UNIVERSITETET I BERGEN
Det samfunnsvitenskapelige fakultet

SPIM – Secure Person Identification Module

Hvordan kan IT bistå til sikker identifikasjon og gjenkjennelse av personer i klinisk forskning (Clinical Trials) i lavinntektsland?

Jo Are Ingvaldsen

Institutt for Informasjons- og Medievitenskap, Universitetet i Bergen

Sammendrag

God, kvalitetssikret forskning i lavressurs omgivelser krever gode metoder som oppfyller internasjonale lover og reguleringer. En av utfordringene er å sørge for en sikker og pålitelig registrering og identifisering av deltakere i slike studier. FNs høykommisær for flyktninger (UNHCR) har utredet behovet for sikker registrering av personer i en flyktningsituasjon, noe som de mener er essensielt for å kunne tilby en holdbar situasjon. Dette kan overføres til gjennomførelsen av kliniske forsøk i lavinntektsland.

I denne oppgaven blir det beskrevet en applikasjon for registrering og identifisering av deltakere i kliniske forsøk i lavinntektsland. Det er blitt utviklet en prototype av en modul, kalt «Secure Person Identification Module» (SPIM), til «Open Mobile Electronic Vaccine Trials» (OMEVAC) rammeverket. Det ble foretatt en evaluering av prototypen i Mbale, Uganda på feltkontoret til «Promoting infant health and nutrition in Sub-Saharan Africa: Safety and efficacy of exclusive breastfeeding promotion in the era of HIV» (PROMISE). Her fikk feltarbeidere og data manageren teste SPIM, og komme med sine kommentarer til sine systemer. Oppgaven viser at et elektroniske og mobilt system for registrering og identifisering av deltakere i kliniske forsøk i lavinntektsland kan komme med et lite bidrag til å heve validiteten på den innsamlede informasjonen, men at det må gjøres flere undersøkelser for å kunne fastsette den reelle verdien av et slikt system.

Innholdsfortegnelse

Tabelliste

Figurliste

Forord

Arbeidet med masteroppgaven har vært spennende og lærerikt. Jeg har fått satt meg inn i og lært et nytt programmeringsspråk. Å utvikle for håndholdte maskiner har vært en nyttig og utfordrende erfaring. Det å utvikle programvare for, og å samarbeide med «Centre for International Health» har vært interessant. Jeg vil gjerne få takke CIH og «Open Mobile Electronic Vaccine Trials»-prosjektet (OMEVAC) for den muligheten jeg fikk med denne oppgaven.

Jeg vil takke min veileder, Weiqin Chen, som har vært til stor hjelp. Også Jørn Klungsøyr må jeg takke, han har fungert som en veileder og konsulent i utviklingen av SPIM. Til slutt vil jeg gjerne få takke Caleb Bwengye som var til stor hjelp under mitt opphold i Mbale, Uganda.

Jo Are Ingvaldsen[1], Bergen 29. Mai 2008.

[1] Jo Are Ingvaldsen
Mobil: 90577981
E-post: jo.ingvaldsen@student.uib.no

1 Introduksjon

1.1 Problemdomenet

Høykvalitets, global helseforskning i lavressurs omgivelser krever sikre, holdbare, nøyaktige, sammenlignbare, effektive og kost-effektive metoder for datainnsamling og styring for å møte de store utordringene innen helse vi har i verden i dag. Kliniske forsøk avgjør sikkerheten og effektiviteten til medisinskbehandling og medikamenter, og må utføres i samsvar med internasjonale reguleringer(ClinicalTrials.gov 2007).

I lavinntektsland kan det være en utfordring å entydig identifisere personer, mange mangler ID-papir og det finnes ofte ikke noen gode felles identifikatorer. Dette gjør at det kan være vanskelig å verifisere om den innsamlede dataen er gjort på rett individ.Alle slike forskningsprosjekter har system for å registrere deltakere, men disse er ofte papirbaserte og personerkan bli registrert mer enn en gang.Dette kan gi personen ulike fordeler, men samtidig kan det medføre komplikasjoner for personen og usikre data for forskerne. Med hjelp av IT-baserte systemer kan disse utfordringene bli enklere å løse.
Innen identifisering og gjenkjenning av personer det er tilknyttet store utfordringer til utførelsen av slike «clinical trials». Det brukes i dag store ressurser på å følge opp personene, sikre riktig behandling, hindre juks og til å generere rapporter i slike prosjekt. Teorien som vil danne en plattform for det arbeidetsom oppgaven går ut i fra er at et IT-system for gjenkjenning og identifisering av personer kan effektiviserer dette arbeidet, sikre dataintegritet, og samtidig forhindre dobbeltregistreringer. Dette vil sikre at det er en entydig kobling mellom innsamlet data og personen denne informasjonen omfatter.

FNs Høykommisær for Flyktninger (UNHCR) har utredet behovet for sikker registrering av informasjon i en flyktningssituasjon, noe som blant annet resulterte i oppstarten av «Project Profile». I referatet fra det første møte om «GLOBAL CONSULTATIONS ON INTERNATIONAL PROTECTION» skriver de at:
«Registration is an essential tool for effective protection, planning, timely delivery of adequate assistance, as well as the pursuit of appropriate durable solutions.»(UNCHR 2001)

Dette kan overføres til problematikken rundt registrering og identifisering i kliniske forsøk (Clinical Trails) i lavkostland.

1.2 Problemstilling

I denne oppgaven vil det bli foreslått og laget en prototype av et IT-system for registrering, identifisering og gjenkjenning av personer i ulike kliniske forsøk i lavinntektsland.

Programvaren som skal utvikles, i samarbeid med Senter for Internasjonal Helse (CIH) ved Universitetet i Bergen (UiB), og tenkes som en modul til et større rammeverk, slik at den kan brukes i ulike situasjoner. Det tenkes at denne modul skal settes inn i et rammeverk kalt «Open Mobile Electronic Vaccine Trials» (OMEVAC) som er finansiert av Norges Forsknings Råd (NFR).

Som identifiseringsgrunnlag tenkes det å undersøkemulighetene for bruk av fire ulike metoder, (1) biometriskdata, (2) bilde, (3) personopplysninger som navn og alder og, (4) andre informasjoner som: familierelasjoner, bosted, kjønn og midlertidig ID-kort. Disse opplysningene vil da kontrolleres mot hverandre for å verifisere identifikasjonen til en person.

I et slikt system dukker det opp flere etiske spørsmål knyttet til vern av personopplysninger. Det må kartlegges hvilke trusler som finnes for å lagre så store mengder med sensitive personopplysninger på samme sted, og hvordan det er mulig å sikre disse dataene mot å bli missbrukt. Det er viktig å få på plass både logisk sikkerhet og fysisk sikring av dataen.

Problemstillingen vil da bli: «*Hvordan kan ITbistå i identifiseringen og gjenkjennelsen av personer i klinisk forskning (Clinical Trials) i lavinntektsland*.»

1.3 Open Mobile Electronic Vaccine Trails (OMEVAC)

Kommende internasjonale standarder og reguleringer vil i løpet av noen få år kreve komplette elektroniske system for å styre et vaksinasjonsforsøk. Kliniske forsøk

gjennomført i lavinntekstland trenger å ha samme kvalitet og pålitelighet som sammenlignbare studier gjennomført i vestlige land. Dette vil kreve datainnsamlings- og håndteringssystem spesielt designet og utviklet for disse kravene. Forskningsdata i lavressurs omgivelser er nå for det meste samlet inn på papirskjemaer, en prosess som er utsatt for feil og ueffektivitet. Den manglende kontrollen og samsvar med studieprotokollen er en stor utfordring.

For å løse disse og relaterte problem vil OMEVAC bytte ut den papirbaserte prosessen med et hel-digitalisert, mobiltsystem for å gjennomføre kliniske forsøk. Forskere og feltarbeidere vil bruke håndholdte datamaskiner og registrere den innsamlede informasjonen direkte. Dette vil føre til drastisk reduksjon av de logistiske utfordringene knyttet til papirbehandling og digitalisering av disse. Dette prosjektet vil gjøre forskningsinstitusjoner i lavinntektsland i stand til å gjennomføre høykvalitets vaksinasjons forsøk.
Vi vil bygge videre på eksisterendeprosjekter:

1) EpiHandy, et generisk, elektronisk, «open source» og gratis mobil datasamlings- og styringssystem, som blir brukt i regulære studier i lavinntekstland.
2) R, et anerkjent og mye brukt «open source» statistiskanalyseverktøy, og
3) Nesstar, et publiseringssystem for forskningsdata. Disse vil danne et komplett og integrert elektroniske datakjede fra innsamlingspunktet, og frem til deling og publisering av dataene.

Hovedmålet til OMEVAC er å utviklet OMEVAC til å bli et komplett, sikkert, mobilt og elektronisk system for datainnsamling og styring av vaksinasjonsforsøk, fra kilde til publikasjon, bruk og som samsvarer med internasjonale standarder og reguleringer. Det vil bli testet i pågående forsøk i Afrika og bli tilgjengelig, kostnadsfritt med all kildekode og dokumentasjon.

OMEVAC har også syv sekundære mål: (1) Å utvikle et generisk system for datainnsamling og forsøksstyring basert på EpiHandy, (2) å utvikle et generisk og

elektroniske studieprotokollverktøy. (3) Oppfylle internasjonale standarder for kliniske forsøk, (4) utvikle sikre og forbedrede metoder for å koble deltakerne med den innsamlede dataen, (5) bedre muligheter for dataanalyse på studiestedene gjennom R statistiskpakke. (6) Bedre muligheter for datapublisering og deling ved kliniske forsøk gjennom Nesstar og (7) valider og test OMEVAC(2008).

Konsortiet, som utgjør OMEVAC, består av The Special Programme for Research and Training in Tropical Diseases/Verdens helseorganisasjon (TDR/WHO), HandheldsForHealth.org (Linux, Bangalore India), Makerere University Uganda, INDEPTH Network / Malaria Clinical Trial Alliacne, Norsk Samfunnsvitenskapelig Datatjeneste (NSD), Promise Consortium og to fakultet ved Universitetet i Bergen. To aktive forsøkskonsortium i Afrika vil teste programvareprototypene i feltet i løpet av utviklingsprosessen for å sikre at systemet blir brukbart og brukervennlig.

Senter for Internasjonal Helse (CIH) er koordinator forOMAVAC prosjektet ogble etablert i 1988. Det er et inter-fakultetlig senter med mål å starte, koordinere og gjennomføre forskning og kapasitetsbygning med betydning for samarbeidet mellom Universitetet i Bergen og lavinntektstland innenfor helsesektoren. I løpet av de årene senteret har eksistert har det utviklet fra et lite forskningssenter til å bli den ledende og største institusjonen innenfor dette feltet i Norge. Dette har blitt oppnådd gjennom en effektiv vitenskapelig og administrativ ledelse som støtter innovative og dynamiske initiativ fra de vitenskapeligansatte (2005).

SPIM er en modul i det rammeverket som utgjør OMEVAC. I prosjektbeskrivelsen til OMEVAC står det at i lavressurs miljø er offisiell og brukbar identifikasjonspapir som ID-kort og personnummer manglende. Noe som kan føre til at journaler blir blandet. Konsekvensene ved feil i koblinger mellom innsamlet data og deltaker, ved for eksempel å skrive inn feil ID-nummer, kan være katastrofale. Det første steget som må utføres før datainnsamlingen av situasjonsrapportskjemaer begynner, er å kunne fastslå identiteten til studiedeltakeren som dataen skal innsamles på. Gjennom SPIM vil det bli aktivt undersøkt potensielle løsninger på slike problem. Dette inkluderer å undersøke bruk av

biometri, bildeteknikker, GID (Geographic Information Systems) og kontekstuell informasjon om studiedeltakerne som familierelasjoner, adresse, kjønn, alder og mer. Forskjellige algoritmer og treffmønstre vil bli utviklet for å kunne fastslå identiteten til en studiedeltaker før det blir innsamlet noe informasjon. Prosjektet vil utforske etiske og sikkerhetsrelaterte problem for å unngå potensiell missbruk av sensitiv informasjon og implementere teknikker for å unngå slik missbruk. Norsk Samfunnsvitenskapelig Datatjeneste (NSD) har her en viktig rolle som ombudsmann for personvern ved Universitetet i Bergen (Klungsøyr 2007).

1.4 The PROMISE Project

«Promoting infant health and nutrition in Sub-Saharan Africa: Safety and efficacy of exclusive breastfeeding promotion in the era of HIV» (PROMISE) er et prosjekt koordinert av CIH, PROMISE er finansiert av EU. Målet med PROMISE er å promotere eksklusiv amming (ExclusiveBreastfeeding (EBF)) i en afrikansk setting for å vurdere hvilken påvirkning EBF har på spedbarns helse, og å finne den optimale perioden for EBF. Noen studier har vist at «peer consoling» kan endre holdninger til EBF drastisk, PROMISE vil gjennomføre de første randomiserte forsøkene for å utvikle og teste modeller for å innføre denne fremgangsmåten i fire afrikanske land. EBF kan senke spedbarns dødeligheten med 13%, og med ytterlige 2% om det ikke hadde vært for det faktum at amming overfører HIV(Tulleskär 2005).

OMEVAC utvikles i samarbeid mellom flere ulike grupper, der både brukerorganisasjoner og utviklerorganisasjoner er representert. En av brukerorganisasjonene er PROMISE, og deres bidrag til utviklingen vil være å komme opp med krav til systemet og være tilgjengelige for testing. I denne sammenheng ble PROMISE valgt som partner i testingen av SPIM.

2 Teori

I dette kapittelet vil det bli engjennomgang av det teoretiske grunnlaget for oppgaven. De ulike teoriene vil bli beskrevet, og valg av teorier begrunnet.

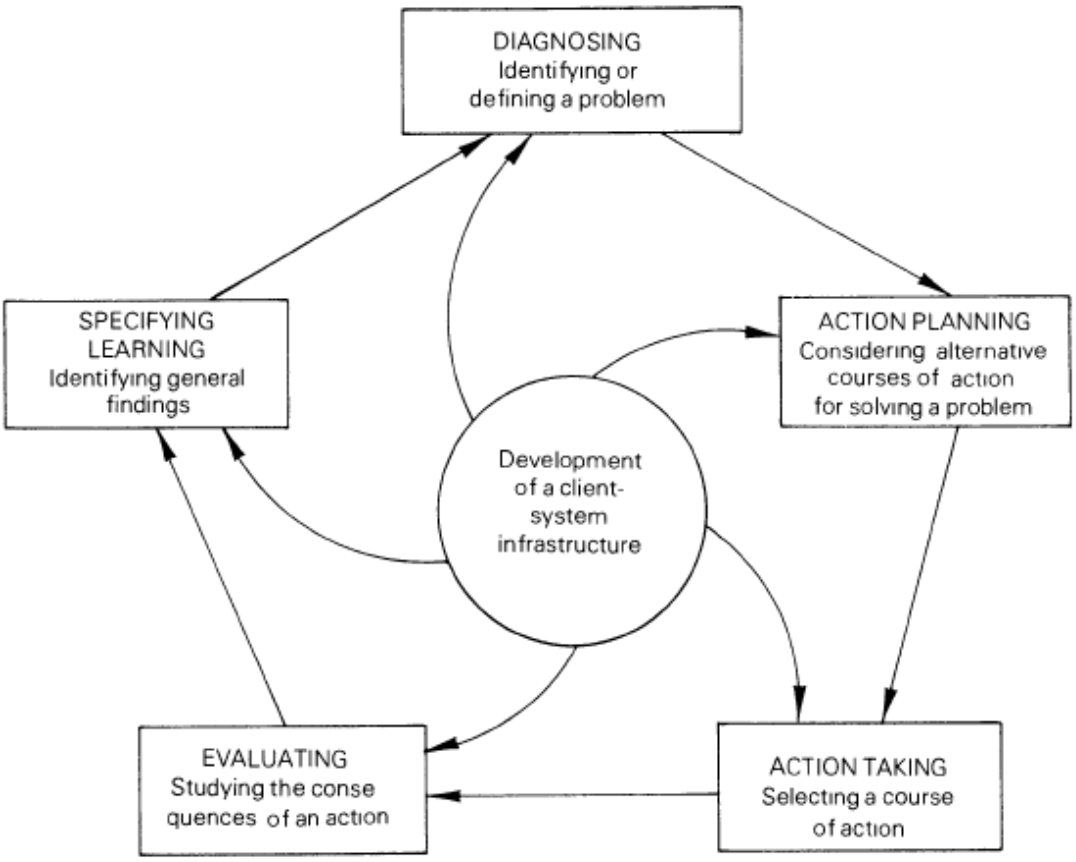
2.1 Action Research

I undersøkelsene vil oppgaven prøve å finne svar på hvordan teknologi kan sørge for sikker identifiserer og gjenkjenne av personer i kliniske forsøk i lavinntekstsland. Det vil i løpet av masteroppgaven bli undersøkt hvordan et informasjonssystem (IS) for identifisering og gjenkjenning av personer, som oppgaven foreslår, kan gjøre identifiseringen enklere og mer pålitelig. Dette for å sikre at innsamlet data kan knyttes entydig opp mot en person. Måten oppgaven vil finne ut dette på er først å utvikle programvaren, så være aktivt med på å teste dette verktøyet i feltet. På grunn av ønsket om å gå inn å aktivt forandre måten arbeidet utføres, vil oppgavenbasere seg på «Action Research»(AR). AR kan defineres som:

«Action Research aims to contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within a mutually acceptable ethical framework.» (Rapoport 1970)

Denne metoden passer for undersøkelser rundt introduksjonen av teknologi i en organisasjon (de Villiers 2005). Det er foreslått et rammeverk med fire dimensjoner for AR, disse fire dimensjonene er: (1) Konseptuelt grunnlag, (2) studie design, (3) forskningsprosessen og (4) forventninger til rollene (Lau 1999).Arbeidet vil bli strukturertetter fem steg for AR (Figur 1). Disse fem stegene er:

(1) Diagnostisering,
(2) planlegging av handling,
(3) utførelse av handling,
(4) evaluering og
(5) spesifiserer læring. (Susman and Evered 1978)



Figur 1 – De fem stegene for Action Research(Susman and Evered 1978)

2.2 Programvareutvikling

I dette avsnittet vil det bli en gjennomgang av ulike teorier og metoder for utvikling av programvaresystemer.

2.2.1 Model-Driven Development (MDD)

Tradisjonelle ingeniør disipliner har en utstrakt bruk av modeller, det ville for eksempel være utenkelig å bygge en bro eller et sykehus uten å modellere dette først. Innen programvareutvikling derimot er det ikke like utstrakt bruk av modeller, og der de er brukt spiller de ofte en sekundær rolle. «Model-Driven Development» (MDD) metoder prøver å lage verktøy for å utnytte fordelen med bruk av modeller. I MDD er hovedfokus og hovedproduktet modeller og ikke programvarekode. Fordelen med dette er at produktet blir uavhengig av den underliggende teknologien, og at domeneekspertene blir bedre integrert i utviklingsprosessen. Modeller er også enklere å utvide og vedlikeholde enn spesifikke implementeringer. En fare ved MDD er at modellene kun blir dokumentasjon, det er derfor en forutsetning at programmer blir automatisk generert fra

modellene. Modeller har fem karakteristikker som gir en stor fordel, den viktigste karakteristikken er abstraksjon. At man kan lage ulike modeller med ulike abstraksjonsnivå for å få større innsikt i den delen av systemet man ønsker. Den andre karakteristikken er at modellen gjør det enklere å forstå systemet. Den tredje karakteristikken en modell må inneha er nøyaktighet. En god modell gir en nøyaktig beskrivelse av det systemet den skal representere. Forutsigbarhet er den fjerde viktige egenskapen en modell må inneha, man bør være i stand til, ut i fra modellen, å forutse interessante, men ikke åpenbare egenskaper til systemet. Den siste egenskapen til en modell er at den må være billigere å konstruere enn systemet den skal modellere (Selic 2003).

2.2.2 Agile Software Development

«*We are uncovering better ways of developing*
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions *over processes and tools*
Working software *over comprehensive documentation*
Customer collaboration *over contract negotiation*
Responding to change *over following a plan*

That is, while there is value in the items on
the right, we value the items on the left more.»

Figur 2 – Manifesto for Agile Software Development (Beck, Beedle et al. 2001)

Agil programvareutvikling, er ikke en metode, men en samlebetegnelse på flere ulike metodikker som har det til felles at de er lette og med fokus på små og raske leveranser av fungerende programvare. I 2001 samlet 17 representanter fra ulike miljø som eXtreme Programming, SCRUM, DSDM og Adaptive Software Development. Disse representantene dannet noe de kalte «the Agile Allinance» og ble enige om en felles

«filosofi» rundt programvareutvikling (Highsmith 2001). Resultatet av dette møtet ble presenteret i noe som ble kalt «Manifesto for Agile Software Development» (Figur 2).

Agil programvareutvikling er et begrep som omfatter programvareutviklingsmetodikker hvor tett samarbeid mellom utviklere og foretningseksperter; ansikt-til-ansikt kommunikasjon (som mer effektivt enn nedskrevet dokumentasjon); gjentatte leveranser av fungerende programvare og tette, selvorganiserende team, blir lagt vekk på (Alliance 2006).

2.2.3 Agile Model-Driven Development (AMDD)

«Agile Model-Driven Development» (AMDD) er en agil form for MDD som navnet sier. Der MDD krever grundige modeller modellert i avanserte verktøy, krever AMDD modeller som er så vidt gode nok. Man skal bruke de enkleste modelleringsverktøyene og lage enkle og raske modeller. Grunn tanken i MDD og AMDD er lik, det vil si at begge bygger på at modellene er sentrale for å kunne utvikle god programvare. Som navnet AMDD sier, bygger denne på agile metodikker, og deler verdier med andre agile metoder. Det vil si at «ansikt-til-ansikt» kommunikasjon, tett samarbeid mellom domeneekspertene og utviklerne, og gjentatte leveranser av fungerende programvare (Ambler 2004). I kapittel tre, Design og Utvikling, kommer oppgaventilbake til AMDD.

2.3 Evaluering av programvare

Evalueringer har tre hovedmål: å vurdere omfanget og tilgjengeligheten til systemets funksjonalitet, å vurdere brukernes opplevelse av interaksjonen og å identifisere spesifikke problemer med systemet (Dix, Finlay et al. 2004). Det finnes mange forskjellige måter å gjennomføre evaluering av programvare på. De som oppgaven vil gå nærmere inn på her er observasjon, intervju og spørreskjema.

2.3.1 Observasjon:

En stor fordel med å bruke observasjon er at den er så direkte. Man spør ikke personene om hva de mener, føler eller hvilke holdninger de har; man ser hva de gjør, og hører hva de sier. Data fra direkte observasjon kan ofte være et nyttig supplement til informasjon innhentet med stort sett alle andre metoder (Robson 2002). Det finnes flere forskjellige typer observasjonsmetoder og Gold (Gold 1958)går igjennom fire typer; «complete

participant», «participant-as-observer», «observer-as-participant» og «complete observer». Der «complete participant» skjuler seg helt blant de som man observerer, denne teknikken er et av ytterpunktene av disse fire observasjonsteknikkene. Med «participant-as-observer» vet både observatøren og den observerte om deres forhold; «observer-as-participant» vil si at den som observerer ikke deltar i arbeidet som utføres, men hvor deltakerne er klar over forskerens rolle. Sist er «complete observer» som er det andre ytterpunktet, her er forskeren helt utenfor den settingen som skal observeres. Forskeren har ingen interaksjon med deltakerne. I denne oppgaven har det blitt benyttet «observer-as-participant» da det er denne formen som passet best med den situasjonen som de observasjoner som ble utført i forbindelse med oppgaven ble gjennomført under.

2.3.2 Intervju

Å intervjue brukere om deres opplevelse av interaksjonen med systemet gir en direkte og strukturert måte å innhente informasjon på. Intervjuer har den fordelen at nivået på spørsmålene kan bli variert ut i fra den gitte konteksten og at forskeren kan undersøke ting nærmere når interessante problemstillinger dukker opp (Dix, Finlay et al. 2004). I denne oppgaven har det blitt benyttet semi-strukturerte intervjuer. I slike intervjuer har intervjueren en liste med emner og spørsmål som det ønskes besvarelse på, men det blir også lagt opp til at svarene fra intervjuobjektet kan utforskes videre (Robson 2002).

2.3.3 Spørreskjema

Spørreskjema tilbyr en enkel måte for å få samlet inn informasjon, og som tilbyr en enkel analyse av den innsamlede informasjonen. I denne oppgaven har det blitt benyttet et spørreskjema kalt «System Usability Scale» (SUS) som er en Likert-skala (Brooke 1996). Dette tilbyr et utseende som deltakerne kan kjenne igjen, siden de ligner på «test deg selv»-tester som man finner i blader og magasiner.Dette gjør at de vil føle seg komfortable med spørreskjemaet (Robson 2002).

2.4 Identifisering av mennesker

Med identifisering av mennesker brukes i denne konteksten Clarkes definisjon (Clarke 1994):

«(..)human identification is the association ofdata with a particular human being.»(Clarke 1994)

Clarke skriver at den opprinnelige grunnen for nødvendigheten av identifikasjon var sosial heller enn økonomisk. Den sosiale dimensjonen av menneskelige kultur blir reflektert av ideen av en person «identifiserer» seg med en gruppe. Familie, venner og bekjente kjenner igjen en person ved utseende, stemmekarakteristikk, kunnskap om personlig informasjon som navn, alder, familie og lignende. Slik identifisering fungere når personer har kjennskap til hverandre, og befinner seg i nærheten av hverandre. Når samfunnet utviklet seg vokste det fram en nødvendighet av at personer uten kjennskap til hverandre kunne identifisere hverandre. Han identifiserer 3 basismetoder for identifisering av mennesker: (1) Kunnskapsbasert identifikasjon, (2) kjennemerkebasert identifikasjon og (3) biometrisk identifikasjon.

(1) Kunnskapsbasert identifikasjon går ut på at en person demonstrer sin identifikasjon ved å vise til at de er i besittelse av informasjon som kun denne personer har kunnskap om. Dette kan være fornavn og etternavn, tidligere navn, fars navn, mors navn, fødsels-dato og -sted, adresse og yrke. Vanlige eksempler på kunnskapsbasert identifikasjon er passord og PIN-koder.
(2) Kjennemerkebasert identifikasjon er identifikasjon ved hjelp av noen som inneholder informasjon om denne personen. Dette kan være et pass, et sertifikat, fødselsattest, adgangskort, kredittkort eller et medlemskort.
(3) Biometrisk identifikasjon blir brukt til å referere til flere forskjellige identifikasjonsmetoder som er basert på fysiske attributter og vanskelig-å-forandre karakteristikker.

Klassifiseringsskjema for biometriske identifiseringsteknikker.
Utseende
Sosial oppførsel
Biodynamikk
Naturlig naturgeografi
Påtvungne fysiske karakteristikker

Tabell1 - Klassifiseringsskjema for biometriske identifiserings teknikker(Clarke 1994)

Tabell 1 viser fem forskjellige grupper av biometriske identifiseringsteknikkeridentifisert av Clarke.

Oppgaven benytterseg mest av den første basismetoden, kunnskapsbasert identifisering, men det er også tatt inn enkelte biometriske elementer som rase. Systemet tillater identifikasjon via kjennemerke, men er ikke tilrettelagt spesielt for dette (ingen egne kort, eller merker).

2.5Tidligere forskning

Sargent og Michael belyser problematikken rundt bruk av informasjonsteknologi og telekommunikasjon (IT&T) i forbindelse med distribusjon av nødhjelp til flyktninger og «Internal Displaced Persons» (IDP) (Sargent and Michael 2005). I denne artikkelen foreslås det å utvikle et rammeverk for å bruke IT i nødhjelp, «Digital Aid Framework». Det kan være lærerikt å se på hvordan de foreslår å løse identifikasjons og gjenkjenningsproblematikken, som vil dele flere karakteristikker med den problematikken denne oppgaven står ovenfor. Å kunne spore en persons deltakelse gjennom hele prosjektets levetid er viktig for i det hele tatt å kunne gjennomføre prosjektet. Shankar fokuserer på å følge deltakere i en studie mens de går igjennom forsøkene (Shankar, Martins et al. 2006). Det er umulig å følge en deltaker gjennom studiet om man ikke er sikker på at det er samme person. Modulen til OMEVAC,som blir presentert i denne oppgaven, vil bidra til å sikre verifikasjon av deltakerens identifikasjon, slik at det er mulig å sikre seg om at det er samme person, også for personer uten offisielle ID-papir.

Håndholdte blir mer og mer brukt i kliniske forsøk (Clinical Trials) og fører til økt datakvalitet og kortere tid for å lukke databasen. Koop og Mösges(Koop and Mösges 2002)hevder at bruk av håndholdte datamaskiner fører til en rekke fordeler fremfor tradisjonell bruk av papirskjemaer. Disse fordelene er bedre tidsverifikasjon av innsamlet data, økt effektivitet gjennom enklere datalagring i kombinasjon med rask og enkel dataprosessering, datakvaliteten er høyere på grunn av at dataen er mer fullstendig, har færre feil, er mer konsistent og det er færre brudd på protokollen. Det er også enkelte ulemper ved bruk av håndholdte datamaskiner ved gjennomførelsen av kliniske forsøk. Sponsorer, data managere og feltarbeiderne må forholde seg til teknologiaspekter, og den initiale investeringen kan være høy. Andre ulemper er at dataen må synkroniseres ofte for

sikkerhetskopiering, og enkelte brukere kan synes at skjermen er vanskelig å tyde. Sist må det gjøres mer arbeid for å sikre elektroniske signaturer og sertifiserte kopier av elektroniske dokument (Koop and Mösges 2002). De fordelen, identifisert av Koop og Mösges (Koop and Mösges 2002), som bruk av slike systemer impliserer, vil ha en positiv effekt for gjennomførelsen av kliniske forsøk i lavinntektsland. Sikker identifisering av deltakerne i slike forsøk er viktig, og når man kan kombinere sikker identifikasjon med bruk av håndholdte enheter kan dette resultere i en gevinst for forskningen.

2.6 Oppsummering

I dette kapittelet har det vært gitt en gjennomgangav de teorier som har blitt brukt i utførelsen av denne masteroppgave. Kapittelet har gått kort inn på AR som har vært den overordnede metodikken, så har det blitt forklart kor hva MDD, Agil utvikling, og AMDD er. MDD er en metodikk der modellene er hovedproduktet og ikke programvaren, MDD krever nøyaktige og gode modeller og har som mål at all kode skal genereres automatisk fra modellene (Selic 2003). Agil utvikling er en samle betegnelse på utviklingsmetodikker som har som et felles sett av kjerneverdier. Hurtige og gjentatte sykluser, hvor fungerende kode blir levert hele tiden, minst mulig dokumentasjon og at utviklerne skal samarbeide med domeneekspertene (Beck, Beedle et al. 2001). AMDD er en agil utgave av MDD, men her skal modellene være så vidt gode nok. Det blir lagt vekt på at modellene skal være raske og enkle, helst ikke utviklet i dyre modelleringsprogrammer, men heller i de enkleste modelleringsprogrammene. Papir og tavle er også metoder som passer inn i AMDD (Ambler 2004).

Så ble det forklart litt om de teoriene som har blitt lagt til grunne for evalueringen som ble gjennomført. Det forklares først kort hovedmålene med en slik evaluering, så ble det gitt en gjennomgang av de tre evalueringsteknikkene som har blitt benyttet i oppgaven. De tre metodene var, observasjon, her var «observer-as-participant» den somble benyttet i oppgaven(Gold 1958). Intervju, og da særlig semi-strukturert intervju (Robson 2002),

og spørreskjema. Det ble benyttet et spørreskjema som har vært i bruk i lengre tid kalt «System Usability Scale» (SUS) (Brooke 1996).

Så ble det gitt en gjennomgang av grunnleggende teori for identifisering av mennesker, hvor detble sett på tre basismetoder for identifiseringen, som beskrevet av Clarke (Clarke 1994). I den siste delen av kapittelet ble det gitt en kort oppsummering av tidligere forskning, og hvor oppgaven ble posisjonert i forhold til den tidligere forskningen.

3 Design og utvikling

SPIM er en modul til systemet OMEVAC, og er lagd for å kunne tilpasses flere forskjellige systemer. SPIM består av 2 biblioteker og en database som kan benyttes av alle prosjekter skrevet for .NET CF. Grunnen til dette er at utviklere da kan gjøre forandringer en plass uten å måtte gjøre tilsvarende forandringer andre plasser. Utviklerne kan for eksempel bytte ut databasehåndteringssystemet med et fra en annen leverandør, og da tilpasser man koden til denne nye databasen kun én plass.
I dette kapittelet vil det bli forklart hvordan utviklingen av SPIM har foregått, og hvordan SPIM er bygget opp.

3.1 Metode

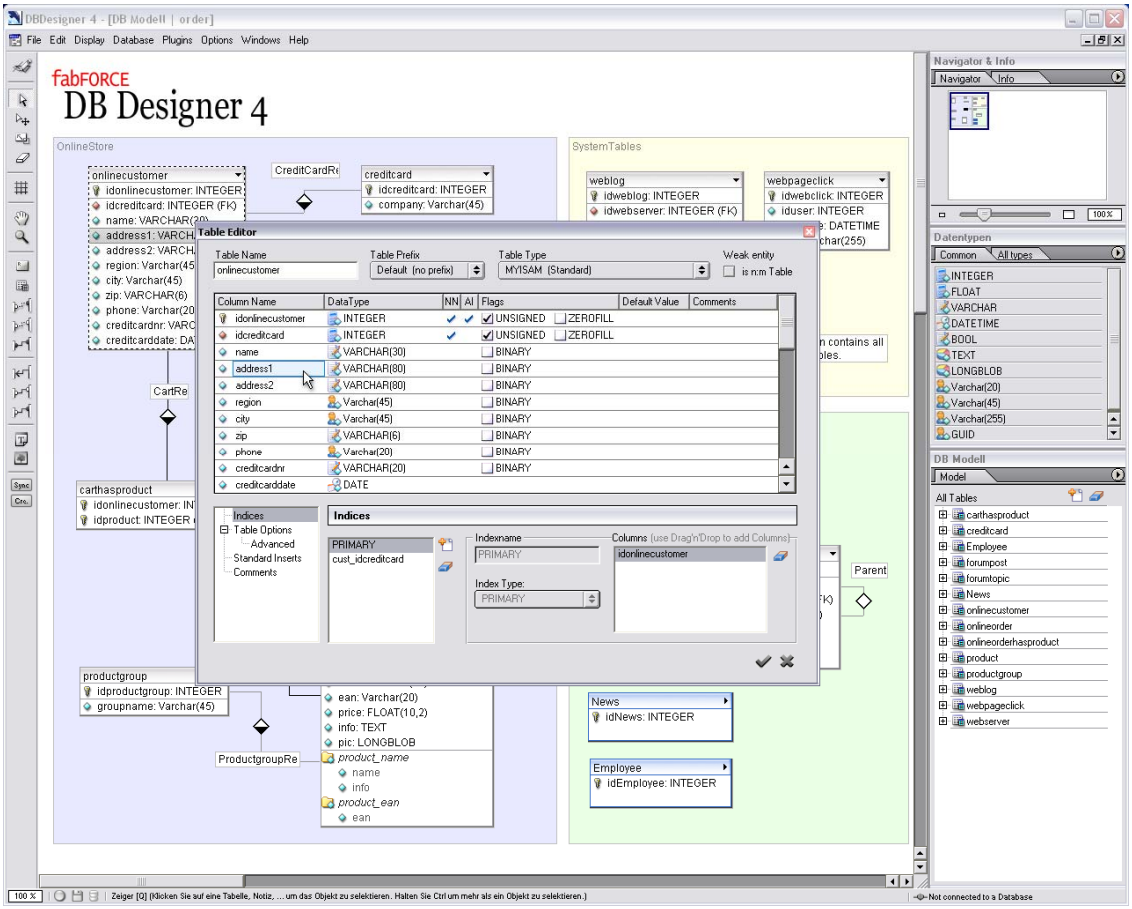
I utviklingen av SPIM har det blitt benyttet en agil utviklingsmetode kalt «Agile Model-Driven Development» (AMDD) (Ambler 2004). AMDD er som navnet sier en agil versjon av «Model-Driven Development» (MDD) (Selic 2003). Hovedforskjellen på MDD og AMDD er at i sistnevnte lager man raske modeller i flere omganger, heller enn store omfattende modeller som i MDD. AMDD går ut på at du modellerer, så implementere (kode) i flere omganger. Utviklerne begynner først med å modellere det overordnede systemet og arkitekturen. Så modellerer man en del av systemet detaljert før man implementerer denne delen, før man går videre å modellerer neste del og implementerer denne i flere iterasjoner eller sykluser. Mye av filosofien bak AMDD er å lage akkurat gode nok modeller, det skal ikke brukes for mye tid på modellene. En enkel håndtegnet modell kan være nok til å gå videre, det trengs ikke en modell tegnet i et avansert program med fine farger.

3.2 Verktøy

Det er blitt benyttet en rekke forskjellige verktøy i arbeidet med SPIM, både proprietære system og åpne. Hovedverktøyet har vært Visual Studio 2008 Beta 2 fra Microsoft. Dette er et kraftig utviklingsverktøy for utvikling av .NET applikasjoner. Den har også en god debug-funksjon som gjør det enkelt å foreta raske tester av applikasjonen som utvikles.

Visual Studio 2008 har også innebygd støtte for å virtualisere ulike håndholdte datamaskiner for å kjøre applikasjoner lagd for Windows Mobile og Windows PocketPC.

For modellering av datamodellen og forretningslogikken har jeg brukt Microsoft Visio 2007. Et verktøy som lar deg lage raske og enkle UML-modeller. For database modelleringen har jeg benyttet meg av en programvare kalt DBDesigner 4 fra fabForce (Figur 3). Dette er et «open source» verktøy som gir deg mulighet for å utføre modell-til-database synkronisering og automatisk plassering av fremmednøkler.



Figur 3–Skjermbilde av tabell-editoren i DB Designer 4.

3.3 Krav

Da SPIM er en modul i OMEVAC-rammeverket, kom brukerne i OMEVAC opp med et sett med krav til applikasjonen. Disse brukerne er feltarbeidere, data managere, prosjekt koordinatorer og lignende. Både funksjonelle og ikke-funksjonelle krav ble gitt. Kravene

er en samling av betingelser som systemet må oppfylle for å kunne være av nytte for CIH, det vil si at for at systemet skal kunne bli tatt i bruk må disse betingelsene oppfylles.

3.3.1 Ikke-funksjonelle krav

Ikke-funksjonelle krav er krav som ikke har noe med hvilke funksjoner og metoder programvaren skal inneholde, men det er krav som går på stabilitet, kjøreplattform og kompatibilitet.Tabell 2 viser de ikke-funksjonelle kravene. Siden SPIM utvikles som en modul til EpiHandy er det sentralt at systemet er kompatibelt med EpiHandy. I arbeidet SPIM skal støtte benyttes det ulike versjoner av Microsoft Windows for håndholdte datamaskiner, derfor må SPIM være kompatibel med Windows PocketPC 2003 og Windows Mobile 5 og 6. For å muliggjøre gjenbruk av koden var det et krav at systemet skulle utvikles innenfor .NET rammeverket og at det skulle støtte en database som kunne distribueres fritt og som var lagd for håndholdte datamaskiner.

Ikke-funksjonelle krav.
Kompatibel med EpiHandy
Kompatibel med Windows PocketPC 2003
Kompatibel med Windows Mobile 5 og 6
Utviklet for .NET rammeverk
Utviklet i enten Visual.NET, C#.NET eller C++.NET
Microsoft SQL CE Server

Tabell 2 – Ikke-funksjonelle krav

3.3.2 Funksjonelle krav

De funksjonelle kravene til et system beskriver hva systemet skal gjøre. Tabell 3 viser de funksjonelle kravene. Det var ikke så mange funksjonelle krav til applikasjonen, men de var viktige og ganske omfattende. Det må være mulig å registrere deltakere i prosjektet med systemet, man skal ha muligheten til å koble sammen de ulike skjemaene og deltakerne. Å kunne søke etter og identifisere deltakere er svært viktig. Å ta vare på alle endringer for senere revisjon er helt sentralt i systemet. Dette er på grunn av at for å kunne forsikre seg om at dataen er riktig, må det være mulighet for å kunne kontrollere alle endringer som er gjort, slik at avvik og forsøk på manipulasjon kan avdekkes.

Funksjonelle krav.
Registrering av deltakere
Koble sammen skjema og deltaker
Søk etter deltakere
Alle endringer skal lagres
Mulighet for identifisering av deltakere
Hver pasientidentifikator må bli koblet til en annen deltaker ID i en studie, uten å være sporbar.
Systemetmåtildele GUID til hver pasient som blir lastet inn i en studie.

Tabell 3 – Funksjonelle krav

3.4 Utviklingen

I utviklingen av SPIM har det blitt benyttet en tredelt arkitektur. Det vil si at man separerer datalaget (modell og database) fra brukergrensesnittet, og lar all samhandling mellom datalaget og brukergrensesnittlaget gå igjennom et kontroll-lag. Dette gjør datalaget og brukergrensesnittlaget uavhengige av hverandre. Under punkt 3.5 beskrives oppbygningen av SPIM mer detaljert.

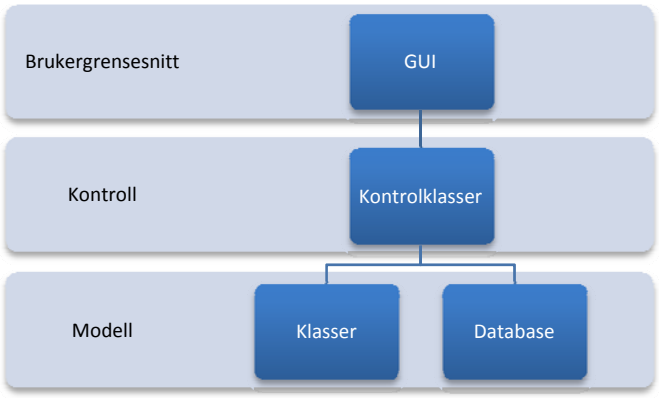
Utviklingen har som beskrevet i punkt 3.1 foregått i flere sykluser, med mål for hver gjennomgang. I begynnelsen av prosjektet ble det arbeidet i en gruppe med tre andre studenter. Det ble lagd noen enkle overordnede modeller før arbeidet med å modellere databasen ble påbegynt. Så ble database-modellen implementerte, før det ble lagd noen enkle modeller av datamodellen. Så ble denne implementert. Det ble gjort forøsk på å benytte Nhibernate for oversettelsen mellom dataobjekter og databasen, men det ble avdekket at Nhibernate ikke støttet .NET Compact Framework. Derfor måtte denne oversettelsen gjøres manuelt og det ble derfor lagd noen modeller for et system med «interfaces», eller kontrakter, og implementasjoner av disse kontraktene. Dette ble gjort fordi det ikke var ønskelit med et rammeverk som var bundet opp til en spesifik databaseteknologi.

Etter at de innledende syklusene var gjennomført, begynte det individuelle arbeidet, og prosjektet gikk inn i en periode med å undersøke ulike teknologier og teknikker for bruk i SPIM. Det fulgte nå en periode med grundigere modellering før implementeringen av disse modellene ble påbegynt. Det første som ble gjort var å utvide de modellene som ble utviklet i det sammarbeidet som ble utført tidlig i prosjektet, før utviklingen av

brukergrensesnittet ble startet. Utviklingen av brukergrensesnittet startet med å lage små skisser av hvordan det ønskede brukergrensesnittet skulle se ut. Disse modellene ble så brukt for å utforme grensesnittet. I løpet av denne perioden ble det gitt innspill fra CIH som endte med å gå tilbake til modelleringen. Her kom en av fordelen med AMDD inn, det at AMDD oppfordrer til å ta imot forandringer gjennom hele utviklingsløpet. Dette førte til at det kom inn flere endringer underveis, og det fungerte bra i forhold til den totale fremgangen til systemet.

Sett under ett gikk utviklingen greit, og selv om det dukket opp problemer som tok lengre tid enn antatt, så gikk det greit å tilpasse utviklingen etter dette. Mye av grunnen til dette var at det hele tiden var korte sykluser, hvor etter hver syklus hadde et system som kompilerte, og som var klart for den neste syklus. Metoden med å modellere i starten av hver syklus viste seg å være effektiv, da det skapte klare mål og det var hele tiden klart hva som skulle implementeres.

3.5 Arkitektur



Figur 4 – Modell-Kontroll-Brukergrensesnitt

SPIM er bygget som etter en tredelt arkitektur (**Error! Reference source not found.**). Modell-laget er todelt og består av datastrukturen, det vil si objektklassene, og databasen. Dette laget inneholder kun data, og har ingen metoder for å transformere dataen. Kontroll-laget består av logikken, og transformasjonen av dataene. Man kan si at dette laget omdanner data til informasjon. Dette laget får forespørsler fra brukergrensesnittet, og omdanner det til en forespørsel som går videre til modell-laget. Den sørger så for at dataen fra modell-laget blir omdannet til informasjon som er forståelig for brukeren. Kontroll-laget tar seg også av database til objekt transformasjonen. Når kontroll-laget får

inn en forespørsel fra brukergrensesnittet, som et objekt, så lager dette laget en databasespørring som tilsvarer forespørselen fra brukergrensesnittet. Resultatet av spørringen vil så bli omdannet til objekter og returnert tilbake til brukergrensesnittet.Brukergrensesnittet er topp laget, og tar seg av all kontakt med brukerne, og står for presentasjonen av informasjonen fra kontroll-laget. Brukeren forholder seg kun til brukergrensesnittet, og alle operasjoner gjort her blir sendt gjennom kontroll-laget.

Grunnen til denne tredelingen er å skille ut de ulike funksjonen til applikasjonen fra hverandre, slik at de endringene somblir gjort på en del av applikasjonen ikke fører til at det blir nødvendig å gjøre endringer i de andre lagene, eller å minimere behovet for slike endringer. Utviklere kan lage en nytt brukergrensesnitt uten å gjøre noen endringer på de to andre lagene. Man kan bytte ut databasen, og bare gjøre endringer i implementeringen av DAO-klassene, som vil bli beskrevet i avsnitt 3.4.3.

3.5.1 Database

Grunnlaget for databasen ble lagd i sammen med Knut Ivar Myklebust, Bjørge Næss og Dag Skjellevik. Det ble tatt utgangspunkt i databasestrukturen til OpenMRS (www.openmrs.com), og så ble denne tilpasset til de behov som CIH har. Grunnen til at det ble valgt å ta utgangspunkt i en eksisterende databasestruktur, var at det ville sikre en hensiktsmessig databasestruktur som fungerer. En struktur som var testet og i bruk. OpenMRS er et mye mer omfattende system enn SPIM og med et annet formål, derfor måtte detbli gjort en rekke endringer som å ta bort en rekke tabeller, legge til enkelte tabeller og endre andre tabeller. Det ble blant annet lagt til «Global Unique Identifier» (GID) verdier til alle ID-feltene i databasen. Noe som var en stor og kritisk mangel i OpenMRS's databasestruktur var mangelen på historietabeller. Tabeller som tar vare på alle forandringer i databasen, noe som er kritisk for revisjon og verifikasjon av dataen lagret i databasen. I all klinisk forskning er det kritisk at alle forandringer på verdier i databasen blir lagret, slik at endringer kan spores og ugyldige handlinger oppdages. Derfor var det en av de viktigste endringene som ble gjort med databasestrukturen til OpenMRS.

Databasestrukturen består av to hoveddeler, de primære tabellene og historie tabellene (Figur 5). Videre er hver av disse hoveddelene organisert i fem grupperinger. Disse er: «Person», «Patient», «User», «Patient_connector» og «Schema_connector». Disse grupperingene finnes som tilsvarende «History-grupper» i historie delen. Før en endring på en rad i en primærtabell blir gjennomført, lagres denne radens tilstand i den tilsvarende historietabellen. Om man for eksempel endrer pasients bostedsadresse blir denne raden først kopiert over i historietabellen med et tidsstempel, før endringen blir gjennomført i primærtabellen. Dette fører til at man kan gå tilbake å se når endringen ble gjennomført, og hvilke verdier som var lagret i raden før endringen. Noe som fører til at ulovlige eller feilaktige endringer kan spores og annulleres. Figur 5viser hvordan databasen er oppbygd.

Person

«Person-gruppen» består av fem tabeller: «person», «person_name», «person_address», «relationship» og «relationship_type». Person-tabellen er hovedtabellen for SPIM, hvor selve «personen» blir opprette. Her blir også person-ID-en opprettet. Denne ID-en følger «person» gjennom hele systemet, og blir brukt som nøkkel i flere andre tabeller.

En «person» har én eller flere «person_address» og hver «person_address» tilhører én «person». Hver «person» har ett eller flere «person_name», og hvert «person_name» tilhører én person. «Relationship» ”eies” av en person, men én «perosn» kan ha mange «relationships». Hver «relationship» har en en-til-mange relasjon til «relationship_type». «Relationship» har relasjon til både «person_a» og «person_b». En «person» trenger ikke å ha noen relasjoner til «relationship».

Patient

«Patient-gruppen» består av fem tabeller; «patient», «patient_identifier», «patient_identifier_type», «location» og «patient_groupe». Patient-tabellene er hovedtabellen i patient-gruppen. Denne tabellen får sin nøkkel fra person-tabellen. Dette er også tabellen som «samler» alle de andre patient-tabeller sammen. Location-tabellen inneholder data om den geografiske plasseringen til pasienten og er koblet sammen med patient_identifier-tabellen. «Patient_identifier» er koblet sammen med

«patient_identifier_type» ved en 1-til-mange-relasjon. Patient_groupe-tabellen holder informasjon om til hvilken pasient-gruppe en pasient tilhører. Den har en en-til-mange-relasjon til patient-tabellen. Patient_groupe-tabellen er koblet sammen med user_groupe-tabellen via en en-til-en-relasjon. Det vil si at én pasientgruppe har en relasjon til én brukergruppe.

User

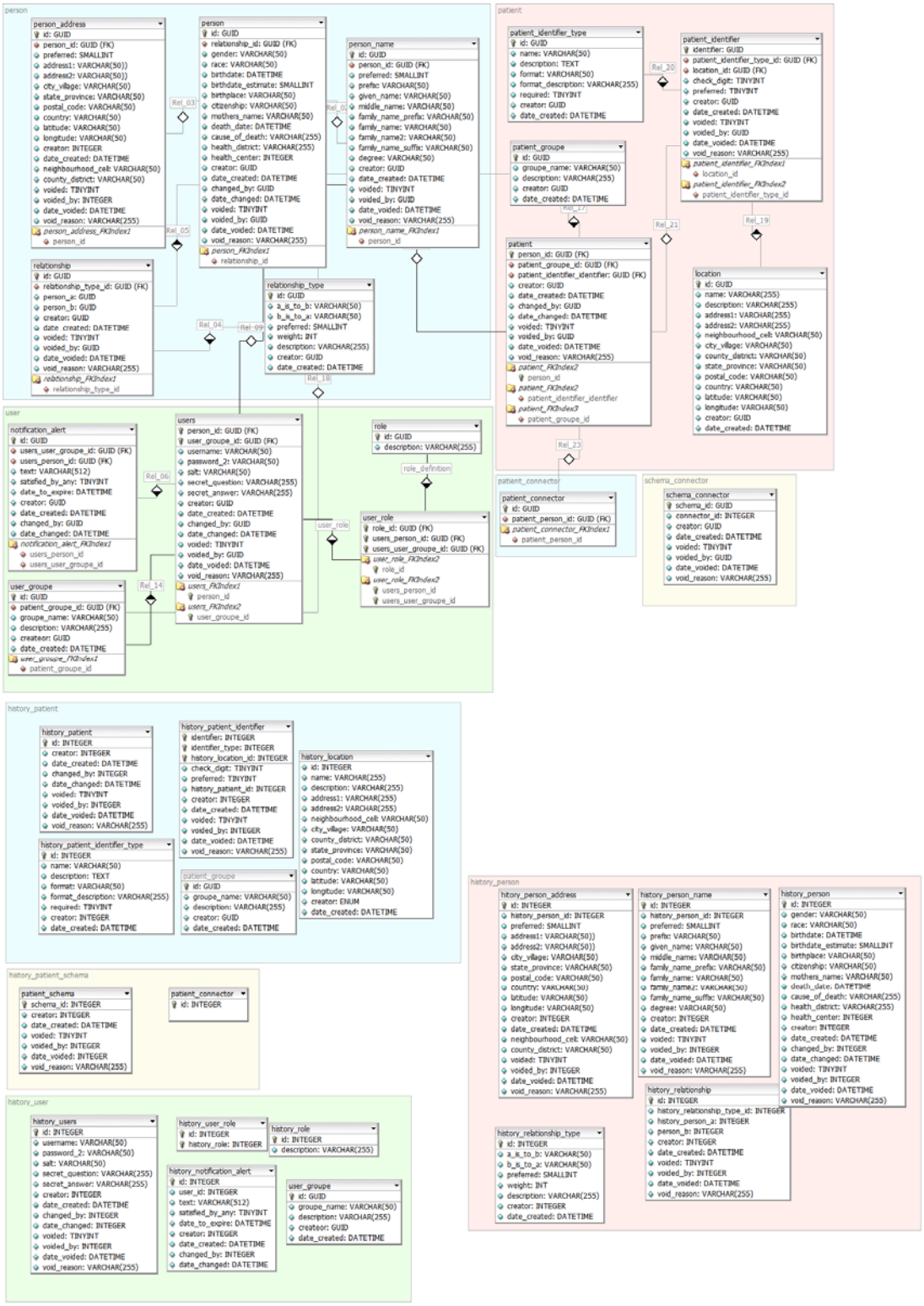
User-gruppen består av fem tabeller; «users», «role», «user_role», «notification_alert» og «user_groupe». Users-tabellen er hovedtabellen i user-gruppen og den som binder sammen user-gruppen. Denne tabellen inneholder blant annet brukernavn og passord. Role-tabellen inneholder ulike roller, og user_role-tabellen er en kobling mellom role-tabellen og user-tabellen. Notification_alert inneholder ulike meldinger til brukerne. User_groupe-tabellen inneholder informasjon om brukergruppen til en bruker.

Patient_connector

Patient-connector-gruppen inneholder kun én tabell. Denne tabellen, «patient_connector», som har som formål å skjule pasient_id-en mot systemer utenfor biblioteket. Denne tabellen har to felt, en ID og person-IDen.

Schema_connector

Schema_connector-gruppen inneholder kun én tabell. Denne tabellen, «schema_connector», har som formål å sikre at riktig skjema kobles til riktig pasient uten å avsløre pasient-id-en. Her lagres skjemanummeret til det skjemaet som tilhører en pasient, og en «connector»-ID fra «patient_connector».



Figur 5- Databasestrukturen til SPIM

3.5.2 Datamodell

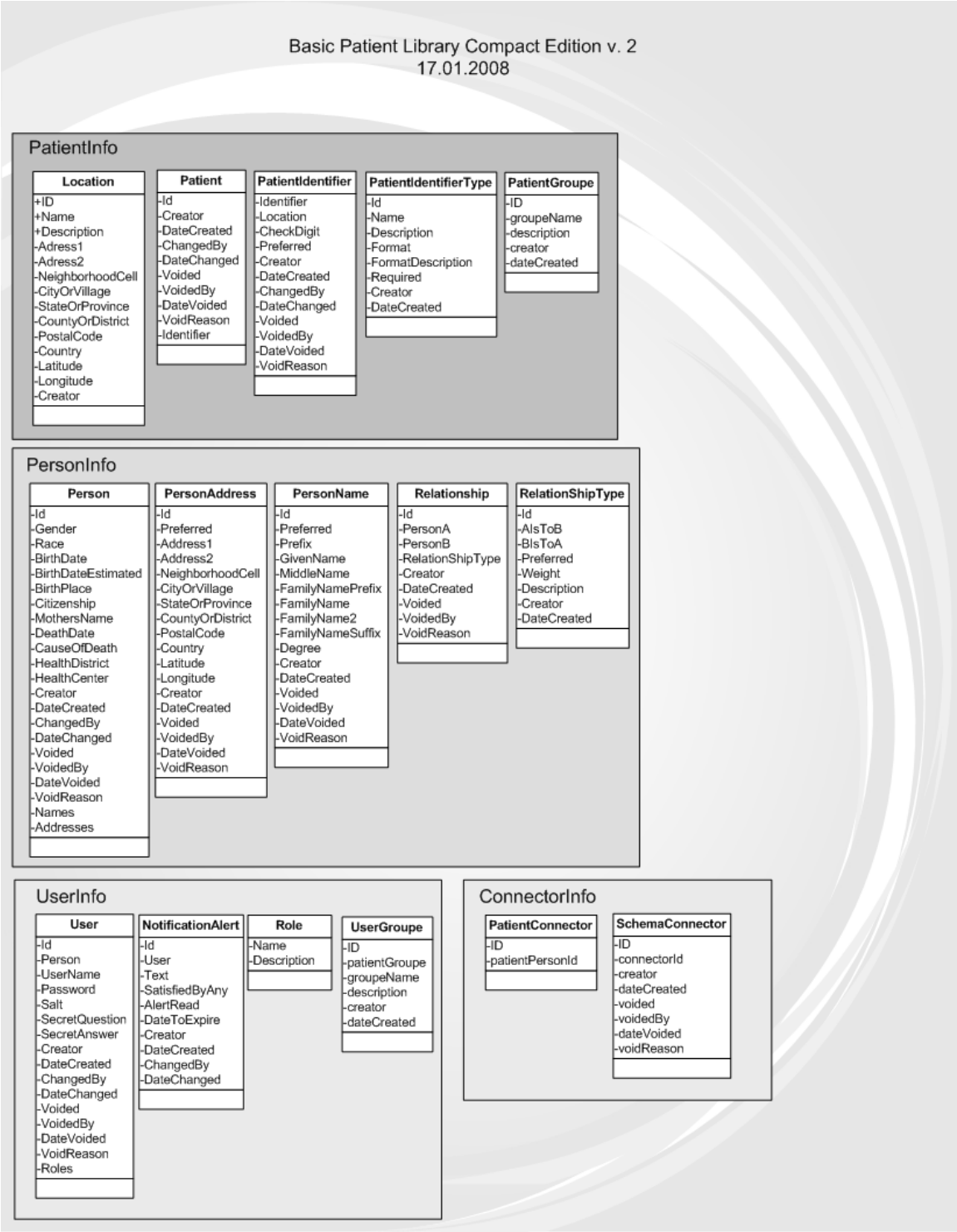
Datamodellen (Figur 6) ble i likhet med databasen utviklet i samarbeid med Knut Ivar Myklebust, Bjørge Næss og Dag Skjellevik. Denne består av de klassene som danner fundamentet for SPIM. Biblioteket som datamodellen danner for ble kalt «Basic Patient Library» (BPL) (Figur 6). Klassene i BPL bygger på primærtabellene fra databasen, og består av 16 klasser fordelt på fire «namespaces» eller grupper. Disse klassene og gruppene ble valgt for å gjøre overgangen mellom databasestrukturen og objektene så liten som mulig. Det er mye enklere å forholde seg til databasen og datamodell om disse samsvarer. Det vil si at de fire gruppen i datamodellen svarer til de fire gruppene i databasestrukturen. «PersonInfo» tilsvarer **«**Person**»**-gruppen i databasen, «PatientInfo» med «Patient», «UserInfo» med «User» og «ConnectorInfo» som er en sammenslåing av «Patient_Connector» og «Schema_Connector»-gruppene i databasestrukturen. Både «Patient» og «User» er subtyper av «Person», det vil si at de arver fra «Person».

PersonInfo

PersonInfo består av fem klasser; «Person», «PersonAddress», «PersonName», «Relationship» og «RelationshipType». Hvor «Person» er hovedklassen og de andre klassene er feltvariabler i «Person». Den har en samling med «PersonAddres» og en samling med «PersonName», noe som lar deg koble flere navn og flere adresser til objektet. Mens «Relationship» er løsere knyttet til «Person». «Relationship» har «RelationshipType» som en feltvariabel.

PatientInfo

PatientInfo består av fem klasser; «Patient», «PatientIdentifier», «PatientIdentifierType», «PatientGroupe», «Location». «Patient» arver fra «Person» men har en «PatientIdentifier» i tilegg. «Location» lagrer addresse og geografiks informasjon, og «PatientIdentifier» kobler sammen pasienten og lokasjonen. «PatientIdentifierType» er en klasse som gir en beskrivelse av pasientidentifikatortypen. «PatientGroupe» er en klasse for pasient grupper. Den inneholder, en Id, gruppenavn og en beskrivelse av pasientgruppen.



Figur 6- Datamodell for Basic Patient Library

3.5.3 Controller

«Controller» er laget mellom brukergrensesnittet og datalaget. Dette laget består av klasser og metoder for å håndtere overgangen mellom objekter og relasjonelldata fra databasen. Kontroll-laget består av forskjellige «namespaces» på forskjellige nivå. Dette laget er bygget som et eget bibliotek som benytter seg av BPL biblioteket. Jeg har kalt kontrollbiblioteket for SPIM-CF der CF står for «Compact Framework».

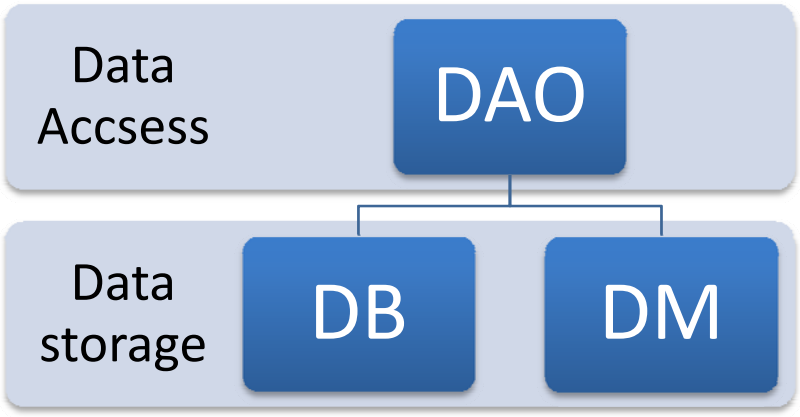
SPIM-CF består av en «DAO» (Data access object) del som er en rekke med kontrakter med metoder for omgjørelsen fra databasen til objekter. Det er også skrevet klasser som oppfyller disse kontraktene og som er skrevet for Microsoft SQL Server Compact Edition (SQL CE). Dette fordi jeg har benyttet meg av Microsoft SQL Server Compact Edition

```
public void DeleteUser(User u)
{
//Since the user is deleted, the object is voided.
        u.Voided = true;

 //Instansiate a new DAO.
NotificationDAO tempDAO = newSqlCeNotificationDAO();

 //Creates an NotificationAlert-object, so we can find all the alerts connected to this user.
NotificationAlert SearchObject = newNotificationAlert();
        SearchObject.User = u; //Sets that you should find every Alert with User = this user.

 //Gets an collection with every NotificationAlert that matches the criteria.
ICollection<NotificationAlert> Alerts = tempDAO.FindNotification(SearchObject);

        //Delete all these NotificationAlerts.
foreach (NotificationAlert n in Alerts)
        {
          tempDAO.DeleteNotification(n);
        }

//Creates the link to the DB.
SqlCeCommand command = SqlCeSingleton.GetInstance().CreateCommand();

 //Updates the users table
        command.CommandText = "UPDATE users SET voided = '1" +
"', voided_by = '" + u.VoidedBy.Id +
"', date_voided = '" + u.DateVoided +
"', void_reason = '" + u.VoidReason + "'" +
            " WHERE id = '" + u.Id + "';";
        command.ExecuteNonQuery();

        //Insert the object in the history_users table
        command.CommandText = "INSERT INTO history_users (user_id, user_groupe_name, username,
password, salt, secret_question, secret_answer, creator, date_created, changed_by, date_changed, voided,
voided_by, date_voided, void_reason) SELECT * FROM users WHERE id = '" + u.Id + "';";
        command.ExecuteNonQuery();
        //Delets the user from the users-table.
        command.CommandText = "DELETE FROM users WHERE id = '" + u.Id + "';";
command.ExecuteNonQuery();
    }//end DeleteUser
```


Figur 7 – DeleteUser-metoden fra SqlCeUserDAO

som grunnlag for SPIM. Hver klasse i BPL har en kontrakt som hører til den klassen, og en klasse som implementerer denne kontrakten for SQL CE. Hver kontrakt har fire metoder som må oppfylles, en for søk med objekt, en for søk med ID, en for å oppdatere eller lagre ny, og en metode for å slette objektet. Figur 7 viser et eksempel på hvordan metoden for å slette en bruker ble implementert.

Måten database til datamodell transformasjonen er blitt løst på er at i bunn ligger databasen og datamodellen. Hver tabell i databasen har en klasse i datamodellen som tilsvarer tabellen. Så er det noen DAO-klasser som foretar selve transformasjonen. Om man vil lagre et objekt man har opprettet i databasen, sender man det objektet inn til tilsvarende DAO-klasse som så kobler seg opp mot databasen og oversetter objektet til databaseverdier som blir lagret i databasen. Om man vil hente ut et objekt fra databasen, sender man inn et objekt med de ønskede verdiene til DAO-objektet. Så kjører DAO-objektet en spørring mot databasen med de ønskede verdiene. Resultatet fra spørringen blir omdannet til en samling med objekter som blir returnert. Figur 8 viser en grafisk fremstilling av denne strukturen.



Figur 8 - Datatilgang og datalagring

Videre har SPIM-CF også et «namespace» kalt «Controller». Dette «namespacet» inneholder kun en klasse som inneholder metoder som lar brukergrensesnittet kommunisere med DAO-klassene, uten å opprette DAO-objekter. Figur 9 viser eksempel fra «Controller-klassen», hvor man får inn et pasient-objekt og lagrer dette i databasen. Fra brukergrensesnittet forholder man deg kun til «Controller-klassen» og «AddPatient-

metoden» når man vil lagre en pasient. Dette vil gjøre brukergrensesnittet uavhengig av DAO-klassene, og vil gjøre det enklere å gjøre forandringer andre steder i systemet. Klassen har metoder for å logge inn, søke etter pasienter, legge til pasienter, slette pasienter og andre operasjoner som er nødvendige. Så fra brukergrensesnittet forholder man seg kun til en klasse, hvor man oppretter et objekt og utfører de ønskede operasjonen gjennom dette objektet.

```
public bool AddPatient(Patient p) {
try
            {
          SqlCePatientDAO newPatient = newSqlCePatientDAO();
              newPatient.SaveOrUpdatePatient(p);
            }
catch (Exception ex)
            {
          return false;
          }
return true;
}
```

Figur 9– Kode eksempel fra Controller-klassen

3.5.4 Plattform og kjøremiljø

Siden EpiHandy kjører på ulike Windows operativsystem, ble det foretatt et valg om å utvikle SPIM for Windows Mobile 5 og 6 samt Windows PocketPC 2003. Testene i Uganda ble gjennomført med SPIM installert på PDAer som kjørte PocketPC 2003. I begynnelsen av utviklingen ble det bestemt at det skulle benyttes .NET CF 3.5, som kjøremiljø for BPL. Derfor ble det ansett som hensiktsmessig å benytte .NET CF 3.5 som kjøremiljø for hele SPIM.

3.6 Brukergrensesnitt

Brukergrensesnittet til SPIM er utviklet for å gjøre det så enkelt som mulig for feltarbeiderne å bruke systemet når de er ute i feltet og samler inn data. Det har blitt prøvd å følge standard grensesnitt for de ulike kjøremiljøene, og selv om det er visse ulikheter i de ulike versjonene av operativsystemene, har det ikke oppstått noen problemer knyttet til de ulike versjonene.

Figur 10 – Hovedmenyen

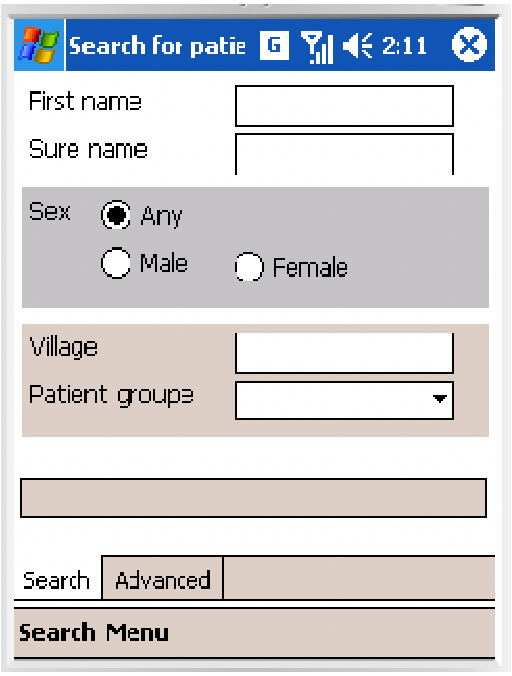
Det første brukeren møter etter å ha logget seg inn, er hovedmenyen (Figur 10). Denne består av to eller tre knapper, avhengig av om man er administrator eller ikke. Administratoren har tilgang til å legge til en brukeren, noe en vanlig bruker ikke har. I tillegg er det en meny som lar brukeren uføre de operasjoner som har egne knapper, logge ut og avslutte applikasjonen. I hovedmenyen er det også et statusfelt som viser ulike meldinger etter hvilke operasjoner brukeren har utført, og resultatene av disse. **Error! Reference source not found.**11 viser meldingen brukeren får i statusfeltet når man har vellykket lagt til et nytt skjema.

Figur 11 – Melding etter du har lagt til et nytt skjema

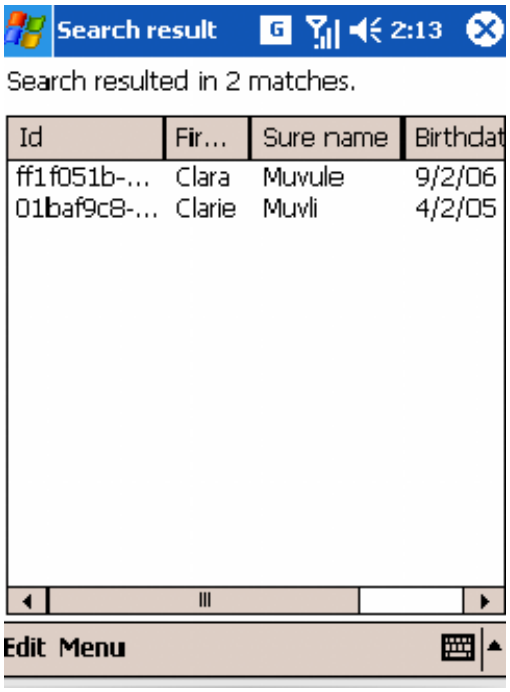
Om man trykket på søkknappen i hovedmenyen kommer man til søkevinduet (**Error! Reference source not found.**12). Her har brukeren to tabulatorer, en for ordinært søk og en for avansert søk. Det er en knapp for å utføre søket og en meny hvor man kan utføre søket, gå tilbake til hovedmenyen, få hjelp, logge av applikasjonen eller avslutte applikasjonen. Vinduet er bygget opp med de mest brukte søkekriteriene øverst. De ulike typene av søkekriterier er delt inn i ulike felt med ulik farge for å tydeliggjøre skillet mellom de ulike typene. Øverst finner man fornavn og etternavn, så kommer kjønn hvor

brukeren kan velge å søke etter begge kjønn eller, menn eller kvinner. Nederst finner man landsby og pasientgruppe. Alle disse kriteriene kan kombineres for å få mer nøyaktige treff. Når brukeren søker etter fornavn og etternavn, trenger ikke brukeren å søke etter hele navnene, det er nok at navnet man søker etter inneholder de bokstavene du bruker i søket. For eksempel kan en bruker søke etter en pasient med fornavn «Clara» med søketermen «Cla». Når brukeren så trykker på søk, aktiveres «progress baren», slik at brukeren får en tilbakemelding og ikke tror at systemet har låst seg eller lignende når søket utføres.

Når søket er gjennomført blir brukeren overført til et nytt vindu med søkeresultatet (Figur 13). Her får brukeren melding om hvor mange treff søket gav, og en liste med resultatene fremstilt med ID-nummer (de åtte første tegnene), fornavn, etternavn og fødselsdato. Nederst i vinduet er to menyer, en «Edit»-meny og en «Menu». I «Edit»-menyen har brukeren tre operasjoner han kan utføre; «New Form», «All Forms» og «Edit Patient». Alle tre operasjonene krever at brukeren har markert det ønskede søkeresultatet. «New Form» sender brukeren videre til et nytt vindu hvor han kan skrive inn skjemanummeret på skjemaet som brukeren vil registrere på pasientent. «All Forms» sender brukeren videre til et vindu som viser alle skjema registrert på den valgte pasienten og «Edit Patient» sender brukeren til det samme vinduet som når man skal registrere en ny pasient, med den forskjellen at all informasjon som finnes om pasienten er utfyllt.



Figur 12– Søkevinduet

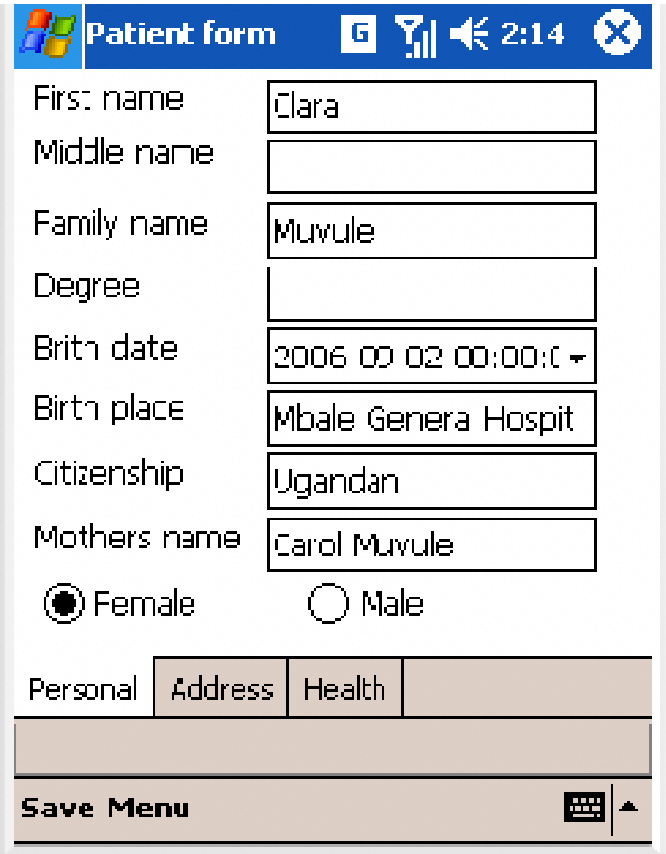


Figur 13 – Søkeresultatet

I «Menu» har brukeren fem valg. Man kan legge til en ny pasient, gå tilbake til søket eller hovedmenyen, få fram hjelpe for vinduet og avslutte applikasjonen. Hjelpen består av en informasjonsboks med en kort tekst om de mulighetene man har i søkeresultatvinduet.

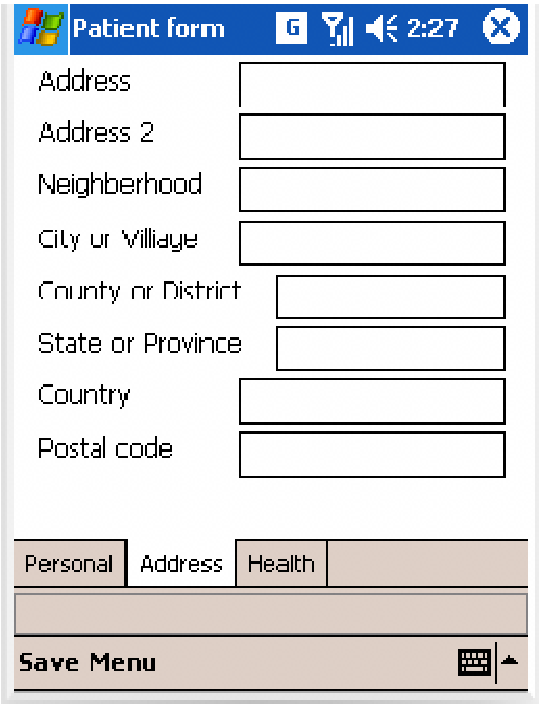
Vinduet for å legge til et nytt skjema er sparsommelig utformet med en tekstboks, hvor brukeren skriver inn skjemanummeret, og to knapper for å lagre og å avbryte registreringen. Nederst er det en meny hvor som lar brukeren; lagre skjemaet, gå tilbake til søk eller hovedmenyen, velge hjelpfunksjonen eller avslutte applikasjonen. Når brukeren har lagret skjemaet blir han overført til hovedmenyen. Det kommer da opp en melding i statusfeltet i hovedmenyen.

Når en bruker velger å se alle skjemaene til en pasient («All Forms»), så kommer han til et vindu som inneholder en liste med alle skjemaene som er registrert på den valgte pasienten. Øverst i vinduet har man en ledetekst hvor pasientens navn blir vist etter fulgt av selve listen. Her får man opp skjemanummeret (de åtte første tegnene) og datoen skjemaet ble registrert. Menyen gir det mulighet til å gå til søkevinduet, hovedmenyen, få opp hjelpen og avslutte applikasjonen.

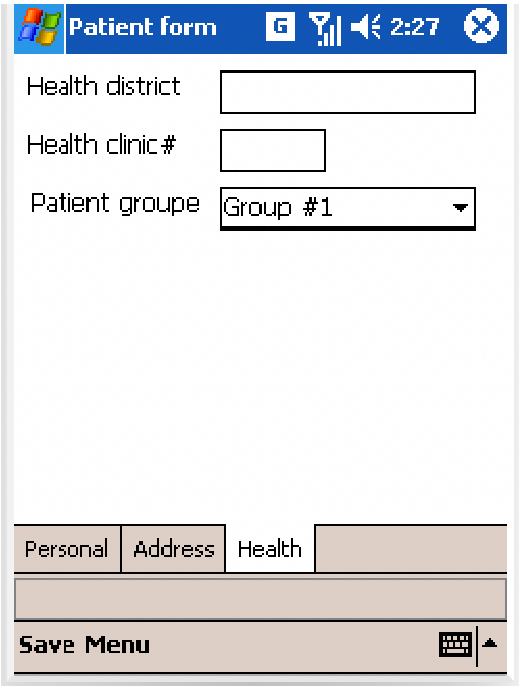


Figur 14 – Pasient vinduet personfanen.

«Add Patient» og «Edit Patient» vinduene er det samme vinduet (Figur 14). Eneste forskjellen er at i «Add Patient» er alle feltene tomme, mens i «Edit Patient» er all informasjon om pasienten som er registrert i systemet utfylt. Vinduet består av tre faner, hvor de forskjellige typene av informasjon er delt opp. All personlig informasjon som navn og fødselsdato er lagt til en fane som er kalt «Personal». All adresseinformasjon er lagt til en fane kalt «Address» og helseinformasjonen, det vil si helsesenter, helsedistrikt og pasientgruppe, er lagt til fanen «Health». Den første fanen, og den som vises når man åpner vinduet, er «Personal». Øverst finner brukeren tekstboks for fornavn, mellomnavn og etternavn. Så kommer ulike felter brukeren skal fylle ut nedover. Nederst finner man radiobokser for kjønnet. Fødselsdatoen er en kalender eller «TimePicker», som lar brukeren velge riktig dato fra en kalender. Du har også muligheten for å skrive inn datoen om det passer bedre. «Address» fanen (Figur 15) er bygget opp likt som «Personal» fanen med ledetekst på venstresiden og tekstboksene på høyre siden. Brukeren begynner øverst med adressene så nabolag, distrikt og land. I denne fanen er det bare tekstbokser. Den siste fanen er «Health» fanen (Figur 16). Her skal brukeren skrive inn navnet på helsedistriktet og nummeret på helseklinikken pasienten er tilknyttet. Det er også en nedtrekksmeny hvor man tilordner pasienten en pasientgruppe. Når brukeren har fylt ut den informasjonen han har tilgang til trykker du på lagreknappen («Save»). Dette kan brukeren gjøre uavhengig av hvilken fane han har oppe. Når brukeren har lagret pasienten blir han overført tilbake til hovedmenyen, og han får opp en statusmelding hvor det står om lagringen var suksessfull eller ikke. I menyen i dette vinduet kan brukeren velge å avbryte registreringen eller editeringen, brukeren kan få hjelp eller han kan avslutte applikasjonen.



Figur 15– Pasient vinduet adressefanen



Figur 16– Pasient vinduet helsefanen.

3.7 Oppsummering

I dette kapittelet har det blittgitt en gjennomgang av utviklingen og strukturen til SPIM. Det har blitt vist hvordan utviklingen har foregått, og hvilken metode som har blitt benyttet. Grunn strukturen i SPIM er bygget på databasestrukturen til Open MRS, noe som sparte tid i starten av utviklingen. Det gjorde at databasen ble bygget på en utprøvd struktur som er i bruk, noe som er en fordel mot å lage en helt ny struktur fra bunnen. Å ha utviklet grunnstrukturen fra bunnen av ville ha medført mer testing og lengre tid for å få en fungerende struktur.
Utviklingen har fulgt en metodikk kalt «Agile Modell-Driven Development» (AMDD), og går ut på at man lager mange raske modeller før man implementerer modellene og går videre med andre modeller. Dette er en iterativ metodikk, det vil si at utviklingen foregår i sykluser med definerte mål for hver syklus. I punkt 3.2 ble det beskrevet hvordan utviklet har foregått, og prøvd å begrunne hvorfor utviklingen ble gjennomført på denne måten.

Det er vist at SPIM er utviklet med en trelags-arkitetktur (Figur 4) som gjør applikasjonen mer fleksibel for videre utvikling. Den er bygget opp med datastrukturene i bunn, databasen og datamodellen, så kommer et lag som består av metoder for å oversette objektene fra datamodellen til datasett som databasen forstår, og omvendt. Dette laget har også en klasse som står for kommunikasjonen med brukergrensesnittlaget. Det tredje laget er brukergrensesnittet som tar seg av all interaksjon med brukerne.

I punkt 3.5 har det blitt foretatt en detaljert gjennomgangav det grafiske brukergrensesnittet, og det er vist flere figurer som illustrerer hvordan brukergrensesnittet ser ut. Det har blitt beskrevet engjennomgang av hvert enkelt vindu og forklart hvordan og hvorfor de er utformet slik som de er. Det har blitt lagt vekt på at brukergrensesnittet skal være enkelt, og at det ikke skal være noen forstyrrende elementer som kan distrahere brukerne.

4 Evaluering

For evalueringen av SPIM ble det valgt å legge denne til et forskningssenter i Uganda. Dette for å evaluere applikasjonen blant sluttbrukerne.Evalueringen ble foretatt på to brukergrupper. Feltarbeiderne, de som er ute i feltet og foretar datainnsamlingen, og data managers, de som bearbeider den innsamlede dataen.

I det følgende kapittelet vil det bli presentert hvordan evalueringen ble foretatt og resultatene fra denne. Først presenteres det hvordan SPIM ble testet med feltarbeiderne og de funn som ble gjort i arbeidet med disse. Deretter blir de funn som ble gjort under evalueringen av SPIM sammen med data manageren ved PROMISE prosjektet i Mbale presentert.

Figur 17 - PROMISE kontoret

4.1 Metode

Det har blitt benyttet flere forskjellige metoder for datainnsamlingen i denne evalueringen av SPIM. Robson (Robson 2002) og Dix et. al. (Dix, Finlay et al. 2004)

skriver i sine bøker om metoder for forskning og evaluering. De kommer frem til flere metoder som passer i denne situasjonen, og sammen med AR. Det har blitt valgt å benytte et spørreskjema, med rom for kommentarer, observasjon og intervjuer. I kapittel 2 ble disse ulike teknikkene grundigere beskrevet.

Feltarbeiderne fikk utlevert noen oppgaver de skulle utføre med SPIM, så fikk de utlevert et spørreskjema med ti utsagn de skulle si seg enig eller uenig i. På skjemaet ble de oppfordret til å skrive noen kommentarer. Under oppgavene ble de observert og etter på ble det foretatt intervjuer med feltarbeiderne. Data manageren ble observert over flere dager for å få et innblikk i hvordan hans oppgaver ble løst. Han fikk også prøve SPIM, og teste ut de forskjellige funksjonene til SPIM før feltarbeiderne gjorde sine oppgaver. Det ble også holdt noen uformelle samtaler om arbeidet før det mot slutten ble holdt et strukturert intervju med ham.

4.2 Feltarbeidere

Feltarbeiderne er de som er ute i feltet og samler inn data. De oppsøker deltakerne i prosjektet og innhenter den data som kreves av prosjektet. I PROMISE har feltarbeiderne brukt håndholdte datamaskiner for innsamling av data, noe som gjør at de er godt vant med bruk av håndholdt datamaskin. For å få en reell test på hvor brukervennlig SPIM er, ble det valgt å holde opplæringen i programmet veldig enkelt med kun en kort gjennomgang av applikasjonen.Deretter fikk brukerne utdelt fireoppgaver som skulle løses. Det var seks feltarbeidere som var tilgjengelige for testingen av SPIMog observatør var tilgjengelig for å yte hjelp under oppgavene for alle utenom den siste brukeren. Til gjengjeld fikk den siste brukeren litt mer opplæring enn resten av brukerne. Dette var fordi observatør ikke kunne være tilgjengelig for å yte bistand under hennes utførelse av oppgavene. Testene foregikk på kontorene til PROMISE i Mbale (Figur 17) og det var en eller to brukere som utførte oppgavene samtidig.

Under utførelsen av oppgavene ble brukerne observert om hvordan de arbeidet. Det ble registrertulike handlingsmønstre og prøvd å se hvor problemene oppsto.

Etter at brukerne hadde løst oppgavene de hadde fått utlevert, ble de gitt et kort spørreskjema, ”System Usability Scale”(SUS)(Brooke 1996),med 10 spørsmål. De 10 spørsmålene gitt på brukbarheten (usability) til applikasjonen og var gradert fra 1, sterkt uenig, til 5, sterkt enig(Brooke 1996). De ble også bedt om å skrive eventuelle kommentarer på baksiden av spørreskjemaet.

Etter testingen og spørreskjemaet ble det foretatt intervjuer. Disse ble gjort med en person om gangen og startet med en liten innføring i hva SPIM er, og hva hensikten med applikasjonen er. Grunnen til at innføringen kom her var for å se hvor enkelt systemet var å bruke. Og det ble oppfattet at det ville fremkomme et ærligere resultat om en ikke gjennomførte en grundig opplæring av systemet i forkant av evalueringen. Og når observatør var tilgjengelig for støtte under utførelsen av oppgavene, ville det være lettere å oppdage hvor problemene var om feltarbeiderne ikke hadde lært hvordan de skulle bruke systemet. Alle brukerne var erfarne når det kom til å bruke PDA-er, og noe av det som var ønsket å undersøke var hvor intuitivt systemet var. Det ble også informert om hensikten med testene og intervjuene.

Intervjuene ble foretatt på kontorene til PROMISE (figur 17) og ble tatt opp. Det ble ikke skrevet så mange notater under intervjuene, siden intervjuene ble tatt opp og lagret digitalt.Dette gjorde at intervjuer hadde mulighet til å konsentrere seg om intervjuobjektene. Intervjuene besto av fire forskjellige kategorier med spørsmål. Disse var: Brukbarhet, praktiskhet, fremtidig utvikling og en åpen del. Hver del av intervjuet hadde fra tre til seks spørsmål, utenom den åpne delen, men spørsmålene ble ikke fulgt notorisk. De ble heller brukt som et utgangspunkt for intervjuet, altså et semi-strukturert intervju (Robson 2002). Semi-strukturert intervju ble valg fordi da har man muligheten til å komme med oppfølgende spørsmål, og utforske svarene fra intervjuobjektet grundigere enn med et strukturert intervju. I enkelte sitater kommer det ikke klart frem hva det er snakk om, her er det lagt til anmerkninger i parentes og ”f. anm.” som står for forfatters anmerkning.

4.2.1 Mål

Målet med evalueringen er å la brukerne få prøve applikasjonen slik at de kan komme med innspill på hva som er styrkereller svakhetermed systemet. Siden det i denne evalueringen har blitt benyttet ekte sluttbrukere, fører dette til at de tilbakemeldingene som blir gitt vil være viktig for den videre utviklingen av SPIM. For at sluttbrukerne skal få størst mulig nytteverdi av den ferdige applikasjonen er det viktig at de får prøve det og komme med innspill. Det var også et mål å få rede på om brukerne så en nytteverdi i en slik applikasjon, og om de trodde at SPIM vil føre til mer valid data.

4.2.2 Oppgaver

Feltarbeiderne fikk utlevert fire oppgaver som skulle løses. Dette tok rundt 10 minutter og ble foretatt under observasjon. Observatør var behjelpelig med instruksjoner om det var noe brukerne ikke fant ut av. Under blir de fire oppgavene presentert med målet for oppgaven og en beskrivelse av denne.

4.2.2.1 Oppgave 1 – ”The first search”

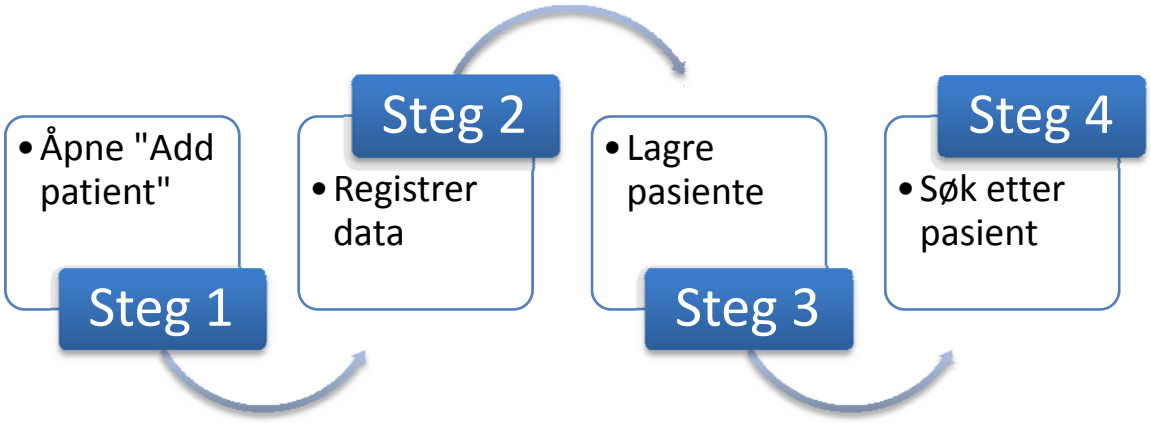
Mål med oppgaven: Å bli kjent med søkesystemet i SPIM

Brukerne skulle først logge seg inn på applikasjonen med utdelt brukernavn og passord. Så skulle de åpne søkevinduet og søke etter en deltaker kalt ”Clara Muvule” ved å søke med frasen ”Cla” i feltet for fornavn.

4.2.2.2 Oppgave 2 – ”Adding a new patient” (Figur 18)

Mål med oppgaven: Å legge til en ny pasient og så søke etter denne.

Brukerne skulle først åpne vinduet for å legge til en ny pasient. Brukerne fikk utdelt hvilken informasjon som skulle lagres om denne brukeren. Det var personopplysninger, som navn, kjønn og fødselsdato, adresseopplysninger og helseopplysninger, hvilken klinikk de tilhørte. Disse tre kategoriene med opplysninger var fordelt over tre faner i vinduet. Så skulle de lagre pasienten. Deretter fikk de beskjed om å søke etter denne personen via søkevinduet. Om de hadde gjort alt riktig ville de få denne personen opp iresultatet fra søket.



Figur 18 - Oppgave 2

4.2.2.3 Oppgave 3 – ”Adding a new form”

Mål med oppgaven: Å legge til et nytt skjema på en pasient.

Brukeren skulle først søke etter en pasient, ”Claire”. Så skulle de velge denne i resultatvinduet, og velge ”Add new form” fra edit-menyen. Så skulle de skrive inn et utdelt skjemanummer og lagre dette på den valgte pasienten.

4.2.2.4 Oppgave 4 – ”Viewing all the forms collected on one patient”

Mål med oppgaven: Å se alle skjemaene registret på en deltaker.

Brukeren skulle først søke etter en person, ”Clara”, så skulle denne personen markeres i søkeresultatet. Deretter skulle brukeren gå inn i edit-menyen og velge ”All forms”. Da skulle det presenteres en oversikt over alle skjemaer registret på denne deltakeren.

4.2.3 Resultat

Alle brukerne greide oppgavene uten store problem, og brukte mellom 5 og 10 minutter på å utføre dem. Noen ”problemer” gikk igjen, særlig at fem av brukerne prøvde hele tiden å gå tilbake ved å trykke på X-en oppe i høyre hjørne. Dette kom av at EpiHandy bruker dette for å gå tilbake til vinduet man kom fra.

“Is a normal thing with EpiHandy. (…) you just tap X, then you go back to the other page you wanted.” (Bruker 3)

En annen ting var at de fleste søkte med så mye informasjon som de hadde tilgjengelig. Selv om det bare var nødvendig å søke på fornavn, de fylte ut alle feltene i søkevinduet. Ett annet problem var å markere resultatene i søkeresultatvinduet. Brukeren måtte markere ID-en, mens de fleste prøvde å markere resultatet ved å trykke på fornavnet.

4.2.3.1 I bruk

Spørreskjemaet brukerne fikk utlevert gikk på brukbarheten (”Usability”) til systemet. Det inneholder 10 spørsmål som går på forskjellige aspekter av brukbarheten til SPIM. Spørreskjemaet tar opp emner som: ”Tror du at du vil bruke systemet regelmessig?”, kompleksiteten til systemet, om det var enkelt å bruke og om man trenger mer opplæring.

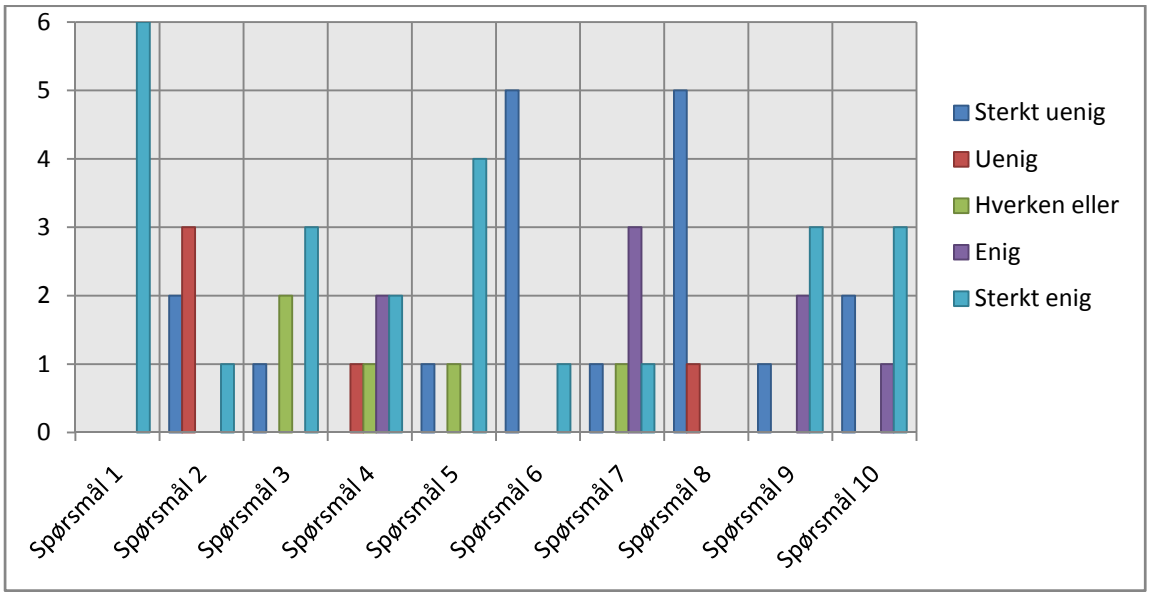
SUS-skjemaet lar deg regne ut et ”brukbarhets-resultat” som kan gi en indikasjon på hvor brukbart brukerne oppfatter systemet. Formel for utregning av SUS-poeng (Brooke 1996):

For oddetall påstandene: Plassering av valgte alternativ - 1

For partall påstandene: 5 – plasseringen av valgte alternativ.

Summer opp resultatene og multipliser med 12,5.

Dette gir deg en poengsum mellom 0 og 100, der et høyere tall indikerer bedre brukbarhet enn et lavt tall. Tallene systemet fikk var mellom 87,5, som det høyeste, og 30, som det laveste. Tre brukere hadde en poengsum på 80 eller høyere, mens gjennomsnittspoengsummen var 69,6. Noe som gir et litt skeivt resultat da fireav seksskjema ga høyere resultat enn gjennomsnittet.



Figur 19 - Graf over resultatene fra spørreskjemaet

Figur 19 viser fordelingen av hvilke svar brukerne gav på de 10 spørsmålene i SUS-skjemaet. Fra denne figuren ser man at alle de seks deltakerne sa seg sterkt enige i spørsmål 1, «I think that I would like to use this system frequently.». Spørsmål 2, «I found the system unnecessarily complex.», bydde på et mer nyansert resultat der to personer sa seg sterkt uenig i spørsmålet, tre personer var uenige og en person var sterkt enig. På det tredje spørsmålet, «I thought the system was easy to use.», svarte én at han var sterkt uenig, to svarte verken eller, og tre sa at de var sterkt enige i at systemet var enkelt å bruke. Én person sa seg uenig, én sa seg verken eller, to sa seg enige og to sa seg sterkt enige i spørsmål 4, «I think I would need the support of a technical person to be able to use this system.». Spørsmål 5, «I found the various functions in this system were well integrated.», viste en uenighet blant deltakerne da én sa at han var sterkt uenig i spørsmålet, én verken eller og fire personer sa seg sterkt enig i formuleringen. Resultatene fra spørsmål 6, «I thought there was too much inconsistency in this system.», viste at fem personer var sterkt uenig i dette, mens én av deltakerne var sterkt enig. Man ser også at på spørsmål 7, «I would imagine that most people would learn to use this system very quickly.»,ga et veldig variert system, der én sa seg sterkt uenig, én verken eller, to var enige og én svarte sterkt uenig på dette punktet. Det var en stor enighet i spørsmål 8, «I found the system very clumbersome to use.», der fem var sterkt uenige og

én var uenig. Spørsmål 9, «I felt very confident using the system.», gav utslag i én sterkt uenig, to enig og tre sterkt enige. Det siste spørsmålet, «I needed to learn a lot of things before I could get going with this system.», viste at deltakerne var splittet i og med at to var sterkt uenige, én var enig og tre var sterkt enige.

Videre fikk de spørsmål om bruken av systemet under intervjuene. Tabell 4 viser de fem grunnspørsmålene under bruk.

De fem grunnspørsmålene under bruk:
1. How did you feel the program functions?
2. What did you miss in the program?
3. What did you find un-logical or hard to do?
4. Why did you try to go back to the previous window by pressing the X in the upper-right corner?
5. Anything else you would like to say about the usability of the system?

Tabell 4- De fem grunnspørsmålene under bruk

Siden det var et semi-strukturert intervju, ble ikke alle spørsmålene stilt alle brukerne, og det kom også opp andre tilleggsspørsmål under intervjuene.

Under intervjuene ble det ytret ulike meninger om brukervennligheten til applikasjonen. De fleste syntes at det var et enkelt og lite komplisert system å bruke, men at de trengte litt mer tid for å bli kjent med systemet. Det var en enighet om at det ikke var noen spesielle ting som var ulogiske eller vanskelige å gjøre. Men en av brukerne uttrykte at han ikke hadde forstått metoden for å benytte seg av systemet:

"I hadn't got the methodof (didn't understand the way to f.anm.) operating it."(Bruker 3)

Mens en annen bruker sa at:

"Every thing was logical about the program. And I think if instructions are closely followed then nothing is hard to do, every thing is viable, is possible and …" (Bruker 1)

og

"There's nothing that is hard, when everything is being explained to you there is nothing hard." (Bruker 5)

Bruker 2 sa at det var ikke noe som var vanskelig å gjøre, men at det var nødvendig å få litt mer tid med systemet slik at du blir vant med å bruke det.

Når det kom til det tilbakevendende problemet med å trykke på X-en øverst til høyre, så viste det seg at dette var noe som EpiHandy tillot deg å gjøre for å gå tilbake til forrige vindu.

"So we thought it would still be the same in this program." (Bruker 2)

Den generelle bruken av programmet ble sagt at var enkel, men at det kunne by på litt problemer grunnet mangelfull opplæring.

"The program is not hard, but more time to, if more time is given to learn it would be good." (Bruker 5),

"It was OK, only that we need more training so that we can get use to the system." (Bruker 2)

og

"I think it's a good program if it being incorporated (with EpiHandy f. anm.) and put in use it's really a good program." (Bruker 6)

Observasjonene stemmer godt overens med resultatene fra spørreskjemaet og fra intervjuene, at de fleste greide å bruke systemet uten særlig opplæring, men at det samtidig var enkelt utfordringer som kunne vært løst med en grundigere opplæring. Under utførelsen av testene hjalp observatør til, men det gikk mest på å forklare hva oppgavene gikk ut på, hva de skulle gjøre. Likevel måtte observatør ved noen anledninger vise hvordan enkelte operasjoner skulle utføres.

4.2.3.2 I praksis

Under intervjuene var det også en kategori med spørsmål som omfattet hvordan brukerne så verdien av SPIM i praksis. Hvordan de oppfattet at det ville hjelpe dem i deres daglige virke (Tabell 5).

De fem grunnspørsmålene under praktiskverdi:
1. How would this program work for you in a real situation?
2. What is the biggest drawback with the program?
3. Anything else?

Tabell 5- Praktiskverdi

Brukerne var positive til å bruke SPIM i deres daglige virke, og så for seg at det ville være til hjelp for dem i arbeidet deres.

”I think with this program is very able, to you are able to identify the people you want to follow up, so easily.” (Bruker 6)

Men også her kom det frem at de ville hatt mer tid på til opplæring, og litt tid for å sette seg inn i applikasjonen for å kunne gi grundig analyser av hvordan programmet ville ha fungert i deres daglige virke.

En bruker oppga at han ønsket at når man klikket på en person i søkeresultatet burde man ha kommet rett til informasjonen om denne personen, og ikke som i dagens versjon der man må markere personen, så gå inn i menyen for å få denne informasjonen.
”When you click on a certain name, you get that information.” (Bruker 3)
Og en annen bruker så det som ønskelig å kunne bruke X-en i øverste høyre hjørne til å gå tilbake med. Siden dette er vanlig i andre applikasjoner de er vant med.
”I think this thing with going back to the previous (window f. anm.) by pressing the X.” (Bruker 6)
Mens andre syntes at de implementerte funksjonen ikke hadde behov for forbedring.
”As by now, I don’t see anything to be improved. To me I feel everything is OK.(…) Maybe when we have used it in the field we can see witch functions to improve.” (Bruker 2)
Når det kom til den største ulempen med SPIM ble enkelte av kommentarene fra de foregående spørsmålene gjentatt, med muligheten til å kunne editere informasjonen og at det var et behov for grundigere opplæring i bruken, noe tre av brukerne kommenterte. Kommentarene gikk på tid til å lære applikasjonen å kjenne, få den inn i fingrene.
”We need to train, train and train, so that it can be a part of us.” (Bruker 4)
Ellers var det ikke noen store ulemper eller haker ved programmet som dukket opp under intervjuene.

4.2.3.3Veien videre

Det var også en gruppe med spørsmål vedrørende den fremtidige utviklingen av SPIM. Om hvordan de så på den videre utviklingen av applikasjonen, hva de ville ha med, hva

som må få større oppmerksomhet og hva som kunne eller burde vært gjort annerledes. I denne delen av intervjuene ble det tatt utgangspunkt iseks grunnspørsmål. (Tabell 6)

De fire grunnspørsmålene under veien videre:
1. What should be done differently?
2. How do you think this will improve during the next years?
3. Are there some benefits for you, or do you believe that this will be a hindrance for you in your work?
4. Are there any functions you feel needs improvement?
5. Where is the greatest room for improvement?
6. Anything else?

Tabell 6 - Veien videre

Når det kom til spørsmål om hvor programmet hadde størst rom for forbedring ble det nevnt at den manglende integreringen med EpiHandy var det som var den største mangelen i SPIM.

«The greatest room for improvement would be if both of them (SPIM and EpiHandy f.anm.) are integrated with EpiHandy (into one system f.anm.)».(Bruker 3)

Også muligheten til å kunne editere informasjonen på de ulike sidene var savnet som bruker 1 sa:

«The greatest room for improvement in the program is the ability to edit on each and every page that I access.»(Bruker 1)

Også hurtigheten på søket ble tatt frem som et område hvor det var behov for å forbedre applikasjonen.

En bruker tok opp at det kunne være en ide å oversette programmet til de lokale språkene i området, slik at brukerne slapp å oversette på direkten til de deltakerne i prosjektet som ikke snakker engelsk. Også bruk av bilder og biometri, som fingertrykk, ble tatt frem som mulige forbedrings muligheter for SPIM. Eller var det nokså entydig at, med god opplæring, ville brukerne ha nytte av programmet. På spørsmål om SPIM vil hjelpe brukerne eller om det vil være en hemsko for dem svarte bruker 6 at:

«It will improve. (…) Because, just as I told you earlier. If you are working with clients whom you are following up, then it is easy for you to track the information and it reduces your ability to make mistakes». (Bruker 6)

og

«When getting use to it (SPIM f. anm.) it can't be of any hindrance to me.»(Bruker 4)

Bruker 2 snakket litt om fordelene med å ha en sikker identifisering av deltakerne for å kunne være sikker på at den innsamlede dataen er korrekt. Og hvordan SPIM vil være behjelpelig med å løse disse utfordringene.

«Avoiding the missing values, the missing data from the respondent.(…) Here what I mean is if the data collected is accurate, has been corrected to the best knowledge of the data collector or the person responsible. That is going to give you good results. I can't say it will be a 100%, but it will be fairly good. And that would eventually lead to proper utilization of resources.»(Bruker 2)

4.3 Data manager

Data manageren er personen som står for å synkronisere dataen fra de håndholdte maskinene og tjeneren på kontoret. Han har også ansvaret for å kontrollere at dataen er korrekt, fullstendig og valid. Han må gå igjennom listen av skjemaene feltarbeiderne har samlet inn. Feltarbeiderne registrere manuelt deltakernavn og ID på hvert skjema. Så må data manageren kontrollere om skjemaet er koblet til riktig deltaker i prosjektet og at deltaker-ID-en er koblet mot riktig navn.Denne registreringen skjer manuelt og her forekommer det feil. I løpet av perioden i Mbale var det flere ganger at data manageren brukte flere timer på å finne ut av de feil registreringene som feltarbeiderne har gjort. For eksempel hadde en feltarbeider skrevet inn deltakernummeret feil (139 i stedet for 193). Dette ble oppdaget av data manageren fordi navnet på det nummeret ikke passet med nummeret på deltakeren i registrene på kontoret. Og da måtte det kontrolleres om det var navnet eller nummeret som var feil. Etter mye arbeid frem og tilbake og etter en samtale med den aktuelle feltarbeideren, fant de ut av problemet.

Et annet problem, som de hadde kommet forbi, var registrerings prosessen. I begynnelsen når de skulle begynne prosessen med registrering av deltakere oppsto det visse problemer knyttet til at flere deltakere fikk likt nummer. Dette løste de ved at data manageren gav feltarbeiderne kort med deltakernummeret skrevet på. Deretter skrev feltarbeiderne på navnene til deltakerne på kortet. Dette førte til at alle deltakerne fikk unike nummer.
Data manageren fikk prøve SPIM, men fikk ikke noen spesifikke oppgaver. Han fikk lov til å utforske applikasjonen og prøve de ulike funksjonene. Observatør var hele tiden til stede og kom med instruksjoner og forklaringer når han ikke fant ut av det selv. En del problemer med SPIM ble oppdaget under denne gjennomgangen (Tabell 7).

Problemer(bugs) med SPIM oppdaget under gjennomgang med data manageren.
Om du vil ha et resultat i søket, må du søke etter enten fornavn eller etternavn.
Mellomnavnet blir tapt når du lagrer pasienten/deltakeren.
Forskjellige navn på navnene (Etternavn er skrevet både som ”surename” og ”family name”).
Om du avslutter programmet vil du ikke kunne starte det igjen uten å restarte PDAen.

Tabell 7 - Problemer med SPIM

4.3.1 Mål

Målet med å observere og intervjue data manageren var å få en annen vinkling på problematikken knyttet til bruk av SPIM og lignende systemer. Å få vite hva en data manager ser etter i et slikt system, og hvordan han tror at SPIM vil kunne løse disse utfordringene. Det var også et mål for meg å la data manageren få en gjennomgang av system med tanke på feil(bugs) og logiske brister, for å prøve å identifisere flest mulig av disse. Videre var hovedmålet med observasjonene og intervjuet å forstå hvordan data manageren arbeider og hvordan SPIM kan være med på å effektivisere og heve kvaliteten på den innsamlede dataen.

4.3.2 Resultat

Resultatene som ble oppnådd var gode, og førte til at det ble utviklet bedre innsikt i data managerens arbeid og oppgaver. Det kom også frem hvilke forventninger data manageren har til et slikt system, som SPIM, og fikk kartlagt hvor SPIM kan være behjelpelig i hans arbeid. Det kom også opp noen punkter som sammenfalt med hva feltarbeiderne hadde

utrykket, samtidig som andre punkter ble belyst. I de følgende avsnittene vil det disse punktenepresentert nærmere.

4.3.2.1 I bruk

Som nevnt i punkt 4.3 ble det gjennomført en gjennom gang av SPIM sammen med data manageren for å la ham bli kjent med applikasjonen, og for å identifisere feil og mangler i denne. Dette ble gjort på kontoret til PROMISE (Figur 17) og data manageren fikk utforske applikasjonen assistanse fra observatør. Det kom opp en del feil og enkelte ting ved brukergrensesnittet som burde forbedres. Tabell 7 viser de feilene (bugs) som dukket opp under denne gjennomgangen. Det ble også observert at data manageren, i likhet med de fleste feltarbeiderne, prøvde å gå tilbake til forrige vindu ved å trykke på X-en opp i høyre hjørne.

Data manageren kommenterte også at i «Add patient» vinduet var det litt forvirrende at «Save» knappen så lik ut som menyen. Han kommenterte også at det burde ha vært mulighet for å lagre fra menyen. At «Save» knappen og menyen så like ut førte til at han ble litt usikker på hvor han skulle trykke for å lagre pasienten.

4.3.2.2 I praksis

På spørsmål om «How do you see an application like SPIM fit in your vision for the future of fieldwork in clinical trails?» svarte data manageren at SPIM bare vil være hjelpsomt om deltakerne blir registrert med riktig informasjon. Om det sniker seg inn en feil under registreringen vil det bli meget vanskelig å finne igjen denne personen.

«If you make an error during the registration, that will be a permanent error.»(Data manager)

Han sier videre at SPIM er en bra applikasjon for identifisering av deltakere, men det kan være problemer med registrering av deltakere om man må fylle ut store spørreskjemaer for å velge ut deltakerne. Data manageren tror at det kan være ugunstig å først registrere deltakerne i SPIM (eller andre lignende applikasjoner), så gå tilbake til feltet ved neste anledning for å registrere alle data som SPIM ikke har tatt høyde for og som er prosjekt spesifikt.

4.3.2.3 Fremtiden ("Veien videre")

For fremtiden mener data manageren at det kan være nyttig å benytte seg av GPS-posisjonering i identifiseringen av deltakere. Slik at når en bruker registrerer en deltaker, så registrerer man samtidig den geografiske posisjonen til denne. Også kort eller andre ting som kan deles ut, kan være greit for identifisering. Samtidig så kan det være et problem om dette kortet forsvinner. Han foreslår heller å feste et kort eller en RFID-brikke på huset til vedkommende, slik at det ikke blir mistet.

Når det kommer til biometri er han mer skeptisk og tror at ressursene kan bli brukt bedre med andre metoder. Biometrisk utstyr er dyrt og i en afrikansk setting ute på landsbygda i Afrika, kan dette utstyret være litt for ømfintlig for støv og sand og lignende. Data manageren tror også det ville føre til litt for mye arbeid før selve intervjuet kommer i gang.

4.4 Oppsummering

Gjennom evalueringen i Mbale, Uganda og PROMISE prosjektet ble SPIM prøvd ut hos de reelle sluttbrukere. Feltarbeiderne gjennomførte 4 oppgaver de fikk utlevert og fikk så utlevert et spørreskjema med 10 påstander om brukbarheten til applikasjonen. Observasjonene fra oppgaveløsningen og svarene fra spørreskjemaet har sammen med intervjuene gitt en bedre innsikt i de fremtidige behovene for feltarbeidere i slike forskningsprosjekter, samtidig med at det har gitt tilbakemeldinger på hvor SPIM står i dag.

Resultatene fra evalueringen viser at feltarbeiderne er positive til en slik applikasjon, og alle svarte at det var svært enig i påstanden: «I think that I would like to use this system frequently.» Det som fremstår som det største problemet var manglende tid til opplæring. Når de fikk forklart hvordan applikasjonen skulle brukes, gikk det greit. Av rene brukergrensesnitt problem som ble identifisert var muligheten for å gå tilbake til forrige vinduet med å trykke på X-en i øverste høyre hjørne det som ble savnet av flest. Fem av seks feltarbeidere som deltok i testene prøvde å gå tilbake ved å trykke på X-en, samt at data manageren også prøvde å gjøre det under sine tester av SPIM. Eller mente de fleste

at SPIM ville være til hjelp for dem i deres arbeid ute i feltet. Tabell 8 viser hvilke forslag til nye funksjoner feltarbeiderne kom med. Opplæring i SPIM var det som gikk igjen som det som burde ha vært bedre og at om feltarbeiderne hadde fått mer tid til opplæring og mer tid til å bli kjent med programmet ville de ha kunnet gi flere tilbakemeldinger på manglende funksjoner i SPIM.

Forslag til nye funksjoner til SPIM
Mulighet til å kunne editere informasjonen på hver side.
Mulighet for å gå tilbake til forrige vindu ved å trykke på X-en.
Lagre bilder av deltakerne.
Oversettelser til lokale språk.
Få opp informasjonen om en person når du klikker på denne.

Tabell 8- Nye funksjoner

Feltarbeiderne fikk også ut noen spørreskjema hvor de skulle gradere hvor enige de var i 10 påstander om brukbarheten til SPIM. Resultatene varierte fra 87,5 av 100 til 30 av hundre, men hvor fire av seks personer ga applikasjonen en gradering på over 70 av 100. Det som trakk mest ned var manglende opplæring. Alle feltarbeiderne sa seg svært enig at SPIM var en applikasjon de tror de ville ha brukt ofte om den hadde vært tilgjengelig for dem.

Data manageren kom også med innspill og han så for seg at enkelte problem som oppstår på grunn av manuell inntasting av skjemanummer kan elimineres ved bruk av SPIM. Han så også for seg at registrerings prosessen kunne bli enklere siden systemet gir deltakerne nummer automatisk. Likevel ser han at det kan oppstå flere problemer knyttet til bruken av SPIM. For eksempel hevdet han at det kunne bli tungvint å benytte seg av SPIM til registrering i prosjekter hvor registreringen er store spørreskjemaer som skal fylles ut før utvelgelsen av deltakerne.
Så selv om det var enkelte feil og mangler i applikasjonen ble SPIM tatt vel i mot av testerne i Mbale. De sa at en applikasjon som SPIM kan være til hjelp for dem i deres virke, samtidig som det fra data managerens side kan føre til litt mer troverdig data.

5 Konklusjon og fremtidig arbeid

5.1 Konklusjon

I denne masteroppgave har det blitt utviklet en applikasjon for registrering og identifisering av deltakere i kliniske forsøk i lavinntektsland. Oppgaven har vært gjort i sammen med CIH og er tilknyttet OMEVAC. Utviklingen av SPIM har vært todelt, der det har blitt lagd modeller av et system for identifisering og registrering, så har det blitt utviklet en implementering av disse modellene. I feltarbeidet i Uganda ble det mest fokus på brukervennligheten til systemet, og da prøve å få en forståelse av hva de mener om de underliggende funksjonene i systemet. Hovedfokus og problemstilling gjennom hele denne tiden har vært: «*Hvordan kan ITbistå i identifiseringen og gjenkjennelsen av personer i klinisk forskning (Clinical Trials) i lavinntektsland*».

Det har blitt undersøkt dette ved å la en gruppe feltarbeidere ved PROMIS's feltkontor i Mbale, Uganda (Figur 16). Hver feltarbeider fikk ett sett med oppgaver som de skulle løse ved hjelp av SPIM, slik at de fikk prøvd de ulike funksjonene min implementering av SPIM tilbyr. Målene med denne testingen var å undersøke om SPIM ville være til hjelp for dem i deres daglige virke, hvordan brukervennligheten var, og å finne muligheter for å forbedre applikasjonen. På feltkontoret i Mbale ble det også foretatt et intervju med data manageren om hans tanker om hvordan et system som SPIM kan være til hjelp i hans hverdag, og hvilke muligheter han så for seg ved bruk av SPIM.

Oppgaven begynte med å planlegge å utvikle SPIM, før testing og evaluering av applikasjonen ble gjort i Uganda. Resultatene fra arbeidet i Uganda gav resultater som kan tyde på at det vil være hensiktsmessig å benytte seg av IT til å registrere og identifisere deltakere i kliniske forsøk i lavinntektsland. Feltarbeiderne gav uttrykk for at et system som SPIM ville være til hjelp for dem i deres arbeid, og at det med riktig opplæring ville føre til færre feil i datainnsamlingen. Det ble også avdekket enkelte forbedrings potensialer i løpet av evalueringen (Tabell 8). Og noe av det første feltarbeiderne ville ha på plass var integrering med EpiHandy.

I denne oppgave har det blitt vist at et IT-basert system for registrering og identifisering av deltakere i kliniske forsøk i lavinntektsland, integrert med et generelt system for gjennomførelse av slike forsøk antageligvis vil kunne være av nytte for de organisasjoner som står for gjennomførelsen. I løpet av den tiden denne oppgaven har vært gjennomført har det blitt utviklet en prototype av et system for identifisering og registrering av personer tilknyttet kliniske forsøk i lavinntektsland kalt SPIM. Denne prototypen og de testene av denne som ble gjort viser at et slikt system kan ha en viss nytteverdi. Prototypen er blitt testet i Mbale i Uganda, i samarbeid med PROMISE prosjektet. Oppholdet i Mbale varte i 16 dager, hvor det blitt gitt en mulighet til å bli bedre kjent med hvordan en slik studie gjennomføres. Oppgaven viser også et eksempel på hvordan man kan benytte en prototype som et verktøy for å undersøke en hypotese.

En slik oppgave, som denne, vil ikke kunne komme med noen endelige bevis for nytten av et system som SPIM. Til det har det vært en for kort tid, med for få ressurser. Oppgaven kan ses på som et lite steg på en lang reise for å utvikle et sikkert system for registrering og identifiseringa av deltakere i kliniske forsøk, og for å kunne fastslå nytten av slike system.

5.2 Fremtidig arbeid

Gjennom oppholdet i Uganda, kom det frem feil og mangler med SPIM. Samtidig kan disse funnene indikere at en videre utvikling av SPIM med en integrering mot EpiHandy og OMEVAC kan være ønskelig. Det er en lang vei å gå før SPIM oppfyller alle krav og forventninger til et slikt system. Det er flere utfordringer som må løses før en endelig versjon kan lanseres. Noen av de største og viktigste funksjonene som må implementeres er integrering med EpiHandy (OMEVAC) og kryptering av lagrede data. Samt gode metoder for å sikre at data og informasjon ikke kommer på avveie. Likevel er det rimelig å anta at fordelene med å kunne tilby en sikker kobling mellom innsamlet data og deltakerne i forsøkene vil kunne veie tyngre enn de utfordringene som må løses før systemet kan lanseres.

For å sikre at SPIM overholder internasjonale krav og reguleringer,som 21 CFR part 11 (Shultz 1997), må det arbeides med å finne metoder for lagring av personopplysninger og behandling av disse som overholder gjellende lover og reguleringer. For at resultatene fra forskningen skal bli godkjent internasjonalt, er det avgjørende at systemene hvor data blir lagret og behandlet oppfyller internasjonale lover. Det må også utredes for hvilke trusler som oppstår og identifisere måter å imøtekomme disse truslene. Personvernet må ivaretas og veies opp mot de positive effektene for samfunnet forskningen vil kunne oppnå. For å sørge for at den innsamlete informasjonen er gyldig og valid er det viktig å kunne koble det innsamlete materialet mot enkelte deltakere. Med en slik kobling blir det enklere å gjennomføre revisjoner med tanke på hvor materialet ble samlet inn, når og på hvilken deltaker og hvem som foretok registreringen. Om et slikt system blir implementert, som forslått med SPIM, er det av svært høy betydning å sikre at det er kun autoriserte brukere med riktige tilgangsrettigheter som kan få ut slike opplysninger. Her må det legges ned betydelig arbeid for å ikke bryte personvernet og lege-pasient konfidensialiteten.

For å sikre en så sikker identifisering av deltakere i forkant av datainnsamlingen kan det være store gevinsteri å undersøke flere forskjellige modeller for identifisering av personer. Ny teknologi kommer på markede og blir billigere, noe som kan føre til at det kan bli mulig å benytte seg av ulike former for biometri og, eller i kombinasjon med, RFID (Eschet 2004) og smartkort (Smartcard). Likevel kan det oppstå problemer verdrørende slike systemer, som i McDonalds (McDonalds 2006) artikkel hvor det blir oppdaget at strekkoden til en pasient er gitt en annen, og hvor nært det var å gi pasientene feil behandling grunnet troen på at strekkoden ikke kunne være gal. Derfor kan det være nyttig å studere hvordan ulike metoder for identifisering kan kombineres for å få en så sikker som mulig registrering og identifisering.

Det kan også være lurt å se på utviklingen av ulike prosedyrer for registrering og identifisering samtidig med utviklingen av selve systemet. Dette for å kunne tilpasse den nye teknologien til nye arbeidsprosedyrer for å maksimere nytteverdien av de nye systemene. Å erstatte manuell registrering og identifisering med et elektronisk system for

tilsvarende prosedyrer uten å redesigne prosedyrene vil ikke gi den maksimale effekten av teknologien.

Referanseliste

(2005, 23-12-2005). "Centre for International Health, University of Bergen, Norway " Retrieved 5 of February, 2008, from http://www.cih.uib.no/index.php?valg=sql&id=25.

(2008, 9 April 2008). "About OMEVAC - EpiHandy." Retrieved 5 of May, 2008, from http://www.epihandy.com/wiki/index.php/About_OMEVAC.

Alliance, T. A. (2006, May 14 2006). "Agile Alliance: What Is Agile Software Development." Retrieved 28. jan, 2008, from http://www.agilealliance.com/show/2.

Ambler, S. W. (2004). The Object Primer: Agile Model-Driven Development with UML 2.0. Cambridge, Cambridge University Press.

Beck, K., M. Beedle, et al. (2001, 2001). "Manifesto for Agile Software Development." Retrieved 28 jan, 2008, from http://agilemanifesto.org/.

Beck, K., M. Beedle, et al. (2001). "Principles behind the Agile Manifesto." Retrieved 01/03, 2008, from http://www.agilemanifesto.org/principles.html.

Brooke, J. (1996). SUS - A 'quick and diry' usability scale. Usability Evaluation in Industry. P. W. Jordan. London, CRC Press**:** 189-194.

Clarke, R. (1994). "Human Identification in Information Systems: Management Challenges and Public Policy Issues." Information technology & people**7**(4): 32.

ClinicalTrials.gov. (2007, 2007/09/20). "Understanding Clinical Trials." Retrieved 02-02, 2008, from http://clinicaltrials.gov/ct2/info/understand.

de Villiers, M. R. (2005). Three approaches as pillars for interpretive information systems research: development research, action research and grounded theory. Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries. White River, South Africa, South African Institute for Computer Scientists and Information Technologists.

Dix, A., J. Finlay, et al. (2004). Human-Computer Interaction. Essex, Pearson Education Limited.

Eschet, G. (2004). "FIPs and PETs for RFID: Protecting Privacy in the Web of Radio Frequency Identification." Jurimetrics,The Journal of Law, Science, and Technology**45**: 54.

Gold, R. L. (1958). "Roles in Sociological Field Observations." Social Forces**36**(3): 7.

Highsmith, J. (2001, 2001). "History: The Agile Manifesto." Retrieved 28. jan, 2008.

Klungsøyr, J. (2007). OMEVAC (Open Mobile Electronic Vaccine Trials), an interdisciplinary
project to improve quality of vaccine trials in low-resource settings. Bergen, Centre for International Health**:** 10.

Koop, A. and R. Mösges (2002). "The use of handheld computers in clinical trials." Controlled Clinical Trials**23**(5): 12.

Lau, F. (1999). "Toward a framework for action research in information systems studies." Information technology & people**12**(2): 148.

McDonalds, C. J. (2006). "Copmuterization Can Create Safety Hazards: A Ba-Coding Near Miss." Annals of Internal Medicine**144**(7): 8.

Rapoport, R. N. (1970). "Three dilemmas in action research." Human relations**23**(6): 499.

Robson, C. (2002). Real World Research. Oxford, Blackwell Publishing.

Sargent, J. and K. Michael (2005). The need for a digital aid framework in humanitarian relief. The 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2005), Orlando, Florida, USA.

Selic, B. (2003). "The Pragmatics of Model-Driven Development." IEEE Software**20**(5): 7.

Shankar, R. D., S. B. Martins, et al. (2006). Epoch: an ontological framework to support clinical trials management. Proceedings of the international workshop on Healthcare information and knowledge management. Arlington, Virginia, USA, ACM Press.

Shultz, W. B. (1997). 21 CFR Part 11. D. o. H. A. H. S.-F. a. D. Administration**:** 38.

Susman, G. C. and R. D. Evered (1978). "An Assessment of the Scientific Merits of Action Research." Administrative science quarterly**23**(4): 582.

Tulleskär, T. (2005). Promoting infant health and nutrition in Sub-Saharan Africa: Safety and efficacy of exclusive breastfeeding promotion in the era of HIV. Burkina Faso, Uganda, Zambia, South Africa, University of Bergen.

UNCHR (2001). Practical Aspect Of Physical and Legal Protection With Regards to Registration. Global Consultations on International Protection.

Appendiks A – Intervjuskjema med feltarbeidere

Interview

Project
This interview is part of the evaluation for the SPIM-project.

Goal
The goal of the tasks, the questionnaire and the interview are to test SPIM in a real-life setting, and to let potential users of the program try, test and evaluate the different functions.

Confidentiality
Respondents are anonymous, and no names or images are related to the questionnaire or the interview. The recording of this interview will be deleted when the project is ended. You have the possibility to stop the interview at any given time.

The interview will last for approximately for fifteen minutes.

Personal information
1. How old are you?
2. Male or female?
3. What is your highest finished education?
4. How long have you been working as a field worker?
5. How long have you been collecting data on PDAs?

Category 1 – Use of the program.
1. How did you feel the program functions?
2. What did you miss in the program?
3. What did you find un-logical or hard to do?
4. Why did you try to go back to the previous window by pressing the X in the upper-right corner?
5. Any thing else you would like to say about the usability of the system?

Category 2 – Practicality in a real setting.
1. How would this program work for you in a real situation?
2. Are there some functions you feel needs improvement?
3. Where are the greatest room for improvement?
4. What is the biggest drawback with the program?
5. Any thing else?

Category 4 – Your thoughts on the development (what should be done).

1. What should be done differently?
2. How do you think this will improve during the next years?
3. Are there some benefits for you, or do you believe that this will be a hindrance for you in your work?
4. Any thing else?

Category 5 – Open part

1. Is there something you would like to add?

Appendiks B – Intervjuskjema for data manager

Interview with Data manager for the PROMISE project, Uganda site.

The interview will be taped, but will be handled with strict confidentiality. The recorded interview will be deleted at the end of my master thesis.

1) What is your main responsibility in the PROMISE study, and what is your title?
2) What does PROMISE stand for, and what are the objectives for the project?
3) When did PROMISE fieldwork in Mbale start and when will they end?
4) Which level trail is conducted here (level 1, 2 or 3 trails)?
5) How many participants are registered for the trails?
6) How do you keep track of the participants?
7) How long have you been working with the use of PDAs for data collection?
8) What is your experience with using mobile devices for the data collection (pro and cons)?
9) What are the biggest challenges by handling data collected by mobile devices?
10) What is the most common reason for errors?
11) How do these errors influence the quality of the gathered data?
12) How can these reasons for errors be prevented?
13) How do you want the way fieldwork is conducted to evolve during the next years?
14) What are your feelings towards Electronic Data Capturing (EDC)?
15) How do you see an application like SPIM fit in your vision for the future of fieldwork in clinical trails?
16) How would an application like SPIM influence your work as a data/an information manager?
17) What do you feel is the most needed functionality for an application like SPIM?

Appendiks C – Oppgaver for sluttbrukere

Task for End-user testing.

Task 1 – The first search

A) Login to SPIM with your username and password.
B) Search for a patient named “Clara Muvule” use term “Cla” as first name.

Task 2 – Adding a patient

A) Add a new patient with the following attributes:
    a. First name: Kizza
    b. Middle name: Yoweri
    c. Family name: Orombi
    d. Birth date: 1974-03-21
    e. Birth place: Kampala
    f. Citizenship: Ugandan
    g. Mothers name: Claire Orombi
    h. Gender: Male
    i. Address: 20 Masaba Road
    j. City: Mbale
    k. County/District: Mbale
    l. Country: Uganda
    m. Health district: Mbale
    n. Patient group : “Groupe #1”
B) Save the patient
C) Search for that patient

Task 3: Adding a new form

A) Login to Spim
B) Search for “Clarie”
C) Chose Claire in the search result window.
D) Chose “New form” from the “Edit” menu
E) Give the form number you want to add to this participant.
F) Save that number

Task 4: Viewing all the forms collected on one participant

A) Login to Spim
B) Search for “Clara”
C) Chose Clara from the search result list.
D) Chose “All forms” from the “Edit” menu.

Appendiks D – System Usability Scale

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Appendiks E – Resultat fra SUS-skjema

B1	1	2	3	4	5
1					x
2	x				
3			x		
4		x			
5			x		
6	x				
7			x		
8	x				
9				x	
10	x				

B2	1	2	3	4	5
1					x
2		x			
3					x
4				x	
5					x
6	x				
7				x	
8	x				
9					x
10	x				

B3	1	2	3	4	5
1					x
2					x
3					x
4					x
5	x				
6					x
7	x				
8	x				
9	x				
10					x

B4	1	2	3	4	5
1					x
2		x			
3					x
4			x		
5					x

6	x				
7				x	
8	x				
9					x
10				x	

B5	1	2	3	4	5
1					x
2	x				
3	x				
4					x
5					x
6	x				
7					x
8	x				
9					x
10					x

B6	1	2	3	4	5
1					x
2		x			
3			x		
4				x	
5					x
6	x				
7				x	
8		x			
9				x	
10					x

Appendiks F – SqlCePatientDAO.cs

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.Text;
using spimCFLib.DAO.Interfaces;
using BplCe.Model.PatientInfo;
using System.Data.SqlServerCe;
using BplCe.Model.PersonInfo;
using BplCe.Model.UserInfo;
using System.Data.SqlTypes;

namespace spimCFLib.DAO.SqlCeImplementation
{
/**
      * This class implements the interface PatientDAO, and is specifically for use
in
      * the context where Microsoft SQL Server Compact Edition is used as the
database.
      */
class SqlCePatientDAO : PatientDAO
   {

       System.Text.RegularExpressions.Regex isGuid = new
System.Text.RegularExpressions.Regex("[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-
F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}");
/**
        * @method FindPatient
        * @param p - patient object which contains the search criterias we wish to
search for in the database.
        * @return a collection of Patient-objects from the database that match the
search criteria of p.
        */
public ICollection<Patient> FindPatient(Patient p)
      {
//Creates the link to the DB.
          SqlCeCommand command =
SqlCeSingleton.GetInstance().CreateCommand();

//The WHERE clauses for the query.
string searchString = GetSearchCriteria(p);

//The query.
```

```
        command.CommandText = "SELECT * FROM patient " + searchString;

//Reads the result from the query
        SqlCeDataReader DataReader = command.ExecuteReader();

//Creates a collection of Patient-objects where I store the patients from the DB.
        ICollection<Patient> PatientCollection = new List<Patient>();

//While there are more rows from the query.
while (DataReader.Read())
        {
            //For every field in each row.
for (int i = 0; i < DataReader.FieldCount; i++)
            {
                Patient tempPatient = new Patient();
string ColumnName = DataReader.GetName(i);

                //Each column name has a different "action"
switch (ColumnName)
                {
case"id":
                        tempPatient.Id = (Guid)DataReader.GetValue(i);
break;
case"patient_groupe_id":
                        tempPatient.PatientGroupeId = new PatientGroupe();
                        if(isGuid.IsMatch(DataReader.GetValue(i).ToString()))
                            tempPatient.PatientGroupeId.Id =
(Guid)DataReader.GetValue(i);
break;
case"creator":
                        tempPatient.Creator = new User();
if (isGuid.IsMatch(DataReader.GetValue(i).ToString()))
                            tempPatient.Creator.Id = (Guid)DataReader.GetValue(i);
break;
case"date_created":
if (DataReader.GetValue(i).ToString() != "") {
string stringDate = DataReader.GetValue(i).ToString();
                            tempPatient.DateCreated = DateTime.Parse(stringDate);
                        }
break;
case"changed_by":
                        tempPatient.ChangedBy = new User();
if (isGuid.IsMatch(DataReader.GetValue(i).ToString()))
                            tempPatient.ChangedBy.Id = (Guid)DataReader.GetValue(i);
break;
case"date_changed":
```

```
if (DataReader.GetValue(i).ToString() != "")
                        {
string stringDate = DataReader.GetValue(i).ToString();
                            tempPatient.DateChanged = DateTime.Parse(stringDate);

                        }
break;
case"voided":
String VoidedTemp = DataReader.GetValue(i).ToString();
bool Voided;
if (VoidedTemp == "0")
                        {
                            Voided = false;
                        }
else
                        {
                            Voided = true;
                        }
                        tempPatient.Voided = Voided;
break;
case"voided_by":
                        tempPatient.VoidedBy = new User();
if (isGuid.IsMatch(DataReader.GetValue(i).ToString()))
                            tempPatient.VoidedBy.Id = (Guid)DataReader.GetValue(i);
break;
case"date_voided":
if (DataReader.GetValue(i).ToString() != "") {
                            string stringDate = DataReader.GetValue(i).ToString();
                            tempPatient.DateVoided = DateTime.Parse(stringDate);
                        }
break;
case"void_reason":
                        tempPatient.VoidReason = DataReader.GetValue(i).ToString();
break;
                }
                //We add the patient to the collection:
                PatientCollection.Add(tempPatient);
            }
        }

        //Next, we find and add any existing identifier(s) for each patient in
PatientCollection:
foreach (Patient tempPatient in PatientCollection)
        {
//Creates the link to the DB.
```

```
            SqlCeCommand command2 =
SqlCeSingleton.GetInstance().CreateCommand();

//We are always interested only those records existing that belong to the current
tempPatient-object.
            command2.CommandText = "SELECT * FROM patient_identifier
WHERE patient_id='" + tempPatient.Id.ToString() + "'";

//Reads the result from the query
            SqlCeDataReader dataReader = command2.ExecuteReader();

//While there are more rows from the query.
while (DataReader.Read())
            {
               //For every field in each row.
for (int i = 0; i < dataReader.FieldCount; i++)
               {
                  PatientIdentifier tempPatientIdentifier = new PatientIdentifier();
string ColumnName = dataReader.GetName(i);

//Each column name has a different "action"
switch (ColumnName)
                  {
case "identifier":
                     tempPatientIdentifier.Identifier =
(String)dataReader.GetValue(i);
break;
case"identifier_type":
                     tempPatientIdentifier.IdentifierType.Id =
(int)dataReader.GetValue(i);
break;
case"location_id":
                     tempPatientIdentifier.Location = new Location();
if(isGuid.IsMatch(dataReader.GetValue(i).ToString()))
                        tempPatientIdentifier.Location.Id =
(Guid)dataReader.GetValue(i);
break;
case"check_digit":
                     tempPatientIdentifier.CheckDigit =
(int)dataReader.GetValue(i);
break;
case"preferred":
String PreferredTemp = dataReader.GetValue(i).ToString().Trim();
bool Preferred;
if (PreferredTemp == "0")
                     {
```

```
                    Preferred = false;
                }
else
                {
                    Preferred = true;
                }
                tempPatientIdentifier.Preferred = Preferred;
break;
case"creator":
                tempPatientIdentifier.Creator = new User();
if(isGuid.IsMatch(dataReader.GetValue(i).ToString()))
                    tempPatientIdentifier.Creator.Id =
(Guid)dataReader.GetValue(i);
break;
case"date_created":
if (dataReader.GetValue(i).ToString() != "")
                {
string tempDate = dataReader.GetValue(i).ToString();
                    tempPatientIdentifier.DateCreated =
DateTime.Parse(tempDate);
                }
break;
case"voided":
String VoidedTemp = dataReader.GetValue(i).ToString();
bool Voided;
if (VoidedTemp == "0")
                {
                    Voided = false;
                }
else
                {
                    Voided = true;
                }
                tempPatientIdentifier.Voided = Voided;
break;
case"voided_by":
                tempPatientIdentifier.VoidedBy = new User();
if(isGuid.IsMatch(dataReader.GetValue(i).ToString()))
                    tempPatientIdentifier.VoidedBy.Id =
(Guid)dataReader.GetValue(i);
break;
case"date_voided":
if (dataReader.GetValue(i).ToString() != "")
                {
string tempDate = dataReader.GetValue(i).ToString();
```

```
                        tempPatientIdentifier.DateVoided =
DateTime.Parse(tempDate);
                    }
break;
case"void_reason":
                    tempPatientIdentifier.VoidReason =
dataReader.GetValue(i).ToString();
break;
                }
//We add the new identifier to the temporary stored patient:
                tempPatient.Identifier.Add(tempPatientIdentifier);
            }
        }
        command2.Dispose();
    }
    ICollection<Patient> newCollection = new List<Patient>();
/* Adds the person-attributes, personName and personAddress from the
database */
foreach (Patient pa in PatientCollection)
    {
string paID = pa.Id.ToString();
if (isGuid.IsMatch(paID) && (paID != "00000000-0000-0000-0000-
000000000000"))
        {
            SqlCePersonDAO pers = new SqlCePersonDAO();
            Person tempPerson = new Person();
if (pers.FindPersonById(pa.Id) != null)
{
                tempPerson = pers.FindPersonById(pa.Id);
                pa.Gender = tempPerson.Gender;
pa.MothersName = tempPerson.MothersName;
                pa.Names = tempPerson.Names;
                pa.Race = tempPerson.Race;
                pa.Addresses = tempPerson.Addresses;
                pa.BirthDate = tempPerson.BirthDate;
                pa.BirthDateEstimated = tempPerson.BirthDateEstimated;
                pa.BirthPlace = tempPerson.BirthPlace;
                pa.CauseOfDeath = tempPerson.CauseOfDeath;
                pa.Citizenship = tempPerson.Citizenship;
                pa.DeathDate = tempPerson.DeathDate;
                pa.HealthCenter = tempPerson.HealthCenter;
                pa.HealthDistrict = tempPerson.HealthDistrict;
            }
        }
        newCollection.Add(pa);
    }
```

```
            PatientCollection = newCollection;

            ICollection<Patient> FinalCollection = new List<Patient>();
foreach (Patient pat in PatientCollection)
            {
if (isGuid.IsMatch(pat.Id.ToString()) && (pat.Id.ToString() != "00000000-0000-
0000-0000-000000000000"))
                {
                    FinalCollection.Add(pat);
                }
            }

            /* Finally, return the collection with the patient-objects that match your
             * patient search-criteria, complete with identifiers:
             */
            command.Dispose();


return FinalCollection;
        }//End FindPatient

/**
         * @method GetSearchCriteria
         *
         * @author Jo Are Ingvaldsen, jin021@student.uib.no
         * @co-author by Knut Ivar Myklebust, kmy041@student.uib.no
         *
         * @desc This method takes the fields that is set from your search object,
         * and puts their value into a search string.
         *
         * @param p - The Patient-object with the search criterias
         * @return string - The search string
         */
privatestring GetSearchCriteria(Patient p)
        {
string search = "WHERE ";

/**
             * These if-sentences builds the search string.
             * If a field is set, then include this in the search string
             * with the value from the object.
             */
if (p.Id != Guid.Empty)
                search += "id = '" + p.Id + "' AND ";
if (p.PatientGroupeId != null&& p.PatientGroupeId.Id != Guid.Empty)
```

```
            search += "patient_groupe_id = '" + p.PatientGroupeId.Id + "' AND ";
if (p.Creator != null&& p.Creator.Id != Guid.Empty)
            search += "creator = '" + p.Creator.Id + "' AND ";
if (p.DateCreated != null)
            search += "date_created = '" + p.DateCreated + "' AND ";
if (p.ChangedBy != null&& p.ChangedBy.Id != Guid.Empty)
            search += "changed_by = '" + p.ChangedBy.Id + "' AND ";
if (p.DateChanged != null)
            search += "date_changed = '" + p.DateChanged + "' AND ";
if (p.Voided != null)
            search += "voided ='" + p.Voided + "' AND ";
if (p.Voided != null&& p.VoidedBy.Id != Guid.Empty)
            search += "voided_by = '" + p.VoidedBy.Id + "' AND ";
if (p.DateVoided != null)
            search += "date_voided ='" + p.DateVoided + "' AND ";
if (p.VoidReason != null)
            search += "void_reason = '" + p.VoidReason + "' AND ";

//Takes away the last 'AND ' from the search string, which could cause some
trouble
if (search != "WHERE ")
        {
if (search.Substring(search.Length - 4, 4) == "AND ")
            search = search.Remove(search.Length - 4, 4);
        }
else
            search = "";

//Returns the search string
return search;
    }//End GetSearchCriteria

publicvoid SaveOrUpdatePatient(Patient p)
    {
        ICollection<Patient> patientExists = new List<Patient>();
if (isGuid.IsMatch(p.Id.ToString()) && p.Id.ToString() != "00000000-0000-0000-
0000-000000000000")
        {
            Patient temp = FindPatientById(p.Id);
patientExists.Add(temp);
        }
else
            patientExists = FindPatient(p);

//Creates the link to the DB.
```

```
        SqlCeCommand command =
SqlCeSingleton.GetInstance().CreateCommand();

int voided = 0;
if (p.Voided == true)
        {
            voided = 1;
        }

//If you are about to save a new Patient, this patient isn't in the patientExists
collection.
if (!(patientExists.Contains(p)))
        {
//SAVING/INSERTING NEW TUPLE

//Save the person-object
            SqlCePersonDAO personDAO = new SqlCePersonDAO();
            Person tempPerson = PatientToPerson(p);
            personDAO.SaveOrUpdatePerson(tempPerson);
            ICollection<Person> personList =
personDAO.FindPerson(tempPerson);
foreach (Person per in personList)
            {
                tempPerson = per;
            }

string sqlText = GetInsertText(p);
            sqlText += GetInsertValues(p, tempPerson);

            command.CommandText = sqlText;

            command.ExecuteNonQuery();

            //into table patient_identifier
if (p.Identifier != null)
            {
foreach (PatientIdentifier pi in p.Identifier)
                {
                    command.CommandText =
"INSERT INTO patient_identifier (" +
"identifier," +
" identifier_type," +
" location_id," +
" check_digit," +
                    " preferred," +
                    " patient_id," +
```

```
" creator," +
" date_created," +
                    " voided," +
" voided_by," +
                    " date_voided," +
" void_reason) " +
"VALUES('" +
                    pi.Identifier + "', '" +
                    pi.IdentifierType + "', '" +
                    pi.Location.Id + "', '" +
                    pi.CheckDigit + "', '" +
                    pi.Preferred + "', '" +
                    p.Id + "', '" +
                    pi.Creator + "', '" +
                    pi.DateCreated + "', '" +
                    pi.Voided + "', '" +
                    pi.VoidedBy + "', '" +
                    pi.DateVoided + "', '" +
                    pi.VoidReason + ")";

                    command.ExecuteNonQuery();
                }//End foreach PatientIdentifier
            }// End if p.Identifier != null
        }
else
        {
//UPDATING VALUES:

            //STEP 1: COPYING DATA TO HISTORY TABLES

            //Copying tuple from table patient to table history_patient
            command.CommandText = "INSERT INTO history_patient (" +
"patient_id, " +
"patient_groupe_id, " +
"timestamp, " +
                                "creator, " +
"date_created, " +
"changed_by, " +
"date_changed, " +
"voided, " +
"voided_by, " +
"date_voided, " +
                                "void_reason" +
") " +
"SELECT * " +
"FROM patient " +
```

```
"WHERE id ='" + p.Id + "';";

            command.ExecuteNonQuery();

            //Copying tuple(s) from table patient_identifier to table
history_patient_identifier
foreach (PatientIdentifier pi in p.Identifier)
            {
                command.CommandText = "INSERT INTO history_patient_identifier
(" +
"identifier, " +
"identifier_type, " +
"location_id, " +
                                    "timpestamp, " +
"check_digit, " +
                                    "preferred, " +
"patient_id, " +
"creator, " +
"date_created, " +
"voided, " +
"voided_by" +
"date_voided, " +
"void_reason" +
                                    ") " +
"SELECT * " +
"FROM patient_identifier " +
"WHERE patient_id = " + p.Id + ";";

                command.ExecuteNonQuery();
            }


            //STEP 2: UPDATE EXISTING TABLES WITH NEW DATA

            //Updating patient table:
            command.CommandText = "UPDATE patient " +
"SET " +
"patient_groupe_id = " + p.PatientGroupeId +
", creator = " + p.Creator.Id +
", date_created = " + p.DateCreated +
", changed_by = " + p.ChangedBy.Id +
", date_changed = " + p.DateChanged +
", voided = " + voided +
", voided_by = " + p.VoidedBy.Id +
                            ", date_voided = " + p.DateVoided +
", void_reason = " + p.VoidReason + " " +
```

```
"WHERE id = " + p.Id + ";";
                command.ExecuteNonQuery();

//Updating patient_identifier_table
foreach (PatientIdentifier pi in p.Identifier)
                {
                    command.CommandText = "UPDATE patient_identifier " +
"SET " +
"identifier = " + pi.Identifier +
", identifier_type = " + pi.IdentifierType +
", check_digit = " + +pi.CheckDigit +
", preferred = " + pi.Preferred +
", patient_id = " + p.Id +
                                        ", creator = " + pi.Creator +
", date_created = " + pi.DateCreated +
", voided = " + voided +
", voided_by = " + pi.VoidedBy +
", date_voided = " + pi.DateVoided +
", void_reason = " + pi.VoidReason +
"WHERE patient_id = " + p.Id + ";";
                    command.ExecuteNonQuery();
                }
            }
        }//End SaveOrUpdatePatient

privatestring GetInsertText(Patient p)
        {
string text = "INSERT INTO patient (id, ";

if(p.PatientGroupeId != null)
                text += "patient_groupe_id, ";
if (p.Creator != null)
                text += "creator, ";
if(p.DateCreated != null)
                text += "date_created, ";
if(p.ChangedBy != null)
                text += "changed_by, ";
if(p.DateChanged != null)
                text += "date_changed, ";
if(p.Voided != null)
                text += "voided, ";
if(p.VoidedBy != null)
                text += "voided_by, ";
if(p.DateVoided != null)
                text += "date_voided, ";
if(p.VoidReason !=  null)
```

```
            text += "void_reason, ";

        //Takes away the last ', ' from the search string, which could cause some
trouble
if (text.Substring(text.Length - 2, 2) == ", ")
            text = text.Remove(text.Length - 2, 2);

        text += ")";
return text;
    }//End GetInsertText

privatestring GetInsertValues(Patient p, Person tempPerson)
    {
string text = " VALUES(";

if (isGuid.IsMatch(tempPerson.Id.ToString()))
            text += "'" + tempPerson.Id.ToString() + "', ";
if (p.PatientGroupeId != null)
            text += "'" + p.PatientGroupeId.Id.ToString() + "', ";
if (p.Creator != null)
            text += "'" + p.Creator.Id.ToString() + "', ";
if (p.DateCreated != null)
            text += "'" + p.DateCreated.ToString() + "', ";
if (p.ChangedBy != null)
            text += "'" + p.ChangedBy.Id.ToString() + "', ";
if (p.DateChanged != null)
            text += "'" + p.DateChanged.ToString() + "', ";
if (p.Voided != null)
        {
int voided = 0;
if(p.Voided.Value)
                voided = 1;
            text += "'" + voided + "', ";
        }
if (p.VoidedBy != null)
            text += "'" + p.VoidedBy.Id.ToString() + "', ";
if (p.DateVoided != null)
            text += "'" + p.DateVoided.ToString() + "', ";
if (p.VoidReason != null)
            text += "'" + p.VoidReason + "', ";

//Takes away the last ", ' from the search string, which could cause some trouble
if (text.Substring(text.Length - 2, 2) == ", ")
            text = text.Remove(text.Length - 2, 2);

        text += ")";
```

```
return text;
        }//End GetInsertValues

publicvoid DeletePatient(Patient p)
        {
//Since the patient is deleted, the object is voided.
            p.Voided = true;

//Creates the link to the DB.
            SqlCeCommand command =
SqlCeSingleton.GetInstance().CreateCommand();

//Updates the patient table with void-related information
            command.CommandText = "UPDATE patient SET " +
"voided = " + p.Voided +
", voided_by = " + p.VoidedBy.Id +
", date_voided = " + p.DateVoided +
", void_reason = '" + p.VoidReason +
" WHERE id = " + p.Id + ";";
            command.ExecuteNonQuery();

//Insert the object in the history_patient table
            command.CommandText = "INSERT INTO history_patient(" +
"patient_id, " +
"patient_groupe_id " +
"creator, " +
"date_created, " +
"changed_by, " +
"date_changed, " +
"voided, " +
                                "voided_by, " +
"date_voided, " +
                                "void_reason) " +
"SELECT * " +
"FROM users " +
"WHERE id = " + p.Id + ";";
            command.ExecuteNonQuery();

            //Updating the patient_identifier table with void-related information
foreach (PatientIdentifier pi in p.Identifier)
            {
                command.CommandText = "UPDATE patient_identifier SET" +
" voided = " + p.Voided +
", voided_by = " + p.VoidedBy.Id +
                                    ", date_voided = " + p.DateVoided +
```

```
", void_reason = '" + p.VoidReason +
                                " WHERE patient_id = " + p.Id + ";";
            command.ExecuteNonQuery();
        }


//Inserting each p.Identifier-object into the history_patient_identifier
foreach (PatientIdentifier pi in p.Identifier)
        {
            command.CommandText = "INSERT INTO history_patient_identifier ("
+
"identifier, " +
"identifier_type, " +
"location_id, " +
"timpestamp, " +
"check_digit, " +
"preferred, " +
"patient_id, " +
"creator, " +
"date_created, " +
"voided, " +
"voided_by, " +
"date_voided, " +
"void_reason" +
") " +
"SELECT * " +
"FROM patient_identifier " +
"WHERE patient_id = " + p.Id + ";";

            command.ExecuteNonQuery();
        }

//Deleteting tuples in patient_identifier table related to this patient
foreach (PatientIdentifier pi in p.Identifier)
        {
            command.CommandText = "DELETE " +
"FROM patient_identifier " +
"WHERE patient_id = " + p.Id + ";";
command.ExecuteNonQuery();
        }

        //Deleteting the patient from the patient-table:
        command.CommandText = "DELETE FROM patient WHERE id = " + p.Id
+ ";";
        command.ExecuteNonQuery();
      } //End DeletePatient
```

```
#region PatientDAO Members

public Patient FindPatientById(Guid id)
        {
            Patient pat = new Patient();
            pat.Id = id;
            ICollection<Patient> patList = FindPatient(pat);
foreach (Patient p in patList)
                pat = p;

return pat;
        }//End FindPatientById

private Patient PersonToPatient(Person pe)
        {

            Patient pat = new Patient();

            pat.Id = pe.Id;
            pat.Addresses = pe.Addresses;
            pat.BirthDate = pe.BirthDate;
            pat.BirthDateEstimated = pe.BirthDateEstimated;
            pat.BirthPlace = pe.BirthPlace;
            pat.CauseOfDeath = pe.CauseOfDeath;
            pat.Citizenship = pe.Citizenship;
            pat.DeathDate = pe.DeathDate;
            pat.Gender = pe.Gender;
            pat.HealthCenter = pe.HealthCenter;
            pat.HealthDistrict = pe.HealthDistrict;
            pat.MothersName = pe.MothersName;
pat.Names = pe.Names;
            pat.Race = pe.Race;

return pat;
        }//End PersonToPatient

private Person PatientToPerson(Patient p)
        {
            Person per = new Person();

            per.Id = p.Id;
            per.Addresses = p.Addresses;
            per.BirthDate = p.BirthDate;
            per.BirthDateEstimated = p.BirthDateEstimated;
            per.BirthPlace = p.BirthPlace;
```

```
            per.CauseOfDeath = p.CauseOfDeath;
            per.Citizenship = p.Citizenship;
            per.DeathDate = p.DeathDate;
            per.Gender = p.Gender;
            per.HealthCenter = p.HealthCenter;
            per.HealthDistrict = p.HealthDistrict;
            per.MothersName = p.MothersName;
            per.Names = p.Names;
            per.Race = p.Race;

return per;

        }//End PatientToPerson

        #endregion
    }
}
```

Appendiks G – SQL kode: person, person_address og person_name

```
CREATE TABLE person (
  id UNIQUEIDENTIFIER DEFAULT NewID(),
  gender NVARCHAR(50),
  race NVARCHAR(50),
  birthdate DATETIME,
  birthdate_estimated SMALLINT,
  birthplace NVARCHAR(50),
  citizenship NVARCHAR(50),
  mothers_name NVARCHAR(50),
  death_date DATETIME,
  cause_of_death NVARCHAR(255),
  health_district NVARCHAR(255),
  health_center INT,
  creator UNIQUEIDENTIFIER REFERENCES person(id),
  date_created DATETIME,
  changed_by UNIQUEIDENTIFIER REFERENCES person(id),
  date_changed DATETIME,
  voided SMALLINT,
  voided_by UNIQUEIDENTIFIER REFERENCES person(id),
  date_voided DATETIME,
  void_reason NVARCHAR(255),
  PRIMARY KEY(id)
);

CREATE TABLE person_address (
  id UNIQUEIDENTIFIER DEFAULT NewID(),
  person_id UNIQUEIDENTIFIER REFERENCES person(id),
  preferred SMALLINT,
  address1 NVARCHAR(50),
  address2 NVARCHAR(50),
  city_village NVARCHAR(50),
  state_province NVARCHAR(50),
  postal_code NVARCHAR(50),
  country NVARCHAR(50),
  latitude NVARCHAR(50),
  longitude NVARCHAR(50),
  creator UNIQUEIDENTIFIER REFERENCES person(id),
  date_created DATETIME,
  voided SMALLINT,
  voided_by UNIQUEIDENTIFIER REFERENCES person(id),
  date_voided DATETIME,
  void_reason NVARCHAR(255),
  neighborhood_cell NVARCHAR(50),
  county_district NVARCHAR(50),
  PRIMARY KEY(id)
);




CREATE TABLE person_name (
  id UNIQUEIDENTIFIER DEFAULT NewID(),
  person_id UNIQUEIDENTIFIER REFERENCES person(id),
  preferred SMALLINT,
  prefix NVARCHAR(50),
  given_name NVARCHAR(50),
```

```
  middle_name NVARCHAR(50),
  family_name_prefix NVARCHAR(50),
  family_name NVARCHAR(50),
  family_name2 NVARCHAR(50),
  family_name_suffix NVARCHAR(50),
  degree NVARCHAR(50),
  creator UNIQUEIDENTIFIER REFERENCES person(id),
  date_created DATETIME,
  voided SMALLINT,
  voided_by UNIQUEIDENTIFIER REFERENCES person(id),
  date_voided DATETIME,
  void_reason NVARCHAR(255),
PRIMARY KEY(id)
);
```