15-12-2007

CepaIR & CepaBackup

Udvikling af software til konfigurering og kontrol af infrarød fjernbetjening, samt backup/restore af brugerdata

Bachelorprojekt i IT | Rune Tinghuus Bundgaard – s040799

Vejleder: Bjarne Poulsen
Firma: Cepa Mobility Aps.
IRN: IMM-B.Eng.-2007-54

Indhold: 95 sider eksl. forside, 1 CD-rom

Institut For Informatik Og Matematisk Modellering
Danmarks Tekniske Universitet

Indholdsfortegnelse

Figurliste

1.1. Forord

Dette bachelorprojekt er udarbejdet af Rune Tinghuus Bundgaard i forbindelse med afvikling af uddannelsen som IT-diplomingeniør ved Institut For Informatik Og Matematisk Modellering på Danmarks Tekniske Universitet.

Projektet gennemføres i samarbejde med Cepa Mobility Aps.(Cepa), hvor jeg har gennemført min praktik og derigennem opbygget gode kontakter med ansatte i udviklingsafdelingen, såvel som salg og ledelse. Den oprindelige kontakt med Cepa er oprettet via en bekendt, som gennem længere tids ansættelse og delvist ejerskab af Cepa gav udtryk for manglen på praktikanter i mindre nystartede virksomheder med begrænsede økonomiske resurser. Det blev derfor efterfølgende undersøgt, hvorvidt det var muligt at få godkendt Cepa Mobility Aps. som praktiksted. Derved resulterede den oprindelige efterspørgsel i ansættelse af undertegnede som firmaets første ing.praktikant.

Rapportens formål er at redegøre og beskrive udviklingsforløbet for prototyper af to systemer, som udvikles delvist til PDA, PC og Server. Der er tale om to adskilte prototyper, hvoraf den ene omhandler omverdenskontrol via en PDA, mens den anden omhandler instant beskyttelse af brugerdata opbevaret på PDA. Da prototyperne varetager hver deres problemstilling, vil de i rapporten blive betragtet og beskrevet som separate løsninger, og rapporten er derfor også delt op i separate dele.

Prototyperne har gennem længere tid befundet sig i Cepas planlagte produktpipeline, men er pga. de begrænsede resurser blevet rykket flere måneder på trods af stor efterspørgsel fra deres eksisterende kundegruppe.

1.2. Introduktion

Formålet med dette kapitel er at give en introduktion til projektet og klarlægge dets baggrund, så læseren får en bedre opfattelse af den konkrete problemstilling når denne introduceres. Yderligere vil der blive redegjort for den valgte løsningsstrategi, der hovedsagligt er baseret på kravspecifikationen, men også på projektets vision. Der indledes kort med en beskrivelse af virksomheden og deres eksisterende produkt for bedre at kunne forestille sig det miljø, hvor projektet skal tilpasses.

1.2.1. Virksomhedsbeskrivelse

Denne virksomhedsbeskrivelse er baseret på en samtale med en af virksomhedens stiftere, Thomas Pedersen, og der vil derfor blive gengivet oplysninger, der ikke nødvendigvis kan findes på virksomhedens hjemmeside[1].

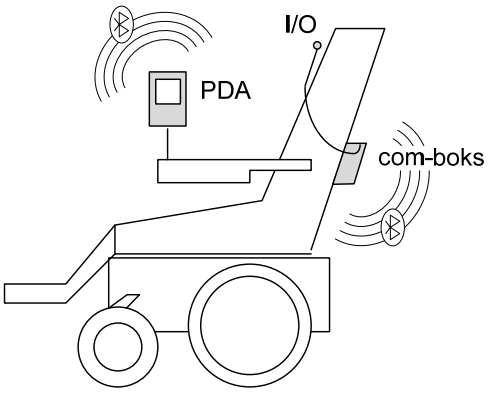
Cepa Mobility Aps. er en yngre virksomhed, som for få år siden begyndte udviklingen af et produkt, der ville gøre det muligt, via en mobil kommunikationsløsning, at telefonere og sms'e ved brug af blot en enkelt bevægelse. Idéen kom fra to af medejerne, der begge sidder i kørestol og som samtidig er stærkt bevægelseshæmmede. De var begge var udstyret med anskaffede hjælpere, men var ærgerlige over det manglende privatliv, i forbindelse med at udføre de dagligdags rutiner, såsom at skrive en sms eller foretage et telefonopkald fra mobiltelefonen. Det de manglede var et specifikt tilpasset supplement til mobiltelefonen, som kunne gøre dette muligt uden nødvendig brug af hænderne. Flere blev samlet omkring projektet og Cepa Mobility Aps. blev dannet. Efter flere måneders udvikling lå produktet klart: CepaCom.

Cepa har i dag 6-7 ansatte hvoraf 4 beskæftiger sig med udvikling og forbedring af produkter. Cepa samarbejder med flere virksomheder i forbindelse med både produktlancering og udvikling heriblandt kan nævnes Microsoft og HP, der begge har udlånt udstyr og egne produkter.

1.2.2. Eksisterende System

Hardware: CepaCom består af en com-boks indeholdende print med microcontroller samt bluetooth-chip. Til denne boks kan tilsluttes et varierende antal I/O-kontakter, der ved benyttelse generer et signal til com-boksen. I/O-kontakter er i forvejen bredt anvendt på hospitaler/i plejesektoren og kan bestå af alt lige fra en knap der trykkes, til et rør der pustes i. Com-boksen benytter sin bluetooth-forbindelse til at kommunikere I/O signaler videre til en PDA med Cepas egen software installeret. Com-boksen fungerer dermed som en kommunikationsformidler mellem bruger og PDA.

[1] http://www.cepamobility.com (2007)



Figur 1 - Skitse af kørestol med Cepas produkt

Com-boksen monteres oftest bag på kørestolen eller under sædet, men kan rent praktisk placeres hvor som helst, så længe kabelføring er mulig mellem com-boks, strømkilde og I/O-kontakter. I/O-kontakter monteres fysisk afhængigt af type og varierer derfor fra bruger til bruger. PDAen med Cepas software monteres foran på kørestolen i en vinkel, distance og højde tilpasset den enkelte bruger. Hvis brugeren har tilstrækkelig bevægelsesevne, monteres PDAen lavt og inden for rækkevidde, så brugeren kan betjene den via touchscreen.

Software: Cepas software benytter et menu-system, som autoscroller (scanner) menupunkterne igennem med en variabel hastighed, mens den samtidig visuelt gengiver menuen på skærmen for brugeren.

Figur 2 - Illustration af CepaCom kørende på PDA

Med denne sammensætning har de udviklet et "ét-klik-system" med integreret telefon og sms som forholdsvist simpelt kan monteres på den bevægelseshæmmedes kørestol.
Som noget af det sidste nye har Cepa integreret CAN-bus[2] i com-boksen, hvilket gør det muligt at anvende CepaCom sammen med flere forskellige typer kørestole. CepaCom er dermed blevet fleksibel nok til, at den fremtidige udvikling kan rettes mod fremstilling af nye I/O-kontakter frem for konstante udvidelser og forbedringer af com-boksen.

Flere oplysninger om virksomheden kan findes på hjemmesiden www.cepamobility.com, hvor der yderligere kan rekvireres brochurer samt uddybende informationer om produktet.

[2] http://en.wikipedia.org/wiki/Controller_Area_Network (2007)

1.3. Baggrund og vision

Dette afsnit beskriver kort baggrunden for de to projekter, samt visionen for den fremtidige udnyttelse af det færdige resultat. Underafsnittet "Baggrund" er delt op i to individuelle beskrivelser, mens underafsnittet "Vision" beskriver et fælles forventet mål for de to prototyper. Det forudsættes, at afsnittene "Virksomhedsbeskrivelse" og "Eksisterende System" er læst.

1.3.1. Baggrund

CepaIR: Cepa har gennem udviklingen af deres produkt fokuseret meget på genanvendelighed i den forstand, at der fremover kun behøves udviklet eksterne enheder, der ved at følge en bestemt protokol, vil være fuldt ud kompatible med com-boksen. Der kunne derfor hurtigt gå's i gang med den konceptuelle udvikling af en ekstern multifjernbetjening, der skulle gøre det muligt for brugere af Cepas produkt at kontrollere døråbnere, tv, anlæg etc. i deres hjem udelukkende ved at være i stand til at benytte en I/O-kontakt.

Cepa færdiggjorde for nylig en prototype af en infrarød fjernbetjening, som tilsluttet com-boksen kan emulere diverse standard fjernbetjeninger vha. data modtaget fra PDA'en. Fjernbetjeningen er dog indtil videre kun blevet anvendt i et lukket miljø nemlig ved brug af konstant data placeret i fjernbetjeningens microcontroller. Cepa udskød midlertidigt projektet, da en potentiel samarbejdspartner i kørestolsindustrien meldte interesse omkring et andet produkt, som dermed blev opprioriteret.

CepaBackup: Cepas kunder melder ofte tilbage med ønskede udvidelser, hvilket i mange tilfælde kan betragtes som behov. Cepas produkt sidder f.eks. fastmonteret på kørestolen og PDAen bliver derfor udsat for vind, regn, stød og slag, da kørestolene ikke altid kan stilles let til side eller i tørvejr. PDAer skal derfor ofte til reparation, hvor de bliver enten nulstillet eller udskiftet, hvorved personlige kontaktoplysninger, beskeder mm. går tabt.

1.3.2. Vision

Visionen for projekterne er at erhverve sig nogle brugbare løsningsforslag, der kan opfylde de behov, som Cepas kunder har udtrykt ønske om, nemlig omverdenskontrol og backup/genskabelse af personlige data. Yderligere gøres det muligt, ved udvikling af prototyperne at finde frem til evt. forhindringer eller flaskehalse, som måtte præsentere sig i interaktionen mellem det nye produkt og Cepas eksisterende hardware og software. Begge systemer vil efter endelig færdigudvikling og test skulle tilbydes til kunder i form af tilkøbspakker til Cepas produktpakke.

Cepa forventer derfor basalt set at kunne minimere tidsforbruget under færdigudviklingen ved at benytte sig af konklusioner og erfaringer tilegnet under projektet.

1.4. Kravspecifikation

Følgende afsnit beskriver kravene formuleret i samarbejde med Cepa. Kravene har til formål at danne en fælles forståelse for løsningen af problemstillingen. Applikationer til PDAen skal udvikles til .NET Compact Framework 2.0, mens applikationer til PC og server skal udvikles til .NET Frameworket 2.0. Dette kræves for at drage nytte af de allerede erhvervede licenser og software development kits (sdk) og for at sikre samspil med de eksisterende omgivelser.

1.4.1. Krav til CepaIR

Til prototypen af CepaIR skal der udvikles:

1. En PC-applikation som kan udtrække oplysninger og koder om fjernbetjeninger fra tekstdokumenter indsamlet på www.lirc.org[3]. Udtrukne data skal struktureres og gemmes på en MySQL-server.
2. En web-applikation som gennem få gentagne brugervenlige trin kan oprette en samling af fjernbetjeninger udvalgt fra MySQL-serveren. Derudfra skal der genereres en fil indeholdende konfigurationsoplysninger og IR-koder fra hver enkelt udvalgt fjernbetjening. Når filen er genereret, skal den gøres tilgængelig for download til PC.
3. Et plugin til Cepas software på PDAen, som gør det muligt for brugeren at kontrollere fjernbetjente enheder i sin bolig. Konfiguration og koder skal sendes til com-boksen når valgt via menu på PDAen. Com-boksen videreformidler herefter signalerne til IR-fjernbetjeningen. Data sendt via bluetooth[4] mellem PDA og com-boks skal sendes i en bestemt pakkestørrelse specificeret af Cepa.

1.4.2. Krav til CepaBackup

Til prototypen CepaBackup skal der udvikles:

1. Et plug-in til Cepas software på PDA'en, som gør det muligt for brugeren via enten GPRS eller WLAN øjeblikkeligt at foretage en backup af personlige data på en fjernserver.
2. Et plugin til Cepas software på PDAen, som gør det muligt for brugeren via enten GPRS eller WLAN at udvælge en ud af flere backups og derefter genskabe de pågældende data. Brugeren skal gøres klart opmærksom på, at genskabelsen af data vil medføre sletning af eksisterende data.
3. En server-applikation som tidsstempler og begrænser antallet af backups til et ikke nærmere defineret antal. Applikationen skal kunne returnere en backup-oversigt, så brugeren kan udvælge en vilkårlig backup. Det er vigtigt at brugeren gøres opmærksom på når max. antallet er nået, da dette vil medføre sletning af den ældste backup.

[3] Open Source projekt til udvikling af PC-styret multifjernbetjening
[4] Cepa har erhvervet et licens til SDKen Franson BlueTools og vil gerne stille dette til rådighed.

1.4.3. Afgrænsninger

Følgende afgrænsninger er besluttet i fællesskab med Cepa og skal sikre, at der ikke anvendes unødvendige resurser.

- Der skal under backup af data ikke tjekkes for duplikater, da personlige data ikke nødvendigvis er unikke.
- Der skal ved udtrækning af data fra tekstdokumenter ikke gemmes oplysninger om apparatets kategori (tv, dvd etc.), da disse informationer ikke eksisterer i samtlige dokumenter.
- Der skal ikke udtrækkes data, hvor IR-koderne er opgivet som ”raw code”, da disse ikke er understøttet af hardwaren.
- Der skal i prototypen af databasen ikke tages hensyn til pladsforbrug. Hermed tænkes der dog ikke på normaliseringen, men på datatyperne.
- Download af konfiguration og koder rettes mod PC da mængden af oplysninger kan variere meget.

1.5. Løsningsstrategi

Dette afsnit har til formål at beskrive og begrunde den valgte løsningsstrategi, som vil blive anvendt under projektet. Udvikling af prototyper giver frihed til selv at prioritere rækkefølge af forløbet og det er derfor muligt, at arbejde kronologisk med projektet. Udarbejdelsen af den infrarøde multifjernbetjening har dog højeste prioritet af de to projekter, da denne er tungest vægtet i Cepas udviklingspipeline og derfor tungest vægtet i rapporten. Den kræver yderligere, at der udvikles flere applikationer, da der ikke kun skal udvikles til anvendelse på PDA'en, men også konfiguration og oprettelse. Det er derfor mest nærliggende, at dette projekt håndteres først, så Cepa, uanset opgavens udfald, har fået noget brugbart ud af samarbejdet. CepaBackup vil derfor blive udviklet sidst af de to, hvis tiden tillader det.

1.5.1. Fremgangsmåde

Cepas nuværende softwareprodukt, der afvikles på PDA, fungerer som tidligere beskrevet via et autoscrolling menusystem, der skifter mellem menupunkterne efter et given interval. I/O-kontakterne tilsluttet com-boksen gør det muligt for brugeren at vælge et menupunkt, der fører ham videre til en undermenu eller en anden funktionalitet. Menuer og undermenuer eksisterer i en stor løkke, hvor skift mellem menupunkter er efterfulgt af en idle tidsperiode. Ved at bruge en løkke til at styre CepaComs GUI, bliver det dog kun muligt at anvende to fremgangsmåder, enten ved at arbejde med hele CepaCom og direkte integrere projektet eller ved at udvikle forholdsvist uafhængige assemblies, som Cepa selv kan integrere efterfølgende. For bedst at kunne vælge mellem de to fremgangsmåder vejes argumenter for og imod:

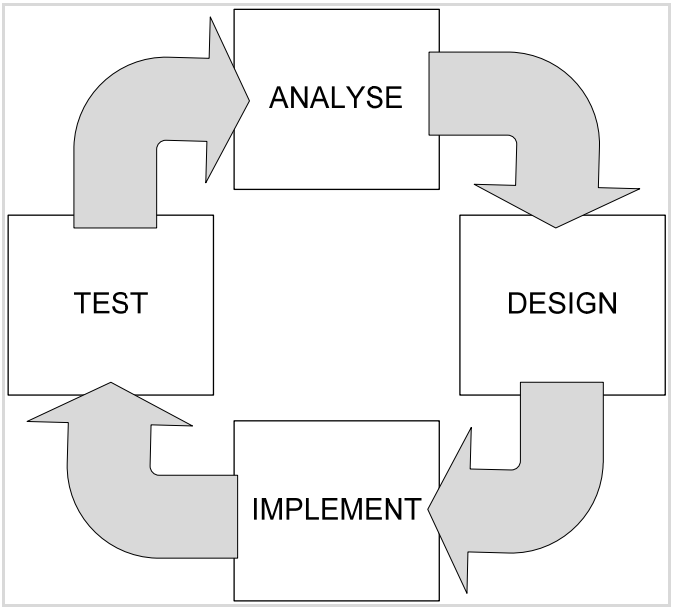
Direkte integreret udvikling:
Denne fremgangsmåde gør det muligt at bruge den eksisterende kode og GUI, til gengæld bliver den svær og rodet at dokumentere, da kun egne løsninger må dokumenteres i rapporten. CepaCom har desuden vist sig uhensigtsmæssig til udvidelser, da der kun er ganske få assemblies og ingen tydelig lagdeling i designet. Direkte integreret udvikling medfører derfor deploy af hele CepaCom, hvilket kan blive dyrt i forhold til tidsforbruget under implementering og samtidig være næsten umuligt at dokumentere.

Uafhængig udvikling:
Denne metode vil kræve udvikling af midlertidige GUIs, der kun har til formål at fremvise funktionalitet, men ellers aldrig vil blive anvendt. Dette vil dog blive lettere at dokumentere, da både design og kode vil være en selvstændig præstation. Fuldstændig uafhængighed vil dog ikke kunne opnås, da bluetooth-kommunikation skal ske gennem Cepas eget protokol-lag. Protokol-laget kræver få udvidelser for at muliggøre afsending og disse skal forsøges dokumenteret uden sammenblanding.

Et sidste faktum som tipper vægten er, at der er tale om en prototype og ikke et endeligt produkt. Største fordel opnås derfor ved at foretage uafhængig udvikling og denne beslutning har Cepa godkendt som tilfredsstillende.

1.5.2. Udviklingsmetode

Projektet vil blive udviklet ved brug af inspiration fra "Agile Software Development"[5]. Agile development er en fordel ved udvikling af prototyper, hvor kunden muligvis ikke ved præcis hvad de ønsker og derfor løbende kan komme med ændringer til kravspecifikationen. Det forventes dog ikke at Cepa vil komme med ændringer som potentielt kan true projektets deadline, men ved brug af denne udviklingsstrategi vil det være muligt i det mindste, at kunne håndtere evt. forslag på fornuftig vis. Grunden til at der kun udvikles med inspiration fra Agile Development skyldes, at projektet har en fast deadline og at det derfor vil være for risikabelt, at udvikle uden en mere omfattende planlægning. Formålet med brug af Agile Software Development vil for dette projekt derfor være, løbende at kunne producere "mini releases" af prototyperne. Disse skal ikke nødvendigvis godkendes eller testes af Cepa, men anvendes i stedet til at udbygge og udvide prototypen indtil denne lever op til kravspecifikationen.



Figur 3 - Udviklingsfaser under projektet

Udviklingsforløbet opdeles i fire faser, analyse, design, implementering og test, som tilsammen udgør en enkelt iteration. Under analysen beskrives krav, behov og miljø, hvilket anvendes under designet for at kunne målrette løsningsmodellen. Endeligt implementeres løsningsmodellen og testes.

Rapporten vil for overskuelighedens skyld kun indholde den endelige iteration for hver prototype.

[5] http://en.wikipedia.org/wiki/Agile_software_development (2007)

1.6. Kapiteloversigt

I dette afsnit præsenteres et punktopstillet kapiteloversigt, som har til formål kort at beskrive indholdet af rapportens kapitler.

- Kapitel 2: ”Analyse af CepaIR”
  - I dette kapitel analyseres der ud fra kravspecifikationen de individuelle problemstillinger for hver applikation under CepaIR projektet. Yderligere analyseres flaskehalse i form af teknologier og formater.
- Kapitel 3: ”Design af CepaIR”
  - I dette kapitel vil der blive designet løsningsmodeller for de analyserede problemstillinger fra forrige kapitel. Hver applikation vil blive designet individuelt, men med hensyntagen til deres tætte samspil.
- Kapitel 4: ”Implementering af CepaIR”
  - I dette kapitel vil et udpluk af de færdigimplementerede elementer blive beskrevet. Der vil i kapitlet lægges vægt på den røde tråd gennem de tre applikationer.
- Kapitel 5: ”Test af CepaIR”
  - I dette kapitel beskrives testen som CepaIR undergår, herunder en use case-test af PC-, server- og PDA-applikationen, udført af en ansat fra Cepas udviklingsafdeling.
- Kapitel 6: ”Konklusion af CepaIR”
  - I dette kapitel vil projektforløbet blive opsummeret, samt udbyttet målt i form af en konklusion fra den studerende og en kommentar fra virksomheden.

1.7. Delkonklusion

Alle forudsætninger for påbegyndelse af analysefasen er nu opfyldt og de to projekter CepaIR og CepaBackup vil fremover blive behandlet separat. CepaIR vil blive udviklet først og CepaBackup sidst. Krav og afgrænsninger for begge systemer er fastlagt og der vil blive udviklet uafhængigt af Cepas egen software grundet bl.a. dokumentationsbesvær ved at udvikle direkte i softwaren.

2. Analyse af CepaIR

Med kravspecifikationen og afgrænsningerne fastlagt er det nu muligt at analysere de nødvendige teknologier og formater som er nødvendige for udviklingen af CepaIR. De anvendte teknologier og formater kan have stor indvirkning på problemstillingerne, og det er derfor vigtigt at analysere dem så tidligt som muligt. Ud fra kravspecifkationen er det yderligere muligt at analysere de enkelte problemstillinger for CepaIR og beskrive dem via use cases. Begge analyser vil senere danne fundament for designet af løsningsmodellerne, og det er derfor relevant at få identificeret forretningsprocesser og evt. flaskehalse.

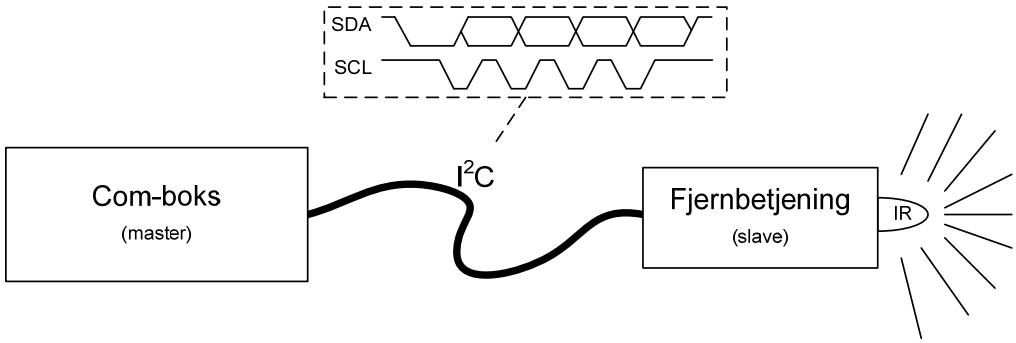
2.1. Teknologi og standard

Ved påbegyndelsen af projektet er det som nævnt vigtigt at sætte sig ind i de anvendte teknologier og standarder for bedst at kunne vurdere deres mest hensigtsmæssige anvendelse. Dette vil selvfølgelig medføre øget tidsforbrug under analysen, men vil give det bedste resultat på længere sigt, da flere problemer evt. kan håndteres eller helt undgås.

2.1.1. Fjernbetjeningen

I det følgende afsnit vil den hardwarebaserede fjernbetjening blive beskrevet, som Cepa ønsker, skal kunne kontrolleres via softwareresultatet af PDA-applikationen.

I com-boksen findes en udgang til kommunikation via en I^2C bus[6], hvor fjernbetjeningen er tilsluttet gennem et kabel. I^2C bussen fungerer via to forbindelser en til clock(SCL) og en til data(SDA), hvor en masternode styrer clock'en, mens en eller flere slavenoder modtager eller sender data. Com-boksen fungerer her som master, mens fjernbetjeningen fungerer som slave.



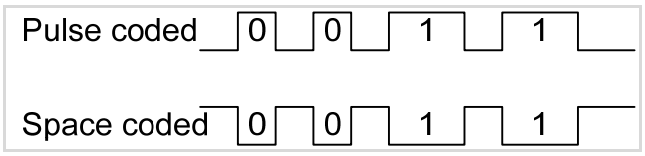
Figur 4 - Illustration af forbindelse mellem com-boks og fjernbetjening

Når dataen er modtaget af fjernbetjeningen via I^2C'en bestemmes datatypen ud fra to kategorier: konfigurationsdata eller IR-kode. Da elektriske apparater fra forskellige producenter ikke nødvendigvis modtager signaler på samme måde, er det nødvendigt at konfigurere fjernbetjeningen før afsendingen af signalet. Til konfigurering af fjernbetjeningen skal der bruges oplysninger om bl.a. følgende:

[6] http://en.wikipedia.org/wiki/I%C2%B2C

Kodningstype

Et IR-signal kan konstrueres på flere måder før modulering og afsending. Ved *Pulse encoding* er det varigheden af den høje værdi, der bestemmer udfald, mens det ved *Space encoding* er varigheden af den lave værdi. Yderligere kan værdierne være omvendte og kan derfor muligvis skulle vendes.



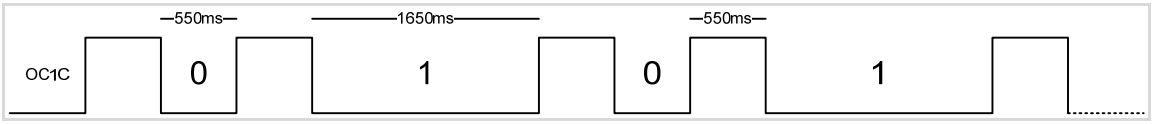
Figur 5 - Illustration af kodningstyper pulse og space

Der eksisterer selvfølgelig flere kodningstyper og varianter, men de hernævnte dækker en stor del af de mest almindelige apparater.

Bølgelængder for 0 og 1

For at de elektriske apparater skal kunne modtage de korrekte data, skal bit værdierne 0 og 1 afsendes med en specifik puls- eller spacelængde. Det er derfor nødvendigt, at der i konfigurationen angives længden af tidsperioden som det digitale signal skal trækkes højt eller lavt, for at afspejle den tilsigtede værdi.

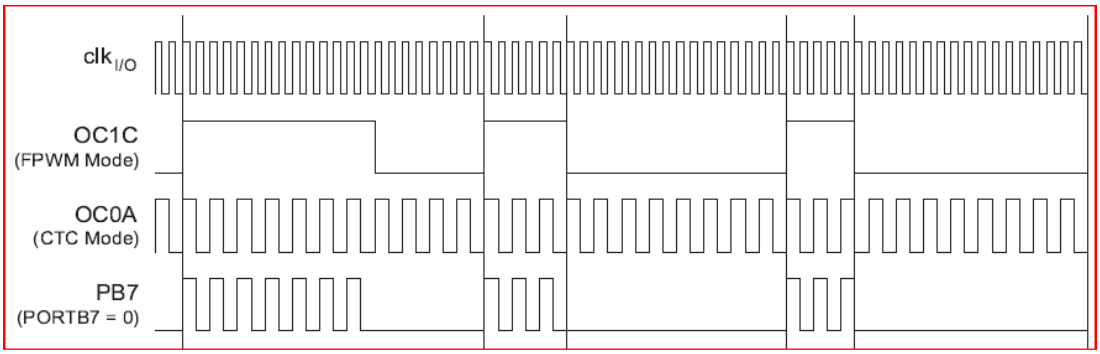
Eksempelvis vil IR-koden ’0101’ konfigureret med værdierne ’0’=550ms, ’1’=1650ms og spaceencoded=550ms generere følgende:



Figur 6 - Illustration af anvendelse for timingværdier

Frekvens af bærebølge

Bærebølger anvendes for at modulere signalet og dermed frafiltrere støj fra dagslys og andre lyskilder. Som standard ligger frekvensen på 38Khz, men kan dog variere noget.
I fjernbetjeningen benyttes bærebølgen sammen med førnævnte bølgelængder af bitværdierne til at generere det endelige IR-signal ud fra IR-koden. Dette fungerer ved at sammenligne et timerregister(bærebølgen) med et register indeholdende IR-koden. Sammenligningen af de to registre sker vha. en effektiv logisk AND-kreds, som findes i visse microcontrollers.



Figur 7 - Illustration af signal-modulering

Øverst i figuren ses microcontrollerens clock (*"clk"*), derunder ses IR-koden i registret *"OC1C"*, som er et register specifikt beregnet til sammenligning af data med timerregistret kaldet *"OC0A"*. Outputtet, som i denne microcontroller sendes til porten *"PB7"*, er nu det modulerede digitale signal, som ønskes sendt via IR-dioden.

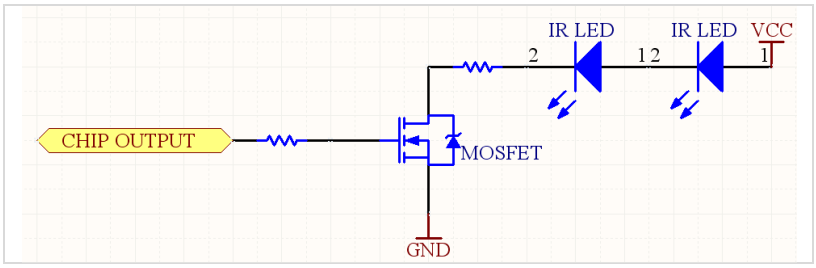
Størrelse af IR-koden

IR-koder kan i princippet fylde alt lige fra 1 til et uendeligt antal bits, men normalen ligger dog omkring de 32bit for de mest anvendte kodningstyper. Ved håndtering af kodningstypen rawcode er størrelsen dog oppe omkring 1Kbit inkl. fyldkode.

Med bl.a. disse informationer kan fjernbetjeningen nu konfigureres til apparatet og IR-koderne kan nu sendes af sted. Følgende er et eksempel på en 32bit IR-kode som kan tænde projektoren MP8620 fra producenten 3M:

STANDBY/ON:	0x0AF5E817

Det modulerede signal sendes gennem et output på microcontrolleren, som er forbundet til en mosfet-transistor med en ekstern spændingskilde. Output-signalet fra microcontrolleren benyttes da til at åbne mosfetten, så kredsløbet lukkes gennem de forbundne dioder.



Figur 8 - Illustration af behandling af chip output

På denne måde genereres og afsendes det infrarøde signal, som styrer det elektriske apparat. Andre fjernbetjeninger kan fungere anderledes, men de elementære funktioner for dannelse af signalet er i store træk ens.

2.1.2. LIRC-filerne

Cepa har valgt at anvende filer fra open source projektet LIRC, som er en applikation til fjernbetjening af apparater via PC. På LIRCs hjemmeside kan man finde flere tusinde brugergenererede filer, som frit kan downloades og anvendes til projekter. At filerne er brugergenererede, betyder, at standarden er åben og at der frit kan arbejdes med indholdet. Det betyder dog også, at der kan forekomme fejl i stavning såvel som syntaks, da ingen af delene nødvendigvis tjekkes før frigivelse.

Filernes syntaks opdeler en fjernbetjening i konfigurationsdata og IR-data. Konfigurationsdata findes mellem de to tags *"begin remote*"-"end remote" og inkluderer bl.a. de tidligere beskrevede timingværdier, størrelser og kodningstype, men ofte også produktnavnet på fjernbetjeningen. IR-Data består af navnet på en til flere knapper, samt ir-koderne for disse og findes mellem de to tags *"begin codes"-"end codes".*
Følgende viser en kraftigt redigeret udgave af lirc-filen " RM-470.txt", som tilhører en Sony cd-afspiller. Den uredigerede version kan ses i "Bilag 1: LIRC-fil "RM-470.txt"".

```
# this config file was automatically generated
# using lirc-0.6.5(any) on Thu Aug 29 23:05:05 2002
begin remote
  name  CD
  bits            7
  flags SPACE_ENC|CONST_LENGTH
  header       2412   588
  one           612   588
  zero         1210   588
  begin codes
          PLAY                     0x0000000000000059
          STOP                     0x0000000000000071
          PAUSE                    0x0000000000000031
  end codes
end remote
```

Som det kan ses fremgår navnet på fjernbetjeningen blot som "name CD", hvilket ikke gør det lettere at bedømme, hvorvidt man har fat i den fil der passer til ens produkt. Dette er den risikable side ved anvendelsen af brugergenererede filer, brugeren skal nemlig selv skrive, hvilket produkt filen hjælper med at emulere, og dette bliver i flere tilfælde gjort sparsomt. For at håndtere denne problemstilling, er det nødvendigt at kunne aflæse filnavnet, samt navnet på mappen den ligger i. Filerne er nemlig placeret i et struktureret mappesystem opdelt efter producent, og hver fil i disse mapper har et navn ofte svarende til produktnavnet på fjernbetjeningen. Resten af de ønskede konfigurationsdata, er opstillet linie for linie som henholdsvis en til to talværdier eller tekststrenge.

Cepas prototype af fjernbetjeningen understøtter pt. kun en begrænset mængde af de konfigurationsdata som kan findes i lirc-filen, men med planer om fremtidige udvidelser, ønsker Cepa størstedelen af de mest almindelige konfigurationsdata gjort tilgængelige gennem CepaIR. Følgende data ønskes som minimum gjort tilgængelige:

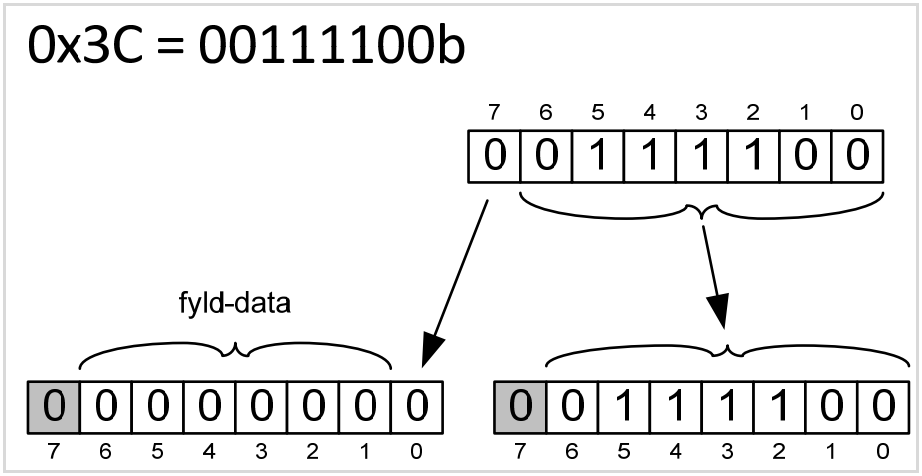
- Flags
- One (begge værdier)
- Zero (begge værdier)
- Repeat (begge værdier)
- Pre_Data
- Post_Data
- Bits
- Header (begge værdier)
- Ptrail
- Gap
- Toggle_bit
- Frequency
- Repeat_Bit

Med ovenstående konfigurationsdata samt specifikke IR-koder, vil det være muligt at emulere et bredt udvalg af fjernbetjeninger med Cepas hardwarebaserede multifjernbetjening.

2.1.3. Protokol og datapakker

CepaCom kommunikerer trådløst via bluetooth med com-boksen for at modtage brugerhændelser. Funktionaliteten som muliggør dette på softwaresiden, har Cepa valgt at købe sig til gennem softwarefirmaet ved navn Franson[7]. Franson specialiserer sig i udviklingen af API'er og leverer en samling licenserede assemblies, som gør programmørens arbejde med bluetooth gennemsigtigt. Cepa har derfor kun skulle udvikle det øverste applikationslag af protokollen, som varetager oprettelse af forbindelse, genetablering af forbindelse, samt modtagelse og udpakning af data, etc..
CepaCom har indtil videre kun skulle kunne modtage data fra com-boksen og protokollens applikationslag og mangler derfor funktionen til at pakke og afsende data.
Cepa benytter sig af en selvudviklet type datapakker, hvor mest betydende bit (MSB) indikerer, hvorvidt der er tale om en operationskode ('1' = opcode) eller '0' = data. Af den grund skal data altid pakkes i størrelser af 7bit, og det skal sikres, at MSB sættes til '0'.

Afsending af en byte data vil derfor kræve afsending af to byte, hvis det skal sikres at, MSB altid er '0'. Følgende eksempel illustrerer tilfældet, hvor en byte med værdien 0x3C ønskes afsendt.



Figur 9 - Illustration af pakkestruktur

Ved afsending af opcodes sættes MSB til '1', dog vil dette ikke skulle sikres funktionelt, da alle opcodes er konstanter og dermed skræddersyede til formålet.

[7] http://www.franson.com (2007)

2.1.4. Microsoft .NET

Cepa har gennem længere tid udviklet deres software ved brug af .NET og det har derfor været et krav fra deres side, at projektet udvikles i samme, for at gøre den evt. videreudvikling lettere. Microsoft .NET er et stykke software, som gør det muligt for applikationer skrevet i et .NET understøttet sprog at kommunikere med det underliggende operativsystem. Microsoft .NET er et direkte modspil til Sun's Java, så hvor JAVA anvender Java Virtual Machine (JVM) til at afvikle applikationer, anvender Microsoft .NET Common Language Runtime (CLR). CLR håndterer applikationernes tildeling af hukommelse samt objekternes levetid via en garbagecollector, og det er derfor som programmør ikke nødvendigt at håndtere oprydning.

Microsoft .NET Framework

Med til .NET hører et omfattende framework, der er skabt af Microsoft for at gøre programmørens arbejde lettere. Microsoft .NET frameworket er en hierarkisk opbygning af bibliotheksklasser, som giver programmøren mulighed for at oprette og nedarve fra basale klasser og derigennem anvende deres funktionalitet. Bibliotheksklasserne er opdelt i en struktur kaldet namespaces. Hvert namespace dækker over sin egen kategori af funktionaliteter, så som håndtering af simple datatyper, håndtering af komplicerede matematikbegreber eller håndtering af den grafiske brugerflade, som gør brugeren i stand til at interagere med softwaren. Frameworket er kompatibelt med flere kendte programmeringssprog herunder Visual Basic og C#, hvoraf den sidstnævnte blev udviklet specifikt til .NET.

Microsoft .NET Compact Framework

Microsoft .NET Compact Framwork (CF) er et framework specielt fremstillet til mobile platforme så som telefoner og PDAer. CF giver adgang til et stort udvalg af klasser også fundet i standard frameworket, men er pga. de ofte få resurser til rådighed på mobile enheder begrænset til et færre antal klasser, metoder og events. Eksempelvis kan nævnes metoder til håndtering af XML og webbaserede løsninger. CF er ligesom standard frameworket opbygget hierarkisk og anvender ofte de samme namespaces til gruppering af bibliotheksklasserne.

2.2. Use cases for CepaIR

I dette afsnit af analysen benyttes i bredt omfang use cases, da de er særdeles velegnede til at beskrive forretningsprocesser og anvendelsesområder[8]. Use cases fungerer ofte godt som kommunikationsmetode, da personer uden kendskab til fagtermerne, kan forstå dem og relatere dem til det ønskede produkt. Use cases benyttes derfor tit som en opfølgende kontrakt direkte baseret på kravspecifikationen.

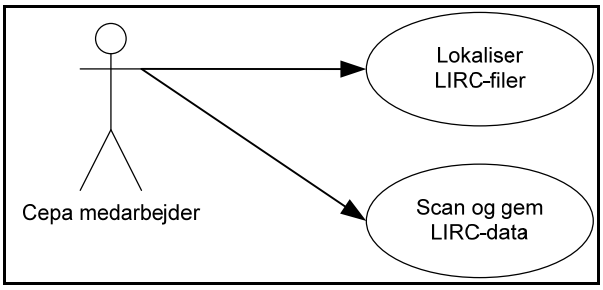
Use cases i dette afsnit er illustrerede i use case diagrammer direkte efterfulgt af en uddybende beskrivelse. De udarbejdede use cases er holdt på et aktør-orienteret niveau, da den underliggende funktionalitet af systemet ikke er fastlagt for prototypen gennem kravspecifikationen. Yderligere er CepaIR-analysen opdelt i tre afsnit for bedre at kunne afspejle systemafgrænsningerne for de tre applikationer under CepaIR projektet.

2.2.1. Use cases for PC-applikationen

Denne applikation har til formål at muliggøre tilgang til LIRC-data uden brug af selve LIRC-filerne i de efterfølgende applikationer. PC-applikationen er oplagt som indfaldsvinkel, da det er her, den røde tråd starter.

Use case-diagram

Nedenstående use case diagram illustrerer PC applikationens anvendelsesområde i samspil med aktøren.



Figur 10 - Use case-diagram for PC-applikation

Aktørbeskrivelse: Cepa medarbejder

Den primære og eneste aktør i denne applikation er Cepa medarbejderen, som har adgang til samlingen af LIRC-filer. Denne aktørs pligt er at holde sig opdateret omkring ændringer i LIRC-filernes format såsom justeringer i syntaks eller formulering. Aktøren bør derfor være en ansat hos Cepa med tilknytning eller kendskab til CepaIR-projektet eller den udviklede prototype af hardwarefjernbetjeningen.

[8] Alistair Cockburn, "Writing Effective Use Cases", Addison-Wesley (2002)

Use Case-beskrivelse: Lokalisér LIRC-filer

Som tidligere nævnt har Cepa valgt at anvende en downloadet filsamling fra open source-projektet LIRC. Aktøren skal informere systemet, om hvor filerne er placeret på PC'en, før at filerne kan scannes. Systemet vil til gengæld for denne oplysning kvittere med et udfald af resultatet. Denne use case er nødvendig for at kunne udtrække oplysninger og koder fra filerne.

ID	Lokaliser LIRC-filer
Formål	At lokaliserer samtlige potentielle LIRC-filer i en specificeret mappe på PCens harddisk og meddele aktøren om udfaldet.
Primær aktør	Cepa medarbejder
Motiver og interesser	At undersøge forekomsten af potentielle LIRC-filer i en given mappe.
Trigger	Brugerstyret kontrolelement
Forudsætninger	At en eller flere LIRC-filer er blevet downloadet og gjort tilgængelige på PCens harddisk.
Succeskriterier	Lagring af forekomster for den givne mappe.
Fejlbetingelser	Den givne mappe er ikke tilstede eller utilgængelig.
Hovedforløb	1. Aktøren vælger en mappe 2. Mappen og undermapper gennemløbes af systemet og evt. forekomster registreres og lagres. 3. Aktøren underrettes om forekomsterne.
Udvidelser	• Valg af enkelte filer • Integritet af filer undersøges før lagring
Hyppighed	Udføres med mellemrum ved tilføjelse af flere LIRC-filer .

Som antydet i beskrivelsen udføres use casen ikke så ofte, hvilket er fælles for samtlige use cases under PC-applikationen. Lokalisering af LIRC-filerne kan forekomme hyppigere end resten, da aktøren ikke nødvendigvis behøver at vælge rigtigt i første forsøg.

Use Case-beskrivelse: Scan og gem LIRC-data

Scanningen af filerne har et primært formål, nemlig at identificere og udtrække data for at gemme dem et andet sted. Scanningen medfører selvklart, at aktøren har lokaliseret filsamlingen via systemet, og at systemet har skriveadgang til det pågældende sted, hvor dataene skal lagres. Dette er hovedfunktionen for PC-applikationen og den endelige løsning for kravspecifikationens punkt 1.

ID	Scan og gem LIRC-data
Formål	At gennemlæse lagrede forekomster og udtrække/gemme nødvendige data i en database.
Primær aktør	Cepa medarbejder
Motiver og interesser	At tilføje nye data til databasen for at øge kompatibilitet.
Trigger	Brugerstyret kontrolelement
Forudsætninger	Udførsel af use case "Lokaliser LIRC-filer"
Succeskriterier	Tilføjelse af nye og valide LIRC-data til databasen.
Fejlbetingelser	Manglende forbindelse til databasen
Hovedforløb	1. Aktøren vælger at scanne og gemme LIRC-data

	2. Systemet scanner LIRC-filen og gemmer LIRC-data i databasen 3. Systemet registrerer udfald (tæller op) *Indtil der ikke er flere lagrede forekomster gentages trin 2 og 3* 4. Aktøren underettes om udvidelsen af databasen
Udvidelser	• Lokal lagring ved manglende databaseforbindelse • Manuel tilføjelse af nye data
Hyppighed	Udføres med mellemrum indtil en tilstrækkelig kompatibilitet er opnået

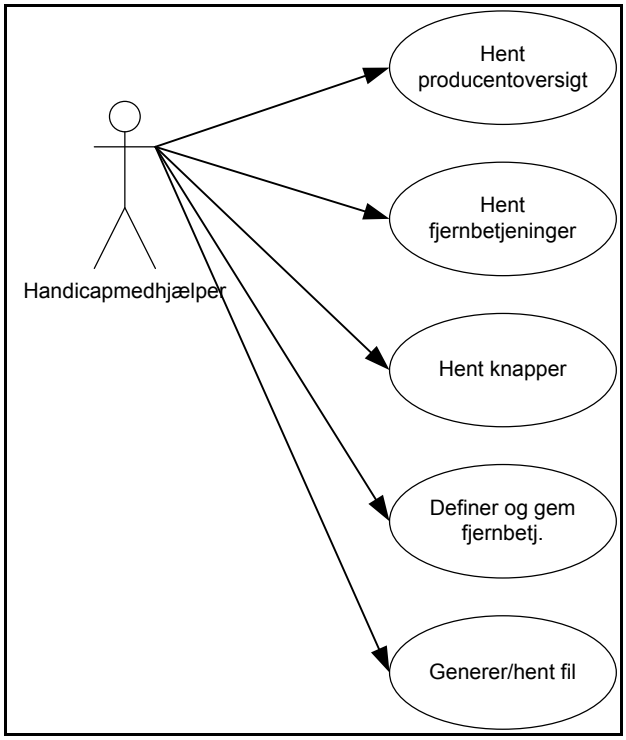
Ikke alle data kan betragtes som valide, da Cepas hardwarefjernbetjening ikke understøtter alle dataformater og kodningstyper. En form for sortering og validering vil derfor være nødvendig før lagring.

2.2.2. Use cases for server-applikationen

Denne applikation skal fungere som en simplificeret webfront til de nu tilgængelige LIRC-data. Applikationen vil gøre det muligt for en simpel computerbruger at sammensætte og hente en konfigurationsfil indeholdende informationer om en til flere fjernbetjeninger. Dette er det mest naturlige næste skridt i den røde tråd.

Use case-diagram

Nedenstående use case-diagram afspejler aktørens tilgang samt applikationens anvendelsesområde og afgrænsninger.



Figur 11 - Use case-diagram for server-applikation

Aktørbeskrivelse: Handicapmedhjælper

Denne aktør kan som udgangspunkt være hvem som helst pånær den bevægelseshæmmede selv. Aktøren kan i visse tilfælde være Cepa-medarbejderen, hvis servicen er forudbestilt, men vil som regel være en ansat medhjælper eller et assisterende familimedlem. Aktøren bør have et basalt kendskab til brugen af PC og internet samt have adgang til eller en liste over de elektriske apparater, som ønskes understøttet.

I de følgende use cases benyttes til tider udtrykket ”indkøbskurven”. Dette udtryk er en midlertidigt term, som blot skal indikere, at aktøren gemmer sit arbejde til senere brug.

Use Case-beskrivelse: Hent producentoversigt

Da det er sandsynligt, at den lettest tilgængelige information om produktet er producenten, er dette den bedste tilgang til indhentning af produktets data. Dette er det første af fire lette trin nødvendige i processen for identificering og konfiguration af en fjernbetjening.

ID	Hent producentoversigt
Formål	At få vist en liste af alle producenter så det bliver muligt at specificere sig frem til et enkelt produkt
Primær aktør	Handicapmedhjælper
Motiver og interesser	At finde en specifik producent af et produkt
Trigger	Åbning af hjemmeside
Forudsætninger	Adgang til internettet
Succeskriterier	At få vist en liste over de tilgængelige producenter
Fejlbetingelser	• Manglende internetforbindelse • Manglende databaseforbindelse
Hovedforløb	1. Aktøren indtaster hjemmesidens adresse i browseren 2. Systemet kalder databasen og viser listen over producenter
Udvidelser	
Hyppighed	Anvendes hver gang en fjernbetjening skal tilføjes ”indkøbskurven”

Da flere af de viste use cases har næsten identiske motiver, udvidelser og aktører, vil use case beskrivelser for ”Hent fjernbetjeninger” og ”Hent knapper” kunne findes i ”Bilag 2: Use Case Beskrivelse: Hent fjernbetjeninger” og ”Bilag 3: Use Case Beskrivelse: Hent knapper”. Kort fortalt beskriver de blot, hvordan aktøren anvender allerede hentede data til at finde og hente flere. De to frasorterede use cases udgør trin 2 og 3 i processen for identificering og konfiguration af en fjernbetjening.

De følgende to use case beskrivelser er beholdt i rapporten, da deres funktionalitet er essentiel for bedst mulig udnyttelse af applikationen. Under beskrivelsen af LIRC-filen blev det nævnt, at navnet på fjernbetjeningen i filen ikke altid var tydeligt og ofte blot et produktid. Det er derfor en nødvendighed for brugervenlighedens skyld, at brugerene af det endelige produkt selv har indflydelse på navngivningen af deres konfigurerede fjernbetjeninger. På denne måde undgås det, at slutbrugeren skal huske tilhørsforhold på navne som ” RM-470” eller ”RM-570”, for at de kan betjene deres apparater.

Use Case-beskrivelse: Definér og gem fjernbetjening

Da det skal være muligt at konfigurere mere end én fjernbetjening, er det nødvendigt for aktøren at kunne gemme sit arbejde. Aktøren skal derfor efter navngivning/definering af den konfigurerede fjernbetjening sørge for at gemme den i systemet. Dette er sidste trin i processen for identificering og konfiguration af en fjernbetjening og ligeledes løsningen til kravet om få brugervenlige trin samt oprettelse af en samling af fjernbetjeninger.

ID	Definer og gem fjernbetjening
Formål	At tilpasse udseende og funktionalitet af fjernbetj., så den bedst muligt svarer til brugerens behov og efterfølgende gemme denne.

Primær aktør	Handicapmedhjælper
Motiver og interesser	At gøre brugen af den endelige fjernbetj. så brugervenlig som muligt og gemme den til senere brug
Trigger	Brugerstyrede kontrolelementer
Forudsætninger	• Udførsel af use casen ”Hent knapper” • Kendskab til apparatets type og placering eller slutbrugerens personlige ønske
Succeskriterier	At få tilpasset fjernbetj. til brugerens specifikke behov og gemt denne til senere anvendelse
Fejlbetingelser	• Manglende internetforbindelse • Manglende databaseforbindelse
Hovedforløb	1. Aktøren markerer en eller flere af knapperne tilhørende den valgte fjernbetj. 2. Aktøren vælger en sigende beskrivelse for den valgte fjernbetj. (f.eks ”TV Stue”) 3. Aktøren vælger at tilføje fjernbetj. til ”indkøbskurven” 4. Systemet henter konfigurationsdata fra databasen og lægger fjernbetj. og knapper i ”indkøbskurven”
Udvidelser	• Tilføjelse af ikoner for individuelle knapper • Tilføjelse af egne udviklede knapper og koder
Hyppighed	Anvendes hver gang en fjernbetjening skal tilføjes ”indkøbskurven”

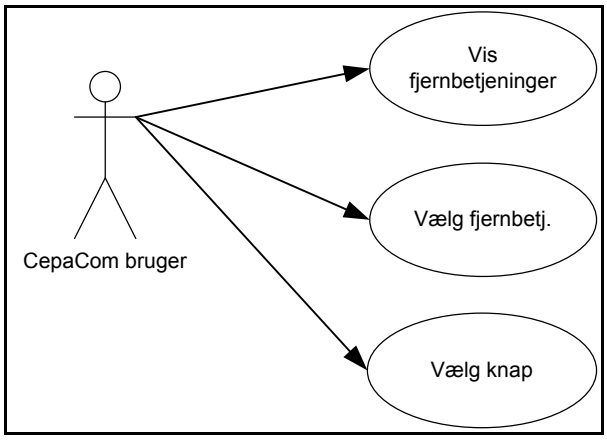
Følgende use case er sidste trin i processen for generering af en fil indeholdende de konfigurerede fjernbetjeninger. Aktøren beder her systemet generere og sende hans arbejde i form af en fil, som genereres ud fra samlingen. I modsætning til tidligere use cases i scenariet udføres denne kun en enkelt gang nemlig ved afslutning af konfigureringsforløbet. Aktøren skal selv sørge for, at filen bliver gemt det korrekte sted - med det samme eller senere.

Use Case Beskrivelse: Generer/hent fil

ID	**Generer/hent fil**
Formål	At kunne hente en fil med alle konfigurerede fjernbetjeninger
Primær aktør	Handicapmedhjælper
Motiver og interesser	At gøre brugen og konfigurering af fjernbetj. fleksibel
Trigger	Brugerstyrede kontrolelementer
Forudsætninger	Udførsel af use casen ”Definer og gem fjernbetjening”
Succeskriterier	At downloade en enkelt fil indeholdende alle konfigurerede fjernbetjeninger
Fejlbetingelser	Manglende internetforbindelse
Hovedforløb	1. Aktøren vælger at hente indhold af ”indkøbskurv” 2. Systemet genererer filen ud fra indholdet 3. Aktøren specificerer placeringen af den genererede fil
Udvidelser	Integritetstjek af filindhold
Hyppighed	Anvendes kun ved endelig afslutning af konfigurationsforløbet

2.2.3. Use cases for PDA-applikationen

Formålet med denne applikation er at anvende filen, som blev konstrueret via server-applikationen. PDA-applikationen vil med disse data gøre det muligt at afsende konfigurations- og IR-data til com-boksen via en leveret bluetooth-forbindelse. Det kan være svært at sammenholde denne applikations use cases med kravspecifikationen, da den detaljerede funktionalitet af denne er en direkte effekt af udarbejdelsen af de to andre.



Figur 12 - Use case-diagram for PDA-applikation

Aktørbeskrivelse: CepaCom bruger

Denne aktør er oftest en stærkt bevægelseshæmmet person, der har Cepas produkt monteret på sin kørestol. Aktøren er stærkt motiveret for at lære nye systemer grundet den medfølgende frihed, og computer applikationer er derfor en stor del af hverdagen. Aktøren bør være i stand til at betjene CepaCom ved brug af I/O-kontakterne og bør have den endelige fjernbetjening tilsluttet sin com-boks.

Use Case-beskrivelse: Vis fjernbetjeninger

Denne use case er nødvendig, da brugeren kan have flere apparater i hjemmet og derfor har flere fjernbetjeningers koder indlagt på PDA'en.

ID	Vis fjernbetjeninger
Formål	At få adgang til samtlige fjernbetjeninger
Primær aktør	CepaCom-bruger
Motiver og interesser	At afgive en specifik kommando til et elektronisk apparat via IR
Trigger	Opstart af CepaIR-modulet under CepaCom
Forudsætninger	Eksistensen af overført fjernbetjeningsfil med LIRC-data
Succeskriterier	At få vist en oversigt af samtlige fjernbetjeninger
Fejlbetingelser	• Manglende adgang til fjernbetjeningsfil • Manglende indhold i fjernbetjeningsfil
Hovedforløb	Systemet henter fjernbetjeningsoversigt
Udvidelser	Udvikling af GUI til samspil med CepaComs menusystem
Hyppighed	Anvendes ofte da omverdenskontrol forekommer dagligt

Følgende use case vil drage fordel af det brugervalgte navn for hver fjernbetjening. I denne use case er det nemlig disse navne, som aktøren skal vælge mellem for at få adgang til tilhørende knapper. Dette lader sig gøre, da listen hentet i forrige use case udgør en oversigt af samtlige fjernbetjeninger i filen på PDA'en.

Use Case-beskrivelse: Vælg fjernbetjening

ID	Vælg fjernbetjening
Formål	At få adgang til knapper tilhørende en bestemt fjernbetj.
Primær aktør	CepaCom-bruger
Motiver og interesser	At afgive en specifik kommando til et elektronisk apparat med IR
Trigger	Brugerstyret kontrolelement
Forudsætninger	Udførsel af use casen "Vis fjernbetjeninger"
Succeskriterier	At få vist en oversigt af knapperne for en fjernbetjening
Fejlbetingelser	• Manglende adgang til fjernbetjeningsfil • Manglende indhold i fjernbetjeningsfil
Hovedforløb	1. Aktøren vælger den ønskede fjernbetj. 2. Systemet henter knapper til den valgte fjernbetj.
Udvidelser	Udvikling af GUI til samspil med CepaCom menusystemet
Hyppighed	Anvendes ofte da omverdenskontrol forekommer dagligt

Den sidste use case for hele CepaIR-projektet kan kortest beskrives som trykket på fjernbetjeningsknappen. Gennem de fleste af de beskrevne use cases har det nok været lettest for læseren at forestille sig hver use case som en knap eller et andet grafisk kontrolelement. I dette tilfælde kan der ikke være tvivl. For bedst muligt at emulere en fjernbetjening må man give brugeren det indtryk, at han sidder med en nærmest normal fjernbetjening ved hånden. Dette gøres med knapper.

Use Case-beskrivelse: Vælg knap

ID	Vælg knap
Formål	At afsende en knaps IR-kode
Primær aktør	CepaCom-bruger
Motiver og interesser	At afgive en specifik kommando til et elektronisk apparat med IR
Trigger	Brugerstyret kontrolelement
Forudsætninger	Udførsel af use casen "Vælg fjernbetjening"
Succeskriterier	At kunne foretage omverdenskontrol via IR-kode
Fejlbetingelser	• Manglende forbindelse til com-boks eller hardwarefjernbetj. • Fejlagtig IR-kode
Hovedforløb	1. Aktøren vælger den ønskede knap 2. Systemet afsender IR- og konfigurations-data til com-boksen
Udvidelser	Udvikling af GUI til samspil med CepaCom-menusystemet
Hyppighed	Anvendes ofte da omverdenskontrol forekommer dagligt

2.3. Konklusion af analyse for CepaIR

Analysekapitlet har givet et godt overblik over miljøet, hvori softwareløsningen CepaIR skal fungere. De udarbejdede use cases vil senere i udviklingsforløbet vise sig nyttige som uddybende referencer til kravspecifikationen. Med dette menes, at de udarbejdede use cases anvendes som byggeklodser på et fundament bestående af kravspecifikationen.
Der er ikke mulighed for den store indflydelse på eksisterende hardware og software, men Cepa har dog erkendt, at den eksisterende software bør være mere modulbaseret f.eks. via flere assemblies.

Teknologi og formater: Kravet omkring generel udvikling i .NET bør ikke blive et problem trods det noget mindre Compact Framework. Microsoft .NET CF er efterhånden så omfattende og udbredt, at flere slanke løsninger stilles til rådighed på nettet for de endnu frasorterede klasser. Der vil dog muligvis skulle tages hensyn til de begrænsede metoder vedrørende parsing af XML. Analysen af formaterne har dog også vist, at der eksisterede nogle svagheder i forbindelse med de brugergenererede LIRC-filer. Den første svaghed relaterede sig til de ringe navngivninger i filerne, som gør det svært for slutbrugeren af PDA-applikationen at vælge den rigtige fjernbetjening når ønsket. Den anden svaghed omkring LIRC-filerne var fejl i formatets syntaks, hvilket må skulle håndteres via designet af løsningsmodellen.

PC-applikationen: Der er gjort rede for PC-applikationens anvendelsesområde og der er ikke under udarbejdelsen konstateret nogen større forhindringer ud over den tidligere nævnte afhængighed af de brugergenererede LIRC-filer, som kan indeholde ugyldig syntaks. Der blev under analysen identificeret to use cases: ”Lokalisér LIRC-filer” og ”Scan og gem LIRC-data”.

Server-applikationen: Ud fra analysen af server-applikationen kan det konkluderes, at den optimale brugervenlighed kan opnås ved at lade aktøren udføre en konfiguration af filen i få og enkle trin. Yderligere har analyseforløbet medført, at et enkelt problem er blevet håndteret nemlig problemet vedrørende navngivningen. Dette problem blev løst ved at gøre aktøren ”handicapmedhjælper” ansvarlig for navngivningen af de konfigurerede fjernbetjeninger. Den udarbejdede løsningmodel vil gøre fjernbetjeningen mere intuitiv for brugeren og dermed mere brugervenlig. Under analysen blev fem use cases identificeret: ”Hent producentoversigt”, ”Hent fjernbetjeninger”, ”Hent knapper”, ”Definer og gem fjernbetj.” og ”Generer/hent fil”

PDA-applikationen: Der er under analysen af PDA-applikationen ikke kommet nye konkrete problemer frem i lyset. Det meste af systemets funktionalitet er efterhånden klarlagt, og de sidste detaljer vil sandsynligvis blive udarbejdet i det følgende designafsnit. Der er dog stadig en smule usikkerhed omkring, hvad der kan lade sig gøre rent programmeringsmæssigt, da erfaringen med C# er stærkt begrænset til web og PDA. Analysen har medført identificering af tre use cases: ”Vis fjernbetjeninger”, ”Vælg fjernbetj.” og ”Vælg knap”.

Ud over de allerede nævnte har de beskrevne use cases ikke afdækket større flaskehalse og samtlige krav synes at være opfyldt med den nuværende beskrivelse af systemet. Det er derfor nu muligt at påbegynde designet af løsningsmodellerne for CepaIR.

3. Design af CepaIR

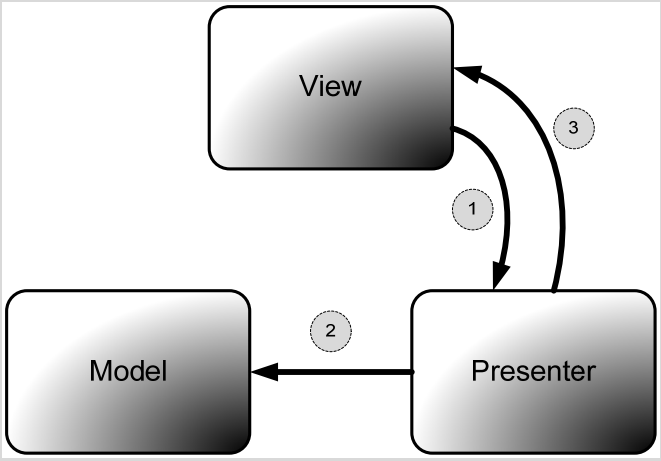
Designfasen for CepaIR har til formål at designe konkrete løsningsmodeller ud fra problemstillingerne beskrevet i analysen via use cases. Der vil i denne fase skulle tages hensyn til brugen af teknologier og formater, hvis anvendelse er krævet i kravspecifikationen samt beskrevet under analysen. I starten af kapitlet vil overvejelserne blive beskrevet omkring brugen af patterns, for bedre at kunne sikre en lav kobling og høj samhørighed. Designet vil blive udarbejdet som overordnet, så der evt. kan gøres plads til kreative indfald ved implememtering.

3.1. Patterns

Brugen af patterns skal betragtes som en guide til god programmørskik nærmere end en regel. Patterns bygger på erfaringer af mere erfarne udviklere, som har valgt at navngive og videregive visse løsningsmodeller til udviklingssamfundet. Nogle patterns bliver med tiden til gode vaner, mens andre forsvinder til fordel for nye eller aldrig vinder indpas. Med udgangspunkt i de gode vaner kan der allerede nu træffes beslutning om brugen af to typer patterns nemlig samspil mellem bruger og applikation og tilgangen til data uden for den nære hukommelse.

3.1.1. Model View Presenter

For applikationerne i CepaIR vil der hele vejen igennem blive anvendt et pattern kaldet Model View Presenter[9] (MVP). MVP har til formål at nedsætte mængden af kobling mellem den grafiske brugerflade (GUI) og resten af applikationen herunder businessmodellen (fremover blot omtalt som "modellen"). MVP er en videreudvikling af det tidligere meget anvendte "model-view-controller"(MVC), men er efter større debat i en udviklerkreds tilknyttet MVP, blevet opdelt i to separate patterns kaldet "Supervising Controller" og "Passive View". Grunden til, at dette pattern i rapporten beskrives som MVP er, at opdelingen endnu ikke er bredt kendt og stadig omtales som MVP.



Figur 13 - Illustration af MVP-funktionalitet

Supervising Controller består af to dele, en controller, også kendt som en presenter, og et view, der som oftest er en GUI. Sammenhold følgende forløb med Figur 13.

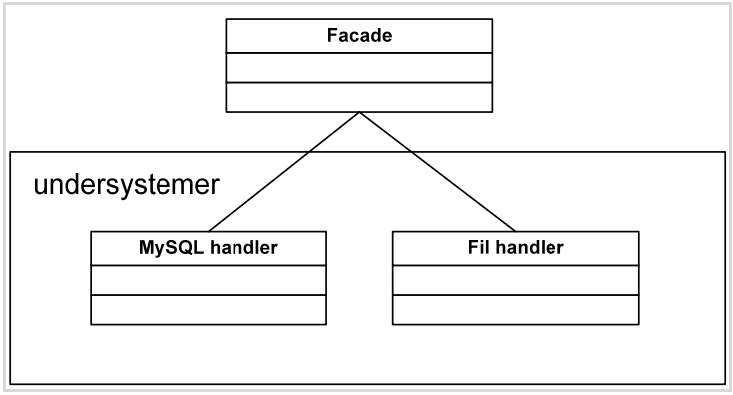
[9] http://martinfowler.com/eaaDev/ModelViewPresenter.html (2006)

(1) Viewet varetager kun simple funktioner og videresender alle brugerinputs i form af events til den underliggende controller, som er blevet tilknyttet viewet i de første faser af dets "livscyklus".
(2) Controlleren benytter så input fra tilsendte brugerevents til at foretage evt. ændringer i modellen eller anden manipulation af data i applikationen.
(3) Hvis synkronisering af model og view er nødvendig efter ændringer i modellen opdateres disse af viewets controller gennem et interface fungerende som gateway. Dette er kendt som Passive View, da viewet ikke selv kender til datakilder og heller ikke beder om opdatering. Ofte anvendes databinding mellem view og model, men i tilfælde, hvor data skal forbehandles eller genbrugelighed sikres, benyttes Passive View med fordel til opdatering.

På denne måde fungerer de to patterns i et tilstrækkeligt tæt samarbejde uden at kende for meget til hinandens eksistens. Brugen af de to patterns muliggør yderligere, at viewet forholdsvist simpelt kan udskiftes og den tilhørende presenter genanvendes så længe, at det eksisterende interface implementeres af det nye view.

3.1.2. Facade Pattern

Et facade pattern[10] anvendes oftest som adgangsgiver for flere forskellige services til underliggende systemer. I CepaIR vil facade pattern i bred udstrækning blive anvendt som adgangsgiver til eksterne datakilder så som MySQL-server, filer på harddisk og bluetooth-transmission.



Figur 14 - Illustration af facade-pattern

Facaden varetager da samtlige kald til undersystemerne fra de overliggende lag i applikationen herunder kald fra controller/presenters, model og evt. undersystemerne imellem. Dette sikrer bl.a., at evt. regler om formatering eller endda forretningsregler kan påtvinges før adgang til undersystemerne. I applikationerne under CepaIR vil facaden hovedsagligt være integreret som et servicelag også kendt som et applikationslag. Facaden vil ikke blive anvendt som en singleton, da read/write adgang efterhånden er godt kontrolleret eksternt for systemet.

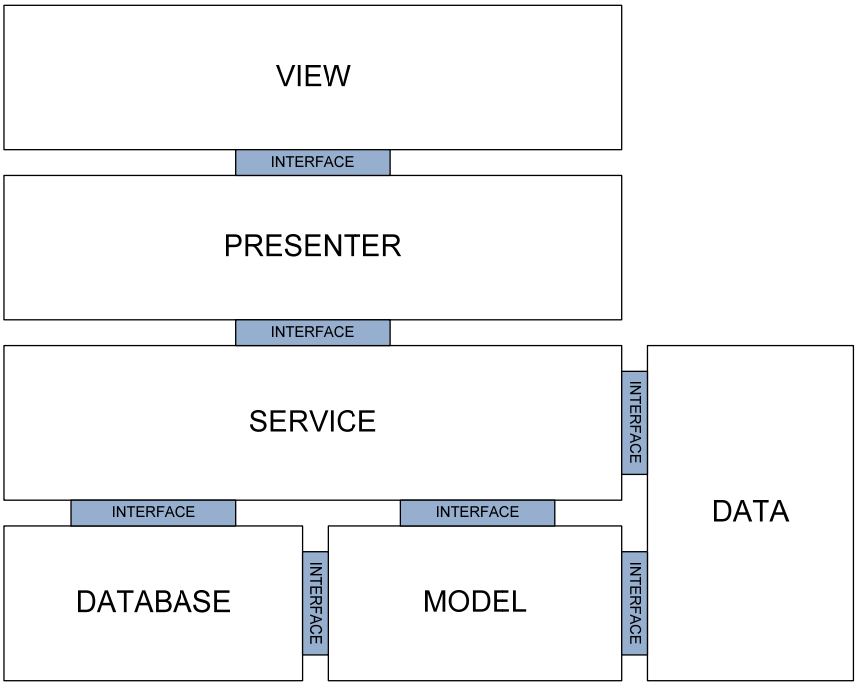
[10] http://www.dofactory.com/Patterns/PatternFacade.aspx (2006)

3.2. Design af PC-applikationen

Før påbegyndelse af det aktuelle design for PC-applikationen er det en fordel at opsummere resultaterne af de tidligere afsnit vedrørende denne. I kravspecifikationen fastsættes, at applikationen skal kunne scanne adskillige LIRC-filer på en harddisk, udtrække specifikke data og gemme dem på en MySQL-server. I analysefasen blev det fastlagt, at brugeren af denne applikation var en Cepa-medarbejder, og at medarbejderen selv skulle sørge for at downloade LIRC-filerne til den lokale PC. Medarbejderen skal så via PC-applikationen kunne lokalisere LIRC-filerne på harddisken og vælge at få dem scannet og gemt. Der skal derfor designes følgende:

- En grafisk brugerflade til interaktion med brugeren
- En presenter til håndtering af events og synkronisering grundet ønske om brug af model-view-presenter
- En fil-søger til lokalisering af LIRC-filerne
- En parser til udtrækning af LIRC-data
- En business-model til strukturering og beskrivelse af objekterne
- En MySQL-database til opbevaring af LIRC-data
- En løsning til forbindelse mellem MySQL-server og PC-applikationen

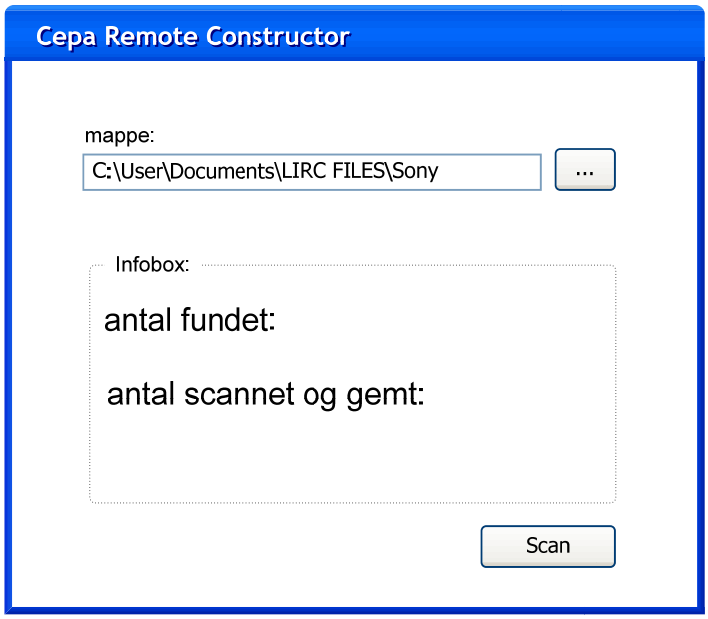
PC-applikationen kan ud fra ovenstående opdeles i 6 lag: View, Presenter, Service, Model, Data og Database som alle vil blive beskrevet nærmere i de følgende afsnit.



Figur 15 - PC-applikationens arkitektur

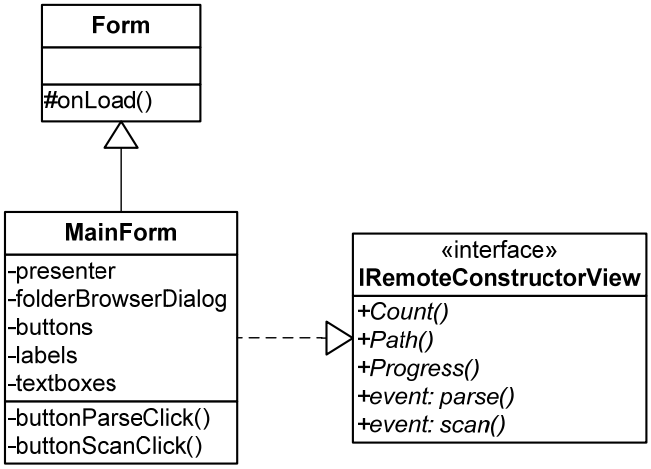
3.2.1. View

GUI'en til PC-applikationen er forholdsvist simpel og kan designes med udgangspunkt i de to use cases "Lokalisér LIRC-filer" og "Scan og gem LIRC-data". De to use cases svarer til to kontrolelementer som i dette tilfælde er knapper placeret i en underliggende container.



Figur 16 - PC-applikationens GUI

Ved brug af .NET's FolderBrowserDialog-klasse er det muligt at lokalisere mappen indeholdende LIRC-filerne ved at trykke på knappen med påskriften "…". Stien til mappen vises efterfølgende i tekstboxen "mappe:" og antallet af potentielle LIRC-filer vises i infoboxen ud for teksten "antal fundet:". Ved at trykke på knappen "Scan" påbegyndes processen med at parse og gemme de fundne LIRC-filer, hvilket afsluttes med at underrette om resultatet ud for teksten "antal scannet og gemt:".

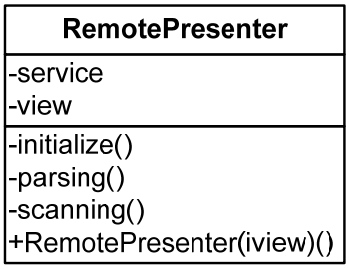


Figur 17 - PC-applikationens view-lag

GUIen nedarver fra .NET's Form-klasse og har derfor den samme livscyklus som en standard-form. Det er derfor muligt at oprette forbindelsen til viewets presenter allerede ved "OnLoad", hvilket er det tidligste tidspunkt i formens levetid, man som udvikler har adgang til. Viewet implementerer yderligere interfacet til kommunikation med presenteren, som ikke kender til anden eksistens af view end dette. Via interfacet implementeres knappernes eventhandlers, parse og scan samt properties til opdatering af attributter.

3.2.2. Presenter

Presenteren som har til formål at håndtere synkronisering af GUI og model, samt håndtering af events, har et yderligere formål i systemet nemlig at oprette forbindelse til service-facaden. Forbindelsen mellem presenter og service foregår via et interface, som implementeres af service-klassen i service-laget.

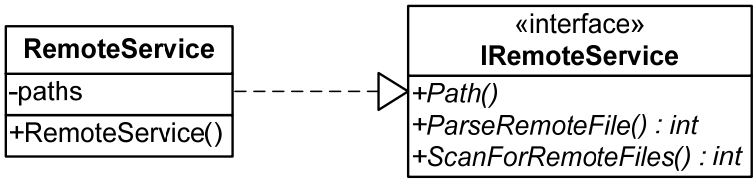


Figur 18 - PC-applikationens presenter-klasse

Presenteren kan efter oprettelse af denne forbindelse foretage metodekald til undersystemerne og stadig referere tilbage og opdatere dens tilknyttede view.

3.2.3. Service

Service-facaden fungerer som en gateway og har kun formålet at hente, hvad den bliver bedt om. Servicen implementerer et interface, som sikrer kommunikationen med Presenteren.

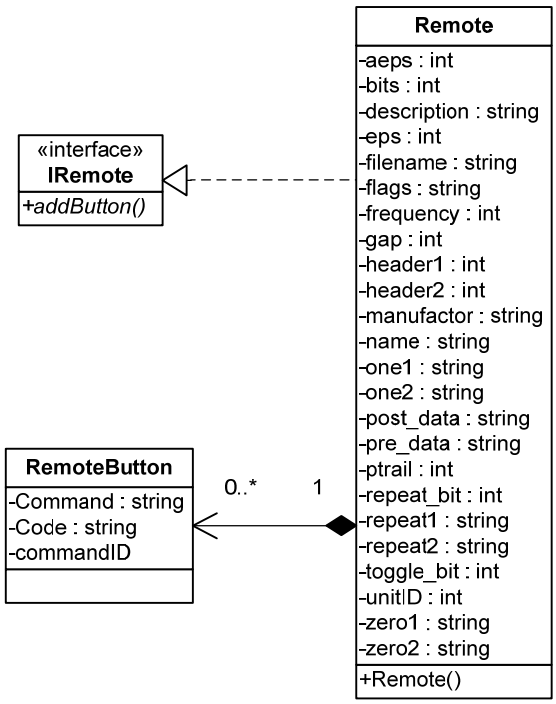


Figur 19 - PC-applikationens service-lag

Da servicen i denne applikation hovedsagligt anvendes til behandling af multiple filer, bliver det dens opgave at gemme listen over stier til potentielle filer, indtil brugeren beslutter sig for at parse dem.

3.2.4. Model

Modellen beskriver fjernbetjeningens opsætning og knapper, som skal udtrækkes af LIRC-filerne. Kommunikation med modellen foregår via et interface, som klassen ”Remote” implementerer.



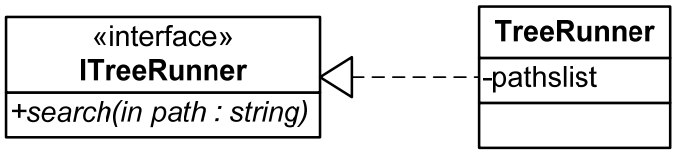
Figur 20 - PC-applikationens model-lag

Remote indeholder samtlige data til konfigurering samt anvendelse af fjernbetjeningen. Flere af attributterne har været nævnt før i analysen af teknologien, men der bør lægges mærke til følgende fire: ”description”, ”filename”, ”manufactor” og ”UnitID”.
I analysen af PC-applikationen blev det opdaget, at LIRC-filerne ikke altid indeholdt navnet på producenten eller fjernbetjeningens navn. Disse oplysninger skulle derfor findes et andet sted for nemmere at kunne identificere fjernbetjeningen senere. Attributterne ”filename” og ”manufactor” er derfor indsat som supplement til attributten ”name”. Attributten ”description” skal anvendes til personlig navngivning af fjernbetjeningen og vil først blive anvendt i web- og PDA-applikationerne. Allerede nu vil den dog blive tilføjet klassen, så modellen kan genbruges fuldt ud. ”UnitID” bliver nødvendig fremover som indekseringsattribut, men vil først blive tildelt af MySQL-serveren, når fjernbetjeningens andre data skal lagres.

RemoteButton består af tre attributter, Command, Code og CommandID. Code er byte-strengen som med tiden bliver til IR-koden og Command er navnet på knappen, som samtidig beskriver kodens funktion. Attributten ”commandID” eksisterer på samme vilkår som ”unitID” og er derfor blot til indekseringsbrug.

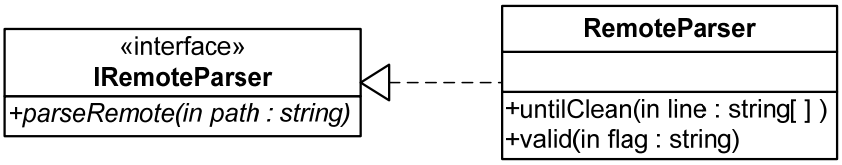
3.2.5. Data

I data-laget placeres klasserne med adgang til filerne på harddisken. Klasserne tilgås via interfaces og altid kun gennem service-laget. Klassen til fil-lokalisering kaldet ”TreeRunner” fungerer ved rekursivt at gennemløbe samtlige undermapper for den angivne sti. I hver mappe søges efter filer med den korrekte filendelse og stien til de fundne filer lagres og returneres endelig til den forespørgende klasse.



Figur 21 - PC-applikationens klasser til filsøgning

Parseren til udtrækning af konfigurationsdata og IR-koder tilgår en enkelt fil ad gangen ud fra en placering leveret som parameter. Parseren gennemlæser filen linie for linie og sammenligner den læste tekst med en liste over ønskede informationer.

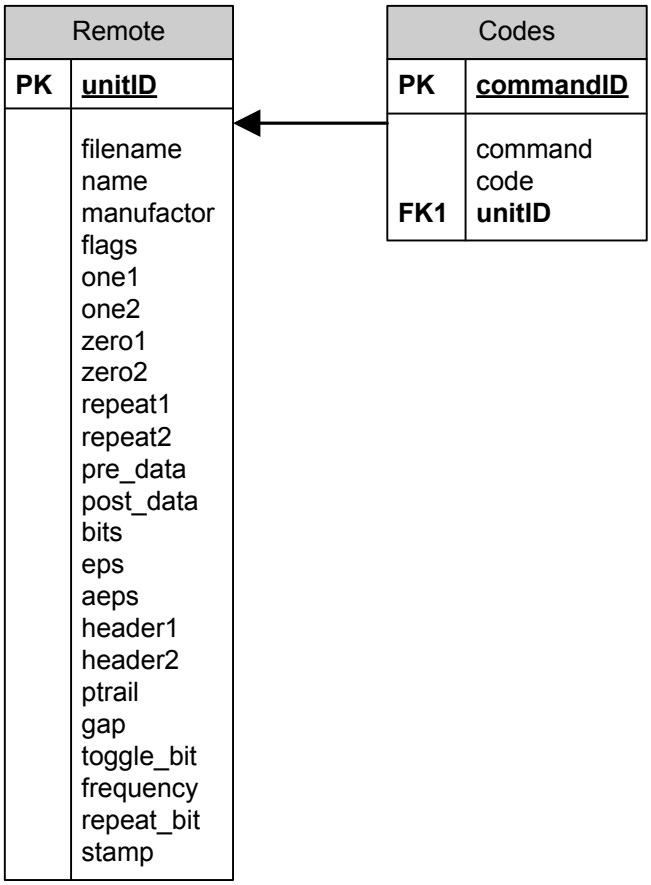


Figur 22 - PC-applikationens klasser til filparsing

Alle linier i filen skal undersøges for fejlagtig formatering og ugyldige karakterer, da dette er den eneste mulighed for håndtering af de brugergenererede fejl i LIRC-filerne. Dette gøres med funktionen ”untilClean”, som læser hver ascii-karakter og evt. udskifter den med en ny. Parseren har yderligere til opgave at vurdere, om indholdet af filen udgør en valid fjernbetjening f.eks. ved at undersøge om kodningstypen er understøttet af hardwarefjernbetjeningen. Dette gøres med funktionen ”valid”, da hardwarefjernbetjeningen ikke understøtter kodningstypen ”raw codes”, og som derfor heller ikke bør understøttes af softwaren.

3.2.6. Database

Før en egentlig beskrivelse af designet for databaselaget skal udarbejdelsen af databasen foretages. Databasens udformning afspejler i høj grad modellen, da det er objekter af denne type som ønskes lagret. Databasen består af de to tabeller Remote og Codes, som indeholder henholdsvis konfigurationsoplysningerne for en fjernbetjening og dens IR-koder. Databasen illustreres bedst med nedenstående relationsdiagram.



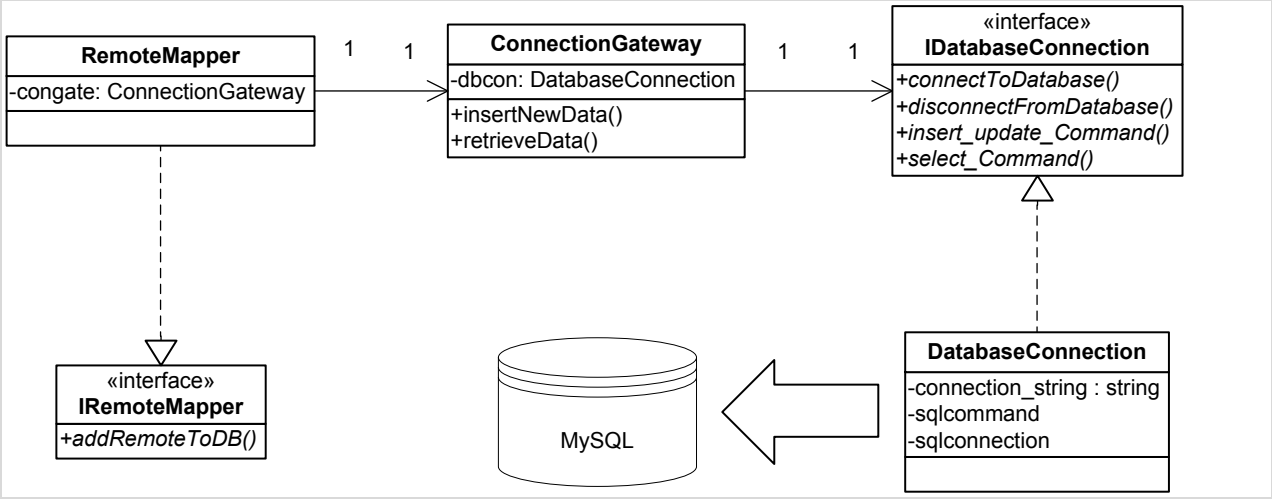
Figur 23 - Relationsdiagram for databasen

Som beskrevet under designet af modellen anvendtes to attributter til indeksering, hvilket i denne database anvendes som primærnøgler. Attributten "unitID" bliver primærnøglen for tabellen "Remote" og fremmednøgle i tabellen "Codes", mens attributten "commandID" bliver primærnøglen for "Codes". Alternativt til primærnøglen "unitID" kunne en kombination af attributterne "manufactor" og "filename" have været anvendt, men da det ønskes[11] at lade MySQL-serveren håndtere nøgle-generering via automatisk optælling, bliver denne løsning fravalgt. Nederst i tabellen "Remote" findes attributten "stamp", som er endnu en server-genereret værdi. Denne anvendes til tidsstempling af de lagrede

[11] Cepa har under designfasen udtrykt ønske om brug af server-genererede nøgler

fjernbetjeninger, og gør det nemmere at sortere eller slette ved fremtidige opdateringer af databasen.

Når designet af databasen er færdigt, kan designet af databaselaget påbegyndes. Databaselaget er sepereret i tre dele, som hver varetager sin specifikke funktion ved kontakt med databasen. Databaselaget tilgås via interfacet IRemoteMapper, som implementeres af klassen RemoteMapper.



Figur 24 - PC-applikationens database-lag

RemoteMapper er den eneste klasse i laget med specifikt kendskab til modellen. Den anvendes til formulering af SQL-sætninger, ud fra de objekter den modtager. Et fjernbetjeningsobjekt kan indeholde flere forskellige datatyper, og frem for at lave flere individuelle databaseforbindelser samles alle dataene i en SQL-sætning. Denne løsning sparer en del tid, da databaseforbindelser ofte er langsommelige at etablere, mens internetbåndbredden bliver stadigt hurtigere. Når RemoteMapper har formuleret en SQL-sætning, opretter den et instance af en ConnectionGateway, som varetager den videre kontakt.

ConnectionGateway er en facadeklasse mellem mapper- og connection-klassen. Denne klasse stiller to simple funktioner til rådighed, som gør det muligt for flere uafhængige mapperklasser, at anvende samme DatabaseConnection, uanset hvilken model de relaterer til. ConnectionGateway opretter forbindelse til DatabaseConnection via interfacet IDatabaseConnection, som implementeres af DatabaseConnection.

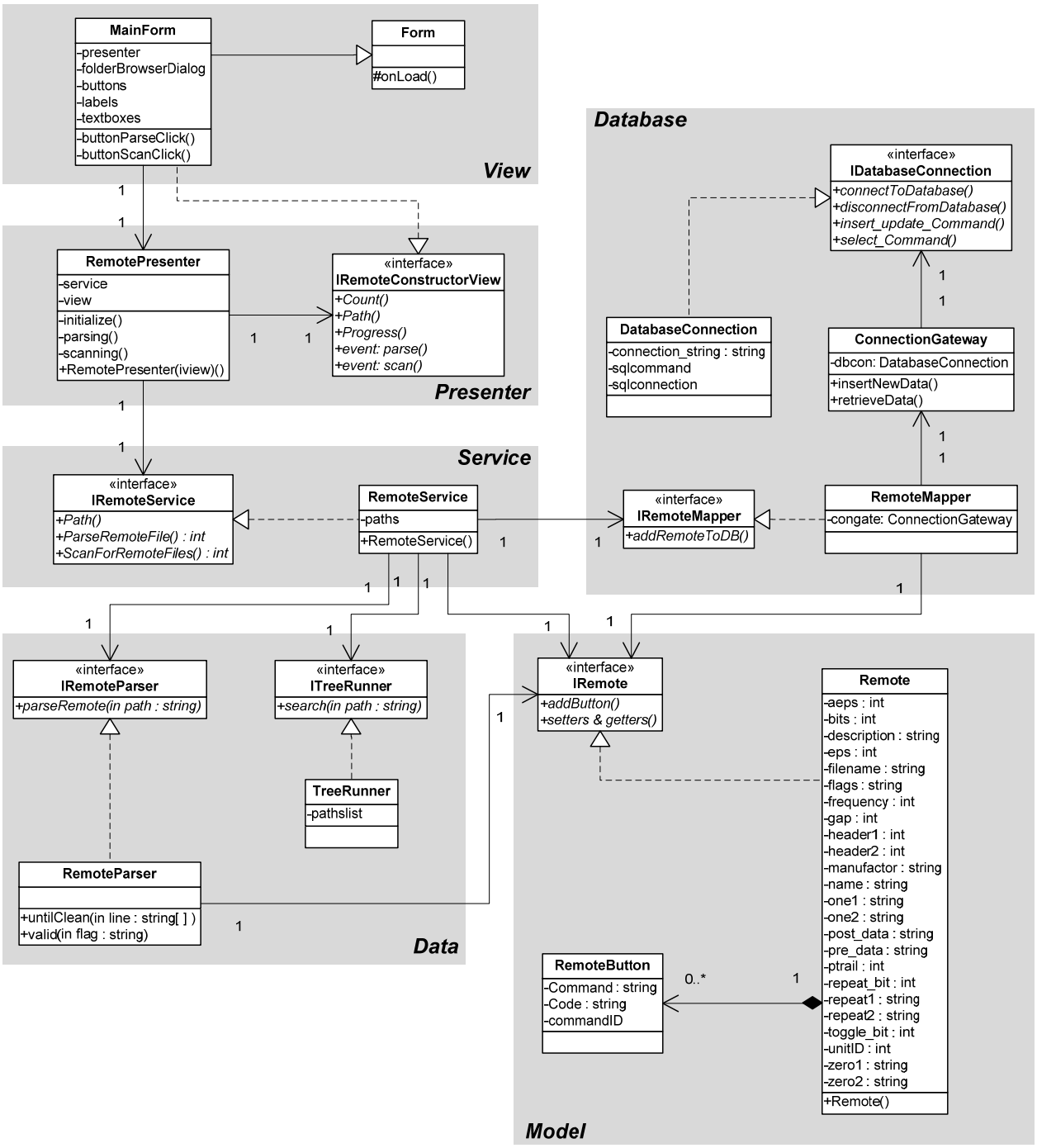
DatabaseConnection er applikationens klasse til håndtering af forbindelser til databasen og skal derfor sørge for, at oprette og lukke forbindelser ved indsætning og udtrækning af data.

Databasen skal ifølge kravspecifikationen oprettes på en MySQL-server og der kan derfor med fordel gøres brug af MySQL's Connector 5.0 til .NET, som er frit tilgængelig fra deres hjemmeside[12]. MySQL Connector 5.0 stiller klasser til rådighed, som muliggør effektiv oprettelse af forbindelse, samt eksekvering af kommandoer. DatabaseConnection er den eneste klasse i applikationen med viden om MySQL-serverens placering (IP), administratornavn og kodeord. Det vil derfor i fremtiden være simpelt, at udskifte leverandør af webhotel, herunder MySQL-server, ved blot at ændre i klassens attribut "connection_string".

[12] http://dev.mysql.com/downloads/connector/net/5.0.html (2007)

3.2.7. Klassediagram

Da samtlige applikationslag nu er skitseret og planlagt, kan de samles til det endelige klassediagram. Klassediagrammets grå boxe repræsenterer applikationslagene, og hver enkelt box er navngivet i et af dens fire hjørner.



Figur 25 - PC-applikationens klassediagram

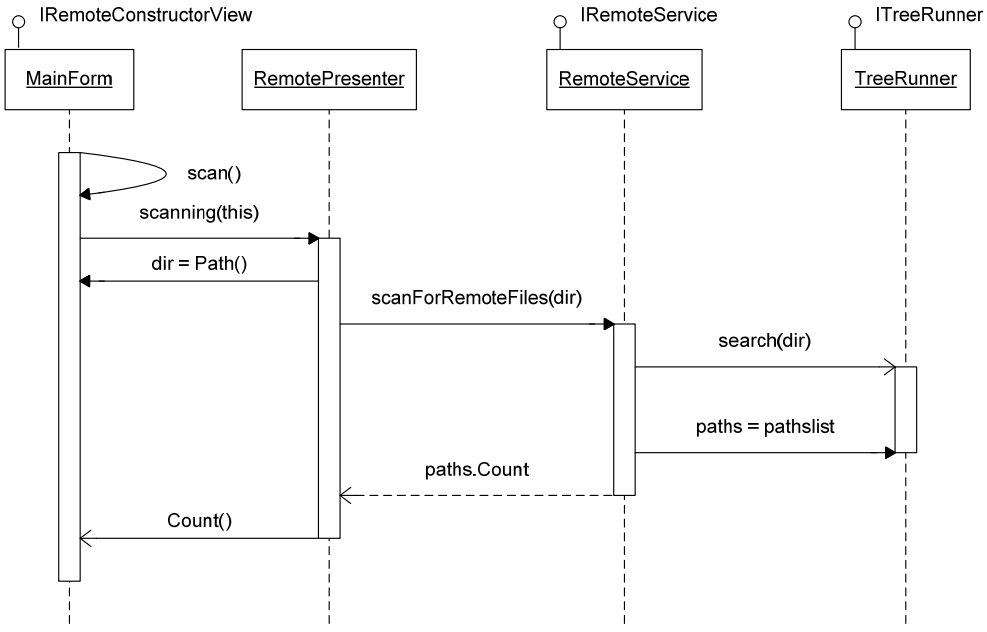
Klassen "MainForm" implementerer interfacet "IRemoteConstructorView", selvom dette interface ikke befinder sig i klassens lag. Dette skyldes, at GUIen skal kunne udskiftes uden redigering og medbringning af gammel kode, men blot ved at implementere interfacet i den

nye. Det bliver dermed muligt blot at fjerne det gamle View-lag og indsætte et nyt. Klassen ”RemotePresenter” i Presenter-laget implementerer som den eneste klasse ikke et interface. Dette skyldes, at klassen aldrig direkte tilgås af de andre klasser, men blot behandler events og videreformidler data. Ud over dette ene tilfælde tilgås samtlige lag via interfaces, og kommunikationen er dermed sikret i tilfælde af fremtidig omstrukturering eller redigering.

3.2.8. Sekvensdiagrammer

Sekvensdiagrammer er et godt supplement til illustrering af en process og kan i denne applikation opstilles parallelt med de udarbejdede use cases. Sekvensdiagrammerne er forsøgt holdt på et overordnet niveau af to årsager først og fremmest for overskuelighed, men også for at give en smule programmatisk frihed under implementeringen. De anvendte funktioner vil dog bestå uanset omfanget af den evt. refactoring. Der bør i samtlige diagrammer lægges mærke til, at der altid kommunikeres gennem interfaces, hvor disse er implementeret.

Det første sekvensdiagram relaterer til use casen ”Lokalisér LIRC-filer”, og brugeren har netop valgt en mappe via knappen med karaktererne ”…”.

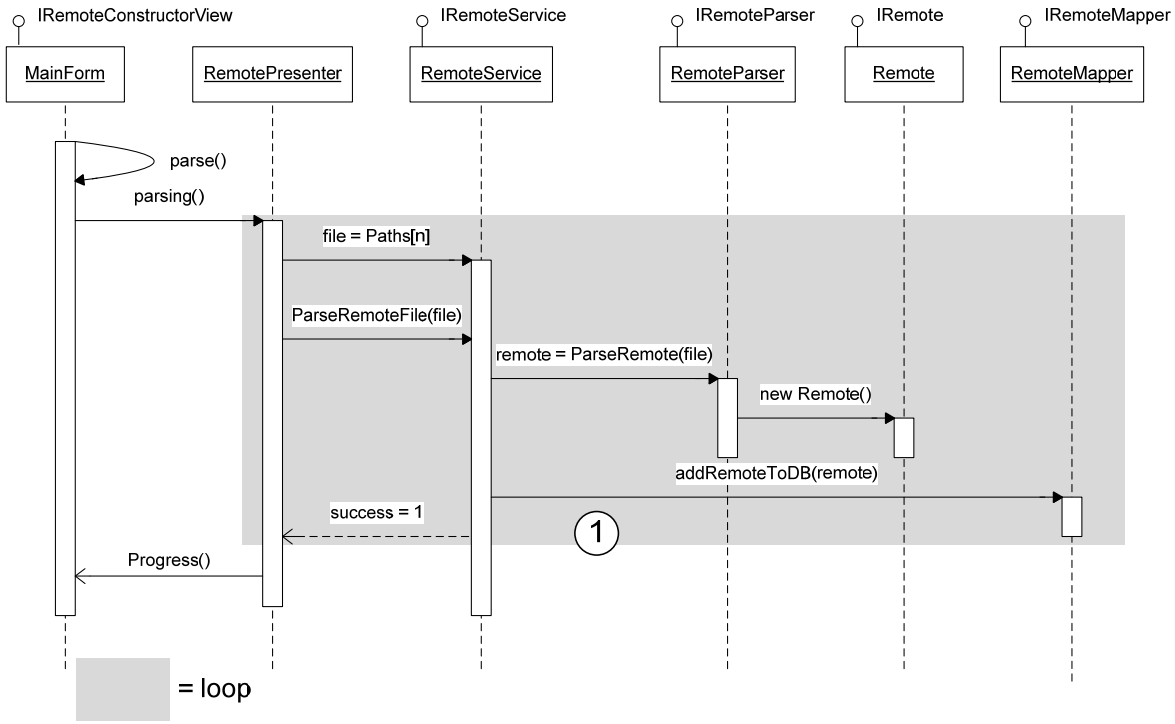


Figur 26 - Sekvensdiagram for "Lokaliser LIRC-filer"

Trykket på knappen genererer eventet ”scan()”, som håndteres af RemotePresenter med funktionen ”scanning(this)”. Parametren ”this” sikrer, at presenteren kan referere tilbage til MainForm og indhente variablen ”dir”, som udgør stien til den valgte mappe. Stien videresendes til RemoteService ved at kalde ”scanForRemoteFiles(dir)”, som igangsætter TreeRunners søgefunktion ved at kalde ”search(dir)”. TreeRunner gennemløber mappen og dens undermapper og opdaterer løbende sin attribut ”pathslist” med fundne filer.

RemoteService indhenter derefter ”pathslist” fra TreeRunner, gemmer den lokalt i sin egen attribut ”paths” og returnerer størrelsen af denne til RemotePresenter. RemotePresenter tilskriver til sidst MainForms ”Count()”, som afspejles i GUI’ens infobox ud for teksten ”antal fundet:”. LIRC-filerne er nu lokaliseret, og stierne ligger i RemoteService til videre behandling.

Følgende sekvensdiagram afspejler use casen ”Scan og gem LIRC-data”. Diagrammet er splittet i to for overskuelighed og er opdelt på det sted i processen, hvor LIRC-data skal skrives til MySQL-serveren. Det lysegrå felt i første sekvensdiagram repræsenterer et iterativt forløb, hvor listen med filstierne gennemløbes indtil sidste element. Det andet sekvensdiagram, som illustrerer kontakten med databasen, skal derfor betragtes som en del af det iterative forløb.

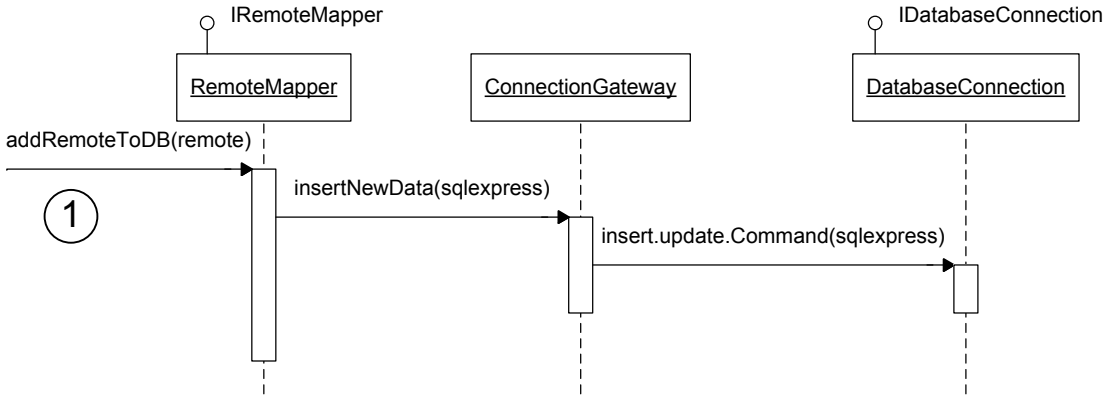


Figur 27 - Sekvensdiagram for "Scan og gem LIRC-data"

Eventet ”parse()” håndteres af RemotePresenter med funktionen ”parsing()”, som enkeltvis henter filstierne fra RemoteService (file = Paths[n]). Hver filsti skal behandles, og det iterative forløb starter derfor her. Den enkelte filsti videresendes til RemoteService ved at kalde ”ParseRemoteFile(file)”, som videresender stien til RemoteParser. RemoteParser opretter et Remote-objekt, som udfyldes med data fundet ved parsing af LIRC-filen. Remote-objektet returneres efter endt parsing af filen til RemoteService, hvilket i diagrammet illustreres med tilskrivningen ”remote = ParseRemote(file)”. Objektet skal nu skrives til databasen via RemoteMappers ”addRemoteToDB(remote)”, men for overskueligheds skyld afspejles dette først i næste sekvensdiagram og skal i dette diagram blot betragtes som en black box. Hvis alt er forløbet succesfuldt inkrementeres en tæller i RemotePresenter, som

efter fuldt gennemløb af listen vises i GUI'ens infobox ud for teksten "antal scannet og gemt:". Brugeren kan nu, hvis ønsket, sammenligne de to resultater og derudfra konkludere, hvor mange ikke-kompatible filer, der befandt sig i mappen.

Det næste sekvensdiagram er som nævnt en del af den iterative proces for "Scan og gem LIRC-data", bedre beskrevet som "Gem LIRC-data". Data fra den parsede LIRC-fil befinder sig her i objektet "remote", som netop er blevet sendt til RemoteMapper via funktionskaldet "addRemoteToDB(remote)".



Figur 28 - Sekvensdiagram for lagring af remote-objekt

RemoteMapper genererer så en enkelt SQL-sætning ved at samle alle remote-objektets attributter i en lang tekststreng (sqlexpress). SQL-sætningen formidles via ConnectionGateway til DatabaseConnection, som åbner en forbindelse til databasen på MySQL-serveren, eksekverer SQL-sætningen og lukker forbindelsen igen. ConnectionGateway kan derfor betragtes som en formidlende facade, som muliggør brugen af DatabaseConnection for evt. andre mapper-klasser.

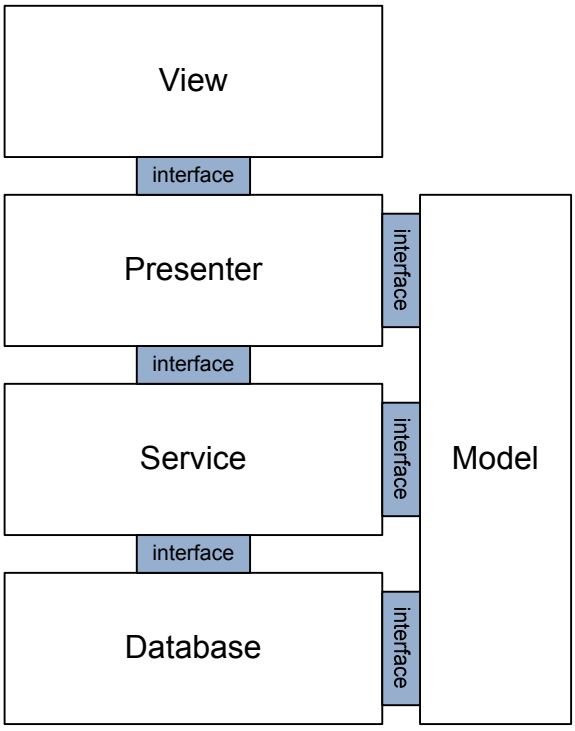
Dette kan betragtes som en valid løsningsmodel for use casen "Scan og gem LIRC-data", og der er dermed udarbejdet et fuldt design af alle features i PC-applikationen.

3.3. Design af server-applikationen

Før påbegyndelse af designet for server-applikationen, opsummeres resultaterne af tidligere afsnit vedrørende denne. Ifølge kravspecifikationen skal konfiguration af fjernebetjeningsfilen kunne udføres via få brugervenlige trin. Fjernbetjeningsfilen skal derefter kunne downloades via server-applikationen og placeres et vilkårligt sted på en PC. I analysen blev brugeren af applikationen identificeret som en handicapmedhjælper, som assisterer den bevægelseshæmmede i sin hverdag. Brugeren skal ud fra de identificerede use cases igennem mindst 5 trin for at identificere en fjernbetjening, redigere dens knapper og beskrivelse, tilføje den til en brugertilknyttet samling og til sidst downloade den server-genererede fil. Dog skal de første fire trin gentages for hver fjernbetjening, som ønskes understøttet af fjernbetjeningen. Følgende skal derfor designes til server-applikationen:

- En grafisk brugerflade til interaktion med brugeren
- En presenter til håndtering af events og synkronisering
- En business-model til strukturering og beskrivelse af objekterne
- En forbindelse mellem MySQL-server og server-applikationen

Server-applikationen skal interagerer med brugeren i form af en hjemmeside, som ud over at give brugeren adgang til data fra databasen også kortvarigt kan lagre brugerens arbejde samt generere en fil baseret på arbejdet. Arkitekturen for server-applikationen skal, vurderet ud fra ovenstående, se ud som følgende:



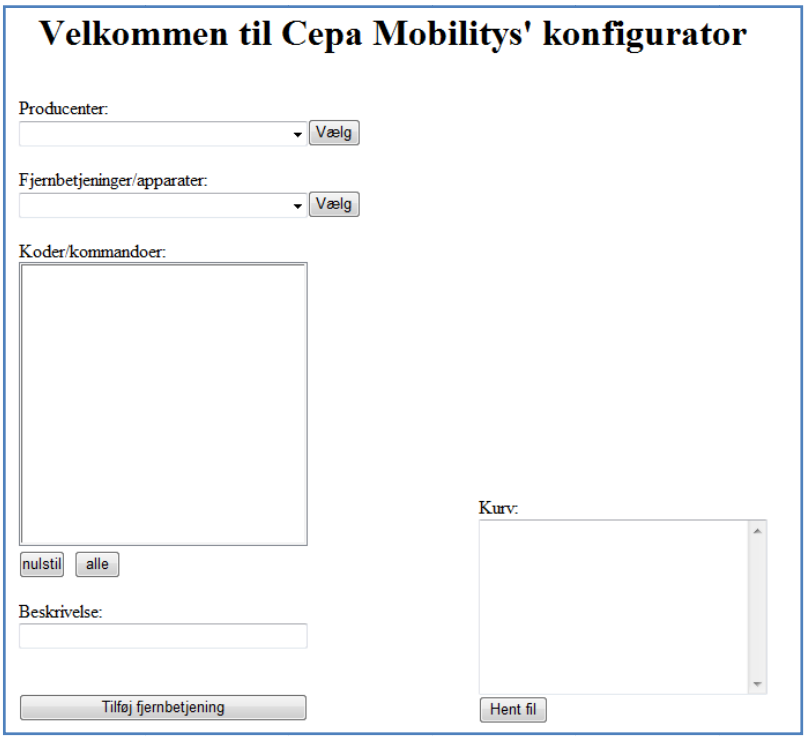
Figur 29 - Server-applikationens arkitektur

Designet af Model- og Database-lag vil i denne applikation være baseret på designet udarbejdet for PC-applikationen. Designet af de to lag kan genbruges i et sådan omfang, at der blot skal tilføjes få metoder til enkelte klasser. Applikationen vil som den forrige gøre

brug af MVP med udgangspunkt i Supervising Controller og Passive View. Der vil derfor skulle tages stilling til fordelingen af ansvar mellem klasserne, da den direkte anvendelse af flere web-baserede funktioner ikke stemmer overens med konceptet for Passive View.

3.3.1. View og Presenter

Den grafiske brugerflade består af en dynamisk hjemmeside med hovedformålet at identificere og gemme fjernbetjeningsdata. Knapperne på hjemmsiden er direkte relaterede til de fem use cases, som blev identificeret under analysen af server-applikationen dog undtagen use casen ”Hent Producentoversigt”.



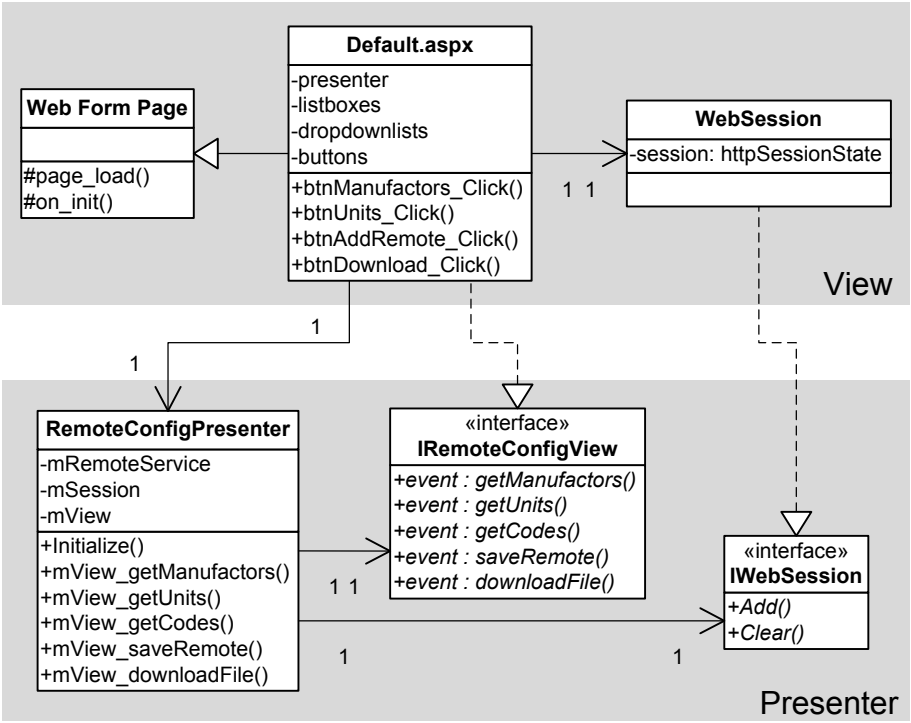
Figur 30 - Server-applikationens GUI

Use casen ”Hent Producentoversigt” er et udgangspunkt for hjemmesiden og skal være opdateret allerede når siden er indlæst. Producentoversigten vil derfor fra start blive vist i dropdownlisten under teksten ”Producenter:”.

Ud for dropdownlisten, indeholdende producentoversigten, ses en knap med teksten ”Vælg”. Denne knap repræsenterer use casen ”Hent Fjernbetjeninger”, som henter samtlige fjernbetjeninger fra den valgte producent og opdaterer dem i dropdownlisten under teksten ”Fjernbetjeninger/apparater:”. På samme måde fungerer det for use casen ”Hent knapper”, som dog opdateres i en listbox da det skal være muligt at vælge/fravælge enkelte knapper. Use casen ”Definér og gem fjernbetjening” håndteres ved, at brugeren selv indtaster en beskrivelse af fjernbetjeningen, som der blev vedtaget i analysen pga. problemet med ringe

navngivning i LIRC-filerne. Brugeren skal efter dette trykke på knappen ”Tilføj fjernbetjening”, som lagrer den redigerede fjernbetjening i en tilknyttet session. De hidtil beskrevne trin gentages indtil samtlige ønskede fjernbetjeninger befinder sig i sessionen. Endelig kan brugeren vælge knappen ”Hent fil” svarende til use casen ”Generer/hent fil”, som igangsætter en generering af filen ud fra sessionsindholdet og tilbyder denne som download for brugeren.

Da konceptet for MVP blev beskrevet tidligere i kapitlet, blev Passive View beskrevet som et view med mindst mulig funktionalitet ud over brugerinteraktion. Håndtering af sessionens indhold ønskes derfor placeret eksternt for viewet. Dette lader sig bedst gøre, ved at oprette en separat klasse kaldet ”WebSession” indeholdende en httpSession, som gøres tilgængelig for presenteren ”RemoteConfigPresenter” via et implementeret interface ”IWebSession”.

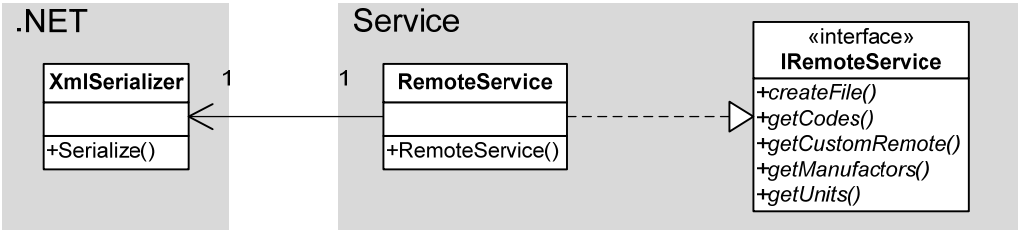


Figur 31 - Server-applikationens view-/presenter-lag

Klassen ”Default.aspx” udgør den dynamiske hjemmeside og nedarver sin basale funktionalitet fra .NET’s ”System.Web.UI.Page” også kendt som ”Web Form Page”. Oprettelse af presenter og websession skal ske tidligst muligt i sidens livscyklus, hvilket er under dens initialisering. Sessionen videresendes med det samme til presenteren, som herfra håndterer dens indhold via interfacet ”IWebSession”. Ud over denne funktionalitet vil samspillet mellem view og presenter, foregå på samme måde som i PC-applikationen.

3.3.2. Service

Service-laget fungerer også i denne applikation som en facade mellem presenteren og de underliggende klasser, men den har her yderligere en funktionalitet relateret til funktionen ”createFile()”.



Figur 32 - Server-applikationens service-lag

Server-applikationen har intet data-lag på trods af XML-håndteringen, som normalt ville ligge i et sådan. XML-generering kan nemlig håndteres ved anvendelse af .NETs ”XmlSerializer”, som gør det muligt at generere en gyldig XML-kode ud fra vilkårlige objekter. Funktionen ”createFile()” vil derfor håndtere oprettelse af XmlSerializer samt returnering af den genererede XML-kode til presenteren.

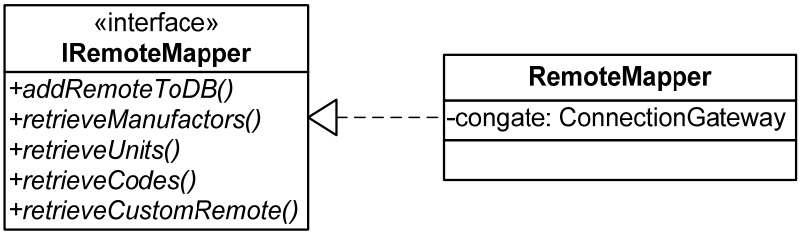
Serialisering af objekter kan tilpasses efter ønske, så krav til struktur af XML-koden kan efterkommes. XML-kode genereret i server-applikationen skal være mulig at gennemlæse og anvende sekventielt, så det ikke er nødvendigt at indlæse hele filen i PDA’ens hukommelse. Serialisering skal derfor tilpasses, så XML-koden nogenlunde anvender følgende struktur:

```
<Remotes>
  <Remote Beskrivelse=”brugerindtastet beskrivelse” ID=”servergenereret id”>
    <Konfigdata_1>???</Konfigdata_1>
    <Konfigdata_2>???</Konfigdata_2>
    ……flere konfig. data……..
    <Knapper>
      <Knap>
        <Kommando>???</Kommando>
        <Kode>???</Kode>
        <ID>???</ID>
      </Knap>
      <Knap>
        <Kommando>???</Kommando>
        <Kode>???</Kode>
        <ID>???</ID>
      </Knap>
      ……flere knapper……..
    </Knapper>
  </Remote>
</Remotes>
```

Det vil med denne pseudo-struktur være muligt at udtrække fjernbetjeningens beskrivelse og ID separat uden at skulle indlæse hele filen i hukommelsen.

3.3.3. Model og Database

Modellen og database-laget kræver som tidligere nævnt ikke nogle strukturelle ændringer. Database-laget skal nu yderligere håndtere udtrækning af data fra databasen, og klassen ”RemoteMapper” skal derfor kunne formulere de nødvendige SQL-sætninger samt returnere resultatet af forespørgslerne.



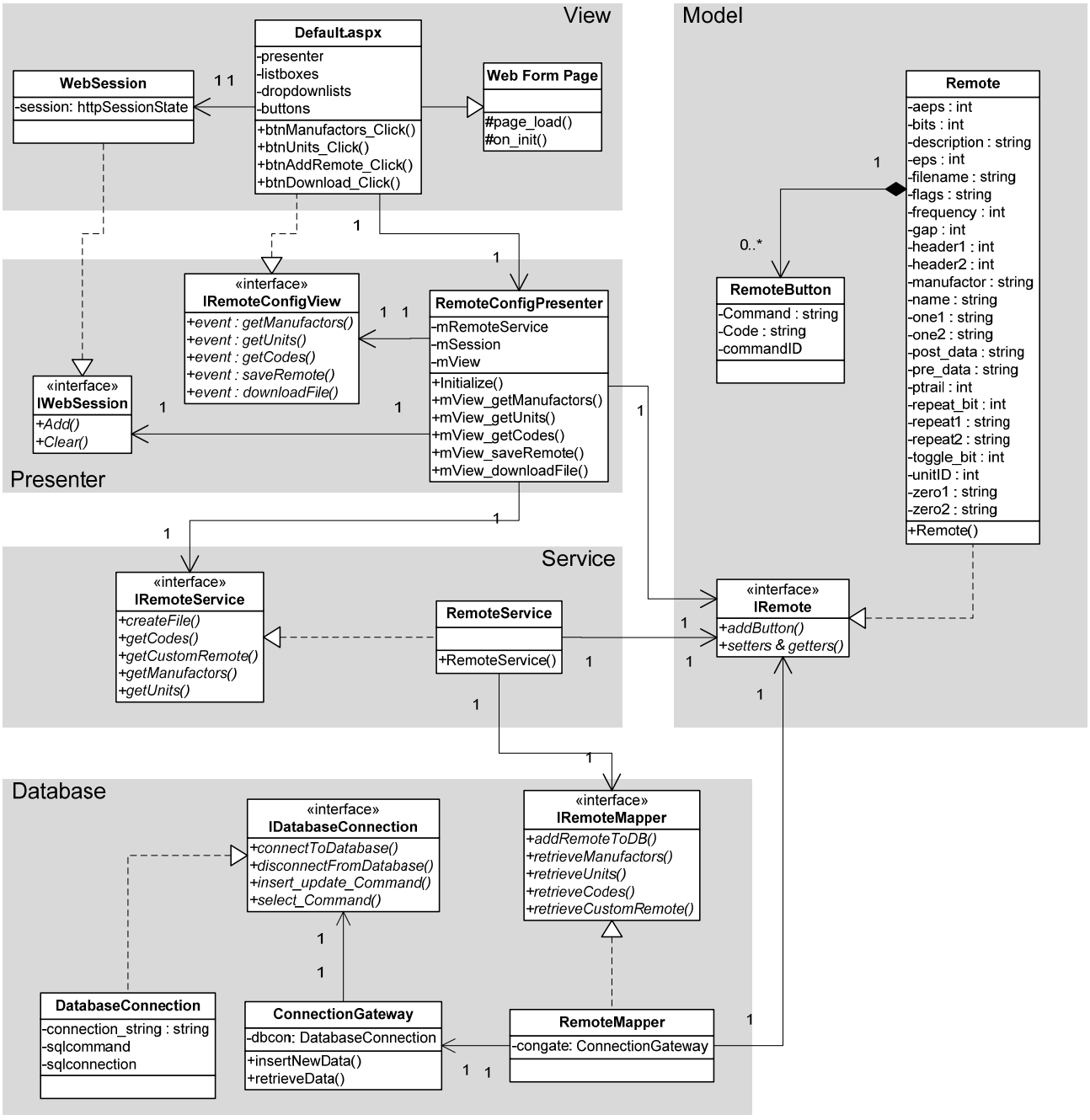
Figur 33 - Server-applikationens udvidelser til RemoteMapper

Interfacet ”IRemoteMapper”, som implementeres af klassen ”RemoteMapper”, udvides derfor med fire funktioner yderligere, som hver repræsenterer en af de udarbejdede use cases.

Modellen behøver ingen tilføjelser i form af funktioner og attributter, men skal grundet den ønskede serialisering have tilføjet to markeringer ved attributterne ”description” og ”unitID”. Markeringerne skal sikre, at disse to attributter serialiseres som XML-attributter frem for XML-elementer. De andre attributter i klassen vil blive serialiseret som elementer, da dette er standard, når intet andet er indikeret.

3.3.4. Klassediagram

Ved sammensætning af diagrammerne for de designede applikationslag fremkommer nedenstående diagram. Afgrænsningerne for hvert applikationslag er illustreret med en grå box, hvor navnet på laget er anført i et af de fire hjørner.



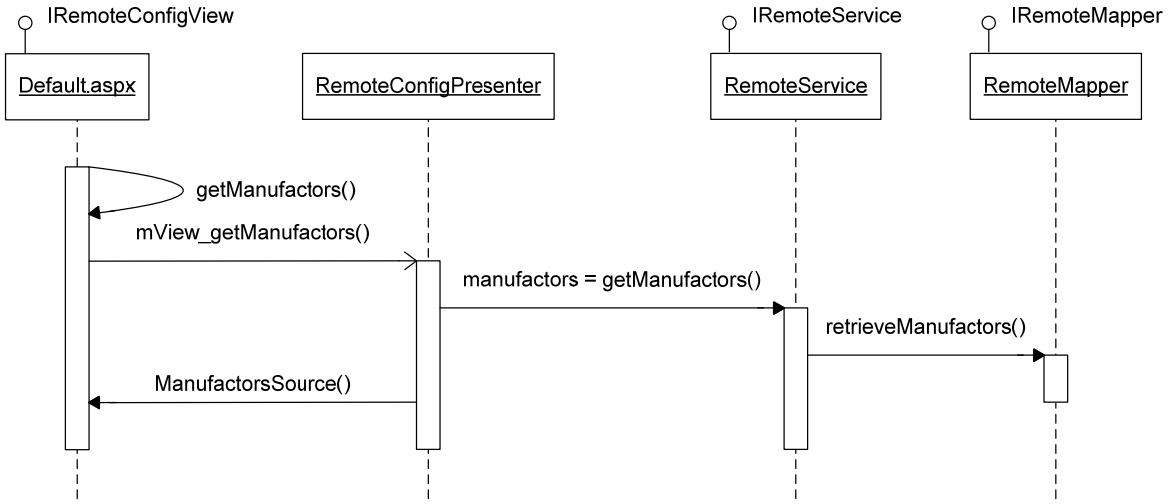
Figur 34 - Server-applikationens klassediagram

Kommunikation mellem lagene foregår via interfaces pånær mellem view- og presenter-lagene. Som illustreret ovenstående er lagene Database og Model nærmest identiske med PC-applikationens, og der kan derfor med fordel også genbruges kode under implementeringen.

3.3.5. Sekvensdiagrammer

Der vil i dette afsnit blive udarbejdet tre sekvensdiagrammer, som har til formål at afspejle tre af de udarbejdede use cases for server-applikationen. Use casen ”Hent producenter” er valgt, fordi den har samme basale funktionalitet som de to næste: ”Hent fjernbetjeninger” og ”Hent knapper”. Andet og tredje sekvensdiagram afspejler use casene ”Definér og gem fjernbetj.”, samt ”Generér/hent fil”. Disse to use cases ønskes diagrammeret fordi udarbejdelse af funktionaliteten bag dem kræver en mere detaljeret og dybdegående gennemgang. Da det tidligere er diagrammeret, hvorledes database-laget fungerer, vil sekvensdiagrammerne springe over denne funktionalitet ved kald til klassen ”RemoteMapper” og fortsætte diagrammeringen ved returdata fra samme klasse.

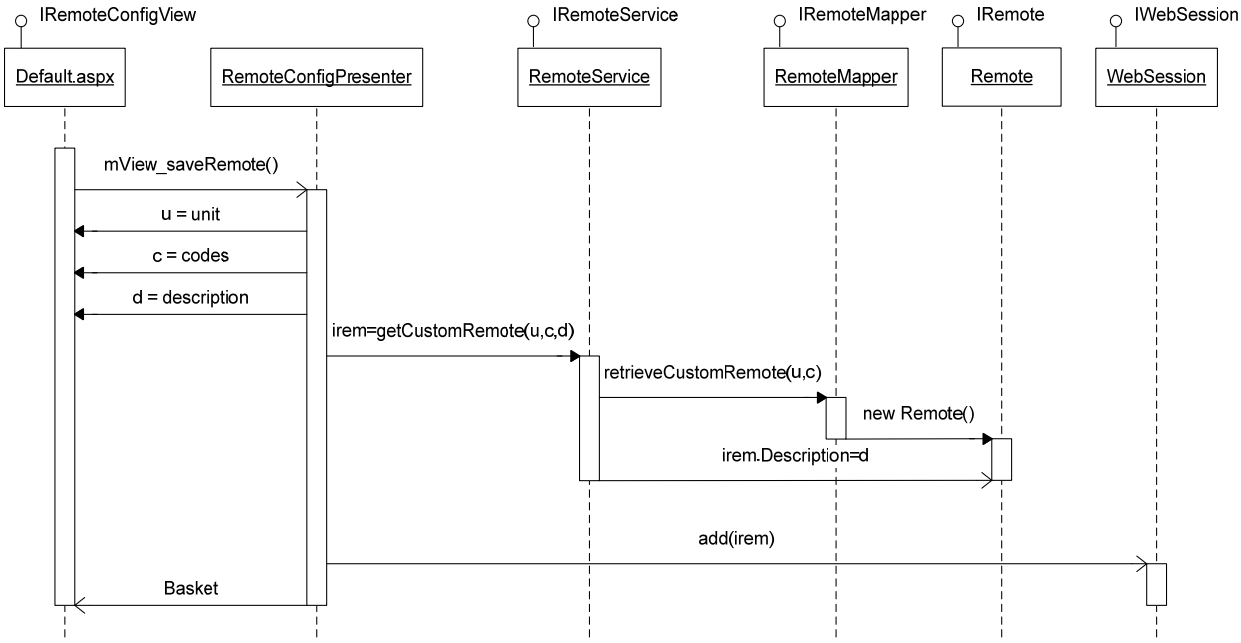
Processforløbet for indhentning af producentoversigten starter ved åbningen af den dynamiske hjemmeside og er første trin for identificering og redigering af en fjernbetjening.



Figur 35 - Sekevensdiagram for "Hent producenter"

”Hent producenter” består hovedsagligt af en række metodekald ned gennem lagene, og havde det ikke været for den fastlagte lagstruktur, ville der muligvis have været en direkte binding til datakilden. Eventet ”getManufactors()” indtræffer når siden loades og håndteres af RemoteConfigPresenter via funktionen ”mView_getManufactors()”. Presenteren forespørger så service-laget, som igen forespørger klassen ”RemoteMapper” i database-laget. Når alle forespørgslerne har returneret de forespurgte data, opdaterer presenteren ”Default.aspx” via dennes ”ManufactorSource”, og producentoversigten vil så være opdateret med de indhentede data. De to ikke-diagrammerede use cases fungerer på samme måde, blot med udtrækning af en parameter fra viewet efter presenteren er blevet kaldt.

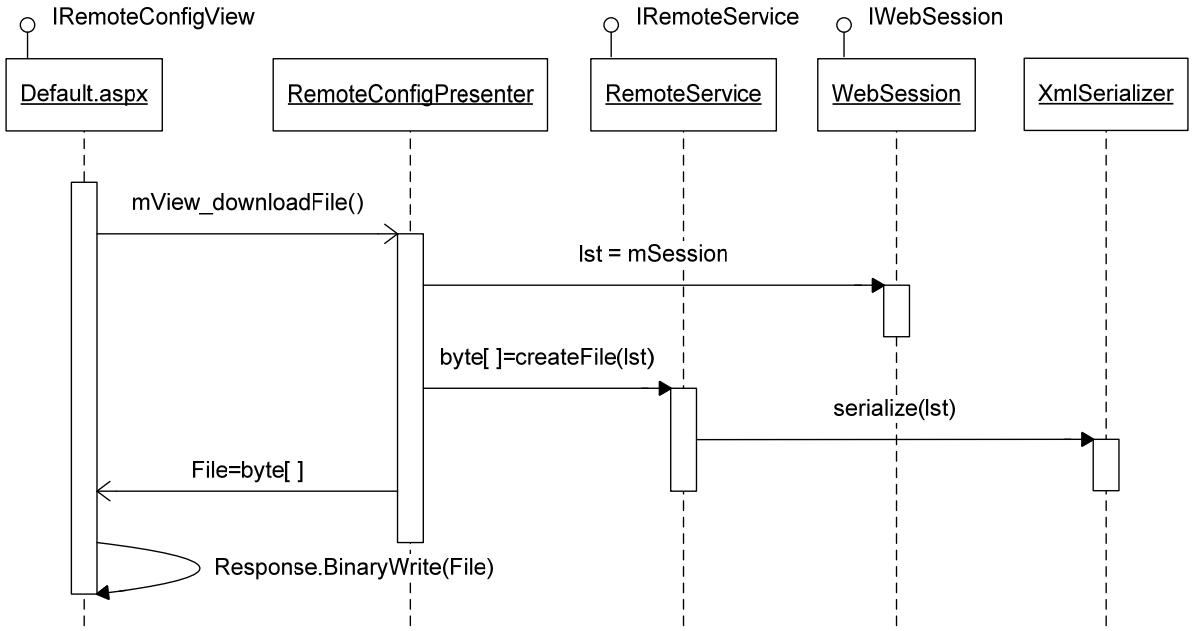
Processon bag use casen ”Definér og gem fjernbetj.” gør brug af samtlige valg brugeren har foretaget fra valg af enhed og knapper til indtastning af beskrivelsen. Følgende sekvensdiagram (Figur 36) afspejler processen fra det øjeblik knappen ”Tilføj fjernbetjening” bliver trykket og eventet ”saveRemote” indtræffer.



Figur 36 - Sekvensdiagram for "Definér og gem fjernbetj."

Eventet håndteres af presenteren med metoden ”saveRemote()”, som efterfølgende indhenter id for unit (fjernbetjeningen), id-samling for codes (knapperne) og endelig den indtastede beskrivelse ”description”. De indhentede data sendes til sevice-lagets ”RemoteService” som forespørgsel, og denne klasse kalder ”RemoteMapper” i database-laget. RemoteMapper henter samtlige konfigurationsdata tilknyttet den efterspurgte fjernbetjening samt alle de efterspurgte knapper. Disse data tilskrives et remote-objekt, som returneres til klassen ”RemoteService”. RemoteService tilføjer efterfølgende beskrivelsen til remote-objektet og returnerer remote-objektet til presenteren. Presenteren, som håndterer data i sessionen, tilføjer endelig remote-objektet til websessionen og opdaterer den tidligere omtalte indkøbskurv i ”Default.aspx”. Den redigerede fjernbetjening ligger nu i sessionen og lagres indtil, at brugeren er færdig og ønsker at generere sin xml-fil.

Det tredje og sidste sekvensdiagram skal afspejle processen for use casen ”Generér/hent fil”. Sekvensdiagrammet starter, hvor brugeren har tilføjet alle ønskede fjernbetjeninger til sessionen og trykket på knappen ”Hent fil”, som dermed genererer eventet ”downloadFile”.



Figur 37 - Sekvensdiagram for ”Generer/hent fil”

Presenteren kaldes med funktionen ”mView_downloadFile()”, som tilføjer indholdet af sessionen til variablen ”lst”, hvilket er en liste af remote-objekter. Listen formidles derefter til service-klassen ”RemoteService”, som bliver bedt om at generere og returnere et byte-array. RemoteService beder klassen ”XmlSerializer” om serialisering af af listen, som returneres iform af et byte-array. Presenteren tilskriver efterfølgende den indhentede buffer til viewets ”File”, som foretager en ”Response.BinaryWrite” af bufferen og genererer den endelige XML-fil til download.

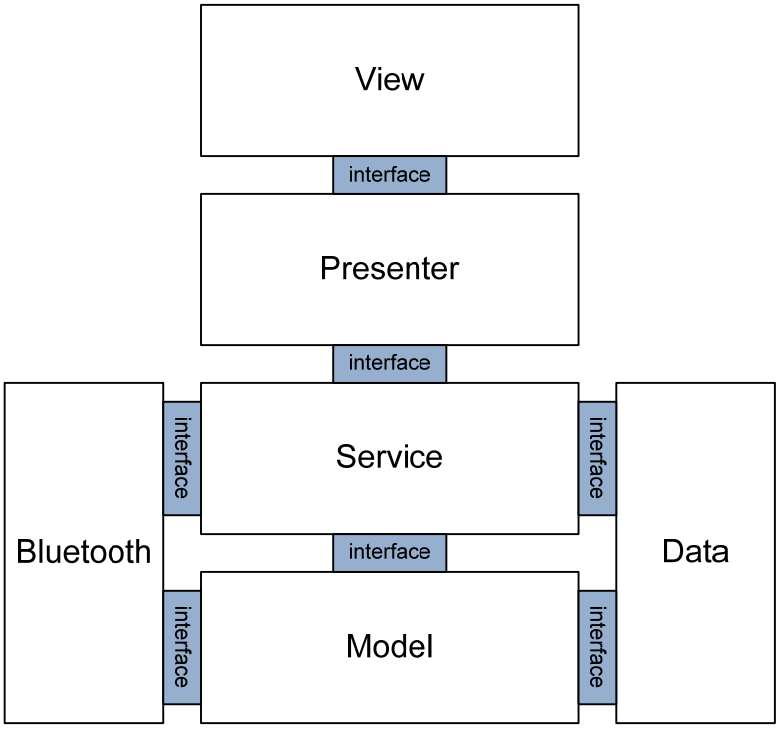
Der er nu udarbejdet et brugbart design for samtlige løsningsmodeller under server-applikationen og denne er derfor parat til implementering.

3.4. Design af PDA-applikationen

Som i de to foregående afsnit vil der før påbegyndelse af designet opsummeres resultater fra tidligere kapitler vedrørende PDA-applikationen. Kravspecifikationen er ikke videre specifik omkring applikationens udformning med den undtagelse, at den skal gøre det muligt at afsende konfigurationsdata og IR-koder via en bluetooth-forbindelse. Brugeren af applikationen blev identificeret som en bevægelseshæmmet person med CepaCom monteret og installeret i kørestolen. Brugeren kan nu ifølge de identificerede use cases under analysen, kunne få præsenteret en liste over de understøttede fjernbetjeninger, vælge en konkret fjernbetjening og se dennes knapper samt vælge en specifik knap og afsende dennes IR-kode. Designet af de to foregående applikationer har høj tilknytning til designet for PDA-applikationen, da business-model og data-struktur nu ligger fast.

PDA-applikationen kan opdeles i seks applikationslag, som hver varetager et separat ansvar og kaldes via interfaces. Vurderet ud fra ovenstående skal der, designes følgende nye funktionaliteter:

1. En dynamisk brugerflade til håndtering af variable mængder grafiske elementer.
2. Et data-lag til effektiv udtrækning af data fra XML-filen.
3. Et bluetooth-lag med mulighed for fremtidig udvidelse/udskiftning.



Figur 38 - PDA-applikationens arkitektur

Følgende afgrænsninger er yderligere foretaget i applikationen:

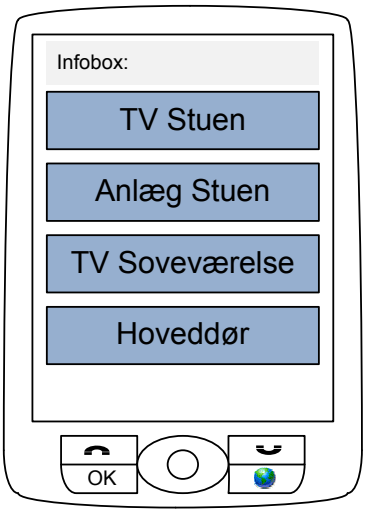
- Grundet tidligere omtalte vanskeligheder vedrørende integrering af brugermenu i det eksisterende system skal der kun udvikles en midlertidig GUI. Denne GUI vil dog danne grundlag for ”proof of concept”.
- Grundet nye informationer omkring Cepas egen udvikling skal pakken ”Franson Bluetools” til kommunikation via bluetooth ikke anvendes. I stedet skal der arbejdes hen mod en anvendelse af Cepas egne kommunikationsklasser, som pt. gennemgår større omskrivning. Data som ønskes sendt til hardwarefjernbetjeningen skal af den grund stadig formidles til bluetooth-laget, men skal ikke afsendes via bluetooth.

For samarbejdet mellem klasserne i view-, presenter- og service-lagene gælder samme principper som tidligere. View-laget samarbejder med presenter-laget i overenstemmelse med MVP, mens service-laget igen fungerer som en facade mellem presenteren og de underliggende klasser. Service-lagets klasse ”PDAService”, som implementerer interfacet ”IPDAService”, får yderligere som ansvar at lagre et remote-objekt efter dette er blevet indlæst fra XML-filen. Dette kan sammenlignes med PC-applikationen, hvor service-laget fik til opgave at lagre listen over stier til potentielle LIRC-filer.

3.4.1. GUI

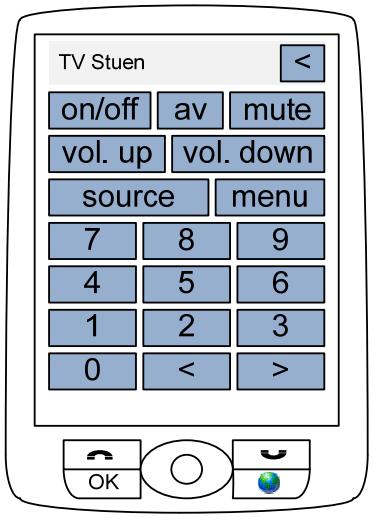
GUI'en for PDA-applikationen har til formål at lade brugeren interagere med sine elektriske apparater. GUIen skal følge de tre identificerede use cases, men ikke det autoscannende menusystem, som anvendes i Cepas egen software CepaCom.

Under designet af server-applikationen blev en struktur fastlagt for XML-koden, som gør det muligt først at udtrække samtlige headers, som indeholder beskrivelse og id for hver fjernbetjening i XML-filen. Det første skærmbillede vil derfor være en repræsentation af de udtrukne headers, som illustreres i form af en knap med hver fjernbetjenings beskrivelse.



Figur 39 - Illustration af PDA-applikationens GUI 1

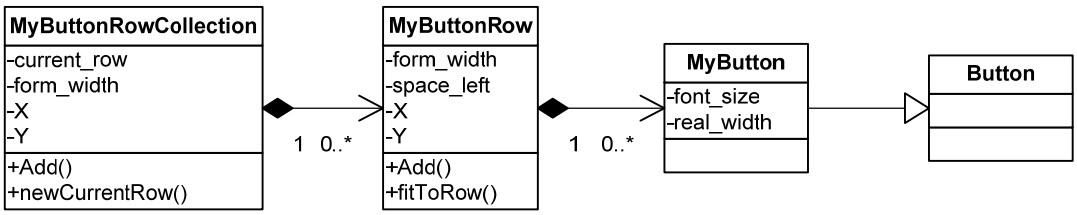
Ved at trykke på en af de viste knapper hentes alle knapper tilhørende den valgte fjernbetjening og illustreres som nye knapper. Hvis brugeren trykker på en af disse afsendes den tilhørende IR-kode, som kontrollerer apparatet. Hermed opfyldes funktionaliteten for de tre use case "Vis fjernbetjeninger", "Vælg fjernbetjening" og "Vælg knap".



Figur 40 - Illustration af PDA-applikationens GUI 2

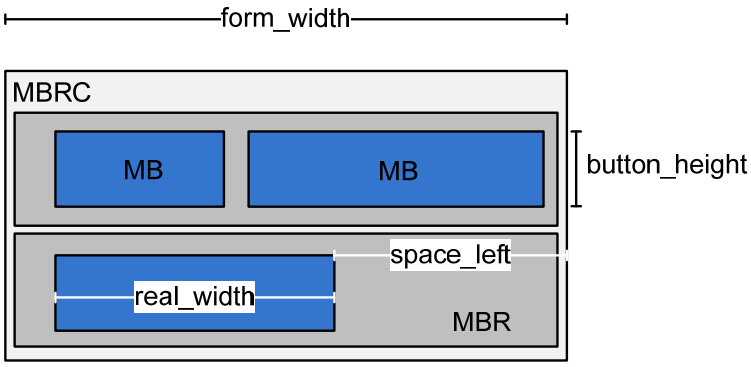
For at kunne håndtere og placere det varierende antal knapper, som er tilknyttet hver fjernbetjening, skal der foretages et sæt beregninger for hver knap samt for hver række af knapper. Til dette formål oprettes tre klasser:

- MyButton
- MyButtonRow
- MyButtonRowCollection



Figur 41 - PDA-applikationens special-klasser til GUI

MyButtonRowCollection (MBRC) svarer til et panel med koordinatsæt (0,0), som kan indeholde et varierende antal MyButtonRows (MBR). Ved oprettelsen af MBRC benyttes formens bredde ”form_width” som parameter og denne anvendes igen ved oprettelsen af MBR , men bliver nu til attributten ”space_left”. Attributten ”space_left” anvendes ved at trække bredden af den tilføjede MyButton (MB) fra ”space_left”, og dermed fremkommer den resterende pladsmængde i rækken. Når en MB tilføjes en MBR, skal MBR derfor trække MB’ens ”real_width” fra ”space_left”, sætte koordinaterne på den tilføjede MB og beregne koordinater for den evt. næste MB.

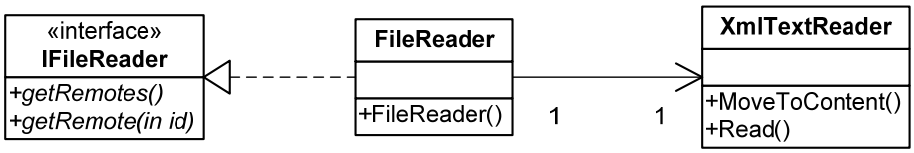


Figur 42 - Illustration af GUI'ens opsætning

Ved oprettelsen af en MB skal der før den tilføjes undersøges, om MBens bredde overstiger den resterende plads, i hvilket tilfælde en ny MBR skal oprettes med MBRCs ”newCurrentRow()”. Yderligere ønskes der maksimalt tre knapper i hver række, og antallet af MBer i rækken skal derfor også undersøges.

3.4.2. Data

Data-laget varetager ansvaret for adgangen til XML-filen, der indeholder fjernbetjeningerne. Klassen ”FileReader” implementerer interfacet ”IFileReader”, som sikrer kommunikationen med de andre lag i applikationen.



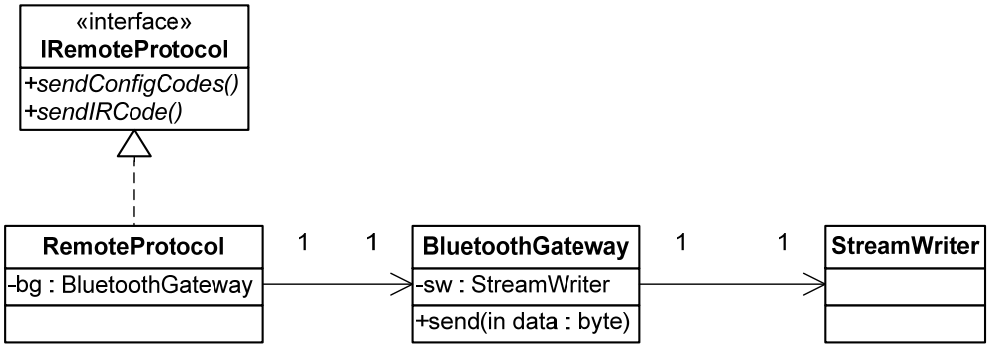
Figur 43 - PDA-applikationens data-lag

FileReader er en XML-parser specielt beregnet til udtrækning af headers og remote-objekter ved at anvende .NET’s ”XmlTextReader”. XmlTextReader er udelukkende fremadrettet og læser en enkelt linje ad gangen. Brugen af XmlTextReader gør applikationen effektiv med henblik på hukommelsesforbrug, men gør det også mere besværligt at udtrække data. I analysen af .NET Compact Framework blev det dog konkluderet, at XML-håndteringen kunne blive besværlig, og strukturen af XML-koden blev derfor tilpasset, så de nødvendige oplysninger for en header kunne udtrækkes ved gennemlæsningen af en enkelt linje. Dette blev opnået ved at tvinge ”description” og ”unitID” igennem serialiseringen som XML-attributter, hvorved det bliver muligt at læse dem som det første og på en enkelt linje.

I funktionen ”getRemotes()” gennemlæser parseren XML-filen sekventielt, udtrækker headers for samtlige fjernbetjeninger og returnerer dem til den forespørgende klasse. Funktionen ”getRemote()” anvender en medbragt parameter ”id”, som er lig et ”unitID” for en af fjernbetjeningerne. Parseren gennemlæser igen XML-filen sekventielt, men går denne gang dybere i XML-strukturen, når den finder det angivne id. Ved at gå dybere i XML-strukturen kan den udtrække samtlige informationer, der er lagret om en konkret fjernbetjening og returnerer denne ved endt gennemløb.

3.4.3. Bluetooth

Bluetooth-laget består af en protokol ”RemoteProtokol”, som implementerer interfacet ”IRemoteProtocol” og en gateway ”BluetoothGateway”, der vil fungerer som en facade til kommunikationsklassen i en endelig løsning. Kommunikationsklassen er dog grundet tidligere omtalte problemer udskiftet i denne løsningen så applikationen som minimum kan anvendes til proof-of-concept. Klassen er derfor erstattet med en StreamWriter, som vil gøre det muligt at skrive ellers normalt afsendte data til en fil i stedet.

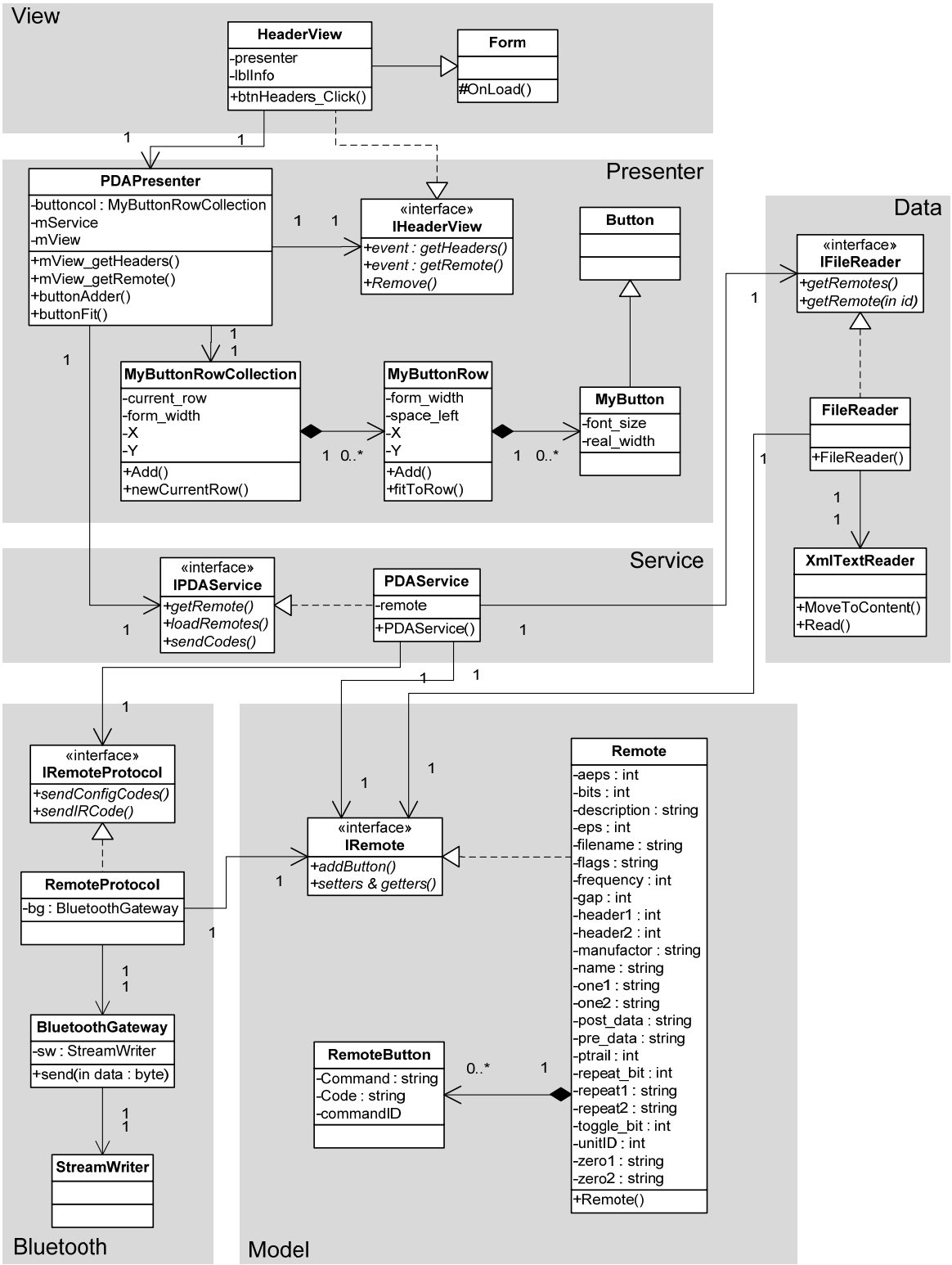


Figur 44 - PDA-applikationens bluetooth-lag

Da størstedelen af bluetooth-lag vil blive udskiftet, vil der ikke blive kigget nærmere på designet af dette. Det oprindeligt påtænkte design af bluetooth-laget er vedlagt som ”Bilag 4: PDA-applikationen – tiltænkt bluetooth-lag”.

3.4.4. Klassediagram

Klassediagrammet er igen sammensat af de designede lag opdelt i navngivne grå boxe. Klasserne ”Form”, ”Button, ”XmlTextReader” og ”StreamWriter” tilhører alle .NETs Compact Framework, som anvendes til udvikling mod mobil og PDA.

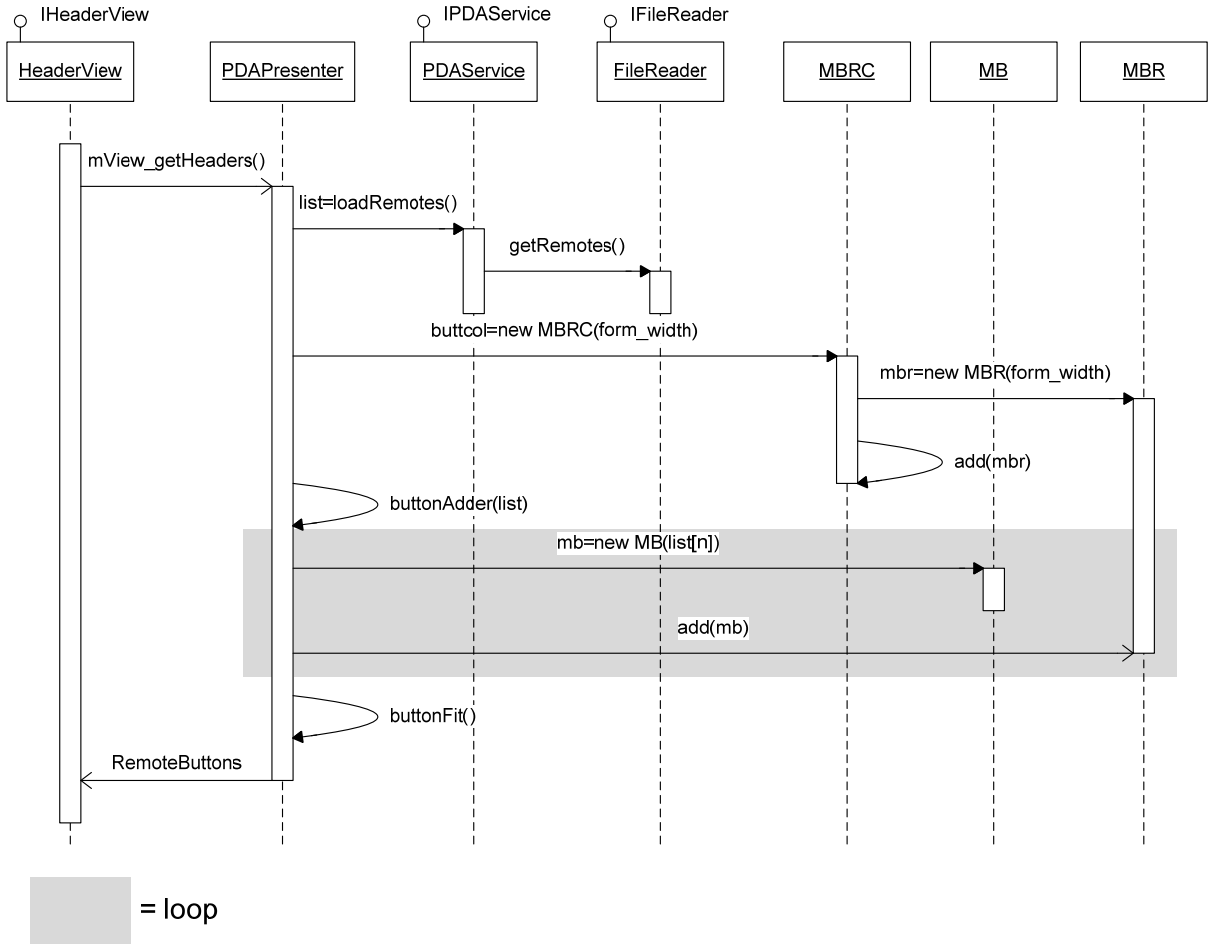


Figur 45 - PDA-applikationens klassediagram

3.4.5. Sekvensdiagrammer

Sekvensdiagrammerne i dette afsnit vil have til formål at afspejle processen for de tre use cases identificeret under PDA-applikationens analyse. De tre use cases kaldet ”Vis fjernbetjeninger”, ”Vælg fjernbetj.” og ”Vælg knap” udgør tre trin, som gør brugeren i stand til at betjene et elektrisk apparat, hvis dette apparats konfigurationsoplysninger og IR-koder befinder sig i XML-filen på PDA’en.

Det første sekvensdiagram afspejler ”Vis fjernbetjeninger”, som præsenterer brugeren for en liste med beskrivelsen for hver fjernbetjening i XML-filen. Det sker ved opstart af applikationen eller ved returnering fra menuen med hver fjernbetjenings knapper.

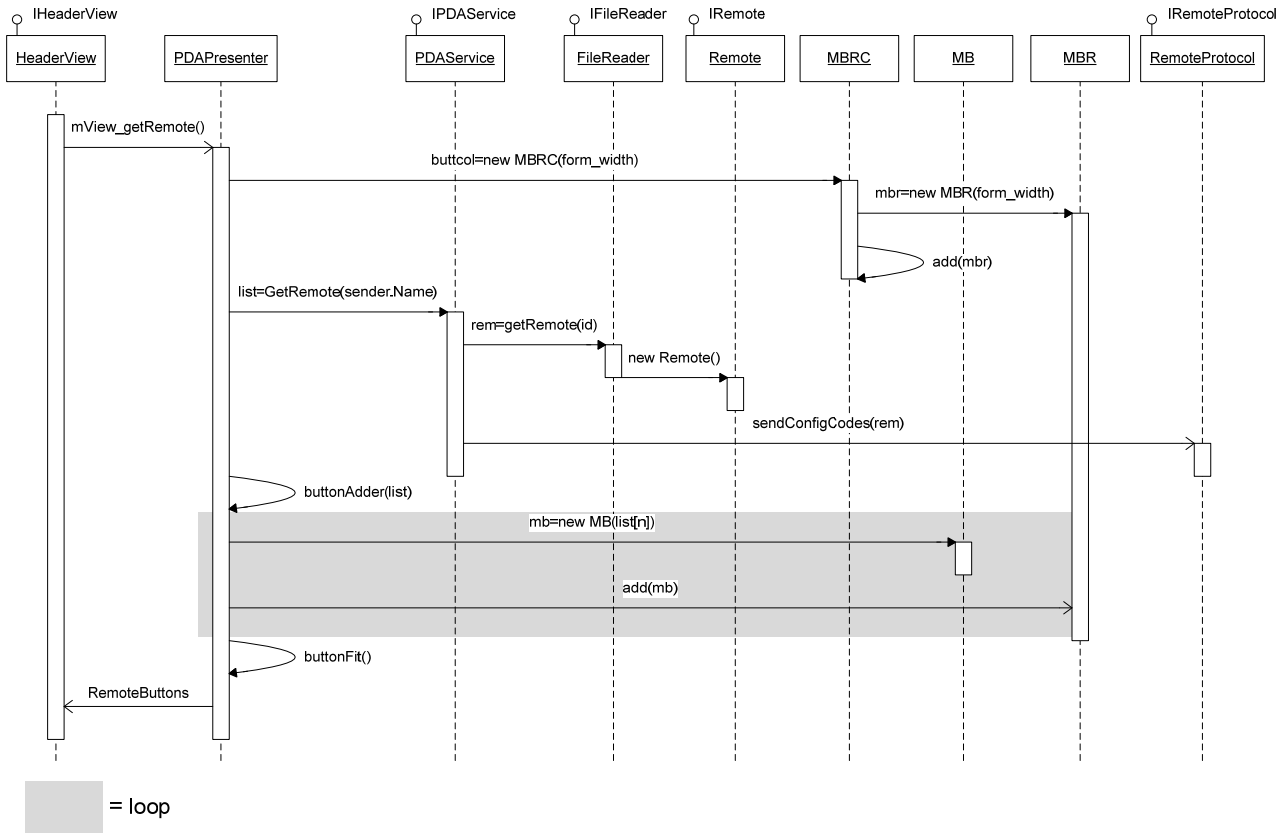


Figur 46 - Sekvensdiagram for ”Vis fjernbetjeninger”

PDAPresenter kaldes med funktionen ”mView_getHeaders()” og presenteren beder efterfølgende service-lagets PDAService om at indlæse og returnere en liste med fjernbetjeningsoversigten. PDAService kalder derfor FileReader i data-laget, som udtrækker og returnerer listen til PDAService. Da presenteren har modtaget listen, skal listen konverteres til knapper og der oprettes derfor en MyButtonRowCollection (MBRC), så der kan foretages en beregning omkring placering af disse. MBRC opretter en ny MyButtonRow (MBR) og tilføjer knapperne til denne, så fremt der er plads i rækken og så længe listen ikke

er endeligt gennemløbet. Presenteren anvender efterfølgende funktionen buttonFit() for at tilpasse knapstørrelsen og for at tilføje hver enkelt knap til HeaderView. Løkken i diagrammet forudsætter, at der ikke er mere end tre knapper, som til sammen ikke fylder mere end rækkens bredde. I tilfælde af dette oprettes en ny MBR, som tilføjes til den eksisterende MBRC.

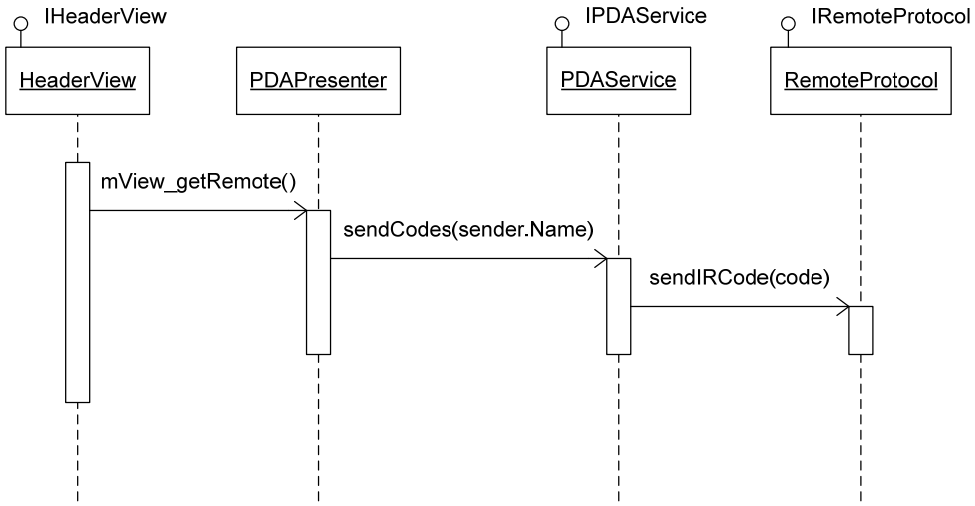
Det andet sekvensdiagram afspejler "Vælg fjernbetj.". Denne indhenter samtlige oplysninger tilknyttet den valgte fjernbetjening, lagrer dem i PDAService, sender konfigurationsdataene til protokollen "RemoteProtocol" og opdaterer HeaderView med fjernbetjeningens knapper. For diagrammet gælder samme regler for løkken som i forrige diagram.



Figur 47 - Sekvensdiagram for "Vælg fjernbetj."

Konfigurationsdata afsendt til RemoteProtocol har til formål at konfigurere hardware-fjernbetjeningen, som efterfølgende vil være klar til at modtage og afsende IR-koder i form af IR-signaler.

De to forrige diagrammer fører til tredje og sidste sekvensdiagram, som afspejler ”Vælg knap”. Hardwarefjernbetjeningen bør nu være klar til at modtage IR-koder og brugeren kan derfor vælge en af de indlæste knapper som illustreret i Figur 48.



Figur 48 - Sekvensdiagram for ”Vælg knap”

Klassen ”PDAPresenter kaldes” med funktionen ”mView_getRemote()”. Denne bestemmer ud fra knappens indhold, hvilken en funktion der skal udføres. Hvis afsenderen af eventet er en knap indeholdende en IR-kode, tilskriver presenteren klassen ”PDAService” med IR-koden, som videresender den til protokollen ”RemoteProtocol” i bluetooth-laget.

For samtlige tilskrivninger af data til ”RemoteProtocol” gælder, at de lagres i en lokal fil til test-formål.

3.5. Konklusion af design for CepaIR

Der er gennem dette kapitel blevet udarbejdet et sæt løsningsmodeller til problemstillingerne identificeret og beskrevet i analyseafsnittet. Formålet har været at strukture de designede applikationer ensartet for bedst muligt at kunne genbruge og dermed sikre en symbiose mellem applikationerne. Brugen af patterns har været effektiv og givet pæne resultater, men har i visse tilfælde krævet yderligere tilpassede løsninger.

PC-applikationen er blevet fuldt ud designet. Problemet omkring ugyldig syntaks i LIRC-filerne forventes at kunne blive håndteret ved at foretage en udrensning før gennemlæsning af hver enkelt linje tekst. Der er ikke blevet erfaret yderligere problemer eller flaskehalse under designet af denne.

Server-applikationen er ligeledes designet fuldt ud og den tidligere omtalte løsning omkring en brugerindtastet beskrivelse af hver redigeret fjernbetjening menes at være integreret effektivt. Problemstillingen omkring GUI'ens brugervenlighed er også blevet løst ved at alle valg foretages i samme GUI, som der løbende synkroniseres med nye data. Der er ikke under udarbejdelsen af designet fundet nye problemer.

PDA-applikationen er ikke blevet designet fuldt ud sammenlignet med kravspecifikationen. Det har ikke været muligt at få indsigt i Cepas nye kommunikationsklasse i tide, og det har derfor været nødvendigt at udarbejde et fleksibelt design med mulighed for en fremtidig udbygning. Ud over dette opfylder designet fuldt ud både kravspecifikation og use cases. Valget af en XML-fil som data-formidler samt den udarbejdede struktur af denne tyder på at være et effektivt valg. Det udarbejdede design af GUI'en til håndtering af multiple knapper, vil muligvis give kvaler under implementeringen, men vil hvis implementeret korrekt give den pæneste og mest fleksible løsning.

Da designet af alle tre applikationer er fundet tilfredsstillende, kan implementeringsfasen startes. Diagrammer samt beskrivelser udarbejdet under designet vil danne grundlag for implementeringsløsningerne, som udmunder i de endelige prototyper.

4. Implementering af CepaIR

Formålet med dette kapitel er, at dokumentere essentielle dele af den implementerede løsning og gennem denne dokumentation drage paralleller til beslutninger og resultater fra forrige kapitler. Da der i dette kapitel kun dokumenteres et mindre uddrag af koden, vil der i den endelige kildekode blive indsat beskrivelser i form af XML og almindelige kommentarlinjer. Med denne metode kan koden læses og forstås uafhængigt af rapporten, hvilket gør fremtidig videreudvikling lettere. Kapitlet opdeles i tre underafsnit, startende med PC-applikationen efterfulgt af server-applikationen og PDA-applikationen.

4.1. Implementering af PC-applikationen

PC-applikationen har, som planlagt i designet, 6 lag, og hvert lag varetager et specifikt ansvar. View-laget, som varetog interaktion med brugeren, skulle ligeledes struktureres ifølge Model View Presenter og stod derfor for oprettelsen af den tilknyttede presenter i formens nedarvede funktion *"OnLoad()"*.

```
private RemotePresenter presenter;

/// <summary>
/// funktionen er nedarvet fra Form-klassen og er tidligste adgangstidspunkt
/// i formens livscyklus. Presenteren oprettes derfor her
/// </summary>
protected override void OnLoad(EventArgs e)
{
   base.OnLoad(e);
   this.presenter = new RemotePresenter(this);
}
```

MainForm.cs linie 14 - 24

Ved oprettelse af presenteren foretages et kald fra presenterens constructor til funktionen *"initialize()"* og overtager hermed håndtering af de brugerforårsagede events.

```
/// <summary>
/// knytter events til presenteren ved initialisering
/// </summary>
private void initialize()
{
   this.view.parse += new EventHandler<EventArgs>(parsing);
   this.view.scan += new EventHandler<EventArgs>(scanning);
}
```

RemotePresenter.cs linie 32 – 39

Et klik på en knap placeret på formen vil derfor håndteres af presenteren, såfremt presenteren har tilknyttet en funktion via en delegate. Et klik på knappen med karaktererne "…" på PC-applikationens form, vil derfor ende ud med et kald til presenterens funktion *"scanning()"*.

```
/// <summary>
/// indhenter valgt sti fra view og videresender til service
/// med henblik på scanning for potentielle LIRC-filer
/// </summary>
private void scanning(object sender, EventArgs e)
{
   string tmp = this.view.Path;    //stien valgt
   if (!String.IsNullOrEmpty(tmp))
   {
      this.view.Path = tmp;
      this.view.Count = this.service.ScanForRemoteFiles(tmp).ToString();
      this.view.Progress = "";
   }
}
```

RemotePresenter.cs linie 41 – 54

Presenteren kan efterfølgende via viewets implementerede interface, indhente den valgte sti, som ønskes scannet for LIRC-filer og anmode service-klassen om, at igangsætte en scanning af den indhentede sti. Resultatet af scanningen opdateres endeligt på formen, via det implementerede interface. Ovenstående fremgangsmåde for kommunikation mellem view, presenter og service er fælles for de tre applikationer og dækker nogenlunde funktionaliteten for Model View Presenter. Fremgangsmåden vil derfor ikke blive beskrevet synderligt i de kommende afsnit, i stedet vil der blive lagt vægt på implementering af essentielle funktioner, samt mere avancerede løsningsmetoder.

Klassen ”TreeRunner” i data-laget indeholder funktionen *”search(string dir)”*, som rekursivt gennemløber undermapperne for en tilsendt sti. Dette er den essentielle funktionalitet for use casen ”Lokaliser LIRC-filer”, som under designet blev beskrevet som en filsøger.

```
/// <summary>
/// foretager rekursivt gennemløb af medsendte sti's undermapper
/// og lagrer fil-stier i listen "pathslist"
/// </summary>
public void search(string dir)
{
   string[] filer = Directory.GetFiles(dir, DEFAULT_EXTENSION);
   foreach (string fil in filer)
   {
      this.pathslist.Add(fil);
   }
   string[] mapper = Directory.GetDirectories(dir);
   foreach (string mappe in mapper)
   {
      if(((File.GetAttributes(mappe))&(FileAttributes.ReparsePoint))!=
           FileAttributes.ReparsePoint)
      search(mappe);
   }
}
```

TreeRunner.cs linie 23 – 40

Funktionen starter med at identificere filer befindende i rodmappen og tilføjer evt. fund til listen *"pathslist"*. Efterfølgende identificeres evt. undermapper og for hver fundet undermappe foretages et rekursivt kald. For ikke ved en fejl at komme ind i en uendelig løkke, benyttes *"FileAttributes.ReparsePoint"* til at stoppe gennemløb før det rekursive kald foretages. Samtlige filer med den korrekte "DEAFULT_EXTENSION" placeres dermed i *"pathslist"*, som er tilgængelig fra service-klassen.

Parseren, som gennemlæser de fundne LIRC-filer, fungerer via en "StreamReader", som læser filen en linie ad gangen. Dog er det muligt at aflæse to af fjernbetjeningens attributter, blot ved at opdele filens absolutte sti.

```
//følgende 3 linjer udtrækker informationer udelukkende indeholdt
//i filens sti: Producent og enhedsnavn
string[] fold = path.Split(new char[] { '\\' });     //opdeler sti i foldernavne
rem.Manufactor = fold[fold.Length - 2];              //udtrækker Producent
rem.Filename = (fold[fold.Length - 1]).Replace(".txt", ""); //udtrækker enhedsnavn
```

RemoteParser.cs linie 47 – 51

Ved at splitte stien bliver det muligt, at aflæse navnet på filens mappe, som i alle tilfælde er producenten, altså attributten *"manufactor"*. Denne attribut kunne ikke altid findes i filens indhold og det blev derfor under analysen besluttet, at hente den fra stien. Den anden attribut, som kan aflæses, er *"Filename"*. Navnet på hver enkelt fjernbetjeningsmodel var kun noteret korrekt i en del af filerne og da filnavnene viste sig at være mere konsistente, var det oplagt at anvende disse som helgardering. Parseren påbegynder efterfølgende gennemlæsningen og navigerer ud fra kendte positioner i filen.

```
if (line.StartsWith("begin codes")){ code = true; }//filens knapsektion påbegyndes
else if (line.StartsWith("end codes"))//filens knapsektion gennemlæst og flag done sættes
{ code = false; done = true; }
else if (!line.StartsWith("#")) //hvis linjen ikke er en kommentarlinje påbegyndes læsning
{
   string[] str = line.Split(new Char[]{ ' ', '\t' });//opdeler linje ved whitespace og tab
   str = untilClean(str);       //sender den opdelte linje til udrensning af andet uønsket
   if (code == false)  //stadig i sektion for konfig.data
   {
      if (!String.IsNullOrEmpty(str[0]))
      {   //sammenligner resterende indhold af linje med ønskede data-navne
         if (str[0].StartsWith("name")) { rem.Name = str[1]; }
```

RemoteParser.cs linie 61 – 74

Hver genkendt position i filen medfører ændringer i indsatte flag, som afgør parserens state. Eksempelvis betyder "begin codes" at filens knapsektion er nået, mens "end codes" betyder at der ikke er flere ønskede data i filen. Linier begyndende med karakteren "#" springes over, da dette er kommentarlinier og ikke skal anvendes. Linien splittes op så kun naturlige tekststrenge resterer og disse placeres i et string-array, som sendes til udrensning af lus. Det blev tidligere erfaret at filerne kunne indholde fejlagtig syntaks, som måtte håndteres med en udrensende funktion *"untilClean(str)"* under implementeringen.

```
private string[] untilClean(string[] p)
{
   string[] str = p;   //buffer
   int j = 0;
   for (int i = 0; i < p.Length; i++)  //checker hver char i string
   {
      p[i] = p[i].Replace("\'", " "); //fjerner "sql-killers"
      if ((p[i] != " ") && (p[i] != null) && (p[i] != "#") &&//fjerner uønskede chars
      (p[i] != "") && (p[i] != "\t"))
      { str[j] = p[i]; j++; }
   }
   return str; //returnerer renset stringsamling
}
```

RemoteParser.cs linie 111 – 123

"UntilClean()" foretager ganske simpelt en gennemlæsning/sammenligning af hver karakter og undlader at kopierer ugyldige karakterer over i string-arrayet som returneres. Indholdet af string-arrayet er efterfølgende klar til udtrækning af data, hvis beskrivelse nu befinder sig på den første plads i arrayet.

Når et komplet remote-objekt er blevet udtrukket af filen skal det lagres i databasen, som befinder sig på en MySQL-server. Selve oprettelsen af forbindelsen er forholdsvis standard og behøves derfor ikke beskrevet nærmere, mens mapperen som anvendes til bl.a. konstruktion af SQL-sætningerne, godt kan beskrives kort.

Mapperen indeholder funktionen *"addRemoteToDB(IRemote rem)"*, som modtager et interface til aflæsning af data fra et remote-objekt.

```
/// <summary>
/// samler og sender de konstruerede sql-sætninger fra "addRemote"
/// og "addRemoteButtons" til connection-gatewayen
/// </summary>
public void addRemoteToDB(IRemote rem)
{
   string sqlexpression = addRemote(rem);
   sqlexpression += LAST_KEY_QUERY + addRemoteButtons(rem);
   congate.insertNewData(sqlexpression);
}
```

RemoteMapper.cs linie 101 - 110

Funktionen kalder de to lokale funktioner *"addRemote()"* og *"addRemoteButtons()"*, der hver konstruerer en SQL-sætning, som til sidst samles og sendes til klassen "ConnectionGateway".

For at kunne gøre brug af tabellen Remote's primærnøgle, er det nødvendigt at lagre denne i en server-variabel. Nøglen er nemlig server-genereret og vil blive overskrevet, så snart tabellen Codes får tildelt en anden primærnøgle, hvilket sker ved lagring af den første fjernbetjeningsknap.

```
private const String LAST_KEY_QUERY = "SET @last := LAST_INSERT_ID();";
```

RemoteMapper.cs linie 14

Hver SQL-sætning konstrueret til indsætning af en fjernbetjeningsknap i tabellen "Codes", indeholder derfor selv, en reference til denne variabel, der skal fungerer som tabellens fremmednøgle.

```
private string addRemoteButtons(IRemote rem)
{
   string sqlexpression = null;
   foreach (RemoteButton rb in rem.RemoteButtons)
   {
      string code_data = String.Format("('{0}','{1}',@last)", rb.Command, rb.Code,
rem.UnitID);
      sqlexpression += String.Format("INSERT INTO {0} VALUES {1};", CODES_TABLE,
code_data);
   }
   return sqlexpression;
}
```

RemoteMapper linie 41 – 50

En hel fjernbetjening kan derfor trods et højt antal knapper, lagres under et enkelt forbindelsesopkald, hvilket sparer tid ved brug af de efterhånden meget almindelige bredbåndsforbindelser. Yderligere er der ved at lade mapper-klassen håndtere de model-specifikke opgaver, holdt en fornuftig afgrænsning for "DatabaseConnection", som holdes komplet uvidende om modellens udformning, mens mapperen holdes uvidende om databasens placering.

4.2. Implementering af server-applikationen

Server-applikationen udvikles i ASP .NET, ved brug af Visual Studios "code behind", som gør det muligt, at udvikle en web-applikation med skarp opdeling mellem html og programmeringskode. Identificering og tilpasning af en ønsket fjernbetjening starter med valg af producent, valg af fjernbetjening og endelig knapper, samt beskrivelse. Den indtastede beskrivelse benyttes som løsning på problemet omkring navngivning af fjernbetjeningen, som ofte blot var et serienummer uden indlysende relation til det elektroniske apparat. Disse oplysninger videresendes af presenteren til RemoteService's *"getCustomRemote()"*, som anmoder database-lagets klasse RemoteMapper om at udtrække og returnere data tilhørende den redigerede fjernbetjening. RemoteService tilføjer derefter beskrivelsen til remote-objektet og returnerer det til presenteren, som tilføjer remote-objektet til dens WebSession. Presenteren opdaterer til sidst kurven Basket, så brugeren er klar over, at hans hidtidige arbejde er lagret.

```
this.mSession.Add(this.mRemoteService.getCustomRemote(this.mView.SelectedUnit,this.
mView. SelectedCodes,this.mView.Description));

for (int i = 0; i < mSession.Count; i++)
{
   if (mSession[i] is Remote)
   {
      txt += "\n" + ((Remote)mSession[i]).Manufactor.ToUpper() + " : " +
((Remote)mSession[i]).Filename;

   }
}
this.mView.Basket = txt;
```

RemoteConfigPresenter.cs linie 120 – 128

Den/de redigerede fjernbetjeninger befinder sig derfor i sessionen, indtil brugeren beslutter sig for at downloade sit arbejde iform af XML-filen.

```
private void mView_downloadFile(object sender, EventArgs e)
{
   List<IRemote> lst = new List<IRemote>();
   for (int i = 0; i < mSession.Count; i++)
   {
      if (mSession[i] is Remote)//sikrer casting
      {
         lst.Add(((IRemote)mSession[i]));
      }
   }
   this.mView.File = this.mRemoteService.createFile(lst);
}
```

RemoteConfigPresenter.cs linie 54 – 65

XML-filen skal genereres ud fra brugerens lagrede arbejde og ovenstående kode viser hvordan hvert remote-objekt udtrækkes fra sessionen af presenteren og tilføjes en liste, som videresendes til RemoteService's funktion *"createFile(List<IRemote> lst)"*.

"createFile()" opretter en StreamWriter, en XmlSerializer, et byte-array og en MemoryStream.

```
StreamWriter stWriter = null;
XmlSerializer xmlser;
byte[] buffer;
try
{
   xmlser = new XmlSerializer(remcol.GetType());
   MemoryStream memStream = new MemoryStream();
   stWriter = new StreamWriter(memStream);
   xmlser.Serialize(stWriter, remcol);
   buffer = memStream.ToArray();
   memStream.Flush(); memStream.Close();
}
catch (Exception ex) { throw ex; }
finally { if (stWriter != null)stWriter.Close(); }
return buffer;
```

RemoteService.cs linie 86 – 100

XmlSerializeren anvendes til serialisering af listen, som i ovenstående kode kaldes *"remcol"*. StreamWriteren benyttes til at tilskrive den oprettede MemoryStream med den serialiserede liste og MemoryStreamens indhold konverteres endelig til byte-arrayet, som returneres til presenteren.

Presenteren tilskriver til sidst indholdet af byte-arrayet til view-lagets klasse *"Default.aspx"*, som gennem sin nedarvning fra Page-klassen, kan generere filen til download for brugeren.

```
if (File != null)
{
   String filename = "remotes.xml";
   Response.Clear();
   Response.AppendHeader("Content-Disposition", "attachment;filename=" +  filename);
   Response.ContentType = "application/xml"; ;
   Response.BinaryWrite(File);
   Response.Flush();
   Response.Close();
}
```

Default.aspx.cs linie 197 – 206

På denne måde hentes data fra databasen, redigeres efter brugerens ønske og placeres i en XML-fil, som endelig overføres til PDAens lagerkort. Videreudviklingen af model- og database-laget er for server-applikationens vedkommende, blevet foretaget i de samme projekter, som anvendes i PC-applikationen. Underprojekterne for de to lag er tilføjet til server-projektet og ændringer/tilføjelser for disse lag skal derfor findes i PC-projektets undermapper.

4.3. Implementering af PDA-applikationen

PDA-applikationen anvender i modsætning til de to andre .NET's Compact Framework (.NET CF), der som nævnt i analysen af dette, viser sig at have begrænsninger omkring XML-håndtering. Der blev derfor under designforløbet udarbejdet en løsningsmodel, som gjorde brug af frameworkets "XmlTextReader" til målrettet at læse og udtrække ønsket data. Ved opstart af applikationen ønskes det at fremvise en fjernbetjeningsoversigt for brugeren og denne anmodes derfor allerede i formens (HeaderView) *"OnLoad()"*, som ligeledes for .NET CF er arvet fra Form-klassen. Dette håndteres efterfølgende af presenteren "PDAPresenter".

```
void mView_getHeaders(object sender, EventArgs e)
{
   this.mView.InfoText = "Alle fjernbetjeninger";//oplyser bruger om menustate
   this.mView.Remove();
   buttcol = new Custom.MyButtonRowCollection(this.mView.fWidth);
   this.buttonAdder(this.mService.LoadRemotes());//henter fj.betj. oversigt
}
```

PDAPresenter.cs linie 72 - 78

PDAPresenter opdaterer først infoteksten på formen og fjerner efterfølgende eventuelle knapper med funktionen *"Remove()"*. PDAPresenter opretter en MyButtonRowCollection (MBRC) med formens bredde som parameter, så indlæste knapper kan tilføjes med det samme. PDAPresenter anmoder derefter service-laget om indlæsning af oversigten med *"LoadRemotes()"*, hvilket service-laget udfører ved forespørge data-lagets klasse *"FileReader"*.

```
public List<string[]> getRemotes()
{
  List<string[]> look = new List<string[]>();
  if (File.Exists(file1))
  {
    string[] head;
    StreamReader streamread = new StreamReader(file1);
    XmlTextReader textread = new XmlTextReader(streamread);
    textread.WhitespaceHandling = WhitespaceHandling.None;

    textread.MoveToContent();
    while (textread.Read())
    {
      if (textread.AttributeCount > 0)
      {
        head = new string[2];
        for (int i = 0; i < textread.AttributeCount; i++)
        {
          textread.MoveToAttribute(i);
          //udtrækker fj.betj.-beskrivelsen
          if (textread.Name == "Description") { head[0] = textread.Value; }
          //udtrækker fj.betj.-ID
          if (textread.Name == "UnitID") { head[1] = textread.Value; }
        }
        look.Add(head);
      }
    }
    textread.Close();   //lukker reader
    streamread.Close(); //lukker stream
```

```
    }
    return look;
}
```

FileReader.cs linie 158 – 189

FileReaders funktion *"getRemote()"* er meget specifik for CepaIRs valgte XML-struktur og gennemlæser vha. StreamReader og XmlTextReader filen fremadrettet og linie for linie. På grund af den valgte XML-struktur er det muligt at konkludere, at der ved samtlige elementer med to attributter, er tale om attributterne "Description" og "UnitID". Disse to blev under designet af server-applikationen besluttet, at anvende som repræsentation af en fjernbetjening, med "Description" som knaptekst og "UnitID" som identificerende ID. Samtlige elementer med to attributter får derfor aflæst attributterne, som lagres i en liste og til sidst returneres til forespørgende klasse. Når PDAPresenter får listen retur fra service-laget, sendes denne til en lokal funktion kaldet *"buttonAdder()"*.

```
private void buttonAdder(List<string[]> strarr)
{
    Custom.MyButton mb;
    foreach (string[] str in strarr)
    {
        mb = new Custom.MyButton(str[0], str[1]);

        //ikke mere plads i row
        if ((mb.RealWidth > buttcol[buttcol.Current_row].Spaceleft) ||
(buttcol[buttcol.Current_row].Count == 3))
        {
            buttonFit();                              //fitter rows når de er fyldt
            buttcol.newCurrentRow(mb.Height + 2);     //lav ny row
        }
        buttcol[buttcol.Current_row].Add(mb);         //tilføjer knap til row
    }
    buttonFit();                                       //fitter den sidste row
}
```

PDAPresenter.cs linie 84 – 100

"buttonAdder()" anvendes til at oprette og tilføje knapper af typen MyButton (MB). For hver knaprække må der højst placeres tre knapper og knapperne må ikke til sammen være breddere end formen. Hvis dette kriterie opfyldes, tilføjes MB til rækken og hver MB's bredde tilpasses til rækken med en anden lokal funktion kaldet *"buttonFit()"*.

```
private void buttonFit()
{
    buttcol[buttcol.Current_row].fitToRow();          //tilpas knapper til fyldt row
    foreach (Custom.MyButton but in buttcol[buttcol.Current_row])
    {
        this.mView.RemotesButton = but;               //tilføjer fitted knap til panel
    }
}
```

PDAPresenter.cs linie 106 – 113

"buttonFit()" meddeler klassen *"MyButtonRow"* om at tilpasse udseendet af knapperne med et kald til funktionen *"fitToRow()"*, som inkrementere hver MB's bredde med én, indtil der ikke er mere restplads i rækken. *"buttonFit()"* tilføjer endelig hver MB i rækken til formens knap-panel, som opdateres af viewets *"RemotesButton"*.

På denne måde udtrækkes en header for hver fjernbetjening, danner en knap som tilføjes formen og gør brugeren i stand til at vælge den fjernbetjening, som ønskes taget i brug.

Når brugeren vælger en fjernbetjening, sendes dennes id (UnitID) på samme måde til FileReaders *"getRemote(string id)"*, som sammenligner id'et med indholdet af hver attribut indtil et identisk findes.

```
while (xtr.Read() && stop)                              //kør indtil slut eller stoppet
{
  if (xtr.NodeType == XmlNodeType.Element)               //dette er et element
  {
    nodename = xtr.Name;
    if (nodename == "Remote"&&xtr["UnitID"].ToString()==id)//fjernbetjening fundet: START
    {
      found = true;                                                 //sæt flaget
      irem.UnitID = Convert.ToInt32(xtr["UnitID"]);                 //gem attributter
    }
    else if (xtr.Name == "Remote" && found) { stop = false; } //fjernbetjening slut: STOP
  }
  if (xtr.NodeType == XmlNodeType.Text)                  //dette er tekst som skal gemmes
  {
    if (found && (xtr.Depth == 3))                      //Konfig.data er på dybden 3
    {
      switch (nodename)
      {
        case "Aeps":
        irem.Aeps = Convert.ToInt32(xtr.Value);
        break;
        case "Bits":
        irem.Bits = Convert.ToInt32(xtr.Value);
```

FileReader.cs linie 39 – 61

FileReader kan derefter ud fra dybden i XML-dokumentets træstruktur, afgøre hvilken type informationer som er til rådighed på dette niveau og udtrække dem til et oprettet remote-objekt. Ovenstående kode viser konfigurationsdataenes placering i dybden 3, mens nedenstående kode viser at knapdata kan aflæses i dybden 5.

```
else if (found && (xtr.Depth == 5))       //knapdata er på dybden 5
{
  switch (nodename)
  {
    case "Command":
      com = xtr.Value;
      break;
    case "Code":
      cod = xtr.Value;
      break;
```

FileReader.cs linie 128 – 137

Remote-objektet returneres efterfølgende til klassen *"PDAService"* i service-laget, som lagrer konfigurationsdataene og udelukkende returnerer de udtrukne knapdata til presenteren. PDAService sender remote-objektet til protokol-klassen *"RemoteProtocol"* i bluetooth-laget, mens PDAPresenter tilføjer knapperne til formen på samme måde som tidligere.

Funktionaliteten af RemoteProtocol og resten af bluetooth-laget er for denne prototype holdt til "proof-of-concept", men det vil alligevel kort blive beskrevet hvorledes det fungerer.
RemoteProtocol tilknytter en funktionsbyte til hver af de enkelte data i remote-objektet og sender disse til klassen "BluetoothGateway", som simulerer en transmission ved at skrive dataene ned i en fil på PDAens lagerkort.

```
public void sendConfigCodes(IRemote rem)
{
   bg = new BluetoothGateway();
   bg.send("********************");
   if (rem.Aeps!=null) bg.send("0x80: " + rem.Aeps.ToString());
   if (rem.Bits != null) bg.send("0x81: " + rem.Bits.ToString());
```

RemoteProtocol.cs linie 33 - 38

Efter at den ønskede fjernbetjening er udtrukket af XML-dokumentet og dennes knapper placeret på formen, indeholder hver af knapperne nu sine egne IR-data og ved et tryk på en af disse afsendes den valgte knaps IR-data. IR-data videreformidles af PDAPresenter og PDAService og ender i RemoteProtocol, som behandler dem på samme måde som konfigurationsdataene, ved at tilføje en funktionsbyte.

```
public void sendIRCode(string code)
{
   bg = new BluetoothGateway();
   bg.send("0x99: " + code);
}
```

RemoteProtocol.cs linie 23 – 27

Samtlige data er dermed sendt til com-boksen i denne simulerede bluetooth-transmission. Bluetooth-laget kan let udskiftes og gøres anvendeligt når den endelige kommunikations-klasse er færdig og dermed kan prototypen gøres brugbar, ved blot at implementere denne.

4.4. Konklusion af implementering for CepaIR

Der er gennem dette kapitel blevet forsøgt, at uddybe funktionaliteten af de essentielle processer, som udgør de tre applikationer og gør disse i stand til at samarbejde om det overordnede mål. Applikationernes opbygning følger det udarbejdede design og der er derfor ikke nogle væsentlige tilføjelser.

PC-applikationen er blevet implementeret fuldt og gør det muligt at udtrække specifikke data fra adskillige LIRC-filer og gemme dem på en MySQL-server. Yderligere er der blevet implementeret en løsning med henblik på håndtering af ukorrekt syntaks filerne, ved at foretage en udrensning af hver enkelt linie før udtrækning af data. Der kan på senere tidspunkt tilføjes en ændring i klassen RemoteParser, som mindsker intensiteten af diskforbruget under parsing. Kravspecifikationen er opfyldt og samtlige identificerede problemer håndteret.

Server-applikationen er ligeledes implementeret fuldt og gør det muligt for en almindelig computerbruger, at identificere, konfigurere og downloade en fjernbetjeningssamling. Implementeringen afspejler designet og der er derfor ligeledes implementeret den udarbejdede løsning omkring vedhæftning af brugerens egen beskrivelse af en fjernbetjening. Designet kunne muligvis have været mere simpelt, men den implementerede løsning gør det muligt, hurtigt og sikkert at udskifte, samt ændre de implementerede komponenter. Kravspecifikationen er ligeledes for server-applikationen opfyldt.

PDA-applikationen er ikke blevet implementeret fuldt i forhold til kravspecifikationen, da den parallelle udvikling i firmaet hverken gjorde det muligt at designe eller implementere det endelige bluetooth-lag. Der er derfor i begge faser forsøgt udarbejdet en fleksibel løsning, som ved fremtidig udvikling kan udskiftes eller ændres efter behov. Cepa har løbende givet adgang til forskellige delvise løsninger af kommunikationsklassen, men der har ikke på noget tidspunkt foreligget et design eller en endelig løsning. PDA-applikationen er dog på trods af dette blevet ret fleksibel, da det bliver muligt at implementere den i CepaCom, blot ved udskiftning af view-laget. View-laget har hverken kendskab til modellen eller de dynamisk tilføjede grafiske komponenter, MBRC, MBR og MB. Ligeledes kan data-laget anvendes direkte i en endelig løsning, hvis Cepa ønsker at beholde den nuværende XML-struktur.

5. Test af CepaIR

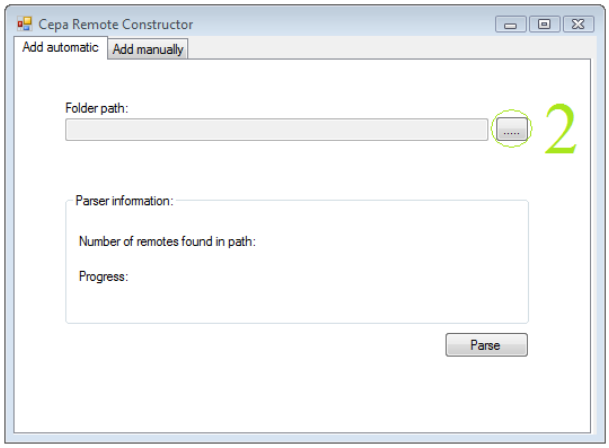
Da implementeringen er fuldendt skal applikationernes funktionalitet testes. Dette skal gøres for at sikre, at de implementerede løsninger fungerer efter hensigten. Til testformålet udabejdes en use case-test af hver applikation, som udføres af en ansat i Cepas udviklingsafdeling, Thomas Pedersen (tp@cepa.dk). Use case-testen udføres på PCen anvendt under udviklingen. Valget af test skyldes applikationernes status som prototyper, da der ellers ville være anvendt en white box test, såsom unit testing. Beskrivelsen af testudførelsen for hver applikation suppleres af en til flere beskrivende skærmbilleder, samt et udførligt sæt instruktioner, som skal udføres af testpersonen.

5.1. Test af PC-applikationen

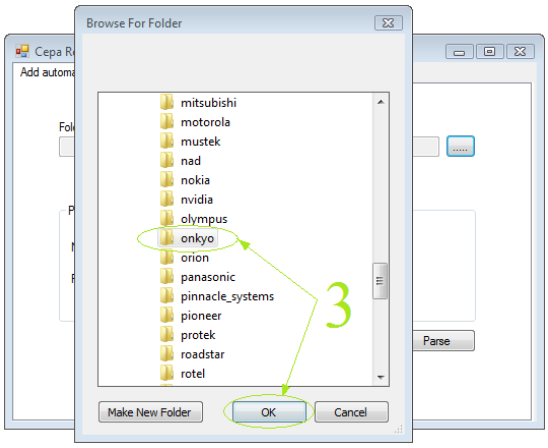
PC-applikationen har kun to use cases og funktionaliteten af disse kan ikke sammenlignes. Af den grund udføres test af samtlige use cases for denne applikation.

5.1.1. Lokaliser LIRC-filer

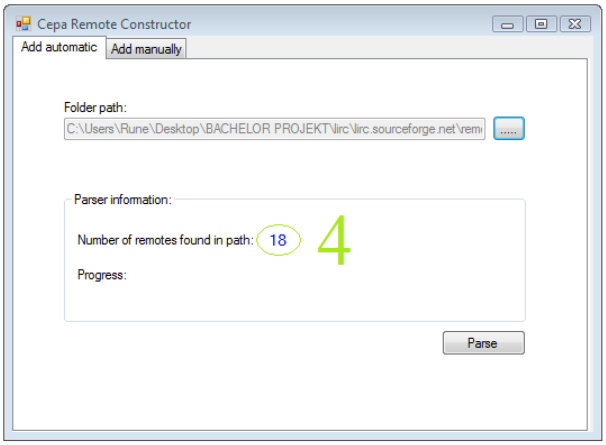
Titel:	**Use case ”Lokaliser LIRC-filer”**
Formål:	At teste den implementerede løsning af use casen ”Lokaliser LIRC-filer”, ved at verificere output som korrekt.
Beskrivelse:	Testen vil blive udført efter valget af en konkret mappe er blevet foretaget.
Forudsætninger:	At LIRC-filerne er blevet downloadet og placeret i en mappe tilgængelig for PCen.
Testforløb:	1. Start PC-applikationen 2. Tryk på knappen med karaktererne ”.....”. (se Figur 49) 3. Vælg mappen ”Onkyo” fra mappen med de downloadede LIRC-filer og tryk på knappen ”Ok”. (se Figur 50) 4. Den valgte mappe gennemsøges og antallet af fundne filer vises i infoboksen. (se Figur 51)



Figur 49 - Screenshot 1 PC-applikation



Figur 50 - Screenshot 2 PC-applikation

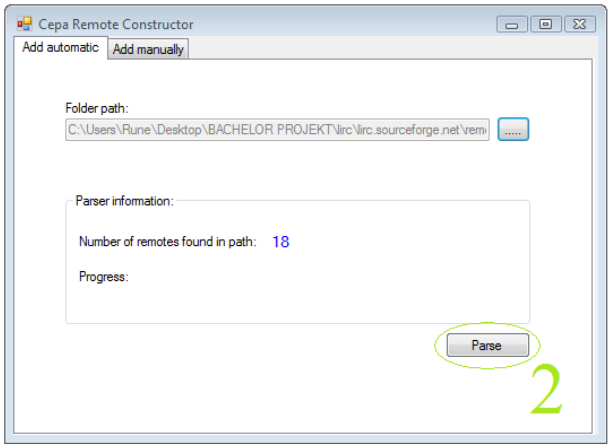


Figur 51 - Screenshot 3 PC-applikation

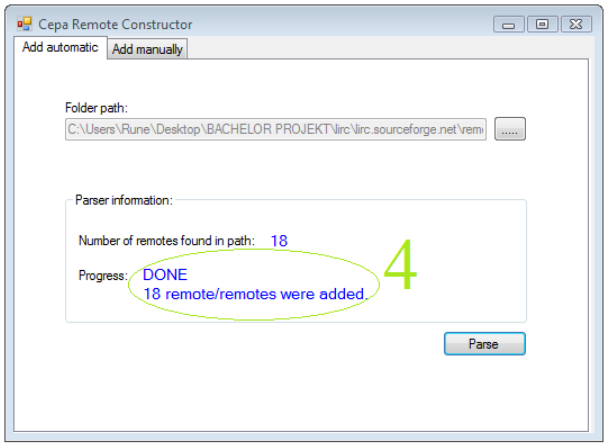
Resultat:	Antallet af LIRC-filer fundet, vises i brugerfladens infoboks
Konklusion:	Succes

5.1.2. Scan og gem LIRC-data

Titel:	Use case ”Scan og gem LIRC-data”
Formål:	At teste den implementerede løsning af use casen ”Scan og gem LIRC-data”, ved at verificere output som korrekt.
Beskrivelse:	Testen vil blive udført efter lokalisering af LIRC-filer er foretaget.
Forudsætninger:	1. Udførelse af ”Lokalser LIRC-filer”. 2. MySQL-server tilgængelig for anvendte PC.
Testforløb:	1. Udfør ”Lokaliser LIRC-filer” og vælg igen mappen ”Onkyo” 2. Tryk på knappen med teksten ”Parse” (se Figur 52) 3. De lokaliserede LIRC-filer parses og LIRC-dataene lagres på MySQL-serveren 4. Antallet at lagrede filer vises i infoboksen (se Figur 53)



Figur 52 - Screenshot 4 PC-applikation



Figur 53 - Screenshot 5 PC-applikation

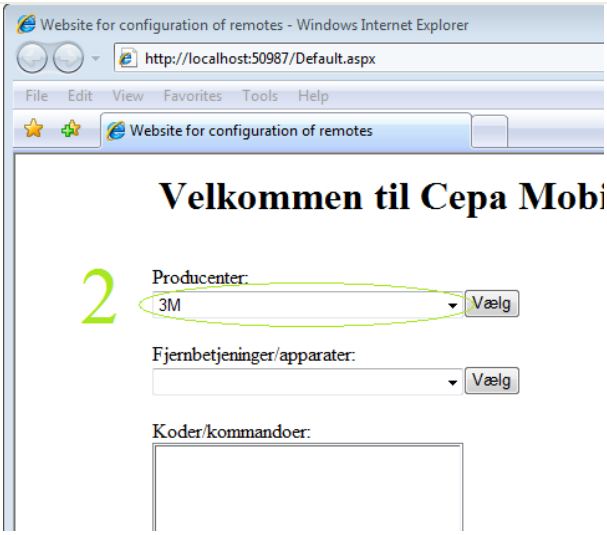
Resultat:	Applikationen underretter om igangværende konvertering af filerne. Da konverteringen er færdig, bliver antallet af lagrede filer vist i infoboksen.
Konklusion:	Succes

5.2. Test af server-applikationen

Flere af server-applikationens use cases minder en del om hinanden og derfor vil der kun være inkluderet tests af følgende: ”Hent Producentoversigt”, ”Definer og gem fjernbetj.”, samt ”Generer/hent fil”.

5.2.1. Hent Producentoversigt

Titel:	**Use case ”Hent Producentoversigt”**
Formål:	At teste den implementerede løsning af use casen ”Hent Producentoversigt”, ved at verificere indhentede data som korrekte.
Beskrivelse:	Testen vil blive udført efter åbning af den dynamiske hjemmeside Default.aspx
Forudsætninger:	1. Parsing og lagring af LIRC-data udført på forhånd. 2. MySQL-server tilgængelig for anvendte PC.
Testforløb:	1. Kør server-applikationen i Visual Studio 2005 2. Listen over producenter hentes og vises i dropdownlisten. (se Figur 54)

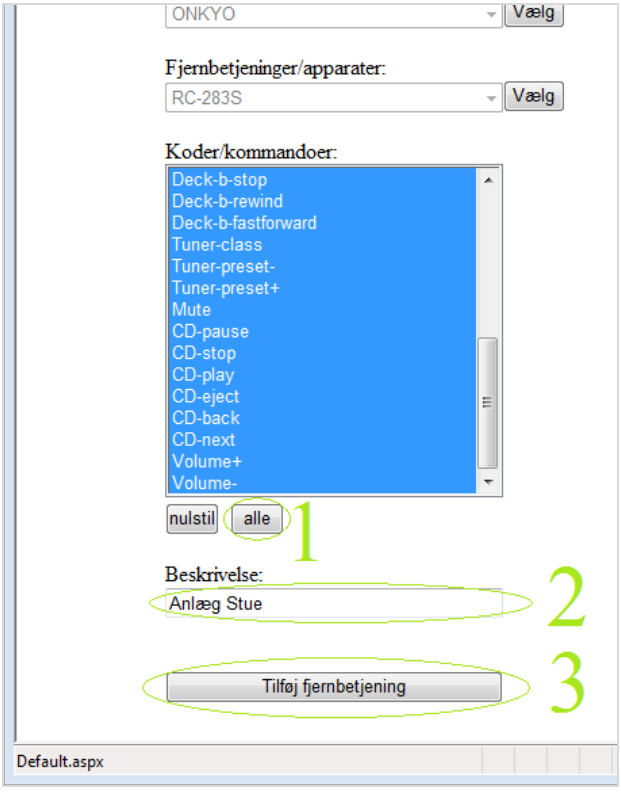


Figur 54 - Screenshot 1 server-applikation

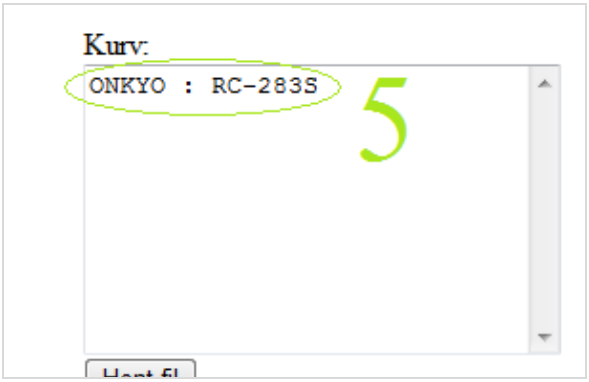
Resultat:	Producentlisten hentes og vises i dropdownlisten
Konklusion:	Succes

5.2.2. Definer og gem fjernbetj.

Titel:	Use case ”Definer og gem fjernbetj.”
Formål:	At teste den implementerede løsning af use casen ”Definer og gem fjernbetj.”, ved at verificere lagring af det udførte arbejde
Beskrivelse:	Testen vil blive udført efter valg af producent (Onkyo)og efter valg af fjernbetjening (RC-283S). Knapperne er derfor på dette tidspunkt hentet og vises listboksen.
Forudsætninger:	1. ”Hent producentoversigt” udført 2. ”Hent fjernbetjeninger” udført 3. ”Hent knapper” udført
Testforløb:	1. Vælg alle knapper fra listboksen, ved at trykke på knappen med teksten ”alle”. (se Figur 55) 2. Indtast beskrivelsen af fjernbetjeningen under teksten ”Beskrivelse:”. Skriv ”Anlæg Stue” i textboksen. (se Figur 55) 3. Tryk på knappen ”Tilføj fjernbetjening”. (se Figur 55) 4. Fjernbetjeningen hentes og lagres i sessionen. 5. Kurven opdateres med oplysninger omkring den lagrede fjernbetjening. (se Figur 56)



Figur 55 - Screenshot 2 server-applikation



Figur 56 - Screenshot 3 server-applikation

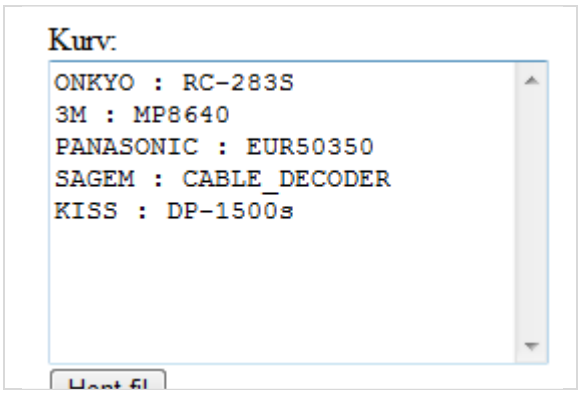
Resultat:	Ved tryk på ”alle” markeres alle fjernbetjeningsknapper. Ved tryk på ”Tilføj fjernbetjening” bliver den redigerede fjernbetjening lagt i kurven.
Konklusion:	Succes

5.2.3. Generer/hent fil

For at få en brugbar fil ud ad denne test, bør foregående uses cases gentages, så kurven indeholder mere end en enkelt fjernbetjening. Gentag med følgende input:

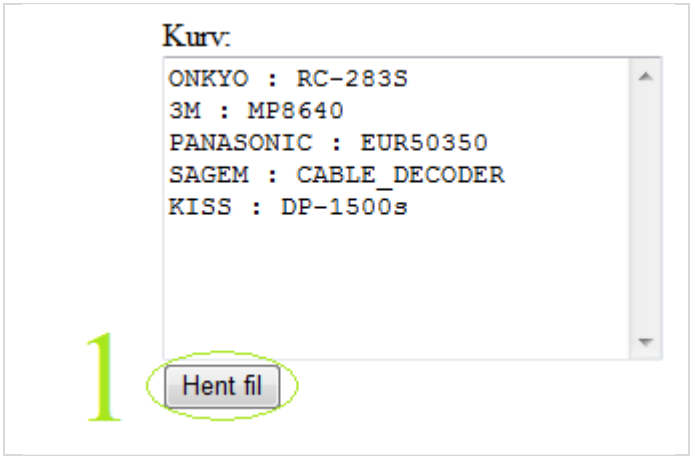
Producent	Fjernbetjening	Knapper	Beskrivelse
3M	MP8640	”Standby/On” ”Volume Up” ”Volume Down” ”Menu”	Projektor Stue
Panasonic	EUR50350	alle	TV Soveværelse
Sagem	CABLE_DECODER	alle	Kabelbox Stue
KISS	DP-1500s	”power” ”eject” ”play” ”stop” ”chap+” ”chap-”	DVD Soveværelse

Hvis udført korrekt, bør kurven nu se ud som følgende:



Figur 57 - Screenshot 4 server-applikation

Titel:	**Use case ”Generer/hent fil”**
Formål:	At teste den implementerede løsning af use casen ”Generer/hent fil”, ved at verificere generering og download af filen
Beskrivelse:	Testen vil blive udført efter tilføjelse af de 5 fjernbetjeninger, omtalt i starten af dette afsnit.
Forudsætninger:	Gentagen udførelse af de tidligere nævnte use cases
Testforløb:	1. Tryk på knappen ”Hent fil”. (se Figur 58) 2. Indholdet af sessionen serialiseres. 3. Filen gøres tilgængelig for brugeren som download. 4. Download og placer filen ”remotes.xml” på PDAens lagerkort



Figur 58 - Screenshot 5 server-applikation

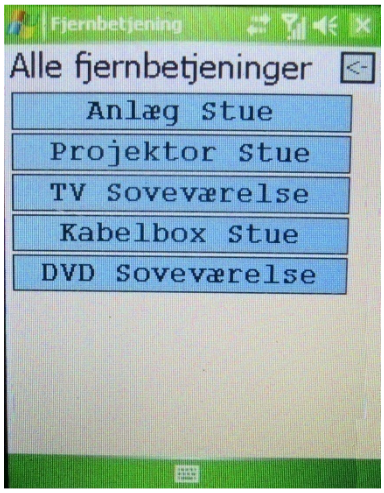
Resultat:	Efter tryk på ”Hent fil” fremkommer en downloadboks med tilbud om download af filen ”remote.xml”.
Konklusion:	Succes

5.3. Test af PDA-applikationen

PDA-applikationen består af tre use cases, som alle vil blive testet i dette afsnit. De tre use case tests vil gøre brug af xml-filen ”remotes.xml”, som blev downloadet og placeret på lagerkortet som afslutning af testen for server-applikationen.

5.3.1. Vis fjernbetjeninger

Titel:	**Use case ”Vis fjernbetjeninger”**
Formål:	At teste den implementerede løsning af use casen ”Vis fjernbetjeninger”, ved at verificere udtrukne xml- data som korrekte.
Beskrivelse:	Testen udføres ved start af PDA-applikationen
Forudsætninger:	At filen genereret under test af server-applikationen er blevet overført til PDAens lagerkort.
Testforløb:	1. Kør PDA-applikationen i Visual Studio 2005 2. Listen over fjernbetjeningernes beskrivelse hentes og vises som knapper. (se Figur 59)

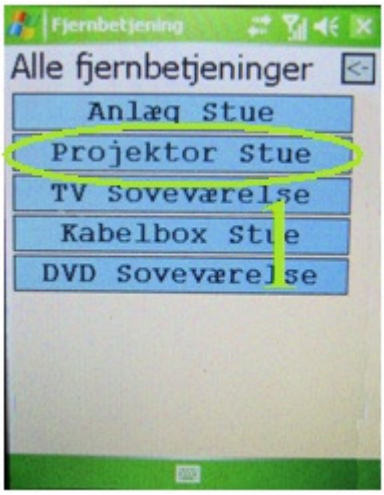


Figur 59 - Screenshot 1 PDA-applikation

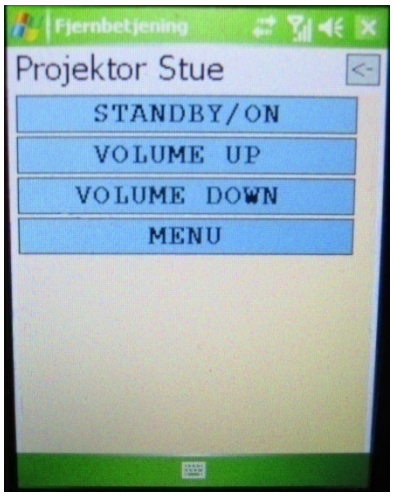
Resultat:	Listen over understøttede fjernbetjeninger fremkommer på skærmen.
Konklusion:	Succes

5.3.2. Vælg fjernbetjening

Titel:	Use case ”Vælg fjernbetjening”
Formål:	At teste den implementerede løsning af use casen ”Vælg fjernbetjening”, ved at verificere udtrukne xml-data som korrekte.
Beskrivelse:	Testen vil blive udført efter tryk på en af knapperne med tekst svarende til en beskrivelse for en fjernbetjening
Forudsætninger:	1. At use case testen for ”Vis fjernbetjeninger” netop er blevet udført. 2. At PDA-applikationen ikke er blevet genstartet siden forrige test.
Testforløb:	1. Tryk på knappen med teksten ”Projektor Stue”. (se Figur 60) 2. Knapper tilhørende den valgte fjernbetjening hentes og vises på skærmen. (se Figur 61)



Figur 60 - Screenshot 2 PDA-applikation

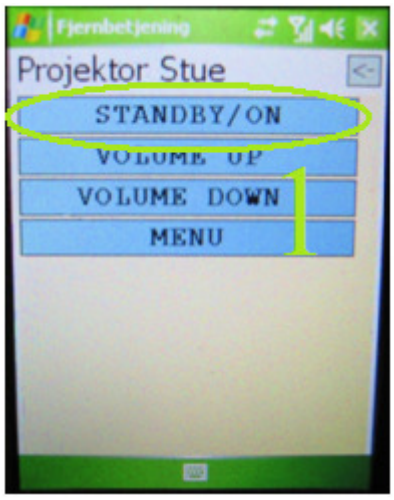


Figur 61 - Screenshot 3 PDA-applikation

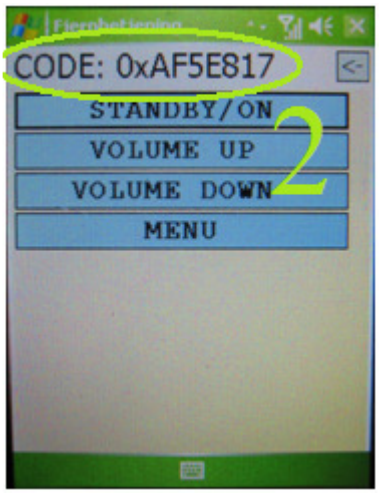
Resultat:	Efter tryk på ”Projektor stue” fremkommer en liste bestående af fjernbetjeningens knapper
Konklusion:	Succes

5.3.3. Vælg knap

Titel:	Use case ”Vælg knap”
Formål:	At teste den implementerede løsning af use casen ”Vælg knap”, ved at verificere udført handling som korrekt.
Beskrivelse:	Testen vil blive udført efter at knapperne for en fjernbetjening er blevet hentet og vist på skærmen.
Forudsætninger:	1. At use case testen for ”Vælg fjernbetjeninger” netop er blevet udført. 2. At PDA-applikationen ikke er blevet genstartet siden forrige test.
Testforløb:	1. Tryk på knappen med teksten ”STANDBY/ON”. (se Figur 62) 2. IR-data for den valgte knap sendes og vises yderligere i tekstboksen over knapperne. (se Figur 63)



Figur 62 - Screenshot 4 PDA-applikation



Figur 63 - Screenshot 5 PDA-applikation

Resultat:	IR-koden tilhørende den valgte fjernbetjeningsknap fremkommer i toppen af skærmen, hvor fjernbetjeningens navn før blev vist.
Konklusion:	Succes

5.5. Konklusion af test for CepaIR

De anvendte use case-tests er blevet udført i et kronologisk forløb, så det ligeledes har været muligt at teste sammenhængen mellem applikationerne. Overordnet har applikationerne virket som forventet og de anvendte tests har kun givet inspiration til ønskede ændringer og tilføjelser.

PC-applikationen:
Tests af PC-applikationen forløb uden uhåndterede fejl, men benytter dog harddisken intensivt under parsing af LIRC-filer. Dette skyldes at parseren ikke er trådstyret, hvilket ville gøre det muligt at indsætte "idle-time" i processen. Resultatet af hver use case-test har ellers bekræftet at applikationen virker som forventet.

Server-applikationen:
Tests af server-applikationen forløb ligeledes uden fejl og der blev ikke konstateret nogle uhåndterede flaskehalse. Applikationen virkede som forventet.

PDA-applikationen:
Applikationen blev testet ud fra de tidligere nævnte forbehold omkring det manglende endelige kommunikations-lag. Der blev derfor med succes anvendt ekstra feedback til brugeren, som kunne aflæse delvise resultater i toppen af skærmen. Det er muligt ud fra resultatet af de udførte use case-tests at konkludere, at applikationen virker som forventet.

6. Konklusion for CepaIR

Formålet med dette kapitel er at opsummere og evaluere det gennemgåede projektforløb, vurdere fremtidige udvidelser og forbedringer, samt endelig konkludere udbyttet for den studerende og for virksomheden. Det sekundære projekt, CepaBackup, blev grundet omfanget af CepaIR aldrig påbegyndt, hvormed denne konklusion kun omfatter CepaIR, der fra begyndelsen fik første prioritet.

Projektets analysefase gjorde det muligt at identificere en del flaskehalse, som medfølger ved anvendelse af uvante formater og teknologier. Analysen gjorde det ligeledes muligt at beslutte i hvilke faser flaskehalsene skulle håndteres og nogle kunne endda håndteres ved minimalt at udvide en aktørs ansvarsområde. Der blev identificeret og udarbejdet use cases for de tre applikationer, som blev defineret og beskrevet i tæt overensstemmelse med kravspecifikationen. I designfasen blev der fokuseret på applikationernes arkitektur og tætte samarbejde, hvilket krævede at der også blev set frem mod applikationen som skulle anvende data genereret af applikationen under udvikling. Der blev udviklet en brugervenlig GUI til alle tre applikationer, som dog let kunne udskiftes, da disse ville blive implementeret under Model View Presenter. Under designfasen blev det besluttet at foretage en afgrænsning omkring anvendelse af Cepas egen kommunikationsklasse til PDA-applikationen, da denne endnu ikke var færdigudviklet. Der blev derfor indgået et kompromis om at færdiggøre prototypen til PDA'en, så Cepa selv kunne indsætte den færdige kommunikationsklasse i designets protokol-lag. Under implementeringsfasen blev de sidste flaskehalse håndteret og løsninger til nogle af teknologiernes begrænsninger udviklet. Applikationerne blev derfor implementeret i overensstemmelse med kravspecifikationen og kompromiserne indgået under designfasen. Testfasen blev udført af Cepa-udvikler Thomas Pedersen, der kunne konkludere, at prototyperne fungerede som planlagt uden opståede fejl. Der kan derfor leveres tre applikationer til Cepa, som kan færdigudvikles og tages i brug.

6.1. Forbedringer

For at kunne gøre omfattende brug af de udviklede prototyper, bør følgende funktioner udvikles og implementeres:

PC-applikationen:

- Design og implementering af trådstyret parser for at nedsætte intensiteten af applikationens udnyttelse af harddisken
- Mulighed for manuel indtastning af nye konfigurations- og IR-data.

Server-applikationen:

- Implementering af auto postback så dropdownlister automatisk opdateres uden tryk på separat knap.
- Redigering af indholdet i kurven herunder Rediger og Slet

PDA-applikationen:

- Implementering af Cepas auto-scannende menusystem.
- Implementering af Cepas endelige kommunikationsklasser.

6.2. Konklusion

Arbejdet med CepaIR har været meget lærerigt og spændende. Det har været muligt at anvende flere forskellige teknologier under Microsoft .NET, hvilket har givet et bedre indblik i mulighederne for udvikling til PC, server og tynde klienter, såsom mobil og PDA. Resultatet af projektet er meget tilfredsstillende, selvom det har været nødvendigt at afgrænse éns egne ambitioner og ideer fordi kravene var opfyldt eller deadlinen var nået. Det har været svært at samle alle trådene op, efter at have måtte slippe projektet under min meget ufrivillige bolignød og det understreger igen vigtigheden i at nedfælde tanker og ideer med det samme. Tilbage på rette spor blev man dog hurtigt fordybet igen og som en forandring fra daglig undervisning, få mulighed at intensivere sit arbejde med et enkelt projekt.

Følgende er kommentarer fra Cepa-udvikler, Thomas Pedersen, som udførte testen. Kommentarerne skal opfattes som feedback til de udarbejdede prototyper og er brugbare ved senere udvidelser eller forbedringer:

PC-applikationen:
Lokalisering og parsing af LIRC-filer er let overskuelig og fungerer fornuftigt. Brugergrænsefladen har et meget minimalistisk design, hvilke gør interaktionen meget brugervenlig, da den er let overskuelig. Programmet kan det vi har ønsket og gemmer dataene ud fra de krav vi har stillet.

Server-applikationen:
Brugerfladen er opbygget meget intuitiv og brugervenligt. Starten med at vælge producent o.s.v. giver et meget naturligt flow igennem processen. Det kunne ønskes at der blev gjort brug af auto-feedback for dropdownlisterne i stedet for knapperne "Vælg". Det er en god ekstra detalje at man kan til-/fra-vælge knapper, i tilfælde af man ikke ønsker at gøre brug af samtlige. Løsningen mht. beskrivelse af fjernbetjeningen er en god ide, da brugerne kan have svært ved at forholde sig til producenternes kryptiske navngivning. Det mere overskueligt at slutbrugeren af PDA-applikationen, får teksten "Projektor stue" i stedet for "3M MP8640".
Applikationen kan mere end det vi har stillet krav til og kan med meget få tilføjelser tages direkte i brug.

PDA-applikationen:
Programmet er en god prototype og giver en godt blik på hvordan det kan implementeres i vores grænsefladestruktur. Tilpasningen af knappernes bredde til skærmens størrelse er en fornuftig løsning fremfor en knap pr. række.
Applikationen opfylder ikke den oprindelige kravspecifikation, men opfylder vores krav ud fra de kompromiser indgået senere i forløbet.

7. Litteraturliste

7.1. Bøger

Titel	Forfatter	Forlag	År
Writing Effective Use Cases	Alistair Cockburn	Addison-Wesley	2002
Applying UML And Patterns 2. Ed	Craig Larman	Prentice Hall	2002
Microsoft .NET Compact Framework	Andy Wigley Stephen Wheelwright	Microsoft Press	2003
Database systems 3. Ed	Thomas Connolly Carolyn Begg	Addison-Wesley	2002
Microsft ASP .NET Programming With Microsoft Visual C# NET. Step By Step	G. Andrew Duthie	Microsoft Press	2003

7.2. Hjemmesider

Emne	Adresse
LIRC open source projektet	http://www.lirc.org
Controller Area Network	http://en.wikipedia.org/wiki/Controller_Area_Network
I^2C	http://en.wikipedia.org/wiki/I%C2%B2C
Agile Software Development	http://en.wikipedia.org/wiki/Agile_software_development
Facade pattern	http://www.dofactory.com/Patterns/PatternFacade.aspx
Model View Presenter pattern	http://martinfowler.com/eaaDev/ModelViewPresenter.html
Database design	http://www.mysql.com
.NET Programmering	http://msdn2.microsoft.com

8. Bilag

Bilag 1: LIRC-fil "RM-470.txt"

```
#
# this config file was automatically generated
# using lirc-0.6.5(any) on Thu Aug 29 23:05:05 2002
#
# contributed by
#
# brand:                       Sony
# model no. of remote control: RM-470
# devices being controlled by this remote: CD player
#

begin remote

  name  CD
  bits            7
  flags SPACE_ENC|CONST_LENGTH
  eps            30
  aeps          100

  header       2412   588
  one           612   588
  zero         1210   588
  ptrail       1210
  post_data_bits  4
  post_data      0x7
  gap          45604
  repeat_bit       0

      begin codes
          PLAY                     0x0000000000000059
          STOP                     0x0000000000000071
          PAUSE                    0x0000000000000031
          NEXT                     0x0000000000000039
          PREV                     0x0000000000000079
          FAST_FORWARD             0x0000000000000069
          FAST_BACKWARD            0x0000000000000019
          FORWARD                  0x0000000000000011
          BACKWARD                 0x0000000000000051
          VOLUME_UP                0x000000000000005B
          VOLUME_DOWN              0x000000000000001B
          RED                      0x0000000000000003
          GREEN                    0x0000000000000029
          YELLOW                   0x0000000000000023
          BLUE                     0x0000000000000043
          QUIT                     0x000000000000004B
```

```
            UP                          0x0000000000000007
            DOWN                        0x0000000000000002
            RIGHT                       0x0000000000000075
            LEFT                        0x0000000000000055
            OK                          0x0000000000000027
            MENU1                       0x000000000000000D
            MENU2                       0x0000000000000065
            1                           0x000000000000007F
            TWO                         0x000000000000003F
            3                           0x000000000000005F
            4                           0x000000000000001F
            5                           0x000000000000006F
            6                           0x000000000000002F
            7                           0x000000000000004F
            8                           0x000000000000000F
            9                           0x0000000000000077
            10                          0x000000000000007D
            11                          0x000000000000003D
            12                          0x000000000000005D
            13                          0x000000000000001D
            14                          0x000000000000006D
            15                          0x000000000000002D
            16                          0x000000000000004D
            17                          0x000000000000007B
            18                          0x000000000000003B
            19                          0x000000000000000B
            20                          0x0000000000000035
            ZERO                        0x000000000000000D
            CHECK                       0x0000000000000027
            CLEAR                       0x0000000000000007
      end codes

end remote
```

Bilag 2: Use Case Beskrivelse: Hent fjernbetjeninger

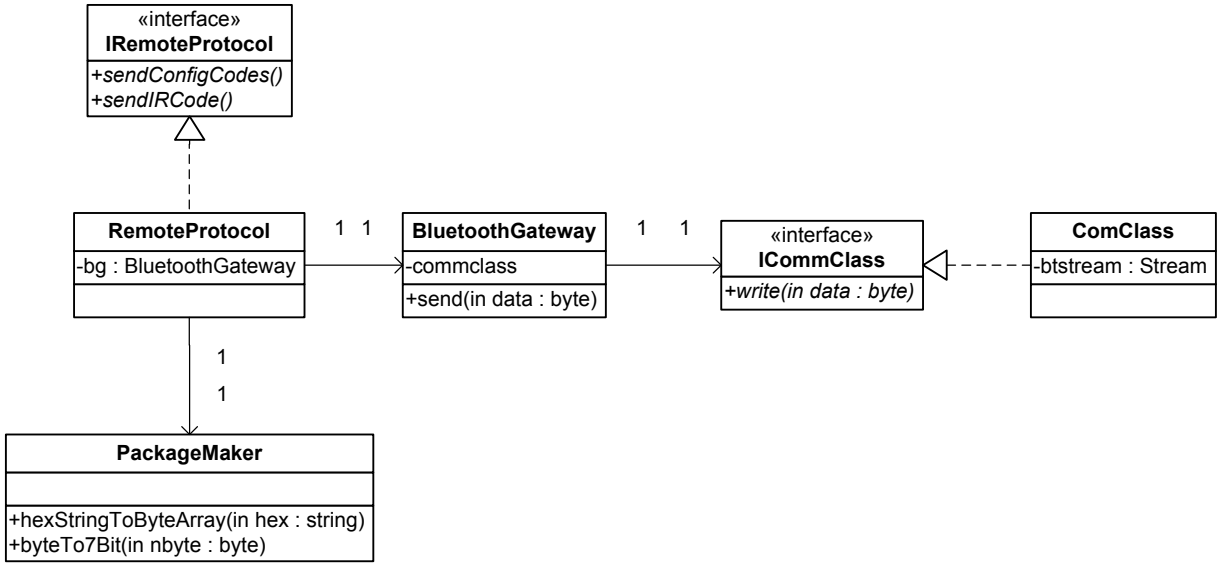
ID	Hent fjernbetjeninger
Formål	At få vist en liste over alle fjernbetjeninger fra en enkelt producent, så det bliver muligt, at finde fjernbetjeningen til et enkelt produkt
Primær aktør	Handicapmedhjælper
Motiver og interesser	At finde en specifik fjernbetjening til et produkt
Trigger	Brugerstyret kontrolelement
Forudsætninger	Udførsel af use casen ”Hent producentoversigt”
Succeskriterier	At få vist en liste over en producents fjernbetjeninger
Fejlbetingelser	3. Manglende internetforbindelse 4. Manglende databaseforbindelse
Hovedforløb	1. Aktøren vælger en enkelt producent fra producentoversigten. 2. Systemet henter en liste over alle fjernbetjeninger for den valgte producent
Udvidelser	
Hyppighed	Anvendes hver gang en fjernbetjening skal tilføjes ”indkøbskurven”

Bilag 3: Use Case Beskrivelse: Hent knapper

ID	Hent knapper
Formål	At få vist en liste over alle knapper tilhørende en fjernbetjening, så det bliver muligt, at vælge ønsket understøttet funktionalitet
Primær aktør	Handicapmedhjælper
Motiver og interesser	At få mulighed for, at vælge ønsket funktionalitet af fjernbetj.
Trigger	Brugerstyret kontrolelement
Forudsætninger	Udførsel af use casen ”Hent fjernbetjeninger”
Succeskriterier	At få vist en liste over en fjernbetjenings knapper
Fejlbetingelser	5. Manglende internetforbindelse 6. Manglende databaseforbindelse
Hovedforløb	1. Aktøren vælger en enkelt fjernbetj. oversigten 2. Systemet henter en liste alle knapper for den valgte fjernbetj.
Udvidelser	
Hyppighed	Anvendes hver gang en fjernbetjening skal tilføjes ”indkøbskurven”

Bilag 4: PDA-applikationen – tiltænkt bluetooth-lag

Bluetooth-laget skulle oprindeligt varetage protokol og datapakning før afsendelse, men blev fortrudt da applikationen efter endt udvikling skulle indsættes i CepaCom, hvor et nyt kommunikationslag var under udarbejdelse.



Figur 64 - PDA-applikationens tiltænkte bluetooth-lag

RemoteProtocol modtager enten et remote-objekt eller en ir-kode, som ønskes afsendt via bluetooth-forbindelsen. RemoteProtocol tilføjer de data-specifikke funktionbytes, så dataene kan adresseres korrekt og beder klassen ”PackageMaker”, om at pakke dem i henhold til tidligere analyserede pakkestruktur. Dataene sendes derefter til gatewayen og videre klassen ”ComClass”.

Bilag 5: CD-rom struktur

- Rapport.pdf : Den færdige rapport i PDF-format
- Forside.pdf : Rapportens forside i PDF-format
- Visual Studio 2005 : Projektmappe
  - RemoteConstructor : PC-applikationens Projektmappe
    - RemoteConfigurator.ModelLayer : PC-/Server-applikationernes model-lag
    - RemoteConstructor.ApplicationLayer : PC-applikationens data-lag
    - RemoteConstructor.DatabaseLayer : PC-/Server-applikationernes database-lag
    - RemoteConstructor.PresenterLayer : PC-applikationens presenter-lag
    - RemoteConstructor.ServiceLayer : PC-applikationens service-lag
    - RemoteConstructor.ViewLayer : PC-applikationens view-lag
  - WebRemoteConfig : Server-applikationens Projektmappe
    - WebRemoteConfig.PresenterLayer : Server-applikationens presenter-lag
    - WebRemoteConfig.ServiceLayer : Server-applikationens service-lag
    - WebRemoteConfig.WebSite : Server-applikationens view-lag
  - PDARemote : PDA-applikationens Projektmappe
    - PDARemote.BluetoothLayer : PDA-applikationens bluetooth-lag
    - PDARemote.DataLayer : PDA-applikationens data-lag
    - PDARemote.ModelLayer : PDA-applikationens model-lag
    - PDARemote.PresenterLayer : PDA-applikationens presenter-lag
    - PDARemote.ServiceLayer : PDA-applikationens service-lag
    - PDARemote.ViewLayer : PDA-applikationens view-lag