AALBORG UNIVERSITET
INSTITUT FOR SUNDHEDSTEKNOLOGI

Fredrik Bajersvej 7 ▪ DK-9220 Aalborg Øst Telefon 96 35 87 76

TITEL:	Softewaresystem til håndtering af intrakraniel trykmåling
TEMA:	Design af sundhedsteknologiske systemer
PROJEKTPERIODE:	6. semester, foråret 2003
PROJEKTGRUPPE:	691

DELTAGERE:
Anne Østervig Boilesen
Thomas Borup
Thomas Bork Hardahl
Rikke Hjort Jensen
Tue Rask Nielsen
Jens Krog Vistisen

VEJLEDER:
Michael Voigt

Synopsis

I diagnosticering af normaltrykshydrocephalus (NPH) er det vigtigt, at lægen har et godt vurderingsgrundlag i form af en intrakraniel trykmåling eller infusionstest. Præsentation heraf er af stor vigtighed for at give lægen de bedste muligheder for at stille den korrekte diagnose. Præsentation af trykmålinger på Aalborg Sygehus foregår i dag ved hjælp af en trykmåler og en analog printer. Gennem objektorientert analyse og design er et softwaresystem til hjælp ved diagnosticering af NPH udviklet. Modelleringssproget Unified Modeling Language er anvendt i analyse- og designfasen, og systemet er implementeret i programmeringssproget Java.
Systemet er inddelt i tre dele: 24-timers ICP-måling, Detektering af B-bølger og Infusionstest. 24-timers ICP-måling udgør den del af systemet, der opsamler og præsenterer det intrakranielle tryksignal under døgnmåling af det intrakranielle tryk. Detektering af B-bølger er den del af systemet, der præsenterer det opsamlede signal fra 24-timers ICP-måling og giver lægen hjælp til at detektere B-bølgeaktivitet. Infusionstest præsenterer tryksignalet under infusionstesten, og giver efterfølgende mulighed for at se en filtreret kurve og udregne R_{out}.
Problemformuleringen er opfyldt, idet der er implementeret et softwaresystem til hjælp ved diagnosticering af NPH, der opfylder de opsatte krav. Der ligger dog stadig forskningsmæssige udfordringer i at fastlægge en parameterkarakteristik for detektering af B-bølgeaktivitet.

OPLAGSTAL:	10
SIDEANTAL:	169
TILLÆG:	CD-ROM og brugermanual
AFSLUTTET:	23. maj 2003

Forord

Temaet for 6. semester Sundhedsteknologi er *Design af sundhedsteknologiske systemer*. Projektet har med udgangspunkt i 5. semestersprojektetet "Diagnosticering af normaltrykshydrocephalus ved intrakraniel trykmåling" [3] forsøgt at løse en konkret problemstilling på neurokirurgisk afdeling på Aalborg Sygehus (AAS).

Problemstillingen på AAS blev analyseret på 5. semester, og projektet vil derfor hovedsageligt bestå af analyse og design af de softwaremæssige dele samt implementation og test af et system til anvendelse på AAS, neurokirurgisk afd. K.

Rapporten henvender sig primært til vejleder og censor samt medstuderende med interesse for emnet. Derfor forudsætter udbytterig læsning samme faglige grundlag som projektgruppen.

Læsevejledning

Rapporten er opdelt i 4 hoveddele: Indledning, Analyse, Design og Implementation og Test. I analyse- og designfasen er der anvendt Unified Modelling Language (UML).For yderligere uddybelse af syntaksen i UML henvises appendiks A og [21].
Den engelske terminologi i UML er enkelte tilfælde ikke forsøgt oversat.

Figurer, diagrammer og tabeller er nummereret fortløbende gennem kapitlerne, ligesom litteraturhenvisninger angives med fortløbende numre i firkantede paranteser. For at gøre det nemmere for læseren at finde de kapitler, der refereres til i 5. semesters projektet, er der i flere tilfælde angivet sidetal efter referencen [3] på følgende måde: [3, s. 1] . En samlet litteraturliste findes bagerst i rapporten.
Forkortelser angives i parentes første gang de bliver nævnt, og vil efterfølgede blive anvendt.

På rapportens bagside findes en vedlagt CD-ROM, der indeholder en række af rapportens bilag. Der er i teksten henvist til de enkelte bilag, der indeholder manualer, datablade m.v. Desuden forefindes en elektronisk udgave af rapporten, samt kildekoden til systemet. Ligeledes vil det være muligt at eksekvere programmet fra CD'en. Der er udarbejdet en brugermanual til systemet, der vedlægges rapporten.

Vi takker

Projektgruppen ønsker at takke neurokirurg Preben Sørensen (AAS), der har givet input til udvikling af et anvendeligt og implementerbart system og til Integra NeuroScience for adgang til protokollen til trykmåleren. Endvidere en tak til alle patienter, der gav projektgruppen lov til at afprøve demo-udgaver til opsamling af data, og en tak tilvejleder Michael Voight for råd og vejledning.

Aalborg den 23. maj 2003, udarbejdet af:

Anne Østervig Boilesen	Thomas Borup
Rikke Hjort Jensen	Thomas Bork Hardahl
Tue Rask Nielsen	Jens Krog Vistisen

INDHOLD

Del I

INDLEDNING

KAPITEL 1

Problembaggrund

Kapitlet introducerer den initierende problemstilling, som er videreført fra 5. semestersprojektet "Diagnosticering af Normaltrykshydrocephalus ved Intrakraniel Trykmåling" [3].

Normaltrykshydrocephalus (NPH) tilhører sygdomsgruppen hydrocephalus, hvor højtrykshydrocephalus er den mest alment kendte og oftest forekommende type.

Navnet hydrocephalus kan oversættes til "vand i hovedet", og navnet refererer til forstørrede hjerneventrikler forårsaget af ophobning af hjernens cerebrospinalvæske (CSF).
Denne ophobning af væske skyldes cirkulationsforstyrrelser af CSF i hjernens ventrikelsystem pga. obstruktion eller utilstrækkelig resorption af væsken. Dette medfører et forhøjet intrakranielt tryk (ICP) [14].
Et forhøjet tryk i hjernen kan ødelægge hjernevævet og medføre atrofi. Når først nervevæv i hjernen er ødelagt, er det ikke muligt at gendanne, og det har derfor store konsekvenser for patienten, hvis ikke denne degenerering forhindres [17]. Med forhøjet ICP falder det cerebrale perfusionstryk (CPP)[1], og hjernens gennemblødning mindskes [14].
Hydrocephalus er i de fleste tilfælde en livslang sygdom, men behandling i form af drænage, ved indoperering af ventil, tillader ofte patienterne at leve et normalt liv. Ubehandlet har patienter med hydrocephalus en mortalitet på 80% se vedlagt CD-ROM [CD, A.1.1] .
Der findes en redegørelse for højtrykshydrocephalus i 5. semestersprojektet [3, s. 83-87] .

NPH er en type hydrocephalus, som oftest rammer ældre mennesker over 60 år.
Incidensen af NPH er ca. 1 af 100.000 personer pr. år, og der er således diagnosticeret 4 tilfælde af NPH på Aalborg Sygehus i 2001 [1]. Karakteristiske symptomer på NPH er gangforstyrrelser (ataktisk gang), urininkontinens, demens og depression [22].
NPH menes at skyldes væskedynamiske og vaskulære abnormaliteter, indbefattet forringet absorption af CSF i sinus sagitalis superior gennem villi arachnoidalis og lav kompliance af hjernevævet. Til forskel fra højtrykshydrocephalus, hvor ICP er konstant forhøjet, har NPH-patienter et i perioder forhøjet tryk med et normalt til lidt forhøjet baselinetryk. Det periodevist forhøjede tryk kan identificeres som B-bølger [10] [9] [14] se yderligere beskrivelse i [3, s. 8-20, 92-94] .

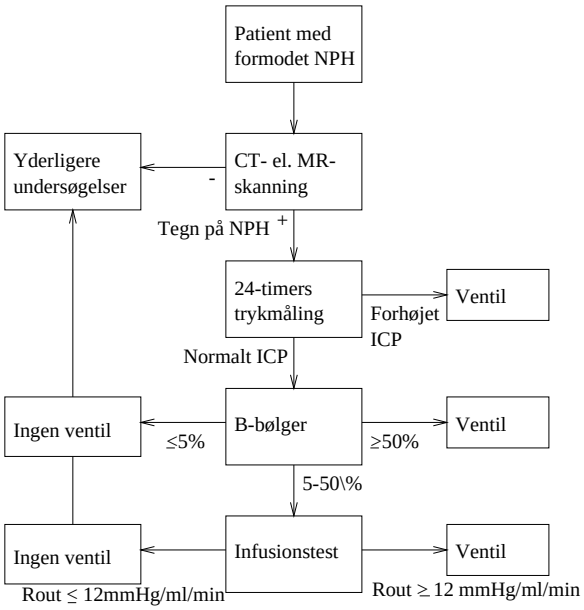
Det er svært at diagnosticere NPH, fordi der findes en lang række differentialdiagnoser med lignende symptomer, og fordi sygdomsbilledet er uklart og diagnosticeringsmetoderne subjektive. Det er især svært at stille diagnosen, hvis der er tale om idiopatisk NPH. Diagnosticeringsmetoderne har derfor lav sensitivitet, hvilket betyder, at ikke alle syge patienter bliver identificeret af testen.
Dette er et problem, idet behandling med en ventil kan være forskellen på et døgnovervåget liv

[1] CPP=MAP-ICP, hvor MAP er Mean Arterial Pressure

på plejehjem med tiltagende demens eller muligheden for at klare sig selv i eget hjem [3, s. 8-20] .

Indoperering af ventil er ikke uden komplikationer. I 1965 da diagnosen først blev stillet, fik alle patienter med umiddelbare tegn på NPH indopereret ventil. Dengang sås en forbedring hos 24-33%, komplikationer hos op til 44% og en dødelighed hos op til 9% af patienterne [9]. Denne prognose skal ses i lyset af, at der siden er udviklet utallige nye ventiler. Alligevel eksisterer der stadig de samme problemer med alle ventiltyper, eksemplevis infektioner og tilstopning jf. CD-ROM [CD, A.1.1] .
Ved indoperering af ventil opleves i dag bedring hos ca. 60% af patienterne med idiopatisk NPH. Komplikationer som følge af ventilbehandling indenfor det første år, vil kunne forventes i op til 40% af alle ventiloperationer, og yderligere operation i op til 22% af tilfældende. Ca. 5% vil opleve store neurologiske problemer eller vil dø [4] [25] [20].

På Rigshospitalet, København er der udviklet en standard for, hvordan sygdommen kan diagnosticeres og denne ses i figur 1.1. Figuren viser diagnosticeringen, som den i dag foretages på neurokirurgisk afdeling AAS. Retningslinierne er baseret på mængden af B-bølgeaktivitet over



Figur 1.1: *Diagrammet angiver Rigshospitalets retningslinier for, hvordan diagnosticering af NPH kan foretages [14] Metoden er baseret på mængden af B-bølger over en 24-timers monitoreringsperiode og på R_{out} målt ved infusionstest. På Aalborg sygehus skal R_{out} være >16 mmHg/ml for, at der gives ventil*

en 24-timers monitoreringsperiode og på infusionstest [14].
Infusionstesten angiver en værdi for modstanden mod udløb af CSF R_{out} [5]. Denne metode benyttes sekundært på Aalborg Sygehus [3, s. 33] . Der gives imidlertid kun ventil, hvis R_{out}> 16 mmHg/ml [3, s. 25] . ICP-monitorering over en 24-timers periode klargør, om der er karakteristiske B-bølger, der indikerer NPH. Grænsen for, om B-bølger anses for værende indikation for indoperering af ventil, er fra Rigshospitalet sat til at være mere end 50% af måleperioden. Denne grænse er sat udfra visuel detekterering af B-bølgeaktivitet [14].

1.1 Initierende problemstilling

Målet er en forbedret diagnostisk test, som gør det muligt kun at operere patienter, som vil få gavn af ventiloperationen. Dermed undgås de risici for komplikationer, der følger en ventiloperation, og fejlbehandling mindskes.

Det er ikke uden risiko at få indopereret en ventil, og det er derfor vigtigt med en diagnostisk test, der sikrer høj specificitet.
Ligeledes har det følger, hvis patienten ikke får indopereret en ventil, hvor denne kunne have været til gavn, og en diagnostisk test skal derfor også kunne sikre høj sensitivitet. Et skridt på vejen mod bedre diagnosticeringsmetoder er at objektivisere diagnosticeringen.

Problematikken leder frem til følgende initierende problemstilling:

Hvordan sikres høj specificitet og sensitivitet ved diagnosticering af NPH-patienter på AAS, så risikoen for at udsætte patienter for unødige konsekvenser i forbindelse med behandling mindskes?

KAPITEL 2

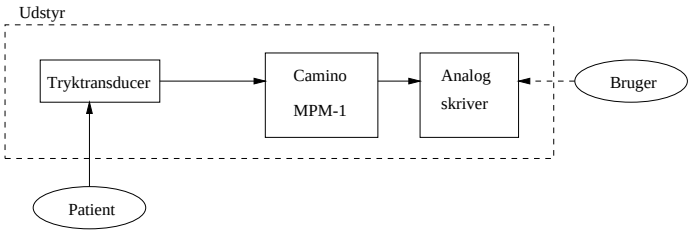
Problemstillinger

Kapitlet indeholder en beskrivelse af problemstillinger vedrørende det benyttede udstyr til diagnosticering af NPH på neurokirurgisk afdeling, AAS.

NPH-patienter kommer oftest i forbindelse med sygehusvæsenet pga. symptomerne dårlig gangfunktion, inkontinens og/eller demens. Ved mistanke om NPH foretages bl.a. en neurologisk udredning af patientens tilstand, og der tages en taptest af CSF for at mindske ICP og vurdere, om dette medfører en kortvarig forbedring af patientens tilstand. Hvis der er tale om NPH, skal alle tre karakteristiske symptomer være tilstede, og demens skal være i et tidligt stadie.
Hvis der efter de indledende undersøgelser stadig er mistanke om NPH, henvises patienterne til neurokirurgisk afdeling, hvor ICP-monitorering og infusionstest foretages. Det er udstyret til diagnosticering benyttet på denne afdeling, der er i fokus i dette projekt.

2.1 Beskrivelse af udstyr

Udstyret, som i dag benyttes ved diagnosticering af NPH på neurokirurgisk afdeling, består af en Camino MPM-1 intrakraniel trykmåler og en analog printer til udskrivning af tryksignalet. Brugerne af dette system er sygeplejersker og læger. Systemet ses i figur 2.1.



Figur 2.1: *Viser oversigt over udstyret benyttet til monitorering af ICP, som det forefindes på afdelingen i dag.*

2.2 24-timers ICP-måling

Ved en 24-timers ICP-måling på AAS, får patienten placeret en intrakraniel tryktransducer frontalt i kraniet, som tilkobles Camino MPM-1 jf. appendiks B. Patienten skal efterfølgende forblive sengeliggende i hele måleperioden for ikke at interferere ICP-signalet, som registreres af Camino MPM-1 og plottes på en papirudskrift. Løbende kontrolleres patienten og udstyret af

sygeplejersker, som noterer årsager til bevægelsesrelaterede trykændringer på kurven. Lægen kan efter et døgns overvågning diagnosticere patienten ud fra ICP-kurven.

2.2.1 Problemstillinger omkring 24 timers ICP-måling

De centrale problemer på afdelingen med hensyn til diagnosticeringsprocessen er arbejdet omkring printeren og diagnosticering ud fra den printede trykkurve. Dette uddybes følgende:

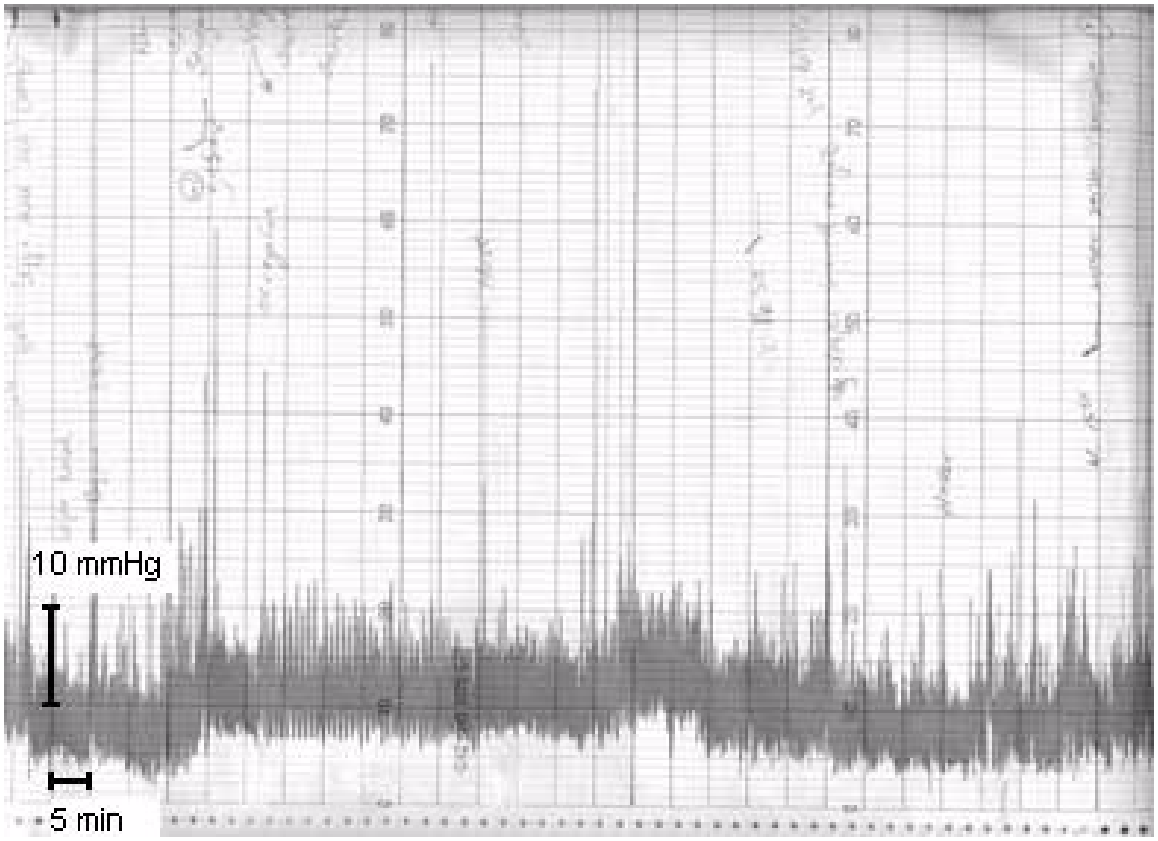
Notering på printet kurve
Sygeplejerskens interaktion med udstyret er at notere patientens lejring, når patienten er urolig, om patienten sover eller er vågen og angive, hvis hovedgærdet eleveres eller sænkes. Denne notering har betydning for at kunne eliminere fejlkilder under lægens efterfølgende diagnosticering [3, s. 27-28,63-64] .
Der er imidlertid i dag ingen standardiserede regler for, hvordan denne notering skal foregå, dvs. *hvad* der skal påskrives kurven og *hvor ofte*. Den nødvendige overvågning for at kunne foretage disse noteringer gør samtidig systemet løntungt.

Dataopløsning vs. kurvelængde
Der er et skisma mellem at få udskrevet en kurve med en længde, der er mulig at overskue ved diagnosticeringen og en udskrift med høj nok opløsning til at muliggøre identifikation af de for diagnosen betydende B-bølger. I dag udskrives kurven med 12 cm/time (2mm/minut) resulterende i en kurve, der efter 24 timers monitorering er ca. 3 meter lang. Der er på afdelingen problemer med både at overskue kurve og identificere bølger [3, s. 41-53]

Datapræsentation
Tiden og trykniveauet defineres kun i starten af målingen, hvilket gør det besværligt for brugeren at aflæse disse.

Subjektiv diagnose
En subjektiv diagnose medfører lille reliabilitet og validitet af testen.
Diagnosticeringen foretaget ud fra papirudskriften er subjektiv. Dette skyldes problemer med at identificere B-bølger, foretage en samlet vurdering af B-bølgeaktivitet på kurven og uoverensstemmelser omkring definitionen af B-bølger for de enkelte læger. Det kræver erfaring at kunne genkende de rigtige mønstre på kurven svarende til B-bølger, og desuden er der ikke enighed om, hvilken periode diagnosticeringen foretages over. Dette kan have betydning, idet det med statistisk signifikans er påvist, at der forekommer flere B-bølger i søvnperioder [11]. For at eliminere subjektivitetens betydning begrænses foretagelsen af diagnosticeringen på afdelingen til at afhænge af samme læge. Dermed begrænses udbredelse af viden og erfaring, hvilket indebærer en lille fleksibilitet i organisationen [3, s. 41-53] . Figur 2.2 viser et uddrag af den udskrift lægen benytter til at stille diagnosen.

Arbejdet omkring printeren
Med hensyn til den analoge printer opleves skæv papirfremføring, problemer vedrørende udskiftning af blækpatron samt for patienterne generende støj fra printerens skrivehoved.
Generelt opleves udstyret som stort og klodset, og gør sygeplejerskens arbejde omkring patienten

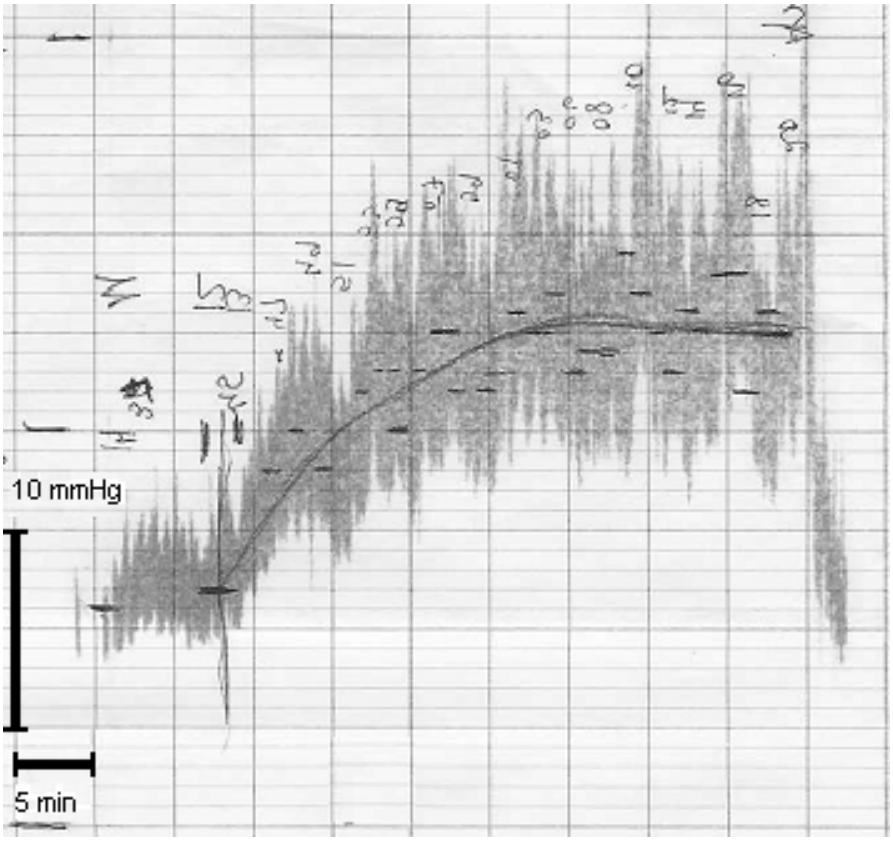


Figur 2.2: *Uddrag af en printet papirudskrift med sygeplejerskernes notater. Kurven er udskrevet med 12 cm/timen. x-aksen viser tiden og y-aksen trykket i mmHg.*

besværligt.
Den udprintede papirkurve giver yderligere ingen mulighed for dataprocessering [3, s. 28-29] , og printeren skal kalibreres inden hver måling.

2.3 Infusionstest

Infusionstest foretages i dag på AAS ved to forskellige metoder. Hvis patienten allerede har fået indlagt en ICP-trykmåler, tilkobles patienten til samme printer, som benyttes ved 24-timers ICP-måling. Er der ikke lagt en trykmåler, kan der tilkobles et manometer til en lumbalpunktur, hvorigennem lægen infuserer væske og løbende aflæser og noterer ICP-værdierne. I begge metoder infuseres der igennem lumbalpunkturen en konstant mængde væske. En metode som kaldes Katzman test. ICP-værdierne registreres gennem hele målingen, der normalt varer mellem 30 og 60 minutter indtil der opnåes et plateau eller trykket stiger faretruende [5] [2]. Infusionstestens indikation for ventiloperationen er den diagnostiske parameter R_{out}, som angiver modstand mod udløb af CSF-væske. Det vil sige jo højere R_{out}, jo større modstand mod CSF udstrømning og jo større indikation på nedsat CSF-absorption.
Ved målingen skal opnås et plateau for at kunne beregne R_{out}. Figur 2.3 viser en infusionstest med lægens notater og fittede kurve.



Figur 2.3: *Viser den printede kurve for en infusionstest med lægens markeringer af gennemsnitstrykket aflæst på Camino-trykmåleren og fittede kurve. X-aksen viser tiden og Y-aksen ICP-trykket i mmHg. ICP_{start} er 12 mmHg og ICP_{slut} er 25 mmHg. Hele testen har taget ca. 45 min..*

2.3.1 Problemstillinger omkring infusionstesten

De centrale problemer med hensyn til beregningen af R_{out} er den subjektive diagnose og vurdering af slutplateau under målingen. Dette uddybes følgende:

Subjektiv diagnose
Det er besværligt at fastlægge ICP start- og stopniveau, idet det målte signal har et lille signal-støjforhold, som det ses på kurven på figur 2.3. Lægen er nødsaget til løbende at plotte midlede trykværdier på kurven og fitte en kurve for at se plateauer. Dette medfører en subjektiv vurdering af den registerede måling, som kan variere fra læge til læge [3, s. 79] .

Målingen
Hos visse patienter ses det under infusionstesten, at trykket stiger faretruende højt, så plateau ikke kan opnås. Dette medfører, at lægen ikke kan beregne R_{out}, og er nødsaget til at stoppe målingen og eventuelt påbegynde en ny måling med mindre infusionsrate.

2.4 Problemformulering

På baggrund af de foregående problemstillinger opstilles en problemformulering, der fokuserer på en teknologisk løsning.

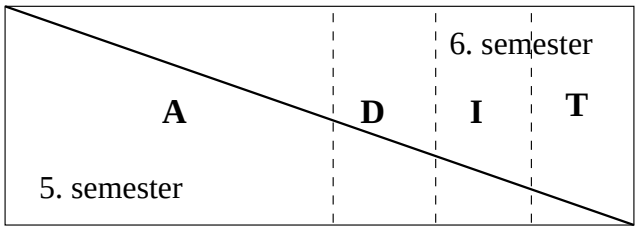
> **Der ønskes udviklet et system, der digitaliserer, visualiserer og behandler det intrakranielle tryksignal. Formålet er et mere anvendeligt og brugervenligt system, der objektiviserer 24-timers ICP-monitorering og infusionstest.**
> **Systemet udvikles med henblik på implementering på neurokirurgisk afdeling AAS.**

KAPITEL 3

Metodebeskrivelse

Kapitlet beskriver, hvordan rapporten er opbygget ud fra udviklingsværktøjet UML

Projektet fra 5. semester [3] bestod af en analyse af diagnosticering af NPH på AAS. Temaet for dette semester var "Teknologi i sundhedssektoren" og projektet skulle derfor analysere og dokumentere problemstillingen med det formål at udarbejde en kravspecifikation til et medikoteknisk udstyr. Efter 5. semester var det oplagt at gå videre med problemstillingen og her designe og udvikle produktet. Grænsen mellem 5.- og 6. semesters-projekterne illustreres i figur 3.1 ved den diagonale skæring i rektanglet.



Figur 3.1: *Figuren illustrerer grænsen, hvor 5.semesters-projektet slutter og 6.semesters-projektet forsætter. Arealet af figuren for hvert semester skal læses som den tid, der benyttes på A=analyse, D=design, I=Implementation og T=test.*

Figuren er delt op i fire udviklingsfaser: Analyse, Design, Implementering og Test. Disse fire udviklingsfaser illustrerer, hvor de to projekter mødes, og hvor ressourcerne er lagt på det pågældende semester.
På nærværende semester, hvor temaet er at udvikle et medikoteknisk udstyr, tages der udgangspunkt i den højre halvdel af rektanglet, hvor kravspecifikationen fra 5. semester revurderes.
Det medikotekniske udstyr består i at udvikle et softwaresystem, der kan vejlede lægen til den rigtige diagnose af NPH.

3.1 Udviklingsværktøjer

UML er et modelleringssprog, der anvendes til at specificere, visualisere, konstruere og dokumentere software-systemer og kan anvendes i alle fire faser af systemudviklingen. Analysedelen inddeles til også at indeholde en requirementanalyse, og processen kan derfor ses som bestående af fem faser. For yderligere beskrivelse af UML-terminologi jf. appendiks A og [21].
Til at modellere UML-diagrammerne i de forskellige udviklingsfaser anvendes det objektorienterede program Together [28]. Implementeringsfasen kræver et objektorienteret programmerings-

sprog, og softwareløsningen er derfor udviklet i Java. Valget er faldet på Java, da der fra studiets side har været undervisning i sproget.

3.2 Requirementanalyse

Kravene fra 5. semester revurderes i samarbejde med primærbrugeren af systemet, og på baggrund heraf udvikles et UML use-case diagram. Dette diagram angiver aktørerne i systemet og beskriver de funktionaliteter, systemet skal levere og interaktionen mellem system og aktører.

3.3 Analyse

Ud fra det modellerede use-case diagram beskrives det, hvordan problemstillingerne skal håndteres af systemet, og hvordan systemet skal opføre sig for, at det har den ønskede funktionalitet. Dette gøres ved at modellere systemet i klasse-, sekvens- og aktivitetsdiagrammer, der ikke baseres på tekniske detaljer men på problemområdet. Analysen foregår i to faser, hvor den første fase modellerer hele systemet i form af aktivitetsdiagrammer ud fra det opstillede use-case diagram. Systemet afgrænses herefter til at bestå af et passende antal funktionaliteter, der kan nå at blive implementeret i projektperioden. Fase 2 består i at beskrive det afgrænsede system yderligere ved sekvens- og klassediagrammer.

3.4 Design

I designfasen videreudvikles UML-diagrammerne fra analysefasen til tekniske detaljerede modeller, der muliggør implementation ved en softwareløsning. Diagrammerne som udvikles i denne fase er klasse-, sekvens- og aktivitetsdiagrammer. I designfasen opstilles kvalitetsfaktorer på baggrund af analysen. Disse kvalitetsfaktorer inddrages i den videre designfase. Designfasen bliver anskuet i to dele hhv. komponent- og procesarkitektur for systemet. Det er vigtigt at understrege, at designfasen ikke er endt efter, at de forskellige diagrammer er opstillet, da design- og implementationsfaserne er iterative processer, som løbende revurderes.

3.5 Implementering

I implementeringsfasen skrives den endelige kode til systemet. Det opnås ved at tage de sidste designbeslutninger og oversætte designdiagrammerne til syntaksen i Java.

3.6 Test

I testfasen foretages enhedstest, integrationstest og systemtest som beskrevet i appendiks F. Afsnittet vil ikke beskrive testprocessen, men vil beskrive udvalgte eksempler på, hvad der er blevet testet, og hvordan det er blevet testet.

Del II

ANALYSE

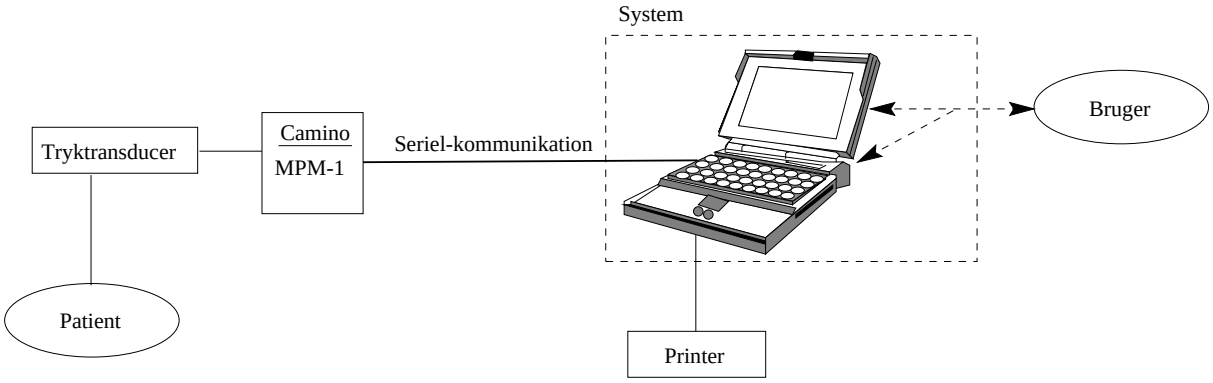
KAPITEL 4

Requirementanalyse

I systemudviklingen anvendes Requirementanalysen til at modellere og dokumentere en fælles enighed mellem brugerens forventninger og krav til softwaresystemet. Ved hjælp af use-case diagrammer skabes der et overblik over systemet, der præsenteres for brugeren.

4.1 Systembeskrivelse

Der ønskes udviklet et system, der kan opsamle trykmålinger ved input fra en tryktransducer. Det udviklede system er et softwareprogram, som skal ligge på en bærbar computer. Systemet er vist i figur 4.1. Figuren angiver desuden systemets relationer, som er interaktion med en bruger,



Figur 4.1: *Viser softwaresystemet, der ønskes udviklet i projektet samt relationer til bruger og en trykmåler*

en Camino MPM-1 trykmåler og en printer.
Tryksignalet analyseres på computeren, der via en trykmåler er forbundet til en tryktransducer. Nærmere beskrivelse af Camino MPM-1 og tryktransducer forefindes i appendiks B og på CD-ROM [CD, A.1.3] .
Computerens skærm og tastatur benyttes som brugergrænseflade og til brugerkommunikation.

4.2 Krav til systemet

Kravene til systemet i 5. semesters-projektet [3] revurderes i samarbejde med overlæge Preben Sørensen, neurokirurgisk afdeling K, AAS via et use-case diagram. Dette har ført til en diskussion, hvor kravene er blevet diskuteret, og de endelige krav er sat til følgende:

- Systemet skal fungere med den på afdelingen benyttede Camino MPM-1 trykmåler.
- Der skal som minimum gives adgang til samme informationer, som trykkurven i dag giver adgang til, dvs. de frekvenser i signalet som benyttes ved vurdering af kurve [3, s. 59,69] samt den information der formidles gennem notater på papirkurven.
- Signalet skal præsenteres ved en opløsning på 30 cm/t, da brugeren har defineret denne hastighed som optimal for en visualisering af signalet [3, s. 63] .
- Systemet skal standardisere definitionen af B-bølger og foretage udregningen af mængden af B-bølgeaktivitet over hele monitoreringsperioden.
- Det er vigtigt, at systemet sikrer brugervenlighed [3, s. 69] . Programmets brugergrænseflade skal derfor være menu- og/eller ikonbaseret, og der skal være adgang til en dansk brugermanual. Desuden skal programmet kunne håndtere alle inputs fra et dansk tastatur, og samtlige beskeder inkl. fejlmeddelelser skal være på dansk.
- Systemet skal kunne visualisere ICP under infusionstest og efterfølgende beregne R_{out}.

4.3 Krav til hardware

- Trykmålingen skal foretages med en Camino MPM-1 med minimum software version 1.32 for at muliggøre seriel kommunikation.

Sampling af signal

- Signalet skal samples således, at alle patologiske og fysiologiske trykbølger kan rekonstrueres. Den hurtigste frekvens i signalet af relevans for diagnosen er pulsen, der i hvile er $<$120 slag/min svarende til 2 Hz [24]. Samplingshastigheden skal derfor ifølge Nyquist samplingsteori være minimum 4 Hz.
- Data skal gemmes løbende på harddisken, så data ikke mistes, hvis strømmen pludselig afbrydes eller computeren af anden grund går ned.

4.4 Krav til sikkerhed

Da systemet skal tilkobles en patient, opsættes der krav til elektrisk sikkerhed for systemet. Desuden undersøges det, hvilke regler der skal overholdes for, at patientrelateret data beskyttes.

4.4.1 Elektrisk sikkerhed

Sikkerhedskravene der ønskes opfyldt er defineret i Elekricitetsrådets Stærkstrømbekendtgørelse Afsnit 135-1, Elektromedicinsk udstyr, 1996. Denne er en oversættelse af den internationale publikation; Medical electrical equipment, IEC Publication 60601-1.

Patienten vil kun være i berøring med tryktransduceren, som er et fiberoptisk system. Intgra NeuroCare dokumenterer, at systemet er elektrisk isoleret [CD, A.1.3] .
Der skal yderligere tages hensyn til kravene i valget af computer systemet skal fungere på.

4.4.2 Datasikkerhed

I forbindelse med den tiltagende anvendelse af IT i sundhedssektoren, er der fra Sundhedsstyrelsen udgivet en IT-sikkerhedsvejledning, der skal sikre, at der på de danske sygehuse opnås et højt IT-sikkerhedsniveau [32]. Deslige er der fra Nordjyllands Amt opstillet en række retningslinier, der skal overholdes på amtets sygehuse jf. bilag [CD, A.1.2] .

For nærværende projekt skal det tilstræbes, at de retningslinier, der i dag benyttes og følges på AAS, skal anvendes og implementeres i systemet.

Kravene til systemet bygger på interview med Susanne Petersen, IT Sundhed afd. AAS, samt retningslinier beskrevet i bilag [CD, A.1.2] og [32].

Retningslinierne gør det klart, at hvis et system indeholder personhenførbare oplysninger, skal systemet sikres med personligt login, og alle data skal gemmes på det pågældende sygehus server. Dette gøres for at sikre sig mod tab af data, idet computeren kunne blive stjålet eller gå i stykker. Hvis systemet skal indeholde oplysninger som navn og CPR-nr, skal systemet kobles op på sygehusets netværk, og der skal oprettes personligt brugernavn og password til samtlige sygeplejersker og læger på afdelingen.

4.5 Aktørbeskrivelser

Aktører repræsenterer roller, som de forskellige brugere af systemet eller tilkoblet udstyr kan påtage sig.

De personlige brugere af systemet er aktive aktører, og inkluderer læger og sygeplejersker på neurokirurgisk afdeling, AAS. Både overlæger og 1. reservelæger skal benytte systemet og kunne stille en diagnose. Det vil være lægerne, som starter og stopper programmet og kobler udstyret til patienten. Sygeplejersker vil løbende indskrive kommentarer vedr. patientomstændigheder.

I det følgende beskrives de forskellige aktører, samt de brugsmønstre eller arbejdsopgaver de er involverede i.

Aktør Sygeplejerske
Sygeplejersken er tiltænkt to brugsmønstre i systemet. Der er tale om en personlig aktør, som inden 24-timers ICP-måling skal registrere patienten i systemet og under 24-timers ICP-måling og infusionstest skal overvåge, at alt forløber normalt. Desuden skal sygeplejersken tilføje relevante kommentarer til målingen.

Aktør Læge
Lægen har flere brugsmønstre i systemet. Der er tale om en personlig aktør, som skal registrere patienten til enten 24-timers ICP-måling eller infusionstest.
Yderligere skal lægen analysere kurven fra ICP-målingen og infusionstesten og stille en diagnose.

Aktør Superbruger
Der er tale om en personlig aktør, som skal administrere brugertilladelser til systemet og kunne ændre i beregningsdefinitioner.

Aktør Camino
Der er tale om en passiv hardwareaktør, der skal sende ICP-data til systemet kontinuerligt.

Aktør Printer
Der er tale om en passiv hardwareaktør, der skal udskrive dokumentation fra systemet.

4.5.1 Sammenhæng mellem aktører og brugsmønstre

Tabel 4.1 angiver hvilke brugsmønstre, det er tiltænkt de forskellige aktører. X angiver en primær rolle, forstået på den måde, at det normalt vil det være denne person, som udfører opgaven. Hvis personen udfører en anden opgave, svarer det til, at personen påtager sig en anden rolle. I tabellen er der ikke medtaget de passive hardware-aktører *Camino* og *Printer*.

Aktør / Opgave	Patient-registrator	ICP-overvåger	Data-behandler	Infusions-overvåger	Admini-strator
Sygeplejerske	X	X			
Læge	X		X	X	
Superbruger					X

Tabel 4.1: Oversigt over aktører, der anvender systemet samt deres brugsmønstre.

4.6 Systemets funktionalitet

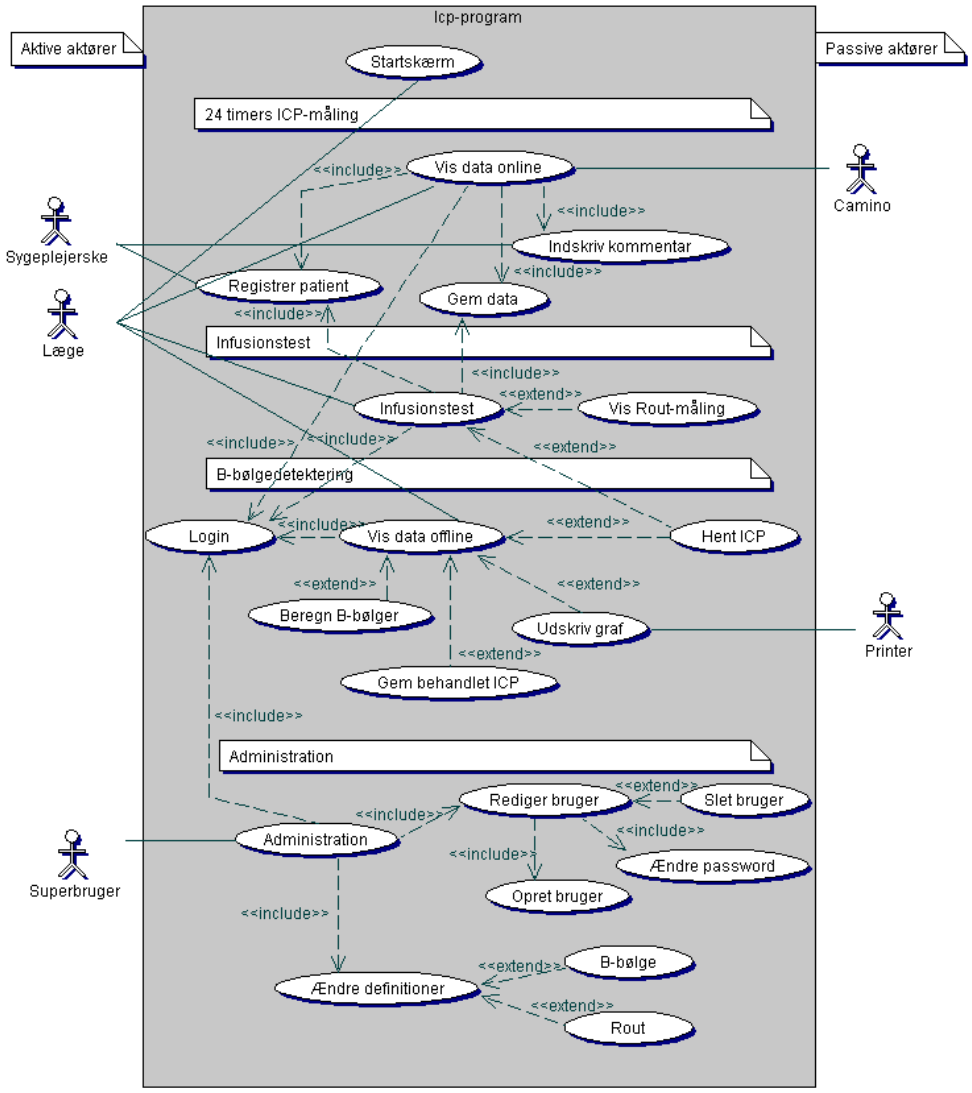
Systemets funktionaliteter er fundet og beskrevet ved samtaler med de kommende brugere af systemet samt analysearbejde på 5. semester. Funktionaliteterne er opstillet i et use-case diagram, for at overskueliggøre de forskellige brugsmønstre og ses i figur 4.2.

Use-case diagramet i figur 4.2 viser, hvordan aktørerne initialiserer de nødvendige funktionelle use-cases for at opnå den ønskede funktionalitet af systemet. I use-case diagrammet anvendes <<*include*>> og <<*extend*>> for at visualisere inkluderinger eller udvidelser af enkelte funktionaliteter. For yderligere information omkring use-case notation henvises til appendiks A.
Funktionelle use-case beskrivelser for hver use-case beskrives følgende.

4.6.1 Use-case beskrivelser

Under de funktionelle use-casebeskrivelser angives use-cases i citationstegn og delsystemer vil fremgå understregede. Alle use-cases i diagram 4.2 beskrives i logisk rækkefølge.

Login
Use-case "Login" ses midt i diagrammet. For at beskytte persondata er det nødvendigt med et login. Brugeren skal opgive godkendt brugernavn og password for at anvende programmet.



Figur 4.2: *Viser et use-case diagram over systemet godkendt af den primære bruger. Diagrammet udtrykker systemets problemområder og «include» og «extend» beskriver, om det er funktionaliteter, der inkluderes eller udvides.*

Startskærm

Ved opstart af programmet skal der vises en startskærm med mulighed for at vælge, om der skal fortages en 24-timers ICP-måling, detekteres B-bølger på en allerede optaget måling, laves infusionstest, eller om indstillingerne i programmet skal ændres.

Registrer patient

Inden Infusionstest eller 24-timers ICP-måling initialiseres af lægen/sygeplejersken ved påbegyndelse af en måling, skal lægen/sygeplejersken registrere patienten ved navn og CPR-nr. Det skal være muligt for brugeren til enhver tid at komme ud af "Registrer patient" uden at have oprettet patienten.

24 timers ICP-måling

Vis data online

Efter at use-case "Registrer patient" er blevet udført, skal use-case "Vis data online" initialiseres. Use-casen skal give brugeren mulighed for at starte en 24-timers ICP-måling, der visualiserer ICP-signalet som en graf på skærmen. På grafen skal starttidspunktet for målingen vises, og tiden skal løbende angives.

Indskriv kommentar

Sygeplejersken skal under visualiseringen af ICP-signalet have mulighed for at indskrive kommentarer på grafen. Dette skal forekomme, hvis patienten laver bevægelser, der influerer målingen. Kommentarerne kan være prædefinerede eller selvvalgte, og skal på grafen markere det tidspunkt, hvor patientens bevægelse har fundet sted. Kommentarer og tilhørende tidspunkter skal gemmes i en fil.

De prædefinerede kommentarer og deres betydning fremgår af tabel 4.2, og er defineret på baggrund af krav i 5. semesters rapporten [3].

Kommentar	**Uddybende forklaring**
Uro	Taler, griner, spiser e.l.
Sover	Rolig med lukkede øjne
Vågen	Åbne øjne og kontaktbar
Sidder	Sengegærdet eleveres over ca. 45° fra vandret
Ligger	Sengegærdet sænkes til vandret
Tilkoblet	Transduceren tilkobles og ICP registreres
Frakoblet	Transduceren frakobles og der registres intet ICP

Tabel 4.2: Tabellen viser de prædefinerede kommentarer, der skal være tilgængelige på brugergrænsefladen under en 24-timers ICP-måling. Desuden defineres, hvornår kommentarerne skal benyttes

Gem data

Samtidig med at ICP-signalet visualiseres på skærmen, skal de opsamlede ICP-data omregnes fra ASCII-kode og gemmes i en fil, således det efterfølgende er muligt at se målingen.

Infusionstest

Infusionstest

Efter at use-case "Registrer patient" er blevet initialiseret, skal det være muligt at initialisere use-case "Infusionstest". Denne use-case skal give brugeren mulighed for at starte en infusionstest, der visualiserer ICP-signalet på en graf på skærmen.

Hent Icp

Det skal for lægen være muligt at hente en tidligere infusionstest fra en fil, som R_{out}-beregningen skal foretages på. Den hentede fil skal visualiseres på en graf.

Vis R_{out}-måling

Når lægen er parat til at starte målingen, skal use-case "Vis R_{out}-måling" initialiseres. Denne use-case skal visualisere ICP-signalet sammen med det beregnede signal. Grafen skal vises realtime på skærmen.

Det skal til enhver tid være muligt for lægen at stoppe målingen. På grafen skal starttidspunk-

tet for målingen vises. Hvis graferne overstiger en vis grænse, skal lægen gøres opmærksom på situationen.

B-bølgedetektering

Vis data offline

Denne use-case skal give lægen mulighed for at analysere signalet over en 24-timers ICP-måling. Dette gøres ved at visualisere det målte signal med eventuelle kommentarer og giver mulighed for at foretage beregninger på B-bølger. Beregningen skal visualiseres på grafen.

Hent ICP

"Hent ICP" skal sætte lægen i stand til at hente en gemt 24-timers ICP-måling og se signalet på skærmen.

Beregn B-bølger

Lægen skal kunne initialisere funktionaliteten "Beregn B-bølger" efter at "Hent ICP" har visualiseret ICP-målingen. Mængden af B-bølger i ICP-signalet skal beregnes.

Rediger B-bølger

Hvis lægen ikke er enig i beregningen af B-bølger, skal der her kunne tilføjes eller slettes markeringer af bølgerne på grafen. Herefter skal den *nye* mængde bølger angives som procentsats.

Gem behandlet data

Lægen skal, ved at initialisere use-case "Gem behandlet data", kunne gemme det behandlede data, så det ikke overskriver det oprindelige signal.

Udskriv graf

Gennem use-case "Udskriv graf", skal lægen have mulighed for at få en udskrift af signalet.

Administration

Ændre definitioner

I use-case "Ændre definitioner" skal det være muligt for superbrugeren, ved benyttelse af password, at ændre definitioner til de beregninger der udføres af programmet.

B-bølger

Denne use-case skal for superbrugeren give mulighed for at ændre parametre for B-bølgedetektion. Det skal til enhver tid være muligt at ændre parametrerne tilbage til de oprindelige indstillinger.

R_{out}

Denne use-case skal for superbrugeren give mulighed for at ændre definitioner for beregningen af R_{out}.

Rediger bruger

I use-case "Rediger bruger" skal det være muligt for superbrugeren at redigere i oplysninger mht., hvem der skal have adgang til systemet.

Slet bruger

Superbrugeren skal i denne use-case have mulighed for at hente en liste af brugere. Superbrugeren skal have mulighed for at slette en bruger, så denne ikke mere har adgang til systemet.

Opret bruger

Superbrugeren skal i denne use-case have mulighed for at oprette en bruger, således denne har adgang til systemet.

Ændre password

Superbrugeren skal i denne use-case have mulighed for at nulstille en brugers password i tilfælde af, at det skulle være glemt.

KAPITEL 5

Systemanalyse

Analysen giver en beskrivelse af, hvorledes programmet skal fungere ved brug af aktivitets-, klasse- og sekvensdiagrammer. Der foretages ligeledes en afgrænsning af systemets funktionaliteter.

Analysen af systemet inddeles i to faser, der adskilles af en afgrænsning af det system, som ønskes designet og implementeret i dette projekt.
I første omgang beskrives systemet ved aktivitetsdiagrammer, hvor der inddeles i delsystemer, som bestemmes af funktionaliteten af de enkelte moduler fra requirementanalysen afsnit 4.2 på side 15. I anden omgang beskrives det afgrænsede system med et klassediagram og sekvensdiagrammer, der ligeledes opdeles i delsystemer.

5.1 Aktivitetsanalyse

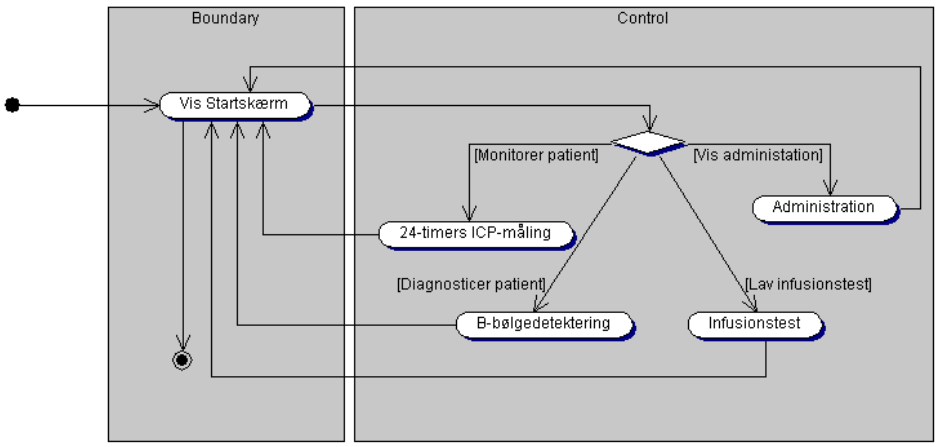
Med udgangspunkt i use-casediagrammet, figur 4.2 på side 19 udarbejdes der aktivitetsdiagrammer over delsystemerne og enkelte use-cases, hvor der vurderes behov for beskrivelse af en funktionalitet ved et diagram. Der beskrives i naturlig kronologisk rækkefølge for almindelig brug af systemet.
Da proceduren for administration af brugere til edb-systemer på AAS ikke er kendt, afgrænses der fra at designe UML-diagrammer for login.
Aktivitetsdiagrammerne er inddelt i *swimlanes* bestående af boundary, control, entity jf. appendiks A.3.

5.1.1 Opstart af program

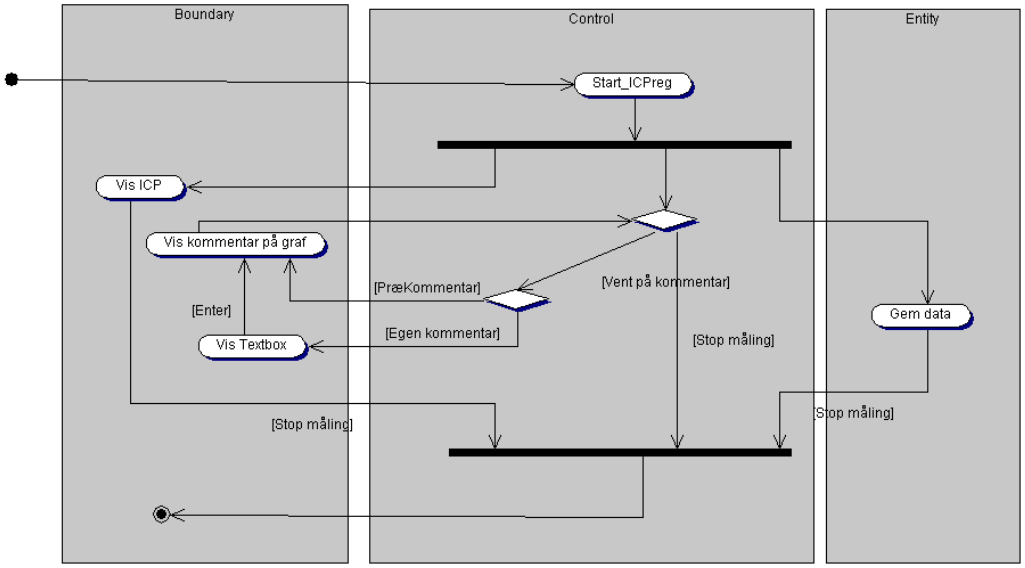
Som det ses illustreret i aktivitetdiagrammet, figur 5.1, vil der, når programmet startes op være fire funktionaliteter at vælge imellem. Aktøren kan vælge 24-timers ICP-måling, og der kan startes en monitorering af patientens ICP. Efter der er foretaget en 24-timers ICP-måling kan lægen vælge B-bølgedetektering og efterfølgende stille en diagnose. Aktøren kan også vælge at lave Infusionstest. Superbrugeren har mulighed for at vælge Administration for at oprette eller redigere brugere samt indstille parametre. Ved afslutning af hver af funktionerne returneres til "Startskærm".

5.1.2 24-timers ICP-måling

På figur 5.2 ses et aktivitetsdiagram over delen, som vedrører 24-timers ICP-måling. Use-case



Figur 5.1: *Aktivitetsdiagram over use-case "Startskærm"*

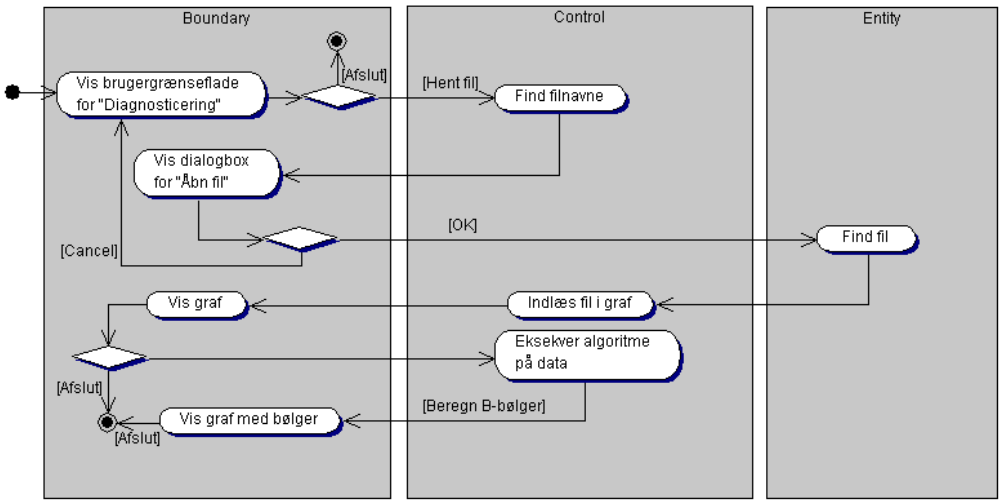


Figur 5.2: *Aktivitetsdiagram over use-casen "24-timers ICP-måling".*

"Vis data online" initialiseres og opsamlingen af ICP kan startes, hvorefter data gemmes i en fil og vises på en graf. Simultant med denne proces er det også muligt at indskrive kommentarer på grafen. Disse kommentarer kan enten være prædefinerede eller være egne indskrivninger.

5.1.3 B-bølgedetektering

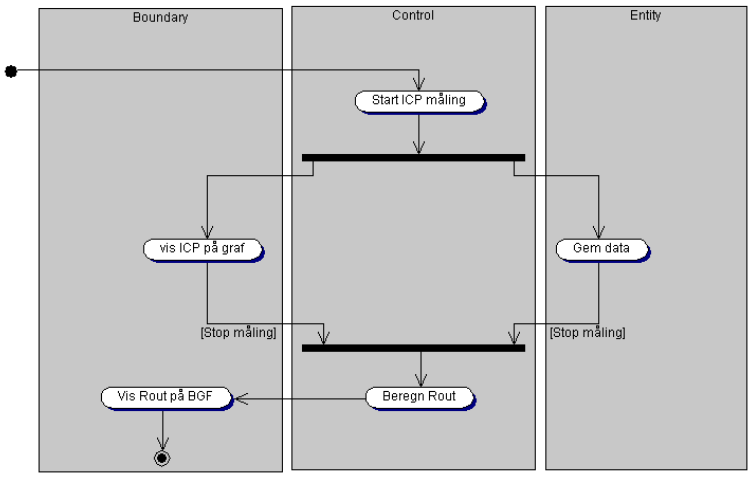
Efter 24-timers ICP-måling er afsluttet, kan lægen foretage en diagnosticering ved detektering af B-bølgerne i signalet. Figur 5.3 viser, hvordan ICP-datafilen for ønskede patienter findes og indsættes i en graf. ICP-data kan herefter viderebehandles, så B-bølgerne kan findes.



Figur 5.3: *Aktivitetsdiagram over B-bølgedetektering*

5.1.4 Infusionstest

Figur 5.4 illustrerer, hvordan infusionstesten startes af lægen, hvorefter data, i en simultan proces, gemmes og vises på en graf. Når målingen stoppes udregnes R_{out} og vises på brugergrænsefladen.

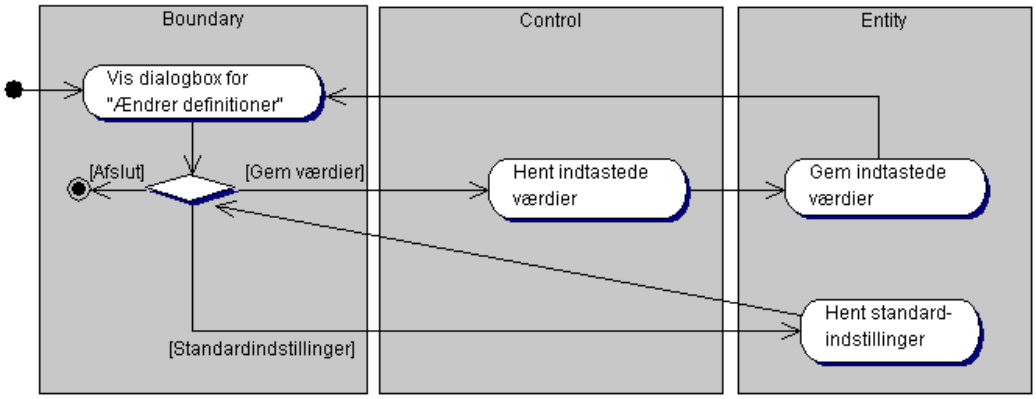


Figur 5.4: *Aktivitetsdiagram over use-case "Infusionstest"*

5.1.5 Administration

Under Administration er der to funktionaliteter. Der kan ændres i detekteringsdefinitioner eller der kan redigeres i brugere. Der vil kun blive fokuseret på use-case "Ændre definitioner" jf. indledningen i afsnit 5.1 på side 23.

Figur 5.5 viser processen for ændring af parametre, hvor der enten kan indtastes nye værdier eller hentes standardværdier.



Figur 5.5: *Aktivitetsdiagram over ændring af definitioner af B-bølger og R_out*

5.2 Afgrænsning af systemet

Systemets funktionaliteter afgrænses, så et fungerende system kan designes og implementeres indenfor semesterets projektperiode.

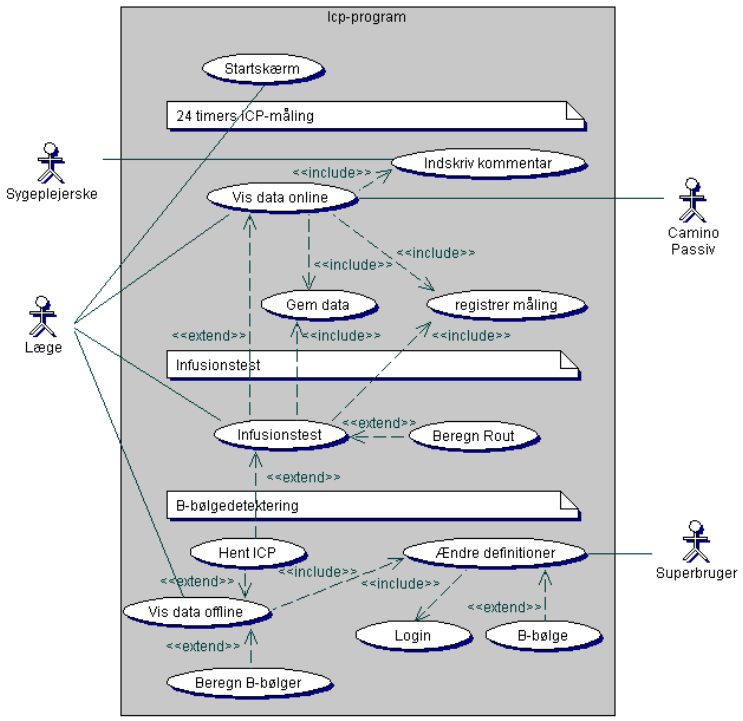
Systemet defineres til at tilhøre forskningsstadiet. Hermed menes, at den digitale opsamling skal give grundlag for yderligere forskning vedrørende NPH samt andre kliniske perspektiver, der involverer B-bølgedetektering og infusionstest. Derfor skal der være mulighed for at opsamle data, at gemme data, at detektere B-bølger samt at beregne R_{out}. Udvikling af en metode til automatisk beregning af R_{out} nedprioriteres, da udvikling har taget udgangspunkt i analysen fra 5. semester [3], som ikke har beskæftiget sig med infusionstest. Derudover har der været mangel på data, hvilket ses som en nødvendighed for at retfærdiggøre udviklingen af en matematisk metode til beregningen af R_{out}. Systemet designes dog således, at lægen kan lave en udregning af R_{out} ved selv at sætte de nødvendige parametre. Hermed kan den analoge printer undværes, og systemet kan overtage alle printerens funktionaliteter.

Projektgruppen vil gerne undgå netværksopkobling, da nærværende projekt er et udviklingsprojekt, og implementering af en netværksopkobling vil være tidskrævende og spild af ressourcer i det første indledende udviklingsforløb. Ifølge reglerne for datasikkerhed er det ulovligt at gemme personhenførbart data på computerens harddisk. Da der derfor ikke vil være personhenførbare data på computeren vælges det at nedprioritere et login-system. I fremtiden, hvis systemet kommer på netværk, skal der naturligvis implementeres et sikkerhedssystem der identificerer brugere via brugernavn og password. Selvom Administration fjernes vælges det dog at videreføre use-case "Ændrer definitioner". Denne beslutning tages, da systemet skal kunne indgå i forskning og det ses derfor nødvendigt at kunne ændre i definitioner vedrørende beregningen af B-bølgeaktivitet.

Til sidst afgrænses der fra at udarbejde en alarmfunktion, da Caminoen allerede tilbyder denne

funktion. Funktionaliteten "Gem behandlet data" afgrænses også, da printerens funktionalitet anses for tilstrækkelig.

Der udarbejdes et nyt use-case diagram, som er en modificering af det første diagram jf. figur 4.2 på side 19.
Det nye use-case diagram, der vil være gældende for design og implementation, ses i figur 5.6.



Figur 5.6: *Nyt use-case diagram, efter afgrænsning af systemet. Diagrammet viser det afgrænsede systems funktionaliteter*

5.3 Analyse af systemet

Der findes klasser i systemet på baggrund af problemområdet og anvendelsesområdet, der er illustreret ved use-case diagrammet i figur 5.6. Metoden for, hvordan klasserne findes er beskrevet i appendiks A.2.
Med udgangspunkt i klassediagrammet udarbejdes sekvensdiagrammer, der giver en mere specifik illustration af interaktionen i systemet med fokus på tid. Metoden for, hvordan sekvensdiagrammer udarbejdes beskrives i appendiks A.4.

5.3.1 Klassediagram

Problemområdet dækker over de elementer, der ønskes styret, overvåget eller kontrolleret, dvs. de elementer, der ønskes håndteret og gemt information omkring. Klasserne for problemområdet er beskrevet i tabel 5.1.

Klasse	Forklaring (problemområde)
OnlineGraf	Håndterer ICP-data ved en grafisk visualisering med ICP som funktion af tiden.
OfflineGraf	Håndterer ICP-data fra fil efter afsluttet ICP-måling.
Patient	Håndterer patientens ICP-data ved at hente og gemme i filer.
IcpData	Henter data fra Caminoen ved seriel kommunikation.
Kommentar	Håndterer indskrivning af prædefinerede og egne kommentarer på grafen.
Parametre	Håndterer detekteringen af B-bølger ved fastsættelse af detekteringsparametrene.

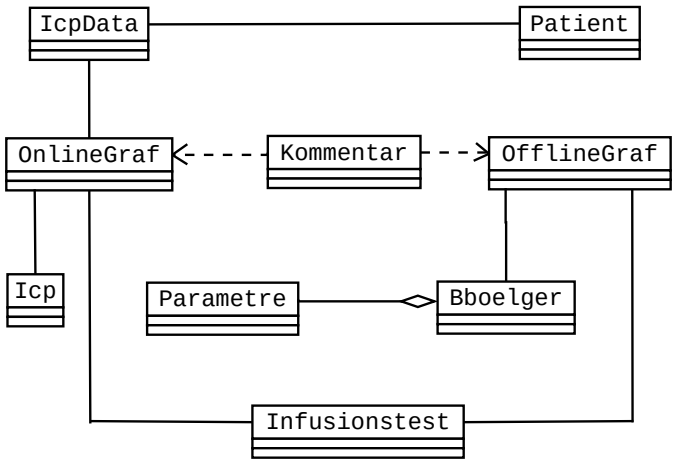
Tabel 5.1: Klasser i problemområdet samt tilhørende forklaring

Anvendelsesområdet for systemet er defineret til de elementer, der udgør grænsefladerne. Klasserne for anvendelsesområdet er beskrevet i tabel 5.2

Klasse	Forklaring (anvendelsesområde)
Icp	Viser den grafiske brugerflade for 24-timers ICP-måling og håndterer tast på knapper.
Bboelger	Viser den grafiske brugerflade for B-bølgedetektering og håndterer tast på knapper.
Infusionstest	Viser den grafiske brugerflade for Infusionstest og håndterer tast på knapper.

Tabel 5.2: Klasser i anvendelsesområdet samt tilhørende forklaring

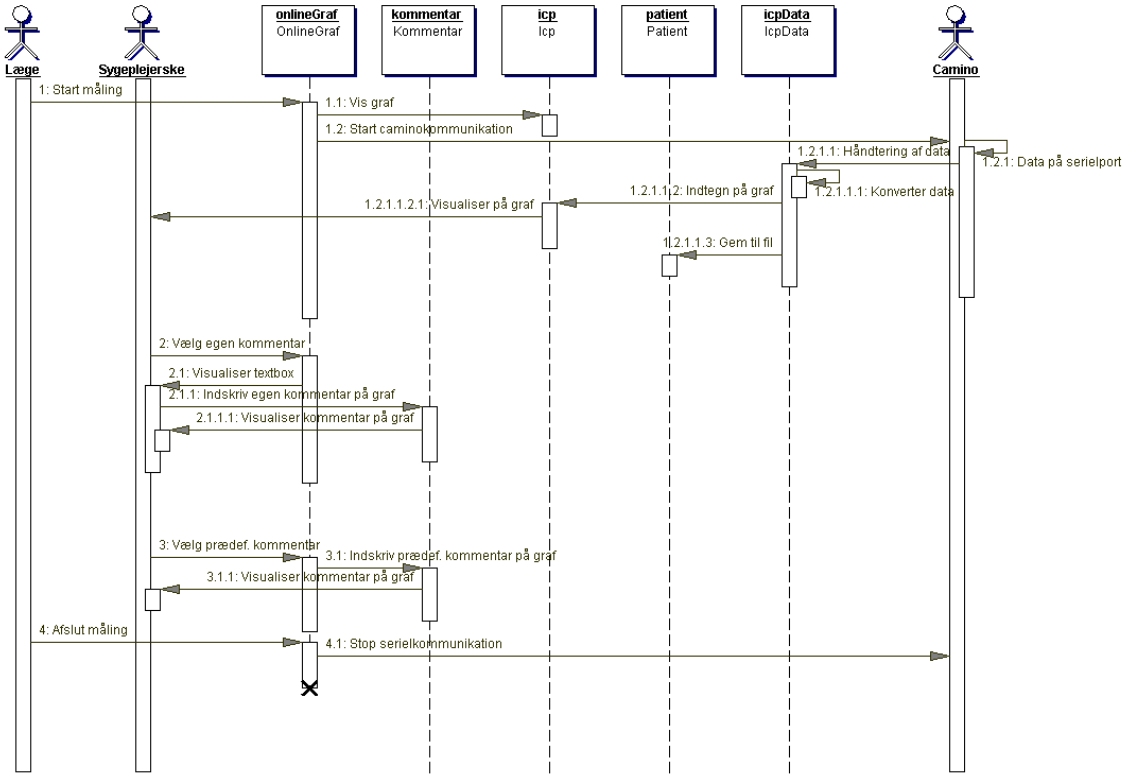
Klasserne fra problemområdet og anvendelsesområdet opstilles i et klassediagram, hvilket fremgår af figur 5.7



Figur 5.7: *Klassediagram der viser relationen imellem klasserne*

5.3.2 24-timers ICP-måling

Interaktionen mellem objekterne i delsystemet 24-timers ICP-måling opstilles i et sekvensdiagram, hvilket ses på figur 5.8.



Figur 5.8: *Sekvensdiagram for 24-timers ICP-måling*

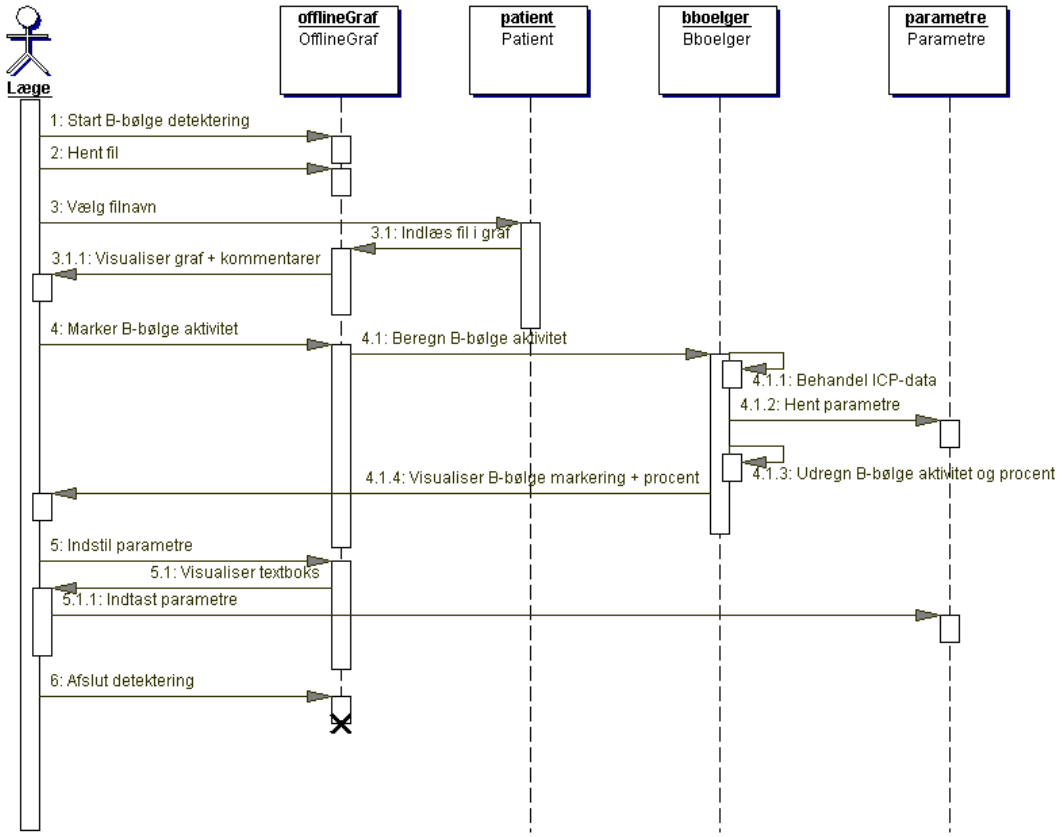
Lægen starter målingen ved tryk på en knap på brugerfladen. Inden målingen kan startes, skal der indtastes et filnavn.
For at undgå de mange krav, der stilles til systemer, som indeholder personhenførbare oplysninger, vælges det at indtaste datoen for målingen, hvormed kravene for datasikkerhed opfyldes. Herved bliver det muligt at finde en gammel måling frem igen, ved at se i patientens journal, hvilken dato målingen blev foretaget på. Dette sikrer, at for at kæde måling og patient sammen *skal* man have adgang til patientens journal.

Efter der er valgt en fil at gemme ICP-data i, oprettes automatisk en kommunikation med Caminoen, hvor data læses på den serielle port. Hver datasample konvertes fra ASCII-kode, således det kan indlæses på en graf, og samtidig skrives til filen.
Det er nu muligt for sygeplejersken at indskrive en kommentar. Der kan vælges mellem en prædefineret kommentar, som er tilgængelig på brugerfladen eller egen kommentar, som kan indskrives via en tekstboks. Kommentarerne gemmes automatisk til det tidspunkt de er skrevet ind.
Efter ca. 24 timer kan lægen stoppe målingen og slukke programmet for efterfølgende at foretage

diagnosticeringen.

5.3.3 B-bølge detektering

For B-bølge detekteringen udarbejdes ligeledes et sekvensdiagram, som kan ses i figur 5.9.



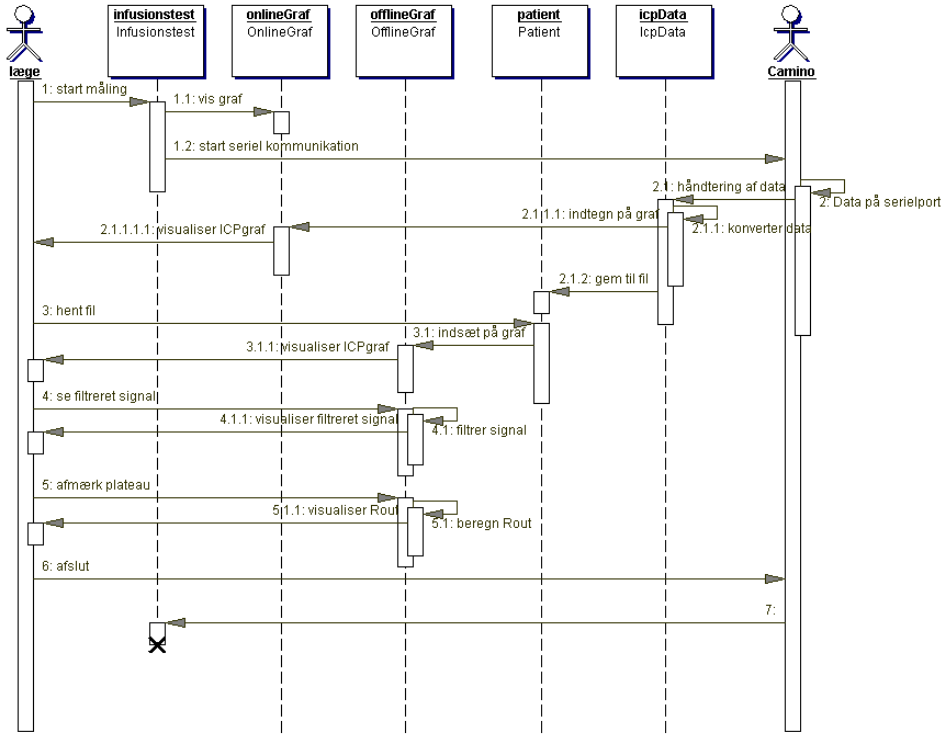
Figur 5.9: *Sekvensdiagram for B-bølge detekteringen*

På brugerfalden skal der først trykkes på en knap, der henter ICP-datafilen. Der er mulighed for at søge i filerne på computeren og vælge den ønskede fil. Denne indlæses i grafen på brugerfladen, således ICP-grafen ses ved en opløsningen på 30 cm/time. Ved tryk på en anden knap, beregnes B-bølgeaktiviteten, og markering af B-bølger vises på grafen. Procenttallet for B-bølgerne vises ligeledes på brugerfladen.
Lægen kan på denne baggrund foretage sin vurdering, og kan derefter afslutte programmet.

5.3.4 Infusionstest

Sekvensdiagrammet for Infusionstest ses i figur 5.10.

Infusionstesten foregår på næsten samme måde som ICP-målingen. Målingen startes, der indtastes filnavn og ICP-data vises i en graf. Grafen skaleres dog således, at hele grafen kan ses på et



Figur 5.10: *Sekvensdiagram for Infusionstesten*

skærmbillede.

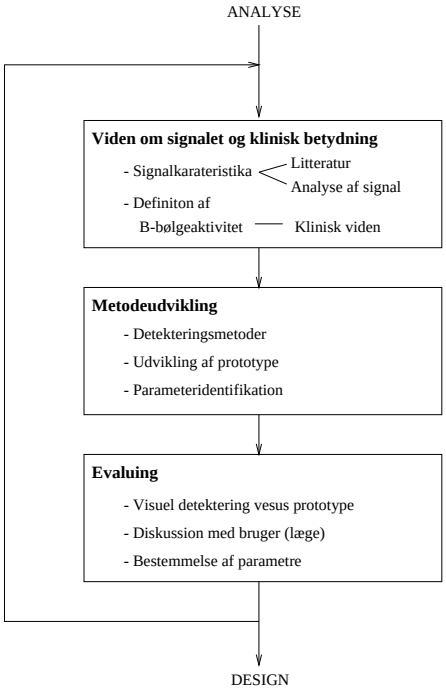
Lægen vurderer, hvornår målingen skal stoppes. Det er muligt at slukke computeren og senere hente datafilen frem igen og se den i grafen. Det er muligt at beregne R_{out} ved at afmærke plateauniveauer og indtaste væske-infusionsraten. R_{out} kan beregnes ligesom lægen ellers ville gøre med en papirudskreven kurve. Der kan vælges at se rådata eller et filtreret signal, hvor støjen er fjernet.

KAPITEL 6

Signalbehandling

Kapitlet indeholder en beskrivelse af signalet ved en 24-timers ICP-måling samt definition af de B-bølger, der ønskes detekteret af systemet. Der vælges en metode til detekteringsalgoritmen for B-bølgedetektering. Slutteligt gives en redegørelse for beregninger omkring R_{out} ved infusionstest.

Der vil blive fokuseret på detektering af B-bølgeaktivitet, indeholdende en analyse af signalet samt udvikling af en algoritme.
Processen for udvikling af en algoritme til detektering af B-bølger er illustreret i figur 6.1. I den videre beskrivelse af algoritimen vil der ikke fokuseres på processen.



Figur 6.1: *Processen ved udvikling af en algoritme, hvor forudsætningen er et vidensgrundlag om signalets karakteristika og betydning i diagnosemæssig sammenhæng. Herefter kan en metode udvikles, og evaluering foretages, for at kunne beslutte, om processen evt. skal starte forfra*

Viden om signalet og dets kliniske betydning

Algoritmen skal bygge på fakta angående signalets karakteristika, signalets frekvenser, og hvordan signalet opfører sig. Det er vigtigt at vide, om der eksemplevis er afvigelser i signalets frekvensspektre fra person til person eller fra situation til situation. Denne viden baseres på en analyse af tilgængelige signaler og på kendt viden fra litteratur.

B-bølgeaktivitet, som har betydning for diagnosen, skal være så præcist defineret, at algoritmen kan udvikles derefter. Denne viden er den kliniske viden, som skal erhverves fra brugeren, som er eksperten på området og fra litteraturstudier.

Metodeudvikling
På baggrund af viden kan algoritmen udvikles. En detekteringsmetode, der kan opfylde kravene, vælges og metoder til detekteringsalgoritmen diskuteres i appendiks C.
En prototype udvikles i Matlab og beskrives.

Evaluering
Prototypen er en model, der foretager beregning ud fra fastlagte parameterværdier, og disse bestemmes.
Brugeren præsenteres for en række papirkurver udskrevet med 12 cm/timen, som i dag benyttes ved diagnoseticering på AAS. Der foretages en visual bedømmelse af, til hvilke tidspunkter der forekommer B-bølgeaktivitet, og der foretages en beregning med den udviklede algoritme. Resultaterne sammenlignes og diskuteres med brugeren, hvor brugeren får mulighed for at bedømme kurverne på pc, dvs. får mulighed for at få præsenteret kurverne ved en ønsket opløsning. Formålet er at nå frem til ny viden i form af større eller ny indsigt omkring signalet, større detaljeringsgrad eller ændring af parameterkarakteristika. Evaluering har til formål at forbedre modellen, indtil et resultat kan accepteres.

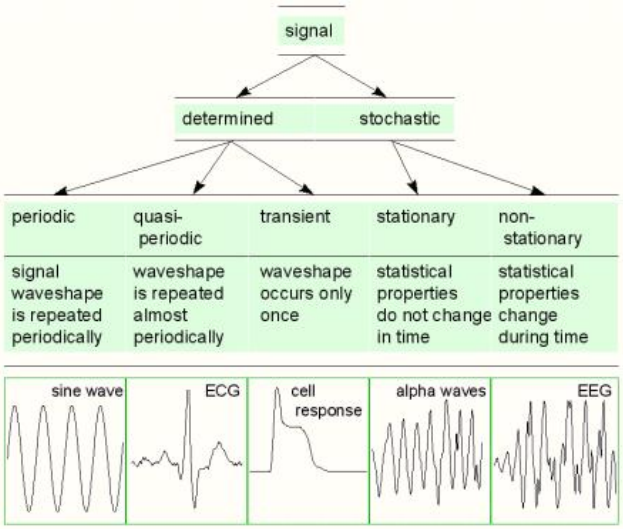
6.1 Viden om signalet og dets kliniske betydning

6.1.1 Signalkarakteristika

Når et signal skal behandles, er det vigtig at vide, hvilken type signal det pågældende signal kan karakteriseres ved.
Det intrakranielle tryksignal er et biologisk signal, der ønskes behandlet digitalt. Der arbejdes derfor med diskrete værdier modtaget fra Camino-trykmåleren.

Signalet kan inddeles efter bølgeformen (stokastisk eller deterministisk) og efter, om signalet varierer som en funktion af tiden (stationært eller non-stationært). Signalet klassificeres som et deterministisk signal, hvis det har en gentagende karakter og dermed er forudsigeligt. Et stokastisk signal danner spontane bølgeformer, og det er ikke muligt at forudsige en værdi på et givent tidspunkt [18]. Figur 6.2 viser klassifikationen af biologiske signaler.

Det intrakranielle tryk består af delvist kendte frekvenser, men klassificeres som et stokastisk signal, idet bølgeformerne er uforudsigelige og optræder til ukendte tidspunkter jf. afsnit 6.1.2. Dermed klassificeres signalet også som non-stationært, da det ændres uafhængigt over tid. Der kan imidlertid tales om periodisk aktivitet ved quasi-perioder (se figur 6.2), som ved forekomst af B-bølgeaktivitet. Ligeledes kan der optræde en A-bølge i signalet, hvilket er en enkeltstående bølge med en karakteristisk bølgeform (se tabel 6.2), som kan opfattes som et transient deterministisk signal (se figur 6.2).



Figur 6.2: *Klassifikation af biologiske signaler, hvor de to hovedgrupper er deterministiske og stokastiske signaler [18].*

6.1.2 Definition af B-bølgeaktivitet

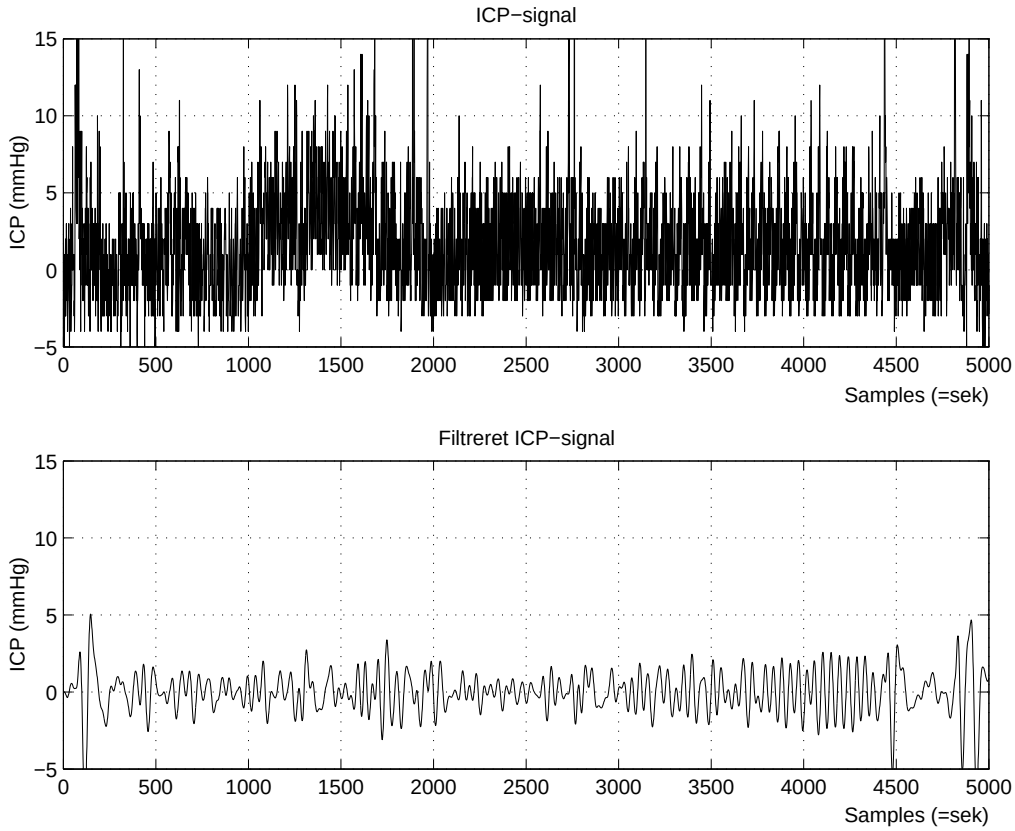
Forskellige faktorer har indvirkning på trykket i hjernen og medvirker til generering af bølger af forskellig karakteristika. Det kan både være fysiologiske og patologiske faktorer. Puls og åndedræt er eksempler på fysiologiske faktorer, der genererer bølger i signalet. Tryksignalet er en sum af disse forskellige bølger.

B-bølger er de betydende bølger for diagnosen, idet de anses for værende patologiske, hvis de forefindes i mere end 50% af en 24-timers monitoreringsperiode. 50%-grænsen er sat ved visual bedømmelse af trykkurver [14]. Det er vigtigt at notere sig, at B-bølger ikke forårsager sygdommen NPH, men er et symptom, der indikerer en lav kompliance i hjernen og/eller cirkulationsforstyrrelser af CSF.
Videnskabelige artikler omhandlende B-bølger refererer alle til inddelingen af trykbølger i A-,B- og C-bølger som beskrevet af Lundberg i 1960 [10] [9] [7] [20] [19].
B-bølgerne, der optræder i ICP-signalet, er blevet tillagt flere forskellige årsager. Hvis den intrakranielle kompliance er lav, har hjernen ikke, eller kun i ringe grad, mulighed for at kompensere for stigninger i trykket. Dette betyder, at der kun skal en lille volumenændring til, for at ICP stiger kraftigt [14] [3, s. 92] . Volumen kan stige pga. ophobning af CSF-væske. Dette kan ske, når der er ubalance mellem produktion og resorption af CSF, hvilket er en af de almindeligste forklaringer på NPH. Der er desuden andre faktorer, der kan medføre en volumenændring. Autoregulation af blodtrykket kan udvide blodårerne, og desuden øger øget koncentration af CO_2 i blodet blodvolumen. Pga. den periodiske aktivitet af B-bølger er det sandsynligt, at der er sammenhæng med blodtrykket [14].

Definition af B-bølger:

> **Rytmiske bølger med $\frac{1}{2}$ til 2 oscillationer pr. minut svarende til bølger af varighed 30-120 sekunder** [19]

For at fastlægge krav til modellens parametre, som algoritmen skal beregne B-bølgeaktivitet ud fra, betragtes et udsnit af ICP-signalet.
Figur 6.3 viser knap halvanden times udsnit af et ICP-signal, hvor det øverste er ubehandlet, og det nederste er filtreret med et båndpasfilter over B-bølgefrekvensspektret. Som det ses nederst



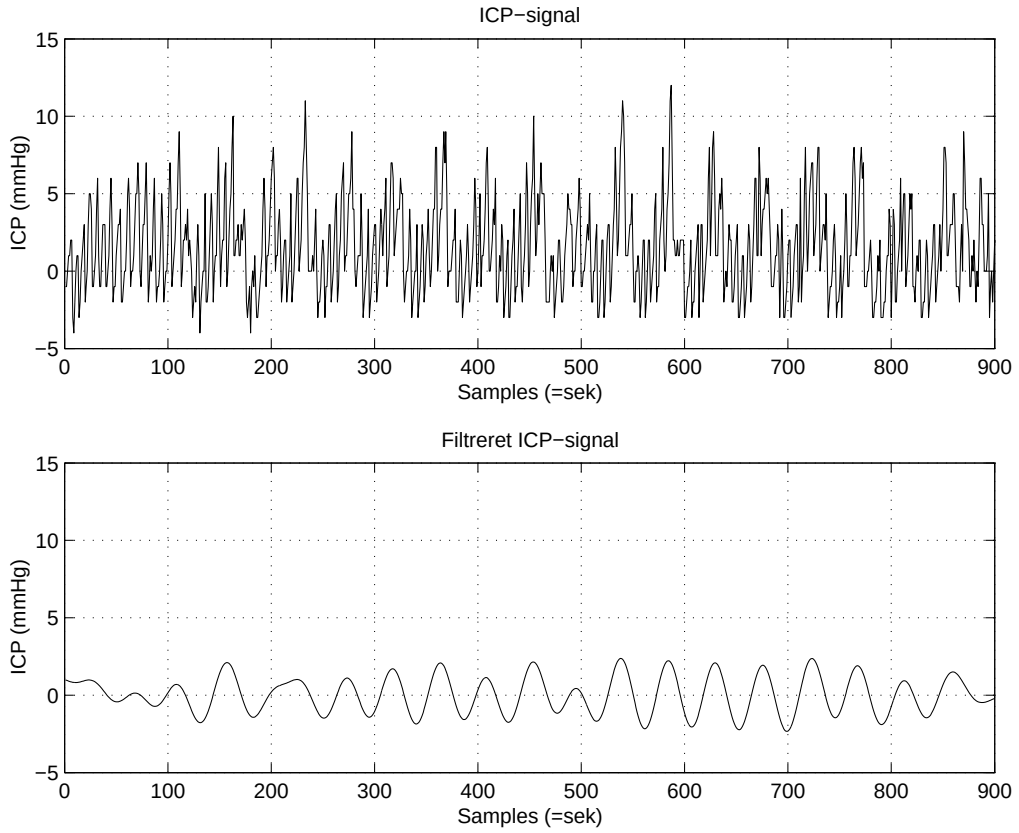
Figur 6.3: *Et typisk $1\frac{1}{2}$-times segment af en ICP-måling.*

på figur 6.3, er B-bølger bredt repræsenteret over hele segmentet. Dette er et typisk udsnit af et signal. Når lægen skal stille diagnosen ud fra kurven, kan han ikke skelne alle disse bølger, men han ser et karakteristisk mønster, som han i dette signaludsnit har identificeret til at ligge omkring samples 3500-4500.
Karakteristisk for dette segment er, at det består af B-bølger med tilnærmelsesvis samme amplitude. Det er sådanne mønstre, der ønskes detekteret, for det er for et sådan mønster, at der er sat en 50%-grænse for, hvornår tryksignalet er patologisk jf. figur 1.1 på side 4.

Parameterkarakterisik:

> **De B-bølger, der ønskes detekteret af brugeren til diagnosticering skal have et karakteristisk mønster. Der skal ikke detekteres enkelte bølger, men bølge*aktivitet*.**
> **Bølgeaktivitet karakteriseres ved aktivitet over ca. 5 min. svarende til ca. 3-10 B-bølger forekommende i forlængelse af hinanden. Disse bølger skal have tilnærmelsesvis samme amplitude, da dette giver det karakteristiske mønster, der ønskes detekteret.**

Det karakteristiske mønster fra figur 6.3 er forstørret i figur 6.4



Figur 6.4: *Figuren viser de karakteristiske bølger fra figur 6.3. Øverste figur viser parameterkarakteristikken, dvs. det karakteristiske mønster af bølger lægerne kigger efter i signalet. Nederste figur viser det samme mønster båndpasfiltreret over B-bølgespektret*

B-bølgerne kan yderligere inddeles efter amplitude og bølgeform som angivet i tabel 6.1. Der ønskes dog ikke at skelne mellem typerne i detekteringen. Definition af de øvrige bølger der ses

Bølgetype	Varighed/Osc.	Ampl./mmHg	Frekvens/Hz	Form
B-bølge Type 1	0,5-2 osc/min	>20 mmHg	0,0083-0,033 Hz	Asymmetriske
B-bølge Type 2	0,5-2 osc/min	>=10 mmHg	0,0083-0,033 Hz	Store symmetriske
B-bølge Type 3	0,5-2 osc/min	<10 mmHg	0,0083-0,033 Hz	Små symmetriske

Tabel 6.1: B-bølgernes karakteristika angivet ved varighed, amplitude, frekvens og form

i signalet er angivet i tabel 6.2.

Nærværende afsnit har beskrevet de frekvenser i signalet, som er kendte og veldefinerede. Der optræder imidlertid også frekvenser forårsaget af støj ved patientbevægelser [3, s. 63-64] . Frekvenser i B-bølgespektret kan forventes at opstå ved længerevarende bevægelse eller anden uro. Det er pålagt patienterne at ligge stille, så sådanne frekvenser vil optræde sjældent i signalet. En eventuel fejlkilde elimineres ved på grafen at gøre lægen opmærksom på, at patienten har været urolig i perioden

Bølgetype	Varighed/Osc.	Ampl./mmHg	Frekvens/Hz	Form
A-bølge	5-20 min	50-100 mmHg	<0,003 Hz	Asymmetriske
C-bølge	4-8 osc/min	<20 mmHg	0,067-0,13 Hz	Små symmestriske
Åndedræt	10-20 osc/min	2-10 mmHg	0,167-0,33 Hz	Symmetriske
Puls	50-90 osc/min	1-4 mmHg	0,83-1,5 Hz	Symmetriske

Tabel 6.2: De forskellige bølgetyper og deres karakteristika angivet ved varighed, amplitude, frekvens og form. Åndedræt og puls er angivet i hviletilstand [10] [9] [3].

I det efterfølgende betragtes alle frekvenser, der ikke indgår i B-bølgespektret, som støjfrekvenser.

6.2 Metodeudvikling

Præcisionen på en visuel detektering af B-bølger er lille. Dette kan konkluderes efter at have foretaget interobservatørundersøgelse på 5. semester [3, s. 41-53] , der indikerede store problematikker med visuelt at identificere bølger og definere B-bølgeaktivitet [3]. Dette viste, at forståelsen af B-bølger ikke er standardiseret. Derfor er et vigtigt krav til algoritmen, at den kan standardisere definitionen.

Der er flere forskellige metoder en detekteringsalgoritme kan baseres på. I appendiks C beskrives og diskuteres de forskellige metoder på baggrund af litteraturstudier.

Der er ikke blevet foretaget undersøgelser af interobservatørvariation blandt læger, der i det daglige har ansvaret for at diagnosticere ud fra en ICP-måling. Det må derfor forventes, at det er vanskeligt at sammenligne resultater med forskellige observatører, og dette vil gøre det svært at teste den udviklede algoritme.

I forskningssammenhænge kunne det være en fordel at lave en objektiv mønstergenkendelsesalgoritme, hvor signalet sammenlignes med sig selv, for at detektere mønstre og finde ud af, om der er konsensus for det brugeren identificerer som et mønster. Dette fokuseres der imidlertid ikke på i dette projekt.

For at kunne opfylde kravene sat til parameterkarakteristikken i afsnit 6.1.2 vælges det at benytte en metode, der er amplitueafhængig.
Der er ikke i litteraturen fastlagt en minimumgrænse for amplituden af B-bølger. Det antages imidlertid, for at muliggøre detektering, at der er en sådan grænse, og antagelsen forsvares med, at små amplituder er mindre relavante i klinisk diagnostisk sammenhæng.

6.2.1 Udvikling af prototype

Det vælges at udvikle to algoritmer, som sammenlignes, idet der ses fordele og ulemper ved begge metoder. Den ene metode detekterer hver enkelt bølge og udregner en RMS-værdi over et antal bølger, hvorimod den anden udregner arealet under kurven i vinduer.

Metode1:

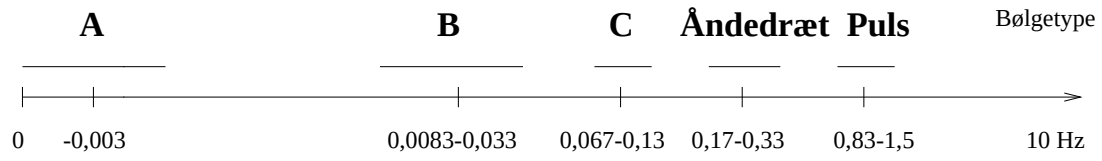
- En amplitudeafhængig metode er afprøvet i praksis med gode resultater [7].
- Metoden giver mulighed for at detektere et mønster med den rette parameterkarakteristik.
- Metoden har brugervenlig indstilling af parametre.

Metode2:

- Der foretages beregninger over ikke-overlappende vinduer, til forskel fra metode1, fordi dette kræver færre beregninger og muliggør simpel implementation i Java. Dette er muligt ved at beregne arealet mellem to samples, hvilket betyder, at der ikke er krav til, hvor vinduet placeres.
- Det antages, at B-bølgernes frekvenser er spredt over frekvensspektret og gennemsnitligt er af samme varighed, for dermed ikke at give fejlkilde ved udregning af arealet under bølgerne.
- Metoden er ikke beskrevet i litteraturen, men er, efter projektgruppens vurdering, et godt alternativ til metoderne beskrevet i artiklen "Two Computerized Methods..." [7].

To prototyper udvikles i Matlab, hvor de testes og evalueres. Kildekoden til Matlab-programmerne forefindes på CD-ROM [CD, A.2.1] .

Båndpasfilter
For begge metoder gælder det, at signalet skal båndpasfiltreres over B-bølgespektret.
Med et båndpasfilter er det muligt at dæmpe de uønskede frekvenser i signalet. De kendte frekvenser ligger relativt spredt, hvilket ses i figur 6.5. Det antages, at signalet maksimalt når



Figur 6.5: *ICP-signalets bølger og deres frekvensspektre skitseret i forhold til hinanden på en logaritmisk skala. Illustration af tabel 6.1.*

en amplitude for A-bølger på 100 mmHg og for C-bølger på 20 mmHg, som angivet i tabel 6.2 på foregående side. Det bemærkes, at i klinisk henseende betragtes ICP < 15 mmHg for værende normalt. Amplituden udtrykt i dB for henholdsvis A- og C-bølger:

$$A_A = 20 \cdot log(100) = 40dB$$

$$A_C = 20 \cdot log(20) = 26dB$$

Et filter skal dermed dæmpe A- og C-bølger med følgende, hvor X er filterets dæmpning:

$$X \cdot log(\frac{0,003}{0,0083}) = 40dB \Leftrightarrow \underline{X = -91dB/decade}$$

Dæmpning af A-bølger (0-0,003 Hz)

$$X \cdot log(\frac{0,033}{0,067}) = 26dB \Leftrightarrow \underline{X = -85dB/decade}$$

Et 5. ordens filter dæmper med 100 dB pr. decade, hvilket er et minimum, for at opfylde kravet om en dæmpning på 91 dB.

Knækfrekvenserne er B-bølgespektrets grænser, hvor disse værdier er dæmpet 3 dB, dvs. knækfrekvenserne for filteret bliver:

$$20 \cdot log(\frac{0,0083}{X}) = 3dB \Leftrightarrow \underline{X = 5,88mHz}.$$

$$20 \cdot log(\frac{0.033}{X}) = -3dB \Leftrightarrow \underline{X = 46,6mHz}.$$

For at undgå faseforskydning laves et 4. ordens filter, hvor signalet først filtreret forfra og derefter bagfra. Filtreringen svarer dermed til et 8. ordens filter, hvilket er mindste orden filter, der er muligt at realisere vha. Matlab, når kravet er et 5. ordens filter.

6.2.2 Metode1

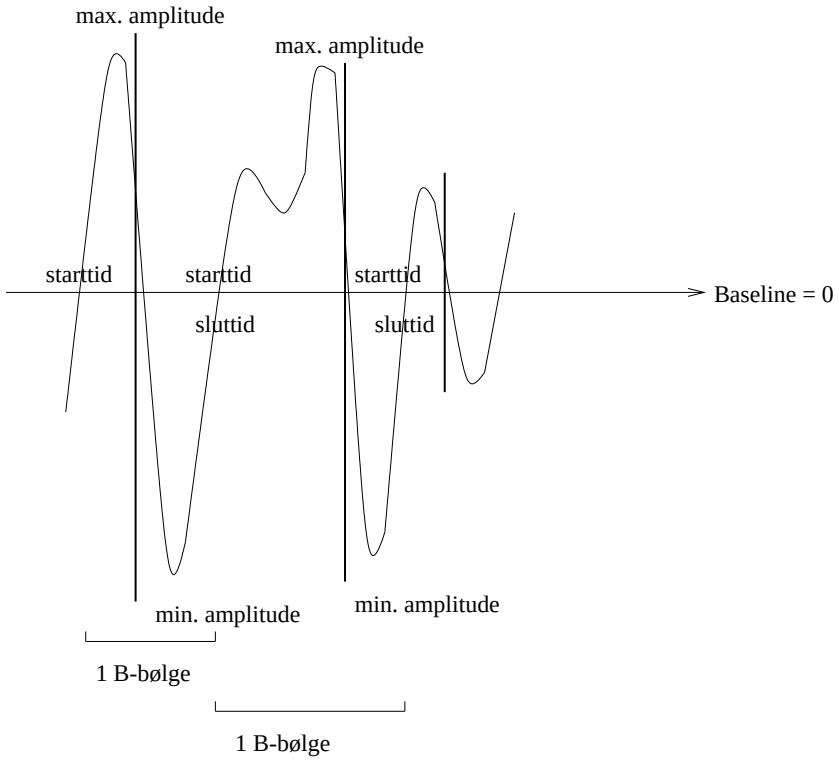
Detektering ved amplituden

Algoritmen identificerer hver enkelt bølge i signalet. For hver bølge identificeres starttidspunkt, globalt maksimum, globalt minimum og sluttidspunkt, som det er vist i figur 6.6. Følgende er de betydende parametre for hver bølge, her vist for bølge1:

- array[bølge1,1]= Starttidspunkt. Er første sample der overstiger 0-linien
- array[bølge1,2]= Globalt maksimum. Er største positive værdi i en enkelt bølge
- array[bølge1,3]= Globalt minimum. Er største negative værdi i en enkelt bølge
- array[bølge1,4]= Sluttidspunkt. Er sidste sample inden 0-linien
- array[bølge1,5]= Amplitude

Amplituden for den enkelte bølge udregnes som:

$$Amplitude = globalt\ maksimum\ -\ globalt\ minimum.$$



Figur 6.6: *Illustrerer de punkter på hver enkelt bølge, som algoritmen identificerer. Signalet er båndpasfiltreret, så baseline ligger på 0-linien, og alle bølger der beregnes på ligger i B-bølgefrekvensspektret.*

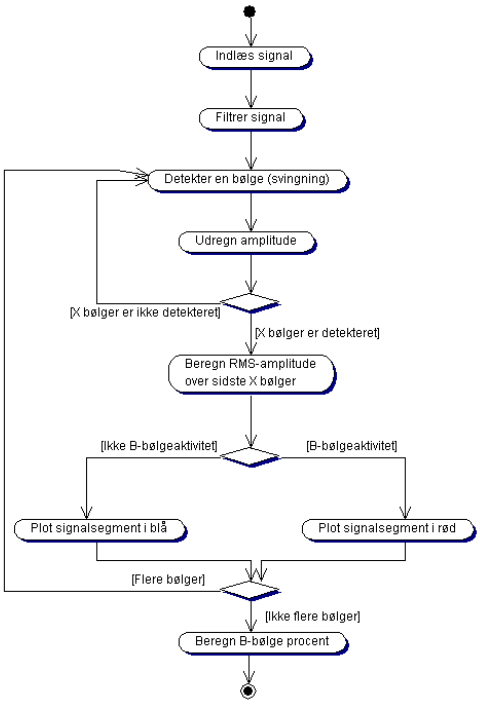
Der er to parametre, der skal fastlægges. Det er perioden for B-bølgeaktivitet og grænsen for minimum RMS-amplitude. Perioden for B-bølgeaktivitet fastlægges ved antallet af B-bølger i perioden. Analyse af udvalgte B-bølgesegmenter i ICP-signalet har vist, at bølgerne kun har tilnærmelsesvis samme amplitudeværdier. Derfor udregnes en RMS-værdi over dette antal af B-bølger, og denne skal overstige minimum RMS-amplituden. En RMS-værdi udregnes for at forøge signal-støjforholdet, og for at detektere sammenhængende perioder i signalet efter brugerens ønske.

Figur 6.7 viser et aktivitetsdiagram over en prototype for metode1. Betydende værdier for hver detekterede bølge i det filtrerede signal lægges i hver sin søjle i et array. For hver detekteret B-bølge udregnes RMS-amplituden for de seneste B-bølger, som er en variabel parameter.

Hvis betingelsene for B-bølgeaktivitet er opfyldte, farves signalet rødt i perioden, ellers farves det blåt.

For at undgå, at én stor amplitude ukorrekt kan forårsage, at RMS-amplituden over antallet af bølger der beregnes på overstiges, sættes en amplitudeværdi over 5 mmHg lig RMS-amplituden.

$$amplitudeværdi > 5mmHg \Rightarrow amplitudeværdi = RMSgrænseværdi$$

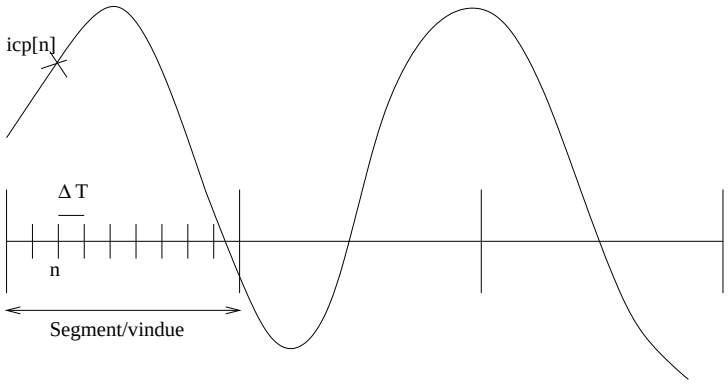


Figur 6.7: *Aktivitetsdiagram over en prototype for metode1*

6.2.3 Metode2

Detektering ved arealet

Det filtrerede signal inddeles i en m×n matrix, hvor n er samples i et vindue, og m er det givne vindue. Denne opdeling er illustreret i figur 6.8. Der foretages en numerisk integration over et



Figur 6.8: *Illustration af hvorledes signalet inddeles i vinduer.*

vindue af det filtrerede signal, hvor $icp[n]$ er den filtrerede ICP-værdi til den givne sample. For at opnå en mere præcis værdi anvendes trapeziodal-metoden, dvs. arealet mellem to samples betragtes som en trapezform, hvor der beregnes i forhold til den næste sample i rækken. Det er valgt at foretage en præcis beregning, fordi der er mange andre usikkerhedsfaktorer, der spiller

ind i detektering af B-bølger.
For at forøge signal-støjforholdet regnes på kvadratet af amplitudeværdien, som angivet i ligning 6.1, hvor t angiver tiden.

$$Integral = \sum_{i=0}^{n-1}(icp[n] \cdot \Delta t + (icp[n+1] - icp[n]) \cdot \Delta t \cdot \frac{1}{2})^2 \tag{6.1}$$

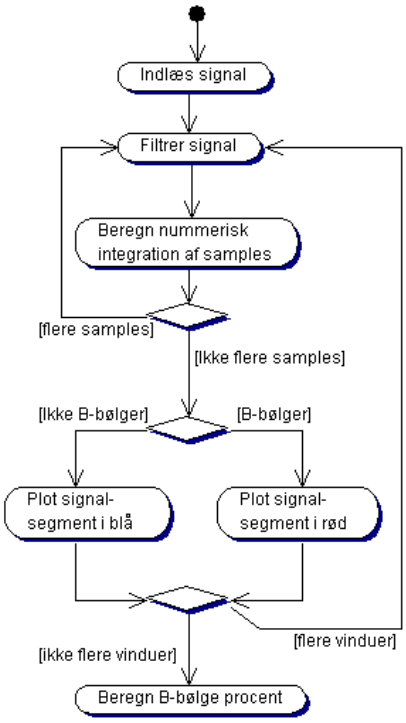
Hvis to samples ligger på hver side af 0-linien, dvs. den ene værdi er positiv og den anden negativ, udregnes integralet ved ligningerne 6.2 og 6.3

$$t = \frac{-icp[n] \cdot \Delta t}{icp[n+1] - icp[n]} \tag{6.2}$$

$$Integral = |t \cdot \frac{icp[n]}{2}| + |(\Delta t - t) \cdot \frac{icp[n+1]}{2}| \tag{6.3}$$

Metoden kan ikke tage højde for ukendte støjfrekvenser, som man ikke kan være sikker på dæmpes fuldstændigt af et båndpasfilter, og som derved ville kunne skabe fejlkilder. Det er dog muligt at frasortere støjpeaks ved at ekskludere disse fra beregningen af *Integral*-værdien. For værdier over 20 mmHg sættes integralet derfor lig 0..

Algoritmen udvikles som vist i aktivitetsdiagrammet figur 6.9. Detekterede B-bølgeperioder



Figur 6.9: *Aktivitetsdiagram over en prototype for metode2*

farves på rådatasignalet, da det er rådata brugeren er vant til at se på.
Den procentvise B-bølgeaktivitet udregnes som den røde del af signalet i forhold til længden af signalet.

Kriterier
Der opstilles kriterier for, hvad der detekteres som B-bølgeaktivitet, og kun hvis disse overholdes farves segmentet. På denne måde er det muligt at forøge vindueslængden uden, at fejlkilder opstår, som det kan ske, hvis et vindue skærer en periode med B-bølgeaktivitet skævt. Kravene sættes som følger:

- $Integral$[m] > arealtærskel
- Der skal gælde for et af de følgende omkringliggende segmenter:

  $Integral$[m-2] > arealtærskel

  $Integral$[m-1] > arealtærskel

  $Integral$[m+1] > arealtærskel

  $Integral$[m+2] > arealtærskel

Det er således muligt at justere på parametrerne arealtærskel og vindueslængde (minutter).

6.3 Evaluering

6.3.1 Præsentation for bruger

Brugeren præsenteres for algoritmernes fund af B-bølgeaktivitet. Signalet præsenteres ved en opløsning svarende til 30 cm/time, hvilket er den opløsning brugeren har identificeret som værende den bedste til visualisering af signalet [3, s. 63] .
Han finder ved visuel detektering gerne sammenhængende perioder, også selvom en lille del af perioden evt. ikke opfylder kravene for B-bølgeaktivitet, da dette ikke fremgår visuelt. Hans detektering er dermed grovere end algoritmens, hvilket accepteres.

Ved præsentation af algoritmernes detektering opnåes enighed om, at når baseline har en hældning, er mønsteret ikke længere karakteristisk visuelt, selvom parameterkarakteristikken er opfyldt. Algoritmen vil derfor i sådanne tilfælde korrekt detektere B-bølgeaktivitet, som brugeren ikke har markeret. Det må derfor forventes, at algoritmen detekterer en større mængde B-bølgeaktivitet end observatøren i sådanne tilfælde.

6.3.2 Metode til evaluering

Gruppen har haft adgang til 2 målinger, der er repræsenteret både analogt og digitalt. Målingerne er opsamlet med programmet MPM- Seriel Data Capture, som er udviklet af Integra Neuro Care, og som kan kommunikere serielt med Camino MPM-1 og vise tryksignalet på en graf.
Målingerne er samplet med 1 Hz, hvilket vurderes tilstrækkeligt til en analyse af B-bølgeaktivitet. ICP-måling1 er foretaget over 18 timer fra kl. 16.30-10.30. Observatøren har i denne måling detekteret 29 % B-bølgeaktivitet.

ICP-måling2 er foretaget over 8 timer fra kl. 16.00-24.00. I denne måling er der af observatøren detekteret en %B-bølgeaktivitet på 7%.

En objektiv fastlæggelse af parameterværdier foretages, og til sammenligning benyttes en erfaren neurokirurgs visuelle bedømmelse som ekspertreference.
Formålet er ikke nødvendigvis, at algoritmen skal detektere, hvad brugeren gerne vil se, men skal detektere ud fra de definitioner, der er opsat for B-bølgeaktivitet. Kravene til B-bølgeaktivitet er imidlertid relativt løse, hvilket betyder, at flere forskellige parametre kan opfylde kravene. Af denne årsag er brugerens detektering benyttet som reference.

Det giver usikkerheder kun at benytte en enkelt observatør og benytte denne detektering som reference, fordi fundet af B-bølger er meget subjektivt [3], men da der kun er en enkelt person med erfaring i en sådan detektering på AAS, må denne begrænsning accepteres. Erfaringen betyder imidlertid, at det kan forudsættes, at kurverne bedømmes på samme måde, dvs. at der er samme bias på begge kurver.
Resultaterne diskuteres med brugeren, og ved diskussion vurderes yderligere 2 målinger registreret på pc. Ud fra dette anbefales parameterværdier.
De to algoritmer sammenlignes, og der foretages et valg af metode.
Observatøren præsenteres for papirkurverne og markerer B-bølgeaktivitet og angiver, hvis der har været tvivl.

De to algoritmer evalueres ved systematisk at ændre på parameterværdierne og beregne B-bølgeaktivitet.
Argumenterne for valg af parameterværdier til evaluering for metode1:

- RMS-amplitude: Et filtreret signal analyseres, og det vurderes, at amplituderne gennemsnitligt ligger omkring 2-4 mmHg. Derfor vælges værdierne 2,5 mmHg, 3,0 mmHg, 3,5 mmHg og 4,0 mmHg.

- Antal bølger: Bølgeaktivitet er defineret til at vare ca. 5 min., og da bølgerne oscillerer 0,5-2 gange i minuttet vælges antallet af bølger til 3, 5, 7 og 9.

Argumenter for valg af parameterværdier til evaluering for metode2:

- Vindue: Et lille vindue vil betyde, at den detekterede B-bølgeaktivitet er kortvarig. Til gængæld vil et stort vindue give fejlkilder, fordi vinduet kan ramme skævt ind på bølgerperioderne. Det er defor fundet passende at lade et mindste vindue være 1,5 min. svarende til minimumlængden af 3 bølger. Det vælges at se på vinduer af størrelserne 1,5 min, 2.0 min, 2,5 min og 3,0 min. Fejlkilden vil dermed for et tre-minutters vindue være $< 2 \times 3$ min., fordi det er muligt at få op til tre minutters fejlkilde i start og slutning af en bølgeperiode.

- Areal: Værdierne er fastlagt ud fra et typisk B-bølgeudsnit. Ved et vindue på 1,5 min sættes arealniveauer til 60, 70, 80 og 90. Ved et større vindue øges arealet med lige så mange procent som vinduet øges.

For hvert signal beregnes følgende, og resultater ses i bilag B.

- Den procentvise B-bølgeaktivitet beregnes ved: samples med B-bølgeaktivitet i forhold til antal samples i signalet.

- Der udregnes en sammenlignelig værdi af den visuelt detekterede mængde B-bølgeaktivitet i forhold til den beregnede mængde B-bølger i signalet.

- Overensstemmelse mellem detektering af B-bølgeaktivitet visuelt og beregnet udregnes ved: fællesmængden mellem samples med beregnet B-bølgeaktivitet og visuelt detekteret B-bølgeaktivitet i forhold til antal samples ved visuelt detekteret B-bølgeaktivitet.

6.3.3 Valg af parametre og sammenligning af metoder

Af resultaterne fremgår det, at beregnet %B-bølger generelt ligger højere end visuelt detekteret %B-bølger. Dette er især tilfældet for måling2.
Brugeren afviser ikke, at det er muligt, at der er mere bølgeaktivitet i signalet, end han først havde fundet.

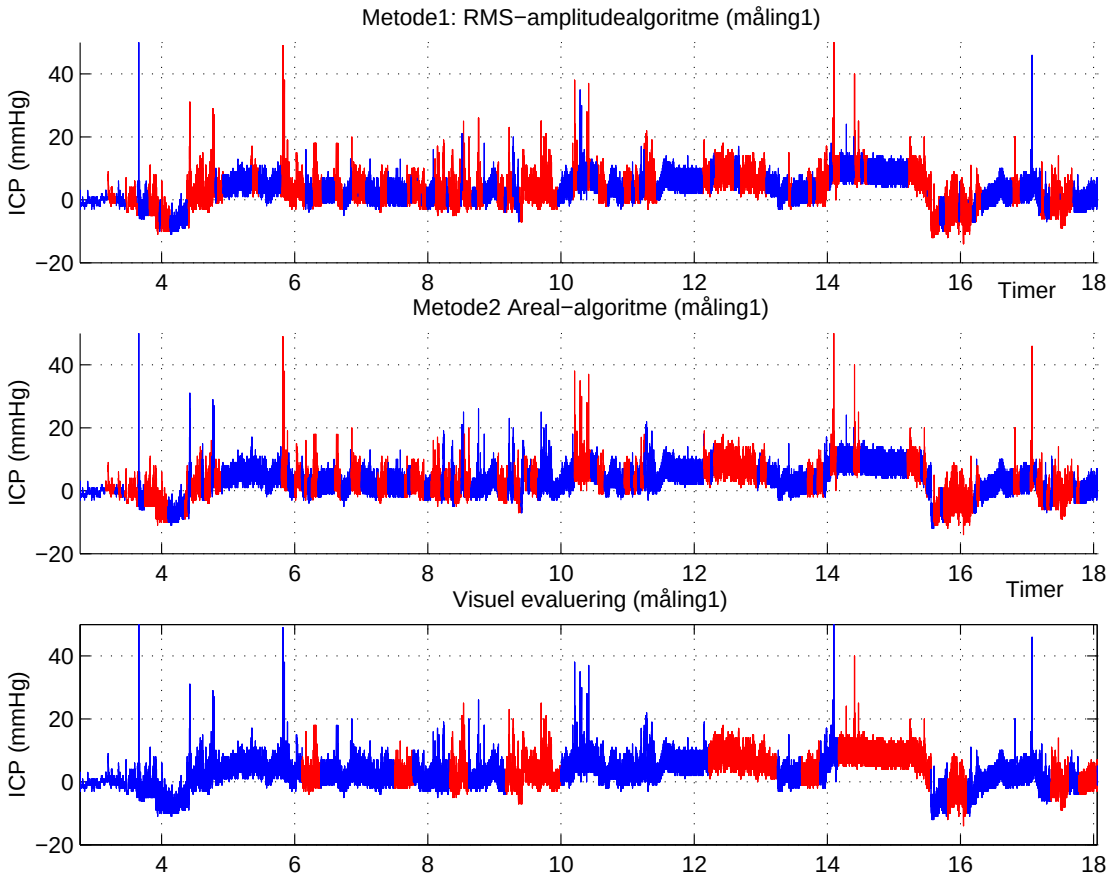
Den højeste overensstemmelse mellem fundet af %B-bølgeaktivitet beregnet og visuelt forekommer ved de lavest målte RMS-amplitudegrænser og arealgrænser i forhold til antallet af bølger/længde af vindue. At denne overensstemmelse ikke kommer over 65% i måling1 skyldes, at observatøren har været usikker i perioden 14-14.30. Ved filtrering viser det sig, at amplituderne i dette interval genemsnitligt ligger på 1,5 mmHg, hvilket ikke kan detekteres af algoritmerne, fordi alle B-bølger dermed vil detekteres, og det er ikke muligt at detektere aktivitet.

Ud fra resultaterne og diskussion med bruger anbefales det, at parametrene for metode1 ligger på RMS-amplitudegrænse 2,5 mmHg og antal bølger 5. For metode2 anbefales det, at parametrene ligger på vindue 3 min og og areal 120.
Pga. det lave antal målinger der har været adgang til, er det vigtigt, at systemet giver mulighed for at ændre på parameterværdier, og at evaluering gentages, når der er adgang til flere ICP-målinger.

I figur 6.10 og figur 6.11 er angivet detekteringen af B-bølgeaktivitet af de to algoritmer samt observatørens visuelle detektering for måling1 og måling2. Sammenlignes de to algoritmer i figur 6.10 ses det, at algoritmerne hovedsageligt adskiller sig fra hinanden i perioden fra sample 9.30-10.15. Ved benyttelse af samme parameterværdier for måling2 ses det i figur 6.11, at der detekteres meget mere B-bølgeaktivitet end observatøren har angivet.

6.3.4 Valg af algoritme

Det kan ikke entydigt ud fra evalueringen konkluderes, hvilken metode der er bedst. Metode1 har den vigtige fordel, at den er mere brugervenlig i forhold til indstilling af parametre. Metode2 har imidlertid vist sig simplere at implementere i Java, og derfor vælges det at implementere metode2.



Figur 6.10: *De to algoritmer baseret på hhv. metode1 og metode2 benyttet på måling1 ses som de to øverste figurer . I nederste figur ses den visuelle detektering af B-bølgeaktivitet. For metode1 er benyttet parametrerne antal Bølger = 5, RMS-amplitudegrænse = 2,5 mmHg og for metode2 parametrerne vindue = 3 min. og arealtærskelværdi = 120*

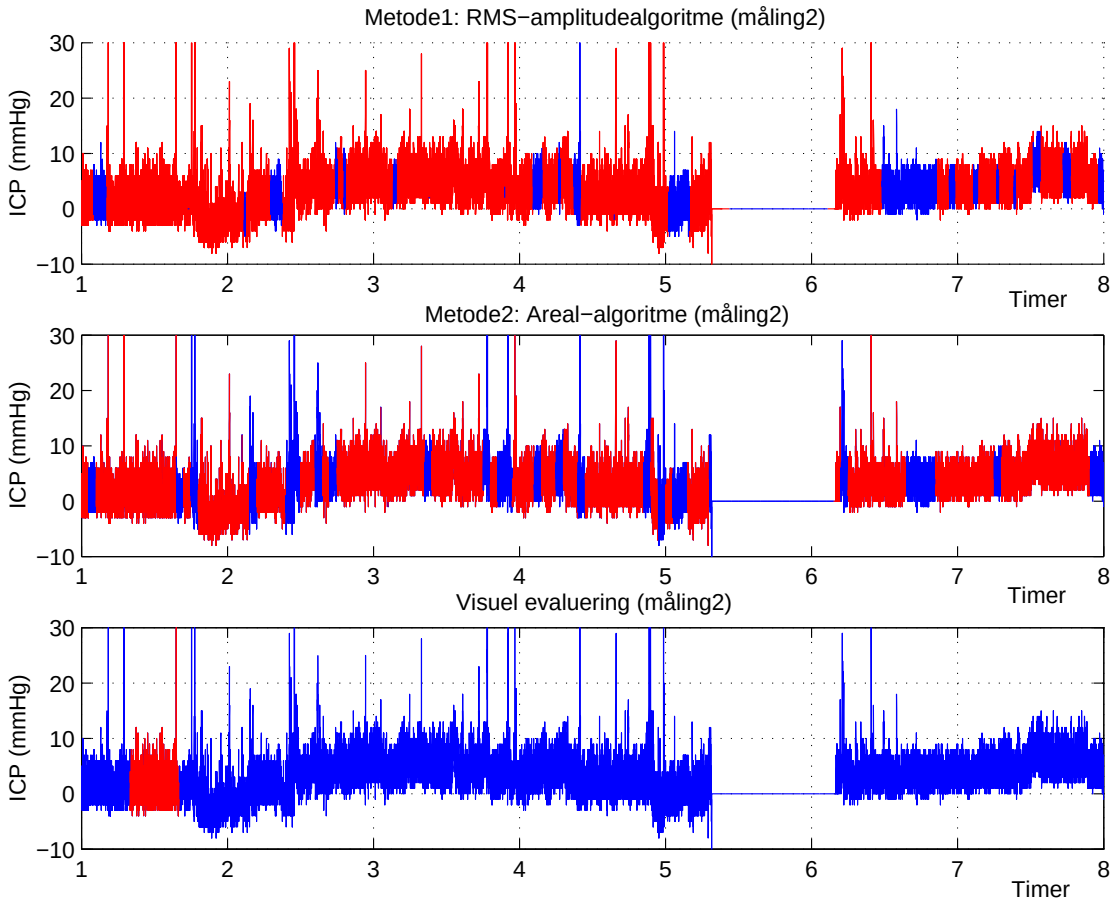
6.4 Infusionstest

Der har i projektperioden kun været adgang til én digital infusionsmåling, hvorfor det vælges ikke at foretage en større analyse af signalets mønster. Udregning af R_{out} foretages i klinisk praksis ved en statisk metode, som det vælges at fokusere på.

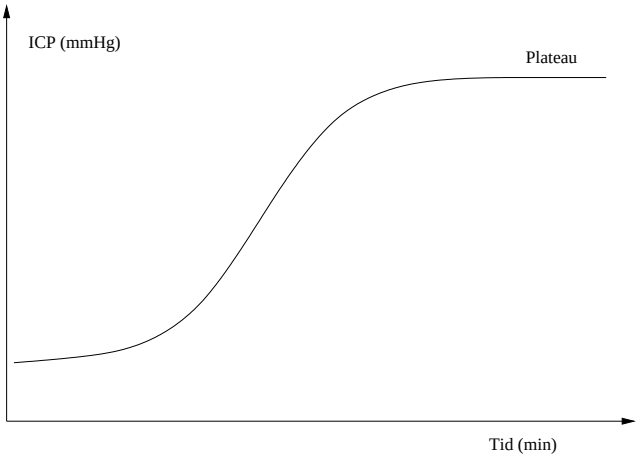
6.4.1 Detekteringsmetode

Det vælges at benytte en statisk metode, som er baseret på, at ICP opnår et nyt trykniveau under infusion af væske i CSF-rummet. I figur 6.12 ses en skitse af den model en infusionstest vil tilnærme sig. Som det ses af figuren er kurven sigmoidformet. Hvor lang tid det tager for et plateau at indfinde sig kan variere fra patient til patient. Infusion af væsken varer under 2 timer, ellers vil målingen oftest afbrydes af lægen, idet han vurderer den ubrugelig [2].

Hvis der ikke opnåes et plateau kan R_{out} ikke beregnes. I situationer hvor ICP-kurven ikke indfinder sig på et nyt plateau, er der indikation for et patologisk forhøjet R_{out} [5].



Figur 6.11: *De to algoritmer baseret på hhv. metode1 og metode2 benyttet på måling2 ses som de to øverste figurer. I nederste figur ses den visuelle detektering af B-bølgeaktivitet. For metode1 er benyttet parametrerne antal Bølger=5, RMS-amplitudegrænse= 2,5 mmHg og for metode2 parametrerne vindue= 3 min og arealtærskelværdi=120.*



Figur 6.12: *Skitse af modellen infusionstesten bygger på. Eksempelvis kunne start-plateauet være 5 mmHg og slut-plateauet 25 mmHg. Tidsrummet, hvor trykket stiger, kunne være 20 min. [2].*

6.4.2 Beregning af R_{out}

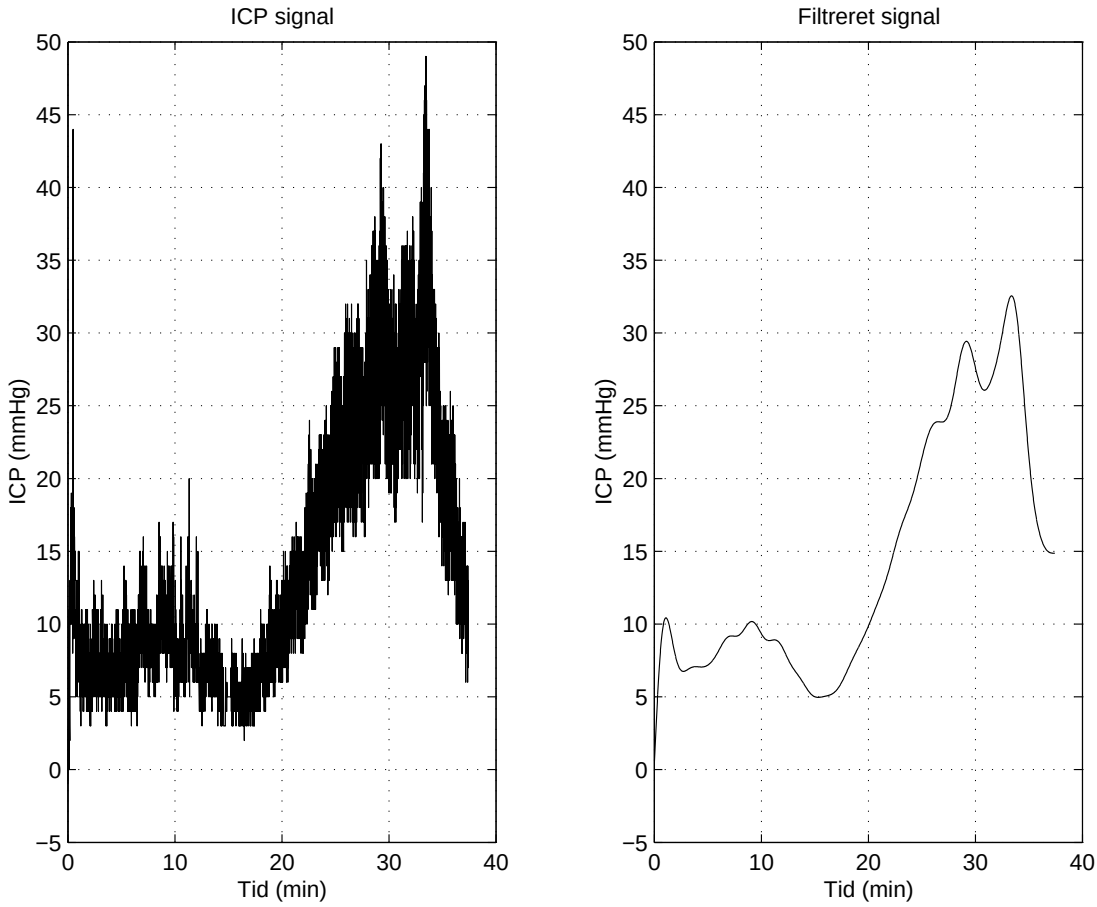
R_{out} er et udtryk for modstanden mod udløb af CSF. CSF absorberes gennem villi arachnoidales til sinus sagitalis superior og løber tilbage i kredsløbet med veneblodet [14].
Lægen beregner R_{out} ud fra ligning 6.4 [5] [14] [2]. For at simplificere udregning anvendes oftest en infusionsrate på 1 ml/min.

$$R_{out} = \frac{ICP_{slut} - ICP_{start}}{V_{inf}} \tag{6.4}$$

ICP_{start} er ICP-trykket (mmHg) ved infusionstestens start.
ICP_{slut} angiver ICP-trykket (mmHg) til ved nyt plateau.
V_{inf} er infusionsraten (ml/min).
R_{out} måles i $\frac{mmHg}{ml/min}$ [5] [14] [2]

6.4.3 Behandling af signalet

I infusionssignalet er det kun væsentligt at detektere, hvordan baseline ændrer sig ved infusion. Frekvenser, som er konstante i alle infusionsmålinger, er uden betydning ved beregning af R_{out}. Konstanter i signalet er de i tabel 6.1 på side 36 og 6.2 på side 37 definerede bølgefrekvenser. Det vælges derfor at lavpasfiltrere signalet og frafiltrere B-bølger og alle højere frekvenser, dvs. bølger med en frekvens højere end 0,0083 Hz. Der vurderes, at et 4. ordens filter er tilstrækkeligt. Det implementerede filter er et 2. ordens Butterworth lavpasfilter med knækfrekvens på 0,006 Hz. Signalet køres gennem filtret forfra, hvorefter sekvensen vendes og signalet filtreres igen for at undgå faseforskydning. Filtrets koefficienter findes via Matlab.
På figur 6.13 ses, hvorledes det ufiltrerede og det filtrerede signal ser ud.



Figur 6.13: *Til venstre ses et ufiltreret ICP-signal fra en infusionstest. Til højre samme signal filtreret med et 4. ordens 0-fase butterworthfilter.*

Del III

DESIGN

KAPITEL 7

Design

Designet af softwaresystemet beskrives ved betragtning af komponent- og processarkitektur. Først beskrives overvejelser og vurderinger omkring kvaliteten af systemet

Der laves en vurdering af de kvalitetsfaktorer, der er vigtige for produktet. Kravene sættes ud fra definitioner i softwareudviklingsmodellen SPU [15].

Der redegøres for arkitekturen ved en statisk og en dynamisk beskrivelse af systemet.
Den statiske beskrivelse fremstilles ved komponentarkitektur, som vil indeholde et designklassediagram og identifikation af metoder for hver af disse klasser.
Den dynamiske beskrivelse fremstilles ved procesarkitektur, der dokumenterer, hvordan og hvornår instanser af klasser (dvs. objekter) initialiseres og viser kommunikationen mellem objekterne ved metodekald.

7.1 Kvalitetsfaktorer

Hver kvalitetsfaktor vurderes på en 5-punkts skala fra *ukritisk* til *særdeles vigtigt* og er angivet i tabel 7.1. Kravene skal forstås som minimumskrav og sættes ud fra de konsekvenser det vil have, hvis kravet ikke opfyldes.

	Ukritisk	Ikke særlig vigtig	Vigtig	Meget vigtig	Særdeles vigtig
Pålidelighed			X		
Vedligeholdelsesvenlighed			X		
Udvidelsesvenlighed				X	
Brugervenlighed					X
Genbrugbarhed	X				
Integritet			X		
Effektivitet				X	

Tabel 7.1: Væsentlige kvalitetsfaktorer vurderet på en 5-punktsskala [15]

Pålidelighed: Er sandsynligheden for, at systemet fungerer i en given periode uden fejl svarende til "oppe-tiden" [15].
Pålideligheden vurderes ud fra, hvilke konsekvenser det har for patienten, at systemet går ned, samt hvilke økonomiske omkostninger et nedbrud vil få. Hvis systemet skulle gå ned under en

måling, vil det i værste fald betyde, at målingen skal tages om, og at patienten skal ligge endnu et døgn til trykregistrering. Dette er til gene for patienten dog ikke med umiddelbare fysiske konsekvenser. Pålideligheden vurderes derfor som *vigtig*.
Punktet tages i betragtning under implementation og test.

Vedligeholdelsesvenlighed: Er den tid det tager at finde eller rette en fejl i et produkt, der er taget i brug svarende til "nede-tiden" [15]
Da systemet er et studieprojekt, der forventes færdigtestet efter det er taget i brug, er det vigtigt, at de fejl der opstår let kan rettes. Det er dog ikke kritisk nødvendigt, at det sker omgående pga. fysiske konsekvenser for patienten. Derfor vurderes vedligeholdelsesvenlighed som *vigtig*.
Punktet tages i betragtning under design af programstruktur og kodning af fejlhåndtering.

Udvidelsesvenlighed: Er den produktivitet, der opnås ved udvidelser af programmet, sammenlignet med den oprindelige produktivitet [15].
Da der er oplagte udvidelsesmuligheder for systemet, er det en fordel, at udvidelsesvenligheden er høj og vurderes derfor som *meget vigtig*.
Systemet er afgrænset i analysen jf. kapitel 5, og derfor indgår visse udvidelsesmuligheder allerede i requirementanalysen i kapitel 4.
Punktet tages i betragtning ved strukturering af design.

Brugervenlighed: Er brugerens oplevelse af produktets betjening [15].
I 5. semesters-projektet [3] fremgår det af spørgeskemaer, at personalet finder brugervenlighed vigtig for at sikre, at systemet vil blive benyttet. Systemet skal ikke benyttes dagligt, og personalet vil derfor sjældent komme i kontakt med systemet, hvilket er endnu et argument for, at det skal være let at sætte sig ind i brugen af systemet. Brugervenlighed vurderes derfor som *særdeles vigtig*.
Punktet tages i betragtning ved udvikling af brugergrænsefladen, ved fremlæggelse af en prototype for brugeren og ved udarbejdelse af en dansk brugermanual.

Genbrugbarhed: Er procentdelen af programmet, der skal kunne genbruges igen [15]
Programmet forventes ikke at skulle genbruges til andre applikationer og vurderes derfor til *ukritisk*.

Integritet: Er produktets evne til at beskytte sine data [15].
Med integritet gælder samme argumentation som for pålidelighed, dvs. at konsekvensen i værste fald er, at patienten skal have foretaget en ny trykregistrering hvis data ikke er blevet gemt. Integritet vurderes derfor som *vigtig*

Effektivitet: Er produktets ydeevne i bestemte situationer [15].
Det tager i dag ca. 5-10 minutter at stille en diagnose fra den udskrevne papirskurve [3], og det er et krav, at det nye system som minimum skal være lige så effektivt som det nuværende. For at systemet vil blive benyttet vurderes effektivitet af systemet, i sammenligning med det i dag benyttede system, at være *meget vigtig*.
Punktet tages i betragtning ved valg af samplingsfrekvens og den pc, som systemet skal fungere på.

KAPITEL 8

Komponentarkitektur

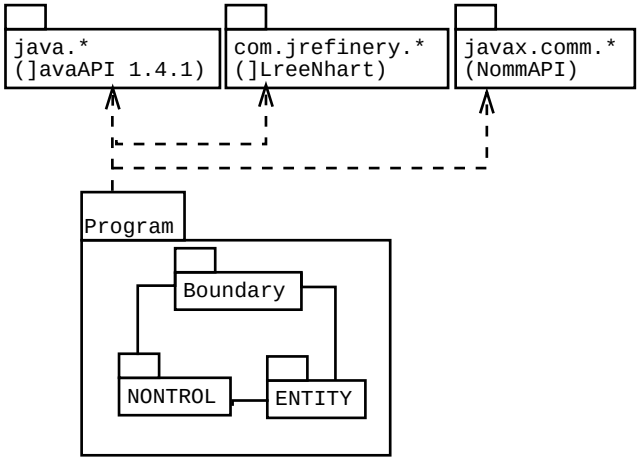
Komponentarkitekturen er en beskrivelse af systemet betragtet som et statisk system. Der fokuseres derfor på klasserne i systemet.

Systemet bygger på seriel kommunikation og visualisering af data i grafer. Det er derfor nødvendigt at arbejde med biblioteker, der ligger udenfor Java's normale Application Programming Interface (API).
Det vælges af tegne graferne ved hjælp af biblioteket JFreeChart, version 0.9.8 af David Gilbert, da dette synes at være det mest udbredte og ligetil samt rummer mange muligheder for konstruktion af grafer [34].

Til seriel kommunikationen benyttes Java's pakke til seriel kommunikation [33].

8.1 Programstruktur

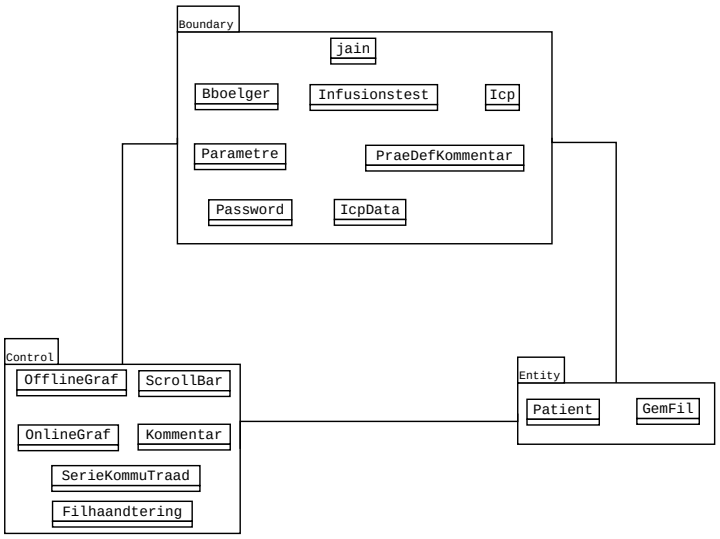
Overordnet kan strukturen illustreres i et diagram, der viser sammenhængen mellem pakkerne i systemet. Dette fremgår af figur 8.1. Det udviklede program er således afhængigt af de importerede pakker.



Figur 8.1: *Diagram der viser pakkerne i systemet og deres relationer med importerede pakker*

Klassediagrammet fra analysen (se figur 5.7 på side 28) udvides med en række nye klasser. Disse klasser er ikke direkte en del af problemområdet men gør, at systemet bliver nemmere at implementere, da håndtering af de forskellige funktioner i systemet kan adskilles. Alle klasserne

ses i figur 8.2, hvor de er inddelt i 3 pakker: Boundary, Control og Entity. Programmets klasser lægges i pakker for at skabe overskuelighed.



Figur 8.2: *Diagram over hvorledes de fundne klasser fordeles i pakker*

Boundary

Boundary er systemets grænseflader. Hovedparten af klasserne herunder håndterer den grafiske brugergrænseflade. Klassen IcpData varetager kommunikation og dataopsamling fra Caminoen. De resterende klasser er Main, Bboelger, Infusionstest, Icp, Parametre, Password og PraeDef-Kommentar.

Control

Når brugeren giver systemet et bestemt input, skal systemet udføre en handling. Denne handling sørger Control-delen for. Pakken består af 6 klasser, hvor 3 af klasserne udfører de handlinger, der foregår under målingen (OnlineGraf, SerielKommuTraad og Kommentar). De sidste 3 klasser (Filihaandtering, OfflineGraf og ScrollBar) sørger for at udføre de handlinger, der skal foregå, når lægen skal se på målingerne og detektere B-bølgeaktivitet.

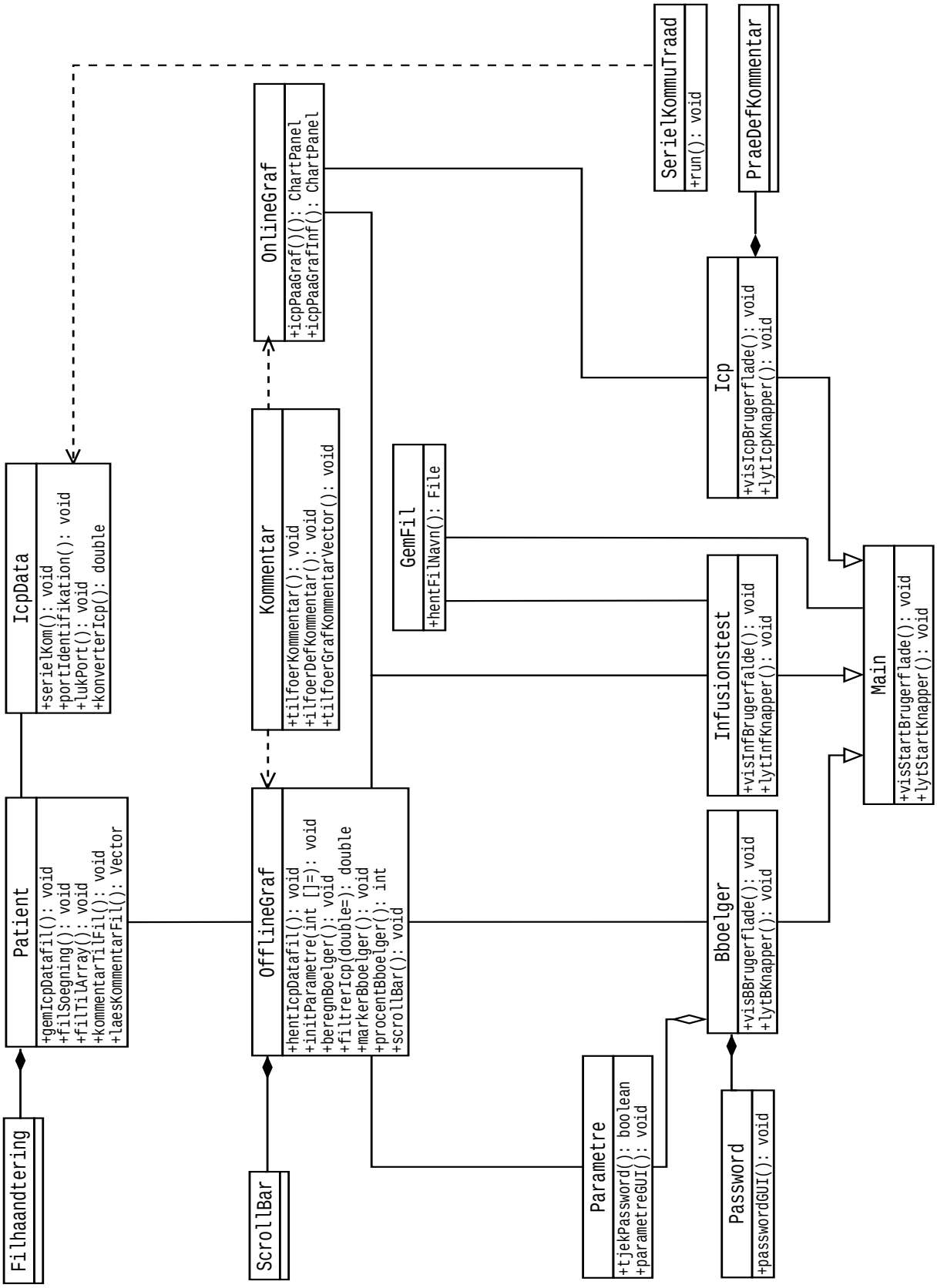
Entity

Den implementerede Entity-del sørger for at gemme data og kommentarer, samt sørger for at data senere kan hentes. Delen er opdelt i de to klasser: Patient og IcpData.

8.2 Klassediagram

Med udgangspunkt i de fundne klasser og metoder fra aktivitets- og sekvensdiagrammer i analyseafsnittet udarbejdes et klassediagram for designfasen. Klassediagrammet indeholder de relevante metoder, der skal implementeres i det samlede program. Klasserne samt deres metoder er yderligere beskrevet i appendiks D.

På figur 8.3 ses klassediagrammet for det samlede system. Det vælges at udelade attributter og kun fokusere på metoderne.



Figur 8.3: Klassediagrammet for designfasen viser relationen imellem klasserne samt hvilke metoder de indeholder

KAPITEL 9

Procesarkitektur

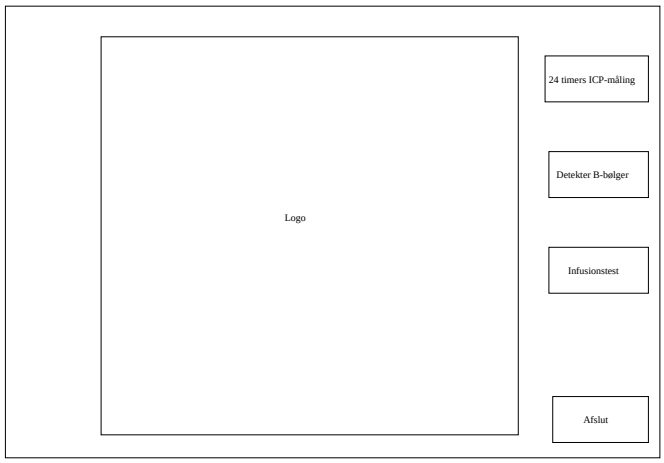
Procesarkitekturen er den dynamiske beskrivelse af systemet, der angiver, hvorledes objekterne interagerer med hinanden.

Procesarkitekturen vil blive beskrevet ved sekvensdiagrammer og tager udgangspunkt i den grafiske brugeflade Startskærm.

Softwareprogrammet eksekveres fra Main, hvor den første brugerflade Startskærm dannes. Et sekvensdiagram vil vise, hvilke objekter Startskærm består af, samt hvilke nye objekter der initialiseres ved tast på de forskellige knapper. På samme måde vises brugerflader for: 24-timers ICP-måling, B-bølgedetektering og Infusionstest.
De efterfølgende afsnit vil beskrive hovedfunktionaliteterne af, hvad der sker i diagrammerne. Detaljer skal læses af diagrammerne selv. Sekvensdiagrammerne forefindes sidst i afsnittet.

9.1 Startskærm

I figur 9.1 ses en skitse af startskærmen.

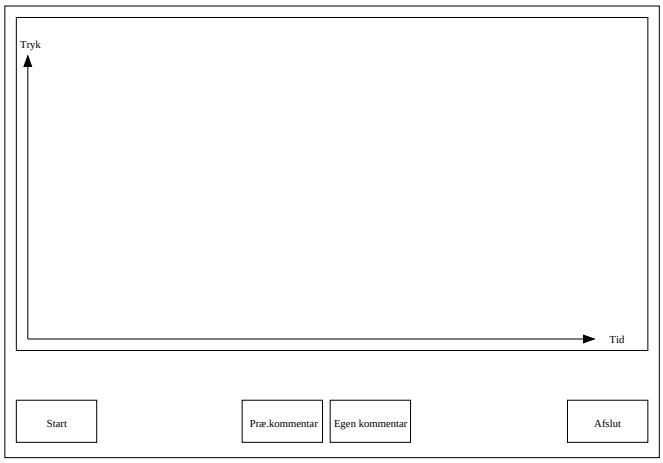


Figur 9.1: *Skitse af brugergrænsefladen for startskærmen*

I sekvensdiagram 9.5 vises det, at et objekt af klassen Main oprettes. Ved oprettelse af brugerfladen dannes knapper og tilhørende ActionListeners. Ved tast på disse knapper oprettes objekter for brugerfladerne for hhv. 24-timers ICP-måling, B-bølgeberegning og Infusionstest.

9.2 24-timers ICP-måling

I figur 9.2 ses en skitse af ICP-brugerfladen.



Figur 9.2: *Skitse af brugergrænsefladen for 24-timers ICP-måling*

I sekvensdiagram 9.6 vises det, hvordan et objekt af 24-timers ICP-brugerfladen oprettes. Der bliver oprettet objekter for opsætning af graf og for ActionListeners til knapperne.

Diagram 9.7 viser, hvad der sker ved tast på knapperne på brugerfladen. Ved tryk på knappen "Start" oprettes objekter til at hente data og starte seriel kommunikation. Endvidere ses det, hvordan det af sygeplejersken kan vælges at indtaste en kommentar. Ved tryk på knappen "Afslut" afbrydes kommunikationen med Caminoen.

9.3 B-bølgedetektering

I figur 9.3 ses en skitse af brugerfladen for B-bølgedetektering.

Figur 9.3: *Skitse af brugergrænsefladen for B-bølgedetektering*

I sekvensdiagram 9.8 beskrives det for B-bølgedetektering, hvordan brugerfladen oprettes. Der oprettes objekter for opsætning af graf og for ActionListeners til knapperne.

Tryk på knappen "Hent ICP-datafil" vises i sekvensdiagrammet 9.9. Her oprettes objekter til at hente en datafil og tilhørende kommentarer med tidspunkter, plotte data og tilføje scrollbar til grafen.

Tryk på knappen "Marker B-bølger" vises i sekvensdiagrammet 9.10. Her oprettes objekter til at beregne B-bølgeaktivitet og plotte det i grafen.

Tryk på knappen "Indstil parametre" vises i sekvensdiagrammet 9.11. Her oprettes efter godkendelse af password et objekt af klassen parametre, som muliggør, at lægen kan ændre de indstillinger B-bølgeaktivitet beregnes ud fra.

9.4 Infusionstest

I figur 9.4 ses en skitse af infusionsbrugerfladen.

Figur 9.4: *Skitse af brugergrænsefladen for Infusionstesten*

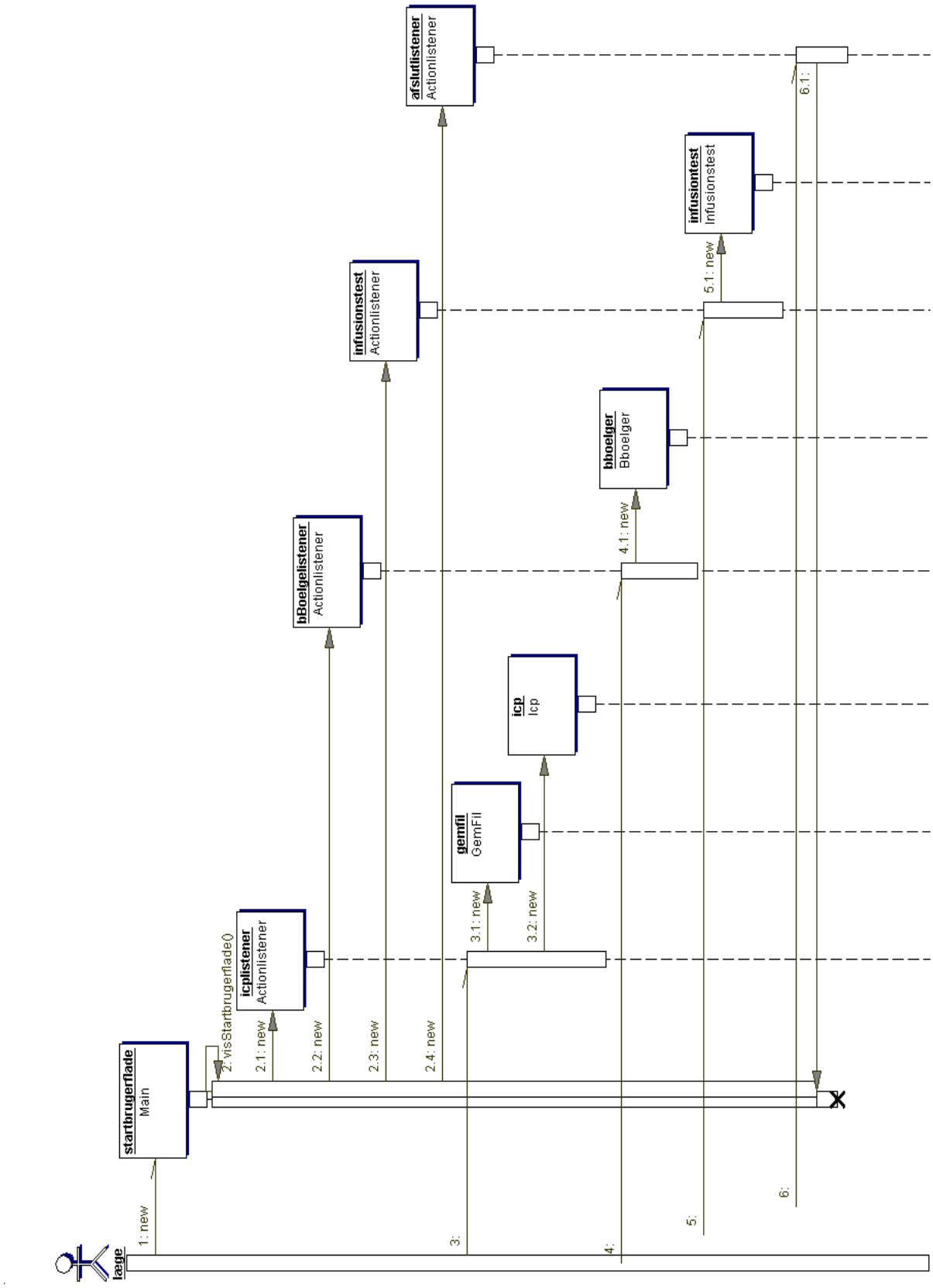
I sekvensdiagram 9.12 beskrives det for infusionstesten, hvordan brugerfladen oprettes. Der oprettes objekter for opsætning af graf og for ActionListeners til knapperne.

Tryk på knappen "Start" vises i sekvensdiagrammet 9.13. Her oprettes objekter til at gemme en datafil, hente data og oprette seriel kommunikation med Caminoen. De sidste to objekter initialiserer ved metodekald, der muliggør at data vises på grafen.

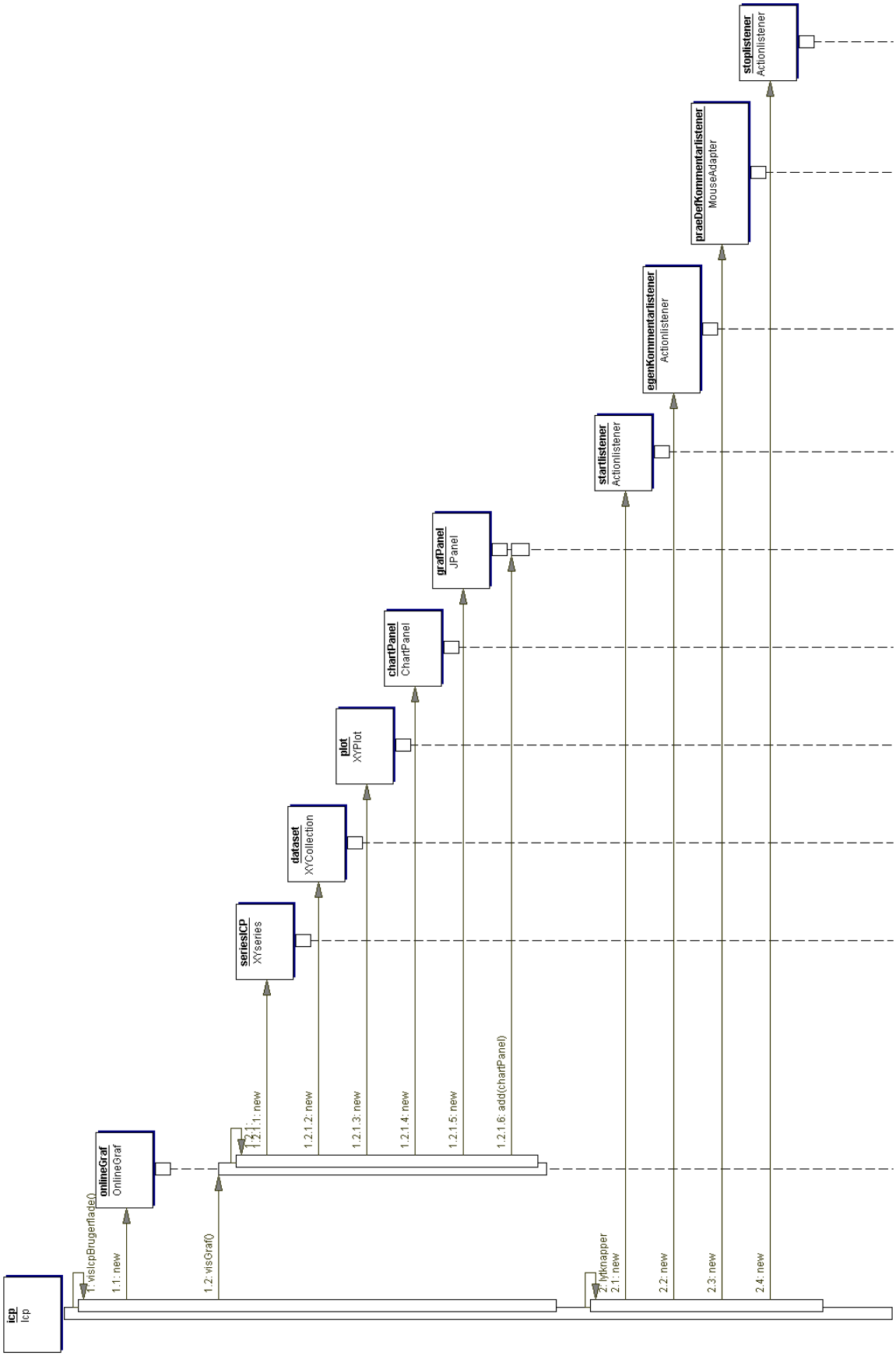
Tryk på knappen "Hent ICP-datafil" vises i sekvensdiagrammet 9.14. Her oprettes objekter til at hente en datafil og vise den på grafen.

Tryk på knappen "Filtrer signal" vises i sekvensdiagrammet 9.15, hvor objekter oprettes til at filtrere data og vise det på grafen.

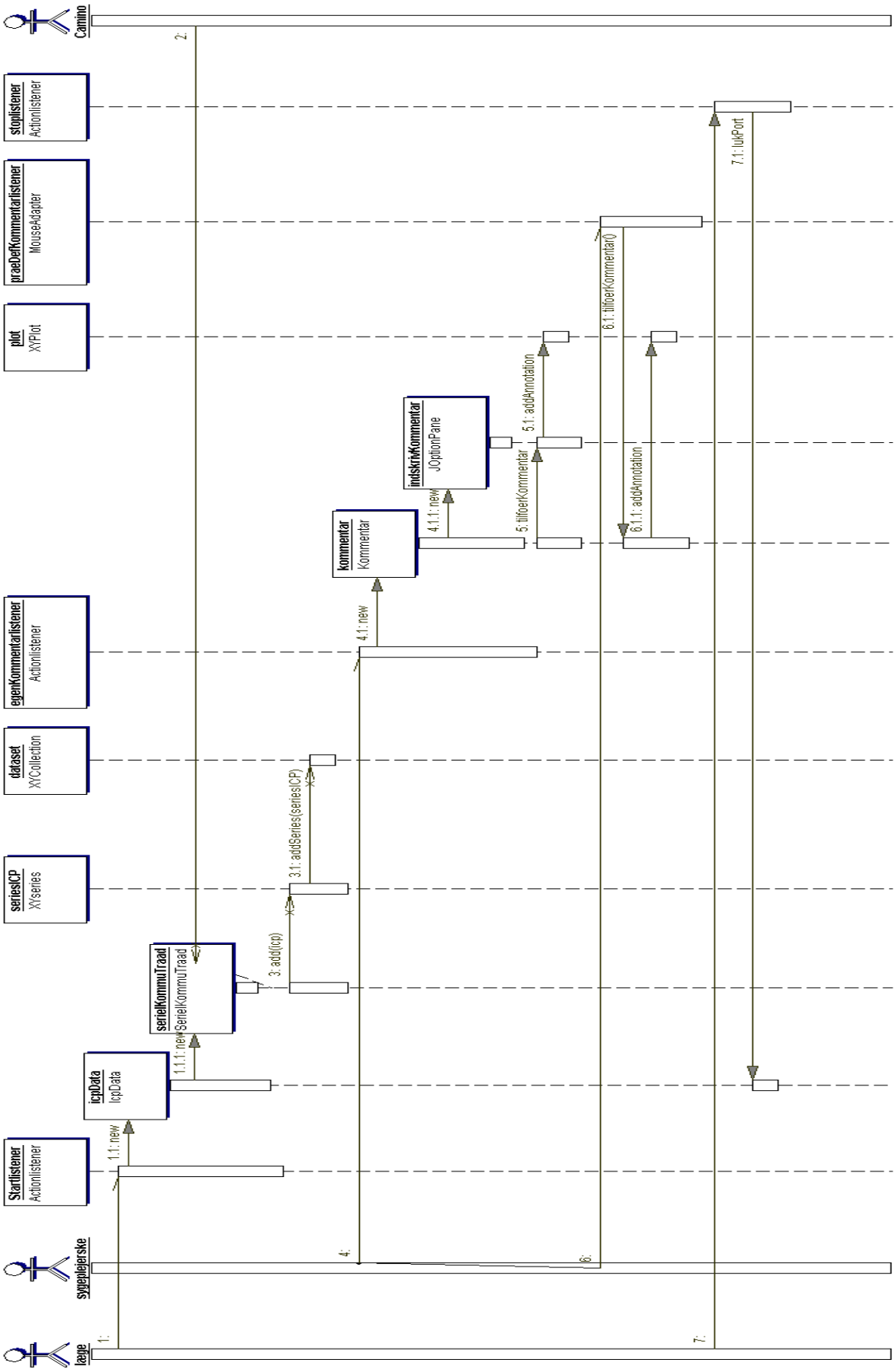
Sekvensdiagram 9.16 viser, at før der kan trykkes på knappen "Beregn R_{out}" skal der initialiseres et objekt ved to museklik i grafen og en indskreven infusionsrate.



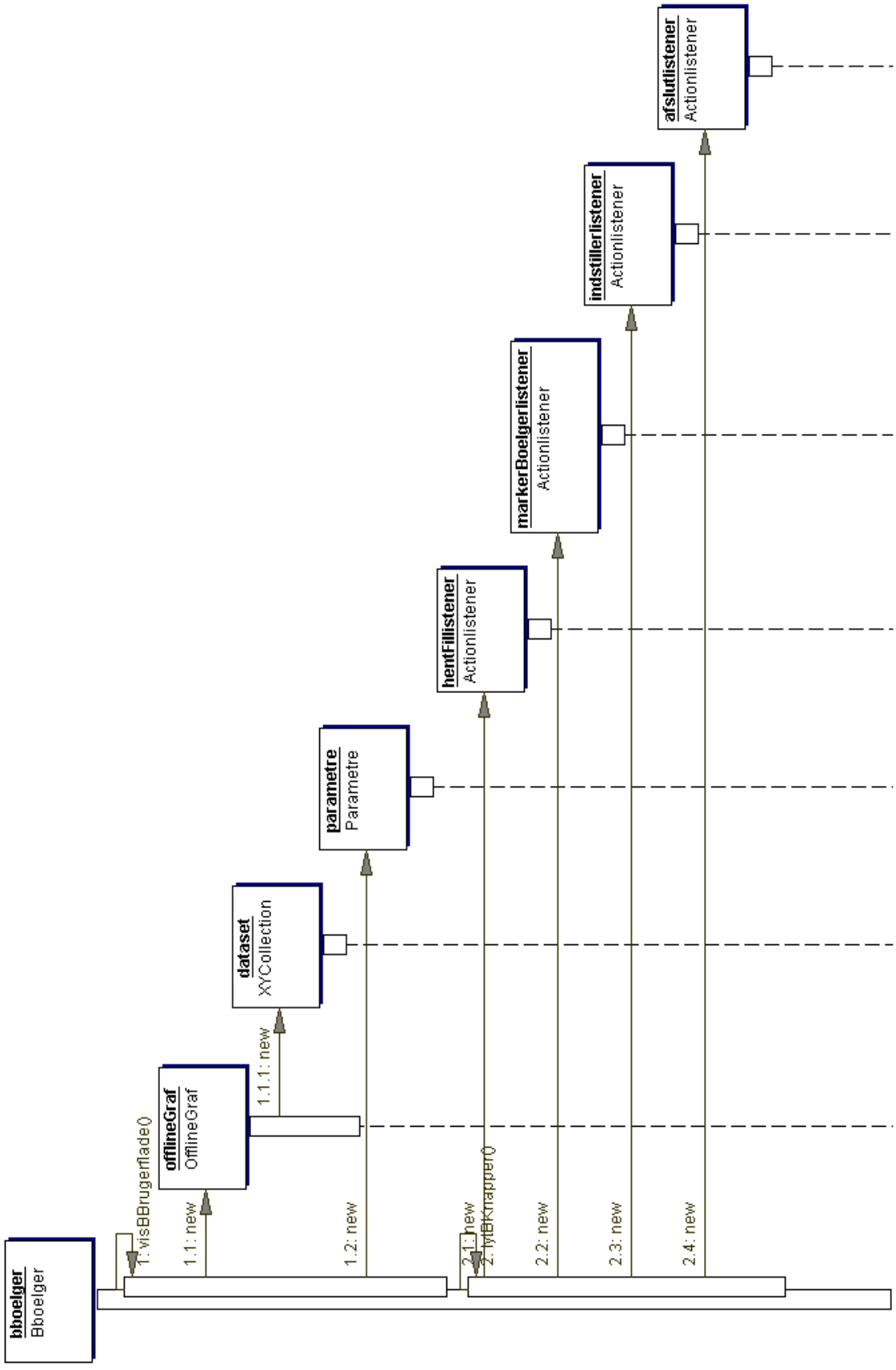
Figur 9.5: Oprettelse af startskærm samt aktørintervention



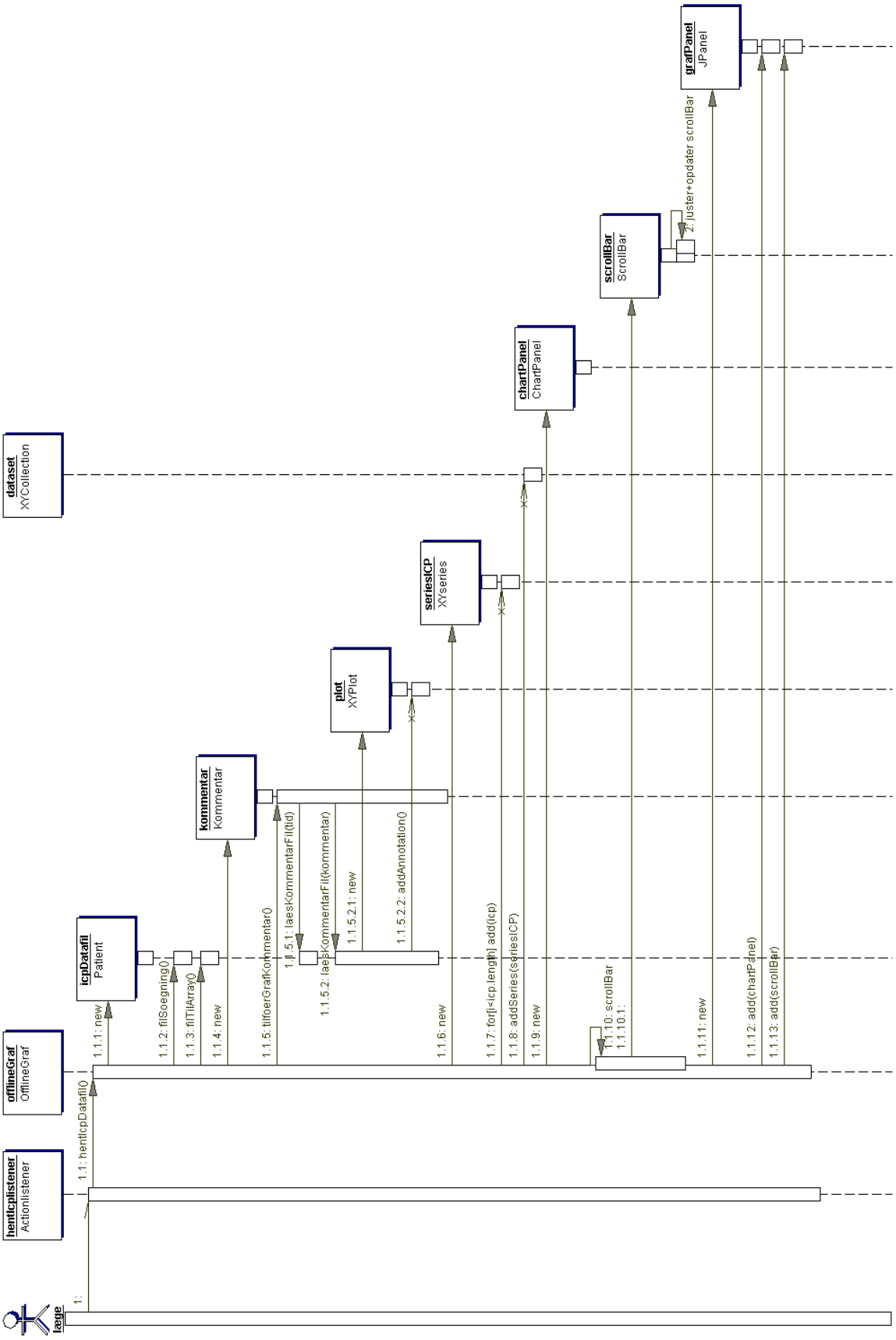
Figur 9.6: Oprettelse af ICP-brugerflade



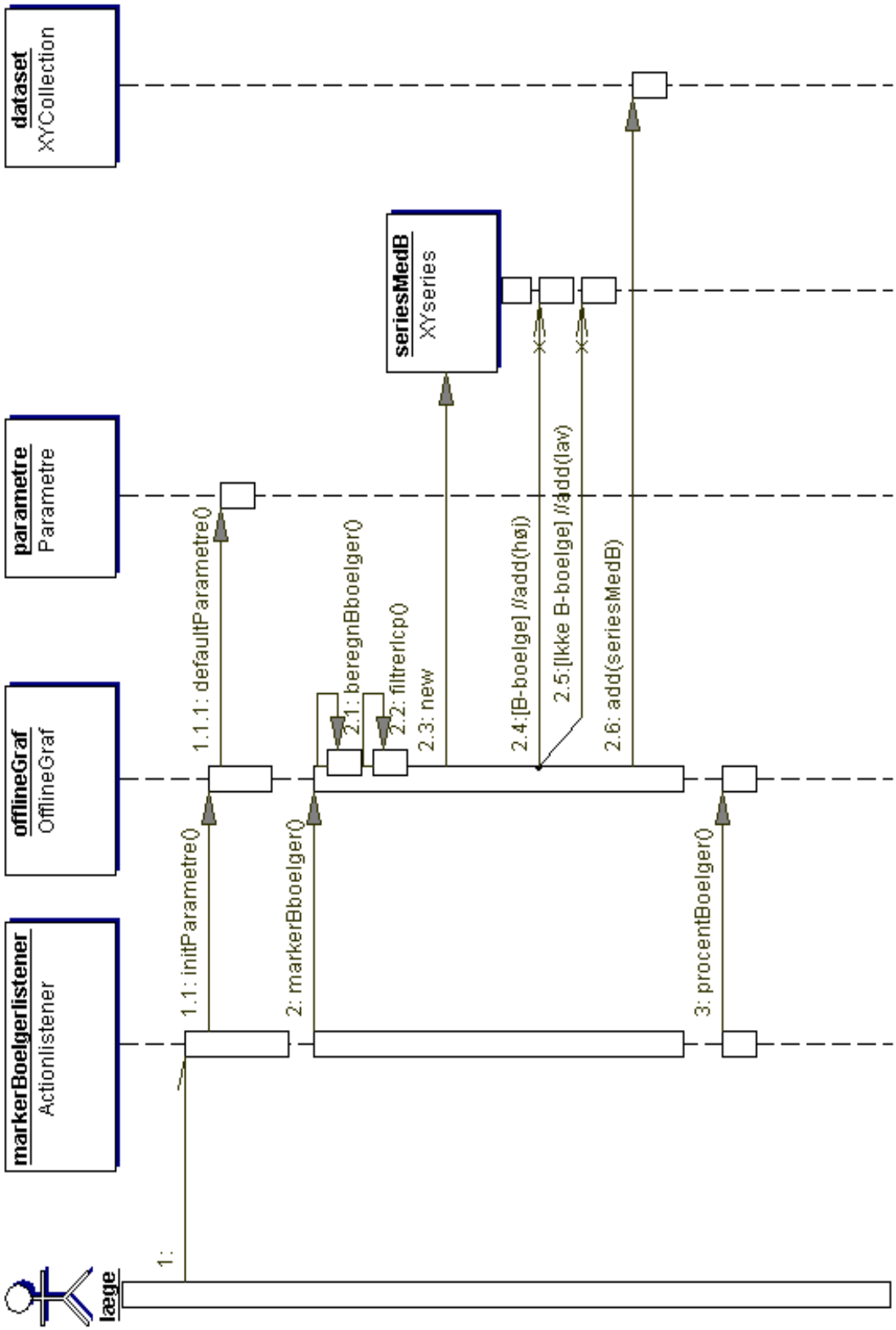
Figur 9.7: Initialisering af knapper på ICP-brugerflade



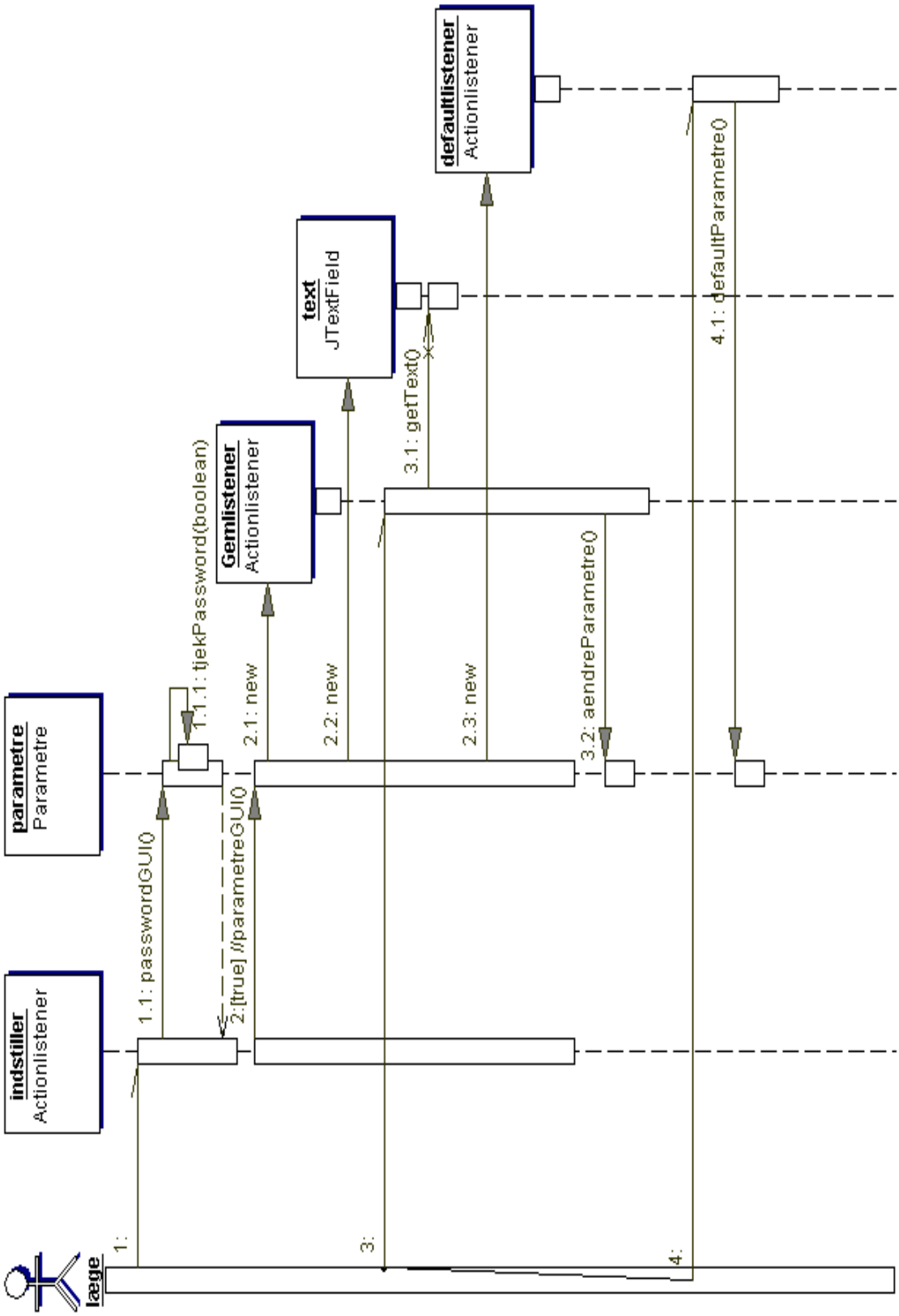
Figur 9.8: Oprettelse af B-bølgedetekterings-brugerflade



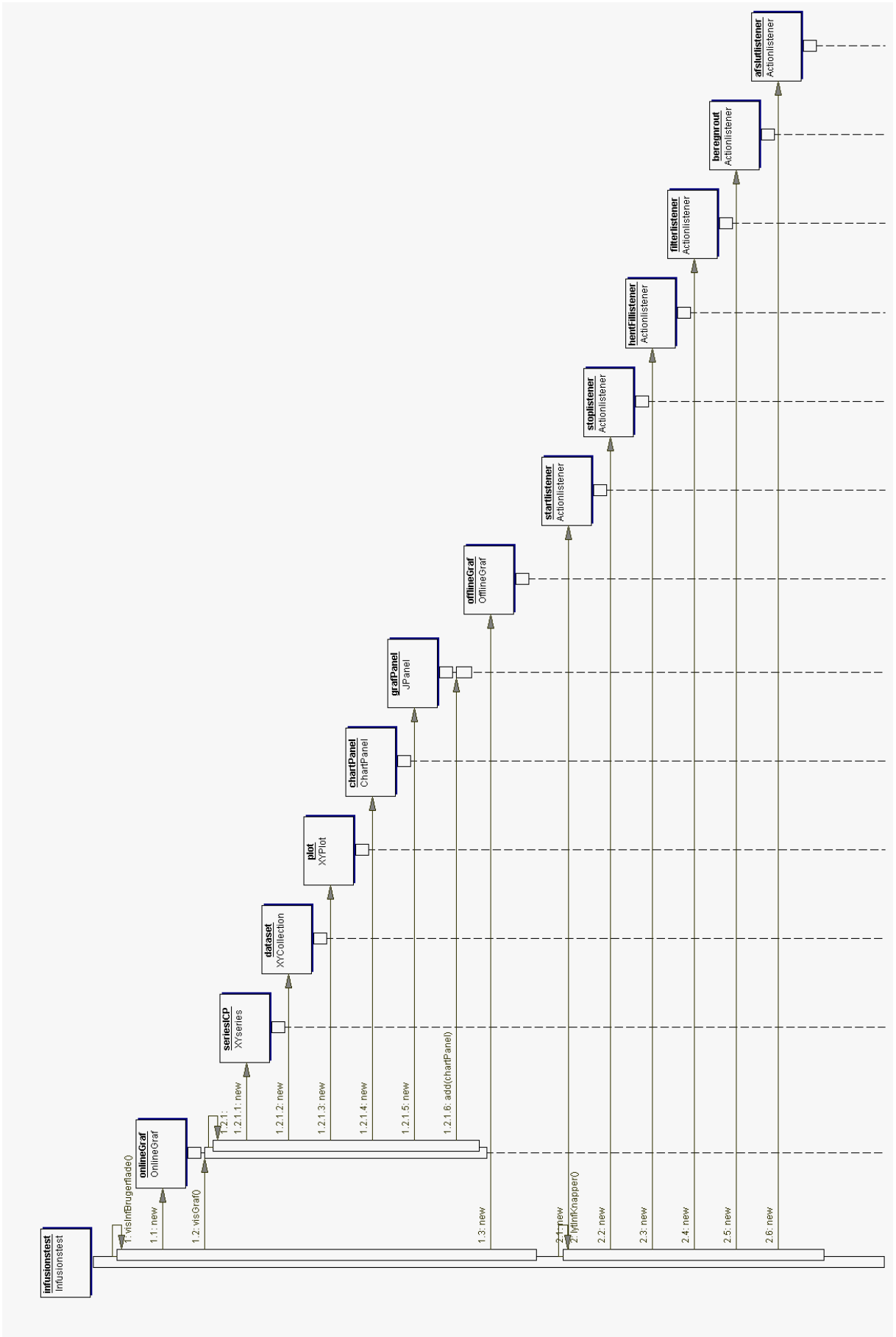
Figur 9.9: Initialisering af knappen "Hent Icp" på brugerfladen



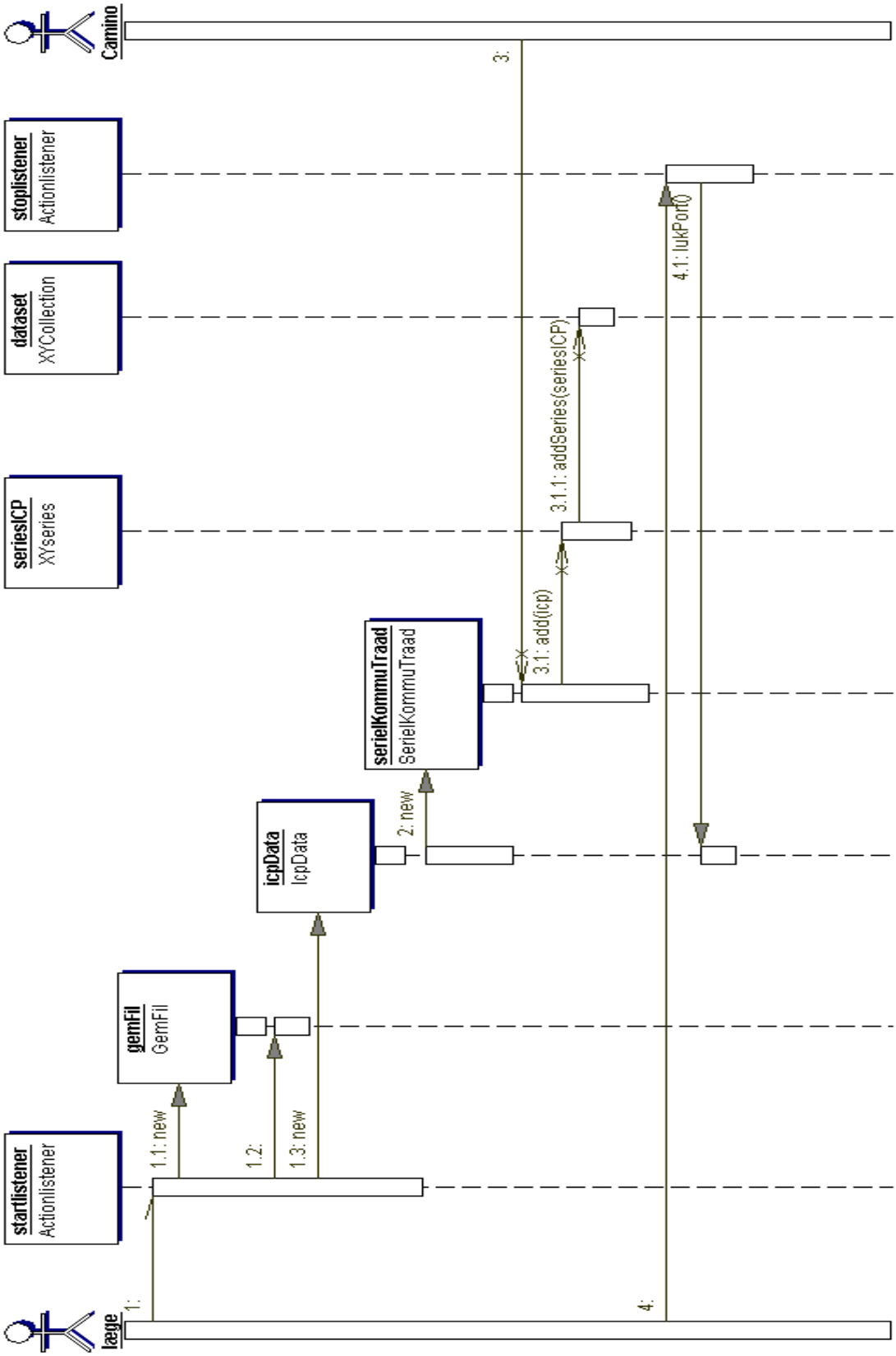
Figur 9.10: Initialisering af knappen "Marker B-bølger" på brugerfladen



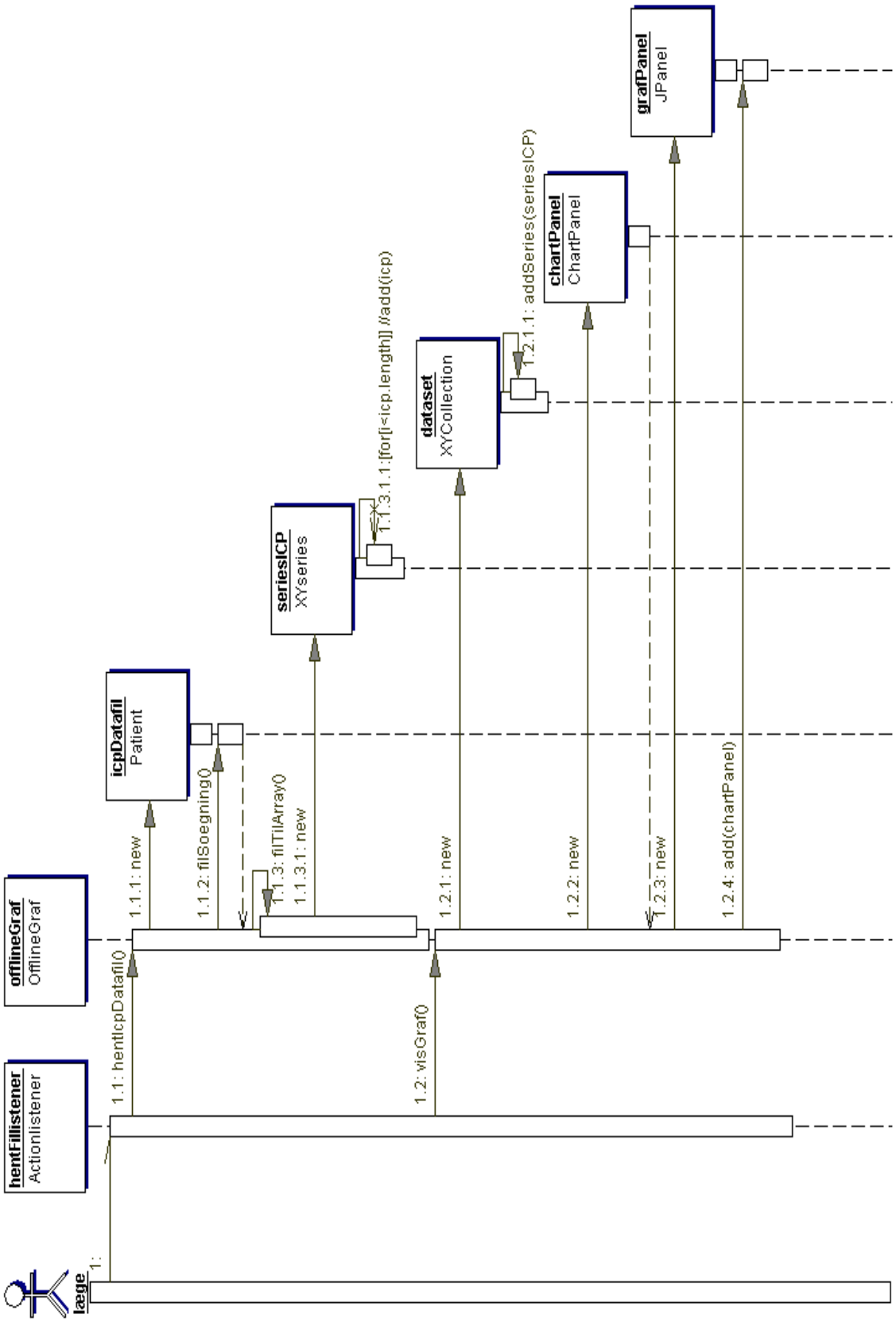
Figur 9.11: Initialisering af knappen "Indstil parametre" på brugerfladen



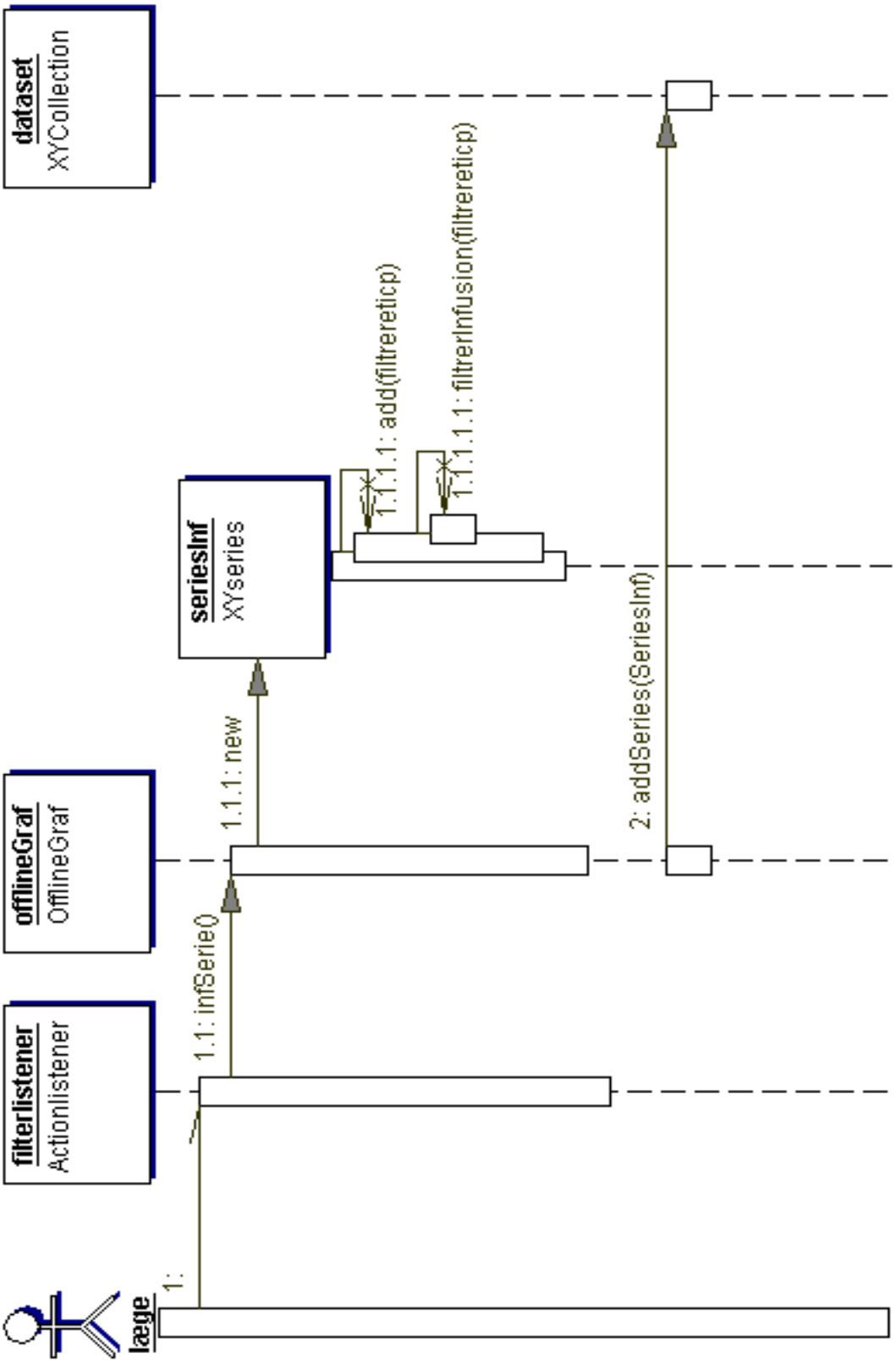
Figur 9.12: Oprettelse af infusionsbrugerflade



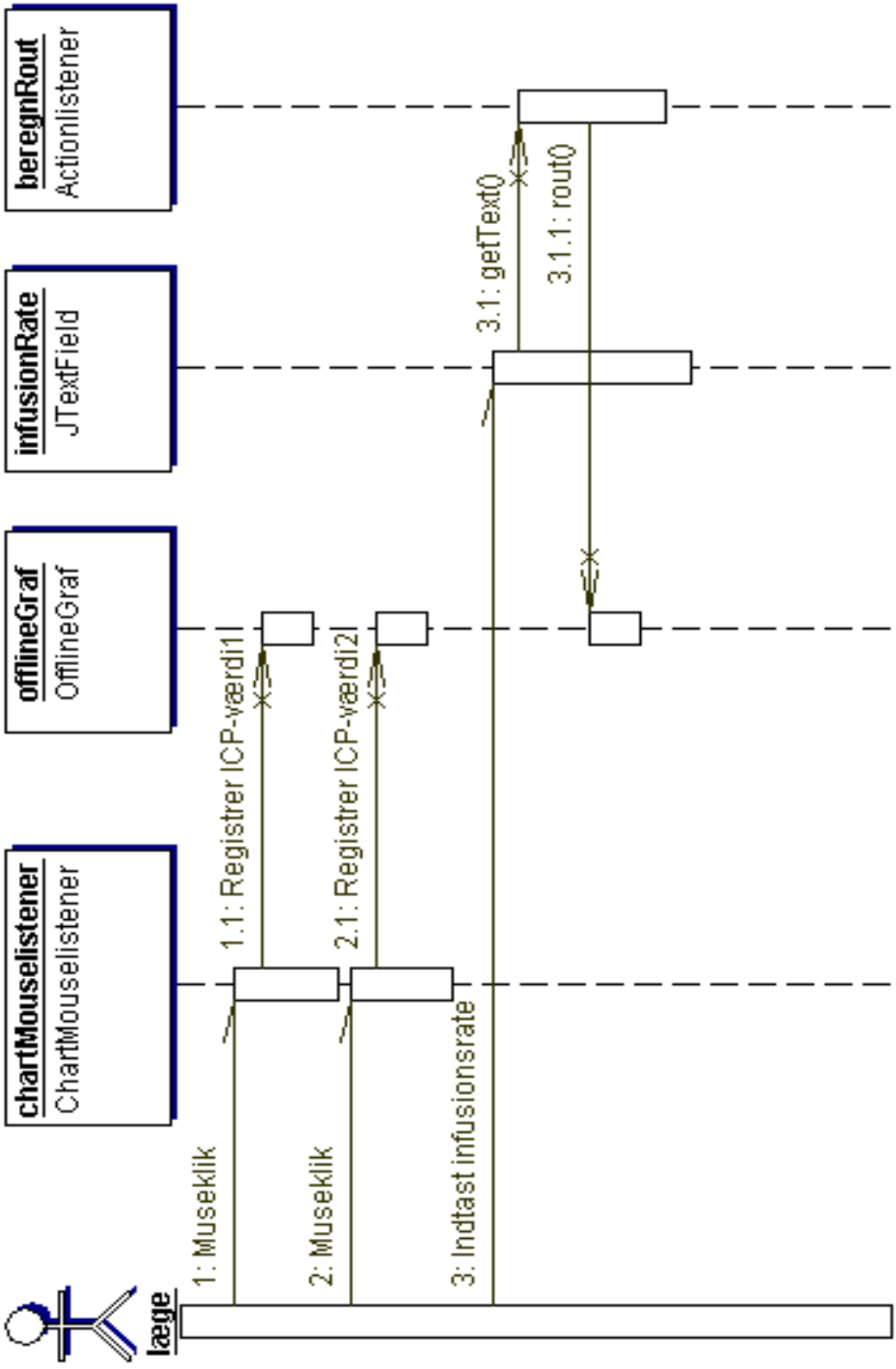
Figur 9.13: Initialisering af knapperne "start" og "stop" på infusionsbrugerfladen



Figur 9.14: Initialisering af knappen "Hent ICP-datafil" på infusionsbrugerfladen



Figur 9.15: Initialisering af knappen "Filtrer signal" på infusionsbrugerfladen



Figur 9.16: Initialisering af knappen "Beregn R_{out}" på infusionsbrugerfladen samt interaktørintervension

Del IV

IMPLEMENTATION OG TEST

KAPITEL 10

Implementation

Der redegøres i kapitlet for emner, der under implementationen har givet anledning til afvigelser fra design eller har vist sig besværlige at implementere. I det følgende beskrives Entity, Control og Boundary, og der beskrives et udpluk af kildekoden fra Boundary og Control.

Den samlede kildekode forefindes i bilag C samt på vedlagte CD-ROM [CD, A.3.1] , hvorfra programmet kan eksekveres.

Systemet er udviklet i Java SDK 1.4.1, og er optimeret til anvendelse på en computer med 14,1" skærm og en opløsning på 1024x768 dpi.

Der redegøres for udvalgte dele af kildekoden, da programmering er i fokus på dette semester. Brugen af JFreeChart-biblioteket er essentielt for systemet, hvorfor det vælges at beskrive klassen OnlineGraf i afsnit 10.2. Ligeledes er seriel kommunikation en vigtig funktion i systemet og speciel i forhold til Java, og derfor forklares håndteringen af dette i afsnit 10.3.

10.1 Entity

GemFil sørger for, at der for hver måling, der foretages, gemmes en datafil med ICP-data, og Patient sørger for, at der gemmes en kommentarfil og en fil med tiden, til hvilken kommentaren er indtastet. Det fandtes nemmest at arbejde med tre filer, hvor kommentar- og tidsfil automatisk åbnes, når en datafil åbnes. Filerne er gemt under samme filnavn, men med forskellige filtypenavne.
Der oprettes derfor tre typer af filer på følgende måde: 2003_05_23.type
Typen er .icp for data, .icpkom for kommentarer indtastet under målingen og .icptid for de tilhørende tider.
GemFil sørger også for at kontrollere, om der allerede ligger målinger for den pågældende dato. Hvis dette er tilfældet bliver brugeren bedt om at indtaste et andet filnavn, og det er ikke muligt at overskrive en fil.

Klassen Patient er implementeret for at sikre, at indtastede kommentarer gemmes og efterfølgende indsættes på offlinegraf til de rigtige tidpunkter.

10.2 Control

OnlineGraf og SerielKommuTraad sørger for, at data der kommer fra den serielle kommunikation bliver plottet på grafen. Det er vigtigt, at data plottes på grafen uden afbrydelser af den serielle

kommunikation, hvorfor der er anvendt en tråd til varetagelse af dette.
En tråd i Java er et sekventielt programforløb, der sikrer, at tråden tildeles den nødvendige CPU-tid.

Grundet den store datamængde der skal plottes på offlinegraf, er det overvejet, hvorledes data skal plottes. Når der laves en graf i JFreeChart visualiseres koordinatsystemet, hvorefter data kan plottes, ved at tilføje datapunkterne til serien, som herefter opdateres automatisk. Dette er en fordel ved visualisering af en onlinegraf.
Ved en offlinegraf bør datapunkterne tilføres serien før grafen visualiseres for at få vist hele dataserien hurtigere.

Det er ikke muligt at få vist klokkeslettet på x-aksen i hverken online- eller offlinegraf. I onlinegraf opstår der problemer med plotningen af data, da værdierne på x-aksen ikke kan tælles med modulus 60, så det passer med et klokkeslet.
Ved offlinegraf opstår der problemer med opdatering af scrollbaren. For at kunne opdatere scrollbaren er det nødvendigt med fortløbende værdier på x-aksen.
Projektgruppen har valgt at løse problemet ved at indskrive tidspunktet ca. hver halve time på offlinegrafen.

10.2.1 Klassen OnlineGraf

OnlineGraf er den klasse i Control, som designer graferne, der plotter opsamlet data, således det kan visualiseres på brugergrænsefladerne 24-timers ICP-måling og Infusionstest. Koden uddybes for de relevante elementer, der indgår i konstruktion af en graf.
For at tegne en graf i JFreeChart oprettes først et XY-datasæt, der kan plottes på brugerfladen. Dette gøres i metoden tegnGraf ved at oprette et nyt objekt af XYSeries, som laver en serie, hvor data indsættes. XYSerien sættes på grafen ved at tilhæfte den et objekt af XYSeriesCollection, som samler de serier, der skal plottes på grafen.
Idet 24-timers ICP-målings- og infusionstestgraferne varierer i udseende, laves to metoder til håndtering af disse. Dog skal grafen laves på samme måde, så denne laves i tegnGraf()-metoden. Begge metoder icpPaaGraf() og icpPaaGrafinf() kalder tegnGraf() og returnerer chartPanel, således disse metoder kan kaldes, hvor brugergrænsefladen laves. Metoden icpPaaGraf() ændrer på x- og y-aksens egenskaber. x-aksen bliver sat til at være autoskaleret, hvilket gør, at x-aksen tilpasser sig løbende med XYplottet. Yderligere bestemmes x-aksens samplesinddeling og y-aksens maksimums- og minimumsskalering.
Metoden icpPaaGrafinf() sætter x-aksen til at have start- og slutværdier, som svarer til 2 timers samples, og y-aksens maksimums- og minimumsskalering sættes.

```
//De nødvendige pakker importeres fra JFreeChart-pakken
import com.jrefinery.data.*;
import com.jrefinery.chart.*;
import com.jrefinery.chart.plot.XYPlot;
import com.jrefinery.chart.annotations.XYTextAnnotation;

/***Klassen OnlineGraf***/
public class OnlineGraf
{
/***Attributter**********/                                                    10
    private ChartPanel chartPanel;
    private XYPlot plot;

/***Metoder**************/
    /*Tegn onlineGraf*/
    public void tegnGraf(String labelxakse, String overskrift)
    {
        XYSeries series = new XYSeries("ICP"); //Laver nyt objekt af XYSeries
        XYSeriesCollection dataset = new XYSeriesCollection(); //Laver nyt objekt af XYSeriesCollection
        dataset.addSeries(series); //Inkluderer serien således den kan vises på grafen   20
        //Laver et chart som er et XY-koordinatsystem
        JFreeChart chart = ChartFactory.createLineXYChart(overskrift, //Overskrift på grafen
                                                labelxakse, //Label på X-akse
                                                "ICP (mmHg)", //Label på Y-akse
                                                dataset, //Data der skal vises
                                                true, //Grafforklaring
                                                false; //Værktøjstip
                                                false); //URL sættes
        plot = chart.getXYPlot(); //Henter plottet som laves i chart
        chartPanel = new ChartPanel(chart); //Sætter chart på et panel så det kan visualiseres   30
    }

    /*Laver grafen til 24-timers ICP-måling*/
    public ChartPanel icpPaaGraf()
    {
        tegnGraf("Tid", "Intrakraniel Trykmåling (ICP)"); //Grafen skal laves først
        ValueAxis xaxis = plot.getDomainAxis(); //Således x-aksen kan manipuleres
        xaxis.setAutoRange(true); //Grafen vil glide over skærmen og opdateres automatisk
        xaxis.setFixedAutoRange(2570.0); //Det visuelle område på skærmen, svarer til 50 min, dvs. ca 30 cm/t
        ValueAxis yaxis = plot.getRangeAxis(); //Således y-aksen kan manipulers              40
        yaxis.setRange(-20.0, 80.0); //Sætter Y-aksens min. og max.
        return chartPanel; //Metoden returnerer et panel, som kan sættes på brugerfladen
    }

    /*Laver grafen til Infusionstesten*/
    public ChartPanel icpPaaGrafInf()
    {
        tegnGraf("Tid (2 timer)", "Infusionstest"); //Grafen skal laves først
        ValueAxis xaxis = plot.getDomainAxis(); //Således x-aksen kan manipuleres
        xaxis.setRange(0.0 ,7200.0); //Der vises 2 timer på skærmen                        50
```

```
        ValueAxis yaxis = plot.getRangeAxis(); //Således y-aksen kan manipulers
        yaxis.setRange(−20.0, 80.0); // Sætter Y-aksens min. og max.
        return chartPanel; //Metoden returnerer et panel, som kan sættes på brugerfladen
    }
}
```

10.3 Boundary

Der fokuseres på grænsefladen til Caminoen, som er speciel for systemet. IcpData er den klasse, der sørger for dataopsamling. Caminoens protokol for seriel kommunikation er vedlagt i appendiks E.
Computeren skal identificere den serielle port, der er tilsluttet Caminoen. Porten bliver herefter opsat til at modtage data ud fra de specifikationer, der er givet for Caminoen. Den serielle kommunikation kører herefter på events fra serielporten, dvs. når der er data på porten modtages og gemmes disse. Uddrag af kildekoden til metoden serialEvent() findes efterfølgende, og er uddybet med kommentarer.
Da det er data i form af bytes, der modtages på porten, og disse ikke direkte kan bruges videre i systemet, konverteres disse til doubleværdier.
Der sørges for, at serielporten lukkes igen, når den ikke længere benyttes. Hvis ikke, kan der ikke foretages en ny dataopsamling medmindre hele programmet lukkes ned og startes op igen.

Uddrag af kildekoden til metoden serielEvent() uddybes i det følgende med kommentarer:

```
/*serialEvent*/
    public void serialEvent(SerialPortEvent event)
        {
        switch (event.getEventType())
        {
        case SerialPortEvent.BI://Break interrupt
        case SerialPortEvent.OE://Overrun error
        case SerialPortEvent.FE://Framing error
        case SerialPortEvent.PE://Parity error
        case SerialPortEvent.CD://Carrier detect                                    10
        case SerialPortEvent.CTS://Clear to send
        case SerialPortEvent.DSR://Data set ready
        case SerialPortEvent.RI://Ring indicator
        case SerialPortEvent.OUTPUT_BUFFER_EMPTY://Output buffer er tom
            break;//I alle ovenstående situationer skal der hoppes ud af metoden
        case SerialPortEvent.DATA_AVAILABLE://Data er tilgængelig på den serielle port
            byte[] readBuffer = new byte[1000];//Oprettelse af byte array med 1000 pladser
                try
                {
                int str = 0;                                                        20
                        byte readBuffer1;
                        //While løkken kører så længe der er data tilgængelig på seriell porten
                        while (inputStream.available() > 0)
                        {
                        int numBytes = inputStream.read(readBuffer);//angiver antallet af bytes i bufferen
```

```
                    if (numBytes>stoerst)
                    {
                    stoerst=numBytes;
                    System.out.println("I stoerst står:" + stoerst);
                    }                                                          30
                    for(str = 0; str < numBytes; str++)//læser alle bytes i bufferen
                    {
                            readBuffer1 = readBuffer[str];//readBuffer1 angives i ASCII
                            if (t == true)
                            {
                            switch (readBuffer1)
                            {
                                    case 65://Situationen at der står B i readBuffer1
                                    case 88://Situationen at der står X i readBuffer1
                                    case 120://Situationen at der står x i readBuffer1   40
                                    case 89://Situationen at der står Y i readBuffer1
                                    case 90://Situationen at der står Z i readBuffer1
                                    t = false; break;
                                    case 82: break; //Der står R i readBuffer1
                                    default:
                                        icpArray[j]=readBuffer[str];
                                                j++;
                            if (j == 4)
                            {
                            taeller=taeller+1;//angiver antal samples               50
                            sum =konverterIcp(icpArray)+sum;
                                    //skal kun starte serielKommuTraad 1 gang i sek.
                                    if (taeller==190)//camino sender data med 190Hz
                                    {
                                    midletIcp=sum/190;//midles til 1Hz
                                    sum=0;
                                    Thread serielKommuTraad = new SerielKommuTraad();
                                    taeller=0;
                                    //Udfører handling, når GUI har tid til at opdatere
                                    SwingUtilities.invokeLater(serielKommuTraad);   60
                                    pw.println(midletIcp);//skriver til fil
                                    }
                                    t = false;
                                    j = 0;
                                    }
                            }
                    }
                else
                    {
                            if (readBuffer[str] == 82)                              70
                            {
                                t = true;
                                i = 0;
                            }
                    }
                    }
}
```

```
    }
        //Ved input output fejl, skrives der en fejlmeddelse til skærmen
        catch (IOException e) {JOptionPane.showMessageDialog(null,                    80
                               "Kan ikke skrive til fil! Prøv igen!",
                               "Filskrivningsfejl", JOptionPane.WARNING_MESSAGE);}
    break;
}
}
```

KAPITEL 11

Test

Dette kapitel har til formål at give et overblik over, hvorledes programmet er blevet testet, og hvorfor det er blevet testet på den måde.

Test af et program er en vigtig del af et systems udvikling. Der bruges ofte op til 40% af udviklingsperioden på at teste, hvilket gør testningen til en bekostelig affære både tidsmæssigt og økonomisk. Alligevel er det ofte en god investering, idet prisen for at rette eventuelle fejl efter udviklingsperioden kan løbe op i 100 gange den pris, det koster under udviklingen [23]. Derfor findes det vigtigt at beskrive, hvordan systemet er testet.
Der vil blive fokuseret på metoderne beskrevet i appendiks F, dvs. whiteboxtest og blackboxtest. Desuden inddrages nogle eksempler fra systemet, som er vigtigte at dokumentere.

11.1 Valg af testmetoder

Gennem udviklingsprocessen har projektgruppen anvendt forskellige tests. Overordnet er der testet ved følgende tre metoder:

- Enhedstest
- Integrationstest
- Systemtest

Enhedstest er anvendt under opbygning af de enkelte klasser, oftest foretaget som whiteboxtest for specifikke datastukturer såsom løkker. Ydermere er de enkelte klasser, hvor muligt, testet vha. blackbox-metoden.

Integrationstesten er blevet anvendt efter de forskellige klasser er blevet programmeret, når disse er blevet integreret i et samlet system. Systemet er gradvist blevet samlet i større og større grupper af klasser, der var naturligt sammenhængende. Disse grupper er blevet testet, hver gang en ny klasse er blevet tilføjet.
Integrationstesten er blevet udført som en blackboxtest, hvor kendt input gav forventet output. Der er anvendt en bottom-up metode [23], hvor små moduler gradvist er samlet til et hovedprogram.

Systemtest er den endelige test af det samlede system, og systemet er blevet testet efter blackbox-metoden af gruppen selv. Den endelige systemtest er en funktionalitetstest. Denne er er analog med en accepttest, forskellen er blot, hvem der tester. I systemtesten er det udviklere, mens det i accepttesten er køberen eller brugeren, der udfører testen.

Enhedstest og integrationstest er blevet benyttet til løbende at finde fejl og rette dem. Under systemtesten ønskes det at teste, om programmet indeholder de funktionaliteter, der er beskrevet i use-case diagrammet figur 5.6 på side 27. De tre hoveddele af systemet: 24-timers ICP-måling, B-bølgedetektering og Infusionstest er blevet testet hver for sig.
Testen har haft til formål at afsløre funktionalitetsfejl. Disse fejl kan opstå som funktionelle fejl, hvilke indikerer, at en funktionalitet er manglende eller ukorrekt, nonfunktionelle fejl, der indikerer, at en funktionalitet udføres for langsomt eller logisk fejl [21].

Da både integrationstest og systemtest er funktionalitetsbetingede og ikke tager højde for, om de implementerede klasser behandler data på den ønskede måde, ønskes det derfor yderligere at verificere:

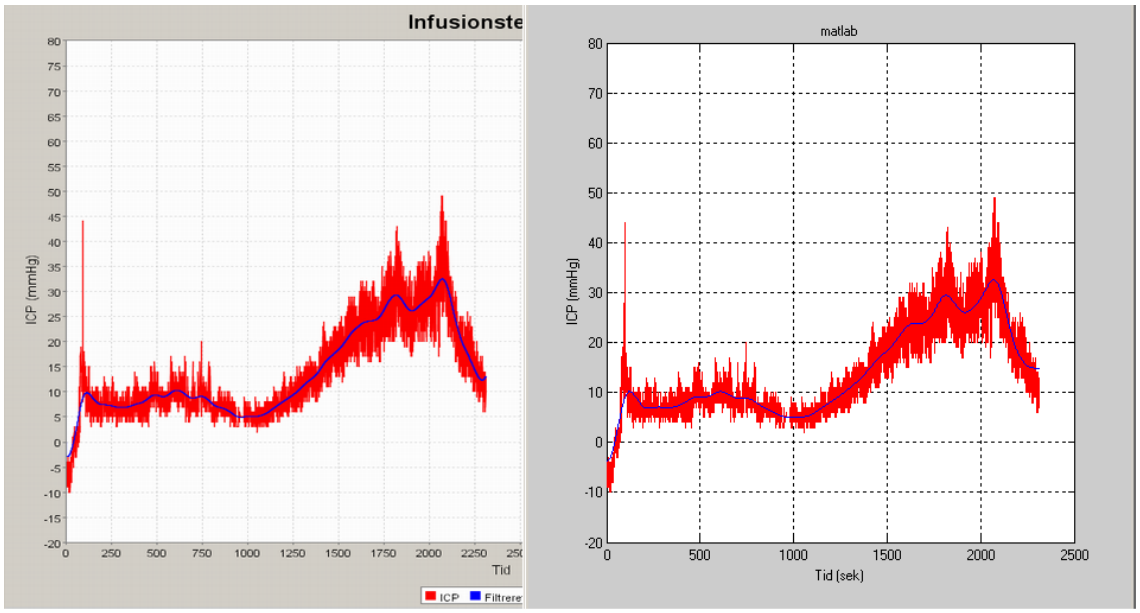
- at data filtreres korrekt
- at B-bølger findes korrekt
- at data modtages korrekt i den serielle kommunikation

11.2 Udførte tests

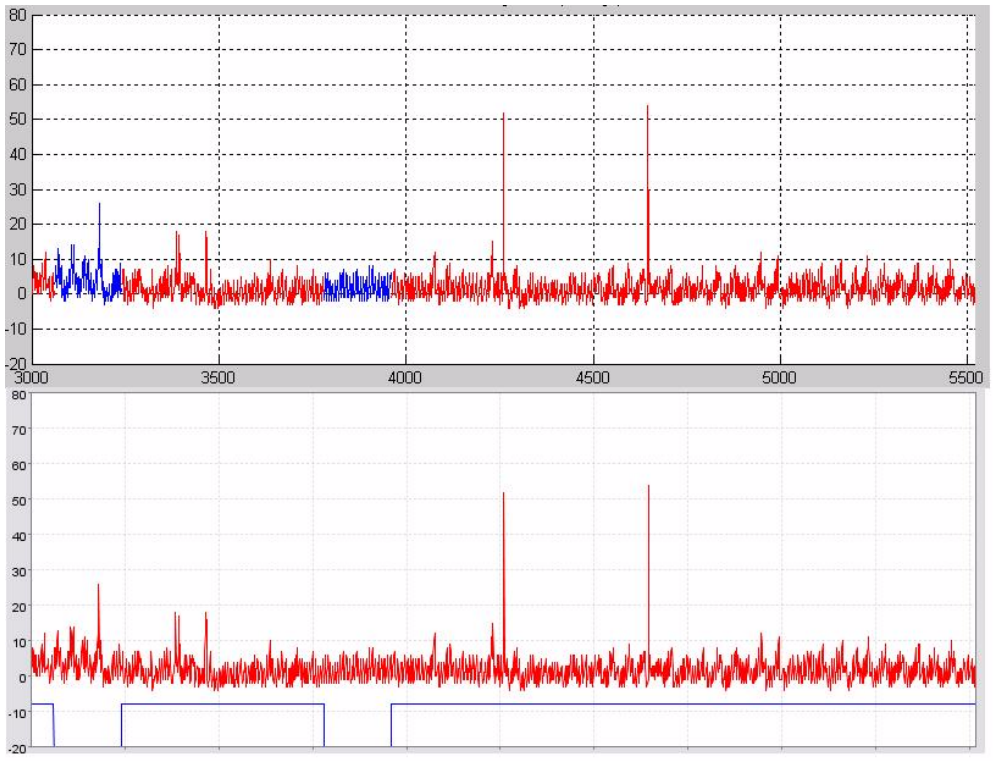
Test:	**Test af filtre**
Formål:	At finde ud af, om programmets filtre er implementeret korrekt.
Metode:	Filtrerne testes ved at filtrere forskellige signaler vha. de implementerede filtre og sammenligne resultatet med Matlab, for derved at kunne teste om output stemmer overens.
Resultat:	De implementerede filtre filtrerer som forventet. Skærmprint fra Matlab og systemet kan ses i figur 11.1

Test:	**Test af B-bølge detektion**
Formål:	At finde ud af, om den implementerede algoritme til detektering af B-bølger er implementeret korrekt.
Metode:	Algoritmen til detektering af B-bølger testes ved at lade algoritmen foretage sin beregning på flere signaler. Det kontrolleres herefter, om der er fundet den samme bølgeaktivitet.
Resultat:	Skærmprint fra Matlab og systemet kan ses i figur 11.2. Som det fremgår af figuren stemmer de markerede områder overens, hvilket betyder, at algoritmen er implementeret korrekt.

Test:	**Test af seriel kommunikation**
Formål:	At teste om seriel kommunikationen er implementeret korrekt
Metode:	Der opsamles et signal gennem den serielle forbindelse til Camino MPM-1, og data plottes på grafen. Samtidig med dette printes signalet ud vha. sygehusets analoge printer. Det ses herefter, om de to signaler stemmer overens. Desuden testes det, om systemet kan håndtere afbrydelse af kommunikationen enten ved at slukke for Caminoen eller ved at trække det serielle kabel ud.
Resultat:	Der er gennem 3×20 timer opsamlet data fra Caminoen. Samtidig er tryksignalet udskrevet på printeren med en papirfremføringshastighed på 30 cm/t. På trods af en samplingshastighed på 1 Hz ses ingen betydelig visuel forskel på målingerne. Hvis Caminoen slukkes eller serielkablet fjernes, stopper opsamlingen af data. Når der tændes eller tilkobles igen, fortsætter dataopsamlingne som forventet. Opsamlingsbufferen på serielporten er sat til at have 1000 pladser. Dette er under testen fundet tilstrækkeligt, da der under normal brug af computer maksimalt blev optaget ca. 500 pladser, mens der under ressourcekrævende arbejde blev optaget op til 850 pladser.



Figur 11.1: *Viser skærmprint af en infusionstest filtreret af hhv. systemet og Matlab. Systemets filtrering ses til venstre. På x-aksen ses tiden (2500 sek) og på y-aksen trykket i mmHg.*



Figur 11.2: *Viser skærmprint af en ICP-måling, hvor B-bølgeaktivitet er detekteret af hhv. Matlab og systemet, hvor systemets detektering er angivet nederst. Systemet har angivet B-bølgeaktivitet ved en eleveret linie, mens Matlab har farvet B-bølgeaktivitet rødt. På x-aksen ses tiden (2500 sek) og på y-aksen trykket i mmHg.*

KAPITEL 12

Diskussion

Systemet er udviklet ved objektorienteret analyse og design, hvor UML er blevet anvendt som modelleringssprog.
Der ses både fordele og ulemper ved brugen af UML. Projektet har taget udgangspunkt i en kravspecifikation udarbejdet på et tidligere semester, men gruppen valgte at revurdere kravspecifikationen for at gennemgå alle faser i UML. Det har imidlertid vist sig problematisk at abstrahere fra tidligere ideer omkring, hvordan systemt først var tænkt. Dette har genspejlet sig i processen, hvor det var svært at adskille faserne, og ikke fra begyndelsen stille teknisk detaljerede krav. Derfor er det gruppens erfaring, at UML er et værktøj, der, for at få det bedste udbytte, skal benyttes fra analysens begyndelse.
Fordelene ved UML har imidlertid vist sig at være, at det indeholder diagrammer, der kan fremstille systemet fra mange vinkler, alt efter hvad fokus er, hvilken fase udviklingen befinder sig i, og hvem diagrammerne henvender sig til.

Under implementering af systemets funktionaliteter har der vist sig visse problemer med benyttelse af Java som programmeringssprog.
Det har været besværligt at implementere graffunktionerne samt håndtere den serielle kommunikation. Desuden er den store datamængde, der skal håndteres ved detektering af B-bølger, tung at arbejde med, når der skal scrolles i grafen. En utålmodig bruger ville muligvis finde dette generende. En umiddelbar løsning kunne være benyttelse af en pc med større kapacitet end den pc, der er benyttet ved udvikling. Her er benyttet en 1066 MHz Dell LATITUDE med 384 MB RAM og Windows XP pro. Ved fremtidig udvikling af lignende systemer, der bygger på signalbehandling, bør andre programmeringssprog tages i betragtning.

Det har ikke været muligt at sætte et klokkeslet på x-aksen, som normalt vil forventes, fordi der var en uoverensstemmelse mellem datatyperne, der skal påføres grafen for hhv. at visualisere klokken og kommentarer.
Kommentarerne blev prioriteret højst, og valget kan forsvares ved, at der hver halve time plottes et tidspunkt på grafen. Dermed vil der altid for hvert skærmbillede være adgang til et tidspunkt, hvilket allerede er en forbedring af det tidligere system.

Der er ikke fastlagt endegyldige parametre for en algoritme til detektering af B-bølgeaktivitet. For det første, fordi der ikke har været et datagrundlag at foretage en undersøgelser på, og for det andet, fordi det ikke er muligt nå frem til et endeligt svar uden at benytte en subjektiv visuel vurdering som reference. Mange parameterværdier kan opfylde parameterkarakteristikken, som er sat ud fra litteraturstudier og ud fra spørgeskemaundersøgelser på 5. semester. Det har været formålet at standardisere en metode, der gør det muligt for lægen at dokumentere, på hvilket

grundlag han træffer sin beslutning og eliminere erfaringsbaseret mønstergenkendelse. Dette vil gøre det muligt at erhverve viden om signalet og dets kliniske betydning og være et grundlag for at sammenligne resultater artikler, hospitaler og læger imellem. Dette er, hvad et endegyldigt svar kræver, og dette er, hvordan systemet tænkes benyttet af brugeren.

Der skal ved benyttelse af systemet ses kritisk på den grænse for 50% B-bølgeaktivitet i en målingsperiode, som i dag er sat for patologisk bølgeaktivitet. Det forventes, at der ved en digital opsamling og behandling af signalet vil detekteres flere perioder med bølgeaktivitet, end hvad visuelt er tilgængeligt.

Der er i designafsnittet sat krav til systemets kvalitetsfaktorer. Udvidelsesvenlighed er vurderet til meget vigtig for at give mulighed for at implementere nye funktionaliteter. Grundet den manglende erfaring med programmering i Java er udvidelsesvenligheden mindre end ønsket. Dog fremgår opbygningen af programmet af diagrammerne.

KAPITEL 13

Konklusion

Der er udviklet et system, der håndterer det intrakranielle tryksignal, og systemet virker efter hensigterne i forhold til de opsatte krav i requirementanalysen.
Det intrakranielle tryk registreres, opsamles og visualiseres på en computerskærm og gemmes i en fil. Afhængigt af om der vil foretaget en 24-timers ICP-måling eller infusionstest, kan datafilen efterfølgende behandles og støtte lægen i diagnosticeringsprocessen.

Formålet med det udviklede system er at forbedre anvendeligheden i forhold til det i dag benyttede udstyr og objektivisere metoden til diagnosticeringen. Med dette forventes at kunne forbedre den diagnostiske kvalitet af 24-timers ICP-måling og infusionstest og dermed optimere sensitiviteten og specifiteten i diagnosticeringen af NPH-patienter.
Der er udviklet et system, som er mere brugervenligt end det nuværende. Det er let for sygeplejerskerne at påføre kommentarer ved 24-timers ICP-måling, da disse er standardiserede, og det er hurtigere for lægen at foretage en diagnose, fordi B-bølgeaktivitet markeres, og fordi den besværlige håndtering af en 3 meter lang udskrift er elimineret. Metoden til detektering af B-bølgeaktivitet bygger på arealberegning af segmenter i signalet og vurderes brugbar i klinisk praksis.
Infusionstesten visualiseres bedre for lægen efter filtrering af signalet. Dog skal lægen stadig selv fastlægge start- og slutplateauer, men systemet muliggør, at denne vurdering lettere foretages, hvorfor det vurderes, at infusionstesten objektiviseres.
Systemet er desuden, til forskel fra det nuværende system, lydløst, hvilket er en fordel for patienter og personale i et miljø, hvor støjniveauet i forvejen er højt.

Udviklingen af systemet bygger på UML, idet objektorienteret programmering har været i fokus i udviklingsprocessen, og anvendelsen af UML ses slutteligt som et stærkt værktøj.
Valget af programmeringssproget Java har betydet, at systemet kan være tungt at arbejde med, men det vurderes, at systemets brugervenlighed og funktionalitet er tilfredsstillende i forhold til brugerens ønsker.

Slutteligt må det fremhæves at udfordringen ikke ligger i at udvikle en algoritme, men at få fastlagt en parameterkarakteristik, der præcist beskriver, hvad der kan defineres som B-bølgeaktivitet og dokumentere, at denne parameterkarakteristik er af klinisk betydning, idet det endnu er udokumenteret, hvad der forårsager B-bølger.
Det ses realistisk, at systemet i nær fremtid med fordel kan benyttes i diagnosticeringsprocessen af NPH-patienter på AAS, og der er til formålet udviklet en dansk brugermanual.

Det kan konkluderes, at systemet objektiviserer metoderne, der danner baggrund for diagnosticeringen, og det vurderes, at systemet danner grundlag for en forbedret diagnostisk kvalitet.

KAPITEL 14

Perspektivering

Implementering og anvendelse af systemet på neurokirurgisk afdeling, AAS ligger indenfor de nære fremtidsperspektiver for systemet. Brugerne skal her gøre erfaringer med udstyret, hvorved der dannes grundlag for yderligere udvikling af systemet.
Men yderligere udvikling kræver, at der foretages en række ICP-målinger, så en endelig parameterværdi for detektering af B-bølgeaktivitet kan fastlægges. Dermed vil processen for udvikling af metoden til B-bølgedetekteringen fortsætte efter implementering af systemet, hvor målet er at sikre, at sensitivitet og specificitet forbedres og validitet og realibilitet højnes.
En større datamængde vil desuden danne grundlag for forskning i udseende og oprindelse af B-bølger, hvorved der kan tilegnes en større viden omkring deres betydning.

Den digitaliserede trykkurve giver mulighed for at præsentere ICP-signalet filtreret. Dette kan give anledning til en ny konsensus omkring detekteringen af B-bølgeaktivitet, idet B-bølger i et ikke filteret signal kan være skjult af støj som fx puls eller alm. bevægelser, hvilket gør det svært at finde alle B-bølger ved visuel detektering.

For at gøre det muligt for systemet at beregne R_{out} kan det være en fordel foretage at et sigmoid-kurvefit. Derved fastlægges plateauerne af beregningen, og det vil ikke være nødvendigt for brugeren selv at markere disse. Dette vil give en mere objektiv bestemmelse af R_{out}, da denne i stedet baseres på en standardiseret matematisk metode.

Der forskes desuden i muligheden for at detektere R_{out} uden opnåelse af slutplateau ved en såkaldt dynamisk metode. [5] [16]. Resultater fra den dynamiske metode bør følges og overvejes i en kommende udviklingsfase.

Hvis afdelingen gør gode erfaringer med udstyret, vil det være nærliggende, at andre afdelinger også kan benytte udstyret.
Systemet bør kunne benyttes på hver af de 4 neurokirurgiske afdelinger i Danmark. I praksis er det mest aktuelt for Aalborg og Odense Sygehus, da Camino MPM-1 på nuværende tidspunkt er implementeret på de respektive neurokirurgiske afdelinger. Århus Kommunehospital og Rigshospitalet anvender en anden type trykmåler, og det vil derfor være nødvendigt at udvikle systemet, så der kan kommunikeres med disse trykmålere. Alternativt kan anskaffelse af en Camino være en mulighed, hvis de anvendte trykmålere ikke giver mulighed for kommunikation med en pc.

Systemets fremtid på det kommercielle plan vil kræve en CE-mærkning af systemet.
CE-mærkning er en handelsmæssig nødvendighed, hvis man ønsker at markedsføre sit produkt inden for EU's grænser. Der stilles desuden lovpligtige krav til produktet, hvis en CE-mærkning

skal opnås.
Direktivet foreskriver, at medicinsk udstyr skal CE-mærkes som et synligt tegn på, at fabrikanten har fulgt procedurerne i direktivet, og at produkterne opfylder de gældende krav. Regler i forhold til CE-mærkning, der specielt er aktuelle for nærværende sytem, kan ses i appendiks G. Afsnittet er et uddrag af reglerne udstedt af Dansk Godkendelse af Medicinsk Udstyr (DGM) [36].

Vejen til CE-mærkning er besværlig, og en række forhold skal godkendes af det bemyndigede organ, som herhjemme er DGM [29].
Producenten skal fremlægge dokumentation, der beviser at produktet overholder de "Væsentlige Krav". Tolkning af disse krav kan findes i de harmoniserede standarder, der er en specifikation af direktiverne fra DGM [36]. Disse gør det nemmere for producenten, idet de harmoniserede standarder blot følges, hvorved dokumentationen bliver fyldestgørende.
Proceduren for godkendelse afhænger af klassifikationen af udstyret. Det udviklede system defineres til at tilhøre klasse IIa, hvilket medfører en bestemt procedure for godkendelse. Dette er yderligere beskrevet i appendiks G. Proceduren afhænger af, hvilken slags firma der drives, og om firmaet er ISO-godkendt. En korrekt ISO-godkendelse muliggør, at firmaet selv har lov til at CE-godkende sit produkt (med undtagelse af klasse III produkter). ISO-godkendelsen baseres på en række forhold omkring produktionen i virksomheden, hvilket ikke vil blive gennemgået.

En markedsføring af produktet vil yderligere kræve overvejelser omkring patent. Deslige vil en markedsføring betyde, at der ikke længere er tale om udstyr i forskningssammenhæng, og at systemet skal udvides i forhold til gældende regler for datasikkerhed. Det forventes, at der skal være en loginprocedure på systemet, at systemet skal på netværk og at data lagres centralt i en database.

En nærværende løsning er midlertidigt at få systemet ind som forskningsprojekt. Derved kan systemet umiddelbart indgå i klinisk praksis på AAS. Dette kræver, at en læge på afdelingen opstiller en forsøgsprotokol, som godkendes i Videnskabsetisk Komité. Tidsperspektivet for godkendelse er omkring 1 måned, og foretages af det lokale udvalg, hvilket dækker Viborg og Nordjyllands Amt [37].

Projektgruppen håber at neurokirurgisk afdeling, AAS vil få glæde af systemet, og at dette vil indgå som en naturlig del af den kliniske hverdag samt være til gavn for NPH-patienter.

Del V

APPENDIKS

APPENDIKS A

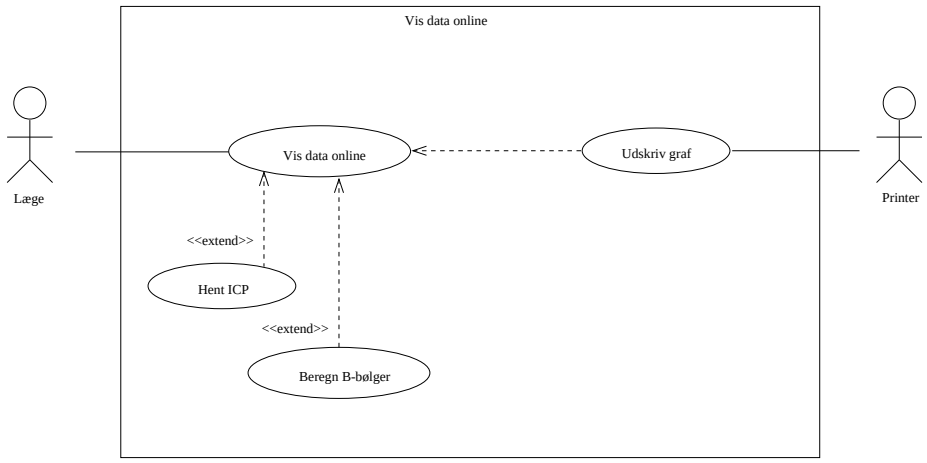
Unified Modelling Language

Der gives i dette appendiks en redegørelse for diagrammer og termer anvendt i brugen af Unified Modelling Language (UML). Beskrivelserne baseres på undervisning i emnet og bogen "UML Toolkit" [21].

A.1 Use-case diagram

Use-case diagrammet repræsenterer et overblik over hele systemet. Dette overblik er vigtigt, da dets indhold er centralt for hele softwareudviklingen ved, at den videre udvikling tager udgangspunkt i use-case diagrammet. Et use-case diagram indeholder en model af tre elementer: systemet, aktører, og use-cases.

Systemet er illustreret ved en boks, der indeholder use-cases til systemet, og hvor delsystemets navn er angivet øverst, dette ses illustreret i figur A.1 med delsystemet *Diagnosticering*.



Figur A.1: *Den firkantede boks illustrerer et system, med navnet på delsystemet øverst*

Aktørene kan enten være personer eller andre systemer (hardware), der interagerer med systemet. Aktørene sender eller modtager beskeder eller udveksler information med systemet. Disse repræsenterer en rolle til systemet, og sender beskeder til use-cases for at initialisere dem, som i figur A.1, hvor *Læge* og *Printer* er aktører.

For at identificere aktørerne til systemet bør følgende spørgsmål overvejes:

- Hvem skal bruge hovedfunktionaliteten af systemet (primær aktør)?

- Hvem har brug for systemets hjælp til deres daglige opgaver?
- Hvem skal sørge for at vedligeholde og administrere systemet?
- Hvilket hardware har systemet brug for at styre?
- Hvem og hvad har interesse for den værdi systemet producerer?

Aktørene der vælges til systemet kan bekræftes ved at have én eller flere associationer med use-cases. Aktøren behøver nødvendigvis ikke at initialisere en use-case, men vil på et eller andet tidspunkt kommunikere med én. Hvis det opfyldes får aktøren et navn til systemet der reflekterer rollen som feks. *Læge* og *Printer* se figur A.1.

Use-case repræsenterer en komplet funktionalitet i systemet, der altid bliver initialiseret af aktøren enten direkte eller indirekte. I UML defineres en use-case som:

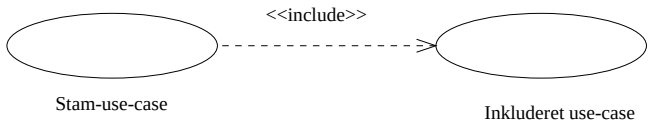
> En use-case specificerer en sekvens af handlinger, inklusiv varianter, som systemet kan udføre, og som giver et synligt resultat af værdi for den pågældende aktør.

Use-casens handlinger kan være at kommunikere med aktører eller foretage beregninger inde i systemet. For at finde use-cases, betragtes de valgte aktører, hvor følgende spørgmål stilles:

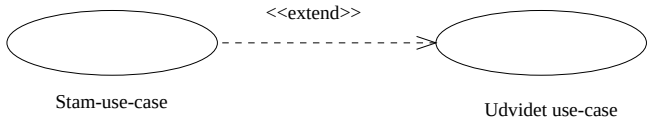
- Hvilke funktioner har aktøren brug for fra systemet? Hvad har aktøren brug for at gøre?
- Har aktøren brug for at læse, oprette, slette, modificere, eller gemme noget information i systemet?
- Har aktøren behov for at blive gjort opmærksom på begivenheder i systemet, eller har aktøren brug for at gøre systemet opmærksom på et eller andet? Hvad repræsenterer de begivenheder i funktionaliteter?
- Hvilke input/output har systemet brug for? Hvor modtages eller sendes input/output fra?
- Hvad er problemerne med det nuværende system? (feks. manuelt system istedet for automatiseret)

I eksemplet fra figur A.1 ses det at lægen har brug for en use-case, hvor det er muligt at diagnosticere. Printeren får input fra systemet ved en use-case "Udskriv graf". Indirekte initialiserer aktør lægen andre use-cases fra use-case "Vis data offline", såsom "Hent ICP", "Beregn B-bølger", "Rediger B-bølger", "Gem behandlet data", og "Udskriv graf".
Når use-cases bliver initialiseret af andre use-cases beskrives relationen imellem dem for at give et indblik for deres opførsel i systemet. Relationen mellem use-cases kan beskrives på to forskellige måder *extend* eller *include*.
«include»
I en include-relation står den inddragne use-case ikke alene, den skal forbindes med en eller flere stam-use-cases, se figur A.2. Det medfører at i tilfælde, hvor der er en include-reltion, er udførelsen af stam-use-casen afhængig af om den inkluderet use-case er aktiveret.
«extend»

Extends bruges når en use-case er en variation af en anden. En use-case repræsenterer selv forskellige varianter af et forløb, men i forbindelse med en extend er der tale om en udvidelse af en use-case, se figur A.3. Stam-use-casen er ikke afhængig af den udvidet use-case, som det er tilfældet ved include.



Figur A.2: *Viser hvordan include-notation i UML, ser ud.*



Figur A.3: *Viser hvordan extend-notation i UML, ser ud.*

A.2 Klassediagram

Et klassediagram viser klassernes statiske struktur og deres relationer i et system. Klasser repræsenterer "ting", der bliver behandlet i systemet.
Formålet med et klassediagram er også at danne et grundlag for andre diagrammer i UML. For at kunne lave et klassediagram skal systemets klasser identificeres.

Udover at klasserne viser ting, der bliver behandlet i systemet, så viser klasserne også stukturen af information samt adfærd i systemet.
Klasserne findes i problemdomænet og defineres ved, at de er nanvneord, der enten bliver gemt eller analyseret i systemet. Klasserne kan typisk findes og defineres ud fra use-case beskrivelsen, som findes i afsnit 4.6.1.
En klasse kan defineres ved: Navn, Attributter og Metoder se figur A.4.

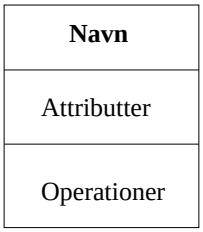
- Navnet skal helst sige noget om hvad det er klassen repræsenterer fra problemdomænet.
- Attributterne er de instanser eller variable, der indgår i klassen, dvs. klassens karakteristika.
- Operationer er de metoder, dvs. funktioner som klassen udfører.

En klasses attributter kan både være *Public* eller *Private*. Dette vises i en klasse som hhv. et plus(+) eller et minus(-). Når en attribut er public betyder det, fx at den kan nedarves til specialiceringer af klassen. Private betyder, at attributten ikke kan tilgåes af andre klasser.

Som tidligere nævnt viser et klasse diagram blandt andet relationerne mellem klasserne. Klassernes relationer kan opdeles i de følgende grupper.

Association

En association er et link mellem klasser, der gør, at klasserne "kender til hinanden". Der findes



Figur A.4: *Viser opbygningen af en klasse.*

flere former for associering, fx or-association eller ordnet association. En associering vises i et diagram som en linie mellem to klasser, se figur A.5.

Aggregering

En aggregering er en speciel form for association, der henviser til at en klasse indeholder en anden klasse. En aggregering vises som på figur A.5. Den ruderformede ende slutter op mod den indeholdende klasse.

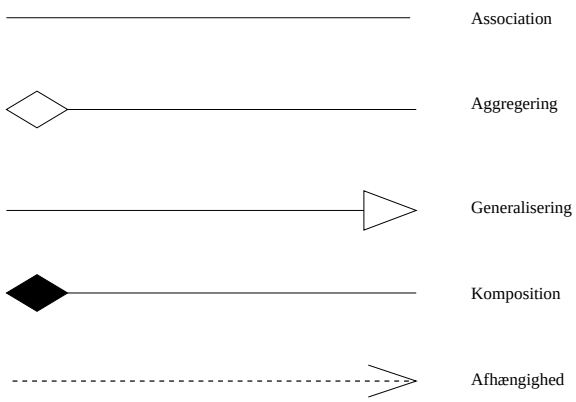
En anden type aggregering er en komposition, der angiver, at den indeholdte klasse ikke kan eksistere uden, at den indeholdende klasse eksisterer. En composition vises ved at ruderformen i er udfyldt, hvilket også ses figur A.5.

Generalisering

En generalisering mellem to klasser viser at de to klasser er bundet sammen via nedarvning fra den ene klasse til den anden. Klasserne benævnes som den generelle og den specielle klasse. En generalisering vises som en pil, der peger mod den generelle klasse, se figur A.5.

Afhængighed

Èn klasse afhænger af en anden, hvis den anvender et objekt at den anden klasse som parameter, tilgår et globalt objekt eller kalder en metode fra den anden klasse. Afhængighed illustreres med en stiplet pil, hvilket ses i figur A.5. Pilen vender mod klassen den afhængige klasse.

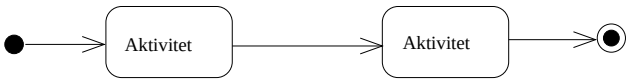


Figur A.5: *Figuren viser 5 UML-notationer der bruges når UML-diagrammer tegnes.*

A.3 Aktivitetsdiagram

Aktivitetsdiagrammets formål er at vise et sekventielt flow af aktiviteter, dvs. handlinger samt resultatet af disse handlinger. Fokus med et aktivitetsdiagram ligger på arbejdet.
Ud fra handlingerne kan der findes operationer eller metoder, der kan bruges i udvikling af kode. De handlinger eller aktiviteter der findes i aktivitetsdiagrammet kan indsættes i "swimlanes". Swimlanes bruges til at gruppere aktiviteterne, og kan med fordel anvendes til fx at vise hvor i et system aktiviteterne foregår. Aktivitetsdiagrammer deles ofte op i tre swimlanes: Boundary, Control og Entity. Boundary er systemets grænseflade, som kan være i form af brugerflade (GUI), styring og kommunikation med input og output enheder. Entity er den del af systemet, der arbejder med hukommelsen dvs. at gemme, hente variable og filer osv. Control er den del af systemet, der udfører og kontrollerer funktionaliteter fx algoritme beregning.

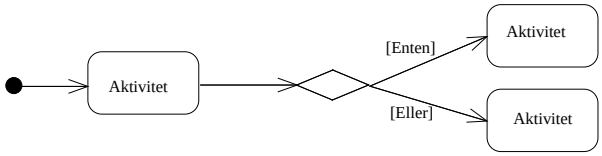
En aktivitet er en handling, der foretages i et program, og hvoraf der ofte kan udledes en metode. Aktiviteten er vist i diagrammet som en firkant med afrundede hjørner. Overgangen mellem to aktiviteter kaldes transitioner, der vises ved en pil som på figur A.6.



Figur A.6: *Viser aktivitet og transition*

Som det også ses af figur A.6 startes og stoppes et aktivitetsdiagram hhv. af en sort cirkel og et såkaldt "bulls-eye".

Transitionerne i diagrammet kan være påhæftet en "Guard-condition", dvs. at en eller anden hændelse skal være sand for at komme til næste aktivitet. Dette giver også mulighed for at benytte et "Decision-point", dvs. hvor der vælges en rute ud af flere mulige. Et "Decision-point" er i UML beskrevet ved en ruder-form som på figur A.7.



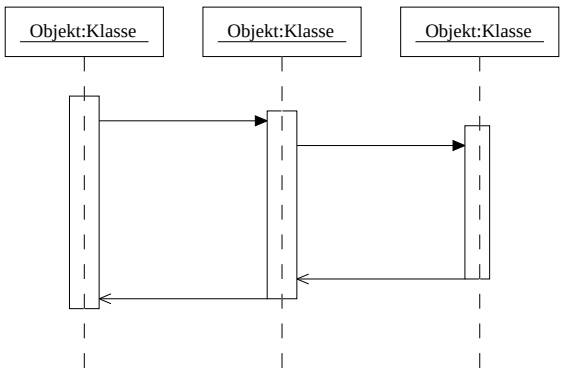
Figur A.7: *Viser et "Decision-point" og dets "Guard-conditions".*

Hvis der i et diagram er sideløbende processer, kan disse vises ved en "bjælke", hvor transitionen deler sig op i to eller flere transitioner. Denne bjælke bruges også, når flere transitioner samles i en enkelt transition, som det kan ses på figur 5.2.

A.4 Sekvensdiagram

Et sekvensdiagram illustrerer, hvordan instanser af klasser (objekter) interagerer med hinanden, hvor det primære fokus er tiden. Der fokuseres på, hvorledes meddelelser transmitteres og modta-

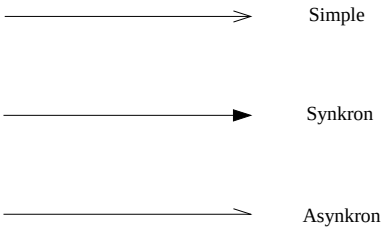
ges objekterne imellem evt. symboliseret ved metodekald. Diagrammet benyttes til at visualisere detaljer om komplicerede og dynamiske situationer, hvilket ses i et eksempel figur A.8. Sekvensdiagrammet supplerer derfor et klassediagram, der beskriver en statiske situation.
Sekvensdiagrammet opbygges ved to akser: en lodret akse, der viser tiden, og en vandret akse,



Figur A.8: *Notation i et sekvensdiagram der viser objekterne og deres tidslinie.*

der viser de objekter, der ønskes beskrevet.

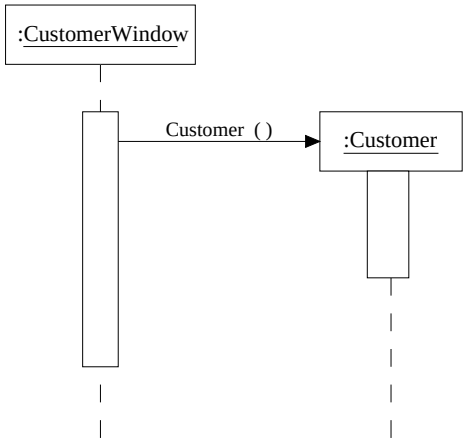
Objekterne på den vandrettte akse repræsenteres ved et rektangel, hvori objektnavnet og dets klasse angives, hvor objektet understreges. Den lodrette stiplede linie fra objektet repræsenterer livslinien for objektet.
Objektets aktiviteter illustreres som rektangler på livslinien i figur A.8. Meddelelser mellem aktiviteter kan illustreres ved tre forskellige pile, hvilke ses i figur A.9. Simple meddelelser



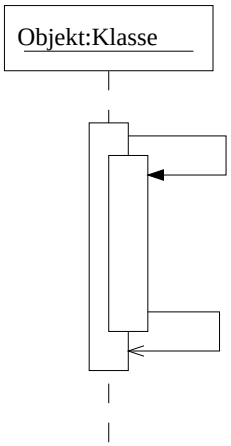
Figur A.9: *Viser notation for meddelelser mellem objekter, der kan være simple, asynkrone eller synkrone.*

repræsenterer kommunikation mellem objekter, hvor der ikke ønskes udspecificeret detaljer omkring kommunikationen. Synkrone meddelelser repræsenterer eksempelvis tryk på en knap, hvor det ikke vides, til hvilket tidspunkt hændelsen indfinder. En asynkron meddelelse symboliserer en hændelse, som sker på bestemte tidspunkter, eksempelvis som et ur der slår. Den asynkrone meddelelse afhænger ikke af at skulle håndteres af et objekt, og anvendes ofte i realtime systemer for eksempelvis seriel kommunikation.

En meddelelse eller metodekald kan foruden at aktivere en ny aktivitet også skabe et nyt objekt, hvilket illustreres i figur A.10, og et sekvensdiagram kan illustrere, at et objekt aktiverer en funktionalitet tilhørende objektet selv som angivet i figur A.11. Et objekt kan desuden slås ihjel, så det ikke længere er muligt at tilgå. Dette illustreres ved på objektets livslinie at angive et X.



Figur A.10: *Et metodekald skaber et nyt objekt.*



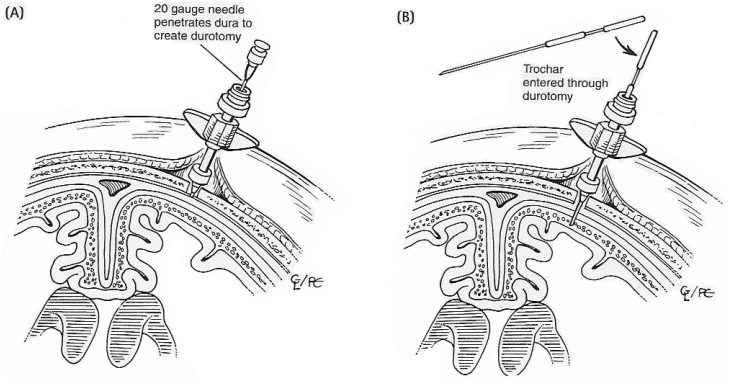
Figur A.11: *Et objekt aktiverer en funktionalitet tilhørende objektet selv.*

APPENDIKS B

Beskrivelse af Camino MPM-1

Afsnittet indeholder en beskrivelse af Camino MPM-1 ud fra de oplysninger, der har været mulige at få fat på.

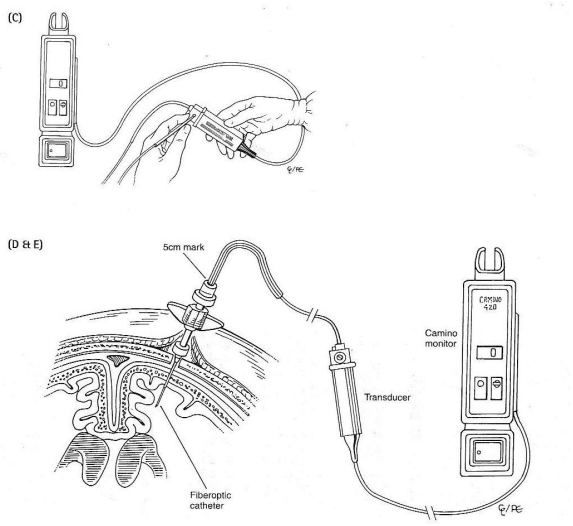
Det anvendte udstyr på AAS til intrakranial trykmåling er et fiberoptisk system til foretagelse af en intraparenchymatøs trykmåling. Udstyret består af en tryktransducer, som er tilkoblet Camino MPM-1. Måleprincippet bygger på en reflektion af lys fra en trykfølsom membran i spidsen af et optisk kabel. Det reflekterede lys transporteres via det optiske kabel til et måleapparat, hvor trykket beregnes og vises på en monitor.
Denne procedure har være kommerciel siden 1985, udviklet af Camino Laboratories, og er nu blandt de mest populære metoder, fordi den er meget præcis og er årsag til meget få komplikationer [26] [CD, A.1.3] .
Ved parankymatøs måling placeres kateteret i hjernevævet i frontallappen. Der under 5% risiko for komplikationer i forbindelse med infektioner og post-operative blødninger [26]. På figur B.1 og B.2 ses, hvorledes et kateter lægges ved parenkymatøs måling.



Figur B.1: *Trin A-B, hvor det ses hvor trykmåleren placeres [8]*

Patienten kobles til en Camino MPM-1 med display, der viser trykket i mmHg. Camino-trykmåleren er mest udbredt, fordi den er mere pålidelig end fx konkurrenten Codman MicroSensor [31]. Da sygehuset har gode erfaringer med og har benyttet Camino-trykmåleren gennem en årrække, er denne valgt til måling af det intrakranielle tryk på AAS [2]. På figur B.3 ses en Camino MPM-1 trykmåler.

Præcisionen af trykket er $\pm$ 1 mmHg, og transduceren er groft set liniær fra en skale på -20 til 300 mmHg, dog understøtter denne kun et måleområde fra -10 til 250 mmHg, se yderligere manualen for Camino MPM-1, der findes på vedlagte CD-ROM [CD, A.1.3] .



Figur B.2: *Trin C-E ved en parenkymatøs trykmåling, hvor trin C er specielt for en Camino-trykmåler, idet den kaliberes efter atmosfærisk tryk [8].*

Figur B.3: *En Camino-trykmåler [30].*

På displayet vises der en trendkurve samt det aktuelle ICP. Derudover har den en RS-232 port, således det er muligt at opsamle data digitalt. Desuden er der en analog udgang til en printer. Detaljeret information om måleprincipper har ikke været tilgængelige, men en manual kan findes på vedlagte CD-ROM [CD, A.1.3] .

APPENDIKS C

Metoder til detektering af B-bølgeaktivitet

Kapitlet præsenterer metoder, som kan benyttes ved B-bølgedetektering. Afsnittet vil være opdelt efter metoder, der agerer i frekvensdomænet og i tidsdomænet. Fordele og ulemper diskuteres, og der foretages et valg angående, hvilken metode systemet skal benytte.

C.1 Frekvensdomæne

Fourier-analyse

En oplagt mulighed for detektering af frekvenser i signalet er en Fourier-analyse, hvor signalet repræsenteres i frekvensdomænet. Dette findes også beskrevet i litteraturen for ICP-bølger [12] [13].

Fourier-transformationen har visse ulemper og begrænsninger i forhold til formålet B-bølgedetektering, idet en sådan transformation kun giver en opløsning i frekvens og ikke i tid. Dermed kan alle frekvenser i signalet bestemmes, men det er ikke muligt at relatere disse til et tidspunkt, hvilket er essentielt for systemet, hvor tidspunkterne skal visualiseres for brugeren [35].

Fourier-transformationen beregnes ud fra et fastlagt tidsvindue, der har en konstant tidsopløsning. Produktet af tidsopløsningen og frekvensopløsningen ($\triangle T_r \cdot \triangle F_r$) er konstant, hvilket betyder, at forbedring af tidsdomænet kun kan ske på bekostning af opløsningen i frekvensdomænet. Diskret Fourier-transformation antager desuden, at signalet er periodisk, hvilket ikke er hensigtsmæssigt i håndteringen af et stokastisk signal [6].

Waveletanalyse

En waveletanalyse giver en løsning på problemerne ved Fourier-transformationen.

Waveletanalysen bygger på en komposition af basale funktioner kaldet wavelets, ved hvilke signalet kan fragmenteres og beskrives [18]. Signalet analyseres udfra en "mother-wavelet", der er afgrænset i tid, og kan tilpasses signalets frekvenser. Det bliver muligt at analysere højfrekvente komponenter i et vindue, der har et lille tidsinterval, da perioden er kort for høje frekenser, og lavfrekvente komponenter i et vindue med et stort tidsinterval. Således tilpasses behovet for de enkelte frekvenser, og medfører den bedste opløsning [6].

Resultatet af waveletanalysen er en transformation af signalet, hvor det er muligt at bestemme, til hvilken tid frekvenserne optræder.

Metoderne vil kunne skelne mellem de langsomme og hurtige frekvenser i B-bølgespektret, men dette ønskes imidlertid ikke.

C.2 Tidsdomæne

Mønstergenkendelse

Et signal kan analyseres digitalt, på samme måde som lægerne i dag foretager diagnosticeringen, ved at løbe signalet igennem og identificere et mønster. Her betragtes signalet som et deterministisk signal [18].

Mønstergenkendelse finder sted, når man på grundlag af en måling af et objekts karakteristika prøver at finde ud af, hvilket objekt der er tale om [18]. I praksis identificeres et mønster, som er karakteristisk for B-bølgeaktivitet. Dette signal, indeholdende mønsteret, lader man glide hen over signalet og trækker trykværdierne fra hinanden, til der er et match, der kan accpteres.

Mønstergenkendelse kan også benyttes til objektivt at fastlægge, om der er mønstre i signalet, der går igen. Dette er muligt ved at sammenligne perioder af det filtrerede signal med signalet selv. Et resultat ville kunne indikere, om der er gentagne mønstre, der kan relateres til en diagnose. Metoden kan benyttes til sammenligning med den kliniske erfaringsbaserede viden, som i tilfældet B-bølgedetektering er begrænset til de mønstre, der visuelt kan skelnes.

Amplitudedetektering

Der er forskellige muligheder for at implementere en amplitudeafhængig algoritme.

Én metode er direkte at detektere den enkelte bølges amplitude, og der skal derved i udregningen holdes styr på hver enkelt bølge. Da hver bølge ikke ønskes detekteret (jf. afsnit 6.1.2) ville det være en fordel at udregne RMS-værdien af amplituderne over en mængde bølger, som det er beskrevet i artiklen *"Two computerized methods..."* [7]. En anden metode kunne være at udregne arealet under bølgerne for en defineret periode, hvor arealet forventes proportional med en række bølger med samme amplitude, dvs. hvis enkelte bølger har lav amplitude vil arealet være mindre. I ovennævne artikel beskrives to amplitudeafhængige metoder.

Data er samplet med 250 Hz, og der er beregnet et gennemsnit over 1 sek. Data filtreres med et digitalt butterworth 5. ordens båndpasfilter med knækfrekvenser svarende til B-bølgefrekvensspektret. Dermed fjernes offset, så bølgerne ligger omkring 0-linien.

I første metode analyseres hver enkelt bølge og klassificeres som B-bølge, hvis amplituden minimum har værdien af en fastlagt amplitudegrænse. En bølge klassifiseres som en ny bølge, når 0-linien overskrides.

I anden metode deles det filtrerede signal ind i 10-minutters blokke med overlap på 1 minut. For hver blok udregnes RMS-værdien af amplituderne. Alle blokke med en RMS-værdi over en fastlagt grænse defineres som B-bølgeaktivitet.

Resultaterne sammenlignes med en visual bedømmelse foretaget af to erfarne neurologer, der ved uenighed har konfereret til enighed. Resultaterne fra artiklen for disse to metoder ses i tabel C.1. Der findes god korrelation mellem metoderne for tærskelværdier op til 3 mmHg. Anden metode finder til forskel fra første metode god overensstemmelse med den visuelle bedømmelse ved tærskelværdier på 3 mmHg og 3,25 mmHg.

Begge metoder vil kunne benyttes til detektering, men for at få sammenhængende perioder med aktivitet, og derved opfylde kravene sat til parametkarakteristikken, foretrækkes artiklens metode2.

Tærskel	$B\%_1/B\%_{visuel}$	$B\%_2/B\%_{visuel}$	$B\%_1/B\%_2$
0.25	0.69	0.31	0.78*
0.5	0.82*	0.81*	0.97**
0.75	0.79*	0.75*	0.99**
1.0	0.74*	0.57	0.96**
1.25	0.69	0.47	0.95**
1.5	0.6	0.36	0.92
1.75	0.58	0.33	0.91*
2.0	0.52	0.36	0.91*
2.25	0.46	0.37	0.93*
2.50	0.39	0.40	0.93*
2.75	0.37	0.56	0.92*
3.0	0.35	0.74*	0.81*
3.25	0.33	0.72*	0.19
3.5	0.32	0.65	0.05
3.75	0.30	0.65	0.06
4.0	0.32	0.65	0.02

Tabel C.1: Sammenhæng mellem estimeret B% som funktion af minimum amplitude tærskelværdier. *statistisk signifikans p<0,05. **statistisk signifikans p<0,01

[7]

APPENDIKS D

Klasser og metoder

Forklaring til de enkelte klasser og deres metoder, klasserne står i alfabetisk rækkefølge

ActionJList	Lytter og håndterer valg af de prædefinerede kommentarer.

Bboelger	Viser den grafiske brugerflade for B-bølgedetektering og håndterer tast på knapper.
+visBBrugerflade():void	Opretter brugerfladen, dvs. den grafiske ramme, panel, graf og knapperne.
+lytBKnapper():void	Sørger for der kan udføres en række sekvenser ved tryk på en knap.

Filhaandtering	Et filter, der bruges til at begrænse søgningen på filer.
+addExtension()	Tilføjer en extension, som gør, at GemFil kan finde filer med den extension.
+setDescription()	Tilføjer en beskrivelse til en extension.

GemFil	Giver mulighed for at gemme filer.
+hentFilNavn():File	Opretter filnavnet ved at hente datoen.

ICP	"24-timers ICP-måling"
+visIcpBrugerflade():void	Opretter brugerfladen, dvs. den grafiske ramme, panel, graf og knapperne
+lytIcpKnapper():void	Sørger for der kan udføres en række sekvenser ved tryk på en knap

IcpData	Starter dataopsamling og konverterer data bytes til ICP-værdi (mmHg).
+serielEvent():void	Opsamler data fra porten og frasorterer uønsket data.
+portIdentifikation():void	Identificerer porten, der kommunikere med Caminoen.
+lukPort():void	Sørger for at lukke porten, når denne ikke bruges mere.
+konverterIcp():double	Konverterer databytes til ICP.

Infusionstest	Henter brugergrænsefladen for infusionstesten og håndterer tast på knapper.
+visInfBrugerflade():void	Opretter brugerfladen, dvs. den grafiske ramme, panel, graf og knapperne.
+lytInfKnapper():void	Sørger for der kan udføres en række sekvenser ved tryk på en knap.

Kommentar	Håndterer indskrivning af kommentarer og tilføjer dem til online graf.
+tilfoerKommentar():void	Tilfører kommentar, indtastet af brugeren til online graf.
+tilfoerDefKommentar():void	Tilfører prædefinerede kommentarer til graf.
+tilfoerGrafKommentarvector():void	Tilfører de kommentarer, der er gemt fra online graf til offline graf.

Main	Viser brugergrænseflade for startskærm og håndterer tast på knapper (indeholder Main-metoden).
+visStartBrugerflade():void	Opretter startbrugerfladen, dvs. den grafiske ramme, panel og knapperne.
+lytStartKnapper():void	Sørger for der kan udføres en række sekvenser ved tryk på en knap.

OfflineGraf	Opretter offline grafen, håndterer beregningen af B-bølger og filtrer signalet fra infusionstesten.
+hentIcpDatafil():void	Henter ICP data fil gemt fra 24 timers-ICPmåling.
+initParametre(int[]):void	Sætter parametrene, der skal bruges til at beregne B-bølger.
+beregnBoelger():void	Beregner hvilke dele af signalet, der er B-bølge aktivitet.
+filterInfusion(double):double	Filtrerer ICP signalet foretaget i infusionstesten.
+markerBboelger():void	Markerer den del af signalet metoden beregnBoelger() har beregnet er B-bølgeaktivitet.
+procentBboelger():int	Beregner den procentuelle del af signalet, som er B-bølgeaktivitet på baggrund af resultatet fra beregnBoelger().

OnlineGraf	Opretter graf til plotning af data ved 24 timers ICPmåling.
+icpPaaGraf():chartPanel	Opretter grafen og opsætter akserne for grafen til 24-timers ICPmåling.
+icpPaaGrafInf():chartPanel	Opretter grafen og psætter akserne for grafen til infusionstest.

Parametre	Håndterer parametrene for beregning af B-bølger.
+passwordGUI():void	Viser brugerflade, således der først kan indtastes et password.
+tjekPassword():boolean	Tjek af det indtastede password.
+parametreGUI():void	Brugerflade så ændring af parametrene kan foretages.

Patient	Henter datafiler og sørger for at data kan plottes på OfflineGraf.
+filSoegning():void	Giver mulighed for at søge efter en given måling, hvorefter der sørges for at hente data, kommentar og tidsfiler.
+filTilarray():void	Henter gemt ICP vektor og lægger værdierne i et array.
+kommentarTilFil():void	Læser kommentarene ind i en vektor, som bliver læst ind i en fil.
+laesKommentarFil():vector	Henter kommentar vektor fra fil og læser kommentarerne fra vektoren og tilføjer kommentarerne til grafen.

PraeDefKommentar	Oprette de prædefinerede kommentarer og visualiserer dem på brugergrænsefladen.

ScrollBar	Opretter scrollbarfunktion samt håndterer brugen af scrollbaren.

SerielKommuTraad	Tråd, der sørger for, at der kan plottes på grafen.
+run():void	Sætter data på grafen således det vises online.

APPENDIKS E

Seriel kommunikation

Som tidligere nævnt kommunikeres serielt med Camino MPM-1 gennem et RS-232 kabel. Protokollen for Caminoen vil i det følgende blive beskrevet.

For yderligere oplysninger om den serielle kommunikation se protokollen, der findes på vedlagte CD-ROM [CD, A.1.4] . Caminoen benytter en DB-9 seriel port til kommunikationen gennem et standard RS-232 kabel. Data sendes med en hastighed på 19200 baud med 7 data bit, ét stop bit og ulige paritet. Data bliver sendt i alfanummerisk ASCII-format periodisk separeret af et linieskift (<**LF**> = 10).

Data sendes i pakker af 5 bytes startende med et alfabetisk header ID fulgt af nummeriske værdier: **a n1 n2 n3 n4** .

Header ID

Caminoen kan sende en række forskellige oplysninger, men for dette system er det kun en enkelt Header ID - **R** - der benyttes. Se tabel E.1.

Header ID	ASCII(decimal)	Forklaring
R	82	Filtreret (gennemsnit over 4 værdier) ICP, ca 190 Hz
I	73	Mean ICP, 1 Hz
S	83	Systolisk ICP, 1 Hz
D	68	Diastolisk ICP, 1 Hz
A	65	MAP, 1 Hz
C	67	Mean CPP, 1 Hz
T	84	Mean ICT, 1 Hz
O	79	Trend ICP(1 værdi), sendes hvert 2.7 minut
P	80	Trend CPP(1 værdi), sendes hvert 2.7 minut
M	77	Trend ICP buffer, buffer sendes når 't' modtages serielt
N	78	Trend CPP buffer, buffer sendes når 't' modtages serielt
E	69	Fejlkode
X	88	Ingen alarmer, 1 Hz
Y	89	Hørbar alarm, 1 Hz
Z	90	Stille alarm, 1 Hz
B	66	Indeks af skalering

Tabel E.1: Tabel over de Header ID, der kan blive sendt fra Caminoen

Nummeriske værdier

De nummeriske værdier for **R** (ICP) sendes i enheden $\frac{1}{10}$mmHg og negative værdier er indikeret med et (-). Dvs. at koderne læses på følgende måde:

R0122 = ICP 12,2 mmHg eller

R-053 = ICP -5,3 mmHg

Det betyder, at ICP værdier kan sendes i området -99,9 til 999,9 mmHg.

<LF>

Der indsættes et <LF> (line-feed character) periodisk ved opsamling af ICP. Dette sker ved hver 15. ICP datapakke (header ID: R). Desuden sendes der et <LF> før og efter en informationsdatapakke, som består af 6 parametre (header ID: I, S, D, A, C og T).

APPENDIKS F

Test

Dette afsnit har til formål at give en teoretisk gennemgang af whitebox- og blackboxtest, der er de to mest benyttede testmetoder.

Formålet med at teste sin kode er at finde fejl, og en succesfuld test er derfor en test, der finder uopdagede fejl, mens en test, der ikke finder fejl, blot sætter spørgsmålstegn ved om testen var udført grundigt nok [23].
Whitebox og blackbox er to forskellige testmetoder, på den måde at whiteboxtest håndterer den logiske struktur i koden og blackboxtest håndterer programmets funktionalitet [23].

F.1 Whitebox

Logiske struktur indebærer, at whiteboxmetoden går meget procedural tilværks, dvs. gennemgår koden "slavisk", hvor det især er løkker og betingelser, der er i fokus.

Hvis et program bliver testet vha. whiteboxmetoden vil man teoretisk kunne forvente, at koden er 100% fejlfri, men dette kan ikke lade så gøre i praksis, da whiteboxtest hurtigt bliver meget omfattende. Dette kan vises i et eksempel:

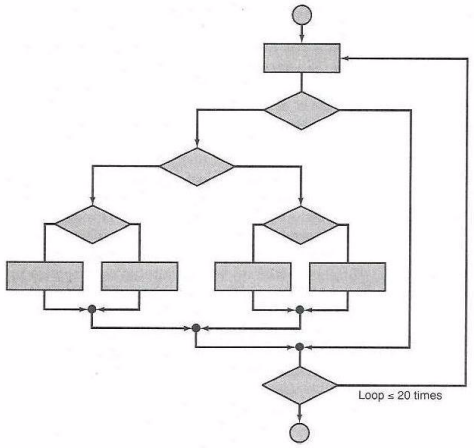
> ...a 100 line PASCAL program with a single loop that may be executed no more than 20 times. There are approxmately 10^{14} possible paths that may be executed!
> ...(if a) processor can develop a test case, execute it, and evaluate the results in 1 millisecond. Working 24 hours a days, 365 days a year, the processor would work for 3170 years to test the program...(se figur F.1) [23].

Da whitebox er så omfattende bliver man nødt til at indkredse testen til nogle færrere og mindre problemområder.

Whitebox test bruges til at lave test-cases, der:

1. Garanterer at alle uafhængige stier i et modul bliver gennemkørt mindst én gang.
2. Udfører alle logiske valg, som både sand og falsk.
3. Gennemkører alle løkker ved deres grænser og indenfor deres operationsområder.
4. Kan afprøve data strukturernes validitet.

Men hvorfor bruge meget tid på at teste logik og struktur, når programmets funktionaliteter kan testes vha. en blackboxtest?



Figur F.1: *Flowchart diagram over det i eksemplet brugte program [23]*

Dette skyldes at blackboxtest højst sandsynlig *ikke* finder de fejl, som fx tastefejl, fejl som ligger i programmets "hjørner" samt i de løkker og logiske valg, der gennemkøres [23].

Whiteboxtest udføres oftest tidligt i implementations- og testfase.

F.2 Blackbox

Blackboxtest fokuserer som tidligere nævnt på programmets funktionelle krav, dvs. sætter en softwaretester i stand til give programmet input, gennemkøre alle programmets funktionaliteter og sammenligne output med forventede output uden at tænke på programmets kode.
Blackboxtest er dog ikke et alternativ til whiteboxtest, men benyttes til at afsløre en anden slags fejl [23]:

1. Forkerte eller manglende funktioner.
2. Grænsefladefejl.
3. Fejl i datastrukturer eller ekstern databasetilgang.
4. Fejl i ydeevne.
5. Initialiserings- eller afslutningsfejl.

Blackboxtest udføres sent i udviklings- og testfase, og der vil her blive omtalt to testmetoder: *Equivalence Partitioning* og *Boundary Value Analysis* (BVA).

F.2.1 Equivalence Partitioning

Ved Equivalence Partitioning er formålet at begrænse antallet af totale testcases. En testcase for Equivalence Partitioning baseres på evaluering af input, dvs. hvorvidt et input er valid eller ikke valid. Input defineres her som en specifik nummerisk værdi, et område af værdier, et "sæt" af værdier eller en boolean [23].

F.2.2 Boundary Value Analysis

Boundary Value Analysis (BVA) (analyse af grænseværdier) benyttes, fordi der af uvisse grunde er større tendens til, at der opstår fejl i grænserne af et inputdomæne [23]. BVA opererer med samme input som Equivalence Partitioning og retningslinierne ser ud på følgende vis:

1. Hvis input er et område af værdier fra a til b skal testcasen udføres, så der testes på værdierne a og b samt værdierne lige over og under hhv. a og b.
2. Hvis input er specifikke værdier testes minimum og maximum samt værdier under og over hhv. minimum og maksimum.
3. For output værdier sættes samme krav, som i 1 og 2.
4. For interne datastukturer med forudsatte grænser, kan der fx. testes på disse grænser.

APPENDIKS G

CE-mærkning

Følgende er uddrag fra reglementet vedrørende CE-mærkning af medicinsk udstyr, udgivet af Dansk Godkendelse af Medicinsk Udstyr (DGM) [36]. Centrale punkter er udvalgt for at give et overblik over de vigtigste overvejelser idet et produkt ønskes CE-mærket.

Hvad er medicinsk udstyr?

Medicinsk udstyr defineres som alt det udstyr, som fra fabrikantens side er beregnet til at blive anvendt ved behandling af mennesker, uanset om det drejer sig om diagnosticering, overvågning, forebyggelse, behandling eller lindring.
Også svangerskabsforebyggende midler og hjælpemidler til handicappede falder ind under begrebet medicinsk udstyr.
Lægemidler, kosmetiske produkter, personlige værnemidler m.v. vil fortsat være reguleret efter deres egne regelsæt [36].

CE-mærket

Direktivet foreskriver, at medicinsk udstyr skal CE-mærkes som et synligt tegn på, at fabrikanten har fulgt procedurerne i direktivet, og at produkterne opfylder de krav, der gælder for dem. Dette gælder dog ikke udstyr, der er specialfremstillet til en bestemt patient eller beregnet til klinisk afprøvning eller præsentation på messer m.v [36].

Krav til produkterne

I bekendtgørelsen er hovedkravene til det medicinske udstyr beskrevet i overordnede vendinger. De "Væsentlige Krav" til produkterne kan dog inddeles i 6 generelle og en lang række mere specifikke krav [36]:

De generelle krav:

- Højt sikkerhedsniveau
- Indbygget beskyttelse
- Egnethed
- Stabilitet i ydeevne
- Transportsikkerhed
- Afvejning af fordele mod ulemper

De specifikke krav:

- Materialesikkerhed
- Bioforlignelighed
- Sterilitetssikkerhed
- Mekanisk sikkerhed
- Elektromagnetisk kompatibilitet
- Brand- og eksplosionssikkerhed
- Nøjagtighed og stabilitet
- Strålebeskyttelse
- Alarmfunktioner
- Elektrisk sikkerhed
- Mærkningskrav
- Krav til brugsanvisning

Til hjælp for den enkelte fabrikant er der udviklet særlige harmoniserede standarder, der tolker, hvad de acceptable niveauer er til opfyldelse af de "Væsentlige Krav".

Europæiske harmoniserede standarder
Der findes i dag næsten 100 europæiske, harmoniserede standarder til medicinsk udstyr, som afspejler direktivernes krav på en mere konkret og detaljeret måde.
Disse standarder er udarbejdet i de europæiske standardiseringsorganisationer CEN og CENELEC efter mandat fra EU-Kommissionen. Følger fabrikanten disse standarder, anses de krav i bekendtgørelsen, som standarderne omhandler, automatisk for opfyldt.
Eksempler på sådanne harmoniserede standarder for produkterne er EN 60601-1 om elektrisk sikkerhed, EN 60601-1-2 om elektromagnetisk forlignelighed (EMC) samt serien EN 550/54/56 om sterilitet.
Det er dog frivilligt for fabrikanterne, om de vil følge standarderne, eller om de på anden måde kan dokumentere, at kravene er opfyldt [36].

Klassifikation
Da den generelle bekendtgørelse om medicinsk udstyr dækker et meget bredt spekter af produkter, har man ikke fundet det hensigtsmæssigt, at alt medicinsk udstyr skal gennemgå den samme godkendelsesprocedure.
Bekendtgørelsen opdeler derfor det medicinske udstyr i 4 forskellige klasser (I, IIa, IIb og III). Endvidere er klasse I underopdelt i udstyr, der sælges sterilt og udstyr, der sælges med en målefunktion.
Klassifikationen afspejler den risiko, der er forbundet med at anvende produktet, sårbarheden af de legemsdele, udstyret skal anvendes på, og hvor lang tid dette foregår over.

Den højeste risikoklasse (III) omfatter således blandt andet produkter, der kommer i kontakt med centralnervesystemet, hjertet eller det centrale kredsløb samt medicinsk udstyr, der indeholder lægemidler.
Opdelingen sikrer således, at kontrollen står i et rimeligt forhold til den risiko, der kan være forbundet med brugen af udstyret [36].

Klassificering af Systemet

I henhold til Indenrigs- og sundhedsministeriets bekendtgørelse nr. 105 af 27. februar 2002 om medicinsk udstyr, Bilag IX, Klassificeringskriterier [27], kan det sluttes at systemet udviklet tilhører klasse IIa, hvilket er defineret på baggrund af følgende:

I DEFINITIONER
1.6. *Aktivt udstyr, der er bestemt til diagnosticering*

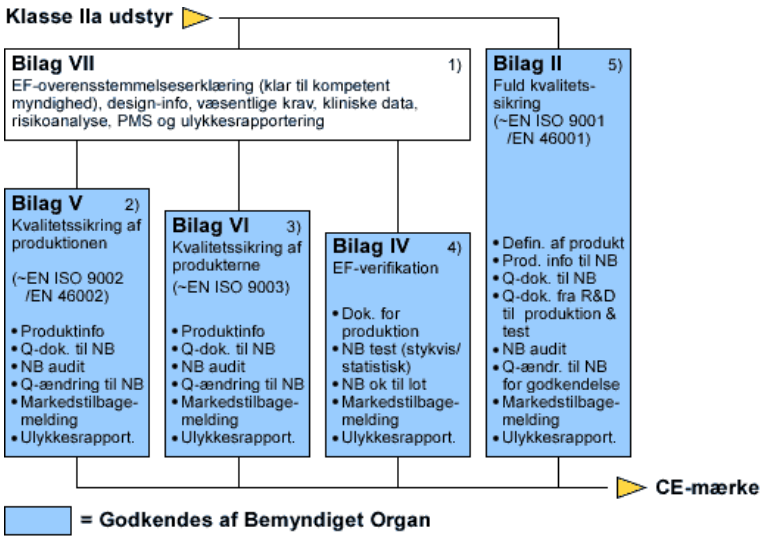
> Ethvert aktivt medicinsk udstyr, som anvendes enten alene eller sammen med andet medicinsk udstyr til at tilvejebringe oplysninger med henblik på detektion, diagnosticering, overvågning eller behandling af fysiologiske tilstande, helbredstilstande, sygdomme eller medfødte misdannelser [27].

III KLASSIFICERING
3.2. *Regel 10*

> Aktivt udstyr, der er beregnet til diagnosticering, henhører under klasse IIa
> ...
> hvis det er beregnet til at muliggøre en direkte diagnosticering eller overvågning af vitale fysiologiske processer... [27]

Klasse IIa udstyr

Figur G.1 viser processen for CE-mærkning af et produkt at klasse IIa.



Figur G.1: *Illustration af vejen til et CE-mærke [36].*

Del VI

BILAG

BILAG A

CD-ROM

Oversigt over den vedlagte CD-ROM, derudover kan datafiler med optaget ICP-data findes, samt projektrapporter for 5. og 6. semester og brugermanual til DIKTI, se yderligere README.txt.

A.1 Bilag

A.1.1 Slideshow af Preben Sørensen vedrørende Hydrocephalus

Filnavn: Hydrocephalus.ppt

A.1.2 Regler for IT-sikkerhed i Nordjyllands Amt

Filnavn: IT-sikkerhed.pdf

A.1.3 Manual til Camino MPM-1

Filnavn: Caminomanual.pdf

A.1.4 Protokol for serielkommunikation med Camino MPM-1

Filnavn: Serielprotokol.pdf

A.2 Signalbehandling

A.2.1 Matlabprototyper af algoritmerne

Filnavn: amplitudemetode.m (metode1)
Filnavn: arealmetode.m (metode2)

A.3 Kode

A.3.1 Kildekode til programmet

Bibliotek: DIKTI **Filnavn:** DIKTI.jar (Eksekverbar fil)

BILAG B

Resultater

Resultater for algoritmernes detektering af B-bølgeaktivitet ved forskellige parameterværdier.

Måling1 Amplitude (metode1)

Bølger	RMS	B-Beregn	B-Visuel	Længde	$\%B_b$	$\%B_v$	$\%B_v/\%B_b$	$\%B_{faelles}$
3,0	2,5	38926	18790	64536	60	29	48	65
3,0	3,0	32750	18790	64536	51	29	57	54
3,0	3,5	25998	18790	64536	40	29	72	39
3,0	4,0	20430	18790	64536	32	29	92	31
5,0	2,5	41834	18790	64536	65	29	45	50
5,0	3,0	34713	18790	64536	54	29	54	57
5,0	3,5	28697	18790	64536	44	29	65	49
5,0	4,0	22742	18790	64536	35	29	83	36
7,0	2,5	43823	18790	64536	68	29	43	62
7,0	3,0	37027	18790	64536	57	29	51	48
7,0	3,5	29877	18790	64536	46	29	63	48
7,0	4,0	24296	18790	64536	38	29	77	27
9,0	2,5	44584	18790	64536	69	29	42	47
9,0	3,0	36033	18790	64536	56	29	52	42
9,0	3,5	31477	18790	64536	49	29	60	48
9,0	4,0	23737	18790	64536	37	29	79	39

Måling2 Amplitude (metode1)

Bølger	RMS	B-Beregn	B-Visuel	Længde	$\%B_b$	$\%B_v$	$\%B_v/\%B_b$	$\%B_{faelles}$
3,0	2,5	23223	1800	25759	90	7	8	100
3,0	3,0	19372	1800	25759	75	7	9	100
3,0	3,5	16945	1800	25759	66	7	11	90
3,0	4,0	12722	1800	25759	49	7	14	96
5,0	2,5	23773	1800	25759	92	7	8	100
5,0	3,0	21726	1800	25759	84	7	8	100

5,0	3,5	17772	1800	25759	69	7	10	100
5,0	4,0	17772	1800	25759	69	7	10	100
7,0	2,5	24528	1800	25759	95	7	7	0
7,0	3,0	22768	1800	25759	88	7	8	100
7,0	3,5	18894	1800	25759	73	7	10	100
7,0	4,0	13375	1800	25759	52	7	13	100
9,0	2,5	24986	1800	25759	97	7	7	0
9,0	3,0	22825	1800	25759	89	7	8	100
9,0	3,5	19444	1800	25759	75	7	9	100
9,0	4,0	14477	1800	25759	56	7	12	100

Måling1 Areal (metode2)

Vindue	Areal	B-Beregn	B-visuel	Længde	$\%B_b$	$\%B_v$	$\%B_v/\%B_b$	$\%B_{faelles}$
1,5	60	28521	18790	64536	44	29	66	51
1,5	70	24471	18790	64536	38	29	77	42
1,5	80	22401	18790	64536	35	29	84	40
1,5	90	20781	18790	64536	32	29	90	35
2,0	80	28701	18790	64536	44	29	65	53
2,0	93	24921	18790	64536	39	29	75	40
2,0	107	22281	18790	64536	35	29	84	42
2,0	120	20961	18790	64536	32	29	90	38
2,5	100	28807	18790	64536	45	29	65	52
2,5	116	25407	18790	64536	39	29	74	46
2,5	134	21807	18790	64536	34	29	86	40
2,5	150	19207	18790	64536	30	29	98	36
3,0	120	27441	18790	64536	43	29	68	47
3,0	139	25281	18790	64536	39	29	74	44
3,0	161	20061	18790	64536	31	29	94	33
3,0	180	17721	18790	64536	27	29	106	34

Måling2 Areal (metode2)

Vindue	Areal	B-Beregn	B-visuel	Længde	$\%B_b$	$\%B_v$	$\%B_v/\%B_b$	$\%B_{faelles}$
1,5	60	18540	1800	25759	72	7	10	95
1,5	70	18090	1800	25759	70	7	10	95
1,5	80	18000	1800	25759	70	7	10	95
1,5	90	16560	1800	25759	64	7	11	85
2,0	80	19920	1800	25759	77	7	9	93
2,0	93	18360	1800	25759	71	7	10	93
2,0	107	16920	1800	25759	66	7	11	93
2,0	120	16200	1800	25759	63	7	11	93
2,5	100	20100	1800	25759	78	7	9	83
2,5	116	18600	1800	25759	72	7	10	75
2,5	134	17750	1800	25759	69	7	10	75
2,5	150	16500	1800	25759	64	7	11	75
3,0	120	20111	1800	25759	78	7	9	97
3,0	139	19031	1800	25759	74	7	9	97
3,0	161	17591	1800	25759	68	7	10	97
3,0	180	16691	1800	25759	65	7	11	97

$\%B_b$:	Algoritmens detektering af B-bølgeaktivitet i procent
$\%B_v$	Observatørens detektring af B-bølgeaktivitet i procent
$\%B_{faelles}$:	Fællesmængden mellem detektering af B_b og B_v i procent forhold til B_v

BILAG C

Kildekode

C.1 Boundary

Bboelger.java

```
package DIKTI.Boundary;
import DIKTI.Control.*;
import DIKTI.Entity.*;

import java.awt.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.JTextArea;
import com.jrefinery.data.*;                                                    10
/**********************************
** Brugergrænseflade for detektering af B-bølger
**********************************/

public class Bboelger extends JFrame
{
/*********************************Attributter*****************************************/
    private Container contentPane;
    private JPanel buttonPanel, labelPanel;
    private JButton hentIcp, markerBoelger, indstilParametre, afslutDetektering;   20
    private JLabel procenttal;
    private OfflineGraf offlineGraf;
    private int procent, sletgraf = 0;
    private static Bboelger bboelgerRef;
/*********************************Constructor*****************************************/
    public Bboelger()
    {
        super("Detekter B-bølger");
        offlineGraf = new OfflineGraf();
        visBrugerflade();                                                       30
        lytKnapper();
        setVisible(true);
        bboelgerRef = this;
        new Parametre();
    }
    /*****************************Metoder***********************************/
    /*Kan returnere det pågældende objekt*/
```

```
public static Bboelger bboelgerRef()
{
    return bboelgerRef;                                                    40
}
/*Opsæt brugerfladen for StartFladen*/
public void visBrugerflade()
{
    /***Ramme/panel sættes op****/
    contentPane = this.getContentPane();
    this.setSize(1024, 738);

    JPanel panel = new JPanel();
    panel.setBorder(BorderFactory.createEmptyBorder(                       50
                                            20, //top
                                            3, //left
                                            3, //bottom
                                            3) //right
                                            );
    contentPane.add(panel, BorderLayout.SOUTH);
    panel.setLayout(new GridLayout(0, 7));
    contentPane.add(offlineGraf.visGraf(true, false, "Detektering af B-bølger"),BorderLayout.CENTER,1);
    /***Knapper******/
    hentIcp = new JButton("Hent ICP-datafil");                             60
    hentIcp.setPreferredSize(new Dimension(0,50));
    markerBoelger = new JButton("Marker B-bølger");
    indstilParametre = new JButton("Indstil parametre");
    afslutDetektering = new JButton("Afslut");
    JLabel mellem1 = new JLabel("");
    JLabel mellem2 = new JLabel("");
    procenttal = new JLabel("");
    panel.add(hentIcp);
    panel.add(markerBoelger);
    panel.add(mellem1);                                                    70
    panel.add(procenttal);
    panel.add(mellem2);
    panel.add(indstilParametre);
    panel.add(afslutDetektering);
    /*Laver genvejstasterne Alt og h,b,i,a*/
    hentIcp.setMnemonic(KeyEvent.VK_H);
    markerBoelger.setMnemonic(KeyEvent.VK_B);
    indstilParametre.setMnemonic(KeyEvent.VK_I);
    afslutDetektering.setMnemonic(KeyEvent.VK_A);
    /*Inaktiverer knapper fra start*/                                      80
    markerBoelger.setEnabled(false);
    indstilParametre.setEnabled(false);
}
/*Opretter klasser så der lyttes på knapperne*/
public void lytKnapper()
{
    /*Lytter på knappen Hent ICP  */
    ActionListener hentIcplistener = new ActionListener()
        {
```

```
            public void actionPerformed(ActionEvent e)                                    90
            {
                  if (offlineGraf.hentIcpDatafil("icp")) //er sand vælges en datafil
                  {
                        contentPane.remove(1);
                        contentPane.add(offlineGraf.visGraf(
                              false, false, "Detektering af B-bølger"),BorderLayout.CENTER,1);
                        contentPane.validate();
                        Kommentar kommentaren = new Kommentar();
                        Patient patient = new Patient();
                        kommentaren.tilfoerGrafKommentarVector(patient.LaesKommentarFil(          100
                        offlineGraf.filnavn()+"kom"), patient.LaesKommentarFil(
                        offlineGraf.filnavn()+"tid"), offlineGraf.plottet());
                        indstilParametre.setEnabled(true);
                        markerBoelger.setEnabled(true);
                  }
            }
        };
    hentIcp.addActionListener(hentIcplistener);
/*Lytter på knappen Beregn B-bølger*/
    ActionListener markerBoelgerlistener = new ActionListener()                           110
        {
            public void actionPerformed(ActionEvent e)

            {
                  offlineGraf.sletSerie();
                  offlineGraf.initParametre(Parametre.parametreRef().defaultParametre());
                  offlineGraf.markerBboelger();
                  procenttal.setText("B-bølgeaktivitet:  " +offlineGraf.procentBoelger() +"%");
                  contentPane.validate();
            }                                                                             120
        };
    markerBoelger.addActionListener(markerBoelgerlistener);
/*Lytter på knappen Indstill parametre*/
    ActionListener indstillerListener = new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                Password password = new Password(new JFrame());
            }
        };                                                                                130
    indstilParametre.addActionListener(indstillerListener);
/*Lytter på knappen til startbrugerflade*/
ActionListener afslutlistener = new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                new Main();
                Bboelger.this.dispose();
            }
        };                                                                                140
    afslutDetektering.addActionListener(afslutlistener);
```

```
    /*Lyt på om vinduet bliver lukket og stop program*/
        WindowAdapter w = new WindowAdapter()
            {
                public void windowClosing(WindowEvent e)
                {
                    Bboelger.this.dispose();
                    System.exit(0);
                }
            };
        this.addWindowListener(w);
    }
    /*Bestemmer aktiveringen af indstil parametre knappen*/
    public void aktiverIndstil(boolean sat)
    {
        indstilParametre.setEnabled(sat);
    }
}
```

Icp.java

```
package DIKTI.Boundary;
import DIKTI.Control.*;
import DIKTI.Entity.*;
import java.awt.*;
import java.util.*;
import javax.swing.*;
import java.awt.event.*;
import com.jrefinery.chart.annotations.XYTextAnnotation;
/*********************************
** Brugergrænseflade for 24-timers ICP-måling
*********************************/
public class Icp extends JFrame
{
/*********************************Attributter*****************************/
    private Container contentPane;
    private JPanel buttonPanel, kommentarPanel;
    private JButton egenKommentar, start, afslut;
    private JOptionPane option;
    private boolean startet = false;
    public IcpData kommu;
    public static String inputVar;
    public static OnlineGraf graf;
    public static double doubleTid;
/*********************************Constructor*****************************/
        public Icp()
        {
        super("24-timers ICPmåling");//overskrift på BGF
        inputVar = null;//sikrer mod nullpointer
        visIcpBrugerflade();
```

```
        lytKnapper();                                                          30
        setVisible(true);
        }
/*****************************Metoder*************************************/
/*Opsæt brugerfladen for 24-timers måling*/
        public void visIcpBrugerflade()
        {
                contentPane = this.getContentPane();//Rammepanel opsættes
                this.setSize(1024, 738);//sætter pixel størrelse
                /*Opsætter et knappanel i rammepanel*/
                buttonPanel = new JPanel();                                    40
                buttonPanel.setBorder(BorderFactory.createEmptyBorder(
                                                        5, //top
                                                        5, //venstre
                                                        5, //bund
                                                        5) //højre
                                                        );
                buttonPanel.setLayout(new GridLayout(0,6 ));
                contentPane.add(buttonPanel, BorderLayout.SOUTH);
                /*Opsætter labels og knapper til rammepanel*/
                JLabel mellem1 = new JLabel("");                               50
                JLabel mellem2 = new JLabel("");
                egenKommentar = new JButton("Egen kommentar");
                egenKommentar.setPreferredSize(new Dimension(0,20));//Angiver størrelsen af knappen
                start = new JButton("Start");
                afslut = new JButton("Afslut");
                buttonPanel.add(start);//sætter knapper på panelet
                buttonPanel.add(mellem1);
                buttonPanel.add(new PraeDefKommentar());
                buttonPanel.add(egenKommentar);
                buttonPanel.add(mellem2);                                      60
                buttonPanel.add(afslut);
                /*Disabler nogle knapper fra start*/
                egenKommentar.setEnabled(false);
                PraeDefKommentar.ajl.setEnabled(false);
                graf = new OnlineGraf();
                //tilføjer graf x-akse: 2520 samples, y-akse:fra -20 til 80 (mmHg)
                contentPane.add(graf.icpPaaGraf(), BorderLayout.CENTER);
                /*Laver genvejstasterne Alt + bogstaverne s,a,e*/
                start.setMnemonic(KeyEvent.VK_S);
                afslut.setMnemonic(KeyEvent.VK_A);                             70
                egenKommentar.setMnemonic(KeyEvent.VK_E);
        }
/*Opretter klasser, så der kan lyttes på knapperne*/
public void lytKnapper()
{
/*Lyt på om vinduet bliver lukket og afslut program*/
                WindowAdapter w = new WindowAdapter()
                        {
                                public void windowClosing(WindowEvent e)
                                {                                              80
                                        Icp.this.dispose();
```

```
                        System.exit(0);
                    }
                };
            this.addWindowListener(w);
/*Lyt på startknap og gør knapper brugelige*/
            ActionListener startlistener = new ActionListener()
                    {
                        public void actionPerformed(ActionEvent e)
                        {                                                          90
                            kommu = new IcpData();//starter serielkommunikationen
                            egenKommentar.setEnabled(true);
                            start.setEnabled(false);
                            afslut.setEnabled(true);
                            PraeDefKommentar.ajl.setEnabled(true);
                            startet = true;
                        }
                    };
                start.addActionListener(startlistener);
/*Lyt på afslutknap og gør knapper brugelige*/                               100
            ActionListener afslutlistener = new ActionListener()
                    {
                        public void actionPerformed(ActionEvent e)
                        {
                            Main main = new Main();//opretter Main
                            Icp.this.dispose();//afslutter 24 ICP-måling
                            GemFil gemFil = new GemFil();
                            //gemmer tid og referencesamples i fil
                            Kommentar.kommentarerReturner().add(
                              SerielKommuTraad.tidspunkt());                   110
                            Kommentar.tidspunkterReturner().add(
                              ""+SerielKommuTraad.returnerSampleNr());
                            Patient.kommentarTilFil(
                              Kommentar.kommentarerReturner(),
                              gemFil.icpFilNavn()+"kom");
                            Patient.kommentarTilFil(
                              Kommentar.tidspunkterReturner(),
                              gemFil.icpFilNavn()+"tid");
                            if (startet)//tjekker om der er startet en måling
                            {                                                  120
                            kommu.lukPort();//com-porten lukkes
                            IcpData.pw.close();//lukker for skrivning til fil
                            }
                        }
                    };
                afslut.addActionListener(afslutlistener);
/*Lyt på egenKommentar-knap og aktiver knapper*/
            ActionListener egenKommentarlistener = new ActionListener()
                    {
                        public void actionPerformed(ActionEvent e)             130
                        {
                            option = new JOptionPane();
                            inputVar = option.showInputDialog(null,"Indskiv kommentar: ",
```

```
                                        "Tilføj på graf", JOptionPane.PLAIN_MESSAGE);
                                        Kommentar kommentar = new Kommentar();
                                        kommentar.tilfoerKommentar();
                                    }
                                };
                        egenKommentar.addActionListener(egenKommentarlistener);
                    }                                                                    140
}
```

IcpData.java

```
package DIKTI.Boundary;
import DIKTI.Control.*;
import DIKTI.Entity.*;
import java.io.*;
import java.lang.*;
import java.util.*;
import javax.comm.*;
import javax.swing.*;
import com.jrefinery.data.Millisecond;
/*****************************************************                              10
** Opretter seriel kommunikation med Camino MPM-1
** konverterer tid og ICP fra Ascii-bytes til doubleværdi
*****************************************************/
public class IcpData implements SerialPortEventListener
{
/***************************Atrributter*************************************/
        private GemFil gemFil;
        private Patient patient;
        private FileOutputStream ud;
        private InputStream inputStream;                                             20
        private boolean portFound;
        private int taeller=0, stoerst = 0;
        private double sum =0;
        private static Millisecond milli;
        private static String defaultPort;
        private static Enumeration portList;
        private static CommPortIdentifier portId, portIdcom1;
        private static boolean t = false;
        private static int i = 0, j = 0;
        private static long ms;                                                      30
        private static double timer, minutter, sekunder, midletIcp;
        public SerialPort serialPort;
        public static Date tid;
        public static PrintWriter pw;
        public static byte[] array, icpArray= new byte[4];
/************************Constructor*************************************/
    public IcpData()
        {
                try
                {                                                                    40
                GemFil gemFil = new GemFil();//åbner gem fil dialogboks
```

```
            Patient patient = new Patient();
            //Opretter FileOutputStream
            FileOutputStream dataFil = new FileOutputStream(gemFil.icpFilNavn());
            pw = new PrintWriter(dataFil);
            }
            //Udskriver fejlmeddelelse til skærmen
            catch (FileNotFoundException f){JOptionPane.showMessageDialog(null,
                                        "Der kunne ike gemmes i filen. Prøv igen!",
                                        "Filskrivningsfejl!", JOptionPane.WARNING_MESSAGE)
        try
        {
            serialPort = (SerialPort) portIdcom1.open("SimpleReadApp", 2000);//Serill porten åbnes
        }
        //Udskriver fejlmeddelelse til skærmen når porten er i brug
        catch (PortInUseException e) {JOptionPane.showMessageDialog(null,
                                "Porten er i brug!Afslut all programmer der bruger COM-port 1",
                                "Kommunikationsfejl!", JOptionPane.WARNING_MESSAGE);}
        try
        {
            inputStream = serialPort.getInputStream();//Data fra serielporten tilføjes til inputStream
        }
        catch (IOException e){}
        try
        {
            serialPort.addEventListener(this);//Eventlistner der kigger på seriell porten oprettes
        }
        catch (TooManyListenersException e) {}
        serialPort.notifyOnDataAvailable(true);
        try
        {       //opsætning af seriel forbindelse: 19200 baud, 7 bit, 1 stopbit, ulige paritet
            serialPort.setSerialPortParams(19200, SerialPort.DATABITS_7,
                                        SerialPort.STOPBITS_1,
                                        SerialPort.PARITY_ODD);
        }
        catch (UnsupportedCommOperationException e){}
}
/*********************************Metoder*********************************/
/*serialEvent*/
    public void serialEvent(SerialPortEvent event)
        {
        switch (event.getEventType())
        {
        case SerialPortEvent.BI://Break interrupt
        case SerialPortEvent.OE://Overrun error
        case SerialPortEvent.FE://Framing error
        case SerialPortEvent.PE://Parity error
        case SerialPortEvent.CD://Carrier detect
        case SerialPortEvent.CTS://Clear to send
        case SerialPortEvent.DSR://Data set ready
        case SerialPortEvent.RI://Ring indicator
        case SerialPortEvent.OUTPUT_BUFFER_EMPTY://Output buffer er tom
            break;//I alle ovenstående situationer skal der hoppes ud af metoden
```

```
case SerialPortEvent.DATA_AVAILABLE://Data er tilgængelig på den serielle port
   byte[] readBuffer = new byte[1000];//Oprettelse af byte array med 1000 pladser
      try
      {
      int str = 0;
            byte readBuffer1;
            //While løkken kører så længe der er data tilgængelig på seriell porten
            while (inputStream.available() > 0)
            {
            int numBytes = inputStream.read(readBuffer);//angiver antallet af bytes i bufferen
                  if (numBytes>stoerst)
                  {
                  stoerst=numBytes;
                  System.out.println("I stoerst står:" + stoerst);
                  }
                  for(str = 0; str < numBytes; str++)//læser alle bytes i bufferen
                  {
                        readBuffer1 = readBuffer[str];//readBuffer1 angives i ASCII
                        if (t == true)
                        {
                        switch (readBuffer1)
                        {
                              case 65://Situationen at der står B i readBuffer1
                              case 88://Situationen at der står X i readBuffer1
                              case 120://Situationen at der står x i readBuffer1
                              case 89://Situationen at der står Y i readBuffer1
                              case 90://Situationen at der står Z i readBuffer1
                              t = false; break;
                              case 82: break; //Der står R i readBuffer1
                              default:
                                 icpArray[j]=readBuffer[str];
                                       j++;
                        if (j == 4)
                        {
                        taeller=taeller+1;//angiver antal samples
                        sum =konverterIcp(icpArray)+sum;
                              //skal kun starte serielKommuTraad 1 gang i sek.
                              if (taeller==190)//camino sender data med 190Hz
                              {
                              midletIcp=sum/190;//midles til 1Hz
                              sum=0;
                              Thread serielKommuTraad = new SerielKommuTraad();
                              taeller=0;
                              //Udfører handling, når GUI har tid til at opdatere
                              SwingUtilities.invokeLater(serielKommuTraad);
                              pw.println(midletIcp);//skriver til fil
                              }
                              t = false;
                              j = 0;
                              }
                        }
                  }
```

```
                    else
                        {
                                if (readBuffer[str] == 82)
                                {
                                    t = true;                                    150
                                    i = 0;
                                }
                        }
                        }
            }
        }
            //Ved input output fejl, skrives der en fejlmeddelse til skærmen
            catch (IOException e) {JOptionPane.showMessageDialog(null,
                            "Kan ikke skrive til fil! Prøv igen!",
                            "Filskrivningsfejl", JOptionPane.WARNING_MESSAGE);}  160
        break;
    }
    }
/*lukker port*/
public void lukPort()
{
    serialPort.close();//Seriell porten lukkes
}
/*returnerIcp*/
    public static double returnerIcp()                                           170
    {
        double icp = midletIcp;//Værdien fra midletIcp gemmes i variablen af typen double
            return icp;
    }
/* konverterIcp*/
    public static double konverterIcp(byte[] onlineArray)
    {
    double icpKonverteret;
    icpKonverteret=0;
    int i=0;                                                                     180
        if  (onlineArray[0]==48)//Hvis første byte er 0
            {
            icpKonverteret= ((((onlineArray[1]-48)*100.0)
                        +((onlineArray[2]-48)*10.0)+(onlineArray[3]-48))/10.0);
            }
        else
            if (onlineArray[0]==45) //Hvis første byte er et -
            {
            icpKonverteret = (-(((onlineArray[1]-48)*100.0)
                          +((onlineArray[2]-48)*10.0)+(onlineArray[3]-48)))/10.0;  190
            }
            else
            if (onlineArray[0]>48) //Hvis første byte er større end 0
            {
                icpKonverteret= (((onlineArray[0]-48)*1000.0)
                            +((onlineArray[1]-48)*100.0)
                            +((onlineArray[2]-48)*10.0)+(onlineArray[3]-48))/10.0;
```

```
                    }
                return icpKonverteret;
        }
/*Portidentifikation*/
public static void portIdentifikation()
{
        boolean portFound = false;
        defaultPort = "COM1";
        portList = CommPortIdentifier.getPortIdentifiers();
        while (portList.hasMoreElements())
        {
                portId = (CommPortIdentifier) portList.nextElement();
                if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL)
                {
                    //Hvis elementet i port listen er deafult porten com1 så udfør
                    if (portId.getName().equals(defaultPort))
                        {
                            System.out.println("Found port: "+defaultPort);
                            portFound = true;
                            //Værdien fra portId, som er deafultporten, gemmes i portIdcom1
                            portIdcom1 = portId;
                        }
                }
         }
        //Hvis porten ikke findes skal der udskrives fejlmeddelse til skærmen
        if (!portFound)
        {
                JOptionPane.showMessageDialog(null,
                                        "port " + defaultPort + " er ikke fundet.",
                                        "Serielkommunikationsfejl", JOptionPane.WARNING_MESSAGE);
        }
}
}
```

Infusionstest.java

```
package DIKTI.Boundary;
import DIKTI.Control.*;
import DIKTI.Entity.*;
import java.awt.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.*;
import com.jrefinery.chart.annotations.XYTextAnnotation;
/**************************************************************
** Brugergrænseflade for Infusionstesten
**************************************************************/
public class Infusionstest extends JFrame
{
/*********************************Attributter*****************************/
```

```
    private Container contentPane;
    private JPanel buttonPanel;
    private JButton start, stop, filtrerSignal, hentFil, afslut, beregnRout;
    private JLabel rout1, rout2, visRout, indtast, tekst, enhed;
    private JTextField rate;
    private String infusionsrate;                                              20
    private IcpData kommu;
    private OfflineGraf offlineGraf;
    public static String inputVar;
    private boolean startet = false;
/**********************************Constructor*****************************/
        public Infusionstest()
        {
                super("Infusionstest"); //Nedarver fra JFrame constructor
                inputVar = null;
                offlineGraf = new OfflineGraf();                               30
                visInfBrugerflade();
                lytKnapper();
                setVisible(true);
        }
/*****************************Metoder**************************************/
/*Opsæt brugerfladen for Infusionstest*/
        public void visInfBrugerflade()
        {
                contentPane = this.getContentPane();
                this.setSize(1024, 738);                                       40
                buttonPanel = new JPanel();
                contentPane.add(buttonPanel, BorderLayout.SOUTH);
                buttonPanel.setBorder(BorderFactory.createEmptyBorder(
                                                        5, //top
                                                        8, //left
                                                        5, //bottom
                                                        8) //right
                                                         );
                buttonPanel.setLayout(new GridLayout(0,8));
                //Definerer knapper                                            50
                start = new JButton("Start");
                start.setPreferredSize(new Dimension(0,50));
                stop = new JButton("Stop");
                filtrerSignal = new JButton("Filtrer ICP");
                hentFil = new JButton("Hent ICP-datafil");
                beregnRout = new JButton("Beregn R_out");
                afslut = new JButton("Afslut");
                visRout = new JLabel("");
                indtast = new JLabel("");
                visRout();//Vedrørende beregning af Rout                       60
                //tilføjer knapper til panelet
                buttonPanel.add(start);
                buttonPanel.add(stop);
                buttonPanel.add(hentFil);
                buttonPanel.add(filtrerSignal);
                buttonPanel.add(beregnRout);
```

```
            buttonPanel.add(indtast);
            buttonPanel.add(visRout);
            buttonPanel.add(afslut);
            /*Laver genvejstasterne Alt og bogstag*/                                    70
            start.setMnemonic(KeyEvent.VK_S);
            stop.setMnemonic(KeyEvent.VK_P);
            hentFil.setMnemonic(KeyEvent.VK_H);
            filtrerSignal.setMnemonic(KeyEvent.VK_F);
            beregnRout.setMnemonic(KeyEvent.VK_B);
            afslut.setMnemonic(KeyEvent.VK_A);
            /*Disabler nogle knapper fra start*/
            stop.setEnabled(false);
            filtrerSignal.setEnabled(false);
            beregnRout.setEnabled(false);                                             80
            rout1.setEnabled(false);
            tekst.setEnabled(false);
            enhed.setEnabled(false);
            /*Viser koordinatsystem, hvor grafen skal plottes*/
            OnlineGraf onlineGraf = new OnlineGraf();
            //tilføjer online graf: x-akse 7200 samples(2timer ved 1Hz), y-akse fra -20 til 80
            contentPane.add(onlineGraf.icpPaaGrafInf(), BorderLayout.CENTER,1);
        }
        public void visRout()
        {                                                                             90
            //Definerer knapper, labels og tekstområde
            visRout.setLayout(new GridLayout(2,1));
            rout1 = new JLabel("R_out : ");
            rout2 = new JLabel("");
            visRout.add(rout1);
            visRout.add(rout2);
            indtast.setLayout(new GridLayout(2,1));
            tekst = new JLabel(" Infusionsrate  ");
            JLabel indstil = new JLabel();
            indstil.setLayout(new GridLayout(1,2));                                   100
            rate = new JTextField(infusionsrate);//Tager teksten i feltet
            rate.setText("1.0");//1.0 sættes som default i tekstfeltet
            enhed = new JLabel("  ml/min");
            indstil.add(rate);
            indstil.add(enhed);
            //Sætter labels og tekstfelt på panel
            indtast.add(tekst);
            indtast.add(indstil);
        }
/*Lyt på om vinduet bliver lukket og stop program*/                                 110
        public void lytKnapper()
            {
                WindowAdapter w = new WindowAdapter()
                {
                    public void windowClosing(WindowEvent e)
                    {
                        Infusionstest.this.dispose();//afslutter BGF for infusionstest
                        System.exit(0);//afslutter program
```

```
                }
            };
        this.addWindowListener(w);
/*Lyt på startknap og gør knapper brugelige*/
        ActionListener startlistener = new ActionListener()
                {
                    public void actionPerformed(ActionEvent e)
                    {
                        GemFil gemIcpFil = new GemFil();
                        if (gemIcpFil.hentFilNavn(".inf") != null)
                        //bliver null hvis der tastes cancel
                        {
                                startet = true;
                                //starter serielkommunikation
                                kommu = new IcpData();
                                start.setEnabled(false);
                                stop.setEnabled(true);
                                filtrerSignal.setEnabled(false);
                                hentFil.setEnabled(false);
                        }
                    }
                };
            start.addActionListener(startlistener);
/*Lyt på stopknap og gør knapper brugelige*/
        ActionListener stoplistener = new ActionListener()
                {
                    public void actionPerformed(ActionEvent e)
                    {
                        start.setEnabled(true);
                        filtrerSignal.setEnabled(false);
                        hentFil.setEnabled(true);
                        stop.setEnabled(false);
                        if (startet)//tjekker om der er startet en måling
                        {           //ville ellers give nullpointer
                        kommu.lukPort();//com-porten lukkes
                        IcpData.pw.close();//lukker for skrivning til fil
                        }
                    }
                };
            stop.addActionListener(stoplistener);
/*Lyt på hentFilknap og gør knapper brugelige*/
        ActionListener hentFillistener = new ActionListener()
                {
                    public void actionPerformed(ActionEvent e)
                    {
                    contentPane.remove(1);//fjerner grafen, så en ny kan plottes
                    contentPane.add(offlineGraf.visGraf(
                        true, true, "Infusionstest"),BorderLayout.CENTER,1);
                    //true hvis der ikke tastes cancel
                    if (offlineGraf.hentIcpDatafil("inf"))
                    {
                        contentPane.validate();
```

```
                                        filtrerSignal.setEnabled(true);
                                        start.setEnabled(false);
                                        beregnRout.setEnabled(true);
                                        rout1.setEnabled(true);
                                        tekst.setEnabled(true);
                                        enhed.setEnabled(true);
                                }
                                else
                                offlineGraf.grafPanel.updateUI();//opdaterer BGF
                                }                                                                    180
                        };
                hentFil.addActionListener(hentFillistener);
/*Lyt på behandelSignalknap og gør knapper brugelige*/
        ActionListener behandelSignallistener = new ActionListener()
                        {
                                public void actionPerformed(ActionEvent e)
                                {
                                        //Filtrer signalet og viser det på grafen
                                        offlineGraf.lavFiltreretInf();
                                        filtrerSignal.setEnabled(false);                             190
                                        start.setEnabled(false);
                                        stop.setEnabled(false);
                                }
                        };
                filtrerSignal.addActionListener(behandelSignallistener);
/*Lyt på beregnRout og beregner Rout*/
        ActionListener routlistener = new ActionListener()
                        {
                                public void actionPerformed(ActionEvent e)
                                {                                                                    200
                                        infusionsrate = rate.getText();//henter infusionrate
                                        double infusionsRate = Double.parseDouble(infusionsrate);
                                        if(infusionsRate>0.0)
                                        {
                                        //Beregner Rout udfra ICPslut-ICPstart/Vinf
                                        double endeligRout = (offlineGraf.Rout())/infusionsRate;
                                        //Afrunder til heltal og viser på brugerfladen
                                        rout2.setText(+Math.round(endeligRout) +" mmHg/ml/min");
                                        }
                                        else                                                         210
                                        {
                                        JOptionPane.showMessageDialog(null,
                                                "Infusionsrate ikke anvendelig",
                                                "Advarsel", JOptionPane.WARNING_MESSAGE);
                                        }
                                }
                        };
                beregnRout.addActionListener(routlistener);
/*Lyt på afslutknap og gør knapper brugelige*/
        ActionListener afslutlistener = new ActionListener()                                         220
                        {
                                public void actionPerformed(ActionEvent e)
```

```
                                {
                                        new Main();//opretter Main
                                Infusionstest.this.dispose();//afslutter infusionstesten
                                if (startet)//tjekker om der er startet en måling
                                        {                       //ville ellers give nullpointer
                                        kommu.lukPort();//com-porten lukkes
                                        IcpData.pw.close();//lukker for skrivning til fil
                                        }                                                                    230
                                }
                        };
                afslut.addActionListener(afslutlistener);
        }
}
```

Main.java

```
package DIKTI.Boundary;
import DIKTI.Control.*;
import DIKTI.Entity.*;
import java.io.*;
import java.awt.*;
import java.util.*;
import java.lang.*;
import java.awt.event.*;
import javax.comm.*;
import javax.swing.*;                                                                                        10
/**********************************************************************
** MAIN - opsætter startskærmen og åbner port til Caminokommunikation
**********************************************************************/
public class Main extends JFrame
{
/*********************************Attributter*****************************************/
        private Container contentPane;
        private JPanel buttonPanel, labelPanel;
        private JLabel navn, version, udgivelse;
        private JButton icp, bBolger, infusion, indstillinger, exit;                                         20
/*********************************Constructor*****************************************/
        public Main()
        {
                super("DIKTI");//Titel på rammen
                visStartskaerm();
                lytKnapper();
        }
/*******************************Metoder***********************************/
        /*Opsæt brugerfladen for Startskærm*/
        public void visStartskaerm()                                                                         30
        {
                contentPane = this.getContentPane();
                this.setSize(1024, 738);//Angiver pixel størrelse
                /*Opsætter et knappanel i rammepanel*/
                buttonPanel = new JPanel();
                buttonPanel.setBorder(BorderFactory.createEmptyBorder(
```

```
        5, //top
        10, //venstre
        10, //bund
        10) //højre                                                    40
        );
contentPane.add(buttonPanel, BorderLayout.EAST);//sætter knappanel til højre
buttonPanel.setLayout(new GridLayout(8,0));//rækker, kolonner for knappanel
/*Laver knapper, labels der opsættes på knappanel*/
icp = new JButton("24 timers ICP måling");
//icp.setPreferredSize(new Dimension(170,0));//knapstørrelse, gælder alle
JLabel mellem1 = new JLabel("");
JLabel mellem2 = new JLabel("");
bBolger = new JButton("Detekter B-bølger");
JLabel mellem3 = new JLabel("");                                      50
JLabel mellem4 = new JLabel("");
infusion = new JButton("Infusionstest");
exit = new JButton("Afslut");
buttonPanel.add(mellem1);
buttonPanel.add(icp);
buttonPanel.add(mellem2);
buttonPanel.add(bBolger);
buttonPanel.add(mellem3);
buttonPanel.add(infusion);
buttonPanel.add(mellem4);                                             60
buttonPanel.add(exit);

/*Laver en label til billed*/
JLabel billede = new JLabel();
Image myImage = Toolkit.getDefaultToolkit().getImage("ventrikler_negative2.JPG");
billede.setIcon(new ImageIcon(myImage));//sætter billede på startskærm
JPanel billedPanel = new JPanel();
billedPanel.add(billede);
contentPane.add(billedPanel, BorderLayout.CENTER);//sætter billedet i midten
/*Opsætter et labelpanel*/                                            70
labelPanel = new JPanel();
labelPanel.setLayout(new GridLayout(30,0)); //rækker,kolonner for labelpanel
contentPane.add(labelPanel, BorderLayout.WEST);//sætter labels til venstre
/*Laver labels der opsættes på labelpanel*/
navn = new JLabel(" DIKTI");
//navn.setPreferredSize(new Dimension(150,0));
version = new JLabel("    Version 1.00");
udgivelse = new JLabel("    Maj 2003");
labelPanel.add(version);
labelPanel.add(udgivelse);                                            80
labelPanel.add(new JLabel(""));
labelPanel.add(navn);
labelPanel.add(new JLabel(""));
labelPanel.add(new JLabel("  Digital"));
labelPanel.add(new JLabel("  Intra"));
labelPanel.add(new JLabel("  Kraniel"));
labelPanel.add(new JLabel("  Trykmåling"));
labelPanel.add(new JLabel("   og"));
```

```
        labelPanel.add(new JLabel("  Infusionstest"));
        /*Laver genvejstasterne Alt og t,b,i,a*/                                    90
        icp.setMnemonic(KeyEvent.VK_T);
        bBolger.setMnemonic(KeyEvent.VK_B);
        infusion.setMnemonic(KeyEvent.VK_I);
        exit.setMnemonic(KeyEvent.VK_A);
        setVisible(true);
    }

public void lytKnapper()
    {
        /*Lyt på om vinduet bliver lukket og stop program*/                         100
        WindowAdapter w = new WindowAdapter()
                {
                    public void windowClosing(WindowEvent e)
                    {
                        Main.this.dispose();//Main afsluttes
                        System.exit(0);//Programmet lukkes
                    }
                };
                this.addWindowListener(w);
        /*Lytter på knappen 24 timers ICP måling */                                 110
        ActionListener icplistener = new ActionListener()
                    {
                        public void actionPerformed(ActionEvent e)
                        {
                            //Objekt, der kan gemme fil
                            GemFil gemIcpFil = new GemFil();
                            //"ser" kun filer, der hedder .icp
                            gemIcpFil.hentFilNavn(".icp");
                            if (gemIcpFil.disposeMain)//hvis filen gemmes rigtigt
                            {                                                       120
                                new Icp();//24timers ICP måling startes
                                Main.this.dispose();//Main afsluttes
                            }
                        }
                    };
                icp.addActionListener(icplistener);//tilføjes icp-knappen
        /*Lytter på knappen Detekter B-bølger*/
        ActionListener bBolgerlistener = new ActionListener()
                    {
                        public void actionPerformed(ActionEvent e)                  130
                        {
                            //starter brugerfladen for B-bølger detektering
                            Bboelger bboelger = new Bboelger();
                            Main.this.dispose();//Main afsluttes
                        }
                    };
                bBolger.addActionListener(bBolgerlistener);//tilføjes bBolger-knappen        /*Lytter på knappen Infusionstest
        ActionListener infusionlistener = new ActionListener()
                    {
                        public void actionPerformed(ActionEvent e)                  140
```

```
                        {
                                //brugerflade for inf.test
                                Infusionstest infusionstest = new Infusionstest();
                                Main.this.dispose();//Main afsluttes
                        }
                };
            infusion.addActionListener(infusionlistener);//tilføjes infusions-knappen          /*Lytter på knappen Exit*/
        ActionListener exitlistener = new ActionListener()
                {
                        public void actionPerformed(ActionEvent e)                              150
                        {
                        Main.this.dispose();//Main afsluttes
                        System.exit(0);//Programmet afsluttes
                        }
                };
            exit.addActionListener(exitlistener);//tilføjes exit-knappen
    }

/*Mainmetoden*/
    public static void main(String[] args)                                                      160
    {
        IcpData.portIdentifikation();//metode, der sørger for at opsætte seriel kommunikation
        Main mainMetode = new Main();//starter Main

    }
}
```

Parametre.java

```
package DIKTI.Boundary;
import DIKTI.Control.*;
import DIKTI.Entity.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/************************************************************
 ** Håndeterer parametre for B-bølge detektering
 ************************************************************/
public class Parametre extends JFrame                                                          10
{
/*********************Attributter*********************************/
    private JPanel panel;
    private JPanel defaultPanel;
    private Container contentPane;
    private JTextField tekst1, tekst2, tekst3;
    private JButton gemButton, defaultButton, afslut;
    private final String energi =  "120.0";//energi
    private final String vindue = "3.0";//vindueslængde
    private final String stoej = "20.0";//støj                                                  20
    private static String parameter1 = "120.0";
    private static String parameter2 = "3.0";
    private static String parameter3 = "20.0";
```

```
    private int [] parametreArray;
    private static Parametre parametreRef;
/*************Constructor*****************************************/
    public Parametre()
    {
        super("Indstil parametre");
        lytLukVindue();
        parametreRef = this;
    }
/*************Metoder*********************************************/
        public static Parametre parametreRef()
        {
                return parametreRef;
        }
    /*Initialisere parametrerne*/
    public int [] defaultParametre()
    {
        parametreArray = new int[3];//array med 3 pladser
        parametreArray[0]=(int)Double.parseDouble(parameter1);
        parametreArray[1]=(int)Double.parseDouble(parameter2);
        parametreArray[2]=(int)Double.parseDouble(parameter3);
        return parametreArray;
    }
    /*laver en brugerflade, der muliggør parameterændring*/
    public void parametreGUI()
    {
        contentPane = this.getContentPane();

        JLabel label = new JLabel("Energi: ");
        JLabel label1 = new JLabel("Vindueslængde: ");
        JLabel label2 = new JLabel("Støj: ");
        //skriver eksisterende værdier i felterne
        tekst1 = new JTextField(parameter1);
        tekst2 = new JTextField(parameter2);
        tekst3 = new JTextField(parameter3);
        gemButton = new JButton("Gem");
        gemButton.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                    parameter1 = tekst1.getText();//henter værdi i teskstfelt
                    parameter2 = tekst2.getText();
                    parameter3 = tekst3.getText();
                }
            });
        label.setLabelFor(gemButton);
        defaultButton = new JButton("Standardindstillinger");
        defaultButton.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                    parameter1 = energi;//sætter "standard" værdier
```

```
            parameter2 = vindue;
            parameter3 = stoej;
            tekst1.setText(energi);//skriver "standard" værdier i tekstfeltet
            tekst2.setText(vindue);
            tekst3.setText(stoej);                                              80
        }
    });
label.setLabelFor(defaultButton);
afslut = new JButton("Afslut");
afslut.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
                Bboelger.bboelgerRef().aktiverIndstil(true);
            Parametre.this.dispose();//afslutter BGF for parametre              90
        }
    });
label.setLabelFor(afslut);
defaultPanel = new JPanel();
defaultPanel.setLayout(new GridLayout(0, 3));
JLabel mellem = new JLabel();
defaultPanel.add(defaultButton);//sætter knapper på panel
defaultPanel.add(gemButton);
defaultPanel.add(afslut);
contentPane.add(defaultPanel, BorderLayout.SOUTH);                              100
panel = new JPanel();
panel.setBorder(BorderFactory.createEmptyBorder(
                                            5, //top
                                            5, //left
                                            5, //bottom
                                            5) //right
                                             );
JLabel label4 = new JLabel();
JLabel label5 = new JLabel("min");
JLabel label6 = new JLabel("mmHg");                                             110
panel.setLayout(new GridLayout(3, 3));
//sætter tekstfelter og labels på panel
panel.add(label);
panel.add(tekst1);
panel.add(label4);
panel.add(label1);
panel.add(tekst2);
panel.add(label5);
panel.add(label2);
panel.add(tekst3);                                                              120
panel.add(label6);
contentPane.add(panel, BorderLayout.NORTH);
pack();
setVisible(true);
/*sætter genvejstaster Alt og g, a, s*/
gemButton.setMnemonic(KeyEvent.VK_G);
afslut.setMnemonic(KeyEvent.VK_A);
```

```
        defaultButton.setMnemonic(KeyEvent.VK_S);
    }
    /*Lyt på om vinduet bliver lukket og stop program*/
    public void lytLukVindue()
    {
        WindowAdapter w = new WindowAdapter()
            {
                public void windowClosing(WindowEvent e)
                {
                        Bboelger.bboelgerRef().aktiverIndstil(true);
                    Parametre.this.dispose();//afslutter objekt for parametre
                }
            };
        this.addWindowListener(w);
    }
}
```

Password.java

```
package DIKTI.Boundary;
import DIKTI.Control.*;
import DIKTI.Entity.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.util.Vector;
import javax.swing.*;
/*****************************************************************
** Håndterer og tjekker indtastning af password ved indstilling af parametre
*****************************************************************/
public class Password extends JDialog
{
        private JLabel passwordLabel;
        private Container contentPane;
        private JPasswordField passwordField;
        private static String password;
        private char[] pw;
/*****************Constructor************************/
public Password (Frame parent)
{
        super(parent, "Indskriv password", true);  //JDialog constructor
        password();
}
/*laver gui til passwordpopup*/
public void password()
{
        contentPane = this.getContentPane();
        contentPane.setLayout(new GridLayout(1, 2));
        passwordLabel = new JLabel("Password: ", JLabel.RIGHT);
        passwordLabel.setForeground(Color.black);
        passwordField = new JPasswordField();//sætter stjerne i stedet for tekst
        ActionListener aListener = new ActionListener()
```

```
        {
                public void actionPerformed(ActionEvent e)
                {
                        pw = passwordField.getPassword(); //henter indtastet password
                        password = new String(pw).trim();
                        Password.this.setVisible(false);
                        if (password.equals("icpTest"))//tester om indtastet er lig krævet password
                        {
                                Bboelger.bboelgerRef().aktiverIndstil(false);
                                Parametre.parametreRef().parametreGUI();
                        }
                        else
                        {
                        JOptionPane.showMessageDialog(null, "Password ikke korrekt",
                                                "Advarsel", JOptionPane.WARNING_MESSAGE);
                        }
                }
        };
        passwordField.addActionListener(aListener);
        contentPane.add(passwordLabel);//tilføjer label til panel
        contentPane.add(passwordField);//tilføjer passwordfelt til panel
        this.setSize(200, 60);//sætter størrelse af password-dialogboks
        this.setVisible(true);
}
/*returnerer det password der er indtastet*/
private static String getPassword()
{
        return password;
}
}
```

PraeDefKommentar.java

```
package DIKTI.Boundary;
import DIKTI.Control.*;
import DIKTI.Entity.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/****************************************
** Laver liste med prædefinerede kommentarer
****************************************/
public class PraeDefKommentar extends JPanel
{
        public static String pttKommentar;
        public static ActionJList ajl;  //objekt af ActionJList
/*Laver en GUI-liste med prædefinerede kommentarer*/
public PraeDefKommentar()
{
        setLayout(new GridLayout(1,1));
        String[] items =
        { "Uro","Sover","Vågen","Sidder","Ligger","Tilkoblet","Frakoblet"};
```

```
        ajl = new ActionJList(items);                                          20
        ajl.setFixedCellWidth(133);// cellebredde
        ajl.setVisibleRowCount(3);//sætter tre rækker synlige
        ajl.addActionListener(new ActionListener()
        {
           public void actionPerformed(ActionEvent ae)
                  {
                      pttKommentar = ajl.getSelectedValue().toString();
                      Kommentar kommentar = new Kommentar();
                      kommentar.tilfoerDefKommentar();
                  }                                                            30
        });//gemmer kommentar i vektor
        JScrollPane jsp = new JScrollPane(); //gør det muligt at scrolle i listen
        jsp.getViewport().add(ajl);//sætter ActionJList på JScrollPane
        add(jsp);
}
}
```

C.2 Control

ActionJList.java

```
package DIKTI.Control;
import DIKTI.Boundary.*;
import DIKTI.Entity.*;
import javax.swing.*;
import java.awt.event.*;

/*****************************************************************************
** Håndterer valg ved prædefineret kommentar i forhold til tryk på knapper
*****************************************************************************/
                                                                               10
public class ActionJList extends JList
{
        ActionListener al;

public ActionJList(String[] it)
{
        super(it); //nedarver frame
        addMouseListener(new MouseAdapter()
        {
                public void mouseClicked(MouseEvent me)                        20
                {
                        if (al == null) //hvis der ikke er valgt noget
                        return;
                        Object ob[] = getSelectedValues();//læser valgt kommentar
                        if (ob.length > 1) //Hvis der er valgt mere end en værdi sker ingenting
                        return;
                        if (me.getClickCount() == 2) //lytter på doubleklik
                        {
                        al.actionPerformed(new ActionEvent(this,
```

```
                                        ActionEvent.ACTION_PERFORMED,                    30
                        ob[0].toString()));
                                //Konsumerer this event så det ikke processeres per default
                                //af kilden der skabte det.
                        me.consume();
                        }
                }
        });
        addKeyListener(new KeyAdapter() //lytter på enter-tast
        {
                public void keyReleased(KeyEvent ke)                                     40
                {
                if (al == null)
                        return;
                Object ob[] = getSelectedValues(); //læser valgt kommentar
                if (ob.length > 1) //Hvis der er valgt mere end en værdi sker ingenting
                        return;
                if (ke.getKeyCode() == KeyEvent.VK_ENTER)//skal kun ske noget ved tast på enter
                        {
                        al.actionPerformed(new ActionEvent(this,
                        ActionEvent.ACTION_PERFORMED,                                    50
                        ob[0].toString()));
                        ke.consume();
                        }
                }
        });
   this.setSelectedIndex(0);
}

public void addActionListener(ActionListener al)
{                                                                                        60
        this.al = al; //adderer frame til actionJListener
}
}
```

FlytlinieThread.java

```
package DIKTI.Control;
import DIKTI.Boundary.*;
import DIKTI.Entity.*;
import java.lang.*;
import java.util.*;
import javax.swing.*;

/**********************************************************************
** Opdaterer grafen når linien flyttes over skærmen i Infusionstesten
**********************************************************************/                  10

public class FlytlinieThread extends Thread
{
public void run()
{
```

```
        OfflineGraf.grafPanel.updateUI();//opdaterer grafen
}
}
```

Kommentar.java

```
package DIKTI.Control;
import DIKTI.Boundary.*;
import DIKTI.Entity.*;
import java.io.*;
import java.util.*;
import java.util.StringTokenizer;
import com.jrefinery.chart.plot.XYPlot;
import com.jrefinery.chart.annotations.XYTextAnnotation;
import java.awt.Font;
import com.jrefinery.data.XYDataset;                                    10
import com.jrefinery.data.XYSeries;
import com.jrefinery.data.XYSeriesCollection;
import com.jrefinery.chart.plot.XYPlot;
import com.jrefinery.data.Millisecond;
/*********************************************************
** Håndterer indskrivning af kommentarer på grafer
*********************************************************/
public class Kommentar
{
/*********************************Attributter*****************************/   20
private static Millisecond milli;
private static long ms;
private static double timer, minutter, sekunder;
public static Date tid;
public static Vector kommentarer=new Vector();
    public static Vector tidspunkter=new Vector();

/*********************************Metoder*********************************/
/*Tilfører egen kommentar til et grafplot*/
public void tilfoerKommentar()                                          30
{
/*inputVar == null hvis der tastes "cancel"*/
if ((Icp.inputVar != null) && (Icp.inputVar != ""))
        {
                //tilføjer kommentar på grafen på grafen for 24-tims måling
        OnlineGraf.icpPlot().addAnnotation(new XYTextAnnotation(Icp.inputVar,
                                        new Font("SansSerif", Font.PLAIN, 10),
                                        SerielKommuTraad.returnerSampleNr(), 70.0));
                //skriver kommentar og tilhørende tid til vektor
                kommentarer.add(Icp.inputVar);                          40
                tidspunkter.add(""+SerielKommuTraad.returnerSampleNr());                }
}
/*Tilfør defineret kommentar til et grafplot*/
public void tilfoerDefKommentar()
{
        if ((PraeDefKommentar.pttKommentar != null) && (PraeDefKommentar.pttKommentar != ""))
```

```
        {
            if (PraeDefKommentar.pttKommentar=="Tilkoblet" | PraeDefKommentar.pttKommentar=="Frakoblet")
                {
                        //Skriver kommentar og tid på graf for kommentaren "tilkoblet"                    50
                        OnlineGraf.icpPlot().addAnnotation(
                         new XYTextAnnotation(PraeDefKommentar.pttKommentar,
                                              new Font("SansSerif", Font.PLAIN, 10),
                                              SerielKommuTraad.returnerSampleNr(), 65.0));
                        //gemmer kommentar med tilhørende tid til vektorer
                        kommentarer.add(PraeDefKommentar.pttKommentar+" "
                                        +SerielKommuTraad.tidspunkt() );
                        tidspunkter.add(""+SerielKommuTraad.returnerSampleNr());
                }
            else                                                                                        60
                {
                        //skriver kommentar på graf
                        OnlineGraf.icpPlot().addAnnotation(
                         new XYTextAnnotation(PraeDefKommentar.pttKommentar,
                                              new Font("SansSerif", Font.PLAIN, 10),
                                              SerielKommuTraad.returnerSampleNr(), 62.0));
                        //gemmer oplysninger i fil
                        kommentarer.add(PraeDefKommentar.pttKommentar);
                        tidspunkter.add(""+SerielKommuTraad.returnerSampleNr());
                }                                                                                       70
        }
 }
/*Tilfør de valgte kommentarer gemt i vektorer til graf*/
public void tilfoerGrafKommentarVector(Vector kommentarerVector,
                                       Vector tidspunkterVector, XYPlot plottet)
{
        for (int i =0; i<=kommentarerVector.size()-2;i++)
        {
        switch (i%3)
        {                                                                                               80
        case 0: plottet.addAnnotation(new XYTextAnnotation(kommentarerVector.elementAt(i).toString(),
                                      new Font("SansSerif", Font.PLAIN, 10),
                                      konverterVectorTidTilDouble(
                                      tidspunkterVector.elementAt(i).toString()), 52.5));
            break;
        case 1: plottet.addAnnotation(new XYTextAnnotation(kommentarerVector.elementAt(i).toString(),
                                      new Font("SansSerif", Font.PLAIN, 10),
                                      konverterVectorTidTilDouble(
                                      tidspunkterVector.elementAt(i).toString()), 50.0));
            break;                                                                                      90
        case 2: plottet.addAnnotation(new XYTextAnnotation(kommentarerVector.elementAt(i).toString(),
                                      new Font("SansSerif", Font.PLAIN, 10),
                                      konverterVectorTidTilDouble(
                                      tidspunkterVector.elementAt(i).toString()), 47.5));
            break;
        }
        }
}
```

```
/*Returnerer vektoren med indskrevne kommntarer*/
public static Vector kommentarerReturner()
{
        return kommentarer;
}

/*Returnerer vektoren med indskrevne tidspunkter til kommentarer */
public  static Vector tidspunkterReturner()
{
        return tidspunkter;
}
/*Konverterer tid til en doubleværdi, fordi der skal sættes en doubleværdi på graf*/
public static double konverterVectorTidTilDouble(String doubleTiden)
{
        double doubleTid = Double.parseDouble(doubleTiden);
        return doubleTid;
}
}
```

OfflineGraf.java

```
package DIKTI.Control;
import DIKTI.Boundary.*;
import DIKTI.Entity.*;
import com.jrefinery.chart.*;
import com.jrefinery.chart.entity.*;
import com.jrefinery.chart.ChartMouseEvent;
import com.jrefinery.chart.axis.ValueAxis;
import com.jrefinery.chart.plot.*;
import com.jrefinery.data.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
/************************************************************
** Laver graf og håndterer beregningen af B-bølger samt infusionstest
************************************************************/
public class OfflineGraf extends JFrame implements ChartMouseListener
{
/*************Atributter*****************************************/
    private Patient icpDatafil;
    private XYSeries seriesMedB,seriesInf,seriesICP;
    private ChartPanel chartPanel;
    private XYPlot plot;
    private JFreeChart chart;
    private ScrollBar scrollBar;
    private int fs,minutter,samplesIvindue,antalvindue,nedretaerskel,oevretaerskel,stoej;
    private int bProcent, sProcent, ICPtaeller = 0, sletGraf=0, musflyttet=0;
    private int [] parametre;
    private double Rout;
    private double integralSample,n,n1;
```

```
    private double y,x0, y1=0, y2=0, y3=0, y4=0, y5=0, y6=0,
        x1=0, x2=0, x3=0, x4=0, x5=0, x6=0;
    private double [] tid, icp,integral, icpfilt, filtarrayBagfra;
    private double [] ICPtal = new double[15];
    private XYSeriesCollection dataset;
    public static JPanel grafPanel;

/**************Metoder***********************************************/
/********Henter fil og læser ind i Grafen***********/
public String filnavn()
{
        return icpDatafil.icpFilNavnHent();
}
/*Henter filen*/
public boolean hentIcpDatafil(String txt)
{
        icpDatafil = new Patient();
        if (icpDatafil.filSoegning(txt))//er sand hvis der er valgt en datafil
        {
                icpDatafil.filTilArray();
                icp = icpDatafil.icp();//Indlæser data i et icp

                icpfilt = new double[icp.length];
                //Anvendes kun ved B-bølge detektering
                for(int k=icp.length-1; k>0; k--) //Filtrere signalet første gang
                {
                        icpfilt[k] = filtrerIcp(icp[k]);
                }
                seriesICP = new XYSeries("ICP");
                for(int i=0; i<icp.length; i++)
                {
                        seriesICP.add(i,icp[i]); //Sætter ICP på grafen
                }
        dataset.addSeries(seriesICP);
        return true;
        }
        return false;
}
/**************Laver grafen************************/
/*Laver grafen og sætter på et panel*/
public JPanel visGraf(boolean udenScroll, boolean medStreg, String overskrift)
{
      dataset = new XYSeriesCollection();
      chart = ChartFactory.createLineXYChart(overskrift, //Titel på graf
                                        "Tid", //x axis label
                                        "ICP (mmHg)", //y axis label
                                        dataset, //Datasæt
                                        true, //Vis legend
                                        true, //Værktøjstip
                                        false //URLs
                                        );
        chartPanel = new ChartPanel(chart);
```

```
        plot = chart.getXYPlot();
        ValueAxis yaxis = plot.getVerticalValueAxis();
        yaxis.setRange(-20.0,80.0); //Skalere Y-akse
        //Vandret streg der vedhæftes musen, når dette flyttes
        if(medStreg) //True:Infusionstest, False:B-bølge detektering
        {
                chartPanel.setVerticalAxisTrace(true);
                chartPanel.addChartMouseListener(this);                                    90
        }
        //Om det er graf med eller uden scrollbar
        if(udenScroll) //For Infusionstest
        {
                ValueAxis xaxis = plot.getDomainAxis();
                xaxis.setRange(0.0, 7200.0);
                grafPanel = new JPanel(new BorderLayout());
                grafPanel.add(chartPanel, BorderLayout.CENTER);//Graf sættes på panel
                return grafPanel;
        }                                                                                  100
        else //For B-bølge detektering
        {
                dataset.addSeries(seriesICP);
                scrollGraf();
                grafPanel = new JPanel(new BorderLayout());
                grafPanel.add(chartPanel, BorderLayout.CENTER); //Graf sættes på panel
                grafPanel.add(scrollBar, BorderLayout.SOUTH); //Scrollbar sættes på panel
                return grafPanel;
    }
}                                                                                          110
/*Returnerer chart*/
public JFreeChart charten()
{
        return chart;
}
/*Returnerer plottet*/
public XYPlot plottet()
{
        return plot;
}                                                                                          120
/*Laver og håndterer scrollbaren*/
public void scrollGraf()
{       //Scrolllængde = AntalSamples/ScrollMax * BlockIncrement
        int scrollMax = icp.length*3;
        int scrollMin = 1;
        int scrollExtent = 10;
        //Horisontal scrollbar, der scroller i grafen
        scrollBar = new ScrollBar(JScrollBar.HORIZONTAL,scrollMax/2,
        scrollExtent, scrollMin, scrollMax, chartPanel, this);
        scrollBar.setBlockIncrement(2000*3);//scroller 2000 samples                         130
        scrollBar.setUnitIncrement(scrollBar.getBlockIncrement()/2);//scroller ved tast på pil
        plot.addChangeListener(scrollBar);//tilføjer listener på scrollbaren til grafen
}
```

```
/**********Beregner og markerer B-bølger******************/
/*Tager kvadratet*/
public static double sqr(double x)
//Kaldes af beregnIntegral()
{
        return x*x;                                                                    140
}
/*Filtrer en inputværdi*/
public double filtrerIcp(double x)
{
        x0=x;
        //4. ordens butterworth filter(filtfilt): Koefficienter fundet i matlab
        y = 3.62142299122964*y1 - 4.94267270294415*y2 + 3.01758045977983*y3 - 0.69642928463559*y4 +
        (0.01380545367933*x0  - 0.02761090735866*x2 + 0.01380545367933*x4);
        y4 = y3;
        y3 = y2;                                                                       150
        y2 = y1;
        y1 = y;
        x4 = x3;
        x3 = x2;
        x2 = x1;
        x1 = x0;
        return y;
}
/*Sætter parametre */
public void initParametre(int [] parametre)//Kaldes af beregnIntegral()                160
{
        //Indtastede parametre, eller default parametre
        nedretaerskel=parametre[0];
        minutter=parametre[1];
        stoej=parametre[2];
        oevretaerskel=800;
        fs=1;
        //Variable til beregning af integralet
        samplesIvindue=minutter*60;
        antalvindue= icp.length/samplesIvindue;                                        170
}
/*Beregner Integrale for et vindue og lægger i array*/
public void beregnBboelger() //Kaldes af markerBbolger()
{
        integral =  new double[antalvindue];
        double sum=0;
        for (int j=0; j<antalvindue; j++) //Gennemløber antal vinduer
        {
                for (int i=0; i<samplesIvindue; i++) //Gennemløber samples i det pågældende vindue
                {       //Den filtrerede værdi til pågældende sample                   180
                n=filtrerIcp(icpfilt[i+j*samplesIvindue]);
                n1=filtrerIcp(icpfilt[i+j*samplesIvindue+1]);
                if(n<stoej)//Hvis under støj grænsen at
                        {
                                if ((n>0 & n1<0) |(n<0 & n1>0))//To samples er hhv. positiv og negativ
                                {
```

```
                        double skaering = (-n*fs)/(n1-n);
                        integralSample=Math.abs(skaering*(n/2))
                        +Math.abs((fs-skaering)*(n1/2));
                        }
                        else //Beregner integraleværdien for den pågældende sample
                        {
                        integralSample=sqr(n*(1/fs)+ (n1 - n)*(fs/2));
                        }
                else //Eller tages integraleværdien ikke i betragning
                        {
                        integralSample=0;
                        }
                sum=sum+integralSample;//Integralet i et vindue
                }
                integral[j]= sum;
                sum=0;
        }
}
/*Serie der markerer B-bølger slettes*/
public void sletSerie()
{
        if(sletGraf>0)
        {
                dataset.removeSeries(seriesMedB);
        }
        sletGraf++;
}

/*Plotter hvis betingelser er opfyldt*/
public void markerBboelger()
{
        beregnBboelger();
        seriesMedB = new XYSeries("B-bølgeaktivitet");
        bProcent = 0;
        sProcent = 0;
        for(int r=2; r<(antalvindue-2); r++)
        {       //Hvis de opsatte kriterier er opfyldt for den pågældende integraleværdi
                if((nedretaerskel<integral[r]) && (integral[r]<oevretaerskel) &&
                        ((nedretaerskel<integral[r-1]) || (nedretaerskel<integral[r+1]) ||
                        (nedretaerskel<integral[r-2]) || (nedretaerskel<integral[r+2])))
                {
                        bProcent++;
                        for(int s=0; s<samplesIvindue; s++)//B-bølgeaktivitet
                        {
                                seriesMedB.add((int)s+r*samplesIvindue,(int)-8.0);
                        }
                }
                else
                {
                        sProcent++;
                        for(int s=0; s<samplesIvindue; s++)//Ikke B-bølgeaktivitet, ej synlig
                        {
```

```
                        seriesMedB.add((int)s+r*samplesIvindue,(int)-25.0);
                    }
                }
            }
        dataset.addSeries(seriesMedB);
}
/*Berenger B-Bølge procenten*/
public int procentBoelger()
{
        int procentVaerdi;
        if(sProcent==0)
                procentVaerdi = 0;
        else
                //B-bølgeprocenten beregnes udfra markeringer
                procentVaerdi=((bProcent*100)/(bProcent+sProcent));
        return procentVaerdi;
}

/********* Behandler infusionstesten **********************/
/*Filtrer en inputværdi*/
public double filterInfusion(double x)
{
        x0=x;
        // 2. orden butterworth filter(filtfilt): Koefficienter fundet i matlab
        y = 1.94669754075618*y1 - 0.94808170610674*y2 + (0.00034604133763910*x0
                + 0.00069208267527820*x1 + 0.00034604133763910*x2);
        y2 = y1;
        y1 = y;
        x2 = x1;
        x1 = x0;
        return y;
}
/*Filtrer ICP ved infusionstest og viser det på grafen*/
public void lavFiltreretInf()
{
        seriesInf = new XYSeries("Filtreret");
        //Filtrerer første gang
        filtarrayBagfra = new double[icp.length];
        for(int k=icp.length-1; k>0 ; k--)
        {
                filtarrayBagfra[k] = filterInfusion(icp[k]);
        }
        //Filtrerer anden gang og sætter på graf
        for(int i=0; i< icp.length; i++)
        {
                seriesInf.add(i,filterInfusion(filtarrayBagfra[i]));
        }
        dataset.addSeries(seriesInf);
}

/*Lytter på musen, aktuel for infusionstesten*/
public void chartMouseClicked(ChartMouseEvent e)
```

```
{
        if(ICPtaeller>1) //Efter 2 museklik
        {
        plot.clearRangeMarkers(); //Vandrette streger fjernes
        ICPtaeller=0;
        }
         chartPanel.setVerticalAxisTrace(true);//Vedhæfter vandret streg til musemarkør
         ChartEntity entity = e.getEntity();//Punktet under musemarkøren
         if(entity!=null)//Hvis den er over dataen plottet
         {                                                                    300
                String punkt = entity.getToolTipText();//Henter koordinater
                String [] splittet = punkt.split(" ");//Splitter strengen
                if((splittet[4]).length()>3)//Tager Y-værdien (ICP-værdien)
                    ICPtal[ICPtaeller] = Double.parseDouble(splittet[4]);
                else
                    ICPtal[ICPtaeller] = (double) Integer.parseInt(splittet[4]);
                //Tegner er vandret streg, hvor der blev klikket med musen
                plot.addRangeMarker(new Marker(ICPtal[ICPtaeller],
                Color.blue, new BasicStroke(1.0f), Color.blue, 1.0f ) );
                if(ICPtaeller!=0)//Rout beregnes efter andet museklik          310
                    {
                    //Tager det nummeriske værdi af forskellen fra 1. og 2. markering
                    Rout=Math.abs((ICPtal[(ICPtaeller-1)])-(ICPtal[ICPtaeller]));
                    ICPtaeller++;
                    }
                if(ICPtaeller==2)
                        chartPanel.setVerticalAxisTrace(false);             }
        }
}
/*Lytter om musen flyttes over grafen*/                                      320
public void chartMouseMoved(ChartMouseEvent e)
{
        Thread flytlinie = new FlytlinieThread(); //Opdaterer
        SwingUtilities.invokeLater(flytlinie);
}
/*Forskel mellem de to markeringer med musen*/
public double Rout()
{
        return Rout;
}                                                                          330
}
```

OnlineGraf.java

```
package DIKTI.Control;
import DIKTI.Boundary.*;
import DIKTI.Entity.*;
import com.jrefinery.data.*;
import com.jrefinery.chart.*;
import com.jrefinery.chart.*;
import com.jrefinery.chart.plot.XYPlot;
import com.jrefinery.chart.axis.ValueAxis;
```

```
import com.jrefinery.chart.annotations.XYTextAnnotation;
/*****************************************                                    10
** Laver graf således data kan vises online
*****************************************/
public class OnlineGraf
{
/**************************Atrributter*************************************/
        private ChartPanel chartPanel;
        private XYSeriesCollection dataset;
    private static XYSeries series;
        private JFreeChart chart;
    private static XYPlot plot;                                                20
/***************************Metoder*****************************************/
/*Tegn onlineGraf*/
public void tegnGraf(String overskrift, String labelxakse)
{
        series = new XYSeries("ICP");
        dataset = new XYSeriesCollection();
        dataset.addSeries(series); //Serien inkluderes i datasættet
        chart = ChartFactory.createLineXYChart(overskrift, //Titel på graf
                                               labelxakse, //Label på x-akse
                                               "ICP (mmHg)", //Label på y-akse   30
                                               dataset, //Datasæt
                                               true, //Vis legend
                                               false, //Værktøjstip
                                               false); //URLs
        plot = chart.getXYPlot();
        chartPanel = new ChartPanel(chart);
}
/*Graf for 24-timers ICP-måling*/
public ChartPanel icpPaaGraf()
{                                                                              40
        tegnGraf("Intrakraniel trykmåling (ICP)","Tid (sek)");
        ValueAxis xaxis = plot.getDomainAxis();
        xaxis.setAutoRange(true); //Data glider automatisk over skærm
        xaxis.setFixedAutoRange(3120.0); //Der kan ses 52 min over skærm, 30 cm/t
        ValueAxis yaxis = plot.getRangeAxis();
        yaxis.setRange(-20.0,80.0); //Y-aksen skaleres
        return chartPanel;
}
/*Graf for Infusionstest*/
public ChartPanel icpPaaGrafInf()                                              50
{
        tegnGraf("Infusionstest","Tid (sek) 2 timer i alt");
        ValueAxis xaxis = plot.getDomainAxis();
        xaxis.setRange(0.0,7200.0); //Der ses 2 timer på skærmen
        ValueAxis yaxis = plot.getRangeAxis();
        yaxis.setRange(-20.0,80.0); //Y-aksen skaleres
        return chartPanel;
}
/*Der kan referes til plottet andet sted fra*/
public static XYPlot icpPlot()                                                 60
```

```
{
        return plot;
}
/*Der kan referes til sereien andet sted fra*/
public static XYSeries series()
{
        return series;
}
}
```

ScrollBar.java

```
package DIKTI.Control;
import DIKTI.Boundary.*;
import DIKTI.Entity.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.event.AdjustmentEvent;
import javax.swing.*;
import com.jrefinery.data.*;
import com.jrefinery.chart.*;
import com.jrefinery.chart.event.*;                                          10
import com.jrefinery.chart.ChartPanel;
import com.jrefinery.chart.axis.ValueAxis;
/*********************************************
** Laver en scrollbar til den hentede ICP-graf
*********************************************/
public class ScrollBar extends JScrollBar implements PlotChangeListener, AdjustmentListener
{
/**************Atributter*********************************************/
        private OfflineGraf grafen;
        private ValueAxis valueAxis;                                         20
        private ChartPanel chartPanel;
        private double offsetPercentage, displayMin, displayMax;
        private double rangeMin, rangeMax, dataMin, dataMax, displayRange;
        private boolean doAdjust;
    private int max;
/**************Constructor*******************************************/
public ScrollBar(int orientation, int value, int extent, int min, int max,
                                ChartPanel chartPanel, OfflineGraf grafen)
{
        super(orientation, value, extent, min, max);//nedarver frame         30
        this.chartPanel = chartPanel;
        this.valueAxis = chartPanel.getChart().getXYPlot().getDomainAxis();
        this.doAdjust = true;
        this.adjustScrollBar();
        this.addAdjustmentListener(this);
        this.grafen = grafen;
        this.startAkser();
}
/**************Metoder***********************************************/
/*Lytter når der skal scrolles*/                                         40
```

```
public void plotChanged(PlotChangeEvent event)
{
        this.adjustScrollBar();
}

/*Beregner værdier for justering af grafen*/
public void adjustScrollBar()
{
        if (doAdjust)//hvis scrollbaren skal justeres, dvs. der er trykket på baren
        {                                                                                    50
                getMinMaxValues();
                offsetPercentage = (rangeMin - displayMin) / displayRange;
                double value = this.getMinimum()+offsetPercentage*(this.getMaximum()-this.getMinimum());
                double ratio = (rangeMax - rangeMin)/displayRange;//regner datamængden relativt

                this.doAdjust = false;
                this.model.setValue((int)value);
                this.setVisibleAmount(5317);
                this.doAdjust = true;
        }                                                                                    60
}
/*Sætter startakserne*/
public void startAkser()
{
        this.getMinMaxValues();
        double value = ((double)(this.getValue()));
        //beregner samplenr for venstre
        double leftBound = displayMin+(displayRange)*value/this.getMaximum();
        double rightBound = leftBound+2517.0;//definerer vinduesstørrelsen til at være 2517 samples(30cm/t)
        double punkter = rightBound - leftBound;                                             70
        doAdjust = false;
        (grafen.charten()).getXYPlot().getDomainAxis().setRange(leftBound, rightBound);
        doAdjust = true;
}
/*Sætter grænserne til grafen der vises på skærmen*/
public void adjustmentValueChanged(AdjustmentEvent e)
{
        if (!e.getValueIsAdjusting())
        {
                if (doAdjust)                                                                80
                {
                        double value = ((double)(e.getValue()));
                        this.getMinMaxValues();
                        //beregner samplenr for venstre
                        double leftBound = displayMin+(displayRange)*value/this.getMaximum();
                        //definerer vinduesstørrelsen til at være 2517 samples(30cm/t)
                        double rightBound = leftBound+2517.0;
                        doAdjust = false;
                        (grafen.charten()).getXYPlot().getDomainAxis().setRange(leftBound, rightBound)
                        doAdjust = true;                                                     90
                }
        }
```

```
}
/*Sætter grænseværdier for hvor meget der skal vises*/
private void getMinMaxValues()
{
        dataMin = 0;
        rangeMin = valueAxis.getMinimumAxisValue();//henter minimum sampleværdi, dvs 0
        dataMax = max/3;//Antal samples i signalet
        rangeMax = valueAxis.getMaximumAxisValue();//henter maximum sampleværdi                    100
        displayMin = Math.min(dataMin, rangeMin);//finder hvilken værdi der er mindst
        displayMax = Math.max(dataMax, rangeMax);//finder hvilken værdi der er størst
        displayRange = displayMax - displayMin;
}
}
```

SerielKommuTraad.java

```
package DIKTI.Control;
import DIKTI.Boundary.*;
import DIKTI.Entity.*;
import java.lang.*;
import java.util.*;
import javax.swing.*;
import com.jrefinery.chart.annotations.XYTextAnnotation;
import java.awt.Font;
/****************************************
** Tråden sætter ICP på onlingraf                                                              10
** når ny data er tilgængelig fra Camino
****************************************/
public class SerielKommuTraad extends Thread
{
        private double icp;
        private static int timer, minutter, sampleNr=0;
        private static double doubleTid;
        private static boolean sammeDag= true;
        private static String time, minut;
        public static Date tid;                                                                 20
/*Kører tråden når der er data fra Caminotrykmåler*/
public void run()
{
        sampleNr= sampleNr+1; //tæller op hver gang der skal plottes på graf
        OnlineGraf onlineGrafObject = new OnlineGraf();
        icp = IcpData.returnerIcp();//metode, der returnerer midlet icp
        onlineGrafObject.series().add(sampleNr, icp);//sætter icp på grafen
        //plotter tid på grafen ved start og hver halve time
        if ((sampleNr==1)|(sampleNr%1950==0))
        {                                                                                       30
                OnlineGraf.icpPlot().addAnnotation(new XYTextAnnotation(tidspunkt(),
                                        new Font("SansSerif", Font.PLAIN, 10),
                                        SerielKommuTraad.returnerSampleNr(), 70.0));
                Kommentar.kommentarer.add(""+tidspunkt());
                Kommentar.tidspunkter.add(""+sampleNr);
        }
```

```
}
/*Laver den tid der skal sættes på graf hver halve time og til kommentarerne tilkoblet og frakoblet*/
public static String tidspunkt()
{
        Date tid = new Date();
        timer = tid.getHours();
        minutter = tid.getMinutes();
        if(timer==24)
                timer = 0;//ellers bliver tiden talt op til 24 -> fx 24:14
        if (timer<10)
                time = "0"+Integer.toString(timer);//sikrer fx 01:23
        else
                time = Integer.toString(timer);

        if (minutter<10)
                minut = "0"+Integer.toString(minutter);// sikrer fx 13:03
        else
                minut = Integer.toString(minutter);

        String tidspunkt = time+":"+minut;
        return tidspunkt;
}
/*Returnerer antallet af samples sat på grafen*/
public static int returnerSampleNr()
{
        return sampleNr;
}
}
```

C.3 Entity

GemFil.java

```
package DIKTI.Entity;
import DIKTI.Boundary.*;
import DIKTI.Control.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.lang.*;
import javax.swing.*;
/*************************************************
** Gemfil håndterer at gemme filer (både .icp og .inf)
*************************************************/
public class GemFil extends JPanel
{
/*****************************Attributter*********************************/
        private static File f;
        private static Date dato;
        private static String nymaaned, nydag;
```

```
    public static boolean disposeMain = false;
    public static boolean cancel = true;                                    20
/*************************Metoder************************************/
/*Viser directory hvor et filnavn kan søges  */
    public static File hentFilNavn(String extension)
    {
        dato = new Date();//henter dato, tid etc...
            int dag = dato.getDate();
            if (dag<10)
            {
                String dagen = Integer.toString(dag);
                nydag = "0"+dagen;//sikrer at dato vises som fx 03          30
            }
            else
            {
                nydag = Integer.toString(dag);
            }
            int maaned = dato.getMonth();
            maaned = maaned+1;//måneder tælles fra 0 til 11
            if (maaned<10)
            {
                String maaneden = Integer.toString(maaned);                 40
                nymaaned = "0"+maaneden;//sikrer at måned vises som fx 04
            }
            else
            {
                nymaaned = Integer.toString(maaned);
            }
            //Året regnes fra 1900 (fx 103)
            String nyaar = Integer.toString(1900 + dato.getYear());
            File datoen = new File(nyaar+"_"+nymaaned+"_"+nydag+extension);
        /*Laver en filechooser - specificerer i hvilket**                 50
        **directory der skal startes med at søge i******/
        JFileChooser fc = new JFileChooser(new File("\\Windows/Desktop"));
        Filhaandtering filter = new Filhaandtering();//laver filter til filsøgning
        String ext = extension.substring(1);//fjerner . i extension-variablen
        filter.addExtension(ext);//tilføjer extension icp eller inf
        filter.setDescription("datafiler");//viser at icp og inf er datafiler
        fc.setFileFilter(filter);//tilføjer filteret til filechooser
        fc.setSelectedFile(datoen);//tilføjer et i forvejen valgt navn(datoen) på filechoosere
        int returnVal = fc.showSaveDialog(null);
        if (returnVal == JFileChooser.APPROVE_OPTION)//ved OK               60
        {
            f = fc.getSelectedFile();
            //laver en liste over filer i dir'et
            File[] findes = fc.getCurrentDirectory().listFiles();
            for(int i=0;i<findes.length;i++)
            {
                //hvis gemt fil = eksisterende fil -> vis Advarsel
                if (findes[i].equals(f))
                {
                    JOptionPane.showMessageDialog(null,                     70
```

```
                                    "Der findes en fil med dette navn",
                                    "Vælg et andet filnavn!", JOptionPane.WARNING_MESSAGE);
                                hentFilNavn(extension);
                                 return f;
                        }
                }
                if (extension.equals(".icp"))
                        disposeMain = true;
                return f;
        }
        else//ved cancel
        {
        Object[] options = { "Ok", "Cancel" };
        int brugerVaelg= JOptionPane.showOptionDialog(null,
        "Indskriv et filnavn for at starte måling", "Warning",
        JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE,
        null, options, options[0]);//Besked, hvis der trykket på cancel
        if (brugerVaelg ==0)//hvis ok -> hentes en filechooser
        {
                hentFilNavn(extension);
                return f;
        }
        else
        {
                f=null;
                return f;
        }
        }
}
/*Returnerer det navn ICP datafilen gemmes under*/
public static String icpFilNavn()
{
        return f.toString();
}
}
```

Patient.java

```
package DIKTI.Entity;
import DIKTI.Boundary.*;
import DIKTI.Control.*;
import java.io.*;
import java.util.*;
import java.awt.BorderLayout;
import javax.swing.*;
import com.jrefinery.chart.*;
import com.jrefinery.data.*;
/*****************************************
** Håndterer patientdata fra og til filer
*****************************************/
public class Patient extends JFrame
{
```

```
/**************Atributter*****************************************/
    public static Vector kommentarerTilGraf;
    private Vector icpVektor;
    private File fil;
    private int getext;
    private double [] icpArray;
    private String filename;
    public File filen;
/**************Metoder*******************************************/
/*kommentarTilFil*/
    public static void kommentarTilFil(Vector kommentarTilFil, String pathToFile)
    {
        try
            {
                FileOutputStream kommentarFil = new FileOutputStream(pathToFile);
                PrintWriter pw = new PrintWriter(kommentarFil);
                for (int i=0; i<kommentarTilFil.size();i++)
                    {//skriver bogstaverne fra kommentaren til fil
                        pw.println("" +kommentarTilFil.elementAt(i));
                    }
                pw.close();//lukker for skrivning til fil
            }
        catch (FileNotFoundException filen)
            {
                JOptionPane.showMessageDialog(null,
                                        "Der kan ikke skrives til filen" + pathToFile,
                                        "ADVARSEL", JOptionPane.WARNING_MESSAGE);
            }
    }
/*Læser kommentarer fra fil og returnerer en vector*/
    public static Vector LaesKommentarFil(String pathToFile)
    {
        String line = null;
        BufferedReader kommentarInd = null;
        try
            {
                kommentarerTilGraf = new Vector();
                kommentarInd = new BufferedReader(new FileReader(pathToFile));
                line = kommentarInd.readLine();//læser fra fil
                kommentarerTilGraf.add(line);//tilføjer til graf
                while (line != null)//sidste linie er tom - giver ellers nullpointer
                {
                    line = kommentarInd.readLine();//læser fra fil
                    kommentarerTilGraf.add(line);//tilføjer til graf
                }
                kommentarInd.close();//lukker fillæseren
            }catch (FileNotFoundException f){}
            catch(IOException i){}
        return kommentarerTilGraf;
    }
/*******Henter ICP-datafilen*******/
    public boolean filSoegning(String extension)
```

```
    {
        JFileChooser fc = new JFileChooser();//Laver en filechooser
        Filhaandtering filter = new Filhaandtering();
        filter.addExtension(extension);//filføjer filextension til filter                    70
        filter.setDescription("datafiler");//beskrivelse af extension
        fc.setFileFilter(filter);//tilføjer filter til filechooser
        int returnVal = fc.showOpenDialog(null); //åbningsdialogboks
        if (returnVal == JFileChooser.APPROVE_OPTION) //Hvis der vælges en fil indlæses den
            {
                filen = fc.getSelectedFile();//henter sti og navn på valgt fil(File)
                filename = filen.getAbsolutePath();//giver sti og navn på fil(String)
                getext = filename.lastIndexOf(".");//finder sidste '.' i stien
                if(filename.substring(getext+1).equals(extension))//sammenligner fil-endelser
                    return true;                                                             80
                else
                    {
                        JOptionPane.showMessageDialog(null,
                                        "Vælg korrekt filformat (.inf)",
                                        "Forkert filformat", JOptionPane.INFORMATION_MESSAGE);
                        return false;
                    }
            }
        else
            {                                                                                90
                JOptionPane.showMessageDialog(null,
                                        "Vælg venligst en datafil for at starte",
                                        "Vælg en fil!", JOptionPane.INFORMATION_MESSAGE);
                return false;
            }
    }
/*Metode, der læser fra fil og sætter i et array*/
    public void filTilArray()
    {
        /*Læser fil ind i vektor*/                                                         100
        icpVektor = new Vector();
        if (filename!=null)
            {
                fil =  new File(filename);
                /*Læser fra filen*/
                try
                    {
                        BufferedReader filIbuffer =  new BufferedReader(new FileReader(fil));
                        String thisLine;
                        while ((thisLine =  filIbuffer.readLine()) != null)                  110
                            {
                                if ((thisLine.length()) > 0)
                                    {
                                        StringTokenizer ftk =  new StringTokenizer(thisLine);
                                        String tokenString =  ftk.nextToken();
                                        icpVektor.addElement(tokenString);
                                    }
                            }
```

```
            }catch (IOException e){JOptionPane.showMessageDialog(null,
                                    "Der kan ikke læses fra filen: " + e,
                                    "ADVARSEL", JOptionPane.WARNING_MESSAGE);}

            /*Indlæser fil til array*/
            icpArray = new double [icpVektor.size()];
            for(int i=0; i<icpVektor.size() ; i++)
                {
                    //Cast af vector til string
                    String vectorToString = (String)(icpVektor.elementAt(i));
                    //så string kan parses til integer
                    icpArray[i] = Double.parseDouble(vectorToString);
                }
        }
    }
    public double [] icp()
    {
        return icpArray;
    }
    public String icpFilNavnHent()
    {
        return filename;
    }
}
```

LITTERATUR

[1] Patientregistreringssystem på Neurokirurgisk afd., Aalborg Sygehus Syd.

[2] *Preben Sørensen, overlæge, neurokirurgisk afdeling K, Aalborg Sygehus Syd*, mundtlig kilde - sep-dec. 2002 og feb-maj 2003.

[3] Diagnosticering af Normaltryks-hydrocephalus ved Intrakraniel Trykmåling. *Boilesen A, Jensen R.H, Rask T, Vistisen J.* Aalborg Universitet, 5. sem, Sundhedsteknologi 2002.

[4] Cusimano MD A.O. Hebb. Idiopathic normal pressure hydrocephalus: a systemic review of diagnosis and outcome. *J Neurosurg*, Nov;49(5):1166-84, 2001.

[5] Rachid Bech-Azeddine. The lumbar infusion test in the diagnotic evaluation of adult patients with hydrocephalus. *Phd thesis, The Liquordynamic Laboratory University Clinic of Neurosurgery The Neuroscience Centre Rigshospitalet, Copenhagen, Denmark*, 2002.

[6] Suresh R. Devasahayam. *Signals and systems in biomedical engineering: Signal processing and physiologial systems modeling.* Kluwer Academic / Plenum Publishers, 2000. ISBN 0-306-46391-1.

[7] A. Eklund et. al. Two computerized methods used to analyse intracranial pressure b waves: comparison with traditionel visual interpretation. *J Neurosurg*, vol. 94; 392-396, 2001.

[8] A. H. Kaye et. al. *Operative Neurosurgery.* Churchill Livingstone, 1999. ISBN 044305827X.

[9] C. Raftopoulos et. al. Morphological quantitative analysis of intracranial pressure waves in normal pressure hydrocephalus. *Neurological Research*, 'Dec. vol. 14; 389-396, 1992.

[10] J.J. Lemaire et. al. Slow preessure waves in the cranial enclosure. *Acta Neurochirurgica*, 144: 243-254, 2002.

[11] J.K. Krauss et. al. The relation of intracranial pressure b-waves to different sleep stages in patients with suspected normal pressure hydrocephalus. *Acta Neurochir (Wien)*, 1995.

[12] M. Walter et. al. Online analysis of intracranial pressure waves. *Acta Neurochir*, (Suppl)81: 161-162, 2002.

[13] Mattias H.Morgalla et. al. A computer-based method for continuous single pulse analysis of intracranial pressuer waves. *Journal of the Neurological Science*, 168: 90-95, 1999.

[14] O.B. Paulsen et. al. *Klinisk Neurologi og Neurokirurgi.* FADL, 3. udg. 1997. ISBN 87-7749-111-4.

[15] S. Biering-Sørensen et. al. *Struktureret Program Udvikling.* Ingeniøren|bøger, 2000. ISBN 87-571-1046-8.

[16] S.E. Børgesen et. al. Computerized infusion test compared to steady pressure constant infusion test in measurement of resistance to csf outflow. *Acta Neurochir*, 119:12-16, 1992.

[17] Kent M. Van De Graff. *Human Anatomy.* McGraw-Hill, 2002. ISBN 0-07-112283-4.

[18] M.A. Musen J.H. van Bemmel. *Handbook of Medical Informatics.* Springer, 1997. ISBN 3-450-63351-0.

[19] Niels Lundberg. Continuous recording and control of ventricular fluid pressure in neurosurgical practice. *Acta psychiatricia et neurologica Scandinavica (suppl)*, vol. 36; 149;1-193, 1960.

[20] Shizuo Oi. Hydrocephalus chronology in adults: confused state of the terminology. *Crit Rev Neurosurg*, Vol. 8; Issue 6; 346-356, 1998.

[21] Eriksson P. *UML Toolkit.* John Wiley and Sons, 1998. ISBN 0-471-19161-2.

[22] I.K Pople. Hydrocephalus and shunts: What the neurologist shold know. *J Neurol Neurosurg Psychiatry*, 73 (suppl 1): i17-i22, 2002.

[23] Roger S. Pressman. *Software Engineering - a practitioner's approach.* McGraw-Hill, 1992. ISBN 0-07-112779-8.

[24] Stefan Silbernagl and Agamemnon Despopoulos. *Color Atlas of Physiology 4th ed.* Thieme, 1991. ISBN 0-86577-382-3.

[25] J.A.L. Vanneste. Diagnosis and management of normal-pressure hydrocephalus. *J Neurol*, 247: 5-14, 2000.

[26] *Intracranial pressure (ICP) and cerebrospinal fluid (CSF) dynamics.* http://www.-panarabneurosurgery.org/Journal/Vol4_2/ICP_and_CSF/JournalVol4_2_ICP_and-_CSF.htm, 3. nov 2002.

[27] *Bekendtgørelse om medicinsk udstyr, BEK nr 105 af 27/02/2002 (Gældende).* http://www.medicoindustrien.dk/Library/Pdf/bekendtg_105_270202.pdf, maj 2003.

[28] *Borland hjemmeside for beskrivelse af programmet Together.* http://www.togethersoft.com/, 20 maj 2003.

[29] *DGM Dansk Godkendelse af Medicinsk Udstyr.* http://www.dgm-nb.dk, maj 2003.

[30] *Integra LiveScience.* http://www.integra-ls.com/index_java.html, 20. maj 2003.

[31] *Intracranial pressure transducers.* http://www.rmpd.org.uk/research/critical_care-_physics/Intracranial_pressure_and_clinical_status.htm, 21. maj 2003.

[32] *IT-sikkerhedsvejledning for sygehuse.* http://www.sst.dk/publ/Publ2002/IT_sikkh_sgh-_korr.pdf, 20. maj 2003.

[33] *Java(tm) Communications API Users Guide.* http://java.sun.com/products/javacomm/-javadocs/API_users_guide.html, marts 2003.

[34] *JFreeChart.* http://www.jfree.org/jfreechart/index.html, marts 2003.

[35] *A really friendly guide to wavelets.* http://perso.wanadoo.fr/polyvalens/clemens/wavelets/-wavelets.html, 7. mar 2003.

[36] *Regler - Medicinsk udstyr.* http://www.dgm-nb.dk/regler_medicinsk/, maj 2003.

[37] *Videnskabsetisk komité.* http://www.nja.dk/Serviceomraader/SundhedOgSygehuse/-VidenskabsetiskKomite/VidenskabsetiskKomite.htm, maj 2003.