NTNU
Det skapende universitet

Integrert presentasjon av sanntids datastrømmer

Ved hjelp av Health Layer 7 Fast Healthcare Interoperability Resource

Vegar Eggereide

Helseinformatikk
Innlevert: desember 2017
Hovedveileder: Pieter Jelle Toussaint, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknologi og informatikk

Vegar Eggereide

Integrert presentasjon av sanntids datastrømmer

Ved hjelp av Health Layer 7 Fast Healthcare Interoperability Resource

Masteroppgave i Helseinformatikk

Trondheim, desember 2017

Veileder: Pieter Jelle Toussaint

Norges Teknisk Naturvitenskaplige Universitet
Fakultet for Informasjonsteknologi og Elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Sammendrag

Når man høster kliniske data er det viktig at disse standardiseres slik at man oppnår interoperabilitet. Det bør man gjøre for å unngå fragmentering av informasjon og tap av kontekst. For å vise at det er hensiktsmessig ble det laget et artefakt for å demonstrere proof-of-concept.

Artefaktet samlet datastrømmen fra to virtuelle sprøytepumper lastet med medisiner for å regulere blodtrykket. Blodtrykket observeres av personalet og medisineringen styres deretter.

Health Layer 7 Fast Healthcare Interoperability Resource ble brukt for å standardisere informasjonen slik at det er mulig å utveksle den.

Blodtrykkskurven og medisinkurvene ble fremstilt i en integrert visualisering for å vise at man kan høste, standardisere og om ønskelig utveksle data slik at man unngår fragmentering og bevarer konteksten.

Abstract

When harvesting clinical data it is important that these are standardized to achieve interoperability. This should be done to avoid fragmentation of information and loss of context. To demonstrate that it is expedient, an artifact was made to demonstrate proof-of-concept.

The artifact collected the data stream from two virtual syringe pumps loaded with medications to regulate blood pressure. The medication administered is determined by the blood pressure observed by the staff.

Healt Layer 7 Fast healthcare Interoperability Resource was used to standardize the information so that it is possible to exchange it.

The blood pressure and medication curve were made in an integrated visualization to show that you can reap, standardize and if desired exchange data to avoid fragmentation and preserve the context.

Forord

Helseinformatikkstudiet er for både IT- og helsepersonell. Det første året er det et todelt løp, der en følger tilrettelagt undervisning avhengig av hvilken bakgrunn man har. Med min kombinerte helse- og tekniske bakgrunn kunne i teorien valgt begge løpene. Siden jeg de siste 12 årene har jobbet utelukkende som helsepersonell, så jeg det som mest hensiktsmessig å følge løpet for personer med helsebakgrunn.

Siden en ikke blir bedre enn de man har rundt seg, hadde jeg et sterkt ønske om å finne en teknisk oppgave med problemstilling hentet fra klinikken.

Grunnen til at jeg velger å gjennomføre en ren IT-basert masteroppgave er, i tillegg til min tekniske interesse, at dette er kunnskap jeg ikke har mulighet til å tilegne meg gjennom mitt daglige virke. Derfor har jeg valgt en ren teknisk problemstilling hentet fra klinikken hvor jeg jobber i til daglig og kjenner godt.

Takk til Professor Emeritus Kjell Bratbergsengen som brukte av fritiden sin til innledende veiledning som dannet grunnlaget for den aktuelle problemstillingen.

Takk til klinikken som ga meg denne unike muligheten.

En stor takk til Professor Pieter Jelle Toussaint for veldig god veiledning.

Til slutt vil jeg gjerne takke den tålmodige familien min, spesielt samboer Gunn Berit som har tatt seg av barna og gitt meg arbeidsro.

Innholdsfortegnelse

Figurliste xi
Forkortelser xii

1. INNLEDNING 1

1.1. Egne erfaringer fra klinikken 1
1.2. Noen av dagens utfordringer 4
1.2.1. Fragmentering av kliniske data 4
1.2.2. Kliniske data som mangler kontekst 5
1.3. Problemstilling 6

2. TEORI 9

2.1. Standarder og interoperabilitet 9
2.1.1. Hva er interoperabilitet? 9
2.1.1.1. De ulike typene interoperabilitet 9
2.1.2. Hvorfor trenger vi interoperabilitet? 10
2.1.3. Hva er en standard? 11
2.1.3.1. Hvordan forvaltes standardene? 11
2.1.4. Health Layer 7 11
2.1.4.1. HL7 v2 12
2.1.4.2. HL7 v3 12
2.1.4.3. Fast Healthcare Interoperability Resource 12
2.1.4.4. Hva er ressurser? 13
2.1.4.5. Representational State Transfer 15
2.1.4.6. Clinicians on FHIR 16
2.2. Medisinsk utstyr 16
2.2.1. Datakommunikasjon 16
2.3. Hva har andre gjort 18
2.3.1. Utveksling mellom apparater og HIS 18
2.3.2. Utveksling mellom IKT systemer 18
2.3.3. Hva skiller denne oppgaven fra andre 19

3. METODE 21

3.1. Design Science 21
3.1.1. Et rammeverk for Design Science forskning 21
3.2. Konstruksjon av prototypen 23
3.2.1. Sprøytepumpe 23
3.2.2. Pasientmonitor 24
3.2.3. Datastrømmen 25
3.2.4. Kravspesifikasjon 25

4. RESULTAT 27

4.1. Scenarioet 27
4.1.1. Modellering 27
4.1.2. Oppbygningen av datastrømmen 29
4.2. Utvikling av prototypen 30
4.2.1. Hvordan er problemstillingen adressert 30
4.2.1.1. Virtuelle pumper 31
4.2.1.2. Skjematisk fremstilling 32
4.2.1.3. Gjennomgang av hver enkelt funksjon 33

4.3. Simuleringen 39
4.3.1. Resultatet av plottingen 39
4.3.2. Datastrømmen som lagres i JSON 40
4.3.2.1. Objektet medicationAdministration.status 40
4.3.2.2. Objektet medicationAdministration.medication 40
4.3.2.3. Objektet medicationAdministration.effectivePeriod 40
4.3.2.4. Objektet medicationAdministration.identifier 41
4.3.2.5. Objektet medicationAdministration.dosage.rateQuantity 41
4.3.2.6. Objektet medicationAdministration.device 41
4.3.2.7. Hvordan blir medisineringen dokumentert 41
4.3.2.8. Journalnotatene 41
5. DISKUSJON 45
5.1. Vurdering 45
5.1.1. Vurdering av datastrømmen og medicationAdministration 45
5.1.1.1. medicationAdministration.status 46
5.1.1.2. medicationAdministration.medication 46
5.1.1.3. medicationAdministration.effectivePeriod 47
5.1.1.4. medicationAdministration.dosage.rateQuantity 48
5.1.1.5. medicationAdministration.device 49
5.1.1.6. medicationAdministration.identifier 49
5.1.1.7. HL7 FHIR som grunnlag for sanntidsdata 49
5.1.2. Vurdering av artefaktet 49
5.1.2.1. Virtuelle sprøytepumper 49
5.1.2.2. Oversette datastrømmen 49
5.1.2.3. Skriver til dictionary og lagrer JSON-fil 49
5.1.2.4. Lage blodtrykkskurve 50
5.1.2.5. Plotting av kurvene 50
5.1.3. Tilbakemelding fra klinikken 50
5.1.4. Håndtere alle typer data 50
5.2. Naturlige begrensninger i denne oppgaven 51
5.2.1. Tiden 51
5.2.2. Størrelsen på og erfaringen til teamet 51
5.2.3. Begrenset erfaring med programvareutvikling 51
5.2.4. Forslag til forbedring av artefaktet 51
5.2.4.1. Innholdet i JSON-filen 51
5.2.4.2. Multithreading 52
5.2.4.3. Formatering av grafene 53
6. KONKLUSJON 55
7. REFERANSELISTE 57
APPENDIKS A: KILDEKODEN TIL ARTEFAKTET
APPENDIKS B: MALEN TIL MEDICATIONADMINISTRATION

Figurliste

Figur 1.1: Hjerte-lungemaskin 2
Figur 1.2: Overvåkningskurver 4
Figur 2.1: Lagvis modell over interoperabilitet 10
Figur 2.2: Strukturen til ressursen medicationAdministration. 15
Figur 2.3: Oppbygningen av en URL. 16
Figur 2.4: Et eksempel på en RS232-protokoll. 17
Figur 3.1: Information Systems Research Framework. 23
Figur 3.2: En Alaris CC med Guardrails sprøytepumpe. 24
Figur 3.3: Philips IntelliVue X2 pasientmonitor. 25
Figur 4.1: Scenarioet ved senkning av blodtrykket. 28
Figur 4.2: Scenarioet ved økning av blodtrykket. 28
Figur 4.3: Skjermdump av de to virtuelle sprøytepumpene. 31
Figur 4.4: Oversikt over artefaktets funksjoner. 32
Figur 4.5: Leser data fra virtuelle pumper. 33
Figur 4.6: Oversetter datastrømmen. 34
Figur 4.7: Viser en liste med tre datasampler lagret i en liste. 34
Figur 4.8: Skriver datastrømmen til JSON. 35
Figur 4.9: Formaterer y-verdiene. 36
Figur 4.10: Lager blodtrykkskurve. 37
Figur 4.11: Flytskjema for plotting av kurvene. 38
Figur 4.12: Integrert presentasjon av overvåknings- og medisineringskurve. 39
Figur 5.1: Viser et eksempel på journalnotat med alle samples. 48
Figur 5.2: En rack med infusjons- og sprøytepumper. 52
Figur 5.3: Eksempel på brukergrensesnitt med integrerte kurver. 53

Forkortelser

ANSI	American National Standards Institute
CEN	European Comittee for Standardisation
CDA	Clinical Document Administration
CTS	Clear to send
CVP	Central Venous Pressure
IEEE	Institute of Electrical and Electronics Engineers
IKT	Informasjons- og kommunikasjonssystemer
irDA	Infrared Data Association
ISO	International Standardisation Organization
JSON	JavaScript Object Notation
REST	Respresentational State Transfer
RFH	Resources For Health
RIM	Reference Information Model
RTS	Request to send
RxD	Receive data
SDC	System and Device Connectivity
TxD	Transmit data
XML	Extensible Markup Language

1. Innledning

Når man er i kontakt med helsevesenet forventer de fleste at helsepersonellet har nødvendig informasjon tilgjengelig og den kunnskapen de trenger for å kunne gi best mulig behandling. Det er ikke bestandig tilfelle i dag. For eksempel har hver spesialist sitt spesialistsystem med begrenset eller ingen mulighet til å utveksle kliniske data i den konteksten informasjonen ble høstet (1,2).

Utstyr kommuniserer spesielt dårlig med annet utstyr som ikke er fra samme leverandør. Da må gjerne IT-avdelingen lage et grensesnitt for å få systemene til å utveksle data. Dette svikter ofte og man må gå tilbake til å utveksle informasjonen på papir (1,3). Det er ikke uten grunn at helsevesenet er det største gjenværende markedet for penner, papir og faksmaskiner (4).

Vi mennesker genererer enorme mengder verdifulle data om mulige helsetilstander. I tillegg til at denne informasjonen gir oss oversikt over aktuell tilstand, kan den også gjenbrukes i fremtiden til å ta lærdom av tidligere helsetilstander og forløp. Dette forutsetter at informasjonen forvaltes riktig. Det kan gi oss mer og bedre kunnskap som grunnlag for bedre prosedyrer, tryggere og mer tilpasset behandling (5).

Informasjon blir i dag samlet på bakgrunn av plikten til å dokumentere når man behandler en pasient. Skal en forske samles kun en begrenset mengde forhåndsdefinerte data inn i forskningsprosjektets kontekst. Alt det man velger å ikke samle inn vil være tapt. Det blir feilaktig antatt at kliniske funn er overførbart til alle pasientkategorier uavhengig av øvrige diagnoser. Når en medisin virker som ønsket på en utvalgt gruppe pasienter vil den gjøre det på alle (6).

Å ha systemer for å samle inn og lage en tilpasset integrert visualisering av alle informasjonsstrømmer kan skape effektiv gjenbruk av data ved at det senker terskelen for forskning og statistikk. I tillegg har en mulighet til å supplere med data og endre vinkling underveis. Man trenger ikke å vite på forhånd hvilke data man trenger til forskningen (5).

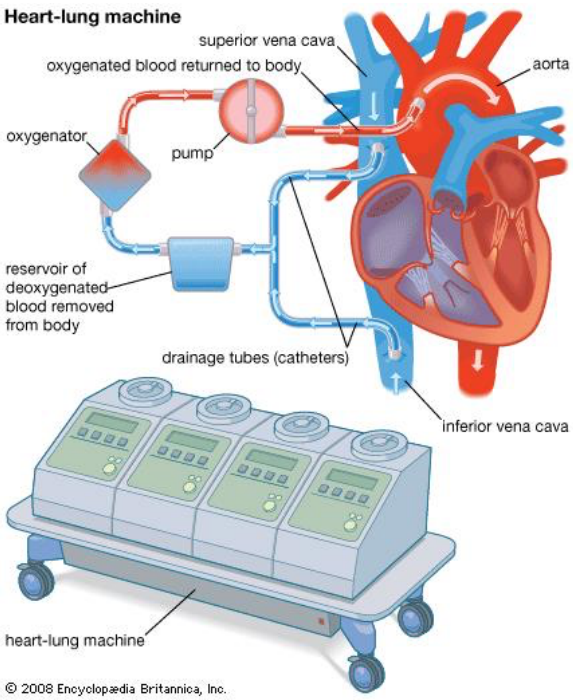
Hvis dette skal bli en realitet må interoperabilitet mellom systemene være på plass. For å oppnå interoperabilitet mellom systemer må informasjonen standardiseres.

Også innenfor kirurgien er det viktig med standardisering og interoperabilitet slik at man kan integrere datastrømmene og lage et felles brukergrensesnitt (7). Kirurgi er en tverrfaglig behandling og manglende integrering av informasjonen mellom de ulike spesialitetene som bidrar til utredning av pasientene kan få katastrofale følger (8). I denne oppgaven skal en benytte seg av en ny standard for utveksling av journaldata til å integrere og visualisere datastrømmer fra et klinisk scenario.

1.1. Egne erfaringer fra klinikken

Eksempelet fra klinikken er basert på egne erfaringer etter mange års klinisk arbeid.

Det er normalt anestesien som overvåker pasienten, administrerer narkosen, medisiner og dokumenterer behandlingen i den elektroniske kurven. Ett unntak er ved åpne hjerteoperasjoner.



Figur 1.1: Hjerte-lungemaskin

Bildet viser hvordan hjerte-lungemaskinen er koblet til kroppen og på den måten skilles både hjertet og lungene mekanisk fra resten av kroppen. De blå slangene tar blodet ut av kroppen og den røde setter det oksygenerte blodet tilbake i kroppen (9).

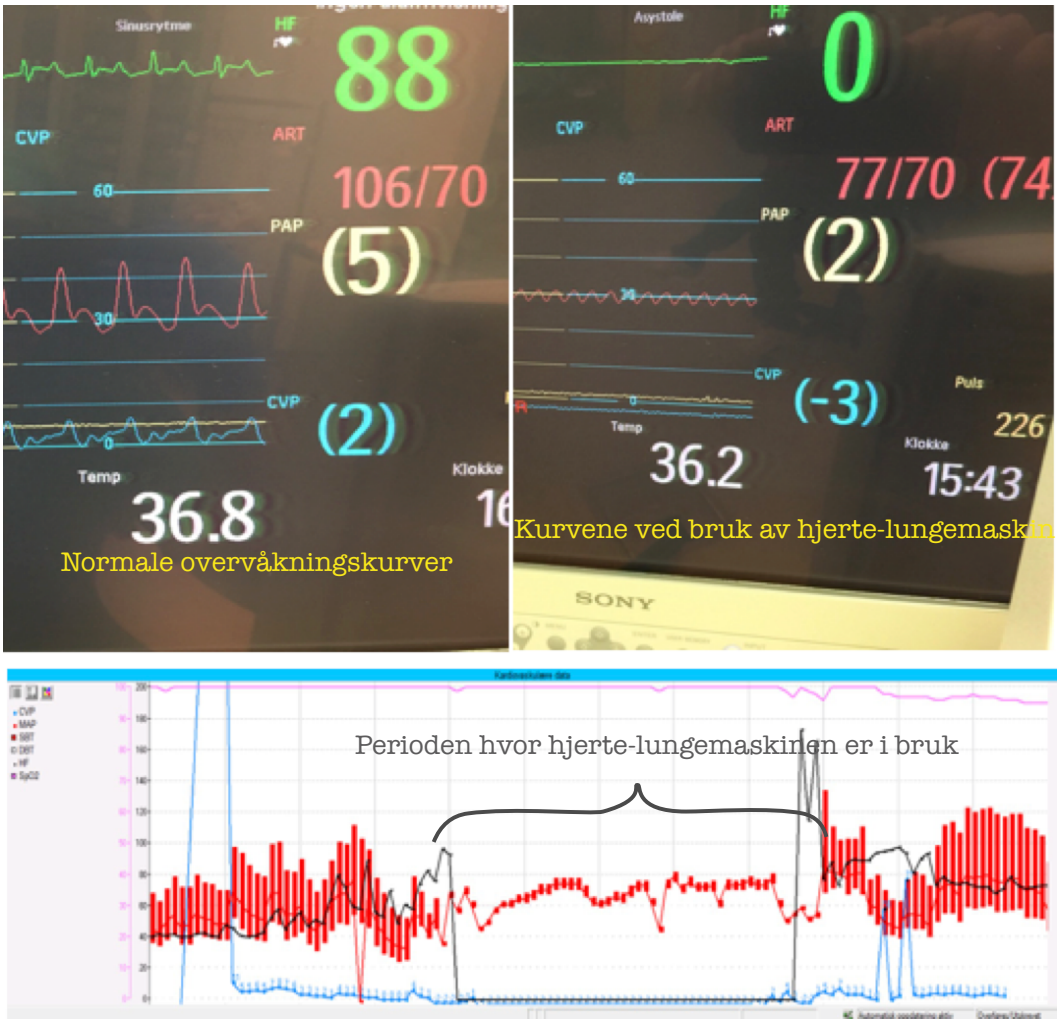
Mens man opererer på hjertet må det stoppes. Da er ikke pasienten i stand til å sirkulere blodet rundt i kroppen på egen hånd, se figur 1.1. Når ikke hjertet pumper går det ikke noe blod gjennom lungene. Derfor fungerer hverken hjertet eller lungene under en åpen hjerteoperasjon. For å erstatte hjertet og lungenes funksjon bruker man en hjerte-lungemaskin.

I den perioden hjerte-lungemaskinen er i bruk lagres flere av de kliniske parameterne som overvåkes i hjerte-lungemaskinens kurvesystem. Det genereres en del metadata i forbindelse med bruk av hjerte-lungemaskinen som gir supplerende kontekst til det totale bildet. Derfor hadde det vært ønskelig med utveksling av informasjon mot anestesiens kurvesystem.

På grunn av mangelen på interoperabilitet mellom systemet som anestesien bruker og systemet som brukes på hjerte-lungemaskinen kompletteres ikke anestesikurven med den manglende informasjonen som oppstod i perioden hjerte-lungemaskinen var i bruk, se figur 1.2. Overvåkningsdataene lagret på hjerte-lungemaskin følger derfor ikke med videre i behandlingen og man har ingen mulighet til å komplettere kurvene som dokumenterer behandlingsforløpet. Eller skaffe seg oversikt over uønskede hendelser på hjerte-lungemaskin i denne perioden. Fordi informasjonen lagres på forskjellige systemer, uten mulighet for utveksling, blir dokumentasjonen fragmentert.

Når informasjonen fragmenteres på denne måten forsvinner også konteksten den ble høstet i. Det er viktig å bevare konteksten i denne sammenhengen fordi noen av de fysiologiske overvåkningskurvene endrer utseende og bruksområdet som følge av hjerte-lungemaskinen. Sentralt venetrykk, CVP[1], sier noe om fyllingen av hjertet i dets hvilefase (10). Fordi hjerte-lungemaskinen henter blodet ut av pasienten ved hjelp av høydeforskjellen forsynes hjerte-lungemaskinen via en hevert. Mens hjerte-lungemaskinen er i bruk sier CVP noe om hvor godt heverten fungerer. En CVP på -3mmHg sier at det er god drenasje. Vanligvis vil en CVP på -3mmHg bety at pasienten lider av hypovolemi, altså for lite blodvolum og at hjertets pumpeevne dermed reduseres (11). Da vil man vanligvis fylle pasienten med væske. Et annet eksempel er EKG-kurven som til vanlig brukes til å overvåke hjertets elektriske aktivitet og rytme, brukes under operasjonen til å kontrollere at hjertet ikke har elektrisk aktivitet. Hvis man ikke kjenner konteksten i slike tilfeller kan det f.eks. se ut som pasienten har fått hjertestans.

[1] Man bruker den engelske forkortelsen CVP for Central Venous Pressure i dagligtalen.



Figur 1.2: Overvåkningskurver

Øverst til venstre viser en normalsituasjon med EKG og trykkurve. Øverst til høyre viser de samme kurvene etter at hjerte-lungemaskinen har overtatt. Nederste bilde viser forskjellen i anestesiens kurvesystem med og uten hjerte-lungemaskin. Pasienten ligger på hjerte-lungemaskin i den perioden i midten der kurvene er flate.

1.2. Noen av dagens utfordringer

Interoperabilitet er grunnlaget for at alle vet det de trenger å vite, når de trenger å vite det (7,12). Her beskrives et par eksempler på hvordan mangelen på interoperabilitet skaper utfordringer i det kliniske arbeidet.

1.2.1. Fragmentering av kliniske data

Helse og omsorg handler om kommunikasjon. Det moderne helsevesenet er avhengig av lagarbeid og kommunikasjon. Interoperabilitet er nødvendig for å gjøre informasjon tilgjengelig der det er behov for den (4).

Likevel er det slik at de sykeste pasientene, som gjerne har flere diagnoser og som går til behandling hos mange spesialister sier at manglende utveksling av informasjon fører til bekymring. Det er spesielt i overføringen mellom de behandlende institusjonene og oppfølging rundt

medisinering at utveksling av klinisk informasjon svikter (13). Utilstrekkelig utveksling av informasjon er en av de vanligste årsakene til feilmedisinering (14).

Ved mangelfull utveksling av informasjon, blir den gjerne liggende lokalt. Når man er i kontakt med ulike behandlende og utredende institusjoner blir det da lagret små biter av informasjon ved hver plass man har vært. Dette kalles fragmentering. Fragmentering av kliniske informasjon fører til ufullstendige journaler, feil, dobbeltføring og bortkastet arbeid (4).

Når kliniske data blir spredt over mange datalager, ved at alle som behandler pasienten lagrer sine observasjoner i egne system. Så inneholder ingen av datalagrene det totale bildet og dermed blir muligheten til anvendelse av all informasjon som er samlet inn av ulike spesialister, f.eks. i forbindelse med en utredning eller behandlingsopphold, begrenset (6).

Det er ønskelig å dele informasjon, men det er en risiko for at man kan bli oversvømt av informasjon (15). Derfor er det viktig at en har gode muligheter til å søke og filtrere på den informasjonen en trenger til gjeldende kontekst og behandling.

Tross i all teknologien som brukes på dagens moderne operasjonsstuer blir fremdeles kirurger isolert fra resten av sykehuset nå de opererer og kommunikasjonen fragmentert (7).

1.2.2. Kliniske data som mangler kontekst

Kliniske data som blir tatt ut av journalen, lagret i egne registre og administrert av personer som ikke er involvert i pasientbehandlingen, kalles for frikoblede data (6). Disse skaper flere utfordringer fordi informasjonen mister konteksten den er lagret i, har begrenset verdi i forskning og det er ikke mulig å gi tilbakemeldinger til det miljøet der dataene oppstod.

Det er viktig å bevare konteksten de kliniske dataene oppstod i for at de skal gi mening (16). Det er mulig å frikoble data, men at dette krever en del merarbeid. For at gjenbruk av data skal ha noen verdi er det viktig å vite hvordan informasjonen oppstod. Det sikrer riktig fortolkning av informasjonen (6).

Standardiserte metadata en lavt hengende frukt som gjør informasjonsdeling tilgjengelig mellom de som behandler pasienten (4).

1.3. Problemstilling

Som tidligere nevnt står utveksling av informasjon helt sentralt i behandlingen av pasienter. Likevel er informasjonsflyten en utfordring. Det genereres en stor mengde data fra mange kilder som lagres flere plasser under en åpen hjerteoperasjon. I dag integreres ikke data fra de forskjellige journalsystemene som brukes og bare noe av informasjonen følger med pasienten videre. Det gjør at det mangler en helhetlig dokumentasjon av behandlingsforløpet.

Det optimale er om all denne informasjonen integreres og følger med pasienten videre under oppholdet slik at en lett kan skaffe seg overblikk over behandlingen som er utført.

Informasjonen skal kunne utveksles, tolkes og brukes uten at det er tvil om betydingen av innholdet. For å kunne gjøre dette må informasjonen standardiseres. HL7 FHIR gjør dette mulig ved hjelp av åpne webstandarder.

Noen argumenter for å benytte HL7 FHIR:

- Laget for utveksling av medisinsk informasjon
- Stor brukergruppe
- Alle sykehus støtter nativt HL7 FHIR
- Lett å implementere og gjennomføre
- Fritt tilgjengelig rammeverk, ingen innkjøpskostnader. Ingen restriksjoner på bruk.
- Ferdig tilrettelagt for interoperabilitet ved hjelp av ressurser.
- Bruker Web-standarder som f. eks. XML, JSON, HTTP og OAuth.
- Støtte for RESTful arkitekturer som gir sømløs utveksling av informasjon.
- Konkrete og lett forståelige spesifikasjoner.
- Har en solid ontologi

(17)

Problemstillingen er:

Kan HL7 FHIR brukes til å legge til rette for integrering av sanntids datastrømmer?

Leseguide

Innledningen, kapittel 1, setter leseren inn i konteksten for oppgaven. Og avslutter med en isolert problemstilling som er knyttet til noen av utfordringene i dagens kliniske praksis.

Teori, kapittel 2, knytter sammen eksisterende kunnskap og hva andre har gjort i et overordnet rammeverk for å løse problemstillingen. Her gjøres rede for begreper og annen teori som benyttes senere i oppgaven.

Metodekapitlet, kapittel 3, beskriver hvordan jeg har løst oppgaven.

Resultat, kapittel 4, består av to deler. Starter med en beskrivelse av utviklingen av prototypen, deretter en beskrivelse av simuleringen.

Diskusjonsdelen, kapittel 5, er også todelt. Først en vurdering av artefaktet siden det ikke er mulig å gjennomføre en fullstendig evaluering. Her drøfter jeg resultatene mine og setter de i en sammenheng. Deretter en beskrivelse av egne erfaringer med arbeidet og gjennomgang av noen forbedringer som må gjøres for å gå fra proof-of-concept til en fungerende prototype.

I kapittel 6 kommer konklusjon.

2. Teori

2.1. Standarder og interoperabilitet

I innledningen beskrives behovet for å standardisere informasjonen slik at den lettere kan utveksles og tolkes automatisk mellom systemer og institusjoner. I denne oppgaven er det FHIR, uttales ´fire´, den nyeste standarden fra HL7 som benyttes. I dette kapittelet blir det en gjennomgang av teori i forbindelse med standarder og interoperabilitet, hva HL7 FHIR er og hvordan den kan benyttes til å løse denne problemstillingen.

2.1.1. Hva er interoperabilitet?

Institute of Electrical and Electronics Engineers (IEEE) har definert interoperabilitet som «muligheten for to eller flere systemer eller enheter til å utveksle og forstå informasjonen som utveksles». Interoperabilitet gjøres mulig ved å implementere standarder (18).

Interoperabilitet er et ikke-funksjonelt krav som står sentralt for å oppnå suksess med helseinformasjonssystemer. Det er helt klare fordeler med å ha det beste informasjonsgrunnlaget tilgjengelig når man behandler pasienter.

For å oppnå interoperabilitet må man blant annet definerer den syntaktiske og semantiske meningen med informasjonen som utveksles. Konsistente implementerbare standarder reduserer risikoen, tidsforbruket og kostnadene på prosjekter knyttet til utvikling av helseteknologi (19).

Interoperabilitet er essensielt for bedre behandling. I dag fungerer det meste av utstyr og maskiner som frittstående enheter uten kommunikasjonsmuligheter (7).

2.1.1.1. De ulike typene interoperabilitet

Definisjonen av interoperabilitet legger til grunn flere typer interoperabilitet. For å kunne utveksle informasjon er man avhengig av teknisk interoperabilitet og for at mottakeren skal kunne nyttiggjøre seg av informasjonen trenger man semantisk interoperabilitet. Derfor kan man betrakte interoperabilitet lagvis, se figur 2.1. Der de ulike typene interoperabilitet danner grunnlaget for neste nivå. De typene som er nødvendig for å løse denne oppgaven er teknisk, syntaktisk og semantisk interoperabilitet. Da kan man få et system som kan høste data, standardisere informasjonen, gjøre den lesbar, tolkbar, forståelig både for mennesker og datamaskiner og utveksle den fritt.

Teknisk interoperabilitet

Teknisk interoperabilitet er at informasjon flyttes fra ett system til et annet, uavhengig av innhold. I denne oppgaven vil det være når en overfører datastrømmen fra sprøytepumpene til artefaktet som oversetter dataene til HL7 FHIR standarden.

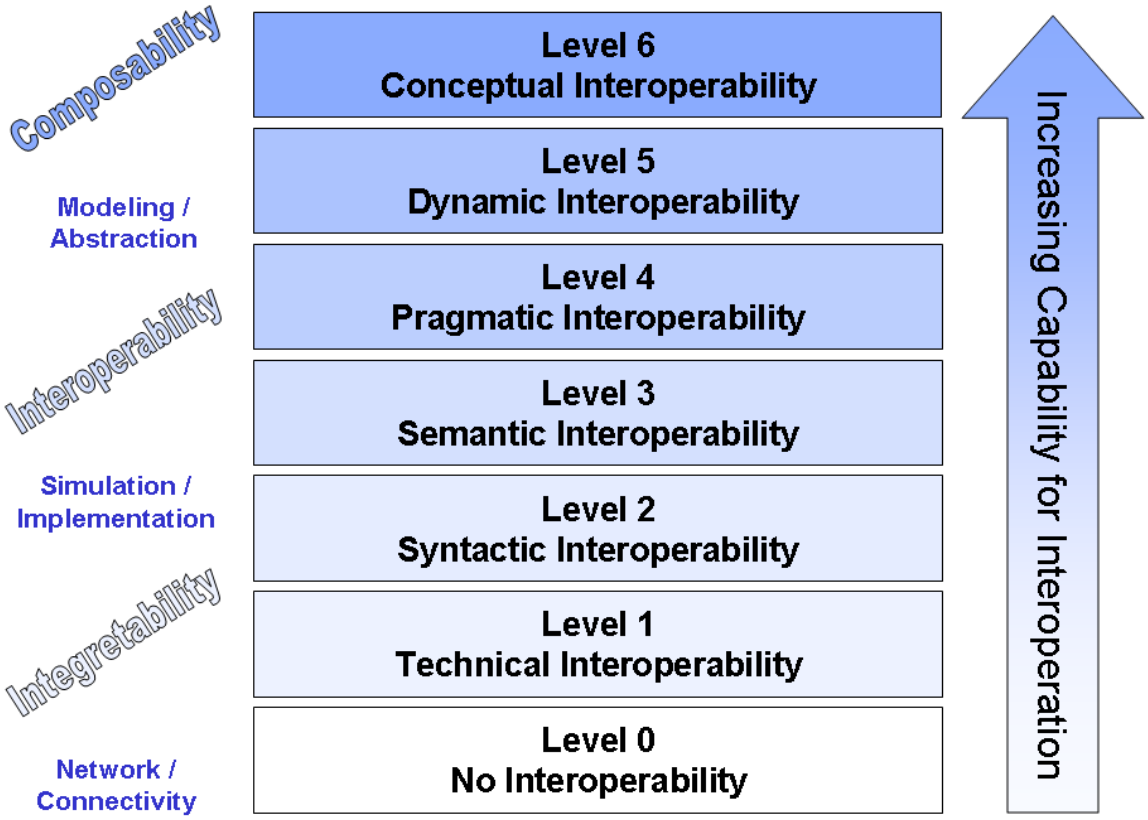
Syntaktisk interoperabilitet

Når to eller flere systemer som kan utveksle informasjon med hverandre gjør det på et standard dataformat kalles det syntaktisk interoperabilitet. JSON er en slik standard som danner grunnlaget for syntaktisk interoperabilitet.

Semantisk interoperabilitet

Den syntaktiske interoperabiliteten må være på plass før man kan oppnå semantisk interoperabilitet. Det er når systemene kan dele, forstå, tyde og bruke informasjonen uten at det er tvil om betydningen av disse (4,20).

Figur 2.1 viser hvordan de ulike typene av interoperabilitet henger sammen. De ulike lagene bygger på hverandre for å oppnå høyere grad av interoperabilitet.



Figur 2.1: Lagvis modell over interoperabilitet

Hvert nivå danner grunnlaget for det neste (21).

2.1.2. Hvorfor trenger vi interoperabilitet?

I det daglige arbeid med pasienter handler helsevesenet i stor grad om kommunikasjon mellom personer og organisasjoner. Interoperabilitet er nødvendig for å gjøre informasjonen forståelig og tilgjengelig der det er behov for den, når det er behov for den. Det vil også redusere merarbeidet og bedre sikkerheten i form av færre manuelle datahøstinger hvor det kan oppstå feil (4).

I en kartlegging av erfaringene til de sykeste pasientene, som ofte har utstrakt bruk av helsevesenet, hadde på tvers av spesialiteter ble det blant annet avdekket dårlig kommunikasjon, spesielt mellom institusjoner, feilbehandling, ikke optimal behandling og feilmedisinering (13). Dette viser at høy spesialisering av behandlingen og tilhørende fragmentering av informasjonen stiller store krav til informasjonsutveksling.

Helsevesenet mangler utstrakt bruk av standarder for å oppnå interoperabilitet. Uten slike standarder kan ikke data som deles forstås av systemene som mottar de. Utvikling av standarder vil danne grunnlaget for utvikling av interoperabilitet (7). Det er et behov for å utvikle tilstrekkelige standarder som kan tas i bruk av helsevesenet (22).

2.1.3. Hva er en standard?

Definisjonen på en standard er «en teknisk spesifikasjon som beskriver hvordan ulike objekter skal kunne defineres på en entydig måte» (23).

To sentrale begreper innen standardisering er konsensus, som er den generelle enigheten om hvordan det bør være, og en internasjonal standardiseringsorganisasjon som er allment akseptert. Selve standardiseringen består av prosesser for å formulere, utstede og implementere standarder. I læreboken «Principles of Health Interoperability» står det at interoperabilitet i helsevesenet er basert på bruken av standarder (4).

En av flere utfordringer knyttet til analysering av store datamengder er å få standardisert dataene. Dette er nødvendig siden flertallet av pasientene gjerne kommer fra eller er innom flere sykehus eller deler av helsevesenet. Derfor er det stor fragmentering av informasjon og ingen har total oversikt (24).

Likevel har det vært vanskelig å oppnå en enkelt, godt utbredt standard (19).

2.1.3.1. Hvordan forvaltes standardene?

ISO

Den Internasjonale standardiseringsorganisasjonen (ISO) ble etablert i 1947 med fokus på internasjonale standarder. Det er en medlemsorganisasjon, som baserer seg på at hvert land har sin egen standardiseringorganisasjon som er medlem i ISO.

IEEE

Institute of Electrical and Electronics Engineers (IEEE) er en ideell organisasjon innenfor elektronikk og elektroteknikk. De har mer enn 360 000 medlemmer i 175 land. IEEE definerer standarder innenfor en rekke områder som for eksempel datateknikk, biomedisin, telekommunikasjon, elektroteknikk, elkraftteknikk, romfarts- og forbrukerelektronikk (18).

CEN og Standard Norge

Det finnes en europeisk standardiseringsorganisasjon, European Comittee for Standardisation (CEN), som ble etablert i 1961. I Norge har vi Standard Norge som ble stiftet i 2003, men har røtter helt tilbake til 1923. Standard Norge er medlem i både ISO og CEN og sikrer at norske interesser blir ivaretatt når internasjonale standarder utformes (25,26).

ANSI

Den amerikanske medlemsorganisasjonen i ISO heter American Nasjonal Standards Institute (ANSI) (27).

Wien-avtalen

Wien-avtalen regulerer konflikter mellom standarder. I hovedsak er det slik at internasjonale standarder har prioritet foran nasjonale standarder (4,26).

2.1.4. Health Layer 7

HL7 er en ideell, ANSI-akkreditert standardiseringsorganisasjon. De ble etablert i 1987 for å utvikle standarder forutveksling av helseinformasjon.

HL7 sin visjon er «A world in which everyone can securely access and use the right health data when and where they need it» (28).

HL7 er den mest utbredte standarden for utveksling av informasjon i helsevesenet. Og består av over 1600 medlemmer fra mer enn 50 land. De utvikler standarder for å oppnå bedre inte-

roperabilitet. Deres to første meldingsutvekslingsstandarder het HL7 v2 og HL7 v3. Disse kom i henholdsvis 1989 og 2005.

2.1.4.1. HL7 v2

HL7 v2 ble utviklet til det formål å forene ulike sykehussystemer. Det var designet slik at 80% av grensesnittene var definert i spesifikasjonene til HL7 og 20% ved lokale tilpasninger. Ett av problemene med HL7 v2 var at det ikke hadde en ontologi (19). Dette førte til at det ble mange lokale tilpasninger av standarden, som igjen gikk ut over interoperabiliteten (1). Så mye som 94% hadde en form for syntaks- eller semantisk feil. Det mest vanlige problemet var datafelt som ikke ble brukt eller data i feil felt (29).

Den høye graden av tvetydighet og behovet for lokale tilpasninger førte til utviklingen av HL7 v3 (3).

2.1.4.2. HL7 v3

Utviklingen startet i 1995 og formålet var å utbedre svakhetene i v2. Men man startet på nytt istedenfor å videreutvikle v2. Vinklingen var også annerledes. I v2 var formålet å forene ulike sykehussystemer, mens i v3 valgte de en ovenfra og ned tilnærming. I en slik tilnærming skisserer man hele systemet først, for så å beskrive ett og ett undersystem i detalj. Man definerte strukturen på semantikken i en «Reference Information Model» (RIM). Dette ga en høy grad av informasjonsintegritet, men bidro også til å øke kompleksiteten (3,30).

Til tross for at HL7 v3 benyttet XML-teknologi og objekt orientert tilnærming, førte den høye kompleksiteten til at svært få tok i bruk v3. Sannsynligvis fordi det er svært vanskelig å modellere helsevesenet i sin helhet (2).

HL7 v3 hadde en omfattende ramme, omhyggelig detaljert og konsekvent i anvendelse. På dette området var v3 en suksess (4). Men v2 og v3 var ikke direkte kompatible. Overføring av data mellom v2 og v3 krevde egen programvare for å oversette informasjonen. En gruppe utviklere brukte 18000 arbeidstimer på å utvikle et verktøy for å implementere v3 (19). Terskelen for å komme i gang ble veldig høy og HL7 v3 ble aldri særlig utbredt. Eneste unntaket var HL7 v3 CDA. En standard som beskriver semantikken og strukturen i kliniske dokument med den hensikt å kunne utveksle informasjon mellom de ulike interessentene (3).

2.1.4.3. Fast Healthcare Interoperability Resource

HL7 v3 hadde oppfylt sine egne mål, men ikke HL7 sitt overordnede mål om å gjøre interoperabilitet billigere og enklere.

Derfor nedsatte HL7 i 2011 noe de kalte «fresh look task force». En arbeidsgruppe uten bestemte føringer til resultatet, som skulle se på hvordan HL7 best kunne lage løsninger som fremmet interoperabilitet (4,19).

Først måtte man utrede styrker og svakheter med de standardene man allerede hadde. Deretter så man etter eksempler fra virkeligheten hvor interoperabilitet fungerte bra.

En gruppe HL7-utviklere benyttet muligheten de fikk til å tilnærme seg problemet med utveksling av helseinformasjon ved å bruke eksisterende web-standarder. I starten het prosjektet «Resources for Health» (RFH), senere skiftet de navn til «Fast Healthcare Interoperability Resources» (FHIR) (19,31).

I motsetning til v2 som benyttet en formålsrettet utviklingsmetode og v3 som hadde en ovenfra og ned tilnærming, valgte man i HL7 FHIR å benytte seg av smidig metode. Det vil si at en utvikler systemet stegvis hvor en hele tiden jobber mot delmål i en gjentagende prosess. En

fordel er at man jobber tett opp mot brukeren og har en kontinuerlig dialog gjennom utviklingen av systemet.

FHIR er bakoverkompatibelt med v2 og v3 og fremhever det beste fra tidligere versjoner, mens den utnytter det siste innen web-baserte standarder med fokus på å gjøre implementeringen enklere.

Fleksibiliteten og modulariteten til FHIR danner grunnlaget for nye muligheter til kommunikasjon mellom medisinsk utstyr og helseinformasjonssystemer. Og bidrar til å viske ut tidligere barrierer mellom medisinsk utstyr og klinisk IKT (3).

Det er vanlig at et helseinformasjonsystem filtrerer ut mye av observasjonene som gjøres av det medisinske utstyret fordi det er ikke konstruert for å samle på og analysere store datamengder. Eksisterende helseinformasjonsystemer er konstruert for å lagre funn og resultater, ikke kontinuerlige observasjoner. Med FHIR kan man sammenfatte informasjon fra eksisterende datalagre og gjeldende data ved hjelp av underliggende funksjonalitet for prosessering av diversifiserte data (3).

2.1.4.4. Hva er ressurser?

Ressurser beskrives som en logisk enhet som inneholder informasjon som kan lagres eller utveksles i integrerte helseløsninger. FHIR representerer kliniske data som ressurser. En ressurs er kort fortalt et dataelement, som for eksempel et dokument eller et journalnotat, brukt til å lagre eller utveksle data i integrerte helseløsninger (32).

Ressurser står sentralt i FHIR. Det er dataelementer som inneholder definisjonen på alt innholdet i dokumentene og legger grunnlaget for informasjonsutvekslingen. Det er et artefakt som kombineres på ønsket måte for å danne grunnlaget for et fungerende system hvor kontekst og behandling kan dokumenteres i journalen på et format som både er leselig for mennesker og maskiner.

Hver ressurs er logisk uttrykt med veldefinerte objekter og datatyper. Definisjonene er konkrete og intuitive som f.eks. medicationAdministration (30).

En ressurs kan ha flere representasjoner. Det vil si at dokumentet kan eksistere på flere forskjellige format med det samme innholdet (33). Som i tilfelle med HL7 FHIR-dokumenter for eksempel XML og JSON.

Man kombinerer disse ressursene slik at de relevante variabler blir benyttet i en lokalt tilpasset profil. En profil er definert som et sett med regler som omhandler innholdet i en ressurs. Slike regler som begrenser bruken av en eller flere ressurser danner grunnlaget for en profil som er tilpasset lokal bruk (34).

Bruken av ressurser og web-standarder gjør at FHIR er veldig fleksibelt i bruk og kan tilpasses mobilbruk, skylagring, meldingsutveksling og mye mer.

Når man skal lage en fungerende løsning kombinerer man de tilgjengelige ressursene slik at en får et fungerende system for akkurat den arbeidsflyten en skal løse. På den måten setter man sammen sin egen lokale løsning som baserer seg på lokale use-cases. Resultatet er at de lokale dataene standardiseres. Dette gjør gjenbruken av informasjonen fleksibel og at den enkelt kan deles i den konteksten den oppstod.

I spesifikasjonene er det forhåndsdefinert mange forskjellige ressurser (35).

En ressurs beskrives av HL7 som en entitet hvor:

- En har kjent identitet i form av en URL som kan adresseres.

- En av ressursene er definert i HL7 FHIR spesifikasjonen
- Innholdet er et strukturert datasett som beskrevet i henhold til ressursens definisjon
- En har en identifiserbar versjon som oppdateres når ressursen oppdateres.

(36)

Maturity level

Maturity level sier noe om hvor godt testet og stabil ressursen er. HL7 FHIR har et system for rapportering av utprøving slik at ressursene modnes etterhvert som de testes ut i de ulike utviklingsmiljøene rundt om kring.

Modenheten til en ressurs deles inn i 7 nivåer fra 0 til 6. På det laveste nivået er ressursen ansett som et utkast. Når en ressurs er godt utprøvd, stabil og betraktet for å være klar for klinisk bruk er den på nivå 6 (37). Så langt har ingen ressurser oppnådd nivå 6 (38).

Eksempel på en ressurs

Som eksempel brukes samme ressurs som benyttes i artefaktet. Ressursen `medicationAdministration` er laget for å beskrive hendelsen forbundet med å ta eller få medisiner. Den dekker alt fra tabletter til kontinuerlig infusjon. Relaterte ressurser knytter denne til forordning og andre møter mellom helsepersonell og pasient (39). Figur 2.2 viser strukturen i en ressursen `medicationAdministration`.

Forklaring på kolonnene:

- Name: Inneholder navnet på elementet
- Flags: Inneholder informasjon om elementet som påvirker hvordan man skal håndtere de i implementeringen.
- Card.: Angir kardinaliteten til elementet. Nedre og øvre grense for hvor mange ganger elementet skal eller kan brukes i ressursen.
- Type: Type element og hva det inneholder.
- Description & Constraints: Beskriver elementet, hva det er og ikke er, samt dets begrensninger.

11.2.3 Resource Content

Structure | UML | XML | JSON | Turtle | R2 Diff | All

Structure

Name	Flags	Card.	Type	Description & Constraints
MedicationAdministration	I		DomainResource	Administration of medication to a patient *+ Reason not given is only permitted if NotGiven is true* *+ Reason given is only permitted if NotGiven is false* Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension
identifier		0..*	Identifier	External identifier
definition	Σ	0..*	Reference(PlanDefinition | ActivityDefinition)	Instantiates protocol or definition
partOf	Σ	0..*	Reference(MedicationAdministration | Procedure)	Part of referenced event
status	?! Σ	1..1	code	in-progress | on-hold | completed | entered-in-error | stopped | unknown MedicationAdministrationStatus (Required)
category		0..1	CodeableConcept	Type of medication usage MedicationAdministrationCategory (Preferred)
medication[x]	Σ	1..1		What was administered SNOMED CT Medication Codes (Example)
medicationCodeableConcept			CodeableConcept	
medicationReference			Reference(Medication)	
subject	Σ	1..1	Reference(Patient | Group)	Who received medication
context		0..1	Reference(Encounter | EpisodeOfCare)	Encounter or Episode of Care administered as part of
supportingInformation		0..*	Reference(Any)	Additional information to support administration
effective[x]	Σ	1..1		Start and end time of administration
effectiveDateTime			dateTime	
effectivePeriod			Period	
performer	Σ	0..*	BackboneElement	Who administered substance
actor	Σ	1..1	Reference(Practitioner | Patient | RelatedPerson | Device)	Individual who was performing
onBehalfOf		0..1	Reference(Organization)	Organization organization was acting for
notGiven	?! Σ	0..1	boolean	True if medication not administered
reasonNotGiven	I	0..*	CodeableConcept	Reason administration not performed SNOMED CT Reason Medication Not Given Codes (Example)
reasonCode	I	0..*	CodeableConcept	Reason administration performed Reason Medication Given Codes (Example)
reasonReference		0..*	Reference(Condition | Observation)	Condition or Observation that supports why the medication was administered
prescription		0..1	Reference(MedicationRequest)	Request administration performed against
device		0..*	Reference(Device)	Device used to administer
note		0..*	Annotation	Information about the administration
dosage	I	0..1	BackboneElement	Details of how medication was taken *+ SHALL have at least one of dosage.dose or dosage.rate[x]*
text		0..1	string	Free text dosage instructions e.g. SIG
site		0..1	CodeableConcept	Body site administered to SNOMED CT Anatomical Structure for Administration Site Codes (Example)
route		0..1	CodeableConcept	Path of substance into body SNOMED CT Route Codes (Example)
method		0..1	CodeableConcept	How drug was administered SNOMED CT Administration Method Codes (Example)
dose		0..1	SimpleQuantity	Amount of medication per dose
rate[x]		0..1		Dose quantity per unit of time
rateRatio			Ratio	
rateQuantity			SimpleQuantity	
eventHistory		0..*	Reference(Provenance)	A list of events of interest in the lifecycle

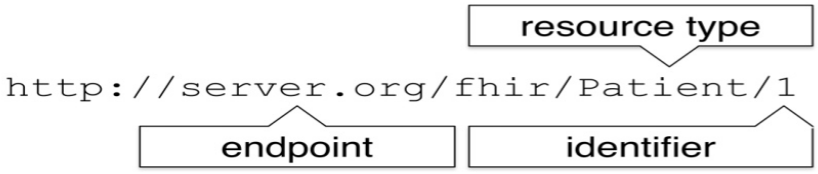
Figur 2.2: Strukturen til ressursen medicationAdministration.

(39)

2.1.4.5. Representational State Transfer

REST er en måte å legge til rette for interoperabilitet mellom datamaskiner på internett (40). HL7 FHIR benytter seg av det som kalles RESTful, som har litt løsere retningslinjer enn REST. I praksis åpner RESTful for noen flere tolkninger enn REST. For eksempel kunne man ikke hatt representasjoner av informasjonen i JSON hvis man skulle fulgt REST slavisk. Bruken av URL som peker på bestemte ressurser og bruk av en tilstandsløs kommunikasjonsprotokoll, hvor hverken sender eller mottaker tar ansvar for innholdet i dataene som sendes, har lagt grunnlaget for den svært skalerbare og desentraliserte strukturen til World Wide Web (19,41).

Ved å bruke RESTful kan man, ved å lagre journaldata på strukturert form ved hjelp av XML eller JSON, benytter seg av web-baserte løsninger, hvor hvert dokument får sin egen nettadresse, se figur 2.3. Disse dataene lagres på en web-server og hentes fram igjen på samme måte som en henter fram en nettside. Man kan søke i og filtrere dokumentene på samme måte som en søker på nettsider eller etter informasjon på nettet (32).



Figur 2.3: Oppbygningen av en URL.

(32)

2.1.4.6. Clinicians on FHIR

HL7 FHIR består av et stort miljø av klinikere og utviklere. Det er høy aktivitet og det skjer mye rundt denne standarden. En samling av verktøy som er utviklet for å lette samarbeidet, kommunikasjonen og forståelsen mellom klinikere og utviklere er *Clinicians on FHIR* (42). Ett av verktøyene heter *scenario builder*. Den tar utgangspunkt i en use-case og brukes til å konstruere ulike kliniske scenarioer ved å kombinere de nødvendige ressursene. En kliniker trenger ikke å lære og forstå informasjonsteknologi, men kun ta utgangspunkt i dagens praksis og sette sammen de ressursene som tar vare på informasjonen som genereres av gjeldene use-case. Fordelen er at semantikken blir bevart og programmereren kan lage profiler basert på de ulike scenarioene direkte uten å ha noe kunnskap om helse.

2.2. Medisinsk utstyr

Målet med oppgaven er å beskrive en mulig løsning på overføring av klinisk informasjon fra medisinsk utstyr, hvor kliniske målinger og observasjoner gjøres, til helseinformasjonsystemet hvor behandlingen dokumenteres.

Forskrift for Medisinsk Utstyr §1.5 definerer Medisinsk Utstyr som:

> Ethvert instrument, apparat, utstyr, programvare, materiale eller annen gjenstand som brukes alene eller i kombinasjon, herunder programvare som av produsenten er tiltenkt å brukes spesielt til diagnostiske og/eller terapeutiske formål og som kreves for riktig bruk, og som er ment å skulle brukes på mennesker.
>
> Utdrag fra (43)

Medisinsk utstyr er helt uunnværlig i behandlingen av pasientene. Utstyret er kilde til det meste av den kliniske informasjonen som observeres. Samtidig kan en se for seg at utstyret benyttes som grunnlag for dokumentasjon inn i helseinformasjonsystemet.

2.2.1. Datakommunikasjon

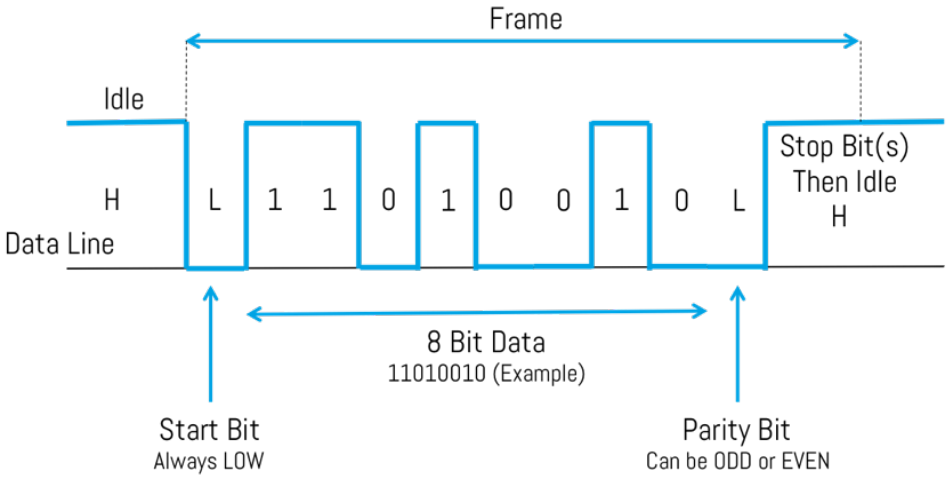
> «If standards are the language we use in healthcare, then the industry is missing the telephone lines connecting everyone speaking it. Though so advan-

> ced in so many other ways, communication in healthcare currently lives in the land of telegraphs and the pony express» (44).

Veldig mye av det medisinske utstyret mangler oppdaterte tilkoblingsmuligheter.

RS232-protokoll

En av de eldre standardene som fremdeles er utbredt på medisinsk utstyr er RS232-protokollen. Den sender data serielt og det krever at man oversetter informasjonen før den sendes inn i helseinformasjonsystemet.



Figur 2.4: Et eksempel på en RS232-protokoll.

(45)

Figur 2.4 viser en mulig konfigurasjon av et RS232-signal. Ett tegn består av 8 bit og en typisk konfigurasjon for RS232 inneholder 10 bit. 1 start bit, 8 databit for tegnet som sendes og et stop bit. Det finnes mange forskjellige konfigurasjoner, ca. 30 totalt (45).

IEEE 11073

IEEE 11073[2] er en nyere standard for utveksling av informasjon mellom medisinsk utstyr og/eller mot eksterne informasjonssystemer. Den gir mulighet for automatisk høsting av kliniske data og metadata.

Det er to målsettinger med standarden:

- Sanntids plug-and-play interoperabilitet mellom medisinsk utstyr.
- Sømløs utveksling av kliniske data hvor som helst, når som helst.

Problemene man ønsker å løse med denne standarden er:

- I mangelen på en standard blir data enten høstet manuelt eller med proprietært utstyr eller ikke høstet i det hele tatt.
- Manuell høsting av data er ekstraarbeid, for sjelden dokumentert eller utsatt for menneskelige feil.

[2] Kalles også Medical Health Device Communication standard

- Bruk av proprietære løsninger øker kostnaden, blir bare brukt på de mest nødvendige pasientene og låser brukeren til en bestemt løsning eller firma.
- Videreutviklingen og utrullingen av integrerte behandlingsløsninger hindres fordi man ikke kan sammenfatte data og lære av det.
- Ved manglende standardisering er det ingen konsistens i informasjonen som blir lagret.

(46)

IEEE 11073 er en familie av standarder bestående av:

- IEEE 11073-20702 som ivaretar muligheten for å utveksle data på det grunnleggende nivået.
- IEEE 11073-10207 som definerer strukturen på informasjonen som utveksles. Dette sikrer semantisk konsistent fortolkning av dataene.
- IEEE 11073 20701 beskriver den overordnede arkitekturen og definerer sammenhengen mellom de to tidligere nevnte standardene.

Kombinasjonen av disse spesifikasjonene sikrer semantisk interoperabilitet på tvers av utstyrsprodusenter (47).

Selv om dette er den eneste standarden for plug-and-play interoperabilitet med medisinsk utstyr, er den ikke veldig utbredt. Årsaken kan være kompleksiteten eller at leverandørene ønsker å holde sine grensesnitt proprietære (48).

Når en snakker om hele familien av IEEE 11073 standarder kalles den IEEE 11073 SDC.

2.3. Hva har andre gjort

HL7 FHIR er en ny standard som er i kontinuerlig utvikling. Det finnes noen artikler som omhandler prosjekter der man har prøvd å innføre HL7 FHIR. Her blir det en kort gjennomgang av lignende prosjekter som bygger på HL7 FHIR.

2.3.1. Utveksling mellom apparater og HIS

Interoperabilitet mellom enheter er fundamentalt for å kunne integrere ny teknologi. Dessverre har ledende produsenter investert betydelige midler i proprietære løsninger. Dette hindrer åpen utveksling av informasjon på tvers av produsentene (48).

I et samarbeid mellom klinikk og industri ble det konstruert en operasjonsstue hvor alt det medisinske utstyret og kliniske IKT-systemet snakket sammen. Dette er et fleksibelt system hvor alt er tilpasset bestemte use-case. Det har toveis kommunikasjon slik at f.eks. alle apparater kirurgen betjener med en fotpedal kan betjenes med én og samme fotpedal. Dette gir ett grensesnitt mot alle apparater og en unngår ett grensesnitt mot hvert apparat og like mange pedaler. Alt er basert på åpne standarder og man benyttet den nye standarden IEEE 11073 SDC for kommunikasjon mot det medisinske utstyret (47).

2.3.2. Utveksling mellom IKT systemer

Med unntak av det tyske OR.NET prosjektet (47,49) så fokuserer det meste av HL7 FHIR litteraturen på utveksling mellom kliniske IKT systemer. Enten mellom stasjonære systemer (30), mot mobile enheter (2), eller overføring fra gamle IKT-systemer til HL7 FHIR (3,50).

Det ble utviklet et system som høstet informasjon fra sensorer på kroppen ved hjelp av bluetooth, for å overvåke eldre på et pleiehjem. Systemet bearbeidet og analyserte informasjonen, men lagret informasjonen i HL7 FHIR ressurser (51).

2.3.3. Hva skiller denne oppgaven fra andre

Denne oppgaven skiller seg fra det som ellers står i litteraturen ved at den henter data som kommer fra den gamle, men svært utbredte RS232-protokollen, oversetter og lagrer dette til en relevant HL7 FHIR ressurs.

3. Metode

Formålet med prototypen er å demonstrere en løsning på problemstillingen ved å standardisere datastrømmen fra eksisterende medisinsk utstyr. Dette distribueres til en overvåkningskurve som det er mulig å oppdatere i sanntid og lagre i et journalnotat ved å bruke HL7 FHIR-standarden.

Design science er forskningsmetoden som benyttes i denne oppgaven til å utvikle et artefakt for å løse problemstillingen.

I prototypen blir det benyttet to tekstbaserte datastrømmer som simulerer to sprøytepumper. Datastrømmen fra disse oversettes og lagres i nødvendige JSON-filer som er formatert i henhold til malen for ressursen `medicationAdministration` (39). Datastrømmen blir i tillegg presentert grafisk for å vise at en kan benytte HL7 FHIR som grunnlag for integrering av sanntidsinformasjon. Kurven viser pasientens kontinuerlige blodtrykk og medisinkurvene som plottes når det gis medisiner for å endre på blodtrykket.

Rapporten er strukturert etter retningslinjer for rapportskriving og er bygd opp som følger: innledning, teori, metode, resultat, diskusjon og konklusjon. De forskjellige punktene i design science metodens fremgangsmåte sorterer naturlig under de forskjellige kapitlene i rapporten.

3.1. Design Science

Design science er en forskningsmetode som kan benyttes om man ønsker å lage en ny løsning eller forbedre en eksisterende løsning. Definisjonen er «the scientific study and creation of artefacts as they are developed and used by people with the goal of solving practical problems of general interest» (52).

3.1.1. Et rammeverk for Design Science forskning

Det er to paradigmer som er fundamentale for forskning på informasjonssystemer, *behavioral science* og *design science*. Å kombinere disse gir muligheter for å utnytte de komplementære forskningsprosessene til disse to metodene, se figur 3.1 (53).

Behavioral science gjøres i forbindelse med evalueringen når man forsker på informasjonssystemer. Det anvendes for å forutse eller forklare fenomener i forbindelse med bruk av artefaktet. Kontrollere brukbarheten og observere virkningen det har på brukerne eller organisasjonen (53).

Man kan dele fremgangsmåten for et design science prosjekt inn i fem aktiviteter. Det er ikke vanlig for et prosjekt å fordype seg i alle fem fasene, da det fort kan bli veldig tid- og ressurskrevende. Ofte omtales noen kort, mens en går i dybden på andre (52).

Skal nå gå igjennom fremgangsmåten for dette design science prosjektet sett i sammenheng med de to paradigmene.

Forklare problemet

Man starter med å formulere problemet så presist som mulig og begrunne betydningen av å løse utfordringen (52).

Problemet i denne oppgaven er basert på litteratursøk og domenekunnskap. Domenekunnskap er kunnskap om miljøet som systemet skal brukes i (54,55).

Domenekunnskapen er i hovedsak basert på forfatterens egne erfaringer etter mange års klinisk arbeid, supplert med uformelle diskusjoner med kolleger. Denne fasen står i innledningskapitlet.

Fasen hører til *rigor cycle* i figur 3.1. Ved å utføre litteratursøk og bruke domenekunnskap benytter man seg av kunnskapsbasen. Denne kunnskapen tar man med seg inn i prosjektet (53). Til slutt når prosjektet er ferdig tilfører resultatet ny kunnskap til kunnskapsbasen.

Definere kravene

Under dette punktet er målet å skissere en mulig løsning på den isolerte problemstillingen og definere krav til artefaktet (52).

Å definere kravene er beskrevet i teorikapitlet, men det består ikke av en utfyllende kravspesifikasjon. Det vil være det ikke-funksjonelle kravet interoperabilitet som er fokuset i denne oppgaven, for å kunne lage et artefakt som oversetter eksisterende datastrøm til HL7 FHIR-standard.

Utvikling av et artefakt i fasen *utarbeide og utvikle programvare* for å oppfylle kravet om interoperabilitet vil være godt dekket.

Definisjon av kravene hører til under *Relevance cycle* i rammeverket, se figur 3.1. Man har et miljø bestående av mennesker, organisasjoner og teknologi som alle har sine utfordringer og begrensinger. Fra dette kommer ønsker og behov som man kan bruke til å definere kravene. Det ferdige produktet anvendes i det miljøet det ble utviklet for.

Utarbeide og utvikle programvare

I denne delen inngår programmeringen av artefaktet som skal adressere problemet og innfri de utarbeidede kravene (52).

Det hører til *design cycle* som er forsknings- og utviklingsprosessen av artefaktet, se figur 3.1. Man starter med utviklingen. Deretter demonstrerer man det foreløpige resultatet, evaluerer det og fortsetter å foredle det til man har et ferdig produkt som oppfyller kravene (53).

Denne fasen er beskrevet i metode og resultat. I metodekapitlet beskrives de teknologier som benyttes, hvordan de fungerer og hvorfor de er valgt til å løse denne oppgaven. I resultatkapitlet blir utviklingen av prototypen gjennomgått. Hvordan en i praksis har tenkt og hva en må ta hensyn til for å utvikle en relevant løsning.

Demonstrere programvare

Her viser man fram prototypen som er utviklet og brukbarheten med et fungerende konsept (52). Dette er en del av *design cycle* i figur 3.1 (53).

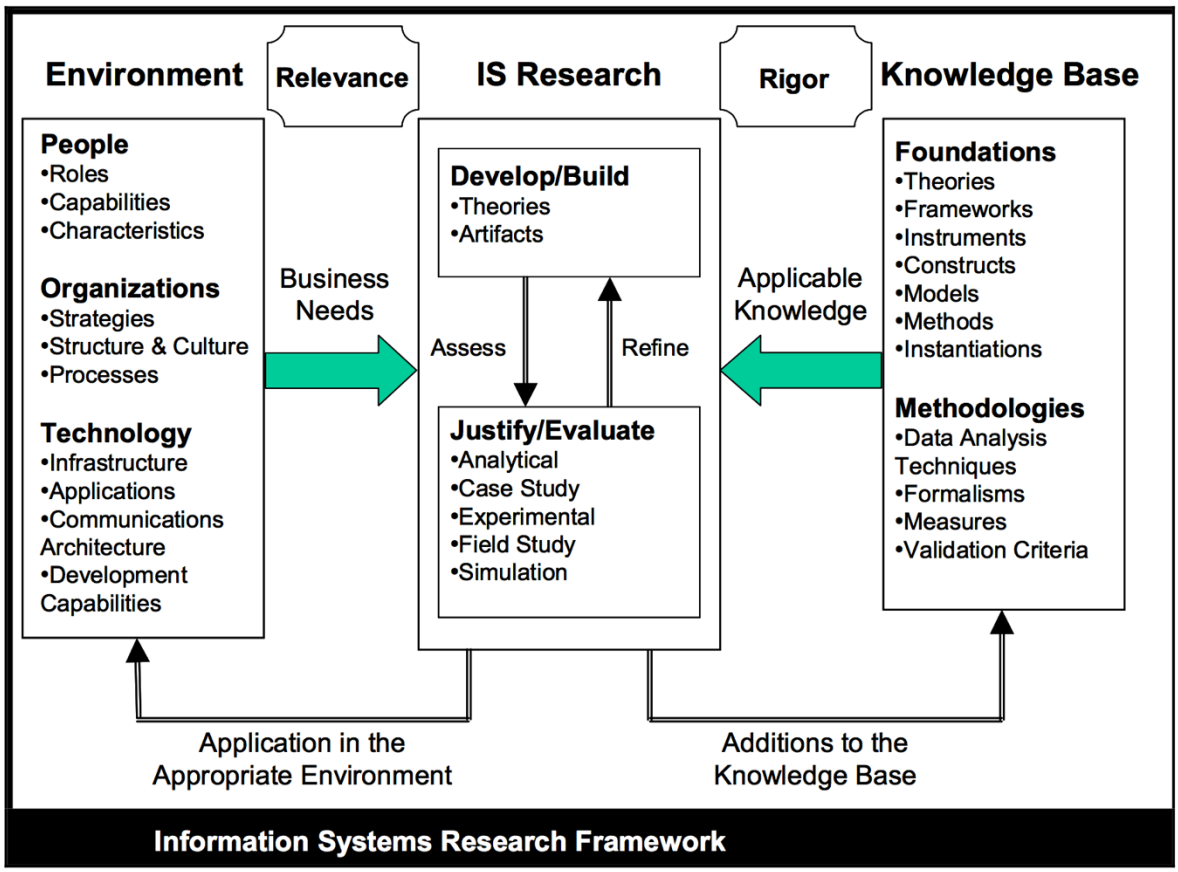
Evaluering av programvaren

I evalueringen utreder man hvor godt programvaren oppfyller kravene og i hvilket omfang den løser problemet som var utgangspunkt for prosjektet (52).

Evalueringen er også en del av *design cycle* i figur 3.1 (53).

Evalueringsfasen er beskrevet i diskusjonskapitlet. Det vil ikke være en fullstendig evaluering av løsningen. Det er to årsaker til det. Tiden og ressursene er begrensende faktorer i en slik oppgave. Og i figur 3.1 står det at man skal ha «application in appropriate environment». Det er ikke gjennomførbart. Ikke er det forsvarlig med den løsningen som er utviklet så langt. I tillegg er det strenge krav som må oppfylles for å kunne gjøre dette i klinikken.

Derfor blir det en vurdering basert på domenekunnskap og tilbakemeldingene når prosjektet presenteres for resten av klinikken. Vurderingen blir den største delen av diskusjonskapitlet.



Figur 3.1: Information Systems Research Framework.

llustrerer et rammeverk for å forstå, utføre og evaluere forskning på informasjonssystemer (53).

3.2. Konstruksjon av prototypen

For å bevise proof-of-concept er tanken å oversette to datastrømmer til ressurser i JSON-format i henhold til HL7 FHIR-standard og samtidig plotte informasjonen i integrerte kurver. Den integrerte kurven består av en kanal inn i form av en observasjon og to kanaler ut i form av dokumentasjon på medisinering med to sprøytepumper.

3.2.1. Sprøytepumpe

I denne oppgaven ble det benyttet to virtuelle sprøytepumper, se figur 4.3, som kilde til datastrømmen. En sprøytepumpe er medisinsk utstyr som brukes til å gi en kontinuerlig medisininfusjon direkte inn i pasientens blodsirkulasjon, se figur 3.2. Den kan brukes frittstående eller kobles til en rack sammen med flere sprøytepumper. Datastrømmen ble laget ved å koble en pumpe til et testoppsett for datahøsting inne på verkstedet til medisinsk teknisk avdeling. Pumpen ble satt opp og programmert til å tro at den var lastet med medisiner. Rådataene som ble lest av ble lagret til en tekstfil. Denne tekstfilen ble senere duplisert og redigert slik at en hadde to filer som hver inneholdt data om en type medisininfusjon. På denne måten fikk en to virtuelle sprøytepumper med hvert sitt medikament. Noe som forenklet arbeidet med artefaktet som skulle oversette datastrømmen og en unngikk et stort fysisk testoppsett.

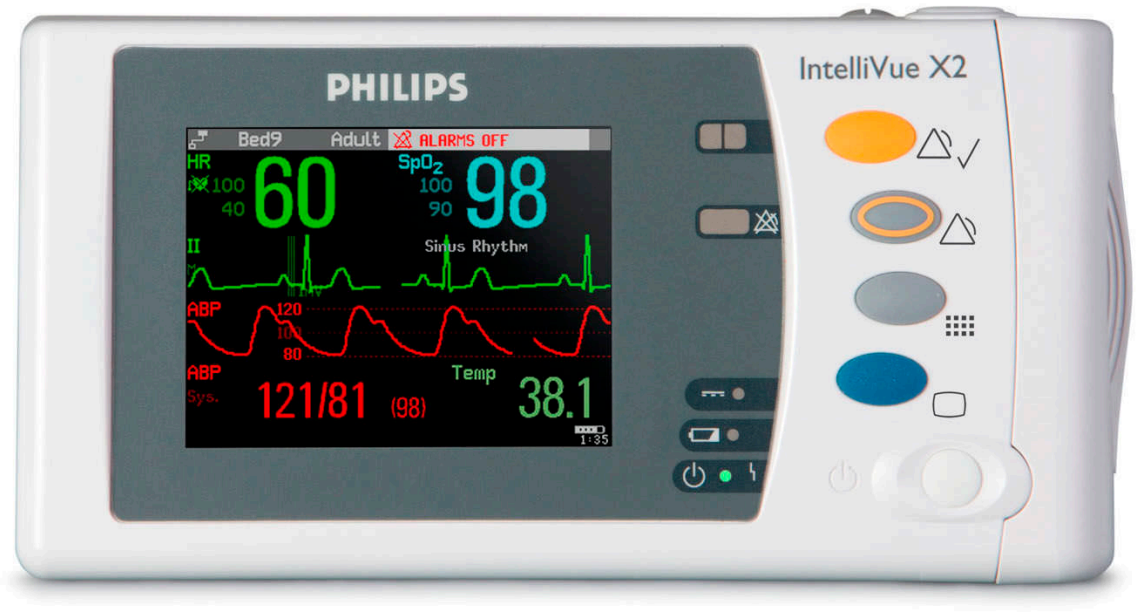
Figur 3.2: En Alaris CC med Guardrails sprøytepumpe.

(56)

Sprøytepumpen sender data i henhold til RS-232 protokollen, enten via infrarød sender (irDA) eller kabel. Konfigurasjonen er avhengig av modell, men var 38400 baud, 8 databit, 1 stopbit, ingen paritet for denne modellen. For å sette i gang datastrømmen fra pumpen benytter den hardware flow control (handshaking). Det betyr at man i tillegg til RxD (sender), TxD (mottaker) og jord bruker en leder for å sende en forespørsel om å sende data (RTS) og en leder for å gi beskjed om at den er klar for å motta data (CTS).

3.2.2. Pasientmonitor

En pasientmonitor, se figur 3.3, overvåker pasientens tilstand i sanntid. De ulike parameterne som overvåkes i dette scenarioet er systolisk blodtrykk, diastolisk blodtrykk, elektrokardiogram, sentralvenøst trykk, oksygenmetning, temperatur og puls. I utgangspunktet var tanken å sample eksempeldata med det samme testoppsettet som ble brukt på sprøytepumpen inne på verkstedet til medisinsk teknisk avdeling. Men det var ikke like lett. Problemet er at man ikke får lesbare data ut som en kan redigere og lagre til en tekstfil. Pasientmonitoren er forskjellig fra sprøytepumpen på den måten at den generer ikke data selv, den leser kliniske data fra pasienten. Derfor måtte man ha koblet en person til for å lage eksempeldata, noe som ikke er etisk forsvarlig. Løsningen ble å programmere en realistisk blodtrykkskurve i artefaktet.



Figur 3.3: Philips IntelliVue X2 pasientmonitor.

(57)

3.2.3. Datastrømmen

For å komme i gang var det nødvendig å få tilgang til en datastrøm fra medisinsk utstyr. For å gjøre eksperimentet så enkelt som mulig ble det valgt å bruke sprøytepumper til å generere en datastrøm. Fordelen er at de genererer sine egne data basert på brukerens innstillinger, derfor trenger man ikke å koble til en pasient for å få ut en realistisk datastrøm.

En sample fra en sprøytepumpe består av to linjer med tekst. På første linje står tidsstempelet. På andre linje står alle variablene adskilt med en ^. Sprøytepumpene sender ut et signal ca. hvert 1,2s. Under vises et eksempel på en sample:

```
{bh}{27/03/2017,14:25:42.864}
!INF^8003-35698^-^SET^4.20^ml/h^Adrenalin^0.073^ml^-2.23^mmHg
^06:19:30^EVENT^36053|A235
```

3.2.4. Kravspesifikasjon

Utgangspunktet er at man ikke er fornøyd med dagens løsning eller det mangler en løsning. Man ønsker en endring i fra dagens praksis og hvilke ønsker man har til den nye løsningen.

Det skilles mellom to typer krav. Det første er kundens krav, ønsker og forventninger til den nye løsningen. Kravene fra kunden som bestiller den nye løsningen utformes i et dokument med et naturlig språk. Den andre typen krav er systemkrav. Det er et mer detaljert teknisk dokument som beskriver eksakt hva som skal implementeres. Dette utarbeides av utviklerne som skal lage den nye løsningen i samarbeid med kunden.

Systemkravet kan deles inn i to grupper. De funksjonelle kravene og de ikke-funksjonelle kravene. De funksjonelle kravene sier noe om det nye systemets indre. Hvilke tjenester det skal tilby, hvordan det skal håndtere inndata og hvordan det skal reagere i bestemte situasjoner.

De ikke-funksjonelle kravene legger føringer for systemet som helhet og ikke hver enkelt funksjon det skal inneholde. Eksempeler er krav til hvor sikkert systemet skal være eller krav til oppetid (33).

Krav knyttet til problemstillingen

Utgangspunktet for oppgaven var problemet med å høste informasjon automatisk på et standardisert format for å integrere data og skape interoperabilitet mellom systemer, personer, avdelinger og institusjoner.

Fra dette kommer det naturlig noen krav:

- Interoperabilitet
- Bedre datakvalitet
- Lettere å gjenbruke informasjonen

I denne oppgaven er det kun fokusert på det ikke-funksjonelle kravet interoperabilitet som står sentralt for å løse problemet beskrevet i problemstillingen.

4. Resultat

Dette kapitlet er delt i to deler. Først blir det en gjennomgang av prototypens oppbygning og virkemåte. Deretter en beskrivelse av simuleringen og sluttproduktet.

4.1. Scenarioet

Å lese av datastrømmen fra sprøytepumper når man administrer medisiner til pasienten er bare en liten del av den store sammenhengen. På bakgrunn av konteksten setter man ønskede grenseverdier for å sikre optimal behandling av pasienten. Scenarioet som skisseres her er på bakgrunn av en hjerteoperasjon hvor det er viktig med god regulering av blodtrykket under og den første tiden etter operasjonen for at ikke anostomosene[3] skal løsne og begynne å blø. Intensivpasienter er bemannet med en til to spesialsykepleiere som er til stede inne på rommet hele tiden og kontinuerlig overvåker tilstanden. Trykket vises på en overvåkningsskjerm hvor grenseverdiene er lagt inn. Den varsler hvis trykket endrer seg ut over innstilte verdier. Når dette blir observert av personalet administrer de, avhengig av om trykket er høyt eller lavt, riktig medisin etter gjeldende prosedyre. Mens tiltaket gjennomføres følger de med på om det har ønsket effekt. Denne overvåkningen og justeringen foregår kontinuerlig så lenge det er nødvendig. Figur 4.1 viser scenarioet ved høyt blodtrykk og figur 4.2 ved lavt blodtrykk.

Vanligvis vil man illustrere prosedyrene ved hjelp av use-case. Disse brukes for å beskrive en bestemt arbeidsflyt før den implementeres av utviklerne. Denne oppgaven har fokus på et lite område av den ferdige programvaren. Derfor er det uhensiktsmessig å lage use-case fordi samhandlingen vil skje etter at datastrømmen er parset over på HL7-standarderen. Use-case danner også grunnlaget for bruk av *scenario builder* som videre danner grunnlaget for profiler.

Scenario builder

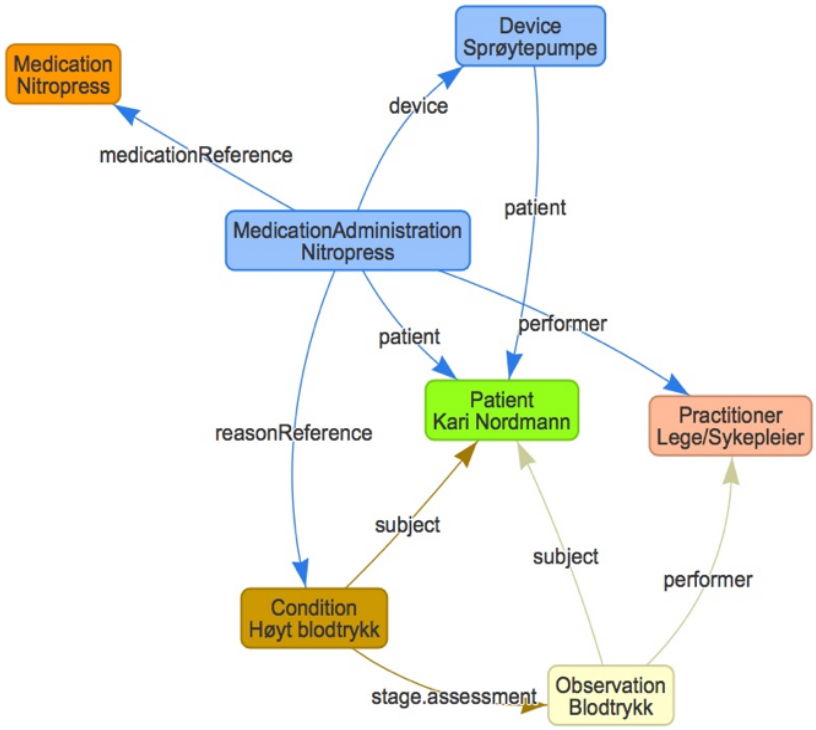
Her brukes *scenario builder* for å gi en oversikt over hvordan denne oppgaven henger sammen med det store bildet. I og med at prosedyren er å regulere blodtrykket opp og ned vil de to skjemaene på figur 4.1 og 4.2 være veldig begrenset.

Skjemaene viser hvordan de forskjellige ressursene som er i bruk dokumenterer kliniske data når personalet observerer en pasients blodtrykk. Man observerer at tilstanden enten er høy eller lav og at dette krever et tiltak. Ressursen `medicationAdministration` dokumenterer medisineringen av en pasient som gjøres med en sprøytepumpe. Det er denne ressursen som brukes i artefaktet som et proof-of-concept på om det er mulig å standardisere og lagre kliniske data for å bedre datanøyaktigheten, datakvaliteten, gjøre informasjonen søkbar og lettere tilgjengelig for gjenbruk. I tillegg til å lage en integrert visualisering av behandlingen.

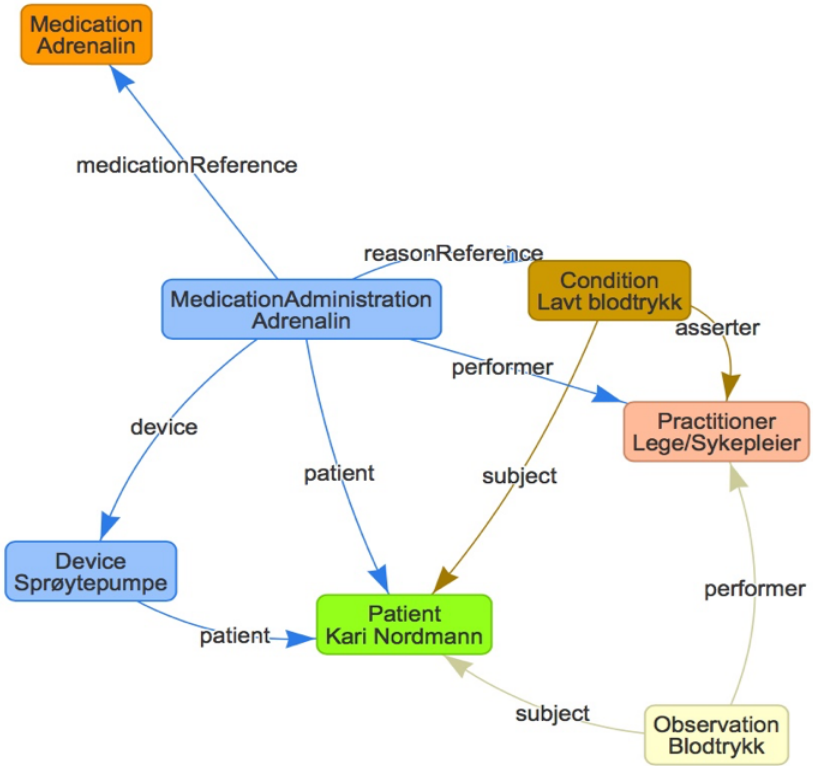
4.1.1. Modellering

Figurene 4.1 og 4.2 viser hvordan datastrømmen fra administrasjon av blodtrykksmedisinene henger sammen med helsepersonellets observasjon av pasientens blodtrykk, beslutningen som tas og det medisinske utstyret som brukes.

[3] Sammenføyningen mellom to blodkar.



Figur 4.1: Scenarioet ved senkning av blodtrykket.



Figur 4.2: Scenarioet ved økning av blodtrykket.

4.1.2. Oppbygningen av datastrømmen

```
{bh}{27/03/2017,14:25:42.864}
!INF^8003-35698^-^SET^4.20^ml/h^Noradrenalin^0.073^ml^-
2.23^mmHg^06:19:30^EVENT^36053|A235
```

Over vises ett datasample i sin helhet. Her kommer en gjennomgang av datastrømmen som skal oversettes og lagres i ressursen. Det var ikke mulig å finne informasjon om datastrømmen i bruker- eller servicemanualen. Derfor er kildene til denne informasjonen ansatte ved medisinsk teknisk avdeling og anestesi- og intensivklinikken. Hvis informasjonen blir tatt vare på av artefaktet står det beskrevet i hvilket objekt den lagres.

`{27/03/2017,14:25:42.864}`

Dette tidsstempelet er oppgitt med et millisekunds nøyaktighet og kommer på en egen linje. På neste linje kommer selve datasettet. Tidspunktet for start, endring og stopp lagres i objektet `medicationAdministration.effectivePeriod`.

`INF`

Sier noe om hvilken type datastrøm det er. I dette tilfellet en infusjon.

`8003-35698`

Er serienummeret på infusjonspumpen. Det brukes som en unik identifikator på pumpen. Denne informasjonen lagres i objektet `medicationAdministration.device`.

`SET`

Er status på infusjonen. Det betyr at riktig dose er satt, bekreftet og at infusjonen er i gang. Alternativene er `TIT`, som betyr titrerer, sier at noen gjør endringer på dosen som ikke er bekreftet ennå. `BOL` som indikerer at det gis en bolus, som er en stor dose på veldig kort tid. Og `HLD` som sendes ut når pumpen står i pause.

`4.20`

Er dosen som gis i øyeblikket. På pasienter som opereres eller overvåkes på intensiv endres dosen kontinuerlig på bakgrunn av tilstanden. Når man begynner å endre på dosen skifter status på infusjonspumpen fra `SET` til `TIT`. Med en gang man bekrefter endringen endres status tilbake til `SET`. Dosen lagres i objektet `medicationAdministration.dosage.rateQuantity`.

`ml/h`

Beskriver enheten på dosen, i dette tilfellet ml i timen. Andre eksempler på vanlige enheter er `mg/kg/h`, milligram per kilo per time, `U/h`, enheter per time og `µg/kg/h`, mikrogram per kilo per time.

`Noradrenalin`

Er navnet på medikamentet som gis. Siden proof-of-concept er å registrere blodtrykk i dette scenarioet er det noradrenalin og nitropress som er aktuelle medikamenter. Noradrenalin øker blodtrykket ved å minske diameteren på blodårene slik at motstanden øker. Og nitropress senker blodtrykket med å øke diameteren på blodårene. Informasjon om medikamentet lagres i objektet `medicationAdministration.medication`.

`0.073`

I dette feltet står den totale mengden som er gitt av medikamentet. Mengden inkrementerer for hver sample.

`ml`

Angir enheten for den totale mengden.

`-2.23`

Måler trykket i slangen medikamentet går og gir alarm hvis det blir for høyt. Dette er en sik-

kerhetsfunksjon hvis f.eks. slangen går tett eller at kanylen ikke ligger korrekt plassert i pasientens blodåre. Da er det en indikasjon på at medikamentet ikke flyter fritt og pumpen skal varsle, eventuelt stoppe slik at ikke pasienten blir skadet. Får man alarm må personalet feilsøke slik at man får startet medisineringen igjen. Fordi sprøytepumpen ikke var koblet til en pasient, var det ikke mulig med trykkmåling i dette forsøket.

`mmHg`
Er måleenheten pumpen bruker for trykk.

`06:19:30`
Er feltet som viser varigheten på medisininfusjonen.

`EVENT`
Er tittelen til neste felt som inneholder hendelsesnummeret.

`36053|A235`
Er hendelsesnummeret. Det er todelt. 36053 er nummeret for akkurat denne dosen og endrer seg først når en endrer og bekrefter innstillingen på dosen til medikamentet. `A235` er nummeret for akkurat dette ene samplet og inkrementeres for hver linje som kommer. Den første delen av hendelsesnummeret `36053` lagres i objektet `medicationAdministration.identifier`. Og brukes i tillegg av artefaktet til å registrere endringer i dosen for å starte å dokumentere i ny fil.

4.2. Utvikling av prototypen

Utgangspunktet var den allerede eksisterende datastrømmen inne på en operasjonsstue og behovet for å sammenfatte disse. Det kan skape grunnlag for bedre kvalitet på informasjonen som blir lagret under et sykehusopphold. Etter å ha lest om interoperabilitet og tenkt på hvordan den nye HL7- standarden FHIR kunne benyttes i en mulig løsning, ble det besluttet å lage en proof-of-concept ved å integrere datastrømmene fra flere sprøytepumper som danner datagrunnlaget for journalen og sanntidskurver.

Prototypen tar en sample med rådata, splitter den opp og lagrer den i en liste. Innholdet i listen kan da enkelt lagres under tilhørende objekt i JSON-skjemaet. JSON-skjemaet lagres kontinuerlig for ikke å miste informasjon ved en eventuell strømstans eller nedetid av andre årsaker. Videre sender artefaktet dataene til et verktøy for tegning av grafer. Der integreres datastrømmene til grafer i samme koordinatsystem.

I prototypen er det kun brukt de objektene som må være med i henhold til innhold i datastrømmen med utgangspunkt i malen til `medicationAdministration`. Resultatet som blir lagret til filen og arkiveres i journalen inneholder derfor bare det minste antall objekter som må være med.

For å kunne demonstrere proof-of-concept måtte man først ha et eksempel på en datastrøm fra medisinsk utstyr. Man valgte å bruke en sprøytepumpe. Årsaken er at de generer sine egne data og er derfor ikke avhengig av å kobles til en pasient for å lage en realistisk datastrøm. Derfor trenger man ikke godkjenning fra Regional Etisk Komitè fordi man kan lage en ekte datastrøm med fiktive data.

4.2.1. Hvordan er problemstillingen adressert

For å lage artefaktet som skal adressere problemet og vise at konseptet fungerer må skriptet bygges opp av flere funksjoner. Her følger en gjennomgang av hvordan artefaktet er konstruert.

4.2.1.1. Virtuelle pumper

Man kobler seg fysisk til pumpen(e) med enten kabel eller infrarød sender og mottaker. Når man starter en medisininfusjon vil pumpen etter å ha fått klarsignal fra mottakeren sende en sample med ca. 1,2s mellomrom. Dette krever at en rigger seg til med utstyr for overføring av data. For å gjøre arbeidet med utviklingen av skriptet lettere ble det, med hjelp fra medisinsk teknisk avdeling og deres testoppsett, lastet ned et eksempel på en datastrøm til en tekstfil. Se figur 4.3.

Nitropress.txt

```
{bh}{27/03/2017,14:25:42.864}
!INF^8003-36000^-^SET^7.40^ml/h^Nitropress^0.073^ml^-2.23^mmHg^06:19:30^EVENT^36063|A235

{be}

{bh}{27/03/2017,14:25:43.983}
!INF^8003-36000^-^SET^7.40^ml/h^Nitropress^0.075^ml^-2.23^mmHg^06:19:29^EVENT^36063|60EB

{be}

{bh}{27/03/2017,14:25:45.204}
!INF^8003-36000^-^SET^7.40^ml/h^Nitropress^0.076^ml^-2.23^mmHg^06:19:28^EVENT^36063|5E4B

{be}

{bh}{27/03/2017,14:25:46.389}
!INF^8003-36000^-^SET^7.40^ml/h^Nitropress^0.077^ml^-2.23^mmHg^06:19:27^EVENT^36063|B658
```

Noradrenalin.txt

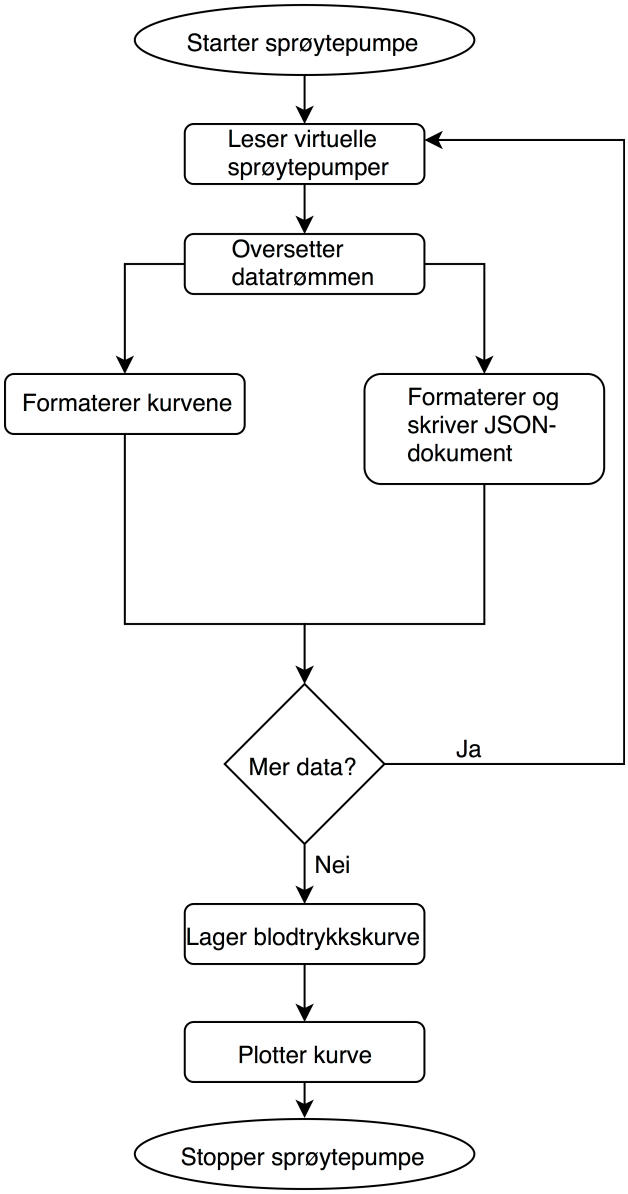
```
{bh}{27/03/2017,14:25:42.864}
!INF^8003-35698^-^SET^4.20^ml/h^Noradrenalin^0.073^ml^-2.23^mmHg^06:19:30^EVENT^36053|A235

{be}

{bh}{27/03/2017,14:25:43.983}
!INF^8003-35698^-^SET^4.20^ml/h^Noradrenalin^0.075^ml^-2.23^mmHg^06:19:29^EVENT^36053|60EB

{be}

{bh}{27/03/2017,14:25:45.204}
!INF^8003-35698^-^SET^4.20^ml/h^Noradrenalin^0.076^ml^-2.23^mmHg^06:19:28^EVENT^36053|5E4B

{be}

{bh}{27/03/2017,14:25:46.389}
!INF^8003-35698^-^SET^4.20^ml/h^Noradrenalin^0.077^ml^-2.23^mmHg^06:19:27^EVENT^36053|B658

{be}

{bh}{27/03/2017,14:25:47.581}
!INF^8003-35698^-^SET^4.20^ml/h^Noradrenalin^0.079^ml^-2.23^mmHg^06:19:26^EVENT^36053|5E94
```

Figur 4.3: Skjermdump av de to virtuelle sprøytepumpene.

Den lagrede datastrømmen ble kopiert i to eksemplarer. Så ble de redigert slik at de inneholdt navn, dosering og tidspunkt for infusjon av to forskjellige medikamenter som brukes til å senke og heve blodtrykket. De to tekstfilene med de forskjellige datastrømmene representerer hver sin virtuelle sprøytepumpe.

Fordelen med å gjøre det slik er at en slipper å ha det aktuelle medisinske utstyret tilgjengelig og være avhengig av et stort fysisk testoppsett med potensielle feilkilder. Selv om alt foregår på datamaskinen vil simuleringen være lik bruken i klinisk praksis. Andre fordeler er at man enkelt kan jobbe i team og er ikke avhengig en fysisk arbeidsplass hvis en skulle gjort dette som en fullskala prosjekt.

4.2.1.2. Skjematisk fremstilling



Figur 4.4: Oversikt over artefaktets funksjoner.

I prototypen er fokuset på datastrømmene knyttet til overvåkning og justering av blodtrykket. Det er simulert en blodtrykkskurve og så er datastrømmen fra sprøytepumpene som brukes til å regulere trykket i henhold til observasjon lest av, oversatt, standardisert og lagret.

For å vise hvordan man ser for seg en mulig løsning på programvaren ble det laget et flytskjema som viser de ulike stegene i artefaktet, se figur 4.4. Dette viser hvordan én datastrøm håndteres av systemet.

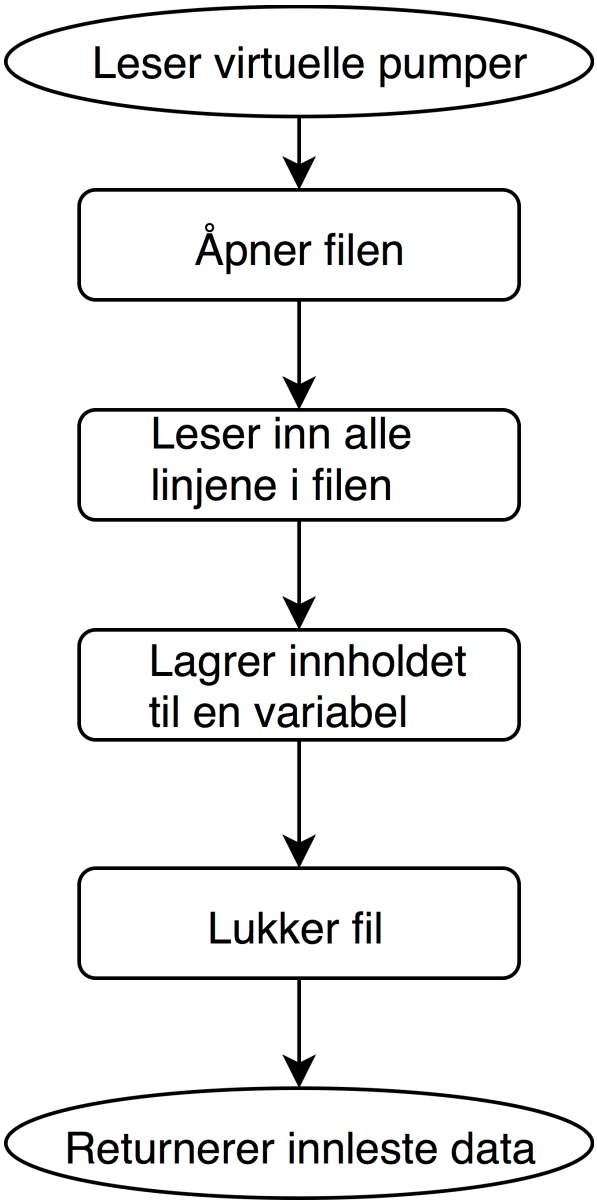
Når skriptet starter leser første funksjonen datastrømmen. Så oversettes og lagres datastrømmen i lister. Listene danner grunnlaget for de elektroniske kurvene og informasjonen som skal inn i journalnotatet. Dette gjøres for hver linje med data som kommer. Så hentes blodtrykkskurven inn og kurvene plottes til slutt.

Informasjonen som lagres i ressursen lagres kun når medisineringen pågår. Artefaktet lagrer kontinuerlig data i pasientjournalen. Disse kan brukes til å tegne en graf over tidspunkt og dosen som ble gitt både i sanntid og retrospektivt.

4.2.1.3. Gjennomgang av hver enkelt funksjon

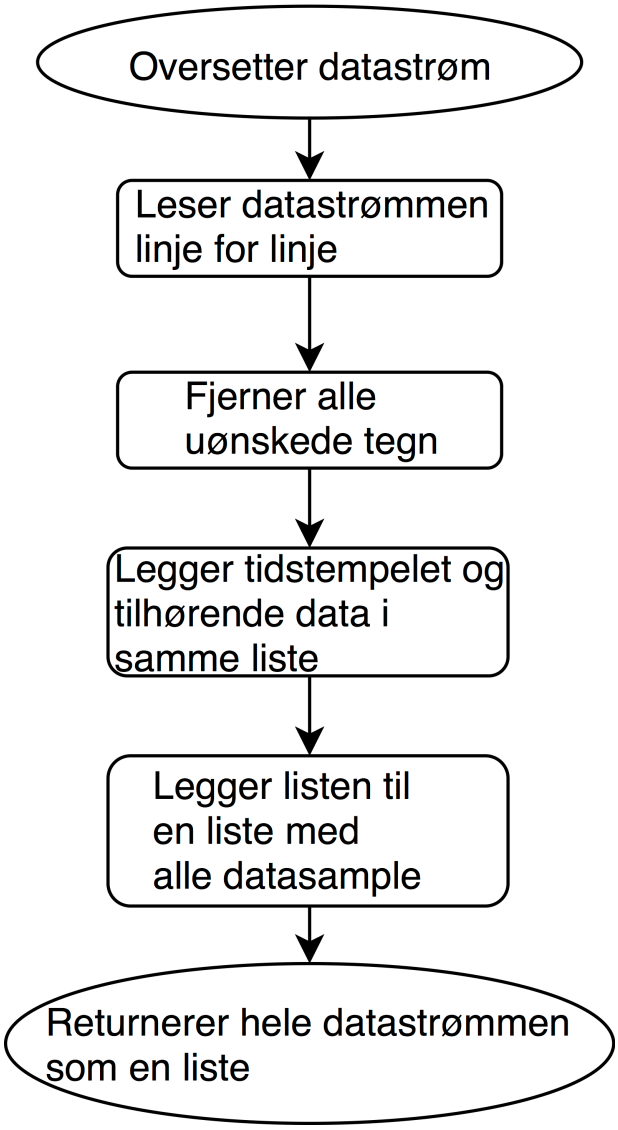
Med utgangspunkt i figur 4.4 følger en gjennomgang som beskriver kildekoden funksjon for funksjon.

Leser virtuelle pumper



Figur 4.5: Leser data fra virtuelle pumper.

I stedet for å motta et signal inn på en port på datamaskinen blir datastrømmen lest fra en fil. Funksjonen som leser fra den virtuelle sprøytepumpen heter `lesFil()`. Figur 4.5 viser flytskjema for funksjonen. Dataoverføringen starter med å kalle på funksjonen og har tekstfilen som inneholder datastrømmen som argument.



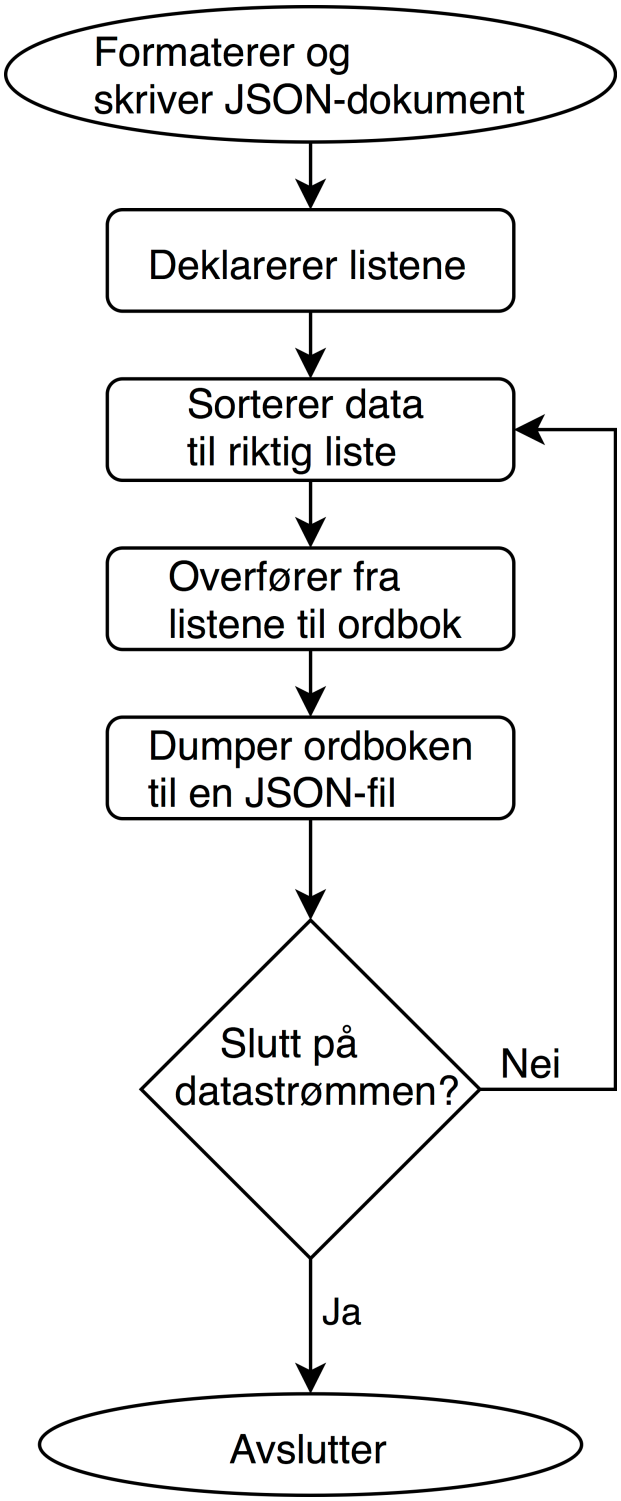
Figur 4.6: Oversetter datastrømmen.

Etter at datastrømmen er lest inn fra en tekstfil sendes den til en funksjon om oversetter datastrømmen, se figur 4.6, og lagrer den til en liste, se figur 4.7. Grunnen er å strukturere innholdet og gjøre det lett tilgjengelig ved å kunne hente ut enkeltverdier fra listen og skrive det til en JSON-fil som formateres i henhold til HL7 FHIR-malen som brukes. Samtidig er det enkelt å sende enkeltverdier til funksjonen som plotter kurven. Det starter med at man kaller på funksjonen og bruker innholdet fra filen som ble lest inn som argument.

```
[['27/03/2017,14:25:42.864', 'INF', '8003-36000', '-', 'SET', '7.40', 'ml/h', 'Nitropress', '0.073', 'ml', '-2.23', 'mmHg', '06:19:30', 'EVENT', '36063lA235'],
['27/03/2017,14:25:43.983', 'INF', '8003-36000', '-', 'SET', '7.40', 'ml/h', 'Nitropress', '0.075', 'ml', '-2.23', 'mmHg', '06:19:29', 'EVENT', '36063l60EB'],
['27/03/2017,14:25:45.204', 'INF', '8003-36000', '-', 'SET', '7.40', 'ml/h', 'Nitropress', '0.076', 'ml', '-2.23', 'mmHg', '06:19:28', 'EVENT', '36063l5E4B']]
```

Figur 4.7: Viser en liste med tre datasampler lagret i en liste.

Skriver til JSON og lagrer fil

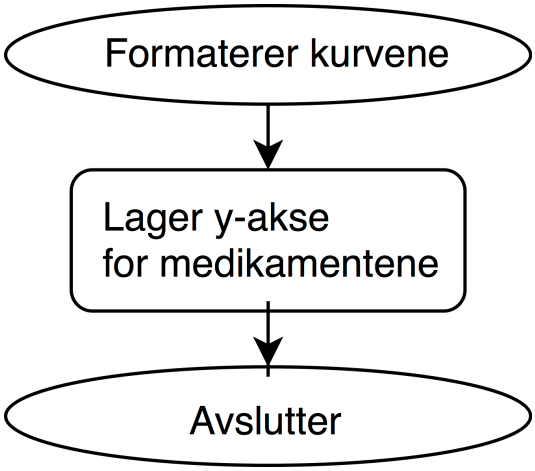


Figur 4.8: Skriver datastrømmen til JSON.

Etter at datastrømmen er lagret i en liste, sendes den videre til en funksjon som henter ut ønskede data og lagrer det til en dictionary. Innholdet i dictionary dumpes deretter til en JSON-fil som lagres i journalen.

Formaterer kurvene

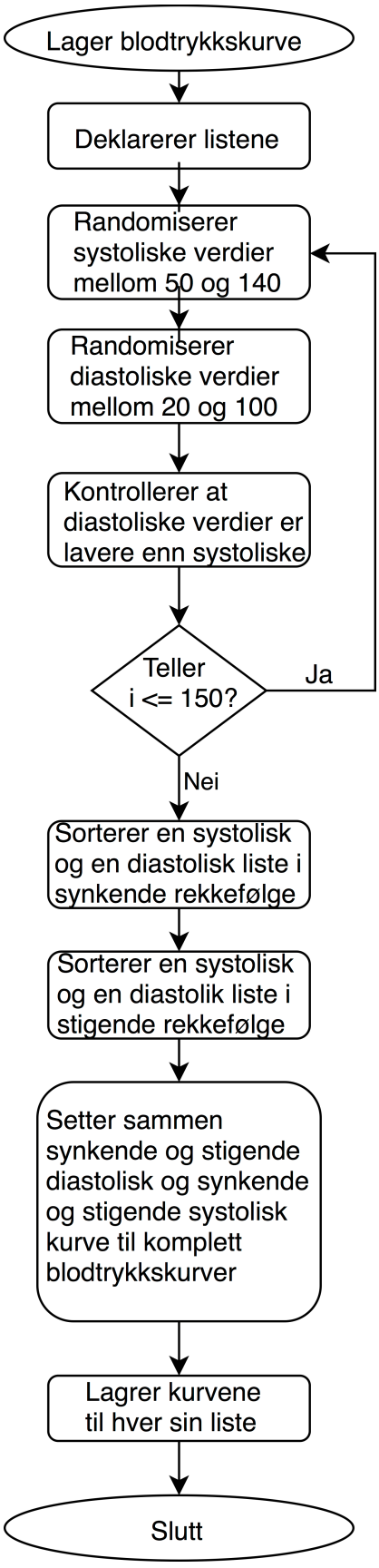
Før man kan kalle på funksjonen som plotter kurvene må kurvene formateres, se figur 4.9.



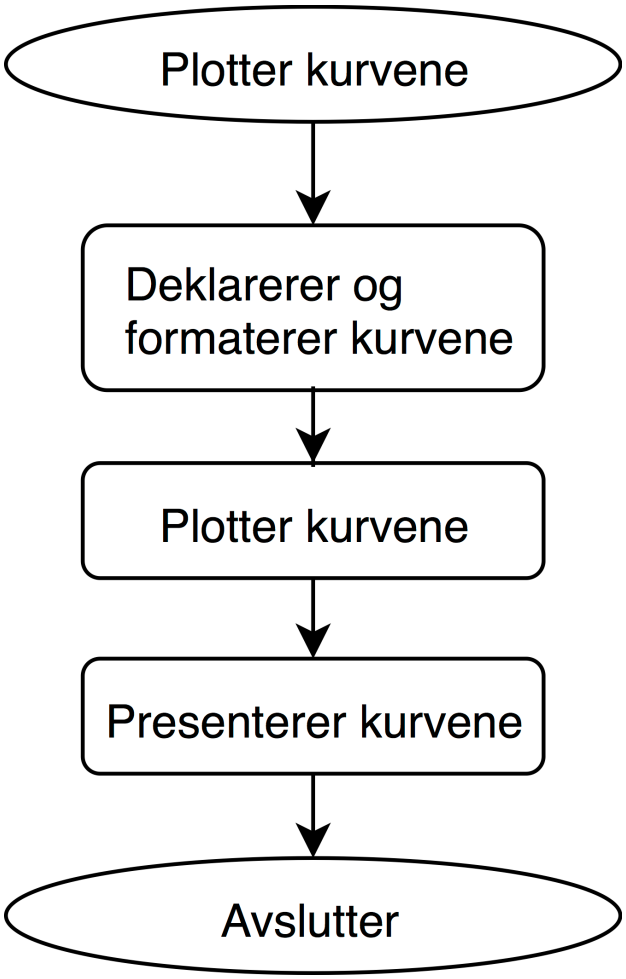
Figur 4.9: Formaterer y-verdiene.

Lager blodtrykkskurve

Fordi det var urealistisk å kopiere en datastrøm fra en pasientmonitor, måtte det programmeres en funksjon som lager en fiktiv, men realistisk blodtrykkskurve. Det starter med systolisk blodtrykk på 140mmHg som er høyt i denne sammenhengen. Deretter faller trykket når det gis nitropress, en medisin som utvider diameteren på blodårene slik at motstanden synker og trykket blir lavere. Det som simuleres fysiologisk er at blodtrykket forsetter å synke selv om man trapper ned på medisindosen og til slutt stopper infusjonen. Da er blodtrykket blitt lavt og pasienten må ha medisiner for å øke blodtrykket. Så startes det en infusjon med noradrenalin for å minske diameteren på blodårene og øke motstanden slik at trykket stiger. Blodtrykkskurven ble ikke laget som en egen funksjon fordi en funksjon i Python ikke kan returnere to variabler. Den ligger i samme funksjon som plottingen av kurvene og blodtrykksverdiene legges rett inn i en kurve for systolisk trykk og en kurve for diastolisk trykk. Se figur 4.10.



Figur 4.10: Lager blodtrykkskurve.



Figur 4.11: Flytskjema for plotting av kurvene.

Variabelen som inneholder datastrømmen blir til slutt brukt til å integrere blodtrykkskurven og medisinkurvene i en elektronisk kurve. Fordi det er disse kurvene, som oppdateres i sanntid, personalet er interessert i når de overvåker og behandler en pasient. De må være oversiktlige og gi god informasjon om tilstanden bare ved å kaste et hurtig blikk bort på skjermen.

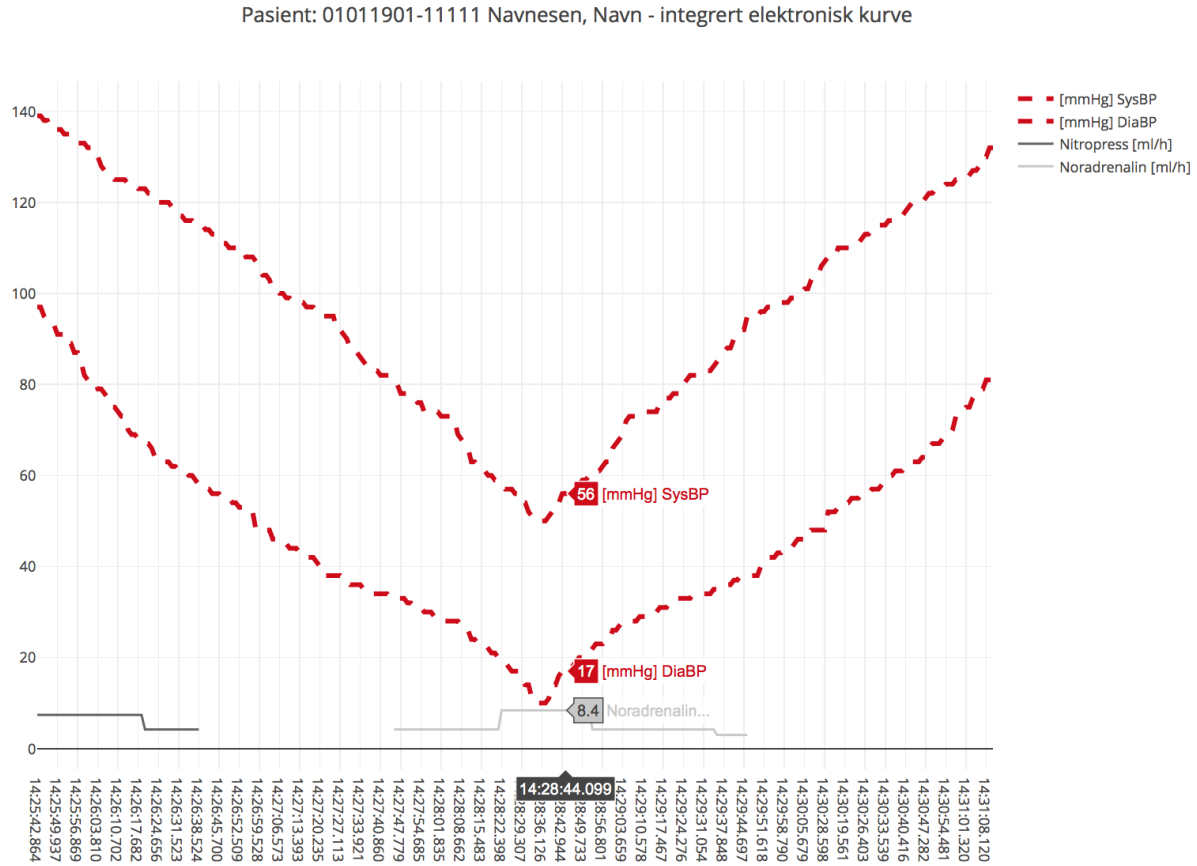
Resultatet er en integrert visualisering av datastrømmene og gevinsten av å sammenfatte disse. Det viser at proof-of-concept fungerer og at problemstillingen lar seg løse.

4.3. Simuleringen

Scenarioet som simuleres er overvåkning av pasientens blodtrykk under eller tiden like etter en hjerteoperasjon. I et slikt scenario er det viktig med god kontroll på blodtrykket for å unngå blødninger ved at anastomosene løsner før de har grodd fast. Høsting av informasjon når det gis medisiner for å endre på trykket, enten det er for høyt eller for lavt, gir mulighet for god oversikt over tilstanden. Kurvene som plottes gir en integrert presentasjon av hvordan de ulike datastrømmene henger sammen.

Man leser av den røde blodtrykkskurven som viser både det systoliske og diastoliske blodtrykket. Ut i fra gjeldende prosedyre, som varierer med kontekst, har man grenseverdier som sier når man skal iverksette tiltak for å endre på blodtrykket. Når man starter blodtrykksmedisinene leser systemet datastrømmen som inneholder blant annet dose og tidspunkt for medisineringen.

4.3.1. Resultatet av plottingen



Figur 4.12: Integrert presentasjon av overvåknings- og medisineringskurve.

Denne grafen svarer også på problemstillingen ved at man standardiserer datastrømmene og bruker informasjonen som grunnlag for integrert presentasjon av sanntidsdata. Da unngår man fragmentering av informasjonen. Man vil kunne visualisere både overvåkningsdata og data fra behandlingen i samme skjermbilde og skaffe seg forløpende oversikt over behandlingen som gis samtidig som dokumentasjonen lagres i journalen. På denne måten bevarer man også konteksten.

Figur 4.12 viser hvordan man kan kombinere datastrømmen fra overvåkningen av blodtrykket med datastrømmen fra to sprøytepumper med medisiner som brukes til å regulere blodtrykket.

Dette sikrer enkelt oversikt over pasientens nåværende tilstand, hvilken behandling som gis og hvordan tilstanden utvikler seg. Under en hjerteoperasjon og oppholdet på intensivavdelingen i timene etterpå er det viktig med kontinuerlig og helhetlig oversikt fordi tilstanden kan endre seg fra minutt til minutt.

Det gir ingen mening å samle inn alle datastrømmer hvis de ikke kan integreres. Ved å integrere foredler man informasjonen og den gir merverdi.

4.3.2. Datastrømmen som lagres i JSON

For å dokumentere behandlingen blir medisineringen dokumentert i en JSON-fil som legges i journalen.

Siden målet her er å vise at konseptet fungerer er ikke malen brukt i sin helhet, men kun de objekter som passer med innholdet i datastrømmen for å dokumentere medisinseringen. Objektene blir fylt med informasjon fra datastrømmen som kommer fra de virtuelle pumpene.

Samtidig som det lagres en JSON-fil blir dose og tidspunkt fra hver sample brukt til å tegne en kurve som presenteres sammen med blodtrykkskurven. Dette er bare en demonstrasjon på anvendelse av informasjonen som kommer ut av artefaktet. Samt et eksempel på hvordan man kan kombinere inngang og utgang i samme skjermbilde for enkelt å skaffe seg oversikt over tilstand og behandling.

Datastrømmen lagres til en JSON-fil som arkiveres kontinuerlig i journalen så lenge medisineringen pågår. Dette sikrer at de data som allerede er samlet inn ikke forsvinner hvis det blir problemer med dataoverføringen ved f.eks. strømbrudd.

All dokumentasjon på objektene og datatypene som beskrives i delkapitlene under er hentet fra (39).

4.3.2.1. Objektet medicationAdministration.status

Så lenge infusjonen går er status satt til ”in-progress”. Hvis man setter en infusjon på pause og en regner med at den skal startes opp igjen settes status til ”on-hold”. Dette er ikke uvanlig på operasjonsstuen hvor man forordner en pakke med medikamenter basert på type kirurgi og forventet forløp med utgangspunkt i morbiditet og mortalitet. Prosedyrene er basert på erfaring og litteratur, tilstandene endrer seg fort og derfor har man de nødvendige medikamentene i beredskap. Når medisineringen er fullført settes status til ”completed”. Det er totalt seks statuskoder. Hvis en medisin blir lagt inn ved en feil brukes ”entered-in-error”. Stopper man infusjonen før den er fullført skal status endres til ”stopped” og hvis man ikke vet hvilken av de andre statusene som skal brukes settes den til ”unknown”.

4.3.2.2. Objektet medicationAdministration.medication

Her beskrives hvilken medisin som ble gitt. Man kan skrive inn medisinkoden eller referere til medisinens ressurs. Den inneholder identifikasjon og definisjon på medisinen. I dette artefaktet benyttes medisinnavnet fra datastrømmen for enkelhets skyld.

4.3.2.3. Objektet medicationAdministration.effectivePeriod

Dette objektet sier i hvilken tidsperiode medisinen er gitt og inneholder start-/endrings- og sluttidspunkt. Ressursen må inneholde minst en og bare en periode i henhold til kardinalitet. Dette gjør at man ikke kan legge inn tidspunktet for hvert enkelt datasample, noe som ville gitt en fullstendig x-akse hvis man ønsker å rekonstruere medisinkurven.

4.3.2.4. Objektet medicationAdministration.identifier

Beskriver en numerisk eller alfanumerisk verdi som er assosiert med et objekt eller entitet i et gitt system. Her brukes samme identifikasjonsnummer for den aktuelle hendelsen som sprøytepumpen benytter. Sprøytepumpen endrer hendelsesnummer når det gjøres endringer i f.eks. dose på pumpen. Den gir også ut et unikt identifikasjonsnummer for hver enkelt sample som sendes ut. Slik at en kan knytte sammen både hendelse, tidspunkt, total mengde og dose som ble gitt for å skaffe god oversikt over forløpet. Det er ikke åpning for det i dagens ontologi.

4.3.2.5. Objektet medicationAdministration.dosage.rateQuantity

Dette objektet inneholder hastigheten på medisinen som gis til pasienten. Infusjonsraten hentes direkte fra datastrømmen og oppgis i denne oppgaven uendret på formatet 7.40ml/h for enkelhets skyld.

4.3.2.6. Objektet medicationAdministration.device

Her beskrives utstyret som ble brukt i administreringen av medisinen. Det vil vanligvis være en kobling til ressursen som inneholder detaljert informasjon om utstyret. I dette artefaktet brukes serienummeret fra datastrømmen.

4.3.2.7. Hvordan blir medisineringen dokumentert

Medisineringen for å senke blodtrykket blir lagret i en JSON-fil og arkivert i pasientens journal. De objektene som er tatt med i dette proof-of-concept er med fordi det omfatter de data fra sprøytepumpen som må være med for å dokumentere behandlingen.

4.3.2.8. Journalnotatene

Under kommer utskrift av alle journalnotatene artefaktet lager mens medisineringen pågår. De første dokumentene som lages er når blodtrykket er høyt og må senkes. Hver gang dosen endres starter artefaktet å skrive til ny fil. Når medisineringen er ferdig endres status til `completed` i filen når siste datasample er mottatt. Hvis datahøstingen avbrytes før medisineringen er ferdig blir status stående som `in-progress`.

Dokumentasjon på senking av blodtrykket.

Første filen gir nitropress med en dose på 7.40ml/h fordi trykket er høyt. Når trykket faller trappes dosen først ned til 4.20ml/h. Da avsluttes skrivingen til den første filen og artefaktet fortsetter dokmentasjonen i en ny fil. Når medisineringen avsluttes endres status til `completed` og artefaktet avsluttes. Når medisineringen inneholder flere doser vil det skrives en ny fil hver gang dosen endres. Derfor inneholder objektet `medicationAdministration.effectivePeriod` bare starttid i første fil og i de filene som lages når dosen endres. Da kan man se i ettertid når det ble gjort endringer. Sluttiden kommer i siste fil (58).


```
{
    "MedicationAdministration":                                    [
        {
            "status":                                  "in-progress",
            "medication":                               "Nitropress",
            "effectivePeriod":                                      [
                {
                    "start":              "27/03/2017,14:25:42.864",
                    "stop":                                      ""
                }
            ],
```

```
            "device":                                   "8003-36000",
            "identifier":                                    "36063",
            "dosage":                                              [
                {
                    "rateQuantity":          "7.40            ml/h"
                }
            ]
        }
    ]
}
```


Siste filen innholder både start og stopptidspunkt fordi dosen endres først, også avsluttes medisineringen.


```
{
    "MedicationAdministration":                                    [
        {
            "status":                                   "completed",
            "medication":                              "Nitropress",
            "effectivePeriod":                                     [
                {
                    "start":              "27/03/2017,14:26:20.020",
                    "stop":               "27/03/2017,14:26:38.524"
                }
            ],
            "device":                                   "8003-36000",
            "identifier":                                    "36065",
            "dosage":                                              [
                {
                    "rateQuantity":          "4.20            ml/h"
                }
            ]
        }
    ]
}
```


Dokumentasjon på økning av blodtrykket.

Noen pasienter er ustabile på grunn av deres kardiovaskulære[4] sykdom. Blodårene er stive og forkalkede og forsøk på å kontrollere blodtrykket kan gi store utslag. Etter at nitropress er avsluttet fortsetter trykket å synke og man beslutter å starte noradrenalin for å øke trykket igjen. Man starter med en dose på 4.20 ml/h, men trykket forsetter å synke derfor øker man dosen til 8.40 ml/h. Når dosen endres begynner artefaktet å skrive til en ny fil. Når trykket begynner å stige trapper man igjen ned dosen og for hver gang man gjør en endring starter artefaktet på en ny fil. Siden man bare kan dokumentere én dose per journalnotat, blir det fort mange journalnotat. Det første notatet inneholder bare starttidspunkt.


```
{
    "MedicationAdministration":                                    [
        {
            "status":                                 "in-progress",
```


[4] Noe som har med eller henger sammen med hjerte og/eller blodsirkulasjonen.

```
            "medication":                                   "Noradrenalin",
            "effectivePeriod":                                          [
                {
                    "start":                "27/03/2017,14:25:42.864",
                    "stop":                                          ""
                }
            ],
            "device":                                     "8003-35698",
            "identifier":                                      "36053",
            "dosage":                                                  [
                {
                    "rateQuantity":          "4.20           ml/h"
                }
            ]
        }
    ]
}
```


Legg merke til at de to neste journalnotatene kun har starttidspunkt. Dette representerer tidspunktet for endring av dosen.


```
{
    "MedicationAdministration":                                        [
        {
            "status":                                    "in-progress",
            "medication":                               "Noradrenalin",
            "effectivePeriod":                                          [
                {
                    "start":                "27/03/2017,14:26:20.020",
                    "stop":                                          ""
                }
            ],
            "device":                                     "8003-35698",
            "identifier":                                      "36055",
            "dosage":                                                  [
                {
                    "rateQuantity":          "8.40           ml/h"
                }
            ]
        }
    ]
}


{
    "MedicationAdministration":                                        [
        {
            "status":                                    "in-progress",
            "medication":                               "Noradrenalin",
            "effectivePeriod":                                          [
                {
                    "start":                "27/03/2017,14:26:51.372",
                    "stop":                                          ""

            ],
            "device":                                     "8003-35698",
```

```
            "identifier":                                    "36057",
            "dosage":                                                [
                {
                    "rateQuantity":          "4.20            ml/h"
                }
            ]
        }
    ]
}
```


Det siste journalnotatet inneholder både start- og stopptidspunkt. Det betyr at første ble dosen endret. Før medisineringen avsluttes litt senere.


```
{
    "MedicationAdministration":                                  [
        {
            "status":                                  "completed",
            "medication":                             "Noradrenalin",
            "effectivePeriod":                                    [
                {
                    "start":              "27/03/2017,14:27:33.921",
                    "stop":               "27/03/2017,14:27:44.332"
                }
            ],
            "device":                                 "8003-35698",
            "identifier":                                 "36059",
            "dosage":                                             [
                {
                    "rateQuantity":          "3.00            ml/h"
                }
            ]
        }
    ]
}
```

5. Diskusjon

HL7 FHIR er egentlig en standard for utveksling av informasjon. Sammen med nye standarder som IEEE 11073 SDC vil høsting direkte fra medisinsk utstyr være mulig. Det går også an å rekonstruere eller presentere behandlingen retrospektivt i kurveform.

I denne oppgaven brukes HL7 FHIR som grunnlag for integrering av sanntids datastrømmer til elektronisk overvåkningskurve og automatisk dokumentere til den elektroniske pasientjournalen.

Litt av målet med dette prosjektet var å lage et artefakt som demonstrere proof-of-concept. For at dette ikke bare skal være en løsning kun til glede lokalt er det viktig at løsningen er generaliserbar til alle områder hvor man trenger å utveksle eller høste klinisk informasjon.

I dag lagres den kliniske informasjonen i forskjellige systemer som ikke utveksler informasjon med hverandre. Av all informasjonen som genereres under en åpen hjerteoperasjon, er det bare summen av væskebalansen som muntlig formidles mellom personalet og føres inn i anestesiens kurvesystem som en diskret verdi. Det er ingen informasjon eller kontekst knyttet til når og hvorfor det tilsatt eller fjernet væske fra det sirkulerende blodet. Utover væskebalansen videreføres ingenting av informasjonen som er høstet på hjerte-lungemaskinen.

For å unngå fragmentering og oppnå interoperabilitet må informasjonen standardiseres. HL7 FHIR er siste skudd på treet av standarder for utveksling av helseinformasjon. Den benytter seg av godt utbredte og åpne web-standarder for lagring og utveksling av klinisk informasjon.

I utgangspunktet er ikke HL7 FHIR et system for plotting av overvåkningskurver i sanntid, men en standard for utveksling av journaldata. Fordelene er bedre søkemuligheter, datagrunnlag til forskning og utveksling av informasjon.

For at det skal gå an å høste informasjon fra alle pumpene må artefaktet kunne høste data parallelt. For å gjøre en proof-of-concept i denne oppgaven var det ikke nødvendig da data ble høstet serielt. Først for å senke blodtrykket, så for å øke blodtrykket. Men i en klinisk setting vil det gå mange medisiner samtidig. En måte å løse dette på er å benytte seg av multithreading.

Resultatene i denne oppgaven viser at HL7 FHIR kan brukes i flere kliniske sammenhenger, ikke bare til journalføring.

5.1. Vurdering

Normalt i et design science prosjekt vil man utføre en evaluering av resultatet. En masteroppgave har sine begrensninger i tilgjengelige ressurser, tid og omfang. Derfor er det ikke mulig å gjennomføre en fullstendig evaluering. Men siden det er en realistisk problemstilling, hentet fra egen hverdag er det til en viss grad mulig å gjennomføre en vurdering. Den baserer seg på egne erfaringer, diskusjoner med kolleger og tilbakemeldinger fra presentasjon av prosjektet i klinikken.

5.1.1. Vurdering av datastrømmen og medicationAdministration

Datastrømmen som kommer fra pumpen er veldig detaljert og nøyaktig. Den sier veldig mye om medsineringen og danner et godt datagrunnlag for å kunne betrakte behandlingen retrospektivt. Ulempen er at det fort kan bli veldig store datamengder som tar mye båndbredde og kapasitet hvis en skal lagre alle data som kommer ca. hvert 1,2s. Dette kan gå ut over lesbarheten til dokumentet og kapasiteten til serveren. Det er ikke sikkert all informasjon er like relevant og en kan spare mye båndbredde ved å være selektiv. På en annen side er det masse informasjon i en slik datastrøm som kan gi veldig detaljert oversikt over behandlingen hvis en

tar vare på all informasjonen. Men det er rom både for effektivisering og komprimering uten at det nødvendigvis går ut over datakvaliteten, slik at man kan rekonstruere hendelsene svært nøyaktig.

Maturity level

En må også ta med i vurderingen at `medicationAdministration` har en *maturity level* på 2 som har følgende definisjon:

> the artifact has been tested and successfully exchanged between at least three independently developed systems leveraging at least 80% of the core data elements using semi-realistic data and scenarios based on at least one of the declared scopes of the resource.

(37)

Det betyr at det ennå er mye testing og utbedring igjen før denne ressursen er klarert for klinisk bruk.

5.1.1.1. medicationAdministration.status

```
{
    "MedicationAdministration": [
        {
            "status": "in-progress",
```

Artefaktet automatiserer prosessen med å lagre data fra sprøytepumpene. Når den mottar data settes status til ”in-progress” og når siste sample er mottatt settes status til ”completed” automatisk. Men noen manuelle oppgaver virker det som man ikke kan unngå foreløpig. Hvis man f.eks. legger inn feil dose og korrigerer det, skal status på feildoseringen settes til ”entered-in-error”. Dette kan man ikke gjøre fra pumpen, da må man inn i journalen og korrigere dette etterpå.

5.1.1.2. medicationAdministration.medication

```
{
    "MedicationAdministration": [
        {
            "medication": "Nitropress",
```

Det er noen grunnleggende forskjeller på et journalsystem og et kurvesystem. Ved avdelinger der man behandler og overvåker de sykeste pasientene brukes gjerne en stor mengde medisiner intermitterende i korte eller lengre perioder. Man gir medisinene basert på observasjoner og ordner formalitetene i etterkant.

I dette proof-of-concept benytter artefaktet medisininformasjon fra sprøytepumpen. I figurene 4.1 og 4.2 vises medisinen som en egen ressurs. Normalt vil det være en kobling til denne ressursen i dette feltet (59).

Man kan f.eks. automatisere denne prosessen helt ved at artefaktet benytter navnet på medisinen til å søke opp riktig ressurs og legge til koblingen. Ulempen kan være at det blir for uspesifikt å søke bare på medikament ”nitropress”. Det kan være flere produsenter, forskjellig styrke og innpakning. Dette skaper gjerne utfordringer med automatiseringen og derfor må ofte medisinene bekreftes manuelt. Et alternativ som krever litt større investeringer er å benyt-

te seg av strekkodeleser til å skanne medikamentet slik at man bare bekrefter at det er riktig medisin og slipper arbeidet med å taste inn manuelt.

5.1.1.3. medicationAdministration.effectivePeriod

```
{
    "MedicationAdministration": [
        {
            "effectivePeriod": [
                {
                    "start": "27/03/2017,14:26:20.020",
                    "stop": "27/03/2017,14:26:38.524"
                }
```

Sprøytepumpen sender ut datasample ca. hvert 1,2s med et millisekunds nøyaktighet. Da skriver den ut både dato og tidspunkt for hvert enkelt datasample. Men for å kunne rekonstruere en medisininfusjon trenger man egentlig bare start- og stopptidspunktet. Det er ofte mange pumper som brukes samtidig og det er et poeng at det ikke sendes data oftere enn nødvendig til serveren. Ved at klienten sender start- og stopptidspunkt vil datamengden reduseres kraftig, samt øke lesbarheten på dokumentet.

For å integrere datastrømmene ble alle tidsstemplene og korresponderende dose brukt for å få en så nøyaktig grafisk fremstilling av hendelsen som mulig. Hvis man skal følge kardinaliteten slavisk der man bare noterer start-, endrings-, stopptid og dose vil man ikke få med alle nyansene i en fil. For en medisininfusjon som går kontinuerlig på samme dose er det uansett en kontinuerlig rett strek som skal tegnes, men hvis det gjøres mange endringer på dosen underveis vil det bli mange filer. Det kan skape utfordringer å få med seg alle filene hvis man vil rekonstruere hendelsen. Da kan det være mer hensiktsmessig å lagre hele medisineringen i en fil slik at en har kontroll på alle dataene, men da må det tillates i ontologien at man kontinuerlig lagrer tidsstempelet og korresponderende medisindose, se eksempel i figur 5.1.

```
{
    "status": "in-progress",
    "device": "8003-36000",
    "medication": "Nitropress",
    "effectivePeriod": "27/03/2017,14:25:42.864, 27/03/2017,14:25:43.983, 27/03/2017,14:25:45.204,
    27/03/2017,14:25:46.389, 27/03/2017,14:25:47.581, 27/03/2017,14:25:48.753, 27/03/2017,14:25:49.937,
    27/03/2017,14:25:51.121, 27/03/2017,14:25:52.327, 27/03/2017,14:25:53.461, 27/03/2017,14:25:54.606,
    27/03/2017,14:25:55.732, 27/03/2017,14:25:56.869, 27/03/2017,14:25:57.987, 27/03/2017,14:25:59.173,
    27/03/2017,14:26:00.365, 27/03/2017,14:26:01.522, 27/03/2017,14:26:02.643, 27/03/2017,14:26:03.810,
    27/03/2017,14:26:04.928, 27/03/2017,14:26:06.133, 27/03/2017,14:26:07.258, 27/03/2017,14:26:08.428,
    27/03/2017,14:26:09.570, 27/03/2017,14:26:10.702, 27/03/2017,14:26:11.880, 27/03/2017,14:26:13.076,
    27/03/2017,14:26:14.210, 27/03/2017,14:26:15.427, 27/03/2017,14:26:16.551, 27/03/2017,14:26:17.682,
    27/03/2017,14:26:18.872, 27/03/2017,14:26:20.020, 27/03/2017,14:26:21.146, 27/03/2017,14:26:22.284,
    27/03/2017,14:26:23.482, 27/03/2017,14:26:24.656, 27/03/2017,14:26:25.824, 27/03/2017,14:26:26.963,
    27/03/2017,14:26:28.104, 27/03/2017,14:26:29.241, 27/03/2017,14:26:30.384, 27/03/2017,14:26:31.523,
    27/03/2017,14:26:32.643, 27/03/2017,14:26:33.861, 27/03/2017,14:26:34.979, 27/03/2017,14:26:36.172,
    27/03/2017,14:26:37.292, 27/03/2017,14:26:38.524, ",
    "resourceType": "medisinadministrasjon",
    "identifier": "36063",
    "dosage": [
        {
            "text": "8003-36000",
            "method": "INF",
            "dose": "7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40
            ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h,
            7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40
            ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 7.40 ml/h, 4.20 ml/h, 4.20 ml/h, 4.20 ml/h, 4.20 ml/h, 4.20 ml/h,
            4.20 ml/h, 4.20 ml/h, 4.20 ml/h, 4.20 ml/h, 4.20 ml/h, 4.20 ml/h, 4.20 ml/h, 4.20 ml/h, 4.20 ml/h, 4.20
            ml/h, 4.20 ml/h, 4.20 ml/h, "
        }
    ]
}
```


Figur 5.1: Viser et eksempel på journalnotat med alle samples.

Denne figuren viser et eksempel på en JSON-fil der man ivaretar effictivePeriod som x-aksen og de korresponderende dosene som y-verdier i ett og samme journalnotat. Da får man ett journalnotat for hver hendelse. Ulempen er at det blir mere trafikk på nettet.

I denne oppgaven ble den interne klokken i sprøytepumpen benyttet. Sykehuset eier nesten 1000 sprøytepumper som alle må stilles manuelt. Det blir en ressurskrevende oppgave å holde orden på tiden. I tillegg er ikke formatet på tiden standardisert i henhold til formatet HL7 FHIR bruker. Derfor bør man bruke en tidsserver. Det sikrer også at alle tidspunkt kan standardiseres og kan formateres riktig i objektet `medicationAdministration.effectivePeriod` (58).

5.1.1.4. medicationAdministration.dosage.rateQuantity

```
{
    "MedicationAdministration": [
        {
            "dosage": [
                {
                    "rateQuantity": "4.20 ml/h"
```

Dette objektet er beskrevet slik at man bare kan lagre én dose. Hvis dosen endres skal det dokumenteres i et nytt journalnotat ved at det lages en ny fil (60).

Som beskrevet i kapittel 5.1.1.3 kan det fort bli mange filer når det er mange pumper som endres kontinuerlig. Dette kan gjøre det uoversiktlig og utfordrende å rekonstruere hendelsen i ettertid. En må være sikker på å få med seg alle filene tilhørende én infusjon. Man kan ikke tegne kurven basert på én fil, det gir bare en rett strek.

5.1.1.5. medicationAdministration.device

```
{
    "MedicationAdministration": [
        {
            "device": "8003-36000",
```

Her er det kun serienummeret til utstyret som blir brukt. Normalt vil det være en link til en egen ressurs for utstyret som heter device (61). Se også figur 4.1 og 4.2. Denne inneholder detaljert informasjon om historikken til utstyret, hvilke pasienter det har vært brukt på, eventuelle merknader fra andre brukere og tilstanden. Det gir god oversikt over når, på hvem og til hva utstyret er brukt. Ved en uønsket hendelse har en full oversikt og kan følge opp med nødvendige undersøkelser. Dette vil være en forbedring fra dagens praksis som gir mulighet til detaljert tilbakemelding til servicepersonell og produsent.

5.1.1.6. medicationAdministration.identifier

```
{
    "MedicationAdministration": [
        {
            "identifier": "36065",
```

Det er en referanse til et eksternt identifikasjonsnummer. Identifikasjonsnummeret som er brukt her hendelsesnummeret som kommer fra sprøytepumpen for den aktuelle dosen som gis (62).

5.1.1.7. HL7 FHIR som grunnlag for sanntidsdata

Ved å bruke HL7 FHIR som grunnlag for sanntidsdata, vil informasjonen kunne presenteres på pasientovervåkningen samtidig som den lagres i journalen fortløpende. Ved å lagre kontinuerlig bevarer man mest mulig informasjon i tilfelle nedetid.

5.1.2. Vurdering av artefaktet

En gjennomgang av artefaktet, hvorfor de ulike løsningene ble valgt og forslag til forbedring.

5.1.2.1. Virtuelle sprøytepumper

Å Laste inn datastrømmen i tekstfiler og benytte de som input til artefaktet har gjort det mulig å jobbe fleksibelt med utviklingen av et proof-of-concept. Da man ikke har vært avhengig av å ha en sprøytepumpe tilgjengelig eller å sitte på jobben når en har programmert.

5.1.2.2. Oversette datastrømmen

Det er i denne funksjonen selve parsingen av data har foregått. Fra data som kommer på RS232-protokoll og over i lister som kan brukes som datagrunnlag til å fylle ut `medicationAdministration` ressursen med. Det er mulig at det finnes bedre måter og løse dette på, men med utgangspunkt som en nybegynner har dette vært en ryddig og oversiktlig måte å gjennomføre programmeringen på.

5.1.2.3. Skriver til dictionary og lagrer JSON-fil

I de første versjonene av artefaktet var output en XML-fil fordi oppfatningen var at det var den beste måten å løse det på. Det ble en veldig komplisert kode til sammenligning med JSON. Python har innebygd støtte for JSON, så det ble mye enklere å skrive innholdet til fil.

5.1.2.4. Lage blodtrykkskurve

Siden det ikke var mulig å lese av en blodtrykkskurve fra en pasientmonitor uten å koble til en pasient måtte den lages. Det er en funksjon som lager vilkårlige tall og sorter de først i synkende rekkefølge, deretter i stigende rekkefølge. Når kurven tegnes og integreres med datastrømmen fra de virtuelle sprøytepumpene får man kurver som viser gevinsten av å integrere de ulike datastrømmene.

Dette kunne også vært løst ved at artefaktet hadde lest inn verdiene fra en tekstfil som en virtuell pasientmonitor. Men en funksjon i Python kan ikke returnere to parametre. Derfor ligger denne funksjonen inne i selve artefaktet.

5.1.2.5. Plotting av kurvene

Opprinnelig var planen å bruke et bibliotek som tegner kurvene i sanntid mens artefaktet kjørte. Dette viste seg å ikke være like lett som det så ut. Derfor ble det brukt et bibliotek fra Plotly til å plotte kurvene. Da tegnes kurvene i en webapplikasjon hvor man har muligheten til å tilpasse og eksperimentere med datasett, utseende og presentasjon. Noe som kan være en fordel i utviklingsfasen.

5.1.3. Tilbakemelding fra klinikken

Etter å ha presentert prosjektet for klinikken kom det bare positive tilbakemeldinger på arbeidet og resultatet. Det virket som det ga mening, selv for en konservativ gruppe som klinikere ofte er, med en integrert presentasjon av klinisk informasjon.

5.1.4. Håndtere alle typer data

Det er nødvendig at artefaktet kan tilpasses slik at det kan håndtere alle typer data. Det kan bli store datamengder som endrer seg kontinuerlig. Dette kan øke belastningen både på infrastruktur og lagringskostnadene (32).

HL7 FHIR har en ontologi som støtter alle typer data. Det er viktig å se på muligheten for å kunne lagre alle typer datastrømmer som lyd, bilde og video.

Denne løsningen vil ha allmenn interesse fordi den demonstrerer ikke bare løsningen på de konkrete problemene beskrevet i denne oppgaven. Den viser også at det er mulig å løse de fleste utfordringer knyttet til manglende datahøsting og integrering av informasjon.

5.2. Naturlige begrensninger i denne oppgaven

Det er praktiske utfordringer knyttet til utviklingen av dette artefaktet, som gjør at det ikke er et ferdig produkt. Disse er knyttet til begrensningene i en masteroppgave, tiden og kandidatens kompetanse. Personlig har jeg ønsket å lære og forstå mest mulig av dette grunnleggende problemet i helsevesenet nå når jeg har muligheten.

Fokuset her har vært å vise at konseptet fungerer, har en allmenn interesse og kan være verdt å jobbe videre med. Under utdyper jeg noen av utfordringene som har begrenset omfanget av denne oppgaven.

5.2.1. Tiden

Tiden har vært en begrensende faktor. I tillegg til at dette er en masteroppgave som har et naturlig begrenset omfang, har jeg familie og er i full jobb. Jeg har brukt all ledig tid på jobb og fritiden for å gjennomføre denne oppgaven. Det har ikke vært et mål å lage et ferdig produkt, men vise at dette kan være en farbar vei ved å demonstrere et proof-of-concept.

5.2.2. Størrelsen på og erfaringen til teamet

Helt klart en begrensende faktor. Jeg har jobbet alene med denne oppgaven. Selv om jeg har teknisk bakgrunn er det en utfordring at jeg ikke har praktisert dette på 12 år. Det har gått med mye tid til å lese og forstå teori og spesifikasjoner. Men erfaringen fra helsevesenet har helt klart hjulpet meg med å isolere problemstillingen.

5.2.3. Begrenset erfaring med programvareutvikling

Jeg har kun hatt grunnleggende innføringskurs i programmering. Har derfor brukt mye tid på å lese meg opp for å løse selv de enkleste problemene. Fordelen er at programmering handler om mengdetrening. Derfor har det gått lettere etterhvert.

5.2.4. Forslag til forbedring av artefaktet

Det har ikke vært meningen å lage et skript som kan brukes i produksjon. Tanken var å lage et konsept som viser at dette er gjennomførbart og kan jobbes videre med for å løse problemet.

Den praktiske tilnærmingen har vært veldig lærerik. Jeg startet med en idé om hvordan artefaktet skulle være, også tester man underveis hva som fungerer og hva som ikke fungerer. Når man jobber med det over lengre tid ser man etterhvert hvordan artefaktet kan forbedres og hvordan det egentlig fungerer i forhold til hvordan en tenkte det skulle fungere. Det er mye rom for forbedring og hadde en jobbet i et team ville nok produktet sett helt annerledes ut.

En av de mest opplagte forbedringene er å kjøre ferdig hele artefaktet for hver datasample det mottar. Slik det fungerer nå leser det hele datastrømmen, tolker og oversetter hele datastrømmen og lagrer en og en sample. Det var for å demonstrere at dataene ble tatt vare på selv om det skulle bli strømbrudd el.l. I en klinisk setting må hele skriptet kjøre for hver datasample som kommer før det mottar neste.

5.2.4.1. Innholdet i JSON-filen

På HL7 FHIR sine nettsider ligger det eksempler på riktig bruk av de forskjellige ressursene til ulike scenarioer. Artefaktet lagrer ikke innholdet til JSON-filen på samme måte som eksempelet gjør. Det er små forskjeller som ikke det er tatt hensyn til for å bevise proof-of-concept. Samtidig har ikke hele scenarioet vært implementert slik at det ikke har vært mulig å fylle ut alle objektene. Slik som f.eks. koble å pasientobjektet mot pasientressursen, sprøytepumpen mot ressursen for utstyr osv.

I den videre utviklingen bør man tilpasse artefaktet slik at filen som lages fungerer med og drar nytte av de andre ressursene som henger sammen i et scenario, se figur 4.1 og 4.2.

5.2.4.2. Multithreading

En begrensning i prototypen er at den bare håndterer serielle datastrømmer. Det vil si at den mottar data og gjør seg ferdig med en datastrøm av gangen. I denne prototypen har ikke det noen praktisk betydning da man ikke gir både blodtrykksøkende og blodtrykkssenkende medisiner samtidig. For å kunne parse parallelle data må det lages en løsning som benytter seg av multithreading. Multithreading gjør det mulig å kjøre flere deler av programmet samtidig. Det krever litt mer arbeid og er ikke nødvendig for å vise at dette artefaktet virker, derfor ble ikke det gjort i forbindelse med utviklingen av dette konseptet. Men skal skriptet brukes i klinikken må denne endringen implementeres. Det er vanlig at de sykeste pasientene får mer enn én medisin i gangen, se figur 5.2.

Figur 5.2: En rack med infusjons- og sprøytepumper.

På de tyngste intensivpasientene er ikke dette uvanlig.

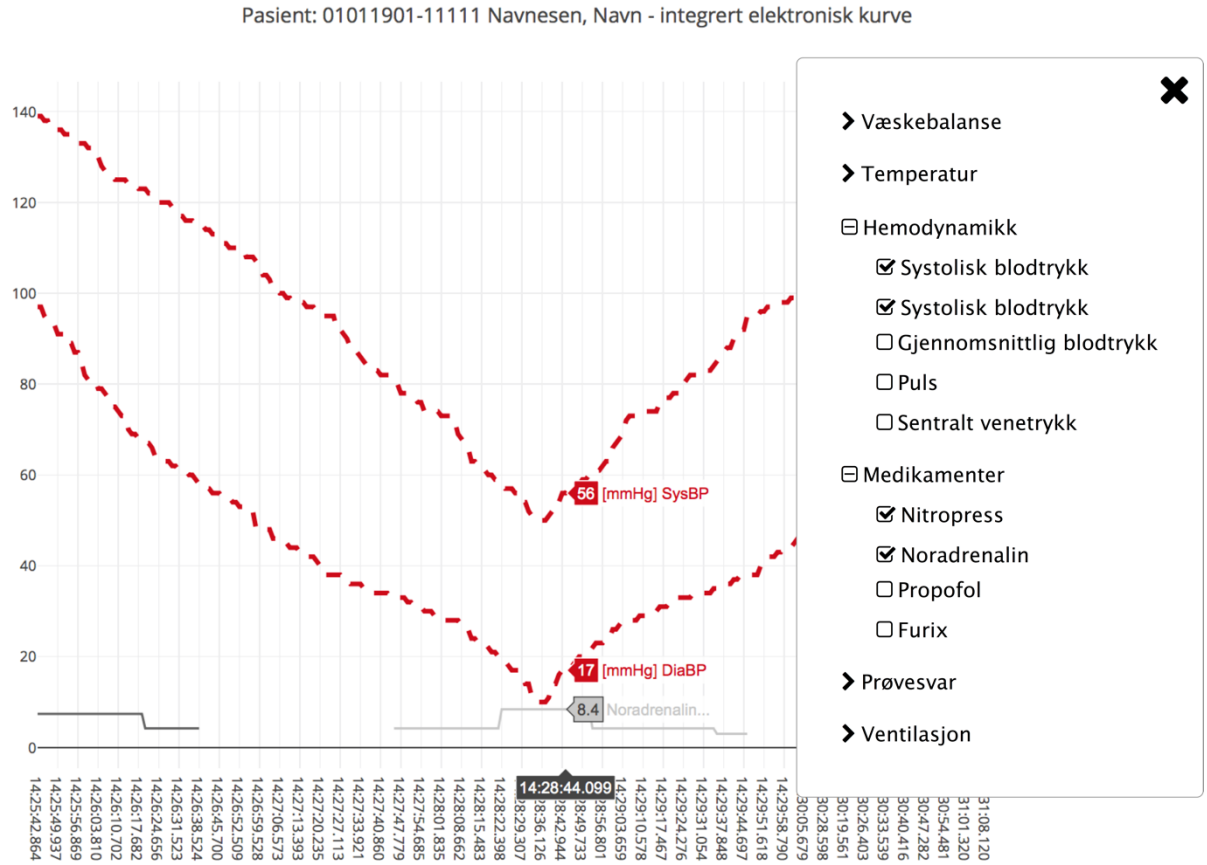
Noe av det første som må på plass hvis en skal gå videre å å lage en fullt fungerende prototype er å kunne kjøre flere deler av programmet samtidig. Dette kan f.eks. løses ved å lage en

funksjon eller et bibliotek av koden som oversetter datastrømmen slik at den kan kalles hver gang det legges til en ny datastrøm. Da vil prototypen håndtere flere tråder samtidig.

5.2.4.3. Formatering av grafene

Plot.ly gir gode muligheter til å teste ut presentasjon av informasjonen. Det bør lages en tilpasset presentasjon som gir personalet god oversikt over pasientens tilstand tilpasset behandlingen som gis (63).

En av fordelene når man har standardisert alle datastrømmene er at en kan sammenfatte de slik man ønsker og synes mest hensiktsmessig for akkurat den behandlingen som utføres. Figur 5.3 viser et eksempel på brukergrensesnitt som enkelt kan tilpasses ønsket kontekst.



Figur 5.3: Eksempel på brukergrensesnitt med integrerte kurver.

6. Konklusjon

I denne oppgaven har en sett nærmere på om det er mulig å benytte HL7 FHIR som er laget for å standardisere og utveksle journaldata til å sammenfatte og presentere integrerte datastrømmer. Dette er essensielt for å unngå fragmentering og miste konteksten informasjonen oppstår i.

Metoden som ble benyttet for å løse problemstillingen var design science. Ved hjelp av to datastrømmen fra virtuelle sprøytepumper ble informasjonen oversatt og lagret i henhold til HL7 FHIR-standarden. Samtidig ble datastrømmen presentert sammen med en observert blodtrykkskurve slik at man fikk en integrert visualisering.

Det vil være meningsløst å samle inn alle mulig data hvis de ikke kan integreres. Dette gjelder også data fra elektronisk kurve og elektronisk journal.

Resultatet viser at det er mulig å sammenfatte, presentere integrerte datastrømmer og lagre disse til en elektronisk pasientjounal. Men når en benytter seg av HL7 FHIR-standarden får man ikke den helt optimale lagringen av en datastrøm i kontinuerlig endring. Da kan det bli uoversiktlig hvis man ønsker å studere en hendelse retrospektivt. Det blir fort mange filer og man kan risikere å ikke få med all informasjonen.

7. Referanseliste

1. Fox R, Sahay R, Hauswirth M. PPEPR for Enterprise Healthcare Integration. In: Electronic Healthcare [Internet]. Springer, Berlin, Heidelberg; 2008 [cited 2017 Nov 5]. p. 130–7. (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering). Available from: https://link.springer.com/chapter/10.1007/978-3-642-00413-1_16

2. Lamprinakos GC, Mousas AS, Kapsalis AP, Kaklamani DI, Venieris IS, Boufis AD, et al. Using FHIR to develop a healthcare mobile application. In: 2014 4th International Conference on Wireless Mobile Communication and Healthcare - Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH). 2014. p. 132–5.

3. Andersen B, Kasparick M, Ulrich H, Schlichting S, Golatowski F, Timmermann D, et al. Point-of-care medical devices and systems interoperability: A mapping of ICE and FHIR. In: 2016 IEEE Conference on Standards for Communications and Networking (CSCN). 2016. p. 1–5.

4. Benson T, Grieve G. Principles of Health Interoperability: SNOMED CT, HL7 and FHIR. Springer; 2016. 465 p.

5. Botsis T, Hartvigsen G, Chen F, Weng C. Secondary Use of EHR: Data Quality Issues and Informatics Opportunities. Summit Transl Bioinforma. 2010 Mar 1;2010:1–5.

6. Ainsworth J, Buchan I. Combining Health Data Uses to Ignite Health System Learning. Methods Inf Med. 2015;54(6):479–487.

7. Cleary K. Medical Robotics and the Operating Room of the Future. In: 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference. 2005. p. 7250–3.

8. Sexton JB, Makary MA, Tersigni AR, Pryor D, Hendrich A, Thomas EJ, et al. Teamwork in the Operating RoomFrontline Perspectives among Hospitals and Operating Room Personnel. Anesthesiol J Am Soc Anesthesiol. 2006 Nov 1;105(5):877–84.

9. Encyclopedia Britannica. heart-lung machine | medical device [Internet]. Encyclopedia Britannica. 2008 [cited 2017 Nov 8]. Available from: https://www.britannica.com/topic/heart-lung-machine

10. Opdahl H. sentralt venetrykk. In: Store medisinske leksikon [Internet]. 2017 [cited 2017 Nov 5]. Available from: http://sml.snl.no/sentralt_venetrykk

11. Opdahl H. hypovolemi. In: Store medisinske leksikon [Internet]. 2017 [cited 2017 Nov 5]. Available from: http://sml.snl.no/hypovolemi

12. Mouttham A, Kuziemsky C, Langayan D, Peyton L, Pereira J. Interoperable support for collaborative, mobile, and accessible health care. Inf Syst Front. 2012 Mar 1;14(1):73–85.

13. Schoen C, Osborn R, Huynh PT, Doty M, Zapert K, Peugh J, et al. Taking the pulse of health care systems: experiences of patients with health problems in six countries. Health Aff Proj Hope. 2005;Suppl Web Exclusives:W5-509–25.

14. Pirnejad H, Niazkhani Z, Berg M, Bal R. Intra-organizational communication in healthcare--considerations for standardization and ICT application. Methods Inf Med. 2008;47(4):336–45.

15. Duftschmid G, Rinner C, Kohler M, Huebner-Bloder G, Saboor S, Ammenwerth E. The EHR-ARCHE project: Satisfying clinical information needs in a Shared Electronic Health Record System based on IHE XDS and Archetypes. Int J Med Inf. 2013 Dec 1;82(12):1195–

207.

16. Berg M, Goorman E. The contextual nature of medical information. Int J Med Inf. 1999 Dec;56(1–3):51–60.

17. Health Layer 7. Summary - FHIR v3.0.1 [Internet]. [cited 2017 Nov 5]. Available from: http://www.hl7.org/implement/standards/fhir/summary.html

18. IEEE. Standards Glossary [Internet]. IEEE Standards University. [cited 2017 Nov 5]. Available from: https://www.standardsuniversity.org/article/standards-glossary/

19. Bender D, Sartipi K. HL7 FHIR: An Agile and RESTful approach to healthcare information exchange. In: Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems. 2013. p. 326–31.

20. Wikipedia. Semantic interoperability. In: Wikipedia [Internet]. 2017 [cited 2017 Nov 5]. Available from: https://en.wikipedia.org/w/index.php?title=Semantic_interoperability&oldid=790371960

21. Wikipedia. Conceptual interoperability. In: Wikipedia [Internet]. 2017 [cited 2017 Nov 12]. Available from: https://en.wikipedia.org/w/index.php?title=Conceptual_interoperability&oldid=794505524

22. Pronovost P, Palmer S, Ravitz A. What Hospitals Can Learn from Airlines About Buying Equipment [Internet]. Harvard Business Review. 2017 [cited 2017 Nov 5]. Available from: https://hbr.org/2017/06/hospitals-are-dramatically-overpaying-for-their-technology

23. Wikipedia. Standard. In: Wikipedia [Internet]. 2016 [cited 2017 Nov 5]. Available from: https://no.wikipedia.org/w/index.php?title=Standard&oldid=16301006

24. Marcheschi P. Relevance of eHealth standards for big data interoperability in radiology and beyond. Radiol Med (Torino). 2017 Jun 1;122(6):437–43.

25. Health Layer 7. European Committee for Standardization. In: Wikipedia [Internet]. 2017 [cited 2017 Nov 18]. Available from: https://en.wikipedia.org/w/index.php?title=European_Committee_for_Standardization&oldid=791118404

26. Standard Norge. Standard Norge [Internet]. [cited 2017 Nov 5]. Available from: http://www.standard.no/toppvalg/om-oss/standard-norge/

27. ANSI. About ANSI [Internet]. [cited 2017 Nov 18]. Available from: https://www.ansi.org/about_ansi/overview/overview?menuid=1

28. Health Layer 7. About Health Level Seven International [Internet]. [cited 2017 Nov 5]. Available from: http://www.hl7.org/about/index.cfm?ref=common

29. Ferreira RJT, Correia MECD, Gonçalves FNR, Correia RJC. Data Quality in HL7 Messages – A Real Case Analysis. In: 2015 IEEE 28th International Symposium on Computer-Based Medical Systems. 2015. p. 197–200.

30. Mandel JC, Kreda DA, Mandl KD, Kohane IS, Ramoni RB. SMART on FHIR: a standards-based, interoperable apps platform for electronic health records. J Am Med Inform Assoc. 2016 Sep 1;23(5):899–908.

31. Health Layer 7. Documentation - FHIR v3.0.1 [Internet]. [cited 2017 Nov 5]. Available from: http://www.hl7.org/implement/standards/fhir/documentation.html

32. Schuler A, Franz B, Krauss O. A Comparison of Data Traffic in Standardized Personal Health Monitoring Solutions. Int J Electron Telecommun. 2015 Jan;61(2):143–149.

33. Sommerville I. Software Engineering, Global Edition [Internet]. 10th ed. Pearson Education Limited; 2015. Available from: https://www.akademika.no/software-engineering-global-edition/ian-sommerville/9781292096131

34. Health Layer 7. Profiling - FHIR v3.0.1 [Internet]. [cited 2017 Nov 5]. Available from: http://hl7.org/fhir/profiling.html

35. Health Layer 7. Resourceguide - FHIR v3.0.1 [Internet]. [cited 2017 Nov 5]. Available from: http://www.hl7.org/implement/standards/fhir/resourceguide.html

36. Health Layer 7. Resource - FHIR v3.0.1 [Internet]. [cited 2017 Nov 5]. Available from: https://www.hl7.org/fhir/resource.html

37. Health Layer 7. Versions - Maturity Level - FHIR v3.0.1 [Internet]. [cited 2017 Nov 17]. Available from: http://www.hl7.org/fhir/versions.html#maturity

38. Health Layer 7. Resourcelist - FHIR v3.0.1 [Internet]. [cited 2017 Nov 27]. Available from: http://www.hl7.org/fhir/resourcelist.html

39. Health Layer 7. MedicationAdministration - FHIR v3.0.1 [Internet]. [cited 2017 Nov 3]. Available from: http://www.hl7.org/fhir/medicationadministration.html

40. Wikipedia. Representational state transfer. In: Wikipedia [Internet]. 2017 [cited 2017 Nov 5]. Available from: https://en.wikipedia.org/w/index.php?title=Representational_state_transfer&oldid=806691400

41. Wikipedia. Stateless protocol. In: Wikipedia [Internet]. 2017 [cited 2017 Nov 5]. Available from: https://en.wikipedia.org/w/index.php?title=Stateless_protocol&oldid=807055121

42. Clinicians on FHIR. clinFHIR Launcher [Internet]. [cited 2017 Nov 5]. Available from: http://clinfhir.com/

43. Lovdata. Forskrift om medisinsk utstyr - Kapittel 1. Innledende bestemmelser - Lovdata [Internet]. [cited 2017 Nov 5]. Available from: https://lovdata.no/dokument/SF/forskrift/2005-12-15-1690/KAPITTEL_1#%C2%A71-5

44. Bonney L. The Connectivity Problem [Internet]. [cited 2017 Nov 16]. Available from: https://www.redoxengine.com/blog/the-connectivity-problem

45. maxEmbedded. The USART of the AVR [Internet]. maxEmbedded. 2013 [cited 2017 Nov 5]. Available from: http://maxembedded.com/2013/09/the-usart-of-the-avr/

46. Wikipedia. ISO/IEEE 11073. In: Wikipedia [Internet]. 2017 [cited 2017 Nov 5]. Available from: https://en.wikipedia.org/w/index.php?title=ISO/IEEE_11073&oldid=805710702

47. Rockstroh M, Franke S, Hofer M, Will A, Kasparick M, Andersen B, et al. OR.NET: multi-perspective qualitative evaluation of an integrated operating room based on IEEE 11073 SDC. Int J Comput Assist Radiol Surg. 2017 Aug 1;12(8):1461–9.

48. Pfeiffer JH, Dingler ME, Dietz C, Lueth TC. Requirements and architecture design for open real-time communication in the operating room. In: 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO). 2015. p. 458–63.

49. Kasparick M, Schlichting S, Golatowski F, Timmermann D. New IEEE 11073 standards for interoperable, networked point-of-care Medical Devices. In: 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). 2015. p. 1721–4.

50. Andersen MV, Kristensen IH, Larsen MM, Pedersen CH, Gøeg KR, Pape-Haugaard LB.

Feasibility of Representing a Danish Microbiology Model Using FHIR. In 2017. p. 13–7.

51. Dagale H, Anand SVR, Hegde M, Purohit N, Supreeth MK, Gill GS, et al. CyPhyS+: A Reliable and Managed Cyber-Physical System for Old-Age Home Healthcare over a 6LoWPAN Using Wearable Motes. In: 2015 IEEE International Conference on Services Computing. 2015. p. 309–16.

52. Johannesson P, Perjons E. An introduction to design science. Vol. 9783319106328. Cham: Springer International Publishing; 2014.

53. Hevner AR, March ST, Park J, Ram S. Design science in information systems research. MIS Q Manag Inf Syst. 2004 Mar 1;28:75–105.

54. Hjørland B, Albrechtsen H. Toward a new horizon in information science: Domain-analysis. J Am Soc Inf Sci. 1995 Jul 1;46(6):400–25.

55. Wikipedia. Domain knowledge. In: Wikipedia [Internet]. 2017 [cited 2017 Nov 6]. Available from: https://en.wikipedia.org/w/index.php?title=Domain_knowledge&oldid=784257696

56. BD. Alaris CC Plus-sprøytepumpe med Guardrails - BD [Internet]. [cited 2017 Nov 3]. Available from: https://www.bd.com/no-no/our-products/infusion/infusion-devices/alaris-plus-platform-with-guardrails-safety-software/alaris-cc-plus-syringe-pump-with-guardrails

57. Philips. Philips - IntelliVue MMS X2 Measurement module and patient monitor [Internet]. Philips. [cited 2017 Nov 5]. Available from: https://www.philips.no/healthcare/product/HC865039/intellivue-mms-x2-measurement-module-monitor

58. Health Layer 7. Datatypes - Period - FHIR v3.0.1 [Internet]. [cited 2017 Nov 8]. Available from: http://www.hl7.org/fhir/datatypes.html#Period

59. Health Layer 7. MedicationAdministration - medication - FHIR v3.0.1 [Internet]. [cited 2017 Nov 17]. Available from: http://www.hl7.org/fhir/medicationadministration-definitions.html#MedicationAdministration.medication_x_

60. Health Layer 7. MedicationAdministration - Definitions - Dosage rate - FHIR v3.0.1 [Internet]. [cited 2017 Nov 5]. Available from: http://www.hl7.org/implement/standards/fhir/medicationadministration-definitions.html#MedicationAdministration.dosage.rate_x_

61. Health Layer 7. MedicationAdministration - device - FHIR v3.0.1 [Internet]. [cited 2017 Nov 17]. Available from: http://www.hl7.org/fhir/medicationadministration-definitions.html#MedicationAdministration.device

62. Health Layer 7. Datatypes - Identifier - FHIR v3.0.1 [Internet]. [cited 2017 Nov 16]. Available from: http://www.hl7.org/fhir/datatypes.html#Identifier

63. Plotly. Plotly | Make charts and dashboards online [Internet]. [cited 2017 Nov 5]. Available from: https://plot.ly/

APPENDIKS A: KILDEKODEN TIL ARTEFAKTET

I selve oppgaven er det brukt flytskjema for å gjøre det oversiktlig og lettlest. Kildekoden i sin helhet med kommentarer ligger vedlagt her.

```
# -*- coding: utf-8 -*-
#Først importeres alle bibliotekene en har bruk for
import json
from random import randint
import plotly.plotly as py
import plotly.graph_objs as go

#Deretter deklareres alle funksjonene
#Første funksjon lager blodtrykkskurve og plotter alle kurvene
def kurve(medisin1,medisin2):
    #Deklarerer de nødvendige listene før det
    #genereres en fiktiv, men realistisk blodtrykkskurve
    systoliskBlodtrykkFallende = []
    diastoliskBlodtrykkFallende = []
    systoliskBlodtrykkStigende = []
    diastoliskBlodtrykkStigende = []

    #Denne for-løkken genererer tilfeldig tall for systolisk og
diatolisk blodtrykk
    #og legger de inn i riktig liste.
    for i in range(150):
        sbp = randint(50,140)
        systoliskBlodtrykkFallende.append(sbp)
        systoliskBlodtrykkStigende.append(sbp)
        dbp = randint(20,100)
        if dbp < (sbp - 20):
            diastoliskBlodtrykkFallende.append(dbp)
            diastoliskBlodtrykkStigende.append(dbp)
        else:
            dbp = sbp - 40
            diastoliskBlodtrykkFallende.append(dbp)
            diastoliskBlodtrykkStigende.append(dbp)

    #Sorterer verdien for de to listene systolisk og diastolisk
blodtrykk i fallende orden.
    systoliskBlodtrykkFallende.sort(reverse=True)
    diastoliskBlodtrykkFallende.sort(reverse=True)
```

```
    #Sorterer verdien for de to listene systolisk og diastolisk blodtrykk i stigende orden.
    systoliskBlodtrykkStigende.sort()
    diastoliskBlodtrykkStigende.sort()


    #Setter sammen listene med synkende og stigende blodtrykk til en kurve
    heleDenSystoliskeBlodtrykkskurven = systoliskBlodtrykkFallende + systoliskBlodtrykkStigende
    heleDenDiastoliskeBlodtrykkskurven = diastoliskBlodtrykkFallende + diastoliskBlodtrykkStigende


    #Deklarerer listene som er knyttet til de forskjellige kurvene som skal tegnes og aksene
    #X-aksen er tidspunktet for datamottak. Det er forskjellig opp-start på de to pumpene.
    #Derfor høster den tidspunkt to ganger.
    x_akse_medisin1 = []
    x_akse_medisin2 = []
    #Langs y-aksen plottes dosen som gis for hvert sample den mot-tar.
    y_akse_medisin1 = []
    y_akse_medisin2 = []


    #Skriver ut grafen for nitropress
    #Legger datastrømmen fra sprøytepumpen inn i y-aksen
    for i in range(len(medisin1)):
        y_akse_medisin1.append(medisin1[i][5])


    #Skriver ut grafen for noradrenalin
    #Legger datastrømmen fra sprøytepumpen inn i y-aksen
    for i in range(len(medisin2)):
        y_akse_medisin2.append(medisin2[i][5])


    #Lagrer x-aksen til egen tabell
    fil = open('tidsstempel.txt','r')
    tidsstempel = fil.readlines()
    fil.close()
    for line in tidsstempel:
        line = line.split(',')
        x_akse_medisin1.append(line[1])
```

```
    #Lagrer x2-aksen til egen tabell
    #x2-aksen er tidsstemplingen for hendelse 2: oppstart av norad-
renalin
    fil = open('tidsstempel_forsinket.txt','r')
    tidsstempel = fil.readlines()
    fil.close()
    for line in tidsstempel:
        line = line.split(',')
        x_akse_medisin2.append(line[1])

    #Plotter kurvene
    trace0 = go.Scatter(
        #Plotter den systoliske blodtrykkskurven
        x = x_akse_medisin1,
        y = heleDenSystoliskeBlodtrykkskurven,
        name = '[mmHg] SysBP', #Navngir kurven
        #Formaterer kurven
        line = dict(
            color = ('rgb(205, 12, 24)'),
            width = 4,
            dash = 'dash')
    )
    trace1 = go.Scatter(
        #Plotter den diastoliske blodtrykkskurven
        x = x_akse_medisin1,
        y = heleDenDiastoliskeBlodtrykkskurven,
        name = '[mmHg] DiaBP',
        line = dict(
            color = ('rgb(205, 12, 24)'),
            width = 4,
            dash = 'dash')
    )
    trace2 = go.Scatter(
        #Plotter datastrømmen fra den første medisinen som kommer
        x = x_akse_medisin1,
        y = y_akse_medisin1,
        name = medisin1[0][7] + ' [ml/h]',
        line = dict(
            color = ('rgb(100, 100, 100)'),
```

```
            width = 2,
            dash = 'line')

    )
    trace3 = go.Scatter(
        #Plotter datastrømmen fra den andre datastrømmen som kommer
        x = x_akse_medisin2,
        y = y_akse_medisin2,
        name = medisin2[0][7] + ' [ml/h]',
        line = dict(
            color = ('rgb(200, 200, 200)'),
            width = 2,
            dash = 'line')

    )
    layout = go.Layout(
        #Skriver tittel på kurven
        title='Pasient: 01011901-11111 Navnesen, Navn - integrert
elektronisk kurve'
    )
    data = go.Data([trace0, trace1, trace2, trace3])
    fig = go.Figure (data=data, layout=layout)
    py.plot(fig,  filename = 'basic-line')

#Funksjonen parser datastrømmen
def parser_innleste_data(lesteLinjer):
    #Oppretter lister
    tabell = []
    datastrom = []

    #Rydder i filen og legger linjen inn i en liste
    for linje in lesteLinjer:
        linje = linje.strip('{beh}!')
        linje = linje.strip()
        if linje != '':
            tabell.append(linje)

    #Ordner datastrømmen i en nøstet liste
    for i in range(len(tabell)):
```

```
        if i%2 == 0: #Alle tidsstemplene ligger på partallslinjer. Sjekker om det er en partallslinje.
            tidspunkt = []
            tidspunkt.append(tabell[i].rstrip('}'))   #Legger tids-stempelet til i linjen
        else:
            infusjon = tabell[i].split('^') #Deler datastrømmen vha delingstegnet og legger den i en liste
            datastrom.append(tidspunkt + infusjon)  #Setter sammen listene med tidsstempel og datastrøm og legger de til i en liste.
            infusjon = []
    return datastrom


#Denne funksjonen dumper til JSON-fil
def jsonDump(datastrom):
    test = datastrom[0][5]
    dosering = ''
    starttid = datastrom[0][0]
    sluttid = ''
    event = ''
    filID = datastrom[0][14].split('|')
    filID = filID[0]

    for i in range(len(datastrom)):
        #Her kontrolleres at det er samme dose. Hvis det er endring skal filen lagres og ny fil opprettes.
        if datastrom[i][5] != test:
            test = datastrom[i][5]
            dosering = ''
            starttid = datastrom[i][0]
            sluttid = ''
            event = ''
            filID = datastrom[i][14].split('|')
            filID = filID[0]

        status = 'in-progress' #Setter status til in-progress

        #Endrer status til completed når absolutt siste datasample kommer.
        if i == len(datastrom)-1:
            status = 'completed'
            sluttid = datastrom[-1][0]
```

```
        event += datastrom[i][14] + ', '
        dosering = datastrom[i][5] + ' ' + datastrom[i][6]
        #Oppretter dictionary med innhold fra medicationAdministra-
tion malen
        medisinAdministrasjon        =        {'MedicationAdministra-
tion':[{'identifier':filID,
        'medication':datastrom[0][7], 'status':status,
        'effectivePeriod':[{'start':starttid, 'stop':sluttid}],
        'device':datastrom[0][2],                                    'do-
sage':[{'rateQuantity':dosering}]
        }]}

        #Dumper dictionary til json file
        with   open(datastrom[0][7]   +   '_'   +   'hendelsesID_'   +
str(filID) + '.json','w') as jsonFil:
            jsonFil.write(json.dumps(medisinAdministrasjon,       in-
dent=4, separators=(',', ': ')))
            jsonFil.close()

#Henter datastrøm fra virtuell pumpe
def lesFil(filnavn):
    lesFil = open(filnavn, 'r')
    lesteLinjer = lesFil.readlines()
    lesFil.close()
    return lesteLinjer

#I hoveddelen kjøres selve programmet ved å kalle på de ulike funk-
sjonene

#Starter datastrøm fra virtuell pumpe med nitropress
lesteLinjer = lesFil('Nitropress.txt')
medisin1 = parser_innleste_data(lesteLinjer)
jsonDump(medisin1)

#Starter datastrøm fra virtuell pumpe med noradrenalin
lesteLinjer = lesFil('Noradrenalin.txt')
medisin2 = parser_innleste_data(lesteLinjer)
jsonDump(medisin2)

#Plotter e-kurve
kurve(medisin1, medisin2)
```

APPENDIKS B: MALEN TIL MEDICATIONADMINISTRATION

I et ferdig system vil det være aktuelt å kombinere mange ressurser sammen til en komplett løsning. Også kalt profil i HL7 FHIR sammenheng.

Under kommer et praktisk eksempel med JSON-skjemaet for administrasjon av medisiner. Denne ressursen brukes når enten pasienten tar medisiner selv eller det på andre måter blir administrert medisiner til pasienten. Alt fra tabletter til infusjoner.

```
{
  "resourceType" : "MedicationAdministration",
  // from Resource: id, meta, implicitRules, and language
  // from DomainResource: text, contained, extension, and modifierExtension
  "identifier" : [{ Identifier }], // External identifier
  "definition" : [{ Reference(PlanDefinition|ActivityDefinition) }], // Instantiates protocol or definition
  "partOf" : [{ Reference(MedicationAdministration|Procedure) }], // Part of referenced event
  "status" : "<code>", // R!  in-progress | on-hold | completed | entered-in-error | stopped | unknown
  "category" : { CodeableConcept }, // Type of medication usage
  // medication[x]: What was administered. One of these 2:
  "medicationCodeableConcept" : { CodeableConcept },
  "medicationReference" : { Reference(Medication) },
  "subject" : { Reference(Patient|Group) }, // R!  Who received medication
  "context" : { Reference(Encounter|EpisodeOfCare) }, // Encounter or Episode of Care administered as part of
  "supportingInformation" : [{ Reference(Any) }], // Additional information to support administration
  // effective[x]: Start and end time of administration. One of these 2:
  "effectiveDateTime" : "<dateTime>",
  "effectivePeriod" : { Period },
  "performer" : [{ // Who administered substance
    "actor" : { Reference(Practitioner|Patient|RelatedPerson|Device) }, // R!  Individual who was performing
    "onBehalfOf" : { Reference(Organization) } // Organization organization was acting for
  }],
  "notGiven" : <boolean>, // True if medication not administered
  "reasonNotGiven" : [{ CodeableConcept }], // C? Reason administration not performed
  "reasonCode" : [{ CodeableConcept }], // C? Reason administration performed
```

```
  "reasonReference" : [{ Reference(Condition|Observation) }], // Condition or Observation that supports why the medication was administered
  "prescription" : { Reference(MedicationRequest) }, // Request administration performed against
  "device" : [{ Reference(Device) }], // Device used to administer
  "note" : [{ Annotation }], // Information about the administration
  "dosage" : { // Details of how medication was taken
    "text" : "<string>", // Free text dosage instructions e.g. SIG
    "site" : { CodeableConcept }, // Body site administered to
    "route" : { CodeableConcept }, // Path of substance into body
    "method" : { CodeableConcept }, // How drug was administered
    "dose" : { Quantity(SimpleQuantity) }, // Amount of medication per dose
    // rate[x]: Dose quantity per unit of time. One of these 2:
    "rateRatio" : { Ratio }
    "rateQuantity" : { Quantity(SimpleQuantity) }
  },
  "eventHistory" : [{ Reference(Provenance) }] // A list of events of interest in the lifecycle
}
```