Visuel komponent til DSB tog-historik

Henrik Christoffersen
s083146

Kongens Lyngby 2012
IMM-B.Eng-2012-75

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk IMM-B.Eng-2012-75

Abstract

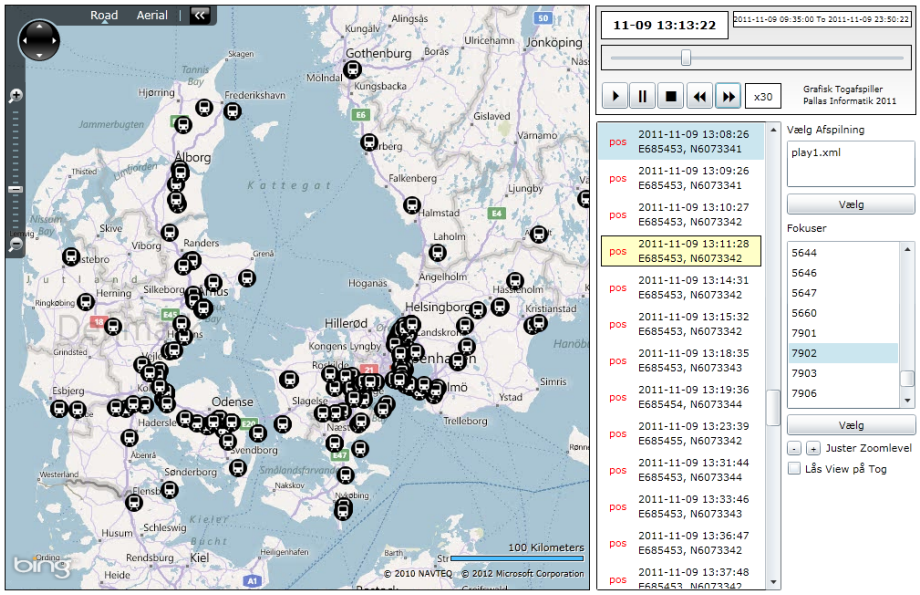
This bachelor thesis deals with the investigation of posibilities, offered by the application framework Microsoft Silverlight, to solve the problem of visualising specific position-data on a map representation supported by a web-mapping service.

This technology investigation was issued by small-company Pallas Informatik A/S, to build a potential product for Danish railway company DSB (Danske Statsbaner). The company belives, that by using data that is already being harvested from trains to make a product, that can visualize operations, the operations can be analyzed and improved to provide a better performance on a larger scale.

The study required a deep understanding of how Microsoft Silverlight worked, and it required a thorough research to draw conclusions. To develop a software system, that solves the problem, the author has used an interative and incremental approach by following principles of Unified Process.

The result of the investigation provides a softwareprototype, that has been demarced to providing what the user should experience. The prototype allows a user to visualize position-data of trains on the web-mapping service Bing Maps, and it allows a user to control the visualization of the data as a media player.

The thesis provides Pallas Informatik A/S with basis to decide, if the product is worth developing and a prototype to present to DSB to try and get them to invest in the project.



Figur 1: Screenshot af prototypen sidst i udviklingsforløbet.

Lyngby, 21-January-2012

Henrik Christoffersen

Henrik Christoffersen
s083146

Forord

Denne rapport er resultatet af mit bachelorprojekt, som er udarbejdet i perioden 29. august 2011 til 23. januar 2012.

Rapporten markerer afslutningen på professionsbachelor uddannelsen "Diplom-IT" ved Danmarks Tekniske Universitet (DTU). Rapporten afspejler mine kompetencer for projektarbejde og softwareudvikling, som er grundlaget for mit afgangsprojekt.

Projektet, som rapporten beskriver, er lavet i samarbejde med virksomheden Pallas Informatik A/S, hvor jeg gennemførte mit praktikforløb for uddannelsen.

Jeg vil i denne sammenhæng gerne rette en særlig tak til min virksomhedsvejleder Anders Spaabæk, som er seniorudvikler ved Pallas Informatik. Igennem mit praktikforløb og bachelorprojekt-forløb har Anders Spaabæk været min kontaktperson ved Pallas Informatik A/S, hvor han har hjulpet og støttet mig.

Projektet er ligeledes udført i samarbejde med min DTU-vejleder Anne Elisabeth Haxthausen, der er lektor ved Instituttet for Matematisk Modellering (IMM) på DTU. En særlig tak bør også tilgå hende, da hun har hjulpet mig i den rigtige retning med start- og slutfasen af mit bachelorprojekt.

Projektet forsvares d. 9. februar 2012, med ekstern censor udvalgt af Anne Elisabeth Haxhausen.

Indholdsfortegnelse

Figurer

Kapitel 1

Indledning

1.1 Pallas Informatik A/S

Dette projekt er lavet i samarbejde med virksomheden Pallas Informatik A/S. Denne virksomhed udvikler og implementerer avancerede softwareløsninger - primært i 4 brancher, hvoraf transportbranchen er den ene.

Pallas Informatik blev grundlagt i 1983 og har idag ca. 60 medarbejdere. I sin tid beskæftigede en afdeling i virksomheden sig med at udvikle og vedligeholde softwaresystemer til taxafirmaer. I løbet af de seneste 10 år har denne afdeling skiftet retning. Idag hedder afdelingen TRIT (Tele- og radiobaseret informationssystem til tog) og arbejder eksklusivt med tog-branchen. Pallas Informatik arbejder eksklusivt med DSBs langtoge, og DSB er afdelingens eneste kunde.

I foråret 2011 gennemførte jeg min ingeniørpraktik, som varede 20 uger, hos Pallas Informatik, og jeg har derfor fået et personligt forhold til virksomheden.

Mit bachelorprojekt er også udført i samarbejde med Pallas Informatik. Virksomheden havde en ide til et projekt, som de ønskede at udvikle til DSB. De foreslog, at jeg kunne lave en teknologisk undersøgelse for et produkt. I den sammenhæng har jeg lavet en protype på deres ide.

Pallas Informatik har længe forsøgt at sælge produktet til DSB. Igennem de første 2 måneder af projektperioden var det endnu ikke lykkedes at sælge dem ideen. Interessen fra DSB's side er dog steget den seneste tid.

Systemet der skal udvikles er afgrænset til to områder. Jeg har valgt at udvikle det ene som mit bachelelorprojekt. Eftersom DSBs interesse er steget, er Pallas Informatik begyndt at udvikle den anden.

Intentionen er at samle de to projekter til en fuldent prototype.

1.2 Problemstilling

DSB ejer en række af langtoge, som både bruges til at rejse uden- og indenrigs. Til at koordinere togene bruges flere værktøjer. Pallas Informatik A/S har i sin tid udviklet en boks, der er installeret i alle DSBs langtoge. Boksen kaldes en Tog Basis Enhed (TBE), og bruges til at dokumentere tog-hændelser.

Enheden noterer alle registrerede ændringer for toget. Enheden foretager også regelmæssige positionsmeldinger med GPS.

Ved at bruge det mobile netværk, rapporterer TBEen hændelserne til et centralt system. TBEen sørger for, at centralen altid har de nyeste data.

Pallas Informatik har ansvaret for centralen. Når DSB skal undersøge et problem, der er opstået, er de nødsaget til at kontakte Pallas Informatik. Pallas Informatik laver derefter en udskrift af den centrale database. DSB bruger denne udskrift til at finde fejlen. Alle hændelser må kontrolleres en ad gangen. Denne proces kan være meget besværlig, hvis der på forhånd ikke vides ret meget om fejlen. Nogle fejl kan være umulige at finde inden for en deadline.

På baggrund af den metode, der bruges idag, ønsker Pallas Informatik at sammensætte et nyt system til DSB. Systemet skal bruges, som et analyse-værktøj. Løsningen skal bruge central-databasen til at opsætte en data-model. Datamodellen skal kunne fortolkes grafisk. Det skal være muligt at "afspille" de registrerede hændelser, ved at animere et togsæts færden. Animationen skal foregå på et kort. Det skal være muligt at animere togsættet på en rute, opbygget af de meldte GPS-koordinater. Ruten skal opbygges igennem en flydende overgang mellem koordinaterne.

Pallas Informatik ønsker at lave en prototype, der undersøger teknologien Silverlight. Silverlight er en teknologi, der understøttes af webbrowsere.

Kapitel 2

Begreber og forkortelser

Her følger en kort forklaring af begreber og forkortelser, der er brugt i rapporten. En kort ordforklaring kan ses i Appendix A.

Ord	Ordforklaring
DSB	Danske Statsbaner
TRIT	Tele- og radiobaseret informationssystem til tog
GUI	Graphical User Interface
TBE	Tog Basis Enhed.
UP	Unified Process.
UML	Unified Modelling Language.
GPS	Graphical Positioning System.
GIS	Geographic Information System.
WMS	Web-Mapping Service.
XML	eXtensible Markup Language.
XAML	eXtensible Application Markup Language.
OMG	Object Management Group.
VPN	Virtual Private Network.
MVVM	Model-View-ViewModel
FURPS	Functionality, Usability, Reliability, Performance, og Supportability.

Kapitel 3

Teknologi

3.1 Forord

Dette kapitel forklarer, hvilke teknologier og tekniske detaljer projektet har fastsat. Det antages i resten af rapporten, at læseren har kendskab til dem.

3.2 Udviklingsmiljø og værktøjer

Til at udvikle projektet har Palls Informatik A/S stillet en *Dell Latitude E6410* bærbar computer til rådighed. Styresystemet er Microsoft Windows 7 Professional 64-Bit installeret med Service Pack 1.

Computeren er installeret med Microsoft Office 2010 pakken, og udviklingsmiljøet Microsoft Visual Studio 2010. Licenserne til alt software på den udleverede bærbar computer ejes af Pallas Informatik.

Da jeg skal lave den teknologiske undersøgelse i Silverlight[Micc], skal udviklingen foregå i Microsoft Visual Studio.

Til at illustrere et kort har jeg behov for en web-mapping service (WMS). Denne service kan automatisk generere verdens-billeder over internettet. Jeg har valgt at benytte mig af "Bing Maps"[Mica], udviklet af Mirosoft. Bing Maps stiller et bibliotek til rådighed ved navn "Bing Maps Silverlight Control v. 1.0"[Micb]. Biblioteket gør at "Bing Maps" let kan integreres i applikationen.

Indholdet af databasen fortolkes i en model. Modellen præsenteres i et fastsat struktur. Strukturen er defineret i XML[BPSM+06]. Fordelen ved XML er, at data kan opsættes i en child/parent relation. Dette betyder, at data opsættes som objekter, der kan nedarve mere data. Formatet gør det samtidig muligt at præsentere alle data-typer.

Modellen beskriver alle detaljer omkring hændelser for togsæt. Jeg har valgt at betegne modellen, som "Oplagringsmodellen".

3.3 Silverlight

Silverlight er udviklet af Microsoft. Formålet er at give en bruger flere muligheder i en webbrowser. Silverlight tilbyder vektorgrafik, animationer, samt lyd og video. Silverlight kan køres på de fleste platforme. Mange anser Silverlight, som en af de største konkurrenter til det anerkendte alternativ Adobe Flash. Silverlight udkom i første udgave i April 2007. Den nyeste version 4.0, er udgivet i Oktober 2011.

For at køre en Silverlightapplikation kræves det først, at Silverlight er installeret. Webbrowseren behandler herefter 2 lag af kode. De to lag udgør tilsammen hele Silverlight-applikationen.

Det første lag består af XAML[Mice]. XAML er et sprog, der deklarerer Silverlight-objekter i brugergrænsefladen. Det bagerste lag kan kontrollere de deklarerede objekter. Dette lag er baseret på .NET-frameworket. Laget håndter brugerens handlinger på objekterne. Til sammen udgør de to lag hele applikationen.

Silverlight er baseret på Microsoft Windows Presentation Foundation (WPF). WPF blev udviklet lang tid før Silverlight. Silverlight er et markant mindre framework, men har meget til fælles med WPF. Forskellen på de to platforme er, at WPF udvikler applikationer, der køres på computerens styresystem. Silverlight derimod, udvikler applikationer, der køres igennem en webbrowser.

Som supplement til Silverlight og WPF kan man bruge Microsoft Expression Blend. Expression Blend bruges til at designe bugergrænsefladen. Expression Blend er

et tilvalg, der ikke er påkrævet for at udvikle i Silverlight. I mit projekt er Expression Blend kun brugt i mindre grad.

3.4 Rapport

Bachelorrapporten er skrevet i LaTeX som er et opmærkningssprog til tekstformatering. Dokumentet følger de anbefalede retningslinjer, der er opsat af IMM for skrivning af speciale.

Formatet i rapporten er bestemt af *Luke Herbert*[Rap] på baggrund af arbejde udført af *Henrik Aalborg Nielsen*, *Thomas Fabricius*, Jan Larsen og *Finn Kuno*.

Kapitel 4

Planlægning

4.1 Forord

For at udvikle et produkt er det nødvendigt at fastlægge en fremgangsmåde. Jeg vil i dette afsnit beskrive den metode og de værktøjer, som jeg har har brugt.

4.2 Udviklingsmetode

Til at udvikle systemet har jeg valgt at bruge principperne fra systemudviklingsprocessen, Unified Process (UP)[Lar01,]. Jeg har valgt at ændre lidt i den typiske fremgangsmåde. For at forstå den proces, som jeg har valgt at bruge, er det nødvendigt at have kendskab til ideen bag UP.

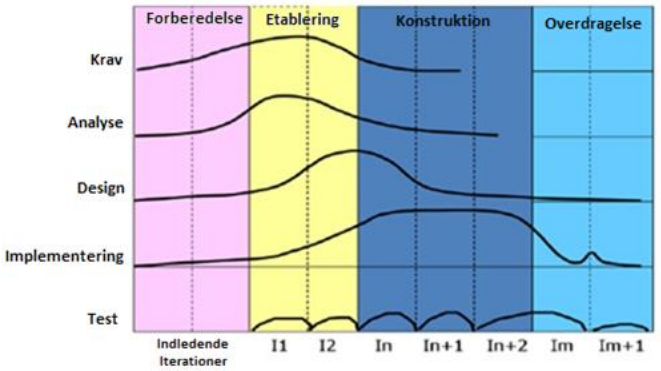
4.2.1 Unified Process

UP er en udviklingmodel, der beskriver et projekts livscyklus fra start til slut. Modellen falder ind i kategorien af "iterative aktivitets-centrerede" modeller.

UP fokuserer på fastsatte aktiviteter, som behandles i iterationer (over flere gange).

UP har 4 fastsatte tidsintervaller i forløbet. Disse intervaller kaldes også "cycles". (se figur 4.1). Man kan betragte et interval, som et forløb der typisk slutter med en prototype af produktet. Navnene på intervallerne er *Forberedelse*, *Etablering*, *Konstruktion*, og *Overdragelse*.

UP ligger også særlig vægt på 5 arbejdsforløb (workflows). Navnene på disse er forløb er *Krav*, *Analyse*, *Design*, *Implementering*, og *Test*. I mit projektforløb har jeg valgt at ændre lidt på disse.



Figur 4.1: Overblik over Unified Process modellen.

UP beskrives ofte med de to ord *Iterativ*, og *Inkrementel*. Iterativ betyder at man, for hver cycle, arbejder under en eller flere iterationer. Der tillader man arbejde på tværs af forskellige arbejdsforløb. Inkrementel betyder, at de 5 arbejdsforløb fortsætter igennem hver cycle.

Lad os betragte de 4 cycles i Unified Process, og beskrive deres formål i udviklingsprocessen.

4.2.2 Cycles i Unified Process

Unified Process opdeler projektet i udviklingsforløb (cycles).

4.2.2.1 Forberedelse

Denne fase fungerer som den indledende fase i projektet. Forberedelse er en kort fase, hvor man fastsætter de kritiske krav. Kravene bestemmes ved at analysere forestillingen omkring slutproduktet. I slutningen af fasen foretages en risikoanalyse. Denne analyse beslutter hvilken vej, systemet skal udvikles.

Forberedelse ligger vægt på at identificere nøglefunktionerne i systemet. Funktionerne beskrives som brugerscenarier mellem en aktør og systemet. Disse scenarier kaldes også use cases. Use cases bruges til at bestemme, hvordan funktionaliteten i systemet skal implementeres.

Når fasen er gennemført, skal projektets proces og værktøjer være valgt.

4.2.2.2 Etablering

Denne fase fastlægger en grundlæggende arkitektur, for det endelige system. Den efterlader en fortåelse af, hvordan systemet skal implementeres.

Der ligges særligt vægt på at bestemme interaktionen mellem systemet og brugeren. Etablering beskriver, hvordan use cases skal udføres. Fasen fastlægger også sekvens diagrammer, der bestemmer fremgangmåden for det interne system.

Etablering slutter med at tage en beslutning omkring, hvorvidt det kan betale sig at udvikle produktet.

4.2.2.3 Konstruktion

Denne fase er den største fase i projektet. Fasen tager udgangspunkt i behandlede use-case beskrivelser.

Målet med denne fase er at konstruere og implementere systemet. Systemet skal opfylde de krav, som blev sat i forberedelsesfasen. Når denne fase er slut skal systemet kunne gennemføre de fleste af sine test.

Konstruktion slutter med at tage en beslutning, omkring hvorvidt projektet har opnået sine krav og forventninger. Der besluttes om produktet kan overdrages til kunden.

4.2.2.4 Overdragelse

I denne fase skal projektet overdrages til kunden. Overdragelsen sker ved, at kunden og udviklerne i samarbejde, tilpasser systemet. Systemet skal kunne gennemføre alle tests med positive resultater.

Systemet implementeres hos kunden.

4.2.3 Arbejdsforløb

I denne sektion vil jeg beskrive de arbejdsforløb, som jeg har brugt i projektet. Jeg har begrænset mig til at bruge de udviklingsmetoder, som bidrager bedst til mit projekt.

4.2.3.1 Forberedelse

Dette arbejdsforløb forbereder projektet ved at lave en projektafgrænsning. Der bestemmes, hvad der skal udvikles, og der opsættes krav. Forberedelse bestemmer de funktioner, som skal implementeres, og kortlægger dem som use-cases.

Forløbet tager stilling til de risici der løbes i udviklingensfasen. Der opsættes kriterier for, hvornår prototypen er en succes.

4.2.3.2 Analyse

Dette arbejdsforløb bestemmer, hvad produktet skal kunne udføre. Forløbet udarbejder en domænemodel, som definerer problemområdet.

Forløbet bestemmer rammerne for de identificerede use-cases. Der bestemmes de direkte interaktionsforløb mellem bruger og system.

4.2.3.3 Design

Dette arbejdsforløb bestemmer designet for produktet. Forløbet bestemmer opbygningen af systemarkitekturen. Der laves en analyse af arkitekturmuligheder, og der vælges en.

Designforløbet bestemmer de byggeklodser, der skal udgøre arkitekturen. Den interne kommunikations-strøm bestemmes igennem sekvens diagrammer.

4.2.3.4 Implementering

Dette arbejdsforløb implementerer produktets byggeklodser. Byggeklodserne er valgt ud fra den valgte system arkitektur, og bestemt i Design-arbejdsforløbet.

Implementeringen håndterer de teknologiske begrænsninger, den teknologiske tilgang.

4.2.3.5 Test

Dette arbejdsforløb tester produktet ved at undersøge resultaterne af udførte use-cases. Forløbet er en succes, når alle succeskriterier er opfyldt.

4.3 Unified Modelling Language

Unified Process bruger teknologien Unified Modelling Language (UML)[Lar01], som designværktøj. UML bruges til at lave grafiske, visuelle modeller af objekts-orienteret software. Heriblandt domænemodeller og sekvens diagrammer.

I 1997 anerkendte organisationen Object Management Group (OMG) UML, som en af deres standarder. OMG er en international non-profit organisation. Virksomheden fastsætter standarder i industrien for softwareintegration i virksomheder.

4.4 Projektplan

Her ses min arbejdsplan for projektforløbet. Jeg har forsøgt at arbejde efter planen så godt som muligt, igennem hele projektforløbet. De grå felter repræsenterer de områder, hvor mit fokus er lagt.

	Ugenr.																		
Forløb	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	01	02
Forberedelse																				
Etablering																				
Konstruktion																				
Test																				
Overdragelse																				
Afslutning																				

Figur 4.2: Plan over arbejdsfordeling igennem projektperioden.

Kapitel 5

Forberedelse

5.1 Forord

I dette kapitel vil jeg beskrive forløbet "Forberedelse". Forløbet er defineret i sektion 4.2.3.1. Kapitlet beskriver baggrunden for problemet, og analyserer problemet som det er idag.

Forløbet afgrænser projektet, og opsætter krav. Kravene fortolkes som use-cases, der skal implementeres.

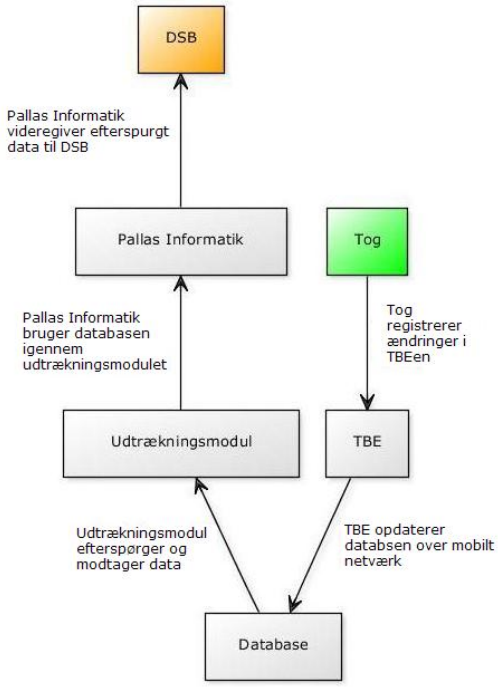
I slutningen af forløbet foretages en risikoanalyse. Analysen bruges til at tage en beslutning omkring, hvorvidt projektet bør fortsætte. Succeskriterier opsættes for at bestemme, hvornår projektets formål er opfyldt.

5.2 Problembeskrivelse

Problemstillingen blev defineret i sektion 1.2. Den beskriver den nuværende proces for problemområdet. Når DSB skal analysere en togdrift for at finde fejl, er det en omfattende proces. De har brug for Pallas Informatiks assistance for at få datagrundlaget.

Pallas Informatik råder selv over den centrale database. Virksomheden er derfor nødsaget til selv at forsyne DSB med udtræk. Disse videresendes idag til DSB over e-mail.

Følgende proces beskriver, hvordan data indsamles til en analyse.



Figur 5.1: Nuværende dataproces for analyse af togdata.

Når DSB modtager en kopi af dataudtræk fra Pallas Informatik, vil det have følgende format.

Tog nr: 2060

TID	TOGNR	AKT	LITRA	TRITID	TEKST
04-08 13:41:21	2060	msr	4398	1:624	
04-08 13:41:22	2060	isc	4398	1:624	4598 Mfb
04-08 13:41:24	2060	isc	4398	1:624	4598 Mfb
04-08 13:41:26	2060	isc	4398	1:624	4598 Mfb
04-08 13:41:29	2060	fst*	4398	1:624	
04-08 13:41:30	2060	alm	4398	1:624	Snekkersten, forsink. -810, hast. 0

Figur 5.2: Kopi af databaseudtræk for Langtog 2060

Databaseudtræk kan forekomme meget uoverskuelige. Det er en besværlig og

tidsomfattende opgave at analysere disse data.

Når en analyse skal foretages, kræver det en indsats både fra DSB og Pallas Informatik. På baggrund af dette problem ønsker Pallas Informatik at udarbejde en løsning. Det skal være lettere at foretage en analyse af togdriften, og det skal kunne gøres mere dynamisk.

Der skal udvikles en visuel komponent til en webbrowser. Komponenten skal fortolke dataudtræk grafisk. Løsningen skal give frihed til at styre afspilningen, som en videoafspilning.

Når systemet samles, skal en bruger kunne efterspørge data direkte. Det er er hensigten, at DSB skal kunne udføre en analyse, uden at inkludere Pallas Informatik.

5.3 Projektafgrænsning

Projektet som helhed er meget omfattende, og jeg har derfor valgt at afgrænse mit projekt til at fokusere på den del, som handler om at visualisere togdata i en browser.

Løsningen skal med tiden fungere som en service-applikation for DSB. Systemet skal kunne håndtere store mængder af data, og derfor er det vigtigt at kunne behandle dem hurtigt. Der kræves derfor en opdeling af systemetstrukturen.

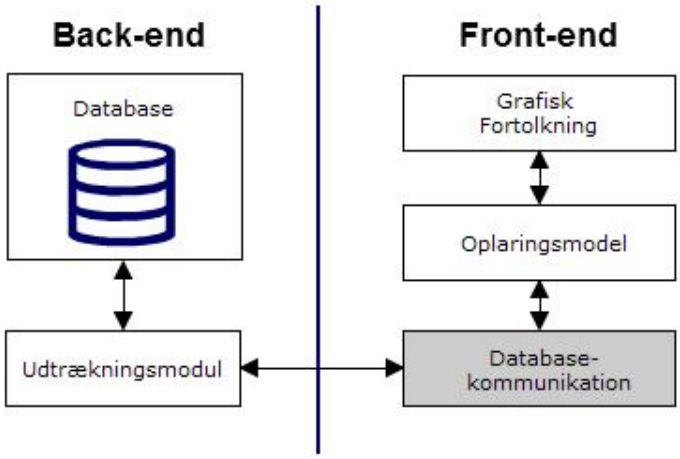
Systemet og database opdeles i et server/client forhold.

Server-delen fungerer som en back-end, der behandler databasen og frembringer en fortolket kopi. Denne kopi har jeg valgt at kalde for "Oplagringsmodellen".

Til at fortolke databasen bør implementeres et særligt modul. Dette modul betegner jeg, som "databasekommunikation".

Client-delen fungerer som en front-end, og bruger et modul til at kommunikere med serverdelen. Jeg har valgt at kalde modulet for "udtrækningsmodulet".

Det er meningen, at oplagringsmodellen skal fortolkes grafisk i webbrowseren, som en afspilning. Afspilningen kan styres igennem værktøjer, der gør det muligt at analysere indholdet.



Figur 5.3: Afgrænsning af det komplette System

Jeg har valgt at fokusere mit bachelorprojekt på front-end delen. Jeg skal derfor fortolke oplagringsmodellen visuelt. Jeg vil ikke implementere udtrækningsmodulet eller databasekommunikation. Disse moduler vil gøre projektet for omfattende.

Jeg har nu behov for at bestemme rammerne for *hvad* mit system skal kunne. Senere i rapporten vil jeg beskrive *hvordan* systemet skal gøre det.

5.4 Systemkrav

Til at bestemme kravene til systemet, har jeg brugt principperne fra FURPS+[BD04, p. 126-127]. FURPS er betegnelsen for en model, der beskriver 5 kvalitetsområder i software. De 5 områder er *Functionality* (Funktionalitet), *Usability* (Brugbarhed), *Reliability* (Pålidelighed), *Performance* (Præstation), og *Supportability* (Sikkerhed).

Senere er +'et tilføjet til modellen, for at markere 4 kategorier af projektbegrænsninger. Disse kategorier har jeg valgt at se bort fra. Istedet vil jeg fokusere på de første 5 kategorier for kvalitetsområder i software, og beskrive dem for projektet.

Jeg har omstruktureret resultatet af min FURPS model i to kategorier. *Funk-*

tionelle krav og *ikke-funktionelle* krav.

De funktionelle krav bestemmer de funktioner, som systemet skal kunne. Dette vil være de ting, som en bruger kan bruge systemet til. De funktionelle krav vil blive beskrevet med use cases og dækker over kategorien *Functionality*, i FURPS-modellen.

De ikke-funktionelle krav omhandler de ting, der forventes af systemets tilstand. Kravene giver et billede af systemets fysiske tilstand, når det overdrages. De ikke-funktionelle krav dækker over de resterende krav, der beskrives med FURPS-modellen.

5.4.1 Funktionelle krav

I samarbejde med Pallas Informatik har jeg bestemt en række funktionelle krav. Kravene er sat ud fra ønsker om, hvad systemet skal kunne udføre. De funktionelle krav er bestemt ved at idenficere dem som use cases.

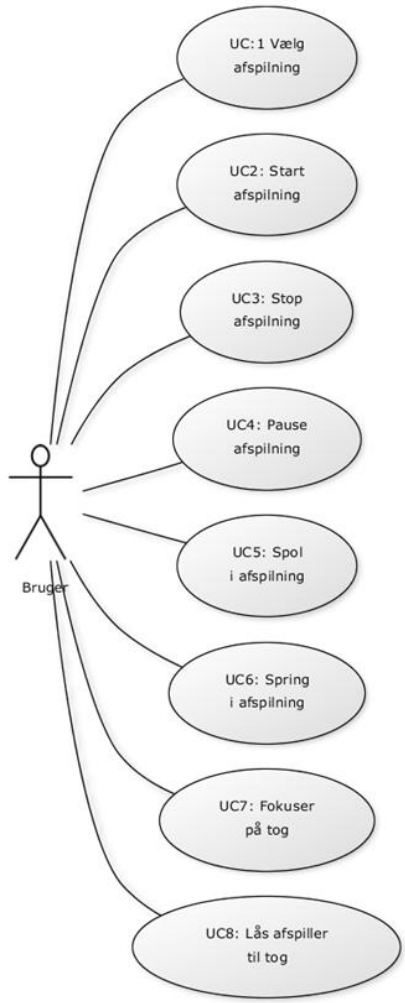
Jeg har udført en analyse til at bestemme, hvilke use cases, der først bør implementeres.

5.4.1.1 Identificering af use-cases

En use case kan som udgangspunkt blive udført af forskellige system-brugere (aktører). Brugerene kan have forskellige rettigheder i systemet. En aktør er en person, der har et særligt formål i systemet. Det kræver altid en aktør for at udføre et use-case. I mit projekt antages det, at der kun er en aktør, som tilmed har alle rettigheder. Aktøren betegnes i systemet som "Bruger".

Følgende liste beskriver de funktionelle krav til systemet, i form af use cases. Use cases er identificeret i sammarbejde med Pallas Informatik.

#	Navn	Beskrivelse
UC1	Vælg afspilning	En afspilning skal kunne kan vælges iblandt flere, i gennem en liste.
UC2	Start afspilning	Afspilleren starter fra begyndelsen. Herefter vil togsæt blive fortolket på kortet, som togsæt-elementer.
UC3	Stop afspilning	Afspilleren stopper, og dermed nulstillet.
UC4	Pause afspilning	Afspilningen sættes til at stå stille (pause). Der foretages ingen opdateringer af nogen art, indtil afspilningen igen bedes fortsætte, eller stoppes.
UC5	Spol i afspilning	Der vælges en ændring i afspilningshastigheden. Afspilleren vil nu begynde at vise data hurtigere eller langsommere end den før gjorde.
UC6	Spring i afspilning	Der vælges et nyt tidspunkt i afspilningen, som det ønskes at fortsætte fra. For at styre afspilningen bruges konceptet omkring en "Slider", som er et kontrolelement, der både kan vise og ændre en værdi.
UC7	Fokuser på tog	Der vælges et togsæt fra en liste, som dermed sættes i fokus. Når et togsæt er sat i fokus, vil en liste *EventList*, som repræsterer oplagringsmodellen blive opsat. Samtidig vil togsæt-elementer på kortet illustrere når en hændelse er sket.
UC8	Lås afspiller til tog	Afspilleren låser sig fast på det tog der er sat i fokus, og kortet flytter sit billede, så det passer med toget, når det bevæger sig rundt.



Figur 5.4: use-case diagram, for systemet

5.4.1.2 Prioritering af USE-Cases

Der kan opstå vanskelige problemer, når man skal implementere et helt system. Oftest kan det ikke forudsiges, hvilke udfordringer, der vil opstå. Derfor er det en fordel at bestemme implementerings-rækkefølgen af use cases. Prioteringen bør laves ud fra, hvor realistisk implementeringen forekommer.

Jeg har lavet min prioriterede liste af de identificerede use cases. Listen er prioriteret på baggrund af en analyse. Den kortligger nødvendigheden for use casen og vanskeligheden ved at implementere den.

Prioritet (P) beskrives med karakterskalan 1-3. 1 beskriver højeste prioritet, og 3 beskrives laveste prioritet.

Risiko (R) beskrives også med karakterskalane 1-3. 1 beskriver laveste risiko, og 3 beskriver højeste risiko.

Ved at multiplicere de to tal med hinanden, får jeg en score. Scoren bruger jeg til at vurdere de forskellige use cases imod hinanden. Use-cases med lavest score, bør implementeres først

Jeg har beregnet score for alle use-cases, og sorteret dem. Det første element i listen har lavest score, og det sidste element i listen har højest score.

UC1: Vælg afspilning		
Prioritet: 1	Risiko: 1	Score: 1
For at kunne styre en afspilning er det nødvendigt, at den rigtige afspilning er valgt. Denne use case fungerer derfor som forgænger til de andre use cases. Risikoen ved at implementere den er knap så høj, da den formodentligt ikke beskæftiger sig direkte med Silverlight teknologi.		

UC2: Start afspilning		
Prioritet: 1	Risiko: 3	Score: 3
Det skal være muligt at starte en afspilning før den kan analyseres, og derfor er prioriteten for denne use case høj. Implementeringen kræver både kode til at styre afspilningen, samt kode til at fortolke og vise den. Der kræves mange ting af Silverlight og derfor følger der en høj risiko ved at implementere denne use case.		

UC6: Spring i afspilning		
Prioritet: 1	Risiko: 3	Score: 3
Hvis man skal kunne analysere en afspilning effektivt kræves det, at man kan springe i forløbet for at finde, hvad man leder efter hurtigt. Derfor prioriteres denne use case højt. Use casen kræver, at afspilleren kan behandle meget data hurtigt, og sætter jeg risikoen højt.		

UC7: Fokuser på tog		
Prioritet: 1	Risiko: 3	Score: 3
For at analysere et tog er det vigtigt, at filtrere de data man gerne vil se. Det skal være muligt at fokusere på et enkelt togsæt, og dermed få vist togsættets data eksklusiv på en overskuelig måde. Prioteten sættes derfor højt i forhold til projektets formål. Hvis der skal vises eksklusivt data, kræver det samspil mellem elementerne i brugergrænsefladenGUIen og koden der styrer dem. Jeg sætter derfor risikoen til mellem.		

UC3: Stop afspilning		
Prioritet: 2	Risiko: 2	Score: 4
En bruger skal kunne stoppe en afspilning når det ønskes, at nulstille, hvad der vises. Prioritet sættes til mellem. For at nulstille, hvad der vises bør det ikke kræve særlig funktionalitet, dog sættes Risiko til mellem da det er en forudsætning, at en afspilning først skal kunne vælges.		

UC4: Pause afspilning		
Prioritet: 2	Risiko: 2	Score: 4
En bruger skal kunne pause afspilningen og fortsætte den igen. Det er en vigtig del af oplevelsen, at man kan fryse afspilningen for at analysere billedet. Prioritet sættes til mellem. For at pause afspilningen bør det ikke kræve særlig funktionalitet, dog sættes Risiko til mellem da det er en forudsætning, at en afspilning først skal kunne vælges.		

UC5: Spol i afspilning		
Prioritet: 2	Risiko: 2	Score: 4
En bruger skal kunne spole i afspilningen for at opleve den hurtigere. Det er en vigtig egenskab at have, hvis systemet skal kunne bruges til at foretage en analyse. Prioritet sættes til mellem. For at spole i afspilningen bør det ikke kræve særlig funktionalitet, dog sættes Risiko til mellem da det er en forudsætning, at en afspilning først skal kunne vælges.		

UC8: Lås afspiller til tog		
Prioritet: 3	Risiko: 2	Score: 6
Hvis man ønsker at opleve, hvordan et enkelt tog har opført sig på en strækning, vil man med fordel kunne låse afspilleren fast til et togsæt, der animeres på kortet. Prioriteten sættes dog lavt i forhold til de resterende use cases. Risikoen ved at implementere denne use case er, at kortet i forvejen skal tilbyde denne mulighed. Jeg sætter derfor Risko til mellem.		

5.4.2 Ikke-funktionelle krav

Denne sektion beskriver de ikke-funktionelle krav. Disse krav kan opfattes som fysiske begrænsninger på produktet. Kravene bestemmer den ønskede tilstand for systemet. Tilstanden defineres ud fra de resterende systemkrav i FURPS.

Systemet er afgrænset i to dele, og jeg skal implementere front-end. Jeg har undladt at implementere database kommunikation. Derfor vil de følgende krav ikke være særligt aktuelle for systemet. Jeg har valgt at beskrive dem alligevel, for at give en bedre forståelse for løsningen.

Brugbarhed
Brugergrænseflade skal have et klart design, der gør det let for en bruger at styre en afspilning. Der skal være orden på de visuelle elementer, som en bruger kan styre. Brugergrænsefladen skal beskrives med vejledende tekst de steder, hvor det er nødvendigt.

Pålidelighed
Det forventes af en afspilning, at tidspunktet stemmer overens med, hvad der vises. Hvis dette ikke er muligt, vil afspilleren ikke kunne bruges som et analyseværktøj.

Præstation
Det forventes, at en afspilning kan vise mindst 10 toge ad gangen, uden at overbelaste systemet.

Det skal være muligt at bruge kortet under en afspilning.

Sikkerhed
Det forventes løsningen skal tilgås igennem et lukket netværk. Systemet skal enten fungere over VPN, eller også skal der opbygges et login-system. Det kræves under alle omstændigheder, at netværkskommunikation er krypteret.

5.5 Risikoanalyse

Ved at være bevidst omkring potentielle risici, er de nemmere at undgå. Risikoanalysen gør det nemmere at løse problemer, når de opstår. Analysen kortlægger potentielle farer for produktet.

Jeg vil nu beskrive de risici, som jeg har identificeret. Jeg har taget stilling til, hvordan de bør løses.

Problemerne er sorteret, så de mest tænkelige risici er nævnt først.

Teknologiske udfordringer
Det vil være stort set umuligt at vide på forhånd, hvilke udfordringer, der kan opstå. Det kan dog antages, hvilke teknologiske områder, der vil være mest udfordrende.

Jeg har i forvejen en smule erfaring med de værktøjer, jeg skal bruge, undtagen Silverlight. Derfor vil Silverlight sandsynligvis volde flest problemer. Samtidig kan der opstå andre teknologiske problemer, som jeg ikke vil kunne forberede mig på.

Måden at løse teknologiske udfordringer på, er ved at anerkende dem. Min erfaring siger mig, at hvis et produkt har et særligt formål, kræves der en særlig løsning. Jeg vil derfor afsætte ekstra tid til at implementere projektet.

Unødvendig refakturering
I hver cycle af software udviklingen vil der være færdiggjort en prototype af produktet. Prototyper implementerer ikke alle use cases ad gangen. Nogle use cases stiller vidt forskellige krav til koden. En radikal ændring af koden vil kræve unødvendig refakturering.

For at komme dette problem til livs er man nødsaget til at gennemtænke systemarkitekturen. Strukturen bør opbygges så man kan centralisere koden. En god arkitektur fordeler ansvar til særlige klasser og opsætter kun få metoder. Disse metoder skal styre systemets flow. På denne måde kan man nøjes med at rette sin kode få steder.

Ændring af krav
Hvis man støder på en teknologisk begrænsning, kan det potentielt eleminere et krav. Dette vil ændre det endelie produkt. Hvis en eller flere use cases ikke kan udføres, skal projektets fremtid tages op til genovervejning.

Når man undersøger ny teknologi, kan man ikke forberede sig på de teknologiske begrænsninger.

Data tab
Produktudviklingen kan sættes markant tilbage, hvis vigtige filer forsvinder. Et sådant tab kan have stor betydning. Hvis filer forsvinder vil det kræve, at kode skal skrives igen. Størrelsen på problemet varierer efter, hvor meget funktionalitet, der er mistet. Der kan være flere årsager til, at data forsvinder.

Det er altid vigtigt, at man sikrer sig en back-up version af sit arbejde. Man kan eksempeltvis opbevare sin backup på en lagerenhed.

Igennem mit projekt har jeg jævnligt taget backup. Jeg har gemt alle stabile versioner af systemet. Udover at tage backup på en lagerenhed, har jeg brugt programmet *Dropbox*. Dropbox er en cloud-løsning, der kan opbevare private filer. Programmet kan synkronisere dine data over internettet, og er gratis.

5.6 Succeskriterier

Jeg har fastsat en række kriterier, som jeg ønsker, at produktet skal kunne opfylde. Løsningen anses som en succes, når alle kriterier er opfyldt. Kriterierne sætter mål, som skal gennemføres og beskriver retningen for udviklingen.

1. Systemet har en play knap, der får et tog til at bevæge sig i en rute.
2. Systemet kan vise events på et tog, som de forekommer.
3. En bruger kan dynamisk skifte mellem det tog han vil have informationer om.
4. En bruger kan operere afspilningen som en normal medieafspiller, uden markant forskel på disse.
5. Man kan låse sig fast til et tog og følge det rundt, som der afspilles.

5.7 Delkonklusion

Jeg har nu foretaget de forberedelser for projektet, som danner fundamentet for produktet. Jeg har beskrevet problemet som det er idag, og afgrænset den del af løsningen, som jeg ønsker at lave.

Jeg har brugt systemkrav til at bestemme brugerscenarier (use cases), der skal danne grundlaget for systemets udvikling.

Riskoanalysen tager stilling til de problemer, som kan opstå i projektet, og succeskriterierne indikerer, hvornår formålet med mit projekt er opfyldt.

På baggrund af risikoanalysen mener jeg ikke der står noget ivejen for, at systemet kan fortsætte. De teknologiske begrænsninger for systemet, som potentielt kunne afslå krav, kan jeg ikke kende på forhånd.

Kapitel 6

Analyse

6.1 Forord

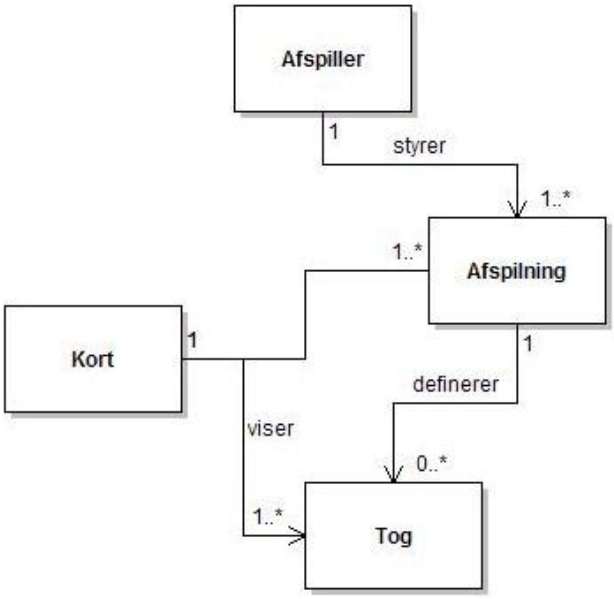
Dette kapitel beskriver hvilke analyser, der er foretaget med henblik på at nå en løsning. Kapitlet redegør for arbejdsforløbet, som er defineret i sektion 4.2.3.2.

Kapitlet bruger principperne fra UP til at analysere domænet for projektet. Jeg foretager en analyse for hver af de identificerede use cases.

6.2 Domænemodel

Formålet med en domænemodel er at vise de fysiske objekter, der beskriver problemdomænet. Domænemodellen viser forholdet mellem dem. En domænemodel bruges til at kortlægge en general ansvarsfordeling for et problem. Objekterne er identificeret ved en navneordsanalyse af use case beskrivelserne.

Domænemodellen er opsat så objekternes relationer markeres med en pil. Mængdeforholdet er noteret i hovedet og bunden af pilen.



Figur 6.1: Domænemodel for systemet.

6.3 Use case beskrivelser

I denne sektion foretages en dybere analyse af de identificerede use cases. De beskrives i tilstanden "fully dressed". Denne model kan beskrive flest detaljer for et scenarie. Jeg har fravalgt overflødige detaljer.

Prækondition beskriver tilstanden, som systemet skal være i eller hændelser, der skal være sket før at use casen kan udføres.

Postkondition beskriver tilstanden, som systemet vil være i efter at use casen er udført.

Primært succes flow beskriver rækkefølgen af interaktioner, der skal foregå mellem brugeren og systemet. Beskrivelsen er udført i punktform og beskriver kun den primære udførelsesfremgang.

Tilføjelser beskriver de punkter i succes-flowet, hvor aktøren er stillet to eller flere valg. De sekundære muligheder for punkterne noteres her.

Teknologi og data variation liste beskriver hvis der er andre teknologiske måder at udføre en use case på. Det noteres også, hvis der er flere datagrundlag for udførelsen.

Frekvens beskriver hvor ofte at use case blive udført i systemet, og under hvilke omstændigheder.

UC1: Vælg afspilning	
Prækondition	Programmet er startet op.
Postkondition	Afspilleren er gjort klar til at afspille data som den er opsat med, og kan nu startes.
Primært succes flow	
Bruger	System
1. Bruger vælger en afspilning fra en liste.	
2. Bruger trykker på knappen "Vælg"	3. Systemet finder det efterspurgte data, opsætter afspilleren, og klargører en afspilning.
Tilføjelser (eller alternativt flow)	Ingen tilføjelser
Teknologi og data-variations liste	Ingen særlige variationer
Frekvens	Hver gang, at en bruger ønsker at afspille et nyt datasæt vil det være nødvendigt, at opsætte afspilleren.

UC2: Start afspilning	
Prækondition	Afspilleren er gjort klar til at afspille data, som den er opsat med, og kan startes.
Postkondition	Data fortolkes på kortet og kan styres igennem en afspillerkontrol.

Primært succes flow	
Bruger	System
1. Bruger trykker på knappen "Play".	2. Systemet begynder at fortolke data fra begyndelsen i afspilningen, og opdaterer animationer på kortet.

Tilføjelser (eller alternativt flow)	Ingen tilføjelser
Teknologi og data-variations liste	Ingen særlige variationer
Frekvens	Når brugeren ønsker at starte en afspilningen fra begyndelsen.

UC3: Spring i afspilning	
Prækondition	Afspilleren er gjort klar ved, at en afspilning er valgt. (Valgfrit)Der er sat "fokus" på et tog og dermed opsat en eksklusiv liste med togdata.
Postkondition	Data fortolkes stadig på kortet, men fra et nyt sted i afspilningen.

Primært succes flow	
Bruger	System
1. Brugeren vælger nyt sted i afspilningen som der ønskes, at fortsætte fra.	2. Systemet fortsætter fra det nye sted i afspilningen, og fortsætter den afspilnings-tilstand som den før havde (Afspillende/Pauset).

Tilføjelser (eller alternativt flow)	**1a** Aktør bruger et slider-kontrolelement ved, at venstre-klikke på det sted i slider-sporet, der ønskes at forsætte fra. **1b** Aktør bruger slider-nålen på slider-kontrolelementet ved at trække den hen til en position, der svarer til det sted i afspilningen, der ønskes at fortsætte fra. **1c** Aktør bruger listen, der repræsenterer events. [1] Aktør finder den hændelse listen der gerne vil skiftes til. [2] Brugeren dobbeltklikker på hændelsen i listen med venstre musseknap.
Teknologi og data-variations liste	Der er to muligheder for at springe i afspilningen. Brugeren bruger Event-List, eller brugeren kan bruge slideren.
Frekvens	Use casen udføres, hver gang det ønskes at skifte til et nyt tidspunkt i afspilningen, og dermed springe noget over.

UC4: Fokuser på tog	
Prækondition	Afspilleren er gjort klar ved, at en afspilning er valgt.
Postkondition	Event-List er opsat med data og den seneste hændselse fra togsættet, som er i fokus vil automatisk være markeret.

Primært succes flow	
Bruger	System
1. Brugeren vælger navnet på et togsæt fra fokus listen, og trykker på knappen "Vælg".	2. Systemet opsætter Event-List med data for det togsæt, der er sat i fokus, og opsætter en besked, der kan ses på kortet når en hændelse finder sted.

Tilføjelser (eller alternativt flow)	Ingen tilføjelser
Teknologi og data-variations liste	Ingen særlige variationer
Frekvens	Hver gang det ønskes at se data for et særligt togsæt skal fokus være sat. Det kræves derfor, at fokus skiftes, hvis det forkerte togsæt er valgt.

UC5: Stop afspilning	
Prækondition	Programmet er startet op, og afspilleren er opsat med data.
Postkondition	Afspilningen er nulstillet, og verdenskortet bliver ikke længere opdateret.

Primært succes flow	
Bruger	System
1. Brugeren trykker på knappen "Stop".	2. Systemets tilstand sættes til, at være stoppet og afspilningen nulstilles til starttidspunktet.

Tilføjelser (eller alternativt flow)	Ingen tilføjelser
Teknologi og data-variations liste	Ingen særlige variationer
Frekvens	Hver gang det ønskes, at stoppe afspilningen.

UC6: Pause afspilning	
Prækondition	Programmet er startet op, og afspilleren er opsat med data.
Postkondition	Hvis der i forvejen afspilles vil afspilningen pauses, og dermed ikke længere opdatere verdenskortet. Hvis der ikke afspilles vil afspilningen fortsætte fra det tidspunkt i afspilningen, hvor den i forvejen er nået til.

Primært succes flow	
Bruger	System
1. Brugeren trykker på knappen "Pause".	2. Systemets tilstand skiftes fra at afspille til at pause, eller fra at pause til at afspille.

Tilføjelser (eller alternativt flow)	Ingen tilføjelser
Teknologi og data-variations liste	Ingen særlige variationer
Frekvens	Hver gang det ønskes at pause afspilningen eller fortsætte den.

UC7: Spol i afspilning	
Prækondition	Programmet er startet op, og afspilleren er opsat med data.
Postkondition	Hastigheden på afspilningen vil være forøget eller formindsket.

Primært succes flow	
Bruger	System
1. Bruger vælger at trykke på knappen "Spol mere" eller knappen "Spol mindre"	2. Systemet forøger hastigheden, hvis "Spol mere" vælges, eller formindsker hastigheden, hvis "Spol mindre" vælges.

Tilføjelser (eller alternativt flow)	Ingen tilføjelser
Teknologi og data-variations liste	Ingen særlige variationer
Frekvens	Hver gang en bruger ønsker at afspille hurtigere, eller mindre. Denne use case vil blive udført ofte, da man ikke kan analysere en afspilning i real-tid særlig effektivt.

UC8: Lås afspiller til tog	
Prækondition	Programmet er startet op, og afspilleren er opsat med data. Et togsæt er sat i fokus.
Postkondition	Kortet er fastlåst på det togsæt, der er i fokus, og lader nu billedet følge togsættet.

Primært succes flow	
Bruger	System
1. Bruger sætter et flueben i en kontrolboks.	2. Systemet sætter togsæt-elementet på kortet i fokus, og lader kortet opdatere billedet så elementet hele tiden ses.

Tilføjelser (eller alternativt flow)	Ingen tilføjelser
Teknologi og data-variations liste	Ingen særlige variationer
Frekvens	Hvis en bruger ønsker at analysere rejsen for kun et enkelt tog, kan det være en fordel at udføre denne use case. Det vil ikke være påkrævet at låse kortet til et togsæt for at udføre en analyse.

6.4 Delkonklusion

Jeg har nu foretaget en analyse af projektet som koncept. Den analyserede domænemodel kan bruges til at opsætte en ansvarsfordeling for designet.

I sektion 5.3 beskrev jeg, at databasekommunikation ikke vil blive implementeret. Det betyder, at oplagringsmodellen ikke modtages direkte af databasen. Jeg har valgt at erstatte modulet med en ekstern fil, som skal indlæses direkte i programmet.

Min afgrænsning gør, at de fleste ikke-funktionelle krav ikke vil blive implementeret. Min løsning vil primært fokusere på at implementere det ikke-funktionelle krav *Brugbarhed*.

Jeg analyserede projektet som koncept og opsatte en domæne model til at kortlægge en ansvarsfordeling. Derefter beskrev jeg de identificerede use cases, for at sætte krav til systemets design.

Kapitel 7

Design

7.1 Forord

Dette kapitel beskriver den proces, der udarbejder strukturen for systemet. Kapitlet redegør for arbejdsforløbet, der er defineret i sektion 4.2.3.3.

Analyse-arbejdsforløbet fokuserede på *hvad* programmet skulle have og kunne. I designfasen vil jeg beskrive hvilke byggeklodser, der skal implementere det.

Da jeg afgrænsede mit projekt, valgte jeg at fokusere på front-end. Denne afgrænsning består af to moduler, som hedder "Oplagringsmodel" og "Grafisk fortolkning". Begge moduler har behov for et design.

Oplagringsmodellen beskriver data for alle togsæt i en efterspurgt periode. Programmet skal fortolke modellen på samme måde, ligegyldig hvad indholdet er. Derfor er strukturen vigtig.

Det andet område er den grafiske fortolkning, som skal udvikles i Silverlight. Til at opbygge apllikationen skal jeg bruge en programarkitektur. Jeg vil derfor først undersøge, hvilke arkitekturiske muligheder jeg har. Undersøgelsen skal ligge vægt på begrænsninger og muligheder.

Det er vigtigt for designet at fastsætte, hvordan systemet skal udføre funktionalitet internt. Jeg bruger principperne fra UP til at opsætte sekvensdiagrammer og illustrere flowet for det implementerede system.

7.2 Datamodel

Når clienten skal foretage en afspilning, er det vigtigt, at den har direkte adgang til oplagringsmodellen. Data skal placeres lokalt, når det skal behandles hurtigt, og der må derfor ikke være behov for databasen under en afspilning.

I sammarbejde med Pallas Informatik har jeg opbygget en struktur til oplagringsmodellen, ved hjælp af XML. Strukturen blev udviklet tidligt i projektforløbet. Kravene til oplagringsmodellen blev senere udvidet, da Pallas Informatik indtrådte i projektet. Strukturen har ændret sig en smule siden da.

Opbygningen af oplagringsmodellen er lavet ud fra konceptet om et array, hvor hvert opslag i modellen forklarer en hændelse for et togsæt.

Databasen indeholder flere typer hændelse, men i prototypen har jeg valgt kun at implementere positionsmeldinger. Det er min hensigt at kunne implementere de andre typer af hændelser, når systemet kan illustrere en togrejse på kortet.

```
<?xml version="1.0" encoding="utf-8" ?>
- <ArrayOfRowInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <RowInfo>
      <TritId>1:349</TritId>
      <Time>2011-11-09T10:43:02</Time>
      <TrainNumber>0</TrainNumber>
      <Litra>4367</Litra>
      <Type>position</Type>
      <Text>East: 800722 | North: 6232460 | Speed: 95 | Delay: -6 | DS: 0 | StaId: | OprTogNr 11049</Text>
    </RowInfo>
      ...
  - <RowInfo>
      <TritId>1:150</TritId>
      <Time>2011-11-09T10:46:48</Time>
      <TrainNumber>14</TrainNumber>
      <Litra>5014</Litra>
      <Type>position</Type>
      <Text>East: 729858 | North: 6170892 | Speed: 0 | Delay: -288 | DS: 0 | StaId: 8600858 | OprTogNr 14</Text>
    </RowInfo>
  </ArrayOfRowInfo>
```

Figur 7.1: Oplagringsmodellen i xml-format, der viser den første og sidste positionsmelding.

Som det kan tydes på figur 7.1 efterligner strukturen et array. Modellen er beskrevet igennem objektet <ArrayOfRowInfo>, hvor <RowInfo> definerer objekter som hændelser.

For at udvikle en prototype, der undersøger de teknologiske muligheder i Silverlight, behøver jeg ikke at bruge alle data i oplagringsmodellen. Jeg ønsker kun at implementere en prototype, der animerer togsættets færden.

Positionshændelserne beskriver flere parametre, men er ikke nødvendige til at styre en animation. For at styre en animation, har jeg kun behov for følgende data i oplagringsmodellen.

Time er det tidspunkt hvor toget har rapporteret en hændelse og registreret den i TBE'en. Tiden kan præcisere ét sekund.

Litra er et særligt ID, der bestemmer et togsæt. IDet bestemmer en kombination af 2 eller flere forbundne vogne og bruges til at definere en togrejse. Litra IDet er typisk et reserveret nummer i DSBs togsystem.

Type fortæller hvilken hændelse, der er registreret. Prototypen fokuserer på hændelsestypen "position". En positionsmelding er når TBEen registrer GPS koordinater til togets position.

Text er en linje, der beskriver data for hændelsen. Text er et objekt, der formateres efter hændelsestypen. Er typen sat til "position" kan vi forvente 7 parametre, adskilt af tegnet |. Til at styre animationer har vi kun behov for parametrene *East* og *West* Til sammen udgør de et GPS-koordinat.

Hvis prototypen skal videreudvikles vil de resterende parametre, kunne bidrage til et mere omfattende projekt.

7.3 Applikationsarkitektur

Denne sektion beskriver to arkitekturiske muligheder for mit design, som begge retter sig mod softwareapplikationer. Jeg vil beskrive opbygningen af begge arkitekturer og redegøre for mit valg.

Beslutningen træffes ud fra 2 kriterier. 1. Hvor godt arkitekturens understøtter integrering af Bing Maps, og 2. Hvor godt arkitekturen understøtter animationer.

Til at beskrive softwarearkitektur bruges ofte følgende 3 koncepter: Model, View, og Controller. De fleste programmer, der udvikles af ingeniører, bygger på en treenighed, som kaldes Model-View-Controller[MVC].

7.3.1 Model-View-Controller (MVC)

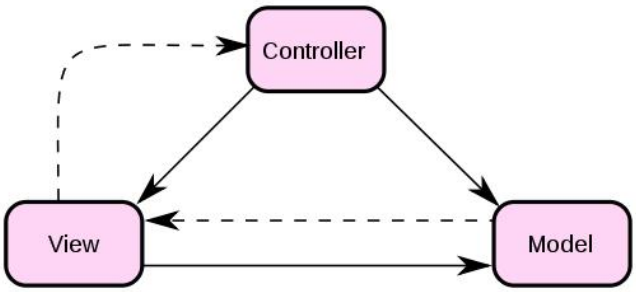
Denne softwarearkitektur består af 3 ansvarsopdelinger, der tilsammen udgør det komplette system.

Model er den opdeling, der indeholder data og definerer tilstanden for systemet. Dataen er som regel indkapslet for at bevare konsistens i systemet. Typisk kan en model kun tilgås igennem metodekald. Data og repræsenteringen af den bør være uafhængige af hinanden.

View betegner brugergrænsefladen og har ansvaret for selv at hente og præsentere data modellen. View fortolker dataindholdet af modellen, ved at fordele den på brugergrænsefladen.

Controller har ansvaret for at forbinde brugerens handlinger med ændringer i systemet. Controlleren fortolker brugerinput, udført i viewet og sørger for at foretage ændringerne direkte på modellen.

MVC bygger på, at en bruger, igennem View, styrer Controlleren, der laver ændringer på modellen. Når modellen rettes, opdateres viewet automatisk.



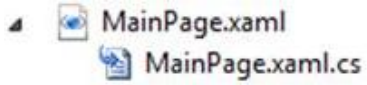
Figur 7.2: Illustration af Model-View-Controller konceptet. Sort streg betyder direkte association. Prikket streg betyder indirekte association

7.3.2 Model-View-ViewModel (MVVM)

Når man skal designe brugergrænsefladen i Silverlight, er man nødsaget til et bruge deklareringssproget XAML[Mice]. XAML beskriver de objekter, der skal vises i Viewet.

Når man skal opbygge en applikationsstruktur i Silverlight, opstår der et problem. Silverlight er lavet på en måde, så viewet har mulighed for at styre sig selv.

Silverlights brugergrænseflade er defineret igennem en XAML-fil, der deklarerer objekter i brugergrænsefladen. XAML-filen har en partnerfil XAML.cs. Partnerfilen indeholder den kode, der styrer objekterne. Når man opretter et nyt Silverlight projekt i Visual Studio 2010 opsættes begge filer automatisk.

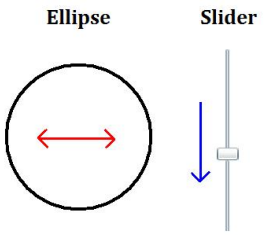


Figur 7.3: XAML filstruktur

På grund af denne ansvarsfordeling, vil View-laget og Model-laget være stærkt forbundet i en Silverlightapplikation. Dette kan give problemer[Lik], som systemet udvikler sig.

I 2005 kom Microsoft arkitekten *John Gossman* med et forslag til en ny softwarearkitektur. Arkitekturen var rettet mod både Silverlight- og WPF-applikationer. For at forstå konceptet bag arkitekturen er det vigtigt at have kendskab til featuren *Binding*.

Når en attribut på et objekt er sat som en binding, betyder det at værdien læses på en fjern variabel. Man kan også sætte en binding mellem to attributer på to objekter. Hvis man eksempeltvis sætter en binding mellem værdien på en "Slider", til bredden på en "Ellipse", vil Ellipsens bredde automatisk ændres, når slideren bruges.



Figur 7.4: Eksempel på en binging mellem slider-værdien, og ellipse-bredden.

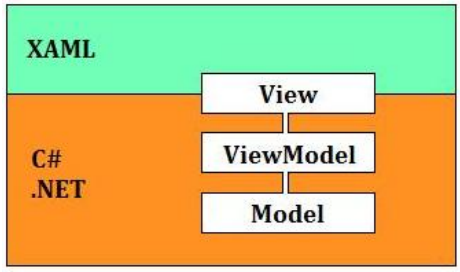
Det er også muligt at sætte bindings mellem attributer på objekter og variabler i klasser. Hvis man sætter en binding til en variabel i en klasse, giver det nye muligheder for at styre viewlaget. Det vil være muligt at opsætte en helt ny programstruktur.

På baggrund af denne forudsætning opsættes arkitekturopdelingen Model-View-ViewModel (MVVM)[Lik]. Arkitekturen er opdelt i tre koncepter.

Model er uændret i den generelle forstand, hvor modellen repræsenteres som et objek og bestemmer datagrundlaget for systemet.

View forekommer også uændret i den generelle forstand, og fortolker en model i viewet.

ViewModel er et nyt koncept, der beskriver en "model af view". ViewModel er en abstraktion af den rigtige model og reflekterer de ændringer, der foretages på modellen. ViewModel kan opfattes som en fortolkningsmodel.



Figur 7.5: Overblik af Model-View-ViewModel arkitekturen

MVVM er en skalerbar arkitektur, som betyder, at kompleksiteten ved at udvikle systemet, ikke stiger med systemets størrelse. Arkitekturen kræver dog en omfattende opsætning.

7.3.3 Designkonklusion

Til et projekt af min størrelse, som samtidig er en prototype, føler jeg ikke, at MVVM arkitekturen er en nødvendighed. Istedet vil jeg bruge en simpel MVC arkitektur.

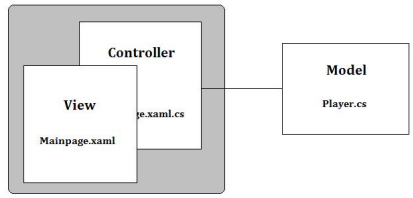
Sektion 7.3.2 forklarer, at der altid hører en partnerfil til en XAML-fil. Partnerfilen håndterer de definerede objekter.

Dynamikken mellem de to filer fungere på følgende måde.

Hvis vi antager, at en bruger trykker på en knap *vælgBtn*, vil en EventHandler *Click* blive håndteret. Man kan definere, hvad der skal ske ved at oprette en

metode og tilføje metoden til Click. Metoder der reagerer på events, placeres som standard i partnerfilen.

Jeg vil opdele min MVC-arkitektur ved at lade XAML-filen definere View, og partnerfilen definere Controller. Til at definere Model opretter jeg klassen *Player*. Denne klassen kræver en struktur, der kan opbevare og behandle togsætdata.



Figur 7.6: Ansvarsfordeling i MVC-arkitektur

Sektion 7.3.2 beskrev problemet ved at opbygge arkitekturer i Silverlight. Jeg er klar over, at ved at have View- og Controller-laget så tæt koblet, kan det give anledning til problemer. Samtidig kan det i små projekter være en fordel at have en monolitisk opbygning af strukturen, hvor systemets elementer kan tilgå hinanden mere direkte.

7.4 Silverlightapplikationen

Denne sektion beskriver, hvordan systemet er designet. Jeg har valgt at lave designet ud fra de tre opdelinger af arkitekturen.

Jeg vil først bestemme View-delen ved at bestemme et generelt design for brugergrænsefladen. For at designet kan implementeres, vil jeg bestemme de Silverlight-objekter, der stilles til rådighed.

Jeg vil beskrive Model-delen ved at udarbejde et design for klassen Player. Designet udgør et klasse hieraki på baggrund af domænemodellen.

Designet af Controlleren tager udgangspunkt i brugerens handlinger, og forbinder brugerinput til at redigere Model. Jeg vil beskrive hvilke metode, der skal opnå dette.

7.4.1 View

Brugergrænsefladen bør afspejle de krav, som er sat for brugbarhed. Vi husker fra sektion 5.4.2 at

"*Brugergrænseflade skal have et klart design, der gør det let for en bruger at styre en afspilning. Der skal være orden på de visuelle elementer, som en bruger kan styre. Brugergrænsefladen skal beskrives med vejledende tekst, de steder hvor det er nødvendigt.*".

Jeg har valgt at opdele mit design af brugergrænsefladen ud fra følgende 5 områder.

Verdenskort: Et kort, der grafisk kan vise verden igennem en web-mapping service.
Afspilninger: En liste af oplagringsmodeller, som man kan opsætte til en afspilning.
Tog-Fokus: En liste af togsæt i oplagringsmodellen, som man kan sætte i fokus.
Event-Liste: En liste, der viser events for det togsæt, der er i fokus.
Afspiller: Et kontrol-modul, hvor man kan styre afspilningen.

Jeg har lavet et design af brugergrænsefladen og besluttet hvilke objekter, der skal udgøre områderne. Områdernes placering i designet er bestemt ud fra min forestilling om et brugervenligt design.

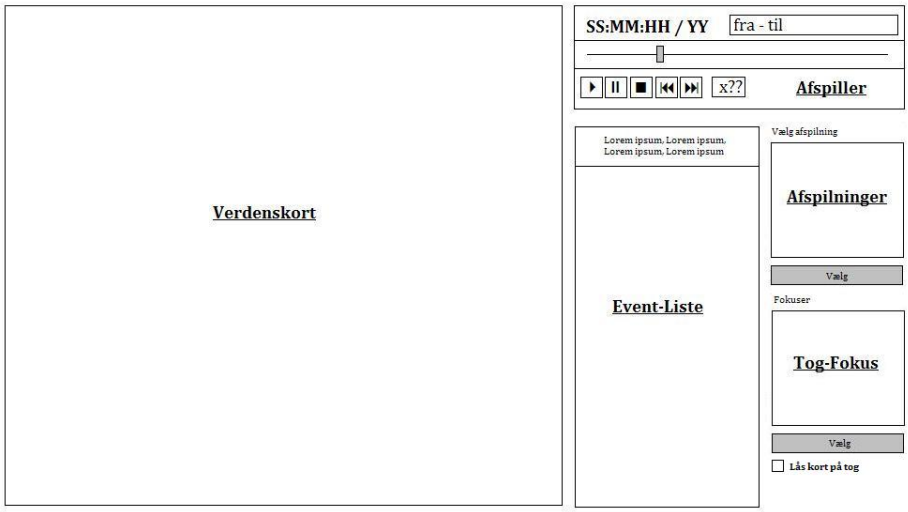
7.4.1.1 Verdenskort

Til at illustrere verdenskortet bruges Bing Maps, som er udviklet af Microsoft. Et andet populært alternativ er Google Maps, men der er endnu ikke udviklet et stabilt bibliotek, der kan integrere kortet i Silverlight. Bing Maps derimod, er forholdsvist simpelt at integrere.

Verdenskortet er et enkelt objekt i brugergrænsefladen.

7.4.1.2 Afspilninger

Dette område fungerer som en liste, der kan vise de tilgængelige afspilninger. Jeg har valgt at tilføre dette område til mit projekt som alternativet til at kommunikere med back-end delen, der endnu ikke er færdigudviklet.



Figur 7.7: Brugergrænseflade-design

Når back-end delen implementeres, er det meningen, at den skal kunne kontaktes direkte og efterspørge en oplagringsmodel. Indtil da vil man kunne vælge en afspilning, ved at markere den i listen, og trykke på knappen *Vælg*.

Oven over listen er der placeret en tekststreng, der viser "Vælg afspilning".

7.4.1.3 Tog-Fokus

Objekterne i dette område fungerer på funktionelt samme måde, som *Afspilninger*. Tog-Fokus indeholder en liste af de togsæt, der eksisterer i afspilningen. Ved at markere et togsæt i listen, kan man bruge knappen *Vælg* til at sætte et togsæt i fokus.

Når et togsæt er i fokus opsættes området *Event-List*. Da vil det også være muligt at sætte flueben i den lille boks under knappen *Vælg*. Når fluebenen er sat, vil det billede, som verdenskortet viser, låses fast til togsættets færden på kortet.

Oven over listen er der placeret en tekststreng, der viser "Fokuser".

7.4.1.4 Event-Liste

Dette områder indeholder en liste, der viser oplagringsmodellen for det togsæt, som er i fokus. Elementerne i listen repræsenterer hændelsen og viser omstændighederne ved den.

Når en afspilning er startet, og et togsæt er sat i fokus, skal listen markere de seneste hændelser. Det element i listen, der repræsenterer den seneste hændelse, vil bliver markeret med en gul baggrund. Det er meningen, at listen automatisk skal opdatere markeringerne.

7.4.1.5 Afspiller

Afspilleren består af flere objekter, der til sammen styrer afspilningen. Personligt er jeg meget glad for en gratis multimedie afspiller, som hedder "Winamp". Jeg har forsøgt at kopiere winamp-designet til min egen afspiller.

Øverst i afspilleren viser en tekstreng tidspunktet for afspilningen. Tekststrengen opdateres konstant og viser derfor altid det tidspunkt, som fortolkes på *Verdenskortet*. Afspilleren bruger en anden tekststreng til at vise, hvor afspilninger strækker sig fra og til (tidsgrænser).

I midten af afspilleren er placeret en Slider, som består af en nål, der kan trækkes over et spor. Sporet i Slideren afspejler tidsstrækningen mellem afspilningens tidsgrænser. Nålens placering på sporet bestemmer det tidspunktet i afspilningen, der fortolkes.

For at implementere UC6 skal det være muligt at klikke et sted på Slider-sporet. Når der klikkes i slider-sporet, skal afspilningen fortsætte fra det sted i afspilningen som positionen i sporet repræsenterer. Det skal også være muligt at venstreklikke på Slider-nålen. Når nålen er aktiveret skal den kunne trækkes til et nyt sted i sporet, give slip, og fortsætte afspilningen derfra.

Afspilleren har 5 knapper, som hver udfører en funktion i afspilningen. Hver knap bruges til at udføre en forskellig use case. Knapperne til afspilleren har følgende formål:

Play: Denne knap sætter afspilningen igang og fortolker den igennem Viewet. Til at vise afspilningen bruges Verdenskortet, Event-Listen (såfremt et togsæt er i fokus), og Afspilleren. Playknappen bruges til at udføre UC2.

Pause: Denne knap sætter afspilningen på pause og sørger for at afspilningen ikke længere fortolkes. Hvis afspilningen i forvejen er sat på pause, når knappen bliver brugt, vil afspilningen fortsætte ved tidspunktet, som slider-nålen repræsenterer. Pauseknappen bruges til at udføre UC4.

Stop: Stopknappen nulstiller afspilningen, ved at sætte slider-nålens værdi til afspilningens begyndelsestidspunkt. Når stopknappen benyttes vil viewet ikke længere opdateres. Stopknappes bruges til at udføre UC3.

Spol frem: Denne knap får hastigheden på afspilningen til at stige. Hver gang der laves en ændring på hastigheden, opdateres teksten i den lille boks ved siden af knapperne. Spol-frem knappen bruges til at udføre UC5.

Spol tilbage: Denne knap får hastigheden på afspilningen til at blive sænket. Hver gang der laves en ændring på hastigheden, opdateres teksten i den lille boks ved siden af knapperne. Spol-tilbage knappen bruges til at udføre UC5.

Ved siden af de fem knapper er en boks, der viser hastigheden på afspilningen. Hastigheden "x1" svarer til at afspilleren viser 1 sekund af afspilningen per 1 sekund. Hastigheden "x60" svarer til at afspilleren viser 60 sekunder af afspilningen per 1 sekund.

7.4.1.6 Controls

For at bygge og implementere en brugergrænseflade i Silverlight skal man bruge de rigtige objekter. I WPF og Silverlight er disse objekter defineret som "Controls" [Micd]. Et control er en samling af elementer, der kan bruges til at udføre brugerhandlinger. Et typsik eksempelt på et control er en knap.

Controls er opbygget med EventHandlere, som håndterer de events, der opstår igennem brugerhandlinger. Silverlight stiller de fleste controles til rådighed, men det er også muligt at designe og opbygge sin egen.

De følgende tabeller beskriver hvilke controls, der skal bruges i applikationen. Tabellerne beskriver også hvilke EventHandlere, der stilles til rådighed.

ListBox	
Beskrivelse	En liste der kan repræsenterer elementer og er opbygget med en vertical slider. Slideren kan rulles, og en bruger kan derfor søge i listen efter et element. Elementerne i listen kan designes til at vise stort set alle elementer og controls.
Bruges af	Afspilninger Tog-Fokus Event-Liste
EventHandlere	
MouseLeftButtonDown	Dette event kaldes, når en bruger trykker venstre musseknap ned. Eventet blive kun håndteret i Tog-Fokus ListBoxen.

Button	
Beskrivelse	En knap som brugeren kan trykke på. Knappen kan indeholde tekst og billede.
Bruges af	Afspilninger Tog-Fokus Afspiller
EventHandlere	
Click	Denne event opstår når en bruger venstreklikker på knappen.

Slider	
Beskrivelse	En slider består primært af to rektangler og en nål. Nålen er placeret i midten af de to rektangler, og placeringen af nålen bestemmer bredden på de to rektangler. Man kan bestemme en ny værdi for nålen ved at trække og slippe den, men man kan også gøre det ved at trykke et sted på et af rektanglerne.
Bruges af	Afspiller
EventHandlere	
DragStarted	Dette event kaldes når en bruger trykker på nålen med venstre musseknap.
DragCompleted	Dette event kaldes når en bruger slipper venstre musseknap, efter at nålen blev trukket.
MouseLeftButtonDown	Dette event kaldes når en bruger trykker venstre musseknap ned på en af de to rektangler.

CheckBox	
Beskrivelse	En checkbox er et felt som en bruger kan krydse af ved at klikke på det, med venstre musseknap. En checkbox kan afkrydses ved at man trykker en gang mere.
Brugt af	Tog-Fokus

EventHandlere	
Checked	Kaldes når CheckBox bliver krydset.
Unchecked	Kaldes når CheckBox bliver afkrydset.

Label	
Beskrivelse	Et element, der repræsenterer en tekststreng i brugergrænsefladen.
Brugt af	Afspilninger Tog-Fokus Afspiller

Map	
Beskrivelse	Map er et Control der stilles til rådighed igennem Bing Maps biblioteket. Map opsættes som et element i brugergrænsefladen, og forbinder Silverlightapplikationen til Bing Maps web-mapping service. Map kræver en nøgle for at virke som skal tildeles fra Microsofts hjemmeside.
Brugt af	Verdenskortet

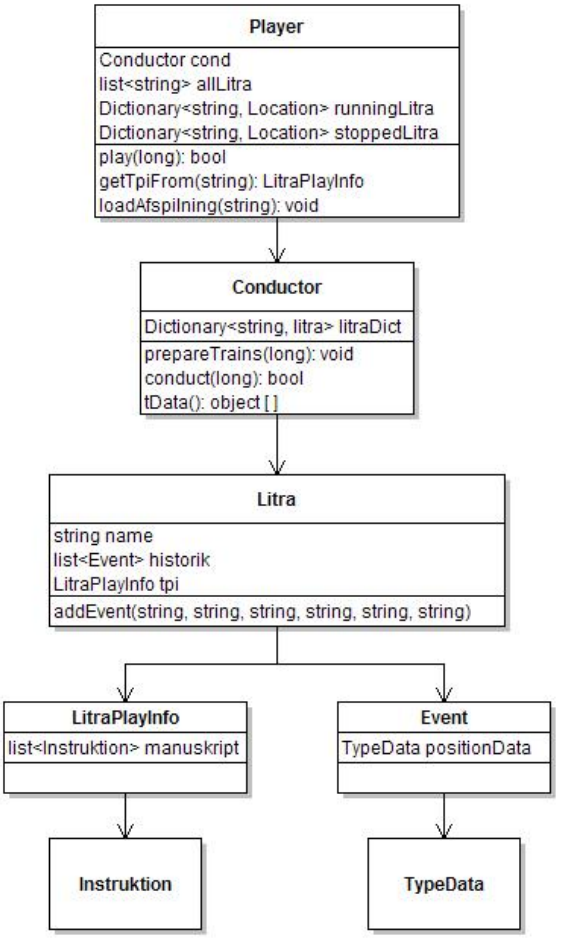
7.4.2 Model

Klassehierakiet skal repræsentere en kongruent opbygning af det data, der skal fortolkes oplagringsmodellen.

Det et krav, at en bruger skal kunne hoppe og spole i en afspilning. Hvis alt data i modellen skal fortolkes, kan systemet nemt blive overbelastet. Klassehierakiet er opbygget ud fra to forestiller omkring systemet:

- Når afspilleren skal fortolke et nyt tidspunkt, skal den ikke fortolke det "næste" billede. Når viewet er opdateret, skal det næste genereres dynamisk. Model skal fortolkes ud fra det tidspunkt, som afspilningen er nået til. På denne måde undgår afspilningen at overbelaste systemet.

- Hvis systemet skal lave alle udregninger direkte på oplagringsmodellen hver gang, der skal animeres et billede, vil det kræve unødvendigt mange ressourcer. Det er derfor oplagt at forberede de fleste udregningerne, som en opslagsmodel. Resultatet af udregningerne skal opfattes som et manuskript, der beskriver, hvordan hændelserne for et togsæt skal fortolkes.



Figur 7.8: Klassediagram for modellen

Lad os betragte klasserne i diagrammet, og beskrive deres formål i systemet.

Player Player klassen inderholder klassehierakiet. Formålet med klassen er at

fremskaffer data til viewet, når det efterspørges.

Player klassen opsættes ved hjælp af oplagringsmodellen igennem metoden *load-Afspilning*. Metoden indlæser oplagringesmodellen som en liste af klassen Event, og opsætter den i klassen Conductor.

Player besider følgende variabler.

cond:	Et objekt af klassen Conductor.
allLitra:	En liste der beskriver navnene på alle togsæt.
runningLitra:	En dictionary der beskriver de togsæt, der skal vises på Verdenskortet.
stoppedLitra:	En dictionary der beskriver de togsæt, der ikke skal vises på Verdenskortet.

En dictionary er en klasse, der kan bruges til at opbevare data og fungerer på samme måde som et Hashtable. Man kan tilgå en værdi ved at bruge en nøgle. Nøglen til de to dictionaries i Player er togsæts ID, og værdien er et koordinat på kortet.

Når playeren skal fortolke et tidspunkt i afspilningen, kaldes metoden *play*. Når metoden returnerer vil *runningLitra*, og *stoppedLitra* være opdaterede.

Conductor

Denne klasse indeholder funktionaliteten til at fortolke afspilningen igennem beregninger. Conductor opsættes samtidig med Player og tildeles en dictionary *litraDict*. Denne dictionary indeholder data for hvert togsæt igennem klassen Litra.

Hvert Litra-objekt i litraDict beskriver sin egen oplagringsmodel igennem variablen *historik*. Når Conductor bliver opsat, fortolkes oplagringsmodellen ved at lave en opslagmodel. Opslagmodellen indeholder støttedata til at udføre animationer i viewet, og resultatet gemmes i klassen LitraPlayInfo under variablen *manuskript*.

Litra

Denne klasse repræsenterer alle data omkring togsæt, og besidder følgende variabler.

name:	Togsættes ID.
historik:	En liste der beskriver togsættes hændelser i oplagringsmodellen igennem klassen Event.
tpi:	Et objekt af klassen LitraPlayInfo, der beskriver omstændighederne ved at animere et togsæt på Verdenskortet. Klassen indeholder den fortolkede udgave af oplagringsmodellen *manuskript*. En dictionary runningLitra beskriver de togsæt, der skal vises på Verdenskortet.

7.4.3 Controller

Controllerdelen bestemmer hvordan View skal samarbejde med Model.

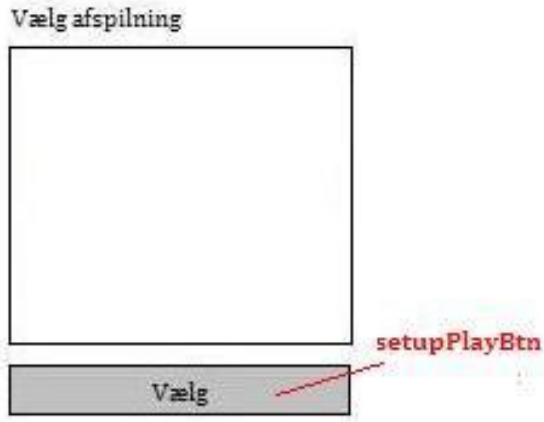
Vi husker, at Controlleren er defineret igennem XAML-filens partnerfil. Partnerfilen styrer objekterne i Viewet ved at definere de metoder, der reagerer på events.

De følgende tabeller beskriver, hvordan de forskellige områder i Viewet er forbundet til Controlleren. Tabellerne beskriver omstændighederne for alle objekter af hver controltype. Beskrivelse af Verdenskort-området er undladt, da den ikke er forbundet direkte til Controlleren.

Tabellerne beskriver følgende parametre:

Control beskriver hvilken controltype, der beskrives, for GUI-området.
Elementer beskriver navnene på controls af samme type, der er defineret i GUI-området.
Events beskriver de EventHandlers, der håndteres i af controltypen, i GUI-området.
Metoder beskriver navnene på de metoder, der håndteres af EventHandlerne.

7.4.3.1 Afspilninger



Figur 7.9: Objekter i Afspilninger delen

setupPlayBtn	
Control	Button
Event	Click
Method	setupPlayBtn_Click
Beskrivelse	Player klassen opsættes med oplagringsmodellen, og fortolker indeholdet til en historik for hvert togsæt. Alle historik elementer opsættes i en særklasse af Litra, og disse elementer opsættes til en dictionary. Player klassen opsætter Conductor med en den nye dictionary som dermed udarbejder manuskipter.

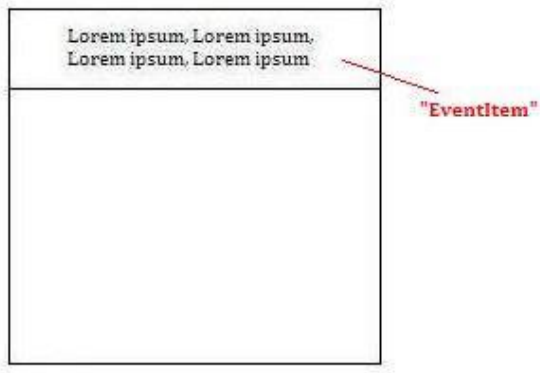
7.4.3.2 Tog-Fokus

Figur 7.10: Objekter i Tog-fokus delen

focusTrainBtn	
Control	Button
Event	Click
Method	focusTrainBtn_Click
Beskrivelse	Event-Listen opsættes med elementer fra listen historik for det togsæt, der er valgt.

lockViewCheckBox	
Control	CheckBox
Event	Checked
Method	lockViewCheckBox_Checked
Beskrivelse	Når CheckBox krydses af bliver kortet låst fast til det togelementet, der tilhører togsættet, som er sat i fokus. Billedet for Verdenskortet vil nu følge togelementet rundt på kortet.
Event	Unchecked
Method	lockViewCheckBox_Unchecked
Beskrivelse	Når CheckBox afkrydses vil kortet ikke længere være fastlåst til togsættet.

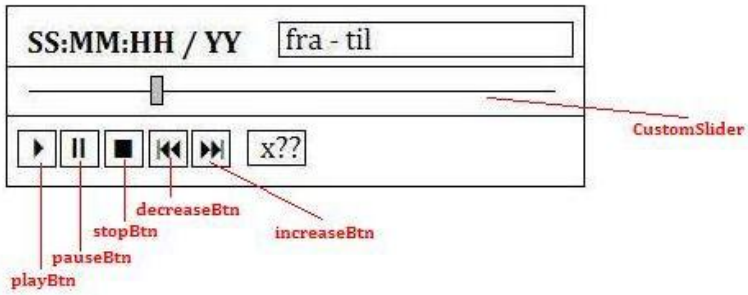
7.4.3.3 Event-Liste



Figur 7.11: Objekter i Tog-fokus delen

EventItem	
Control	ListBoxItem
Event	MouseLeftButtonDown
Method	eventItemClicked
Beskrivelse	Metoden registrerer, hvornår der sidst er klikket og, hvis to klik forekommer hurtigt efter hinanden anses det som et dobbeltklik. Når et element er dobbeltklikket, vil afspilleren fortsætte det sted i afspilningen som elementet repræsenterer.

7.4.3.4 Afspiller



Figur 7.12: Objekter i Afspiller delen

playBtn	
Control	Button
Event	Click
Method	playBtn_Click
Beskrivelse	Denne metode får afspilleren til at starte fra begyndelsen af afspilningen. Dermed aktiveres både Afspiller, og Verdenskort, samt Tog-Fokus, hvis et togsæt er sat i fokus.

stopBtn	
Control	Button
Event	Click
Method	stopBtn_Click
Beskrivelse	Afspilningen stopper helt og nulstilles til begyndelsestidspunktet.

increaseBtn	
Control	Button
Event	Click
Method	increaseBtn_Click
Beskrivelse	Hastigheden på afspilningen sænkes, og der vises færre sekunder af afspilningen pr. sekund.

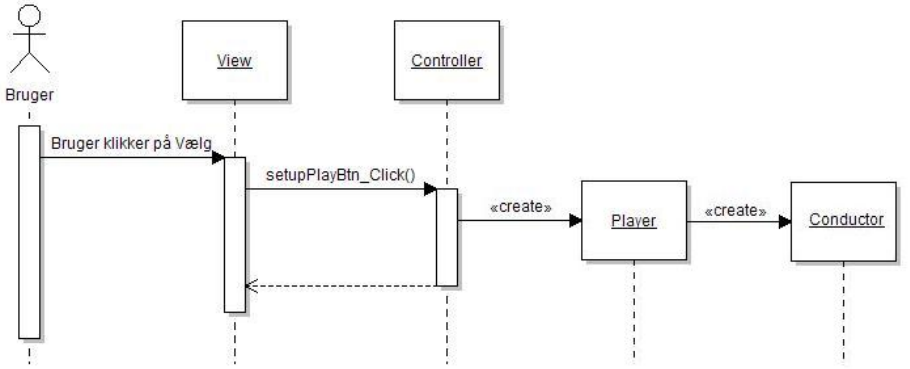
decreaseBtn	
Control	Button
Event	Click
Method	decreaseBtn_Click
Beskrivelse	Hastigheden på afspilningen hæves, og der vises flere sekunder af afspilningen pr. sekund.

CustomSlider	
Control	Slider
Event	DragStarted
Method	thumbDragStarted
Beskrivelse	Metoden kaldes når en bruger trykker på slider-nålen. Når denne metode kaldes, fortsætter afspilningen uændret, men den Label, der beskriver tidsgrænserne for afspilningen vil vises det tidspunkt, som svarer til slider-nåles position i slider-sporet.
Event	DragCompleted
Method	thumbDragCompleted
Beskrivelse	Denne metode håndteres når en bruger giver slip på slider-nålen igen. Metoden springer i afspilningen, og viser grænseværdierne på sin label igen.
Event	MouseLeftButtonDown
Method	moveToMouse
Beskrivelse	Metoden aktiveres når en bruger trykker et sted på Slider-sporet. Metoden flytter Slider-nålen hen til positionen, der er trykket på og afspilningen fortsætter herfra.

7.5 Sekvensdiagrammer

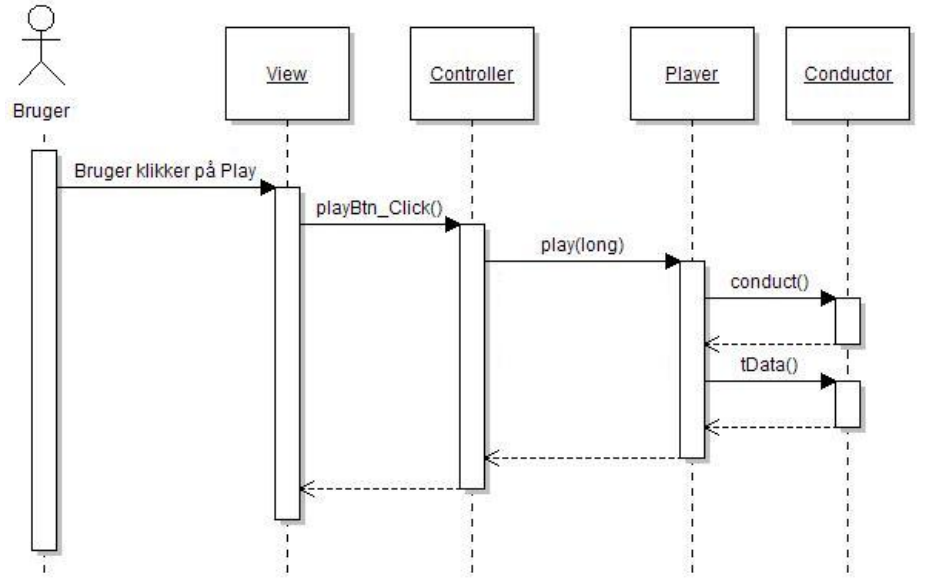
I dette afsnit vil jeg beskrive, hvordan mit system skal opføre sig internt, når en bruger udfører et use case. Til at beskrive den interne proces i systemet med UP, bruges sekvens diagrammer. Diagrammerne illustrerer brugerhandlinger og processer i systemet og rækkefølgen på dem.

7.5.0.5 UC1 Vælg afspilning



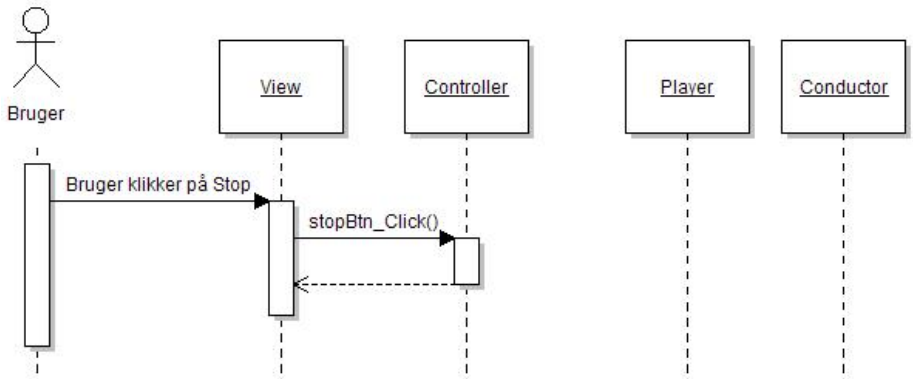
Figur 7.13: Sekvensdiagram for UC1

7.5.0.6 UC2 Start afspilning



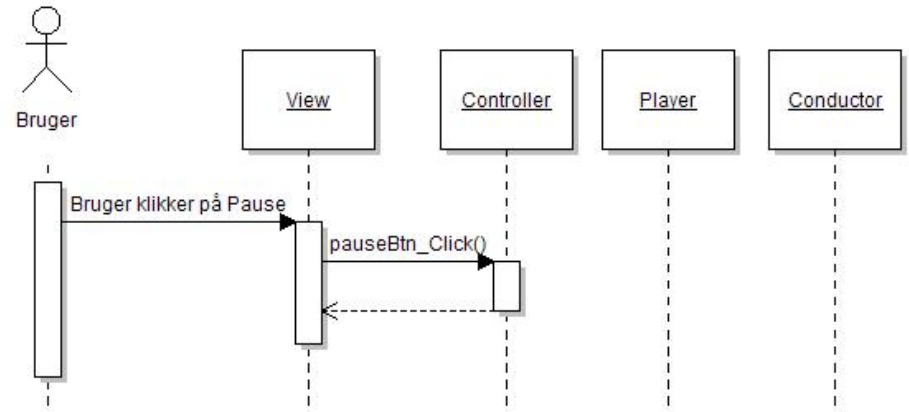
Figur 7.14: Sekvensdiagram for UC2

7.5.0.7 UC3 Stop afspilning



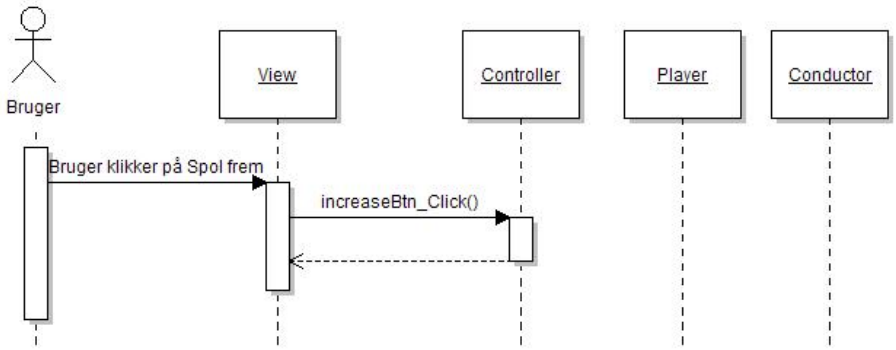
Figur 7.15: Sekvensdiagram for UC3

7.5.0.8 UC4 Pause afspilning



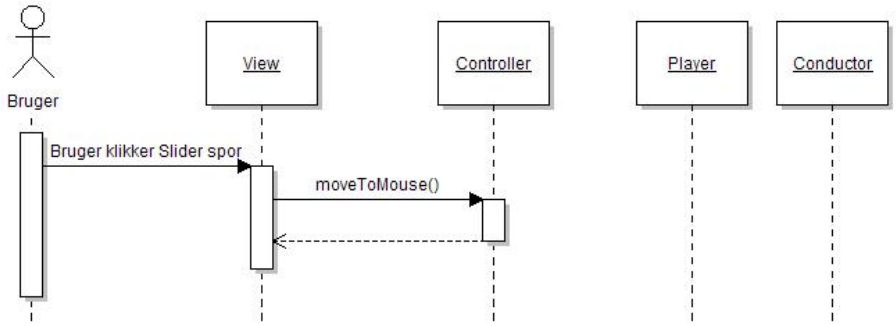
Figur 7.16: Sekvensdiagram for UC4

7.5.0.9 UC5 Spol i afspilning



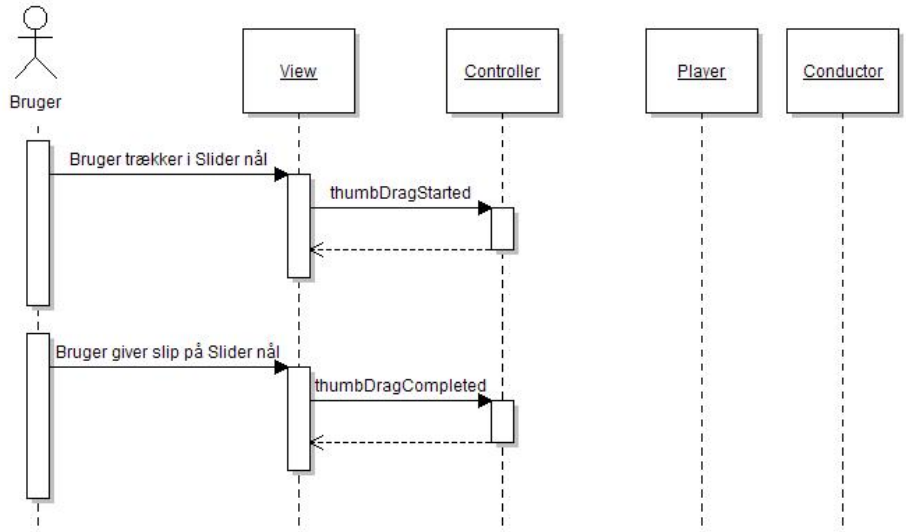
Figur 7.17: Sekvensdiagram for UC5

7.5.0.10 UC6 Spring i afspilning



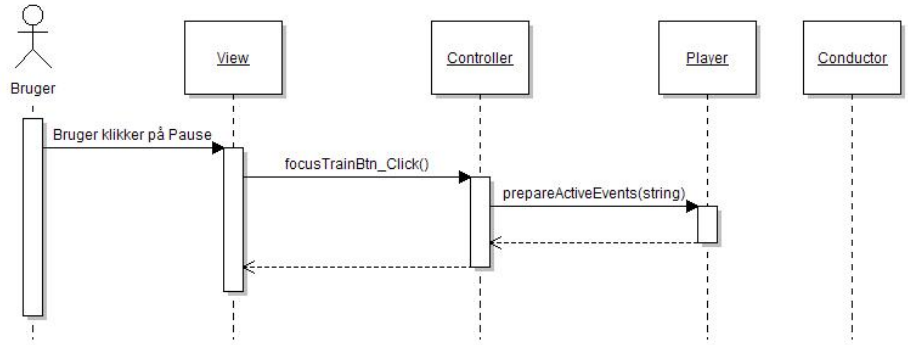
Figur 7.18: Sekvensdiagram for UC6

7.5.0.11 UC6 Spring i afspilning (alternativ)



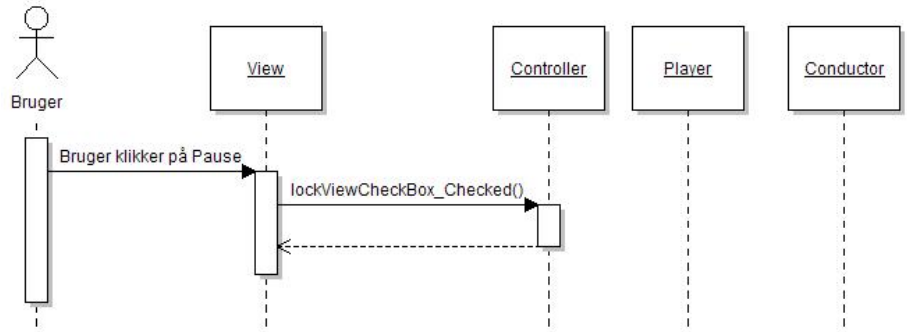
Figur 7.19: Sekvensdiagram for UC6, alternativ version

7.5.0.12 UC7 Fokuser på tog



Figur 7.20: Sekvensdiagram for UC7

7.5.0.13 UC8 Lås afspiller til tog



Figur 7.21: Sekvensdiagram for UC8

7.6 Delkonklusion

Jeg har i design-fasen udarbejdet et design for hele den afgrænsning af løsningen som mit projekt fokuserer på. Strukturen blev lavet ud fra system-arkitekturen MVC. Jeg har designet arkitekturen i tre dele og defineret, hvordan de er forbundet.

Til slut har jeg beskrevet den interne fremgangsmåde i systemet for alle use cases ved at bruge sekvensdiagrammer.

Design forløbets afslutning betyder, at alle komponenter, der skal bruges i arkitekturen, er forberedt, og kan derfor nu blive implementeret.

Kapitel 8

Implementering

8.1 Forord

Dette kapitel beskriver, hvordan systemet er implementeret. Kapitelet redegør for arbejdsforløbet, som er defineret i sektion 4.2.3.4.

Metoden til at animere en afspilning begrænser den måde, som prototypen kan sammensættes. Kapitlet beskriver den valgte animerings-metode, og beskriver det implementere system ud fra systemarkitekturens opdeling.

8.2 Animering i Silverlight

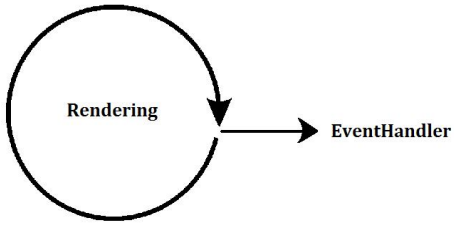
Skal man lave sin egen animation i Silverlight, er der to metoder at gøre det på[Smi]. Den første metode bruger en klasse, der hedder DispatcherTimer, og den anden metode udnytter en EventHandler, der håndterer *frame-rendering*. Jeg har valgt at lave animationer på baggrund af den sidstnævnte metode.

DispatcherTimer er en klasse man kan opsætte til at sammarbejde med programmets dispatcher. Dispatcheren er en mekanisme i programmet, der bestem-

mer sammarbejdet, mellem tråde og processer i systemet.

Ved at opsætte et interval i DispatcherTimer-klassen, kan man få dispatcheren til automatisk at vende tilbage for at udføre en Callback-metode og derefter gå i dvale indtil intervallet slutter. Jeg har fravalgt denne metode, fordi den stiller særlige krav til dispatcheren, og kodemæssigt kræver større kontrol.

Den anden metode at lave animationer på, udnytter en **frame-rendering EventHandler**. Hver gang at Silverlight skal opdatere et billede i grafikken (et frame), vil en EventHandler *Rendering* blive håndteret. Grafikken i Silverlight bliver konstant opdateret, og derfor vil *Rendering* hele tiden blivet håndteret.



Figur 8.1: EventHandleren håndteres hver gang grafikken opdateres.

EventHandlere kan håndtere flere metoder, når der sker en Event, og jeg kan derfor tilføje en brugerdefineret metode til Rendering, som vil blive håndteret sammen med grafikken.

8.2.1 Fremgangsmåde for Animering

Oplagringsmodellen beskriver GPS-koordinaterne til det sted, hvor der blev foretaget en positionsmelding. Til at animere togrejsen for prototypen har jeg valgt kun at animere skiftet fra et koordinat til et nyt koordinat. Dette betyder, at et togsæt vil blive animeret i en lige strækning, og derfor springer skinnerne lidt over nogle steder.

Figur 8.2: Animering mellem to koordinater, der ignorerer skinner.

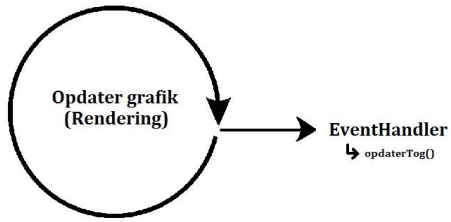
Jeg skal udføre en animation mellem to koordinater igennem en metode, der konstant bliver håndteret og gradvist ændre animationen. For at foretage ændringen i animationen, er jeg nødsaget til at vide følgende detaljer.

- Hvordan skal animationen starte og slutte?
- Hvor lang tid skal animationen vare?
- Hvad tidspunkt påbegyndes animationen?
- Hvor langt tid er der gået efter påbegyndelsen?

Det er muligt at tilføje et element på Bing Maps kortet, der kan repræsentere et togsæt og få det vist på et GIS-koordinatsæt. Bing Maps biblioteket bruger klassen *Location* til at repræsentere longitude og latitude. Hvis man sætter et element til at nedarve denne klasse, ved at tilføje den som ejendom, vil Verdenskortet vise elementet på det givne koordinat.

For at elementet kan bevæge sig flydende på kortet, skal man opdatere værdierne for *Location* klassen på elementet. Først fjernes elementet fra kortet, dernæst ændres værdierne i *Location*, og til slut tilføjes elementet igen. Denne handling kan udføres på kortet flere tusind gange i sekundet, uden det bemærkes i brugeroplevelsen.

Lad os antage, at vi vil animere et togsæt med metoden "opdaterTog()", ved at benytte EventHandleren *Rendering*. Vi erstatter GIS-koordinater med lave tal for forståelsens skyld.



Figur 8.3: EventHandleren håndteres hver gang grafikken opdateres

Et togsæt er placeret på position (5, 0) skal til position (10, 5).
Vi ved derfor, at toget skal animeres +5 vandret, og +5 lodret i forhold til et koordinatsystem.
Animationen starter ved systemtiden 0, og skal slutte ved systemtiden 100.

Hvis systemtiden er 50, kan vi udregne koordinatet for togsættet på følgende

måde.

$$\text{Tid gået} = (50 * 100\%) / 100 = 50\%$$
$$\text{Vandret afstand} = (5 * 50\%)/100\% = 2.5$$
$$\text{Lodret afstand} = (5 * 50\%)/100\% = 2.5$$

Togets position ved tidspunkt 50 vil være (2.5, 2.5).

8.2.2 Systemflow

I denne sektion vil jeg beskrive, hvordan systemet skal udnytte systemarkitekturen til at implementere brugeroplevelsen. Jeg vil beskrive min strategi for at behandle afspilningen igennem brugerens handlinger i GUI'en.

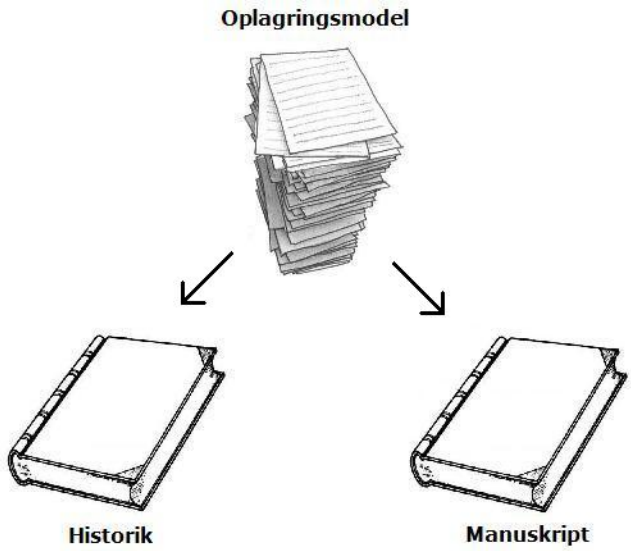
8.2.3 Datagrundlag for animering

Jeg har bestemt strukturen i oplagringsmodellen, og jeg har bestemt hvordan GUI'en skal opføre sig. Systemet har brug for en metode, der dynamisk fortolker indholdet af oplagringsmodellen.

I sektion 7.4.2 bestemte jeg, at man kan opsætte en opslagsmodel *Manuskript* af oplagringsmodellen for at at spare ressourcer.

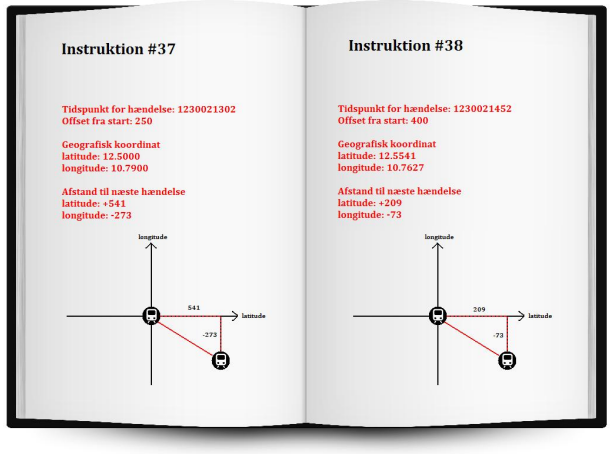
Hvert togsæt i systemet har derfor brug for følgende modeller.

Historik, som beskriver indholdet af opslagmodellen, og
Manuskript, som er en fortolkning af opslagsmodellen og indeholder opslag, til udregningerne af afspilningen.



Figur 8.4: Manuskript fortolker Historik, til en opslagsmodel.

Manuskriptet er en liste af klassen Instruktion, der beskriver omstændighederne ved en hændelse, og indeholder data til at udregne positionen for et togsæt mellem koordinaterne for to hændelser.



Figur 8.5: Manuskriptet indeholder instruktioner for afspilningen.

Begge modeller er en del af klassen Player, som beskrevet i sektion 7.4.2. His-

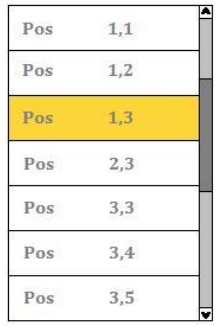
torik vil være en del af klassen Litra, og Manuskript være en del af klassen LitraPlayInfo.

8.2.4 Fremgangsmetode for animering

Når afspilningen startes opsætter en variabel, der noterer hvor længe afspilningen har været igang. Tidspunktet opdateres konstant, og kan derfor bruges til at opdatere kortet så ofte som systemet kan følge med.

Vi husker, at EventHandleren *Rendering* håndteres hver gang at grafikken opdateres, og vi kan derfor tilføje en metode til at håndtere animeringen af Verdenskortet og styringen af Afspilleren. Metoden der skal udføre dette, kalder jeg *updatePlay()*.

Når et togsæt bliver sat i fokus kræves det, at Event-Listen bliver opsat med data fra historikken. For at gøre listen brugbar, skal den fremhæve det element i listen, der repræsenterer den hændelse, der sidst er udført.



Figur 8.6: Event-listen markerer det det sidste hændte event.

Event-Listen er kun aktiveret, når et togsæt er i fokus, og derfor har jeg valgt at konstruere en ekstra, der skal håndteres af *Rendering*, når listen blive aktiveret. Denne metode kalder jeg *manageEvents()*.

8.3 Implementering af View

Implementeringen af view foregår primært igennem filen MainPage.xaml. Elementer der deklareres i XAML filen kan også tilføjes igennem partner-filen

MainPage.cs, som i protypen fungerer som Controller. Jeg vil i denne sektion beskrive, hvordan de forskellige områder i GUI-designet er blevet deklareret.

Indholdet i en GUI'en for en Silverlight applikation defineres ofte ud fra et Canvas, som er er et kvadrat, der kan indeholde elementer eller controls. Når man indsætter et element i GUI'en, kan man bestemme positionen på elementet ud fra relationen til et canvas. De fleste Applikationer bruger et foreslået canvas*LayoutRoot* til at danne en ramme omkring hele applikationen.

8.3.1 Verdenskort

For at integrere Bing Maps kortet i en Silverlight applikation, skal man først referere til de to biblioteksfiler i Visual Studio-projektet. Når referencen er opsat, kan man i XAML koden referere til biblioteket med følgende kode.

Kodestykke 8.1: Definering af Bing Maps bibliotek i projektet.

```
1     xmlns:m="clr-namespace:Microsoft.Maps.MapControl;assembly=
          Microsoft.Maps.MapControl"
```

Koden bestemmer en reference til et namespace (xmlns), igennem variablen "m". Kortet er indsat i viewet med følgende kode.

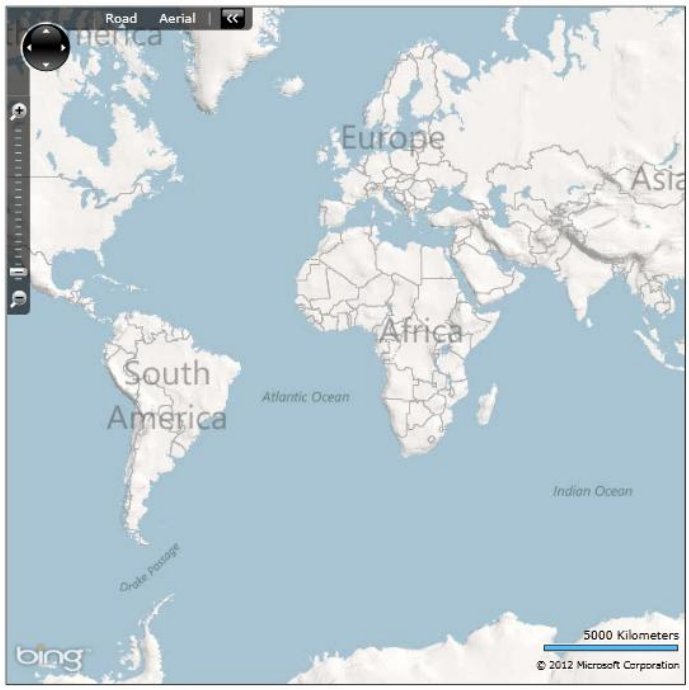
Kodestykke 8.2: Deklarering af Bing Maps i GUI.

```
1     <m:Map x:Name="BingMap" Mode="Road" Height="640" Width="640"
          Margin="0, 0, 0, 0" UseInertia="True"
2             CredentialsProvider="AgLMIj1RqV-
                  GNzkIoaUn84oV3QRu6TAd4XVne_KihohBYH2KHK-
                  UTXZKaTUoeJFh" >
3         <m:MapLayer x:Name="trainLayer"/>
4     </m:Map>
```

Bemærk at kortet er defineret igennem navnet "BingMap", og CredentialsProvider bestemmer en lang kode, der fungerer som kredentialer til at få adgang til Bing Maps serveren. Mappet ejer et MapLayer objekt, ved navn "trainLayer", som bruges til at opbevare elementer, der kan vises på kortet.

8.3.2 Afspilninger

Området Afspilninger havde behov for en ListBox, et Label, og en Button. Hvert control er defineret med position i forhold til canvas og et unikt navn. Control'et



Figur 8.7: Den implementerede udgave af Verdenskortet.

setupPlayBtn definerer også attributen "Click", som beskriver hvilken metode, der håndterer EventHandleren *Click*.

Kodestykke 8.3: Deklarering af Afspilninger i GUI.

```
1     <ListBox Canvas.Left="908" Canvas.Top="199" Height="51" Name="
          playsListBox" Width="141" />
2     <dataInput:Label Canvas.Left="908" Canvas.Top="178" Height="17"
          Name="selectPlayTxt" Width="100" Content="Vælg Afspilning"
          />
3     <Button Canvas.Left="908" Canvas.Top="257" Content="Vælg" Height
          ="23" Name="setupPlayBtn" Width="141"
4     Click="setupPlayBtn_Click" />
```

8.3.3 Tog-Fokus

Dette område fungerer på samme måde, som området Afspilninger. Der defineres en Label, som beskriver området, og en Button man kan trykke på, for

Figur 8.8: Den implementerede udgave af Afspilninger.

at sætte et togsæt i fokus.

Figur 8.9 viser listen Tog-Fokus efter en afspilning er valgt, og listen repræsenterer ID for afspilningens togsæt. Når et togsæt er valgt, kan CheckBoxen bruges til at fastlåse kortet til et togelement og på den måde udføre UC8.

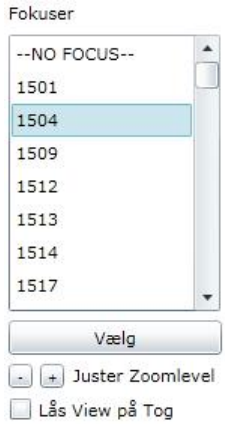
Det har vist sig, at det ikke er muligt at bruge zoomfunktionen på Bing Maps kortet, når billedet er fastlåst på et togsæt. For at komme rundt om problemet, har jeg tilføjet to knapper til at zoome ind og ud på kortet, imens det er fastlåst. Knapperne ændrer på en intern variabel i systemet, som refereres til af kortet, når det låses fast. Knapperne har fået navnene zoomIncreaseBtn og zoomDecreaseBtn.

Kodestykke 8.4: Deklarering af Tog-Fokus i GUI.

```
1 <CheckBox Canvas.Left="908" Canvas.Top="550" Content="Lås View på
      Tog" Height="16" Name="lockViewCheckBox" Checked="
      lockViewCheckBox_Checked" Unchecked="lockViewCheckBox_Unchecked"
       />
2 <Button Canvas.Left="929" Canvas.Top="528" Content="+" Height="16"
      Name="zoomIncreaseBtn" Width="16" FontSize="8" Click="
      zoomIncreaseBtn_Click" />
3 <Button Canvas.Left="908" Canvas.Top="528" Content="-" FontSize="8"
      Height="16" Width="16" Name="zoomDecreaseBtn" Click="
      zoomDecreaseBtn_Click" />
4 <dataInput:Label Canvas.Left="951" Canvas.Top="528" Height="16" Name
      ="adjustZoomLevelLabel" Width="98" Content="Juster Zoomlevel" />
```

8.3.4 Event-Liste

Event-Listen har til formål at vise indholdet af historik for det togsæt, der er sat i fokus. Når der opstår en hændelse for det togsæt, der er i fokus, skal elementet, der svarer til hændelsen blive markeret. Det skal også være mulig at dobbeltklikke på et element i listen og dermed springe i afspilningen.



Figur 8.9: Den implementerede udgave af tog-fokus.

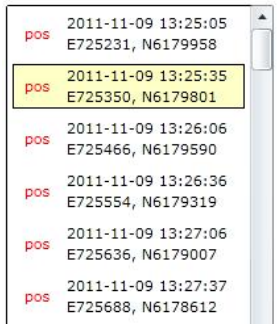
For at tilfredsstille disse krav i Silverlight er man nødsaget til at designe elementerne til ListBoxen. Når ListeBoxen defineres i XAML koden, skal man også definere en skitse for, hvordan elementerne (ListBoxItem) skal repræsenteres. Jeg har defineret to egenskaber i koden, ItemContainerStyle og ItemTemplate.

ItemContainerStyle sætter egenskaber for, hvordan listen skal behandle indholdet, og ItemTemplate bestemmer, hvordan indholdet skal repræsenteres. Når et element i listen skal markeres, skal en sort kant rundt om elementet gøres synlig, mens baggrunden sættes til en gul farve (#FFE88D34).

Kodestykke 8.5: Deklarering af Event-Liste i GUI.

```
1  <ListBox Canvas.Top="178" Canvas.Left="699" Height="514" Name="
       eventListBox" Width="203"
2            BorderBrush="Black" BorderThickness="1" ScrollViewer.
                 VerticalScrollBarVisibility="Visible">
3      <ListBox.ItemContainerStyle>
4          <Style TargetType="ListBoxItem">
5              <Setter Property="HorizontalContentAlignment" Value="
                   Stretch"></Setter>
6          </Style>
7      </ListBox.ItemContainerStyle>
8      <ListBox.ItemTemplate>
9          <DataTemplate>
10             <Border BorderBrush="Black" CornerRadius="3" Background
                   ="#FFE88D34"
11                     BorderThickness="1" HorizontalAlignment="Stretch
                           " >
12                 <Grid Background="Transparent" HorizontalAlignment="
                       Stretch" >
13                     <Grid.ColumnDefinitions>
14                         <ColumnDefinition Width="*" />
```

```
15                      </Grid.ColumnDefinitions>
16                  </Grid>
17              </Border>
18          </DataTemplate>
19      </ListBox.ItemTemplate>
20  </ListBox>
```



Figur 8.10: Den implementerede udgave af event-liste.

8.3.5 Afspiller

Afspilleren er det område i GUI'en, der bruger flest Controls. Den er opdelt i et skelet af Borders for at give brugeren et bedre overblik.

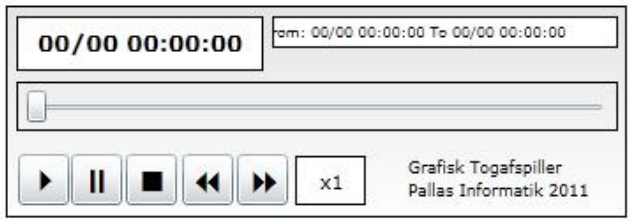
Afspilleren består af en *Button* for hver af de forskellige afspilnings-funktioner og labels, der forklarer tidspunktet. Label'en der viser grænsen, implementerer en marquee effekt, som får indholdet til at søge fra venstre mod højre. Effekten gør, at teksten kræver mindre plads for at blive set, fordi den flytter sig.

Der opstod en særlig udfordring med implementeringen af den Slider, som Afspilleren skulle bruge. De Events, som Slideren skulle tilbyde for at udføre UC6 (Spring i afspilning), blev ikke understøttet som standard. Ligeledes var det heller ikke muligt at trykke et sted i Slider-sporet og fortsætte afspilningen derfra.

For at tilfredsstille disse krav, måtte jeg designe mit eget Slider-control *CustomSlider*, og tilføje nye EventHandlers. Jeg indsætter mit *CustomSlider* i viewet, ved hjælp af følgende kode.

Kodestykke 8.6: Deklarering af Afspiller i GUI.

```
<my:CustomSlider Cursor="Hand" SmallChange="1" Value="0" x:Name="
    CustomSlider"
    Style="{StaticResource CustomSlider}" IsEnabled="True" Width
        ="331" Canvas.Left="5" Canvas.Top="4" />
```



Figur 8.11: Den implementerede udgave af Afspiller.

8.4 Implementering af Model

Denne sektion beskriver de tre klasser, der til sammen udgør modellen. I designkapitlet blev klassehierakiet bestemt, og jeg vil nu se nærmere på de funktioner, der skal implementere systemflowet, beskrevet i sektion 8.2.2.

8.4.1 Player

Klassen Player repræsenterer data-modellen, og har følgende felter.

Kodestykke 8.7: Fields i Player-klassen.

```
    private Conductor conductor;
    public DateTime playFrom;
    public DateTime playTo;
    public Dictionary<string, Location> running;
    public Dictionary<string, Location> stopped;
    public List<string> trainNames;
    public List<Event> activeEvents;
```

DateTime klasserne playFrom og playTo, repræsenterer grænseværdierne "fra", og "til" for afspilningen. De to dictionarys running og stopped fungerer som opslag til Location-data for alle togsæt. trainNames bestemmer indeholder alle

ID for togsæt i afspilningen, og listen activeEvents repræsenterer en kopi af historikken for det togsæt, der er i fokus.

Når Player-klassen opsættes fortolkes oplagringsmodellen til historik for hvert togsæt. Player opsætter en liste af klassen Litra, og bruger den til at opsætte Conductor.

Kodestykke 8.8: Fortolkning af opslagsmodel

```
1     // store everything in a dictionary
2     Dictionary<string, Litra> litraDict = new Dictionary<string,
          Litra>();
3
4     foreach (XElement ri in RowInfo)
5     {
6         // get old or make new train
7         string litra = innerRead(ri.Element("Litra")).TrimStart("(")
              .TrimEnd(")");
8         Litra t = (litraDict.ContainsKey(litra)) ? litraDict[litra]
              : new Litra(litra);
9
10        string time = innerRead(ri.Element("Time"));
11        string trainnumber = innerRead(ri.Element("TrainNumber"));
12        string tritid = innerRead(ri.Element("TritId"));
13        string type = innerRead(ri.Element("Type"));
14        string tekst = innerRead(ri.Element("Text"));
15        t.addEvent(time, trainnumber, litra, tritid, type, tekst);
16        litraDict[litra] = t;
17    }
18
19    // setup conductor
20    conductor = new Conductor(litraDict, playFrom.Ticks);
```

Player-klassen har en metode *play*, der tager et tidspunkt som argument og returnerer en boolean.

Når man arbejder med tid i computerprogrammer, repræsenteres system tiden, som et tal igennem datatypen long. Værdien repræsenterer, hvor mange 100-dele nanosekunder, der er gået siden 1. Januar, 1970 kl. 00:00:00 (også kaldet Unix epoch). Fordelen ved at repræsentere tidspunkter i tal er, at man nemt kan sammenligne to tidspunkter med hinanden.

Metoden *play* kaldes med det tidspunkt i afspilningen, der skal vises, som argument. Hvis tidspunktet ikke befinder sig inden for tidsgrænerne, antages det, at afspilningen er slut, og metoden returnerer falsk. Hvis tidspunktet eksisterer, foretages de nødvendige beregninger, og metoden returnerer sand.

Metoden *play* kalder metoden *conduct* i klassen Conductor, og bruger det samme argument, som den selv blev kaldt med. Metoden indsamler de nye værdier for Location for hvert togsæt og returnerer dem til de to lister *running* og *stopped*, i Player-klassen.

Når metoden play returnerer, vil Controlleren have direkte adgang til de to lister i Player og kan opdatere kortet med de elementer, der skal vises for tidspunktet i afspilningen.

8.4.2 Conductor

Conductor klassen bruges til at udregne Location-koordinater for alle togsæt i forhold til et tidspunkt i afspilningen. For at foretage disse udregninger bruges manuskriptet.

Manuskriptet er defineret ud fra klassen Instruktion, som opbevarer omstændighederne ved at animere et tog mellem to punkter. Omstændigheder blev bestemt i sektion 8.2.1.

Når Conductor bliver opsat bruges oplagringsmodellen til at fortolke manuskriptet. Oplagringsmodellen noterer positionsdata i formatet for GPS, men Bing Maps kan kun fortolke koordinater i GIS. Når manuskriptet opsættes, foretages en konvertering imellem de to formater for alle hændelser.

Når Conductor skal udregne Location for et togsæt betragtes manuskriptet. Manuskriptet er en sorteret liste af fortolkede hændelser (instruktioner), og derfor kan man finde den korrekte instruktion, ved at sammenligne tidsgrænserne for hver hændelse.

Til at udregne positionen mellem to koordinater bruges følgende kode, inspireret af udregningene forklaret i sektion 8.2.1.

Kodestykke 8.9: Beregning af position, mellem to koordinaterl

```
1         // handle train
2         long totalTicks = manuskript[eventNr + 1].moment -
              manuskript[eventNr].moment;
3         long progress = thisMoment - manuskript[eventNr].offset;
4         double percent = progress * 100.0 / totalTicks;
5
6         // sæt ny togPos.
7         double xPctChange = (manuskript[eventNr].xChange / 100) *
              percent;
8         double yPctChange = (manuskript[eventNr].yChange / 100) *
              percent;
```

```
9
10          // lat / long
11          litra.tpi.togPos = new Location(manuskript[eventNr].xStart -
                xPctChange,
12              manuskript[eventNr].yStart - yPctChange);
```

Hvis det ikke er muligt at finde en instruktion for et togsæt til et af tidspunkterne i afspilningen, noteres det i enum-variablen togState, i klassen LitraPlayInfo. Hvis et togsæt ikke registrerer en instruktion, vil togState markeres som "stopped".

Når metoden *tData* i Conductor returnerer Location-data, fordeles togsæt, der skal vises på kortet i listen running, og dem der ikke skal vises, fordeles til listen stopped.

8.4.3 Litra

Litra klassens ansvar er at opbevare alle data omkring et togsæt. Klassen indeholde følgende felter.

Kodestykke 8.10: Fields i Litra-klassen

```
1     public string name { get; set; }
2     public LitraPlayInfo tpi { get; set; }
3     public List<Event> historik { get; set; }
```

TogPlayInfo spiller en vigtig rolle for systemet og besider data, der bestemmer, hvordan togsættet skal fortolkes i afspilningen. Når metoden *conduct* i klassen Conductor kaldes, vil værdierne i LitraPlayInfo blive opdateret. togPos bestemmer den nye position, playEvent bestemmer den seneste hændelse, og playState bestemmer om togsættet skal vises på kortet.

Kodestykke 8.11: Fields i LitraPlayInfo-klassen

```
1     public class LitraPlayInfo
2     {
3         public List<Instruction> manuskript { get; set; }
4         public Location togPos { get; set; }
5         public int playEvent { get; set; }
6         public enum playState { running, stopped }
7         public playState state;
8     }
```

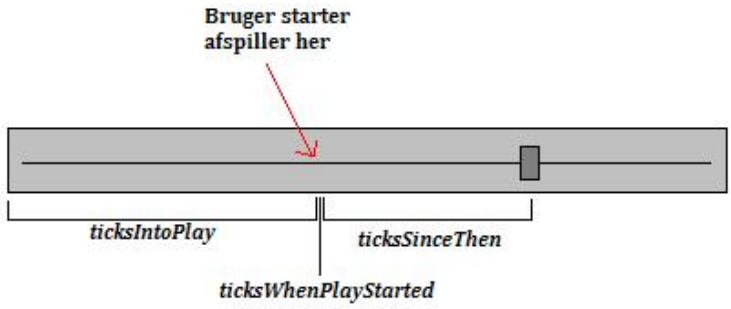
8.5 Implementering af Controller

I sektion 7.3.3 beskrev jeg, at View- og Controller-laget er meget tætte, og det betyder at Controller-klassen (MainPage.xaml.cs) kan styre View-klassen (MainPage.xaml) direkte. Denne opdeling gør, at Controller klassen kan virke uoverskuelig, men er strukturmæssigt meget simpel.

For at styre afspilningen er systemet afhængig af system-tid, og derfor introduceres 3 long-variabler i koden, *ticksWhenPlayStarted*, *ticksInfoPlay*, og *ticksSinceThen*.

ticksWhenPlayStarted svarer til tidspunktet, som en bruger satte afspilningen igang, ticksIntoPlay svarer til den tilbagelagte tid i afspilningen, da afspilningen blev sat igang, og ticksSincethen bestemmer, hvor længe der er gået siden afspilningen blev startet. Det er vigtigt for strukturen i koden, at have en klar opdeling mellem de tidspunkter, der svarer til hændelserne i afspilningen, og de tidspunkter, der styrer afspilningen.

Når man skal styre en afpilning, vil man bruge de tre elementer til at definere tidspunktet i afspilningen, der skal efterspørges af Player. Ved at trække værdien ticksWhenPlayStarted fra den nuværende systemtid, vil man få værdien ticksSinceThen. Denne værdi lægges sammen med ticksIntoPlay, og giver så det tidspunkt, som afspilleren bør fortolke i afspilningen.



Figur 8.12: Styring af system tid.

Værdien for det system ændres hele tiden, og derfor vil denne udregning give et nyt resultat hver gang den foretages.

Vi bestemte i sektion 8.2.4 to metoder, der skal håndteres af EventHandleren *Rendering*. jeg vil nu forklare, hvordan disse metoder er implementet.

8.5.1 updatePlay()

Formålet med denne metode er at opdatere togsæt-elementerne på Verdenskortet og de controls, der udgør Afspilleren. Metoden bliver ved med at håndteres af Rendering, indtil metoden *play* i klassen Player returnerer falsk.

Kodestykke 8.12: Håndtering af togsæt-elementer på Verdenskortet

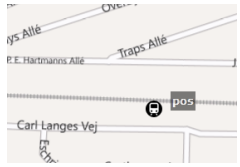
```
1             // update map with new data
2             foreach (string key in player.running.Keys)
3             {
4                 Ellipse t = trainLayer.FindName(key) as Ellipse;
5                 trainLayer.Children.Remove(t);
6                 t.SetValue(MapLayer.PositionProperty, player.running
                      [key]);
7                 t.SetValue(MapLayer.VisibilityProperty, Visibility.
                      Visible);
8                 trainLayer.Children.Add(t);
9             }
10            foreach (string key in player.stopped.Keys)
11            {
12                Ellipse t = trainLayer.FindName(key) as Ellipse;
13                trainLayer.Children.Remove(t);
14                t.SetValue(MapLayer.PositionProperty, player.stopped
                      [key]);
15                t.SetValue(MapLayer.VisibilityProperty, Visibility.
                      Collapsed);
16                trainLayer.Children.Add(t);
17            }
```

8.5.1.1 Flow for updatePlay metoden

1. opdater ticksSinceThen.
2. kald metoden play med tiden (ticksIntoPlay + ticksSinceThen)
3. Hvis play-metoden returnerer false. Ryd op i Bing Maps kortet. nulstil afspilleren. fjern updatePlay metoden fra Rendering EventHandleren.
4. Hvis play returnerer sand. Opdater kortet med nyt data.
5. Hvis kortet skal låses på togsættet i fokus, opdater da kortets position (view).
6. Opdater værdier i afspilleren

8.5.2 manageEvents()

Formålet med denne metode er at vedligeholde markeringen af elementer i Event-List, der repræsenterer hændelser, og det sker ved, at baggrunden på elementet bliver gjort gult. Samtidig indsætter metoden en besked på Verdenskortet ved det togsæt, som er i fokus, der markerer, at der er sket en hændelse.



Figur 8.13: Markering på kortet, omkring en hændelse.

For at kontrollerer et event-skift i afspilningen, bruges klassen LitraPlayInfo for det togsæt, der er i fokus. LitraPlayInfo klassen indeholder værdien *playEvent*, der repræsenterer den Instruktion, som toget udfører. Controller klassen registrerer denne værdi og sammenligner den med sin tidligere version for at fastslå, om der er sket en ændring.

De to værdier bestemmer om listen skal nulstilles eller skifte. Hvis vi antager, at playEvent var 6 og skifter til 7, vil markeringen skifte til det næste element.

Kodestykke 8.13: Håndtering af togsæt-elementer på Verdenskortet

```
1             //change highlight
2             if (oldEventNr != -1)
3             {
4                 ListBoxItem lbi = eventListBox.Items[oldEventNr] as
                      ListBoxItem;
5                 Border b = lbi.Content as Border;
6                 b.BorderBrush = new SolidColorBrush(Colors.
                      Transparent);
7                 b.Background = new SolidColorBrush(Colors.
                      Transparent);
8             }
9             if (newEventNr != -1)
10            {
11                ListBoxItem lbi = eventListBox.Items[newEventNr] as
                      ListBoxItem;
12                Border b = lbi.Content as Border;
13                b.BorderBrush = new SolidColorBrush(Colors.Black);
14                b.Background = new SolidColorBrush(Color.FromArgb
                      (255, 255, 255, 200));
15            }
```

Når der sker et hændelsesskift for togsættet i fokus vil metoden også markere skiftet på Verdenskortet ved at opsætte en tekstbox, der er synlig i 3 sekunder. For at administrere de synlige tekstboxe, har jeg opsat et system igennem en ArrayList *ActiveTips*, der opbevarer navnene på de elementer på kortet, samt systemtiden, der bestemmer, hvornår de skal fjernes.

Jeg vedligeholder tekstboxene ved at sammenligne tiden de skal fjernes, med den nuværende systemtid. Hvis en tekstbox ikke længere skal vises, bliver den fjernet, og hvis en tekstbox stadig skal vises, tilføjes den til en ny liste *newList*. Når alle objekter er kontrolleret, overskrives den gamle liste *ActiveTips* med indeholdet fra *newList*.

Kodestykke 8.14: Håndtering af tekstboxe på Verdenskortet

```
1                var newList = new List<object[]>();
2
3                foreach (object[] o in ActiveTips)
4                {
5                    var name = (string)o[0];
6                    var duetime = (long)o[1];
7
8                    if (duetime < DateTime.Now.Ticks)
9                    {
10                       Border train = trainLayer.FindName(name) as
                             Border;
11                       trainLayer.Children.Remove(train);
12                   }
13                   else
14                       newList.Add(o);
15               }
16               ActiveTips = newList;
```

8.5.2.1 Flow for manageEvent metoden

1. If no train is in focus, clear tips if requested, and return.
2. Hent tpi fra tog i fokus, og bestem newPlayEvent.
3. Hvis clear tips er kaldt, skift markering, ellers fjern de tips der er forældede.
4. Håndtering af events. Returner hvis der ikke laves naturligt skift.
5. Skift markering, tilføj ny tekstbox på kortet, og rul eventliste for brugeren.

8.6 Styring af afspilning med Control events

Når systemet bruges til at udføre et use-case vil GUI'en blive brugt til at styre, hvad der skal ske. Hvis en bruger ønsker at starte en afspilning (UC2), opsætter han først systemet og trykker på knappen playBtn. Brugerens handling håndteres af metoden playBtn_Click i Controlleren, der nu kan dirigere systemflowet, udelukkende ved hjælp af metoderne updatePlay og manageEvents.

Controlleren bruger en boolean variabel *playing*, som er sand, når der afspilles og falsk, når der ikke gør. Metoden playBtn_Click starter afspilningen ved at lade Rendering håndtere updatePlay.

Kodestykke 8.15: Håndtering af metoden playBtn_Click

```
1             if (!playing)
2             {
3                 // start player
4                 ticksWhenPlayStarted = DateTime.Now.Ticks;
5                 ticksIntoPlay = (long)CustomSlider.Value - player.
                      playFrom.Ticks;
6                 CompositionTarget.Rendering += new EventHandler(
                      updatePlay);
7                 playing = !playing;
8                 clearTips = true;
9             }
10            else
11            {
12                //restart player
13                CustomSlider.Value = CustomSlider.Minimum;
14                ticksWhenPlayStarted = DateTime.Now.Ticks;
15                ticksIntoPlay = (long)CustomSlider.Value - player.
                      playFrom.Ticks;
16                clearTips = true;
17            }
```

De resterende events til controls i viewet kan ses i appendix X, side XYZ.

8.7 Delkonklusion

Jeg har nu implementeret hele systemet, og det er hensigten, at de definerede use cases nu skal kunne udføres med succes.

Det lykkedes mig at implementere alle definerede krav, men med en lille undtagelse. Det viste sig, at det ikke var muligt at bruge Verdenskortets zoom-

funktion, når et togsæt var sat i fokus. Jeg har løst problemet ved at tilføje to knapper til Tog-Fokus området, der kan bruges til hhv. at zoome ind, og ud.

Silverlight krævede også en særlig tilgang til at implementere animationer. Jeg måtte derfor implementere to metoder, *updatePlay* og *manageEvents*, til at håndtere animationen af brugergrænsefladen.

Implementering-forløbets afslutning markerer det færdige produkt. Det implementerede produkt mangler nu kun at blive kontrolleret igennem tests.

Kapitel 9

Test

Dette kapitel har til formål at teste systemet, før det kan overdrages. Kapitelet redegør for arbejdsforløbet defineret i sektion 4.2.3.5.

Der er flere måder at teste et system på, med flere formål. For at teste mit system, har jeg valgt at udføre greybox-testing.

9.1 Greybox-testing

Hvis en person udfører en blackbox-test på et produkt, har han ikke kendskab til produktet. Testen udføres ved at give produktet et input og undersøge resultatet. Lever resultatet op til forventningen, er produktet en succes.

Hvis en person udfører en greybox-test[Gre] på et produkt, har han kendskab til, hvordan produktet er lavet. Conceptet bag en greybox-test er, at en person, der ved hvordan et produkt er opbygget, bedre vil kunne teste det. En vigtig detalje bag en greybox-test er, at man som tester kun må teste produktet udefra.

Ønsker man at teste applikationen, kan man udføre unit tests. Unit tests bruges til at teste dele af koden individuelt for at se, om alle resultater stemmer overens.

Jeg har i mit projekt valgt ikke at udføre unittests, da jeg har vurderet dem til at være for omfattende til en projekt af min størrelse. Unit tests kan nemt opsættes, og det vil være en fordel at bruge dem på vigtige dele af funktionaliteten i min kode, hvis mit produkt fik et større omfang.

I de følgende sektioner vil jeg for hvert gennemgå en greybox-test for hvert use-case i systemet. Fremgangsmåden for hver use-case er at udføre de forskellige trin i use-casen og kontrollere resultatet. Jeg vil for hver trin i hvert test også beskrive, hvilket resultat jeg forventer, og hvilket resultat systemet giver mig.

9.1.1 Test af UC1: Vælg afspilning

Prækondition
Programmet er opstartet.

Trin	Aktivitet	Forventet resultat	Resultat	OK
1	Tryk på knappen "Vælg" i Afspilninger-delen	Afspilleren er opsat ved, at den en viser et tidspunkt i afspilleren, og den er opsat med tidsgrænseværdierne for afspilningen	Afspilleren er opsat som forventet.	✓
2	Prøv slider	Når man rykker slidernålen, vil slider-tiden, der vises i afspilleren skiftes.	Slider-tiden skiftes.	✓

9.1.2 Test af UC2: Start afspilning

Prækondition
Afspilleren er opsat ved, at en afspilning er valgt.

Trin	Aktivitet	Forventet resultat	Resultat	OK
1	Tryk på play-knappen i Afspilleren	Nålen begynder gradvist at bevæge sig og kortet viser togsæt, der flydende rykker sig på kortet	Afspilningen kører som forventet	✓
2	Zoom ind på kortet og kontroller et togsæt	Når afspilningen er igang vil togsæt bevæge sig	Togsæt bevæger sig som forventet.	✓

9.1.3 Test af UC3: Stop afspilning

Prækondition
Afspilleren er opsat ved, at en afspilning er valgt, og der afspilles.

Trin	Aktivitet	Forventet resultat	Resultat	OK
1	Tryk på stop-knappen i Afspilleren	Afspilningen nulstilles ved, at alle elementer, der vises på kortet bliver fjernet. Tiden der viser, hvor der afspilles bliver nulstillet til at vise starttidspunktet for afspilningen.	Afspilningen nulstilles som forventet	✓

9.1.4 Test af UC4: Pause afspilning

Prækondition
Afspilleren er opsat ved, at en afspilning er valgt, og der afspilles.

Trin	Aktivitet	Forventet resultat	Resultat	OK
1	Tryk på pause-knappen i Afspilleren	Afspilningen standses ved det tidspunkt, der er nået. Ingen elementer rykker sig og tiden i afspilleren skifter ikke.	Afspilningen pauses som forventet	✓
2	Zoom ind på kortet og kontroller et togsæt	Når afspilningen er pauset vil alle togsæt stoppe med at bevæge sig.	Togsæt er standset som forventet.	✓

9.1.5 Test af UC5: Spol i afspilning

Prækondition
Afspilleren er opsat ved, at en afspilning er valgt, og der afspilles.

Trin	Aktivitet	Forventet resultat	Resultat	OK
1	Brug slideren ved enten at trække i nålen, eller trykke på tidslinjen.	Afspilningen fortsætter normalt ved det valgte tidspunkt.	Afspilningen fortsætter ved et nyt tidspunkt som forventet.	✓
2	Zoom ind på kortet og find et togsæt. Brug slideren til at springe i afspilningen og bekræft ændring på kortet.	Togsæt vil fortsætte på kortet et nyt sted.	Togsæt fortsætter et nyt sted, som forventet.	✓

9.1.6 Test af UC6: Spring i afspilning

Prækondition
Afspilleren er opsat ved, at en afspilning er valgt, og der afspilles.

Trin	Aktivitet	Forventet resultat	Resultat	OK
1	Brug slideren ved enten at trække i nålen eller trykke på tidslinjen.	Afspilningen fortsætter normalt ved det valgte tidspunkt.	Afspilningen fortsætter ved et nyt tidspunkt som forventet.	✓
2	Zoom ind på kortet og find et togsæt. Spring i afspilningen med slideren og bekræft ændring.	Togsættet vil fortsætte på kortet et nyt sted	Togsæt fortsætter et nyt sted som forventet.	✓

9.1.7 Test af UC7: Fokuser på tog

Prækondition
Afspilleren er opsat ved, at en afspilning er valgt.

Trin	Aktivitet	Forventet resultat	Resultat	OK
1	Vælg togsæt fra liste og tryk knappen "Vælg" i Tog-Fokus delen.	Event-List bliver opsat med data omkring events fra togsættets historik. Den seneste hændelse markeres med det samme.	Event-List bliver opsat som forventet	✓
2	Vælg et togsæt og kontroller om event-skift markeres på kortet.	Togsættet markerer på kortet når en event opstår.	Togsættet markerer på kortet som forventet	✓

9.1.8 Test af UC8: Lås afspiller til tog

Prækondition
Afspilleren er opsat ved, at en afspilning er valgt, og et togsæt er i fokus.

Trin	Aktivitet	Forventet resultat	Resultat	OK
1	Sæt et flueben i CheckBoxen i Tog-Fokus delen	Såfremt afspilningen er igang vil Bing Maps kortet følge togsættet, der er i fokus ved at flytte billedet mens det bevæger sig.	Bing Maps kortet følger togsættet som forventet	✓

9.2 Delkonklusion

Jeg har nu udført tests for systemet, igennem alle use cases. Det er lykkedes mig at implementere alle 8 use cases, og få positive resultater igennem testene.

Succeskriterierne der blev sat i sektion 5.6 må hermed anses som opfyldte.

Kapitel 10

Konklusion og Perspektivering

Jeg har nu færdiggjort mit produkt igennem systemudviklingsprocessen Unified Process.

Baggrunden for projektet var, at DSB idag bruger en besværlig proces, når de skal analysere fejl i togdriften. DSB er nødsaget til at inddrage Pallas Informatik og kan derfor ikke løse opgaven alene.

Formålet med mit projekt er at lave en prototype på en løsning til problemet, ved at afprøve teknologien Silverlight. Prototypen skal bruges af Pallas Informatik til at konkludere, hvorvidt Silverlight kan løse problemet, og om det er en god platform at bruge til at udvikle et komplet system.

Jeg forberedte projektet ved at afgrænse den del, som jeg ville fokusere på at løse. Jeg opsatte systemkrav, og identificerede use cases til at bestemme, hvad systemet skulle kunne. Jeg foretog en risikoanalyse til at identificere potentielle trusler og konkluderede, at systemet fortsat burde udvikles. Jeg opsatte succeskriterier, der bestemmer, hvornår systemets formål er opfyldt.

Jeg analyserede projektet som koncept og opsatte en domæne model til at kortlægge en ansvarsfordeling. Derefter beskrev jeg de identificerede use cases for at sætte krav til systemets design.

Jeg udarbejdede et design for hele min projektafgrænsning. Jeg bestemte en tredelt arkitektur for Silverlight applikationen med MVC og definerede, hvordan den var forbundet. Ved hjælp af sekvensdiagrammer beskrev jeg, hvordan arkitekturen skulle bruges til at udføre alle use cases.

Jeg implementerede systemet igennem Silverlight-platformen, og tilpassede fremgangsmåden for animationer. Det er dog ikke lykkedes mig at lade zoomfunktionen være aktiveret, når et togsæt sættes i "fokus".

Jeg testede systemet igennem greybox-tests ved at gennemgå alle use-cases. Dermed anser jeg succeskriterierne for opfyldte.

På baggrund af udviklingsprocessen må jeg konkludere, at systemet godt kan bruges til at udvikle en løsning for opgavens problemstilling. Jeg har kun lavet prototypen til en afgrænsning af systemet, men jeg mener, at Silverlight som platform vil være en god basis for et komplet system.

Prototypen som jeg har udviklet kan være med til at lægge grundarbejdet for den løsning som Pallas Informatik vil lave, hvis DSB køber projektet. Mit bachelorprojekt kan derfor være med til at implementere en løsning for DSB, der kan bruges til at forbedre togdriften, og dermed forbedre serviceoplevelsen for den almindelige bruger.

Mit bachelorprojekt er lavet på en måde, så det retter sig mod DSBs langtoge, men vil også kunne bruges i andre brancher. Løsningen kan visualisere en rute på et verdenskort, der defineres ud fra GPS-koordinater. Programmet vil derfor kunne administrere en "flåde" af andre transportmidler, så som skibe og lastbiler.

Fremtiden for min prototype vil være at implementere de resterende hændelsestyper og visualisere dem på kortet. Man ville også kunne arbejde videre på fremstillingen af den rute, som togsæt bevæger sig efter. Oplagringsmodellen indeholder flere detaljer, bl.a. hastigheden på togsættet, da GPS-meldingen blev registreret. Ved at inkludere disse data i udregningen, vil man kunne præcisere togsættets placering endnu bedre.

En billede af implementeringen kan ses på figur **??**.

En implementering af systemets prototype kan ses på webaddressen
http://www.hench.dk/GTA/gta.html.
(Bemærk at det kræver en installation af Silverlight).

Appendix A

Ordforklaring

Ord	Ordforklaring
Togsæt	En sammensætning af 2 eller flere togvogne, der tilsammen udgør enhed, der kan medbringe passagerer.
Litra	Synonym for togsæt. Bruges ofte i teknisk sammenhæng.
TBE	TBE er en forkortelse for Tog Basis Enhed. En TBE er en boks, der er placeret i alle vogne af DSBs langtoge.
Event	Et Event er en hændelse der sker i softwareprogrammer, der typisk aktiveres direkte af brugeren. Dette kan eksempelvis opstå ved hjælp af tastatur eller mus.
EventHandler	En EventHandler er en program rutine, der køres når et event opstår i et stykke software. Dette kan eksempelvis være at lukke computeren igennem Windows, når en bruger klikker på sluk ikonet.
GPS	Forkortelse for Graphical Positioning System. GPS bruges til at beregne geografisk position og evt. højde over vandets overflade. For at modtage en geografisk position med GPS kræver det en GPS-modtager, der er forbundet til en GPS-satellit.

GIS	Forkortelse for Geographic Information System. GIS er et avanceret system, der kan beskrive geografiske positioner for hele jorden. GIS er mere generisk anlagt end GPS.
WMS	Web-mapping service er et protokol-concept, der leverer billeder over internettet genereret af en kort-server ved at bruge data fra en GIS-database.
XML	Forkortelse for Extensible Markup Language. XML er et opmærkningssprog, der bruges til at opbevare data i et format, der både kan læses af mennesker og computer.
XAML	Forkortelse for eXtensible Application Markup Language. XAML er et XML-sprog, der bruges til at deklarere og tildele værdier til objekter i .NET Frameworket.
Framework	Det engelske ord for en programmeringsplatform. Et framework beskriver det udviklingsmiljø, der bruges til at udvikle et program.
Use-case	En liste af trin eller en beskrivelse, der definerer en interaktion mellem en bruger og et system.
Sekvens diagram	Et diagram der viser hvordan software-processer interagerer med hinanden i et system. Processerne er defineret imellem to objekter.
Domænemodel	En domæne model er en konceptmodel, der beskriver roller og forhold i et problem-område.
Client	En client (klient) er en applikation eller et system, der tilgår en service gjort ledigt af en Server.
Server	En server er et computerprogram, der modtager forespørgsler fra andre programmer og udfører en service.
VPN	Forkortelse for Virtual Private Network. VPN er et netværk, der primært bruger offentlig telekommunikation så som internettet til at sikre tunnel-kommunikation mellem flere computere. Netværket er krypteret og kræver autenmtificering for bruges.
Slider	En slider er et kontrol objekt i Silverlight, som kan tilføjes på brugergrænsefladen. En slider repræsenterer en værdi, der befinder sig mellem to grænser. Slideren repræsenterer værdien igennem en nål, der kan køres over et spor.
Control	Et control er en samling af brugergrænseflade-elementer, der kan bruges af en systembruger. Controls deklareres i XAML som objekter. Et control fungerer som et værktøj, der kan stille funktionalitet til rådighed for en bruger.
Dictionary	En dictionary er en klasse, der kan bruges til at opbevare data og fungerer på samme måde som et Hashtable. Man kan tilgå en værdi i en dictionary ved at bruge en nøgle.

Bibliography

[BD04] Bernd Bruegge and Allen H. Dutoit. *Object-Oriented Software Engineering: Using UML, Patterns and Java*. Prentice Hall, second edition edition, 2004.

[BPSM+06] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml) 1.0 (fourth edition). http://www.w3.org/TR/2006/REC-xml-20060816, August 2006.

[Gre] What do you understand by gray box testing? http://productdevelop.blogspot.com/2011/11/what-do-you-understand-by-gray-box.html.

[Lar01] Craig Larman. *Applying UML and Patterns: An introduction to Obect-Oriented Analysis and Design and the Unified Process*. Prentice Hall, second edition edition, 2001.

[Lik] Jeremy Likness. Model-view-viewmodel (mvvm) explained. http://csharperimage.jeremylikness.com/2010/04/model-view-viewmodel-mvvm-explained.html.

[Mica] Microsoft. Bing maps platform - overview. http://www.microsoft.com/maps/product/overview.aspx.

[Micb] Microsoft. Bing maps silverlight control 1.0 released. http://www.bing.com/community/site_blogs/b/maps/archive/2009/11/09/bing-maps-silverlight-control-1-0-released.aspx.

[Micc] Microsoft. Microsoft silverlight - overview. http://msdn.microsoft.com/en-us/library/bb404700(VS.95).aspx.

[Micd] Microsoft. System.windows.controls namespace. http://msdn.microsoft.com/en-us/library/ms590941(v=VS.95).aspx.

[Mice] Microsoft. Xaml overview. http://msdn.microsoft.com/en-us/library/cc189036(v=vs.95).aspx.

[MVC] Mvc design pattern brings about better organization and code reuse [online]. http://www.oracle.com/technetwork/articles/javase/index-142890.html.

[Rap] When you produce graduate theses and reports - guidelines and recommended typesettings. http://www2.imm.dtu.dk/teaching/thesis/.

[Smi] Gozzo Smith. Custom animations in silverlight. http://www.silverlightshow.net/items/Custom-Animations-in-Silverlight.aspx.