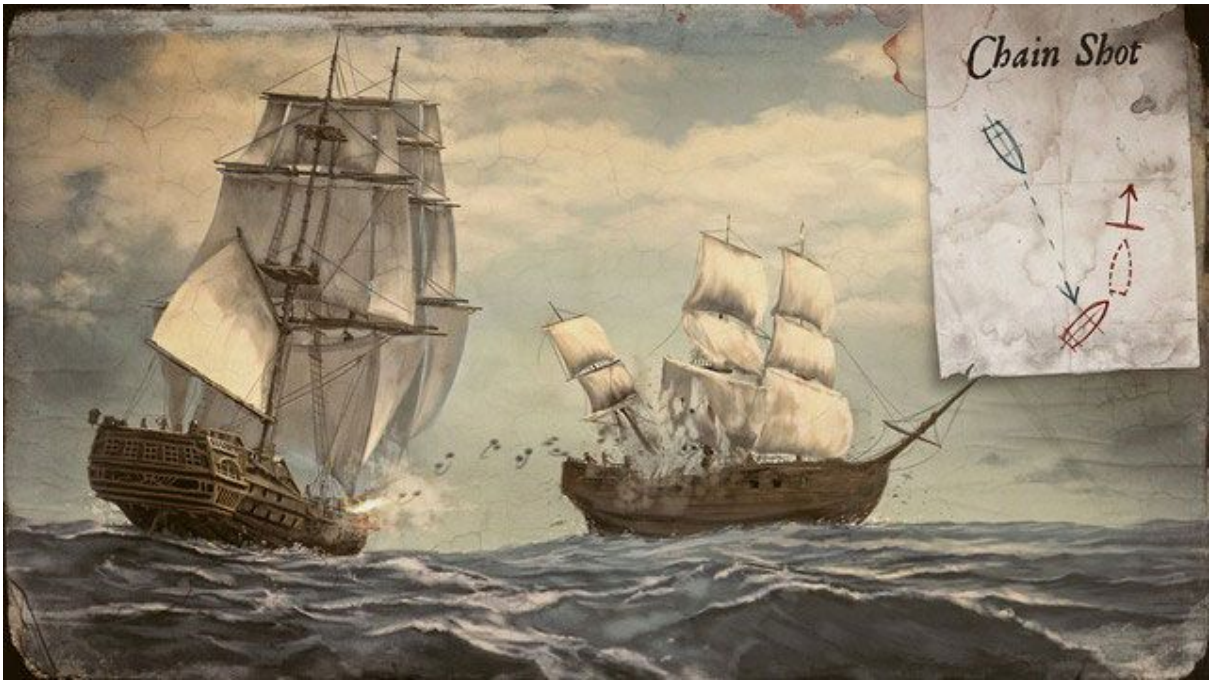
Modulopgave 3 - Ships & Sails

Frederik Lundbeck Jørgensen, Marcus Bender Knudsen,
Nichlas Birkehøj List, Søren Holmgrün Carlsen

KEA
DAT18C

Teachers: Asger, Jarl, David



Abstract

This report will take you, the reader, through all the design and implementation considerations we had from the beginning of the project all the way to the end. Simply put, the project itself is to build the game Ships & Sails and document all the design and implementations considerations we had throughout the project. This is done utilizing UP and iterating through UP's 4 phases showcasing essential use cases, diagrams and more. As we came closer and closer to the deadline we realized we wouldn't be able to complete the entire game. The requirements section along with the limitations section will go through what we managed to finish and what is left for future aspects. In conclusion we derive that the short timeframe vs the size of the project just didn't add up and thus ending the project with an incomplete game.

Indholdsfortegnelse

Abstract 1

Indholdsfortegnelse 2

Indledning 3

Ordliste 3

Fase 1 - Inception 4

Kravspecifikation 4

Risikoanalyse 6

Afgrænsning 6

FURPS+ 7

Fase 2 - Elaboration 8

Use Cases 8

UC#1: Opret spil 8

UC#2: Tilknyt spil 8

UC#3: Ammunition 8

UC#4: Skibskollision 9

UC#5: Ildkamp 9

UC#6: Tur 9

UC#7: Skibe 10

UC#8: Reload 10

UC#9: Bevægelse 11

UC#10: Map/Grid 12

Use Case Diagram 13

Domæne model 14

ERD Diagram 15

System Sequence Diagram 16

System Diagram 17

Class Diagram 18

Fase 3 - Construction 19

Konklusion 19

Fase 4 - Transition 19

Future Aspects 19

Indledning

Denne rapport gennemgår design og implementerings overvejelserne til modul opgaven Ships & Sails. Opgaven i sig selv består i at designe og implementere et turbaseret brætspil i programmeringssproget Java med Spring som framework til en brugergrænseflade samt MySQL til databasehåndtering. I design delen benytter vi arbejdsmetoden UP, til at danne et overblik over hvad spillet skal kunne og hvordan vi kommer i mål med struktureringen af koden. I rapporten er det beskrevet med hhv. diagrammer, use cases, kravspecifikation afgræsning osv. hvordan vi bevæger os igennem de 4 faser. Slutmålet for projektet er et komplet spil der via TCP/IP opkobling både kan fungere som både server og klient.

Ordliste

Dette er en ordliste af termer, som optræder en eller flere gange i rapporten som umiddelbart kan være forkortelser eller kræve teknisk forståelse på forhånd. De bliver her uddybet.

SOTL	Ship Of The Line - Skibstype.
MAW	Man At War - Skibstype.
Brig	Brigade - Skibstype.
Grape shot	Ammunition der er lavet til at ramme mandskab på skibet.
Chain shot	Ammunition der er lavet til at ramme masterne på skibet.
Canon shot	Ammunition der er lavet til at ramme skroget på skibet.
Grid	Et grid er en samling af hexagoner der tilsammen udgør det søkort hvorpå skibe kan manøvrere.
Crit-skade	En form for skade der er kritisk i den forstand, at den skader mere end normalt.
TCP/IP	En forbindelses protokol til at snakke mellem server og klient.
Hull	Skroget/Livet på vores skibe
GUI	Graphical User Interface - grafisk brugergrænseflade

Fase 1 - Inception

Kravspecifikation

Projektet skal indeholde og være bygget op omkring hele UP pakken. Det skal implementeres ved hjælp af Java, Spring og inter-computer kommunikation (TCP/IP opkobling). Yderligere har projektet et krav til inddeling af kode elementerne i projektet. Hvert medlem af gruppen skal skrive og stå til ansvar for en deres andel af koden.

Marcus (Gun Control)
Denne del står for alt der har med kanonerne og skade uddeling at gøre. F.eks. når en spiller vælger at skyde mod et andet skib skal der udregnes hvor meget skade der bliver uddelt til de forskellige tiles.

Nicklas (Movement)
Movement skal stå for at holde styr på at rykke skibene og sørge for at et træk overholder de regler som er givet. Samt skal der udvikles alle de klasser der har med movement eller koordination at gøre f.eks: Coordinate.java som holder på et x og y koordinat. Direction som holder på en retning osv.

Frederik (Communication)
Denne del skal håndtere alt kommunikation der kan opstå mellem spillerne. F.eks når man skal tilslutte sig en server, skal det håndteres med sockets. Samt skal der laves en kommunikations protokol så de to programmer kan sende information om deres træk til hinanden osv.

Søren (UI)
UI delen består mest af javascript kode og html samt css. Desuden skal dette kobles sammen med nogle metoder fra vores HomeController klasse. Grafikken til spillet er også en del af ansvaret.

Til slut skal spillet køres ud fra et regelsæt som vi vil forsøge at overskueliggøre i nedenstående punktform.

- Alle deltagere spiller deres tur på samme tid.
- Spillerens skibe bevæger sig på et givent scenarie med specificerede startpositioner.
- Alle skibe har en orientering (**N, NE, NW, S, SE, SW**). Hvilket har indflydelse på hvordan skibene kan bevæge sig.
- Spillebrættet består af hexagoner og dertil findes 3 scenarier med hhv. **12x12, 24x24 og 32x32** tiles.

- Spillebrættet indeholder vind der har en orientering men ingen styrke. Vindretningen påvirker skibenes bevægelse. Vindretningen (**N, NE, NW, S, SE, SW**).
- Alle skibe har unikke fordele og ulemper (se skibs specifikationen nedenfor)
- Hvis et skib fra begge spillere ender på samme felt, vil spillet udregne skade på de givne skibe efter en kollision formular.
- Udskiftning af ammunition koster en tur og der er 3 typer af ammunition
    - **Cannon ball** - kort til lang rækkevidde. Skaden påføres skibet skrog. Et Critical Hit kan springe hele skibet i luften.
    - **Chain** - Middel rækkevidde. Skaden påføres på skibets rigning.
    - **Grape shot** - Kort rækkevidde. Skaden påføres skibets mandskab.
- Genladning af kanoner koster en tur
- Et tur består af følgende punkter (disse kan kombineres efter spillerens eget ønske og er ikke ensbetydende med at **alle** funktionerne skal udføres for en hver tur)

Bevægelse af skibene:
Alle skibe kan bevæge sig ud fra en regelsæt der tilhører hvert skib.
Skud placering:
Spilleren vælger et felt på mappet og en skadeformular udregner skaden på modstanderes skib, såfremt et af modstanderes skib rammes.
Genladning af kanoner:
Genladning af kanonerne på et givent skib koster en tur. Dvs. kanonerne på det skib der genlader ikke kan skyde før næste tur.
Udskiftning af ammunitionstype:
Udskiftning af ammunitionstype på et givent skib koster en tur. Dvs. skibet der har udskiftet ammunition først kan skyde på næste tur.

Skribsspecifikation

Type (navn)	Antal kanondæk	Antal kanoner pr. dæk	Antal sømænd til kanon kontrol	Antal sejl	Antal sømænd til sejl kontrol
Brigade (Brig)	1	8	24	4	24
Ship of the Line (SoTL)	2	16	48	10	60
Man at War (MaW)	3	28	84	24	144

Type (navn)	Antal kanondæk	Max antal sømænd	Max hull	Max fart	Max fart ændring	Max sejl	Antal orientation skift
Brigade (Brig)	1	60	25	2	1	30	1
Ship of the Line (SoTL)	2	160	60	5	2	60	2
Man at War (MaW)	3	340	140	4	1	80	1

Risikoanalyse

Risiko	Årsag	Sandsynlighed	Konsekvens	Risikotal	Handleplan
Sygdom		1	10	10	Gruppekontrakt
Dårlig kommunikation	Tingene bliver diskuteret for meget i gruppen. Da vi mangler konkret information omkring opgaven.	3	7	21	En projektleder som vil kunne tage det afsluttende valg, og få gruppen til at arbejde videre derfra.
Kort tidsramme	Et for kort semester.	2	6	12	Mere konkret viden i f.ek.s spring.
Manglende evner	For lidt undervisning i følgende emner.	3	10	30	Dele opgaven op, så hver elev i gruppen har sit ansvarsområde og kan "nørde" sit emne.
Ikke når deadline		3	4	12	Afgrænsning

Afgrænsning

Hovedparten af projektets afgrænsning består i problematikken omkring projektstørrelse vs. timeframe. I følgeskab med en obskur projektbeskrivelse, samt en mangelfuld protokol for client/server kommunikationen. Giver dette anledning til en markant afgrænsning af Construction fasen. Størstedelen af projektforløbet er gået med diskussioner, omkring hvorvidt vi forstod reglerne og omstændighederne for spillets gang og dets udvikling. Ovenstående, i forbindelse med risikoanalysen, bringer os frem til følgende afgræsning.

- Spillets logik. Grundet mange misforståelser vedrørende spillets regelsæt. Har vi ikke færdiggjort logikken der styrer de fundamentale elementer i spillet.
- Beregnings formularer på skade og fuldtræffere: Vi har ikke nået at implementerer dette da vi ikke kunne få det til hænge sammen uden nogen GUI.

- Brugergrænsefladen. Vi mangler de fundamentale kundskaber i hhv. Spring, Thymeleaf til at kunne færdiggøre et fungerende GUI.
- Server Klient kommunikation. Mangel på kommunikations protokol har været en stor hindring i færdiggørelsen af kommunikationen.

Med afgrænsningen taget i betragtning vil vi forsøge at implementere koden på en generisk måde så vi i fremtiden kan tilføje de manglende elementer.

FURPS+

Functionality	Programmet skal i starten prompte brugeren om man vil tilslutte sig eller lave et nyt spil. Under spillet skal programmet kunne modtage, samt sende data til den anden socket (TCP/IP). Disse data er baseret på en bestemt kommunikations-protokol, så vi er sikre på hvad vi sender og modtager. Derudover skal programmet præsentere en grafisk brugergrænseflade (GUI) der illustrerer spillets gang, mellem skibene og de valg man kan træffe.
Usability	Et af vores mål er at lave et intuitivt design så brugeren nemt kan bruge vores spil. Vi bruger bootstraps CSS stylesheet til vores HTML-elementer. Dette giver forhåbentlig brugeren en følelse, af en mere moderne og interessant side at interagere med.
Reliability	Spillets kørsel er afhængig af en TCP/IP forbindelse til server eller klient, altså et andet menneske i den anden ende. I forbindelse med dette, vil der altid være uundgåelige problemer. Et af disse kunne f.eks være at en spiller vælger at lukke forbindelsen/spillet. Selve spillets kode og dokumentation vil være uploadet til frit brug på en github side, tilgængelig for alle der kunne have interesse. Der vil derfor ikke gå noget tabt hvis en person vælger eller kommer til at slette filerne fra computeren.
Performance	Programmet skal udelukke eller ihvertfald vise spiller responstid, så man har en idé om forbindelses-kvaliteten til sin modstander. Den grafiske brugergrænseflade er i 2D og vi bruger ikke nogen krævende grafiske renderinger, så vi regner med alle computere der kan køre HTML5 også kan bruge vores program.
Supportability	Programmet kræver brugeren har et Java runtime environment (JRE) på computeren og en browser (Explorer, Opera, Chrome etc) installeret. Derudover skal computeren kunne køre en

	moderne browser med support til HTML5.
Licens	Vores software har en MIT licens. Licensen blev udarbejdet og oprettet parallelt med vores GitHub mappe. MIT Licensen gør softwaren frit tilgængelig for andre interesserede, det indebærer at de kan “bruge, kopiere, ændre, distribuere og sælge” vores kode uden nogle former for konsekvens.

Fase 2 - Elaboration

Use Cases

UC#1: Opret spil

Vi skal kunne oprette et spil på vores egen server. Serveren skal kunne oprette connection til en klient. Forbindelses protokollen bliver brugt af TCP/IP.

UC#2: Tilknyt spil

Precondition: En player har hosted et spil og det er tilgængeligt via. TCP/IP.
Vi skal kunne tilknytte os til et game på en anden server og udfordre en anden spiller.

UC#3: Ammunition

Scope	Spillerne og spil applikationen
Level	Ammunition bliver udskiftet og det koster en tur.
Primary actor	Spilleren
Stakeholder	Spillerne fordi de “spiller” samt os, fordi vi ønsker at spillerne har det sjovt.
Precondition	1. Skibet er ikke sunket.
Main success scenario	1. Spilleren vælger ammunition type (Chain, Cannon ball, Grape shot). 2. Ammunitionen bliver puttet i kanonerne og det koster en tur. 3. På næste tur er kanonerne klar til at skyde.

UC#4: Skibskollision

Scope	Spillerne og spil applikationen.
Level	To skibe kolliderer på en givent tile.
Primary actor	Spilleren og modspilleren.
Stakeholder	Spillerne fordi de “spiller” samt os, fordi vi ønsker at spillerne har det sjovt.
Precondition	1. Begge spillere har afsluttet deres tur. 2. Et skib fra begge spillere har passeret samme tile på samme tidspunkt.
Main success scenario	1. Turen bliver beregnet og ved kollision bliver skaden påført skibene. 2. Skibene stopper på kollisions positionen.

UC#5: Ildkamp

Scope	Spillerne og spil applikationen
Level	Der bliver udvekslet skud fra begge spillere når turen er slut.
Primary actor	Spilleren
Stakeholder	Spillerne fordi de “spiller” samt os, fordi vi ønsker at spillerne har det sjovt.
Precondition	1. Skibet er ikke sunket. 2. Kanonerne er ladt med den valgte ammunition. 3. Kanonerne er bemandet af 3 sømænd
Main success scenario	4. Spilleren vælger den tile hvor skibet skal skyde. 5. Baseret på en metode bliver der nu udregnet en damage som bliver placeret på den bestemte tile.

UC#6: Tur

Scope	Scopet for turen involvere spilleren og modstanderen i og med at de begge interagere med spillet imens turen tages. For eksempel når skibene skal rykkes skal det gøres aktiv af spillerne. Selve applikationen er også indenfor scopet.
Level	Begge spillere foretager alle træk og afslutter deres tur.

Primary actor	Spilleren fordi hele turen drejer sig om hvilket valg der tages.
Stakeholders	Spillerne fordi de “spiller” samt os, fordi vi ønsker at spillerne har det sjovt.
Precondition	Spillet er i gang
Main success scenario	1. Spilleren vælger hvor spillerens skibe skal sejle hen (se UC:8). 2. Spilleren vælger nu om der skal lades, skiftes ammo eller skydes (se UC:3, UC:5). 3. Når begge spillere har foretaget deres valg bliver turen sendt med en submit knap.

UC#7: Skibe

Der er tre typer af skibe til rådighed for spilleren: Brig, SOTL og MAW. Alle skibe har x antal kanoner samt sejl. En kanon skal bemandes af 3 sømænd for at være kampdygtig. Sejl skal bemandes for et skib kan sejle. Sømænd kan bemande kanoner og sejl på samme tid. Alle skibe har et max antal felter de kan selje. Max antallet af felter et skib kan sejle, bestemmes af skibstypen samt vindretningen og vindstyrken. Hvis der ingen sømænd er til at bemande sejlene på et givent skib kan skibet ikke sejle.

UC#8: Reload

Scope	Spillerne og spil applikationen
Level	Kanonerne bliver succesfuldt genladt.
Primary actor	Spilleren
Stakeholder	Spillerne fordi de “spiller” samt os, fordi vi ønsker at spillerne har det sjovt.
Precondition	1. Skibet er ikke sunket.
Main success scenario	1. Spilleren vælger at genlade kanonerne. 2. Kanonerne bliver ukampdygtige indtil turen er gået. 3. Når næste tur starter er kanonerne genladt.

UC#9: Bevægelse

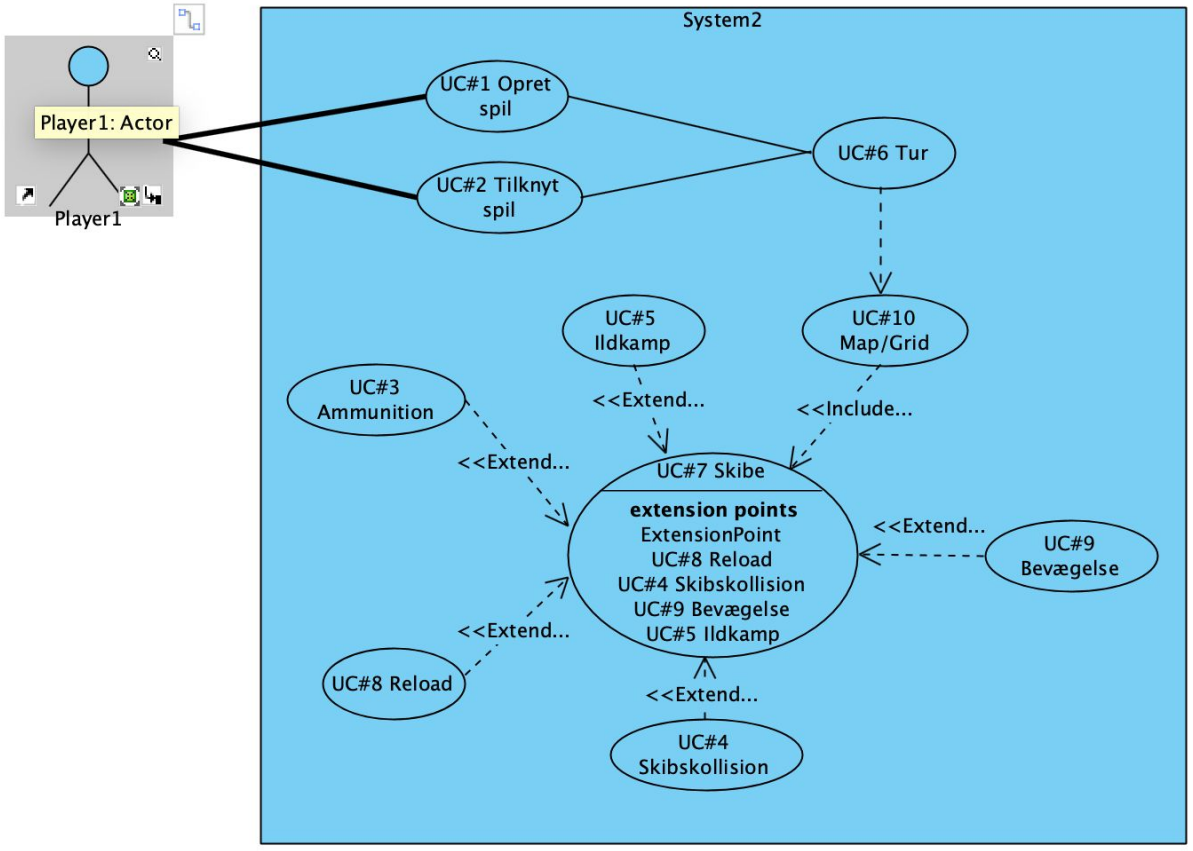
UC 9	**Skibene kan bevæges**
Scope	Scopet for denne use case er udelukkende applikationen
Level	Spillernes skibe bevæger sig på spillebrættet som spillerne ønsker inden for spillet bevægelses regler.
Primary Actor	Spilleren
Stakeholder	Spiludviklerne, Spillerne.
Precondition	Der skal være alt andet end meget modvind for at nogle skibe kan bevæge sig. Desuden skal skibets sejl være i stand og have bemanding. Skibet må ikke være sunket (HULL > 0).
Success Guarantee	Formålet er at skibet rykker til det ønskede koordinat valgt af spilleren.
Main success scenario	1. Du klikker først på det ønskede skib du gerne vil rykke. 2. Du vælger en af de tiles rundt om dig du vil rykke til og klikker med musen på den. 3. Programmet regner ud om det ønskede tile er lovligt. 4. Koordinaterne sendes nu over en socket. Der afventes information om den anden persons træk. 5. Programmet regner nu ud om der eventuelt opstår en kollision. 6. Programmet rykker nu på koordinaterne af skibene. Selve bevægelsen animeres nu i den grafiske brugergrænseflade.
Extensions	1. **Du klikker på fjendes skib** En metode tjekker om skibet er dit eller fjendens, hvis det ikke er dit, vil den prompte dig med beskeden “This option is not valid, only friendly ships can be moved” 2. **Du vælger en tile der er for langt væk til at skibet kan rykke dertil på nuværende tur.** En metode tjekker omgående om trækket er lovligt, hvis trækket er ulovligt vil der promptes denne besked til spilleren: “The ship doesn’t have any actions left” 3. **Du vælger en tile der kræver for mange turns på nuværende tur.** En metode tjekker omgående om trækket er lovligt,

	hvis trækket er ulovligt vil der promptes denne besked til spilleren: “The ship isn’t allowed to do this many turns. ”. 4. **Modstanderens socket bliver pludselig lukket** I tilfælde af en pludselig lukning af modstanderens socket, vil vores program håndtere dette, så applikationen ikke forårsager et crash.

UC#10: Map/Grid

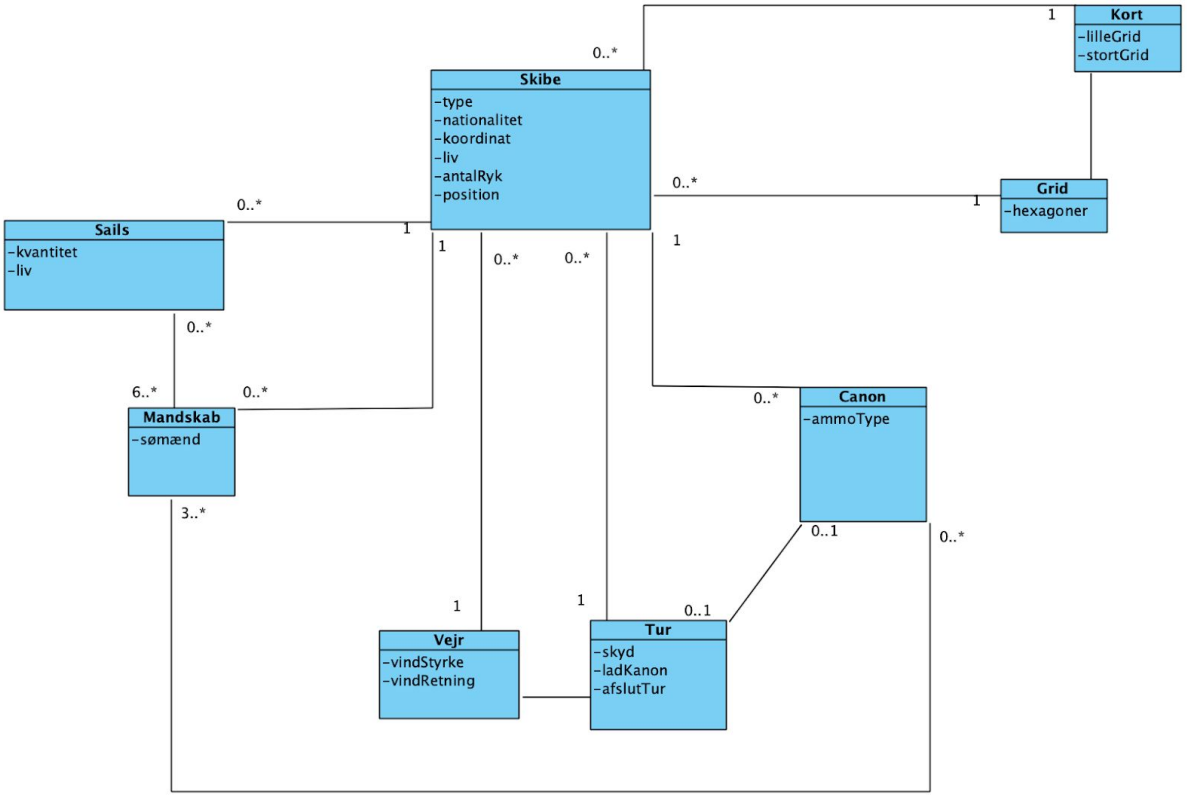
“Mappet” hvor slaget foregår, består af et hexagonalt “grid” på størrelsen 12x12, 24x24 eller 32x32. Grid er en gruppe tiles (Hexagonale tiles), som hver især har en unik position (X, Y).

Use Case Diagram



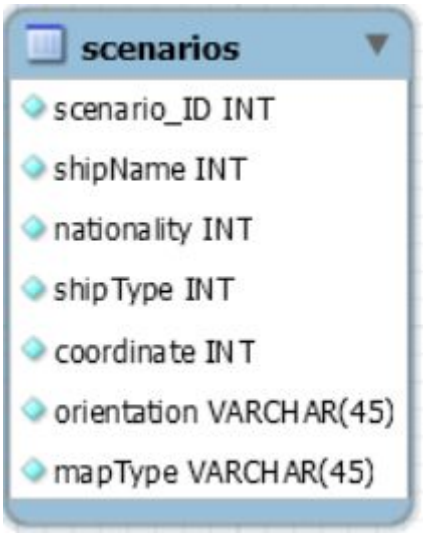
På diagrammet foroven har vi forsøgt at skabe et overblik over hvilke use cases der arbejder sammen samt hvilken relation de har til hinanden. Hvis man tager et kig på UC# 10 Map/Grid. Illustrerer diagrammet at denne Use Case har en include-relation til UC#7 Skibe og at UC#7 har en ekstension-relation til flere forskellige Use Cases, som bl.a. UC#9 Bevægelse.

Domæne model



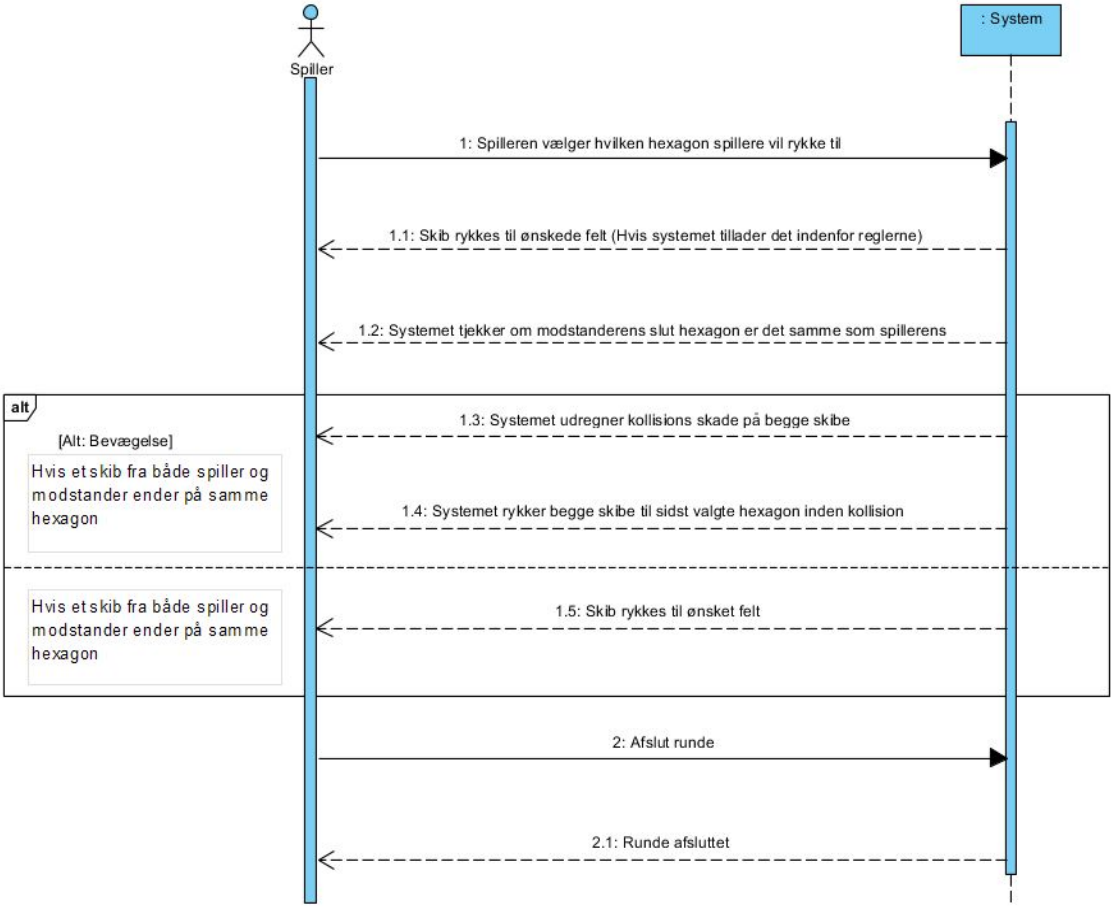
Efter vi har udformet vores use cases med tilhørende use case diagram til, kunne vi begynde at danne et overblik over hvilke konceptuelle klasser der skulle bruges til at lave en Domænemodel. Det er er gjort ved at kigge vores use cases igennem efter navneord primært, de navneord har så udviklet sig til attributter i vores konceptuelle klasser. Som man kan se i Use case diagrammet extender skibe til næsten alle vores use cases, det samme sker i vores domæne model.

ERD Diagram



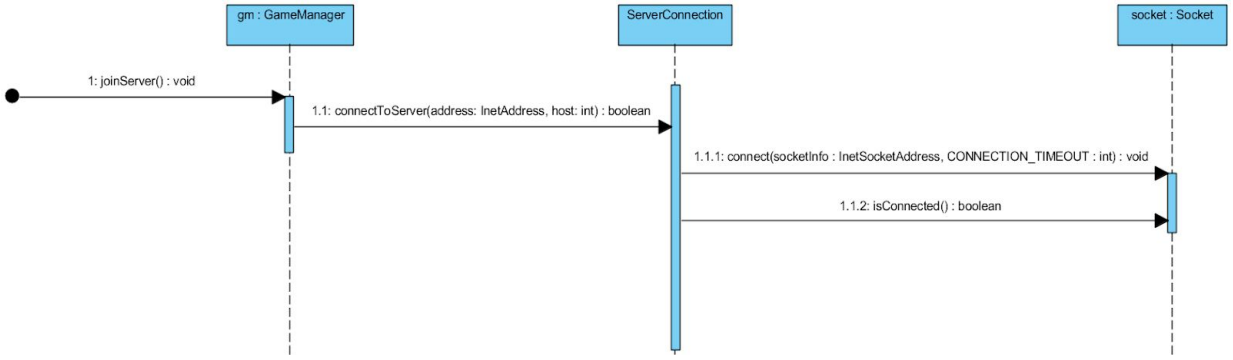
ERD diagrammet viser vores table med følgende data. Derudover kan vi se datatypen på de forskellige data.
Vi har valgt ikke at lave flere tabeller, da vi mener det holder sig indenfor de 3 normalformer, og kunne derfor ikke se idéen med dette.

System Sequence Diagram



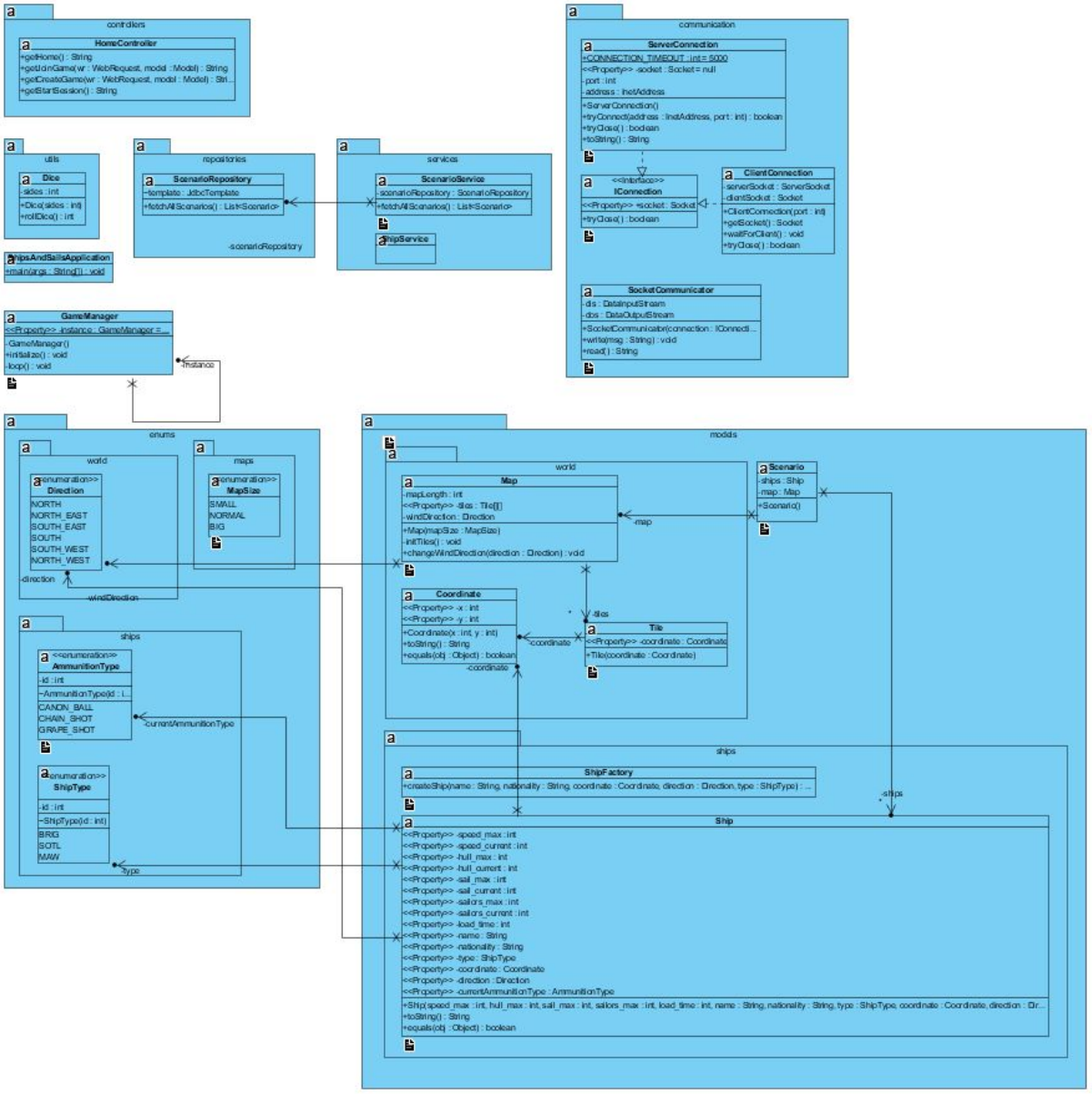
SSD:
System sequence diagrammet visualisere hvordan spilleren i et specifikt scenario får håndteret sine forespørgsler. I dette tilfælde kan vi se hvordan spilleren interagere med systemet, når spilleren skal rykke sit skib.

System Diagram



SD:
Vores SD diagram viser hvad der sker når en spiller vil slutte sig til en server. Det første kald starter fra vores GameManager som er en singleton der står for at holde styr på selve spillets gang: At starte, pause og slutte spillet etc.
Vi kalder så metoden joinServer som kreere et objekt af typen ServerConnection, derefter kalder vi på en serverConnection metode der hedder connectToServer der retunere en boolean. I ServerConnection objektet ligger der et Socket objekt som nu bliver instantieret. Nu kalder vi connect metoden fra vores socket objekt, i parametrene ligger vi et objekt af typen InetSocketAddress som holder på vores port og ip information.

Class Diagram



Fase 3 - Construction

Konklusion

Set ud fra projektets størrelse og tidsramme løb vi ind i en del problemer, som i har læst igennem rapporten. De problemer vi er stødt på, har været alt mellem manglende evner til diskussioner på flere timer. Det vi kan tage med os fra denne opgave og videre frem i forløbet er bedre strukturering omkring arbejdsansvar, fordeling af opgaver og overholde deadlines. Vi som udviklerer føler noget af ansvaret ligger til dels på vores skuldre samt opgave givernes, fordi der ikke blev udarbejdet ordentlige kravspecifikationer til start og vi måtte bruge første uge af projektet på at få de helt rigtige specifikationer skrevet ned.

Fase 4 - Transition

Future Aspects

Spillet er ikke udgivet. Det ligger i venteposition indtil videreudvikling af de manglende spil elementer som regel logikken, GUI, klient/server kommunikation bliver implementeret.