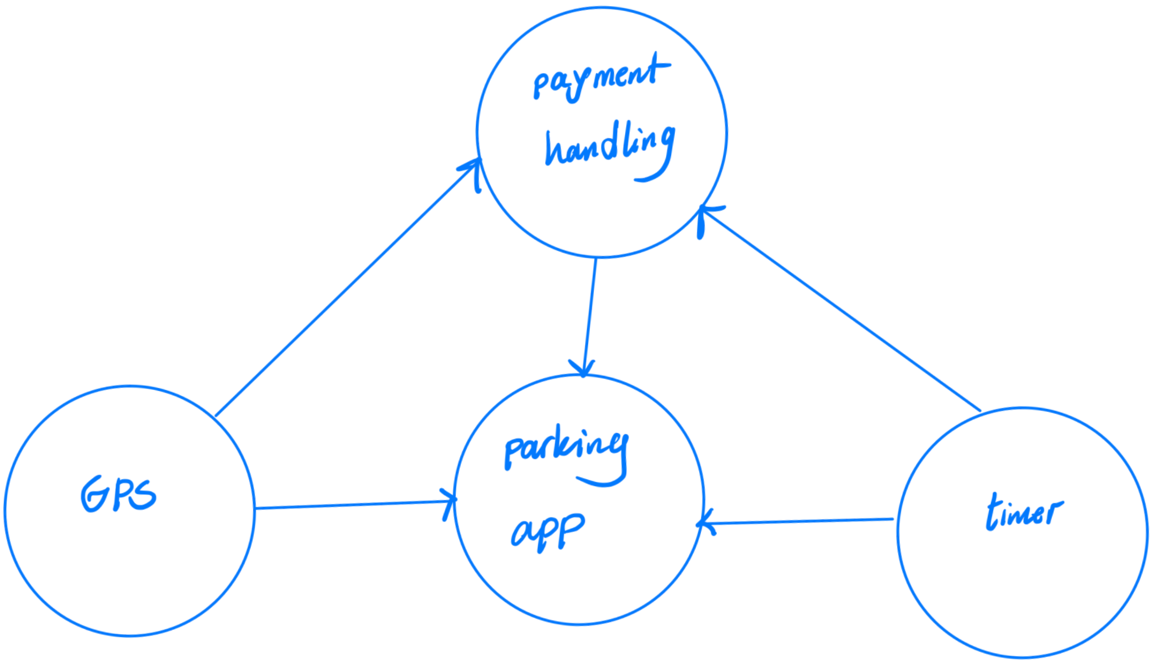
Estimator	**UC1 & 100 & 101 (R1)**	**UC2 & 200 & 201 (R1)**	**UC3 & 300 & 301 (R1)**	**UC1 & 100 & 101 (R2)**	**UC2 & 200 & 201 (R2)**	**UC3 & 300 & 301 (R2)**
Kenan	40	25	5	45	20	8
Abud	50	20	12	45	20	8
Hassan	45	20	10	45	20	8
Noah	40	15	4	45	20	8
Mahdi	50	20	4	45	20	8
Serkan	72	24	8	45	20	8

Step 2 - Traditional

UC1 (LOC)	**UC2 (LOC)**	**UC3 (LOC)**	**UC1 (time)**	**UC2 (time)**	**UC3 (time)**
120	80	45	33	15	4

Comparison

Traditional estimation techniques measure the size of a software project by lines of code or function points, requiring a large amount of work and often vary depending on the language and the skill of the programmer. Agile estimation techniques use user stories and story points, which are relative measures that consider the skill level and domain knowledge of each developer. Additionally, agile estimations consider both ideal and elapsed time. Task switching between projects, misunderstandings of requirements, and other possible obstacles must also be taken into account.



Results

Software System Alpha

- **Architecture Selected**

☑ Hardware platforms identified.
☑ Technologies selected.
☑ System boundary known.
☑ Key technical risks agreed to.

Team Alpha

- **Seeded**

☑ Mission defined.
☑ Growth mechanisms in place.
☑ Required commitment level clear
☑ Size determined.

- **Formed**

☑ Enough members recruited.
☑ Roles understood.
☑ Members introduced.
☑ Members accepting work.
☑ External collaborators identified.

- [x] Communication mechanism defined.
- [x] Members commit to the team.

New Checks

Requirement

- **<u>Bounded</u>**
- [x] Shared solution understanding exists.
- [x] Requirements management in place.

- **<u>Coherent</u>**
- [x] Conflicts addressed.
- [x] Essential characteristics clear.

- **<u>Acceptable</u>**
- [x] Acceptable solution described.
- [x] Value to be realised clear.
- [x] Clear how opportunity addressed.
- [x] Testable.

Opportunity alpha

- **<u>Viable</u>**
- [x] Solution outlined.
- [x] Solution possible with constraints.
- [x] Solution profitable.
- [x] Reasons to develop solution understood.

The requirement and opportunity alphas have progressed as a result of the tests and system design.

Assignment 3

Understanding of the scenario

The scenario is used to illustrate how Essence can be used in order to check the status of a project. Their methodological approach is very clear, as they choose to go through one alpha at a time, since it is not possible to go through them all simultaneously. As a result of their approach, they could determine what needs to be done next.

Impression of playing game within software engineering

Using planning poker technique has clear advantages and some drawbacks in software engineering.

Advantages includes:

1. Collaborative approach: Planning poker encourages collaboration and communication among team members. Each team member has a say in the estimation process, and the final estimate is based on a consensus of the group.
2. More accurate estimates: Planning poker helps to improve the accuracy of project estimates. Since team members bring their unique perspectives and expertise to the table, the estimates are more reliable.
3. Transparency: Planning poker promotes transparency in the estimation process. Each team member's estimation is visible to the group, and this helps to avoid bias and encourages honesty.
4. Speeds up planning: Planning poker is a quick and efficient way of estimating project efforts. Since it involves a group estimation process, it saves time compared to other estimation techniques.
5. Improved project planning: By using planning poker, software engineering teams can better plan their projects.

Drawbacks includes:

1. Time-consuming: Poker planning can be time-consuming, especially when used for estimating large and complex projects. It requires all team members to participate in the estimation process, which can be challenging to coordinate.
2. Bias: The estimation process in poker planning can be influenced by various biases, such as anchoring bias, groupthink, and availability bias, which can affect the accuracy of the estimates.
3. Lack of accuracy: The estimates produced using poker planning are not always accurate, as they are based on subjective judgments and assumptions. This can lead to inaccurate planning, delays, and cost overruns.
4. Overreliance on estimates: Teams may become too reliant on estimates, which can lead to a lack of flexibility and adaptability when changes occur.
5. Focus on individual estimates: Poker planning can focus too much on individual estimates, leading to a lack of collaboration and discussion among team members. This can result in suboptimal planning and decision-making

Impression of software engineering with the alphas

The "*Seven Essential Things"* is a good way of keeping track of the important parts of the project. The alphas give every stakeholder access to the information, thus every member of the team knows what needs to be done.

A problem when grasping the whole domain of software engineering with the alphas is that it requires a combination of theoretical knowledge and practical experience. Theoretical knowledge provides the foundation for understanding the principles and concepts of software engineering, but practical experience is essential for developing the skills and expertise necessary to apply this knowledge in real-world settings. Without practical experience, it can be difficult to fully understand the nuances and complexities of software engineering, and to develop the critical thinking and problem-solving abilities required to succeed in the field.

Assignment 4

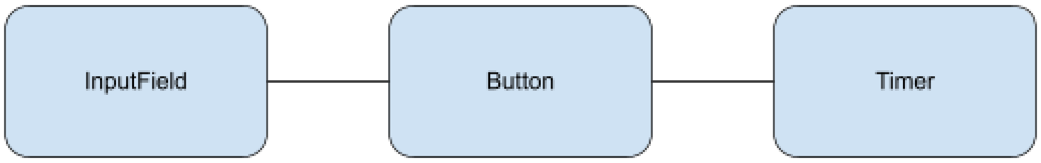
Implementation

Requirement: UC3, timer
Effort required: We approximate that the timer with the user interface will take 8 hours. Since we will not implement the user interface in this task we approximate that it will take 2.5 hours to implement a working timer.
Risk analysis: Servers which contain information about the current time could be down. It is also possible that we will not implement the timer in time.

Design

We are going to utilise only one class since there will not be many lines of code.

First we need to fetch the current time of the user. The user will then be able to write the desired departure time in an input field. When the user presses the button the parking will automatically start.



Pseudo code for timer:

```
● Get user time
● User input time
● If time is ok
      - Start timer
● Else
      - User input time again
● Continue with the next program steps here.
```

Test-first programming

Try inputting a time, click the button and check whether the correct time is displayed and ticks down.

Experience

Pair programming

Advantages	Disadvantages
● Better code quality. ● Knowledge sharing. ● Faster problem-solving. ● Improved team communication.	● Cost. ● Compatibility. ● Fatigue. ● Reduced autonomy.

Test first programming

Advantages	Disadvantages
● Clear picture of what is to be done. ● Confirmation if code is working. ● Better code quality. ● Improved design.	● Hard to implement new vital functions. ● If big change is needed a lot of tests need to be rewritten, thus consuming much time. ● Time consuming. ● Over-reliance on the tests.

Refactoring

Advantages	Disadvantages
● Improved code quality. ● Increased productivity. ● Reduced technical debt. ● Better collaboration.	● Time consuming ● Risk of introducing bugs ● Potential for disruptions ● Cost

Design before coding

Advantages	Disadvantages
● More efficient coding if one makes a plan beforehand. ● Easier to spot mistakes if you have a blueprint.	● Difficult to evaluate the needed steps for a code.

Estimation of effort required

We estimated that it would take us approximately 2.5 hours to implement the requirement. Instead it only took us 1.5 hours. It is hard to estimate the effort required for us because we are inexperienced developers. It is also hard to determine the complexity of the software and there are also external factors to take into account.

Assignment 5

Requirements Alpha

- *Conceived state:* Passed
  Stakeholders agree that the system is to be produced, the stakeholders are students, administrators and faculty. The users of the management system will be the administrators of the faculty. The university and faculty will fund the project. The opportunity is clear, the administrators will easily manage the courses.

- *Bounded state:* Passed
  Development stakeholders are identified, there will be four developers, an IT-director and the faculty members which will provide feedback. The purpose of the system is agreed and clear, it will benefit the administrator. The team is having meetings with the stakeholders to discuss the given feedback. There are also assumptions about how much and what the system will be capable of doing.

- *Coherent state:* Passed
  The coherent state has passed due to the interviews which the team has conducted. The requirements have been shared and the purpose of them is clear. The impact is understood because it makes the work of the administrative staff easier.

- *Acceptable state:* Failed
  "The "unhappy" stakeholders have stopped complaining about lack of functionality. They now accept the new solution and pledge to migrate to the new system this coming semester."
  This quote is one good example why the acceptable state is not passed yet. There is not yet a clear and acceptable solution described since new implementations need to be discussed.

Team Alpha

- *Seeded state:* Passed
  This state passes because the mission is clear, and the team knows its capabilities and constraints.
- *Formed state:* Passed
  This state passes because a team with enough members has been formed. The individuals in the group understand their role, and are all responsible and committed to doing their work. Moreover, oral communcation has been determined as the team communication mechanism.
- *Collaborating state:* Failed
  The collaborating state fails because there seem to be some issues between the team members. The following quote illustrates the problem: "The faculty agreed with the first developer, and the second developer felt somewhat put out that his opinion did not seem to matter. Since this was not the first time that the second developer's ideas had not been accepted, little by little he stopped communicating with the team". It is clear and evident that this developer does not communicate as much as he should with the team, and there are obviously some problems regarding communication within the team. Without good communication, collaborating is difficult and It can therefore be concluded that the collaborating state fails.