Danmarks Tekniske Universitet

Link til Projekt GitHub

CDIO Final - Gruppe 22


Mads Storgaard-Nielsen
s180076

Sander Eg Albeck Johansen
s195453


Jonas Løvenhardt Henriksen
s195457

Frederik Verne Henriksen
s173394


Mikkel Hillebrandt Thorsager
s195470

Alexander Lambrecht
s195482

1 Intro

1.1 Abstract

1.2 Indledning

2 Timeregnskab

Arbejdsområde	Mads	Frederik	Jonas	Mikkel	Sander	Alexander	**I alt**
Projektplanlægning	2	2	2	2	2	2	**12**
Kravspecifikation	2	2	2	2	2	2	**12**
Use Cases	2	2	2	2	2	2	**12**
Programmering/Versionsstyring	0	0	0	0	0	0	**0**
Tests	0	0	0	0	0	0	**0**
Rapportering	0	0	0	0	0	0	**0**
Total	**6**	**6**	**6**	**6**	**6**	**6**	**36**

Indholdsfortegnelse

3 Kravspecifikation

Kravene er inddelt i kategorier efter features, og herefter prioriteret efter MoSCoW principperne.

(M) = Must Have

(S) = Should Have

(C) = Could Have

(W) = Won't Have

3.1 Kravliste efter kategorier

Basale Brætspil Funktioner:

R1: (M) Der er en spil plade

R2: (M) Der skal kunne være 3-6 spillere

R3: (M) Man skal kunne indtaste sit navn

R4: (M) Man skal kunne bevæge sig rundt med uret på spilpladen

R5: (M) Når du passerer start får du 4000 kr.

R6: (M) Man har to seks øjet terninger man slår, udfaldet flytter dig på brættet

R7: (S) Hvis man slår to ens, må man slå igen

R8: (C) Hvis man slår to ens, tre gange i træk ryger du i fængsel

R9: (S) Man vælger mellem forskellige typer brikker for spillerne som UFO, biler, skibe.

Køb af Ejendomme:

R10: (M) køb af grund

R11: (S) dobbelt husleje når alle felter i farve ejes

R12: (S) ekstra husleje ved huse eller hotel

R13: (S) betal ekstra ved eje af flere færgelinjer

R14: (S) Betal antal af øjne gange 100 ved bryggerier

R15: (S) ved eje af begge bryggerier skal man i stedet betale 200 gange antal af øjne man slår

R16: (S) Bygge huse/hoteller

R17: (S) Når du har 4 huse, og køber endnu et, bliver det til et hotel

R18: (C) Auktion hvis en spiller vælger ikke at købe grunden

R19: (C) Pantsætning

Chancekort:

R20: (S) Når man lander på et prøv lykken felt, trækkes et lykkekort, og handlingen på kortet udføres.

Indkomstskat:

R21: (C) Betale indkomstskat

Fængsel:

R22: (S) Man kan ryge i fængsel

R23: (S) Du kan komme ud ved at slå to ens

R24: (S) Du kan maks sidde i fængslet i 3 ture i træk

R25: (C) Kan komme ud af fængsel ved at betale 1000 kr.

R26: (C) Kan købe grunde i fængslet

Konklusion af Spil:

R27: (M) Mangler penge til at betale funktion, gå fallit

R28: (C) Vælg hvilke grunde/boliger du vil sælge

R29: (M) Vinder funktion

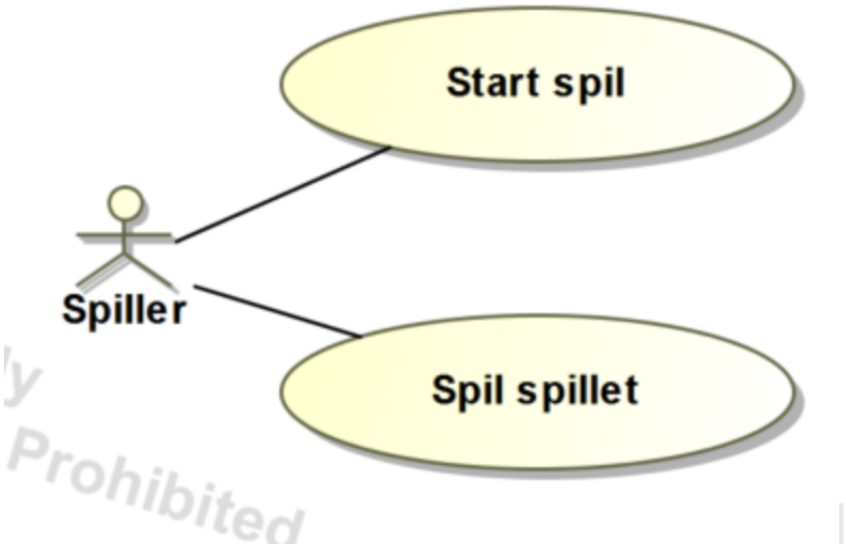
Non-funktionelle krav

- Systemet skal have en hurtig responstid.
- Systemet skal være let at bruge
- Programmet skal være let at vedligeholde.
- Programmet skal kunne køre på almindelige computere.
- Programmet skal nemt kunne oversættes til andre sprog
- Koden skal kunne genbruges[1]

[1] `https://en.wikipedia.org/wiki/Non-functional_requirement`

4 Analyse

4.1 Use Case Diagram



Her ses spillets to use cases.

4.2 Use Cases

U1 Start Spillet

- Ved start af spillet vælges hvor mange spillere der er, derefter enkeltvis vælger man sin brik og indtaster dit navn. Spillerne indtaster deres navn og vælger brik i rækkefølgen yngst til ældst. Efter dette modtager hver spiller 30000 kr.
  - U1A Spilleren der starter kaster terningerne og udfører sin tur.

U2 Spil Spillet

- Spilleren rykker deres bil frem til et felt baseret på et terningekast, alt efter hvilket felt der landes på, er der forskellige udfald.

4.3 Use Cases - fully dressed

U1 - Start Spil

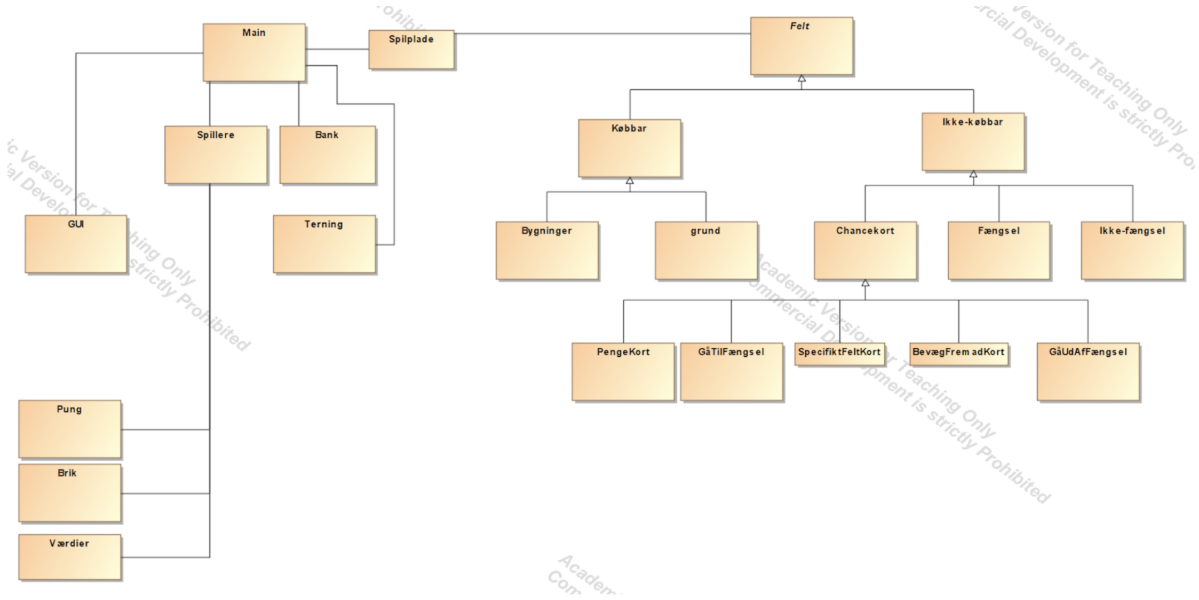
Use Case: Start Spillet	
ID	U1
Level	Spillet skal begynde og spilleren skal være parat til at starte U2
Primær aktør	En spiller
Stakeholders	IOOuterActive og spilleren
Precondition	Spilleren skal have åbnet programmet og være klar til at spille
Postcondition	Alle spillere indtaster navn.
Basic flow and extensions	1. Programmet starter 2. Spilleren bliver bedt om at vælge antal spillere 3. Spilleren bliver bedt om at indtaste sit navn 4. Spilleren indtaster sit navn 5. Spilleren bliver bedt om at vælge en brik. 6. Spilleren får 30000 kr hver.
Frequency of occurence	Dette er use casen som starter programmet, og vil derfor ske en gang pr. spiller.

U2 - Spil spil

Use Case: Spil Spillet	
ID	U2
Level	For hvert terningekast lander spilleren på et bestemt felt, og ud fra feltet
Primær aktør	Spilleren
Stakeholders	IOOuterActive og spilleren
Precondition	Spilleren skal have udført usecase U1
Postcondition	Når terningerne er slået lander spilleren på et felt, informationen fra dette felt gives til spilleren
Basic flow og extensions	1. Man slår med 2 seks sidede terningerne 2. Spilleren rykker det antal øjne frem ad på spilpladen 3. Spilleren lander på et felt • Spilleren lander på et ikke-ejet købbart felt – Hvis spilleren lander på et ejendomsfelt, kan man vælge at købe det eller sætte det på auktion. – Hvis man lander på en grund-felt, kan man vælge at købe det eller at sætte det på auktion

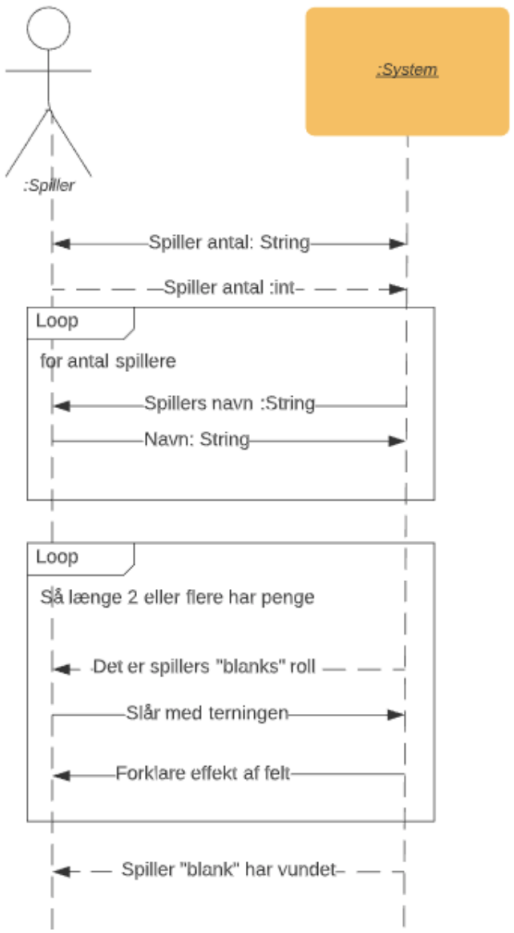
Use Case: Start Spillet	**Fortsat**
Basic flow and extensions	• Spilleren lander på et ejet købbart felt – Hvis det er et bryggeri, skal man betale med det antal af øjne man slår med en sekssidet terning og ganger ∗ Hvis ejeren af bryggeriet kun ejer et bryggeri, skal man gange antallet af øjne med 100. ∗ Hvis ejeren af bryggeriet kun ejer begge bryggeri, skal man gange antallet af øjne med 200. – Hvis det er et rederi, skal man betale et beløb udfra hvor mange af de 4 andre færgelinjer som den spiller der ejer feltet du landede på har. ∗ Et rederi 500 kr ∗ Et rederi 1000 kr ∗ Et rederi 2000 kr ∗ Et rederi 4000 kr – Hvis spilleren lander på et ikke købbart felt. ∗ Hvis spilleren lander på fængsletfeltet, sker det intet ∗ Hvis spilleren lander på parkeringsfeltet, sker der intet ∗ Hvis spilleren lander på prøv lykken, skal de trække et chancekort og følge det · Bevæg sig hen til et specifikt felt · Få penge · Giv penge til banlken eller de andre spillere · Gå-Ud-af-fængslet kort ∗ Hvis spilleren lander på Start eller passere start får de 4000 kr ∗ Hvis spilleren lander på Gå-i-fængsel feltet, skal de gå i fængsel, hvis de passerer start får de ikke 4000 kr. · Ved starten af spillerens næste tur, kan de betale 1000 kr for at komme ud · Ellers skal spilleren kaste med med 2 seksidet terninger og få to ens for at komme ud. · Når spilleren har kastet 3 gange, uden at få nogle ens må de ved næste tur, komme ud af fængslet. · De kan brug deres slip ud af fængslet kort, hvis de har det og slippe for fængslettid – I løbet af en spillers tur kan spillerens tur kan de altid vælge at pantsætte, købe hus eller opgraderer til hoteller. – En spiller løber tør for penge ∗ Kan vælge et pantsætte sine værdier ∗ Sælge sin Værdier ∗ Kan Ikke pantsætte eller sælg noget, går spilleren fallit og overdrever sin resten værdier til banken og hvis spilleren ikke kunne betale husleje, giver man dem til den spiller.
Frequency of occurrence	Det er programmet primære use case, og derfor vil den optræde altid ved brug af programmet, bortset fra når du starter programmet.

4.4 Domænemodel



Her ses et system sekvens diagram

4.5 System sekvensdiagrammer



Her ses et system sekvens diagram

5 Design

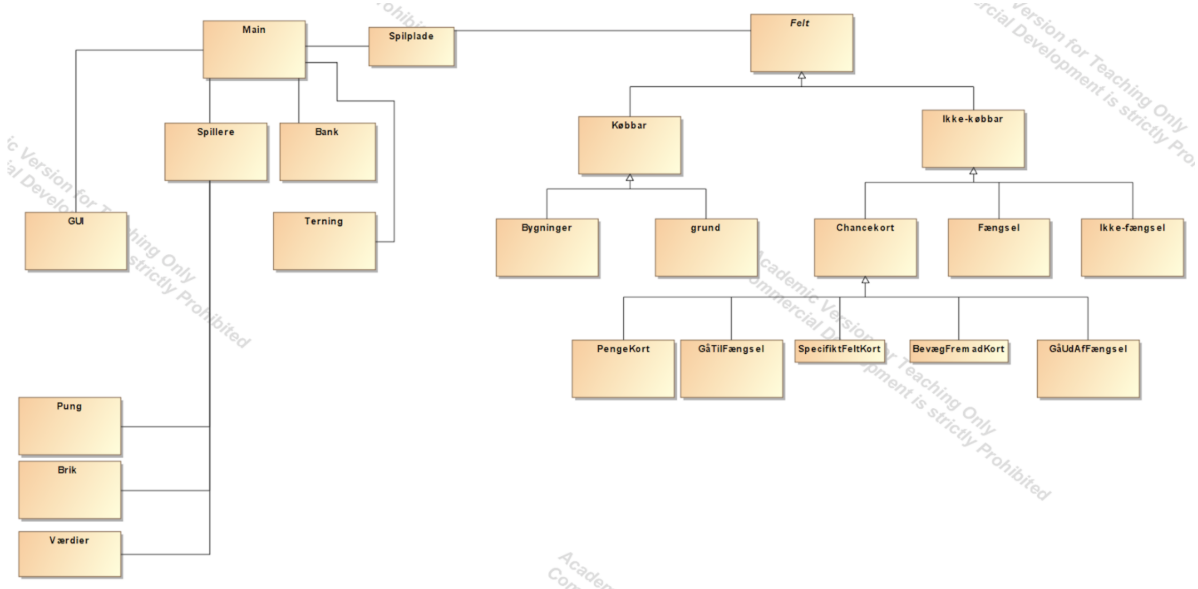
5.1 Design klasse diagram

5.2 Sekvens Diagram

6 Implementering

- Hvilke tanker har vi gjort os?
  - Vi har i projektet arbejdet efter Unified Process, og dermed arbejdet i iterationer. Vi startede vores projekt med at sætte os grundigt ind i reglerne for Matador, og dannede os et overblik over projektet.
- Opdeling af krav
  - Vi opstillede herefter en kravspecifikation til produktet ud fra spillets regler, via MoSCoW principperne. Herefter lavede vi en tidsplan ud fra disse, og fulgte denne. Dette gav et godt overblik over hvad vi skulle nå til hvornår, og herefter gav det god mening at lave vores Use Cases. Da vi havde et godt overblik over projektet, tegnede vi for hver iteration en opdateret version af vores UML diagrammer der hjalp os med at få overblik over vores spil.
- Arkitektoniske overvejelser
  - Efter vi havde gjort os nogle generelle overvejelser om arkitekturen på vores projekt, gik vi i gang med at få et Minimum Viable Product. Dette gjorde også at vi hurtigt kunne komme i gang med at løse eventuelle problemer med den umiddelbare arkitektur vi havde besluttet os for, men også at vi kunne få et produkt på plads med de mest basale features, så vi herefter kunne uddelegere arbejdet, uden at flere lavede det samme. Vi brugte overvejelser og nogle af de simple java-klasser fra de tidligere CDIO projekter til at få et produkt på benene hurtigst muligt.

Midlertidligt klassediagram



ud fra disse udarbejdede vi et midlertidigt klassediagram

7 Dokumentation

- Vi har forsøgt at overholde GRASP principperne
  - dvs. vi har f.eks lavet en player klasse som indeholder informationerne om spilleren og en playercontroller klasse der står for at beregne mm. omkring spilleren.
- information holders
  - hvilke klasser er det
- controllers
  - hvilke klasser er det
- creator
  - hvilke klasser er det

7.1 Lav kobling

- har vi lav kobling, hvis ja hvor

7.2 Samhørighed

- har vi lav samhørighed, hvis ja, hvor

7.3 Evt

- hvad ville vi havde gjort anderledes?

8 Test

8.1 Test cases

MovementController

Test Case ID	TC01
Summary	Test at spilleren bliver placeret korrekt på Gameboarded
Requirements Tested	R1, R4
Precondition	getFacevalue() skal havde returneret et terningekast
Postcondition	Spillet fortsætter
Test Procedure	Sammenligning af spillerplacering og forventet placering efter terningekast
Test Data	Metode test, data ikke nødvendigt
Expected Result	AssertEqual = true
Actual Result	True
Status	Passed
Tested By	Mikkel Thorsager
Date	14/1-2019
Test Environment	IntelliJ IDEA

addToBalance

Test Case ID	TC02
Summary	Test af spillerens balance, og om den opdaterer korrekt vha. deposit metoden
Requirements Tested	Betaling af div. køb
Precondition	Spiller modtager/mister penge
Postcondition	Spilleren har korrekt balance
Test Procedure	Sammenligning af forventet og aktuel balance
Test Data	Metode test, data ikke nødvendigt
Expected Result	True
Actual Result	True
Status	Passed
Tested By	Jonas Henriksen
Date	10/1-2019
Test Environment	IntelliJ IDEA

setOwner

Test Case ID	TC03
Summary	Test af setOwner
Requirements Tested	R10, R11, R12, R13, R14, R15, R16
Precondition	En spiller har købt en grund
Postcondition	Feltet har korrekt owner
Test Procedure	Sammenligning af aktuel og aktuel owner
Test Data	Metode test, data ikke nødvendigt
Expected Result	True
Actual Result	True
Status	Passed
Tested By	Jonas Henriksen
Date	10/1-2019
Test Environment	IntelliJ IDEA

9 Konfigurationsstyring

9.1 Krav til styresystem

OBS[2]

Software

- 64-bit Windows (10, 8, 7)
- MacOS 10.11 eller nyere
- Java Runtime Environment (JRE)
- Java 8 Development kit (JDK, denne pakke inkluderer JRE)
- IntelliJ IDE

Hardware

- 2 GB ram minimum, 8 GB anbefalet.
- 2.5 GB Harddisk plads, SSD harddisk anbefalet.
- 1024x768 minimum skærm opløsning.

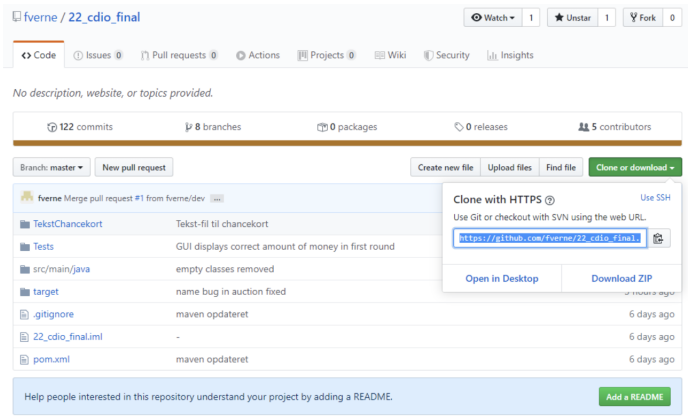
Installation af nødvendig software

1. Java 8 Windows - Mac
2. IntelliJ. (Installations vejledning: Windows - Mac) eller Download IntelliJ direkte

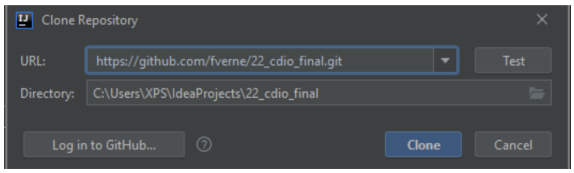
[2]Dele er genbrugt fra gruppe 20a CDIO2 opgave.

9.2 Import af projekt fra GitHub

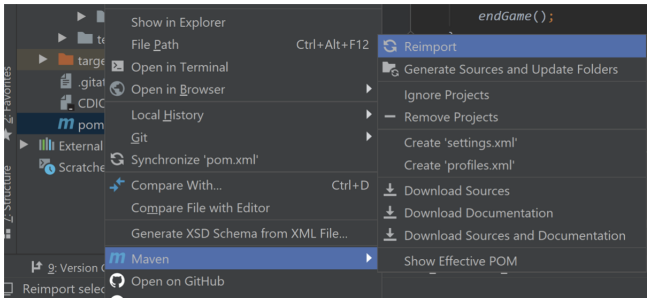
1. gå til linket `https://github.com/fverne/22_cdio_final.git`
2. Importer projektet ved at trykke på clone or download, kopiere linket i POP-UP menu



3. Nu åbner vi Intellij, og klik på Check out from version control, og derefter tryk på Git.



4. Når intelliJ åbner skal man finde pom.xml filen ude i menuen i venstre side, derefter skal man højreklikke og trykke reimport



5. klik derefter på "Kør"ikonet for at kører programmet

10 Reflektion

10.1 Projekt forløb

10.2 Konklusion

11 Bilag

11.1 Bilag A - Link til GitHub

`https://github.com/fverne/22_cdio_final`

11.2 Bilag B - M2

Opsummering:

Vi har på nuværende tidspunkt analyseret, designet og implementeret alle vores "Must have" og "Should have" krav, Bortset fra de sidste chance kort. I løbet af de sidste arbejde, altså den 15-1 til den 18-1 vil gå videre til vores "Could have" krav, samt de før omtalte chancekort samt at revurdere vores UML diagrammer til at passe til det program vi reelt har implementeret. De sidste dage vil vi også teste vores program igennem med unit- , bruger- og systemtest, samt at få færdiggjort vores rapport og finpudsning af programmet.

11.3 Implementering af MoSCoW Krav

Must have

Must have	R1	R2	R3	R4	R5	R6	R10	R27	R29
Implementeret	X	X	X	X	X	X	X		
Ej Implementeret								X	X
Bliver implementeret								X	X

- R27 - Fallit funktion, spillerne skal udgå hvis de løber tør for penge, høj prioritet
- R29 - Vinder funktion, høj prioritet

Could have

Must have	R8	R18	R19	R21	R25	R26	R28
Implementeret	X	X			X	X	
Ej Implementeret			X	X			X
Bliver implementeret			X	X			X

- R19 - Pantsætning af ejendomme, mellem prioritet
- R21 - Betal indkomst skat, mellem prioritet
- R28 - Salg af grunde mellem spillere, mellem prioritet

Should have

Must have	R7	R9	R11	R12	R13	R14	R15	R16	R17
Implementeret	X		X	X	X	X	X	X	X
Ej Implementeret		X							
Bliver implementeret									

R9 - Valg af brik type, lav prioritet

Afvigelse fra den originale arbejdsplan:

Da vi har været markant mere effektive end forventet samt at nogle blive færdig med deres tildelte krav hurtigere end de andre krav i iterationer, så de har de fået lov til at springe videre til den næste iterationer, og har senere så revurderet det de lavede i den forrige iteration. Så det har gjort vi nok er 2 dage foran vores originale tidsplan. Dog har vi stadigvæk nogenlunde overholdt vores originale arbejdsplan i forhold til hvornår vi har designet og implementeret kravene.

Features

Som sagt har vi på nuværende tidspunkt alle vores "must have" og "should have", og på nuværende tidspunkt ligner det ikke at vi vil skære nogle af de resterende features fra da vi estimerer at vi har masser af tid til at designe og implementerer dem. Dog kan vi godt risikere at vi igen er hurtigere end vi forventer og derved kan vi lave nogle seriøs finpudsning.

Ny Arbejdsplan 15-1 til 20-1

Dag	Opgaver	Beskrivelse
15/1	"Could have krav"	De resterende could have krav
16/1	"Could have krav"	De resterende could have krav
17/1	Test, rapportskrivning, UML, finpudsning af program, potentiel ekstra features	**Test:** Lave en grundig systemtest, unittest og brugertest **Rapport:** Lave indledning, lave implementation afsnit, lave konklusionen **UML:** Revurdere systemsekvensdiagram og designklassediagram.
18/1	Test, rapportskrivning, UML, finpudsning af program, potentiel ekstra features Start på Præsentation	**Test:** Lave en grundig systemtest, unittest og brugertest **Rapport:** Lave indledning, lave implementation afsnit, lave konklusionen **UML:** Revurdere systemsekvensdiagram og designklassediagram. **Præsentation:** Find de gyldne ord, lave powerpoint.
19/1-20/1		Ikke aftalt/vurderet endnu, formodentlig mere ekstra tid til finpudsning og forberedelse på eksamen

Gammel Arbejdsplan

Iteration	Indhold (Kravliste)	Beskrivelse
1 [6/1 - 9/1]	1, 2, 4, 5a	Overordnet opsættelse og simpel funktionalitet af spillebrættet og spillere.
2 [10/1 - 13/1]	3, 6, 7acd, 9b, 10,	Simple og centrale regler i matador (ejendomme, betale, chancekort)
3 [14/1 - 17/1]	5i, 7b, 11, 13acd,	Opføring af huse og hoteller, mulighed for fængsel.
4 [18/1- 20/1]	9a, 12, 13b, 14	Avancerede funktioner, som salg af ejendomme, pantsætning, auktion af ejendomme.