Hold A Gruppe 25

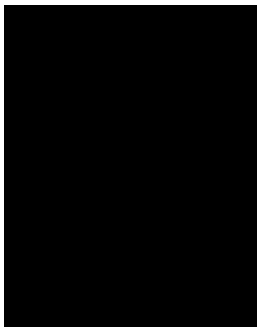
Fag:
02312, 02314, 02315

Afleveringsfrist 29-09-2017

Nicolas Vahman (s164404), Emil Vinkel (s175107) og Rasmus Hjorth (s175120)
Simon Boye(s175118) & Simone Zethof (s175110)







Timeregnskab

Del 1 - Gruppe 25

Timeregnskab **Ver. 2008-09-03**

Dato	**Deltager**	**Design**	**Impl.**	**Test**	**Dok.**	**Ialt**
11/10/2017	**Simon Boye**	**2**	**1**		**3**	**6**
	Nicolas Vahman	**2**	**3**		**1**	**6**
	Emil Vinkel		**3**	**1**	**2**	**6**
	Simone Zethof		**1**	**3**	**2**	**6**
	Rasmus Hjorth		**4**	**1**	**1**	**6**

Abstract

Denne rapport, har til formål at formidle processen i kreationen af vores CDIO-projekt. Vores projekt omhandler udviklingen af et terningspil, som vi skulle producere i fællesskab. Ydermere skulle projektet udvikles i Eclipse. Under udviklingen af programmet, skulle gruppen følge nogle specifikke krav, fra en "opdigtet" kunde. Yderligere var det ønsket, at programmet som minimum skulle opfylde en række specifikke krav. Til udviklingen af programmet, benyttede vi os af metoden Unified process (UP). Dette er en risiko-drevet samt use case baseret metode og der kan ses eksempler på benyttelse af unified process i opgaven. Desuden har vi benyttet os af Universal Modifiering language (UML). Opgaven byder altså på delelementer af vores unified process i form af UML diagrammer, samt test af programmet. Alt dette leder til vores samlede konklusion. For at uddybe dette, kan vi altså konkludere, at vi har overholdt de specifikke krav fra kunden og har udviklet et velfungerende program.

Indholdsfortegnelse

1.0 Indledning

I denne opgave har vi arbejdet med projektet CDIO 1, som omhandler et terningspil. Vi har udarbejdet en rapport som beskriver spillet, samt viser vores udviklingsmetoder, og til sidst vil vi vise vores test af programmet som er skrevet i opgavebeskrivelsen. Vi vil altså vise et sammenspil med fagene, udviklingsmetoder til IT-systemer, indledende programmering og versionsstyring og testmetoder.

Spillet i CDIO 1 er et terningspil, hvor to spillere skal kunne spille imod hinanden. Selve spillet er hvor 2 spillere skiftes til at slå med to terninger. Den spiller der først når summen 40 point vinder.

2.0 Analyse

2.1 Formålet

Formålet med denne rapport er at udarbejde et simpelt terningspil. Ved brug af forskellige analysemodeller, som vi er præsenteret for i undervisningen om UML, kan vi danne overblik over programmets kunnen og metoderne. Selve koden skrives på baggrund af undervisningen fra kurset "Indledende programmering".
Til sidst bruges vores viden fra "versionsstyring og testmetoder" til at teste programmet og dets funktioner.

2.2 Kravspecifikation:

For at forstå programmet der skal udvikles, er det vigtigt at opstille kravene der skal implementeres.

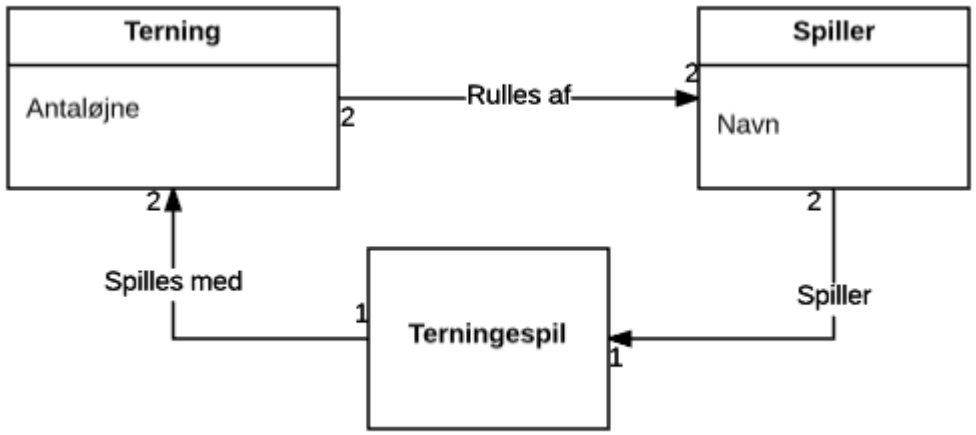
I opgavebeskrivelsen indgår 6 centrale krav, som vi har simplificeret:

Krav1: Spillet skal fungere på databarens computere. (Windows)
Krav2: Der skal være to spillere.
Krav3: Man skal have et raflebærger med to terninger.
Krav4: Resultatet (summen) af terningerne skal vises med det samme.
Krav5: Summen af terningerne skal lægges til ens samlede points.
Krav6: Spillet vindes ved at opnå 40 point.

Vi har valgt at indkludere henholdsvis en domæne-, Use case - og interaktionsmodel i vores videre analyse.

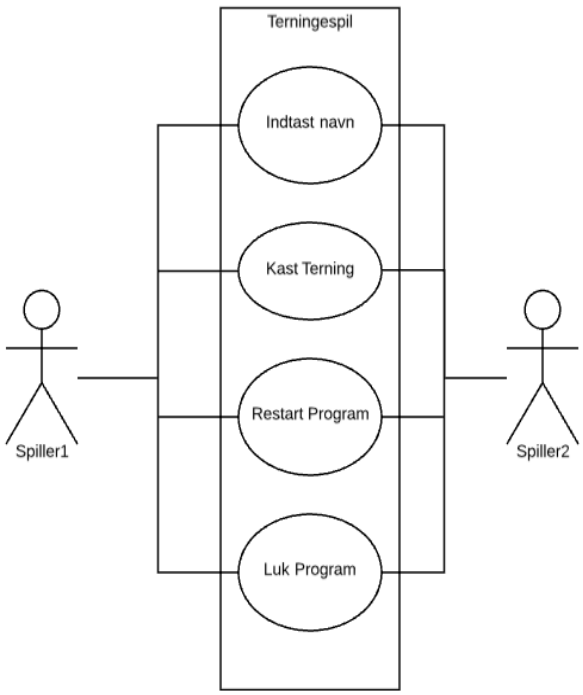
2.3 Domænemodel:

Domænemodellen er medtaget for at vise vores måde at opstille terningspillet med en "virkeligheds vinkel". Dette er med til at vise vores samlede forståelse for brugen af UML men også at vise et af de første trin til at, lave et program.
Vores domænemodel har klasser med, som indeholder attributter, i figur 1 herunder, kan man fx se at klassen Terning, har attributten, Antaløjne. Yderligere kan man se multiplicity(tal), der angiver at 1 Terningspil, spilles af 2 spillere, og at et Terningspil, spilles med 2 Terninger.



Figur 1

2.4 Usecase diagram



Figur 2

I vores spil er der to spillere som interagerer med systemet. De skal både indtaste navn og kaste terning. Derudover har de begge mulighed for at lukke programmet og kan efter endt spil genstarte spillet, hvis dette er ønsket. Her skal det dog tilføjes at spillet spilles på samme brugerflade, så spillerne er nødsaget til at skiftes til at tage deres tur.

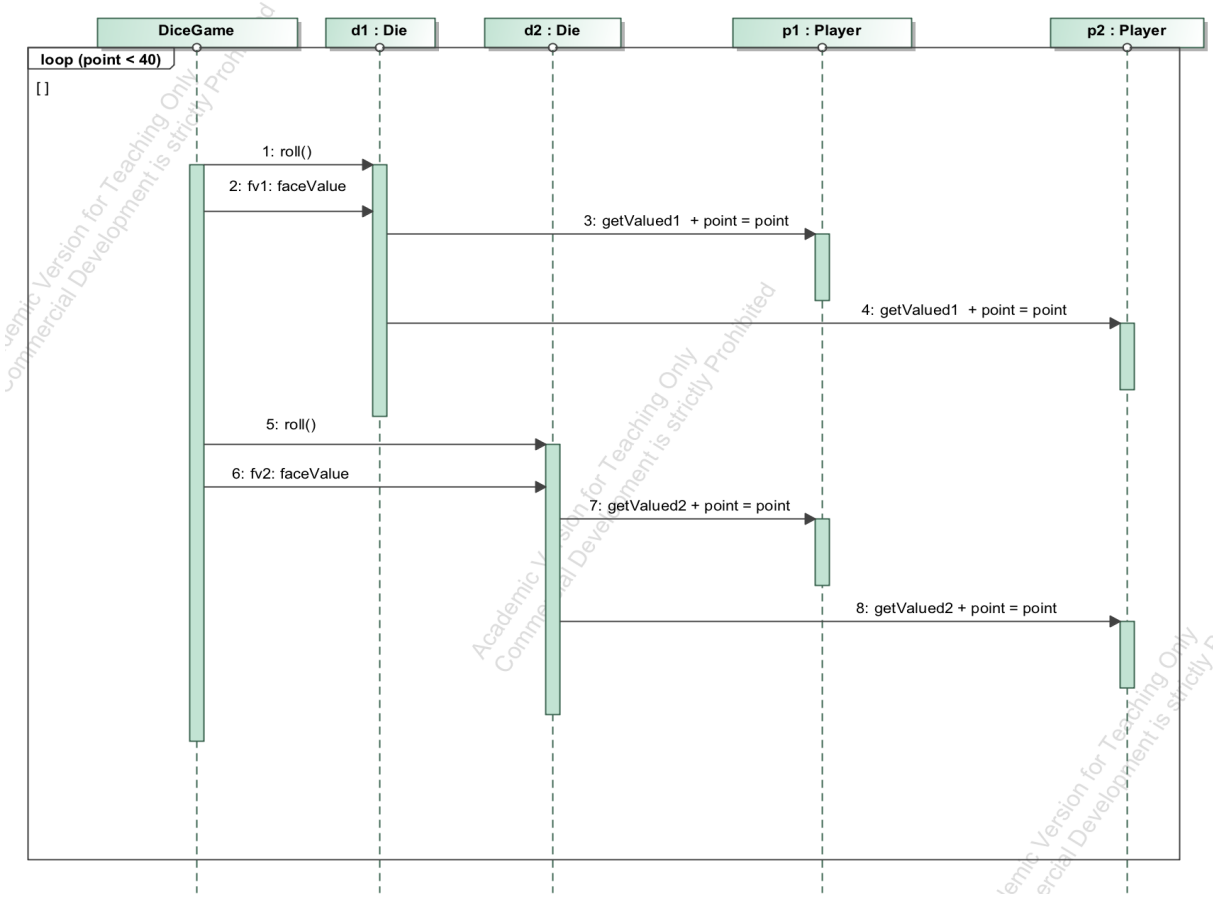
2.5 Usecase

Den nedenstående use case er et eksempel på en use case fra vores program. Den viser den vigtigste sekvens, hvor en bruger starter programmet og spiller spillet ved at kaste med terningen. Dette er med til at vise de enkelte trin som programmet skal opfylde, og vi kan derfor integrere den i vores videre arbejdsprocess, for at komme frem til det endelige produkt/spil.

Use case: kastTerning
ID: UC1
Brief description: Spilleren skal slå med et bæger som indeholder 2 terninger.
Primary actors: Spiller x 2
Secondary actors: terninger x 2
Preconditions: - Spillet skal være startet - Der skal være to spillere
Main flow: 1. *Use case starter når spiller1 kaster de to terninger* 2. *summen af antal øjne adderes til spiller1's point* 3. *Spiller2 kaster de to terninger* 4. *summen af antal øjne adderes til spiller2's point* 5. *…. 1-4 gentages indtil én af spillerne opnår de 40 point*
Postconditions: - *Point skal opdateres.* - *Der bliver udpeget en vinder ved point først til 40.*

2.6 Interaction model:

Vores interaktionsmodel, er det man kalder designmodel. Den er med i forlængelse af vores domænemodel. Design modellen er en viderebygning på domænemodellen. I design modellen begynder man nemlig at sætte metoder på, som vi også har gjort, og generelt er mere uddybende i forhold til et programmeringsmæssigt sprog. Det, der er værd at lægge mærke til i vores design, er vores implementering af et loop. Loopet viser at hele forløbet i modellen, vil gentage sig så længe summen af spillerens point er mindre end 40.



Figur 3

3.0 Implementering

Dette stadie indeholder vores forskellige overvejelser i forhold til koden i programmet. her vil vi uddybe vores overvejelser, samt implementering.

Indsættelse af knapper: Vi har valgt at indsætte forskellige knapper. I starten af spillet har vi implementeret en knap til at kaste terningerne. I Slutningen af programmet har vi implementeret en knap med teksten "Prøv igen?", der starter spillet forfra.

Indtastning af brugernavn: Vi har, når man begynder programmet, indsat et element, som gør det muligt for hver spiller at kunne skrive et brugernavn efter eget valg. Dette er med til at skabe brugervenlighed og skabe en bedre oplevelse for spilleren, at navnet popper op.

4.0 Test

4.1 Sum

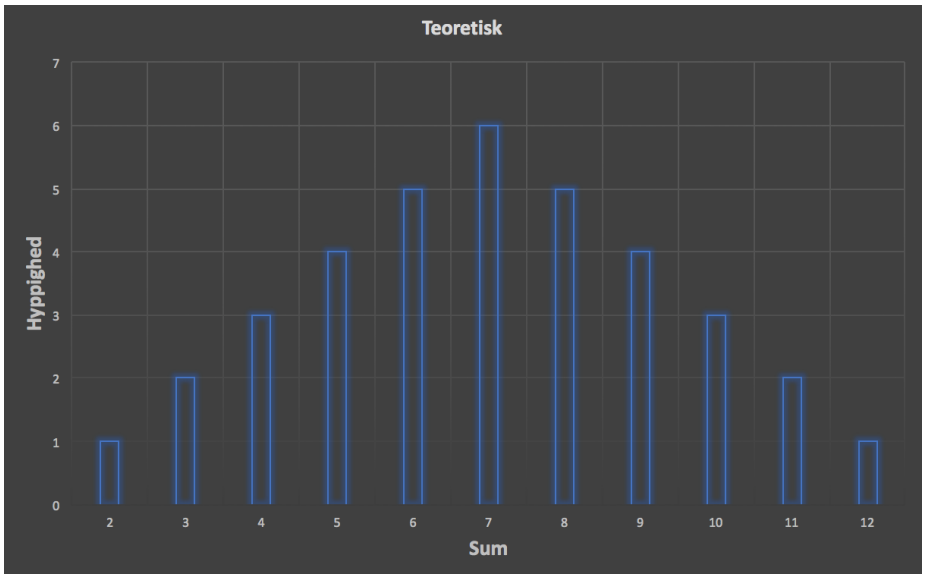
Det første der testes for, er antallet af forekomster for hver mulig værdi af summen (2-12).
Denne test kan ses i filen TestSpil.java

	Teoretisk	
sum	*Kombinationsmuligheder*	*Frekvens*
2	1	0,03
3	2	0,06
4	3	0,08
5	4	0,11
6	5	0,14
7	6	0,17
8	5	0,14
9	4	0,11
10	3	0,08
11	2	0,06
12	1	0,03
	36	

Maks antal kombinationsmuligheder for to terninger er i alt = 6·6 = 36

De teoretiske værdier for antal kombinationsmuligheder og frekvensen beregnes for hvert udfald af summen (2-12).
Se tabel til venstre

Fx. er der kun én kombinationsmulighed for at få en sum på 2, nemlig at begge terninger viser en værdi på 1. Samme sandsynlighed for at summen bliver 12, er den samme som summen på 2, da begge terninger skal vise 6'ere. Sådan følges sandsynligheden ad helt op til summen på 7. Her topper kombinationsmuligheder, og her er sandsynligheden derfor også størst. *(se figur 4)*

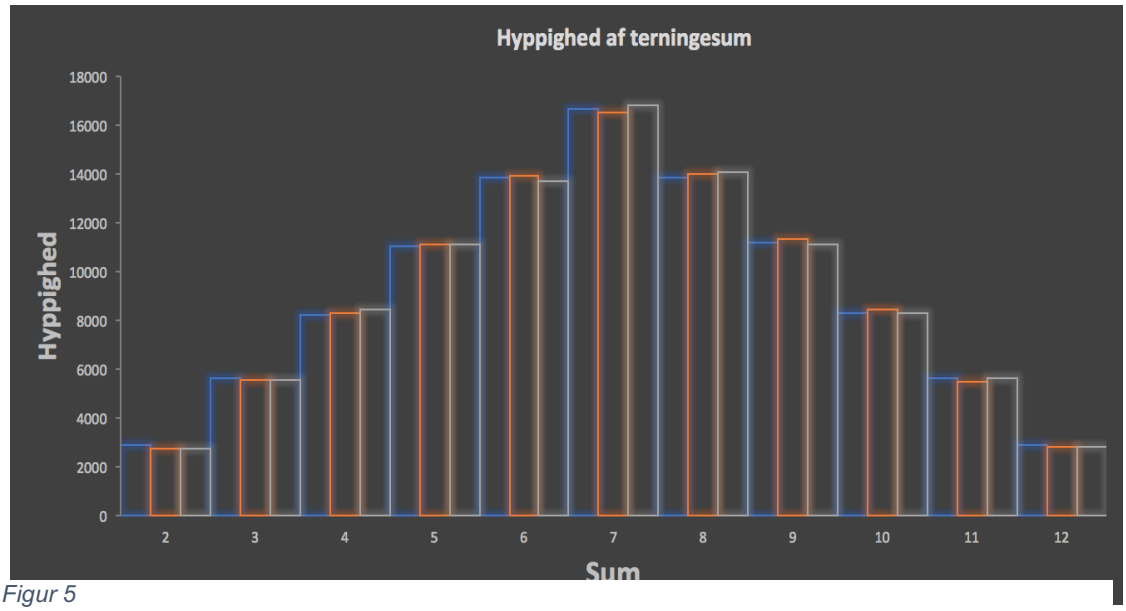


Figur 4. Grafisk illustration over den teoretiske fordeling af hyppigheden/kombinationsmuligheder.

Resultatet af testen på metoden getSum( ) er afbildet i *tabel 1.2*
Der er testet med 100.000 rul af terningerne over tre test, for at have et mere præcist grundlag for fordelingen af summen.

	Hyppighed			Frekvens		
Sum	*Test1*	*Test2*	*Test3*	*Test1*	*Test2*	*Test3*
2	2873	2741	2716	0,02873	0,02741	0,02716
3	5637	5546	5558	0,05637	0,05546	0,05558
4	8218	8285	8389	0,08218	0,08285	0,08389
5	11043	11118	11053	0,11043	0,11118	0,11053
6	13859	13898	13681	0,13859	0,13898	0,13681
7	16619	16503	16771	0,16619	0,16503	0,16771
8	13844	13952	14047	0,13844	0,13952	0,14047
9	11154	11284	11097	0,11154	0,11284	0,11097
10	8268	8424	8305	0,08268	0,08424	0,08305
11	5624	5477	5606	0,05624	0,05477	0,05606
12	2861	2772	2777	0,02861	0,02772	0,02777

Allerede ved at kigge på dataene fra testen, ses det at frekvenserne følger de teoretiske. Det ses også at frekvensen er højest ved summen 7.
For at understøtte dette er der yderligere lavet en grafisk illustration af fordelingen af hyppigheden på de tre test.



Figur 5

Det ses at denne illustration passer rigtig godt med den teoretiske, og derfor vurderer vi, at vores metode getSum() må være tilstrækkelig tæt på et rigtigt terningkast.

4.2 To ens

Programmet testes også for sandsynligheden af, at de to terninger viser ens antal øjne.
Den teoretiske sandsynlighed for at dette forekommer, er 6/36 dvs 0,1667 altså 16,67%.

Testen af terningespillet giver et udfald på henholdsvis 16.629,16.544 og 16.864 ved 100.000 terningeslag. Dvs. en procent på 16,63%, 16,54% og 16,86%. Dette må siges at ligge tæt på de teoretiske 16,67%. Vi kan derfor konkludere at programmets udfald følger de teoretiske sandsynligheder.

Testen kunne yderligere udvides ved at teste udsvinget af vinder og taber.
Det forventes at programmet er neutralt, og derfor at sandsynligheden for at vinde, er nær de 50% for både spiller 1 og spiller 2.
Dog kunne det også tænkes, at det er lettest for spiller 1 at vinde, idet han får lov til at starte, og derfor hele tiden er en runde foran. Her kunne man udvide kravspecifikationerne, så de lød, at når spiller 1 opnår 40 point, får spiller 2 stadig mulighed for at kaste. Herved kan man udvælge en vælger med flest point.

5.0 Projektplanlægning

Vi har løbende arbejdet med projektet, før og efter undervisning, frem mod deadline. Herudover har vi gjort en del ud af, at inkludere alle igennem hele processen. Derfor har vi primært arbejdet samlet i gruppen, så vi således kunne assistere hinanden.
Yderligere har vi i henhold til planlægningen af projektet, arbejdet det ud fra Unified Process. Vi har været igennem en idéfase, debatteret omkring muligt indhold (Inception) og tegnet diagrammer og startet implementationen af kode (Elaboration), lavet den primære implementation (Construction) og afslutningsvis testet vores program (Transition).

Vi har med henhold til projektet arbejdet løbende til afslutningen af forløbet. Vi har desuden ikke arbejdet alene/hver for sig på projektet. Men udelukkede mødt i gruppen.

6.0 Diskussion

I følgende afsnit vil de krav som er angivet i opgavebeskrivelsen blive diskuteret.
I opgavebeskrivelsen havde vi følgende krav:

Krav 1: Spillet skal fungere på databarens computere. (Windows)

Krav 2: Der skal være to spillere.

Krav 3: Man skal have et raflebærger med to terninger.

Krav 4: Resultatet (summen) af terningerne skal vises med det samme.

Krav 5: Summen af terningerne skal lægges til ens samlede points.

Krav 6: Spillet vindes ved at opnå 40 point.

Alle kravene skulle indgå i programmet. I vores rapport har vi implementeret de efterspurgte krav.

Yderligere har vi i testfasen beskæftiget os med hvorvidt programmet kan udføre de stillede krav. Dette har vi (i første omgang) gjort ved, at køre spillet igennem. Hvis de efter testen opnår status - succes, så er kravet korrekt implementeret. Efter at have kørt alle testene igennem, har de stillede krav alle opnået status - succes.

Nedenstående viser hvad vi har gjort for at løse kravspecifikationerne

Krav 1: Vi har arbejdet både på PC (Windows OS) og Mac (OS X), og vores program fungere på begge styresystemer, således kan vi konkludere at krav 1 er opfyldt.
Krav 2: Vi har opfyldt dette krav ved at implementere to løbende summer for henholdsvis en spiller 1 og en spiller 2, der opdateres uafhængigt af hinanden.
Krav 3: Vi har opfyldt dette krav, da spilleren ved tryk på en knap på vores GUI, har mulighed for at kaste 2 terninger, dog har vi ikke haft mulighed for at implementere et raflebærger.
Krav 4: Vi har opfyldt dette krav, efter at spiller "X" har slået, vil man øjeblikket få vist summen af terningerne.
Krav 5: Vi har opfyldt dette krav, efter hver enkelt spiller har slået, bliver de respektive point lagt sammen, og vist det samlede antal point spilleren har opnået.
Krav 6: Når første af de to spillere når 40 point, stoppes spillet og den spiller som har opnået 40 point har vundet.

7.0 Konklusion

Vi kan ud fra vores test af de stillede krav i opgavebeskrivelsen konkludere følgende:

Krav1: Succes
Krav2: Succes
Krav3: Succes
Krav4: Succes
Krav5: Succes
Krav6: Succes

Dermed kan det så siges, at alle de krav som er stillet er blevet opfyldt.

Der er testet antallet af forekomster for hver mulig værdi af summen (2-12), og her kan det konkluderes at der er størst sandsynlighed for at få en sum på 7 og mindst på 2 eller 12.

Desuden har vi lavet en test hvor vi har optalt antallet af kast, hvor terningerne er ens, dvs. den mindst mulige sandsynlighed fra forrige test.
Her kan det konkluderes at den samlede sandsynlighed for at slå to ens i vores spil, er tilnærmelsesvis den teoretiske.

Af de nævnte test, kan der konkluderes at vores terninger i spillet stemmer overens med de teoretiske sandsynligheder for udfaldene.

Gruppearbejdet kan vi konkludere har fungeret rigtig godt. Vi har både formået at arbejde seriøst, samt hygge os omkring projektet. Hvert gruppemedlem har været til stede til hver samling og alle har bidraget med seriøse inputs.

10.0 Glossary

UML (Unified Modeling Language):

En universel standard for udseendet af diagrammer hvormed man kan beskrive strukture og forløb i objekt orienteret programmering.

UP(Unified process):

En process til udvikling af objekt orienteret software. Den er iterativ, man arbejder i iterationer, faserne Inception, Elaboration, Construction og Transition. Den er arkitektur centreret, use case og risk driven. Yderligere er den inkrementel, dvs man får i hver iteration en udvidelse af det færdige system.

Domænemodel:

En model der repræsentere koncepter fra den virkelige verden.

Use Case Diagram:
Den simpleste form for diagram over en brugers interaktion med et system. Et sådant diagram kan være med til at identificere de forskellige bruger af systemet.

Use Case:
En sekvens af handlinger (og variationer), som et system kan udføre, der giver et målbart resultat som har værdi for en der interagere med systemet.

Interaction Model/Design:
Viser interaktionen mellem de involverede klasser og interfaces.

GUI (Graphical User Interface):
En grafisk brugerflade, der gøre det lettere for en person at interagere med systemet.

Multiplicity:
Et tal der repræsenterer hvor mange elementer/objekter på et UML diagram der interagere med hinanden. Det er muligt at der ikke er nogen øvre grænse for interaktionen, det annoteres ved *.

11.0 Litteratur- og kildefortegnelse

Bøger:
"*Applying UML and patterns***"**
- 3rd Edition published 2004
Author: Craig Larman
ISBN-13:9780131489066

"Java Software Solutions"
8th Edition
Authors: John Lewis, William Loftus

Hjemmesider:
https://en.wikipedia.org/wiki/Glossary_of_Unified_Modeling_Language_terms
Sources: Fowler, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (3rd ed.). Addison-Wesley. ISBN 0-321-19368-7.
Tom, Pender (2003). *UML Bible*. John Wiley & Sons. ISBN 0-7645-2604-9.