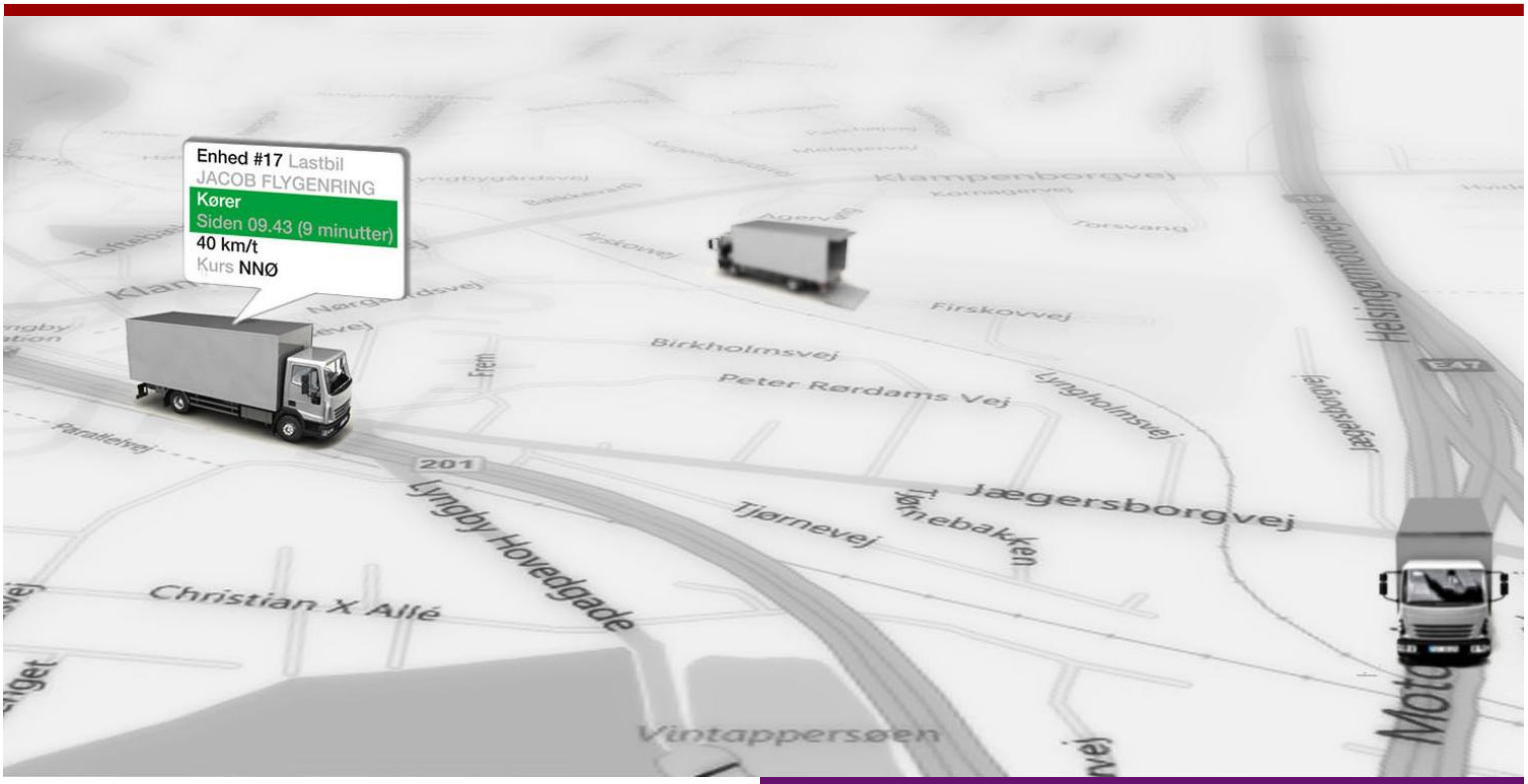
Lokationsbaseret styring af logistiske processer
med optimerede sensordata fra smartphones



DTU Informatik
Institut for Informatik og Matematisk Modellering

Lokationsbaseret styring af logistiske processer med optimerede sensordata fra smartphones
Kandidatspeciale i Software Engineering

Rapporten er skrevet af:
Jacob Flygenring

Vejleder(e):
Ekkart Kindler
Karsten Holm

Institut for Informatik og Matematisk Modellering
Danmarks Tekniske Universitet
Asmussens Alle
Bygning 305
2800 Kongens Lyngby
Danmark
www.imm.dtu.dk
Tel: (+45) 45 25 33 51
E-mail: reception@imm.dtu.dk

Rapportnummer	IMM-M.Sc.-2011-49
Udgivelsesdato	11. juli 2011
Klassifikation	Offentlig
Udgave	1. udgave
Noter	Denne dokument er udgivet som kulminationen på det arbejde, der kræves for at opnå en kandidatgrad (Master of Science in Engineering (M.Sc.)) på Danmarks Tekniske Universitet. Specialet udgør 30 ECTS-point.

Anerkendelser

Jeg vil gerne takke de mennesker, der har hjulpet med at skrive denne afhandling.

Først og fremmest vil jeg gerne takke min vejleder lektor Ekkart Kindler for feedback, vejledning og inspiration gennem hele processen. Din åbne tilgang til projektet og store viden indenfor feltet Software Engineering har været en uvurderlig hjælp, samtidig med at din velstrukturerede og -organiserede arbejdsform har givet mig de helt rette rammer for at kunne gennemføre projektet.

Dernæst vil jeg takke Karsten Holm for hans hjælp og vejledning i forbindelse med de mere tekniske og praktiske sider af projektet, og for inspirerende diskussioner under alle faser i projektet. Jeg vil også takke Soft Design, herunder specielt Anders Petersen, Thomas Petersen og ikke mindst Morten Knudsen for deres hjælp til at projektet kunne lade sig gøre.

Jeg vil også takke de mennesker, der har hjulpet med at korrekturlæse dette speciale. Specielt Thomas Bef Flygenring, Thea Schmidt Poulsen, Kristoffer Negendahl, Sussi Maria Jensen og Morten Krastins for deres særligt store indsats. Også Anne Riis og Torben Flygenring har været til utrolig stor hjælp under hele processen.

Endelig vil jeg sende en særlig tak til Søren Sig Bethge for samarbejdet under mit studie og for ukritisk at hjælpe mig med de problemer, jeg måtte stå med – en uundværlig sparringspartner, i tide og utide.

Jacob Flygenring
Lyngby, juli 2011

Indholdsfortegnelse

Figurliste

Ordforklaring

Dette projekt gør brug af følgende navne, titler og begreber.

Navne og titler

Fokusområde

”Lokationsbaseret styring af logistiske processer med optimerede sensordata fra smartphones”.

Soft Design

Firmaet Soft Design A/S der er beliggende i Søborg.

Det eksisterende system

Produktporteføljen ”Synchro**nicer**”, der udbydes af Soft Design. Omtales også blot som *Systemet*.

Den centrale server

Systemets back-end, der tager hånd om kommunikation med database og forretningslogik på højere niveau end den enkelte enhed. Omtales også blot som *Serveren*.

Enheden

Den mobile enhed hvorpå applikationen skal køre. Forventes at være en smartphone baseret på Android-platformen.

Platformen

Smartphone-operativsystemet Android. Dette projekt antager at applikationen bliver afviklet under version 2.2 af Android.

Applikationen

Produktet vis virkefelt og funktionalitet bliver beskrevet gennem dette projekt.

Centrale begreber

Android

Smartphone-operativsystemet, som antagelser i dette projekt er baseret på.

Applikation

Under Android omtales et program, der interagerer med brugeren, som en Applikation.

Applikationskomponent

Under Android består applikationer af nul eller flere komponenter. Disse komponenter kan være aktiviteter, services og broadcast-receivere

Broadcast-receiver

En applikationskomponent der har til formål at lytte efter opdateringer, der bliver udsendt fra forskellige kilder, for eksempel andre applikationskomponenter eller sensorer.

Wi-Fi

Trådløs kommunikation mellem elektroniske enheder. I dette projekt dækker denne betegnelse over muligheden for at være forbundet til internettet i overensstemmelse med IEEE 802.11 standarden.

Brugergrænseflade

Den del af en applikation, der sætter brugeren i stand til at interagere med applikationen.

Workflow

De interne sammenhænge og skift mellem processer i applikationen.

Passiv enhed

Enhed der ikke foretager datavalidering eller -behandling eller lader sig påvirke af ændringer i tilstandene i resten af det system den indgår i. Bruges her om en enhed der udelukkende sender GPS-positioner til serveren med et fast tidsinterval.

Aktiv enhed

Enhed der tilpasser sig til de forhold den opererer under. Bruges her om en enhed der behandler data inden den sender dem til serveren, og kan tilpasse dataopsamlingen efter hvad der er nødvendigt i den igangværende proces.

Sensordata

Informationer der kommer direkte fra en sensor. Disse er oftest meget simple data.

Simple data

Data der ikke indeholder en kontekst i sig selv, men skal sammenholdes med andre data for, at kunne sige noget om for eksempel en specifik hændelse.

Værdifulde data

Data der er blevet behandlet eller på anden måde indsat i en kontekst, og for eksempel blevet bundet til en specifik hændelse. Disse anses for værdifulde fordi de siger noget i sig selv, og ikke behøver at blive kædet sammen med andre data for at bringe vigtig viden til et system.

Logistiske processer

Processer der beskriver en unik handling eller hændelsesforløb i forbindelse med udførelsen af en logistisk opgave. Det kan for eksempel være transporten af en container fra den bliver hentet af en lastbil og til den bliver afleveret ved sin destination, eller at en kunde kvitterer for modtagelsen af en vare.

Tidsplan

Plan over de opgaver der er planlagt, at medarbejderen skal udføre i løbet af arbejdsdagen.

Registrering

En afgrænset mængde data der indsendes til serveren. I det eksisterende system er dette for eksempel en enkeltstående GPS-position.

Lokation

En velkendt geografisk placering, for eksempel en adresse. I kontrast til en position, der blot er punkt på et kort.

Forkortelser

AIDL (Android Interface Definition Language)

Et sprog der bruges til at beskrive grænseflader mellem applikationskomponenter under Android.

API (Application Programming Interface)

En applikationskomponents grænseflade udadtil, der sætter andre udviklere i stand til at udnytte komponentens funktioner uden at kende dens implementering i detaljer.

GPS (Global Positioning System)

Netværk af satellitter, der kan benyttes til at bestemme positioner ved hjælp af en trianguleringslignende metode.

GUI (Graphical User Interface)

Grafisk brugergrænseflade. Under Android er der i praksis ingen ikke-grafiske brugergrænseflader.

IPC (InterProcess Communication)

En metode der bruges til at kommunikere mellem applikationskomponenter under Android.

SDK (Software Development Kit)

En samling af alle komponenter og API'er, der er nødvendige for at programmere software til et specifikt system.

VM (Virtual Machine)

En måde at afvikle programkode under et andet system end det er beregnet til. Er den mest udbredte måde at afvikle Java, da sproget specifikt ikke er rettet mod nogen bestemt platform.

1 Indledning

Enhver virksomhed der har medarbejdere, som bevæger sig rundt i verden, og udfører opgaver af forskellig karakter, har et vist ønske om og behov for at vide, hvor medarbejderen har været, og hvad medarbejderen har lavet. Det kan for eksempel dreje sig om at en bestemt rute skal følges, eller nogle varer der skal leveres et bestemt sted, til en bestemt person og på et bestemt tidspunkt.

Den mobile medarbejder har også en interesse i at kunne dokumentere sin færden og ikke mindst at kunne blive løbende opdateret om ændringer i sin køreplan og blive dirigeret den rigtige vej til sine planlagte stop. Men arbejdet med at registrere hvilke veje man har kørt ad, hvor man har holdt længere stop, og hvornår man har besøgt en bestemt kunde for at levere en vare eller yde en service, kan være meget omstændeligt. Traditionelt er det foregået med papir og blyant, hvilket har krævet medarbejderens opmærksomhed, hver gang en handling skulle registreres, hvorefter medarbejderen fysisk skulle overlevere sine registreringer til virksomheden inden det udførte arbejde kunne valideres, og der kunne laves tidsplaner på baggrund af deraf. Ydermere havde virksomheden udelukkende medarbejderens og eventuelt kundens ord, for at arbejdet var udført, og eventuelle fastlagte kørselsruter var fulgt.

Efterhånden som teknologien har udviklet sig, kan større og større dele af denne registreringsproces foregå automatisk. En GPS kan bruges til at dokumentere, hvor medarbejderen har været. Stregkoder og elektroniske underskrifter kan verificere vigtige handlinger, som medarbejderen har udført, for eksempel et besøg ved et bestemt sted, en alarm på en vagtrute eller en varelevering hos en kunde. Mobiltelefonnettet kan udnyttes til at sende registreringerne hjem til virksomheden, efterhånden som de sker, endvidere kan der sendes eventuelle opdaterede køreplaner og andre beskeder tilbage til medarbejderen.

Registreringsprocessen bliver lettet meget for både medarbejdere og virksomhed ved at foregå automatisk og digitalt, da medarbejderen kan koncentrere sig mere om sit arbejde, og registreringerne kommer direkte ind i virksomhedens systemer, i takt med at de reelt forekommer. Samtidig har virksomheden bedre mulighed for at holde styr på, hvor dens medarbejdere er henne, og dermed kan virksomheden bedre uddele nye arbejdsopgaver.

Desuden elimineres en fejlkilde, idet der af og til vil ske fejl i manuelle indtastninger af tidsplaner og registreringer.

Hele markedet med automatiske digitale registreringsprocesser og håndtering af mobile medarbejdere kaldes samlet set for *flådestyring*.

I takt med at de tekniske muligheder for at automatisere og optimere medarbejdernes arbejdsgange er blevet bedre, vokser markedet for flådestyringssystemer til stadighed. Ifølge både firmaer i branchen og uvildige institutioner, forventes denne tendens at fortsætte de kommende år[1].

1.1 Baggrund

"Synchro**nicer**" er en produktportefølje der bliver udbudt af firmaet Soft Design. Den centrale funktionalitet i porteføljen er håndtering af opgaver indenfor flådestyring, herunder styring af logistiske processer.

Porteføljen består af en række produkter, der henvender sig til virksomheder, som gør brug af mobile medarbejdere, hovedsagligt indenfor servicebranchen, transportsektoren og den kommunale sektor. Produkterne bliver typisk anvendt i forbindelse med for eksempel servicepersonale, teknikere, bude, chauffører og speditører. Systemet er udbredt i Danmark, hvor det aktuelt benyttes af knapt 100 forskellige virksomheder og kommuner. Virksomheder i flere andre europæiske lande har vist interesse i, at systemet bliver tilgængeligt udenfor Danmarks grænser.

Som systemet fungerer nu, er det en passiv enhed der med faste tidsintervaller, indrapporterer GPS-koordinater til en central server, hvorefter disse behandles så de kan bringe værdifuld information til systemet.

Behandlingen sætter hvert af de indrapporterede GPS-koordinater i sammenhæng med tidligere indrapporterede koordinater og andre kendte data om den igangværende arbejdsgang. Herefter tilpasses det videre forløb for medarbejderens tidsplan i forhold til dette. Når informationen er behandlet og fortolket, kan den tilknyttede medarbejder indsættes i en ny arbejdsgang, og der kan kommunikeres nye instruktioner til enheden, baseret på medarbejderens position.

[1] Dokumentation herfor kan findes i kilderne listet i afsnit 6.4.2 Referencer.

Kommunikationen sker i dag gennem et simpelt interface i en webbrowser på medarbejderens enhed, der viser en webside med medarbejderens køreplan, hvor eventuelle opdaterede informationer er indsat. Denne webside genereres automatisk af systemet, hver gang medarbejderen sender en *registrering* eller anden information til virksomheden.

Lokationsbestemmelse af en medarbejder ud fra GPS-koordinater afsat med faste tidsintervaller kan sige en del om hvor, og hvordan medarbejderen bevæger sig. Det enkelte datapunkt i sig selv giver dog ikke mere information end et punkt på et kort, og ikke kan spores til en bestemt handling fra medarbejderens side. Derfor er det at betragte som data, der er ineffektive i opsamlingsmetode, fleksibilitet og anvendelighed i forhold til andre arbejdsgange. Det kræver også en vis efterbehandling af disse data, ud fra deres kontekst, før end de bliver brugbare i praksis. Dertil kommer at de data der leveres, ikke altid er til at stole på, da forskellige forstyrrelser kan gøre, at der kommer forkerte eller slet ingen data fra enheden.

1.2 Problemformulering

Problemet i forhold til styring af logistiske processer er, at identificere hvilket grundlag processerne skal styres på.

Den traditionelle tilgang er at lade brugeren manuelt styre hvilken proces der er under udførelse og indsende forskellige registreringer til systemet i forbindelse med processen. Sideløbende er brugerens positioner blevet opsamlet med et fast tidsinterval og indsendt til systemet. Efterfølgende sammenholdes de to separate datasæt og et samlet hændelsesforløb kan tilnærmes.

Ved at lade den mobile enhed selv styre hvilken proces der er under udførelse, kan de positioner som enheden opsamler tilpasses til processen og kobles direkte sammen med brugerens handlinger.

Problemet er at kunne identificere de forskellige processer ud fra de data der er til rådighed, og at kunne behandle disse i en grad så de kan knyttes til specifikke handlinger eller hændelsesforløb i de enkelte processer. Hvis det skal kunne lade sig gøre skal opsamlingen af data kunne ske som en aktiv del af udførelsen af de forskellige logistiske processer.

I dette projekt er det derfor målet, at identificere hvilke data der kan opsamles og hvordan de kan optimeres til at styre systemets logistiske processer, og dataopsamlingen dermed kan blive en integreret og tilpasset del af systemets workflow. På denne måde kan det sikres, at der i enhver given arbejdsgang kun blive opsamlet data, der er relevante for netop den arbejdsgang. Hvert stykke data kan forbehandles på enheden, så det er tilpasset bedst muligt til arbejdsgangens behov, hvilket kan gøre enheden selv i stand til at afgøre hvilken proces der er under udførelse. Det gør samtidig, at hvert enkelt datapunkt bliver sporbart til en bestemt handling fra medarbejderen.

1.2.1 Flådestyring via smartphone

Der er aktuelt en markedsudvikling med smartphones, der introducerer en række nye sensorer og muligheder, samt en væsentligt forbedret regnekraft i forhold til tidligere mobile enheder. Dette både kan og vil flådestyringssystemer udnytte.

Idéen i dette projekt er at udnytte den nyere generation af smartphones, der findes på markedet, ved at bruge deres indbyggede sensorer til at give et mere nuanceret billede af hvordan de bevæger sig. Disse sensordata kan hjælpe enheden til at analysere hvornår der skiftes mellem forskellige logistiske processer eller indtræffer andre hændelser der kan være interessante at knytte til en bestemt lokation. På baggrund heraf kan enhedens position så opsamles og indberettes til systemet.

I første omgang er det interessant at klarlægge hvorvidt det kan anvendes til at overtage og forbedre funktionen af de gamle enheder. Som behovet opstår, vil det sidenhen være muligt at tilføje nye funktionaliteter, såsom at måle en maksimal hældning på container under transport, fastslå om en vare har været udsat for stød eller fald undervejs eller give et mål for hvor behageligt en chauffør har kørt med henblik på passagerbefordring.

Fordelene ved disse enheder er, at der findes mange typer og fabrikater, der i store træk overholder de samme specifikationer, og som er meget billige i forhold til de traditionelt anvendte enheder. Fælles for dem er også, at de alle har adskillige sensorer, herunder både GPS og accelerometer, der afgiver mange data, som kan aflæses præcis, når systemet har behov for det. Disse sensorer er meget interessante og anvendelige til at gøre de indhentede data mere sigende og sporbare, i forhold til hvad medarbejderen foretager sig.

Smartphones har også en forholdsvis stor regnekraft, der med fordel kan bruges til at forbehandle data fra sensorerne, så de er mere værdifulde og anvendelige, allerede når de

forlader enheden. Det vil også sætte enheden i stand til at skifte arbejdsgang direkte på baggrund af de opsamlede data, uden at skulle vente på svar fra centralen. En anden vigtig pointe er at enheden vil kunne udføre automatiseret opsamling af data som brugeren ikke behøver at være en del af og ikke behøver at skulle initiere manuelt. Derved fjernes også den fejlkilde at brugeren glemmer at udføre registreringer.

1.2.2 Udfordringer ved flådestyring via smartphones

En udfordring ved disse sensorer er dog, at de producerer en potentielt endeløs strøm af informationer. Det er derfor nødvendigt at sortere grundigt i informationerne for udelukkende at få de nødvendige data.

Der findes ingen implicit måde, at fortælle systemet hvilke typer af informationer, der er interessante og nødvendige i de forskellige processer i systemet. En given information stilles til rådighed ved at angive et fast tidsinterval hvor den ønskes, så yderligere muligheder må tilnærmes ved et samspil mellem tidligere opsamlede sensordata og en efterfølgende justering af intervallerne på baggrund heraf.

1.2.3 Muligheder for optimering af sensordata

Der er flere led i en optimering af brugen af sensorer, der kan gøre systemet mere intelligent og fleksibelt. En stor del af intelligensen kan ses som en vurdering af, hvornår der sker noget 'interessant' i de data der opsamles, hvilket for mobile medarbejdere kan være at holde styr på den fulgte rute, steder hvor medarbejderen har holdt længere stop, og hvor vidt enheden har bevæget sig under fastlagte hvileperioder.

1.2.3.1 Optimeret opsamling af sensordata

En løbende tilpasning af perioden mellem indhentning af GPS-koordinater efter enhedens position og bevægelse, for eksempel gennem brug af accelerometer og andre sensorer, vil give et meget mere nuanceret billede af enhedens bevægelser over tid.

På baggrund af den igangværende proces og arbejdsgang kan der være forskellige krav til hvilke informationer, der er nødvendige i systemet. Det kan være et planlagt stop, hvor en vare skal afleveres til en kunde, og enheden derfor har behov for at vide hvor og hvornår det skete, samt validere leverancen ved for eksempel at skanne en stregkode eller modtage en underskrift. En anden mulighed kan være, at verificere den rute medarbejderen har fulgt ved at anvende informationer om medarbejderens kørestil som grundlag for at indhente positioner.

Kørestilen kan anvendes ved at antage, at behovet for indhentning af GPS-koordinater er direkte proportionelt med den fart, enheden bevæger sig med, da en højere hastighed ved bevægelse langs jorden hænger sammen med længere distancer, der tilnærmer sig rette linjer, og det fordrer lavere hastigheder, des skarpere enheden skal dreje.

En anden mulighed er at bruge andre sensordata, der beskriver kørestilen gennem enhedens bevægelse i det tredimensionelle rum. Dermed kan dataopsamlingen optimeres i forhold til enhedens bevægelse, retning og orientering. Dette skaber grundlag for, at enheden kan registrere, hvor den befinder sig, hver gang der foretages et sving eller en længerevarende standsning, hvorved ruten bliver fuldt ud sporbar, og alle stop kan registreres præcist og automatisk.

1.2.3.2 Forbehandling af opsamlede sensordata

Da der kan være udsving og fejl i de opsamlede data, og der selv efter optimering opsamles flere data end nødvendigt for systemet, er databehandling nødvendig inden data sendes fra enheden.

For at sikre mod fejl i de opsamlede datapunkter kan de valideres i forhold til data opsamlet både før og efter, sted og tidspunkt for opsamling samt forventet fremtidigt forløb, for herved at kunne identificere uventede udslag, der ikke kan forenes med andre opsamlede data og frasortere disse.

En optimering af hvilke datapunkter der tilbagerapporteres, kan bringe større værdi til de observerede data ved kun at vælge de vigtigste, for eksempel steder hvor enheden drejer skarpt (skift mellem veje), og steder hvor enheden gør stop i længere tid ad gangen (adresser og eventuelt optankning, men ikke lyssignaler og almen bykørsel). Dette kan yderligere suppleres med at udelade datapunkter, der ligger tilnærmelsesvist som midtpunkter på rette linjer (kørsel på lige strækninger uden særlige/interessante hændelser).

1.3 Afgrænsning

Det hovedsaglige fokus for dette projekt er at analysere, hvordan sensordata fra en smartphone kan optimeres til at drive de logistiske processer for en mobil enhed i et flådestyringssystem. Det betyder en analyse af dels hvilke processer systemet gennemgår, og dels om enhedens sensorer kan bruges til at styre disse.

Det indebærer, at de forskellige sensorer, der er til rådighed i smartphones, vurderes og analyseres i forhold til hvorvidt de er egnede til positionering og styring af de logistiske processer, herunder også hvordan data fra dem kan opsamles og optimeres.

Målet er at afdække hvilke logistiske processer systemet indgår i, hvordan de hænger sammen og hvilke data der vil være relevante i de individuelle processer og arbejdsgange.

Der udarbejdes et antal prototyper til at evaluere de beskrevne modeller og problemstillingers anvendelighed i praksis. De fungerer som vurderingsgrundlaget for en sandsynliggørelse af systemets funktionalitet, og det er ikke forventeligt, at de fungerer som et brugbart system.

2 Analyse

For at identificere hvilke informationer der er relevante og kan hjælpe i styringen af systemets logistiske processer, er det nødvendigt med en dybere forståelse af systemet og problemerne forbundet med at ændre enhedens funktionalitet dynamisk. I det følgende vil de forskellige aspekter blive anskuet og analyseret.

2.1 Problemanalyse

For at kunne indsætte dataopsamlingen i systemets workflow er det nødvendigt at kende de data, der er til rådighed, samt systemets workflow og de processer og arbejdsgange der indgår der i.

For at få kendskab til hvilke data der er til rådighed, og hvor brugbare de er, er dels undersøgt dokumentationen af platformen og dels udført en række eksperimenter. Alle gængse sensorer er blevet evalueret i forhold til, hvilke informationer de kan bringe til systemet, og i hvilken grad informationerne er anvendelige for systemet. Dette vil blive uddybet i afsnit 2.1.1 Sensorer.

For at have overblik over hvilke processer og arbejdsgange der er i systemets workflow, og hvilke informationer de har brug for, er der foretaget en analyse af systemet på baggrund af dets nuværende anvendelse. Det er blevet vurderet hvilke processer, der essentielt er behov for, for at kunne opfylde de brugssituationer systemet indgår i, og derudfra hvilke informationer der skal til at opfylde kravene for disse processer. Dette vil blive uddybet i afsnit 2.1.2 Processer.

I det efterfølgende er der taget udgangspunkt i nyere smartphones baseret på Android. Nærmere bestemt er der taget udgangspunkt i enheder baseret på version 2.2 (Kodenavn Froyo, API Level 8) af Android, da denne officielt repræsenterer langt over halvdelen af alle aktive enheder[2]. En smartphone der lever op til dette, vil i det efterfølgende blot blive omtalt som en *enhed*.

Ifølge API-dokumentationen for Android bør de relevante dele af enhederne og styresystemet dog være gældende for alle versioner, fra version 1.5 (Kodenavn Cupcake, API Level 3) til version 3.2 (Kodenavn Honeycomb, API Level 12) der i skrivende stund er

[2] http://developer.android.com/resources/dashboard/platform-versions.html

den nyeste. Der er heller intet der tyder på at fremtidige versioner ikke vil være bagudkompatible på dette punkt.

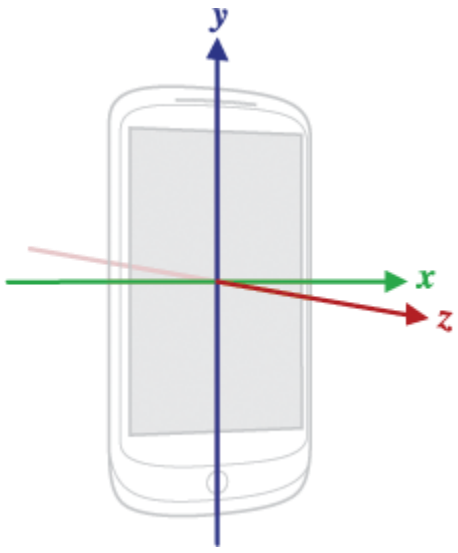
2.1.1 Sensorer

Alle enheder har adgang til en række sensorer. Disse sensorer kan groft opdeles i to typer, henholdsvis primitive og avancerede sensorer.

2.1.1.1 Primitive sensorer

En primitiv sensor er i denne sammenhæng en sensor, der formidler informationer til styresystemet om en enkelt udefrakommende påvirkning af enheden. De er tilknyttet en forholdsvis simpel hardwarekomponent, og data fra komponenten behøver ikke nogen videre behandling, inden de bliver stillet til rådighed for applikationer.

For de sensorer der fortæller noget om enhedens position eller bevægelse i rummet, er koordinatsystemet defineret fast i forhold til enhedens skærm, som vist på Figur 1. X-aksen er vandret og peger til højre, Y-aksen er lodret og peger op og Z-aksen peger ud gennem forsiden af skærmen. I dette system har koordinater bag skærmen negative Z-værdier[3].

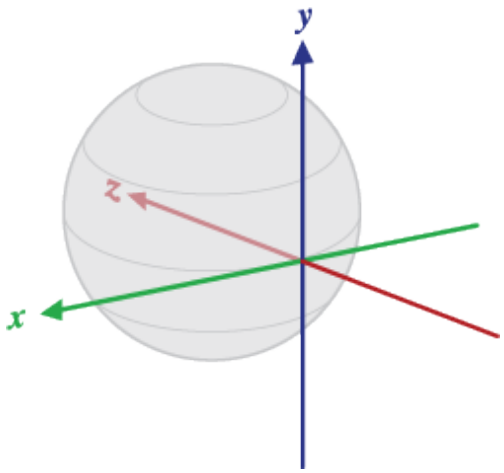


Figur 1: Koordinatsystemet defineret i forhold til enheden

For sensoren der måler enhedens orientering i forhold til jordens overflade, er koordinatsystemet defineret i forhold til det punkt på jorden, hvor enheden befinder sig, som vist på Figur 2. X-aksen er defineret som vektorproduktet af Y og Z, og dermed er den tangent til jorden på enhedens aktuelle placering og peger mod vest. Y-aksen er tangent til

[3] http://developer.android.com/reference/android/hardware/SensorEvent.html

jorden på enhedens aktuelle placering, og peger mod den magnetiske nordpol. Z-aksen peger mod Jordens centrum, og er vinkelret på jordens overflade[4]. Denne information kan bruges til at relatere enhedens relative position og bevægelse til en faktisk position og bevægelse i forhold til jordens overflade.



Figur 2: Koordinatsystemet defineret i forhold til jorden

Følgende primitive sensorer[5] og værdier[6] er understøttet af alle enheder.

- **Accelerometer**

  Måler accelerationen på enheden fordelt på henholdsvis x, y og z-aksen i m/s^2.

  Tyngdekraften påvirker altid den målte acceleration. Af denne grund vil accelerometeret, når enheden ligger på et bord og naturligvis ikke accelererer, vise en acceleration på 9,81 m/s^2 i opadgående retning.
- **Gyroskop**

  Måler hastigheden af enhedens rotation omkring x, y og z-akserne i radianer/sekund.
- **Magnetfelt**

  Måler magnetfeltet omkring enheden på x, y og z-akserne i mikrotesla (µT).
- **Lys**

  Måler lysniveauet omkring enheden i lux-enheder.
- **Orientering**

  Måler enhedens rotation, i forhold til jorden, omkring x, y og z-akserne i grader.

[4] http://developer.android.com/reference/android/hardware/SensorManager.html#getOrientation(float[], float[])
[5] http://developer.android.com/reference/android/hardware/Sensor.html
[6] http://developer.android.com/reference/android/hardware/SensorEvent.html#values

Rotationen omkring z-aksen viser, som et kompas, vinklen mellem y-aksen (magnetiske nordpol) og og enhedens relative y-akse.

- **Tryk**
  Måler styrken af det tryk der påføres enhedens berøringsfølsomme skærm på en arbitrær skala fra 0,0 til 1,0.
- **Afstand**
  Måler afstanden fra enheden til nærmeste objekt foran skærmen i centimeter.
  Visse sensorer giver kun afstanden i binær "nær eller fjern" form.
- **Temperatur**
  Måler temperaturen omkring enheden i grader Celsius.

2.1.1.2 Avancerede sensorer

En avanceret sensor er i denne sammenhæng en komponent i enheden, der kan indsamle mere avancerede former for data, som kræver en større behandling, inden de enten bliver stillet til rådighed for applikationer, eller kan sige noget brugbart om enheden position og bevægelse. Fælles for disse sensorer er også at de normalt ikke blive opfattet som sensorer, da de sædvanligvis har en funktionalitet i sig selv, og ikke blot stiller informationer til rådighed for andre.

Følgende avancerede sensorer er understøttet af alle enheder.

- **GPS-modtager**
  Kan modtage signaler fra satellitter og kan på baggrund af disse udregne enhedens position.
- **GSM-radio**
  Kan kommunikere med mobiltelefonnetværket gennem forskellige sendemaster.
- **Wi-Fi-adapter**
  Kan kommunikere med internettet gennem trådløse access-points.
- **Bluetoothradio**
  Kan kommunikere med andre bluetoothenheder.
- **Kamera**
  Kan tage billeder og optage video.
- **Berøringsfølsom skærm**
  Kan vise grafik og modtage brugerinput.
- **Mikrofon**
  Kan optage lydklip.

2.1.1.3 Brugbare sensorer til lokation

Nogle sensorer kan hurtigt sorteres fra, da de ikke giver informationer, der kan bruges til at bestemme noget om enhedens position eller bevægelse. Disse sensorer er *Lys-, Tryk-, Afstands-* og *Temperatursensorerne*.

Resten af sensorerne kan på forskellige måder give værdifulde informationer til systemet. Her er de listet, efter hvilken type data de kan bidrage med til systemet:

- **GPS-modtager**
  Giver GPS-koordinater der fortæller med stor nøjagtighed, hvor enheden befinder sig. Bruger forholdsvis meget strøm når den er tændt.

  Siden den amerikanske begrænsning på præcisionen blev ophævet i år 2000 og igen da det europæiske system *EGNOS* blev sat i drift i år 2009 steg den teoretiske præcision, først fra 100 meter til 10 meter og siden helt ned til 2 meter.

- **GSM-radio, Wi-Fi-adapter og Bluetooth-radio**
  Hvis positionen af deres respektive tilknyttede centraler er kendt, kan grovere informationer om enhedens position udregnes.
  I praksis er det kun sjældent, at Wi-Fi og bluetooth er anvendeligt i Danmark, da der ikke er tilstrækkelig dækning af centraler med kendt position.
  GSM er på den anden side både meget veldækkende, og bruger ingen ekstra strøm på en mobiltelefon ved almindeligt brug, da det er informationer, der kommer gratis med, når telefonen er tændt.
- **Accelerometer, Gyroskop, Magnetfelt og Orientering**
  Fortæller hvordan enheden bevæger sig i forhold til omverden.
- **Kamera og berøringsfølsom skærm**
  Kan tage visuelle og grafisk input og omdanne dem til informationer om, hvor enheden befinder sig, samt modtage verificering af udførte opgaver.
  Kan også bruges til at dokumentere årsager til at leverancer ikke kan fuldføres, for eksempel ved forkert adresse, trafikomlægninger eller på grund af vind og vejr.
- **Mikrofon**
  Kan modtage talekommandoer, der siger hvor enheden befinder sig, eller er på vej hen, samt modtage verificering af udførte opgaver.

2.1.1.4 Test af sensordata

Gennem forskellige forsøg er udvalgte sensorer blevet evalueret i præcision, usikkerhed og tilgængelighed. Disse er udvalgt i forhold til deres anvendelighed for systemet, og hvor repræsentativ de er for deres respektive sensorgruppe.

2.1.1.4.1 GPS-modtager

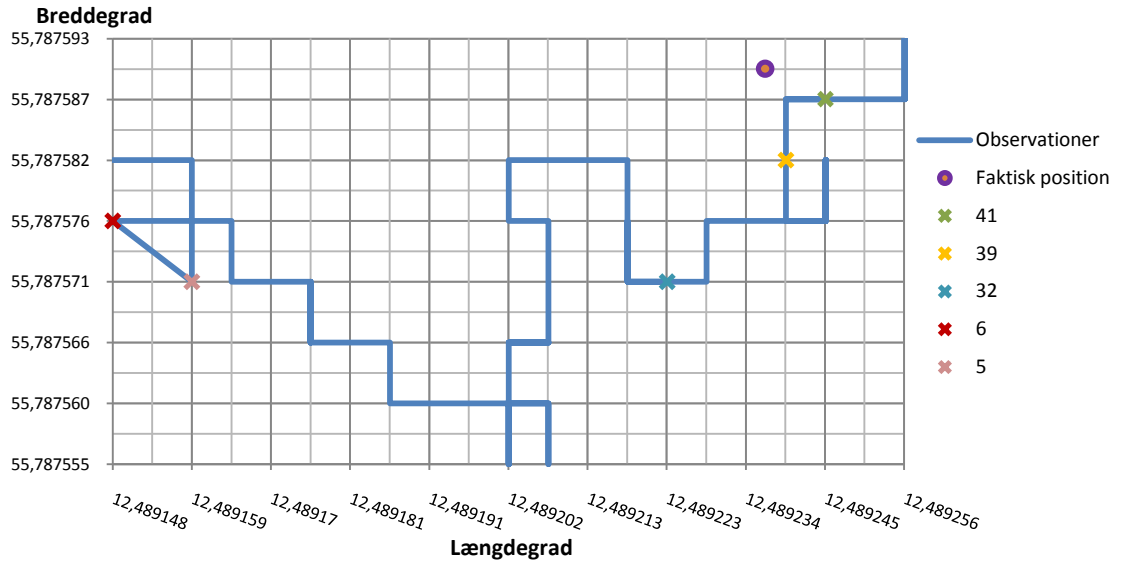
GPS-modtageren er testet i forhold til, hvor præcist den kan give sin position, og hvordan det ændrer sig over tid.

Der er over et antal forsøg opsamlet data over perioder på gennemsnitligt 12 timer, med varierede opsamlingsintervaller på mellem 1 sekund og 10 minutter. For at give et klart billede af hvor konsistente data den producerer, er data opsamlet med en enhed der lå stille under hele perioden. Opsamlingen er fundet sted fra samme position, men på forskellige dage og i forskellige tidsrum i løbet af døgnet.

Disse forsøg viser, at GPS-koordinaterne har en opløsning på under en meter (nord/syd) og under en halv meter (øst/vest) og koordinaterne breder sig over et område på 2,5 meter (nord/syd) og 7 meter (øst/vest). Den længste observerede afstand fra et målt punkt til den faktiske position er 6,2 meter.

Figurerne herunder viser et repræsentativt udsnit på over 40.000 af de observerede koordinater. Figur 3 viser den geografiske placering af koordinaterne og den 'rute' de blev observeret på. Dette viser også, at præcisionen øges over tid, hvilket dog sandsynligvis tilskrives at enheden løbende har fået forbindelse til flere satellitter.

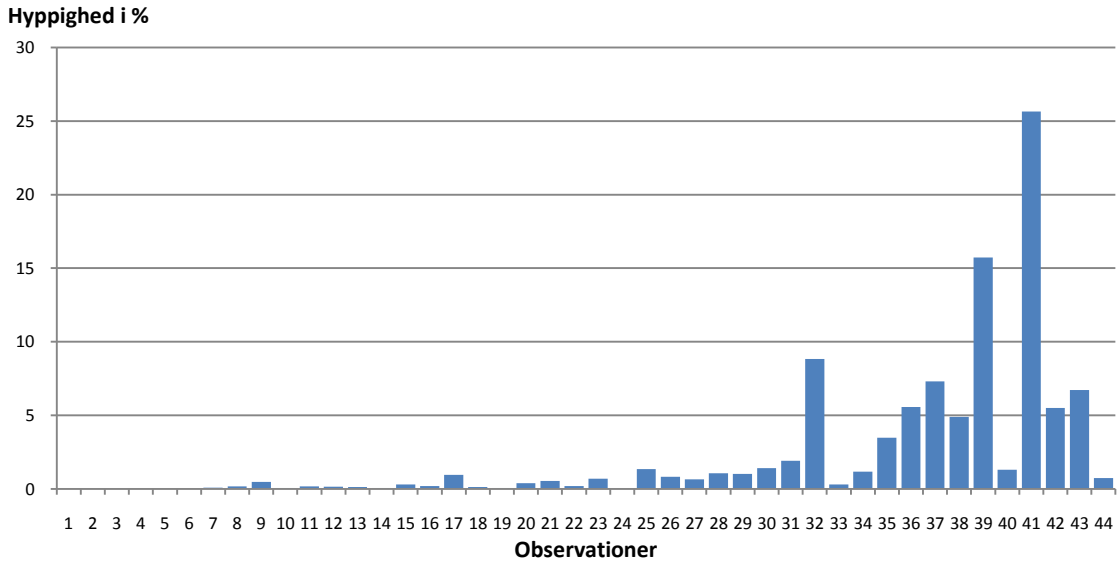
Akserne repræsenterer henholdsvis breddegrad (lodret) og længdegrad (vandret), og hvert 'felt' i grafen er 0,3 meter lodret og 0,33 meter vandret. Større udgaver af Figur 3 og Figur 4 kan findes i Appendiks 6.1.1 Test af GPS-modtager.



Figur 3: Distributionen af GPS-koordinater opsamlet fra en fastliggende enhed

Den faktiske position blev aldrig observeret, hvilket kan skyldes interferens og afskærmning, da enheden var placeret indendørs. I denne sammenhæng er dette dog ikke et problem, da observationerne ligger mindre, end hvad der svarer til to personbil-længder derfra, og det må siges at være en rimelig nøjagtighed i forbindelse med flådestyring.

Figur 4 viser den kvantitative fordeling af koordinaterne. Alle værdier på den vandrette akse stemmer overens med én bestemt observeret position i Figur 3. De er nummereret fortløbende, efter hvornår de er observeret første gang. Den lodrette akse viser den procentvise hyppighed af de enkelte observationer. Fem positioner er markeret for at give en relation mellem de to figurer.



Figur 4: Den kvantitative fordeling af GPS-koordinater opsamlet fra en fastliggende enhed

Som det kan ses, er hyppigheden højere blandt de observationer, der har højere numre, og dermed er observeret senere i forløbet. Disse observationer er også dem, der ligger tættest på den faktiske position, og det viser, at det kun er forholdsvis få (under 13 %) observationer, der ligger længere 2,5 meter fra den faktiske position. Dette er ligeledes, hvad de fem fremhævede punkter hjælper til at give et billede af.

Der er også foretaget forsøg med bevægelse, hvor en kendt rute er fulgt med forskellige transportformer, herunder gåben, cykel, personbil og bus. Observationerne fra disse forsøg er blevet gennemgået manuelt, og den faktiske rute har i alle tilfælde været meget præcist afbildet af observationerne. Nærmere dokumentation for disse forsøg er at finde på den vedlagte CD.

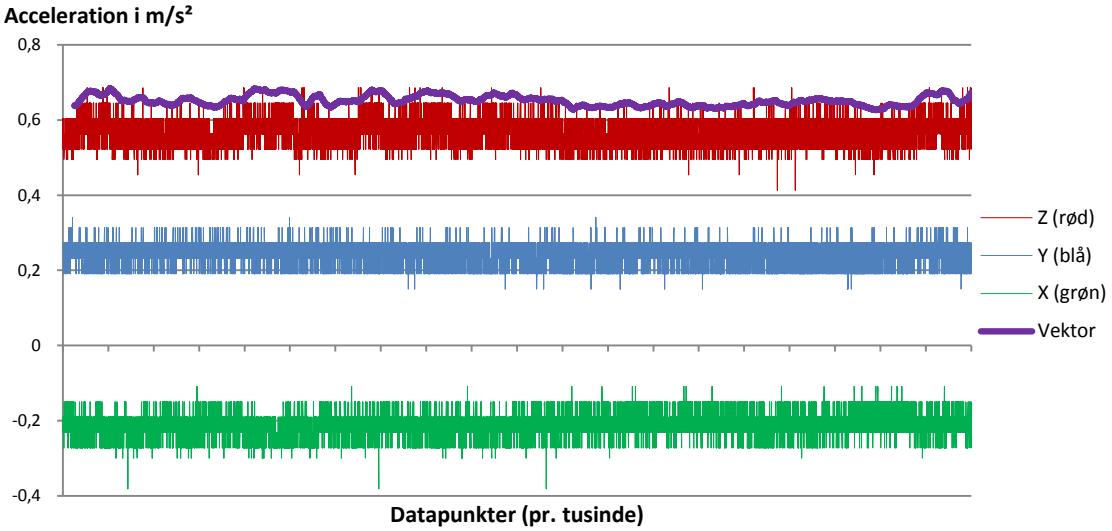
2.1.1.4.2 Accelerometer

Accelerometeret er testet i forhold til, hvor præcist det kan angive enhedens acceleration og hvordan det ændrer sig over tid.

Der er over et antal forsøg opsamlet data over perioder på gennemsnitligt 1 time, med et opsamlingsinterval der svarer til 8 målinger hvert sekund. For at give et klart billede af hvor konsistente data den producerer, er data opsamlet med en enhed der lå stille under hele perioden. Opsamlingen er fundet sted fra samme position, men på forskellige dage og i forskellige tidsrum i løbet af døgnet.

Figuren herunder viser et repræsentativt udsnit på 20.000 af de observerede datapunkter. Forsøgene viser at opløsningen, og dermed også præcisionen, for data på det anvendte accelerometer er 0,040861 m/s^2 og at der er dels et offset og dels en vis støj på målingerne. Støjen kommer til udtryk i en flimren i målingerne på hver af akserne. De tre akser stemmer overens med de akser vist på Figur 1. En større udgave af Figur 5 kan findes i Appendiks 6.1.2 Test af accelerometer.

Fra et teknisk synspunkt er det værd at være opmærksom på, at sensoren giver den totale acceleration der påvirker enheden, inklusiv tyngdeaccelerationen der fordeles ud på de tre respektive akser. For overskueligheden er tyngdeaccelerationen herunder fraregnet.



Figur 5: Målinger af acceleration fra en fastliggende enhed

Offsettet kan, ifølge Figur 5, anses for at være konstant i størrelse og dermed nemt at filtrere fra. Støjen er af størrelsesordenen op til cirka 0,1 m/s^2 på hver akse, hvilket gør den praktisk taget ubetydelig i denne sammenhæng. Dette kan underbygges med en simpel udregning; accelerationen der påvirker en enhed, som bevæger sig omkring hjørnet på en vej.

> Accelerationen A er lig med kvadratet på hastigheden v delt med omdrejningsradiussen r:
>
> $$A = \frac{v^2}{r}$$
>
> Allerede ved en hastighed på 9 km/t (2,5 m/s) omkring et vejhjørne med en radius på 15 meter, påvirkes enheden med en acceleration på
>
> $$A = \frac{v^2}{r} = \frac{(2{,}5\frac{m}{s})^2}{15m} = \frac{6{,}25\frac{m^2}{s^2}}{15m} \cong 0{,}42\frac{m}{s^2}$$

Som det kan ses vil påvirkningen allerede ved denne lave hastighed overstige støjen på alle tre akser tilsammen. Støjen kan også i et vist omfang mindskes gennem simpel databehandling, som for eksempel et løbende gennemsnit over de seneste målinger. Dette vil give en mindre forsinkelse i de opsamlede data, men vil gøre målingerne mere stabile og troværdige.

Ser man på størrelsen af et løbende gennemsnit, taget over de tre aksers resulterende accelerationsvektor[7] ved en periode på 255 punkter (*Vektor* i Figur 5), har denne en gennemsnitlig usikkerhed der ligger helt nede omkring 0,05 m/s^2, hvilket gør støjen endnu mere ubetydelig og muliggør brugen af de opsamlede data til styring af positionering i et flådestyringssystem.

2.1.1.4.2.1 Tidsgrænse for brugbarhed

Ingen af de udførte eksperimenter har vist en nedgang i præcisionen på sensordata over tid. GPS-modtageren, der opsamler absolutte målinger om enhedens position, viste ligefrem forbedring af præcisionen.

Når accelerometeret bruges til at bestemme ændringer i enhedens relative bevægelsesretning, for eksempel som baggrund for opsamling af nye GPS-koordinater, spiller usikkerheden på sensordata, som vist, ingen rolle.

Skal accelerometeret derimod bruges til at kortlægge enhedens bevægelse over længere strækninger, for eksempel i forbindelse med indendørs positionering hvor GPS-modtageren ikke kan anvendes, vil usikkerheden blive akkumuleret over tid, og dermed kan målinger ikke anses som troværdige efter længere perioder. Eksempelvis vil en opsamlingsfrekvens på otte målinger hvert sekund, med en usikkerhed på 0,1 m/s^2, give en akkumuleret usikkerhed på

$$0{,}1\frac{m}{s^2} \times \frac{1}{8}\,s = \frac{1}{80}\frac{m}{s}$$

Det vil betyde, at der allerede efter et minut er en usikkerhed på 0,75 m i forhold til enhedens position. Dette kan forbedres med en højere opsamlingsfrekvens, da usikkerheden er omvendt proportional med opsamlingsfrekvensen.

2.1.1.5 Foreslået brug af sensorer

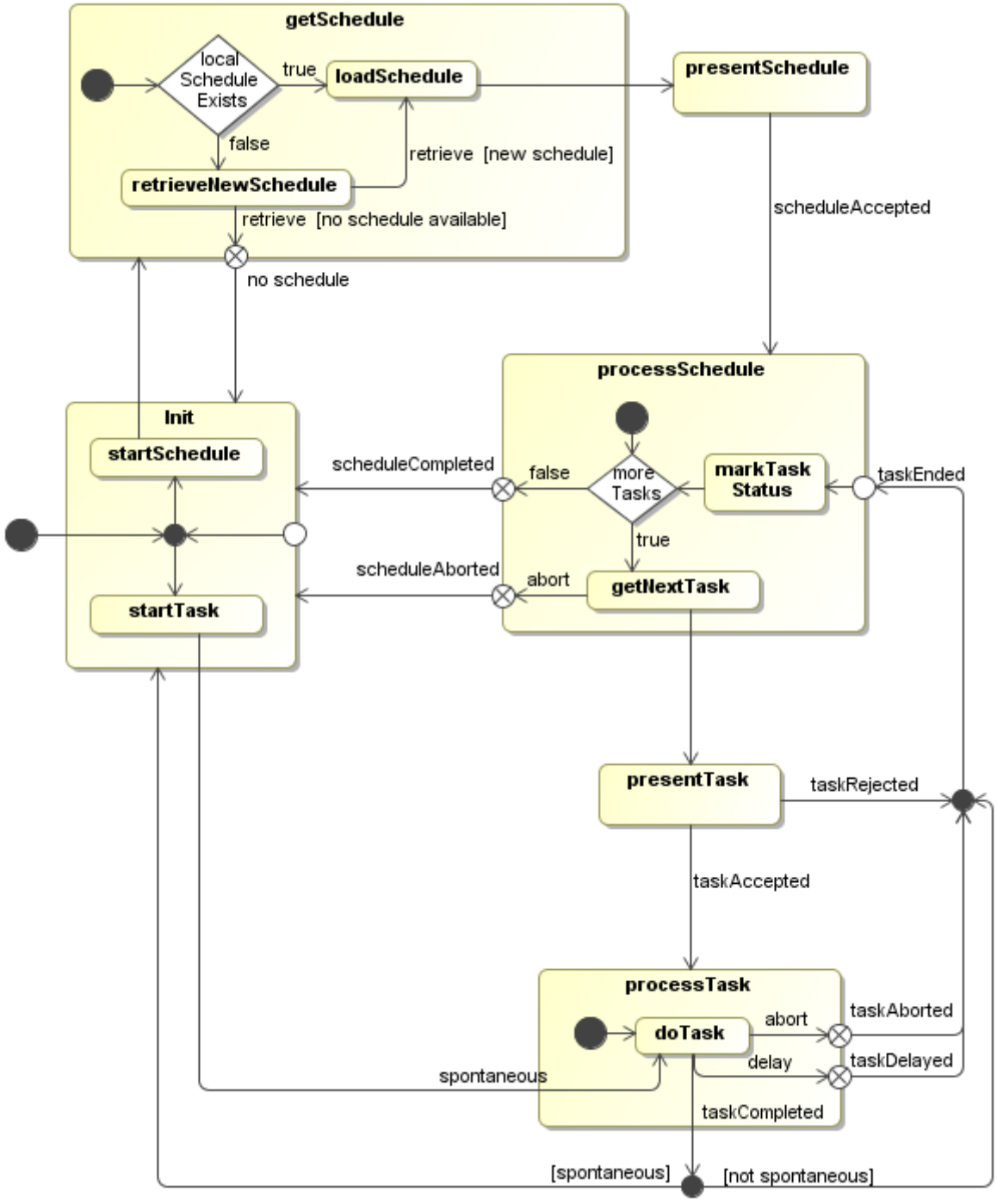
Det kan konkluderes, at accelerometer og GPS er egnede til henholdsvis styring og udførelse af positionering i et flådestyringssystem.

Når det gælder anden brug, som for eksempel indendørs positionering, har de begge svagheder og kan derfor ikke ligge til grund for anvendelser hvor præcision er krævet under disse forhold.

[7] Vektorsummen af de tre aksers accelerationsvektorer

2.1.2 Processer

På baggrund af en analyse af de forskellige sammenhænge systemet indgår i og de dertil hørende logistiske processer, er der udarbejdet en model, for de tilstande systemet antager under brug. Denne model omfatter de arbejdsgange, og muligheder en bruger kan komme ud for.



Figur 6: Tilstandsdiagram over systemets processer

Modellen omfatter ikke opsamlingen af sensordata og positioner, da disse ikke er en del af de logistiske processer, men foregår løbende for at styre og dokumentere udførelsen bedst muligt.

2.2 Kravspecifikation

2.2.1 Overordnet kravspecifikation

Den komplette kravspecifikation kan findes som Appendiks 6.2 Kravspecifikation. Nedenfor er de reelle og væsentligste krav medtaget.

2.2.1.1 Brugergrænseflader

Applikationens brugergrænseflade designes ud fra at brugeren hovedsagligt skal reagere på forespørgsler fra applikationen på baggrund af de data det indhenter fra serveren. Dette holdes så vidt muligt i form af 1 til 6 virtuelle knapper med simple muligheder.

I prototyperne vil der blive lagt minimal vægt på udvikling og udseende af brugergrænsefladen, ligesom brugervenligheden og minimerede svartider på brugerens input ikke er noget der tilstræbes aktivt.

Brugergrænsefladen vil være begrænset til

- At vise informationer der er relevante for det enheden er i gang med, herunder indsamling og behandling af data og kommunikation med serveren
- At tage imod simpelt input udelukkende gennem tryk på virtuelle knapper.

2.2.1.2 Hardwaregrænseflader

Kravene til hardwaregrænseflader på den enhed hvor applikationen skal afvikles, er beskrevet herunder.

2.2.1.2.1 Berøringsfølsom skærm

Da der lægges minimal vægt på brugergrænsefladen vil håndtering af fysiske knapper ikke være en del af prototyperne, så her vil en berøringsfølsom skærm være afgørende for at kunne kommunikere med enheden.

Hvis applikationen bliver afviklet på en enhed, der kan give samme funktionalitet som en berøringsfølsom skærm gennem fysiske knapper, er en berøringsfølsom skærm ikke en nødvendighed. Det kan også være muligt at betjene prototyperne ved brug af fysiske knapper, men det kommer an på enheden, og er ikke noget der tilstræbes.

2.2.1.2.2 GSM-radio

GSM-radioen er hovedsagligt nødvendig til at sørge for kommunikation både til transmittering af data mellem applikationen og serveren, men også i tilfælde hvor eksempelvis SMS eller telefonopkald er nødvendigt for applikationens virke.

Herudover kan informationer om sendemaster og andre data relateret til GSM-nettet bruges som en unøjagtig men hurtig og billig måde, både med hensyn til pris og strømforbrug, at positionere enheden geografisk på en større skala.

2.2.1.2.3 GPS-modtager

GPS-modtageren er en af hjørnestenene i applikationens virkeområde, da det er den mest præcise måde at positionere enheden i stort set alle udendørs situationer. Uden GPS-modtager bliver data mindre præcise end de er i systemet idag.

2.2.1.2.4 Accelerometer

Accelerometeret er en anden hjørnesten i applikationens virkeområde, da det er data fra den sensor den yderligere positionering forbedres ud fra.

2.2.1.3 Softwaregrænseflade

Applikationen skal afvikles under styresystemet Android.

2.2.1.3.1 The Android Operating System

Version: 1.6 eller højere

API niveau: 4 eller højere

Kilde til anskaffelse af softwaren: http://www.android.com

2.2.1.4 Kommunikationsgrænseflade

2.2.1.4.1 Internet over HTTP

Som protokol til kommunikation med serveren benyttes HTTP.

Så længe protokollen er implementeret, så den lever op til standarden, har det ingen indvirken på applikationen præcis hvad der ligger bag, så det lades op til den enkelte enhed hvordan protokollen implementeret og hvilke teknologier der ligger bag.

2.2.1.5 Hukommelsesforbehold

Da mængden af data der skal gemmes lokalt er meget lille i forhold til den hukommelse der er til rådighed i enheder der lever op til ovenstående krav, er der effektivt ingen forbehold.

2.2.1.6 Produktfunktioner

2.2.1.6.1 Brugerhåndtering

Sikrer at applikationen betjenes af den korrekte bruger og forbereder systemet på at modtage data tilknyttet den pågældende bruger.

2.2.1.6.2 Indhente køreplan

Henter en liste over lokationer brugeren skal besøge.

2.2.1.6.3 Udføre køreplan

Håndterer en køreplan og de dertilhørende lokationer.

2.2.1.6.4 Styre den logistiske proces på baggrund af køreplan og position

Ændrer tilstanden af systemet og opdaterer den igangværende logistiske proces i forhold til den aktuelle opgave fra køreplanen og enhedens position.

2.2.1.6.5 Styre positionshåndtering i forhold til proces og lokation

Styrer indhentningen og håndteringen af positioner i forhold til den proces der er under afvikling og den eller de relevante lokationer.

2.2.1.6.6 Behandle positioner

Behandler de indhentede oplysninger om position så de giver værdi i systemet. Behandlingen kan variere efter proces og tilstand i systemet.

2.2.1.6.7 Kommunikere med serveren

Sender informationer om proces, tilstand og position til og modtager informationer og instruktioner fra serveren.

2.2.1.6.8 Individuelle indstillinger

Indstiller enheder til at passe bedst muligt til forskellige brugere og processer.

2.2.1.6.9 Oversættelse af brugergrænsefladen

Gør det muligt at oversætte brugergrænsefladen til forskellige sprog og ontologier.

2.2.1.7 Antagelser og afhængigheder

2.2.1.7.1 Software

Systemet er afhængigt af tilgængeligheden af Android, i den angivne version. Alle nødvendige eksterne funktioner er en del af Android.

Det kan rimeligvis antages at ingen relevante funktioner bliver fjernet fra styresystemet medmindre et bedre alternativ bliver stillet til rådighed.

2.2.1.7.2 Sensorer

Systemet er afhængigt af de angivne sensorer. Uden disse eller tilsvarende kan applikationens funktionalitet ikke opretholdes.

2.2.1.8 Fordeling af krav

Krav til brugerhåndtering, herunder verificering og initiering, kan udelades fra applikationen indtil funktionaliteten dækker tidligere klienter i systemet.

Krav til oversættelse af brugergrænsefladen kan ligeledes udelades fra applikationen indtil funktionaliteten dækker tidligere klienter i systemet.

2.2.2 Specifikke krav

2.2.2.1 Funktionelle krav

2.2.2.1.1 Bruger

2.2.2.1.1.1 Start af applikationen (UA-1)

Brugeren skal kunne starte applikationen fra enhedens brugergrænseflade. Kan applikationen ikke startes bliver brugeren bedt om at kontakte supportafdelingen eller enheden bliver fejlmeldt.

2.2.2.1.1.2 Autentificering af en bruger (UA-2)

Brugeren skal kunne autentificere sig overfor serveren.

2.2.2.1.1.3 Kun autoriseret brug af systemet (UA-3)

Brugeren skal autentificere sig overfor serveren for at bruge applikationen.

2.2.2.1.1.4 Initiere en bruger (UA-4)

Brugeren skal modtage sin personlige opsætning af applikationen fra serveren.

2.2.2.1.1.5 Brugeren stopper applikationen (UA-5)

Brugeren skal kunne stoppe applikationen, der sender den aktuelle opsætning til serveren samt information om brugerens igangværende opgave og køreplan til serveren.

2.2.2.1.1.6 Godkend køreplan (UA-6)

Brugeren skal godkende den køreplan serveren har returneret, for at gå i gang med udførslen af den.

2.2.2.1.1.7 Godkend opgave (UA-7)

Brugeren skal godkende den opgave applikationen er nået til i udførslen af køreplanen, for at gå i gang med udførslen af den.

2.2.2.1.1.8 Afslut opgave (UA-8)

Brugeren skal kunne afslutte den igangværende opgave.

2.2.2.1.1.9 Udsæt opgave (UA-9)

Brugeren skal kunne udsætte den igangværende opgave.

2.2.2.1.1.10 Opret observation (UA-10)

Brugeren skal kunne lave en observation på sin aktuelle position.

2.2.2.1.1.11 Beskriv observation (UA-11)

Brugeren skal kunne tilknytte en beskrivende tekst til en observation.

2.2.2.1.1.12 Anmod om assistance (UA-12)

Brugeren skal kunne anmode om assistance på sin aktuelle position.

2.2.2.1.2 Applikationen

2.2.2.1.2.1 Applikationen initialiseres (AA-1)

Applikationen skal gøre klar til at blive brugt og opsætte forbindelse til og klargøre de aktuelle sensorer.

2.2.2.1.2.2 Verificer bruger (AA-2)

Applikationen skal sende brugerens autentifikationen til serveren, der verificerer brugeren og returnerer brugerens opsætning til applikationen.

Hvis brugeren ikke er autoriseret til at bruge pågældende enhed eller applikation, eller har brugt forkert brugernavn og/eller kode skal applikationen vende tilbage til autentifikationsprocessen og viser information om problemet.

2.2.2.1.2.3 Klargør til bruger (AA-3)

Applikationen skal indlæse en eventuelt returneret opsætning fra serveren, når brugeren er autentificeret.

2.2.2.1.2.4 Indhent køreplan (AA-4)

Applikationen skal indhente den relevante, aktuelle køreplan for brugeren fra serveren.

2.2.2.1.2.5 Præsenter køreplan (AA-5)

Applikationen skal vise brugeren informationer om køreplanen og anmode brugeren om at godkende den.

2.2.2.1.2.6 Afvis køreplan (AA-6)

Godkender brugeren ikke køreplanen skal applikationen informere serveren om, at brugeren har afvist at påbegynde den pågældende køreplan og anmode serveren om en ny.

2.2.2.1.2.7 Start køreplan (AA-7)

Applikationen skal påbegynde afviklingen af køreplanen.

2.2.2.1.2.8 Stop køreplan (AA-8)

Applikationen skal stoppe afviklingen af køreplanen og informere serveren om at afviklingen af den pågældende køreplan er afbrudt.

2.2.2.1.2.9 Præsenter opgave (AA-9)

Applikationen skal vise brugeren informationer om næste opgave i køreplanen og anmode brugeren om at godkende den.

2.2.2.1.2.10 Afvis opgave (AA-10)

Godkender brugeren ikke opgaven skal applikationen informere serveren om, at brugeren har afvist at påbegynde den pågældende opgave og præsentere den næste opgave i køreplanen.

2.2.2.1.2.11 Udfør opgave (AA-11)

Applikationen skal registrere hvor enheden befinder sig i forhold til betingelserne for den igangværende opgave. Brugeren skal løbende opdateres med relevante informationer om enhedens position i forhold til destinationen. Positioner der bliver bedømt som værdiskabende bliver indrapporteret til serveren.

2.2.2.1.2.12 Afslut opgave (AA-12)

Applikationen skal informere serveren om, at den igangværende opgave er afsluttet. Udførelse af næste opgave i køreplanen påbegyndes.

Har brugeren valgt at udsætte opgaven, informeres serveren om dette.

Kan applikationen gennem input fra brugeren og sensorer fastslå at destinationen for opgaven er nået informeres serveren om at opgaven er fuldført.

2.2.2.1.2.13 Verificer afsluttet opgave (AA-13)

Applikationen skal gennem et skærmbillede muliggøre validering, for eksempel i form af stregkodeskanning og underskrift fra modtager.

2.3 Delkonklusion

I dette kapitel er mulighederne der er i at udvide systemets interaktion med omverden blevet analyseret og beskrevet. Det er blevet vurderet hvad der skal til for at kunne afgøre hvilke interaktioner der er interessante at bruge og i hvilket omfang de kan bruges.

Alle de sensorer der er til rådighed er blevet gennemgået med henblik på hvad de kan bidrage til systemet med, og de der kan bruges til at bringe noget nyt til systemet er blevet analyseret dybere i forhold til hvordan de kan udnyttes i systemet og hvilke funktioner de vil kunne medbringe.

Den sensor fra hver sensorgruppe, der blev vurderet til at kunne bibringe mest mulig information i sig selv, henholdsvis accelerometer og GPS, er blevet analyseret og testet i dybden hvad angår de data de kan producere og hvor troværdige de er.

Systemets indre workflow er blevet analyseret gennem brugseksempler for de arbejdsgange der er en del af det nuværende system. På baggrund af denne analyse er der udarbejdet et tilstandsdiagram over de tilstande systemet antager under afvikling af realistiske scenarier over nuværende brug af systemet.

Disse scenarier, diagrammer og analyser har ligget til grund for den udarbejdede kravspecifikation, hvorfra de reelle krav er medtaget herover. Kravene er så igen verificeret mod de dokumenter de er bygget over, og gennem interviews med firmaet, for at fastslå om der er arbejdsgange som kravene ikke omfatter.

3 Design

Ud fra analysen af hvilke sensorer der er til rådighed for enhederne og hvilke processer der indgår i det eksisterende system, vil det foreslåede design til applikationen blive gennemgået. Designet er så vidt muligt holdt uafhængigt af platformen, men da Android er valgt til implementering, er visse mekanismer og designbeslutninger henvendt direkte til denne platform. Det betyder dog ikke, at designet ikke kan ligge til grund for en implementering på en anden platform, men blot at det vil kræve en tilpasning til en sådan.

3.1 Arkitektur

Udgangspunktet i designet er en modulbaseret struktur der gør systemet fleksibelt til forskellige formål og anvendelsesområder. Tilsammen udgør disse moduler et API til systemet, som kan håndtere alle de formål og anvendelser systemet aktuelt har.

Hvert modul er designet til at være uafhængigt af de andre moduler og varetager en bestemt type funktioner i systemet. Modulerne har veldokumenterede interfaces til de andre nødvendige moduler så det er muligt at udvide systemet med nye moduler og udskifte enkelte moduler, hvis et nyt brugsområde opstår eller den bagvedliggende teknologi bliver ændret.

Hovedfokus er at hvert modul giver alle de muligheder til systemet der er brug for uden at have unødvendige eller redundante funktioner. Dette er opnået ved at basere designet direkte på analyserne fra foregående kapitel, hvoraf det fremgår hvilke moduler og funktioner systemet vil have brug for.

Systemet er designet i forhold til Android og derfor kan visse dele af designet være specifikke for arkitekturen i dette. De største punkter, hvor dette er tilfældet er brugen af *Services* i systemet og måden hvorpå informationer fra sensorerne bliver stillet til rådighed.

En Android Service er *"en applikationskomponent, der repræsenterer enten en applikations ønske om at udføre en længere kørende operation, mens der ikke interageres med brugeren eller at levere funktionalitet som andre programmer kan anvende"*[8]. Dette er derfor taget med i de designmæssige overvejelser af systemets arkitektur, da flere af systemets

[8] http://developer.android.com/reference/android/app/Service.html

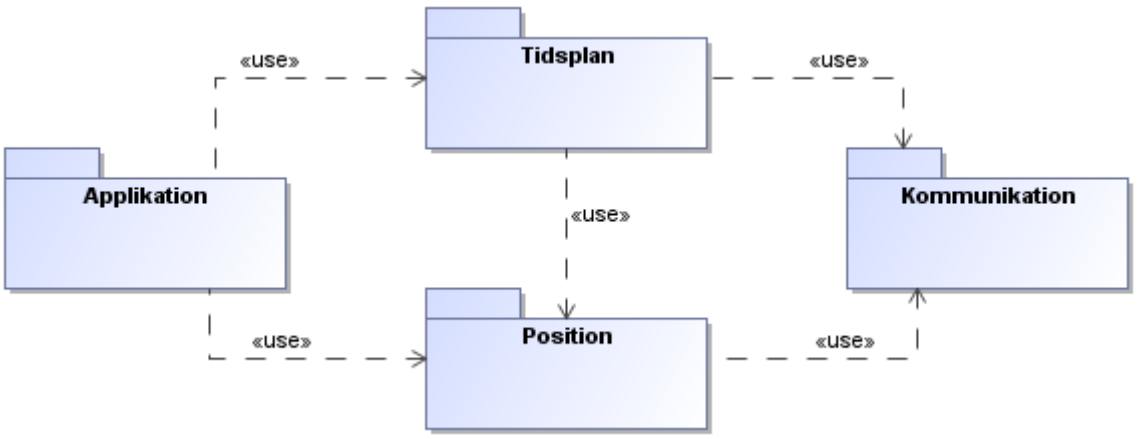
moduler skal kunne arbejde autonomt gennem længere perioder. Derudover har en Service flere gode egenskaber som systemet kan gøre god brug af, hvilket vil blive behandlet senere.

Den måde Android sætter sensordata til rådighed for applikationer gør at der ikke er fuld styring over hvornår data er tilgængelige. For de primitive sensorer og lokationssensorerne gælder det, at man beder om at modtage nye data fra den pågældende sensor med et fast interval og styresystemet så bruger det som en rettesnor for hvornår data skal være til rådighed. På denne måde kan man forvente at data først er klar efter en længere periode end ønsket, men også at de kommer før end ventet. Dette vil også blive diskuteret senere, men i store træk er usikkerheden så lille i forhold til hvor ofte og hvordan sensordata skal bruges, at det ikke spiller den store rolle.

3.1.1 Pakkestruktur

Gennem analysen kan det ses, at der er fire hoveddele i systemet. Disse hoveddele er applikationens moduler der har deres eget afgrænsede funktionsområde.

Hvert modul i systemet er manifesteret som en *Java Package*. Denne pakkestruktur er valgt for at adskille alle modulerne på en logisk og funktionalitetsbaseret måde, der samler alle de komponenter der har med samme funktionsområde at gøre. På denne måde bliver applikationen mere overskuelig fra en udviklers synspunkt og det er mere simpelt at skifte et modul ud med et nyt eller genbruge et modul i en anden applikation, hvis de tekniske specifikationer skulle ændre sig.



Figur 7: Pakkestruktur i systemet

De fire pakker vist i Figur 7 herover udgør den overordnede struktur i systemet. De reflekterer de fire forskellige grundlæggende opgaver systemet skal tage hånd om. Herunder beskrives pakkerne og deres funktionsområder.

3.1.1.1 Position

Pakken *Position* indeholder alt der har med positioner at gøre i systemet.

Pakken varetager tre overordnede funktioner i forhold til resten af systemet, henholdsvis opsamling, behandling og repræsentation af data i systemet.

Positions- og bevægelsesdata fra de forskellige sensorer der er til rådighed i enheden opsamles løbende i alle processer der skal bruge dem. Behandlingen af de opsamlede data sker på baggrund af andre tilgængelige informationer i systemet, herunder informationer om den aktuelle proces og tidligere opsamlede data der er gemt lokalt på enheden. Repræsentationen af de opsamlede data overfor resten af systemet, skal finde sted på en brugbar og let tilgængelig måde. Dette opnås gennem en samling af funktioner til at udføre de nødvendige beregninger og og manipulationer af data.

3.1.1.2 Tidsplan

Pakken *Tidsplan* indeholder alt der har med tidsplaner og opgaver at gøre i systemet.

Pakkens funktioner er at udføre og vedligeholde en tidsplan, herunder gennemløbe tidsplanen og dens opgaver og håndtere deres tilstande, i henhold til Figur 6 i analysen, for dermed at styre de tilknyttede logistiske processer. Hertil kommer funktioner til at modtage opdateringer til tidsplanen og at repræsentere tidsplaner og opgaver overfor resten af systemet.

3.1.1.3 Kommunikation

Pakken *kommunikation* indeholder alt der har med kommunikation at gøre. Det drejer sig hovedsagligt om at transmittere data mellem enheden og systemets centrale server, men det kan også omfatte at modtage instruktioner og andre beskeder fra en operatør. Dette kan foregå over for eksempel internet eller sms.

Pakken omfatter dog også funktioner til at sikre data i tilfælde af kommunikationsnedbrud. Dette omfatter at gemme lokale kopier af modtagne og sendte informationer. Modtagne informationer herunder tidsplaner, skal altid være til rådighed for applikationen og skal derfor gemmes indtil opdaterede informationer er modtaget og

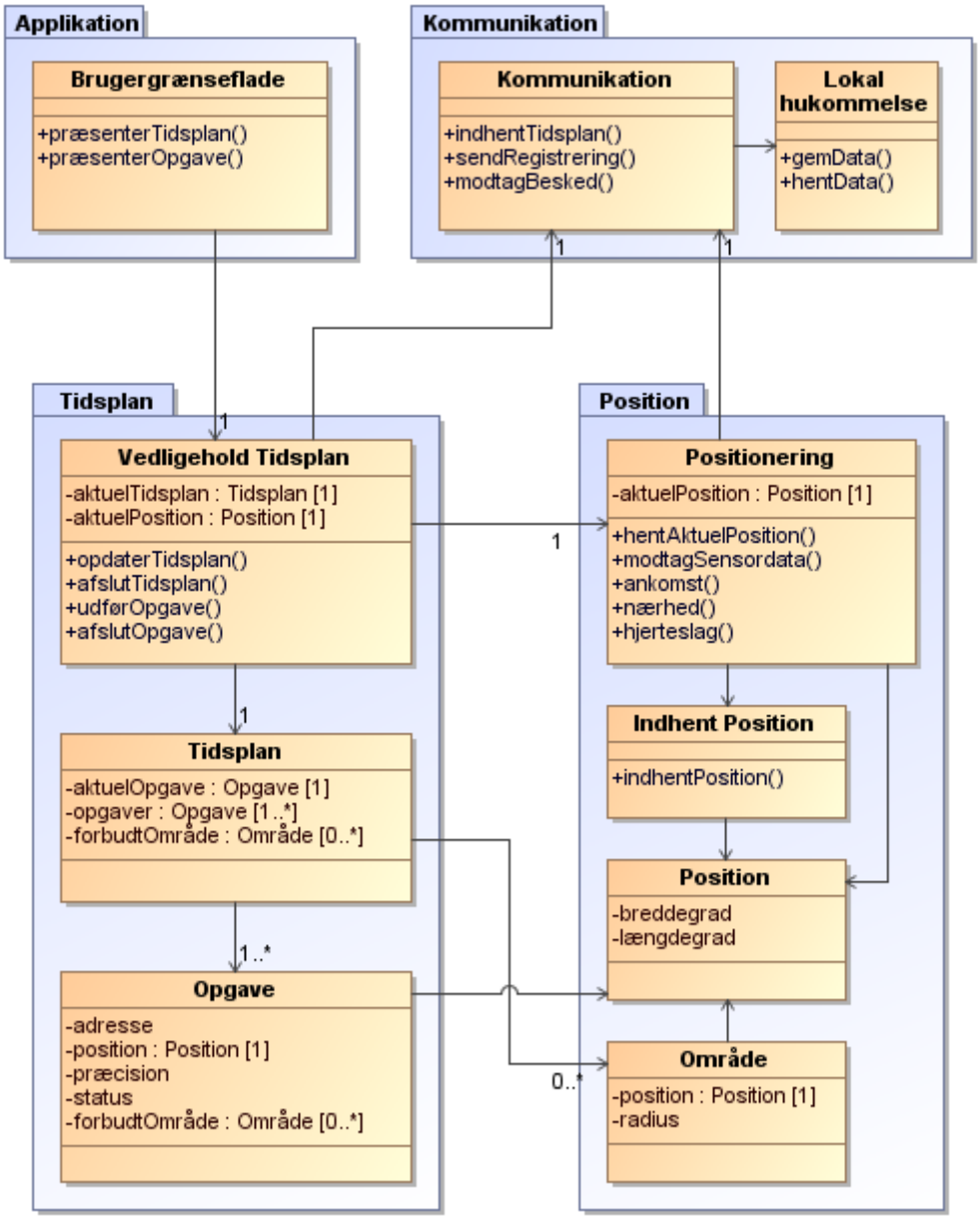
verificeret. Ligeledes skal alle data, der sendes til serveren, gemmes lokalt på enheden indtil der modtages en bekræftelse fra serveren på at data er modtaget. De gemte data bruges også i forbindelse med behandlingen af nye data.

3.1.1.4 Applikation

Pakken *Applikation* indeholder brugergrænsefladen til applikationen, der viser informationer til brugeren og håndterer input fra brugeren til systemet. Den er også ansvarlig for at sætte de andre moduler i gang og initialisere disse.

3.1.2 Klassediagram

Hvert modul i systemet er opbygget af et antal komponenter der kan udføre de relevante funktioner for det pågældende modul. Hver komponent er manifesteret som en *Java Class* og deres overordnede infrastruktur er som vist i Figur 8 herunder.



Figur 8: Klassediagram for systemet

3.2 Design patterns

I designet af systemet er der gjort brug af adskillige *design patterns*. Det mest oplagte er den indirekte brug af *singleton* og *Model-View-Controller*, men en del andre er også benyttet. Disse er beskrevet herunder.

3.2.1 Architectural patterns

Architectural patterns er en type design pattern der retter sig mod opbygningen af selve systemet og ikke blot dele af funktionaliteten.

3.2.1.1 Model–View–Controller

Dette pattern sørger for adskillelsen af indhold, præsentation og funktionalitet. Ved at adskille de tre dele bliver en applikation langt mere overskuelig og det er nemmere at bygge videre på eller udskifte dele af applikationen.

Under Android er det nærmest så indbygget i måden applikationer opbygges, at det er en præstation ikke at følge det. I denne applikation er der således en enkelt klasse der tager sig af grænsefladen til brugeren og hvordan informationer skal formidles. Der er nogle klasser der tager sig af hver deres funktionelle logik og en der overordnet styrer de andres afvikling. Derudover er der et antal klasser der udelukkende tager sig af simpel repræsentation af data.

3.2.2 Creational patterns

Creational patterns er en type design pattern der styrer eller hjælper i forbindelse med at oprette objekter i et program.

3.2.2.1 Singleton

I analysen og designet af applikationen blev det tidligt klart, at der ville være mange fordele ved at anvende services i centrale dele af funktionaliteten. Brugen af disse fører implicit brugen af singletons. En service bliver startet første gang den bliver efterspurgt af en applikation eller anden programkomponent og bliver efterfølgende refereret når den bliver efterspurgt. Den ophører når der ikke længere er nogen anden programkomponent forbundet med den og den ikke udfører nogen funktion, hvorefter den forsvinder fra styresystemets hukommelse når der er brug for den til noget andet.

På denne måde vil der altid kun eksistere én instans af den pågældende service og alle der referer til den, vil arbejde med den samme instans.

3.2.3 Structural patterns

Structural patterns er en type design pattern der letter designet af systemet ved at simplificere forbindelsen mellem visse dele af systemet.

3.2.3.1 Facade

I systemet indgår flere avancerede moduler og af hensyn til både simplificering af systemet og konsistens internt i de enkelte moduler, er det valgt at opretholde en klar opdeling af funktionalitet, der kan tilgås gennem få og relativt simple kommandoer mellem modulerne. Dette kan også ses og anvendes som et API til videre udvilking og anden brug af de enkelte moduler.

Som et enkelt eksempel kan pakken kommunikation (se Figur 8 herover) nævnes, da den modtager simple kommandoer fra resten af systemet og selv styrer hvordan det efterfølgende skal udføres. For eksempel kan kommandoen "sendRegistrering" sende en registrering til en server, men oprettelsen af en forbindelse til serveren og detaljer om hvordan det gøres er skjult for resten af programmet. Ligeledes er det også kun internt i pakken man kan se, at kommandoen også sørger for at gemme de sendte registreringer lokalt, indtil de er kommet en bekræftelse fra serveren på, at de er modtaget.

3.2.4 Behavioral patterns

Behavioral patterns er en type design pattern der letter, simplificerer eller ændrer på den måde objekter opfører sig i systemet.

3.2.4.1 Iterator

Dette pattern kan gøre det muligt at gennemløbe samlinger af objekter der ikke normalt understøtter denne funktionalitet. Dette kan med fordel benyttes i forbindelse med eksempelvis en tidsplan, der blandt andet består af en liste af opgaver (se Figur 8 herover).

Da Android implementerer det meste af Java programmeringssproget, er det oplagt og lige til at benytte dette pattern, da det kun kræver at ens objekt implementerer *Java-interfacet Iterable* og nogle få nødvendige funktioner.

3.2.4.2 Observer

Dette pattern sørger for at et objekt kan registrere sig til at få opdateringer når der sker ændringer i andre objekter. Dette bliver i systemet både brugt til at modtage opdateringer fra sensorer og til at kommunikere mellem services og andre dele af systemet.

3.3 Delkonklusion

Dette kapitel beskriver hvordan den grundlæggende arkitektur i Android-baserede applikationer tager sig ud og hvordan det indvirker på designet og arkitekturen af systemet. Der er flere velkendte designprocedurer der gør sig gældende igennem den brug Android gør af dem. Disse bliver beskrevet blandt de anvendte design patterns, da det er bygget dybt ind i selve platformen.

Der gives et overblik over hvordan applikationen skal skrues sammen, først med et overblik over den pakkestruktur applikationens dele er inddelt i og en beskrivelse af hvad

de enkelte pakker skal tage hånd om, hvilket funktionsområde de varetager og hvilke datarepræsentationer de er ansvarlige for overfor resten af applikationen. Herefter bliver pakkediagrammet udvidet med de konceptuelle klasser der er nødvendige for at fuldbyrde de ansvarsområder der er tildelt de enkelte pakker, og klassediagrammet sætter relationer mellem de forskellige klasser der illustrerer hvordan de indbyrdes interagerer.

Slutteligt er en gennemgang af de design patterns der er anvendt på applikationen, både de der er valgt ud fra egenskaber der vil være systemet til gode og de der er brugt som en del af det framework Android stiller til rådighed. Brugen af disse sikrer en højere grad af konsistens i systemet og bidrager med metoder der er velkendte og udbredte.

4 Implementering

I det foregående kapitel blev systemets grundlæggende struktur designet. De pakker og klasser der skal indgå for at opnå den ønskede funktionalitet er defineret i forhold til deres ansvarsområder og interne funktionalitet.

Den specifikke implementering af systemet er foregået i udviklingsmiljøet Eclipse i version 3.5 (Galileo) med Android SDK senest opdateret til revision 11 (maj 2011). Dette SDK inkluderer blandt andet en fuld version af styresystemet og en VM der kan køre dette som en emulator af en Android-baseret enhed, hvilket gør at emuleringen er så godt som identisk med afvikling på en reel enhed. De eneste umiddelbare forskelle ligger i simulering af hardwarekomponenter, som for eksempel accelerometer og magnetfeltssensor. Andre komponenter, som for eksempel kamera og mikrofon, kan emuleres gennem enheder sluttet til den maskine der udvikles på. GPS kan simuleres gennem et interface hvor man fra udviklingsmiljøet kan sende koordinater til emulatoren.

4.1 Implementeringsovervejelser

4.1.1 Bestemmelse af brugerens lokation

Ifølge den officielle udviklerguide til Android kan *"indhentning af brugerens lokation fra en mobil enhed være kompliceret. Der er flere grunde til, at en aflæsning af lokationen (uanset kilde) kan indeholde fejl og være unøjagtige."*[9]. Guiden fortsætter med at nævne nogle gængse kilder til fejl i brugerens lokation:

- **Mange forskellige kilder til lokation**
  GPS, mobiltelefonmaster, og Wi-Fi kan hver især give et fingerpeg om brugernes placering. Hvilke der skal bruges og stoles på, er et spørgsmål om kompromiser i forhold til nøjagtighed, hastighed, og batteri-effektivitet.
- **Bruger bevægelse**
  Fordi brugerens lokation ændres, må man redegøre for bevægelse ved regelmæssig re-estimering af brugerens lokation.
- **Varierende nøjagtighed**

[9] http://developer.android.com/guide/topics/location/obtaining-user-location.html

Forskellige lokationer fra de forskellige kilder er ikke nødvendigvis af samme nøjagtighed. En lokation indhentet for 10 sekunder siden fra én kilde kan være mere præcis end den nyeste placering fra den samme eller en anden kilde.

Disse og andre fejlkilder er håndteret ved at lade systemet indhente flere forskellige lokationer, på baggrund af en reaktion fra sensorer, der viser en ændring i enhedens bevægelse. Disse lokationer behandles i forhold til hinanden og det vurderes hvorvidt der er flere lokationer der stemmer overens med hinanden. Hvis en lokation kan valideres fra flere kilder eller observationer, sammenlignes den med en forventet lokation baseret på tidligere lokationer. Stemmer dette også overens, bliver den godtaget som en korrekt lokation. Hvis ikke indhentes flere lokationsinformationer til at fortsætte valideringsprocessen.

4.1.1.1 Sensoropdateringer

Under Android er det specificeret, at der ikke stilles nogle garantier omkring præcis hvornår en sensor stiller nye oplysninger til rådighed for applikationer.

Bevægelsessensorerne, og for så vidt også de andre primitive sensorer, stiller opdateringer til rådighed gennem et interface, hvor en applikation registrerer sig til at modtage opdateringer. Ved registreringen skal applikationen angive hvorlænge der skal gå mellem opdateringerne skal modtages. Dette er dog kun en rettesnor for systemet, og dokumentationen skriver da også, at *"Events kan modtages hurtigere eller langsommere end den angivne værdi. Normalt modtages events hurtigere."*[10]. Opdateringshastigheden kan angives i mikrosekunder eller man kan bruge en af fire hastigheder der er defineret på forhånd og som angiveligt er optimeret til henholdsvis ændringer i skærmorienteringen, opdateringer til brugergrænsefladen, sensorbrug i spil og hurtigste mulige opdateringshastighed.

Lokationssensorerne stiller ligeledes opdateringer til rådighed ved, at en applikation registrerer sig til at modtage opdateringer[11]. Ved registreringen kan applikationen angive minimumværdier for hvor mange millisekunder og hvor mange meter der skal være mellem hver opdatering. Det er ikke nærmere specificeret hvor længe man kan risikere, at systemet går over denne tidsgrænse. Det foreslås dog, at baggrundsservices ikke bør sætte

[10] http://developer.android.com/reference/android/hardware/SensorManager.html#registerListener(android.hardware.SensorEventListener, android.hardware.Sensor, int)
[11] http://developer.android.com/reference/android/location/LocationManager.html#requestLocationUpdates(java.lang.String, long, float, android.location.LocationListener)

et minimum for lavt, helst ikke under 60.000 millisekunder, af hensyn til strømforbrug på enheden.

4.1.2 Hvorfor bruge services

Brugen af services i systemet er valgt på grund af disses gode egenskaber i forhold til de funktioner de er valgt til at udføre. Det drejer sig om de to centrale moduler, som tager sig af henholdsvis at gennemløbe tidsplaner og indhente positioner. Disse to moduler har brug for at blive afviklet sideløbende med applikationens brugergrænseflade og har behov for at køre så længe applikationen kører uden dog at afbryde brugergrænsefladen.

Under Android kører applikationer efter en enkelt-tråd model, der foreskriver at selve applikationen og dens brugergrænseflade køres i en enkelt tråd. Dermed er værktøjerne til at tilgå brugergrænsefladen ikke gjort tråd-sikre og al interaktion med brugergrænsefladen bør derfor ske i applikationens hovedtråd[12]. Operationer der tager længere tid bør henvises til arbejdertråde i baggrunden så de ikke låser hovedtråden og får det til at virke som om applikationen ikke svarer.

De to moduler har dog også behov for at kunne kommunikere med andre dele af applikationen, for eksempel for at sende informationer til serveren, få opdateret brugergrænsefladen eller modtage opdaterede data fra et andet modul. Dette kan implementeres ved simple metodekald på tværs af pakker og klasser, hvilket kræver en instantiering af hver klasse der skal kommunikeres med. Dette kan skabe noget rod i synkroniseringen, da flere instantieringer af en klasse kan komme til at sende forskellige informationer til deres respektive ejere.

Problemet med synkroniseringen kan løses ved at gøre brug af et design pattern som Singleton, hvilket sørger for, at der kun er en instans af hver enkelt instantieret klasse i systemet. Et problem ved brugen af singletons kan dog være, at de eksplicit skal oprettes og nedlægges af deres ejere. Hvis applikationens afvikling er pauset for en stund, for eksempel hvis brugeren modtager et telefonopkald på enheden, kan man heller ikke være sikker på, at tilstand og indhold af en singleton er det samme efter som før pausen.

Alle disse behov og problemer løses gennem brugen af services. En service kører i sin egen tråd og er født singleton, da systemet starter servicen første gang en applikation spørger efter den og efterfølgende giver applikationer en reference til den samme instans af

[12] http://developer.android.com/resources/articles/painless-threading.html

servicen. Men da servicen er en separat applikationskomponent, har den også sin egen livscyklus der tager hånd om at starte, pause, genoptage og stoppe servicen. En service stopper ikke så længe en applikationskomponent er forbundet med den og den beholder sin tilstand selvom dens afvikling bliver pauset.

Kommunikation mellem en service og en anden applikationskomponent bliver muliggjort gennem IPC, der gør det muligt at sende data og kommandoer mellem processer. Ved at definere en fælles grænseflade gennem AIDL[13] er både klient og service rustet til at kommunikere frem og tilbage, og selv avancerede datatyper og objekter fra systemet kan sendes over denne protokol.

Både services i sig selv og IPC kræver nogen tid til at sætte sig ind i, og implementere så det virker i første omgang, men det er tid der er givet godt ud og vundet allerede når det bruges anden gang, da det er en meget robust og skalerbar måde at håndtere de ovenstående problemstillinger.

4.1.3 Antagelser i algoritmer

I forbindelse med funktionaliteten af applikationen har der været flere steder hvor der har været grund til at gøre antagelser eller simplificere dele af algoritmer. Dette er hovedsagligt på grund af projektets størrelse i forhold til det eksisterende system og enhedens begrænsninger, herunder regnekraft og batterilevetid.

Det interessante i projektet har ikke været at implementere en kopi af det eksisterende system på en ny platform, men undersøge mulighederne for nye måder at styre den logistiske proces og forbedre opsamlingsprocessen i forhold til enhedens position og bevægelse.

4.1.3.1 Udregning af afstande

Når en afstand mellem to punkter på jordens overflade skal beregnes er der grundlæggende to metoder til at gøre det, se jorden som en meget stor kugle eller som en plan flade.

[13] http://developer.android.com/guide/developing/tools/aidl.html

Det er selvfølgelig mest korrekt og giver mere præcise resultater at antage, at jorden er en meget stor kugle, men det er også meget mere beregningstungt. Den mest udbredte formel til at udregne afstande på overfladen af en kugle, er *Haversine*-formlen[14]:

$$\Delta lon = lon2 - lon1$$
$$\Delta lat = lat2 - lat1$$
$$a = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat1) \times \cos(lat2) \times \sin^2\left(\frac{\Delta lon}{2}\right)$$
$$c = 2 \times \arcsin\left(\min\left(1, \sqrt{a}\right)\right)$$
$$d = R \times c$$

Den finder afstanden mellem de to punkter P1(*lat1*, *lon1*) og P2(*lat2*, *lon2*), hvor *R* er jordens tilnærmede radius. Resultatet gives i variablerne *c* og *d* – i *c* som radianer og i *d* med samme enhed som *R*.

I dette projekt er det dog antaget, at afstandene er af en størrelsesorden hvor usikkerheden er ubetydelig når afstandene udregnes mellem punkter på en flade, efter den pythagoræiske læresætning:

$$a^2 + b^2 = c^2$$

Dermed bliver afstanden, mellem de samme punkter som før, udregnet som

$$d = \sqrt{(lat2 - lat1)^2 + (lon2 - lon1)^2}$$

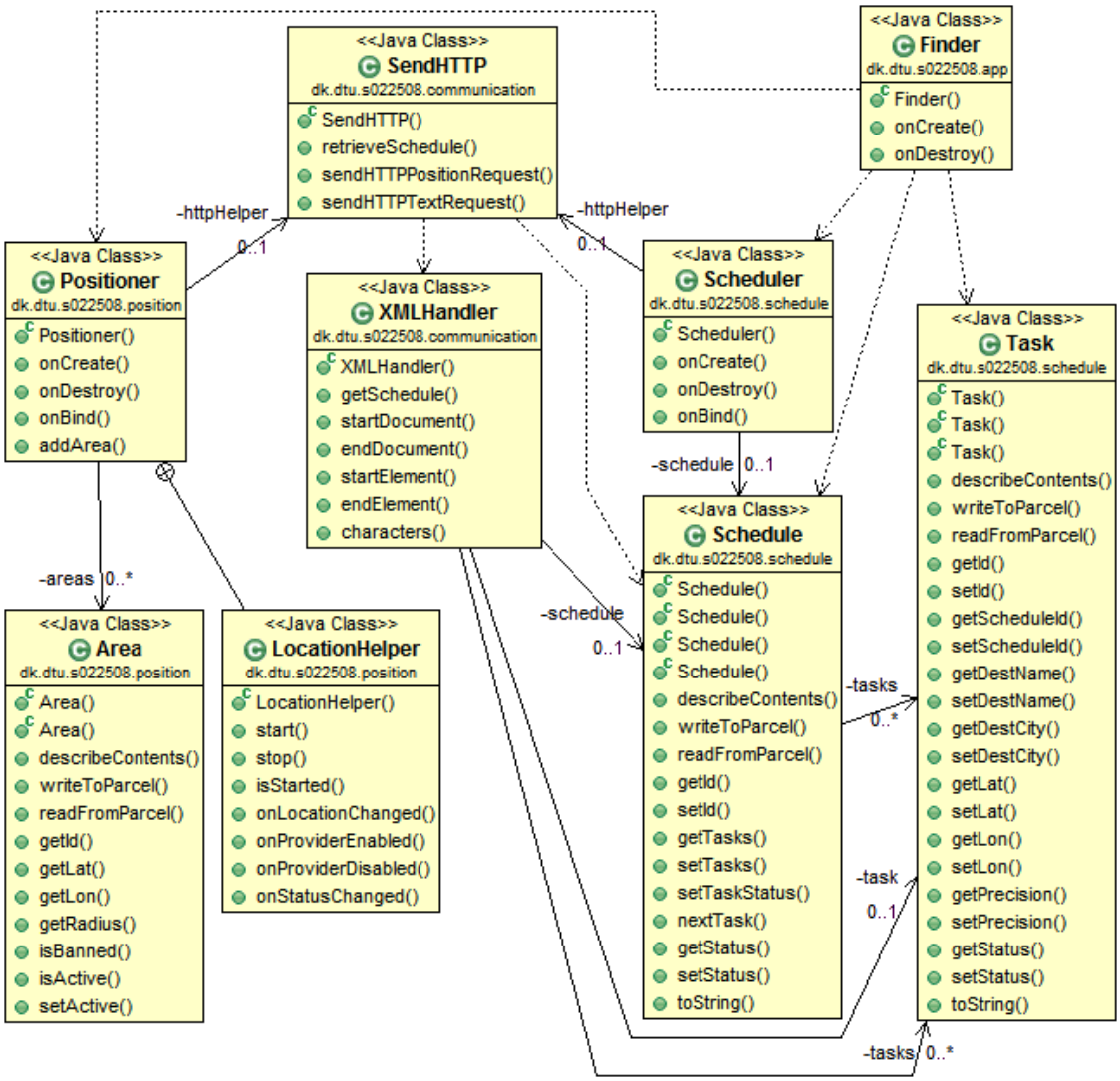
Denne udregning er væsentlig mindre beregningstung og på afstande op til 750 km ses en afvigelse ned til cirka 0,15 %, hvilket svarer til en misvisning på under 2 kilometer på en afstand der svarer til det dobbelte af Jyllands længde, fra Skagen til Flensborg.

I den forbindelse er det også relevant at have et mål for hvornår enheden er ved at nærme sig sin destination. En fast afstand ville selvfølgelig være langt simplest, men en bruger vil opfatte det at nærme sig målet forskelligt om ruten har været 500 meter eller 100 km lang. En anden mulighed ville være, at opsætte en længere formel der giver et mere nuanceret og relativt billede af hvornår ankomsten er nær, men i dette projekt er det meget simpelt sat til at være en afstand på 5 % af den totale afstand fra start til slut.

[14] http://www.faqs.org/faqs/geography/infosystems-faq/ Q5.1: What is the best way to calculate the great circle distance

4.2 Implementering af klasser

Klasserne i Figur 8 tager sig i systemets implementering ud efter følgende diagram.



Figur 9: Klassediagram over implementeringen af systemet

Som det kan ses er der nogle forskelle, da det af implementeringsmæssige hensyn er nødvendigt at have visse specifikke funktioner der er bundet til den aktuelle implementering af systemet og dets protokoller og teknologier. For eksempel kan nævnes brugen af HTTP som kommunikationsprotokol og XML som format til dataoverførsel.

Det kan også noteres, at nogen funktionalitet er lagt ud til hjælper-klasser, eksempelvis konverteringen af XML til de interne datastrukturer og indhentningen af lokationer fra sensorer.

4.3 Delkonklusion

Da implementeringen er bundet til en bestemt platform er der nogle specifikke begrænsninger og muligheder der skal tages højde for. Dette kapitel behandler hvordan platformen adskiller sig fra andre mobile platforme og fremhæver dele af platformen som applikationen kan drage nytte af.

En dybdegående gennemgang beskriver hvordan sensormekanismerne fungerer og hvordan services er ideelle til at varetage kernefunktionerne i systemet. Der er dog også nogle uhensigtsmæssigheder i den måde platformen er designet, men når de er klart beskrevet er det muligt at omgå dem uden, at de er til videre besvær.

Gennem implementeringen er der foretaget en del antagelser for at beskrive systemet på en måde, der på engang er tilnærmet komplet og beregningsmæssigt let nok til at være egnet til afvikling på en mobil enhed i længere perioder ad gangen. Det er beskrevet hvordan, hvorfor og på hvilket grundlag det er besluttet at opfatte jorden som en flade og hvilken kvantitativ indvirken det har på systemet.

5 Konklusion

Igennem dette projekt er det vurderet hvordan et flådestyringssystem kan drage fordel af nyere generationer af smartphones. Analysen af dette er baseret på et specifikt flådestyringssystem der aktuelt er i brug, men resultaterne og de beskrevne metoder henvender sig til alle lignende produkter og viser lovende resultater for fremtidig brug af platformen til at udføre denne slags opgaver med større præcision og bedre interaktion mellem den enkelte bruger og systemet i sin helhed.

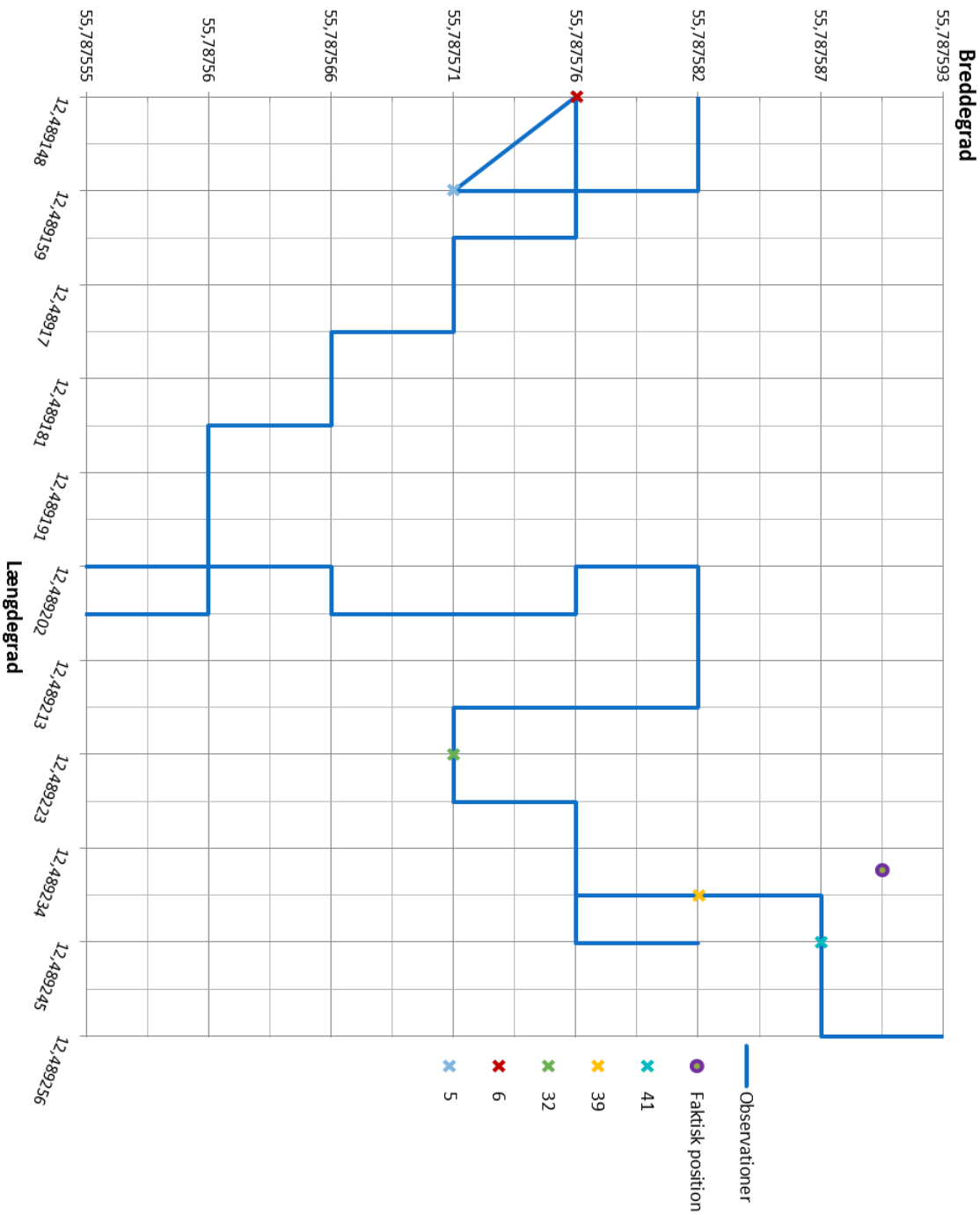
Der er gjort mange studier i hvordan det eksisterende system fungerer og bliver anvendt, og hvordan det vil kunne gøres bedre og mere intuitivt for alle parter. Der er visse fremtrædende bevægelsesmønstre der gør sig gældende for størstedelen af de brugere der er til et system af den art, hvilket kan udnyttes med stor fordel.

Et resultat af arbejdet med raffinering af systemet har været en model for hvordan virksomheder kan spare ressourcer og samtidig få et mere præcist billede af hvordan deres mobile medarbejdere bevæger sig. Dette opnås da der kan skelnes mellem informationer der bliver genereret løbende og informationer der kan knyttes til enkeltstående handlinger fra brugerens side, hvorved der kun er behov for at transmittere en brøkdel af informationerne enheden opsamler. Dette vil give en mærkbar nedgang i dataforbruget på den enkelte enhed og en opgang i sporbarheden af medarbejderens handlinger.

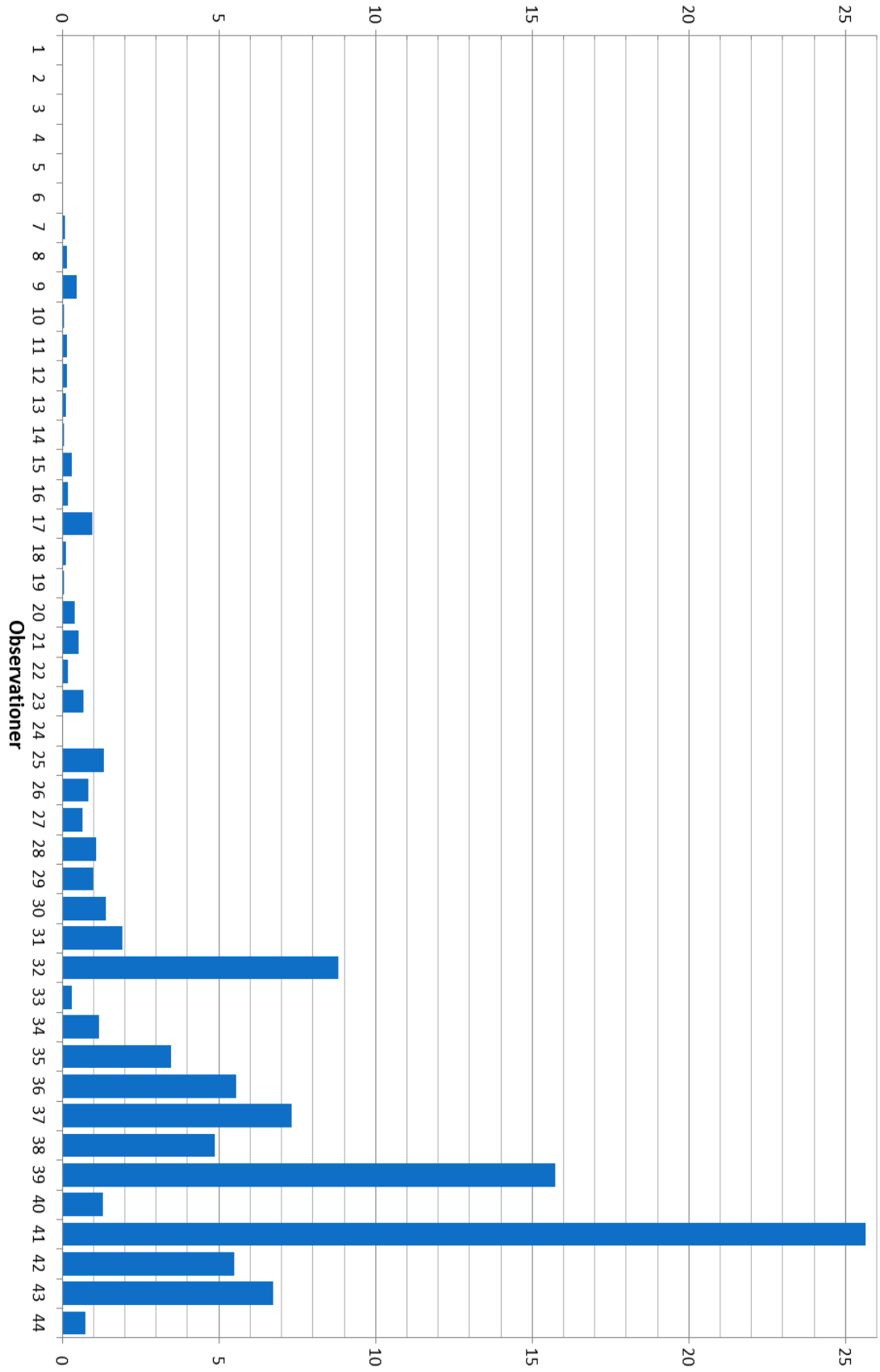
6 Appendiks

6.1 Grafer

6.1.1 Test af GPS-modtager

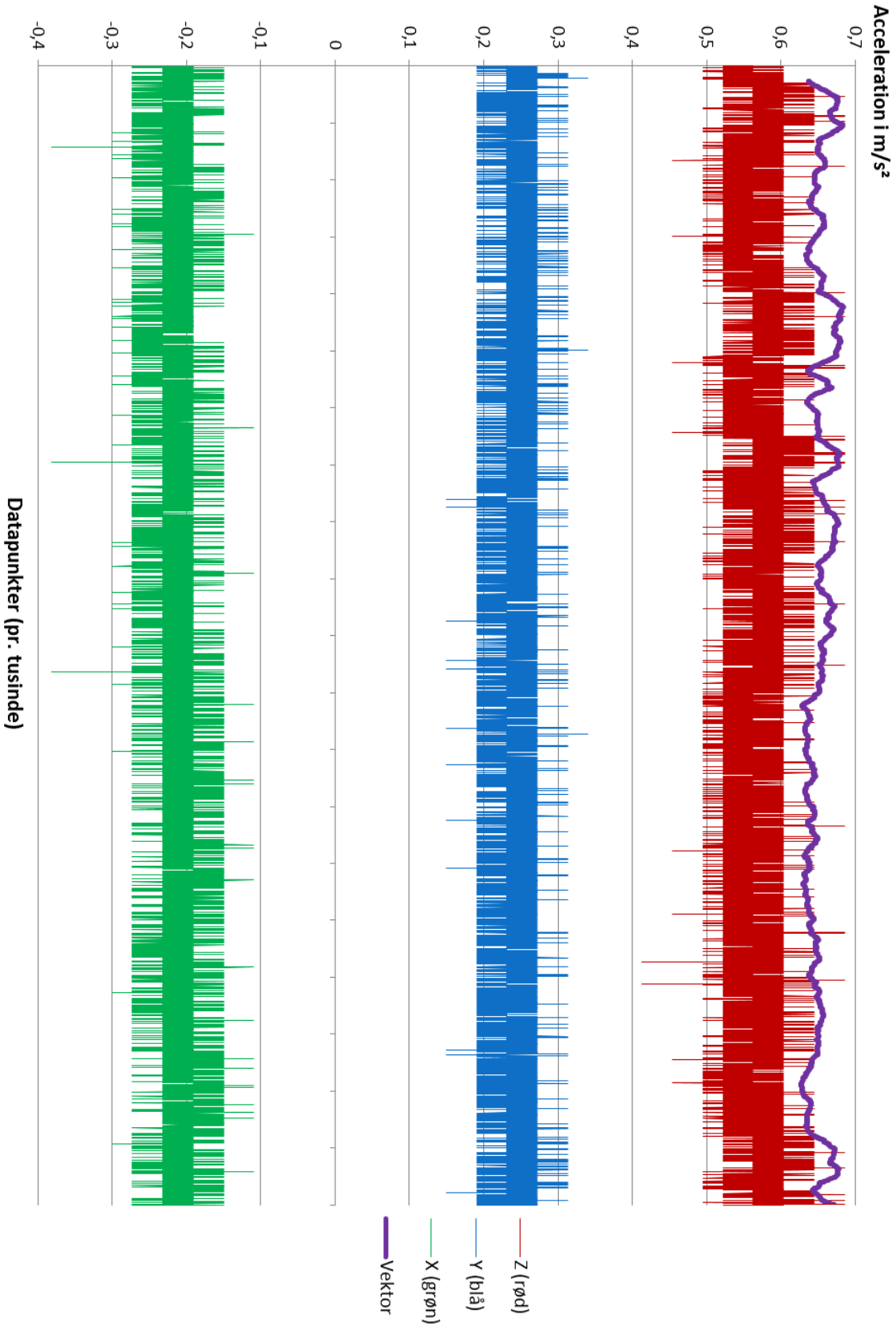


Figur 10: Større udgave af Figur 3



Figur 11: Større udgave af Figur 4

6.1.2 Test af accelerometer



Figur 12: Større udgave af Figur 5

6.2 Kravspecifikation

I henhold til retningslinjerne for en god kravspecifikation(Van Vliet, 2003, pp. 224-226) (The Institute of Electrical and Electronics Engineers, Inc., 1998, p. 4), er punkterne under kravspecifikationen opstillet således at de opfylder følgende krav:

En god kravspecifikation bør være:

1. Korrekt
2. Entydig
3. Fuldstændig
4. Konsistent
5. Arrangeret efter betydning og / eller stabilitet
6. Verificerbar
7. Modificerbar
8. Sporbar

For at synliggøre punkt 5 herover vil kravspecifikationen følge forslaget i punkt 4.3.5.1(The Institute of Electrical and Electronics Engineers, Inc., 1998, p. 7) og rangordne krav ved at klassificere krav som afgørende, betingede og kan udelades. Disse klassifikationer er nærmere beskrevet i punkt

Punkt 5 (The Institute of Electrical and Electronics Engineers, Inc., 1998, p. 10) angiver retningslinjer for hvorledes strukturen for en sådan punktliste kan sammensættes; listen herunder er lavet derudfra.

6.2.1 Introduktion

6.2.1.1 Formål

Formålet med dette dokument er at beskrive samtlige krav til det system, der skal udvikles til *lokationsbaseret styring af logistiske processer med optimerede sensordata fra smartphones*.

Dokumentet er udarbejdet til brug for udvikling af en smart-klient til produktporteføljen *Synchronicer* der er udbydes af firmaet Soft Design A/S. Det er rettet mod systemarkitekter og -udviklere, og er ment som anker i designfasen.

6.2.1.2 Tiltænkte modtagere

De tiltænkte modtagere af denne kravspecifikation er udviklerne af applikationen som både prototype og færdig del af systemet. For brugerne af systemet vil det primært være afsnit 6.2.3.2 Funktionelle krav, der kan have interesse.

6.2.1.3 Brugsområde

På baggrund af kravspecifikationen vil en prototype blive implementeret.

Prototypen har til formål at påvise muligheden for at optimere styring af logistiske processer gennem brugen af sensorer, samt anskueliggøre potentielle forbedringer af Synchronicer.

Denne prototype vil demonstrere

a) Dynamisk indhentning af positioner, styret på baggrund af tidligere positioner og andre sensordata
b) Indrapportering af positioner der anses for væsentlige
c) Udførelse og vedligehold af en køreplan

Prototypen vil ikke implementere

a) Navigationsvejledning til brugeren
b) Autentificering af brugere
c) Individuel brugeropsætning af applikationen

6.2.1.4 Dokumentkonventioner

Kravspecifikationen er udformet efter IEEE 830-1998 standarden(The Institute of Electrical and Electronics Engineers, Inc., 1998).

De enkelte krav i kravspecifikationen identificeres ved hjælp af en kort bogstavkode, efterfulgt af et rækkefølgenummer. Fx har alle krav til den del, som vedrører brugerens handlinger (User Action) ”UA” et fortløbende nummer, eks. ”UA-1”.

Bemærk, at der kan være spring i den fortløbende nummerering, idet hvert enkelt krav bibeholder sit nummer, selvom det enkelte krav er blevet fjernet i forbindelse med opdatering af kravspecifikationen.

6.2.1.5 Definitioner, akronymer, og forkortelser

6.2.1.5.1 Navne og titler

Dette dokument gør brug af følgende benævnelser for navne og titler

Fokusområdet

”Lokationsbaseret styring af logistiske processer med optimerede sensordata fra smartphones”

Firmaet

Soft Design A/S

System(et)

Produktporteføljen ”Synchro**nicer**” udviklet af Soft Design A/S

Serveren

Systemets back-end, der tager hånd om kommunikation med database og forretningslogik på højere niveau end den enkelte enhed

Enheden

Den mobile enhed hvorpå applikationen skal køre. Forventes at være en Android-baseret smartphone

Applikationen

Det produkt der er beskrevet i dette dokument

Prototypen

Implementeringen af det produkt der er beskrevet i dette dokument, under navnet ”DTU Finder”

6.2.1.5.2 Klassificering af krav

Ved samtlige funktionelle krav angives en klassifikation, der angiver kravets betydning i det samlede system. Disse klassifikationer inddeles som nedenstående

Afgørende.

Kernefunktion, der kan spores tilbage til projektaftalen. Indebærer, at applikationen ikke vil være acceptabelt, hvis dette krav ikke er opfyldt som aftalt.

Betinget.

Funktion der er ønskelig, men som ikke er strengt nødvendig i en tidlig udgave af systemet. Indebærer, at dette krav vil forbedre applikationen, men ikke vil gøre det uacceptabelt, hvis det ikke er opfyldt.

Kan udelades.

Funktion, som ikke er nødvendig for at systemet overordnet kan fungere, men som stadig er ønskelig. Indebærer, at dette krav ikke er direkte relateret til applikationens centrale funktionalitet og ikke vil gøre det uacceptabelt, hvis det ikke er opfyldt.

6.2.1.6 Referencer

Flygenring, J., & Kindler, E. (7. Januar 2011). Projektaftale; Lokationsbaseret styring af logistiske processer med optimerede sensordata fra smartphones. *Projektaftale, Kandidatspeciale* . Kongens Lyngby, Danmark: Danmarks Tekniske Universitet.

Holm, K. (Januar-april 2011). Møder og brevveksling. (J. Flygenring, Interviewer)

6.2.1.7 Oversigt

Sektion 2.2.1 *Overordnet* af denne kravspecifikation vil give en mere generel oversigt over systemet, mens sektion 2.2.2 *Specifikke krav* vil behandle de enkelte krav.

6.2.2 Overordnet beskrivelse

6.2.2.1 Produktperspektiv

Produktet er en smart-klient der skal indgå som en del af det eksisterende system. Systemet indbefatter i sin nuværende udgave en server der håndterer og gemmer data, en række klienttyper samt en kommunikationsprotokol til at transmittere data mellem klienter og server.

Applikationen skal derfor arbejde med eksisterende databaser og etablerede services, hvilket mindsker muligheden for fleksibilitet. En øget fleksibilitet kan opnås gennem bedre behandling og udnyttelse af data.

Applikationen skal kodes stringent og veldokumenteret, for at gøre det lettere at sætte sig ind i koden, hurtigere og mere præcist at indføre rettelser og så der er større og bedre basis for en fremtidig videreudvikling.

Brugervenligheden i applikationens brugerinterface skal være høj nok til at alle potentielle brugere kan bruge det, uden det store behov for manualer eller efteruddannelse. I afsnit 6.2.2.3 Brugerkarakteristika uddybes hvad der menes med potentielle brugere.

6.2.2.1.1 Brugergrænseflader

Applikationens brugergrænseflade designes ud fra at brugeren hovedsagligt skal reagere på forespørgsler fra applikationen på baggrund af de data det indhenter fra serveren. Dette holdes så vidt muligt i form af 1 til 6 virtuelle knapper med simple muligheder.

I prototypen vil der blive lagt minimal vægt på udvikling og udseende af brugergrænsefladen, ligesom brugervenligheden og svartiden på brugerens input ikke er noget der tilstræbes aktivt.

Brugergrænsefladen vil være begrænset til

- At vise informationer der er relevante for det enheden er igang med, herunder indsamling og behandling af data og kommunikation med serveren
- At tage imod simpel input udelukkende gennem tryk på virtuelle knapper.

6.2.2.1.2 Hardwaregrænseflader

Kravene til hardwaregrænseflader på den enhed hvor prototypen skal afvikles er følgende

6.2.2.1.2.1 Berøringsfølsom skærm

Betinget.

Da der lægges minimal vægt på brugergrænsefladen vil håndtering af fysiske knapper ikke være en del af prototypen, så her vil en berøringsfølsom skærm være afgørende for at kunne kommunikere med enheden.

Hvis applikationen bliver afviklet på en enhed der kan give samme funktionalitet som en berøringsfølsom skærm gennem fysiske knapper, er en berøringsfølsom skærm ikke en nødvendighed. Det kan også være muligt at betjene prototypen ved brug af fysiske knapper, men det kommer an på enheden og er ikke noget der tilstræbes.

6.2.2.1.2.2 GSM-radio

Afgørende.

GSM-radioen er hovedsagligt nødvendig til at sørge for kommunikation både til transmitering af data mellem applikationen og serveren, men også i tilfælde hvor eksempelvis SMS eller telefonopkald er nødvendigt for applikationens virke.

Herudover kan informationer om sendemaster og andre GSM-net relaterede data bruges som en unøjagtig men hurtig og billig måde, både med hensyn til pris og strømforbrug, at positionere enheden geografisk på en større skala.

6.2.2.1.2.3 GPS-modtager

Afgørende.

GPS-modtageren er en af hjørnestenene i applikationens virkeområde, da det er den mest præcise måde at positionere enheden i stortset alle udendørs situationer. Uden GPS-modtager bliver data mindre præcise end de er i systemet idag.

6.2.2.1.2.4 Accelerometer

Afgørende.

Accelerometeret er en anden hjørnesten i applikationens virkeområde, da det er data fra den sensor den yderligere positionering forbedres udfra.

6.2.2.1.3 Softwaregrænseflade

Prototypen skal afvikles under styresystemet Android.

6.2.2.1.3.1 The Android Operating System

Afgørende.

Version: 1.6 eller højere

API niveau: 4 eller højere

Kilde til anskaffelse af softwaren: http://www.android.com

6.2.2.1.4 Kommunikationsgrænseflade

6.2.2.1.4.1 Internet over HTTP

Afgørende.

Som protokol til kommunikation med serveren benyttes HTTP.

Så længe protokollen er implementeret så den lever op til standarden har det ingen indvirken på applikationen præcis hvad der ligger bag, så det lades op til den enkelte enhed hvordan protokollen implementeret og hvilke teknologier der ligger bag.

6.2.2.1.5 Hukommelsesforbehold

Da mængden af data der skal gemmes lokalt er meget lille i forhold til den hukommelse der er til rådighed i enheder der lever op til ovenstående krav, er der effektivt ingen forbehold.

6.2.2.2 Produktfunktioner

6.2.2.2.1 Brugerhåndtering

Sikrer at applikationen betjenes af den korrekte bruger og forbereder systemet på at modtage data tilknyttet den pågældende bruger.

6.2.2.2.2 Indhente køreplan

Henter en liste over lokationer brugeren skal besøge.

6.2.2.2.3 Udføre køreplan

Håndterer en køreplan og de dertilhørende lokationer.

6.2.2.2.4 Styre den logistiske proces på baggrund af køreplan og position

Ændrer tilstanden af systemet og opdaterer den igangværende logistiske proces i forhold til den aktuelle opgave fra køreplanen og enhedens position.

6.2.2.2.5 Styre positionshåndtering i forhold til proces og lokation

Styrer indhentningen og håndteringen af positioner i forhold til den proces der er under afvikling og den eller de relevante lokationer.

6.2.2.2.6 Behandle positioner

Behandler de indhentede oplysninger om position så de giver værdi i systemet. Behandlingen kan variere efter proces og tilstand i systemet.

6.2.2.2.7 Kommunikere med serveren

Sender informationer om proces, tilstand og position til og modtager informationer og instruktioner fra serveren.

6.2.2.2.8 Individuelle indstillinger

Indstiller enheder til at passe bedst muligt til forskellige brugere og processer.

6.2.2.2.9 Oversættelse af brugergrænsefladen

Gør det muligt at oversætte brugergrænsefladen til forskellige sprog og ontologier.

6.2.2.3 Brugerkarakteristika

Et endeligt produkt baseret på dette dokument skal kunne bruges af alle uanset uddannelse, erfaring og teknisk ekspertise, med et minimum af optræning.

Prototypen bliver implementeret med forventningen, at brugere er indforstået med hvordan hele systemet, enhedens sensorer og de bagvedliggende mekanismer fungerer.

6.2.2.4 Antagelser og afhængigheder

6.2.2.4.1 Software

Systemet er afhængigt af tilgængeligheden af Android styresystemet, i den angivne version. Alle nødvendige eksterne funktioner er en del af Android.

Det kan rimeligvis antages at ingen relevante funktioner bliver fjernet fra styresystemet medmindre et bedre alternativ bliver stillet til rådighed.

6.2.2.4.2 Sensorer

Systemet er afhængigt af de angivne sensorer. Uden disse eller tilsvarende kan applikationens funktionalitet ikke opretholdes.

6.2.2.5 Fordeling af krav

Krav til brugerhåndtering, herunder verificering og initiering, kan udelades fra applikationen indtil funktionaliteten dækker tidligere klienter i systemet.

Krav til oversættelse af brugergrænsefladen kan ligeledes udelades fra applikationen indtil funktionaliteten dækker tidligere klienter i systemet.

6.2.3 Specifikke krav

6.2.3.1 Eksterne grænseflader

6.2.3.1.1 Input

Følgende er de data og informationer som applikationen modtager fra systemet. Disse kan komme på baggrund af enten en forespørgsel fra enheden eller en stimuli i en anden del af systemet.

6.2.3.1.1.1 Køreplan

En samling af arbejdsopgaver brugeren skal udføre.

6.2.3.1.1.2 Beskeder

Meddelelser som serveren kan bruge til, at gøre brugeren opmærksom på vigtige forhold. Kan for eksempel bruges til, at gøre opmærksom på opdateringer i køreplanen.

6.2.3.1.1.3 Kommandoer fra systemet

Meddelelser som serveren kan sende direkte til enheden, hvorved den vil reagere i overensstemmelse med indholdet. Kan for eksempel bruges til, at få systemet til at indhente en ny, opdateret køreplan.

6.2.3.1.2 Output

Følgende er de data og informationer som applikationen forventes at sende til systemet. Disse kan komme på baggrund af enten en fysisk påvirkning af enheden eller stimuli fra brugeren.

6.2.3.1.2.1 Positioner

Opsamles løbende i forhold til enhedens position og bevægelse, og sendes til serveren efterhånden som de bliver anset for værdiskabende.

6.2.3.1.2.2 Kommandoer til systemet

Meddelelser som enheden sender til serveren, enten autonomt eller på baggrund af stimuli fra brugeren. Kan for eksempel informere om, at en opgave er afsluttet eller anmode om en ny køreplan.

6.2.3.2 Funktionelle krav

6.2.3.2.1 Bruger

6.2.3.2.1.1 Start af applikationen (UA-1)

Afgørende

Brugeren skal kunne starte applikationen fra enhedens brugergrænseflade. Kan applikationen ikke startes bliver brugeren bedt om at kontakte supportafdelingen eller enheden bliver fejlmeldt.

6.2.3.2.1.2 Autentificering af en bruger (UA-2)

Kan udelades

Brugeren skal kunne autentificere sig overfor serveren.

6.2.3.2.1.3 Kun autoriseret brug af systemet (UA-3)

Kan udelades

Brugeren skal autentificere sig overfor serveren for at bruge applikationen. Hvis brugeren ikke er autoriseret til at bruge pågældende enhed eller applikation, eller har brugt forkert brugernavn og/eller kode vender applikationen tilbage til autentifikationsprocessen og viser information om problemet.

6.2.3.2.1.4 Initiere en bruger (UA-4)

Kan udelades

Brugeren skal modtage sin personlige opsætning af applikationen fra serveren.

6.2.3.2.1.5 Brugeren stopper applikationen (UA-5)

Kan udelades

Brugeren skal kunne stoppe applikationen, der sender den aktuelle opsætning til serveren samt information om brugerens igangværende opgave og køreplan til serveren.

6.2.3.2.1.6 Godkend køreplan (UA-6)

Afgørende

Brugeren skal kunne godkende den køreplan serveren har returneret, for at gå i gang med udførslen af den.

6.2.3.2.1.7 Godkend opgave (UA-7)

Afgørende

Brugeren skal kunne godkende den opgave applikationen er nået til i udførslen af køreplanen, for at gå i gang med udførslen af den.

6.2.3.2.1.8 Afslut opgave (UA-8)

Afgørende

Brugeren skal kunne afslutte den igangværende opgave.

6.2.3.2.1.9 Udsæt opgave (UA-9)

Betinget

Brugeren skal kunne udsætte den igangværende opgave.

6.2.3.2.1.10 Opret observation (UA-10)

Betinget

Brugeren skal kunne lave en observation på sin aktuelle position.

6.2.3.2.1.11 Beskriv observation (UA-11)

Afgørende

Brugeren skal kunne tilknytte en beskrivende tekst til en observation.

6.2.3.2.1.12 Anmod om assistance (UA-12)

Afgørende

Brugeren skal kunne anmode om assistance på sin aktuelle position.

6.2.3.2.2 Applikationen

6.2.3.2.2.1 Applikationen initialiseres (AA-1)

Afgørende

Applikationen skal gøre klar til at blive brugt og opsætte forbindelse til og klargøre de aktuelle sensorer.

6.2.3.2.2.2 Verificer bruger (AA-2)

Betinget

Applikationen skal sende brugerens autentifikationen til serveren, der verificerer brugeren og returnerer brugerens opsætning til applikationen.

Hvis brugeren ikke er autoriseret til at bruge pågældende enhed eller applikation, eller har brugt forkert brugernavn og/eller kode vender applikationen tilbage til autentifikationsprocessen og viser information om problemet.

6.2.3.2.2.3 Klargør til bruger (AA-3)

Betinget

Applikationen skal indlæse en eventuelt returneret opsætning fra serveren, når brugeren er autentificeret.

6.2.3.2.2.4 Indhent køreplan (AA-4)

Afgørende

Applikationen skal indhente den relevante, aktuelle køreplan for brugeren fra serveren.

6.2.3.2.2.5 Præsenter køreplan (AA-5)

Betinget

Applikationen skal vise brugeren informationer om køreplanen og spørge om at godkende den.

6.2.3.2.2.6 Afvis køreplan (AA-6)

Betinget

Godkender brugeren ikke køreplanen skal applikationen informere serveren om, at brugeren har afvist at påbegynde den pågældende køreplan og anmode serveren om en ny.

6.2.3.2.2.7 Start køreplan (AA-7)

Afgørende

Applikationen skal påbegynde afviklingen af køreplanen.

6.2.3.2.2.8 Stop køreplan (AA-8)

Betinget

Applikationen skal stoppe afviklingen af køreplanen og informere serveren om at afviklingen af den pågældende køreplan er afbrudt.

6.2.3.2.2.9 Præsenter opgave (AA-9)

Betinget

Applikationen skal vise brugeren informationer om næste opgave i køreplanen og anmode brugeren om at godkende den.

6.2.3.2.2.10 Afvis opgave (AA-10)

Betinget

Godkender brugeren ikke opgaven skal applikationen informere serveren om, at brugeren har afvist at påbegynde den pågældende opgave og præsentere den næste opgave i køreplanen.

6.2.3.2.2.11 Udfør opgave (AA-11)

Afgørende

Applikationen skal registrere hvor enheden befinder sig i forhold til betingelserne for den igangværende opgave. Brugeren skal løbende opdateres med relevante informationer om enhedens position i forhold til destinationen. Positioner der bliver bedømt som værdiskabende bliver indrapporteret til serveren.

6.2.3.2.2.12 Afslut opgave (AA-12)

Betinget

Applikationen skal informere serveren om, at den igangværende opgave er afsluttet. Udførelse af næste opgave i køreplanen påbegyndes.

Har brugeren valgt at udsætte opgaven, informeres serveren om dette.

Kan applikationen gennem input fra brugeren og sensorer fastslå at destinationen for opgaven er nået informeres serveren om at opgaven er fuldført.

6.2.3.2.2.13 Verificer afsluttet opgave (AA-13)

Kan udelades

Applikationen skal gennem et skærmbillede muliggøre validering, for eksempel i form af stregkodeskanning og underskrift fra modtager.

6.2.3.3 Performancekrav

I implementationen af en prototype baseret på dette dokument, er der ingen krav til performance.

6.3 Brugseksempler

6.3.1 Eksempler på brug af det eksisterende system

Synchro**nicer**, i sin nuværende form, bliver i store træk anvendt til fire forskellige typer mobile medarbejdere og deres arbejdsopgaver. Naturligvis er der flere typer, men i bund og grund består alle medarbejderes arbejdsopgaver af en eller flere af de fire atomare arbejdsopgaver, beskrevet gennem deres stereotype medarbejdertype.

De fire omtalte er

- *Teknisk forvaltning* der foretager og udbedrer *Observationer*
- *Brugeren* anmoder om *Assistance*
- *Buddet* der tager sig af *Pakkespedition*
- *Lasbilschaufføren* der står for *Containerfragt*

I dette afsnit vil jeg beskrive de fire medarbejdertyper og deres typiske arbejdsdag.

6.3.1.1 Observation

En observation er noget bemærkelsesværdigt som en medarbejder ser under udførslen af sine normale arbejdsopgaver. En bemærkelsesværdig ting i denne sammenhæng, er et problem eller en uhensigtsmæssighed der skal rettes, fx et bøjet vejskilt, et væltet træ eller nedfaldne højspændingsledninger.

For at der kan blive gjort noget ved observation skal den først observeres og noteres, så skal den indrapporteres og systematiseres, og til sidst uddelegeres og udbedres.

6.3.1.1.1 Eksempel på en arbejdsdag

Morten Jensen møder ind på sin arbejdsplads om morgenen. Han tager sig en kop kaffe, henter sin arbejdsplan for dagen og sætter sig med sine kollegaer og læser den igennem. Der er ikke ligefrem noget overraskende.

Han drikker sin kaffe færdig og går med sin makker ned til deres bil. Første opgave er en de ikke fik klaret dagen før. Et vejskilt er forsvundet så de skal sætte et nyt op. Da de kommer frem ser de at pælen som skiltet skal sidde på, er lidt bøjet. De har heldigvis værktøjet med til at klare opgaven og går straks i gang. Da de er færdige sætter Morten og hans makker sig i bilen igen og begynder at køre videre til næste opgave. Mens de kører noterer Morten på sin køreplan, at de har sat skiltet op og skriver at de også rettede pælen da de var der.

På vej til næste stop ser Mortens makker, at en stor gren er brækket af et træ og nu ligger ovenpå el-ledningerne. Da de ikke er uddannet og udstyret til at håndtere den slags,

skriver Morten ned på et stykke papir hvad der var galt og hvor de havde set det. Papiret ryger ned i lommen og turen fortsætter.

Da de når frem til næste opgave, *"Knus glas på offentlig vej – fej det hele sammen og tag med tilbage",* ser de at det smadrede glas de er sendt ud for at fjerne fra gaden, er et stort stykke glaskunst og det kommer til at tage lang tid at samle det hele sammen. De går i gang med at feje efter at have lagt en hurtig plan for hvordan og hvorledes de vil gribe det an.

Da de har udført opgaven kan frokosten indtages ude i felten og Morten og hans kollega kan dermed nemmere komme videre til næste opgave.

Næste opgave på arbejdsplanen er græsslåning af en mindres stribe græs langs en landevej. Mellem arbejdsplanen blev udarbejdet og Morten og hans kollega når frem til stedet har det dog regnet kraftigt, hvilket medfører at de ikke har mulighed for at udføre opgaven. Morten må derfor notere på et nyt stykke papir at de ikke kunne udføre opgaven grundet vejret.

Eftersom dette var den sidste opgave på arbejdsplanen må Morten og hans kollega vende tilbage til arbejdspladsen før tid. De har ikke mulighed for at indhente informationer om andre foreliggende opgaver og må derfor afslutte dagen tidligere end forventet. Når Morten og hans kollega vender tilbage til arbejdspladsen sætter de sig ved hver deres computer og må manuelt indtaste dagens udførte arbejde. Morten tager de håndskrevne notater han har skrevet i løbet af dagen og indtaster dem på computeren. Han indser dog at nogle af dem er ubrugelige da han ikke kan læse sin egen håndskrift. Når dagen er omme indgiver de deres rapporter og en ny arbejdsplan kan udarbejdes til dem.

6.3.1.2 Assistance

Familien Jensen bestående af mor Bente, far Jørgen og deres tre børn Magnus, Andreas og Sofie er på vandretur i de østrigske alper. De har ladet bilen stå på en parkeringsplads og vil de næste par dage gå med oppakning og overnatte i telt.

Efter 7 timers vandring falder Andreas og vrikker om på foden. Forældrene mistænker at han muligvis har brækket anklen og vil tilkalde en ambulance. Jørgen beslutter sig for at ringe til den østrigske alarmcentral. I mellemtiden har han dog glemt, hvor på kortet de er og i tillæg til dette skal han prøve at forklare situationen og, hvor de befinder sig på et sprog han slet ikke behersker. Efter lang tid lykkedes det familien at forklare situationen og en ambulance når frem til stedet og fragter dem til en nærliggende skadestue.

Da Andreas endelig er blevet tilset af en læge er klokken blevet mange og familien indser at de må prøve at finde et lokalt hotel, hvor de kan sove natten over. Ingen i familien ved dog, hvor de skal finde et hotel i byen – og om de overhovedet har plads til dem.

6.3.2 Eksempler på brug af applikationen

6.3.2.1 Rapportering af observationer

Denne arbejdsgang bliver udført af forskellige typer kommunalt ansatte ved siden af deres normale funktion. Kan også udvides[15] til at lade borgere[16] udføre[17] den, i et vist omfang.

I det følgende henvises til den der udfører arbejdsgangen som 'brugeren'

Arbejdsgangen starter når brugeren ser noget under udførsel af sine normale gøremål, der er eller kan blive et problem, fx huller i veje, skader på vejbelægning, manglende dæksler, bøjede, væltede eller manglende vejskilte, væltede træer, trærødder der vokser op gennem fortov og vejbelægning, nedfaldne ledninger eller lignende. Brugeren tager nu sin smartphone frem og åbner programmet til at rapportere observationer.

6.3.2.1.1 Start indberetning

Brugeren vælger at starte en indberetning.

- Programmet registrerer dato og tidspunkt for indberetningens start.
- Programmet registrerer brugerens medarbejdernummer eller anden form for ID (fx for borgere).
- Programmet går i gang med at rekvirere brugerens position, gennem GPS og GSM.

6.3.2.1.2 Beskriv observation

Brugeren kan

- skrive en kort tekst i programmet der beskriver observationen
- vælge observationens prioritet på en forud defineret liste
  Listen varierer gradvist fra prioriteten '**note**', der skal holdes øje med men ikke nødvendigvis handles overfor foreløbigt, (fx trærødder der er ved at bryde vejbelægningen og let bøjede vejskilte) til '**akut**', der kan være et spørgsmål om liv og død hvis det ikke bliver taget hånd om hurtigt, (fx manglende dæksler og nedfaldne højspændingsledninger).
- vælge at tage et billede af observationen for bedre at illustrere hvad observationen er

6.3.2.1.3 Rekvirer position

Programmet finder brugerens position ud fra forskellige sensorer, alt efter forhold og tilgængelighed.

---

[15] http://hulivejen.dk/
[16] http://www.forsyningen.dk/system/givosetpraj/ og http://www.forsyningen.dk/system/mmsbilleder/
[17] http://www.furesoe.dk/Nyheder/2011/Januar/GivOsEtTip.aspx

Hvis det ikke umiddelbart er muligt at finde en tilstrækkeligt præcis position, kan brugeren spørges om at indtaste en beskrivelse af lokationen, fx en adresse eller en beskrivelse af et bestemt sted fx i en park. Denne beskrivelse suppleres med eventuelle rekvirerede positionsdata.

6.3.2.1.4 Afsend

Programmet sender beskrivelsen af observationen og den rekvirerede position til den tilknyttede server.

6.3.2.2 Udbedring af rapporterede observationer

De rapporterede observationer kommer løbende ind til en central server, hvorfra der sendes information videre til den kommunale kontakt.

Det resterende på denne side foregår i kunden regi, og det har vi i dag ikke med i vores løsning, men det kunne meget vel komme hvis der er efterspørgsel.

Processen på den kommunale server fortsætter ved, at de indkomne observationer sorteres efter prioritet.

Hvis en observation ikke er tilknyttet en lokation, giver systemet besked til en medarbejder, der manuelt gennemgår de tilknyttede data, positionsbeskrivelse, eventuel GSM position og eventuelt billede, for at estimere lokationen.

Observationerne inddeles dernæst efter type og geografisk beliggenhed og sendes til de relevante medarbejdere. Observationer med højest prioritet bliver sendt direkte til medarbejderne, så de kan blive udbedret straks. Observationer med lavere prioritet bliver sat på medarbejdernes liste over arbejdsopgave for den eller de følgende dage.

Når en medarbejder møder ind på arbejde bliver listen over arbejdsopgaver udleveret og medarbejderen går i gang med at løse dem.

Modtager medarbejderen i løbet af dagen en opgave med højere prioritet end dem på listen bliver den sat øverst på listen hvorefter arbejdet fortsætter.

Hvis alle observationer ikke er taget hånd om når medarbejderen har fyraften bliver disse sat øverst på listen af arbejdsopgaver til den følgende dag.

6.3.2.3 Anmodning om assistance

Denne arbejdsgang bliver udført af forskellige typer mobile medarbejdere, fx biologer i feltarbejde.

I det følgende henvises til den der udfører arbejdsgangen som 'brugeren'

Arbejdsgangen starter når brugeren under udførsel af sine gøremål bevæger sig rundt. Pludselig får brugeren behov for assistance i forbindelse med en observation eller hændelse.
Dette kunne fx være biologen der går langs et forurenet vandløb, for at finde kilden, og så pludseligt ser et rør der løber selvlysende blåt slim ud af, eller som under kontrol af vandkvaliteten sidder fast i mudder og ikke kan komme fri

Brugeren tager sin smartphone og med en enkelt knap åbner programmet til at anmode om assistance.

6.3.2.3.1 Anmod om assistance

Programmet går straks i gang med at tilkalde assistance.

Hvis kilderne er tilgængelige går programmet i gang med at bestemme brugerens position gennem GPS og GSM, men fejler dette afsendes melding uden disse informationer efter en vis forsinkelse.

6.3.2.3.2 Positionsdata

Er GPS og/eller GSM data om brugerens position tilgængelige bliver disse gemt, ellers fortsætter programmet med at forsøge at rekvirere dem.

6.3.2.3.3 Afsend

Brugerens position, ud fra indhentede data, sendes med anmodningen om assistance.

6.3.2.4 Svar på anmodning om assistance

6.3.2.4.1 Anmodning modtages

Er der knyttet en position til anmodningen indhentes denne og tilknyttes i form af markering på et kort.

6.3.2.4.2 Meddel operatør om anmodningen

Den operatør der er tilknyttet brugeren modtager straks meddelelse om anmodningen.

6.3.2.4.3 *Operatør ringer til medarbejderen der ønsker assistance*

Da der ikke er nogen form for prioritet tilknyttet anmodningen må operatøren behandle alle anmodninger som kritiske, hvorfor operatøren straks ringer til brugeren for at finde ud af hvilken type assistance det drejer sig om. Operatøren stiller typisk spørgsmål i stil med

- Er DU okay?
- Hvad er der galt?
- Hvilken assistance er der brug for

Under samtalen fortsætter brugerens enhed med at forsøge at bestemme positionen, og eventuelt nye informationer sendes til operatøren.

Hvis brugerens enhed stadig ikke har fået en GPS position kan det være nødvendigt med uddybende spørgsmål, da GSM positioner typisk er et område med en relativt stor radius

- Hvor er det?

6.3.2.4.4 *Operatør finder medarbejderens lokation*

Ud fra alle de data operatøren nu er kommet frem til kan brugerens position nogenlunde præcist bestemmes. På baggrund af dette kan operatøren også finde den nærmeste brugbare vej til assisterende køretøjer.

6.3.2.4.5 *Operatør tilkalder passende assistance*

Operatøren kontakter den nødvendige assistance og sender dem til brugerens lokation.

6.3.2.4.6 *Assistance modtages*

Brugeren bliver glad.

6.3.2.5 Pakkespedition

I det følgende henvises til den der udfører arbejdsgangen som 'chaufføren'

6.3.2.5.1 *Start*

Chaufføren sætter sig ind i sin vogn og læser sin køreplan.

Køreplanen er optimeret i forhold til tid og distance, da opsamlinger og leverancer kan ske i vilkårlig rækkefølge.

6.3.2.5.2 *Opsamling*

Enhver tur starter med at chaufføren henter en pakke.

6.3.2.5.3 *Leverance*

Når vognen når frem til sin destination og leveres pakken.

6.3.2.5.4 *Tankning*

Når som helst kan det være nødvendigt at tanke, men dette skal systemet naturligvis også holde øje med.

6.3.2.6 Containertransport

I det følgende henvises til den der udfører arbejdsgangen som 'chaufføren'

6.3.2.6.1 *Start*

Chaufføren sætter sig i sin vogn og tilmelder sig systemet hvorved en tur tildeles med dagens planlagte stop.

Systemet logger hvor vognen befinder sig og det er derfor muligt efterfølgende at verificere at en eventuel given rute blev fulgt. Systemet kontroller ikke rute automatisk.

6.3.2.6.2 *Opsamling*

Enhver tur starter med at chaufføren henter en container.

Indeholder denne container farligt gods ('ADR') kræver det at chaufføren er certificeret til at køre med den type ADR. Desuden medfører det oftest visse rutekrav og andre fartgrænser end normalt.
Det er muligt for systemet at overvåge hvor chaufføren befinder sig på hvilke tidspunkter, men det er op til chaufføren selv at overholde bestemmelserne.

6.3.2.6.3 *Pre-announce*

Pre-announce er en ordning hvor chaufføren kan melde sin ankomst på forhånd, så containerens indhold og andre forhold kan forhåndsgodkendes. Bliver fx brugt i forbindelse med terrorsikrede havne hvor det forkorter behandlingstiden af vognen og containeren betydeligt.

Informationer der fx sendes i forvejen er

- Vognnummer / -ID
- Vognmands ID
- Chauffør ID
- Last ID
- Der sendes evt. også information om plomberinger numre og segl (told og veterinær)

Hermed skal tolderen kun kontrollere at de fremsendte informationer passer med vognen og at containerens plombering er ubrudt.

6.3.2.6.4 Leverance og deponering

Når vognen når frem til sin destination og alt er kontrolleret er containeren oftest regnet som leveret.

Skulle det ske at leverancen sker uden for åbningstiden deponeres containeren i et afgrænsede område hos modtageren, men anses ikke for modtaget endnu.

6.3.2.6.5 Tankning

Når som helst på en tur kan det være nødvendigt at tanke, men dette skal systemet naturligvis også holde øje med.

6.4 Bibliografi og referencer

6.4.1 Bibliografi

Flygenring, J., & Kindler, E. (7. Januar 2011). Projektaftale; Lokationsbaseret styring af logistiske processer med optimerede sensordata fra smartphones. *Projektaftale, Kandidatspeciale* . Kongens Lyngby, Danmark: Danmarks Tekniske Universitet.

Holm, K. (Januar-april 2011). Møder og brevveksling. (J. Flygenring, Interviewer)

The Android Open Source Project. (2008, September 23). *Package Reference | Android Developers*. Retrieved July 11, 2011, from Android Developers: http://developer.android.com/reference/packages.html

The Android Open Source Project. (2008, September 23). *The Developer's Guide | Android Developers*. Retrieved July 11, 2011, from Android Developers: http://developer.android.com/guide/index.html

The Institute of Electrical and Electronics Engineers, Inc. (1998). *IEEE Std 830-1998: Recommended Practice for Software Requirements Specifications.* The Institute of Electrical and Electronics Engineers, Inc.

Van Vliet, H. (2003). *Software Engineering: Principles and Practice* (2nd ed.). John Wiley & Sons, Ltd.

6.4.2 Referencer

Aberdeen Group. (2008, March 24). *Sector Insight: Small Service Fleets Use GPS to Complete More Work Orders per Technician per Day.*

ABI Research. (2008, August 21). *Press Release: Business Benefits Drive Fleet Management Systems Growth.*

ABI Research. (2008, November 18). *Press Release: Robust Future for Fleet Management Systems through 2013.*

Berg Insight. (2010). *Fleet management and Wireless M2M.*

Berg Insight. (2011). *Fleet management in the Americas.*

Xata. (2006-2011). *Xata Annual Report.*