

Orientering Tracker



P2 PROJEKT
GRUPPE A401
SOFTWARE
AALBORG UNIVERSITET
DEN 27. MAJ 2015



AALBORG UNIVERSITET
STUDENTERRAPPORT

Første Studieår v/ Det Teknisk-Naturvidenskabelige Fakultet

Software

Strandvejen 12-14

9000 Aalborg

Titel:

Orienteringsløb

Projekt:

P2-projekt

Projektperiode:

Februar 2015 - Maj 2015

Projektgruppe:

A401

Deltagere:

Christian Dannesboe
Frederik Børsting Lund
Karrar Al-Sami
Mark Kloch Haurum
Søren Lyng

Hovedvejleder:

Jacob Nørbjerg

Synopsis:

Denne rapport beskriver analysen og udviklingen af et program til andet semesterprojekt. Projektet er udarbejdet fra projektoplægget "IT i foreninger". Analysen går ind i hvad orienteringsløb(o-løb) er og hvordan en o-løbs træning foregår. Ud fra analysen vil projektet handle om at optimere en o-løbs træning med et stykke software. Næste del af rapporten handler om udviklingen af software løsningen, teorien bag den, implementeringen og test af løsningen. Tilslut vil der være en diskussion af projektet, samt en konklusion og perspektivering.

Oplagstal: 10

Sidetal: 65

Appendiks: 2

Afsluttet 27-05-2015

Rapportens indhold er frit tilgængeligt, men offentliggørelse (med kildeangivelse) må kun ske efter aftale med forfatterne.

Christian Dannesboe

Frederik Børsting Lund

Karrar Al-Sami

Mark Kloch Haurum

Søren Lyng

Forord

Dette projekt er udarbejdet i samarbejde mellem fem software-studerende på andet semester fra Det Teknisk-Videnskabelige Fakultet på Aalborg Universitet.

I udarbejdelsen af projektet, har gruppen taget udgangspunkt i Aalborg-modellen, i form af problem- og projektbaseret læring. Der tages udgangspunkt i et problem, hvor læringen sker i form af projektarbejde i grupper.

Gruppen vil gerne takke vejlederen Jacob Nørbjerg, for hans vejledning gennem hele projekt forløbet. Derudover vil gruppen gerne takke Claus Bobach for deltagelsen i interviewet, der gav indblik i o-løbstræningen, samt Jens Børsting for hans hjælp gennem hele projektet.

Læsevejledning

Kildehenvisning

I dette projekt bruges Harvard-metoden, også kendt som Chicago-metoden, til kilde henvisning. Hvis der henvises til en bestemt kilde, efter eksempelvis en sætning, påstand eller et citat, henvises der på følgende måde: Sætning/påstand/citat [Forfatter, udgivelsesår].

Hvis kilden anvendes til hele afsnit, sættes kildehenvisningen efter punktummet, således: Afsnit.[Forfatter, udgivelsesår]

I afsnittet "Litteratur" vil kilde henvisningerne blive sorteret i alfabetisk rækkefølge. Hvis det eksempelvis var en hjemmeside der blev brugt som kilde, ville det se således ud:

Kilde henvisningen fra rapporten. Forfatter. *Titel.* URL. Udgivelsesår. Dato siden er set og evt. sidetal.

Dansk Idrætsforbund, 2013. Dansk Idrætsforbund. *Medlemstal.*

http://www.dif.dk/da/om_dif/medlemstal, 2013. Set d. 27/2-2015.

Hvis nogle af disse informationer mangler, eksempelvis udgivelsesår, udelades de.

Figurhenvisning

Igennem rapporten vil der blive henvist til figurer og illustrationer, hvor det vil blive anvist ud fra hvilket afsnit det befinner sig i, samt hvilket nummer figuren er i det omtalte afsnit. Herudover skal der være en beskrivende tekst, der forklarer figuren, eksempelvis:

Figur 2, afsnit 5: Figurtekst

Figur 5.2: Figurbeskrivelse

Indholdsfortegnelse

Kapitel 1 Indledning	1
Kapitel 2 Problemanalyse	2
2.1 O-løb	2
2.1.1 Begreber	3
2.2 Interessentanalyse	3
2.2.1 Aktører	3
2.2.2 Prioriteringen	4
2.3 Typisk o-løbs træning	5
2.3.1 Problemer	6
2.4 Eksisterende løsninger	8
2.4.1 Endomondo	8
2.4.2 EMIT brikker	8
2.4.3 QuickRoute	9
2.4.4 TracTrac	10
Kapitel 3 Problembeskrivelse	11
3.1 Problemafgrænsning	11
3.2 Problemformulering	12
Kapitel 4 Kravspecifikationer	13
4.1 Optimale løsningsforslag	13
4.1.1 Krav til optimal løsning	14
4.1.2 Mockup af optimal løsning	15
4.2 Reelle løsning	19
4.2.1 Brugervenlighed	20
4.2.2 Løsningsstrategi	21
Kapitel 5 Teori	22
5.1 Længde- og breddegrader	22
5.2 Universal Transverse Mercator systemet	22
5.3 Konvertering	23
5.4 Opsummering	24
Kapitel 6 Implementering	25
6.1 Brugergrænseflade	25
6.2 Programstruktur	27
6.2.1 Klassebeskrivelse	27
6.3 Kildekoden	28
6.3.1 Helper	28
6.3.2 Coordinate	29
6.3.3 ControlPoint	29

6.3.4	ControlPointTime	30
6.3.5	Runner	30
6.3.6	RunnerData	32
6.3.7	Leg	32
6.3.8	Player	32
6.3.9	MainForm	33
Kapitel 7	Test	40
7.1	Monkey testning	40
7.1.1	GUI tester	40
7.2	Blackbox testning	41
7.3	Unit testning	42
Kapitel 8	Diskussion	44
8.1	Problemanalyse	44
8.2	Program design	44
8.3	Programmeringsvalg	45
8.4	Test	45
8.5	Opsumming	45
Kapitel 9	Konklusion	47
Kapitel 10	Perspektivering	49
Litteratur		50
Appendiks A	Interview transskribering	52
Appendiks B	E-mail korrespondance med Jens Børsting	56

Indledning 1

Der findes mange forskellige typer af foreninger i Danmark, nogle er større end andre. De mindre foreninger har færre ressourcer til rådighed, og arbejdsbyrden på de enkelte medlemmer kan være stor.

Denne rapport vil forholde sig til orienteringsløb i mindre foreninger, i Danmark er der 76 foreninger med omrent 7.000 medlemmer [Dansk Idrætsforbund, 2013]. Igennem samtaler og mailkorrespondancer med Jens Børsting, som har løbet og været engageret i o-løb i over 30 år, blev det konkluderet, at der er meget arbejde for medlemmerne i o-løbsforeninger, og at det er begrænset hvor meget software der findes til at hjælpe o-løbere.

I et orienteringsløb skal en løber med kort og kompas hurtigst muligt finde et antal forudbestemte poster, typisk i en skov. Da løberne undervejs er svære at holde under opsyn, er det ikke attraktivt at være tilskuer til et o-løb. Der er samtidigt ikke mange unge o-løbere til trods for, at Danmark har nogle af verdens bedste o-løbere [IOF World Ranking, 2015].

Hidtil har posterne været opsat fysisk, og ruten der er tiltænkt løbet er planlagt på forhånd. Dette arbejde indebærer flere timers arbejde både før og efter en træningsgang eller løb. Derudover kan det være svært at sammenligne de vejvalg, den enkelte løber har taget på ruten, da sammenligning kun kan ske ud fra løbernes hukommelse. Dette gør det svært for træneren at fortælle hvad løberen kunne have gjort anderledes. Løberen kan heller ikke sammenligne sine vejvalg med de andre løbere, for at finde svagheder i sit eget løb.

Den initierende problemstilling i dette projekt lyder derfor sådan:

Hvordan kan planlægningen, afviklingen og opfølgningen af træningen for amatør o-løbere forbedres/effektiviseres vha. en IT-løsning?

- Hvordan foregår en o-løbstræning?
- Hvilke problemer forekommer der under en o-løbstræning?
- Hvilke teknologiske redskaber bruges til en o-løbstræning?

Problemanalyse 2

I dette kapitel vil o-løb blive introduceret samt den initierende problemstilling blive analyseret. Dette gøres i form af interressentanalyse der skal hjælpe til at finde de væsentlige interesserter til projektet. På baggrund af interressentanalysen, bliver der foretaget interviews med udvalgte interesserter. Til sidst i kapitlet vil eksisterende løsninger blive beskrevet og analyseret.

2.1 O-løb

Følgende afsnit vil omhandle hvad et o-løb er, termer fra sporten og forklare hvordan et typisk o-løb fungere. Denne information er fundet på Dansk Orienterings-Forbunds hjemmeside [Schmidt, 2015], samt interviews med Jens Børsting, der blev introduceret i indledningen, og Claus Bobach, som er klubtræner hos Aalborg Orienteringsklub.

O-løb er en sport hvor det, som så mange andre sportsgrene, gælder om at være hurtigst. O-løb er anderledes, da det er op til løberen selv at finde vej, ved hjælp af et kort, og måske et kompas. Inden løbet starter, bliver alle løbere udstyret med et kort, med detaljeret visning af stier, bakker og andre udfordringer i terrænet. På dette kort vises de forskellige poster løberen skal finde inden vedkommende har gennemført løbet. Hvis kortet viser en bestemt rækkefølge posterne skal besøges i, så er det påkrævet at gøre det i den rækkefølge.

Kortlæsning er en af de vigtigste aspekter ved o-løb. Det gør løberen i stand til at finde rundt, optimere sin rute og vælge den bedste vej rundt om forhindringer. ”Vi kan med god ret sige, at det er her sjælen i sporten ligger gemt”, skriver Dansk Orienterings-Forbund.

Ved hver post, står en orange/hvid skærm (figur 2.1). På skærmen er oplyst et nummer, så det er muligt at tjekke om det er den rigtige post der er fundet. Desuden er der ved mange løb opstillet elektronisk poster, (figur 2.2) som gør det muligt at se hvornår løberen har været ved posterne.



Figur 2.1. Almindelig skærmpost



Figur 2.2. Elektronisk post

Der findes en række forskellige discipliner inden for o-løb. Først er der Sprint, Mellem, Lang og Ultralang, der er enkeltmandsløb, hvor den eneste forskel er distancen der løbes. Sprint er den korteste, og blive ofte afviklet i byer for at gøre det så simpelt og hurtigt som muligt, og Ultralang er den længste, hvor vindertiden typisk er højere end ved maratonløb. Uddover standard løbene, findes også Nat, Stafet og Hold, der adskiller sig ved, henholdsvis at foregå om natten, eller have flere løbere.

2.1.1 Begreber

EMIT brikken bruges til tidtagning af o-løb. EMIT brikken er en lille firkantet brik, på ca. 5x10x1cm, som sidder fast på løberens finger vha. en elastik. Ved hver post er en kontrolenhed, der gemmer postens nummer og tiden i brikken. Ved endt løb aflæses brikken og tiderne kan skrives ud.

Orienteringsløbere eller o-løbere, er en person der deltager i o-løb, uanset om det er på professionelt niveau, eller som en fritidsaktivitet.

Stræk er stykket mellem to poster i et orienteringsløb. Distancen for et stræk er meget individuelt, da løberne ikke nødvendigvis tager den samme rute for at nå fra en post til en anden.

Delstræk er mindre dele af et stræk, der ikke er fra post til post. Dette snakkes mest om hvis der er flere svære forhindringer på et stræk.

Stræktider er tiden det tog at komme fra en post til en anden.

2.2 Interessentanalyse

I dette afsnit vil indflydelses- og medvirken matrixen blive anvendt, for at undersøge og prioritere interesserter for dette projekt. Dette undersøges for, at finde ressourcepersoner som gruppen kan gøre brug af gennem interviews. Disse personer vil være respondentgruppen til undersøgelse af den initierende problemstilling. Jens Børsting har hjulpet til med at finde interesserter til projektet, og har senere hjulpet med et interview.

2.2.1 Aktører

I dette projekt har gruppen fundet fire aktører, som gruppen mener er relevante til projektet. De forskellige aktører er henholdsvis o-løberne, trænerne, sportens forbund og foreninger og frivillige.

O-løbere

For at den enkelte o-løber skal kunne forbedre sig, er det vigtig at kunne sammenligne løberens rute detaljeret med andre. Lige nu er tiderne mellem hver post (stræktiderne) det eneste der kan sammenlignes og analyseres på. Her er det interessant for løberen at kigge på vejvalg og hastighed mellem posterne, og endda helt ned til de forskellige faser af delstrækkene. Til dette mangler der mere detaljeret data om løbet. Problemet håndteres i dag ved at sammenligne skemaer med stræktider og hvis muligt manuelt indtegne vejvalg på kortet efter løberens hukommelse. Derfor har den enkelte o-løber interesse i dette projekt, da der arbejdes med afvikling og opfølgning af træningen.

Træneren

Træneren har interesse i at mindske arbejdstiden brugt på forberedelse og planlægning af løb, da dette giver mere tid til træning af de enkelte løbere. Samtidigt vil træneren gerne kunne analysere den enkelte løbers tur detaljeret, ved at sammenligne løberens rute med andre løberes rute. Hvis løberen ikke kan huske hvor vedkommende har løbet, eller var faret vild, har træneren svært ved at give sikker og brugbar kritik, da det ikke kan ses på tiderne, præcis hvor den enkelte løber har været. Trænere har derfor interesse i et værktøj som kan hjælpe med planlægning og afviklingen af træningen, samt evaluering af den enkelte løbers tur.

Forbund og foreninger

O-løbernes forbund hedder Dansk Orienterings Forbund, også kaldt for DOF, som ligger under Dansk Idrætsforbund, DIF. Der er i alt 76 foreninger i DOF, med lidt under 7.000 medlemmer [Dansk Idrætsforbund, 2013]. DOF er med til at drive landsholdet, samt står for talentudviklingen inden for orienteringsløb. Dette gør DOF og foreningerne til interessenter i dette projekt, da de bl.a. ønsker deres løbere skal blive så gode som mulige. [Dansk Idrætsforbund, 2013]

Frivillige

I foreninger er frivillige vigtige, da 80% af alle lokale foreninger er udelukkende benytter frivillig arbejdskraft. Derudover er mange af de frivillige engageret i o-løb, da de er villige til at lave et stykke ubetalt arbejde, for at holde gang i klubben. [Frivillighed.dk, 2015]

2.2.2 Prioriteringen

Indflydelse- og medvirken matrixen deles op i fire rum, hvor hvert rum er skaleret efter ”indflydelse på projektet” og ”afgørende medvirken i projektet”. Dette giver rum for diskussion af de enkelte interesserter, for at undersøge hvilke interesserter, der skal tages kontakt til, for at få svar på den initierende problemstilling. Gruppen har i denne rapport meget få interesserter, hvilket medfører, at de enkelte interesserter hurtigt bliver ressourcepersoner.

Eksterne interesserter har hverken en stor indflydelse eller en stor medvirken i projektet. Det betyder at der ikke nødvendigvis skal tages stor hensyn til denne gruppe af interesserter.

Den grå eminence har stor mulighed for at påvirke beslutninger i projektet, men deres medvirken er ikke særlig stor i projektet. Dette vil ofte være en person med en magtfuld stilling eller ledelsesposition.

Gidsler har ikke stor mulighed for at træffe beslutninger i projektet, men deres aktive medvirken er vigtig for projektet.

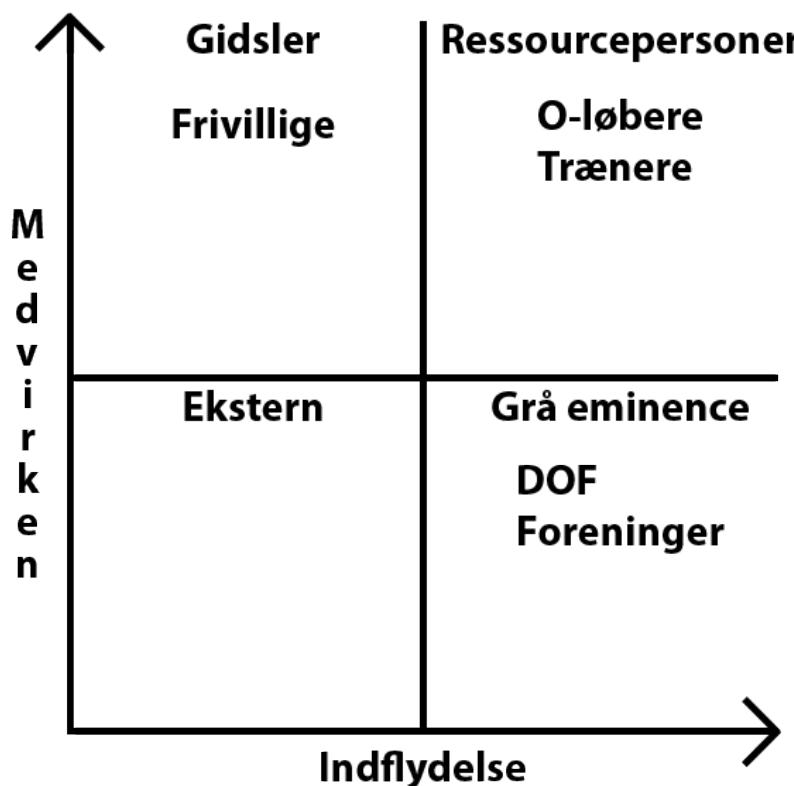
Resourcepersoner har både stor indflydelse og stor medvirken, da det er denne gruppe der skal inddragges i projektet. Denne gruppe har erfaring eller faglige kompetencer indenfor området.

O-løbere er i dette projektet sat som ressourceperson, da de kan give råd og vejledning til, hvordan deres træning og løb fungere, samt undersøge om der er ting der kan forbedre deres løb. Dette gælder både inden- og efter løbet.

Trænere er sat som ressourceperson for projektet, da de ligesom o-løberne har et stort indblik i hvordan orienteringsløb fungere, og hvordan det kan optimeres eller forbedres.

DOF og foreningerne er i dette projekt grå eminence, da de kan have en indflydelse på projektet. Herudover kan de have nogle krav og regler til en løsning. Deres medvirken er dog ikke nødvendig for at projektet skal kunne blive en succes.

I dette projekt har gruppen valgt at placere de frivillige som gidsler. De frivillige har ikke nogen indflydelse på hvordan projektet bliver lavet, men de laver stadig et stykke arbejde, som er værd at tænke over i løsningen.



Opsummering

O-løbere og trænere er sat som ressourcepersoner, derfor vil gruppen trække på den viden som disse grupper kan give. Dette vil ske i form af interview og e-mail korrespondancer. Disse vil desuden være målgruppen for projektet.

2.3 Typisk o-løbs træning

I dette afsnit vil en typisk o-løbs træning blive beskrevet, og der bliver analyseret på nogle af de problemer o-løbere støder på i forbindelse med en træning. Afsnittet er udarbejdet ud fra information af Jens Børsting.

Før en træningssession kan starte, skal der først søges tilladelse hos Naturstyrelsen, til den skov, hvor der skal løbes.

Til hver træning beslutter træneren hvilke fokuspunkter der skal trænes, eksempelvis tekniske fokuspunkter, som højdekurve læsning, eller mere fysisk som konditionstræning. Herefter planlægges løbet i et computerprogram, som f.eks. Condes, hvor de enkelte ruter/baner tegnes på orienteringskort. Desuden udarbejdes der også et orienteringskort, som bruges til udsætning af

poster til træningen. Det meste af dette arbejde kan gøres hjemmefra, dog er det ofte sekretæren i klubben, der står for at printe orienteringskortene. Dagen før træningen, bliver alle poster typisk hentet i klubhuset og sat ud i skoven, denne proces tager omkring to timer. Der er flere forskellige typer af poster. De mest simple er en skærm, altså en farvet stofkasse, der blot indikere hvor posten er. Der findes også en større og besværligere udgave, der er en pind i jorden på ca. 1 m, med en skærm omkring og en elektronisk aflæser på toppen, som løberne kan bruge EMIT brikker til (der kan læses mere om EMIT brikker i afsnit 2.1.1). Det tager cirka en times ekstra arbejde hvis der vælges en af de større elektroniske poster.

På dagen mødes alle løbere og får instruktion om løbets fokuspunkter. Herefter uddeles baner alt efter niveau og kondition, hvor der er typisk 3-7 baner at vælge imellem. Løberne bliver derefter sendt ud i skoven, med et kort, kompas og evt. EMIT brik hvis det er en mulighed. Når løberne er færdige med deres tur, får de en udskrift over hvilke poster de har været ved, og hvor lang tid der er gået mellem hver post, hvis EMIT brikkerne har været taget i brug. Efter løbeturen, har løberne mulighed for at evaluere deres tur, ved at snakke sammen med andre løbere, og sammenligne vejvalg og strækninger. Vejvalg foregår udelukkende efter hukommelse og det er ikke muligt at se forskel i hastighed på delstræk, kun hele stræk mellem 2 poster. Efter træning, eller dagen efter, skal alle poster samles ind igen og pakkes ind i klubhuset.

Hvordan en typisk o-løbstræning foregår afhænger af klubben og løberne, men som en generel hovedregel kan det siges, at der inden en træning er blevet lavet 3 eller flere ruter der kan løbes. Hvordan tidtagningen foregår, og om der overhovedet er nogen form for tidstagning til træningen.

2.3.1 Problemer

Ud fra interviewet med Claus Bobach, er to problemstillinger blevet belyst, i forbindelse med en typisk o-løbs træning. Disse to problemer, er forberedelsestiden og evalueringen af træningen. Problemerne er her beskrevet.

Der bliver brugt en time til halvanden på forberedelse af en træning eller et løb, og derudover skal alle posterne sættes ud, og efterfølgende pakkes sammen igen. Dette svarer til yderligere 2,5 til 3 timers arbejde.

Problemet er at der skal folk til at gøre det, og i små frivillige foreninger, er der ikke nogen der kan blive betalt løn for at gøre det. Der kan derfor udelukkende satses på frivillige, der gider at tage ansvaret for det.

Derudover er det et endnu større arbejde, hvis der ønskes en form for tidtagning på posterne, da de elektroniske poster tager en time mere for henholdsvis at stille op og samle sammen.

Problemet med evaluering af træningen for både træner og deltagere, er problemerne med at kunne se tider på delstræk og vejvalg. Fordi to personer har løbet cirka lige hurtigt mellem to poster, behøver det ikke at betyde at de begge har fundet den hurtigste vej. Det kan fx være at den ene var hurtigere på den første del på grund af vejvalg, mens den anden var hurtig på den sidste del, og det i virkeligheden ville være hurtigere at vælge en kombination af de to ruter. Som det gøres nu sammenlignes der primært ud fra de enkelte løberes hukommelse, hvilket gør sammenligningen upræcis. Nogle løbere evaluere selv derhjemme ud fra GPS dataene fra deres GPS ure. Konsekvensen ved dette er at der ikke sammenlignes med andre løbere, og dermed vil vedkommende have svært ved at se hvor der kunne være taget bedre vejvalg på ruten.

I samarbejde med træner Claus Bobach, har gruppen belyst to problemstillinger, der opstår i

forbindelse med en typisk o-løbs træning. Disse to problemer, er først det tidskrævende arbejde, der ligger i forberedelsen af en træning, og dernæst den mangel på evaluering, der foretages, hvis ikke løberne selv ligger en aktiv indsats i at huske deres vejvalg, og få snakket med de andre løbere.

2.4 Eksisterende løsninger

I dette afsnit vil de nuværende IT-løsninger, som har indvirken på problematikker vedrørende o-løbstræning, blive forklaret og analyseret.

2.4.1 Endomondo

Endomondo er en applikation som bruges til løb, hvori en træningsplan kan udformes. Applikationen vil her have to versioner, en gratis og en udvidelses-version som kan købes. Endomondo bruger Google Maps som kort i deres applikation [Endomondo, 2015]. Funktionaliteten der her vil blive beskrevet er for gratis versionen: Hovedmenu med otte funktioner:

- Newsfeed, som bruges til at kommunikere med andre brugere af Endomondo, hvor brugere kan opmuntre, samt udfordre hinanden til løb og andre udfordringer.
- Notifikationer, hvor brugeren kan se sine udfordringer og lignende fra andre brugere. Her kan brugeren acceptere eller afslå udfordringer.
- Historik, hvor tidligere løb oplagres til genvisning og statistik.
- Kort, hvor brugeren ved hjælp af mobilens GPS-enhed kan få et kort over ruten der er løbet.
- Opgrader-nu, hvor gratis versionen kan betales til opgradering.
- Venner, hvor brugeren kan overvære venner fra sociale medier.
- Træningsplanen, der bruges til at indstille applikationens hjælpefunktioner til træning, hvor brugeren selv sætter mål for træningen, intensiteten ved træningen, og hvilken type af træning der udøves.
- Indstillinger, hvor de generelle indstillinger for programmet kan tilpasses.

Dette er en velfungerende applikation til formålet, løb og træningsplan, men i det at en o-løber skal bruge visningen af løbet på et o-løbskort, kan dette blive problematisk, da brugeren ikke selv kan sætte kort ind i programmet.

2.4.2 EMIT brikker

En EMIT brik, er et elektronisk apparat, der kan registrere hvilke poster der besøges. EMIT brikken sidder fast på fingeren, vha. en elastik. Et billede af EMIT brikken kan ses på figur 2.3.

Måden en EMIT brik fungere på, er at den stilles på en startpost ca. 5 sekunder før løbet bliver sat i gang. Dette vil genstarte EMIT brikken, således at den kan notere de rigtige tider. Ved hver post, der skal besøges, er der en kontrolpost, der ligner startposten, som EMIT brikken skal lægges på i ca. $\frac{1}{2}$ sekund. Dette vil registrere hvor lang tid der er gået mellem forrige post og nuværende post.

Til sidst i løbet, vil der være en målpost, hvor EMIT brikken endnu engang skal lægges på, således at den sidste tid bliver noteret. Herefter skal EMIT brikken afleveres til de ansvarlige,

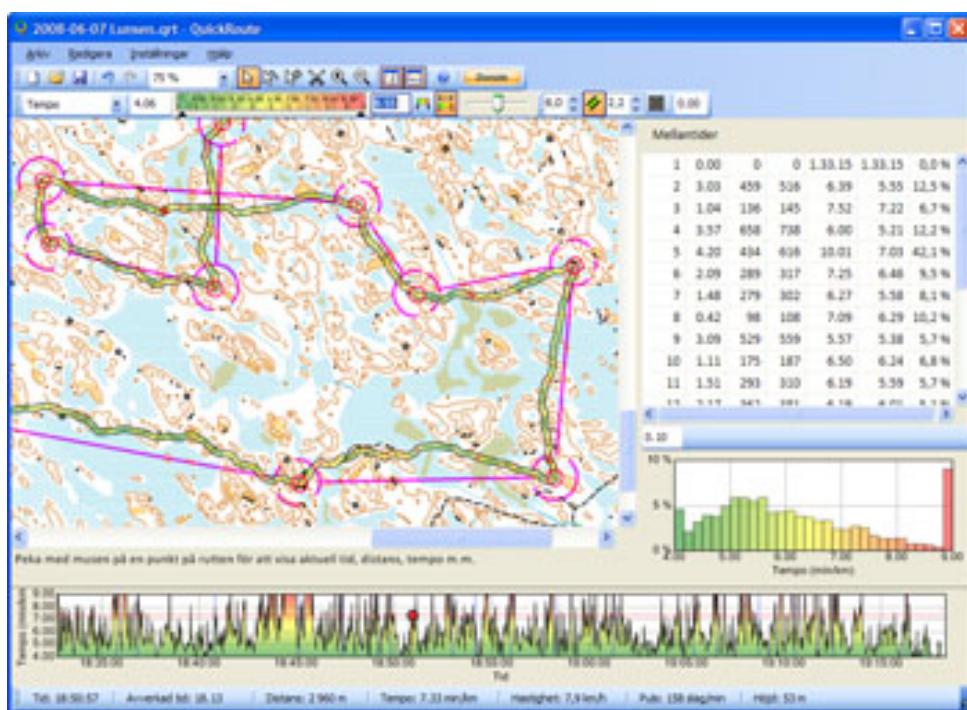


Figur 2.3. EMIT brik

som vil give løberen en udskrift af tiderne. Disse tider kan så derefter anvendes til sammenligning og diskussion med andre løbere. [Odense Orienteringsklub, 2015]

2.4.3 QuickRoute

Igennem gruppens interview, beskrev Claus, at QuickRoute er en eksisterende løsning til kortlæggelse og rutevisning. For at bruge denne løsning skal løberen have et Garmin ur, eller andre GPS enheder, som kan generere en GPX fil over ruten. I QuickRoute kan et kort fra OCAD (OCAD er et program til at lave o-løbskort) lægges ind, og ved hjælp af Garmin ure kan en rute blive vist, hvor forskellige parametre kan blive afbildet. Hastighed, minutter per kilometer, hjertefrekvens, højdemeter og afvigelse fra retningen mellem to punkter kan blive afbildet. Dette bliver vist ved en farvekode der følger ruten, og gennem hele ruten vil denne farvekode variere efter hvert interval af GPS-signal. Med den rigtige serie af Garmin ure, kan der tilmed tages tid på, hvornår en post er nået, så en statistisk model kan beskrive tiderne mellem posterne. Hvis musen holdes over et punkt på ruten, kan følgende information om punktet vises: Klokkeslæt, tid brugt i alt på rute, samlet distance løbet til det punkt, nuværende minutter per kilometer, nuværende hastighed i km/t, nuværende hjertefrekvens, nuværende højdemeter, nuværende afvigelse mellem to punkter, nuværende længde- og breddegrader. Mange af disse informationer kan afbildes på et histogram. [QuickRoute, 2015]



Figur 2.4. Screenshot af QuickRoute

Derudover kan ruten efterfølgende integreres på Google Earth, så der kan ses en 3D model af ruten der er løbet. Denne løsning er meget gennemført, hvor mange funktioner kan benyttes, dog kan antallet af funktioner og parametre virke overflødig for en amatør o-løber. For en rutineret eller professionel o-løber vil denne løsning give et godt indblik i, hvordan løbet er foregået og hvor personen kan udvikle sig. QuickRoute kan også bruges til sammenligning mellem løbere, dette kræver dog at begge løbere GPX-filer bliver lagt over på samme computer, og loaded ind i programmet.

2.4.4 TracTrac

TracTrac er en samlet løsning til livetracking med replay funktion af forskellige sport events. Til dette bruges TracTrac's egne GPS enheder. Disse er ca. på størrelse med en cigaretpakke og vejer 113 gram.

TracTrac bruges til o-løb så der live kan følges med i hvor o-løberne er ude i terrænet, som f. eks. til stævner og konkurrencer. Derudover bruges TracTrac i høj grad til at analysere de enkelte løberes ture efter de har løbet, da TracTrac har en velfungerende replay funktion.

Når TracTrac skal bruges i forbindelse med o-løb, laves der først et o-løbskort som uploades til TracTracs servere. Derefter indsættes præcise punkter på kortet som repræsentere hvor hver enkelt post ligger, så TracTrac kender positionerne på alle posterne. Herefter skal hver enkelt GPS enhedsnummer sættes sammen med en løber, så det bliver tydeligt hvem der løber hvor. Under løbet sender GPS enhederne deres position til serverne som viser disse på kortet. Alt dette sker live. En af funktionerne som gør TracTrac ekstra brugbar i forbindelse med o-løb, er dens mulighed for at flytte løbere tilbage til start og afspille deres tur samtidig, så der kan ses præcis hvordan de løb i forhold til hinanden, selvom de i virkeligheden startede forskudt. Dette kan også gøres selvom løbet er live. En løber der er foran, kan flyttes tilbage, så vedkommende løber samtidig, som en anden løber der er startet senere. Dette giver mulighed for grundig analyse og sammenligning af løbernes vejvalg og hastighed, både under og efter løbet.

TracTrac kan stort set alt der er brug for til live visning og analyse af o-løb. Selve GPS-enheden koster 150 EUR, altså lige godt 1120 DKK. Derudover skal der bruges et sim, data kort og enhedslicens til 88 EUR om året pr. enhed, hvilket vil svare til ca. 650 DKK. Som det sidste skal en system-licens bruges, denne koster 990 EUR årligt, hvilket er ca. 7380 DKK. Dette produkt er i sådan en prisklasse, at de små amatørklubber ikke har råd til det. [TracTrac, 2015]

Opsummering

Ud fra ovenstående analyse af eksisterende løsninger, kan det konkluderes, at der findes metoder der kan hjælpe o-løbere med at optimere deres træning. I form af QuickRoute, er der mulighed for at analysere de forskellige ruter løberen har kunnet vælge imellem, samtidig sammenligne med andre løbere. Dette kræver dog at løberne har en GPS tracker, som et Garmin ur, hvilket gør en sammenligning vanskelig. TracTrac giver mulighed for at det kan følges live, og dermed er der også mulighed for at se replay af løbeturen, herudover kan der også sammenlignes med tal ud fra ruterne. Denne løsning kræver dog ikke Garmin ure, men kræver nogle TracTrac enheder, og noget licens, der hurtigt kan blive for dyrt, for de mindre foreninger og derfor ikke helt optimalt. Gruppen vurderer derfor, at en mulig løsning på dette, er anvendelse af mobiltelefoner som GPS enhed. En mobiltelefon er let tilgængelig, og der er mulighed for GPS tracking på den, og der behøves ikke ekstra omkostninger, som fx. ved et Garmin ur eller TracTrac licens. Endomondo bruger mobilens GPS signal, men da der ikke en nogen replay funktion eller mulighed for at lægge andet end Google Maps' kort bag ruten, er denne løsning ikke velegnet til o-løb.

Problembeskrivelse 3

I dette kapitel vil problematikkerne i denne problemanalyse blive beskrevet, og herefter vil rapporten afgrænses til et delområde, hvoraf en problemformulering vil uddrages.

3.1 Problemafgrænsning

I udarbejdelsen af problemanalysen blev der fundet to problematikker ved o-løbs træninger. Den første omhandler opsætningen af poster, normale og elektroniske, da trænerne bruger flere timer på at forberede en træning. Posterne skal både ud før træning og samles ind igen efter, hvor de elektroniske poster er mere tidskrævende end de små poster. Udover dette, er arbejdskraften typisk frivillig, så trænere og andre klubmedlemmer må satse på, at en person frivilligt vælger at bruge tid på forberedelsen. En anden problemstilling er evaluering af løbere under træning. Problematikken opstår i det, at evalueringen kun kan ske ved, at to løbere sætter sig ned og kigger på kortet over den udførte rute, og prøver at erindre hvilke vejvalg de har taget. De har ikke nødvendigvis taget den samme rute, trods de har en næsten ens samlet tid på løbet, hvor den hurtigste rute er muligvis en kombination af de to ruter. Gruppen har valgt at afgrænse sig til evaluering af træningen for o-løbere, da gruppen ser størst potentiale i dette. Dog ser gruppen også en mulighed for at integrere en løsning til opsætning af poster.

I de eksisterende løsninger blev der fundet en del løsninger, dog løste de ikke alle problematikker som gruppen er kommet frem til, eller der opstod nye, i form af prisen på produktet. Endomondo anvender Google Maps, hvilket ikke er præcist nok til o-løb, da skove kun er repræsenteret med grøn farve. EMIT brikker anvendes til tidtagning under træningen, men denne løsning er ikke særligt anvendelig, da den blot giver strækvider, så løbere kan kun evaluere hvor hurtig de på de specifikke stræk, men ikke de vejvalg de har foretaget. QuickRoute er et godt værktøj til evaluering af løberen, men kræver et GPS-ur. Derudover er det vanskeligt at sammenligne med andre løbere med dette værktøj. TracTrac er en allerede kendt løsning, men som beskrevet i eksisterende løsninger, skal brugere have licens, samt firmaets egne GPS-enheder, hvilket bliver for dyrt for de mindre foreninger.

Ud fra problemanalysens resultater, har gruppen konkluderet, at en software løsning skal bruge GPS-enheten i en mobiltelefon, og dermed være en billigere løsning end eksempelvis TracTrac. Løsningen skal også kunne indsætte o-løbskort, så løberne kan se præcist hvor de har været i terrænet, i stedet for at bruge Google Maps som Endomondo.

Dette kan optimeres betydeligt med en telefonbaseret softwareløsning, hvor der i afsnittet om eksisterende løsninger er blevet beskrevet, at denne løsning skal kunne hjælpe til evaluering af træningen, uden brug af et GPS-ur.

3.2 Problemformulering

Ud fra ovenstående afgrænsning er følgende problemformulering blevet udarbejdet:

Hvordan kan en telefonbaseret softwareløsning optimere evaluering og traening af o-løbere?

- Hvordan kan løberne sammenlignes?
- Hvordan følges løberen rundt på ruten?
- Hvordan kan løsningen gøres brugervenlig?

Kravspecifikationer 4

Dette kapitel omhandler en kravspecifikation som er blevet udarbejdet, ud fra problemerne fra problemanalysen og idéer til projektets løsning, som løsningen vil tage udgangspunkt i. Først vil en optimal løsning blive forklaret, hvilket er idéer uden nogen form for begrænsninger af tid, erfaring eller kunnen. Dernæst vil gruppens reelle løsning være beskrevet, som så vil tage hensyn til de fornævnte begrænsninger. Brugervenlighed og projektets løsningsstrategi vil også blive beskrevet.

4.1 Optimale løsningsforslag

Den optimale løsning vil bestå af to dele, en mobiltelefon med GPS og en webservice som kan vise dataene sendt fra mobilen.

Mobilen skal kunne:

- Måle, sende og optage GPS positionen på løberen undervejs på ruten.
- Tilslutte sig en bestemt bane, som træneren har opsat og lagt på serveren inden træning.
- Mobilen må ikke hjælpe løberen undervejs i løbet, med at vise positionen på ruten.

Webservicen skal kunne:

- Opsætte baner inden løbet.
- Have et klub- og brugersystem, hvor brugeren kan tilknytte sig en bane inden løbet.
- Være kompatibelt med GPS-ure.
- Vise løbernes position på kortet.
- Vise diverse statistik og data for løberens tur på banen. Dette vil være tider, afstande og hastigheder, samt en grafisk visning af den rute løberen har løbet.
- Sammenligne to løberes tur på samme rute.
- Vise et grafisk replay af den rute løberen har løbet, med mulighed for at afspille flere løbere samtidig og dermed sammenligne vejvalg.

Mobilapplikationen vil have en simpel brugergrænseflade hvor brugeren vil have mulighed for at vælge den bane vedkommende skal løbe. Herefter vil brugeren have mulighed for at trykke start, hvorefter mobilen kun vil vise hvor lang tid der er brugt indtil videre og en afslut knap. Under løbet sender mobilen løbende data om GPS position og tiden siden start. Når turen er løbet færdig trykker brugere på afslut og applikationen vil sende de sidste data og vise nogle resultater. Dette kan eksempelvis være tid i alt, gennemsnitshastighed, men også data om løberen sammenlignet med andre på samme bane. Dette kan f.eks. være hvor mange sekunder, efter den hurtigste løber, der er brugt på de enkelte stræk.

Inden løbet skal træneren eller arrangøren, kunne uploadet den bane som personen har lavet, til serveren. Banerne kan vælges fra applikationen på mobilen, af de forskellige løbere. Efter løbet

skal brugerne som sagt kunne uploadede sine resultater til webservicen fra mobil applikationen, under og efter et endt løb. Disse resultater skal så kunne sammenlignes med de andre løbere, der har løbet den samme bane. Ud over at sammenligne statistik over løberens rute, som beskrevet ovenfor, skal den også kunne vise den rute som løberen har løbet grafisk. Derudover skal der være en replay funktion, så de vejvalg som løberen tager, kan ses. Så kan der indsættes flere løbere på den grafiske visning, så flere løberes vejvalg kan sammenlignes.

4.1.1 Krav til optimal løsning

Ud fra ovenstående har gruppen udarbejdet en række krav, disse blev sendt til Jens Børsting, som kom med rettelser og tilføjelser.

- Applikationen tilknyttet løsningen skal kunne køre på android smartphones og apple iphones.
- Løsningen skal være en webservice.
- Samme login-system for både applikation og webservice.
- Webservicen skal kunne oprette og gemme baner i form af o-løbskort med indtegnede poster.
- Webservicen skal grafisk kunne vise en eller flere GPS ruter uploadet fra applikationen ovenpå den tilknyttede bane.
- Webservicen skal have følgende replay funktionaliteter:
 - Det skal være muligt at til/fravælge løbere til den viste bane, evt. mulighed for at tilpasse halelængde (grafiske visning af løberens seneste antal punkter) og farve.
 - Zoom funktion på kortet.
 - Det skal være muligt at starte og pause afspilningen af GPS Dataene.
 - Det skal være muligt at sætte tempoet til 1, 2, 5, 10 og 20 gange normal hastighed.
 - Scrolling funktion for at komme hurtigt frem til et bestemt tidspunkt.
 - Det skal være muligt at vælge én post (punkt som alle passerer) hvor alle viste løbere bliver samlet ved. Der skal kunne defineres/vælges flere poster/punkter.
- Webservicen skal kunne vise følgende data for hvert stræk:
 - Den enkelte løbers navn.
 - Den enkelte løbers placering i løbet efter strækket.
 - Den enkelte løbers stræk længde.
 - Den enkelte løbers tilbagelagte rute.
 - Den enkelte løbers samlede tid efter strækket.
 - Den enkelte løbers tid på strækket.
 - Den enkelte løbers tid i forhold til den hurtigste på strækket.
 - Den enkelte løbers gennemsnitsfart på strækket.
 - Den enkelte løbers top fart på strækket.
 - Evt grafisk sammenligning mellem to løbere ved at tegne streger mellem de to løberes placering på samme tid efter udgangspunktet.
- Applikationen skal kunne optage gps-data under løb og ved løbets afslutning sende disse til webservicen.
- Applikationen skal kunne tilknytte sig en bane, oprettet på webservicen.
- Applikationen skal kunne sortere banerne ud fra klub.
- Applikationen må kun vise tid brugt hidtil og en afslutningsknap på skærmen under løbet.
- Applikationen skal kunne tilgå og grafisk vise webservicens replay funktionalitet.
- Applikationen skal være brugervenlig, i den forstand, at det skal være let at vælge en bane, starte et løb, afslutte et løb og bruge replay-funktionen. Navigering i applikationen skal være simpel.

4.1.2 Mockup af optimal løsning

Mobilenhed

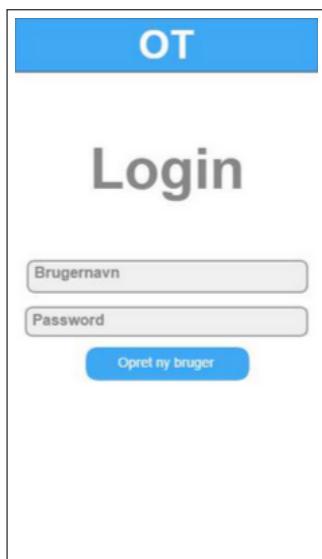
Gruppen har gjort sig nogle tanker om hvordan den optimale løsning skal se ud og hvilke funktioner den skal have. Udfra dette er der blevet lavet nogle mockups af hvordan gruppen forestiller sig den optimale løsning kunne komme til at se ud.

Figur 4.1 viser det første der ses når applikationen åbnes på ens smartphone. Der er to muligheder på denne side, enten kan der logges ind med en eksisterende bruger, eller også kan der oprettes en ny bruger. Hvis det lykkes at logge ind på applikationen, vil det næste der ses være figur 4.2, hvor der ses tre knapper. Der er mulighed for at påbegynde et nyt løb, der kan ses på de løb brugeren allerede har løbet, eller der kan ændres i ens indstillinger.

Hvis menupunktet ”Nyt løb” vælges, bliver brugeren ført videre til figur 4.3. Øverst i applikationen kan der ses en dropdown menu, hvor der er mulighed for at vælge den o-løbs forening som der løbes for. Under den valgte o-løbs forening, vil der så være et antal baner, brugeren kan vælge at løbe. Det er foreningen selv der skal uploadde de baner som er muligt at løbe, hvor det så er planen at en forening skal have et administrator login, for at kunne uploadde disse baner.

Trykkes der på en af banerne, eksempelvis bane 4, føres brugeren ind på siden set på figur 4.4. Her ses nogle informationer om det valgte løb, samt en stor ”Start” knap, hvor løbet selvfølgelig startes. De informationer der findes om løbet er sværhedsgraden på banen, ca. længden på banen, samt hvor mange poster løberen skal igennem. Hvis bane 4 er den bane brugeren gerne vil løbe, trykker brugeren på ”Start” knappen. Brugeren kommer så ind på siden vist på figur 4.5. Denne figur afbilleder hvad der kan ses under løbet. Det er ikke muligt at se noget kort, da applikationen ikke skal vise løberens position på kortet. Det eneste der kan ses, er hvilken bane der er valgt, den tid der er gået siden der er trykket ”Start” og en slide knap til at afslutte løbet.

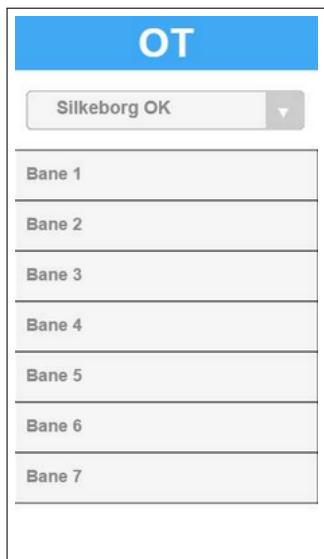
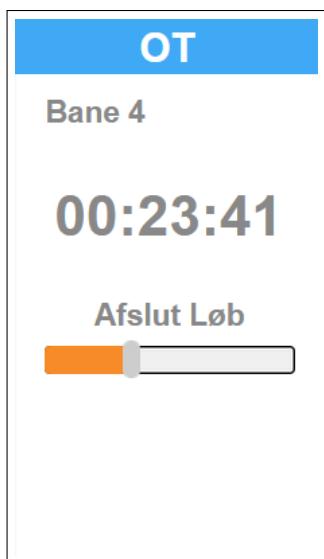
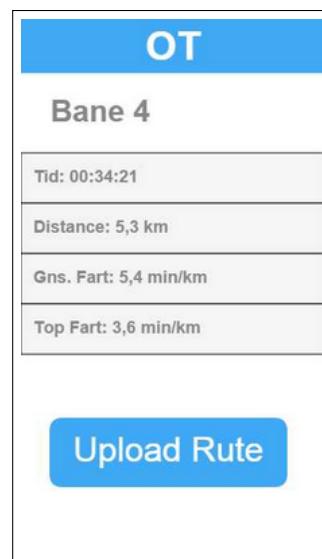
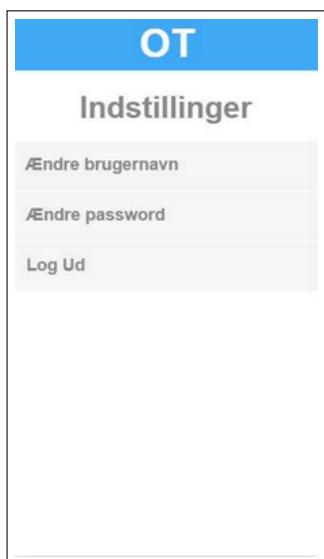
Når løberen har været igennem alle poster og kommet i mål, bruges slide knappen ”Afslut løb”, der fører brugeren ind på figur 4.6. Denne side skal vise informationer om den netop færdiggjorte bane. Her ses tiden fra start til slut, den distance løberen har løbet og gennemsnitsfart. I bunden af siden ses en ”Upload Route” knap. Trykkes der på denne knap uploades ens rute, over den



Figur 4.1. Login system



Figur 4.2. Mainmenu

**Figur 4.3.** Menupunktet - Nyt løb**Figur 4.4.** Informationer af valgte bane**Figur 4.5.** Applikationen underløbet**Figur 4.6.** Applikationen ved afsluttet løb**Figur 4.7.** Menupunktet - Indstillinger**Figur 4.8.** Menupunktet Mine løb

bane der lige er blevet løbet, til webserveren. Nu kan løberens resultater sammenlignes med andre løberes resultater, på den specifikke bane.

Kigges der igen på figur 4.2 og menupunktet "Indstillinger" vælges, bliver brugeren ført ind til figur 4.7. Her kan brugeren ændre brugernavn, password og vælge at logge ud fra applikationen.

Hvis menupunktet "Mine løb" vælges i figur 4.2, kan brugeren se en liste over sine gennemførte løb, som ses på figur 4.8. Listen er sorteret efter dato, hvor de nyeste løb vil være øverst. Derudover kan der ses hvilken bane der er blevet løbet, samt hvilken forening der er blevet løbet ved. I bunden af siden er der en "Find andre baner" knap, hvor der kan søges efter alle de baner der findes i databasen. Derudover kan der søges efter en specifik o-løbs forening eller dato, på denne måde kan der findes løb, der er blevet løbet på samme bane og dag, som brugeren selv har løbet, som ses på figur 4.9. De fundne baner kan så bruges til at sammenlignes med brugerens egne resultater på samme bane.

Hvis brugeren har valgt at ville sammenligne løbere på en specifik bane, vil brugeren se en side lignende figur 4.10. Denne side viser replay funktionen, som vil være den samme som er i webservicen. Replay funktionen er beskrevet nedenfor.

Wbservice

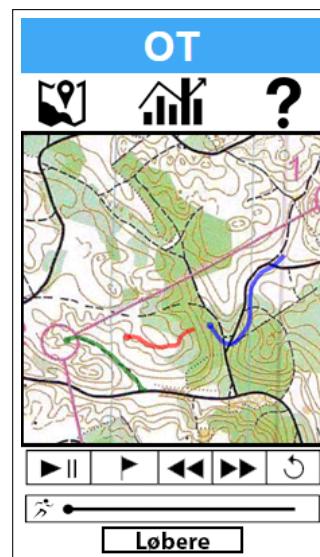
Interfacet på webservicen ligner meget interfacet på applikationen fra figur 4.10. I venstre side, på figur 4.11, ses en menu, hvor første punkt er kortet som ses på billedet. Andet punkt er en detaljeret statistik over løberne og sidste punkt er en hjælpefunktion.

På figur 4.11 ses funktionen "Kort". Kort funktionen viser et orienteringskort, hvor løbernes rute kan genspilles. Øverst i venstre hjørne sidder et lille kort, der fungere som et oversigtskort. Oversigtskortet skal gøre det lettere at navigere rundt, da brugeren kan have en lettere forståelse af hvor de befinder sig på det store kort. Nede i højre hjørne findes en til og fravalgs-funktion af de løbere, der er blevet tilvalgt til sammenligningen.

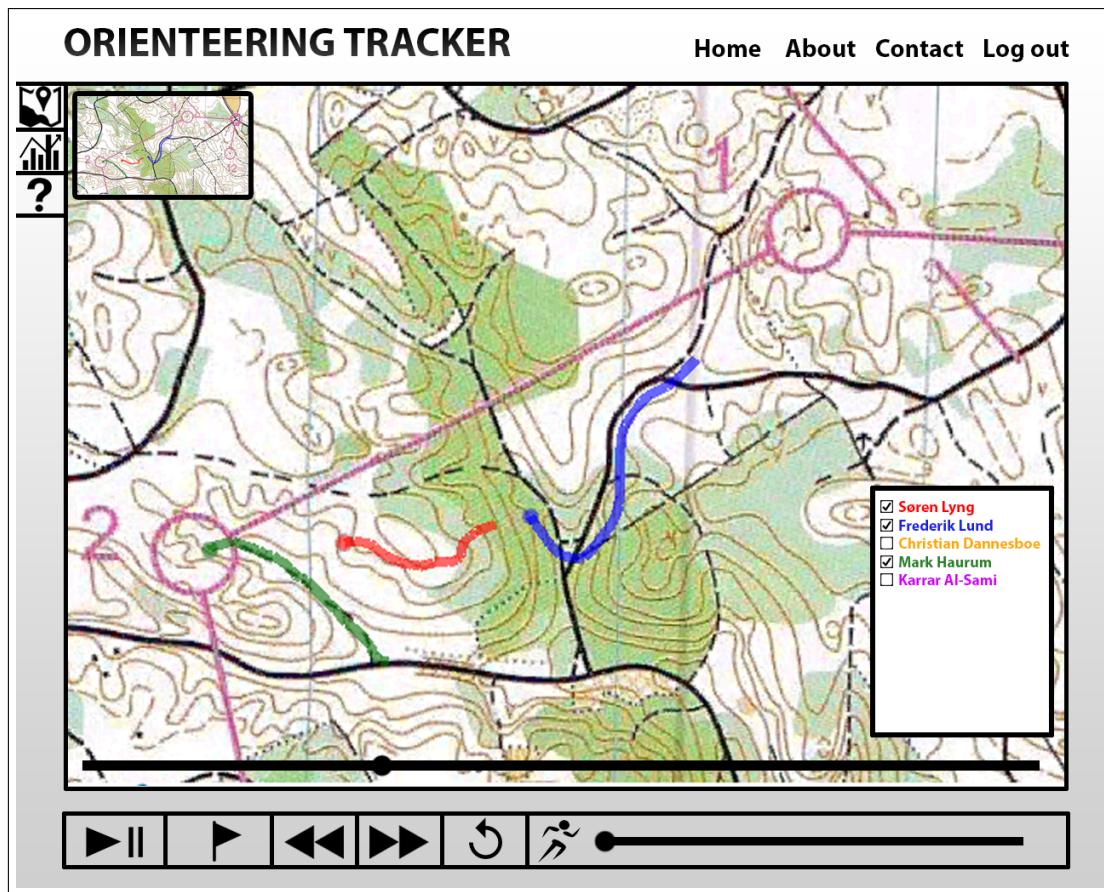
I bunden af figur 4.11 ses en medioplayer. På denne findes der seks forskellige funktioner, hvor de her under vil blive beskrevet fra venstre mod højre:



Figur 4.9. Find andre baner søgefunktion



Figur 4.10. Evaluering- og sammenligningsværktøj



Figur 4.11. Interface til webservice

- Play/pause knap. Bruges som det kendes fra andre mediaplayere, til at starte og pause i sin afspilning.
- Vælg post funktion. Her kan der vælges mellem forskellige poster, hvorfra brugeren ønsker at se løberne løbe fra. Dette gør at løberne kan ses begynde fra samme post samtidigt, for bedre at kunne sammenligne deres vejvalg på deres stræk mellem posterne.
- Forrige post. Sætter alle løbere til at løbe fra forrige post.
- Næste post. Sætter alle løbere til at løbe fra næste post.
- Refresh funktion. Fjerner alle indstillinger der er blevet tilføjet, og afspiller løbet som normalt.
- Afspilningshastighed. Her kan afspilningshastigheden ændres.

Lige over mediaplayeren kan der ses hvor langt, brugeren er inde i genafspilningen af løbet. Den runde sorte plet, angiver hvor brugeren er nået til. Her kan brugeren så også valgfrit kan springe rundt i genafspilningen.

Figur 4.12 viser menupunktet om detaljeret statistik over løberne. Under denne side kan der ses en række informationer om løberen under genafspilingen af løbet. Dette skal være en anden måde at sammenligne løberne, frem for kun at have den grafiske sammenligning. De ting der er tiltænkt, at skulle vises er løberens:

- Navn.
- Position i løbet.
- Hvor lang en distance de har tilbagelagt på det sidste gennemførte stræk.
- Tiden for løbet i alt.

ORIENTEERING TRACKER								
	Home	About	Contact	Log out				
	Navn	Pos.	Stræk	Tid	Stræk tid	Tidsforskel	Hastighed	Top hastighed

Figur 4.12. Detaljeret sammenligning af løbere

- Tiden på det sidste gennemførte stræk.
- Tidsforskellen mellem de forskellige placeringer, eksempelvis tidsforskellen mellem nummer et og nummer to i løbet.
- Gennemsnitshastighed.

Sidste menupunkt i venstre side er hjælpefunktionen. Meningen med denne funktion er, at den kunne hjælpe forvirrede brugere med brugen af programmet, hvis dette skulle blive en nødvendighed.

4.2 Reelle løsning

Ud fra listen over krav til den optimale løsning i afsnit 4.1.1, har gruppen valgt at prioritere hvad løsningen skal indeholde. Gruppen har ud fra den resterende tid og gruppens erfaring, valgt at prioritere replay funktionen med tilhørende kort højest. Selve projektet handler om at kunne sammenligne og evaluere en o-løbs træning, derfor er det vigtigt at kunne se de gode og dårlige beslutninger, en o-løber har taget på hans eller hendes rute. Som udgangspunkt vil der blive arbejdet på at kunne genspille en rute med én løber, derefter at kunne tilføje flere løbere, for at kunne lave en sammenligning. Derudover har gruppen valgt at prioritere den tekstdbaserede sammenligning/statistik over løberne lige under replay funktionen, som ses på figur 4.12. Dette gøres for at kunne give brugeren flere sammenlignings muligheder. Replay funktionen giver et godt overblik over bl.a. vejvalg, men den tekstdbaserede sammenligning er eksempelvis mere detaljeret i forhold til hastighed, tid og distance.

Mobil-delen i projektet har gruppen valgt at simulere, ved at optage GPS signaler på en mobiltelefon, filerne med længde- og breddegrader lægges over på computeren, og arbejdes med derfra. Samtidigt vil et kort anvendt i programmet kræve, at brugeren selv har et geostationært kort, dette vil blive nærmere forklaret senere i rapporten.

Nedenfor ses de krav som gruppen har valgt at tage udgangspunkt i.

- Det skal være muligt at til/fravælge løbere til den viste bane, evt. mulighed for at tilpasse halelængde og farve.
- Zoom funktion på kortet.
- Det skal være muligt at starte og pause afspilningen af GPS dataene.
- Det skal være muligt at sætte tempoet til 1, 2, 5, 10 og 20 gange normal hastighed.
- Scrolling function for at komme hurtigt frem til et bestemt tidspunkt.
- Det skal være muligt at vælge én post (punkt som alle passerer) hvor alle viste løbere bliver samlet ved.

Den tekstdbaserede løsning skal kunne vise følgende den enkelte løbers:

- Navn.
- Placering i løbet efter strækket.
- Placering på strækket
- Stræk længde.
- Tilbagelagte rute.
- Samlede tid efter strækket.
- Tid på strækket.
- Tid i forhold til den hurtigste på strækket.
- Gns. fart på strækket.

4.2.1 Brugervenlighed

I gruppens problemformuleringen lød en af underpunkterne ” Hvordan kan løsningen gøres brugervenlig? ”. Gruppen har taget udgangspunkt i Rolf Molichs råd om brugervenlighed, som har et speciale i software engineering: http://www.dialogdesign.dk/Om_brugervenlighed.htm.

- **Nyttigt** – Programmet skal løse de opgaver, som brugeren ønskes løst.
- **Let at lære** – Programmet skal være let at lære at benytte, hvor indlæringstiden for at lære at løse bestemte opgaver skal være så kort som mulig.
- **Let at huske** – Her menes det at genindlærings tiden skal være så lav som muligt, for de brugere der har været væk fra programmet i en længere periode eller ikke bruger det så ofte.
- **Effektivt at bruge** – Programmet skal køre og løse opgaver så hurtigt muligt. Der skal være så få fejl som muligt, hvis det skulle ske der forekommer en fejl, så skal kvaliteten af fejlmeldingen være god.
- **Rart at bruge** – Dette er brugerens totaloplevelse af programmet, hvad deres mening er om det.

Rolf Molich nævner at alle disse egenskaber kan måles objektivt, hvor han giver eksempler som med et stopur eller spørgeskemaundersøgelser. Brugergrænsefladen som gruppen har udviklet, er derfor tforsøgt lavet så simpel og overskuelig som overhovedet mulig. Da dette dog er det første store program som gruppen har lavet, har gruppen taget den beslutning at fokusere mere på

at få funktionerne i programmet til at virke, frem for at tænke på brugervenlighed. Dette gøres pga. manglende erfaring med større programmer og den lille mængde tid der er til projektet. Brugervenlighed er dog stadig vigtigt i et godt program, så det vil gøres så simpelt som muligt.

4.2.2 Løsningsstrategi

I dette afsnit vil planen for programmet blive beskrevet, med en overordnet beskrivelse af løsningsstrategi, samt en beskrivelse af valg mht. værktøjer brugt i udviklingen af programmet.

I henhold til gruppens kravspecifikation, vil GPS dataen til programmet blive simuleret af en app til en smartphone. Gruppen har brugt en app kaldet, myTracks til dette formål. myTracks optager og sender GPS signaler i form af en GPX-fil, som bliver tilsendt via e-mail til brugeren. Som beskrevet i kravspecifikationen, skal brugeren sørge for at kortet anvendt skal være geostationært. Dette betyder, at der sammen med kortet medfølger en tekstfil med tal, der beskriver kortets placering på jordkloden. Med denne fil kan koordinater beskrevet med UTM-systemet (beskrives i teoriafsnittet), oversættes til pixels, og derved gøre det muligt at tegne afstandstro GPS-ruter på kortet. I dette projekt anvendte gruppen QLandkarteGT til at geostationære et kort, som blev anvendt til udvikling af programmet.

myTracks anvender koordinater i Lat/Long-formatet, hvor det geostationære kort oversætter UTM-koordinater til pixels. Dette betyder, at en konvertering fra Lat/Long til UTM er nødvendig, denne konvertering vil blive beskrevet i teoriafsnittet.

Programmet vil her bestå af to dele, hvor den første del vil behandle data i form af pixels til at tegne ruter på kortet. Anden del af programmet vil bruge UTM-koordinater til at udregne data over løberne, som f.eks. hastighed og distance.

Til udviklingen af dette program har gruppen valgt at benytte Visual Studios Windows Forms, da gruppen ønsker at have en grafisk brugergrænseflade.

Teori 5

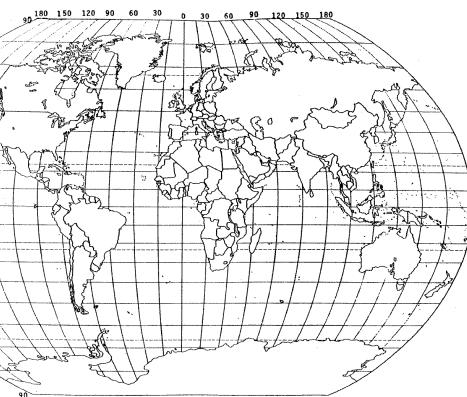
I dette kapitel vil de anvendte teorier blive beskrevet. Herunder længde- og breddegrader, Universal Transverse Mercator systemet og konverteringen mellem dem.

5.1 Længde- og breddegrader

I et geografisk koordinatsystem, er jorden inddelt i længde- og breddekredse. Idet jorden er rund, er disse længde- og breddekredse målt i grader, og de bliver i stedet kaldt længde- og breddegrader.

Længdegrader beskriver de vertikale linjer på verdenskortet på figur 5.1, som går fra nordpolen til sydpolen. Disse bliver også betegnet som meridianer. Meridianen der går gennem Greenwich kaldes nulmeridianen, hvorefter den første vestlige længdegrad bliver udtrykt som 1° vestlig længde, og det samme gælder for den første østlige længdegrad, som vil være 1° østlig længde. Disse længdegrader bliver herefter talt op, i hver deres retning. Idet en cirkel er 360° , er de sidste meridianer på hver sin side af verdenskortet kaldet 180° østlig/vestlig længde. Det skal desuden også nævnes, at længdekredsene ikke nødvendigvis har den samme afstand, da længdekredsene har mindre indbyrdes afstand jo tættere på polerne de er.

Breddegrader beskriver de horisontale linjer i verdenskortet på figur 5.1. Disse linjer er parallelle med ækvator, og deres indbyrdes afstand bliver derfor hverken mindre eller større, men deres omkreds bliver kortere alt efter hvor tæt på polerne de befinder sig. [Britannica, 2015]



Figur 5.1. Verdenskort med længde- og breddegrader

5.2 Universal Transverse Mercator systemet

Universal Transverse Mercator systemet, forkortet UTM, er en anden måde at finde lokationer på jordkloden. Denne metode gør brug af, at jordkloden, som reelt set er rund, bliver omformet til en flade, ligesom et stykke papir. Her bliver jorden inddelt i 60 zoner, hvor hver zone er opdelt fra

øst til vest, med numre, og syd til nord med bogstaver. De østlige og vestlige, startende fra 180 meridianen, regnes fra vest til øst med 6° mellemrum. Enkelte zoner er dog blevet gjort bredere eller smallere. Hver zone er nummereret efter deres placering fra vest.

Nordligt og sydligt er inddelt mellem 80° S bredde, og 84° N bredde, med en afstand på 8° , udover en enkelt zone der er 12° . Hver zone er navngivet ved et bogstav, fra C-X, startende med C i den sydligste zone. Alle zonerne i UTM systemet er baseret på en Transverse Mercator projektion, som gør det muligt at kortlægge en region uden stor forvrængning af jordklodens ellers runde form. Hvis jorden blev kortlagt i én stor region, vil dette forårsage store afstandsfejlberegninger, da der ikke vil blive taget højde for jordklodens runde form. Det er her at Transverse Mercator projektionen er smart, da den sørger for en lav grad af forvrængning, da det er smalle breddegrader (6°) der typisk anvendes til zonerne.

Ved positionering gennem UTM systemet, oplyses følgende: Først en zone fra UTM, som f. eks. 17T. Dernæst en østlig og en nordlig afstand. Den østlige udregnes ud fra den centrale meridian. Den centrale meridian er den længdegrad som går igennem midten af zonen. Denne er sat som 500.000 m østlig for at undgå negative tal, dvs. et punkt vest for meridianen giver et tal under 500.000 m, hvor et tal øst for vil være over 500.000 m. Den nordlige afstand beregnes på følgende måde: Hvis punktet er på den nordlige halvkugle, beregnes den nordlige afstand som afstanden til ækvator. Hvis punktet er på den sydlige halvkugle, beregnes den nordlige afstand som 10,000,000 m minus afstanden til ækvator, for at undgå negative tal. [Agency, 2009]

5.3 Konvertering

Som beskrevet i afsnittet løsningsstrategi, er en konvertering mellem længde- og breddegrads systemet til UTM systemet nødvendig. Dog er udregningen ret kompleks og gruppen har derfor valgt ikke at redegøre for formlen i dette teoriafsnit.

Udover konverteringen fra længde- og breddegrads systemet til UTM, skal en konvertering fra UTM til pixels også bruges i projektet. Dette sker ved brugen af en world file. World filen bruges til at oversætte UTM koordinater til pixels på kortet, så det GPS data der loades ind, kan findes på kortet. Dette projekts world file ser således ud:

```
mpx 0.84964441
roty 0.00000000
rotx 0.00000000
mpy -0.84964441
x0 539276.35483168
y0 6250863.73506770
```

mpx - Antal meter pr pixel i x-retning

roty - Rotationsværdi om y-aksen

rotx - Rotationsværdi om x-aksen

mpy - Antal meter pr pixel i y-retning (oftest negativ)

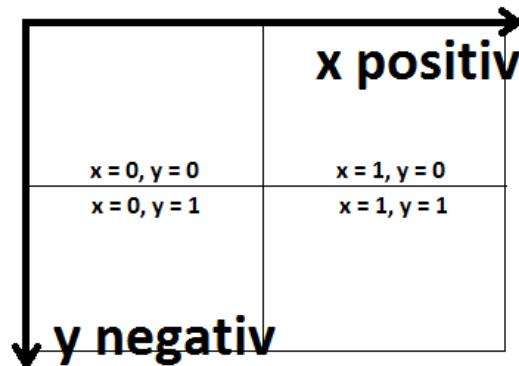
x0 - X koordinatet for midten af pixlet i øverste venstre hjørne

y0 - Y koordinatet for midten af pixlet i øverste venstre hjørne

Da o-løbskort aldrig er roteret, vil roty og rotx altid være 0, og der vil kigges bort fra disse værdier.

Grunden til at mpy oftest er negativ, er fordi pixels begynder fra øverste venstre hjørne, hvor et UTM koordinatsystemet begynder fra nederste venstre hjørne. Tages der udgangspunkt i figur

5.2 kan det forklares således:



Figur 5.2. Figur til forklaring af UTM til pixel

Koordinatsættet (0,0) vil svare til (x0,y0). Koordinatsættet (1,0) vil svare til (mpx + x0,y0)
 Koordinatsættet (0,1) vil svare til (x0,mpy + y0) Koordinatsættet (1,1) vil svare til (mpx + x0, mpy + y0)

For at finde pixel-position (x' , y') ud fra UTM koordinat (x,y) bruges følgende formel:

$$x' = (x - x_0) / \text{mpx}$$

$$y' = (y - y_0) / \text{mpy}$$

5.4 Opsummering

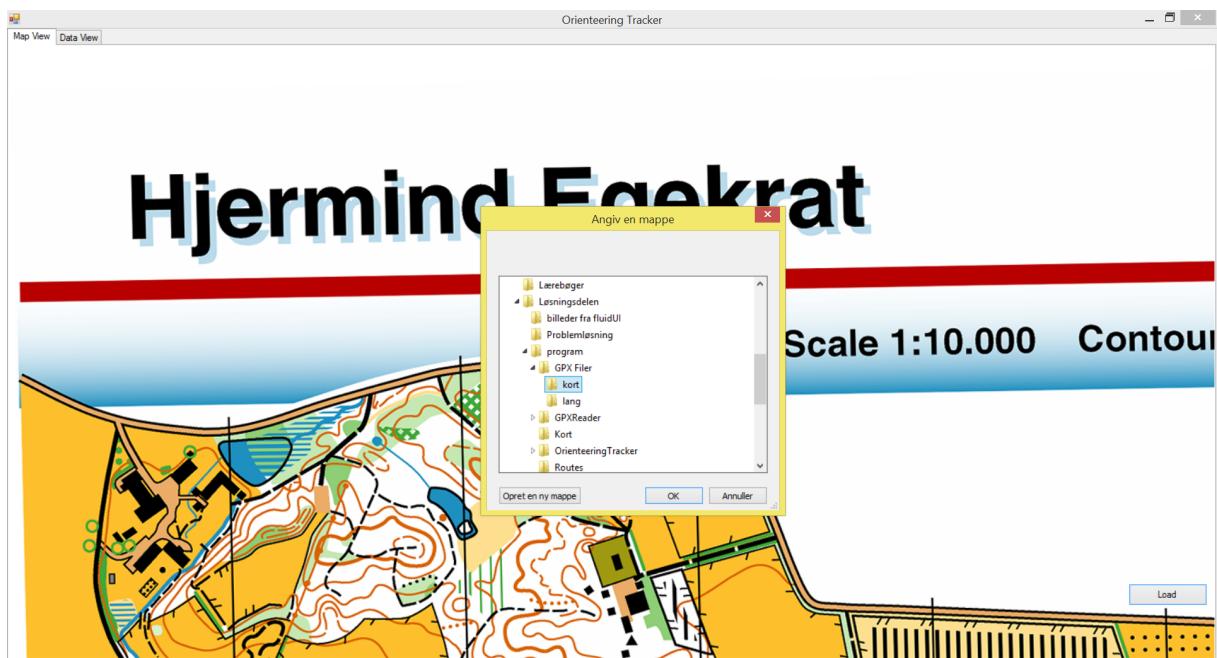
Længde- og breddegrader og Universal Transverse Mercator systemet er blevet kort forklaret. Gruppen har valgt ikke at forklare den brugte formel i konverteringen fra længde- og breddegrader til UTM koordinater. Det lykkedes at opstille formlen for konverteringen fra UTM koordinater til pixels. Denne teori vil blive brugt i kapitlet "Implementeringen".

Implementering 6

Dette kapitel omhandler det fremstillede programs brugergrænseflade, et afsnit om programstruktur med tilhørende klassebeskrivelse, samt implementeringen af kildekoden.

6.1 Brugergrænseflade

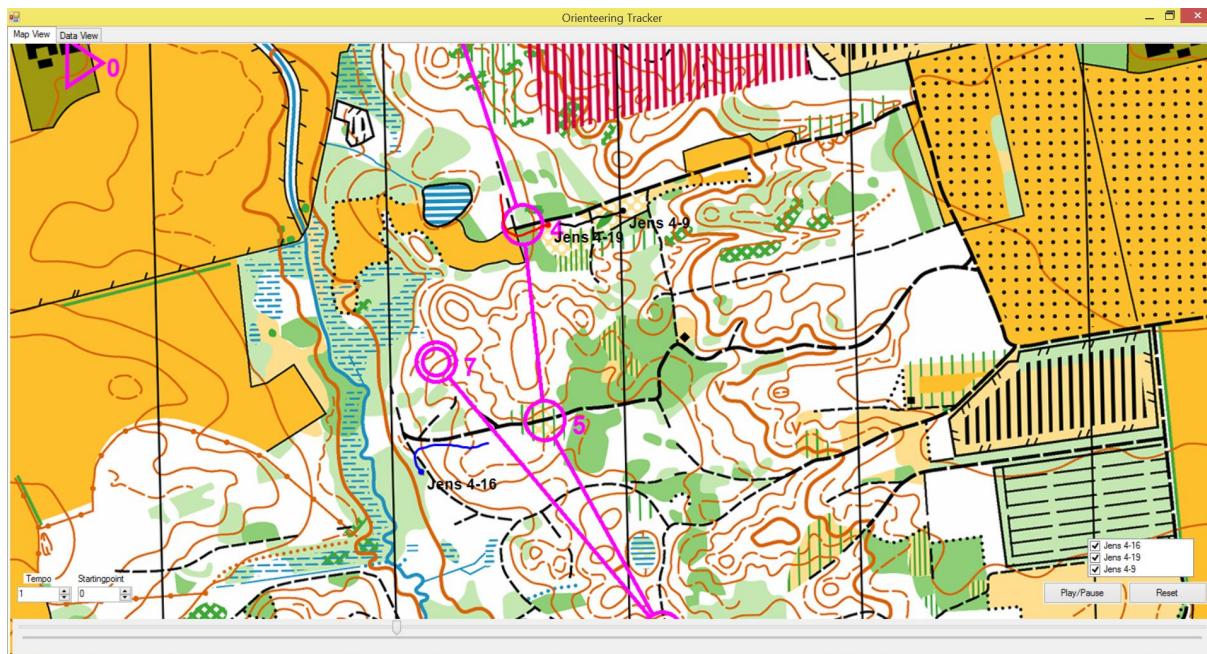
Ved programmets opstart vises det statiske kort som gruppen har valgt at benytte under udviklingen af dette program, samt en load knap. Dette er at finde i tabben "Map View". Tabben "Data View" vil ikke indeholde noget information ved opstart af programmet. Efter tryk på load knappen, popper en dialogbox op, hvorefter brugeren skal vælge den mappe som indeholder løbernes GPX-filer, samt koordinater for posterne til den bane løberne skal løbe. Dette kan ses på figur 6.1.



Figur 6.1. Billede af folder-browser-dialogboxen

Når GPX-filer og koordinaterne for posterne er loadet ind i programmet, vil de forskellige mediaplayer funktioner komme frem, som ses på figur 6.2.

I bunden af programmet ses programmets trackbar, der gør det muligt at spole frem og tilbage under afspilningen. Nede i venstre hjørne findes to forskellige mediaplayer funktioner. Den yderste til venstre bruges til at vælge tempoet for afspilningen. Den anden bruges til at vælge startpost for løberne, hvilket gør det muligt at samle løberne, for lettere at kunne sammenligne dem med hinanden. På højre side ses en play/pause og reset knap. Play/pause bruges til at starte/pause



Figur 6.2. Billede af Map View, med løbere og punkter løbet ind

afspilningen. Reset knappen bringer programmet tilbage til dets opstarts stadie. Lige over disse to knapper findes en checkbox. I denne checkbox kan alle løberne ses. Checkboxen afgør hvilke løberne der vises på kortet, hvilket gør det muligt at skjule løberne, hvis der vil fokuseres på nogle bestemte løbere. Løberne vises ude på kortet med en prik i hver deres farve, samt en hale efter denne, som viser hvordan de har løbet de sidste 30 sekunder.

Når der trykkes på tabben "Data View" vil alle løbernes data for løbet vises. I "Data View" kan der ses løbernes slut position, navn, tid, tidsdifference til førsteladsen, tilbagelagt distance, hastighed i minut pr. kilometer og tiden for hvert stræk. Der kan ses et eksempel på dette på figur 6.3 herunder.

0 - 7												
Position	Name	Time	Difference	Distance	Speed	0 - 1	1 - 2	2 - 3	3 - 4	4 - 5	5 - 6	6 - 7
1	Jens 4-9	00:22:59	00:00:00	3301.60400589844	6,96126285656085	00:04:59	00:01:31	00:02:18	00:04:19	00:03:00	00:04:02	00:02:50
2	Jens 4-19	00:23:55	00:00:56	3362.46961872819	7,11282758763269	00:05:13	00:01:20	00:02:14	00:04:56	00:02:59	00:04:13	00:03:00
3	Jens 4-16	00:36:00	00:13:01	5972.38789609404	6,02773976277464	00:05:49	00:01:15	00:01:37	00:01:27	00:02:48	00:03:14	00:19:50

Figur 6.3. Billede af Data View for hele løbet

Klikkes der på et felt for et stræk, kan der ses mere detaljerede informationer om det specifikke stræk. Som det kan ses på figur 6.4, indeholder dette de samme informationer som når der ses på hele løbet, med en enkelt undtagelse, her kan løberne se hvilken position de fik på det specifikke stræk.

1 - 2						
Position	Name	Time	Difference	Distance	Speed	Final Position
1	Jens 4-16	00:01:15	00:00:00	238.060980080101	5,25075549793759	3
2	Jens 4-19	00:01:20	00:00:05	236.073639690861	5,64795516805407	2
3	Jens 4-9	00:01:31	00:00:16	247.751629919985	6,1217225529452	1

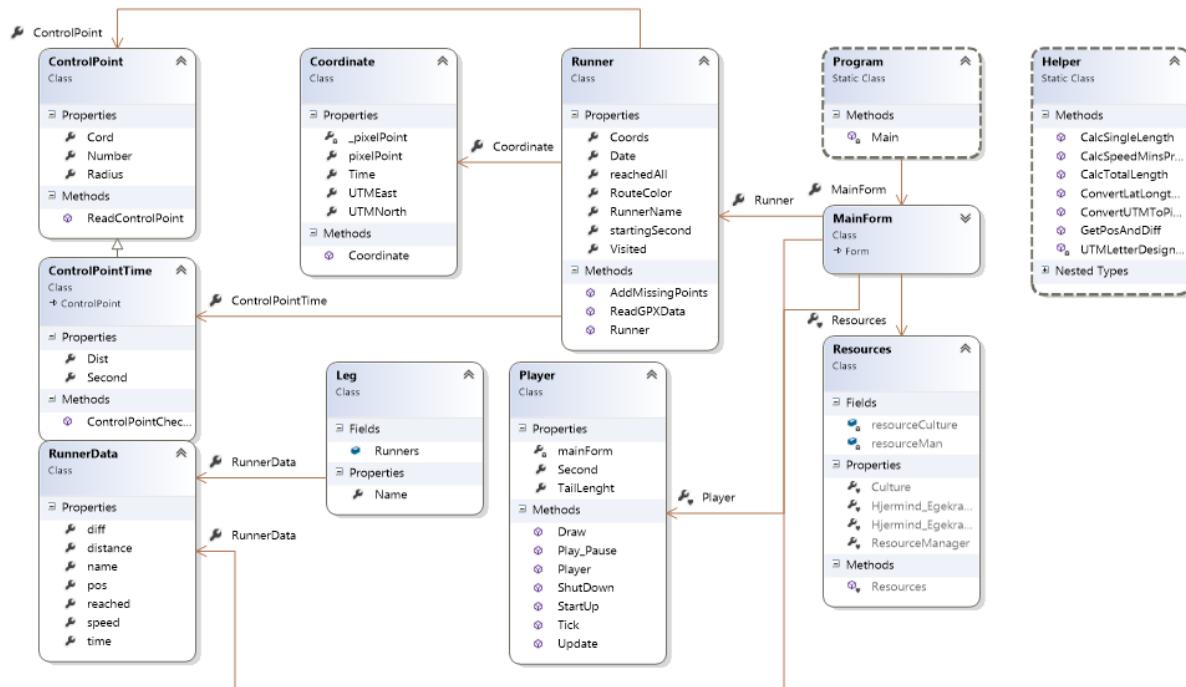
Figur 6.4. Billede af Data View for et specifikt stræk

6.2 Programstruktur

Da ingen af medlemmerne i gruppen tidligere havde erfaring med udvikling af store applikationer i C# og Windows Forms, blev meget af udviklingen lavet ud fra “trial and error” principippet. Med det menes at gruppen forsøgte sig meget frem, for at lære at bruge Windows Forms, og der ikke blev lavet en decideret plan eller design for programmet, før udviklingen blev påbegyndt. Dette medførte at store dele af programmet blev lavet i samme klasse, hvilket forårsagede et forholdsvis lavt abstraktionsniveau. Senere i projektforløbet, blev der opsat flere klasser og metoderne blev delt passende ud i disse.

6.2.1 Klassebeskrivelse

Gruppen har vha. Visual Studio udarbejdet et klassediagram, for at give et bedre overblik over programmets klasser. I firkanterne i klassediagrammet ses klassens navn, dens fields, properties og metoder.



Figur 6.5. Klassediagram over projektets program

- **Helper** klassen indeholder de metoder der bruges til at lave udregninger i programmet, eksempelvis konverteringen fra koordinater i længde- breddegrader til UTM koordinater.
- **Coordinate** den bliver brugt til at holde koordinatsæt til Runners og ControlPoints.
- **ControlPoint** repræsentere posterne i løbet. Den indeholder en posts radius, når løberen kommer indenfor denne radius af posten, vil posten blive markeret som besøgt. Derudover indeholder klassen nummer og henter dens koordinater fra Coordinate.
- **ControlPointTime** har nedarvet fra ControlPoint, den implementerer Second og Dist som vil være tid i sekunder og distancen fra løberen til posten.
- **Runner** indeholder informationer om en løber, dette er bl.a. løberens koordinater på ruten, om løberen har besøgt alle poster og løberens navn.

- **RunnerData** gemmer på en mængde data om hver enkelt løber, som hastighed, distance løbet og position i løbet, dette vil være de informationer som kan findes under tabben som gruppen har kaldt ”Data view”.
- **Leg** er det engelske ord for ”stræk”. Leg indeholder en liste af RunnerData samt et navn på strækket.
- **Player** sørger for at tegne løberen på kortet, samt den hale som skal være efter løberen. Derudover viser, skjuler og opdatere den forskellige funktioner i programmet når det køres. Dette bruges i den tab som gruppen har kaldt ”Map view”.
- **MainForm** holder sammen på programmet. Det er her GUI'en bliver lavet og events bliver håndteret.
- **Resources** indeholder kortet som bruges I programmet, samt world-filen.

Klasserne er kort beskrevet for at give et overblik over dem, og de vil blive beskrevet mere dybdegående i næste afsnit om ‘kildekoden’.

6.3 Kildekoden

I dette afsnit vil kildekoden til programmet blive beskrevet. Der vil være mere fokus på de større metoder i klasserne, frem for de mindre metoder.

6.3.1 Helper

Helper er en statisk klasse, der er lavet for at have nogle funktioner tilgængelige overalt i programmet, uden at skulle instantiere et nyt objekt der kunne udføre handlingerne, eller copy-paste funktionerne ind i de klasser der skal bruge dem.

Helper indeholder konverteringsfunktionerne, i henholdsvis ConvertLatLongToUTM, og ConvertUTMToPixel. ConvertLatLongToUTM funktionen, er lånt fra brugeren Hactic fra Worldwindcentral’s forum [Hactic, 2007]. Derudover bliver Helper klassen også brugt til at lave beregninger på løbernes ture, altså distance og hastighedsberegninger.

GetPosAndDiff, er en metode der bruges til at sortere en liste af RunnerData efter hvem der er hurtigst. Først bruges .OrderBy på runnerdata, for at sortere efter tiden. Derefter itereres der over dataen, for at kunne sætte en position ind på de respektive RunnerData. Der testes efter om reached propertien er sat til True. Hvis ja, sættes positionen og differencen til førstepladsen, og hvis nej, sættes positionen til -1, og tidsdifferencen sættes til 0.

```

1  /* Sortere en liste af RunnerData efter tid, og saetter positionen og differencen til foerstepladsen */
2  public static List<RunnerData> GetPosAndDiff(List<RunnerData> runnerData)
3  {
4      int pos = 1;
5      runnerData = runnerData.OrderBy(runner => runner.time).ToList();
6      for (int i = 0; i < runnerData.Count; i++)
7      {
8          if (runnerData[i].reached)
9          {
10              runnerData[i].pos = pos;
11              if (runnerData[i].pos != 1)
12              {
13                  runnerData[i].diff = runnerData[i-1].diff + runnerData[i].time - runnerData[i-1].time;
14              }
15              pos++;
16          }
17      }
18 }
```

```

19     {
20         runnerData[i].pos = -1;
21         runnerData[i].diff = new TimeSpan(0);
22     }
23     return runnerData;
}

```

ConvertUTMToPixel metoden konvertere UTM koordinatsæt, til en bestemt pixel på kortet. Metoden benytter formlerne beskrevet i teoriafsnittet, og en world-fil gemt i Resources.

```

/* Laver omregningen fra UTM til Pixel koordinater */
2 public static void ConvertUTMToPixel(double UTM_north, double UTM_east, out float x, out float y)
{
4     /* Indlaeser kort fra Resources */
5     MemoryStream ms = new MemoryStream(OrienteeringTracker.Properties.Resources.Hjermind_Egekrat_ref_ref1);
6     string line;
7     List<float> worldTal = new List<float>();
8     StreamReader sr = new StreamReader(ms, Encoding.UTF8);

10    /* Iterere gennem de enkelte linjer i filen, og tilfoejer til listen af floats, worldTal */
11    while ((line = sr.ReadLine()) != null)
12    {
13        worldTal.Add(float.Parse(line, CultureInfo.InvariantCulture));
14    }
15    /* Beregner outputparametrene x og y */
16    x = Convert.ToInt32((UTM_east - worldTal[4]) / worldTal[0]);
17    y = Convert.ToInt32((UTM_north - worldTal[5]) / worldTal[3]);
18}

```

6.3.2 Coordinate

Coordinate klassen indeholder både UTM koordinatsættet og pixel koordinatsættet. Derudover bliver en tid også gemt i Coordinate.

6.3.3 ControlPoint

Til at håndtere poster, er der lavet en klasse kaldet ControlPoint. Denne klasse indeholder et koordinat, Coord, for hvor posten befinner sig, og et nummer, Number, der indikere hvilket nummer i rækken, posten skal besøges. Derudover indeholder ControlPoint en funktion ReadControlPoint, der tager to parametre, line, en tekststreng der indeholder koordinaterne, og nr, denne posts nummer i rækkefølgen.

```

/* Indlaeser data om instansen af ControlPoint */
2 public void ReadControlPoint(string line, int nr)
{
4     string[] coordinatesString = line.Split(';');
5     this.Cord = new Coordinate(float.Parse(coordinatesString[2],
6         System.Globalization.CultureInfo.InvariantCulture),
7         float.Parse(coordinatesString[3], System.Globalization.CultureInfo.InvariantCulture), DateTime.Now,
8         float.Parse(coordinatesString[0], System.Globalization.CultureInfo.InvariantCulture),
9         float.Parse(coordinatesString[1], System.Globalization.CultureInfo.InvariantCulture));
10    this.Number = nr;
}

```

6.3.4 ControlPointTime

ControlPointTime er en klasse der håndterer løbernes interaktion med posterne. ControlPointTime nedarver fra ControlPoint, og indeholder derfor de samme properties og metoder som ControlPoint. Derudover er der også tilføjet et sekund og en distance, der indikere hvornår løberen har nået den givne post, og hvor tæt vedkommende var på den.

Metoden ControlPointChecker, tager et ControlPoint og en Runner, og tjekker om den pågældende løber rent faktisk rammer posten. Hvis den gør, så udfylder den data'en i instansen af ControlPointTime. Måden den gør det på, er ved at iterere igennem alle koordinaterne i løberens rute, indtil et af koordinaterne ligger inden for den givne radius af posten. Derefter tager funktionen alle de efterfølgende koordinater og ligger i en liste, indtil den igen rammer en der ikke er indenfor den givne radius af posten. Det koordinat der ligger tættest posten på fra den liste, bliver gemt som dataen i den nuværende ControlPointTime instans.

```

/* Tester om en runner har ramt det paagaeldende ControlPoint */
2 public void ControlPointChecker(ControlPoint cp, Runner r)
{
4     List<ControlPointTime> distList = new List<ControlPointTime>();
5     ControlPointTime cpt = new ControlPointTime();
6     double doubleDist = 0;
7     foreach (Coordinate coord in r.Coods)
8     {
9         doubleDist = Helper.CalcSingleLength(coord.pixelPoint.X, coord.pixelPoint.Y, cp.Cord.pixelPoint.X,
10            cp.Cord.pixelPoint.Y);
11         if (doubleDist < 25)
12         {
13             for (int i = r.Coods.IndexOf(coord); i < r.Coods.Count; i++)
14             {
15                 if (Helper.CalcSingleLength(r.Coods[i].pixelPoint.X, r.Coods[i].pixelPoint.Y,
16                   cp.Cord.pixelPoint.X, cp.Cord.pixelPoint.Y) > 25)
17                 {
18                     ControlPointTime thisCpt = distList.OrderBy(distance => distance.Dist).First();
19                     this.Cord = thisCpt.Cord;
20                     this.Dist = thisCpt.Dist;
21                     this.Number = thisCpt.Number;
22                     this.Second = thisCpt.Second;
23                     return;
24                 }
25                 cpt = new ControlPointTime();
26                 cpt.Cord = cp.Cord;
27                 cpt.Number = cp.Number;
28                 cpt.Second = r.Coods.IndexOf(coord);
29                 cpt.Dist = Helper.CalcSingleLength(r.Coods[i].pixelPoint.X, r.Coods[i].pixelPoint.Y,
30                   cp.Cord.pixelPoint.X, cp.Cord.pixelPoint.Y);
31                 cpt.Second = i;
32                 distList.Add(cpt);
33             }
34         }
35     }
36     this.Cord = null;
37     this.Dist = 0;
38     this.Number = 0;
39     this.Second = 0;
40     return;
41 }
```

6.3.5 Runner

Runner er en klasse til at repræsentere en løbers tur, der opbevare data om den rute løberen har løbet, tiden, distancen med videre. Den har følgende properties:

```

1 public string RunnerName { get; set; }
2 public DateTime Date { get; set; }
3     public System.Drawing.Color RouteColor { get; set; }
4     public bool reachedAll { get; set; }
5     public List<int> startingSecond { get; set; }
6     public List<Coordinate> Coords { get; set; }
7     public List<ControlPointTime> Visited { get; set; }

```

Til at gemme ruten, er der på hver Runner lavet en liste af koordinater, kaldet Coords. Disse koordinater er de punkter løberen har været på i løbet af sin tur, og bliver brugt både til at vise turen på et kort, og til at se hvor tæt løberen har været på posterne, for at se om vedkommende har nået disse punkter. Derudover indeholder Runner klassen også en liste af ControlPointTimes. Denne liste hedder Visited, og indeholder data om hvorvidt løberen har nået en post. Hvis løberen har nået den pågældende post, ligger dataen i listen under det nummer som posten har. Hvis ikke, så indeholder ControlPointTime kun tomt data.

Runner klassen indeholder også funktionalitet til at læse data fra en .gpx fil, altså en fil der indeholder data fra en gps-modtager, som fx en mobil der har været med på løbeturen. Dette bliver gjort i følgende klassemethode, ReadGPXData:

```

1 /* Bruger input GPX data til at indsætte data i instansen af Runner */
2 public void ReadGPXData(FileStream GpxStream)
3 {
4     GpxReader reader = new GpxReader(GpxStream);
5     reader.Read();
6     this.Date = reader.Track.Segments[0].TrackPoints[0].Time;
7     this.RunnerName = Path.GetFileNameWithoutExtension(GpxStream.Name);
8
9     /* Kører gennem hvert koordinat i GPX-filen */
10    foreach (GpxPoint gp in reader.Track.Segments[0].TrackPoints)
11    {
12        double UTMNorthing;
13        double UTMEasting;
14        string Zone;
15        Helper.ConvertLatLongtoUTM(gp.Latitude, gp.Longitude, out UTMNorthing, out UTMEasting, out Zone);
16
17        float x;
18        float y;
19        Helper.ConvertUTMToPixel(UTMNorthing, UTMEasting, out x, out y);
20
21        Coordinate c = new Coordinate(x, y, gp.Time, (float)(UTMEasting), (float)(UTMNorthing));
22        this.Coords.Add(c);
23    }
24    GpxStream.Close();
25    this.AddMissingPoints();
26}

```

Denne metode bruger en GpxReader klasse, hentet fra nettet[DLG.Krakow.pl, 2011]. Readeren har en funktion der hedder Read, som ud fra en FileStream, altså en fil, læser de nødvendige data, og gemmer det i sin Track property. Denne Track property indeholder alle koordinaterne i en liste, men gemmer dem i længde- og breddegrader. Derfor bruges en hjælpefunktion fra en Helper-klassen, til at konvertere til UTM koordinater, og igen til pixels. Efter konverteringen gemmes koordinaterne i listen af koordinater, Coords, der blev beskrevet ovenfor. Efter koordinaterner er læst ind fra filen, kaldes metoden AddMissingPoints. Problemet med de GPX-filer der er blevet brugt i projektet, er at der ikke nødvendigvis er et koordinat for hvert sekund, men at der godt kan være sprunget et sekund eller flere over, hvis GPS-modtagelsen har været dårlig.

AddMissingPoints udfylder de sekunder der ikke er noget data, så det fremstår som at løberen ikke bevæger sig. På den måde er det lettere at finde ud af hvor lang tid løberen har løbet, da starttidspunktet er kendt.

6.3.6 RunnerData

RunnerData indeholder en række informationer, som bruges i "Data View"-tabben.

6.3.7 Leg

Leg klassen indeholder en liste af RunnerData, samt et navn på det specifikke Leg.

6.3.8 Player

Player klassen styrer den grafiske afspilning af løbernes ruter. Player indeholder en reference til MainForm, så den kan bruge nogle af dens data.

Player klassens vigtigste property er integeren "Second". Denne property indeholder informationer om hvilket sekund programmet er kommet til i afspilningen, i forhold til startpunktet.

Draw metoden sørger for at tegne de forskellige løberes position og hale ind på kortet. Draw metoden itererer igennem alle Runner objekter i listen Runners fra MainForm, dvs. alle løbere. Først tjekkes at den pågældende Runner ikke er tjekket fra i checkboxen i Map View tabben, hvis dette er tilfældet hopper programmet videre til næste løber, og dermed vil denne løber ikke blive tegnet på kortet.

Hvis løberen ikke er tjekket fra i checkboxen, tjekkes der efter om programmet er nået enden på gps data for løberen, hvis enden er nået, bliver denne løber ikke tegnet med på kortet.

```

/* Tegner loebernes ruter paa kortet */
2 public void Draw(Graphics g)
{
4     SolidBrush brush;
5     Pen pen;
6     int tempTailLength = TailLength;
7     g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
8
9     /* Iterere gennem alle loeberne */
10    foreach (Runner runner in mainForm.Runners)
11    {
12        if (!mainForm.RunnersCheckBox.CheckedItems.Contains(runner.RunnerName) || Second < 2
13            || Second >= runner.Coords.Count() -
14            runner.Visited[(int)mainForm.StartpointUpDown.Value].Second)
15        {
16            continue;
17        }
18
19        brush = new SolidBrush(runner.RouteColor);
20        pen = new Pen(brush, 3);
21
22        if (Second < TailLength)
23        {
24            tempTailLength = Second;
25        }
26
27        PointF[] RunnerToDraw = new PointF[tempTailLength];
28        for (int Index = 0; Index < tempTailLength; Index++)
29        {
30            RunnerToDraw[Index] = runner.Coords[Second - tempTailLength + Index +
31                runner.Visited[(int)mainForm.StartpointUpDown.Value].Second].pixelPoint;
32            RunnerToDraw[Index].X *= mainForm.ZoomFactor;

```

```

32     RunnerToDraw[Index].Y *= mainForm.ZoomFactor;
33 }
34
35     g.DrawLines(pen, RunnerToDraw);
36     g.FillEllipse(brush, RunnerToDraw[RunnerToDraw.Length - 1].X - 4,
37                   RunnerToDraw[RunnerToDraw.Length - 1].Y - 4, 8, 8);
38     g.DrawString(runner.RunnerName, new Font("Arial", 14, FontStyle.Bold), new SolidBrush(Color.Black),
39                   RunnerToDraw[RunnerToDraw.Length - 1].X + 5, RunnerToDraw[RunnerToDraw.Length - 1].Y + 5);
40 }

```

Hvis programmet ikke er nået enden på data for løberen, laves der et nyt PointF array (PointF er et floating-point koordinatsæt), kaldet RunnerToDraw. Dette skyldes at metoden DrawLine fra Graphics klassen, tager et array af punkter ind og tegner en streg gennem alle disse punkter. Herefter indsættes de punkter der skal tegnes, på nuværende tidspunkt, ud fra Second propertien og startpunkt. Dette gøres i en for-løkke som iterere lige så mange gange som halen skal være lang, hvilket i dette program vil være 30. I for-løkken findes det ønskede punkt ved at indeksere Runner.Coords listen, udfra følgende formlen:

“Index i Coord listen” = “Sekunder fra startpunkt” - “halelængde” + “index fra for-lykken” + “sekundet løberen passerede valgte startpunkt”

Grunden til at “sekundet løberen passerede valgte startpunkt” lægges til, skyldes at Second kun beskriver hvor lang tid der er gået siden valgte Startpunkt.

Dernæst bliver RunnerToDraw's X og Y koordinat multipliceret med ZoomFactor og bliver gemt som den nye X og Y værdi. Dette gøres så de indtegnede løbere følger med kortet, når der zoomes.

Metoden Tick kaldes hver gang programmets timer ticker. Tick tjekker, om der er data fra Runners tilknyttet til det Second, programmet er nået til. Hvis ikke, stoppes timeren og TrackBarens indikator sættes lig det Second programmet er nået til, så den rykker sig med tiden kronologisk. Til sidst inkrementeres Second med 1.

```

1 /* Taeller sekunder op hvis der er mere data. Refreshes mappen */
2 public void Tick()
3 {
4     if (Second >= mainForm.Runners.Max(r => r.Coods.Count -
5         r.Visited[(int)mainForm.StartpointUpDown.Value].Second))
6     {
7         mainForm.PlayTimer.Stop();
8     }
9     mainForm.PlayBar.Value = Second;
10    mainForm.Map1.Refresh();
11    Second++;
}

```

Derudover findes metoder som Play/Pause, Update, StartUp og ShutDown. Play/Pause bruges til at starte og stoppe afspilningen. Update opdatere hvor stor TrackBaren er, så den ikke kan trækkes længere end der er data til. StartUp viser alle mediaplayer funktionerne, hvor ShutDown skjuler alle mediaplayer funktionerne.

6.3.9 MainForm

MainForm er hoved klassen i programmet og indeholder store dele af koden. Den håndtere UI'en og en del logik i programmet.

Eventhandlerne Map1_MouseMove og Map1_MouseDown sørger for at kortet rykker sig, i takt med at brugeren trækker i kortet med venstre musetast holdt nede.

Map1_MouseWheel håndtere programmets zoomfunktion, ved at ændre på ZoomFactor, når brugeren scroller med musehjulet.

Map1_MouseEnter sørger for at kortet har fokus, så snart musen er over kortet.

LoadButton_Click kaldes når "Load" knappen klikkes på, og viser en folder-browser-dialogbox til valg af mappe som indeholder GPX-data for løberne og GPS-data for ControlPoints. Efter valg af mappe, loades ControlPoints og GPX-data ind i systemet. Herefter skjules "Load" knappen, og mediaplayeren vises ved at kalde Player.StartUp().

PlayButton_Click og PlayTimer_Tick kalder henholdsvis Player.Play_Pause() og Player.Tick().

Map1_Paint sørger for, at kortets højde og bredde bliver skaleret efter ZoomFactor, så kortet tegnes i skaleret forhold, og kalder Player.Draw() og sender kortets grafik med.

PlayBar_Scroll sørger for, at når brugeren trækker i TrackBar indikatoren følger Second med, på den måde er TrackBar og Second synkroniseret.

TempoUpDown_ValueChanged ændre på intervallet mellem ticks for timeren, i forhold til værdien brugeren sætter det til.

ResetButton_Click sætter programmet tilbage til opstarts stadiet, når der trykkes på knappen, under kørslen af programmet.

StartpointUpDown_ValueChanged bestemmer startpunkt for afspilingen og sætter Second propertien til 0 og kører Player.Update som opdatere TrackBarens størrelse.

For at styre tabeloversigten over løbernes tid, position, hastighed osv., er der brugt en række funktioner. Den første hedder Setup_Table, der ganske enkelt sætter overskrifterne i tabellen således:

```
DataTable.Columns[I].HeaderText = string;
```

De første 6 ændre sig aldrig, og bliver derfor hardcoded. Efter det, er resten af overskrifterne afhængige af om der bliver set på et delstræk, eller hele turen. Til dette, er defineret en global variabel isLeg, der hele tiden holder styr på hvilket mode tabellen er i. Hvis det er et delstræk, bliver den sidste overskrift sat til "Final position", altså løberens endelige position i løbet. Hvis det derimod er hele løbet, sættes alle delstrækkenes navne til overskrifterne, fx "1-2" eller "2-3" osv.

Derefter bruges funktionen Put_Data, der tager et Leg som parameter. Det er vigtigt at bemærke at hele strækket fra start til slut, altså hele løbet, også er gemt som et Leg. Det er derfor variablen isLeg er nødvendig for at kunne skelne dem fra hinanden. Herefter bliver informationen for hver løber gemt i Leg, skrevet i tabellen. Der bliver tjekket om løberne har nået posterne, og tilføjet til tabellen på baggrund af den information. Derudover er der, som tidligere nævnt, forskel på om det kigges på et delstræk eller den samlede rute. I tilfældet hvor det er hele strækket der vises, bliver der i de første seks søger, udskrevet den samlede information for løberne. I de resterende søger, bliver udskrevet løbernes (rækker) tider i forhold til de pågældende delstræk (søjler).

```
/* Tager et leg som input parameter, og skriver data i tabelen */
2 private void Put_Data(Leg leg)
```

```

4     DataTitle.Text = leg.Name;
5     Setup_Table();
6     DataTable.Rows.Clear();
7     List<string> row;
8     string time = "";
9     string pos = "";
10    foreach(RunnerData rd in leg.Runners)
11    {
12        if (rd.time <= new TimeSpan(0))
13        {
14            time = "X";
15        }
16        else
17        {
18            time = rd.time.ToString();
19        }
20        if (rd.pos == -1)
21            pos = "X";
22        else
23            pos = rd.pos.ToString();
24
25        /* Hvis det er et stræk: */
26        if (isLeg)
27        {
28            row = new List<string> { pos, rd.name, time, rd.diff.ToString(), Math.Round(rd.distance,
29).ToString(), Math.Round(rd.speed, 2).ToString() };
30            foreach (RunnerData mainRd in MainLeg.Runners)
31            {
32                if (mainRd.name == rd.name)
33                {
34                    if (mainRd.pos == -1)
35                        row.Add("X");
36                    else
37                        row.Add(mainRd.pos.ToString());
38                }
39            }
40        /* Ellers er det hele loebet */
41        else
42        {
43            row = new List<string> { pos, rd.name, time, rd.diff.ToString(), Math.Round(rd.distance,
44).ToString(), Math.Round(rd.speed, 2).ToString() };
45            foreach (Leg l in Legs)
46            {
47                for (int runnerIndex = 0; runnerIndex < l.Runners.Count; runnerIndex++)
48                {
49                    if (rd.name == l.Runners[runnerIndex].name)
50                    {
51                        if (l.Runners[runnerIndex].time <= new TimeSpan(0))
52                            time = "X";
53                        else
54                            time = l.Runners[runnerIndex].time.ToString();
55                        row.Add(time);
56                    }
57                }
58            }
59        }
60        DataTable.Rows.Add(row.ToArray<string>());
61    }
62    /* Sortere efter position */
63    DataTable.Sort(DataTable.Columns[0], ListSortDirection.Ascending);
64 }

```

LoadRunners er en funktion der tager en række GPX filer ind som parametre, og opretter en

Runner og RunnerData instans for hver fil. RunnerData der oprettes her, hører til i MainLeg, altså hele ruten og ikke bare et delstræk. Et foreach loop kører igennem hver fil i GPXFiles (inputparametren). En ny Runner oprettes, og ReadGPXData kaldes på instansen, for at fylde data i den pågældende Runner. Efter Runneren er oprettet, kører endnu et foreach loop igennem alle ControlPoints, og opretter nye ControlPointTimes, ved at bruge funktionen ControlPointChecker. ControlPointTimes bliver tilføjet til Visited listen på Runner instansen. En variabel, reached, holder styr på om løberen er nået igennem alle poster. En ny instans af RunnerData oprettes derefter, og hvis alle posterne er nået, altså reached siger sandt, udfyldes RunnerData med løberens data for sin tur. Efter dette er sket for alle GPX filer, kaldes GetPosAndDiff metoden på Helper klassen, for at bestemme løbernes position, og differencen mellem dem.

```

1  /* Laver nye instanser af Runners og RunnerData, og kalder passende funktioner der laeser data ind */
2  private void LoadRunners(string[] GPXFiles)
3  {
4      int ColorCount = 0;
5      Runner runner;
6      MainLeg.Name = string.Format("0 - {0}", ControlPoints.Count - 1);
7      ControlPointTime cpt = new ControlPointTime();
8      bool reached;
9
10     foreach (string file in GPXFiles)
11     {
12         runner = new Runner();
13         runner.ReadGPXData(new FileStream(file, FileMode.Open));
14         runner.reachedAll = true;
15         runner.RouteColor = Colors[ColorCount % 6];
16         ColorCount++;
17         reached = true;
18
19         /* Tester om Runner har naaet posterne */
20         foreach (ControlPoint cp in ControlPoints)
21         {
22             cpt = new ControlPointTime();
23             cpt.ControlPointChecker(cp, runner);
24             runner.Visited.Add(cpt);
25             if (cpt.Cord == null)
26             {
27                 reached = false;
28             }
29         }
30
31         RunnerData runnerdata = new RunnerData();
32         runnerdata.name = runner.RunnerName;
33         if (reached)
34         {
35             runnerdata.distance = Helper.CalcTotalLength(runner, runner.Visited[0].Second,
36                 runner.Visited[runner.Visited.Count - 1].Second);
37             runnerdata.time = TimeSpan.FromSeconds(runner.Visited[runner.Visited.Count - 1].Second -
38                 runner.Visited[0].Second);
39             runnerdata.speed = Helper.CalcSpeedMinsPrKm(runnerdata.distance,
40             (int)(runnerdata.time.TotalSeconds));
41         }
42         else
43         {
44             runnerdata.distance = 0;
45             runnerdata.time = new TimeSpan(0);
46             runnerdata.speed = 0;
47         }
48
49         runnerdata.reached = reached;
50         MainLeg.Runners.Add(runnerdata);
51         Runners.Add(runner);
52     }
53 }
```

```

51     }
52     MainLeg.Runners = Helper.GetPosAndDiff(MainLeg.Runners);
53 }
```

LoadControlPoints er en funktion der tager en fil som input, og bruger den til at oprette posterne i programmet. Ved et loop oprettes der en ny ControlPoint for hver linje i filen, og ReadControlPoint kaldes på det ControlPoint, og sender linjen fra tekstfilen med. Når posterne er læst ind, skal de tegnes på kortet. Et loop kører gennem ControlPoints listen, og der bliver tjekket for nummeret på posten. Hvis posten er den første tegnes en trekant, hvis posten er den sidste tegnes en cirkel inde i en cirkel, og ellers tegnes en enkelt cirkel. Et if-statement, tjekker til sidst om ControlPoint er den første i listen, og hvis ikke, så tegnes der en streg til den forrige. Dette gøres ved at beregne vinkel og distancen til den forrige post, og derefter tegne en streg baseret på den information.

```

1 /* Indlaeser ControlPoints fra en fil */
2 private void LoadControlPoints(string RouteFile)
3 {
4     ControlPoint newControlPoint;
5     int cpNr = 0;
6     foreach (var line in File.ReadLines(RouteFile))
7     {
8         newControlPoint = new ControlPoint();
9         newControlPoint.ReadControlPoint(line, cpNr);
10        ControlPoints.Add(newControlPoint);
11        cpNr++;
12    }
13
14    Graphics g = Graphics.FromImage(Map1.Image);
15    Pen p = new Pen(Color.Magenta);
16    p.Width = 5;
17    int i = 0;
18
19    /* Iterere gennem posterne, og tegner dem paa kortet */
20    foreach (ControlPoint cp in ControlPoints)
21    {
22        Point[] points = {new Point(Convert.ToInt32(cp.Cord.pixelPoint.X + 30),
23                             Convert.ToInt32(cp.Cord.pixelPoint.Y)),
24                           new Point(Convert.ToInt32(cp.Cord.pixelPoint.X - 15),
25                             Convert.ToInt32(cp.Cord.pixelPoint.Y-30)),
26                           new Point(Convert.ToInt32(cp.Cord.pixelPoint.X - 15),
27                             Convert.ToInt32(cp.Cord.pixelPoint.Y+30))};
28        if (ControlPoints.First() == cp)
29        {
30            g.DrawPolygon(p, points);
31        }
32        else if (ControlPoints.Last() == cp)
33        {
34            g.DrawEllipse(p, cp.Cord.pixelPoint.X - 17, cp.Cord.pixelPoint.Y - 17, 34, 34);
35            g.DrawEllipse(p, cp.Cord.pixelPoint.X - 25, cp.Cord.pixelPoint.Y - 25, 50, 50);
36        }
37        else
38        {
39            g.DrawEllipse(p, cp.Cord.pixelPoint.X - 25, cp.Cord.pixelPoint.Y - 25, 50, 50);
40        }
41
42        using (Font myFont = new Font("Arial", 24, FontStyle.Bold))
43        {
44            g.DrawString(cp.Number.ToString(), myFont, Brushes.Magenta, new
45            Point(Convert.ToInt32(cp.Cord.pixelPoint.X + 30),
46                  Convert.ToInt32(cp.Cord.pixelPoint.Y - 25 / 2)));
47        }
48    }
49 }
```

```

47     if (ControlPoints.First() != cp)
48     {
49         float xDiff = ControlPoints[i - 1].Cord.pixelPoint.X - cp.Cord.pixelPoint.X;
50         float yDiff = ControlPoints[i - 1].Cord.pixelPoint.Y - cp.Cord.pixelPoint.Y;
51
52         float angle = (float)Math.Atan2(yDiff, xDiff) - (float)(Math.PI);
53
54         float distance = (float)Math.Sqrt(xDiff * xDiff + yDiff * yDiff);
55
56         float newX = (float)(ControlPoints[i - 1].Cord.pixelPoint.X + (Math.Cos(angle) * (distance -
57             25)));
58         float newY = (float)(ControlPoints[i - 1].Cord.pixelPoint.Y + (Math.Sin(angle) * (distance -
59             25)));
60
61         g.DrawLine(p, new Point(Convert.ToInt32(ControlPoints[i - 1].Cord.pixelPoint.X +
62             Math.Cos(angle) * 25),
63             Convert.ToInt32(ControlPoints[i - 1].Cord.pixelPoint.Y + Math.Sin(angle) * 25)),
64             new Point(Convert.ToInt32(newX), Convert.ToInt32(newY)));
65     }
66     i++;
67 }
68 Map1.Refresh();
69 }
```

LoadLegs indsætter data i listen Legs, der indeholder informationer om de enkelte stræk. Dette gøres ved at lave et for-loop for hvert ControlPoint, og inde i dette for-loop laves et foreach-loop for hver Runner i Runner listen. Inde i dette foreach-loop tjekkes, om den enkelte Runner har nået det specifikke ControlPoint. Hvis dette er opfyldt, bliver distance, tid og hastighed udregnet for det specifikke stræk og lægges ind i et nyt objekt af RunnerData klassen. Hvis dette ikke er opfyldt, bliver værdierne sat til 0. Herefter tilføjes RunnerData objektet til et Leg objekt, som indeholder information om alle Runners, for det enkelte stræk. Når alle Runners er blevet loadet ind, udregnes løbernes position i forhold til hinanden, og deres tids difference til den hurtigste. Dette gøres vha. helper klassens GetPosAndDiff metode. Til sidst tilføjes dette stræk til listen Legs, som indeholder alle stræk.

```

1 /* Oprettet de legs der haandtere stræk for loeberne */
2 private void LoadLegs()
3 {
4     for (int i = 1; i < ControlPoints.Count; i++)
5     {
6         Leg leg = new Leg();
7         leg.Name = string.Format("{0} - {1}", i - 1, i);
8         foreach (Runner r in Runners)
9         {
10            RunnerData runnerdata = new RunnerData();
11            if (r.Visited[i].Cord != null)
12            {
13                runnerdata.reached = true;
14                runnerdata.distance = Helper.CalcTotalLength(r, r.Visited[i - 1].Second,
15                    r.Visited[i].Second);
16                runnerdata.time = TimeSpan.FromSeconds(r.Visited[i].Second - r.Visited[i - 1].Second);
17                runnerdata.speed = Helper.CalcSpeedMinsPrKm(runnerdata.distance,
18                    (int)(runnerdata.time.TotalSeconds));
19            }
20            else
21            {
22                runnerdata.reached = false;
23                runnerdata.distance = 0;
24                runnerdata.time = new TimeSpan(0);
25                runnerdata.speed = 0;
26            }
27        }
28    }
29 }
```

```
27     runnerdata.name = r.RunnerName;
28     leg.Runners.Add(runnerdata);
29 }
30 leg.Runners = Helper.GetPosAndDiff(leg.Runners);
31 Legs.Add(leg);
32 }
```

Test 7

Gruppen har foretaget nogle tests i forbindelse med udviklingen af programmet, disse metoder vil blive beskrevet herunder, samt hvilke software og metoder gruppen har anvendt for at teste programmet. De anvendte testtyper begrundes ved at brugeren ikke skal kunne give forkert input eller uønsket kunne manipulere med data.

7.1 Monkey testning

Denne test-type kan kategoriseres på forskellige måder, da der er forskellige typer af monkey testning. Der er Smart monkey test, og Ignorant monkey test (også kaldet Dumb monkey test), hvor begge typer er automatiske tests udført af software. De kategoriseres således: Smart monkey test: Her gøres der brug af unit test, da der laves test-suites, hvor ”aben” kender til input-type og ved hvilke metoder der skal testes. Denne test-type skal oftest købes, da det er mere avanceret software, som integreres med den kode som skal testes.

Smart monkey tests er hurtig til at finde fejl i programmet, hvis det er sat ordentligt op. Fejl fundet ved denne test-type er fra store til små, og kan typisk findes i en log-fil efter testen er kørt.

Ignorant monkey test: Denne test-type er en blackbox test, da den ikke har viden om programmets kode, og derfor tester random inputs på enhver mulig måde. Ignorant monkey tests kan findes som gratis software, bl.a. ”GUI tester”, som anvendes i dette projekt. ”Aben” kræver tæt på ingen opsætning af program for at kunne køre, og laver stress-test af programmet, og kan køre i flere dage, hvis ikke den finder fejl der får programmet til at crashe. Ved fejl i denne test, skal testeren typisk sidde og overvåge testen, da der typisk ikke er beskrivende log-filer over testen efter kørsel. Ignorant monkey test finder typisk store fejl i programmet, hvor de mindre fejl i koden ikke opdages. [Exforsys, 2011]

Gruppen har valgt at anvende denne test-type, for at sikre, at brugeren ikke har mulighed for at intaste noget forkert input.

7.1.1 GUI tester

GUI tester er et simpelt ignorant monkey test software, udviklet af Luigi Poderico, som er anvendt i dette projekt. Programmet køres ved at have det testede program i forgrunden, og derefter klikke på højre CTRL knap. GUI testeren vil herefter klikke på forskellige steder i programmet, og intaste forskellige inputs hvis muligt. Problematikker ved dette software er, at det kan åbne op for andre programmer, samtidig med at test-softwaret køres, og derved prøve at ”teste” de andre programmer. [Poderico, 2007]

Ved brug af GUI tester i dette projekt, blev OrienteeringTracker startet, og GUI testen sat i gang. Der blev lavet følgende tests:

- 1: Test uden at have manipuleret programmet. Der er her ikke loadet noget ind i programmet,

og programmet består derfor kun af et kort, Map View tab, Data View tab og Load-knap.

2: Test ved tryk af Load-knap, så dialogboksen åbnes, og testen derfor kan vælge hvilken som helst mappe.

3: Test efter programmet har loadet filer. Her er både løbere, kort, poster og data tilstede i programmet.

I første testcase åbnes programmet, og GUI test startes. ”Aben” trykker mange steder, og ender med at minimere programmet flere gange før den får trykket på ”Load”-knappen. Der er ikke sket nogle fejl i programmet på dette stadie.

I anden testcase har ”aben” klikket på Load, får valgt en mappe som ikke indeholder de rigtige filer, og programmet kaster en ArgumentNullException. Løsningen på denne fejl er at tjekke om mappen med filerne indeholder filtyper der ender på .gpx og utm.

I tredje testcase er de rigtige filer loadet ind i programmet, og ”aben” får lov til at teste både Map View og Data View. Den skifter selv mellem tabs, og prøver, i Map View, at indsætte forskellige værdier i både Tempo og Startpoint. Disse to værdier bliver derved sat til henholdsvis det mindst mulige (ved for lav værdi) eller det højest mulige (hvis værdien indsats er højere end muligt for programmet). Den slår nogle af løberne fra i checkboksen, og får både prøvet at starte løbet, og pauser løbet. I Data View valgte den en celle, og skrev en værdi i cellen. Dette skulle ikke ske, da en bruger så ville kunne manipulere med det viste. Dette blev ændret ved at gøre cellerne ReadOnly. ”Aben” får både klikket sig ind på et stræk, og gjort brug af tilbage-knappen.

7.2 Blackbox testning

Blackbox testning er en test-type, hvor testeren indtaster input til programmet, og ser på hvorvidt outputtet er korrekt. I dette projekt vil denne test-type bruges til at se, hvorvidt filer der loades ind håndteres korrekt. Her vil testcases indkludere følgende:

Testcase 1: Indlæsning af flere af samme person, for at observere hvorvidt Data View tabbet giver korrekt information, og hvordan flere persons rute afbildes i Map View. Her forventes, at Data View kan sorteres per kolonne, hvor en passende sortering finder sted, og at flere versioner af samme person kan afbildes i Map View uden exceptions.

Testcase 2: Indlæsning af personer, som ikke når alle poster på ruten. Her forventes at deres data bliver sat til 0 eller ”X”.

Testcase 3: Lukke dialogboksen uden valg af mappe, efter tryk på Load-knappen.

I testcase 1, hvor flere af samme person læses ind, bliver Map View vist korrekt, og alle ”personer” af samme persons rute, blev lagt ovenpå hinanden. I Data View blev data vist korrekt, men sortering af position fejlede. De øverste positioner ved sortering blev: 1, 10, 11, 12... 2, 21, 22... Dette skyldes, at sorteringen skete ved sammenligning af strenge, og ikke ved tal. Dette er blevet rettet, således at det kommer i den rigtige rækkefølge.

I testcase 2 blev personen, som ikke nåede alle poster, håndteret som forventet. Tider blev udskiftet med ”X” i de tilfælde personen ikke nåede en post, og difference i tid, samt distance løbet, blev sat til 0. Løberen indgår med korrekte tider i de stræk, som løberen nåede.

I testcase 3 crashed programmet med en ArgumentOutOfRangeException. Denne fejl blev rettet, ved at der ikke bliver loadet noget ind i programmet, før der bliver valgt en mappe og trykkes ”OK” i dialogboksen.

7.3 Unit testning

Unit testning er en metode at teste software på, ved at teste forskellige dele af programmet individuelt. I objektorienteret programmering, som softwaren til dette projekt er lavet efter, vil opdelingen typisk ske på klasseniveau. Fordelene ved at teste de enkelte klasser individuelt, er den sikkerhed det giver, både da funktionaliteten af delene garanteres at virke, men også at den videre udvikling af programmet bliver lettere. Skal koden på et tidspunkt skrives om på grund af problemer med fx ydeevne, er det let at garantere, at programmet ikke pludselig holder op med at virke, så længe den del af programmet stadig kan gennemføre sine tests. Tests bliver på den måde også en form for dokumentation over hvilken funktionalitet en klasse eller funktion i programmet har. Hvis en anden programmør end den oprindelige skaber af koden, skal bruge en funktion, kan vedkommende kigge på de test der er skrevet, for at se hvad det forventede output er, og hvordan den del opfører sig i forhold til de forskellige inputs der måtte være. På den måde kan en anden programmør hurtigt finde ud af hvordan den del af koden bruges, fremfor at skulle bruge lang tid på at tyde koden, og forestille sig alle tænkelige scenarier der måtte være.

Af Unit-tests er blevet lavet 3 tests. Den ene tester CalcSingleLength på Helper klassen. CalcSingleLength, beregne grundlæggende bare distancen mellem to punkter, så derfor er testen udført ved at tjekke om funktionen returnere 5, som er distancen mellem punkterne (-2,1) og (1,5).

```
[TestCase]
2 public void CalcSingleLength_TwoPoints_Return5()
{
4     double dist = Helper.CalcSingleLength(-2, 1, 1, 5);
    Assert.AreEqual(dist, 5);
6 }
```

Derefter er ConvertLatLongToUTM funktionen testet, ved at sammenligne med et andet konverteringsprogram, på nettet. [LatLong.net, 2015]

```
[TestCase]
2 public void ConvertLatLongtoUTM_LatLong_UTM()
{
4     double easting;
5     double northing;
6     string zone;
7     Helper.ConvertLatLongtoUTM(57.121332, 9.953613, out northing, out easting, out zone);
8
9     Assert.AreEqual(Math.Round(easting, 2), Math.Round(557740.21, 2));
10    Assert.AreEqual(Math.Round(northing, 2), Math.Round(6331295.72, 2));
11    Assert.AreEqual(zone, "32V");
12 }
```

ReadControlPoint er blevet testet, ved at kalde funktionen med linjen "539446.2;6249967;200;1056". Der testes derefter, om pixelkoordinatet på den pågældende instans af ControlPoint, er blevet sat til (200,1056).

```
[TestCase]
2 public void ReadControlPoints_LineAndNr_PopulatedCP()
{
4     ControlPoint cp = new ControlPoint();
5     string line = "539446.2;6249967;200;1056";
6     int nr = 1;
7     cp.ReadControlPoint(line, nr);
8 }
```

```
10     PointF point = new PointF(200,1056);
11     Assert.AreEqual(cp.Cord.pixelPoint, point);
12 }
```

Diskussion 8

I dette kapitel vil gruppen diskutere nogle af de større beslutninger der er blevet truffet gennem projektarbejdet, specielt vil gruppen diskutere de fejl, gruppen mener, der er blevet lavet gennem projektet.

8.1 Problemanalyse

Projektets problemanalyse er bygget meget på input fra Jens Børsting og Claus Bobach. Begge har stor erfaring inden for o-løb og indsigt i hvad den enkelte o-løber har af behov. Gruppen mener dog, at det er svært at generalisere ud fra en respondentgruppe på to personer. Gruppen havde også kontaktet andre personer, som fx. topatleter, og de fleste lovede et svar. Desværre skete dette dog aldrig, og gruppen skulle have været mere ihærdig efter at få svar fra dem, samt have skrevet til flere personer.

Vidensdelingen med Jens var til stor gavn, men var dog svær at dokumentere, da dette tit foregik gennem uformelle Skype-samtaler og respons på det tekstuelt gruppen har skrevet, samt programmet. Gruppen burde eventuelt have lavet et formelt interview med Jens, udover de allerede besvarede spørgsmål via en mail korrespondance.

I begyndelsen af projektet, kontaktede gruppen Claus, for at sætte et interview op. Interviewet blev foretaget i Aalborg OK's klubhus, hvor tre gruppemedlemmer var til stede. Gruppen havde forberedt et semistruktureret interview, som gik bedre end forventet. Det at der var tre interviewere til stede, gjorde at der kom flere uddybende spørgsmål i løbet af interviewet. Claus kom med mange gode besvarelser og kommentarer, dog var mange af besvarelserne noget gruppen havde forventet, og gav derfor en begrænset mængde ny viden.

8.2 Program design

Som nævnt i afsnit 6.1 om programstruktur, havde gruppen før dette projekt ikke arbejdet med større programmer i C# eller arbejdet i Windows Form. Dette medførte, at da gruppen påbegyndte programmeringen, var det ikke særlig objektorienteret. Dette resulterede i, at der ikke var en generel struktur i programmet. Store dele af programmet blev lavet i de samme klasser, men langsomt fordelt ud på mindre klasser. I nogle af de senere OOP-kursusgange, blev gruppen præsenteret for Model-View-Controller (MVC), som er et koncept til at strukturere og opbygge et program. Programmet burde have været opbygget efter dette koncept, men eftersom at programmet var tæt på at være færdigt, da gruppen lærte om MVC, ville det kræve for meget tid at implementere.

En god struktur for programmet, inden påbegyndelsen af programmeringen, vil gøre programmet lettere at programmere, og samtidig gøre det lettere for læsere at forstå kode og struktur. Det gør også at programmet bliver lettere at opdatere og rette i.

8.3 Programmeringsvalg

I gruppens kildekode har gruppen brugt to stykker kode fundet på internettet, et til at konvertere længde- og breddegrader til UTM koordinater og et bibliotek til at læse GPX filer. Ejerne af begge stykker kode har givet lov til ubegrænset brug af deres kode. Dog ville ejeren af GPX-biblioteket gerne have sit licens dokument med i projektet. I begyndelsen af projektet følte gruppen, at det på en måde var snyd at hente kode fra nettet, som andre havde lavet. Senere i projektet følte gruppen dog det var smart, da der ikke er nogen grund til at skrive noget kode, som allerede er blevet skrevet og stillet til rådighed.

Problemet ved at hente kode fra en tredjepart, er at det ikke vides hvorfor koden er skrevet som det er skrevet. Dette gør det svært at få fuld forståelse for alt koden og kan dermed give problemer hvis der opstår fejl i programmet, da det hentede kode vil være svært at debugge. Det samme kan siges om de allerede implementerede klasser og biblioteker i .NET frameworket. Gruppen mener selv, at det er i orden at bruge koden, så længe outputtet verificeres.

Ved konverteringen fra længde- og breddegrader til UTM koordinater er input og output allerede kendt af gruppen. Gruppen kan dog ikke forklare den matematiske formel, eller forklare hvorfor den ser ud, som den gør.

Gruppens program indeholder mange klasser, som gruppen selv mener kunne være skrevet sammen. Dette ville gøre programmet mere overskueligt og simpelt, eksempelvis kunne RunnerData og Runner skrives sammen, da de indeholder mange af de samme informationer. Dette kunne have været undgået hvis gruppen havde planlagt sit program bedre.

8.4 Test

Under test fandt gruppen en del uforudsete fejl, som blev rettet, hvilket har ført programmet til et tilfredsstillende stadie, med henhold til fejl i programmet. Dog ville gruppen gerne have testet GPS'en i mobilens præcision og undersøgt hvor stor præcision der kræves før programmet kunne bruges af o-løbere. Dog beskæftiger dette projekt sig kun med et bestemt kort/område. Da gruppen vurdere at GPS dataen indenfor dette område har været tilstrækkelig præcis, til at en sammenligning er mulig, har gruppen valgt ikke at udføre test af præcision i andre områder og/eller med andre mobiler og/eller med andet software. Dette vil dog være nødvendigt i videreudviklingen af programmet. Da dette er et centralt spørgsmål, fordi for stor usikkerhed på mobilens GPS vil gøre programmet ubrugelig.

8.5 Opsummering

Gruppens problemanalyse er bygget efter få kilder, det er troværdige kilder, idet både Jens Børsting og Claus Bobach har meget erfaring med o-løb. Her ville det have været mere optimalt, hvis gruppen havde forsøgt at få et svar fra de personer, der havde lovet svar, men som ikke svarede på mail.

I forhold til gruppens design af programmet, var det meste meget empirisk, altså forsøg og forstå, og dette medførte at strukturen i programmet var meget rodet eller ikke eksisterende.

De valg gruppen foretog i forbindelse med udviklingen af programmet, var bl.a. at anvende kode fra tredjepart, som gruppen mente var snyd i begyndelsen, men senere indså at det ville være dumt at skrive kode der allerede er skrevet og sat til rådighed. Gruppen har ikke set det

nødvendigt at teste mobilens gps, i dette projekt, gruppen er dog bevidst om at det er et område der burde undersøges i videreudviklingen af programmet.

Konklusion 9

I gennem projektarbejdet med **IT i foreninger**, har gruppen afgrænset til IT i o-løbsforeninger, og haft fokus på at optimere træningen for o-løbere. Gennem problemanalysen blev en række interesser fundet, hvor der blev foretaget interview og mailkorrespondancer, for at finde frem til projektets problemstillinger. Projektets kravspecifikation blev udformet ud fra problemstillingerne, og blev afgrænset efter evner, erfaring og tid. Dette kapitel vil gennemgå om løsningsdelen overholder kravsspecifikationen og problemformuleringen.

Kravspecifikationen er opdelt i to dele. Den ene del er den grafiske afbildning, og den anden er en tekstbaseret afbildning.

Programmets grafiske afbildning kan manipulere kortet, via zoom og flytning, afspille data, med pause og tempo funktioner, samt mulighed for at springe frem i afspilningen via en scrolling funktion (trackbar). Derudover findes funktionalitet til at til- og fravælge løbere, samt vælge hvilken post løberne skal sammles og afspilles fra. Dette gør at programmet overholder alle krav for den grafiske afbildning. Dog er nogle kun delvist løst. Tempo-funktionaliteten er skaleret anderledes end beskrevet, ca. forhold: 1x, 4x, 9x, 16x, 25x. Brugeren har heller ikke mulighed for at tilpasse halelængden og farven på løberne.

Programmets tekstuclerede afbildning viser de fleste af de ønskede informationer fra kravene. Kravet om visning af placering i løbet efter strækket og samlet tid efter strækket er ikke opfyldt. Der er blevet tilføjet information om placering for hele løbet til alle stræk.

Udover at kravspecifikationerne skal overholdes, skal projektets problemformuleringen også besvares, dette har kravspecifikationerne hjulpet med. Projektets problemformulering som er fundet frem til i problemanalysen, lød således:

Hvordan kan en telefonbaseret softwareløsning optimere evaluering og træning af o-løbere?

- Hvordan kan løberne sammenlignes?
- Hvordan følges løberen rundt på ruten?
- Hvordan kan løsningen gøres brugervenlig?

OrienteeringTracker kan sammenligne løbere på to forskellige måder: Tekstbaseret, ved Data View, og grafisk ved Map View, som viser vejvalg mv.

Løberen kan følges på ruten gennem Map View, og som beskrevet i kravene, kan brugeren skifte mellem poster og hastighed. Dette gør, at brugeren kan sammenligne løbernes vejvalg på ruten synkront.

Efter afgrænsning fra krav til en optimal løsning, valgte gruppen at se bort fra brugervenlighed, til trods for at det er et vigtigt punkt i ethvert program.

OrienteeringTracker er hidtil ikke et fuldt ud telefonbaseret softwareløsning, da programmet ikke er kørbart på en telefon. Dette skyldes, at gruppen har taget afstand fra applikationer til telefoner, og har derfor lavet et computer-program, og simuleret informationer der skulle sendes

fra telefon-applikationen. Den simulerede information fra telefonen er givet ved en tredjeparts applikation.

Programmet optimerer evaluering og træning af o-løbere, da det giver mulighed for sammenligning af både vejvalg, tider, hastighed og mere.

Perspektivering 10

Dette kapitel omhandler hvilke målsætninger og planer gruppen har for projektet, hvis gruppen skulle arbejde videre med projektets eksisterende løsning.

I gruppens kravspecifikation, har gruppen i afsnit 4.1 lavet et optimalt løsningsforslag. I det afsnit er der beskrevet hvordan gruppen ønsker løsningen skulle se ud. Nedenfor er beskrevet den fremadrettede løsningsstrategi, som viser i hvilken rækkefølge gruppen ville tilføje de ønskede funktioner.

Fremadrettet skal denne løsning kunne loade kort ind dynamisk, samt danne kontrolpunkter ved at trykke på kortet, hvor posten skal være. Derefter ønsker gruppen at lave programmet om til en webservice, med et klub- og brugersystem.

Gruppen vil da lave en mobil-applikation så GPX-filerne ikke skal komme fra et tredjepartsprogram. Herefter implementeres funktionalitet til live-tracking. Mobilapplikationen skal desuden udvides, således, at den indeholder de samme funktionaliteter som webservicen. Langsigtet vil programmet blive udvidet til at kunne håndtere kamera optagelser. Dette giver en ny dimension til tilskuer oplevelsen, samt giver træneren mulighed for at “se igennem løberes øjne”.

Gruppen ønsker også at lave en bedre brugergrænseflade, som skal være simpel og nem at navigere rundt i. Derudover skal hjælpe-tabben implementeres, der skal hjælpe brugeren i gang med programmet, hjælpe-tabben er beskrevet i afsnit 4.1.2.

Den optimale løsning, eller dele af den, vil kunne bruges i mange andre sammenhænge. Den vil kunne bruges i alle sportsgrene hvor man bevæger sig fra et punkt til et andet. Alle former for løb, cykelløb, mountain bike, adventure race og motorsport. Alle former for sejlsport og motorbåds sport, roning, kajak, ridning og væddeløb af alle slags. Hertil kommer alle de aktiviteter der ikke nødvendigvis har noget med sport at gøre, men det ønskes at vide hvor folk er og hvad de ser. Politi, beredskab, falck, lastbiler, budbringere, taxa, osv. Listen er nærmest uendelig. Det behøver ikke kun at være mennesker, det kan også være maskiner eller dyr. Alt hvor der ønskes viden om position og rute.

Litteratur

- Agency, 2009.** National Geospatial-Intelligence Agency. *Military Map Reading 201*.
<http://earth-info.nga.mil/GandG/coordsys/mmr201.pdf>, 2009. Set d. 04/5-2015.
- Britannica, 2015.** Britannica. *Latitude and Longitude*.
<http://global.britannica.com/EBchecked/topic/331993/latitude-and-longitude>, 2015. Set d. 04/5-2015.
- Dansk Idrætsforbund, 2013.** Dansk Idrætsforbund. *Medlemstal*.
http://www.dif.dk/da/om_dif/medlemstal, 2013. Set d. 27/2-2015.
- DLG.Krakow.pl, 2011.** DLG.Krakow.pl. *C# Gpx Reader/Writer*.
<http://dlg.krakow.pl/gpx/>, 2011. Set d. 10/4-2015.
- Endomondo, 2015.** Endomondo. "Google Maps can find my location but Endomondo can't".
<https://support.endomondo.com/hc/en-us/articles/201868967-GPS#googlemaps>, 2015.
Set d. 04/3-2015.
- Exforsys, 2011.** Exforsys. *What is Monkey Testing*.
<http://www.exforsys.com/tutorials/testing-types/monkey-testing.html>, 2011. Set d. 20/5-2015.
- Frivillighed.dk, 2015.** Frivillighed.dk. *Fakta: Det frivillige arbejde i organisationer og foreninger*.
<http://www.frivillighed.dk/Webnodes/Frivillige+i+organisationer+og+foreninger/16446>, 2015. Set d. 2/3-2015.
- Hactic, 2007.** Hactic. *C-Sharp code to convert DD to UTM .. here it is !!*
<http://forum.worldwindcentral.com/showthread.php?9863-C-code-to-convert-DD-to-UTM-here-it-is->, 2007. Set d. 10/4-2015.
- IOF World Ranking, 2015.** IOF World Ranking. *Rangeringsliste*.
<http://ranking.orienteering.org/>, 2015. Set d. 27/2-2015.
- LatLong.net, 2015.** LatLong.net. *Convert Lat Long to UTM*.
<http://www.latlong.net/lat-long-utm.html>, 2015. Set d. 4/5-2015.
- Odense Orienteringsklub, 2015.** Odense Orienteringsklub. "Google Maps can find my location but Endomondo can't".
<http://odense-ok.viggonet.dk/websites/odenseok/files/Emit%20instruktion.pdf>, 2015. Set d. 04/3-2015.
- Poderico, 2007.** Luigi Poderico. *GUI Tester*. <http://www.poderico.it/guitester/index.html>, 2007. Set d. 20/5-2015.
- QuickRoute, 2015.** QuickRoute. *QuickRoute Features*.
<http://www.matstroeng.se/quickroute/en/features.php>, 2015. Set d. 04/3-2015.

Schmidt, 2015. Preben Schmidt. *Orientering*. <http://www.do-f.dk/hvad/>, 2015. Set d. 02/3-2015.

TracTrac, 2015. TracTrac. *TracTrac Club*.
<http://www.tractractrac.com/files/TracTracClub-sailing.pdf>, 2015. Set d. 04/3-2015.

Interview transskribering A

Dette er et interview med Claus Bobach, foretaget af Frederik, Mark og Søren.

Søren: Vi går på software på Aalborg Universitet, og har valgt et projekt som skal omhandle små foreninger, og Jens Børsting foreslog at tage kontakt til dig.

Frederik: Lige nu er vi i en fase, hvor vi forestiller os en form for hjælp med jeres træning, med noget udstyr og software. Så vi vil gerne have dig til at fortælle lidt om hvordan en træningsgang forløber. Så kunne vi senere snakke om du har nogle forestillinger om, hvad der kunne hjælpe til træning, af software. Så hvis du kan forklare os hvordan en træningsgang forløber, helt fra i planlægger den, til udførelse og efter udførelsen.

Claus: Vi starter med et blankt kort, på et program der hedder Ocad, der har vi så et program til at lægge lagene ovenpå, Condes, et forholdsvis simpelt program som er udvidet så mange gange at det er blevet rigtig godt. Her har vi mulighed for at lægge en simpel bane, eller noget mere kompliceret som at fjerne dele af kortet. Disse to programmer har så inden for de seneste år kunnet arbejde sammen, og de lag der så ligger i Ocad er delt om i farver og symboler, og i Condes kan man så bestemme hvad af det man vil have med. Så kan man eventuelt vælge kun at se kurvebilledet, når de to programmer arbejder sammen. Hvis ikke man har en cad-fil, så kan man bruge JPG filer, men så har man ikke samme mulighed for at ændre kortet. Så når jeg har lavet banen på et kort, så sender jeg det videre til en som printer det, og sender det til dem som sætter posterne ud. Jeg har så en stak kort til de folk som møder op. Så løber folk ellers ind i skoven og kommer hjem igen.

Til de fleste træninger står folk selv for at skulle tage tid, hvis det er noget de vil. På de ugentlige træninger har vi ikke noget tidstagnings udstyr. Det har vi dog til de lidt mere specielle træninger, blandt andet traditionsløb og sådan, hvor vi har elektroniske enheder ude ved posterne. Her har vi så en enhed vi kan se det hele på, og printe ud fra.

Frederik: Kan du sige lidt om, hvor lang tid det tager at lave kort og sætte poster ud?

Claus: Jeg bruger normalt en time eller halvanden på at lave de tre baner vi løber på til almindelig træning, men vi har en gruppe på otte som skiftes til at sætte posterne ud. At sætte posterne ud tager typisk et sted mellem halvanden til to timer. Så der er selvfølgeligt noget forberedelse. Også i form af at hente posterne igen, men det prøver vi på at gøre i forbindelse med træningen. For eksempel i dag, der løb jeg bare som den bagerste og samlede posterne ind.

Frederik: Du sagde, at i kunne vælge kun at have højdekurverne på kortet. Hvordan er dette relevant?

Claus: Blandt andet her om onsdagen, der prøver vi på at øve forskellige teknikker, nu hvor orienteringsløb er rimelig komplekst, så vi deler det lidt op, så man kan øve forskellige ting. Det er der vi bruger det. Kurvebilledet er en af de ting der er vigtige at træne og læse. Så for at få det mere simpelt til træningen, har vi nogle gange kun kurvebilledet på kortet.

Søren: Når i skal evaluere hvordan i løber, hvordan gør i det?

Claus: Det optimale er selvfølgeligt at vi havde enheder på hver gang, men det

tager allerede halvanden time til to timer at sætte poster ud, og hvis vi så skal have enheder med hver gang, tager det en hel time mere. Og det at løbe bagerst og samle posterne ind, vil heller ikke være lige så nemt. Men der er to muligheder, som man også kan kombinere, men at bruge tiderne mellem de enkelte poster eller at bruge GPS-ur er de to muligheder der typisk bruges. Så de fleste scanner bare kortet ind og lægger GPS-dataen oven i.

Frederik: Ved du hvilket program de bruger til det? Altså til at samle GPS dataen.

Claus: Der er en orienteringsløber oppe i Stockholm, eller sådan noget, det hedder Quickroute. Hvor han henter alt data hjem der ligger i gps uret, det vil sige puls og højde osv. Så man kan få graferne sammen og få det visuelt på kortet at hvis man eksempelvis vælger hastighed, så har løberen en skala fra rød til grøn.

Frederik: Så man kan se hvor man har løber hurtigt og langsomt?

Claus: Ja.

Frederik: Okay. Kan den replay det? Altså så man kan se hvor langt man kom?

Claus: Ja, jeg har faktisk ikke prøvet funktionen, men ja. Jeg tror han har lavet det og lavet det i forbindelse med Google Earth. Hvor Quickroute ligger o-kortet ind i Google Earth også laver den en replay der.

Frederik: Okay, det tror jeg vi skal have kigget nærmere på hvad det er for noget.

Claus: Der er helt klart også noget interessant der.

Frederik: Når i nu ikke har tidtagning med der ude, hvordan evaluere i så bagefter? Det er måske bare lidt for at få erfaringen og holde formen ved lige?

Claus: En ting er jo at nogle har deres GPS ur og selv går hjem og evaluere, men vi opfordre selvfølgelig også folk til at snakke sammen. Bare det at have set hinanden i skoven, kan man godt fornemme hvilke udfordringer de andre har haft. Det man har gjort helt tilbage fra starten af, er jo man har sat sig ned og snakket om hvad gjorde du og hvad gjorde jeg.

Søren: Så vidt som de kan huske selvfølgelig.

Claus: Ja selvfølgelig.

Søren: Hvad med andre redskaber der kan bruge til at evaluere folk i deres træning?

Frederik: Er der f.eks. nogle der bruger deres mobil telefon?

Claus: Jeg tror det er mest til GPS urene, der er nok nogle få der bruger Endomondo, men ellers tror jeg ikke det er noget der er så udbredt. Det er så det Quickroute, hvor det er meget for den individuelle løber. Der er også nogle andre programmer, hvor arrangørerne, altså efter en konkurrence, at de lægger et kort op, hvor alle så kan lægge deres GPS rute ind oven i. De bliver desværre ikke brugt så voldsomt meget lige i forløbet, det var meget populært for en 3-4-5 år siden. Det var som om der var et enkelt program som var rigtig brugervenligt, som folk brugte, men det var der så øbenbart ikke økonomi i, så det svenske forbund valgte at begynde at bruge noget andet, det har man så ikke rigtig for arrangører og løber ind til at støtte op om. Det kræver så, for virkelig at få noget ud af det, skal man have 20-30% af løberne til at bruge det.

Frederik: Ja okay, det er vigtigt de alle sammen er med på det.

Claus: Ja, så derfor kan det godt være vigtigt som arrangør, hvor vi nu skal arrangere et løb her om $2\frac{1}{2}$ uge for 1.500 løbere, hvis vi f.eks. som vi nogle gange har gjort lægger kortet op i et af de der programmer, så er det vigtigt at vi reklamere for det, ellers bliver det jo bare udvasket.

Frederik: Kan du huske hvad det hed?

Claus: Altså det de stoppede med at bruge, det hed RunOWay.

Søren: Vi har også hørt om TracTrac, om det er noget der er udbredt? Men som vi kunne forstå, var det forholdsvis dyrt.

Claus: Det bliver mest brugt til at live rapportere. Enten hvis vi sidder der hjemme, når der er VM et eller andet sted ude i verden, så kan vi se med hvor. De bruger det også til at have en

stormskærm på pladsen, når der er konkurrencer.

Frederik: Det er mest til større events?

Claus: Ja det er det. Jeg er lidt i tvivl om landsholdet måske, har brugt det lidt.

Frederik: Jeg tror de bruger det, det mente Jens Børsting i hvert fald.

Claus: Det tror jeg faktisk også. Det udstyr der bliver brugt til de store konkurrencer herhjemme, har landsholdet jo til dagligt. Så jeg tror måske de bruger det lidt, men jeg ved faktisk ikke hvor meget.

Frederik: Jeg tænkte om du måske havde nogle idéer til områder, hvor du kunne forestille dig der var et eller andet en eller anden form for optimering i forhold til noget software du havde tænkt over? Inden vi snakker vores idé.

Claus: Vi har et enkelt problem i øjeblikket, men det tror jeg også er lidt et spørgsmål om vilje herfra. Når vi arrangerer et o-løb herfra, så for at få data ind fra skoven, ikke GPS data, men bare tidsdata, bare fra en enkelt post eller to, så er vi meget afhængige af... Det er vi nød til at kable ud til. Altså, at vi deciderer har et fysisk kabel, og det begrænser lidt muligheden for hvor langt udefra man kan få meldingen. Så der vil vi jo gerne have en enhed som kunne sende det, eller nogle programmer der kan håndtere det, for jeg tror nok at enheden findes. Jeg tror nok de bruger det nede i Sønderjylland, men de bruger et andet program der skal modtage det. Så det, ja, det er en af vores udfordringer i øjeblikket.

Mark: Men er det i forbindelse med at få data direkte, eller når i kommer tilbage og skal evaluere.

Claus: Det er med det samme. Vi har en mand på pladsen der sidder og speaker om hvordan det går. Lige nu bruger vi det bare sådan at han har en forvarslig på hvem der kommer næst, altså så han kan forberede sig, men specielt på de lange distancer ville det være rare at have en post halvvejs han kunne speake nogle tider fra.

Frederik: Ja men vi kunne måske snakke lidt om hvad vi har tænkt på. En af de første ting vi tænker, var virtuelle poster, hvor du får en form for lyd, når du er i nærheden. Men vi fandt noget der faktisk var det vi havde forestillet os, og hvis det er ude, og det ikke bliver brugt, så tænkte vi at det nok ikke var noget der var så aktuelt. I stedet, så vil vi gerne prøve at lave noget der ligne TracTrac, hvor du bruger din mobil som GPS-sender, for så vidst vi har forstået, så er der ikke nogle af GPS-urene der kan sende data direkte. Nu må vi se hvor præcise GPS'erne i mobilerne er, men så ville det jo være noget man kunne gøre live, og så ville det være noget hvor man, igen som du siger, lave en eller anden form for program som arrangørerne laver, og så få lagt kort ind, og få samlet alle folk ruter og sådan nogle ting, så de kan sammenligne, meget ligesom TracTrac, men så prøve at gøre det lidt billigere fordi det er via mobilen. Men det ved jeg ikke hvad du tænker om?

Claus: Jamen det tror jeg da helt klart kunne være interessant.

Frederik: Er det noget til sådan en almindelig træning som i dag, tror du der er nogle der ville bruge det der, eller ville det være til arrangementer?

Claus: Altså det live mæssige ville vi nok ikke bruge til træning, men jo hvis det er noget man nemt bagefter ville kunne tage frem, så tror jeg da helt sikkert at det...

Frederik: Altså jeg tænkte at hvis man nu havde en projekter op her på væggen, så kunne man...

Claus: Så kunne man bruge det umildbart efter ja.

Frederik: Det var i hvert fald det vi havde i tankerne

Claus: Jamen det er da rigtigt, det tror jeg godt man kunne få nogle til at bruge

Frederik: Så er i også fri for de der tidtagninger ude i skoven, så behøver i ikke dem. Så kan i jo bare se.

Claus: Nej nej

Frederik: Det er vores tanke lige nu

Claus: Der er nogle af de der, der har gjort noget lignende. Det vi bruger posterne ude i skoven til, er jo at få mellemtider. Der er nogle af de der programmer der faktisk har kunne gøre det ved hjælp af GPS'en.

Frederik: Så den registrere om man er kommet tæt nok på posten, og at man så regner med at man har fundet den.

Claus: Ja, et eller andet estimat af hvornår folk har været ved posten, så man ved hvor lang tid man har brugt mellem posterne, uden at man skal sidde og.. ja..

Søren: Jeg tror egentlig også vores primære ide, var at man kunne sammenligne de vejvalg og de stræk man tager, så vi kunne tage fra en post, og...

Claus: Ja ja

Frederik: Jeg havde ikke tænkt over den med at det tager længere tid for jer at sætte de poster ud der skal brik til, end dem der ikke skal, men det er klart.

Claus: Men det gør det. Fordi det vi sætter ud nu, det er skærme i den her størrelse, det koster jo ingenting at have med.

Frederik: Så du skal selv ligesom, hvad skal man sige. Der er ikke noget papir for at validere at du har været derude.

Claus: Nej, vi hænger heller ikke stifteklemmer derude, for det er jo også. Hvis du løber rundt med nogle poster med en snor på, og der hænger noget tungt ude for enden, så i løbet af 5 minutter, så er de snore viklet sammen.

Frederik: Jeg tror heller ikke vi har så meget mere.

Søren: Nej, egentlig ikke.

Frederik: Så er det pizza tid!

E-mail korrespondance med Jens Børsting B

Hvordan foregår en træningsgang af o-løb?

Man starter med at få en skovtilladelse hvor løbet skal foregå. Træneren beslutter hvad der skal trænes og planlægger løbet i et computer program der hedder "Condes". Her lægges baner og tegnes kort. Herefter udskrives kort for alle de baner der skal løbes. Dette inkluderer et kort til udsætning af alle poster. Det meste af dette arbejde kan gøres hjemmefra. Dog er det ofte sekretæren i klubben der printer kortene. Typisk dagen før træningen hentes alle poster i klubhuset og sættes ud i skoven. På træningsdagen mødes alle løbere og får instruktion i løbets momenter i dag og baner fordeles alt efter niveau og kondition. Typisk vil der være 3-7 baner at vælge imellem. Løberne sendes i skoven. Alle løber med en elektronisk brik der registrerer hvornår man har været ved her post. Ved hjemkomst får man en udskrift over hvilke poster man har været ved og hvor lang tid der er gået mellem hver post (stræktider) Man har så mulighed for at gennemgå løbet ved at snakke med andre løbere og trænerne. Her diskuteres vejvalg og hvad der gik godt og hvad der gik mindre godt. Stræktider sammenlignes. Vejvalg foregår udelukkende efter hukommelse og det er ikke muligt at se forskel i hastighed på kortere stræk end hele strækket mellem 2 poster. Hvis løberen ikke er sikker på hvor han/hun har været kan det være meget svært at analysere hvad der gik galt. Dagen efter løbet skal alt udstyret der er placeret i skoven hentes ind igen og pakkes ned i klubhuset hvis det ikke skal hænges til tørre først.

Hvordan evaluerer trænere deres løbere? (Hvad vurderer han på, og hvordan observerer han dette?)

Eneste mulighed træneren har er at snakke med løberen efter løbet og ud fra stræktider og løberens hukommelse af vejvalg diskutere hvad der virkede og hvad der ikke gjorde.

Hvordan evaluere respondenten sin egen tur? (hvordan kan respondenten selv se fremskridt eller fejl ved sin egen træning?)

På samme måde som med træneren kan løberen læse sine stræktider og evt. sammenligne med andre løbere der har løbet sammen stræk. Det kræver dog at disse løbere er til stede. Ved de ugentlige træninger bliver stræktider ikke offentliggjort, så man kan ikke sammenligne online efter løbet. Ved større løb bliver alle stræktider offentliggjort på nettet og man kan sidde hjemme i ro og mag senere og analysere sit løb i forhold til andre. Det er vigtigt at man kan sammenligne med andre da der er stor forskel på løbsterræn og dermed hastighed i terræn ved forskellige løb. Man kan dog sammenligne med dem man normalt lige op med løber med og kan se om man har løbet hurtigere eller langsommere end den på dette løb i forhold til tidligere løb.

Hvordan kan denne evaluering gøres mere præcis, eller endda inkludere flere aspekter af løbet?

Ved at have gps tracking som kan følges live under løbet kan træneren se hvordan løbet forløber og nemmere snakke vejvalg og andet teknisk træning efter løbet. Evt. kan vejledning gives under

løbet, hvis løberen er faret helt vild. Tracks kan sammenlignes med andre for at se hvor på strækket der har været fart på og hvor der ikke har. Samlet afspilning af løberne fra en post til en anden kan give et godt billede af udviklingen af løbet. Det at have gps med på løbet kan også give mulighed for at give elektronisk tilbagemelding om at man er ved posten, kan man reducere arbejdsmængden ved udsætning og indhentning af poster. Samtidig er løberen uafhængig af hvornår der er poster i skoven og kan løbe løbet når det passer ham (dog skal der være løbstilladelse i skoven).

Hvilke redskaber og metoder kender respondenten til, og hvilke bliver generelt set brugt i træningen?

Kort, stræktider og hukommelse er det normale. Sammenligning af stræktider med andre ved løbets afslutning eller ved større løb online hjemme i de efterfølgende dage. Nogle få kender til Travel Tales og PDFmaps, men det bliver stort set aldrig brugt da det ikke giver en samlet løsning. TracTrac og lignende systemer bliver kun brugt af landsholdsløbere og til nogle få eliteløbere til de store stævner.

Hvordan mener respondenten, at en træningsgang kan blive optimeret? (Hvilke evaluerings punkter er mangelfulde)

Ved at kunne sammenligne gps tracks kan man nemmere vurdere om hvilke vejvalg der har virket og hvor det er gået galt/langsomt. Her kan man se om det kunne betale sig at løbe udenom eller det var nemmere at løbe igennem en tæthed eller over en stejl bakke mv.