Implementering og evaluering av en HL7 FHIR integrasjonsbuss for kommunikasjon mellom klientapplikasjoner og e-helsesystemer

Karina Øverland Haugen

Masteroppgave i programutvikling ved

Institutt for data- og realfag, Høgskulen på Vestlandet

Institutt for informatikk, Universitetet i Bergen

Veileder

Yngve Lamo

Institutt for data- og realfag, Høgskulen på Vestlandet

Juni 2017

Abstrakt

Bakgrunn: Den økende digitaliseringen av helsetjenestene krever at informasjonen i de elektroniske pasientjournalene er klinisk strukturerte ved hjelp av datastandarder og kliniske terminologier for riktig prosessering av pasientinformasjonen. Pasientjournalene må også være tilgjengelige når behovet er der. Dagens mangel på en felles standard for datastruktur og datautveksling medfører at systemene og applikasjonene har problemer med å samhandle og dele viktig pasientinformasjon, noe som påvirker effektiviteten hos helsevesenet.
Mål: Dette forskningsprosjektet vil evaluere og vurdere gevinstene med en tjenestearkitektur som bruker en kjent datautvekslingsstandard, HL7 FHIR, som en integrasjonsbuss mellom klientapplikasjoner og bakenforliggende e-helsesystemer.
Metoder: Vi implementerte en mindre tjenestearkitektur som består av en FHIR integrasjonsbuss som kommuniserer med et openEHR basert pasientjournalsystem. Implementasjonen er vurdert ved hjelp av medisinapplikasjonen LEMI som er integrert mot FHIR bussen for henting av pasientinformasjon, medisininformasjon og ordineringsinformasjon.
Resultater: FHIR bussen tilbyr et grensesnitt basert på REST spesifikasjonen som abstraherer den komplekse dokumentstrukturen til openEHR. Forespørslene sendt til FHIR bussen krever ingen domenekunnskap om tjenestene som FHIR bussen bruker som datakilder. Ved hjelp av profilering av FHIR ressursene kunne vi konvertere de obligatoriske datapunktene i openEHR arketypene over til tilhørende datapunkter i FHIR ressurser uten store problemer.
Konklusjon: FHIR er strukturert for å forbedre data- og meldingsutveksling mellom applikasjoner og systemer, og vår tjenestearkitektur med en FHIR integrasjonsbuss mellom medisinapplikasjonen LEMI og openEHR pasientjournalsystemet demonstrerte de store integrasjonsgevinstene som FHIR tilbyr. Vårt openEHR pasientjournalsystem demonstrerte hvordan FHIR kan brukes for å abstrahere domene- og implementasjonskunnskapen som trengs for å kunne prosessere openEHR informasjonen og utføre openEHR spørringer. FHIR bussen gir også tilgang til klinisk strukturerte testdata som forhindrer at utviklerne etablerer sine egne datastrukturer som ikke er kompatible med eksisterende e-helsesystemer.

Forord

Jeg vil takke min veileder Yngve Lamo for sin tålmodighet, sin smittende motivasjon for arbeidet, og sine gode råd og veiledning gjennom hele masterprosjektet. En takk til de involverte i Intromat-prosjektet hos Helse Bergen som har flere ganger kommet med innspill underveis. Jeg vil også takke studentene på masterlesesalen som har vært med på å gjøre dagene på skolen mye lettere. En ekstra stor takk til min fantastiske familie som hele tiden har støttet meg og som alltid har hatt troen på arbeidet mitt. En spesiell takk til min forlovede Ruben, som har holdt ut med meg gjennom alle disse årene og som alltid har motivert meg til å fortsette å jobbe hardt. Uten dere rundt meg ville jeg aldri vært der jeg er i dag.

Dedikert til min tapre mamma, Wenche,
som tapte kampen mot kreft januar 2016,
og som har vært min store drivkraft gjennom hele masterprosjektet

Innhold

Forkortelser **vii**

Figurer **viii**

Listings **xi**

Tabeller **xiii**

1 Introduksjon **1**

1.1 Motivasjon 1

1.2 Mål og problemstilling 5

1.3 Oppgavestruktur 5

2 Bakgrunn **7**

2.1 Integrasjon 8

2.1.1 Integrasjonsutfordringer i elektroniske pasientjournaler 9

2.1.2 Hvordan kan utfordringene løses 10

2.1.3 Semantiske integrasjonsproblemer 11

2.2 Standarder innenfor helseinformasjonsteknologi 12

2.2.1 Klinisk terminologi 13

2.3 openEHR 15

2.3.1 EHR strukturen 15

2.3.2 Arketype og mal 17

2.3.3 Template Designer - verktøy for utvikling av maler 19

2.3.4 To-nivå modellering 21

2.3.5 Archetype Query Language (AQL) 22

2.3.6 openEHR sin rolle i Norge 23

2.4 HL7 FHIR 23

2.4.1 Struktur - FHIR ressurs . . . 25
2.4.2 Profilering av ressurser . . . 26
2.4.3 Forge - profileringsvertøy . . . 28
2.4.4 FHIR sin rolle i Norge . . . 29
2.5 Bruk av FHIR til integrasjon . . . 30
2.5.1 HL7 FHIR og OpenMRS . . . 30
2.5.2 HL7 FHIR og i2b2 . . . 31
2.5.3 HL7 FHIR og HL7 CDA . . . 31
2.5.4 HL7 FHIR og OMOP CDM . . . 32

3 Referanseimplementasjon 33
3.1 Brukerhistorie . . . 34
3.2 Mål for systemet . . . 35
3.2.1 FHIR buss . . . 35
3.2.2 openEHR journalsystem . . . 36
3.3 Funksjonelle krav . . . 36
3.4 Ikke-funksjonelle krav . . . 38

4 Systemarkitektur og implementasjon 39
4.1 Utviklingsverktøy og teknologi . . . 40
4.1.1 HAPI FHIR . . . 41
4.1.2 EtherCIS . . . 41
4.1.3 Docker . . . 42
4.1.4 MediCloud og IBM Bluemix . . . 42
4.1.5 Node.js og Express . . . 43
4.2 Systemarkitektur . . . 43
4.3 Sikkerhetshåndtering . . . 45

5 Konverteringsprosessen 47
5.1 Arbeidsflyten for konverteringsprosessen . . . 49
5.2 Representasjon av observasjonskonsepter . . . 49
5.2.1 openEHR OBSERVATION . . . 50
5.2.2 FHIR Observation . . . 52
5.3 Representasjon av evalueringskonsepter . . . 54
5.3.1 openEHR EVALUATION . . . 55
5.3.2 FHIR Evaluation . . . 56
5.4 Kartlegging av likheter . . . 57

5.4.1 Blodtrykk . . . 57
5.4.2 Allergi . . . 58
5.4.3 Diagnose . . . 58
5.5 Profilering av FHIR ressurser . . . 59
5.5.1 Profilering av Observation . . . 59
5.5.2 Profilering av AllergyIntolerance og Condition(Problem) . . . 60
5.6 Oppretting av openEHR maler . . . 61
5.6.1 Oppretting av mal for arketypen Blodtrykk . . . 61
5.6.2 Oppretting av maler for arketypene «Adverse reaction risk» og «Problem/Diagnosis» . . . 62
5.7 Strukturering av konverteringfil . . . 64
5.7.1 Konverteringsfil for blodtrykk . . . 65
5.7.2 Konverteringsfil for allergi . . . 66
5.7.3 Konverteringsfil for diagnose . . . 66

6 Integrering av medisinapplikasjonen LEMI 69
6.1 LEMI - LEgeMIddelapp . . . 69
6.2 Integrere LEMI mot FHIR bussen . . . 72
6.2.1 Funksjonalitet: Hente medisinliste for en pasient . . . 74
6.2.2 Funksjonalitet: Oppdatere medisinstatus . . . 78

7 Resultat og Evaluering 85
7.1 Resultater . . . 86
7.2 Vurdering . . . 89
7.2.1 Vurdering av FHIR integrasjonsbuss . . . 89
7.2.2 Funksjonelle krav . . . 91
7.2.3 Ikke-funksjonelle krav . . . 95
7.3 Tilgjengelig funksjonalitet i FHIR bussen . . . 97
7.4 Diskusjon . . . 100
7.4.1 Hva er fordelene med en tjenestearkitektur som har et FHIR grensesnitt mellom klientapplikasjonene og de bakenforliggende systemene? . 100
7.4.2 Hvordan kan en skyløsning være med på å forbedre integrasjonen mellom klienter og systemer? . . . 104
7.4.3 Relatert arbeid . . . 105
7.5 Begrensninger . . . 107

8 Konklusjon 109

8.1 Videre forskning 110
8.1.1 Utvide FHIR med partisjoner 111
8.1.2 Konvertere flere arketyper og ressurser 111
8.1.3 Sammenligne antall lignende datapunkter 111
8.1.4 Sikkerhet 112
8.1.5 Automatisk AQL oppretting 112

Bibliografi **113**

A Bruk av verktøyet Forge **119**

B Detaljerte modeller og diagrammer **121**

C Vurderingsbidrag **133**

D FHIR integrasjonsbuss dokumentasjon **139**

Forkortelser

ADL Archetype Definition Language.

API Application Programming Interface.

AQL Archetype Query Language.

B2B Business-to-Business.

B2C Business-to-Client.

CDA Clinical Document Architecture.

CDM Common Data Model.

CIMI Clinical Information Modeling Initiative.

DSTU Draft Standard for Trial Use.

EHR Electronic Health Record.

EHR-S Electronic Heath Record System.

EPJ Elektronisk pasientjournal.

FHIR Fast Healthcare Interoperability Resources.

HAPI HL7 Application Programming Interface.

HIMSS Healthcare Information and Management Systems Society.

HL7 Health Level Seven.

HTML HyperText Markup Language.

HTTP The Hypertext Transfer Protocol.

i2b2 Informatics for Integrating Biology and the Bedside.

ICD International Classification of Diseases.

IHE Integrating the Healthcare Enterprise.

IHTSDO International Health Terminology Standards Development Organisation.

ISO International Organization for Standardization.

JSON JavaScript Object Notation.

LOINC Logical Observation Identifiers Names and Codes.

OMOP The Observational Medical Outcomes Partnership.

RDF Resource Description Framework.

REST Representational state transfer.

SNOMED CT Systematized Nomenclature of Medicine - Clinical Terms.

SQL Structured Query Language.

STU Standard for Trial Use.

URI Uniform Resource Identifier.

XDS Cross-Enterprise Document Sharing.

XHTML Extensible Hypertext Markup Language.

XML eXtensible Markup Language.

XPath XML Path Language.

Figurer

1.1 Kommunikasjon mellom interessenter og institusjoner 2
1.2 Et integrasjonsmiljø med en «FHIR buss» i sentrum 4

2.1 SNOMED CT eksempel 14
2.2 Hvordan LOINC og SNOMED kan brukes i lag 15
2.3 openEHR journalstruktur 16
2.4 openEHR COMPOSITION struktur med obligatoriske felter 18
2.5 Konseptmodell - Mal for rapportering av blodtrykk 20
2.6 Template Designer - Mal for rapportering av blodtrykk 20
2.7 To-nivå modellering 21
2.8 FHIR ressurs struktur 25
2.9 FHIR kardinalitet regler 26
2.10 Profilering av FHIR ressursen Observation for blodtrykk 28
2.11 Forge - Verktøy for profilering av ressurs 29

3.1 En medisinrunde for sykepleiere 34

4.1 Høynivå av vår tjenestearkitektur 39
4.2 Teknologiarkitektur 40
4.3 Systemarkitektur 44

5.1 Modell over de ulike dataformatene i bruk i vår arkitektur 48
5.2 Oppretting av nytt endepunkt i FHIR bussen for ny arketype 50
5.3 Grafisk representasjon av arketypen Blodtrykk 52
5.4 Forenklet FHIR Observation struktur 53
5.5 Oversikt over EVALUTATION struktur 55
5.6 FHIR evalueringsressurser 56
5.7 «Discriminator» og SNOMED kode spesifisert i verktøyet Forge 60

5.8 Profilering av AllergyIntolerance og Condition 61
5.9 Utsnitt av Template Designer for å opprette en mal for blodtrykk 62
5.10 Mal utviklet for arketypene «Risiko for overfølsomhetsreaksjon» og «Problem/Diagnosis» 63

6.1 Utsnitt av pasientliste fra medisinapplikasjonen LEMI 70
6.2 LEMI - utsnitt fra forberedelse og skanning av pasientarmbånd 71
6.3 LEMI - Pasientliste hvor pasienten Roland har fått medisinene han skal ha . 72
6.4 Systemarkitektur med openEHR pasientjournalsystem, FHIR integrasjonsbuss og LEMI medisinapplikasjon 73
6.5 openEHR mal og FHIR profil for ordineringer 75
6.6 openEHR terminologikoder for instruksjonsstatus 79
6.7 Prosessmodell av applikasjonen LEMI 81
6.8 Oppdatering av medisinstatus til status lik 1 81
6.9 Oppdatering av medisinstatus til status lik 2 82
6.10 Oppdatering av medisinstatus til status lik 3 82
6.11 openEHR mal av arketypen *Medication action* 83

7.1 Utsnitt av representasjon av endepunkt for å opprette en pasientjournal. . . . 92
7.2 Utsnitt fra Postman for å opprette en ny mal i openEHR pasientjournalsystemet 93
7.3 Intern arkitektur av FHIR bussen 94
7.4 Strukturering av FHIR buss for vedlikeholdbarhet 96
7.5 Oppretting av AQL spørring uten og med en FHIR integrasjonsbuss 103

A.1 Blodtrykk profil i Forge 119

B.1 Oppdatering for å tillate flere felter i arketype og ressurs 121
B.2 Full FHIR Observation struktur i UML diagram 122
B.3 Komplett openEHR adverse reaction risk arketype og FHIR AllergyIntolerance ressurs 123
B.4 Komplett openEHR Family history arketype og FHIR FamilyMemberHistory ressurs 124
B.5 Komplett openEHR Goal arketype og FHIR Goal ressurs 125
B.6 Komplett openEHR Problem/Diagnosis arketype og FHIR Condition (Problem) ressurs 126
B.7 Komplett openEHR Health risk assessment arketype og FHIR RiskAssessment ressurs 127

B.8 Beslutningstre for konvertering fra openEHR til FHIR 128
B.9 Beslutningstre for konvertering fra FHIR til openEHR 129

C.1 Utsnitt fra Postman for å opprette en ny komposisjon i openEHR pasient-journalsystemet 135
C.2 Utsnitt fra Postman for å hente en eksisterende komposisjon fra openEHR pasientjournalsystemet 135
C.3 FHIR MedicationOrder DSTU2 versjon 136
C.4 FHIR MedicationRequest STU3 versjon 137

Listings

2.1 AQL eksempel BMI verdier 23
5.1 Blodtrykk arketype i XML format 51
5.2 Utsnitt fra observasjonsressurs for blodtrykk 54
5.3 En tom konverteringsfil 64
5.4 Utsnitt av konverteringsfilen for blodtrykk mellom FHIR og openEHR 65
5.5 Utsnitt av konverteringsfilen for ressursen AllergyIntolerance og arketypen Adverse reaction risk 66
5.6 Utsnitt av konverteringsfilen for ressursen Condition(Problem) og arketypen Problem/Diagnosis 67
6.1 Medisinliste representert som FHIR Bundle med en liste med MedicationStatement ressurser 76
6.2 AQL spørring etter aktiv medisinliste 78
6.3 Resultat av AQL spørring etter medisinlisten for pasient 78
6.4 Utsnitt fra ADL representasjon av arketypen Medication action for prosesstrinn brukt i LEMI 80
B.1 Konverteringsfil for arketypen Legemiddelordinering 130
C.1 GET respons med unik identifikator for FHIR ressursen Medication 133
C.2 Utsnitt fra kildekoden til klassen MainServlet fra FHIR bussen 134
C.3 Eksempel for hvordan en ny «provider» for en ressurs opprettes i FHIR bussen. 134

Tabeller

2.1 Pakkene i referansemodellen til openEHR 22

5.1 FHIR evalueringsressurser med tilhørende openEHR arketyper 57

7.1 Kort sammenligning openEHR og FHIR 87
7.2 Liste med tilgjengelige endepunkter i FHIR bussen som bruker intern Cloudant database 98
7.3 Liste med tilgjengelige endepunkter i FHIR bussen som bruker openEHR pasientjournalsystemet 99
7.4 Sammenligning av masterprosjektet med relatert arbeid 106

Kapittel 1

Introduksjon

1.1 Motivasjon

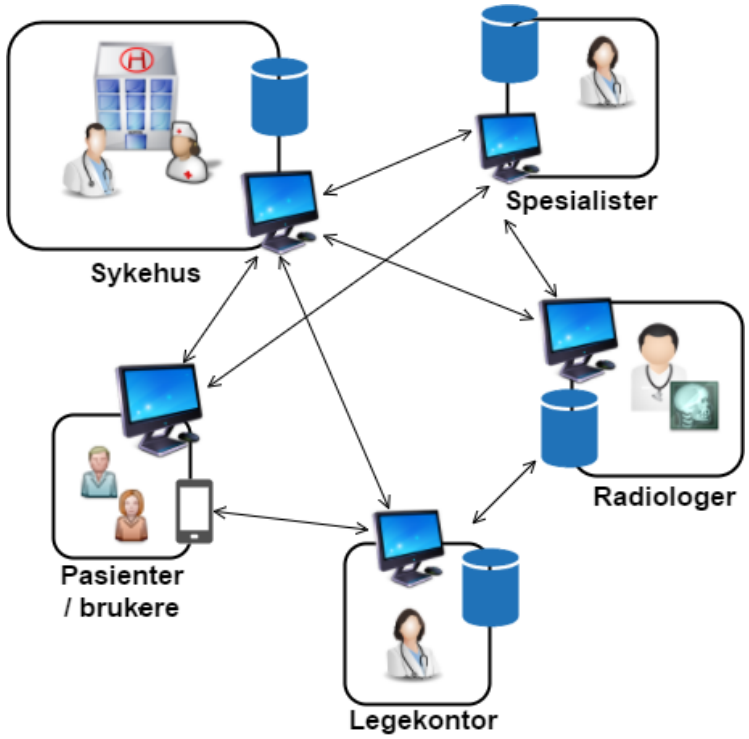
Som en konsekvens av den økende digitaliseringen av helsetjenestene øker mengden av tilgjengelig elektronisk helsedata jevnlig [1]. De kliniske dataene er lagret i elektroniske pasientjournaler, som er en digital versjon av pasientens papirjournal. Ved bruk av den elektroniske pasientjournalen får en tilgang til pasientens demografi, allergier, medikasjon, testresultater fra laboratoriet, tidligere behandlinger og andre kliniske data. Det må være mulig å forstå informasjonen uansett hvilken klinisk institusjon som får tilgang til journalen. Elektroniske pasientjournaler kan ha en positiv påvirkning på kvaliteten av behandlingen, pasientsikkerheten og kan gi bedre effektivitet hos helsevesenet [2–4]. For å oppnå disse gevinstene er det ikke nok å bare lagre dataene elektronisk, men det krever at pasientjournalen inneholder nøyaktige og relevante data i et brukbart format som det er mulig å få tilgang til. Det er viktig at andre aktører enn pasientens fastlegekontor, for eksempel en spesialist som er involvert i pasientens behandling, får tilgang til journalen når behovet er der. For å oppnå dette må standarder for datastrukturering og datautveksling bli utviklet og implementert. En standard kan bli definert i ulike formater, men hovedsakelig inneholder en standard et sett med regler og definisjoner for hvordan en skal utføre en prosess eller strukturere ulike dataelementer. Å ha felles standarder gir muligheten for to eller flere aktører å samarbeide [2]. For å kunne gi pasientene rett diagnose, rett behandling og råd er sykepleierne og legene avhengige av å ha informasjonssystemer tilgjengelig som lar de hente ut relevant informasjon om pasientene.

Dessverre ser vi i dag at helsetjenestene i Norge og i andre land består av flere systemer

som har problemer med å samhandle og dele på informasjonen. Dette er blant annet fordi systemene mangler god datastruktur og felles standard for datautveksling. Flere systemer i dag bruker egne tjenestestrukturer for deling av data og disse er ofte vanskelige å få tilgang til. Dette forhindrer helsevesenet i å gi pasientene best mulig behandling [5]. Som en respons til dette har det blitt utviklet ulike internasjonale integrasjonsstandarder. En gruppe av standardene kommer fra organisasjonen Health Level Seven (HL7) International som har standarder som er i stor grad både akseptert og brukt i dag [6].

På grunn av manglende felles standard for hvordan dokumenter, databaser, objekter og tjenester skal struktureres må hver integrasjon mellom systemene implementeres hver for seg. Dette krever både mye tid og penger. Jo flere systemer som eksisterer i dag, jo flere integrasjoner må bli gjennomført. Siden det ikke er brukt offentlige tjenestestandarder må hver institusjon selv implementere hver integrasjon. Dessverre ser vi også at systemene internt i en institusjon også har problemer med å dele pasientinformasjonen mellom seg.

I figur 1.1 har vi ulike interessenter og institusjoner som ønsker å kommunisere; sykehus, pasienter/brukere, spesialister, radiologer og legekontor.

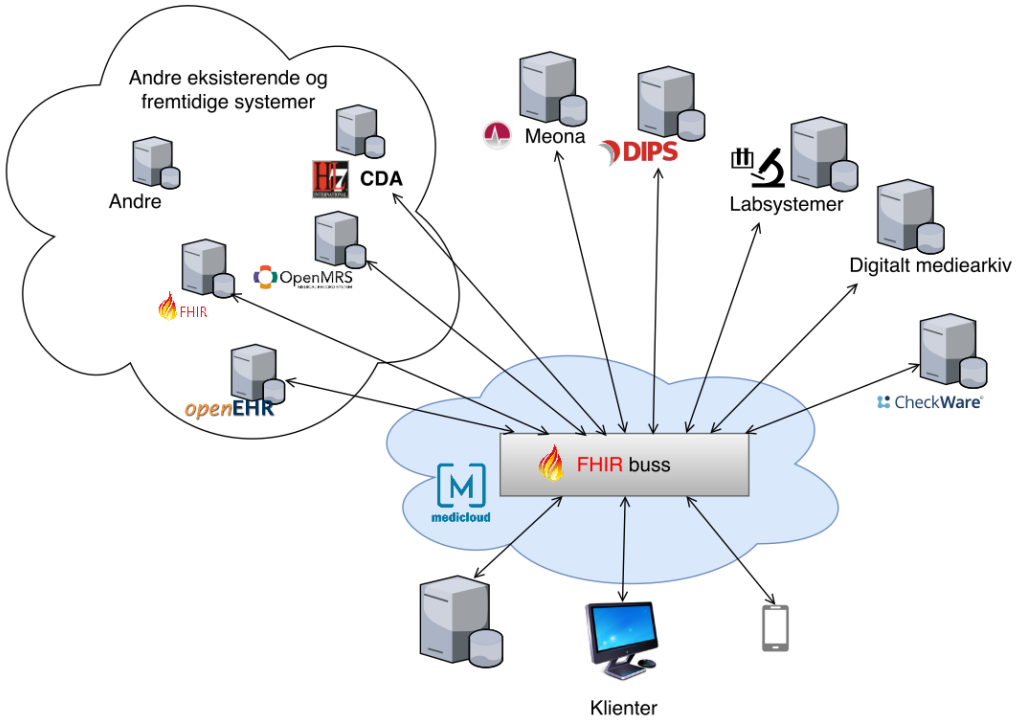


Figur 1.1: Figuren viser et forenklet bilde av kommunikasjon mellom interessenter og institusjoner. Siden det ikke eksisterer en felles kommunikasjonskanal må hver aktør ofte implementere en integrasjon mot de andre aktørene de ønsker å kommunisere med.

Som et eksempel kan en tenke seg at et sykehussystem ønsker å kommunisere med både ulike pasientapplikasjoner, legekontorsystemer, spesialistsystemer og systemer for håndtering av bilder hos radiologen. Slik systemene er bygget opp i dag må hver av disse integrasjonene gjøres hver for seg på grunn av ulike datastrukturer og ulike tjenester for datautveksling. Skulle det skje en endring i enten datastrukturen eller tjenestestrukturen som sykehussystemet tilbyr må alle involverte interessenter endre på sine integrasjoner mot sykehussystemet. Dette er ikke en skalerbar løsning når antall interessenter øker.

To kjente og populære standarder i dag er openEHR og HL7 FHIR. openEHR er en datastandard som fokuserer på hvordan de kliniske dataene skal struktureres, mens FHIR er en meldingsstandard som spesifiserer hvordan ulike meldinger og data kan deles via tjenester. openEHR er ikke den eneste populære standarden som spesifiserer hvordan dataene kan struktureres. Blant annet er både HL7 CDA (Clinical Document Architecture) for dokumentstruktur og OpenMRS for medisinske journalsystemer brukt i ulike e-helsesystemer. Systemer som er baserte på disse standardene krever at brukerne har domenekunnskap. På grunn av dette er det tidligere forsket på hvordan blant annet standardene HL7 CDA og OpenMRS kan bruke FHIR som et grensesnitt for å forenkle integrasjonen mot deres system for andre brukere.

I masterprosjektet blir det designet en tjenestearkitektur med en FHIR integrasjonsbuss mellom brukerne og de bakenforliggende e-helsesystemene. Denne integrasjonsbussen skal være med på å forbedre integrasjonen mot e-helsesystemer som har komplekse eller manglende offentlige datastrukturer. For å teste denne tjenestearkitekturen vil vi implementere en mindre arkitektur som består av en FHIR integrasjonsbuss som vil fungere som en meldingstjeneste mellom brukerne og et openEHR basert pasientjournalsystem. Denne løsning vil kombinere det beste ved openEHR og det beste ved FHIR. FHIR bussen vil gjøre det mulig for andre systemer kompatible med FHIR å både lagre og hente ut informasjon fra pasientjournalsystemet. Å konvertere mellom openEHR og FHIR vil gjøre det mulig å evaluere effekten ved å bruke FHIR som en «buss-tjeneste» mellom andre bakenforliggende systemer og klientapplikasjoner for å forbedre og forenkle integrasjonen mellom de aktuelle aktørene. En slik buss-tjeneste vil være tilgjengelig via en skyløsning og vil tilby ulike tjenester. Disse tjenestene kan tenkes å bli brukt av både pasientapplikasjoner, forskningsbaserte systemer, og andre e-helsesystemer som mangler tjenester for å hente og lagre kliniske data. Integrasjonen mellom openEHR og FHIR bussen er bare en av flere integrasjoner som kan utføres via FHIR bussen. En FHIR buss kan være en kanal for flere systemer som er baserte på ulike standarder. Figur 1.2 viser en enkel modell av arkitekturen for hvordan et slikt miljø kan se ut. På en side har vi klienter som kjører på ulike enheter og kan være både



Figur 1.2: Figuren gir et generelt bilde over hvordan et tenkt integrasjonsmiljø med en «FHIR integrasjonsbuss» i sentrum kan se ut. Alle brukerne av de bakenforliggende e-helsesystemene kommuniserer med FHIR bussen, som videre kommuniserer med de aktuelle systemene.

applikasjoner og forskningsbaserte systemer. På den andre siden kan vi finne systemene som Meona [7] for elektronisk kurve og legemiddelhåndtering, DIPS [8] for elektroniske pasientjournaler, laboratoriesystemer, digitale mediearkiv for lagring av bilder og CheckWare [9] for selvrapportering av psykologiske data for pasienter. Det kan også være andre bakenforliggende systemer som ønsker å tilby sine tjenester gjennom FHIR bussen. Antall systemer vil antageligvis øke de neste årene og det kan tenkes at flere av systemene er baserte på en av standardene HL7 CDA, OpenMRS, openEHR eller HL7 FHIR. Selv om FHIR er en meldingsstandard er det mulig å utvikle systemer som baserer seg på standarden og dermed er også FHIR inkludert som et system. Disse standardene er bare noen få av mange standarder som eksisterer i dag, og det vil nok også komme flere nye standarder i fremtiden.

Vår FHIR integrasjonsbuss vil være kjørende i skytjenesten MediCloud som har ulike tjenester tilgjengelig. MediCloud er en dedikert versjon av skytjenesten IBM Bluemix og styres av Sykehuspartner HF [10]. En tjeneste tilgjengelig i MediCloud kan være å hente ut informasjon om en pasient. Hvor informasjonen kommer fra vil være konfigurert i FHIR bussen.

Klientene trenger da bare å integrere seg mot FHIR bussen og bruke tjenesten for å hente ut informasjonen. En fordel med en slik arkitektur er at klientene av FHIR bussen er skjermet for endringer hos systemene som FHIR bussen bruker som datakilder. Dersom det skulle skje en endring i for eksempel journalsystemet DIPS er det kun FHIR bussen som må oppdateres. Klientene er kun avhengige av tjenestene som FHIR bussen tilbyr og dermed slipper de å oppdatere seg.

1.2 Mål og problemstilling

Det overordnede målet i masterprosjektet er å utforske hvordan en kan forbedre og forenkle integrasjonen mot bakenforliggende e-helsesystemer ved å tilføre et ekstra grensesnitt mellom brukerne og de systemene. Dette evalueres ved å strukturere en arkitektur med en «integrasjonsbuss» basert på meldingsstandarden HL7 FHIR, som vil kommunisere med et pasientjournalsystem basert på datastandarden openEHR. HL7 FHIR spesifikasjonen er en spesifikasjon for et REST basert API. For å kunne evaluere i hvor stor grad dataene kan konverteres mellom de to ulike standardene vil en medisinapplikasjon modifiseres for å hente sine data fra et openEHR pasientjournalsystem gjennom FHIR bussen. For å kunne oppnå målet blir det gjennom masterprosjektet gjort et forsøk på å svare på følgende forskningsspørsmål:

1. Hva er fordelene med en tjenestearkitektur som har et FHIR grensesnitt mellom klientapplikasjonene og de bakenforliggende systemene?
   (a) Hvilke gevinster har en FHIR buss for et openEHR pasientjournalsystem?
   (b) Hvilke modeller er openEHR og FHIR bygget opp av, og lar det seg gjøre å konvertere mellom openEHR arketyper og FHIR ressurser?
2. Hvordan kan en skyløsning være med på å forbedre integrasjonen mellom klienter og systemer?

1.3 Oppgavestruktur

Denne oppgaven er delt opp i følgende kapitler:

1. **Introduksjon** - Gir en innføring til problemstillingen, motivasjonen og forskningspørsmålene for masterprosjektet.

2. **Bakgrunn** - Gir en oversikt over hva integrasjon er og introduserer ulike standarder som er aktuelle for masterprosjektet. I tillegg vises det til ulik forskning som er gjort innenfor område.

3. **Referanseimplementasjon** - Beskriver en brukerhistorie hvor en sykepleier går en medisinrunde for pasienter på et sykehus. I tillegg beskrives et hovedmål og flere delmål for arbeidet i masterprosjektet. Utifra brukerhistorien og målene er det utviklet en liste med funksjonelle og ikke-funksjonelle krav.

4. **Systemarkitektur og implementasjon** - Gir en oversikt over hvordan FHIR bussen er utviklet og strukturert. Diskuterer hvilke verktøy og teknologi som blir brukt for å tilfredstille målene og kravene fra kapittel 3.

5. **Konverteringsprosessen** - Beskriver hvordan arketypene «Blodtrykk», «Risiko for overfølsomhetsreaksjon» og «Problem/Diagnosis» kan bli konvertert over til ressursene FHIR Observation, FHIR AllergyIntolerance og FHIR Condition henholdsvis.

6. **Integrering av medisinapplikasjonen LEMI** - Beskriver hvordan medisinapplikasjonen LEMI ble modifisert og integrert mot vår FHIR integrasjonsbuss for å hente pasienter og medisinlister.

7. **Resultat og Evaluering** - Diskuterer våre resultater fra masterprosjektet og beskriver hvordan disse kan besvare våre forskningspørsmål. Gir også en evaluering av hvordan arbeidet er utført basert på de funksjonelle og ikke-funksjonelle kravene som er spesifisert i kapittel 3.

8. **Konklusjon** - Oppsummerer våre resultater og erfaringer oppnådd gjennom masterprosjektet. Diskuterer også forbedringer og videre forskningsarbeid som kan gjøres innenfor forskningsområdet.

Kapittel 2

Bakgrunn

Før vi starter med presentasjonen av vårt forskningsbidrag gis det i dette kapittelet en introduksjon til de viktigste emnene som masterprosjektet berører. Først skal vi se på hva integrasjon er, hvilke utfordringer som eksisterer og hvordan disse kan løses. Deretter skal vi se på ulike helsestandarder vi har i dag, hvor vi først presenterer hva en terminologistandard er og hvorfor de er viktige for klinisk korrekt tolkning av helsedataene. Vi går i dybden på standardene openEHR for strukturering av data og HL7 FHIR for spesifikasjon av hvordan data og meldinger kan deles. Til slutt skal vi se på tidligere forskningsprosjekter som ser på bruken av et FHIR grensesnitt over eksisterende e-helsesystemer.

2.1 Integrasjon

Selve prosessen å kunne sende helserelaterte data mellom systemer blir ofte kalt for sømløs deling av kliniske data eller *integrasjon*. Når denne delingen blir utført problemfritt mellom systemene sier vi at systemene er godt integrerte mot hverandre. For å få en god forståelse om hva integrasjon betyr og hvorfor vi trenger integrasjon mellom systemene kan vi starte med å forstå hva en elektronisk journal faktisk er. International Organization for Standardization (ISO) har en egen definisjon [11]:

> «The basic-generic definition for the EHR (Electronic Health Record) is a repository of information regarding the health status of a subject of care, in computer processable form»

Som definisjonen forklarer skal innholdet i en elektronisk pasientjournal være strukturert slik at det er mulig å prosessere informasjonen. I tillegg til å kunne prosessere informasjonen bør den også være tilgjengelig gjennom hele livsløpet til pasienten. I Norge skiller en mellom informasjon som er lagret i en pasientjournal og informasjon lagret i et register. Informasjonen lagret i et register blir brukt til blant annet forskning, læring og forbedret behandling av pasientene [12]. Det er viktig at helsepersonellet får tilgang til den nødvendige pasientinformasjonen når behovet er der og for å få til dette kreves det integrasjon.

Det finnes flere definisjoner for hva interoperabilitet er og hvilke nivå som eksisterer. HIMSS (Healthcare Information and Management Systems Society) publiserte i 2013 en artikkel for hva interoperabilitet er og beskrev tre ulike nivåer [13]. Det første nivået er *grunnleggende integrasjon* som innebærer at informasjonen kan bli sendt fra et system til et annet, men garanterer ikke at det mottakende systemet kan tolke dataene. Det andre nivået er *syntaktisk integrasjon*. Dersom systemene har syntaktisk integrasjon har de en mulighet å kommunisere og dele strukturerte data mellom seg. Informasjonen kan bli lest og tolket, men garanterer ikke klinisk korrekthet. Syntaktisk integrasjon krever definerte datastrukturer og felles kommunikasjonsprotokoller. Et eksempel på en datastruktur som kan bli brukt er XML. Det tredje nivået er *semantisk integrasjon* som gjør det mulig for systemene å dele informasjon som kan bli automatisk tolket av det mottakende systemet. Dette gjøres ved hjelp av eksisterende definerte domenekonsepter som systemene kan prosessere og tolke. Semantisk integrasjon krever at det brukes felles godkjente terminologier som gir informasjonen en mening og kan tolkes automatisk [14, 15].

I kapittel 2 i masteroppgaven «Enabling medical research on clinically collected data using openEHR archetypes» av Leykun Melkamu Gebeyehu [14] presenterte studenten en god

oversikt over hvilke utfordringer en møter på når en ønsker å integrere systemene med hverandre. I samme kapittel henviste studenten til ulike løsninger som kan være med på å løse integrasjonsproblemene. I følgende to avsnitt presenteres et utkast av de ulike problemene og løsningene som studenten beskrev i masteroppgaven.

2.1.1 Integrasjonsutfordringer i elektroniske pasientjournaler

Integrasjon mellom systemer er nødvendig for en sømløs deling av data, men det eksisterer fremdeles endel problemer som må løses før vi oppnår integrasjon. En av de større årsakene til mangel på integrasjon hos de større institusjonene er at det eksisterer ulike elektroniske enheter på avdelingene som kjører på ulike versjoner av en plattform. Disse enhetene kan være koblet mot ulike databaser som er ulikt strukturert og det eksisterer flere ulike programvareapplikasjoner for ulike formål. De kliniske dataene til pasientene er strukturerte etter en *informasjonsmodell*. Hvordan en bygger opp en informasjonsmodell kan deles opp i tre nivåer [14, 16]:

- **Fritekst:**
  Når en journal er bygget opp uten en struktur er dataene lagret som ustrukturert fritekst. Fordelen med denne tilnærmingen er at dersom det skjer endringer i de kliniske modellene trenger ikke journalsystemet å endre seg. Derimot er det vanskelig å kunne hente ut relevante data når det trengs.

- **Klasse eller databasetabell for hvert klinisk konsept:**
  Noen journalsystemer er implementert med en egen klasse for hvert eneste kliniske konsept. Dette medfører store databaseskjemaer og disse er vanskelig å vedlikeholde ved oppdateringer.

- **To-nivå modellering:**
  Tilnærmingen som baserer seg på to-nivå modellering (videre beskrevet i avsnitt 2.3.4) handler om å utvikle journalsystemer utifra en referansemodell (nivå 1) med generiske datatyper og konsepter. Dette medfører at en kan lagre hva som helst av kliniske data så lenge de er strukturert utifra referansemodellen. For å opprette objekter for kliniske konsepter opprettes *arketyper* (nivå 2) som bruker de ulike datatypene og konseptene fra referansemodellen.

Problemet i dag er at det eksisterer situasjoner der hver metode kan være god nok. Noen systemer har så lite funksjonalitet at fritekst er godt nok for et fungerende system, mens andre krever en referansemodell. Dette gjør at målet om å oppnå 100% integrasjon mellom

systemene kan være vanskelig å oppnå.

2.1.2 Hvordan kan utfordringene løses

Det eksisterer en generell enighet blant folket at standarder er løsningen [17]. Det bør eksistere en felles nasjonal standard som blir tatt i bruk, men målet til slutt bør være en internasjonal felles standard. Pasientene er i dag mer mobile og reiser mer enn før. Dermed er det viktig at systemene der pasientene er har tilgang til rett data til rett tid. I tillegg til standarder kan også åpen kildekode hjelpe. Prosjekter som blant annet HAPI FHIR [18] og openEHR [19] tilbyr biblioteker med åpen kildekode. Disse kan utviklerne ta i bruk for å gjøre det enklere å komme i gang med utviklingen av et system som baserer seg på en felles standard. I [14, 20] henvises det til fire forutsetninger som må være på plass før vi har semantisk integrasjon:

1. *Standardisert referansemodell brukt i den elektroniske journal.*
   Dette handler om å ha en generisk informasjonsstruktur for pasientjournalsystemet. Dette medfører at systemet ikke må oppdateres for hver gang en ønsker å støtte lagring av nye kliniske konsepter.

2. *Standardisert modell for grensesnittene mellom tjenestene.*
   Dette handler om å ha strukturerte grensesnitt som eksisterer mellom ulike elektroniske journaltjenester og andre tjenester som demografi, terminologi, og sikkerhetstjenester.

3. *Standardiserte domenespesifikke konseptmodeller.*
   Dette handler om å ha en formell modell som beskriver et domenekonsept (arketyper) og maler for ulike kliniske konsepter og brukstilfeller.

4. *Standardiserte terminologier.*
   Terminologier er språket som er brukt i arketypene for å gi innholdet en gitt betydning. Terminologiene inneholder koder som kan brukes for automatisk tolkning av informasjonen.

De første to forutsetningene kreves også for å ha syntaktisk integrasjon.

Det eksisterer flere standarder i dag som er brukt i ulike systemer og med årene vil det nok komme nye standarder og endringer av eksisterende standarder. Siden vi har så mange helsestandarder å velge mellom møter vi på nye integrasjonsproblemer som er en del av hva denne oppgaven addresserer; *hvordan oppnå integrasjon mellom de ulike standardene som*

eksisterer i dag.

2.1.3 Semantiske integrasjonsproblemer

Selv om vi utvikler ulike standarder for strukturering av data møter vi fremdeles på blant annet semantiske integrasjonsproblemer som må løses. I artiklene [21] og [22] diskuterte forfatterne flere konflinkter som oppstår ved en semantisk konvertering og i disse to artiklene utarbeidet forfatterne mulige løsninger på problemene. I [21] fokuserte forfatterne på navnkonfliktene ved semantisk integrasjon. En kan møte på navnkonflikter når en ønsker å konvertere mellom elementer utifra navnet på elementet. Årsakene til disse konfliktene er bruk av synonymer og homonymer. I noen tilfeller kan det være at to ulike navn blir brukt, men som betyr det samme (synonym). En automatisk semantisk integrasjon vil sannsynligvis ikke klare å finne likhetene mellom disse to elementene dersom det kun sammenlignes med navn. I andre tilfeller kan det være at samme navn blir brukt for ulike elementer (homonym), og dermed kan det være at to elementer blir tolket som like utifra navnet selv om det er to helt forskjellige elementer. En mulig løsning for dette problemet var å bruke en internettbasert ordbok. Denne ordboken kunne sammenligne to navn og returnerte et tall om hvor like disse navnene var. I [22] ble det videre forsket på konflikter på datanivå. Dette omhandler datatyper, dataformater, dataverdier og dataskaleringer.

- **Datatype** - Konflikt mellom datatyper kan oppstå dersom det ene systemet bruker «string» for å beskrive et tall, mens et annet system bruker «double».

- **Dataformat** - Konflikt mellom dataformater kan oppstå dersom det ene systemet bruker formatet «MMM dd yy» for datoer, mens et annet system bruker formatet «dd.MM.yy».

- **Dataverdi** - Konflikt mellom dataverdier kan oppstå dersom det ene systemet bruker ordet «male» for mann, mens et annet system bruker ordet «man».

- **Dataskalering** - Konflikt mellom dataskaleringer kan oppstå dersom det brukes ulike enheter for et konsept, for eksempel kilogram istedenfor pounds og fahrenheit istedenfor celsius.

2.2 Standarder innenfor helseinformasjonsteknologi

Standarder innenfor helseinformasjonsteknologi er beskrivelser som spesifiserer terminologier, protokoller, metoder og egne spesifikasjoner for innsamling, deling og lagring av kliniske data [23, 24]. Som nevnt tidligere trengs slike standarder for å la ulike kliniske systemer kommunisere og dele viktige kliniske data om pasientene.

Hovedfokuset til standardene kan deles opp i fire grupper; dataformat, terminologi, informasjonsstruktur og datadeling. Standardene innenfor dataformat spesifiserer hvordan dataene representeres, for eksempel XML, JSON, HTML eller RDF.
Terminologistandarder som tilbyr ulike språk og koder for kliniske konsepter er blant annet SNOMED CT, LOINC og ICD. SNOMED CT og LOINC er mye brukt av både openEHR og FHIR, og vil dermed bli beskrevet i avsnitt 2.2.1. ICD (International Classification of Diseases) er en standard for å beskrive ulike sykdommer, skader og lidelser [25]. ICD-10 brukes mye i Norge for å klassifisere ulike sykdommer og andre beslektede helseproblemer.

Noen av de mer kjente standardene som tilbyr en spesifikasjon for hvordan informasjonen bør struktureres er openEHR, HL7 CDA og CIMI. HL7 CDA (Clinical Document Architecture) er en standard som spesifiserer strukturen og semantikken til kliniske dokumenter i XML format [26]. CIMI (Clinical Information Modeling Initiative) er et internasjonalt samarbeid med partnere som HL7, IHTSDO, openEHR, SMART og ulike organisasjoner i land som USA, Sør-Korea og England. CIMI jobber for å tilby et felles format for hvordan innholdet i helsedataene bør struktureres [27]. Til slutt har vi meldingsstandardene som fokuserer på hvordan dataene blir delt mellom de ulike aktørene. Eksempler på slike standarder er HL7 Version 2, HL7 Version 3 og HL7 FHIR [28].

Selv om en standard har et hovedområde kan den brukes til andre formål. Et eksempel er FHIR som hovedsakelig er for deling av data ved hjelp av et REST grensesnitt, men som de selv skriver i spesifikasjonen kan bli brukt for å opprette kliniske dokumenter («Composition») på en måte som tilfredsstiller kravene beskrevet i «Electronic Heath Record System» (EHR-S) [29]. Også openEHR kan brukes til ulike formål. Hovedformålet til openEHR er å spesifisere hvordan dataene bør struktureres og hvilke funksjoner et elektronisk journalsystem bør ha, men man kan finne beskrivelser for hvordan dataene kan deles ved hjelp av et REST API [30]. Et grensesnitt som følger REST spesifikasjonen er strukturert etter ulike prinsipper for hvordan en web tjeneste skal være designet. Det er fem ulike prinsipper som bør følges; «Client-Server», «Stateless», «Cache», «Uniform interface» og «Layered system» [31]. Prinsippene er videre beskrevet i kapittel 7.

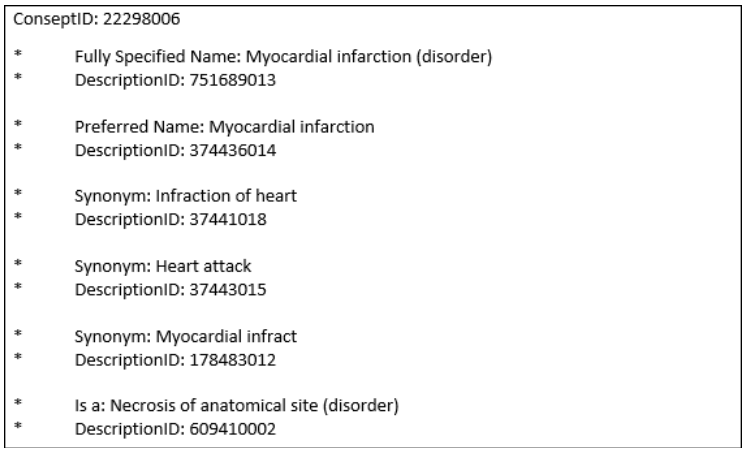
openEHR og FHIR er en datastandard og meldingsstandard henholdvis som begge har en økende interesse internasjonalt. På grunn av dette vil det videre fokuseres på disse to standardene. I de neste delkapitlene blir det introdusert litt bakgrunnstoff om terminologiene SNOMED CT og LOINC, og deretter en introduksjon til standardene openEHR og FHIR.

2.2.1 Klinisk terminologi

Klinisk terminologi er et strukturert ordforråd som blir brukt i ulike kliniske praksiser for å beskrive behandling og pleie av pasienter [32]. En klinisk terminologi dekker konsepter slik som diagnoser, operasjoner, behandlinger og medisiner. I tillegg blir det blant helsepersonell ofte brukt for å dokumentere ulike funn og målinger gjort på pasientene. Det er viktig at den kliniske informasjonen som blir delt mellom helseinstitusjonene er lesbar og at informasjonen blir tolket på rett måte. Terminologistandardene brukes for å oppnå dette.

SNOMED CT

Systematized Nomenclature of Medicine - Clinical Terms (SNOMED CT) er den mest omfattende, flerspråklige kliniske ordforrådet som er tilgjengelig [32]. SNOMED International (tidligere International Health Terminology Standards Development Organisation (IHTSDO)) er en ikke-kommersiell organisasjon som både eier, utvikler og administrerer SNOMED CT. IHTSDO ble grunnlagt i 2007 av ni land (Australia, Canada, Danmark, Litauen, Nederland, New Zealand, Sverige, Storbritannia, og USA). Standarden er i dag brukt i over 50 land og er linket til andre internasjonale standarder [4]. 1. januar 2017 ble Norge et offisielt medlem av SNOMED International, og Direktoratet for e-helse vil nå arbeide med å integrere terminologien videre inn i helsevesenet [33]. Dette er et lovende samarbeid for å øke bruken av terminologistandarder i kliniske dokumenter. SNOMED CT er bygget på ontologier og har over 300 000 aktive konsepter, beskrivelser og forhold, og hvert konsept representerer en klinisk idé. I figur 2.1, hentet fra [34], demonstreres det hvordan et hjerteinfarkt kan beskrives ved hjelp av SNOMED CT. Hvert konsept har en unik identifikator og kan sorteres i et hierarki i lag med flere konsepter. Konseptet kan utvides med ulike kliniske termer eller uttrykk kalt *beskrivelser* som kan deles opp i «Fully Specified Name» (unikt navn), «Preferred Name» (mest kjente navn for konseptet) og synonymer (andre navn for å beskrive samme konsept). Et konsept kan også linkes til andre konsepter dersom begge konseptene har lik betydning. Slike lenker kan beskrives som *ontologiske forhold* og



Figur 2.1: Figuren viser hvordan SNOMED CT kan brukes for å beskrive et hjerteinfarkt. Det finnes flere synonymer for et hjerteinfarkt og dette hierarkiet kan beskrives i SNOMED CT.

det mest brukte forholdet er «Is a» forhold som brukes for å demonstrere et hierarki av konsepter [34]. Dette hierarkiet gjør det mulig for brukerne å søke etter et klinisk konsept og også få returnert alle relevante kliniske konsepter.

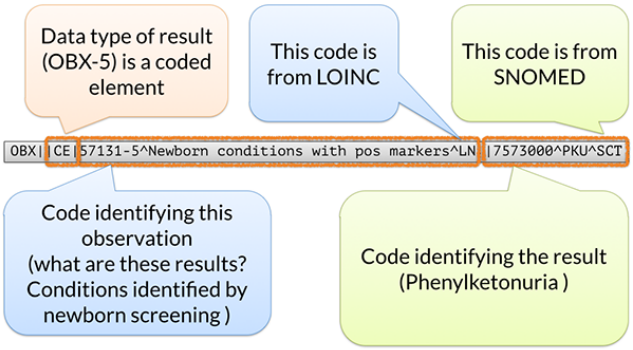
LOINC

Logical Observation Identifiers Names and Codes (LOINC) er en terminologistandard opprettet i 1994 av Clem McDonald, direktør i Lister Hill National Center for Biomedical Communications [35], og arbeidet ble senere videreført av Regenstrief Institute. Regenstrief Institute er en internasjonal ikke-kommersiell medisinsk forskningsinstitutt som samarbeider med universitetet i Indiana [36]. LOINC er et felles språk som inneholder et sett med identifikatorer, navn og koder som kan bli brukt i kliniske og laboratoriske observasjoner. I tillegg kan LOINC inneholde annen informasjon som synonymer, enheter som ble brukt for målinger, og andre beskrivelser for hvert klinisk konsept.

Bruke SNOMED CT og LOINC i lag

Juli 2013 inngikk SNOMED International og Regenstrief Institute et samarbeid for å knytte LOINC og SNOMED CT sammen, forhindre for mange kodeduplikater, og videreutvikle terminologiene for å forbedre informasjonen lagret i pasientjournalene [37]. SNOMED International, med hjelp av Regenstrief Institute, arbeider med retninglinjer for hvordan terminologiene kan brukes i lag. Kort fortalt inneholder LOINC koder som beskriver blant annet kliniske spørsmål og målingsbeskrivelser, mens SNOMED CT inneholder koder for å

blant annet besvare disse spørsmålene og inneholder kliniske verdier som kan brukes for målinger [38]. Figur 2.2, hentet fra [39], viser hvordan LOINC kan brukes i lag med SNOMED CT for å dokumentere en observasjon utført på et nyfødt spedbarn. LOINC blir brukt for å dokumentere selve observasjonsbeskrivelsen, mens SNOMED blir brukt for å dokumentere observasjonsresultatet.



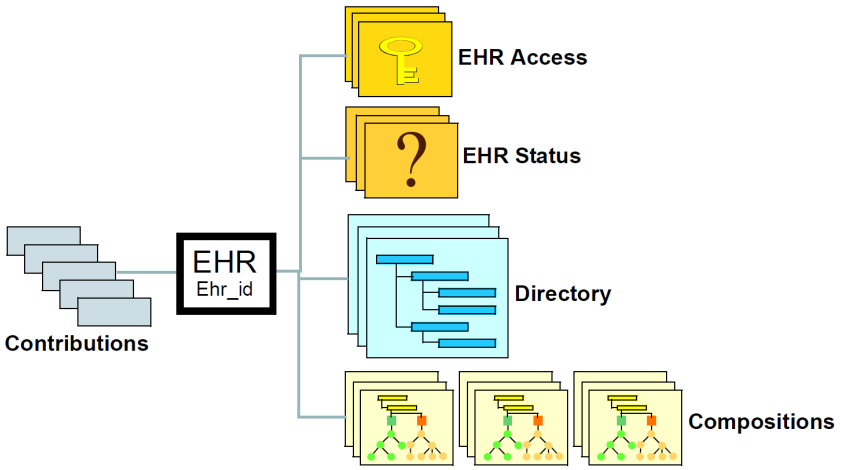
Figur 2.2: Figuren viser et eksempel hvor LOINC brukes i lag med SNOMED CT for å beskrive en observasjon på et nyfødt spedbarn. LOINC koden «57131-5» brukes for observasjonen «Newborn conditions with positive marker». SNOMED CT bruker koden «7573000» for å beskrive observasjonen «phenylketonuria».

2.3 openEHR

openEHR spesifikasjonen er utviklet av et ikke-kommensielt nettverk som vedlikeholdes av openEHR Foundation. Hovedformålet til openEHR er å utvikle en spesifikasjon for et elektronisk pasientjournalsystem som er enkel å ta i bruk og som forsikrer en universell integrasjon av elektronisk data [19]. openEHR tilbyr ulike verktøy som har åpen kildekode som bidrar til å forenkle utviklingen av et openEHR basert system. Standarden er bygget opp ved hjelp av en generisk referansemodell, også kalt for en informasjonsmodell, som definerer den logiske strukturen til elektroniske pasientjournaler. Informasjonen lagret i et openEHR basert pasientjournalsystem er definert ved hjelp av denne referansemodellen.

2.3.1 EHR strukturen

I openEHR blir en pasientjournal omtalt som en EHR (Electronic Health Record), en trestruktur med rotnoden *EHR* som inkluderer elementene; *EHR Access*, *EHR Status*, *Directory*, *Contributions* og *Compositions*. Strukturen er vist i figur 2.3, som er hentet fra [40].



Figur 2.3: En pasientjournal består av et rot-objekt EHR. EHR er en trestruktur med 5 elementer under seg som brukes for å beskrive hvem som har tilgang til journalen, hvem pasienten er, dokumentene som er lagret og hvilke endringer som er gjort i pasientjournalen.

En beskrivelse av hvert element i pasientjournalen er hentet fra [14] og er strukturert ved hjelp av seks punkter.

1. **EHR** er selve rotobjektet som er identifisert ved hjelp av en universell unik EHR identifikator.

2. **EHR Access (versjonert)** er objektet som inneholder informasjonen som har med kontrollering av tilgang til pasientjournalen.

3. **EHR Status (versjonert)** er objektet som inneholder ulik informasjon om status og kontroll. Noen ganger kan den også inneholde en identifikator til pasienten som journalen tilhører.

4. **Directory (versjonert)** er et hierarki av mapper som kan bli brukt for å organisere dokumentene av typen «composition».

5. **Compositions (versjonert)** er ulike beholdere (dokumenter) av dataene relatert til kliniske og administrative innhold i journalen.

6. **Contributions (versjonert)** er objekter som inneholder informasjon om endringene utført i pasientjournalen.

Den kliniske informasjonen er lagret i klasser av typen Composition og disse er videre delt opp i enten seksjoner og/eller elementer. Klassen Entry er elementet hvor informasjonen er lagret og har to ulike underklasser; Admin Entry og Care Entry. Administrativ informasjon blir presentert ved hjelp av Admin Entry. De kliniske konseptene er lagret i form av en

av de fire underklassene av Care Entry; Observation, Evaluation, Instruction eller Action. Seksjoner blir brukt for å organisere en eller flere elementer [40]. Elementene av typen Composition, Observation, Evaluation, Instruction og Action er også kalt for *arketyper*, som er beskrevet i neste avsnitt. Mer informasjon om strukturen til Observation og Evaluation klassene finnes i kapittel 5.

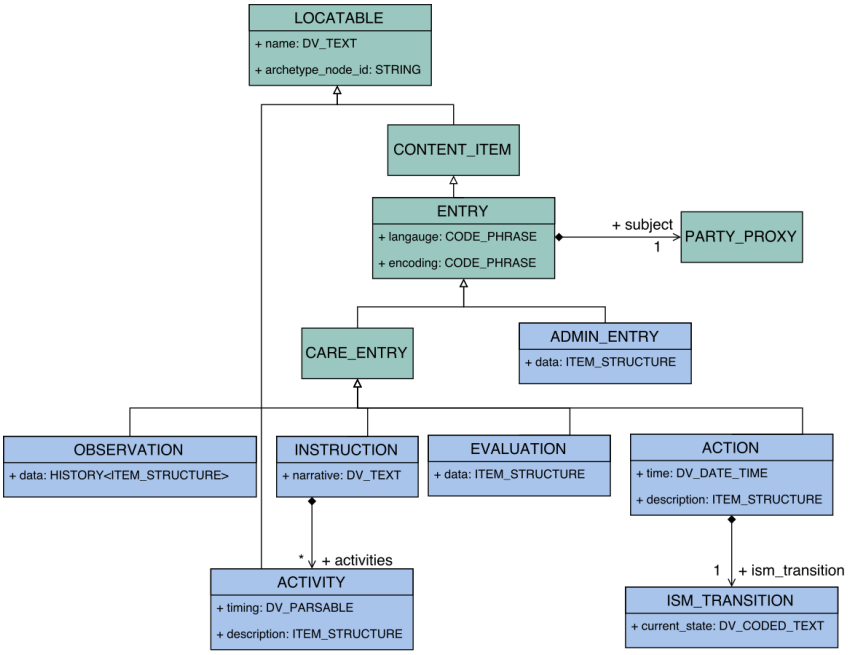
2.3.2 Arketype og mal

Et viktig konsept som openEHR standarden bygger på er *arketype*. En arketype er en modell og mønster for hvordan et klinisk konsept skal presenteres [41]. Det er en maskinlesbar beskrivelse for hvordan en kan lagre pasientdata ved hjelp av openEHR sin referansemodell. En arketype er beskrevet ved hjelp av språket Archetype Definition Language (ADL), men kan også bli representert ved hjelp av XML og JSON om ønskelig. Hver arketype beskriver et komplett klinisk konsept som blant annet *diagnose* eller *blodtrykk*. Flere arketyper har støtte for å lagre informasjonen ved hjelp av terminologistandarder, som medfører at informasjonen vil ha samme betydning uansett hvor den blir brukt i pasientjournalen og uansett hvilket journalsystem som håndterer arketypen. Denne egenskapen er viktig for å bevare den semantiske samhandlingen mellom systemene; hvert system tolker informasjonen likt. Det er kliniske eksperter som er med på å utforme en arketype. Dette medfører at klinikerne ikke lengre er passive brukere av systemet, men er aktivt med på å utforme hvordan en pasientjournal vil fungere og se ut [14]. openEHR sin visjon er å utvikle arketyper med datapunkter for alle mulige brukstilfeller og dermed er hver arketype strukturert med noen obligatoriske felter og flere valgfrie felter. Dette gjør at en arketype skal kunne tilpasses hvilket som helst brukstilfelle. Når en arketype skal brukes tar en i bruk en eksisterende mal (template) eller lager en egen mal ved hjelp av eksisterende verktøy som «Template Designer». En mal består av en eller flere arketyper som er blitt avgrenset og spesialisert ved å fjerne valgfrie felter for et spesifikk brukstilfelle.

Struktur - arketype

En arketype kan deles inn i tre deler; *header*, *definition* og *ontology*. I første del kan en finne blant annet informasjon om personene involvert i utviklingen av arketypen, hvilket formål arketypen har og versjon. I ontologiseksjonen finner man informasjon om hvilke terminologier som er representert i arketypen og hva hvert datapunkt betyr. Datapunktene kan være representert i ulike språk. Selve dataene er lagret i definisjonsseksjonen i arketypen.

Som nevnt tidligere inneholder Composition arketypen en av elementene Observation, Instruction, Evaluation eller Action. Figur 2.4 viser en forenklet modell over hvordan de ulike elementene er bygget opp, hvor de blå elementene er instanser og de grønne elementene er abstrakte klasser. Komposisjonsarketypene kan sammenlignes med kliniske dokumenter som inneholder kliniske data i form av enten OBSERVATION, EVALUATION, INSTRUCTION eller ACTION arketyper [42]. Alle obligatoriske egenskaper som en komposisjon kan ha vises i figuren. Hele strukturen til en komposisjon kan studeres i [40]. De obligatoriske



Figur 2.4: Modellen viser hvordan en komposisjon (COMPOSITION) er bygget opp. Hver arketype vil ha de obligatoriske feltene *name*, *archetype_node_id*, *language*, *encoding* og *subject*. Klassen ADMIN_ENTRY brukes for å beskrive administrativ informasjon som konsultasjoner.

egenskapene som hver arketype har fra ENTRY er *language*, *encoding* og *subject*. I *language* spesifiseres det hvilket språk informasjonen er skrevet i og språket er som regel det samme for hele komposisjonen. *encoding* blir brukt for å spesifisere hvilken koding språket bruker, for eksempel UTF-8. *subject* er en referanse til en pasientidentifikasjon dersom pasienten ikke er anonym. En egenskap som er valgfri å inkludere, men som ofte brukes, er *protocol* arvet fra CARE_ENTRY. Denne egenskapen brukes blant annet for å gi detaljer om hvordan en observasjon ble utført, hvordan en kom fram til en evaluering eller hvordan en skal utføre en instruksjon.

Flere av feltene representeres av en abstrakt datastruktur av typen ITEM_STRUCTURE. Det eksisterer fire ulike datastrukturer; en liste (ITEM_LIST), et tre (ITEM_TREE),

en tabell (ITEM_TABLE) og et enkelt element (ITEM_SINGLE). ITEM_SINGLE og ITEM_LIST består av ett objekt og flere objekter henholdsvis som er av typen ELEMENT, mens ITEM_TABLE inneholder en liste med klynger av typen CLUSTER, som inneholder en eller flere ELEMENT objekter. Hver klynge representerer en rekke i tabellen. Elementene og klyngene er begge en instans av den abstrakte klassen ITEM. ITEM_TREE er den vanligste strukturen som brukes i dag og inneholder en liste med ITEM objekter. Dette gjør det mulig å implementere en datastruktur som kan inneholde både klynger og elementer.

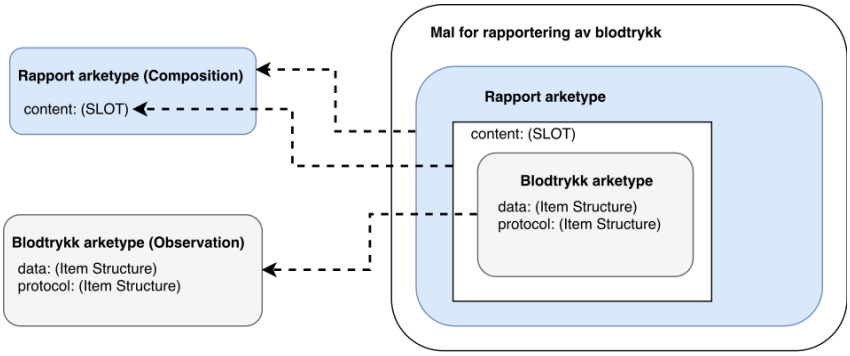
Struktur - mal

En mal har en trestruktur hvor roten er representert ved hjelp av en komposisjonsarketype. Hver komposisjon har et innholdsfelt av typen *SLOT* hvor ENTRY arketypen er lagret. Et SLOT felt gjør det mulig å strukturere arketyper etter ulike behov. En mal som opprettes for en observasjonsarketype har arketypen «Rapport» som rot. Dersom en ønsker å opprette en mal for rapportering av blodtrykk lagres arketypen for blodtrykk i innholdsfeltet til «Rapport» arketypen. Selve innholdsarketypen kan også utvides ved hjelp av andre arketyper dersom et felt er av typen SLOT. Når en mal er opprettet er alle nødvendige arketyper lagt til i tilhørende SLOT felt og andre felter er blitt begrenset eller fjernet om nødvendig for å tilpasse et brukstilfelle.

2.3.3 Template Designer - verktøy for utvikling av maler

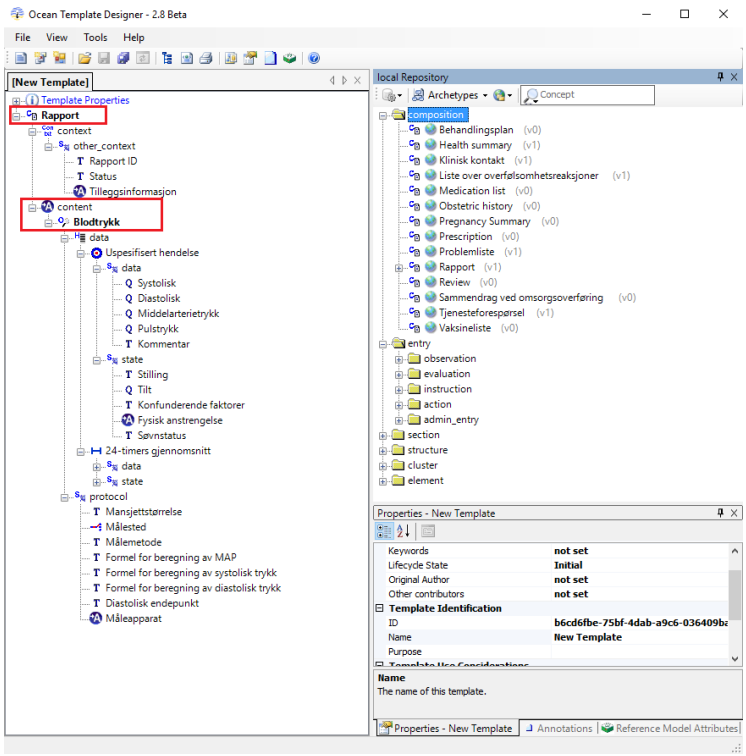
Gjennom nettsiden til openEHR kan en få tilgang til verktøyet *Template Designer* som er utviklet av Ocean Informatics. Ocean Informatics er et australsk helse IKT firma som fokuserer på utvikling av elektroniske pasientjournaler [43, 44].

Når verktøyet åpnes vises det en blank side hvor en kan starte å bygge en ny mal. For å opprette en mal velges det først en komposisjonsarketype hvor arketypen for et klinisk konsept skal være lagret. En komposisjonsarketype har to hovedfelter; *context* og *content*. Arketypen som skal inneholde klinisk informasjon blir lagret i «content» feltet av typen SLOT. Dette utføres enkelt ved hjelp av en «drag-and-drop» funksjonalitet i verktøyet. Brukeren velger en arketype som en ønsker å legge til fra en meny og drar arketypen til SLOT feltet. Ikonet formet som en blå sirkel med bokstaven «A» er ikonet for en SLOT. Oppretting av mal for rapportering av blodtrykk er et fint eksempel. Figur 2.5 viser en forenklet modell av hvordan en mal ser ut. Lik mal for rapportering av blodtrykk er opprettet i Template Designer, vist i Figur 2.6. Når «Rapport» er valgt som rot-arketype (første røde rektangel



Figur 2.5: En mal (høyre figur) er opprettet ved hjelp av en komposisjonsarketype som inneholder enten en observasjons-, instruksjons-, evaluerings- eller handlingsarketype. I figuren er det opprettet en mal for rapportering av blodtrykk. Malen består av komposisjonsarketypen for rapportering, som har en observasjonsarketype for blodtrykk i sitt «content» felt.

i figuren) og observasjonsarketypen for blodtrykk er blitt plassert i «content» feltet (andre røde rektangel i figuren), kan en velge å enten endre kardinaliteten til de ulike feltene eller legge til andre arketyper i de tilgjengelige SLOT feltene.



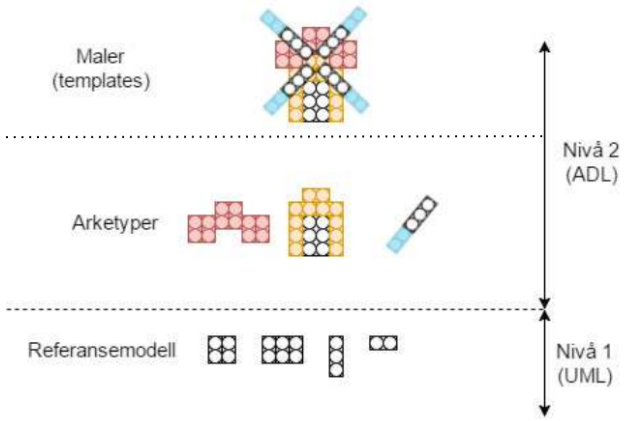
Figur 2.6: Et utsnitt fra Template Designer hvor det er opprettet en mal for rapportering av blodtrykk.

Når malen er ferdig utviklet kan en eksportere malen i formatet «Operational Templa-

te»(.opt). Malen i dette formatet sendes til openEHR systemet for å spesifisere at arketypene som er strukturert etter malen er gyldige.

2.3.4 To-nivå modellering

openEHR strukturen baserer seg på modellering ved hjelp av to nivåer. Denne metoden separerer den generiske referansemodellen og den mer domenespesifikke kliniske modellen. Referansemodellen er strukturert ved hjelp av datatyper som kan brukes for å representere hva som helst av kliniske konsepter. Den kliniske modellen er brukt for å strukturere data og her finnes det mer formelle definisjoner av kliniske konsepter. Referansemodellen er en form for en metamodell, hvor den kliniske modellen fungerer som en modell basert på metamodellen. Det er referansemodellen som gjør at openEHR kan oppnå syntaktiske integrasjon ved å sikre at det er en effektiv overføring av data mellom to systemer, mens det er den kliniske modellen som gir både mening og kontekst til informasjonen som da gjør det mulig å oppnå semantiske integrasjon [45]. Den kliniske modellen i openEHR er representert i form av arketyper og maler.



Figur 2.7: I det nederste nivået har en elementer i referansemodellen i form av enkle legoklosser. Nivået over består av kliniske konsepter i form av arketyper og maler. Nivå 2 er delt opp i to mindre delnivåer for å vise at malene er modeller basert på arketypemodellene.

En god forklaring på to-nivå modelleringen kan gis ved hjelp av legoklosser [46, 47]. Figur 2.7, hentet fra [46, 47], viser hvordan referansemodellen kan beskrives som et sett med alle mulige enkle legoklosser som en kan bruke. Når en modellerer en arketype bruker man byggeklossene som finnes i referansemodellen. I figuren er det laget strukturer for et tak, en bygning og en del av en propell. I openEHR er det i dette nivået en arketype blir strukturert og terminologier blir lagt til. Ved hjelp av arketypene kan en nå bygge en mal for en spesifikk struktur, for eksempel en vindmølle. En kan også bygge et vanlig hus ved hjelp av bygningen

og taket definert som arketype. Samme logikk som beskrevet over blir brukt når en definerer en mal for et spesifikk brukstilfelle.

Referansemodellen til openEHR består av 10 pakker. Disse inneholder alle klassene en trenger for å strukturere en pasientjournal og for å opprette arketyper for å representere kliniske konsepter. Alle pakkene er listet opp i tabell 2.1. Mer informasjon om hver pakke finnes i den digitale referansemodellspesifikasjonen i [48].

Pakke	**Beskrivelse**
Common	Generelle klasser for å strukturere en pasientjournal, som «Folder», «Locatable», «Party Proxy» og «Version Object»
Composition	Klassene som en komposisjon består av, som «Observation», «Content Item», «ISM Transition», «Planned Action» og «Section»
Data Structures	De mest generelle datastrukturene som «Cluster», «Point Event», «Item» og «History»
Data Types	Vanlige datatyper som «Boolean», «Amount», «Duration», «State» og «Text»
Demographics	Informasjon for å beskrive en person, som «Actor», «Organisation», «Role» og «Contact»
EHR	Hovedkomponentene som en pasientjournal består av; «EHR», «EHR Access», «EHR Status», «Versioned Composition», «Versioned EHR Access» og «Versioned EHR Status»
EHR Extract	Klasser for å beskrive henting av data fra en pasientjournalsystem, som «Extract», «Message», «Extract Error» og «Message Content»
Integration	En klasse, «Generic Entry», som brukes for å opprette en arketype som ikke er strukturert etter openEHR sine arketyper. Kan brukes for å opprette en arketype utifra en struktur fra en annen standard
Security	En klasse, «Access Control Settings», som kontrollerer hvem som har tilgang til pasientjournalen
Support	Ulike klasser som er med på å strukturere pasientjournalen, som «Archetype ID», «Party Ref», «UID» og «Object ID»

Tabell 2.1: Pakkene i referansemodellen til openEHR

2.3.5 Archetype Query Language (AQL)

Arketypene har sine egne lokale terminologier som beskrives i ontologiseksjonen i ADL dokumentet. Hver term inneholder en kode, en tekst og en beskrivelse. Et eksempel på en slik kode er at0001. En kode kan bli brukt i en av to ulike situasjoner; enten er koden brukt for å semantisk identifisere datanodene i en arketype, ellers så blir koden brukt for å gi en verdi til et datapunkt i arketypen [49]. I tillegg til intern terminologi er det mulig å binde de interne kodene til eksterne terminologier som LOINC og SNOMED. Dette spesifiseres også i ontologiseksjonen.

Strukturen til en arketype gjør det mulig å utføre ulike spørringer etter data. Ved å bruke stien til en arketype kan en hente ut ønskelig data. For å hente ut data i et openEHR basert system kan en bruke *Archetype Query Language* (AQL), et språk som er basert på *Structured Query Language* (SQL) og XPath. En kode som er brukt for å identifisere en datanode er samme kode som blir brukt i stien til arketypen for å utføre en AQL spørring.

Følgende er et AQL eksempel for hvordan en kan hente ut BMI (Body Mass Index) verdiene som er mer en 30 kg/m2 for en spesifikk pasient [50]:

Listing 2.1: AQL eksempel BMI verdier

```
SELECT
  o/[at0000]/data[at0001]/events[at0002]/data[at0003]/item[at0004]/value
FROM
  EHR [uid=@ehrUid]
    CONTAINS COMPOSITION c [openEHR-EHR-COMPOSITION.report.v1]
    CONTAINS OBSERVATION o [openEHR-EHR-OBSERVATION.body_mass_index.v1]
WHERE
  o/[at0000]/data[at0001]/events[at0002]/data[at0003]/item[at0004]/value > 30
```

Siden AQL spørringene er opprettet ved hjelp av navnet på datanodene og interne koder, kreves det kunnskap om både data- og dokumentstruktur for å opprette en AQL spørring. I kapittel 6 vil vi se hvordan en AQL spørring blir utført i praksis mot et openEHR basert pasientjournalsystem for å hente ut en medisinliste for en pasient.

2.3.6 openEHR sin rolle i Norge

Norge har tatt i bruk standarden openEHR i flere prosjekter. Blant annet står Nasjonal IKT bak sitt eget nasjonale bibliotek for arketyper, `http://arketyper.no/ckm/`, hvor arketypene blir publisert. Tre av fire regionale helseforetak; Helse Nord, Helse Vest og Helse Sør-Øst har en avtale med DIPS AS, en industripartner for openEHR, som er leverandør av pasientjournalsystemet «DIPS Elektronisk Pasientjournal» (EPJ). I nyere tid har det pågått et arbeid med å videreutvikle DIPS-systemet. Det nye systemet, DIPS Arena, baserer seg på openEHR standarden [51].

2.4 HL7 FHIR

Fast Healthcare Interoperability Resources (FHIR) er en spesifikasjon som er utviklet av Health Level Seven (HL7) International for å kunne dele helsedata elektronisk. FHIR er

bygget opp av modulbaserte komponenter kalt «Resources» [52]. I motsetning til openEHR har ikke FHIR en egen journal hvor ressursene er lagret. Det er kun ressursene som blir sendt mellom aktørene. Den relevante informasjonen om en pasient, for eksempel pasient-identifikasjon, vil dermed bli representert i ressursen. FHIR tilbyr en spesifikasjon for et REST basert API for hvordan en kan dele ressursene. Hvordan endepunktene bør se ut og hvilke HTTP metoder som bør være tilgjengelige er godt forklart i spesifikasjonen. FHIR bygger videre på det beste av HL7 sine tidligere utgivelser; HL7 Version 2, HL7 Version 3 og HL7 CDA.

FHIR er stadig under endring. Da arbeidet i masteroppgaven startet var det «Draft Standard for Trial Use 2» (DSTU2), utgitt 24 oktober 2015, som var den offisielle versjonen. 31. mars 2017 ble versjonen «STU3» gitt ut. Denne versjonen endret på strukturen til flere av de eksisterende ressursene og la til noen få nye ressurser.

FHIR er en standard som ønsker å forbedre integrasjonen i kategoriene «Business-to-Business» (B2B) og «Business-to-Client» (B2C). B2B er integrasjonen mellom systemene, mens B2C er integrasjonen mellom systemer og klienter. I masterprosjektet er det bruken av FHIR i B2C integrasjon som blir evaluert. Sammenlignet med openEHR er en FHIR ressurs analog med en openEHR arketype. En ressurs kan enten representeres i et JSON, XML eller Turtle RDF format. Hvor omfattende en FHIR ressurs er varierer fra ressurs til ressurs. Det eksisterer mer spesifikke ressurser for blant annet allergier, ordineringer og henvisninger, og noen få generelle ressurser som «Observation». Fordelen med å ha slike generelle ressurser er at det er lettere å opprette ulike kliniske konsepter basert på den generelle ressursen ved å begrense eksisterende felter og utvide ressursen med nye felter. En slik prosess kalles for *profilering* av en ressurs. En profil kan sammenlignes med en openEHR mal, hvor basisressursen kan sammenlignes med en arketype.

I dag er det 117 ulike ressurser i STU3 versjonen og 93 ulike ressurser i DSTU2 versjonen av FHIR. I motsetning til openEHR forsøker FHIR å utvikle ressurser med datapunkter som dekker cirka 80% av alle brukstilfeller. Altså vil det være egenskaper som openEHR har i sine arketyper som ikke finnes i tilsvarende FHIR ressurser. FHIR ressursene har også egenskaper som ikke finnes i tilsvarende arketyper da ressursene eksisterer uten en pasientjournal. Altså vil det være egenskaper som er lagret i en FHIR ressurs som vil bli lagret utenfor openEHR arketypene i journalrelevante seksjoner i meldingen sendt til openEHR systemet.

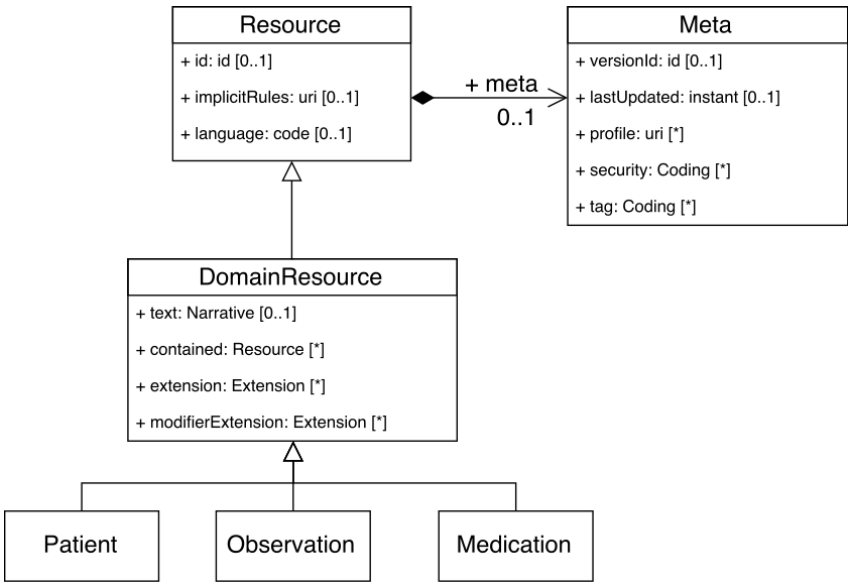
Når en FHIR ressurs mangler egenskaper som en arketype har kan en legge til utvidelser i ressursen ved hjelp av «extensions». Dette er en mekanisme i FHIR for å lage en profilert versjon av ressursen. Det finnes to versjoner av en utvidelse; normale utvidelser og modifise-

ringsutvidelser. Modifiseringsutvidelser påvirker hvordan ressursen blir tolket. Et eksempel på en modifiseringsutvidelse er en utvidelse som forteller at medisinen **ikke** skal bli tatt i en ordineringsressurs. Slike utvidelser SKAL bli tolket av systemet siden det er en viktig utvidelse som har stor betydning for ressursen [53].

2.4.1 Struktur - FHIR ressurs

FHIR spesifikasjonen er delt opp i 5 ulike kategorier i STU3 versjonen; *Foundation*, *Base*, *Clinical*, *Financial* og *Specialized*. Hver kategori har fire eller fem underseksjoner for en mer detaljert kategorisering. Disse fem kategoriene eksisterer bare i FHIR spesifikasjonen for å gi brukerne en bedre navigasjon av ressursene i dokumentasjonen. Strukturen til en FHIR ressurs kan deles opp i fire hovedområder; Identifikasjon og metadata, tekstbasert sammendrag, utvidelser og data. Innhold som pasientnavn, bursdag, kjønn og annen relevant informasjon er lagret i dataseksjonen.

Alle basisressursene unntatt Binary, Bundle og Parameters arver fra domeneressursen *DomainResource*. Dette er en abstrakt ressurs som er en spesialisert versjon av foreldreressursen *Resource*, og inneholder en lesbar XHTML representasjon av innholdet i ressursen, andre ressurser som er relatert til samme ressurs, og utvidelser. Foreldreressursen Resource har fire valgfrie felter; *id*, *meta*, *implicitRules* og *language* [54]. Figur 2.8 viser hovedstrukturen til hver ressurs. FHIR ressursene er bygget opp i en flat struktur hvor ressursene kan ha



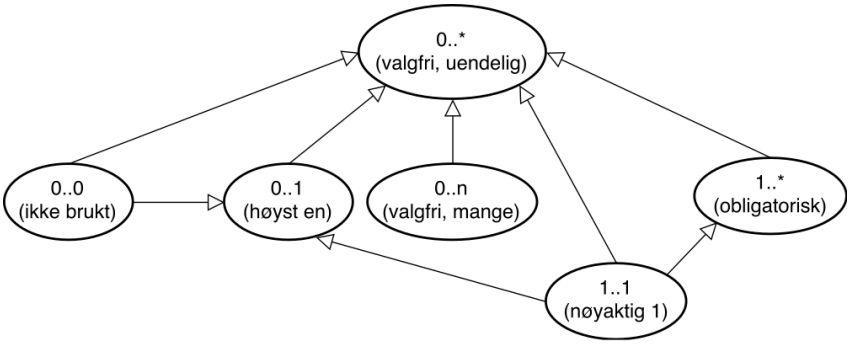
Figur 2.8: Modell over hvordan FHIR ressursene er strukturert.

referanser til hverandre, i motsetning til openEHR som har en hierarkistruktur basert på trestrukturen vist i figur 2.3.

2.4.2 Profilering av ressurser

Profilering av ressurser utføres ved å legge til utvidelser og/eller begrensninger på eksisterende egenskaper funnet i basisressursen. Det er også mulig å utvikle en profil av en profilert ressurs. En profil er beskrevet i en *StructureDefinition* ressurs [55]. Når en profil er utviklet skal den publiseres ved hjelp av et register som gir brukerne en URI (Uniform Resource Identifier) for å hente ut profilen. Denne adressen blir også brukt i ressursen for å spesifisere hvilken profil ressursen er kompatibel med når profilen implementeres.

Ved bruk av en profil kan en begrense kardinaliteten til et element. Reglene for å begrense kardinaliteten kan leses av grafen i figur 2.9. Utifra hvilken kardinalitet som finnes i basisressursen kan en se hvilke kardinaliteter en kan bruke for å begrense elementet i profilen. Når det er en pil fra node A til node B kan en begrense kardinaliteten ved å gå fra kardinaliteten vist i node B til kardinaliteten vist i node A.



Figur 2.9: Grafen tolkes ved å starte på øverste note og deretter gå nedover i treet. Dersom et felt i basisressursen har kardinaliteten «0..*» , kan kardinaliteten endres til blant annet «0..0», «0..1», «0..n», «1..1» eller «1..*» i profilen. Dersom feltet har kardinalitet «0..1» i basisressursen, kan den endres til «0..1» eller «1..1» i profilen.

Når en definerer en profil er det kun mulig å tilføre strengere regler på elementene i basisressursen og aldri motsatt. I tillegg til å begrense antall lovlige repeteringer av et element kan en også begrense hvilke datatyper som er lov å bruke i situasjoner der basisressursen tillater flere ulike datatyper for et element.

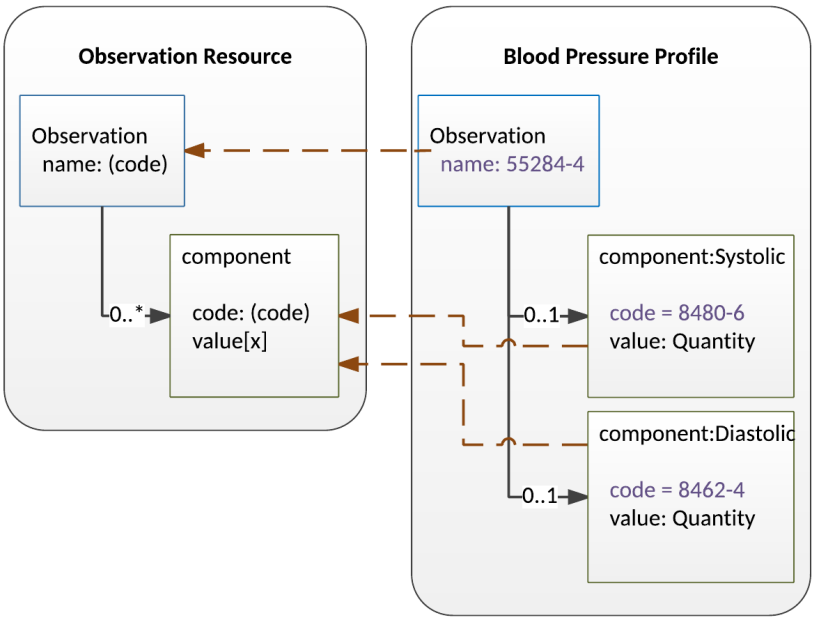
Noen elementer er bundet til lokale koder fra FHIR, standard koder fra terminologier, eller begge. Det er mulig å begrense hvilke koder som er lovlige å bruke i en profil. Denne egenskapen kan bli begrenset ved hjelp av disse fire begrensningene; *required, nivå 1* (kan

kun bruke kodene fra spesifisert terminologi), *extensible, nivå 2* (en kan bruke koder fra andre terminologier gitt at de nye kodene ikke har samme betydning som kodene i den spesifiserte terminologien i ressursen), *preferred, nivå 3* og *example, nivå 4*. De to siste begrensningene gjør det mulig at en kan bruke de kodene en ønsker til lokal bruk. Hver begrensning har i masteroppgaven fått egne nivåer for å lettere kunne beskrive hvordan disse begrensningene kan videreføres til en profil.

Når en tilfører begrensninger på hvilke koder som er lov å bruke i en profil kan en kun bevare samme nivå på begrensningen eller bruke høyere nivå. Nivå 1 er det høyeste nivået, mens nivå 4 er det laveste. Et eksempel er en ressurs som har spesifisert at et datapunkt må bruke koder fra en terminologi, og begrensningen er satt til nivå 2 (extensible). Profilen kan enten bevare samme begrensning eller begrense bruken av koder til *required* med nivå 1.

Slicing

Noen ressurser har lister med elementer. Et eksempel er ressursen FHIR Observation som har en liste med komponenter. I noen tilfeller er det ønskelig å dele listen opp i mindre underlister der hver underliste (*slice*) har egne begrensninger på elementene. Dette kalles for *slicing* eller oppdeling. Figur 2.10, hentet fra [55], viser hvordan en kan opprette en profil for blodtrykk ved hjelp av slicing. Når en oppretter en profil for en blodtrykksobservasjon kan det være ønskelig å dele opp listen med komponenter slik at to komponenter allerede har begrensninger for å representere den systoliske og diastoliske verdien i blodtrykket. For å kunne tolke hver underliste brukes en eller flere felter kalt *discriminator*, som inneholder en sti til et element i underlisten. Elementet som stien peker på har en satt, unik verdi. Dersom en ønsker å dele opp en utvidelse så det alltid URL feltet som er «discriminator» feltet, da dette feltet alltid vil ha en unik verdi blant elementene i underlisten. Dersom rekkefølgen til underlistene har betydning kan dette også spesifiseres. En kan også beskrive regler om at det ikke er lov med flere repeterende elementer (closed) eller at flere repeterende elementer er lov og at 1) de kan være hvor som helst i listen (open) eller 2) de må være spesifiserte etter de spesialiserte underlistene (openAtEnd). Et eksempel hvor det kan være ønskelig å bruke *openAtEnd* er i en blodtrykksprofil, hvor eventuelle ekstra komponenter må være etter den systoliske og diastoliske verdien.



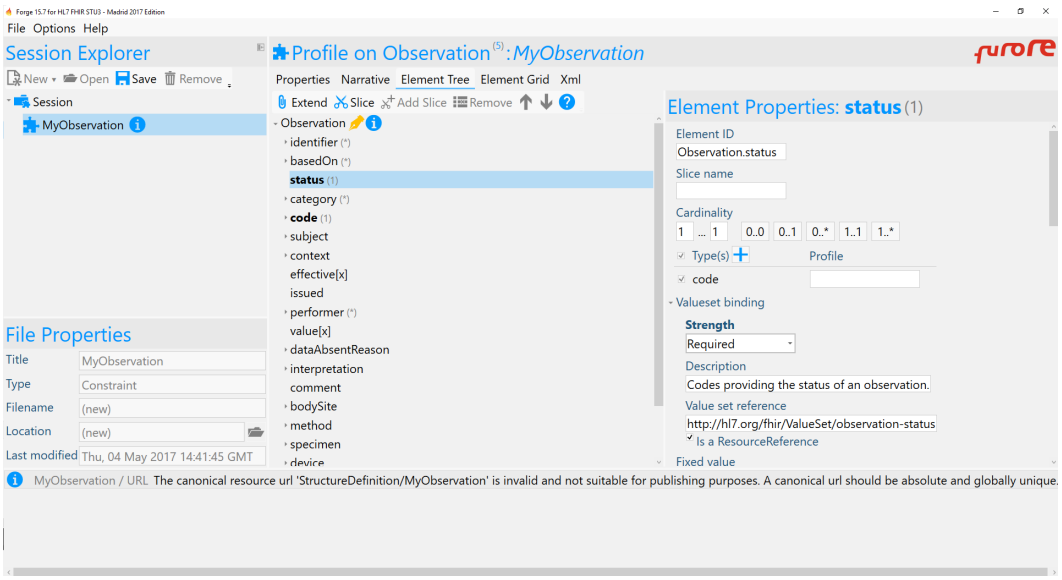
Figur 2.10: Eksempel på hvordan ressursen Observation kan profileres for å beskrive konseptet «Blodtrykk». Her har feltet «name» fått en fast LOINC kode, mens listen med komponenter er delt opp i to mindre elementer. Disse to elementene beskriver den systoliske og diastoliske verdien og har fått en fast LOINC kodeverdi og datatypen er satt til «Quantity».

2.4.3 Forge - profileringsvertøy

I masterprosjektet er verktøyet *Forge* brukt for profilering av ressursene. Forge er utviklet av gruppen Furore [56] som er en arbeidsgruppe i Amsterdam som tilbyr blant annet ulike FHIR verktøy og en FHIR testserver. Forge ble brukt til å legge til begrensninger på feltene; noen ble fjernet og noen ble obligatoriske. Furore oppdaterte sine tjenester 4. mai 2017 til STU3 versjonen. Profilene som er brukt i masterprosjektet ble først utviket for versjon DSTU2, men ble oppdatert til STU3 da den oppdaterte versjonen av Forge ble utgitt.

Furore tilbyr også et FHIR register hvor en kan publisere egne profiler og benytte seg av eksisterende profiler. Registeret heter Simplifier og er tilgjengelig via `www.simplifier.net`. Forge er tett knyttet opp mot Simplifier og lar brukerne publisere profilene direkte fra Forge til Simplifier. Alle profilene som er opprettet i masterprosjektet ble publisert gjennom Simplifier. Forge tilbyr blant annet funksjonalitet for å opprette nye profiler og utvidelser. Figur 2.11 viser et utsnitt av Forge hvor en helt ny profil er startet på for FHIR Observation. Verktøyet er enkel å ta i bruk og er tett knyttet opp mot FHIR spesifikasjonen. Dette gjør det enkelt å kjenne igjen begreper fra FHIR spesifikasjonen, som «extend» og «slice».

Når en profil er opprettet kan FHIR tjenesten som er utviklet validere ressursene mot



Figur 2.11: Utsnitt av verktøyet Forge hvor en har startet på en ny profil for FHIR Observation ressursen. Når en ny profil startes på er profilen lik basisressursen. Nå er det opp til brukeren å legge til utvidelser og endre kardinaliteten der det ønskes. Ruten i midten viser hvordan profilen ser ut nå. På verktøylinjen har en tilgang til «extend» og «slice» for utvidelser og oppdeling av lister. Når et felt er valgt kan en blant annet endre kardinaliteten og legge til egen beskrivelse (se høyre rute).

profilen. En profil kan også bli brukt av utviklere som ønsker å bruke FHIR tjenesten. Utviklerne kan ved hjelp av profilen få spesifisert hvordan en ny ressurs skal struktureres for å være kompatibel med tjenesten, og utviklerne kan bruke profilen for å få beskrevet hvordan ressursstrukturen vil se ut ved søking etter data.

2.4.4 FHIR sin rolle i Norge

Norge har fått utviklet to implementasjonsguider som inneholder ulike ressurser, og gir en god innføring i de norske FHIR profilene. Disse implementasjonsguidene er basert på FHIR versjon DSTU2. Når masteroppgaven ble skrevet var det opprettet 7 ulike norske FHIR profiler; Patient, Perfomer, Procedure Reporting, Diagnosis Reporting, Encounter, Lab Results og Prescriber. En profil for Operation Note er for tiden under utvikling. Det foregår prosjekter i både Helse Sør-Øst, Helse Vest og i Helse Nord. Samtidig har Direktoratet for e-helse flere prosjekter planlagt for 2017 og årene fremover [57].

2.5 Bruk av FHIR til integrasjon

Det har tidligere blitt forsket på hvordan FHIR kan forbedre integrasjonen mot andre standarder enn openEHR. Noen av standardene som har vært i fokus er HL7 CDA, OpenMRS, i2b2 og OMOP CDM. I de følgende avsnittene blir det presentert ulike prosjekter som har testet ut bruken av FHIR som en modul over et eksisterende, mer strukturert og komplekst elektronisk e-helsesystem. Disse fire forskningsprosjektene er plukket ut for å se på tidligere erfaringer med bruk av FHIR til integrasjon. Hvert prosjekt utnyttet FHIR som et grensesnitt ovenfor eksisterende tjenester for å abstrahere komplekse datastrukturer og tilgjengeliggjøre de eksisterende dataene gjennom et utviklingsvennlig REST API.

2.5.1 HL7 FHIR og OpenMRS

I [6] og [58] blir det beskrevet en løsning for å forbedre integrasjonen mot OpenMRS. OpenMRS er en programvareplattform for elektroniske medisinske journaler som blir mye brukt i utviklingsland [59]. Når grensesnittet for OpenMRS ble utviklet var HL7 V2 spesifikasjonen og ulike forslag fra utviklerne brukt som et utgangspunkt for implementasjonen. HL7 V2 manglet blant annet en konsistent datamodell, formell metodikk for elementene i datamodellen, definerte applikasjons- og brukerroller og et skalerbart design. Selv om det har skjedd flere endringer i HL7 V2 grensesnittet de siste årene er det fremdeles flere designvalg som gir en dårlig brukeropplevelse. For å ta grensesnittet i bruk kreves det at utviklerne har stor kunnskap om domenet og det er dermed vanskelig for utviklerne å ta tjenestene i bruk. HL7 FHIR derimot fokuserer på utviklerne og tilbyr en dokumentert spesifikasjon for hvordan et REST API kan implementeres. I tillegg er FHIR spesifikasjonen både enkel og rask å ta i bruk. Som en løsning utviklet forfatterne en FHIR basert modul ved hjelp av HL7 Application Programming Interface (HAPI) FHIR biblioteket som er utviklet av HAPI nettverket. FHIR modulen var strukturert i to lag; et web lag og et API lag. Som et resultat kommuniserte applikasjonene med FHIR modulen via web laget som videre kommuniserte med OpenMRS grensesnittet via API laget. Nå kunne en del av dataene lagret i OpenMRS bli hentet som enten «Patient», «Person», «Location», «Observation», «Encounter» eller «AllergyIntolerance» FHIR ressurser.

2.5.2 HL7 FHIR og i2b2

«Informatics for Integrating Biology and the Bedside» (i2b2) er et klinisk register for analysering av data. I2b2 blir hovedsakelig brukt for å kunne hente ut pasientdata for forskingsrelaterte spørsmål. Selv om i2b2 inneholder pasientdata blir ikke i2b2 plattformen brukt i klinisk arbeid. Istedenfor er i2b2 en kopi av dataene lagret i en elektronisk pasientjournal som gjør at i2b2 kan kjøres parallelt med pasientjournalen. Dermed har i2b2 plattformen fokus på forskning mens pasientjournalen blir brukt i klinisk arbeid [60]. For å kunne gjøre dataene mer tilgjengelig for tredjeparts applikasjoner utviklet forfatterne et grensesnitt for å hente ut data som FHIR ressurser. En av hovedårsakene til at forfatterne valgte FHIR for å tilgjengeliggjøre i2b2 dataene var at FHIR standarden er mye brukt i dag og det er et stort nettverk som investerer mye tid og kunnskap i å gjøre standarden enda bedre. FHIR ressursene som var støttet da artikkelen ble skrevet var «Patient», «Medication», «MedicationPrescription» (heter i dag MedicationRequest), «Observation» og «Condition».

2.5.3 HL7 FHIR og HL7 CDA

Østerrike har et elektronisk journalsystem, «Elektronische Gesundheitsakte» (ELGA), som bruker HL7 CDA og IHE XDS for å strukturere og dele informasjonen i journalsystemet. HL7 Clinical Document Architecture (CDA) er en standard som spesifiserer hvordan kliniske dokumenter kan struktureres og hvilken semantikk som bør brukes i dokumentene [61]. IHE XDS (Integrating the Healthcare Enterprise Cross-Enterprise Document Sharing) er et system med konkrete spesifikasjoner for hvordan en kan dele kliniske dokumenter (for eksempel i HL7 CDA format) mellom systemer og virksomheter, og som er mye brukt i dag. IHE XDS har noen mangler som medfører at det er vanskeligere å søke i dataene lagret i journalsystemet [62]. Forfatterne av artikkel [62] ønsket at det skulle være enklere å få tilgang til informasjonen lagret i journalen og utviklet dermed en mekanisme basert på FHIR for å transformere ELGA CDA dokumenter over til FHIR ressurser. Dette ville da medføre at interessenter som pasienter, applikasjonsutviklere, omsorgsarbeidere og forskere fikk lettere tilgang til ELGA data. CDA dokumentene er strukturert som et XML dokument, og transformeringen mellom CDA dokumentene og FHIR ressursene ble utført gjennom XPATH uttrykk. For strukturering av transformeringsfilen tok de utgangspunkt i JSON representasjonen av FHIR ressursene og for hver CDA dokumenttype som ble transformert utviklet de en avbildning mellom det aktuelle CDA dokumentet og tilhørende FHIR ressurser. For hver ressurs som bygget opp CDA dokumentet hadde de en «cda-path» som refererte til et XML undertre som korresponderte til en FHIR ressurs. Deretter brukte de «cda-value» for

å hente ut en verdi som ble lagret i FHIR ressursen på rett sted. Dersom en transformasjon var nødvendig (for eksempel fra 'male' til 'M') inkluderte de en «cda-mapping». Når transformeringen til FHIR ressursene var ferdig ble de lagret i en FHIR Bundle som ble sendt til klienten.

2.5.4 HL7 FHIR og OMOP CDM

«The Observational Medical Outcomes Partnership» (OMOP) «Common Data Model» (CDM) er en metode for å strukturere dataene i et felles modellformat for å utføre kliniske prediktive analyser på dataene. I artikkel [63] ble OMOP CDM brukt for å trene risikomodeller for å evaluere en risiko for en pasient. Disse modellene er vanskelige og krevende å bruke og prosessere for personer uten domenekunnskap. Forfatterne ønsket å tilgjengeliggjøre resultatene fra risikoevalueringene som FHIR ressurser gjennom en FHIR tjeneste. Klientene sender nødvendig pasientinformasjon til en FHIR tjeneste, som bruker en komponent, «Deployed Predictive Model», for å få tilbake en kalkulert risikoverdi. Evalueringen lagres i en RiskAssessment ressurs hvor en identifikator for ressursen returneres tilbake til klienten. Deretter lagres ressursen i en egen FHIR database. Klienten kan nå bruke den returnerte identifikatoren for å hente ut en ressursen som inneholder en kalkulert risiko. Dette forskningsarbeidet skiller seg fra de tre tidligere forskingsprosjektene ved at de er bare interessert i den kalkulerte risikoverdien, og ikke et helt dokument. Dermed er det ikke utført en konvertering mellom OMOP CDM dataene og FHIR på samme måte som vi har sett tidligere. For å teste implementasjonen ble det utført kalkuleringer på to datasett hvor de sammenlignet responstiden og robustheten til FHIR tjenesten ved å sende flere forespørsler samtidig.

Kapittel 3

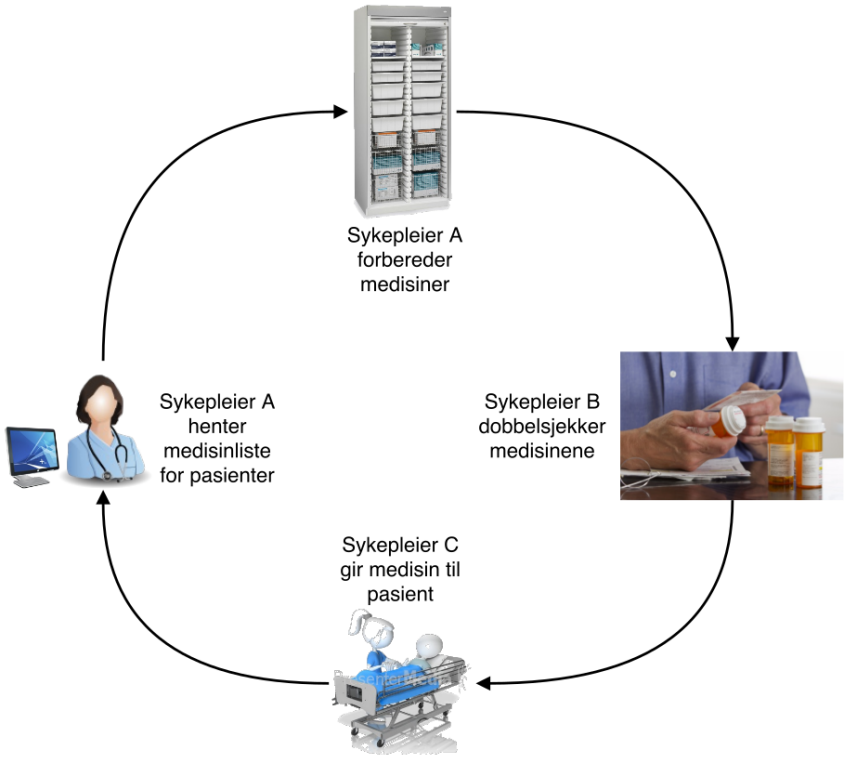
Referanseimplementasjon

Som beskrevet i kapittel 1 skal vi i masterprosjektet strukturere og evaluere en tjenestearkitektur. Arkitekturen skal bestå av en FHIR integrasjonsbuss som tilbyr brukerne kliniske testdata og en spesifikasjon for hvordan helsedataene skal struktureres. FHIR grensesnittet skal abstrahere de mer komplekse data- og dokumentstrukturene ved å konvertere dataene fra de bakenforliggende e-helsesystemene til FHIR ressurser. Det skal vurderes hvordan denne arkitekturen kan forbedre integrasjonsprosessen mellom klientapplikasjoner og bakenforliggende systemer. For å kunne evaluere arkitekturen skal vi opprette en mindre arkitektur hvor vi bare har ett bakenforliggende system; et openEHR basert pasientjournalsystem. Pasientjournalsystemet skal være tilgjengelig for brukerne gjennom FHIR bussen. Klientapplikasjonene skal da kunne hente og lagre openEHR data som FHIR ressurser. Før vi starter å utvikle FHIR bussen og openEHR tjenestene er det viktig å sette opp klare mål og krav for hva vi ønsker å oppnå med tjenestene. Som en kilde for utvikling av krav presenteres en brukerhistorie. Denne brukerhistorien vil senere bli brukt som et utgangspunkt for modifisering av en applikasjon som da skal kunne bruke FHIR bussen (se kapittel 6).

I kapittel 7 skal vi vurdere vår arkitektur bestående av FHIR bussen og openEHR pasientjournalsystemet. Denne arkitekturen skal deretter brukes for evaluering av hvordan en større tjenestearkitektur med en FHIR buss som et grensesnitt for flere bakenforliggende e-helsesystemet kan være fordelaktig for utvikling og integrering av klientapplikasjoner. For å kunne strukturere vår arkitektur er det presentert flere delmål som hjelper til med arbeidet underveis. Målene ble brukt for å utvikle flere funksjonelle og ikke-funksjonelle krav. Kravene ble brukt som veiledning i løpet av oppgaven og brukes i kapittel 7 hvor vår arkitektur skal vurderes og evalueres.

3.1 Brukerhistorie

Hver dag går sykepleiere en eller flere medisinrunder hvor pasientene får medisinene de skal ha. En slik medisinrunde kan deles opp i fire prosesser som er fremvist i figur 3.1.



Figur 3.1: Figuren viser hvordan en medisinrunde oftest foregår på sykehuset. Først leser sykepleier A hvilke medisiner en pasient skal ha. Deretter går sykepleier A til et medisinrom for å forberede medisinene. Når medisinene er gjort klare blir alt dobbelsjekket av sykepleier B for å bekrefte at sykepleier A har forberedt medisinen rett. Når pasienten skal ha medisinene sine henter sykepleier C medisinene som allerede er forberedt og dobbelsjekket, og administrerer de til pasienten.

I denne brukerhistorien håndterer sykepleieren en pasient om gangen. Det første sykepleieren gjør er å hente listen med medisiner som pasienten skal ha. Her må sykepleieren sjekke om listen er oppdatert med gyldige medisiner. Sykepleieren henter også informasjon om dosering og hvilken administreringsform som skal brukes. En administreringsvei hentes også og kan være blant annet oral bruk, intravenøst eller via huden. Etter sykepleieren har lest av medisinlisten kan han forberede rett dose og type av hver medisin som pasienten skal ha. Siden denne forberedelsen skjer manuelt må en annen fra helsevesenet dobbelsjekke og bekrefte at sykepleieren har forberedt rett medisin til pasienten. På større sykehus forberedes ofte medisinene vakten før slik at når neste sykepleier starter vakten sin neste dag så er

medisinene klare for administrering. Sykepleieren som skal administrere medisinene vil også lese av medisinlisten en gang til for å undersøke at ingen oppdateringer har kommet etter medisinene ble forberedt. Når alt er klart og godkjent kan sykepleieren gå til rommet til pasienten for å gi medisinene.

I en medisinrunde kan mye gå galt. Hvert steg utføres manuelt og i en hektisk hverdag er det lett for en sykepleier å gjøre feil. Denne problematikken omtales for *kognitiv overbelastning* [64–66] og er en tilstand som kan oppstå hvor personer utfører oppgaver som er svært krevende over lengre tid og/eller håndterer flere sanseinntrykk samtidig. Dette kan medføre at hjernen blir utslitt og personen kan lide av blant annet rastløshet, uro i kroppen, nedsatt konsentrasjon og irritabilitet, og er et eget forskningsområde innenfor helse.

3.2 Mål for systemet

Oppgaven har ett hovedmål;

Strukturere og evaluere en tjenestearkitektur med en FHIR integrasjonsbuss integrert mot bakenforliggende e-helsesystemer for forbedring av integrasjonsprosessen mellom klientapplikasjonene og e-helsesystemene.

For å kunne nå dette målet skal vi implementere en FHIR buss som skal integreres mot et openEHR basert pasientjournalsystem for å evaluere fordelene med å kunne hente og lagre openEHR arketyper gjennom et FHIR basert REST API. Det er utviklet flere delmål for både FHIR bussen og openEHR pasientjournalsystemet. Disse målene er med på å utvikle både funksjonelle og ikke-funksjonelle krav, som påvirker hvordan systemene blir implementert og hvilke verktøy som blir brukt.

3.2.1 FHIR buss

FHIR bussen skal være funksjonell og bør være konfigurerbar for å integrere seg mot andre standardbaserte systemer. FHIR bussen skal følge den siste versjonen av FHIR spesifikasjonen som er tilgjengelig. Det ble satt opp ulike mål for hvordan FHIR bussen skulle bli utformet:

- **Konfigurerbar** - Det skal være enkelt å endre hvilket system en ønsker å bruke som datakilde uten at FHIR grensesnittet endres.

- **Uavhengighet** - For å integrere seg mot FHIR bussen skal det ikke være nødvendig med kunnskap om strukturen til openEHR systemet (eller andre systemer) den integrert mot.
- **Oppdatert** - Bussen skal være kompatibel med siste utgave av FHIR spesifikasjonen, slik at det er større sannsynlighet for at klientene som bruker bussen også er kompatible med andre FHIR systemer.
- **Mulighet for utvidelse** - Det skal være mulig å utvide bussen med nye tjenester for nye ressurser. I masterprosjektet er det kun noen utvalgte ressurser som er tilgjengelig via tjenester, men det kan være ønskelig å utvide bussen med flere ressurser og legge til flere funksjonaliteter for eksisterende ressurser.

3.2.2 openEHR journalsystem

I masterprosjektet ble det bestemt å bruke et eksisterende openEHR system med åpen kildekode for testing av FHIR bussen. Siden det var flere tilgjengelige openEHR prosjekter med åpen kildekode ble det utviklet ulike kriterier for hjelp med valget av hvilket openEHR system som skulle bli brukt:

- **Oppdatert** - Systemet bør jevnlig oppdateres opp mot openEHR spesifikasjonen og det bør være et aktivt miljø som vedlikeholder systemet slik at en forsikrer seg om at systemet ikke blir utdatert.
- **Støtte for format** - Systemet bør støtte ulike formater som XML, JSON og FLATJSON. Det er ekstra ønskelig at systemet støtter FLATJSON, som er et format som fjerner dokumentstrukturen og inneholder bare de kliniske dataene.

3.3 Funksjonelle krav

For både FHIR bussen og openEHR pasientjournalsystemet ble det utviklet en liste med funksjonelle krav som blir brukt som en mal under utviklingen av tjenestene og valg av teknologi (se kapittel 4). De funksjonelle kravene spesifiserer ulike funksjoner som skal utvikles utifra brukerhistorien og målene, og disse kravene er med på å vurdere systemet i kapittel 7.

1. **Opprette ny pasientjournal når en pasient opprettes**:
   openEHR systemet skal støtte funksjonaliteten om å opprette ny pasientjournal for

en ekstern pasient som er lagret i et annet system. I dette prosjektet skal pasientene være lagret i en egen intern database til FHIR bussen.

2. **Tilby tjenester for henting og lagring av FHIR ressurser**:
   FHIR bussen skal tilby et API for å hente og lagre nye ressurser som pasienter, sykepleiere og medisiner. Ressursene for pasienter, sykepleiere og medisiner skal være lagret i en intern database. Medisinene skal inneholde nok informasjon slik at det er mulig å opprette en ny legemiddelordinering i openEHR systemet utifra medisinen.

3. **Lagre ny mal i openEHR pasientjournalsystemet for å kunne hente og lagre arketyper kompatible med malen**:
   Det skal være mulig å lagre nye maler i openEHR pasientjournalsystemet slik at brukerne kan lagre og hente ulike arketyper som komposisjoner fra pasientjournalsystemet.

4. **Lagre og hente komposisjon fra openEHR pasientjournalsystemet**:
   Når en mal er lagret i pasientjournalsystemet skal det være mulig å både lagre og hente ut en komposisjon fra pasientjournalsystemet.

5. **Oppdatere FHIR bussen med flere ressurser**:
   Det skal være mulig å oppdatere FHIR bussen med endepunkter for nye ressurser. Systemet bør være implementert med en enkel og god struktur som forenkler prosessen med å oppdatere FHIR bussen.

6. **Konvertering mellom openEHR arketyper og FHIR ressurser**:
   openEHR arketypene som er lagret i openEHR pasientjournalsystemet må kunne konverteres over til en FHIR ressurs ved henting av data. En arketype som skal lagres må først lagres som en FHIR ressurs gjennom FHIR bussen som deretter konverteres over til en openEHR arketype.

7. **Lagre og hente arketype som en FHIR ressurs**:
   For hver arketype som openEHR systemet støtter skal det være mulig å hente og lagre arketypen som en FHIR ressurs.

8. **openEHR komposisjonsformat**:
   Komposisjonene som inneholder openEHR arketypene skal være strukturerte i et FLATJSON format. FLATJSON inneholder de kliniske dataene som komposisjonen har. Istedenfor at dataene er lagret i en dokumentstruktur brukes XPATH i lag med openEHR sine lokale koder for å identifisere de kliniske dataene.

3.4 Ikke-funksjonelle krav

Det er et par ikke-funksjonelle krav som også bør oppfylles når implementering av FHIR bussen og valg av openEHR pasientjournalsystem utføres. Ikke-funksjonelle krav beskriver ulike begrensninger for systemets funksjonalitet og hvordan systemet blir implementert. Som i forrige avsnitt er kravene videreutviklet fra målene i avsnittet 3.2.

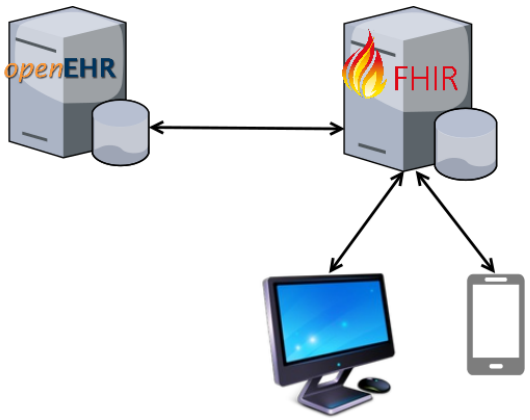
1. **Kjøre i en skytjeneste**:
   Både openEHR systemet og FHIR bussen bør være utviklet i et programmeringsspråk eller bruke teknologier som er støttet av de fleste skytjenester. Dette gjør det enklere å kjøre tjenestene i en skybasert løsning og vi kan senere flytte tjenestene til nye skytjenester om nødvendig.

2. **Vedlikeholdbarhet**:
   Hver komponent i FHIR bussen bør være modulbaserte for å kunne endre, fjerne og legge til funksjonaliteter i systemet uten at hele systemet blir påvirket.

Kapittel 4

Systemarkitektur og implementasjon

I dette kapittelet vil komponentene utviklet i masterprosjektet bli presentert. Hvilke verktøy og rammeverk som er brukt ble avgjort ved hjelp av kravene spesifisert i kapittel 3. Som nevnt i kapittel 1 vil systemet bestå av tre aktører; et openEHR pasientjournalsystem, en FHIR buss og brukere. Figur 4.1 viser et forenklet bilde av hvordan de tre ulike komponentene kommuniserer sammen. I første avsnitt blir det gitt en beskrivelse av hver teknologi



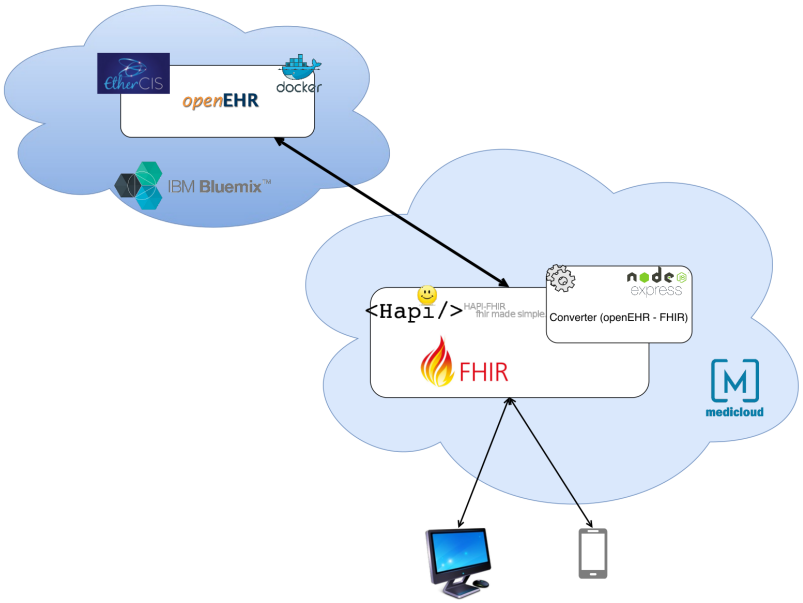
Figur 4.1: Modell over systemet bestående av tre komponenter; openEHR pasientjournalsystem, FHIR buss og brukere.

og rammeverk brukt i masterprosjektet. En begrunnelse for hvorfor de ulike teknologiene ble valgt er gitt i hvert avsnitt. Avsnitt 4.2 består av en detaljert beskrivelse for hvordan de ulike komponentene kommuniserer med hverandre og hvordan systemene er delt opp. I

siste avsnitt diskuterer vi sikkerhetshåndteringen i masterprosjektet og begrunnelse for våre valg.

4.1 Utviklingsverktøy og teknologi

Med hjelp av kravene i kapittel 3 kunne vi velge hvilke teknologier og biblioteker som skulle bli brukt for å sette opp FHIR bussen og openEHR systemet. For både openEHR og FHIR finnes det flere biblioteker og offentlige servere som en kan bruke. For hver mulighet ble det undersøkt om de mulige verktøyene fulgte kravene. Det var også viktig at det ved hjelp av verktøyene var enkelt å sette opp både FHIR bussen og openEHR systemet. Det ble også vurdert hvilket verktøy som hadde den beste funksjonaliteten for konfigurering av systemet etter det var satt opp. En enkel oversikt av arkitekturen og hvilke verktøy og teknologier som ble brukt vises i figur 4.2. Klientene kommuniserer direkte med FHIR bussen, som har en intern konverteringsmodul som kan konvertere ressurser til arketyper og arketyper til ressurser. FHIR bussen kommuniserer direkte med openEHR pasientjournalsystemet.



Figur 4.2: Figuren viser hvordan de ulike teknologiene og verktøyene er brukt i vår arkitektur. Nederst er ulike klienter (presentert som en datamaskin og en smarttelefon) som har tilgang til ulike tjenester via FHIR bussen som bruker biblioteket HAPI FHIR. FHIR bussen bruker en egen konverteringskomponent utviklet ved hjelp av NodeJS og Express som kun tar i mot enten en openEHR komposisjon eller en FHIR ressurs og konverterer den til den motsatte standarden. FHIR bussen og konverteringsmodulen kjører i MediCloud. Når dataene er konvertert sendes en forespørsel videre til EtherCIS openEHR systemet som kjører som en Docker container i et privat område i IBM Bluemix.

Mer informasjon om de ulike teknologiene er beskrevet i følgende avsnitt.

4.1.1 HAPI FHIR

FHIR bussen ble utviklet ved hjelp av biblioteket HAPI FHIR. HAPI (HL7 Application Programming Interface) FHIR er et bibliotek med åpen kildekode utviklet i Java og er en implementasjon av FHIR spesifikasjonen. HAPI FHIR prosjektet er ledet av James Agnew, en ansatt hos University Health Network i Toronto [67]. I dag støtter HAPI FHIR både DSTU2 og STU3. HAPI FHIR biblioteket ble importert i et Spring prosjekt ved hjelp av Apache Maven. Som en demo for hvordan HAPI FHIR kunne bli brukt for å utvikle FHIR bussen ble det offisielle prosjektet funnet på GitHub (`https://github.com/jamesagnew/hapi-fhir`) lastet ned og brukt som et utgangspunkt for FHIR bussen. HAPI FHIR spesifikasjonen som finnes på hjemmesiden [18] ble brukt som dokumentasjon for hvordan HAPI FHIR følger FHIR spesifikasjonen. Begrunnelsen til at HAPI FHIR ble brukt til utvikling av FHIR bussen var at biblioteket tilbyr fullverdig funksjonalitet for å enkelt implementere en tjeneste som følger FHIR spesifikasjonen og at prosjektet fremdeles vedlikeholdes og videreutvikles. Da forskningsarbeidet startet ble DSTU2 versjonen av HAPI FHIR brukt for å implementere FHIR bussen. Underveis kom det en ny utgave, STU3, og FHIR bussen ble oppdatert til å bruke STU3 versjonen av HAPI FHIR.

4.1.2 EtherCIS

Siden hovedarbeidet i masterprosjektet er å utvikle en FHIR buss ble det bestemt å finne et eksisterende prosjekt med åpen kildekode som kunne brukes for pasientjournalsystemet i masterprosjektet. På nettsiden til openEHR kan en finne to ulike prosjekter som tilbyr et openEHR basert system. Det første systemet heter *EHRServer* utviklet av Pablo Pazos som er ansatt i CaboLabs. Dette er et system som er kompatibelt med openEHR og tilbyr en god dokumentasjon for hvordan systemet kan settes opp og hvordan en kan lagre og hente data fra systemet. Systemet støtter at dokumentene lagres i både XML og JSON format. Årsaken til at dette ikke ble brukt var at systemet ikke støttet et FLATJSON lignende format utviklet av leverandøren Marand som står bak prosjektene EhrScape, Think!Meds og Thing!Ehr Platform [68]. EtherCIS er prosjektet som ble valgt som openEHR system i masterprosjektet. Dette er et prosjekt med åpen kildekode som er kompatibelt med openEHR. REST grensesnittet til EtherCIS er basert på EhrScape som er lett tilgjengelig gjennom nettsiden `https://www.ehrscape.com/api-explorer.html`. EtherCIS støtter dokumen-

ter både i XML, JSON, FLATJSON og sitt eget format ECISFLAT. ECISFLAT er en utgave basert på Marand sitt FLATJSON format. Dette formatet lar dokumentene inneholde den viktigste informasjonen i et brukervennlig JSON format, og utviklerne slipper å parse og opprette XML dokumenter. EtherCIS prosjektet finnes på GitHub på nettsiden `https://github.com/ethercis`. En annen grunn til at EtherCIS ble valgt som openEHR system var at det eksisterer en «Docker» versjon av systemet som gjør det lettere å laste opp prosjektet på ulike servere som støtter Docker containere.

4.1.3 Docker

Docker er et prosjekt som ønsker å forbedre distribusjonen av programvare ved hjelp av Docker containere. En Docker container inneholder alt som et program trenger for å kjøre. Dette er blant annet kildekoden, biblioteker som programmet er avhengig av, miljøet som programmet skal kjøre på og andre systemverktøy og innstillinger. Dette gjør det mulig å garantere at programmet vil fungere likt uansett hvilket miljø det kjører på. IBM Bluemix støtter containere og siden det eksisterer en ferdig implementasjon av en EtherCIS Docker container ble det bestemt å utnytte denne funksjonaliteten. Dette gjør det mulig å kjøre samme openEHR server både lokalt og i skytjenester som IBM Bluemix og garantere at systemet vil fungere akkurat likt uansett hvor det kjører.

4.1.4 MediCloud og IBM Bluemix

Da det i masterprosjektet også testes og evalueres hvordan en kan utvikle en FHIR buss i en skytjeneste ble IBM sin skyløsning Bluemix testet ut. IBM Bluemix leveres i tre ulike varianter; *public*, *dedicated* og *local*. Sykehuspartner HF er en organisasjon som jobber med å levere og drifte tjenester i helseregionen Sør-Øst. Organisasjonen står bak den skybaserte plattformen MediCloud, som er en dedikert variant av IBM Bluemix. Dette er en ny løsning som fokuserer på prototyping og eksperimentering av nye tjenester og applikasjoner, og som ønsker å tilby en felles nasjonal skyløsning [10]. En dedikert versjon er, i motsetning til «public» som har alle funksjonene tilgjengelig, en variant som kun har utvalgte funksjoner tilgjengelig. Aktøren som styrer den dedikerte skytjenesten kan også velge hvor dataene skal være lagret. Funksjonene som var tilgjengelig i perioden masteroppgaven ble skrevet var Cloud Foundry Apps og fem tjenester i Services & APIs. Siden det eksisterer en Docker versjon av EtherCIS som var ferdig konfigurert var det ønskelig å laste opp en container i skyen. Dessverre tilbød ikke MediCloud muligheten å utnytte IBM Containers og et eget

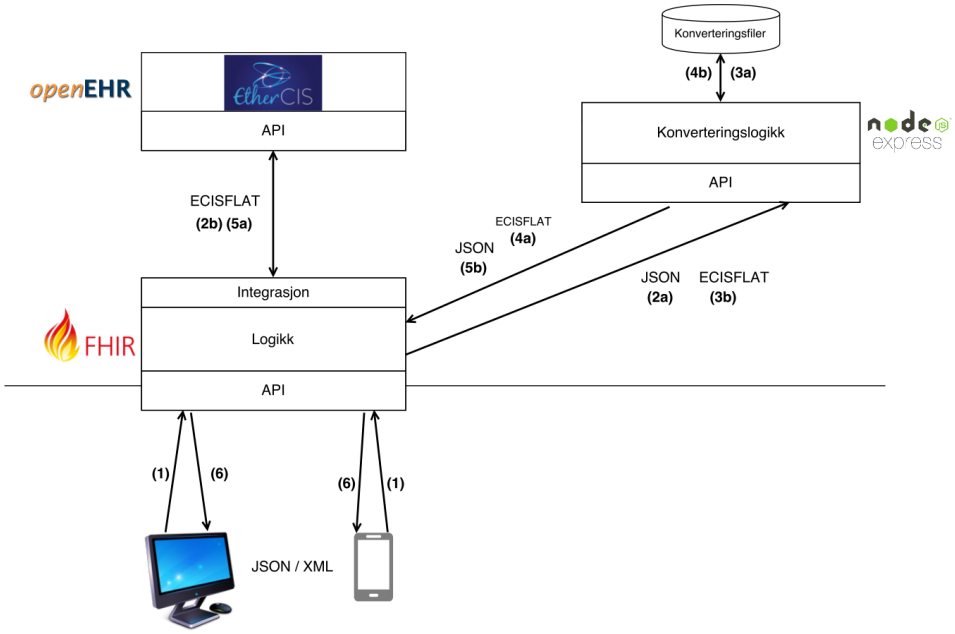
privat IBM Bluemix område ble satt opp. Dette området var en «public» variant av IBM Bluemix. EtherCIS openEHR serveren ble deretter distribuert på det private område i IBM Bluemix. FHIR bussen og konverteringskomponenten ble distribuert i MediCloud som en Tomcat applikasjon og Node.js applikasjon henholdsvis.

4.1.5 Node.js og Express

Logikken for å konvertere mellom openEHR arketyper og FHIR ressurser ble flyttet i egen komponent for å utnytte hvordan JavaScript kan håndtere objekter uten å ha en klasse opprettet på forhånd. Dette gjorde det mulig å utvikle endepunkter som håndterer flere ulike arketyper og ressurser. Node.js er et rammeverk som blir brukt for utvikling av servere og applikasjoner skrevet i JavaScript. Mer informasjon om Node.js finnes på nettsiden `https://nodejs.org/en/`. Node.js i seg selv er ikke nok for å utvikle en tjeneste og dermed ble også rammeverket Express brukt. Express er et lettvektet rammeverk som tilbyr ulike funksjonaliteter for utvikling av web og mobilapplikasjoner. Express ble brukt for å utvikle endepunktene for å konvertere arketypene og ressursene. For dokumentasjon om Express, se `http://expressjs.com/`. Både Node.js og Express er rammeverk som er enkle å ta i bruk og tar liten tid å sette opp. Dette var viktig under utvikling av konverteringskomponenten.

4.2 Systemarkitektur

Systemet er bygget opp av to hovedkomponenter; en FHIR buss og et openEHR pasientjournalsystem. For å få en introduksjon til hvordan systemene er bygget opp og er integrerte mot hverandre vises det i figuren 4.3 hvordan de ulike komponentene er strukturerte i ulike lag. FHIR bussen er delt opp i tre lag; API laget, logikklaget og integrasjonslaget. API laget er et REST basert API utviklet ved hjelp av HAPI FHIR. Her finnes endepunkter for ressursene Observation, AllergyIntolerance, Condition, Patient, Practitioner, Organization, Location, Encounter, Medication, MedicationRequest og MedicationStatement. Disse endepunktene støtter ressursene både i XML og JSON format. Det finnes to ulike scenarioer; enten ønsker klienten å lagre data ellers så ønsker klienten å hente data. Flyten til scenario 1 kan leses av figuren ved å følge tallene i stigende rekkefølge og bokstaven «a». Totalt er det 6 steg; 1, 2a, 3a, 4a, 5a og 6. Flyten til scenario 2 kan leses av figuren ved å følge tallene i stigende rekkefølge og bokstaven «b». Totalt er det 6 steg; 1, 2b, 3b, 4b, 5b og 6.



Figur 4.3: Uansett om det ønskes å hente eller lagre data kommuniserer brukerne med et API lag i FHIR bussen. Kommunikasjon med konverteringsmodulen foregår i logikklaget i FHIR bussen. Integrasjonslaget i FHIR bussen håndterer kommunikasjonen med openEHR systemet. Konverteringsmodulen er delt opp i to lag; API laget og logikklaget. Det er i logikklaget i konverteringsmodulen hvor FHIR ressurs og openEHR komposisjon blir konvertert.

SCENARIO 1+2

(1): Først sendes en forespørsel fra klienten til API laget i FHIR bussen. Deretter blir informasjonen og forespørselen videresendt til logikklaget.

SCENARIO 1 - Lagring av data

(2a): Dersom klienten ønsker å lagre en ressurs sendes en forespørsel videre til konverteringskomponenten der FHIR bussen ønsker å få ressursen konvertert til passende openEHR komposisjon i ECISFLAT format.

(3a): Konverteringsmodulen bruker en database hvor alle konverteringsfiler er lagret. Disse konverteringsfilene forklarer hvordan en spesifikk arketype kan bli konvertert til en spesifikk ressurs, og motsatt.

(4a): Når FHIR ressursen er konvertert til en komposisjon sendes den tilbake til FHIR bussen.

(5a): FHIR bussen lagrer den konverterte komposisjonen i openEHR systemet, som responderer med en unik identifikator for komposisjonen dersom lagring var vellykket.

SCENARIO 2 - Henting av data

(2b): Dersom klienten ønsket å hente ut data sendes det først en forespørsel fra FHIR bus-

sen til openEHR systemet om å hente dataene med identifikator gitt av klienten. EtherCIS sender tilbake en openEHR komposisjon i ECISFLAT format.
(3b): FHIR bussen sender deretter en forespørsel til konverteringskomponenten om å konvertere openEHR komposisjonen over til en FHIR ressurs.
(4b): Komposisjonen blir deretter konvertert ved hjelp av en konverteringsfil.
(5b): Når openEHR komposisjonen er konvertert til en FHIR ressurs sendes den tilbake til FHIR bussen

SCENARIO 1+2
(6): FHIR bussen sender deretter tilbake til klienten en identifikator for ressursen som ble opprettet eller en FHIR ressurs dersom klienten ønsket å hente data.

4.3 Sikkerhetshåndtering

I masterprosjektet er det tatt en beslutning om å ikke fokusere på sikkerhetsaspektet ved både openEHR pasientjournalsystemet og FHIR bussen. Årsaken til dette er at det ble bestemt at vi bør først undersøke hvorvidt det er mulig å konvertere mellom openEHR og FHIR og hvordan dette lar seg gjøre. Når vi kommer frem til en mulig løsning for hvordan vi kan konvertere mellom openEHR arketyper og FHIR ressurser, kan vi senere fokusere på hvordan sikkerheten til dataene håndteres hos begge standardene. I dag eksisterer det en enkel autentisering med brukernavn og passord. Når en bruker er logget inn får brukeren tilbake en «sessionId» som sendes ved hver forespørsel til FHIR bussen. Siden FHIR bussen kun kommuniserer med et openEHR system, brukes samme brukernavn, passord og «sessionId» for både FHIR bussen og openEHR systemet.

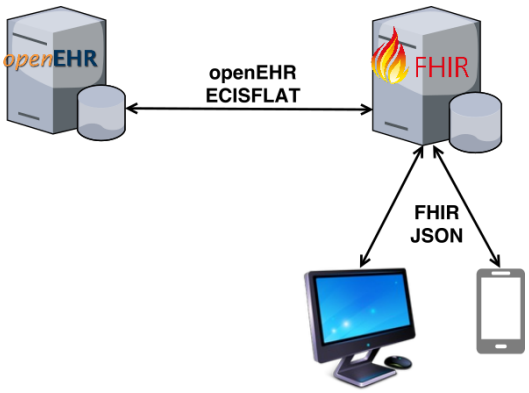
Kapittel 5

Konverteringsprosessen

openEHR og FHIR er to ulike helsestandarder som har valgt ulike fremgangsmåter for hvordan de skal strukturere dataene. openEHR er en datastandard som bruker to-nivå modellering for å strukturere og implementere tjenestene. FHIR er en meldingsstandard som har sine data strukturert i en flat modell med elementer. De kliniske konseptene i openEHR er strukturert ved hjelp av arketyper og FHIR representerer de kliniske konseptene i form av ressurser. Standardene har også ulike visjoner for hvor stor del av brukstilfellene de skal dekke. openEHR ønsker å utvikle arketyper for de viktigste kliniske konseptene og disse arketypene skal være strukturerte med alle relevante datapunkter for det kliniske konseptet. FHIR derimot har som mål at ressursene de tilbyr skal dekke cirka 80% av alle brukstilfellene og lar brukerne tilpasse ressursene ved hjelp av utvidelser.

Siden openEHR og FHIR benytter seg av ulike datastrukturer og dataformater, må det utføres en konvertering mellom disse standardene før en integrasjon er på plass. I masterprosjektet vil informasjonen sendt mellom klientene og FHIR bussen være strukturert som FHIR ressurser i JSON format, mens informasjonen mellom FHIR bussen og openEHR systemet er sendt som komposisjoner i ECISFLAT format. Figur 5.1 øverst på neste side viser et forenklet bilde av hvordan systemet i masterprosjektet er strukturert. Dette er en lignende figur som ble brukt i kapittel 4, og brukes igjen her for å vise hvor konverteringen må utføres og de ulike dataformatene som blir brukt.

Dette kapittelet presenterer en metode for å konvertere dataene mellom FHIR ressurser og openEHR arketyper. For hver konvertering opprettes det en konverteringsfil som beskriver hvordan dataelementene i en arketype kan konverteres til og fra dataelementene i en ressurs.



Figur 5.1: Brukerne sender og henter FHIR ressurser i JSON format fra FHIR bussen, mens arketypene blir sendt og hentet i ECISFLAT format. I tillegg til at dataene er strukturert i ulike dataformat, bruker dataene forskjellige datatyper og er generelt strukturert ulikt.

Selv om en arketype er tilgjengelig via biblioteker som `http://arketyper.no/ckm/` og `http://www.openehr.org/ckm/`, er det ulike stadier en arketype går gjennom før den vil være godkjent og «publisert». Fire av de mer kjente stadiene er *draft*, *Team Review*, *Review Suspended* og *Published*. Når de er blitt publisert har arketypene gått gjennom flere runder med vurdering og refaktorering, og er nærmere en korrekt arketype i forhold til en arketype i «draft» stadiet. I masterprosjektet er det fokusert på arketypene som er publiserte. Det finnes flest slike arketyper innenfor kategoriene OBSERVATION og EVALUATION, og det er disse arketypene som blir evaluert i masteroppgaven. Det er kun de obligatoriske feltene som er tatt med i konverteringen og evalueringen, og i dette kapittelet er det beskrevet hvordan en observasjonsarketype og to evalueringsarketyper kan konverteres til FHIR ressurser.

FHIR versjoner

Da arbeidet startet med å konvertere mellom FHIR ressursene og openEHR arketypene var det DSTU2 som var siste versjon av FHIR spesifikasjonen. På slutten av arbeidet kom en ny versjon ut (STU3). Siden versjonen ble utgitt så nærme leveringstid ble det tatt en beslutning å beholde forskningsarbeidet som er basert på DSTU2. STU3 har endret et par av FHIR ressursene som vi undersøker i dette kapittelet, men endringene er minimale og vil ikke endre vår endelig konklusjon. Selv om konverteringen som er dokumentert i dette kapittelet bruker DSTU2, er både FHIR bussen og tilhørende konverteringsfiler oppdatert til STU3 slik at medisinapplikasjonen som er implementert og dokumentert i kapittel 6 kan bruke en oppdatert FHIR tjeneste.

5.1 Arbeidsflyten for konverteringsprosessen

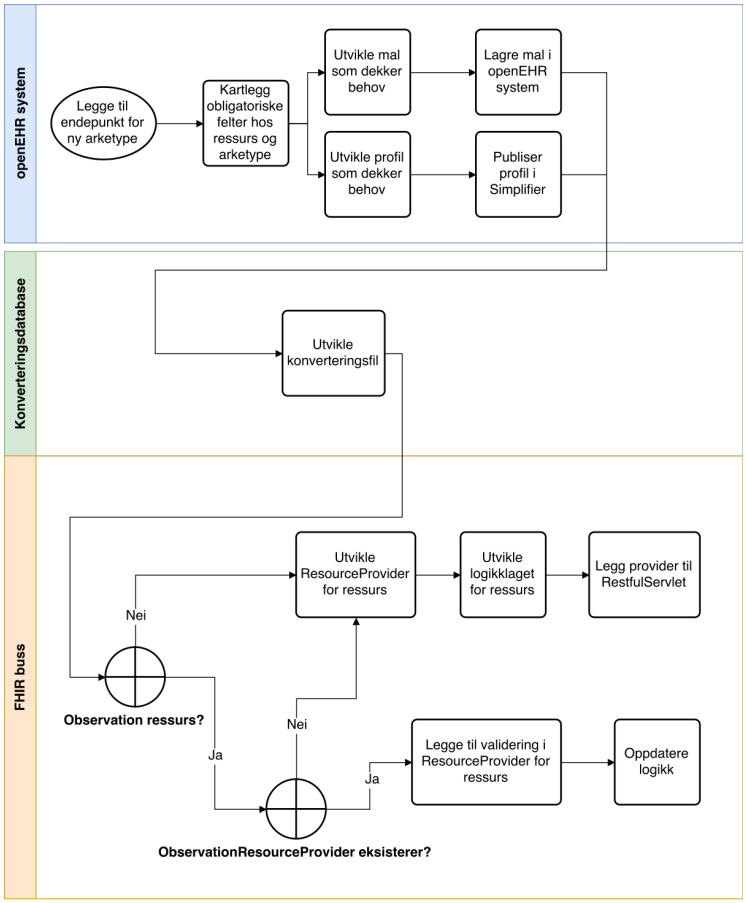
For hver ny arketype som en ønsker at FHIR bussen skal støtte, er det 5 hovedsteg som vi må utføre;

1. Kartlegge sammenheng mellom arketypen og ressursen som en ønsker å konvertere. I dette steget ser vi på hvilke obligatoriske felter som både arketypen og ressursen har og vurderer om det er nødvendig å enten 1) utvide ressursen under profileringen eller 2) utvide aktuell arketype med en ny arketype i et SLOT felt.
2. Opprette en openEHR mal og en FHIR profil som dekker behovene som ble avduket i steg 1.
3. Publisere openEHR malen i openEHR systemet og FHIR profilen i et register. Registeret kan for eksempel være Furore sitt register Simplifier.
4. Utvikle en konverteringsfil som baserer seg på feltene satt til obligatorisk i malen og profilen.
5. Utvikle endepunkt (ResourceProvider) i FHIR bussen med tilhørende logikk.

Hele prosessen kan studeres i arbeidsflytsmodellen i figur 5.2. Her er prosessen delt opp i tre lag; et for openEHR systemet, et for konverteringsmodulen og et for FHIR bussen. Utifra modellen er det endel arbeid som må utføres for hver arketype. I modellen er konverteringsmodulen beskrevet som en konverteringsdatabase. Årsaken til dette er at endringene skjer kun ved å legge til nye konverteringsfiler i databasen. Selve modulen er generisk og logikken er lik for hver arketype og ressurs. Ved å følge pilene i modellen kan en se hver prosess som må utføres før FHIR bussen har et fungerende endepunkt for en ny arketype. Så langt har oppgaven kun håndtert de obligatoriske feltene så godt som mulig, men det er ønskelig at FHIR bussen kan jevnlig utvide seg for å håndtere flere felter. Dette er mulig å gjøre ved å oppdatere FHIR profilen, openEHR malen og konverteringsfilene. Figur B.1 i Vedlegg B er utviklet for å gi en grafisk oversikt over hvordan dette vil fungere for det nåværende systemet.

5.2 Representasjon av observasjonskonsepter

En viktig jobb som sykepleiere og leger utfører hver dag er å *observere*. Allerede når pasienten kommer inn på legens kontor har legen utført en observasjon av pasienten. En slik observasjon er viktig for videre evaluering av pasienten. I tillegg til visuelle observasjoner



Figur 5.2: Modell over arbeidsflyten for å legge til nytt endepunkt for en ny arketype.

utføres det mer tekniske målinger som blodtrykk og urinprøver. openEHR og FHIR har begge elementer for å beskrive en observasjon. openEHR har detaljerte arketyper for hver observasjon og FHIR har en ressurs som kan brukes for de fleste observasjoner. I følgende avsnitt beskrives strukturen for openEHR observasjonsarketypene og FHIR observasjonsressursen.

5.2.1 openEHR OBSERVATION

openEHR har en egen ENTRY kategori for observasjon; OBSERVATION. Alle observasjonsarketypene arver feltene som OBSERVATION inneholder. Obligatorisk for hver observasjon er feltet *data* som inneholder informasjonen som ble observert. Denne er strukturert ved hjelp av datastrukturen HISTORY som inneholder valgfri antall hendelser av typen EVENT. En hendelse i en observasjon kan enten være en observasjon over tid (intervallhendelse) eller en observasjon på et gitt tidspunkt (punkthendelse). Dette beskriver openEHR ved hendelses-

instanser av typen INTERVAL_EVENT og POINT_EVENT henholdsvis. Observasjonen som beskrives i masteroppgaven er av typen punkthendelse. Dataene er lagret ved hjelp av en datastruktur av typen ITEM_STRUCTURE.

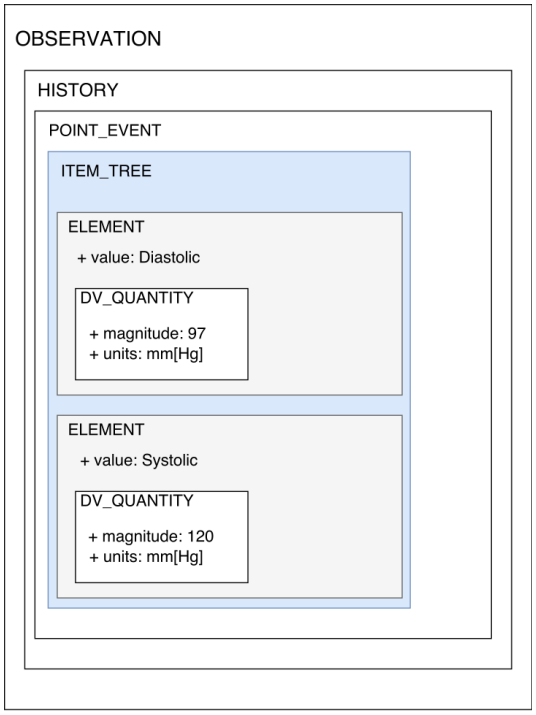
Et eksempel på en observasjon er måling av blodtrykk. openEHR har en egen arketype for blodtrykk som vi bruker videre i kapittelet. Dette er en punkthendelse hvor man finner den systoliske og diastoliske verdien. Listing 5.1 viser en XML versjon av punkthendelsen som er lagret i blodtrykk arketypen. Når en openEHR observasjon blir strukturert i XML format brukes egenskapen «xsi:type» for å spesifisere hvilken instans som XML elementet representerer. Linjene markert i gult viser hvordan instansene OBSERVATION, HISTORY, POINT_EVENT, ITEM_TREE og ELEMENT er spesifisert ved hjelp av «xsi:type».

Listing 5.1: Blodtrykk arketype i XML format

```
<content archetype_node_id="at0000" xsi:type="OBSERVATION">
  ...
  <data archetype_node_id="at0001" xsi:type="HISTORY">
    ...
    <events archetype_node_id="at0006" xsi:type="POINT_EVENT">
      <name>
        <value>any event</value>
      </name>
      <time xsi:type="DV_DATE_TIME">...</time>
      <data archetype_node_id="at0003" xsi:type="ITEM_TREE">
        <name>
          <value>blood pressure</value>
        </name>
        <items archetype_node_id="at0005" xsi:type="ELEMENT">
          <name>
            <value>Diastolic</value>
          </name>
          <value xsi:type="DV_QUANTITY">
            <magnitude>97</magnitude>
            <units>mm[Hg]</units>
          </value>
        </items>
        <items archetype_node_id="at0004" xsi:type="ELEMENT">
          <name>
            <value>Systolic</value>
          </name>
          <value xsi:type="DV_QUANTITY">
            <magnitude>120</magnitude>
            <units>mm[Hg]</units>
          </value>
        </items>
      </data>
    </events>
  </data>
</content>
```

En grafisk representasjon av XML dokumentet i listing 5.1 finnes i figur 5.3.

I tillegg til datafeltet hvor alle hendelsene er lagret kan en observasjon inneholde *protocol* og *state* informasjon. I «state» lagres informasjon som kan være viktig når observasjonen

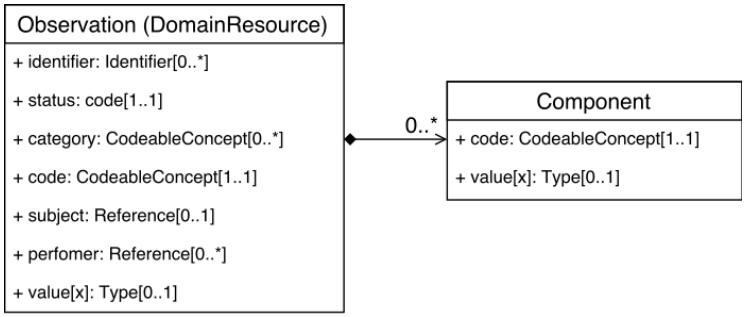


Figur 5.3: Grafisk representasjon av arketypen Blodtrykk. Alle observasjoner er strukturert ved hjelp av OBSERVATION, HISTORY og en type hendelse (punkt eller intervall). Hendelsen for blodtrykksmåling er representert ved hjelp av ITEM_TREE. Dette dataelementet inneholder den kliniske informasjonen til blodtrykket. Hver komponent i blodtrykket, den systoliske og diastoliske verdien, er representert i hvert sitt objekt av typen ELEMENT. Her finnes navnet og verdien til den systoliske og diastoliske komponenten.

skal tolkes og som ikke er tilgjengelig i journalen hvor observasjonen loggføres. Dette kan være informasjon om at pasienten nettopp hadde sovet eller trent før observasjonen ble tatt eller om pasienten var sittende eller stående under observasjonen.

5.2.2 FHIR Observation

FHIR har kun en ressurs, FHIR Observation, som kan tilpasses for alle observasjoner. Mer spesifiserte ressurser opprettes ved hjelp av profilering av FHIR Observation ressursen. En FHIR observasjon har kun to obligatoriske felter; *status* og *code*. Status er en FHIR kode som beskriver statusen til observasjonen og kan ha en av disse 8 kodene; *registered*, *preliminary*, *final*, *amended*, *corrected*, *cancelled*, *entered-in-error* og *unknown*. Feltet *code* beskriver hvilken type observasjon det er og kan for eksempel representeres ved hjelp av SNOMED koder. En forenklet modell av strukturen til FHIR Observation er vist i figur 5.4. I figuren er bare de viktigste egenskapene tatt med som blir diskutert videre i oppgaven. En kan se at en referanse til både pasienten (subject) og personen som utførte observasjonen (performer)



Figur 5.4: FHIR ressursen, *Observation*, er en generell type ressurs som kan brukes for å representere ulike observasjonskonsepter som blodtrykk, kroppsvekt og høyde. Figuren viser et utkast av hvilke felter som er tilgjengelig i ressursen. Feltene «status», «category» og «code» brukes for å beskrive hvilken observasjon ressursen representerer og om observasjonen er komplett.

er lagret i ressursen. Årsaken til dette er at FHIR ikke har lignende beholder som COMPOSITION som kan inneholde denne informasjonen. Et felt som ikke er obligatorisk, men nyttig å ha med for å sortere og klassifisere observasjoner, er *category* hvor en kan spesifisere hvilken type observasjon det er. Dette feltet kan ha en av disse kodene; *social-history*, *vital-signs*, *imaging*, *laboratory*, *procedure*, *survey*, *exam* og *therapy*. For mer informasjon om de tilgjengelige kategorikodene se FHIR spesifikasjonen i [69]. En større detaljert versjon av modellen med alle tilgjengelige felter finnes i figur B.2 i Vedlegg B.

Selve observasjonen kan enten bli lagret i et «value» felt eller ved hjelp av egne komponenter av typen «Component». Hvordan informasjonen blir representert er avhengig av hvilken observasjon en har. Dersom observasjonen har kun en enkel verdi, for eksempel observasjonen av kroppsvekt, kan denne informasjonen bli lagret ved hjelp av «value» feltet og datatypen «Quantity». Ikke alle observasjoner har bare en verdi og da må en opprette egne komponenter for hver verdi i observasjonen. Et eksempel for dette er blodtrykk, som har to verdier for måling; systolisk og diastolisk. En komponent består av et tilsvarende «value» felt og et ekstra felt, «code», som brukes for å beskrive hvilken komponent dette er. Dette kan enten være koder fra LOINC, SNOMED-CT eller andre terminologier.

For å representere blodtrykk som en ressurs lager vi en profilert versjon av FHIR Observation. Et utsnitt av hvordan denne profilering er utført ble tidligere vist i figur 2.10. Arketypen for blodtrykk i listing 5.1 er representert ved hjelp av en FHIR Observation ressurs i listing 5.2.

Listing 5.2: Utsnitt fra observasjonsressurs for blodtrykk

```
{
  "resourceType": "Observation",
  "subject": {
    "reference": "Patient/example"
  },
  "code": {
    "coding": [{
      "system": "http://snomed.info/sct",
      "code": "163020007",
      "display": "On examination-blood pressure reading (finding)"
     }]
  },
  "status": "final",
  "component": [
    {
      "code": {
        "coding": [
          {
            "system": "http://snomed.info/sct",
            "code": "163030003",
            "display": "On examination-Systolic blood pressure reading (finding)"
          },
        ]
      },
      "valueQuantity": {
        "value": 120,
        "unit": "mm[Hg]"
      }
    },
    {
      "code": {
        "coding": [
          {
            "system": "http://snomed.info/sct",
            "code": "163031004",
            "display": "On examination-Diastolic blood pressure reading (finding)"
          }
        ]
      },
      "valueQuantity": {
        "value": 97,
        "unit": "mm[Hg]"
      }
    }
  ]
  ...
}
```

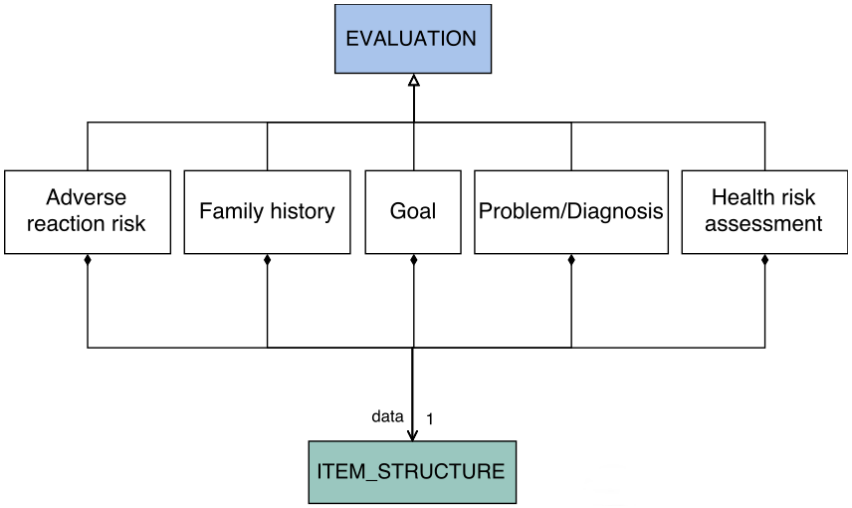
5.3 Representasjon av evalueringskonsepter

Evaluering er et viktig klinisk konsept. Å evaluere er nødvendig for å utføre videre behandlinger, diagnosering og forhindre alvorlige sykdommer. En evaluering kan være et resultat av en observasjon, for eksempel en konklusjon om at en pasient har ulike allergier. Det kan også være innhenting av relevant informasjon om familien til pasienten som kan være med på å avgjøre en risiko for en sykdom. Videre finner vi evaluering om mulige diagno-

ser og bivirkninger. I følgende avsnitt vises det til hvordan openEHR og FHIR har valgt å strukturere slike konsepter ved hjelp av arketyper og ressurser henholdsvis.

5.3.1 openEHR EVALUATION

I openEHR er evalueringskonseptene representert ved hjelp av klassen *EVALUATION* som er en underklasse av ENTRY og CARE_ENTRY. EVALUATION inneholder kun ett data felt som er av typen ITEM_STRUCTURE som vises i figur 5.5. ITEM_STRUCTURE

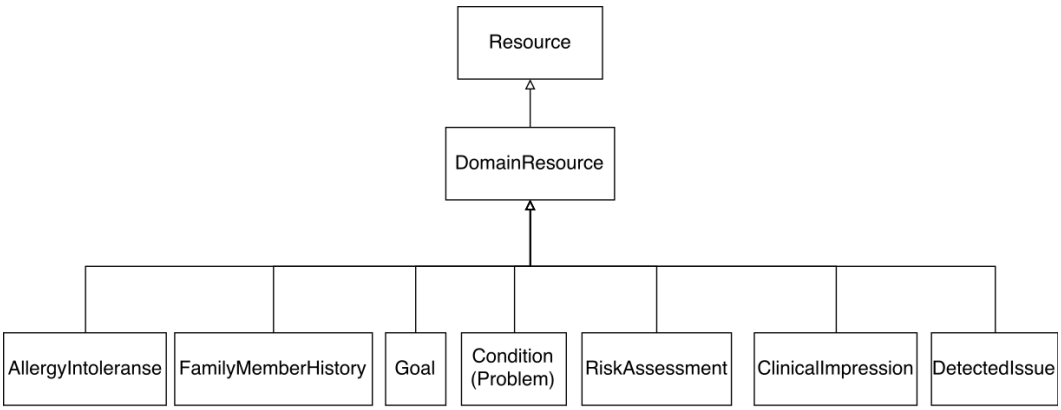


Figur 5.5: Oversikt over hvordan evalueringsarketypene i openEHR er strukturert. Figuren viser bare et lite utvalg av de tilgjengelige evalueringsarketypene. Alle arketypene arver fra klassen EVALUATION og har datafeltet av typen ITEM_STRUCTURE.

er tidligere forklart i avsnitt 2.3.2. I tillegg arves de ulike feltene som ENTRY og CARE_ENTRY inneholder. I en evaluering er det ikke nødvendig med tidsinformasjon slik som i en observasjon. Dette er fordi det er ikke assosiert en spesifikk tid for når en evaluering blir opprettet. Tiden som kan være interessant er tidspunktet for pasientkonsultasjonen hvor evalueringen ble ferdiggjort. Denne informasjonen blir lagret i *COMPOSITION.event_context.start_time* og *COMPOSITION.event_context.end_time*. En annen tidsinformasjon som kan være interessant er tidspunktet for når evalueringen ble lagret i journalen. Dette tidspunktet finnes i *VERSION.commit_audit* og blir satt når evalueringen blir lagret i en komposisjon i journalen. Feltet *provider* finnes i ENTRY klassen og er som regel alltid legen til pasienten. Feltet *protocol* som er arvet av CARE_ENTRY kan brukes om ønskelig for å gi informasjon om hvordan en evaluering ble utført.

5.3.2 FHIR Evaluation

FHIR har ingen generelle ressurser som kan dekke de mest aktuelle evalueringskonseptene slik de gjorde med Observation ressursen. Det ble da tatt en avgjørelse i masterprosjektet om å vurdere hvordan en kan konvertere et utvalgt av de ulike ressursene for evaluering som FHIR har i dag. I DSTU2 har de 7 ulike ressurser som fungerer som en evalueringsressurs; *AllergyIntolerance* for informasjon om allergier, *Condition (Problem)* for informasjon om problemer, diagnoser og sykdom, *ClinicalImpression* for evaluering av pasientens tilstand og evaluering av videre behandlinger, *FamilyMemberHistory* for relevant helseinformasjon om en nær slektning til en pasient, *RiskAssessment* for evaluering av mulige fremtidige risikoer, *Goal* for informasjon om planlagt mål for pasienten etter en viss tid med behandling eller livsstilsendring, og *DetectedIssue* for informasjon om mulige problemer og risikoer med eksisterende/fremtidige kliniske aktiviteter. Den siste ressursen kan inneholde mulige risikoer med to medisiner som blir tatt samtidig, feildosering, problemer med hvor ofte eller hvordan en medisin blir tatt, og mulige risikoer med to ulike behandlinger som utføres samtidig. Et diagram av FHIR evalueringene er vist i figur 5.6.



Figur 5.6: Modelldiagram med de ulike FHIR ressursene som dekker ulike evalueringskonsepter, med tilhørende foreldreressurser. Siden FHIR har en flat strukturering av ressursene arver alle ressursene fra DomainResource.

For hver ressurs er det funnet en tilhørende arketype som skal korrespondere med ressursen. I tabell 5.1 kan en se de 7 ulike FHIR ressurser for evaluering, med tilhørende openEHR arketyper som skal dekke de samme kliniske konseptene.

Ressursen FamilyMemberHistory er for én slektning, mens openEHR arketypen Family history dekker alle relevante slektninger. Dermed må det brukes en liste av FamilyMemberHistory ressurser for å representere arketypen. Siden det ikke eksisterer en felles klinisk metamodell som både FHIR og openEHR baserer ressursene og arketypene på, vil det opp-

openEHR Evaluation arketype	**HL7 FHIR ressurs**
Adverse reaction risk	AllergyIntolerance
Family history	List<FamilyMemberHistory>
Goal	Goal
Problem/Diagnosis	Condition (Problem)
Health risk assessment	RiskAssessment
-	ClinicalImpression
-	DetectedIssue

Tabell 5.1: Her vises det i høyre kolonne 7 ulike ressurser som FHIR tilbyr innenfor evaluering. Venstre kolonne viser de ulike openEHR arketypene som skal dekke samme konsept. De to siste ressursene, ClinicalImpression og DetectedIssue, har ingen korresponderende arketyper og det er markert med en strek.

stå situasjoner der det ikke finnes en arketype for en ressurs, og motsatt. I tabellen vises et slikt tilfelle der FHIR har to ressurser, ClinicalImpression og DetectedIssue, som ikke har tilhørende arketyper i openEHR. openEHR har også flere arketyper for evaluering enn de 5 som er listet i tabellen, men de er ikke tatt med i masteroppgaven. Videre i masteroppgaven blir kun ressursene AllergyIntolerance og Condition(Problem) vurdert og dokumentert. Årsaken til dette er at konvertering, profilering og oppretting av mal vil ha lignende struktur og lik fremgangsmåte uansett hvilken evalueringsressurs som vurderes. Dermed er bare disse to tatt med i vurderingen for å unngå gjentagelse av arbeid. Det er allikevel kartlagt likheter mellom alle de 5 ressursene og tilhørende arketyper i Vedlegg B.

5.4 Kartlegging av likheter

I første steg av konverteringen kartlegges de ulike obligatoriske feltene for både arketypen og ressursen. I neste steg opprettes det en profilert versjon av den aktuelle FHIR ressursen og en mal til openEHR arketypen. I konverteringen er det kun det obligatoriske data-feltet som vurderes i arketypene, og protocol feltet fjernes når en mal blir opprettet.

5.4.1 Blodtrykk

For å representere en ressurs som et blodtrykk profilerer vi en FHIR Observation ressurs. I openEHR arketypen for blodtrykk er ingen felter obligatoriske, men for å representere et

blodtrykk ønsker vi at den systoliske og den diastoliske verdien skal være med. Disse to verdiene er representert i arketypen ved hjelp av datatypen «Quantity», og er bundet til hver sine SNOMED CT koder i ontologiseksjonen i ADL filen. Arketypen har også et felt for å representere tidspunktet for når blodtrykket ble målt. I FHIR Observation vil den systoliske og den diastoliske verdien være representert av hver sine komponenter i listen *component*. Hver komponent har et verdifelt og et kodefelt. Verdifeltet har ingen spesifisert datatype, men ved hjelp av profilering kan vi spesifisere datatypen til å være «Quantity». Kodefeltet kan være en kode fra hvilken som helst terminologi, og i dette tilfellet vil vi at koden er samme SNOMED kode som er spesifisert for blodtrykksarketypen. Ressursen har et felt, *effectiveDateTime*, som beskriver når blodtrykket ble målt. Ressursen har også en referanse til en pasient, *subject*, for å beskrive hvilken pasient blodtrykket tilhører. Denne pasientidentifikasjonen vil være lagret utenfor arketypen for blodtrykk i selve komposisjonen.

5.4.2 Allergi

AllergyIntolerance er en ressurs som kan brukes for å evaluere både allergier og intoleranser mot ulike stoffer. I openEHR arketypen er det kun feltet *substance* som er obligatorisk, som er et felt for å beskrive stoffet som pasienten kan være eller er allergisk mot. I FHIR ressursen er det både feltet *substance* og *patient* som er obligatorisk. En referanse til pasienten er obligatorisk fordi ressursen eksisterer alene, og ikke i en journal slik som openEHR arketypene gjør. Det er flere felter som er valgfri å bruke, blant annet for å spesifisere siste tidspunkt en reaksjon ble observert, hvor kritisk reaksjonen er, status til evalueringen og andre notater som kan være relevante for tolkning av evalueringen. Hele ressursen og arketypen vises i figuren B.3 i Vedlegg B.

5.4.3 Diagnose

Å loggføre et problem eller en diagnose er en viktig del av evalueringen av en pasient. I openEHR finnes det en arketype, *Problem/Diagnosis*, hvor en kan beskrive slike evalueringer. Eneste obligatoriske feltet er navnet på problemet/diagnosen; *Problem/Diagnosis name*. Ellers finnes det blant annet informasjon om hvor på kroppen problemet er identifisert (dersom det er en spesifikk del av kroppen), relevant datoer, alvorlighetsgrad og tidspunkt for når problemet ble løst. FHIR har ressursen *Condition (Problem)* som har tre obligatoriske felter; *patient*, *code* og *verificationStatus*. «patient» er feltet hvor en referanse

til en pasient spesifiseres og denne identifikasjonen finnes i «subject» feltet i komposisjonen hvor arketypen er lagret i openEHR. «code» er feltet hvor en kodet versjon av problemet er lagret, som gjenspeiles i openEHR sitt obligatoriske felt «Problem/Diagnosis name». «verificationStatus» er et felt som brukes for å verifisere om en antatt diagnose eller problem er verifisert, om diagnosen fremdeles er under evaluering eller om det er en feildiagnosering. openEHR har et felt hvor statusen til diagnoseringen er lagret; *Status*. Dette er et SLOT felt hvor en kan bruke arketypen *Problem/Diagnosis status*. Arketypen er en klyngearketype for å gi detaljerte beskrivelser av en status for et spesifikk problem eller diagnose. Klyngen har ingen obligatoriske felter, men et valgfri felt, «Diagnostic status», som kan bli brukt for å lagre vurderingen i FHIR feltet «verificationStatus». Det er allikevel et problem med å konvertere fra «Problem/Diagnosis status» til «verificationStatus». Disse to bruker to ulike terminologier for å beskrive statusen. Her må enten en av standardene forandres (FHIR ressursen kan tillate SNOMED-CT koder) eller så må verifikasjonskoden settes til «unknown» i ressursen. FHIR ressursen har også andre egenskaper som kan være relevant for beskrivelsen av et problem eller diagnose, men disse er det valgfritt å bruke. Blant annet finnes det dato for når evalueringen av diagnosen eller problemet ble lagret, alvorlighetsgrad, første tidspunkt hvor en hypotese om diagnosen ble utført, hvor på kroppen problemet eventuelt er identifisert og andre notater som kan være relevante. Strukturen til arketypen og ressursen finnes i figur B.6 i Vedlegg B.

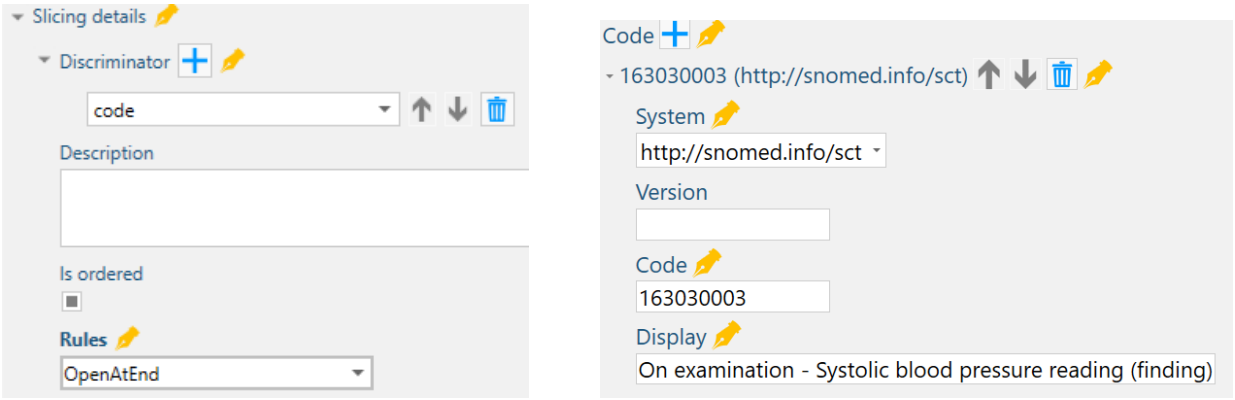
5.5 Profilering av FHIR ressurser

FHIR tilbyr et verktøy for å tilpasse basisressursene for et spesifikk brukstilfelle ved hjelp av profilerte ressurser. En profilert ressurs er en ressurs som er tilpasset ved å enten tilføre nye felter til basisressursen, dele opp eksisterende lister i mindre lister (se avsnitt 2.4.2) eller endre kardinaliteten til et eksisterende felt. Vår konverteringsmetode baserer seg på å opprette profilerte ressurser for hver openEHR arketype som FHIR bussen skal støtte, som vil både bruke oppdeling av lister og utvidelser der det trengs. En profil lar oss spesifisere ovenfor brukerne hvilke felter som en må fylle ut ved oppretting av data eller hvilke felter en kan forvente å få i en respons ved henting av data.

5.5.1 Profilering av Observation

Noen observasjonsarketyper inneholder kun ett verdifelt og da opprettes det en profilert ressurs hvor verdifeltet i ressursen settes til obligatorisk. For de observasjonsarketypene

som har flere verdifelter, for eksempel blodtrykk, fjernes verdifeltet i basisressursen og listen med komponenter settes til obligatorisk. Listen med komponenter deles opp i underlister som tilhører de ulike verdifeltene i arketypen, se figur 2.10. For blodtrykksprofilen blir komponentlisten delt opp i to underlister for den systoliske og diastoliske verdien. En kan ved hjelp av verktøyet Forge spesifisere et «discriminator» felt, se figur 5.7a, og i denne profilen peker dette feltet på «code» verdien i hver komponent da disse vil være unike. I



(a) *discriminator* felt

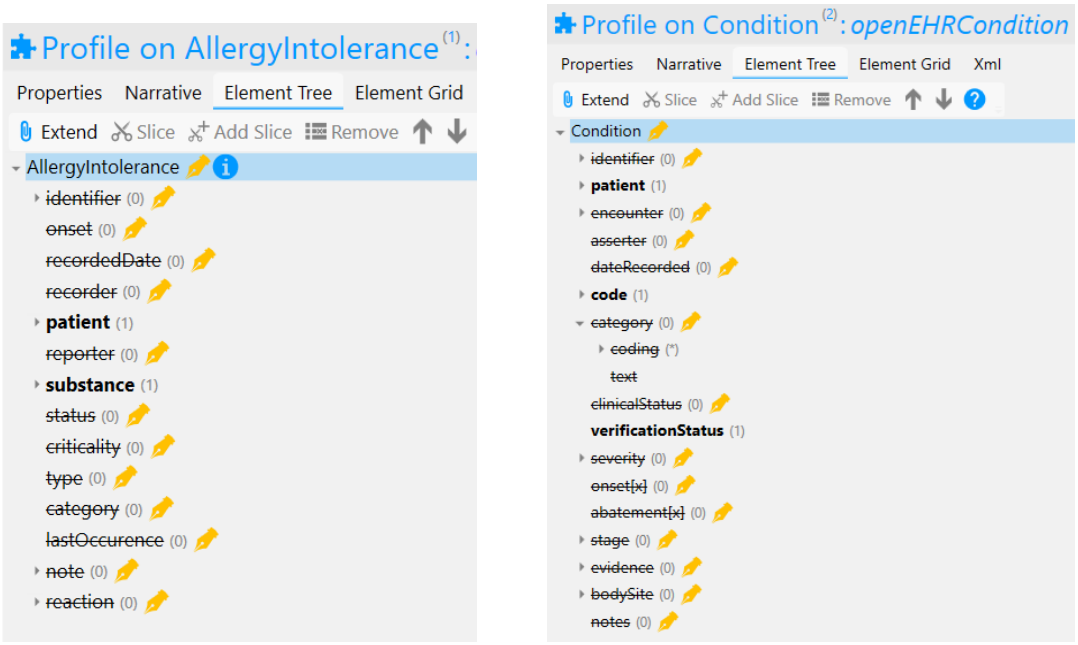
(b) SNOMED-CT kode for systolisk komponent

Figur 5.7: Utsnitt fra Forge hvor en *discriminator* er spesifisert og en SNOMED kode for systolisk verdi er satt i en profil for blodtrykk. Siden «discriminator» feltet er satt til «code», vil systemet bruke «code» feltet for å differensiere komponentene.

profilen er det også satt en fast verdi i «code» feltet til den systoliske og diastoliske verdien. Figur 5.7b viser hvordan en setter en fast kode fra en terminologi i Forge. For utsnitt av hvordan hele observasjonsprofilen ser ut for blodtrykk, se figuren A.1 i Vedlegg A.

5.5.2 Profilering av AllergyIntolerance og Condition(Problem)

Vi ønsker å opprette FHIR profiler av ressursene AllergyIntolerance og Condition. For begge ressursene ble kun de obligatoriske feltene tatt med. FHIR feltene som også vil eksistere i en openEHR mal er «substance» for AllergyIntolerance ressursen og «code» i Condition ressursen. Det var ingen obligatoriske felter i tilhørende arketype som ikke allerede var obligatoriske i ressursen. Figur 5.8 viser et utsnitt fra de to profilene som ble utviklet ved hjelp av verktøyet Forge og som deretter ble publisert i registeret Simplifier. Det var ikke nødvendig å dele opp lister i disse ressursene for å dekke de obligatoriske feltene i openEHR arketypen, og dermed ble det ikke spesifisert en «discriminator» for profilene.



(a) AllergyIntolerance profil

(b) Condition (Problem) profil

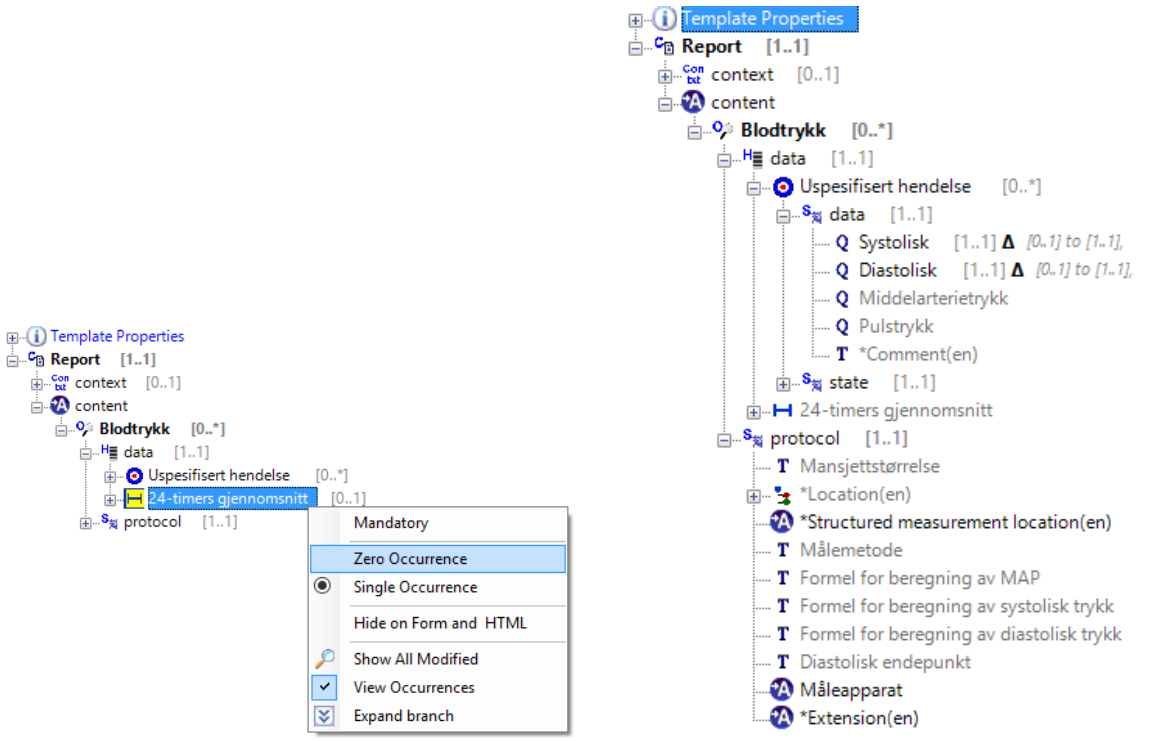
Figur 5.8: Utsnitt fra Forge hvor ressursene AllergyIntolerance og Condition (Problem) er profilert. Feltene som er strøket ut er fjernet, mens feltene som har fet skrift er satt til obligatorisk.

5.6 Oppretting av openEHR maler

For å kunne lagre observasjonsarketypen for blodtrykk og evalueringsressursene for allergi og diagnose må vi opprette tilsvarende openEHR maler som inneholder alle obligatoriske feltene som finnes i tilsvarende openEHR arketype og FHIR ressurs. I følgende avsnitt skal vi se på hvordan vi opprettet tre ulike maler for blodtrykk, allergi og diagnose.

5.6.1 Oppretting av mal for arketypen Blodtrykk

Som beskrevet i 2.3.2 lagres arketypen for blodtrykk i en rapportsarketype. Når arketypen for blodtrykk er lagt til i innholdsfeltet kan en spesifisere hvilke felter som en ønsker å begrense. I blodtrykk fjernes alle underfeltene til «protocol» feltet. I «data» feltet er det to ulike hendelser; «Uspesifisert hendelse» og «24-timers gjennomsnitt». I dette tilfellet ønskes det en «uspesifisert hendelse» og dermed fjernes «24-timers gjennomsnitt» feltet. Template Designer er et interaktivt verktøy hvor en høyreklikker på det aktuelle feltet en ønsker å endre og deretter kan en velge «Zero Occurrence» for å fjerne feltet, som vist i figur 5.9a.



(a) Feltet kan fjernes («Zero Occurrence») eller være obligatorisk («Single Occurrence»).

(b) En openEHR mal for blodtrykk. Alle feltene unntatt «Systolisk» og «Diastolisk» er fjernet (lysgrå farge).

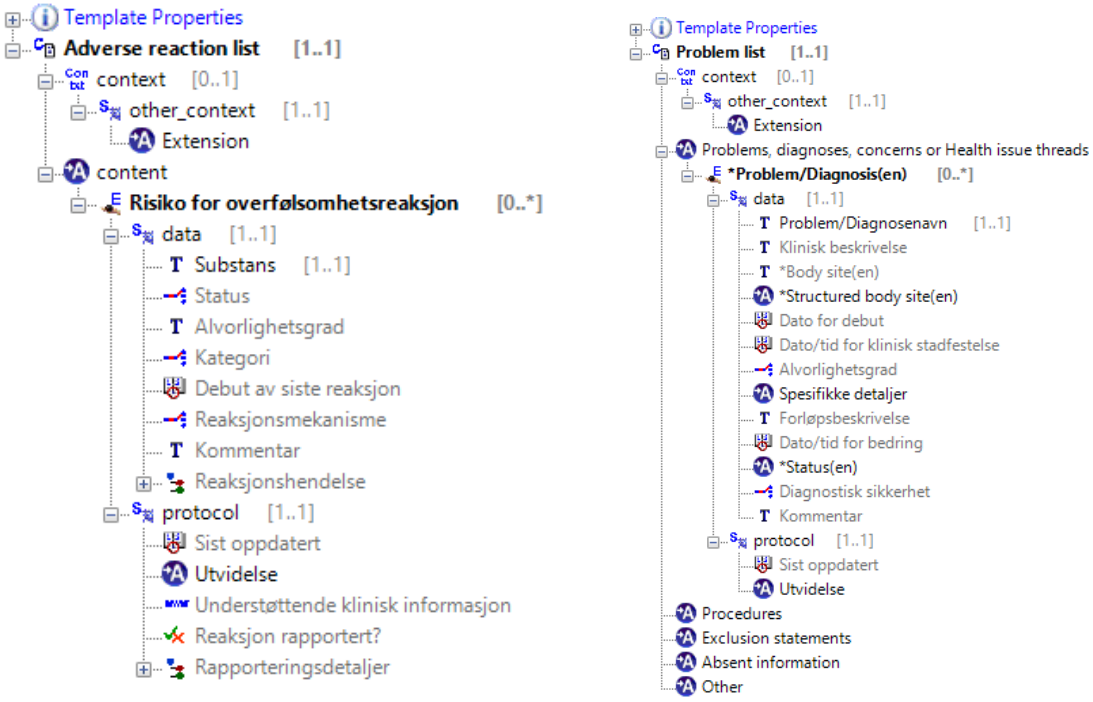
Figur 5.9: Utsnitt av Template Designer for å opprette en mal for blodtrykk

Hendelsen «Uspesifisert hendelse» har to felter, *data* og *state*. Alle underfeltene til «state» er valgfri og disse fjernes. Feltet «data» har fem underfelter; *Systolisk*, *Diastolisk*, *Middelarterietrykk*, *Pulstrykk* og **Comment(en)*. Det systoliske og diastoliske feltet settes til obligatorisk, mens resterende felter fjernes. Feltene som er fjernet vises ved hjelp av en lysere grå farge. Når en setter et felt til obligatorisk kan en se bak feltet hvilken endring som ble gjort. For det systoliske og diastoliske feltet ble kardinaliteten endret fra «0..1» til «1..1». Figur 5.9b er et utsnitt av hovedfeltene i malen for blodtrykk, hvor en kan se feltene som er fjernet i en lysegrå farge, og det systoliske og diastoliske feltet som i malen er obligatorisk.

5.6.2 Oppretting av maler for arketypene «Adverse reaction risk» og «Problem/Diagnosis»

Arketypen Adverse reaction risk og Problem/Diagnosis hadde egne komposisjonsarketyper som rotnode. Adverse reaction risk er representert ved hjelp av komposisjonen «Adver-

se reaction list» som er en komposisjonsarketype som brukes her for å lagre en eller flere allergier og/eller intoleranser hos en pasient. På norsk kalles arketypen for *Risiko for overfølsomhetsreaksjon*. Alle feltene unntatt «Substans» er fjernet (lysgrå farge), se figur 5.10a.



(a) «Adverse reaction risk» mal

(b) «Problem/Diagnosis» mal

Figur 5.10: Utsnitt fra mal utviklet for arketypene «Risiko for overfølsomhetsreaksjon» og «Problem/Diagnosis» i verktøyet Template Designer. Feltene i lysgrå farge er fjernet. I arketypen «Risiko for overfølsomhetsreaksjon» gjenstår bare feltet «Substans», mens i arketypen «Problem/Diagnosis» gjenstår feltet «Problem/Diagnosenavn».

Arketypen Problem/Diagnosis har ingen norsk oversettelse og blir representert som **problem/Diagnosis(en)*. Denne arketypen er representert ved hjelp av komposisjonen «Problem list». Komposisjonen er en arketype som i masteroppgaven blir brukt for å representere en liste med ulike diagnoser og problemer som en pasient kan ha. I malen er det feltet «Problem/Diagnosenavn» som er obligatorisk, mens resten er fjernet. Det er mulig å legge til arketypen «Problem/Diagnosis status» i «*Status(en)» feltet for å kunne konvertere verifikasjonsstatusen på en korrekt måte ved en senere anledning.

5.7 Strukturering av konverteringfil

For hver openEHR arketype eksisterer det en konverteringsfil som brukes under konverteringen. Konverteringsfilen beskriver regler for hvilke dataelementer som skal konverteres og hvordan de skal konverteres mellom standardene. Konverteringsfilene er strukturert ved hjelp av en liste med objekter, heretter kalt konverteringsobjekter. Hvert konverteringsobjekt blir prosessert hver for seg. Konverteringsfilen er basert på arbeidet utført i [62] hvor stien til HL7 CDA dataelementene ble brukt for konvertering over til HL7 FHIR ressurser. Uansett arketype vil hvert konverteringsobjekt inneholde tre felter; *fhirElements*, *openehrPaths* og *openehrType*. En tom konverteringsfil vises i listing 5.3.

Listing 5.3: En tom konverteringsfil

```
1 { "mappings": [
2     {
3       "fhirElements": [],
4       "openehrPaths": [],
5       "openehrType": ""
6     }
7 ]}
```

«fhirElements» er en liste med dataelementer i FHIR ressursen som skal konverteres når konverteringsobjektet blir prosessert. Hvert dataelement er representert ved hjelp av en sti på formatet «**myAddress[0].city**». Eksempelet viser hvordan en kan hente ut navnet på byen i første adresse til en pasient.

«openehrPaths» er en liste med XPATH stier for en arketype i ECISFLAT format. En XPATH sti som bruker kodene funnet i ADL representasjonen til arketypen vil ha lignende struktur: **"/content[openEHR-EHR-ACTION.medication.v0]/description[at0017]/items[at0020]|value"**. Ved hjelp av denne stien kan en hente ut navnet på en medisin. ECISFLAT formatet er dokumentert i GitHub biblioteket til Ethercis; `https://github.com/ethercis/ethercis/blob/master/doc/flat%20json.md`.

Når vi ønsker å konvertere fra et datafelt i en openEHR arketype til et datafelt i en FHIR ressurs hentes openEHR elementet ved hjelp av XPATH stien(e) i «openehrPaths». Deretter brukes «fhirElements» for å spesifisere hvor elementet skal lagres i FHIR ressursen. Samme logikk brukes også når en ønsker å konvertere fra en FHIR ressurs til en openEHR arketype. Siden feltene i arketypene og ressursene kan ha ulike datatyper og ulike dataformater er det tilgjengelig et «openehrType» felt. Dette feltet brukes for å spesifisere hvordan konverteringen mellom feltene skal utføres. Dersom «openehrType» er lik «quantity» vet

systemet at ved å bruke openEHR stien vil systemet få ut en verdi i formatet «magnitude,unit». Systemet kan da verifisere at konverteringsfilen er korrekt strukturert ved å sjekke at «fhirElements» har to elementer; et for å lagre verdien («magnitude») og et for å lagre enheten («unit»). Det finnes også mer avanserte konverteringer som må gjøres, for eksempel konvertere dataverdier (fra «male» til «man»), og dette kan også spesifiseres i konverteringsfilen. Kapittel 6 presenterer en applikasjon for medisinadministrering hvor slike avanserte konverteringer utføres, og dermed vil ikke dette kapittelet beskrive slike konverteringer. I Vedlegg B finnes to beslutningstre som viser logikken bak hvordan «openehrType» brukes for å konvertere mellom en komposisjon og en ressurs.

5.7.1 Konverteringsfil for blodtrykk

Etter vi har kartlagt de obligatoriske feltene for arketypen og ressursen for blodtrykk, beskriver vi i en konverteringsfil hvordan feltene skal konverteres. Eksemplene i dette avsnittet baserer seg på arketypen Blodtrykk vist i listing 5.1. For blodtrykk ønsker vi at den systoliske og den diastoliske verdien, i lag med et tidspunkt for når observasjonen ble utført, skal bli konvertert. Ved hjelp av XML arketypen definert i listing 5.1 finner vi blant annet stien for å hente ut den systoliske verdien;
/content[at0000]/data[at0001]/events[at0006]/data[at0003]/items[at0004]/value/magnitude.
Dette er samme sti som vil bli brukt dersom en skal definere en AQL spørring ved henting av data fra et openEHR basert system. I listing 5.4 vises et lite utsnitt av selve konverteringsfilen, hvor en FHIR sti og en openEHR sti i ECISFLAT format er bundet sammen.

Listing 5.4: Utsnitt av konverteringsfilen for blodtrykk mellom FHIR og openEHR

```
1  {
2    "mappings": [
3      {
4        "fhirElements": [
5          "component[0].value.value.myStringValue",
6          "component[0].value.unit.myStringValue"
7        ],
8        "openehrPaths": [
9          "/content[openEHR-EHR-OBSERVATION.blood_pressure.v1]/data[at0001]/events[
               at0006]/data[at0003]/items[at0004]|value"
10       ],
11       "openehrType": "quantity"
12     },
13     {
14       "fhirElements": [
15         "component[1].value.value.myStringValue",
16         "component[1].value.unit.myStringValue"
17       ],
18       "openehrPaths": [
19         "/content[openEHR-EHR-OBSERVATION.blood_pressure.v1]/data[at0001]/events[
               at0006]/data[at0003]/items[at0005]|value"
20       ],
```

```
21          "openehrType": "quantity"
22      }
23      ...]
24  }
```

5.7.2 Konverteringsfil for allergi

I konverteringsfilen, listing 5.5 for ressursen AllergyIntolerance og arketypen «Risiko for overfølsomhetsreaksjon» (Adverse reaction risk), er det ett obligatorisk felt som er beskrevet. I ressursen er dette feltet «substance», som er representert ved hjelp av datatypen CodeableConcept. Dette vil si at feltet vil ha et terminologisystem («system»), en terminologikode («code») og et beskrivende felt («display»). I openEHR er et slikt felt representert av datatypen DvCodedText. I ECISFLAT formatet er disse tre FHIR-feltene satt sammen til ett felt; «terminology::code|value|». Dermed vil openEHR stien i konverteringsfilen kun ha en sti til «value» feltet av typen DvCodedText.

Listing 5.5: Utsnitt av konverteringsfilen for ressursen AllergyIntolerance og arketypen Adverse reaction risk

```
1  {
2    "mappings": [
3      {
4        "fhirElements": [
5          "substance.coding[0].system.myStringValue",
6          "substance.coding[0].code.myStringValue",
7          "substance.coding[0].display.myStringValue"
8        ],
9        "openehrPaths": [
10         "/content[openEHR-EHR-EVALUATION.adverse_reaction_risk.v1]/data[at0001]/
               items[at0002]|value"
11       ],
12       "openehrType": "codedtext"
13     }
14   ]
15 }
```

5.7.3 Konverteringsfil for diagnose

I konverteringsfilen, listing 5.6 for ressursen Condition og arketypen Problem/Diagnosis, er det også kun ett felt som er obligatorisk; «code» i ressursen og «Problem/Diagnosis name» i arketypen. Disse feltene har lik datatype som feltet «substance» beskrevet i forrige avsnitt. Dermed vil konverteringsfilen ha en lignende struktur, men med ulik openEHR sti og ulike FHIR felter. Konverteringen mellom denne ressursen og arketypen vil ikke være optimal. Årsaken til dette er at ressursen har fremdeles et obligatorisk felt, «verificationStatus», som vi ikke har fått konvertert. Som beskrevet i avsnitt 5.4.3 kan det tenkes at

verifikasjonskoden i FHIR ressursen kommer fra «Problem/Diagnosis status» arketypen som kan brukes i «status-SLOT» feltet i arketypen Problem/Diagnosis. Selv om begge feltene bruker terminologi for å beskrive statusen til diagnosen så brukes det ulike terminologier. I FHIR ressursen er terminologien *Required* (se avsnitt 2.4.2) som vil si at det ikke er mulig å bruke andre terminologisystemer. I openEHR arketypen er det brukt fire lokale koder som er bundet til ulike SNOMED CT koder. I masterprosjektet er dermed verifikasjonskoden i FHIR ressursen satt til «unknown».

Listing 5.6: Utsnitt av konverteringsfilen for ressursen Condition(Problem) og arketypen Problem/Diagnosis

```
{
  "mappings":[
        {
      "fhirElements": [
        "code.coding[0].system.myStringValue",
        "code.coding[0].code.myStringValue",
        "code.coding[0].display.myStringValue"
      ],
      "openehrPaths": [
        "/content[openEHR-EHR-EVALUATION.problem_diagnosis.v1]/data[at0001]/items[
            at0002]|value"
      ],
      "openehrType": "codedtext"
    }
  ]
}
```

Kapittel 6

Integrering av medisinapplikasjonen LEMI

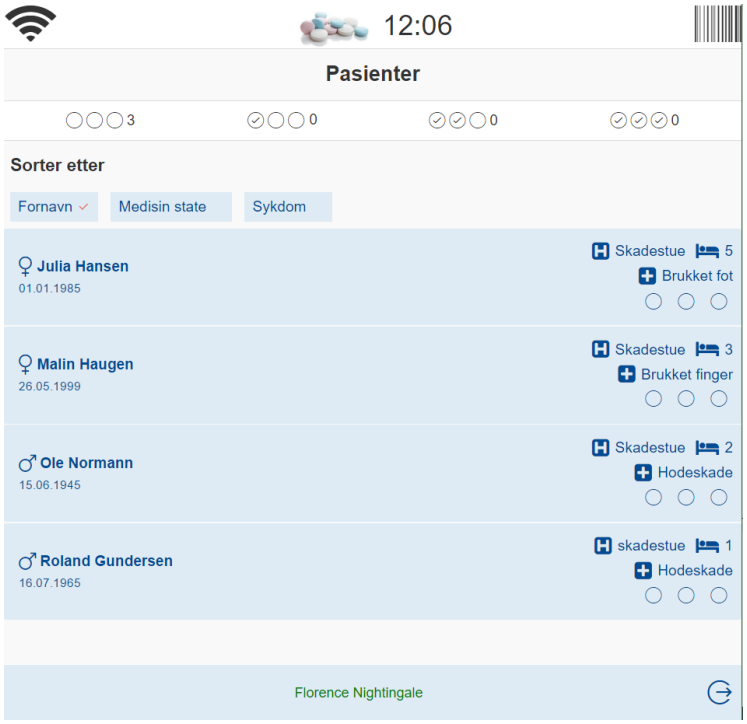
En viktig oppgave sykepleiere og hjelpepleiere utfører hver dag er å gi medisiner til pasienter. Her er det viktig at rett pasient får rett medisin til rett tid. En sykepleiers hverdag er både hektisk og krevende. Medisinadministrering er en oppgave hvor det er lett å gjøre feil, og dessverre ser en i dag at feilmedisinering er vanlig [70]. En pasient kan få feil medisin eller feil dose. Medisinen kan bli gitt på feil tidspunkt eller ikke bli gitt i det hele tatt.

I samarbeid med Helse Vest IKT, Helse Bergen og Høgskulen på Vestlandet ble det utviklet en applikasjon for medisinadministrering. Brukerhistorien som applikasjonen er basert på er gitt i kapittel 3 og i dette kapittelet vil vi dokumentere applikasjonen som er et verktøy brukt i en medisinrunde. Applikasjonen vil også være integrert mot FHIR bussen som er utviklet i masterprosjektet. FHIR bussen og openEHR systemet vil simulere bakenforliggende systemer som blant annet pasientjournalsystemet DIPS for pasientlister og kurvesystemet Meona for medisiner og medisinhåndtering.

6.1 LEMI - LEgeMIddelapp

Applikasjonen LEMI for administrering av medisiner ble utviklet av en tidligere doktorgradsstudent, Xiaoliang Wang, ved Høgskulen på Vestlandet. Denne applikasjonen brukte da hardkodete data for pasienter og medisiner, og manglet integrasjon mot bakenforliggende systemer for testdata. Applikasjonen er en modifisert versjon av en blodtransfusjonsappli-

kasjon som er dokumentert i studentens doktorgradsoppgave «Towards Correct Modelling and Model Transformation in DPF» [71]. LEMI er en medisinapplikasjon som er tenkt å kunne brukes av sykepleiere som skal ta den daglige medisinrunden til pasientene. I dette masterprosjektet blir applikasjonen brukt som en testapplikasjon for å vurdere og evaluere hvordan en applikasjon kan ha nytte av å bruke en FHIR buss for henting av pasientlister og medisinlister, istedenfor å kommunisere direkte med et openEHR basert pasientjournalsystem. Et utsnitt fra applikasjonen vises i figur 6.1. Applikasjonen kjører på et eget nettbrett

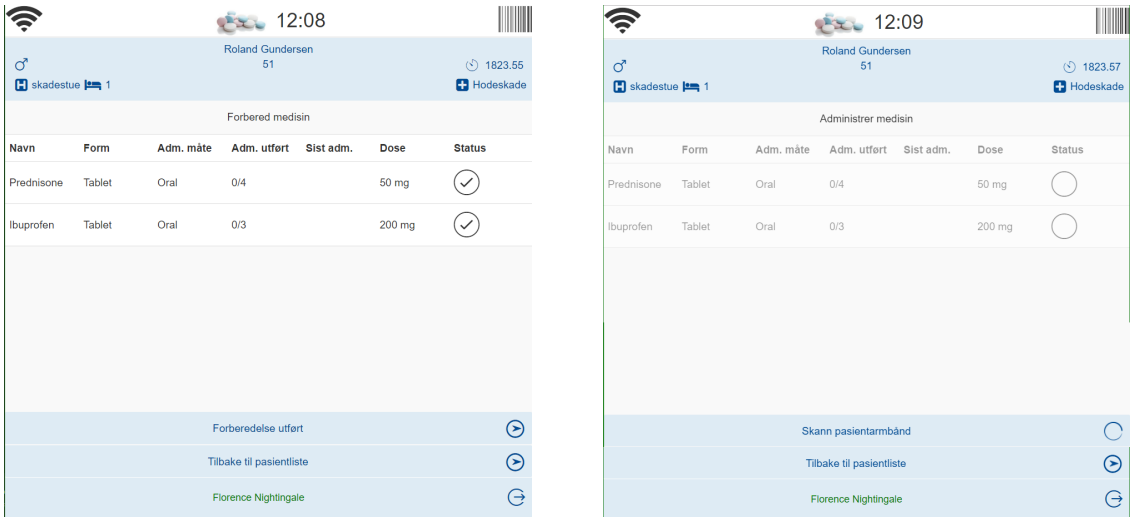


Figur 6.1: Figuren viser et utsnitt av medisinapplikasjonen LEMI. Her vises startskjermen som sykepleieren ser etter han er logget inn. Pasientene er hentet gjennom FHIR bussen. Her kan sykepleieren se både navn på pasienten, kort opppsummering av diagnose, hvilket rom pasienten ligger på og sengnummeret.

som har en integrert strekkodeskanner for skanning av pasientarmbånd. LEMI ble utviklet i JavaScript med hjelp av rammeverket AngularJS. For å kunne kjøre applikasjonen på nettbrettet som en fullverdig Windows applikasjon ble Apache Cordova brukt for å opprette en Windows 8.1 kompatibel applikasjon. Applikasjonen er også utvidet som en webapplikasjon slik at autoriserte brukere med internett kan teste applikasjonen.

LEMI lar autoriserte personer, som sykepleiere, logge seg på og hente ut en liste med pasienter. For hver pasient kan en se alle medisinene som pasienten skal ha på et gitt tidspunkt. Applikasjonen kan brukes i alle fire prosessene beskrevet i avsnittet 3.1.

Når sykepleieren er logget på vises listen med alle pasientene, som vist i figur 6.1. For hver pasient kan sykepleieren se om medisinene til pasienten har 1) blitt forberedt for administrering, 2) dobbelsjekket av en annen autorisert person, eller 3) blitt administrert til pasienten. Disse tre stegene vises ved hjelp av tre sirkler på høyre side av pasientnavnet. I figur 6.1 kan en se at ingen medisiner gjort klare, dobbelsjekket eller administrert. Over pasientlisten kan sykepleieren få en enkel oversikt over antall pasienter som ikke er startet på, antall pasienter hvor medisinene er forberedt, antall pasienter hvor medisinene er dobbelsjekket, og antall pasienter hvor medisinene er administrert. Nå kan sykepleieren velge hvilken pasient som han skal gi medisiner til ved å trykke på pasientnavnet. I vårt eksempel ønsker sykepleieren



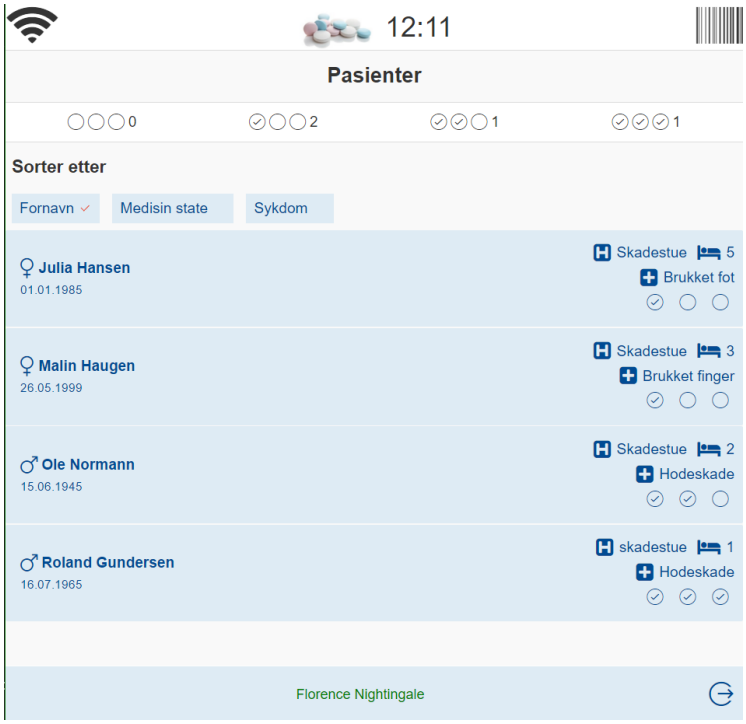
(a) Funksjonalitet for å informere at sykepleieren har forberedt medisin til pasienten.

(b) Funksjonalitet for å skanne pasientarmbåndet for å bekrefte at sykepleieren står ovenfor rett pasient før medisinene administrert.

Figur 6.2: Utsnitt fra medisinapplikasjonen LEMI

å utføre medisinrunden for pasienten Roland Gundersen. Sykepleieren velger pasienten og får opp en liste med medisiner. Dette steget vises i figur 6.2a. I eksempelet skal Roland ha en 50 mg tablett med Prednisone og 200 mg tablett med Ibuprofen. Sykepleieren kan også se hvor mange ganger medisinen er blitt administrert i dag og eventuelt forrige tidspunkt medisinen er administrert. I eksempelet er dette første administrering og et tidspunkt vil dermed ikke vise. Sykepleieren forbereder rett mengde med rett medisin og huker av en «checkbox» under fanen «status» for å fortelle at medisinen er gjort klar. Når alle medisinene er gjort klare får sykepleieren muligheten å gå videre til neste steg i medisinrunden ved å trykke «Forberedelse utført». Når medisinene er hentet og gjort klar for administrering og bekreftet av en annen autorisert person kan sykepleieren gå til pasienten sitt rom for å gi medisinene. Før medisinene blir delt ut må sykepleieren skanne pasientarmbåndet

for å bekrefte at sykepleieren står ovenfor rett pasient. Dette er et viktig steg for å unngå at feil pasient får medisinen. Dette steget er vist i figur 6.2b hvor sykepleieren må skanne armbåndet før han kan bekrefte at pasienten har fått medisinen. Når armbåndet er skannet og medisinen er administrert, kan sykepleieren gå tilbake til hovedskjermen og se at alle tre sirklene er huket av for å informere at pasienten har fått medisinene. Figur 6.3 viser pasientlisten hvor pasienten Roland Gundersen har fått alle medisinene som han skal ha.

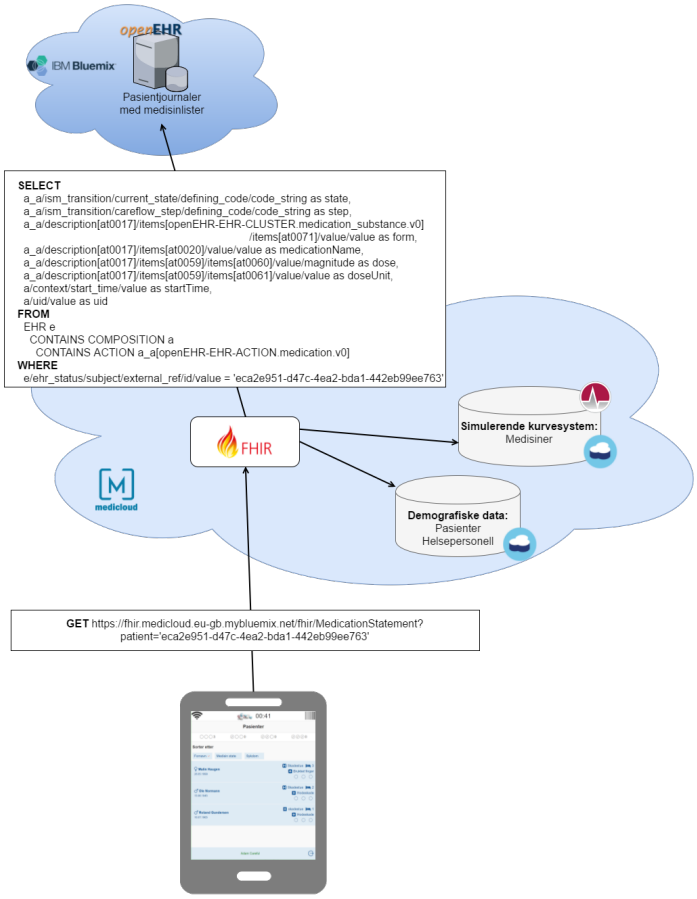


Figur 6.3: Figuren viser et utsnitt av applikasjonen hvor sykepleierne som bruker applikasjonen kan se at pasienten Roland Gundersen har fått medisinene han skal ha på dette tidspunktet. Sykepleierne kan også se at medisinene til Malin og Julia er forberedt, mens medisinene til Ole er dobbelsjekket og klar til administrering.

6.2 Integrere LEMI mot FHIR bussen

Gjennom beskrivelsen av applikasjonen i avsnitt 6.1 har det blitt avduket flere funksjonaliteter som krever en integrasjon mot en tjeneste. Denne tjenesten må tilby både innloggingsfunksjonalitet, henting av pasienter, henting av medisinliste og oppdatering av status for medisinrunden. Informasjonen om sykepleierne, pasientene og medisinene er lagret i egne databaser som FHIR bussen har direkte tilgang til. Informasjon om ordineringer og status

for medisinrunden er lagret i komposisjoner i pasientjournalen for hver pasient. En oversikt over systemet vises i figur 6.4. Systemet har tre hovedkomponenter; openEHR pasientjournalsystem, FHIR buss og LEMI medisinapplikasjon. Når LEMI trenger tilgang til enten pasienter, sykepleiere eller medisinlister kommuniserer LEMI med FHIR bussen. FHIR bussen håndterer hvor informasjonen er lagret og henter informasjonen enten fra egen database eller fra pasientjournalsystemet.



Figur 6.4: Figuren viser hvordan medisinapplikasjonen LEMI kommuniserer med FHIR bussen for å få tilgang til pasienter, sykepleiere og pasientjournaler. Figuren viser også hvordan en forespørsel om å hente medisinlisten til pasienten blir sendt til FHIR bussen og konvertert over til en openEHR kompatibel AQL spørring. Mer informasjon i neste avsnitt.

I følgende avsnitt vil det bli beskrevet hvordan integrasjonen er implementert mot FHIR bussen for å hente medisinlister og å oppdatere medisinstatusen. En sammenligning av hvordan en integrasjon mot openEHR pasientjournalsystemet for samme funksjonalitet ville blitt gjort er også beskrevet.

6.2.1 Funksjonalitet: Hente medisinliste for en pasient

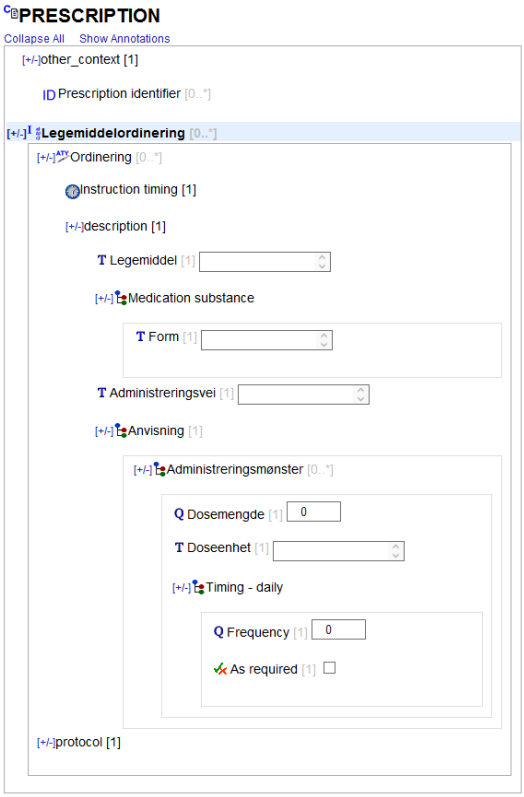
I openEHR representeres en ordinering ved hjelp av INSTRUCTION arketypen *Legemiddelordinering*. En instruksarketype inneholder en eller flere aktiviteter som skal bli utført i fremtiden. Før en aktivitet er fullført kan aktiviteten gå gjennom flere steg, som disponering og administrering av en medisin eller utsettelse av en ordinering. Alle disse stegene er loggført ved hjelp av en ACTION arketype kalt *Medication action*. For å avgjøre om en medisin er blitt forberedt og administrert til pasienten har openEHR et objekt, ISM_TRANSITION, som brukes for å beskrive statusen til en aktivitet. Når en skal tolke om en aktivitet fra en instruksjon er blitt utført eller ikke må en hente den siste ACTION arketypen for instruksjonsarketypen og deretter tolke statusen. Når en ordinering blir opprettet ved hjelp av FHIR bussen opprettes det også en «Medication action» arketype for instruksen som beskriver at dette er en ny instruks hvor ingen aktiviteter er startet.

openEHR mal og FHIR profil for Legemiddelordinering

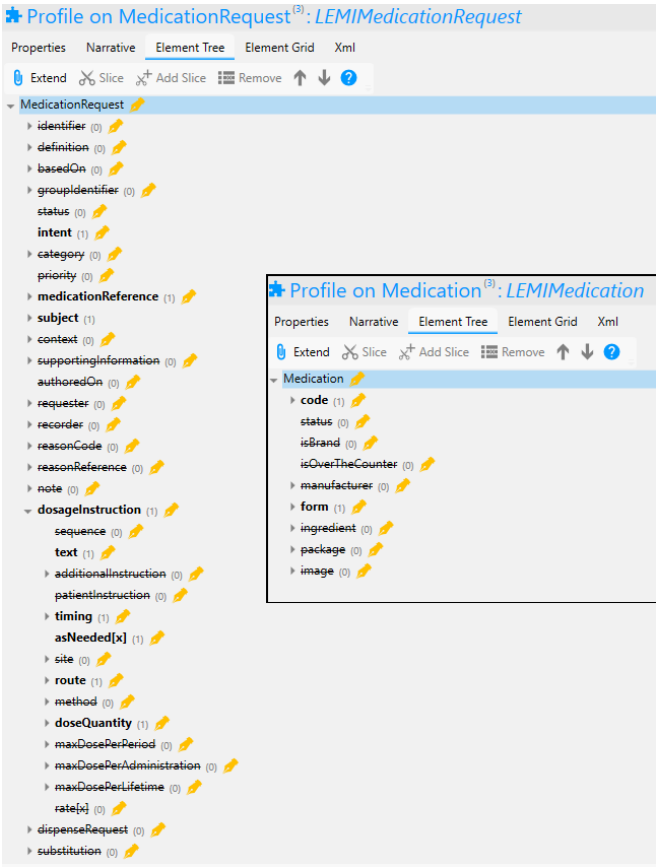
Template Designer ble brukt for å opprette en mal for ordineringer. Arketypen Legemiddelordinering ble begrenset ved å fjerne flesteparten av feltene. Feltene som inneholder informasjon som LEMI trenger er satt til obligatoriske. Template Designer lar også brukerne opprette et HTML skjema for å visualisere malen. Figur 6.5a viser et utsnitt av feltene som er med i malen for ordineringer. Det ble også opprettet en FHIR profil med tilsvarende obligatoriske felter. En ordinering i FHIR er bygget opp av ressursen MedicationRequest. Denne ressursen inneholder informasjon om dosering, tid for administrering av medisin, pasient og en referanse til en medisin. Siden medisinene er lagret i en egen database som er tilgjengelig via FHIR bussen, peker medisinreferansen på en identifikator for en gitt medisin. For å dekke behovene i openEHR malen er både en profilert versjon av MedicationRequest og Medication opprettet, se figur 6.5b. Hele konverteringsfilen for å konvertere fra en MedicationRequest og Medication til en arketype for Legemiddelordinering er listet opp i listing B.1 Vedlegg B.

Forespørsel sendt til FHIR buss

For å hente en medisinliste for en pasient brukes endepunktet for ressursen MedicationStatement. MedicationStatement er en ressurs for å beskrive en medisin som pasienten tar nå. En liste med ressurser av typen MedicationStatement kan tolkes som en medisinliste for pasienten. En av ressursens søkeparametre er «patient». Parameteren brukes for å søke etter



(a) openEHR mal av arketypen *Legemiddelordinering* i form av et HTML skjema. Arketypen inneholder to arketyper, *Medication substance* og *Timing - daily*, for å beskrive formen på medisinen og hvor ofte medisinen skal bli tatt. Det er også obligatorisk å inkludere navnet på medisinen (i kodet format), informasjon om dosering, administreringsvei og om det er lov å administrere medisinen ved behov.



(b) Profilering av ressursene MedicationRequest og Medication. I lag dekker disse profilene alle obligatoriske felter funnet i openEHR malen. Profilen for MedicationRequest inneholder feltet *subject* siden ressursene eksisterer alene og ikke i lag med en pasientjournal.

Figur 6.5

alle MedicationStatement ressursene til en gitt pasient. Alle søkeparameterene som er tilgjengelige for MedicationStatement er dokumentert i FHIR spesifikasjonen via nettadressen `https://www.hl7.org/fhir/medicationstatement.html#search`. Hvilke parametre som kan brukes ved et søk varierer fra implementasjon til implementasjon.

Når LEMI henter ut en medisinliste for en pasient sendes en HTTP GET forespørsel:

GET https://fhir.medicloud.eu-gb.mybluemix.net/fhir/MedicationStatement?patient=eca2e951-d47c-4ea2-bda1-442eb99ee763.

Figur 6.4 viser at forespørselen blir sendt mellom LEMI og FHIR bussen. En egen «session-id» sendes med forespørselen for å autorisere brukeren. Ved en vellykket forespørsel retur-

neres det en Bundle ressurs, se listing 6.1, med alle MedicationStatement ressursene med enten status lik «intended», «active» eller «completed».

Listing 6.1: Medisinliste representert som FHIR Bundle med en liste med MedicationStatement ressurser

```
{
  "resourceType": "Bundle",
  "id": "38d1de8c-57c9-4952-a095-af3b05210693",
  "meta": {
    "lastUpdated": "2017-05-30T18:33:03.779+00:00"
  },
  "type": "searchset",
  "total": 4,
  "link": [
    {
      "relation": "self",
      "url": "http://fhir.medicloud.eu-gb.mybluemix.net/fhir/MedicationStatement?patient=eca2e951-d47c-4ea2-bda1-442eb99ee763"
    }
  ],
  "entry": [
    {
      "fullUrl": "http://fhir.medicloud.eu-gb.mybluemix.net/fhir/MedicationStatement/04ff73a0-74a1-4f6f-a283-fa504f9298fe::vm01.ethercis.org::1",
      "resource": {
        "resourceType": "MedicationStatement",
        "id": "04ff73a0-74a1-4f6f-a283-fa504f9298fe::vm01.ethercis.org::1",
        "meta": {
          "profile": [
            "http://vonk.furore.com/StructureDefinition/c71d5f7f-48ec-4a34-9ffa-d4c9563b256f"
          ]
        },
        "contained": [
          {
            "resourceType": "Medication",
            "id": "1",
            "meta": {
              "profile": [
                "http://vonk.furore.com/StructureDefinition/7dddfe79-4c34-4781-983d-2cf49a4e693d"
              ]
            },
            "code": {
              "coding": [
                {
                  "system": "http://hc-sc-gc-ca",
                  "code": "02244522",
                  "display": "Nexium"
                }
              ]
            },
            "form": {
              "coding": [
                {
                  "system": "http://snomed.info/sct",
                  "code": "385055001",
                  "display": "Tablett"
                }
```

```
51                  ]
52                }
53              }
54            ],
55            ...
56            "status": "intended",
57            "medicationReference": {
58              "reference": "Medication/ab18676a-fbfe-49e9-b331-38da65d2a2ba"
59            },
60            "effectiveDateTime": "2017-05-23T10:31:00Z",
61            "subject": {
62              "reference": "eca2e951-d47c-4ea2-bda1-442eb99ee763"
63            },
64            "dosage": [
65              {
66                "text": "morgen",
67                "asNeededBoolean": true,
68                "route": {
69                  "text": "Oral"
70                },
71                "doseQuantity": {
72                  "value": 500,
73                  "unit": "mg"
74                }
75              }
76            ]
77          }
78        },
79        ...
80    ]
81  }
```

I svaret på forespørselen kan en hente ut dosen og relevant medisininformasjon. FHIR bussen returnerer Medication ressursen som en del av tilhørende MedicationStatement, slik at en kan en hente ut navnet på medisinen og hvilken form medisinen skal ha, uten å sende en ny GET forespørsel.

Forespørsel sendt til openEHR pasientjournalsystem

Tilsvarende forespørsel kan også sendes direkte til openEHR pasientjournalsystemet. Siden en ønsker å hente alle aktive medisiner må det sendes en AQL forespørsel. AQL forespørselen må inneholde hvilken informasjon en ønsker å hente ut, hvor informasjonen ligger og hvilken pasient journalen tilhører. AQL spørringen sendes til pasientjournalsystemet ved hjelp av endepunktet `https://<openEHR-url>/rest/v1/query`. Endepunktet tar inn en «aql» parameter som inneholder en AQL spørring lik eksempelet vist i listing 6.2. Alle AQL spørringer som utføres er hardkodet i selve FHIR bussen. Hver forespørsel som blir sendt inn til FHIR bussen blir prosessert og utifra forespørselen vet FHIR bussen hvilken AQL

spørring som skal utføres. Figur 6.4 viser en AQL spørring som blir sendt til openEHR pasientjournalsystemet utifra en GET forespørsel sendt til FHIR bussen.

Listing 6.2: AQL spørring etter aktiv medisinliste

```
SELECT
  a_a/ism_transition/current_state/defining_code/code_string as state,
  a_a/ism_transition/careflow_step/defining_code/code_string as step,
  a_a/description[at0017]/items[openEHR-EHR-CLUSTER.medication_substance.v0]
                                        /items[at0071]/value/value as form,
  a_a/description[at0017]/items[at0020]/value/value as medicationName,
  a_a/description[at0017]/items[at0059]/items[at0060]/value/magnitude as dose,
  a_a/description[at0017]/items[at0059]/items[at0061]/value/value as doseUnit,
  a/context/start_time/value as startTime,
  a/uid/value as uid
FROM
  EHR e
    CONTAINS COMPOSITION a
      CONTAINS ACTION a_a[openEHR-EHR-ACTION.medication.v0]
WHERE
  e/ehr_status/subject/external_ref/id/value = 'eca2e951-d47c-4ea2-bda1-442eb99ee763'
```

Når AQL forespørselen blir sendt til pasientjournalsystemet returneres det et resultatssett basert på feltene «state», «step», «form», «medicationName», «dose», «doseUnit», «startTime» og «uid» spesifisert i SELECT klausulen. For forespørselen beskrevet over returneres et resultatsett som vist i listing 6.3.

Listing 6.3: Resultat av AQL spørring etter medisinlisten for pasient

```
1  {
2    "executedAQL": "SELECT  a_a/ism_transition/current_state/defining_code/
         code_string as state...",
3    "resultSet": [
4      {
5        "medicationName": "http://hc-sc.gc.ca::02244522|Nexium|",
6        "uid": "04ff73a0-74a1-4f6f-a283-fa504f9298fe::vm01.ethercis.org::1",
7        "dose": "500",
8        "form": "http://snomed.info/sct::385055001|Tablett|",
9        "startTime": "2017-05-32T10:31:00Z",
10       "step": "at0042",
11       "state": "526",
12       "doseUnit": "mg"
13     }
14   ]}
```

FHIR bussen prosesserer resultatsettet returnert fra pasientjournalsystemet ved å bruke konverteringsmodulen for å konvertere resultatsettet over til en eller flere MedicationStatement ressurser.

6.2.2 Funksjonalitet: Oppdatere medisinstatus

Som beskrevet i 6.2.1 er det ACTION arketyper som inneholder informasjonen om hvilke handlinger som er utført for en instruks. Når en skal oppdatere medisinstatusen ønsker en å

informere pasientjournalsystemet at et handling har blitt utført for instruksen om medisinering. For hver oppdatering av medisinstatusen opprettes det en ny ACTION arketype med verdier i feltene «careflow_step» og «current_state». Det sistnevnte feltet er kodet med openEHR koder for å beskrive en instruksjonsstatus. Hele listen med tilgjengelige koder for å beskrive instruksjonsstatusen finnes i figur 6.6 [72].

Terminology: *openehr* Group_name("en"): *"Instruction states"*			
Concept id	**Rubric (en)**	**Description (en)**	**Mappings**
524	"initial"	The instruction is recorded but no state is determined	
526	"planned"	The instruction is planned	
527	"postponed"	The instruction has been posponed - it had not be commenced	
528	"cancelled"	The instruction has been cancelled - it had not been commenced and will not commence in the future	
529	"scheduled"	The instruction has been scheduled to be carried out at a particular time	
245	"active"	The instruction is currently being carried out	
530	"suspended"	The instruction is suspended, it has been activated but is not active at present. It could be active again in the future.	
531	"aborted"	The instruction is aborted, it has been activated but ceased before it has been completed and will not be restarted in the future.	
532	"completed"	The instruction has been completed	
533	"expired"	The instruction has expired, timed out - and assumed to have either been cancelled, aborted or completed	

Figur 6.6: openEHR sine terminologikoder for å beskrive en status for en instruksjon. Figuren er hentet fra terminologispesifikasjonen til openEHR. En instruksjon som er planlagt å bli utført i fremtiden vil bli beskrevet ved hjelp av koden 526. I formatet ECISFLAT vil det se slik ut: «openEHR::526|planned|»

Per dags dato er medisinstatusen representert som et heltall i en FHIR Patient profil; 0 for startstatus, 1 for å beskrive at medisinene er forberedt, 2 for å beskrive at medisinene er dobbelsjekket av en annen autorisert person og 3 for å beskrive at medisinene er administrert til pasienten. Fra ulike medisinstatuser opprettes ulike versjoner av ACTION arketypen «Medication action». Feltet «careflow_step» i arketypen beskriver et prosesstrinn og inneholder lokale koder spesifisert i ADL filen til arketypen. Totalt er det 23 ulike prosesstrinn som en kan velge mellom. Listing 6.4 viser en ADL representasjon som inneholder de tre ulike prosesstrinnene som tilsvarer medisinstatusene 1, 2 og 3. I ADL representasjonen har en tre ulike ISM_TRANSITION elementer. I hvert element eksisterer det et «current_state» og «careflow_step» element. Dersom den lokale koden er at0005 eller at0006 settes «current_state» til openEHR koden 245. Dersom den lokale koden er at0003 kan en velge å

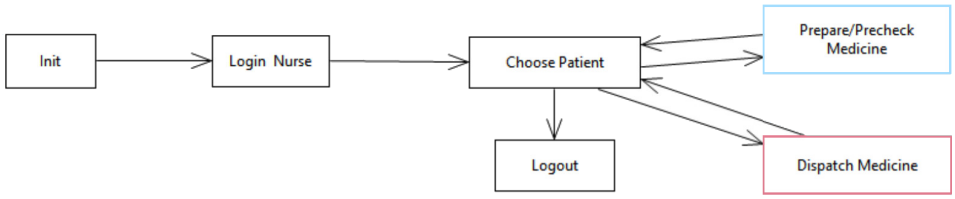
bruke enten openEHR koden 245 eller 524. I FHIR bussen brukes koden 245 for å indikere at administreringen av medisinen gitt i instruksen er satt i gang.

Listing 6.4: Utsnitt fra ADL representasjon av arketypen Medication action for prosesstrinn brukt i LEMI

```
1  ACTION[at0000] matches {        -- Medication action
2    ism_transition matches {
3      ISM_TRANSITION[at0003] matches {  -- Dispense medication (kommentar)
4        current_state matches {
5          DV_CODED_TEXT matches {
6            defining_code matches {[openehr::245, 524]}
7          }
8        }
9        careflow_step matches {
10         DV_CODED_TEXT matches {
11           defining_code matches {[local::at0003]}
12         }
13       }
14     }
15     ISM_TRANSITION[at0005] matches {  -- Review medication (kommentar)
16       current_state matches {
17         DV_CODED_TEXT matches {
18           defining_code matches {[openehr::245]}
19         }
20       }
21       careflow_step matches {
22         DV_CODED_TEXT matches {
23           defining_code matches {[local::at0005]}
24         }
25       }
26     }
27     ISM_TRANSITION[at0006] matches {  -- Administer medication (kommentar)
28       current_state matches {
29         DV_CODED_TEXT matches {
30           defining_code matches {[openehr::245]}
31         }
32       }
33       careflow_step matches {
34         DV_CODED_TEXT matches {
35           defining_code matches {[local::at0006]}
36         }
37       }
38     }
39   }}
```

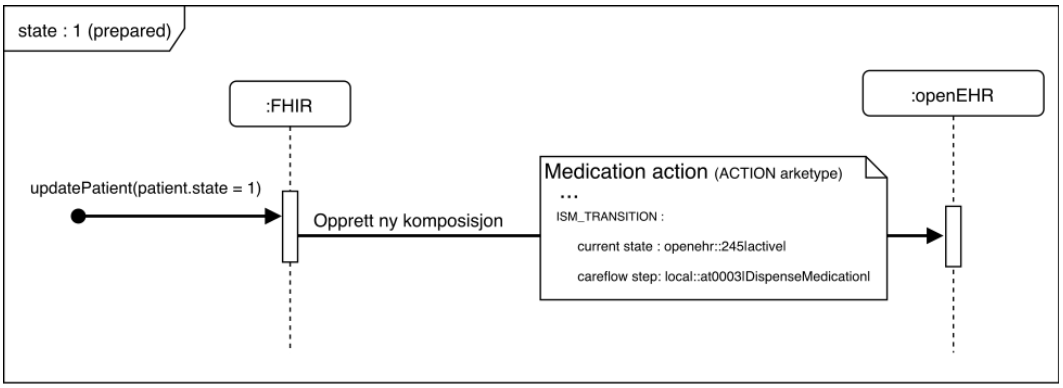
Før applikasjonen ble utviklet opprettet Xiaoliang Wang en prosessmodell med de viktigste funksjonene. Figur 6.7 viser prosessmodellen med applikasjonens hovedfunksjoner; velge pasient utifra pasientliste (Choose Patient), forberede/dobbelsjekke medisin (Prepare/Precheck Medicine) og administrere medisin (Dispatch Medicine). Prosessene markert i blå og rosa er prosesser hvor medisinstatusen blir oppdatert. Prosessen i blått inneholder to funksjonaliteter; «prepare» og «precheck». Årsaken til at to funksjonaliteter vises i en prosess er at disse to er implementert ganske likt i applikasjonen, men med ulikt tekst.

Medisinstatusen brukes i FHIR bussen for å avgjøre om feltet «careflow_step» skal bli satt til den den lokale koden at0003 (medisinstatus 1), at0005 (medisinstatus 2) eller at0006



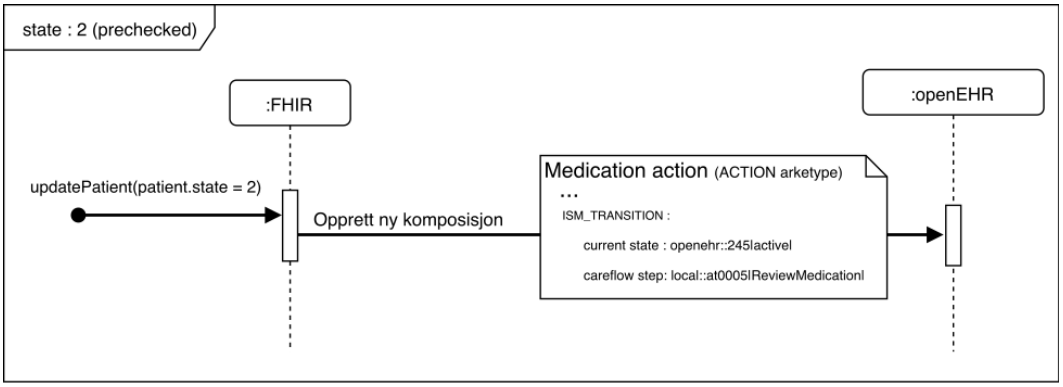
Figur 6.7: Prosessmodell utviklet for applikasjonen LEMI. Viser hovedfunksjonaliteten i applikasjonen. Først logger sykepleieren seg inn og får en liste med pasienter. Utifra listen kan sykepleieren velge en pasient og får opp en liste med medisiner. Utifra hvilket steg medisinadministreringen er på kan sykepleieren enten 1) forberede («prepare») medisinen, 2) dobbelsjekke og bekrefte («precheck») medisinen eller 3) administrere («dispatch») medisinen til pasienten.

(medisinstatus 3). Når medisinstatusen oppdateres til 1 (forberedt), sendes en PUT forespørsel til FHIR bussen om å oppdatere pasienten. FHIR bussen håndterer forespørselen og ved hjelp av konverteringsmodulen opprettes det en ny komposisjon med ACTION arketypen «Medication action». I figur 6.8 vises et forenklet bilde hvordan denne prosessen foregår og hvilken verdi «current_state» og «careflow_step» har.



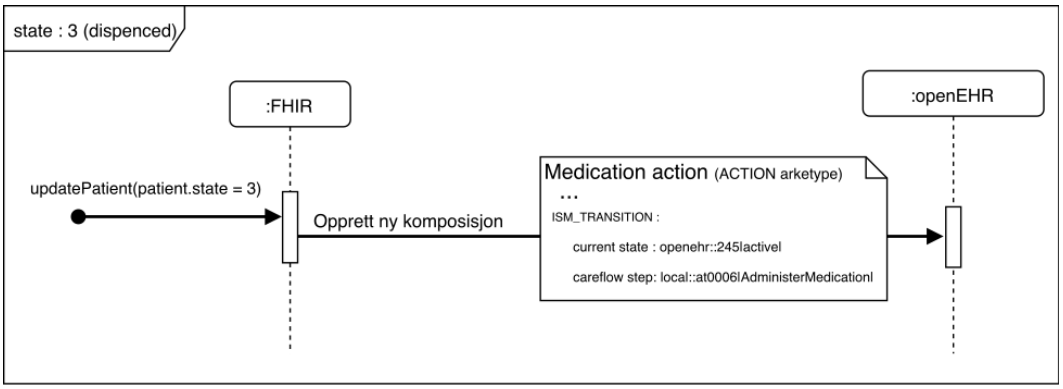
Figur 6.8: Diagram over flyten til oppdatering av medisinstatus til status lik 1

I figur 6.9 oppdateres pasienten med en medisinstatus lik 2 for å beskrive at medisinen er dobbelsjekket og klar til administrering. Ny komposisjon med tilhørende «careflow_step» som beskriver statusen sendes til openEHR pasientjournalsystemet. I siste figur 6.10 har medisinen blitt administrert til pasienten. Legg merke til at «current_state» i komposisjonen ikke er «completed», men fremdeles «active». Årsaken til dette er at dette er en medisin som fremdeles er aktiv for pasienten. Dersom ordineringen for medisinen går ut på dato eller blir avsluttet vil det opprettes en ny komposisjon med «current_state» lik «completed».



Figur 6.9: Diagram over flyten til oppdatering av medisinstatus til status lik 2

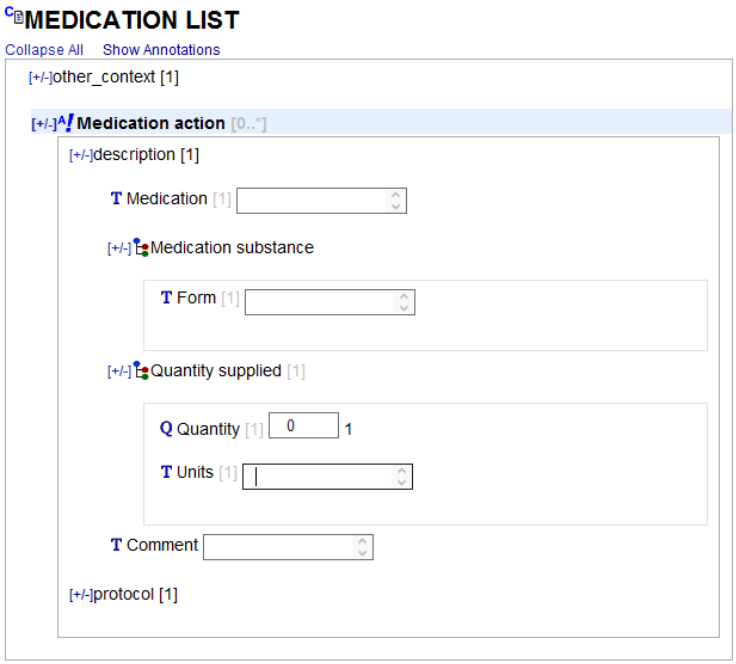
Konseptet ACTION arketyper eksisterer kun i openEHR, og ikke i FHIR. Dermed er det laget en ekstra utvidelse i Patient ressursen som beskriver medisinstatusen. På grunn av dette blir ikke forespørselene sendt til FHIR bussen og pasientjournalsystemet sammenlignet.



Figur 6.10: Diagram over flyten til oppdatering av medisinstatus til status lik 3

openEHR mal for medisin ACTION arketype

Før vi kan opprette en «Medication action» arketype må det opprettes en openEHR mal. Siden arketypen skal bli brukt for å hente ut medisinlisten inneholder malen feltene som LEMI trenger. I figur 6.11 er malen representert i form av et HTML skjema. Feltet «Medication» inneholder navnet på medisinen. Deretter er arketypen utvidet med en ny arketype, «Medication substance», for å beskrive hvilken form medisinen skal ha. Denne utvidelsen finnes også i malen for ordinering. Malen inneholder også informasjon om dosering.



Figur 6.11: openEHR mal av arketypen *Medication action* i form av et HTML skjema. Arketypen er utvidet ved hjelp av en arketype, *Medication substance*, for å beskrive formen på medisinen. Det er også obligatorisk å inkludere navnet på medisinen (i kodet format) og informasjon om dosering.

Kapittel 7

Resultat og Evaluering

I de første seks kapitlene er arbeidet utført i masterprosjektet dokumentert og beskrevet ved hjelp av teori og eksempler. En viktig del av masterprosjektet er å vurdere og analysere arbeidet som er utført. Det første avsnittet i kapittelet presenterer hvilke resultater som er produsert i masterprosjektet. FHIR bussen evalueres først utifra noen retningslinjer og krav for hvordan et REST API skal struktureres. Deretter er arbeidet vurdert utifra hvert krav fra listen over funksjonelle og ikke-funksjonelle krav som finnes i kapittel 3. Til slutt drøfter vi hvordan våre resultater besvarer forskningsspørsmålene som er spesifisert i kapittel 1. Med hjelp av våre resultater diskuterer vi fordelene med å ha en FHIR integrasjonsbuss mellom klientapplikasjoner og andre bakenforliggende systemer. Applikasjonen dokumentert i kapittel 6 er brukt som en del av vurderingsarbeidet for å vurdere nytten av å ha en FHIR integrasjonsbuss som håndterer alle forespørsler.

7.1 Resultater

I dette masterprosjektet er det utviklet en tjenestearkitektur bestående av en FHIR integrasjonsbuss og et openEHR basert pasientjournalsystem. Dette ble implementert for å kunne evaluere hvilke fordeler man har ved å introdusere et FHIR grensesnitt mellom klientapplikasjoner og bakenforliggende e-helsesystemer. FHIR bussen tilbyr ulike tjenester som pasienthåndtering, medisinhåndtering, observasjonshåndtering og ordineringshåndtering. Flere tjenester som FHIR bussen tilbyr utnytter pasientjournalsystemet for henting og lagring av data. For å få til dette var det nødvendig å utføre en konvertering mellom helsestandardene. For et utvalg av openEHR arketypene ble det undersøkt hvordan de var strukturert og hvilke FHIR ressurser som dekket de obligatoriske egenskapene i openEHR arketypen. En kort sammenligning av standardene finnes i tabell 7.1, hvor noen hovedpunkter er plukket ut for sammenligning.

I masteroppgaven ble arketypene for blodtrykk, allergi/intoleranse, diagnose/problem og ordinering konvertert over til tilhørende FHIR ressurser. Konverteringsprosessen startet med å kartlegge de obligatoriske feltene hos både arketypen og tilhørende ressurs(er). Utifra de obligatoriske feltene ble det opprettet en konverteringsfil som er basert på forskningsarbeidet om hvordan en kan konvertere mellom HL7 FHIR og HL7 CDA [62], se avsnitt 2.5.3. Konverteringsfilen ble bygget opp av flere kombinasjoner av en openEHR sti basert på lokale openEHR koder og tilhørende FHIR dataelementer. Når en konverteringsfil var opprettet kunne den brukes i en egen konverteringsmodul. Konverteringsmodulen tok inn enten en openEHR komposisjon eller en FHIR ressurs, og konverterte objektet over til den motsatte standarden med hjelp av tilhørende konverteringsfil. Valideringen av konverteringen ble utført ved å kjøre tester på objektet både før og etter konverteringen. Da vi konverterte fra en openEHR arketype til en FHIR ressurs ble først arketypen validert mot openEHR pasientjournalsystemet ved å lagre arketypen. Pasientjournalsystemet validerer om arketypen er gyldig med hensyn til lagret mal. Etter valideringen ble arketypen slettet. Etterpå konverterte vi arketypen over til en FHIR ressurs som ble testet mot tilhørende FHIR profil. Dersom disse to testene var vellykket konverterte vi ressursen over til arketypen igjen som deretter ble validert mot pasientjournalsystemet ved å lagre og til slutt slette arketypen. Konverteringen mellom openEHR og FHIR ble også validert med å integrere en medisinapplikasjon, LEMI, mot FHIR bussen. Dette gav oss også en enkel evaluering av fordelene ved å integrere seg mot en FHIR tjeneste istedenfor en openEHR tjeneste.

Sammenligning	**openEHR**	**FHIR**
Hovedformål	Strukturering av klinisk data	Deling av klinisk data
Hvor lagres pasientidentifikasjon	**I selve pasientjournalen.** Arketypene eksisterer som en del av en pasientjournal, hvor pasientdata lagres	**I ressursen**. FHIR ressursene eksisterer alene uten en pasientjournal og pasientdata er dermed lagret i ressursen.
Strukturering av elementene	**Hierarki** - Arketypene er delt opp i fire kategorier og bruker et SLOT felt for å utvide en arketype med en annen	**Flat** - Ressursene bruker referanser til andre ressurser
Dataformat	Avhenger av implementasjon. XML, JSON, FLATJSON og ECISFLAT	XML og JSON
Dekning datapunkter	Prøver å utvikle arketyper som inneholder 100% av datapunktene som ønskes for et klinisk konsept	Ressursene skal inneholde cirka 80% av mest brukte datapunkter for et klinisk konsept
Datatyper	Både primitive og komplekse datatyper	Både primitive og komplekte datatyper
Spesialisering og begrensning av dataelement	Mal (templates) hvor arketypene avgrenses og utvides med andre arketyper	Profilering hvor ressurser begrenses og utvides ved hjelp av «extensions»
Spørring etter data	Et lite REST API som fokuserer på AQL spørring. Krever domenekunnskap og kunnskap om hvordan dokumentene er strukturert	Enkel og lesbar REST API. Krever ingen domenekunnskap om strukturen til elementet som hentes
Terminologi	Intern og eksterne koder	Interne og eksterne koder

Tabell 7.1: Kort sammenligning openEHR og FHIR

Som nevnt over startet konverteringsarbeidet ved å kartlegge de obligatoriske datafeltene. Dette var med på å besvare forskningsspørsmålet;

Hvilke modeller er openEHR og FHIR bygget opp av og lar det seg gjøre å konvertere mellom openEHR arketyper og FHIR ressurser?

Det ble her undersøkt hvordan en håndterer situasjonen hvor den ene standarden har et obligatorisk datafelt som den andre standarden ikke har. I vår forskning ble det avduket at de aller fleste feltene som bare FHIR ressursene hadde var felter som inneholdt pasientdata som pasientidentifikasjon. Denne informasjonen blir i openEHR lagret i selve pasientjournalen og ikke i arketypen. Dersom det var andre felter i FHIR ressursen som ikke tilhørende openEHR arketype hadde kunne vi utvide arketypen med andre arketyper i en mal. Denne

situasjonen oppstod svært sjeldent da arketypene er strukturert for å prøve å representere 100% av alle aspekter ved det kliniske konseptet. Dersom openEHR arketypen inneholdt felter som ikke FHIR ressursen hadde utvidet vi FHIR ressursen ved hjelp av profilering. Profilering lar oss legge til nye felter i FHIR ressursen når vi trenger det. Dette gjorde det mulig å utvide basisressursen slik at den ble kompatibel med tilhørende openEHR arketype.

En av de større utfordringene vi måtte håndtere var ulik bruk av terminologi og datatyper. Det hendte at arketypen hadde et felt for kodet tekst, mens tilsvarende felt i ressursen hadde en vanlig tekstrepresentasjon. I noen tilfeller der begge standardene støttet kodet tekst hadde ressursene spesifisert en obligatorisk terminologi som skulle bli brukt. Dette medførte at vi ikke klarte å konvertere datafeltet fra arketypen til ressursen dersom arketypen brukte en annen terminologi.

For hver konvertering ble det opprettet en eller flere FHIR profiler av tilhørende ressurser. Fordelene med å kunne profilere en ressurs er muligheten til å tilpasse ressursene etter ens behov. Dette gjør det mulig å konvertere en FHIR ressurs til et annet objekt fra en annen standard. Den største ulempen med dette er at alle nye utvidelser som blir lagt til en ressurs ikke nødvendigvis støttes av andre FHIR tjenester. Det må også nevnes at FHIR ressursene er profilert for å være kompatible med openEHR arketyper. Dersom en ressurs blir brukt for å hente data fra to ulike systemer som er basert på ulike helsestandarder må FHIR profilen dekke begge standardene. Dersom de to helsestandardene har ulike obligatoriske felter må profilen dekke en union av alle de obligatoriske feltene. Dette kan resultere i at flere datafelter enn nødvendig må fylles ut. Dette kan være problematisk dersom en ønsker å hente data fra system A, men tilhørende profil krever at de obligatoriske feltene fra system B også må fylles ut. En mulighet er å gjøre disse datafeltene til valgfrie, men da har en ikke muligheten å forsikre seg om at alle nødvendige felter er fylt ut ved oppretting av data. I noen tilfeller kan en løse dette problemet ved å bruke ressursen Composition eller Bundle for å kombinere flere ressurser i lag. Dette kan vi gjøre dersom et element fra system A kan representeres av ressursene A1 og A2, mens et element fra system B kan representeres av ressursene A2 og A3. Dette fungerer fint dersom flesteparten av feltene som er like mellom systemene kan representeres ved hjelp av ressurs A2, mens de resterende feltene i system A og system B kan representeres ved hjelp av A1 og A3 henholdsvis.

7.2 Vurdering

I følgende avsnitt vurderer vi implementasjonen av FHIR bussen og valg av openEHR pasientjournalsystem.

7.2.1 Vurdering av FHIR integrasjonsbuss

FHIR bussen er implementert ved hjelp av biblioteket HAPI FHIR og er et REST grensesnitt med tilgjengelige tjenester for ressursene Patient, Practitioner, Observation, Organization, Medication, MedicationRequest, Condition, Location, Encounter og AllergyIntolerance. Disse tjenestene blir vurdert ved hjelp av en liste med prinsipper for hva som karakteriserer et godt REST grensesnitt. Todd Fredrich skrev en artikkel i 2012 med en liste over 5 krav som et grensesnitt må oppfylle for å være et RESTfullt grensesnitt [31]. Vi vil bruke disse kravene og vurdere om vår FHIR buss tilfredstiller disse kravene.

Uniform Interface

En oppnår kravet om å ha et grensesnitt som er uniform dersom disse fire prinsippene er oppfylt:

1. **Resource-Based**
   Prinsippet beskriver at hver ressurs skal være identifisert i en forespørsel ved hjelp av unike URI identifikatorer. Hver ressurs skal derimot ikke være linket til representasjonen som returneres til klienten. Dette vil si at klienten kan returnere ressursen i enten HTML, XML eller JSON, mens dataene er annerledes strukturert i en annen database eller i et annet system. Vår FHIR buss returnerer ressurser som er strukturert etter ressursene i FHIR spesifikasjonen. Dataene som ressursen inneholder kan komme fra blant annet en eller flere openEHR arketyper eller interne Java objekter som er lagret i en database. Brukerne kan selv velge om ressursene skal være strukturerte i enten XML eller JSON format.

2. **Manipulation of Resources Through Representations**
   Prinsippet er oppfylt dersom hver ressurs som blir returnert inneholder nok informasjon for at klienten kan endre eller slette ressursen. FHIR ressursene som returneres MÅ inneholde en identifikator for at HAPI FHIR biblioteket skal returnere ressursen. Altså for hver GET forespørsel returneres den unike identifikatoren som trengs for å både oppdatere og slette ressursen.

3. **Self-descriptive Messages**
   Prinsippet er oppfylt dersom hver melding som blir sendt inneholder nok informasjon for å bli prosessert. For eksempel kan en melding spesifisere hvilken parser som skal bli brukt ved hjelp av «Content-Type» i «Header» feltet i meldingen. Alle «Header» feltene som blir returnert som en respons til GET forespørsler inneholder blant annet feltene «Content-Type», «Server» og «X-Powered-By». Det siste feltet er viktig for klienter av FHIR bussen, da den inneholder informasjon om hvilken versjon av FHIR spesifikasjonen som bussen støtter. Vår FHIR buss returnerer; *HAPI FHIR 2.3 REST Server (FHIR Server; FHIR 1.9.0/DSTU3).*

4. **Hypermedia As The Engine Of Application State (HATEOAS)**
   Prinsippet er oppfylt dersom responsene inneholder lenker for å kunne hente ressursene ved en senere anledning. Blant annet kan denne informasjonen ligge i «body» seksjonen, responskoder eller «Header» seksjonen i en respons. Ved en GET, POST eller PUT forespørsel returnerer FHIR bussen en respons hvor vi finner et «Location» felt i «Header» seksjonen. Dette feltet inneholder hele URI adressen for å kunne hente elementet ut igjen.

Stateless

En tjeneste oppfyller kravet om *Statelessness* dersom hver forespørsel inneholder den nødvendige informasjonen for å prosessere forespørselen. Dette vil si at det er **ikke** nødvendig for tjenesten å lagre data som trengs for fremtidige forespørsler. Dette kan blant annet være «session» informasjon for autentisering og autorisering av brukeren. Alle forespørslene som bruker openEHR pasientjournalsystemet vårt krever en unik session nøkkel som sendes ved hver forespørsel til FHIR bussen i feltet *Ehr-Session* i «Header» seksjonen. Altså lagres ingenting midlertidig i FHIR bussen for å prosessere fremtidige forespørsler.

Cacheable

Hver respons fra en tjeneste skal informere om informasjonen kan bufres eller ikke. Dette innebærer at responsen skal inneholde et felt i «Header» seksjonen som spesifiserer om innholdet kan bufres. Ved en GET forespørsel til FHIR bussen spesifiseres det bare ett felt, «Last-Modified», som spesifiserer når ressursen ble sist modifisert. Dessverre blir denne datoen satt automatisk av systemet til å være samme tid som når forespørselen sendes inn til FHIR bussen. Her bør FHIR bussen utvides til å spesifisere hvilke data som kan bli

bufret av klienten og maks tidsrom dataene kan bli bufret. Siden mesteparten av dataene som returneres er helserelaterte data til en pasient bør ikke dataene bli bufret for lenge. Det er viktig for klienten å prosessere oppdaterte data for å bevare pasientsikkerheten.

Client-Server

Grensesnittet til et REST API skal separere klientene og tjenesten. Dette vil si at FHIR bussen skal kunne endres uten at klientene må endres dersom grensesnitt er bevart. Dette stemmer for vårt system siden logikken i FHIR bussen er abstrahert bak ulike endepunkter. Klienten har altså ikke kjennskap til hvordan forespørslene håndteres i FHIR bussen.

Layered System

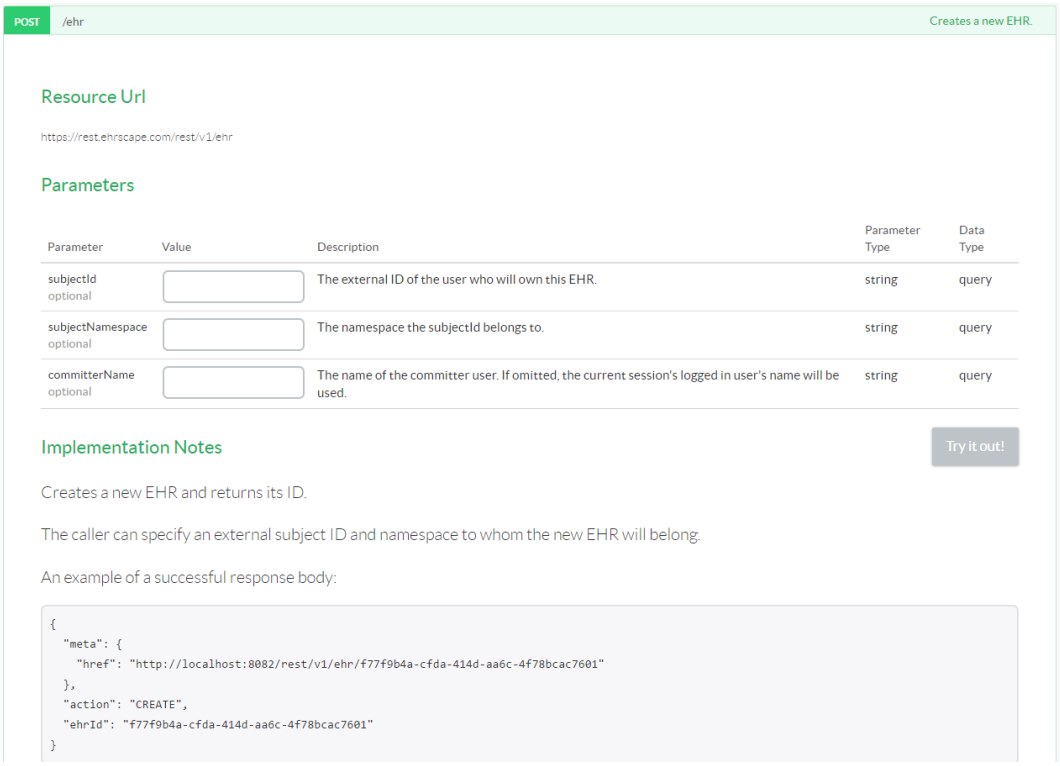
Kravet handler om å et system som er bygget opp i flere nivå. Det er viktig at en REST tjeneste er strukturert i flere lag som separerer grensesnittet, logikken og kommunikasjonen med databasen. Vår FHIR buss er bygget opp av tre lag; et API nivå, et logikknivå og et integrasjonsnivå. Denne strukturen er skjult for klientene og dermed oppfylles kravet om å ha et lagdelt system.

7.2.2 Funksjonelle krav

Dette avsnittet gjentar de funksjonelle kravene som FHIR bussen og openEHR pasient-journalsystemet skal oppfylle. For hvert krav vurderes det i hvor stor grad kravet er blitt oppfylt, med tilhørende eksempler hvor nødvendig.

Opprette ny pasientjournal når en pasient opprettes

openEHR systemet utviklet av EtherCIS kommer med et REST API hvor brukerne har tilgang til et endepunkt for å opprette en ny pasientjournal. Figur 7.1 viser et utsnitt av dokumentasjonen av endepunktet for å opprette en pasientjournal. Dokumentasjonen er hentet fra nettstedet `https://www.ehrscape.com/api-explorer.html`, som er et lignende REST API som EtherCIS bruker som utgangspunkt for strukturering av deres REST API. Som figuren viser kan en legge ved en ekstern identifikasjon til pasienten («subjectId»). Når en pasient lagres i databasen returneres det en unik identifikator for pasienten. Denne identifikatoren brukes for å opprette en ny pasientjournal ved å sette «subjectId» til den nye identifikatoren.



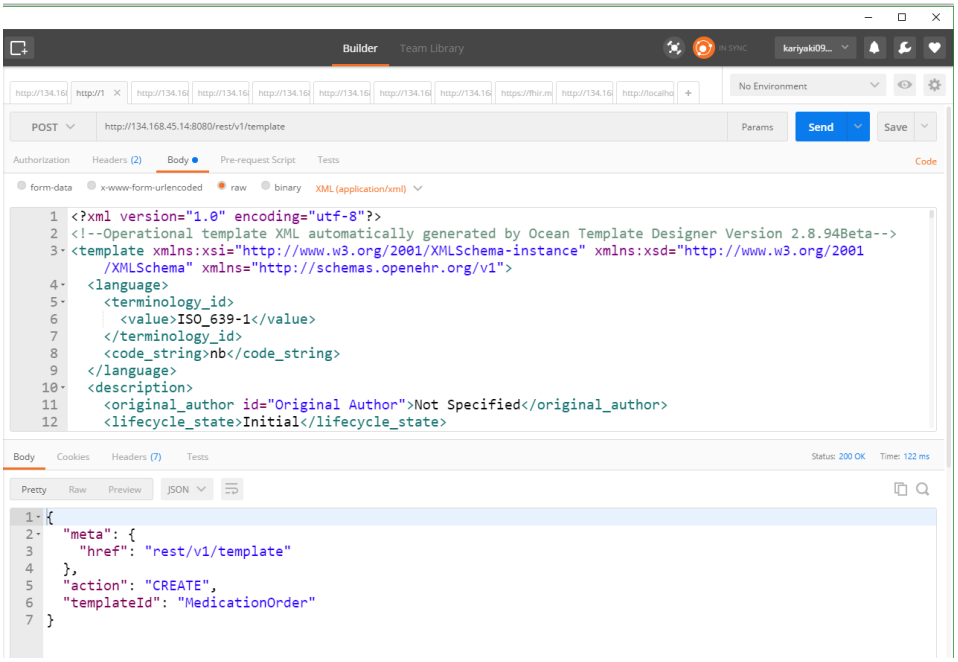
Figur 7.1: Utsnitt av representasjon av endepunkt for å opprette en pasientjournal.

Tilby tjenester for henting og lagring av FHIR ressurser

FHIR bussen har flere endepunkter for 10 ulike FHIR ressurser. Flere av disse ressursene kan opprettes gjennom FHIR bussen. Når en ressurs blir opprettet returneres det en unik identifikator som kan brukes for å hente ressursen igjen. Endepunktene listes opp i avsnitt 7.3.

Lagre ny mal i openEHR pasientjournalsystemet for å kunne hente og lagre arketyper kompatible med malen

EtherCIS kommer med et endepunkt for å opprette en ny mal i pasientjournalsystemet. Dersom en oppretter en mal for legemiddelordinering og vil senere oppdatere malen, brukes samme endepunkt. EtherCIS undersøker om dette er en ny mal eller en oppdatering av en eksisterende mal. Malen må være strukturert i formatet *Operational template (.opt)*. Figur 7.2 viser et utsnitt fra verktøyet Postman hvor en POST forespørsel er sendt til openEHR pasientjournalsystemet. Adressen http://134.168.45.14:8080 er ip-adressen til den kjørende Docker containeren med EtherCIS. Postman er utviklet av Postdot Technologies [73] som kan brukes for å teste eksisterende REST API tjenester. I oppgaven ble Postman jevnlig



Figur 7.2: Utsnitt fra Postman for å opprette en ny mal i openEHR pasientjournalsystemet

brukt for å teste endepunktene underveis. Figuren viser også hvordan svaret på forespørselen ser ut. I figuren er det opprettet en mal med navn «MedicationOrder».

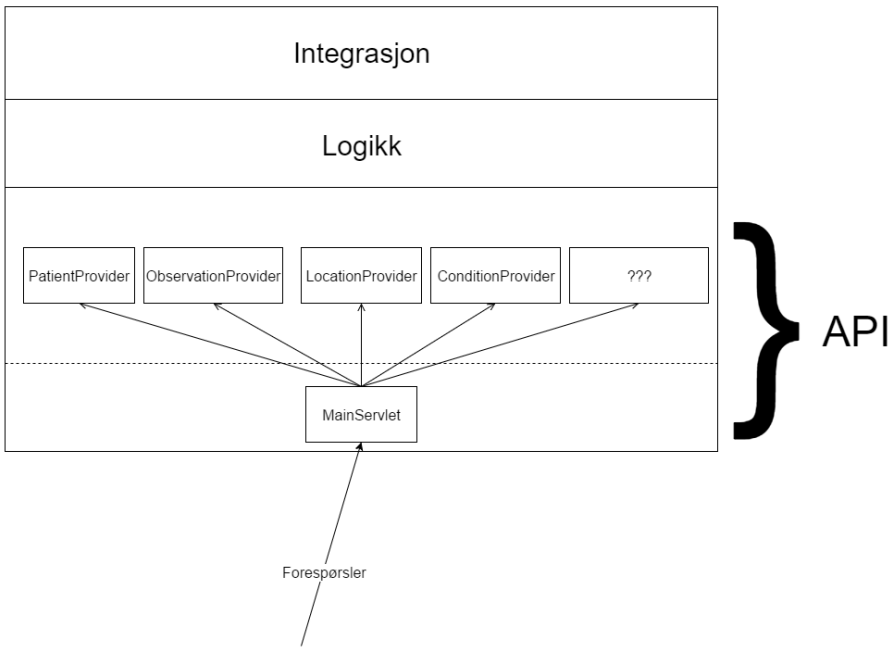
Lagre og hente komposisjon fra openEHR pasientjournalsystemet

EtherCIS har et endepunkt for å lagre og hente en komposisjon. Alle kliniske dokumenter lagres og hentes som komposisjoner og dermed brukes disse to endepunktene for alle arketyper. For å lagre en komposisjon sendes en POST forespørsel til pasientjournalsystemet med komposisjonen i ECISFLAT format i «kroppen». I selve URL-en spesifiseres det hvilken mal som komposisjonen er kompatibel med («templateId»), hvilken pasientjournal som komposisjonen skal lagres i («ehrId») og hvilket format komposisjonen er strukturert i («format»). Når komposisjonen blir lagret returneres det en unik identifikator, «compositionUid», som kan brukes for å hente komposisjonen senere. I figur C.1 i Vedlegg C vises et utsnitt fra Postman hvor en komposisjon er lagret.

For å hente ut samme komposisjon sendes en GET forespørsel til endepunktet «/composition/ID», hvor «ID» inneholder den unike identifikatoren som ble returnert ved forrige POST forespørsel. Når forespørselen er fullført returneres komposisjonen i ECISFLAT format. Figur C.2 i Vedlegg C viser et utsnitt fra Postman hvor komposisjonen som ble opprettet i figur C.1 hentes ut igjen.

Oppdatere FHIR bussen med flere ressurser

Slik FHIR bussen er implementert, ved hjelp av biblioteket HAPI FHIR, kan en legge til flere ressurser om nødvendig. Hver FHIR ressurs har en egen «provider» klasse med egne endepunkter for henting, oppdatering, lagring og søking etter data. FHIR bussen består av en egen servlet klasse, *MainServlet*, som håndterer alle forespørslene som sendes inn. I denne klassen er alle «provider» klassene registrert slik at MainServlet kan prosessere forespørselen og deretter vite hvilken «provider» klasse som forespørselen skal videresendes til. Figur 7.3 viser hvordan FHIR bussen er designet med en MainServlet og en «provider» klasse for et utvalg av de eksisterende ressursene. Denne arkitekturen gjør det mulig å oppdatere FHIR bussen med flere ressurser uten at klassene for de eksisterende ressursene må oppdateres.



Figur 7.3: Figuren viser en intern arkitektur av FHIR bussen som gjør det enkelt å legge til støtte for nye FHIR ressurser. «???» demonstrerer muligheten for å legge til nye ressurser.

Et utsnitt fra implementasjonen av MainServlet finnes i listing C.2 i Vedlegg C. Når en ønsker å legge til en ny ressurs opprettes det først en ny «provider» klasse som deretter legges til i MainServlet. Det vises også i listing C.3 i Vedlegg C hvordan en tom «provider» for ressursen FamilyMemberHistory ser ut. Videre kan en nå legge til endepunkter for oppretting, henting og oppdatering av ressurs.

Konvertering mellom openEHR arketyper og FHIR ressurser

FHIR bussen bruker en egen konverteringsmodul som er beskrevet i kapittel 4 og 5. Konverteringen mellom openEHR arketyper og FHIR ressurser er abstrahert inn i konverteringsmodulen. Årsaken til dette er at FHIR bussen er kun interessert i å konvertere mellom standardene, men FHIR bussen har ingen formening om hvordan dette gjøres. I tillegg er det nå mulig å erstatte konverteringsmodulen med en annen modul dersom en annen konverteringsmetode brukes i fremtiden.

Lagre og hente arketype som en FHIR ressurs

Siden EtherCIS har endepunkt for å opprette og hente hvilken som helst arketype når en tilhørende mal er lagret, er det opp til FHIR bussen å opprette tilhørende «provider» klasser for ressursene som trengs. For hver mal som er lagret i openEHR pasientjournalsystemet er det opprettet en tilhørende «provider» klasse for tilhørende ressurs. For å teste om en MedicationRequest, Condition eller AllergyIntolerance ressurs virkelig kom fra openEHR pasientjournalsystemet kunne vi bruke den unike identifikatoren lagret i ressursen for å hente komposisjonen ut fra openEHR pasientjournalsystemet ved hjelp av endepunktet «/composition/ID».

openEHR komposisjonsformat

Som tidligere beskrevet i kapitlene 4 og 5 støtter EtherCIS både formatet FLATJSON og ECISFLAT. Det er ECISFLAT som ble brukt i masteroppgaven på grunn av bruken av openEHR sti for identifisering av elementene. Siden ECISFLAT er en variant av FLATJSON menes det at kravet er oppfylt. Om ønskelig kan en også bruke FLATJSON, som var formatet kravet spesifiserte. Dersom FLATJSON skal brukes må konverteringsfilene oppdateres til å være kompatible med FLATJSON.

7.2.3 Ikke-funksjonelle krav

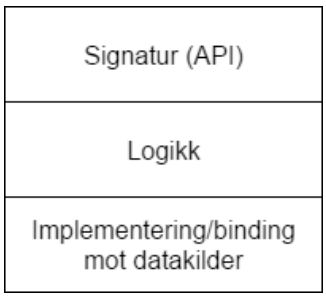
I masteroppgaven ble to ikke-funksjonelle krav spesifisert som påvirket hvordan FHIR bussen ble implementert og hvilke verktøy som ble valgt for openEHR pasientjournalsystemet. I dette avsnittet presenteres disse to kravene og en vurdering i hvor stor grad disse ble oppfylt.

Kjøre i en skytjeneste

Som nevnt tidligere er EtherCIS pasientjournalsystemet bygget som en Docker container og FHIR bussen som en Tomcat applikasjon. Disse teknologiene er mye brukt i dag og er støttet av de fleste skytjenester. Apache Tomcat er en teknologi som er ofte brukt for Java applikasjoner og har vært i bruk i flere år. Docker er en nyere teknologi som kanskje ikke støttes av alle skytjenestene, men å kjøre applikasjoner som Docker containere vil nok bli mer og mer utbredt med årene som kommer. I masteroppgaven fikk vi oppleve at den dedikerte versjonen av IBM Bluemix, MediCloud, ikke støttet Docker containere. Dette er et problem som er forårsaket av aktørene som styrer MediCloud og ikke av IBM Bluemix. IBM Bluemix støtter Docker og for å kunne bruke EtherCIS som en Docker container ble det satt opp en egen versjon av IBM Bluemix med funksjonalitet for Docker containere. I tillegg til IBM Bluemix har blant annet Amazon Web Services [74] og Microsoft Azure [75] støtte for Docker containere.

Vedlikeholdbarhet

Logikken for konvertering er abstrahert inn i en egen modul for å enklere kunne modifisere, fjerne og legge til funksjonalitet uten å påvirke FHIR bussen. FHIR bussen er også utviklet lagvis med egne klasser for hver funksjonalitet og FHIR ressurs. Dette medfører at vi kan modifisere funksjonaliteten for hver FHIR ressurs uten å påvirke de andre ressursene som FHIR bussen støtter. Figur 7.4 beskriver vår struktur av FHIR bussen som gjør det mulig å bevare vedlikeholdbarheten for systemet. Hver FHIR ressurs har tilhørende API klasser («provider» klasser), logikklasser og integrasjonsklasser. Dette gjør det mulig å vedlikeholde funksjonaliteten til en FHIR ressurs uten at det påvirker resten av FHIR bussen.



Figur 7.4: FHIR bussen er strukturert i tre lag for å bevare vedlikeholdbarheten av systemet. API laget inneholder selve *signaturen* for våre FHIR tjenester og er ikke påvirket av de andre lagene. Logikklaget håndterer beregninger og diverse konverteringer som må utføres. Nederste laget håndterer selve bindingen til de ulike bakenforliggende systemene som brukes som datakilder. Disse kan erstattes og endres uten av signaturen til våre tjenester berøres.

7.3 Tilgjengelig funksjonalitet i FHIR bussen

FHIR bussen har ulike tjenester tilgjengelig for brukerne. Flere av endepunktene som er tilgjengelige er et resultat av de funksjonelle kravene. I tabellene 7.2 og 7.3 er det listet opp alle endepunktene som er tilgjengelige i dag. For hvert endepunkt er det gitt en kort beskrivelse av endepunktet. Første tabell har en oversikt over endepunktene som bruker en egen intern database for lagring og henting av data, mens siste tabell inneholder alle endepunktene som bruker openEHR pasientjournalsystemet. Når en diagnose/problem, allergi, observation eller legemiddelordinering blir opprettet hentes pasientidentifikasjonen fra ressursen som sendes ved i forespørselen, og dermed blir ikke pasientidentifikatoren sendt ved i selve forespørselen. I Vedlegg D kan en studere en dokumentasjon over alle FHIR ressursene som støttes med en liste over endepunkter og tilgjengelige søkeparametre. I denne dokumentasjonen finnes også lenker for profilene publiserte via Simplifier.

Tilgjengelige tjenester i FHIR bussen			
HTTP metode	**REST API endepunkt**	**Funksjonalitet**	**Datakilde**
POST	/Patient	Oppretter en pasient	Cloudant database for lagring av pasient, openEHR system for oppretting av pasientjournal
PUT	/Patient/:id	Oppdaterer pasient	Cloudant database
GET	/Patient/:id	Henter pasient med id	Cloudant database
GET	/Patient	Henter alle pasienter	Cloudant database
GET	/Patient?organization=<orgId>	Henter alle pasienter i en organisasjon	Cloudant database
POST	/Practitioner	Oppretter en sykepleier/lege	Cloudant database
GET	/Practitioner/:id	Henter sykepleier med id	Cloudant database
PUT	/Practitioner/:id	Oppdaterer sykepleier	Cloudant database
GET	/Practitioner	Henter alle sykepleiere	Cloudant database
GET	/Practitioner?barcode=<barcode>	Henter sykepleier med strekkode	Cloudant database
POST	/Organization	Oppretter en organisasjon	Cloudant database
GET	/Organization/:id	Henter organisasjon med id	Cloudant database
GET	/Organization	Henter alle organisasjoner	Cloudant database
POST	/Medication	Oppretter medisin	Cloudant database
GET	/Medication/:id	Henter medisin med id	Cloudant database
GET	/Medication	Henter alle medisiner	Cloudant database
POST	/Encounter	Oppretter en ny hendelse	Cloudant database
GET	/Encounter/:id	Henter en hendelse med id	Cloudant database
GET	/Encounter?subject=<patId>	Henter alle hendelsene til en pasient	Cloudant database
POST	/Location	Oppretter et nytt sted (f.eks rom, avdeling eller seng)	Cloudant database
GET	/Location/:id	Henter sted med id	Cloudant database

Tabell 7.2: Liste med tilgjengelige endepunkter i FHIR bussen som bruker intern Cloudant database

Tilgjengelige tjenester i FHIR bussen			
HTTP metode	**REST API endepunkt**	**Funksjonalitet**	**Datakilde**
GET	/Login?username=<usr>&password =<pw>	Autentiserer en bruker og returnerer en session nøkkel dersom autentisert bruker. Nøkkelen legges ved forespørslene i «Header» seksjonen som «Ehr-Session»	Bruker pasientjournalsystemet for å autentisere brukeren
POST	/Observation	Oppretter en observasjon	openEHR pasientjournalsystem
DELETE	/Observation/:id	Fjerner observasjon med id	openEHR pasientjournalsystem
GET	/Observation/:id	Henter observation med id	openEHR pasientjournalsystem
GET	/Observation?patient=<patId>	Henter alle observasjoner til en pasient	openEHR pasientjournalsystem
GET	/MedicationStatement?patient=<patId>	Henter alle aktive medisiner til en pasient	openEHR pasientjournalsystem
POST	/MedicationRequest	Oppretter en legemiddelordinering	openEHR pasientjournalsystem
GET	/MedicationRequest/:id	Henter legemiddelordinering med id	openEHR pasientjournalsystem
GET	/MedicationRequest?patient=<patId>	Henter alle legemiddelordineringer til pasient	openEHR pasientjournalsystem
DELETE	/MedicationRequest/:id	Fjerner legemiddelordinering med ID	openEHR pasientjournalsystem
POST	/Condition	Oppretter en evaluering om en ny diagnose/problem	openEHR pasientjournalsystem
GET	/Condition/:id	Henter diagnose/problem med id	openEHR pasientjournalsystem
DELETE	/Condition/:id	Fjerner diagnose/problem med id	openEHR pasientjournalsystem
POST	/AllergyIntolerance	Oppretter en evaluering om allergi/intoleranse	openEHR pasientjournalsystem
GET	/AllergyIntolerance/:id	Henter allergi/intoleranse med id	openEHR pasientjournalsystem
DELETE	/AllergyIntolerance/:id	Fjerner allergi/intoleranse med id	openEHR pasientjournalsystem

Tabell 7.3: Liste med tilgjengelige endepunkter i FHIR bussen som bruker openEHR pasientjournalsystemet

7.4 Diskusjon

Vi startet masteroppgaven med å presentere motivasjonen for vårt arbeid og kom frem til ulike forskningsspørsmål vi ønsket å besvare i løpet av masteroppgaven. I de neste avsnittene vil vi gå gjennom forskningsspørsmålene som ikke er nevnt tidligere i kapittelet. Første avsnitt tar for seg det første og viktigste forskningsspørsmålet: «Hva er fordelene med en tjenestearkitektur som har et FHIR grensesnitt mellom klientapplikasjonene og de bakenforliggende systemene?». I samme avsnitt vil vi håndtere det andre forskningsspørsmålet: «Hvilke gevinster har en FHIR buss for et openEHR pasientjournalsystem?». Til slutt skal vi se på hvordan skytjenesten IBM Bluemix var med på å forbedre integrasjonen mellom klientapplikasjonen vår, FHIR bussen og openEHR pasientjournalsystemet.

7.4.1 Hva er fordelene med en tjenestearkitektur som har et FHIR grensesnitt mellom klientapplikasjonene og de bakenforliggende systemene?

Vi utviklet en arkitektur med en FHIR buss som tilbyr ulike tjenester for klientapplikasjonene. FHIR bussen kommuniserer med et openEHR basert pasientjournalsystem for å teste hvilke fordeler man får ved å tilby openEHR data via et FHIR grensesnitt. Denne arkitekturen er en mindre versjon av tjenestearkitekturen presentert i kapittel 1 som har flere bakenforliggende systemer. En slik tjenestearkitektur er for omfattende å implementere i masterprosjektet, og dermed fokuserte vi i masterprosjektet på fordelene for openEHR baserte pasientjournalsystemer.

For å få en ekspertevaluering av våre FHIR tjenester hadde vi et intervju med lederen av HL7 Norge om vår FHIR buss og vår tjenestearkitektur. Hun så store potensialer med våre tjenester, og mente vår løsning kunne være med som et startprosjekt for å tilgjengeliggjøre openEHR data via et FHIR grensesnitt. Dette er en løsning som leverandøren DIPS forsøker å gjøre i dag med sitt openEHR baserte pasientjournalsystem DIPS Arena, men som de ikke har kommet i mål med. Vi fikk også en tilbakemelding av en integrasjonsansvarlig for elektronisk kurve i Helse Nord. I lag med tilbakemeldingen fikk vi tilgang til FHIR responser fra deres eksisterende FHIR tjeneste. Disse kan vi ikke publisere, men vi fikk sammenlignet våre FHIR profiler med deres FHIR profiler. Vår struktur for FHIR profilene var tilnærmet like deres. Dette er bra for å forsikre oss at vår FHIR buss tilbyr data som følger en kjent struktur. Han likte godt vår metode med å eksponere AQL spørringer via FHIR tjenester. Våre tjenester er direkte basert på FHIR spesifikasjonen med et rent design og er dermed et

fint utgangspunkt for utvikling av FHIR tjenester uten å kombinere ulike dataformater som XML, JSON og noe fritekst. Disse kombinasjonene blir ofte brukt i dag for å representere et ekte domenekonsept som kommer fra eksisterende systemer som DIPS. Fordelene med vårt design er at vi ønsker å utnytte FHIR profilering fullt ut med bruk av utvidelser for å bevare en god klinisk datastruktur for å unngå å bruke fritekst.

De neste avsnittene tar for seg hvilke gevinster vi kan få ved å bruke vår tjenestearkitektur. Resultatene som er beskrevet her vil ofte være basert på integrasjonen mellom FHIR og openEHR. Dermed vil de samme resultatene som er beskrevet under også besvare forskningsspørsmålet; *Hvilke gevinster har en FHIR buss for et openEHR pasientjournalsystem?*

Kliniske testdata

Et stort problem for applikasjoner som utvikles i dag er mangel på kliniske testdata. I tillegg er det ingen formell datastandard som anbefales i Norge, og dermed blir det ofte etablert en egen datastruktur for hver enkel applikasjon som utviklerne implementerer. Flere av e-helsesystemene i Norge har enten ingen struktur på dataene eller bruker en intern struktur som ikke er offentliggjort. Dermed er det vanskelig for eksterne utviklere å implementere helserelaterte applikasjoner. Dette medfører at flere applikasjoner som blir utviklet i dag må skrives om når de skal integreres mot aktuelle systemer. FHIR bussen tilbyr utviklerne kliniske testdata som er basert på ekte kliniske data fra et system. Dette er viktig for å forsikre seg om at applikasjonene baserer seg på kliniske korrekte datastrukturer. Utviklerne kan da utvikle en applikasjon med testdata som er tilnærmet korrekt istedenfor å etablere deres egen datastruktur. Vi fikk tilsendt kasuistikker (typiske eksempler som ligner ekte kliniske data) fra en sykepleier ved Mottaksklinikken på Haukeland Universitetssykehus med 4 ulike pasienter med hver sine medisinlister med cirka 6-14 medisiner. Disse ble brukt som testdata for medisinapplikasjonen LEMI da applikasjonen ble modifisert. Videre vil det være ønskelig å få flere slike kasuistikker for å tilgjengeliggjøre mest mulig kliniske testdata som inneholder typiske eksempler som helsepersonellet håndterer i dag.

Abstrahering av komplekse data- og dokumentstrukturer

Flere datastandarder i dag, som blant annet openEHR, strukturerer helsedataene i komplekse dokumenter. For å håndtere dataene fra tjenester som baserer seg på disse standardene må utviklerne kjenne til både data- og dokumentstrukturen. Datastandardene kan være komplekse og dermed være utfordrende for en utvikler å lære seg. Å sette seg inn i disse

strukturene tar lang tid og kan være kostbart. FHIR spesifikasjonen er designet for å gi utviklerne tilgang til ressurser som er både lesbare og forståelige for en person som ikke har helserelatert bakgrunn. Istedenfor at alle utviklerne av klientapplikasjonene må lære seg datastandardene til de bakenforliggende systemene er det bare utviklerne av FHIR bussen som trenger å lære seg disse strukturene.

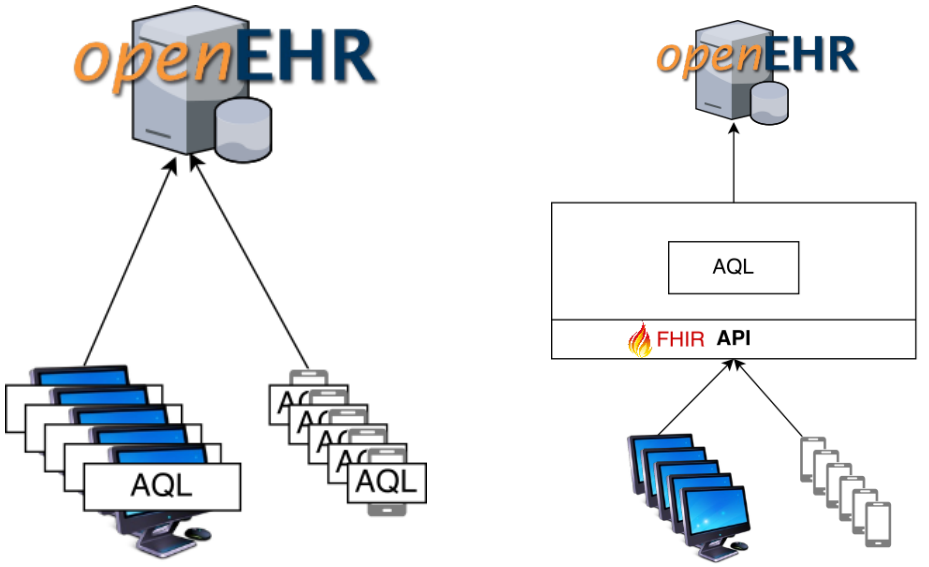
Utviklingsvennlig REST API

FHIR spesifikasjonen er designet med utviklerne i fokus. Dette medfører at REST grensesnittet basert på FHIR er utviklingsvennlig. Som vi så i avsnitt 7.2.1 oppfyller FHIR bussen nesten alle kravene og prinsippene, unntatt «Cacheable». Derimot er det lett å endre FHIR bussen slik at også dette kravet er oppfylt. FHIR spesifikasjonen inneholder god dokumentasjon for hvordan grensesnittet skal brukes og hvilke kliniske konsepter FHIR ressursene representerer. For hver ressurs eksisterer det eksempler hvor ressursene er fylt med relevant informasjon. FHIR grensesnittet følger de mest kjente retningslinjene [76] for hvordan et REST API bør struktureres. Siden FHIR bussen følger FHIR spesifikasjonen vil brukerne av FHIR bussen ha kjennskap til hvordan forespørslene skal struktureres dersom brukerne har erfaring med REST fra før. Dette medfører at det er enklere å kommunisere med FHIR bussen uten å bruke mye tid på å lære seg standarden.

Integrasjon mot de bakenforliggende systemene utføres i FHIR bussen

Tjenestene som FHIR bussen tilbyr i dag er enkle, noe som medfører at forespørselene sendt til FHIR bussen er relativt korte. Som vi så i kapittel 6 kan utviklerne unngå en mer avansert openEHR forespørsel ved å integrere seg mot FHIR bussen istedenfor openEHR pasientjournalsystemet. FHIR bussen tilbyr enkle tjenester for henting og lagring av data, og derfor fungerer det fint å bruke FHIR bussen som en type «konverteringsmodul» mellom brukerne og openEHR pasientjournalsystemet. Figur 7.5a viser hvordan kommunikasjonen vil foregå uten en FHIR buss. Her må hver klientapplikasjon opprette aktuell AQL spørring. Figur 7.5b viser hvordan kommunikasjonen foregår i dag ved hjelp av FHIR bussen. Klientapplikasjonene kommuniserer med FHIR bussen som håndterer opprettingen av AQL spørringene. Dette abstraherer nødvendigheten om domene- og implementasjonskunnskap til de bakenforliggende systemene.

Fordelene derimot med å integrere seg direkte mot openEHR pasientjournalsystemet er muligheten å utføre avanserte og komplekse spørringer etter data uten å utvide openEHR



(a) AQL spørring uten FHIR buss

(b) AQL spørring med FHIR buss, med hjelp at et utviklingsvennlig REST grensesnitt

Figur 7.5: Når klientapplikasjonene skal kommunisere direkte med openEHR pasientjournalsystemet må hver enkel applikasjon opprette AQL spørringen. Med en FHIR integrasjonsbuss vil AQL spørringen bli opprettet bare en gang i selve FHIR bussen.

tjenesten. FHIR er basert på et standard REST grensesnitt hvor en må spesifisere en mer avansert spørring ved å legge til passende parametre. Før en kan utføre en avansert spørring må FHIR tjenesten utvides med denne funksjonaliteten. Dette vil si at for hver ny spørring som ønskes å utføre mot openEHR pasientjournalsystemet, eller andre bakenforliggende e-helsesystemer, må FHIR bussen utvides. Det kan også oppstå situasjoner hvor FHIR mangler støtte for noen typer avanserte spørringer. Å utvide FHIR bussen kan fort ta lang tid og det kan være bedre å sende spørringene direkte til openEHR tjenesten. Dette gjelder allikevel kun i situasjoner hvor det er ønskelig å utføre noen få ulike avanserte spørringer. Dersom en er interessert i enkle tjenester, som henting av pasientdata, medisinlister og observasjoner for en pasient, kan en FHIR buss som abstraherer den avanserte spørringen være fordelaktig i det lengre løp. Dette gjelder også for mer avanserte tjenester vi vet vil bli mye brukt av flere ulike klientapplikasjoner.

Klientapplikasjonene er beskyttet mot endringer i de bakenforliggende systemene ved hjelp av løs kopling

En annen viktig fordel med å bruke en FHIR buss mellom brukerne og de bakenforliggende systemene er beskyttelse mot endringer. Klientapplikasjonene har en løs kopling til de

bakenforliggende systemene ved å kommunisere med FHIR bussen. Dersom et bakenforliggende system endrer sitt REST API eller erstattes så er det bare FHIR bussen som må oppdateres. Alle brukerne kjenner kun til tjenestene fra FHIR bussen og vil dermed ikke påvirkes. Dette medfører at en kan fint endre og oppdatere de bakenforliggende systemene uten ekstra arbeid med å oppdatere alle applikasjonene som bruker deres tjenester via FHIR bussen.

7.4.2 Hvordan kan en skyløsning være med på å forbedre integrasjonen mellom klienter og systemer?

En av fordelene med å bruke en skytjeneste er muligheten til å administrere alle applikasjonene og tjenestene på et sted. IBM Bluemix gjør det mulig å konfigurere hvor mange instanser som kjøres av en tjeneste og hvor mye minne som skal bli tildelt hver instans. Å kjøre flere instanser av en tjeneste gjør det mulig å håndtere flere forespørsler samtidig uten at det går utover ytelsen av tjenesten. MediCloud, en dedikert versjon av IBM Bluemix, har støtte for å kjøre våre egne applikasjoner og tjenester under kategorien «Cloud Foundry Apps» og IBM Bluemix tjenester (blant annet Cloudant database for lagring og IBM Watson for maskinlæring) under kategorien «Services». Vår interne database som lagrer medisinene, pasientene og sykepleierne er en Cloudant database som er en «Service» i IBM Bluemix. Denne kan kobles til våre applikasjoner som kjører i MediCloud. Når en IBM Bluemix tjeneste er koblet til våre applikasjoner eller tjenester blir det delt egne brukernavn, IP-adresser og passord ved hjelp av miljøvariabler. Det vil si at en slipper å lagre denne informasjonen i egne konfigurasjonsfiler. Lagring av brukernavn og passord i egne konfigureringsfiler er ikke en sikker løsning og ved å bruke skyløsningen IBM Bluemix unngår vi dette problemet. IBM Bluemix tilbyr også startprosjekter for ulike programmeringsspråk som er integrert mot en Cloudant database. Dette gjorde det enkelt for oss å komme i gang med utviklingen når vi skulle utvikle applikasjoner og tjenester som skulle kjøres i MediCloud. MediCloud tilbyr også loggføring av våre applikasjoner og tjenester. FHIR bussen bruker en «Logger» klasse som loggfører eventuelle feil som oppstår. Disse feilhåndteringsbeskjedene blir da loggført og tilgjengeliggjort via MediCloud. Dersom det har oppstått et problem med vår tjeneste er det enkelt å se i loggen for å finne ut hvor feilen oppstod. Dette kan spare flere timers arbeid dersom det er en komplisert feil som har oppstått. En skyløsning gir utviklerne av FHIR bussen også tilgang til å loggføre og monitorere alle forespørslene som er sendt til de bakenforliggende systemene via FHIR bussen. Dette er viktig innenfor helse for å kunne styre hvem som har tilgang til hva, monitorere hvem som sendte fore-

spørslene og når disse forespørslene ble sendt. Et slikt abstraksjonslag gjør det også mulig å tilgjengeliggjøre data fra systemer uten å måtte tilgjengeliggjøre de private grensesnittene til de bakenforliggende systemene. Dette er et viktig argument siden dersom en bruker har tilgang til grensesnittene så har de også direkte tilgang til systemene, noe en ikke ønsker for helsenettet i Norge. Vår FHIR integrasjonsbuss kjørende i en skytjeneste kan dermed være med på å løse dette problemet.

7.4.3 Relatert arbeid

I masterprosjektet ble det forsøkt å finne relevante studier som bruker helsestandarden HL7 FHIR som en felles kommunikasjonskanal mellom brukerne og systemer. Da masteroppgaven ble skrevet fant vi ikke et lignende arbeid med å bruke en FHIR integrasjonsbuss (eller modul) ovenfor et openEHR basert pasientjournalsystem. Det nærmeste vi kom et relevant forskningsarbeid var doktorgradsprosjektet til Diego Boscá Tomás [34] som i kapittel 2 beskriver hvordan en kan generere openEHR arketyper (her også kalt for FHIR arketyper) utifra FHIR ressurser, og et masterprosjekt av Eneimi Allwell-Brown som sammenligner openEHR og FHIR på et overordnet nivå [28]. Derimot fant vi fire ulike studier hvor FHIR ble brukt som et integrasjonslag ovenfor helsestandardene OpenMRS [6,58], HL7 CDA [62], i2b2 [60] og OMOP CDM [63]. Deres forskingsarbeid er dokumentert tidligere i masteroppgaven i avsnitt 2.5. Vårt arbeid med bruk av FHIR som en integrasjonsbuss mellom brukerne og et openEHR basert system er en utvidelse av disse fire forskingsprosjektene. De fire forskingsprosjektene, i lag med dette masterprosjektet, er med på å styrke troverdigheten til fordelene med å innføre en FHIR integrasjonsbuss mot diverse bakenforliggende systemer. I alle våre forskingsprosjekter ble det avduket store fordeler for både henting og lagring av data via et FHIR grensesnitt. I tabell 7.4 listes de viktigste punktene med våre forskningsprosjekter som har et FHIR grensesnitt mot bakenforliggende systemer.

Selv om vårt arbeid med å bruke en FHIR tjeneste ovenfor et openEHR basert pasientjournalsystem er en lignende løsning som er beskrevet i de fire relevante forskningsprosjektene, er det noen viktige punkter som skiller vårt forskingsprosjekt fra deres arbeid. Vårt masterprosjekt har evaluert hvordan FHIR kan brukes i en større tjenestearkitektur for å forbedre integrasjonen mellom klienter og bakenforliggende systemer. Vi har også sett hvordan denne tjenestearkitekturen kan være med på å tilgjengeliggjøre kliniske strukturerte testdata. Våre resultater er også bygget videre på resultatene som ble presentert i de fire relevante forskningsprosjektene. Det som er ulikt fra vårt prosjekt er at forfatterne av de andre forskningsprosjektene fokuserte hovedsakelig på å forbedre integrasjonen mot én spesiell stan-

dard. Vi ønsket å ta et steg videre og fokuserte på en større arkitektur for å kombinere flere komplekse e-helsesystemer og tilgjengeliggjøre disse dataene gjennom et FHIR grensesnitt. Dette testet vi ved å abstrahere den komplekse openEHR dokumentstrukturen gjennom FHIR bussen, hvor FHIR bussen er strukturert for å kunne bruke andre e-helsesystemer som datakilder.

Sammenligning masterprosjekt med relatert arbeid					
Tema	**OpenMRS**	**i2b2**	**HL7 CDA**	**OMOP CDM**	**openEHR**
Mål	Bedre grensesnitt mot OpenMRS system	Forenklet og sikker tilgang til registerdata for kliniske applikasjoner	Lettere tilgang til ELGA CDA dokumenter	Tilgjengeliggjøre kliniske prediktive modeller som FHIR ressurser	Abstrahere kompleks dokumentstruktur ved hjelp av FHIR tjeneste
Arkitektur	FHIR modul med to lag; Web lag og API (service) lag.	«FHIR Cell» delt opp i Authentication, REST API, Cache, i2b2-to-fhir converter, semantikk og query engine.	FHIR modul ovenfor HL7 CDA, ingen informasjon om intern systemarkitektur	FHIR tjeneste mellom klienter og «Deployet Predictive Model» komponent.	FHIR buss med tre lag; API, logikk og integrasjonslag. Ekstern konverteringsmodul
FHIR ressurser	Patient, Person, Location, Observation, Encounter, AllergyIntolerance	Patient, Medication, MedicationPrescription, Observation, Condition	Composition, Patient, Practitioner, Medication, MedicationDispense, MedicationOrder.	RiskAssessment	Observation, Patient, Practitioner, Organization, Medication, MedicationRequest, Condition, AllergyIntolerance, Location, Encounter
Validering og kvalitet	Integrert SMART-on-FHIR applikasjon	Integrert SMART-on-FHIR applikasjon	«Proof-of-concept» implementasjon	Måling av responstid på forespørsler og test av robusthet	Integrering av LEMI, validering som RESTfull tjeneste og intervju med lederen av HL7 Norge
Konvertering	Manuell konvertering direkte i Java	Manuell oppretting av XML dokumenter, SQL spørringer og bruk av XQuery	Konverteringsfil for hvert CDA dokument. Bruker XPath til CDA element	Ingen, OMOP CDM kalkulerer et tall som lagres i feltet «prediction» i RiskAssessment	Konverteringsfil for hver openEHR arketype. Bruker kombinasjon av openEHR sti og FHIR element

Tabell 7.4: Sammenligning av masterprosjektet med relatert arbeid. Tabellen sammenligner vårt masterprosjekt med de fire forskingsprosjektene som fokuserte på konvertering mellom FHIR og standardene i2b2, OpenMRS, HL7 CDA og OMOP CDM.

7.5 Begrensninger

Det er noen begrensninger med konverteringsløsningen som er viktig å presisere. I dag støtter konverteringsmodulen følgende openEHR datatyper; quantity, codedtext, «other», «convert» og «choice». De to første datatypene er openEHR datatyper. Andre datatyper som ikke krever ekstra formatering faller under kategorien «other», som dekker alle datatyper som string, integer og boolean. «convert» og «choice» er to interne kategorier i konverteringsmodulen som brukes for å konvertere dataformater eller spesifisere et «valg» fra ADL versjonen av arketypen. Dersom en ønsker å konvertere arketyper som har andre datatyper må en utvide konverteringsmodulen både for å konvertere fra og til en openEHR arketype.

FHIR bussen er oppdatert til å bruke den nyeste versjonen STU3 (Standard for Trial Use 3). Det er dessverre flere implementasjonsguider her i Norge som fremdeles har DSTU2 (Draft Standard for Trial Use 2) profiler. Blant annet tilbyr KULE prosjektet i Helse Vest profiler for medisin og ordinering, men disse er basert på DSTU2 ressurser. Disse profilene er tilgjengelige via nettsiden Simplifier; `https://simplifier.net/HelseVestdosagemappi`. Siden FHIR bussen bruker STU3 kan man ikke benytte seg av disse profilene med mindre en lager en tilnærmet profil basert på STU3. Forskjellene mellom STU3 og DSTU2 versjonene er ikke store, men noen ressurser har fått en stor nok endring med obligatoriske felter som gjør det vanskelig å oversette en DSTU2 profil over til en STU3 profil. Figurene C.3 og C.4 i Vedlegg C viser to utsnitt fra ressursene MedicationOrder fra DSTU2 og tilsvarende MedicationRequest ressurs fra STU3 versjonen. Her kan en sammenligne feltene i hver ressurs.

Kapittel 8

Konklusjon

Målet med masterprosjektet var å evaluere en tjenestearkitektur hvor klientapplikasjonene kommuniserer med bakenforliggende e-helsesystemer ved hjelp av en FHIR integrasjonsbuss. Basert på resultatene vi diskuterte i kapittel 7 kom vi frem til at FHIR bussen kunne tilby brukerne av de bakenforliggende systemene en lettere tilgang til pasientdata og gjorde det enklere å tilby klinisk korrekte testdata. Dette på grunn av strukturen til FHIR ressursene og strukturen til selve FHIR grensesnittet. I masteroppgaven ble forskningsspørsmålene besvart ved hjelp av en implementasjon av en FHIR buss som kommuniserer med et openEHR basert pasientjournalsystem. FHIR bussen, pasientjournalsystemet og vår konverteringsmodul kjører alle i en skytjeneste, som gjorde det lett å både starte flere instanser om nødvendig og undersøke loggfilene dersom et problem oppstod. Standarden openEHR for strukturering av helsedata utfører jobben suverent. Ved hjelp av standardens to-nivå modellering er pasientjournalsystemet implementert ved hjelp av en referansemodell. Alle kliniske konsepter bygger på referansemodellen og tilbyr muligheten for klinikere og andre domeneeksperter å opprette arketyper for alle tenkelige kliniske konsepter. Dette kan gjøres uten å oppdatere pasientjournalsystemet. På grunn av openEHR sin komplekse data- og dokumentstruktur kreves det at brukerne av openEHR sitt REST API har både domenekunnskap og kunnskap om hvordan alle dataene er strukturerte. I tillegg kan vanlige forespørsler bli unødvendig lange på grunn av at de må sendes inn som en AQL spørring. FHIR bussen lar brukerne sende forespørsler til openEHR pasientjournalsystemet uten at brukerne må ha domenekunnskap og implementasjonskunnskap om datastrukturene. Disse FHIR forespørslene er mer deklarative i forhold til AQL forespørslene. Dette vil si at FHIR forespørslene inneholder bare hva vi ønsker å utføre, men hvordan forespørselen prosesseres og hvor dataene er

lagret er ikke spesifisert. En AQL forespørsel derimot spesifiserer konkret hvor informasjonen er lagret ved hjelp av en eller flere openEHR stier.

Siden FHIR bussen er basert på FHIR spesifikasjonen har brukerne tilgang til et REST API som følger kjente retningslinjer for hvordan et REST API bør struktureres. Dette gjør det enklere for brukerne å både hente og lagre openEHR data via FHIR bussen siden REST er en kjent tjenestestruktur. Dessverre må FHIR bussen endres for hver ny funksjonalitet som legges til og for hver ny FHIR ressurs som skal støttes. FHIR spesifikasjonen er altså ikke basert på en to-nivå modellering. De fordelene FHIR bussen har for et openEHR basert pasientjournalsystem vil også være gyldige for andre bakenforliggende systemer som enten har komplekse datastrukturer eller et avansert REST API. Dette så vi var gjeldende for standardene OpenMRS, i2b2, HL7 CDA og OMOP CDM som i de fire relevante forskningsprosjektene brukte et FHIR grensesnitt for å forbedre og forenkle integrasjonen for både henting og lagring av data.

Både FHIR og openEHR tilbyr en spesifikasjon for hvordan dataene kan struktures og deles, og kan i teorien brukes alene i et system. Basert på diskusjonen i kapittel 7 er det større fordeler enn ulemper med å bruke vår tjenestearkitektur som baserer seg på å kombinere det beste ved FHIR og det beste ved openEHR. FHIR bussen bruker en konverteringsmodul for å konvertere mellom FHIR ressursene og openEHR arketypene som fint kan erstattes, endres eller utvides om ønskelig. FHIR bussen er også strukturert i ulike lag for å separere grensesnittet fra integrasjonen mot de bakenforliggende systemene for å kunne utvide FHIR bussen til å bruke andre datakilder enn vårt openEHR pasientjournalsystem. Konverteringsprosessen baserer seg på en direkte naiv konvertering hvor hvert obligatoriske datapunkt i arketypen blir konvertert over til tilhørende datapunkt i en eller flere ressurser. Her er det store muligheter for videre forskning med å sammenligne ulike konverteringsmetoder for hvordan FHIR ressurser kan konverteres fra og til andre helsestandarder.

8.1 Videre forskning

Det er fremdeles mye som kan gjøres innenfor vårt forskningsområde i masterprosjektet. I løpet av vårt arbeid har det dukket opp flere mulige fremtidige forskningsprosjekter som kan være interessante å undersøke. De neste fem avsnittene beskriver ulike forskningsområder som bygger på vårt masterprosjekt.

8.1.1 Utvide FHIR med partisjoner

Det vil være interessant å forske videre på konverteringsmetoden foreslått av Thomas Beale, en kjent senior arkitekt for openEHR.org. Konverteringsmetoden baserer seg på å utvide FHIR spesifikasjonen med egne partisjoner for hver standard som en ønsker å konvertere til FHIR [77, 78]. Hver partisjon har nye ressurser som etterligner entitetene fra referansemodellen for standarden vi ønsker å konvertere fra. I vårt eksempel kunne vi da lagt til en openEHR partisjon med egne FHIR ressurser for COMPOSITION, OBSERVATION, ACTION, EVALUATION og INSTRUCTION arketyper hvor ressursene har samme struktur som openEHR arketypene, men bruker FHIR sine datatyper, identifikasjon og infrastruktur. Deretter kan vi profilere disse for hver spesifikk arketype for et klinisk konsept. FHIR ressursene som eksisterer i dag ville da blitt brukt for systemene som ikke har sine data strukturert, men ønsker å strukturere dataene sine.

8.1.2 Konvertere flere arketyper og ressurser

I dag støtter FHIR bussen ressursene Observation, Patient, Practitioner, Medication, MedicationRequest, MedicationStatement, Condition, Location, Encounter og AllergyIntolerance. Det vil være interessant å konvertere flere arketyper over til FHIR ressurser for å undersøke om en støter på noen utfordringer som vi ikke har hatt. I versjon STU3 ble det lagt til ressurser for arbeidsflyten til klinikerne, blant annet ressursen Task. Task er en ressurs for å loggføre en aktivitet som kan bli utført. Denne ressursen blir oppdatert med ny status for å kunne overvåke statusen til en aktivitet. I fremtiden vil det nok være ønskelig å oppdatere FHIR bussen med funksjonalitet for Task ressurser. Blant annet vil nok medisinapplikasjonen LEMI kunne utnytte disse ressursene for å loggføre hvilken status en medisinadministrering har.

8.1.3 Sammenligne antall lignende datapunkter

Ressursene som FHIR bussen støtter i dag er profilert for å inneholde de obligatoriske datapunktene som både FHIR ressursen og tilhørende openEHR arketype har. Ressursene for medisin og ordinering har i tillegg noen flere datapunkter enn de obligatoriske for å tilfredstille kravene for medisinapplikasjonen LEMI. Det kan være interessant å undersøke videre hvor stor andel av openEHR datapunktene som kan konverteres over til tilhørende FHIR datapunkter i en eller flere ressurser. En kan da se på gjennomsnittlig prosentandel av arketypen som kan konverteres over til tilhørende ressurs(er) uten ekstra utvidelser. Deretter

kan vi se på gjennomsnittlig prosentandel av datapunktene fra openEHR arketypen som krever at FHIR ressursen blir profilert med nye utvidelser.

8.1.4 Sikkerhet

Som vi beskrev i avsnitt 4.3 har vi ikke evaluert sikkerheten hos FHIR og openEHR. Det vil være interessant å se på sikkerhetsaspektet for standardene som kjører i skytjenesten IBM Bluemix. Vår arkitektur som har en FHIR buss kjørende i en skytjeneste er tenkt å kunne være en aktuell arkitektur for hvordan dagens helsenett kan bli tilgjengelig for sikkerhetsklarerte applikasjoner og systemer. For å komme dit må vi undersøke i hvor stor grad en skytjeneste kan være sikker nok for å få til en slik løsning.

8.1.5 Automatisk AQL oppretting

Hver AQL spørring som utføres i dag er hardkodet i FHIR bussen. For videre arbeid kan en undersøke hvordan vi kan konvertere en forespørsel som blir sendt til FHIR bussen over til en tilsvarende AQL spørring. Dersom vi får til en slik konvertering kan vi både konvertere forespørslene og dataene som blir sendt inn. Denne konverteringen kan også abstraheres inn i egen modul slik at konverteringsmetoden av forespørslene kan endres uten at FHIR bussen må endres.

Bibliografi

[1] B. Haarbrandt, E. Tute, and M. Marschollek. Automated population of an i2b2 clinical data warehouse from an openEHR-based data repository. *J Biomed Inform*, 63:277–294, Oct 2016.

[2] Jorge Almeida, Samuel Frade, and Ricardo Cruz-Correia. Exporting data from an openehr repository to standard formats. *Procedia Technology*, 16:1391 – 1396, 2014.

[3] A. Zafar and B. E. Dixon. Pulling back the covers: technical lessons of a real-world health information exchange. *Stud Health Technol Inform*, 129(Pt 1):488–492, 2007.

[4] ihtsdo.org -. `http://www.ihtsdo.org/snomed-ct/what-is-snomed-ct`. [Hentet 27-11-2016].

[5] K. C. Stange. The problem of fragmentation and the need for integrative solutions. *Ann Fam Med*, 7(2):100–103, 2009.

[6] S. N. Kasthurirathne, B. Mamlin, H. Kumara, G. Grieve, and P. Biondich. Enabling Better Interoperability for HealthCare: Lessons in Developing a Standards Based Application Programing Interface for Electronic Medical Record Systems. *J Med Syst*, 39(11):182, Nov 2015.

[7] Meona GmbH. Hjemme - meona gmbh. `http://no.meona.de/`. [Hentet 06-02-2017].

[8] DIPS ASA. Dips - leverandør av elektronisk pasientjournal - epj. `https://www.dips.no/`. [Hentet 06-02-2017].

[9] Checkware. Checkware (no) - checkware. `http://checkware.com/`. [Hentet 06-02-2017].

[10] Digin. Digin.no. `http://www.digin.no/news/gratis-foredrag-18-feb-medicloud`. [Hentet 11-06-2017].

[11] ISO. Health informatics — Electronic healthrecord — Definition, scope and context . `http://tc215.behdasht.gov.ir/uploads/244_514_ISO_TR_20514_2005(E).pdf/`, 2005. [Hentet 25-11-2016].

[12] S. Ajami and R. Arab-Chadegani. Barriers to implement Electronic Health Records (EHRs). *Mater Sociomed*, 25(3):213–215, 2013.

[13] HIMSS. What is interoperability? `http://www.himss.org/library/interoperability-standards/what-is-interoperability`, April 2013. [Hentet 22-04-2017].

[14] Leykun Melkamu Gebeyehu. Enabling medical research on clinically collected data using openEHR archetypes. Master's thesis, Universitetet i Tromsø, Norge, 2012.

[15] Ovidiu Petrun Stan. Techniques for ensuring interoperability in an electronic health record. http://old.utcluj.ro/download/doctorat/Rezumat_Ovidiu_Petru_Stan.pdf. [Hentet 28-11-2016].

[16] L. Bird, A. Goodchild, and Z. Tun. Tun z: Experiences with a two-level modelling approach to electronic health records. In *Journal of Research and Practice in Information Technology 2003*, 2003.

[17] Philip Scott. Meeting the challenges of healthcare interoperability. *Healthcare IT Management*, 4(3):24–25, 2009.

[18] Hapi fhir - the open source fhir api for java. http://hapifhir.io/. [Hentet 02-06-2017].

[19] openehr website. http://www.openehr.org/home. [Hentet 28-11-2016].

[20] Peter Schloeffel. Current EHR Developments: an Australian and International Perspective. *Health Care and Informatics Review Online*, 2004.

[21] Ngamnij Arch-int and Somjit Arch-int. Semantic ontology mapping for interoperability of learning resource systems using a rule-based reasoning approach. *Expert Syst. Appl.*, 40(18):7428–7443, 2013.

[22] Kotchakorn Banlue, Ngamnij Arch-int, and Somjit Arch-int. Ontology mapping and rule-based inference for learning resource integration. *J. Inform. and Commun. Convergence Engineering*, 14(2), 2016.

[23] What is health it (health information technology)? - definition from whatis.com. http://searchhealthit.techtarget.com/definition/Health-IT-information-technology. [Hentet 20-03-2017].

[24] What is health information technology (health it)? | patients & families | healthit.gov. https://www.healthit.gov/patients-families/basics-health-it. [Hentet 20-03-2017].

[25] Who | international classification of diseases. http://www.who.int/classifications/icd/en/. [Hentet 28-11-2016].

[26] R. H. Dolin, L. Alschuler, S. Boyer, C. Beebe, F. M. Behlen, P. V. Biron, and A. Shabo Shvo. HL7 Clinical Document Architecture, Release 2. *J Am Med Inform Assoc*, 13(1):30–39, 2006.

[27] Clinical information modeling initiative. http://opencimi.org/about_cimi. [Hentet 28-11-2016].

[28] Eneimi Allwell-Brown. A Comparative Analysis of HL7 FHIR and openEHR for Electronic Aggregation, Exchange and Reuse of Patient Data in Acute Care. Master's thesis, Stockholm University, Karolinska Institutet, Sweden, 2016.

[29] Ehrsrle\ehrsrle - fhir v1.0.2. http://www.hl7.org/fhir/ehrsrle/ehrsrle.html. [Hentet 27-11-2016].

[30] openehr rest apis - specifications - openehr wiki. https://openehr.atlassian.net/wiki/display/spec/openEHR+REST+APIs. [Hentet 27-11-2016].

[31] Todd Fredrich. Restful service best practices - recommendations for creating web services. http://www.restapitutorial.com/media/RESTful_Best_Practices-v1_1.pdf, Mai 2012. [Hentet 22-04-2017].

[32] Clinical terminology - australian digital health agency. http://www.digitalhealth.gov.au/get-started-with-digital-health/what-is-digital-health/clinical-terminology. [Hentet 27-11-2016].

[33] SNOMED International Anna Adelof. Norway joins snomed international as our 30th member. http://www.snomed.org/news-articles/norway-joins-snomed-international-as-our-30th-member. [Hentet 22-04-2017].

[34] Diego Boscá Tomás. Detailed clinical models and their relation with electronic health records. Master's thesis, Polytechnic University of Valencia, Valencia, Spain, November 2015.

[35] Clem mcdonald | u.s. national library of medicine. https://lhncbc.nlm.nih.gov/personnel/clem-mcdonald. [Hentet 27-11-2016].

[36] About loinc — loinc. https://loinc.org/background. [Hentet 27-11-2016].

[37] Regenstrief, the SNOMED International are working together to link LOINC, and SNOMED CT. Snomed international — loinc. https://loinc.org/collaboration/snomed-international/. [Hentet 22-04-2017].

[38] Daniel J. Vreeman. Guidelines for using loinc and snomed ct together. https://danielvreeman.com/guidelines-for-using-loinc-and-snomed-ct-together-without-overlap/. [Hentet 22-04-2017].

[39] Getting started with loinc — loinc. https://loinc.org/get-started/02.html. [Hentet 27-11-2016].

[40] The openEHR Foundation. openehr - ehr information model. http://www.openehr.org/releases/RM/latest/docs/ehr/ehr.html#latest_issue_date. [Hentet on 09-01-2017].

[41] Sam Heard Heather Leslie. Archetypes 101. HIC, 2006.

[42] Heather Leslie. Introduction to archetypes and archetype classes - health information models - openehr wiki. https://openehr.atlassian.net/wiki/display/healthmod/Introduction+to+Archetypes+and+Archetype+classes. [Hentet 08-03-2017].

[43] Home - | ocean health systems. http://oceanhealthsystems.com/. [Hentet 20-03-2017].

[44] openehr - ocean informatics. http://www.openehr.org/industry_partners/ocean_informatics. [Hentet 20-03-2017].

[45] Sam Heard Thomas Beale. openehr architecture overview. *The openEHR Foundation*, 2008.

[46] The openEHR Foundation. 10 archetypes and templates. http://www.openehr.org/releases/1.0.2/html/architecture/overview/Output/archetyping.html. [Hentet 28-01-2017].

[47] openEHR Foundation. openehr medinfo2015 brazil sponsor session. http://www.slideshare.net/openehr/openehr-medinfo2015-brazil-sponsor-session, August 2015. [Hentet 28-01-2017].

[48] openEHR Foundation. Reference model (rm) - latest. http://www.openehr.org/releases/RM/latest/docs/index. [Hentet 22-04-2017].

[49] The openEHR Foundation. 12 terminology in openehr. http://www.openehr.org/releases/1.0.2/html/architecture/overview/Output/terminology.html. [Hentet 11-01-2017].

[50] The openEHR Foundation. openehr architecture overview. http://www.openehr.org/releases/BASE/Release-1.0.3/docs/architecture_overview/architecture_overview.html#_openehr_package_structure. [Hentet on 09-01-2017].

[51] DIPS AS. Verdensleder på open ehr | dips. https://www.dips.com/no/verdensleder-pa-open-ehr. [Hentet 27-02-2017].

[52] FHIR Infrastructure Work Group. Introducing hl7 fhir. https://www.hl7.org/fhir/summary.html. [Hentet on 09-01-2017].

[53] FHIR Infrastructure Work Group. Extensibility - fhir v1.0.2. https://www.hl7.org/fhir/extensibility.html. [Hentet 11-01-2017].

[54] FHIR Infrastructure Work Group. Base resource - fhir v1.0.2. https://www.hl7.org/fhir/resource.html. [Hentet 10-01-2017].

[55] FHIR Infrastructure Work Group. Profiling - fhir v1.0.2. http://www.hl7.org/FHIR/profiling.html. [Hentet 15-01-2017].

[56] Furore. Forge conformance designer. http://fhir.furore.com/Forge. [Hentet 11-01-2017].

[57] HL7 Norge. Hl7 fhir - norsk profil. http://www.hl7.no/index.php/standarder/hl7-fhir-norsk. [Hentet 11-01-2017].

[58] S. N. Kasthurirathne, B. Mamlin, G. Grieve, and P. Biondich. Towards Standardized Patient Data Exchange: Integrating a FHIR Based API for the Open Medical Record System. *Stud Health Technol Inform*, 216:932, 2015.

[59] B. W. Mamlin, P. G. Biondich, B. A. Wolfe, H. Fraser, D. Jazayeri, C. Allen, J. Miranda, and W. M. Tierney. Cooking up an open source EMR for developing countries: OpenMRS - a recipe for successful collaboration. *AMIA Annu Symp Proc*, pages 529–533, 2006.

[60] K. B. Wagholikar, J. C. Mandel, J. G. Klann, N. Wattanasin, M. Mendis, C. G. Chute, K. D. Mandl, and S. N. Murphy. SMART-on-FHIR implemented over i2b2. *J Am Med Inform Assoc*, Jun 2016.

[61] Hl7 standards product brief - cda® release 2. http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7. [Hentet 20-04-2017].

[62] C. Rinner and G. Duftschmid. Bridging the Gap between HL7 CDA and HL7 FHIR: A JSON Based Mapping. *Stud Health Technol Inform*, 223:100–106, 2016.

[63] M. Khalilia, M. Choi, A. Henderson, S. Iyengar, M. Braunstein, and J. Sun. Clinical Predictive Modeling Development and Deployment through FHIR Web Services. *AMIA Annu Symp Proc*, 2015:717–726, 2015.

[64] Patricia Potter, Laurie Wolf, Stuart Boxerman, Deborah Grayson, and Jennifer Sledge Clay Dunagan. An analysis of nurses ' cognitive work: A new perspective for understanding medical errors, 2015.

[65] Vimla L. Patel and David R. Kaufman. *Cognitive Science and Biomedical Informatics*, pages 109–148. Springer London, London, 2014.

[66] Elisabeth Wigaard Trine Lise Bakken. Sensoriske dysfunksjoner og kognitiv overbelastning hos mennesker med utviklingshemning | stiftelsen sor. SOR Rapport nr 4, 2015. [Hentet 11-03-2017].

[67] © 2017 Centre for Global eHealth Innovation. James agnew - centre for global ehealth innovation. http://ehealthinnovation.org/people/james-agnew/. [Hentet 25-02-2017].

[68] marand. Vendor-neutral, open health data platform | marand. http://www.marand.com/. [Hentet 02-06-2017].

[69] FHIR Infrastructure Work Group. Valueset-observation-category - fhir v1.0.2. http://www.hl7.org/fhir/valueset-observation-category.html. [Hentet 02-02-2017].

[70] Tonje Grimstad. - hundrevis dør av feilmedisinering - nrk norge - oversikt over nyheter fra ulike deler av landet. https://www.nrk.no/norge/--hundrevis-dor-av-feilmedisinering-1.6200529. [Hentet 21-04-2017].

[71] Xiaoliang Wang. *Towards Correct Modelling and Model Transformation in DPF*. PhD thesis, University of Bergen, 2016.

[72] The openEHR Foundation. terminology.pdf. http://www.openehr.org/releases/1.0.2/architecture/terminology.pdf. [Hentet 03-01-2017].

[73] Postdot Technologies. Postman | supercharge your api workflow. https://www.getpostman.com/. [Hentet 21-04-2017].

[74] What is docker? – amazon web services (aws). https://aws.amazon.com/docker/?sc_channel=PS&sc_campaign=acquisition_ND&sc_publisher=google&sc_medium=docker_b&sc_content=docker_general_e&sc_detail=amazon%20web%20services%20docker&sc_category=docker&sc_segment=161186757331&sc_matchtype=e&sc_country=ND&s_kwcid=AL!4422!3!161186757331!e!!g!!amazon%20web%20services%20docker&ef_id=VxEcHwAAAM-tTvI@:20170427162350:s. [Hentet 21-04-2017].

[75] Azure container service | docker swarm | microsoft azure. https://azure.microsoft.com/en-us/services/container-service/. [Hentet 21-04-2017].

[76] Mahesh Haldar. Restful api designing guidelines — the best practices. https://hackernoon.com/restful-api-designing-guidelines-the-best-practices-60e1d954e7c9. [Hentet 21-04-2017].

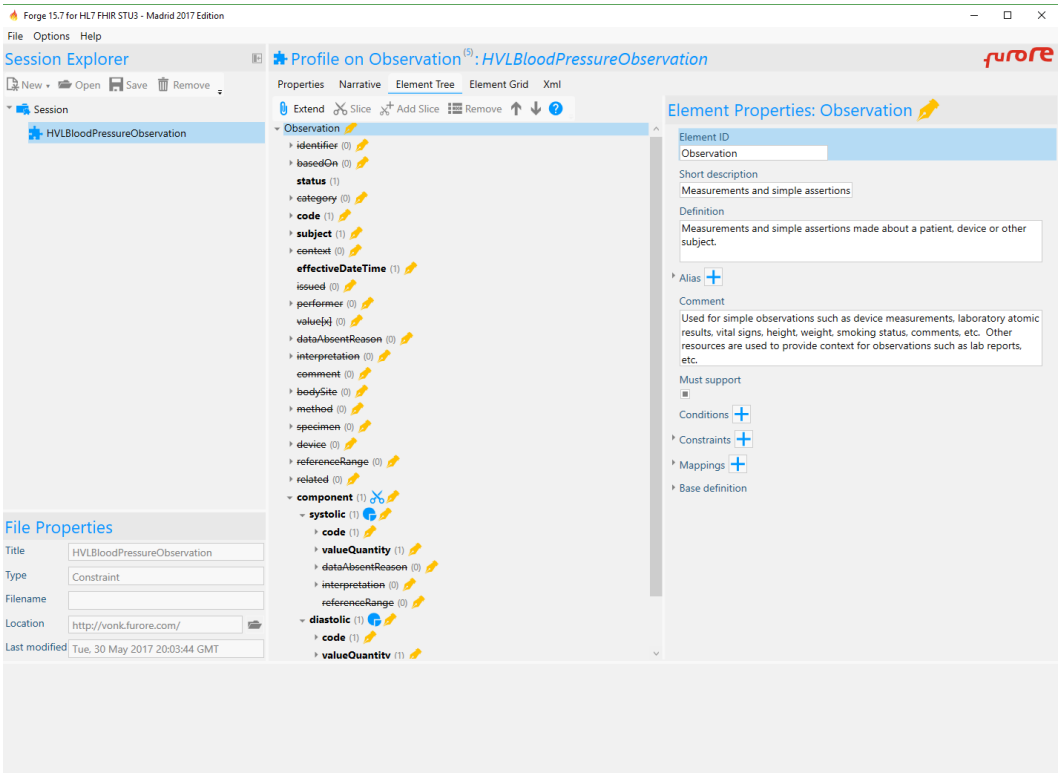
[77] Thomas Beale. Making fhir work for everybody | woland's cat. https://wolandscat.net/2015/12/20/making-fhir-work-for-everybody/. [Hentet 21-04-2017].

[78] Thomas Beale. Fhir compared to openehr | woland's cat. https://wolandscat.net/2017/01/29/fhir-compared-to-openehr/#more-1113. [Hentet 21-04-2017].

Vedlegg A

Bruk av verktøyet Forge

Profilering av blodtrykk

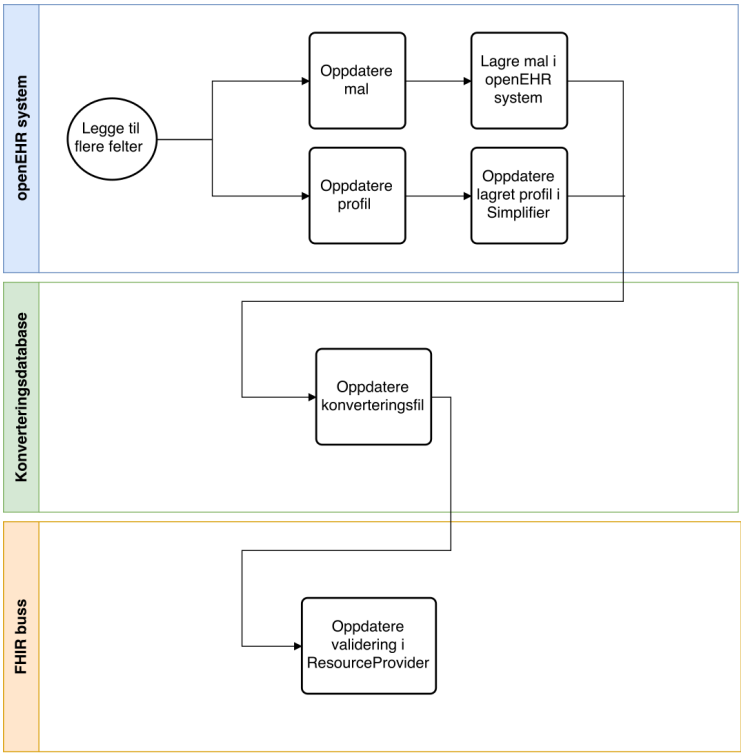


Figur A.1: Utsnitt av hvordan konseptet blodtrykk kan profileres ved hjelp av Forge. Elementene med strek over er fjernet fra ressursen, mens elementene med tykk skrift er obligatoriske. De obligatoriske feltene er *status*, *code*, *subject*, *effectiveDateTime* og komponentene *systolic* og *diastolic*. Profilen har også en liste som er delt opp; *component*. Listen er delt opp med tilhørende systolisk og diastolisk komponent. Disse to komponentene har verdifeltet satt til datatype «Quantity».

Vedlegg B

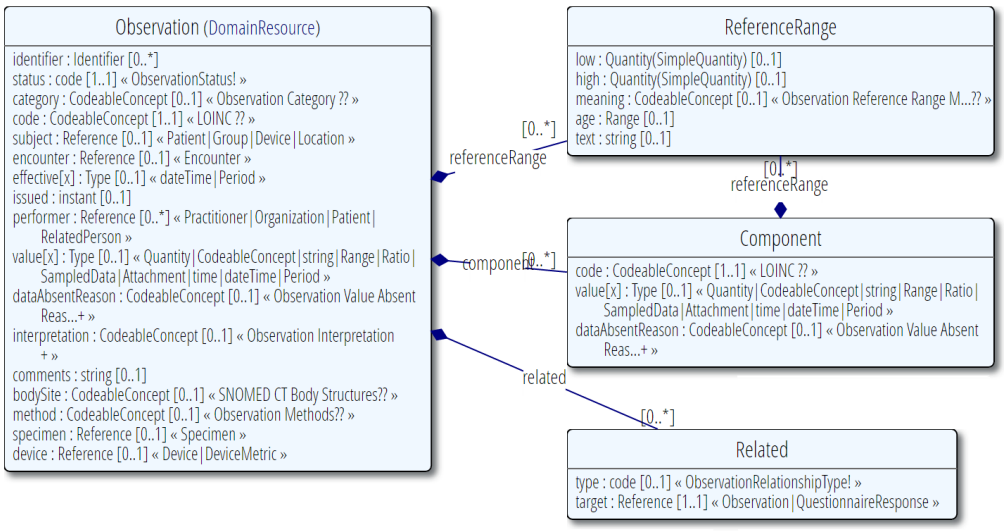
Detaljerte modeller og diagrammer

Arbeidsflyt



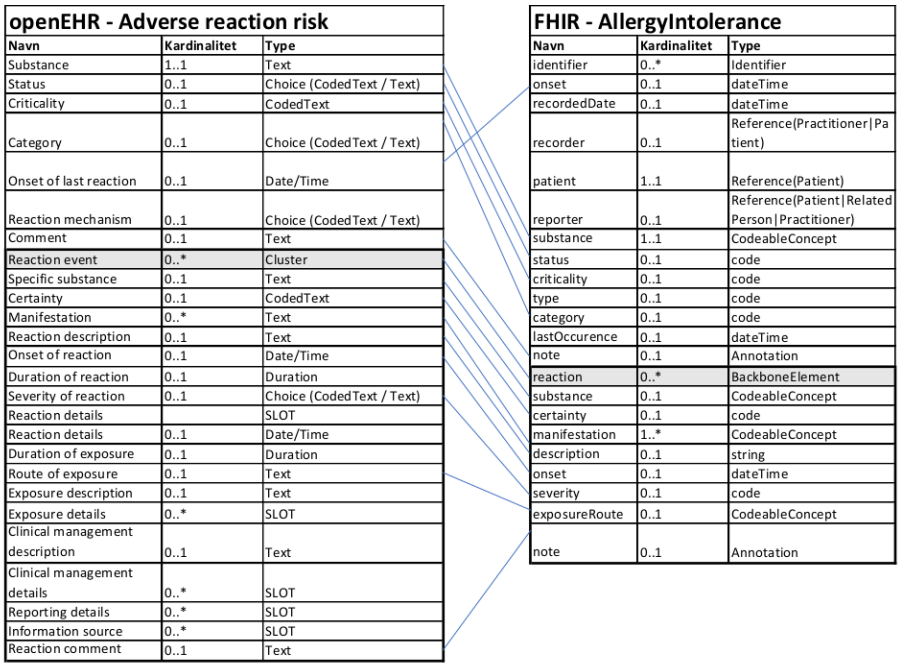
Figur B.1: Modell over arbeidsflyten når en ønsker å legge til flere felter i ressursen og arketypen.

FHIR Observation



Figur B.2: FHIR Observation fra DSTU2 representert i form av et UML diagram som er hentet fra FHIR spesifikasjonen. Her er alle feltene som en kan bruke tatt med. Noen felter er kun en referanse til en annen ressurs, blant annet *subject* og *performer*. Bak hver referanse kan en se hvilke ressurser en kan referere til. Felter kan også være i form av ulike datatyper og dette er spesifisert ved hjelp av «[x]». En observasjon har feltet *value[x]* som kan være representert ved hjelp av datatypene Quantity, CodeableConsept, string, Range, Ratio, SampledData, Attachment, time, dateTime eller Period. Denne funksjonaliteten kan bli begrenset i en profil der en ønsker at kun en spesifikk datatype blir brukt. En observasjon kan også ha informasjon om relaterte observasjoner gitt i objektet *Related*. Noen observasjoner inneholder flere verdier, som blodtrykk, og dette kan beskrives ved hjelp av en liste med komponenter (*Component*). Det er også mulig å beskrive hva som er en lav og høy verdi av observasjonen ved hjelp av *ReferenceRange*. Dette kan bli brukt for å tolke observasjonen.

Kartlegging av likheter mellom 5 evalueringsressurser og tilhørende arketype



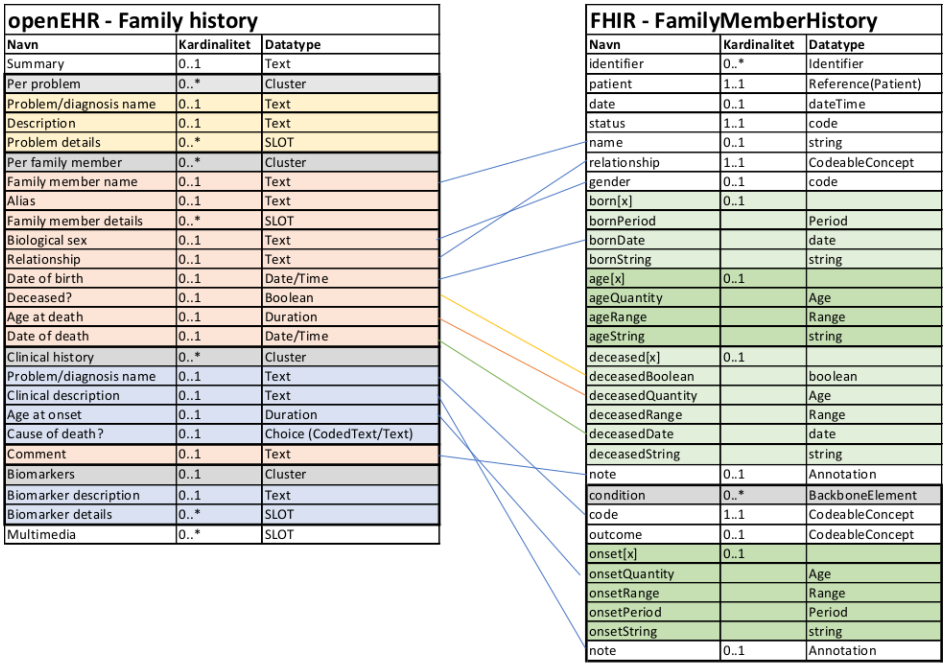
openEHR - Adverse reaction risk

Navn	Kardinalitet	Type
Substance	1..1	Text
Status	0..1	Choice (CodedText / Text)
Criticality	0..1	CodedText
Category	0..1	Choice (CodedText / Text)
Onset of last reaction	0..1	Date/Time
Reaction mechanism	0..1	Choice (CodedText / Text)
Comment	0..1	Text
Reaction event	0..*	Cluster
Specific substance	0..1	Text
Certainty	0..1	CodedText
Manifestation	0..*	Text
Reaction description	0..1	Text
Onset of reaction	0..1	Date/Time
Duration of reaction	0..1	Duration
Severity of reaction	0..1	Choice (CodedText / Text)
Reaction details		SLOT
Reaction details	0..1	Date/Time
Duration of exposure	0..1	Duration
Route of exposure	0..1	Text
Exposure description	0..1	Text
Exposure details	0..*	SLOT
Clinical management description	0..1	Text
Clinical management details	0..*	SLOT
Reporting details	0..*	SLOT
Information source	0..*	SLOT
Reaction comment	0..1	Text

FHIR - AllergyIntolerance

Navn	Kardinalitet	Type
identifier	0..*	Identifier
onset	0..1	dateTime
recordedDate	0..1	dateTime
recorder	0..1	Reference(Practitioner|Patient)
patient	1..1	Reference(Patient)
reporter	0..1	Reference(Patient|RelatedPerson|Practitioner)
substance	1..1	CodeableConcept
status	0..1	code
criticality	0..1	code
type	0..1	code
category	0..1	code
lastOccurence	0..1	dateTime
note	0..1	Annotation
reaction	0..*	BackboneElement
substance	0..1	CodeableConcept
certainty	0..1	code
manifestation	1..*	CodeableConcept
description	0..1	string
onset	0..1	dateTime
severity	0..1	code
exposureRoute	0..1	CodeableConcept
note	0..1	Annotation

Figur B.3: Her kan en se feltene til arketypen «Adverse reaction risk» og ressursen «AllergyIntolerance». For hver likhet er det en blå strek fra FHIR felt til openEHR felt. Arketypen har kun et obligatorisk felt; «Substance». FHIR ressursen har to obligatoriske felter; «substance» og «patient». Som en kan se i figuren er feltene «Substance» i arketypen og «substance» i ressursen bundet sammen. Når et felt har datatypen «Text» er det mulig å enten bruke datatypen «Text» eller «CodedText». I dette tilfellet er det ønskelig å bruke «CodedText» som kan bli konvertert til FHIR sin datatype «CodeableConcept». Feltet «patient» får sin informasjon fra selve journalen hvor arketypen er lagret. Ved oppretting av ressursen brukes dette feltet for å hente ut pasientjournalen med gitt pasientidentifikasjon. Det er også flere valgfri felter som kan konverteres.



openEHR - Family history

Navn	Kardinalitet	Datatype
Summary	0..1	Text
Per problem	0..*	Cluster
Problem/diagnosis name	0..1	Text
Description	0..1	Text
Problem details	0..*	SLOT
Per family member	0..*	Cluster
Family member name	0..1	Text
Alias	0..1	Text
Family member details	0..*	SLOT
Biological sex	0..1	Text
Relationship	0..1	Text
Date of birth	0..1	Date/Time
Deceased?	0..1	Boolean
Age at death	0..1	Duration
Date of death	0..1	Date/Time
Clinical history	0..*	Cluster
Problem/diagnosis name	0..1	Text
Clinical description	0..1	Text
Age at onset	0..1	Duration
Cause of death?	0..1	Choice (CodedText/Text)
Comment	0..1	Text
Biomarkers	0..1	Cluster
Biomarker description	0..1	Text
Biomarker details	0..*	SLOT
Multimedia	0..*	SLOT

FHIR - FamilyMemberHistory

Navn	Kardinalitet	Datatype
identifier	0..*	Identifier
patient	1..1	Reference(Patient)
date	0..1	dateTime
status	1..1	code
name	0..1	string
relationship	1..1	CodeableConcept
gender	0..1	code
born[x]	0..1	
bornPeriod		Period
bornDate		date
bornString		string
age[x]	0..1	
ageQuantity		Age
ageRange		Range
ageString		string
deceased[x]	0..1	
deceasedBoolean		boolean
deceasedQuantity		Age
deceasedRange		Range
deceasedDate		date
deceasedString		string
note	0..1	Annotation
condition	0..*	BackboneElement
code	1..1	CodeableConcept
outcome	0..1	CodeableConcept
onset[x]	0..1	
onsetQuantity		Age
onsetRange		Range
onsetPeriod		Period
onsetString		string
note	0..1	Annotation

Figur B.4: Her vises feltene til arketypen Family history og ressursen FamilyMemberHistory. Dette er et tilfelle der en ikke kan direkte konvertere en ressurs til en arketype. Arketypen brukes for å beskrive alle relevante familiemedlemmer, mens ressursen brukes for ett familiemedlem. Dette kan løses ved å konvertere arketypen til en liste med FamilyMemberHistory ressurser. I arketypen er det brukt farger for å skille klustre fra hverandre. Arketypen har to klustre; «Per problem» og «Biomarkers». Klusteret «Per problem» har igjen to klustre «Per family member» og «Clinical history». Det er klusteret «Per family member» som kan konverteres til ressursen FamilyMemberHistory, siden arketypen har en liste med «Per family member» klustre hvor hvert kluster beskrivet et familiemedlem. I ressursen er det også brukt ulike grønnfarger for å gruppere felter. Noen felter i FHIR har en valget mellom flere datatyper. For eksempel kan feltet «born[x]» bli beskrevet ved hjelp av datatypen «Period», «date» eller «string». En kan også se at noen linjer mellom feltene har fargene gul, rød og grønn. Disse tre linjene beskriver at under konverteringen må det gjøres et valg. Siden FHIR ressursen kan kun ha ett felt av «deceasedBoolean», «deceasedQuantity» og «deceasedDate» kan ikke alle tre tilsvarende felter i arketypen bli brukt. Dette gjelder feltene «Deceased?», «Age at death» og «Date of death». Arketypen har ingen obligatoriske felter. Ressursen derimot har tre obligatoriske felter; «patient», «status» og «relationship». Første felt med pasientidentifikasjonen får sin informasjon hentet fra journalen hvor arketypen er lagret. Feltet «relationship» kan konverteres til feltet «Relationship» i klusteret «Per family member». Feltet «status» beskriver hvilken status denne informasjonen har. Dette feltet kan enten ha verdien *partial*, *completed*, *entered-in-error* eller *health-unknown*. Utifra arketypen er det ikke mulig å avgjøre hvilken status denne informasjonen har.

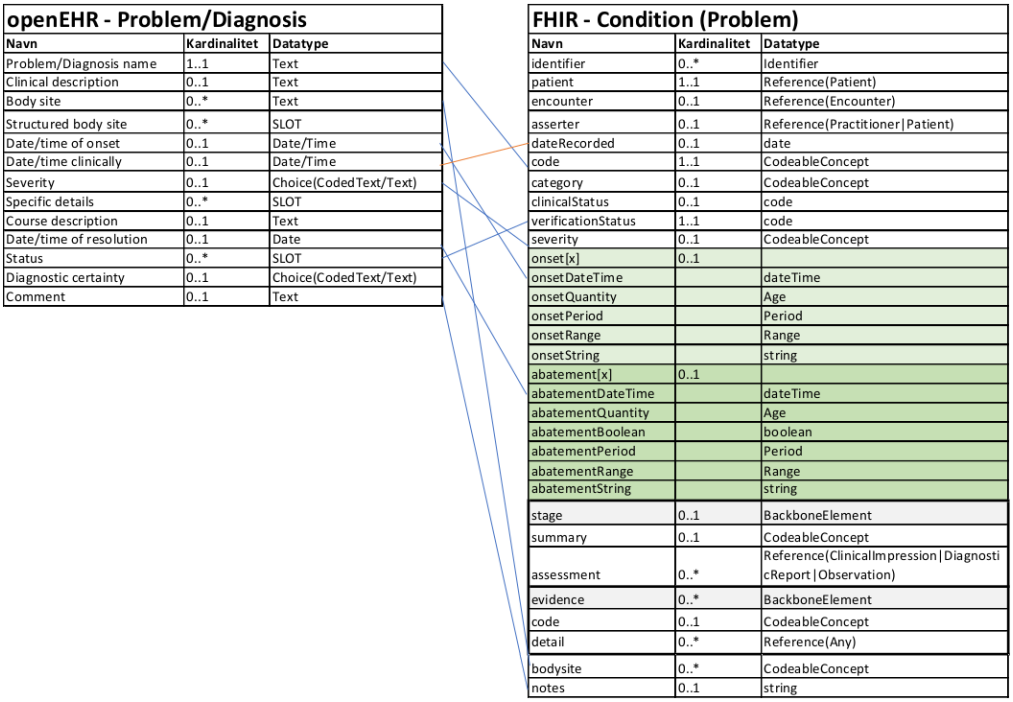
openEHR - Goal

Navn	Kardinalitet	Datatype
Goal name	1..1	Text
Goal description	0..1	Text
Clinical indication	0..*	Text
Goal start date	0..1	Date/Time
Goal proposed data	0..1	Date
Goal end date	0..1	Date
Goal outcome	0..1	Choice (CodedText/Text)
Goal comment	0..1	Text
Readiness for change	0..*	SLOT
Target	0..*	Cluster
Target name	0..1	Text
Target	0..1	Choice(Interval,Duration,Quantity,Count,Proportion,Tex
Target description	0..1	Text
Target path	1..1	URI
Target proposed date	0..1	Date
Target end date	0..1	Date
Target outcome	0..1	Choice(CodedText/Text)
Target comment	0..1	Text

FHIR - Goal

Navn	Kardinalitet	Datatype
identifier	0..*	Identifier
subject	0..1	Reference(Patient|Group|Organization)
start[x]	0..1	
startDate		date
startCodeableConcept		CodeableConcept
target[x]	0..1	
targetDate		date
targetQuantity		Duration
category	0..*	CodeableConcept
description	1..1	string
status	1..1	code
statusDate	0..1	date
statusReason	0..1	CodeableConcept
author	0..1	Reference(Patient|Practitioner|RelatedPerson)
priority	0..1	CodeableConcept
addresses	0..*	Reference(Condition|Observation|MedicatioStatement
note	0..*	Annotation
outcome	0..*	BackboneElement
result[x]	0..1	
resultCodeableConcept		CodeableConcept
resultReference		Reference(Observation)

Figur B.5: I figuren er arketypen Goal og ressursen Goal presentert. Arketypen har kun et obligatorisk felt; «Goal name». Ressursen har to obligatoriske felter; «description» og «status». Feltet «description» i ressursen kan konverteres til feltet «Goal description» i arketypen. Feltet «status» vil ha samme problemstilling som beskrevet under figur B.4, det er ikke mulig å avgjøre statusen. Feltet «Goal name» kan ikke direkte konverteres til et tilsvarende felt i ressursen, men her kan det være mulig å profilere ressursen med et nytt felt for å beskrive navnet. I tillegg til blå linjer som markerer mulige konverteringer finnes det en rød strek. Denne spesifiserer det er mulig å konvertere «statusDate» i ressursen over til «Goal end date» i arketypen når «status» feltet i ressursen er satt til *achieved*.



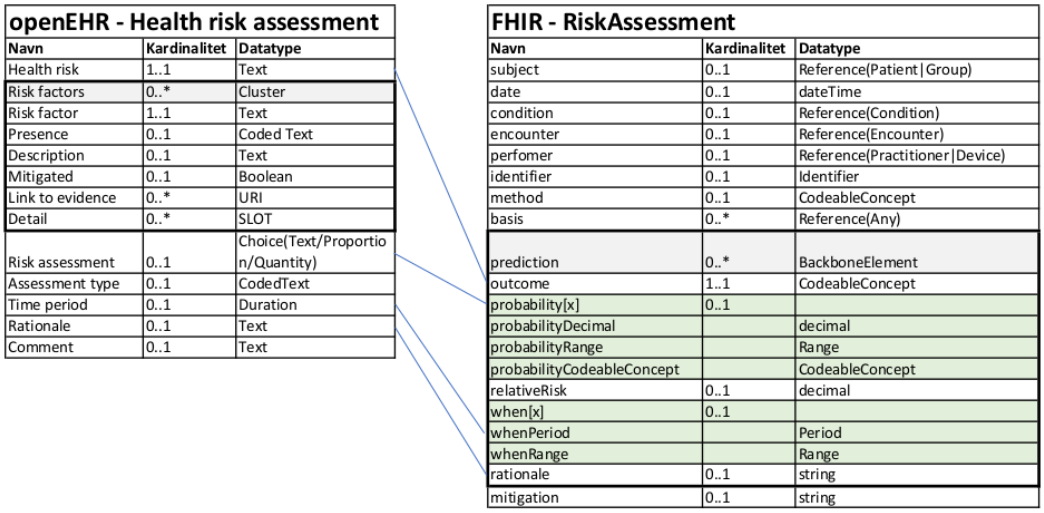
openEHR - Problem/Diagnosis

Navn	Kardinalitet	Datatype
Problem/Diagnosis name	1..1	Text
Clinical description	0..1	Text
Body site	0..*	Text
Structured body site	0..*	SLOT
Date/time of onset	0..1	Date/Time
Date/time clinically	0..1	Date/Time
Severity	0..1	Choice(CodedText/Text)
Specific details	0..*	SLOT
Course description	0..1	Text
Date/time of resolution	0..1	Date
Status	0..*	SLOT
Diagnostic certainty	0..1	Choice(CodedText/Text)
Comment	0..1	Text

FHIR - Condition (Problem)

Navn	Kardinalitet	Datatype
identifier	0..*	Identifier
patient	1..1	Reference(Patient)
encounter	0..1	Reference(Encounter)
asserter	0..1	Reference(Practitioner|Patient)
dateRecorded	0..1	date
code	1..1	CodeableConcept
category	0..1	CodeableConcept
clinicalStatus	0..1	code
verificationStatus	1..1	code
severity	0..1	CodeableConcept
onset[x]	0..1	
onsetDateTime		dateTime
onsetQuantity		Age
onsetPeriod		Period
onsetRange		Range
onsetString		string
abatement[x]	0..1	
abatementDateTime		dateTime
abatementQuantity		Age
abatementBoolean		boolean
abatementPeriod		Period
abatementRange		Range
abatementString		string
stage	0..1	BackboneElement
summary	0..1	CodeableConcept
assessment	0..*	Reference(ClinicalImpression|DiagnosticReport|Observation)
evidence	0..*	BackboneElement
code	0..1	CodeableConcept
detail	0..*	Reference(Any)
bodysite	0..*	CodeableConcept
notes	0..1	string

Figur B.6: Her er arketypen Problem/Diagnosis og ressursen Condition presentert. Arketypen har et obligatorisk felt, «Problem/Diagnosis name», som beskriver hvilken diagnose, sykdom eller problem pasienten har. Ressursen har tre obligatoriske felter; «patient», «code» og «verificationStatus». Navnefeltet i arketypen kan konverteres til kodefeltet i ressursen, som begge beskriver hvilke diagnose eller sykdom pasienten har. Verifikasjonsstatusen i ressursen kan tenkes å konverteres til et felt som eksisterer i en «status-arketype» som kan brukes i åpningen «Status» i «Problem/Diagnosis» arketypen. Informasjon rundt dette og informasjon om konvertering er beskrevet i avsnitt 5.4.3 og avsnitt 5.7.3. I tillegg til blå linjer som markerer mulige konverteringer finnes det en rød strek. Årsaken til dette er at det kan tenkes at datoen for når en helsepersonell evaluerte at pasienten hadde diagnosen er den samme for når diagnosen ble loggført. Dette er en konvertering som fremdeles er under vurdering, og er dermed satt til rød strek. Det kan også tenkes at datoen i ressursen må hentes fra når arketypen ble først lagret i openEHR systemet.



openEHR - Health risk assessment

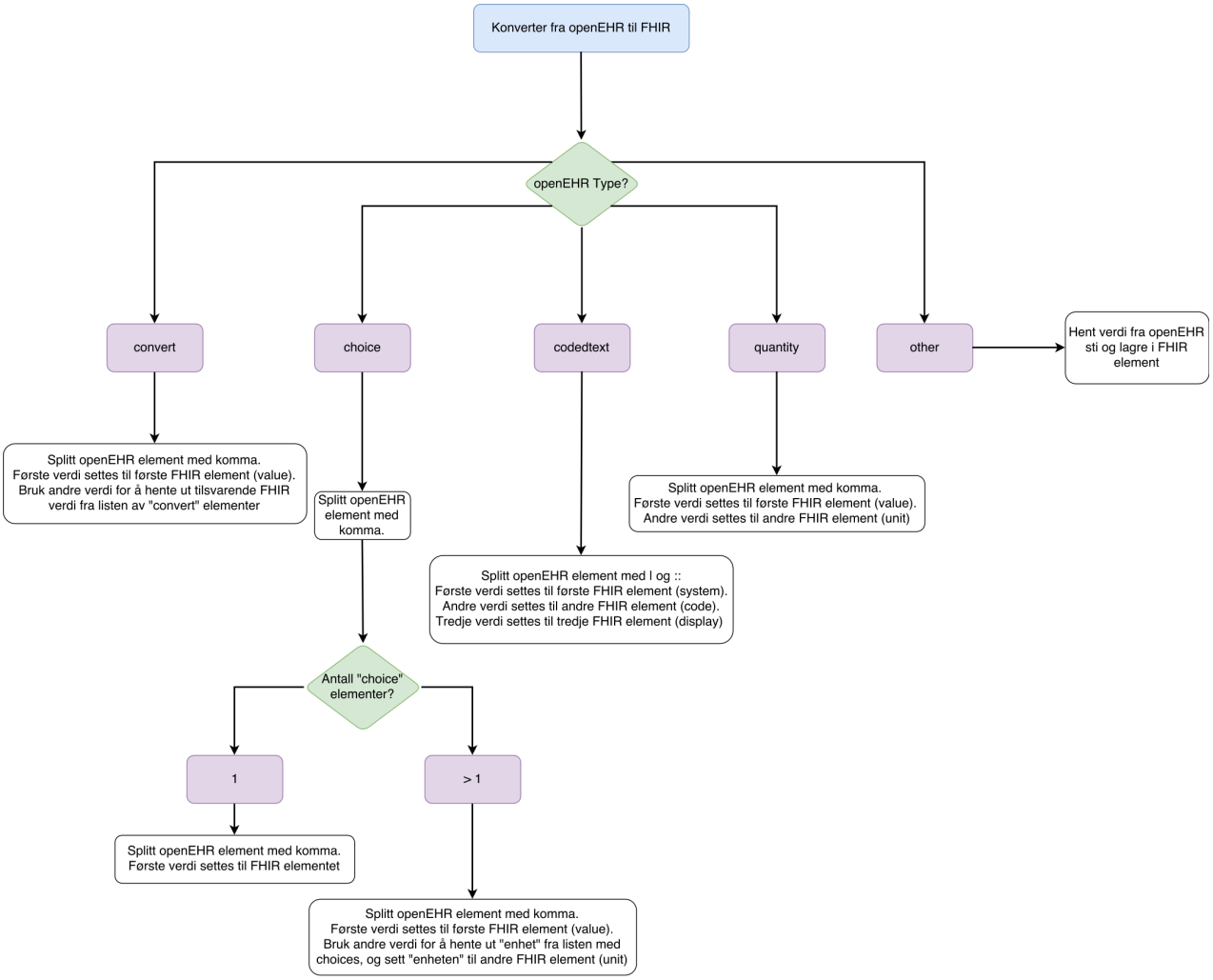
Navn	Kardinalitet	Datatype
Health risk	1..1	Text
Risk factors	0..*	Cluster
Risk factor	1..1	Text
Presence	0..1	Coded Text
Description	0..1	Text
Mitigated	0..1	Boolean
Link to evidence	0..*	URI
Detail	0..*	SLOT
Risk assessment	0..1	Choice(Text/Proportion/Quantity)
Assessment type	0..1	CodedText
Time period	0..1	Duration
Rationale	0..1	Text
Comment	0..1	Text

FHIR - RiskAssessment

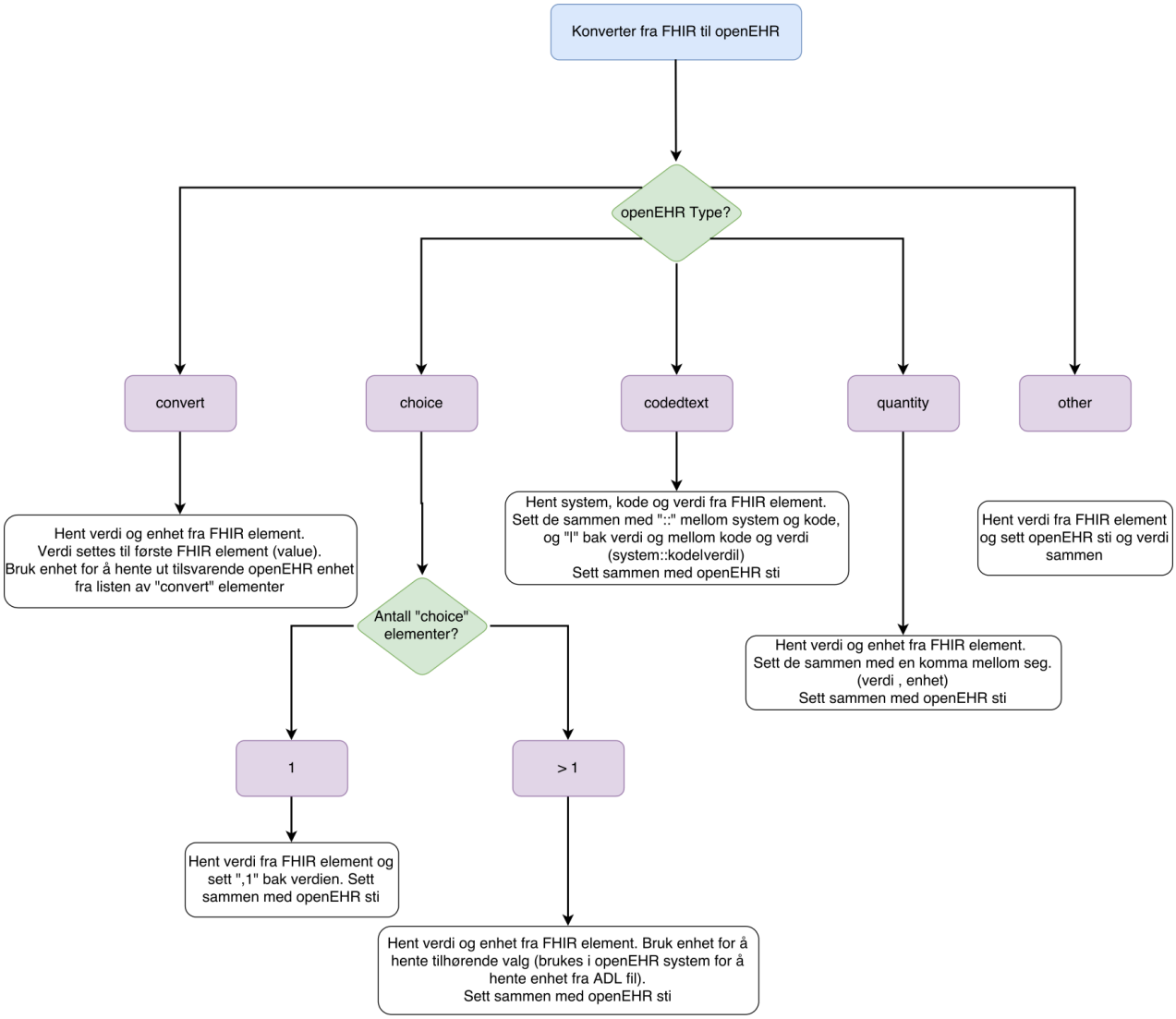
Navn	Kardinalitet	Datatype
subject	0..1	Reference(Patient|Group)
date	0..1	dateTime
condition	0..1	Reference(Condition)
encounter	0..1	Reference(Encounter)
perfomer	0..1	Reference(Practitioner|Device)
identifier	0..1	Identifier
method	0..1	CodeableConcept
basis	0..*	Reference(Any)
prediction	0..*	BackboneElement
outcome	1..1	CodeableConcept
probability[x]	0..1	
probabilityDecimal		decimal
probabilityRange		Range
probabilityCodeableConcept		CodeableConcept
relativeRisk	0..1	decimal
when[x]	0..1	
whenPeriod		Period
whenRange		Range
rationale	0..1	string
mitigation	0..1	string

Figur B.7: Her er arketypen Health risk assessment og ressursen RiskAssessment presentert. Arketypen har kun et obligatorisk felt, «Health risk», som er en identifikasjon om en mulig fremtidig sykdom, tilstand eller helseproblem. Ressursen har ingen obligatoriske felter, men feltet «outcome» i ressursen i elementet «prediction» beskriver samme identifikasjon om en mulig fremtidig sykdom, tilstand eller helseproblem. Arketypen og ressursen er strukturert på to ulike måter, som påvirker hvordan de kan konverteres til hverandre. De viktigste feltene som kan tenkes å være mulig å konvertere er beskrevet med blå strek.

Beslutningstre for konvertering



Figur B.8: Beslutningstreet viset et enkelt bilde over hvordan konverteringen foregår ved hjelp av en konverteringsfil fra en openEHR komposisjon til en FHIR ressurs. De hvite rutene er prosesseringen som blir utført utifra hvilken datatype openEHR stien peker på.



Figur B.9: Beslutningstreet viser et enkelt bilde over hvordan konverteringen foregår ved hjelp av en konverteringsfil fra en FHIR ressurs til en openEHR komposisjon. Dersom det kun er et «choice» element vet konverteringsmodulen at valg nummer 1 skal bli tatt.

Konverteringsfil for legemiddelordinering

Listing B.1: Konverteringsfil for arketypen Legemiddelordinering

```
{
  "mappings": [
    {
      "fhirElements": [
        "medication.code.coding[0].system.myStringValue",
        "medication.code.coding[0].code.myStringValue",
        "medication.code.coding[0].display.myStringValue"
      ],
      "openehrPaths": [
        "/content[openEHR-EHR-INSTRUCTION.medication_order.v0]/activities[at0001]/
            description[at0002]/items[at0070]|value"
      ],
      "openehrType": "codedtext"
    },
    {
      "fhirElements": [
        "medicationRequest.dosageInstruction[0].timing.repeat.frequency.
            myStringValue",
        "medicationRequest.dosageInstruction[0].timing.repeat.periodUnit.
            myStringValue"
      ],
      "openehrPaths": [
        "/content[openEHR-EHR-INSTRUCTION.medication_order.v0]/activities[at0001]/
            description[at0002]/items[at0056]/items[at0058]/items[openEHR-EHR-
            CLUSTER.timing_daily.v0]/items[at0003]|value"
      ],
      "openehrType": "convert",
      "convert": {
        "d": {
          "value": "1/d"
        },
        "min": {
          "value": "1/min"
        },
        "s": {
          "value": "1/s"
        },
        "h": {
          "value": "1/h"
        }
      }
    },
    {
      "fhirElements": [
        "medicationRequest.dosageInstruction[0].dose.value.myStringValue"
      ],
      "openehrPaths": [
        "/content[openEHR-EHR-INSTRUCTION.medication_order.v0]/activities[at0001]/
            description[at0002]/items[at0056]/items[at0058]/items[at0144]|value"
      ],
      "openehrType": "choice",
      "choice": {
        "empty": {}
      }
    },
    {
      "fhirElements": [
        "medicationRequest.dosageInstruction[0].dose.unit.myStringValue"
      ],
      "openehrPaths": [
        "/content[openEHR-EHR-INSTRUCTION.medication_order.v0]/activities[at0001]/
            description[at0002]/items[at0056]/items[at0058]/items[at0145]|value"
```

```
56          ],
57          "openehrType": "other"
58        },
59        {
60          "fhirElements": [
61            "medication.form.coding[0].system.myStringValue",
62            "medication.form.coding[0].code.myStringValue",
63            "medication.form.coding[0].display.myStringValue"
64          ],
65          "openehrPaths": [
66            "/content[openEHR-EHR-INSTRUCTION.medication_order.v0]/activities[at0001]/
                  description[at0002]/items[openEHR-EHR-CLUSTER.medication_substance.v0]
                  /items[at0071]|value"
67          ],
68          "openehrType": "codedtext"
69        },
70        {
71          "fhirElements": [
72            "medicationRequest.dosageInstruction[0].asNeeded.myStringValue"
73          ],
74          "openehrPaths": [
75            "/content[openEHR-EHR-INSTRUCTION.medication_order.v0]/activities[at0001]/
                  description[at0002]/items[at0056]/items[at0058]/items[openEHR-EHR-
                  CLUSTER.timing_daily.v0]/items[at0024]|value"
76          ],
77          "openehrType": "other"
78        },
79        {
80          "fhirElements": [
81            "medicationRequest.dosageInstruction[0].route.text.myStringValue"
82          ],
83          "openehrPaths": [
84            "/content[openEHR-EHR-INSTRUCTION.medication_order.v0]/activities[at0001]/
                  description[at0002]/items[at0091]|value"
85          ],
86          "openehrType": "other"
87        },
88        {
89          "fhirElements": [
90            "medicationRequest.dosageInstruction[0].text.myStringValue"
91          ],
92          "openehrPaths": [
93            "/content[openEHR-EHR-INSTRUCTION.medication_order.v0]/activities[at0001]/
                  description[at0002]/items[at0056]/items[at0058]/items[openEHR-EHR-
                  CLUSTER.timing_daily.v0]/items[at0026]|value"
94          ],
95          "openehrType": "other"
96        }
97      ]
98    }
```

Vedlegg C

Vurderingsbidrag

Listing C.1: GET respons med unik identifikator for FHIR ressursen Medication

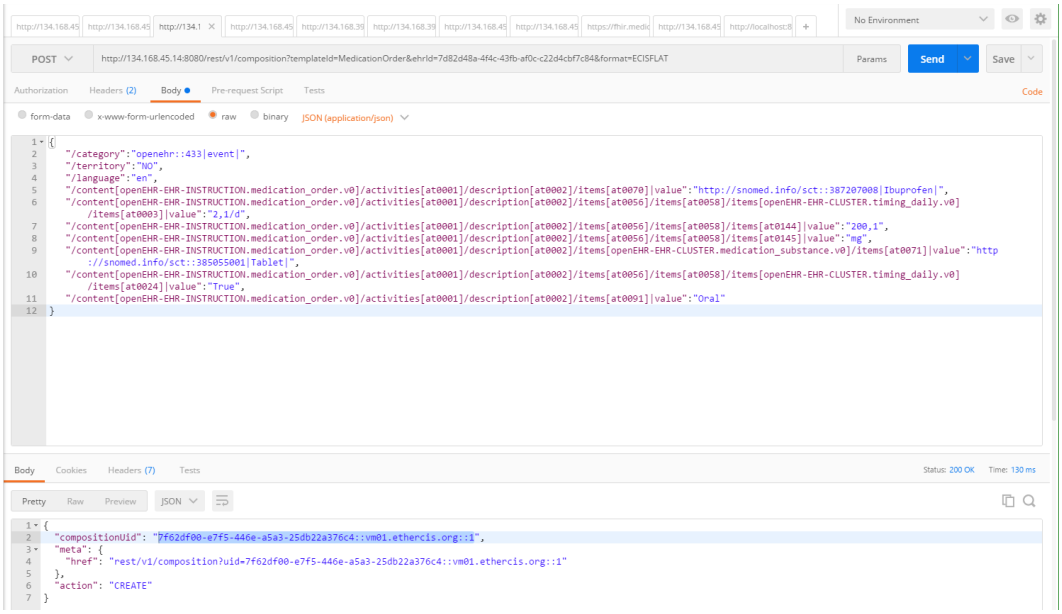
```
{
  "resourceType": "Bundle",
  "id": "45a890ac-8beb-4317-aa4b-d5f7bdb0cdf5",
  ...
  "link": [
    {
      "relation": "self",
      "url": "http://fhir.medicloud.eu-gb.mybluemix.net/fhir/Medication?_format=
          json"
    }
  ],
  "entry": [
    {
      "fullUrl": "http://fhir.medicloud.eu-gb.mybluemix.net/fhir/Medication/2
          df0b91a-91a0-47e0-9f29-57887140a49f",
      "resource": {
        "resourceType": "Medication",
        "id": "2df0b91a-91a0-47e0-9f29-57887140a49f",
        ...
      }
    },
    ...
  ]
}
```

Listing C.2: Utsnitt fra kildekoden til klassen MainServlet fra FHIR bussen

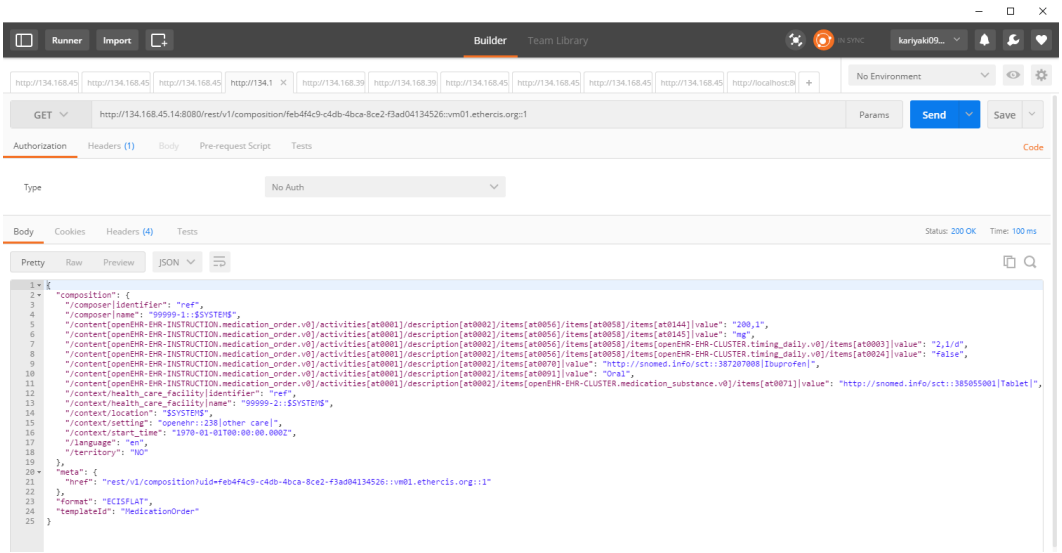
```
public class MainServlet extends RestfulServer {

    public MainServlet() {super(FhirContext.forDstu3());}

    @Override
    protected void initialize() throws ServletException {

        List<IResourceProvider> providers = new ArrayList<IResourceProvider>();
        providers.add(new ObservationResourceProvider());
        providers.add(new PatientResourceProvider());
        providers.add(new AllergyIntoleranceResourceProvider());
        providers.add(new ConditionResourceProvider());
        providers.add(new MedicationRequestResourceProvider());
        providers.add(new MedicationStatementResourceProvider());
        providers.add(new PractitionerResourceProvider());
        providers.add(new MedicationResourceProvider());
        providers.add(new OrganizationResourceProvider());

        setResourceProviders(providers);
        ...
    }
}
```

Listing C.3: Eksempel for hvordan en ny «provider» for en ressurs opprettes i FHIR bussen.

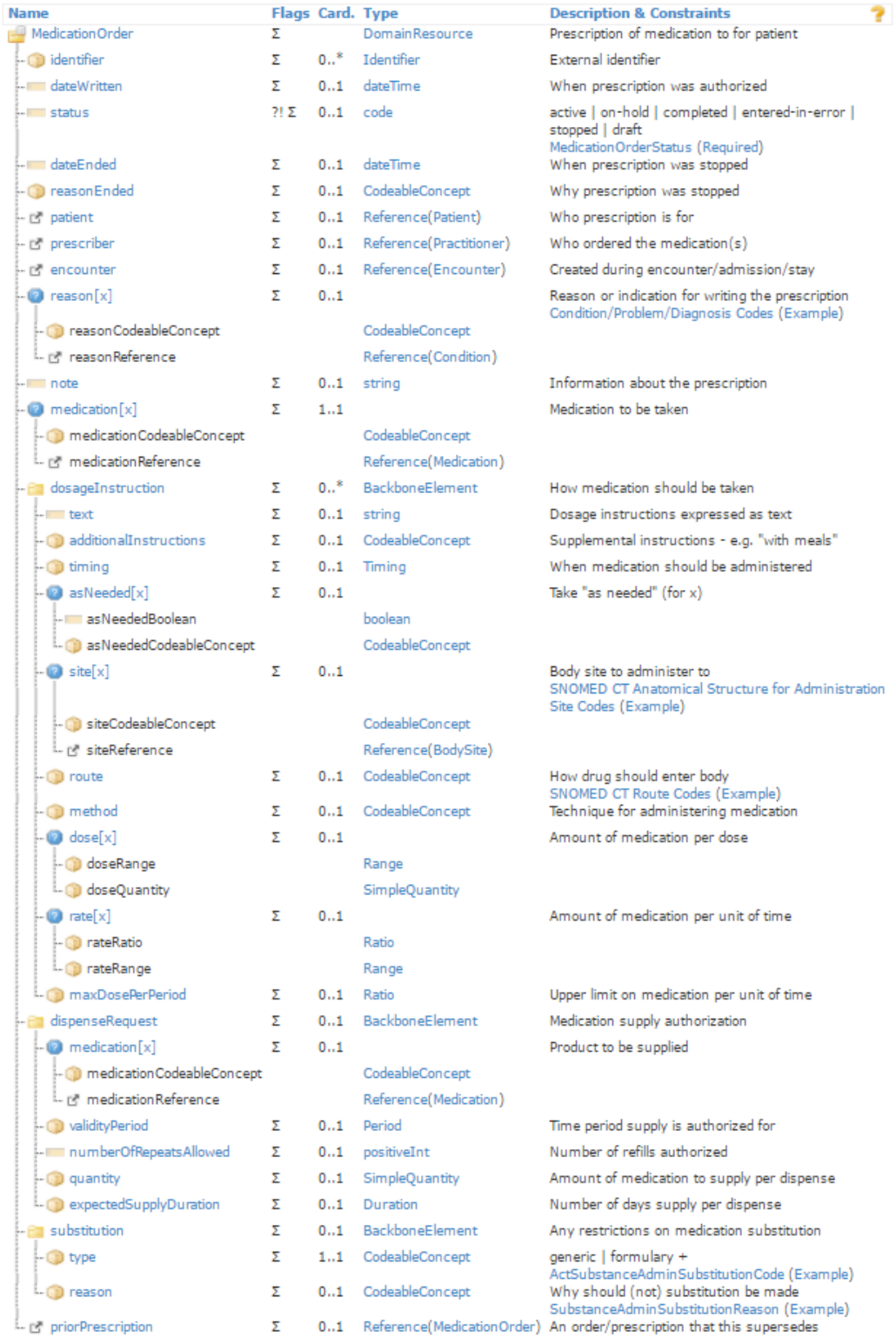
```
import ca.uhn.fhir.rest.server.IResourceProvider;
import org.hl7.fhir.dstu3.model.FamilyMemberHistory;
import org.hl7.fhir.instance.model.api.IBaseResource;

public class FamilyMemberHistoryResourceProvider implements IResourceProvider {

    @Override
    public Class<? extends IBaseResource> getResourceType() {
        return FamilyMemberHistory.class;
    }
}
```



Figur C.1: Utsnitt fra Postman for å opprette en ny komposisjon i openEHR pasientjournalsystemet



Figur C.2: Utsnitt fra Postman for å hente en eksisterende komposisjon fra openEHR pasientjournalsysteme. Som en kan se innholder denne komposisjonen flere felter enn det vi hadde i vår komposisjon da vi opprettet den. Årsaken til dette er at flere av feltene er valgfrie å fylle ut. Dersom brukeren ikke fyller dem ut setter EtherCIS disse feltene til egne standardverdier.



Name	Flags	Card.	Type	Description & Constraints
MedicationOrder	Σ		DomainResource	Prescription of medication to for patient
identifier	Σ	0..*	Identifier	External identifier
dateWritten	Σ	0..1	dateTime	When prescription was authorized
status	?! Σ	0..1	code	active | on-hold | completed | entered-in-error | stopped | draft MedicationOrderStatus (Required)
dateEnded	Σ	0..1	dateTime	When prescription was stopped
reasonEnded	Σ	0..1	CodeableConcept	Why prescription was stopped
patient	Σ	0..1	Reference(Patient)	Who prescription is for
prescriber	Σ	0..1	Reference(Practitioner)	Who ordered the medication(s)
encounter	Σ	0..1	Reference(Encounter)	Created during encounter/admission/stay
reason[x]	Σ	0..1		Reason or indication for writing the prescription Condition/Problem/Diagnosis Codes (Example)
reasonCodeableConcept			CodeableConcept	
reasonReference			Reference(Condition)	
note	Σ	0..1	string	Information about the prescription
medication[x]	Σ	1..1		Medication to be taken
medicationCodeableConcept			CodeableConcept	
medicationReference			Reference(Medication)	
dosageInstruction	Σ	0..*	BackboneElement	How medication should be taken
text	Σ	0..1	string	Dosage instructions expressed as text
additionalInstructions	Σ	0..1	CodeableConcept	Supplemental instructions - e.g. "with meals"
timing	Σ	0..1	Timing	When medication should be administered
asNeeded[x]	Σ	0..1		Take "as needed" (for x)
asNeededBoolean			boolean	
asNeededCodeableConcept			CodeableConcept	
site[x]	Σ	0..1		Body site to administer to SNOMED CT Anatomical Structure for Administration Site Codes (Example)
siteCodeableConcept			CodeableConcept	
siteReference			Reference(BodySite)	
route	Σ	0..1	CodeableConcept	How drug should enter body SNOMED CT Route Codes (Example)
method	Σ	0..1	CodeableConcept	Technique for administering medication
dose[x]	Σ	0..1		Amount of medication per dose
doseRange			Range	
doseQuantity			SimpleQuantity	
rate[x]	Σ	0..1		Amount of medication per unit of time
rateRatio			Ratio	
rateRange			Range	
maxDosePerPeriod	Σ	0..1	Ratio	Upper limit on medication per unit of time
dispenseRequest	Σ	0..1	BackboneElement	Medication supply authorization
medication[x]	Σ	0..1		Product to be supplied
medicationCodeableConcept			CodeableConcept	
medicationReference			Reference(Medication)	
validityPeriod	Σ	0..1	Period	Time period supply is authorized for
numberOfRepeatsAllowed	Σ	0..1	positiveInt	Number of refills authorized
quantity	Σ	0..1	SimpleQuantity	Amount of medication to supply per dispense
expectedSupplyDuration	Σ	0..1	Duration	Number of days supply per dispense
substitution	Σ	0..1	BackboneElement	Any restrictions on medication substitution
type	Σ	1..1	CodeableConcept	generic | formulary + ActSubstanceAdminSubstitutionCode (Example)
reason	Σ	0..1	CodeableConcept	Why should (not) substitution be made SubstanceAdminSubstitutionReason (Example)
priorPrescription	Σ	0..1	Reference(MedicationOrder)	An order/prescription that this supersedes

Figur C.3: Figuren viser et utsnitt fra FHIR ressursen MedicationOrder i DSTU2 versjonen av FHIR spesifikasjonen.

Name	Flags	Card.	Type	Description & Constraints
MedicationRequest			DomainResource	Ordering of medication for patient or group Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension
identifier		0..*	Identifier	External ids for this request
definition	Σ	0..*	Reference(ActivityDefinition | PlanDefinition)	Protocol or definition
basedOn	Σ	0..*	Reference(CarePlan | MedicationRequest | ProcedureRequest | ReferralRequest)	What request fulfills
groupIdentifier	Σ	0..1	Identifier	Composite request this is part of
status	?! Σ	0..1	code	active | on-hold | cancelled | completed | entered-in-error | stopped | draft | unknown MedicationRequestStatus (Required)
intent	?! Σ	1..1	code	proposal | plan | order | instance-order MedicationRequestIntent (Required)
category		0..1	CodeableConcept	Type of medication usage MedicationRequestCategory (Preferred)
priority	Σ	0..1	code	routine | urgent | stat | asap MedicationRequestPriority (Required)
medication[x]	Σ	1..1		Medication to be taken SNOMED CT Medication Codes (Example)
medicationCodeableConcept			CodeableConcept	
medicationReference			Reference(Medication)	
subject	Σ	1..1	Reference(Patient | Group)	Who or group medication request is for
context		0..1	Reference(Encounter | EpisodeOfCare)	Created during encounter/admission/stay
supportingInformation		0..*	Reference(Any)	Information to support ordering of the medication
authoredOn	Σ	0..1	dateTime	When request was initially authored
requester	Σ I	0..1	BackboneElement	Who/What requested the Request *+ onBehalfOf can only be specified if agent is practitioner or device*
agent	Σ	1..1	Reference(Practitioner | Organization | Patient | RelatedPerson | Device)	Who ordered the initial medication(s)
onBehalfOf	Σ I	0..1	Reference(Organization)	Organization agent is acting for
recorder		0..1	Reference(Practitioner)	Person who entered the request
reasonCode		0..*	CodeableConcept	Reason or indication for writing the prescription Condition/Problem/Diagnosis Codes (Example)
reasonReference		0..*	Reference(Condition | Observation)	Condition or Observation that supports why the prescription is being written
note		0..*	Annotation	Information about the prescription
dosageInstruction		0..*	Dosage	How the medication should be taken
dispenseRequest		0..1	BackboneElement	Medication supply authorization
validityPeriod		0..1	Period	Time period supply is authorized for
numberOfRepeatsAllowed		0..1	positiveInt	Number of refills authorized
quantity		0..1	SimpleQuantity	Amount of medication to supply per dispense
expectedSupplyDuration		0..1	Duration	Number of days supply per dispense
performer		0..1	Reference(Organization)	Intended dispenser
substitution		0..1	BackboneElement	Any restrictions on medication substitution
allowed	?!	1..1	boolean	Whether substitution is allowed or not
reason		0..1	CodeableConcept	Why should (not) substitution be made SubstanceAdminSubstitutionReason (Example)
priorPrescription		0..1	Reference(MedicationRequest)	An order/prescription that is being replaced
detectedIssue		0..*	Reference(DetectedIssue)	Clinical Issue with action
eventHistory		0..*	Reference(Provenance)	A list of events of interest in the lifecycle

Figur C.4: Figuren viser et utsnitt fra FHIR ressursen MedicationRequest i STU3 versjonen av FHIR spesifikasjonen. Ressursen er en videreutviklet utgave fra DSTU2 versjonen av MedicationOrder ressursen. MedicationRequest har blant annet et nytt obligatorisk felt; «intent» som beskriver hvilken type ordinering dette er. Feltet «patient» fra MedicationOrder har fått nytt navn, «subject» og er nå et obligatorisk felt. Ellers finnes flere endringer i navn og struktur.

Vedlegg D

FHIR integrasjonsbuss dokumentasjon

Alle FHIR ressursene som FHIR bussen støtter per i dag er listet opp. For hver ressurs er tilhørende profil oppført. Det er også dokumentert hvor FHIR bussen krever en «Ehr-Session» nøkkel i «Header» seksjonen i forespørselen. Noen ressurser har mulighet for å utføre et søk (SEARCH). Et søk er en GET forespørsel hvor en enten får en liste med alle eksisterende elementer av gitt ressurs eller en mindre liste ved hjelp av søkeparametre. Hvor SEARCH er listet opp og søkeparametre mangler har en muligheten om å hente listen med alle ressursene av gitt type. Dersom både SEARCH og søkeparametre er listet opp vil det stå bak SEARCH dersom begge funksjonalitetene er implementert. Dette gjelder for eksempel ressursen Patient. Dersom dette ikke står er det kun implementert å bruke søkeparameteren.

AllergyIntolerance

Profil: `https://simplifier.net/HVL/21cbf6e3-586c-4dfe-8c00-0a357cdc09bb`
FHIR operasjoner:

- CREATE (Krever «Ehr-Session»)
- READ (Krever «Ehr-Session»)
- DELETE (Krever «Ehr-Session»)

Condition

Profil: `https://simplifier.net/HVL/8615cd6e-9d42-40d0-b0eb-5835d2e86c45`
FHIR operasjoner:

- CREATE (Krever «Ehr-Session»)
- READ (Krever «Ehr-Session»)
- DELETE (Krever «Ehr-Session»)

Encounter

Profil: `https://simplifier.net/HVL/370d0dd8-736a-4f62-a6fc-24f71f042644`
FHIR operasjoner:

- CREATE
- READ
- SEARCH (Krever «Ehr-Session»)

Search parameters

- Encounter.subject

Location

Profil: `https://simplifier.net/HVL/f9d34c78-4050-492d-8a6f-a13a5dbbd77a`
FHIR operasjoner:

- CREATE
- READ

Medication

Profil: `https://simplifier.net/HVL/7dddfe79-4c34-4781-983d-2cf49a4e693d`
FHIR operasjoner:

- CREATE
- READ
- SEARCH

MedicationRequest

Profil: `https://simplifier.net/HVL/61d9bf96-ef7c-427f-98ae-779d62b35c43`
FHIR operasjoner:

- CREATE (Krever «Ehr-Session»)
- READ (Krever «Ehr-Session»)
- SEARCH (Krever «Ehr-Session»)
- DELETE (Krever «Ehr-Session»)

Søkeparametre:

- MedicationRequest.patient

MedicationStatement

Profil: `https://simplifier.net/HVL/c71d5f7f-48ec-4a34-9ffa-d4c9563b256f`
FHIR operasjoner:

- SEARCH (Krever «Ehr-Session»)

Søkeparametre:

- MedicationStatement.patient

Organization

Profil: `https://simplifier.net/HVL/52ef8785-e22e-4004-bec9-2ddc15da868e`
FHIR operasjoner:

- CREATE
- READ
- SEARCH

Patient

Profil: `https://simplifier.net/HVL/0767a47f-dac8-4a34-a640-95d7c2fb99c8`
FHIR operasjoner:

- CREATE (Krever «Ehr-Session»)
- READ
- UPDATE (Krever «Ehr-Session»)
- SEARCH (Både uten og med søkeparametre)

Søkeparametre:

- Patient.organization

Practitioner

Profil: https://simplifier.net/HVL/643faf3e-5ff5-4aa4-bbe7-fee519014fe9
FHIR operasjoner:

1. CREATE
2. READ
3. UPDATE
4. SEARCH

Observation

Profil for blodtrykk: https://simplifier.net/HVL/ceb86c08-f29f-45cf-adda-b4ed4819a401
Profil for kroppsvekt: https://simplifier.net/HVL/88744c02-c09c-48e5-b0c8-94d89e53f2ec

FHIR operasjoner:

- CREATE (Krever «Ehr-Session»)
- READ (Krever «Ehr-Session»)
- DELETE (Krever «Ehr-Session»)
- SEARCH

Søkeparametre:

- Observation.patient