62531 02314 62532

Indledende programmering, udviklingsmetoder til IT-systemer, Versionsstyring og testmetoder

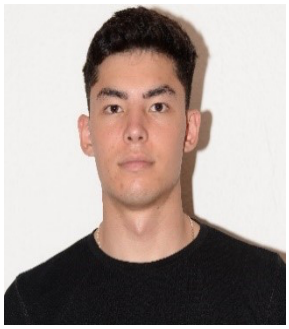
CDIO 1

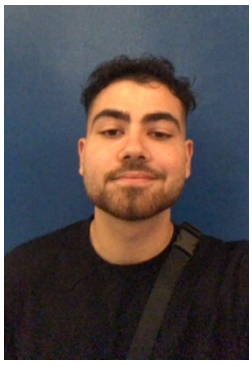
Gruppe 25

Mikkel Nørgaard
S224562

Johan Holmsteen
S224568

Alexander Szabo
S205793





Jason Yüksel
S224576

16. September 2022

DTU Diplom
Center for Diplomingeniøruddannelse

Resumé

Vi startede med at se på projektkravet og lå mærke til, at kravene ikke var godt nok specificeret for os. Vi stillede nogle spørgsmål, for at få afklaret, hvilke krav, vi skulle have til vores spil. Da vi fik afklaret kravene, gik vi i gang med use cases og sub use cases, for at forstå, hvordan vores spil skulle spilles. I og med at vi fik styr på vores use cases, og vores succesguarantee, gik vi videre med vores domænemodel, hvor vi blandt andet fik styr på vores aktører og vores stakeholders. Så fik vi lavet vores usecase modeller og til sidst fik vi lavet en risikovurdering af vores projekt, og da vi fik et overblik over de risici, vi havde i vores projekt og fik evalueret dem, kom vi frem til, at vi var klar til at lave vores spil. Vores terningspil er fuldstændig blevet fuldført efter de nødvendige krav, men de ekstra krav, som der var stillet til opgaven. Da vi så blev færdige med koden og spillet udførte vi en test på 1000 kast, og kom frem til, at vores spil, lever op til de nødvendige krav, og er et færdigt spil, som kan spilles efter hensigten

Timeregnskab

Samlet tidsregnskab: 32 timers arbejde samlet

Indholdsfortegnelse

1. Indledning

Et terningspil kan basalt set defineres som et spil, der benytter én eller flere terninger til at fremføre handlingen i det involverende spil. Vores opgave i dette forløb har været at programmere et sådan spil, som kan spilles i konsollen. Terningspillet foregår ved at der er to spillere, som på skift slår med to 6-sidet terninger. På hvert kast med de to terninger, ligger man antal øjne sammen og den første spiller til at opnå 40 point har vundet.

2. Projekt-planlægning

Vi burde have lavet noget projekt-planlægning inden vi gik i gang med CDIO 1. Desværre glemte vi dette punkt. Vi kommer til i fremtidige opgaver, at have mere fokus på projektplanlægning og sørger for, at have mere fokus på dette punkt, og sørger for at det er en større del, af vores næste projekt.

3. Krav/Analyse

Vi skulle få styr på de krav, som var til det spil, vi skulle lave. Vi fik en kravbeskrivelse, som vi mente ikke var fyldestgørende, da kravene ikke var specificeret på en måde, og nogle manglende beskrivelser. Efter vi stillede spørgsmålene til projektlederne, fik vi afklaret, de spørgsmål, som vi havde til kravene til spillet. Vi kom frem til, at disse var de fulde krav til spillet:

"Vi vil gerne have et system, der kan bruges på i databarerne på DTU, som skal kunne køre på Windows 11 og Java JDK 18. Det skal være et spil mellem 2 personer. Spillet skal gå ud på at man slår med et raflebæger med to terninger og ser resultatet med det samme. Spillerne skiftes efter hvert sl Summen af terningernes værdier lægges til ens point. Vinderen er den, der opnår 40 point, hvor der ikke tages højde for mange slag, hver spiller har haft. Hvis der er ressourcer til det, er der følgende ekstraopgaver:

1. *Spilleren mister alle sine point hvis spilleren slår to 1'ere.*
2. *Spilleren får en ekstra tur hvis spilleren slår to ens og det kan fortsætte uendeligt. Dette gælder også hvis spilleren slår to 1'ere*
3. *Spilleren kan vinde spillet ved at slå to 6'ere, hvis spilleren også i forrige kast slog to 6'ere.*
4. *Spilleren skal slå to ens for at vinde spillet, efter at man har opnået 40 point. Det vil sige, hvis du slår 2 ens, som sender dig over 40 point, skal du slå to ens igen. Hvis du dog slår to 1'ere, så mister spilleren alle sine point, selv hvis de har over 40 point.*

Vi forventer at alle, der ikke kan kode, alle som kan læse på det sprog vi vælger programmet til at være og alle der kan bruge en computer bør kunne spille dette spil.

4. Design

4.1 Use case

Vi mener at der kun er en usecase til spillet.
- Spil terningespillet

Dertil kommer der følgende subusecases:
Rul terning
Summere score

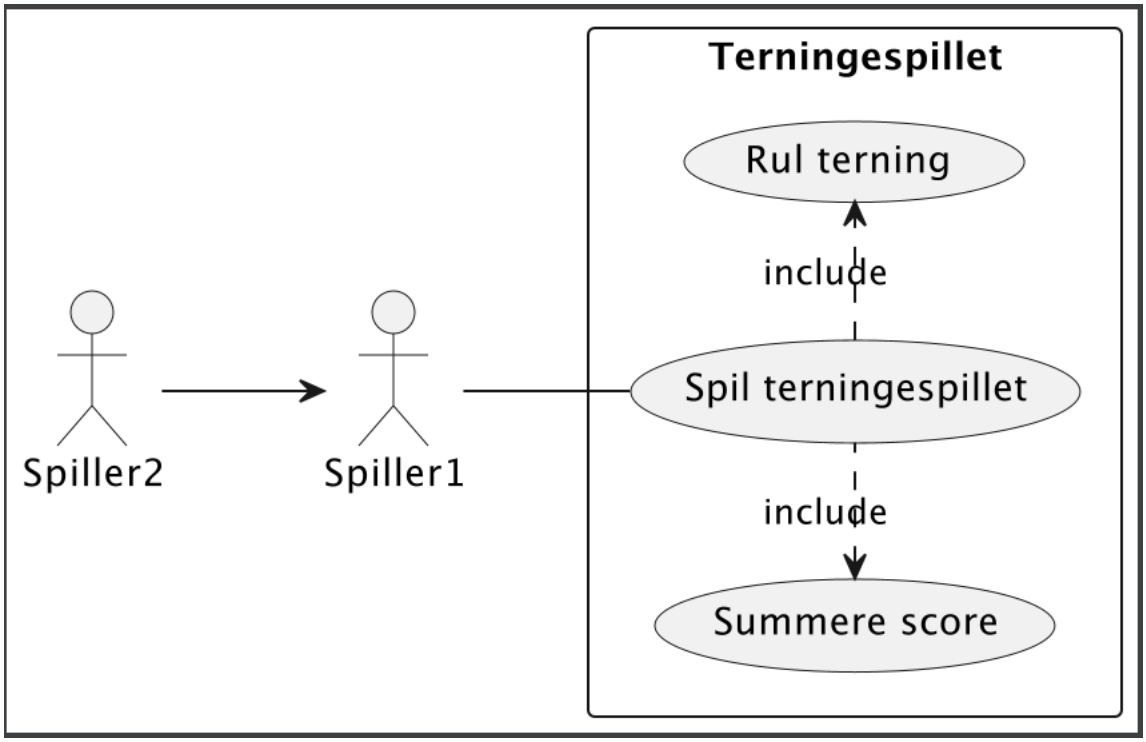
4.2 Beskrivelse af use cases

Spil terningespillet:
Spilleren ruller med terningerne og lægger de to tal sammen. Spilleren afslutter sin tur. Spilleren gør dette til de opfylder kravene for at vinde spillet.

Rul terning:
Spilleren ruller med 2 terninger, som viser et antal øjne som bliver brugt til at beregne score.

Summere score:
Spilleren lægger tallene fra terningerne som spilleren har rullet sammen, og lægger det til summen af spillerens tidligere kast.

4.3 Use case diagram:



Figur 4.1: Use case diagram

4.4 Success Guarantee

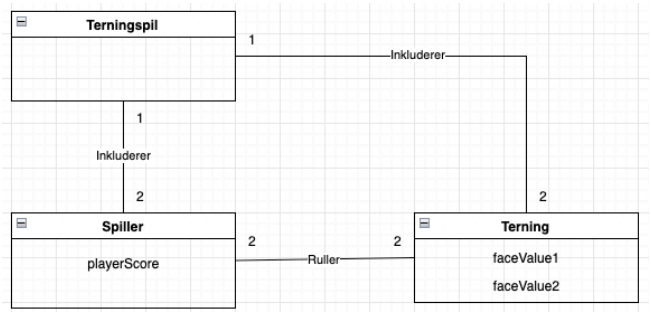
Spilleren har 40 point
Spilleren ruller to seksere to gange i streg - denne gør sig kun gældende såfremt vi når til ekstraopgaverne
Spilleren har 40 point eller derover og slår to ens, men ikke to et'ere - denne gør sig kun gældende såfremt vi når til ekstraopgaverne

4.5 Alternative Flow

Ingen alternative flows er mulige i vores spil:

4.6 Domæne

Objekter i domænet:
Terningespil
Terning x2
Spiller x2



Figur 4.2: Domænemodel

4.7 Vision

Vi vil gerne have et system, der kan bruges på maskinerne (Windows) i databarerne på DTU. Det skal være et spil mellem 2 personer. Spillet skal gå ud på at man slår med et raflebæger med to terninger og ser resultatet med det samme. Summen af terningernes værdier lægges til ens point. Vinderen er den, der opnår 40 point.

4.8 Aktører:

Spiller 1
Spiller 2
Det er de eneste aktører da det er de eneste der kan lave noget og spille spillet.

4.9 Stakeholders:

Spiller 1 og spiller 2. Det er dem der gerne vil have at spillet kører ordenligt.

4.10 Vores risikovurdering:

	Severity	Likelihood	Rating
Vi er ikke gode nok til at programmere	3	1	3
Integration med GUI går dårligt	1	3	3
Vi må ekskludere dele af projektet grundet tidsmangel	3	2	6
Vi har undervurderet størrelsen af projektet	3	1	3
Medlemmer forlader projektet	4	1	4
Vi når ikke at blive færdige til tiden	5	2	10

Figur 4.3: risikovurdering

Severity						
5	5	10	15	20	25	
4	4	8	12	16	20	
3	3	6	9	12	15	
2	2	4	6	8	10	
1	1	2	3	4	5	
	1	2	3	4	5	Likelihood
				20-25		Stop projektet
				12-19		Gør noget ASAP
				5-11		Hold øje
				1-4		Ignorer foreløbigt

Figur 4.4: Risikomatrix

5. Implementering

Se bilag, vi har ud fra vores domænemodel og vores use cases lavet kode som slår med terninger og spiller terningespillet.

6. Test

Vi har lavet to succesfulde tests, hvor vi har testet, at vores terning kun kan slå 1, 2, 3, 4, 5, og 6, hvilket er pointen, da vi selvfølgelig vil spille med en sekssidet terning.
Vi har også lavet en anden succesfuld test, hvor vi har kastet 1000 gange, hvor vi udregner gennemsnittet af øjnenes udfald, som altid rammer 3,5 (eller meget tæt på). Det viser at slår alle de 6 mulige slag lige meget, da gennemsnittet af de mulige kast er 3,5. Test-programmerne kan ses i vores GIT repository og i vores java filer; under mappen ”Test” har vi de to forskellige tests om vi har kørt på vores terninger.

7. Konklusion

Efter en kravanalyse og en masse design, fik vi kodet vores spil, som lever op til kriterierne. Vi har lavet vores terningspil, som fungerer, som det skal og vi har udført tests, som også viser at det fungerer, som det skal. Vi har lavet vores spil, som lever op til alle de nødvendige krav, men ikke de ekstra opgaver, som vi også kunnet have lavet. Men når vi ser på det spil vi har, som lever op til kravene, som består de tests vi laver, så har vi opnået at få lavet et meget tilfredsstillende spil.

8. Kilder

Vores terning (Die.java) er lavet med kraftig inspiration fra terningen i Java™ Software Solutions, Kapitel 4. De er næsten ens og har kun få ting der adskiller dem, primært dokumentationen deri.

9. Bilag

Source code ligger som javafiler i afleveringen.