Systemdesign af LabInfo

Kalle Christensen
s960453

Kgs. Lyngby 2004
IMM-THESIS-2004-14

Systemdesign af LabInfo

Kalle Christensen – s960453

Kgs. Lyngby 2004-29-02

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-THESIS: ISSN 1601-233X

Systemdesign af LabInfo

Eksamensprojekt

Informatik og Matematisk Modellering, IMM

Danmarks Tekniske Universitet, DTU

Udarbejdet af Kalle Christensen – s960453

Vejleder: Tom Østerby

IMM-Thesis-2004-14

29. februar 2004

Forord

Dette eksamensprojekt med titlen "Systemdesign af LabInfo" har været mit afsluttende projekt indenfor civilingeniøruddannelsen på DTU. Projektet er udarbejdet i perioden fra d. 1. september 2003 til d. 29. februar 2004 og svarer til 30 ETCS point. Projektet er skrevet på instituttet for Informatik og Matematisk Modellering, IMM, på Danmarks Tekniske Universitet, DTU.

Docent Tom Østerby, IMM, har været min vejleder på projektet.

Jeg vil benytte lejligheden til at takke Tom Østerby for vejledning op til og under projektet.

Samtidig vil jeg takke Jens Mikkelsen, EDB Gruppen A/S, for at have stillet systemet LabInfo til min rådighed.

Lyngby, DTU, 29. februar 2004

_____

Kalle Christensen

Abstrakt

Dette eksamensprojekt fokuserer på at forbedre systemdesignet for et eksisterende informationssystem, LabInfo systemet. Informationen og kendskabet til LabInfo systemet er begrænset, hvorfor denne viden først må indsamles. Dette gøres ved hjælp af de første aktiviteter indenfor programmelkonstruktion.

For systemet gennemføres en programanalyse og en programspecifikation for at præcisere det eksisterende systemdesign. På baggrund heraf træffes beslutningen om et nyt og forbedret systemdesign.

Ideen bag det nye systemdesign er at lægge systemets arkitektur om til en "four-tier client-server" model. Dermed er der skabt mulighed for en forbedret kommunikation mellem systemets delsystemer.

Der er samtidig skabt bedre overblik af det nye system i forbindelse med udfærdigelsen af use case diagrammer, kravspecifikationer og sekvensdiagrammer.

Abstract in english

This M. Sc. Thesis is focused on improving the system design for an existing system, LabInfo. The information and the general knowledge of LabInfo are very limited and must therefore be obtained. This is done through the first activities within the process of software engineering.

A program analysis and program specification is made in order to specify the existing system design. Using that as a background the new and improved system design is made.

The idea behind the new system design is to chance the systems architecture to a four-tier client-server model. This way it is made possible to improve the communication between the subsystems.

Furthermore an improved overview of the system is given by use case diagrams, requirements specification and sequence diagrams.

1 Problembeskrivelse

LabInfo systemet formidler laboratorieundersøgelser indenfor en række laboratoriespecialer. Disse laboratorieundersøgelser kategoriseres efter IFCC-IUPAC kodesystemet. Til en del af disse undersøgelser knyttes der i Danmark yderligere uddybende information, kompendium information. Denne information bliver fastlagt hos Sundhedsstyrelsen, som er det centrale system. Det står frit for landets hospitaler, at føre en lokal og modificeret version af kompendium informationen. Derved opstår der decentrale systemer.

Projektet sker i samarbejde med virksomheden Edb Gruppen A/S i Ballerup.

Projektet har to formål:

- Gennemføre en programmelanalyse for det centrale LabInfo system.
- Ud fra ovenstående at skabe bedre kommunikation mellem det centrale og de decentrale systemer.

Målet er at få samlet nok viden omkring LabInfo systemet til at kunne redegøre for dets fremtidige systemdesign. Denne viden forventes opnået ved gennemførelse af ovenstående formål med vægt på programmelanalysen.

Indledning

2 Indledning

2.1 LabInfo Systemet

Sundhedsstyrelsen dannede i 1998 styregruppen "Kontor for laboratorie-informatik". Dens opgave var at sikre udvikling og implementering af et internationalt kodeskema IFCC-IUPAC[1]. Kodeskemaet anvendes ved kommunikation af laboratorierekvisitioner og svar på analyser inden for områderne: klinisk biokemi og mikrobiologi. Styregruppen udviklede i samarbejde med konsulent firmaer programmet LabInfo til behandling og formidling af kodeskemaet.

LabInfo blev udvidet til at omfatte mere specialiseret information omkring: allergologi, thrombose og hæmostase, genetik, farmakologi, immunologi og reproduktion og fertilitet. Til hvert af disse fagområder eksisterer der en laboratoriemedicinsk specialforening. Information, der opsamles fra laboratorieundersøgelser i Danmark, samles under et til Kompendium 2000.

Sundhedsstyrelsen er ansvarlig for videreudvikling og vedligeholdelse af det nationale kodesystem. De laboratoriemedicinske specialforeninger varetager informationen bag Kompendium 2000.

2.2 LabInfo tages i brug

Offentliggørelsen af første version af LabInfo medførte nye krav fra brugerne. Kravene omfattede nye funktioner og modificering af eksisterende.

Århus Universitets Hospital[2] dannede på baggrund af LabInfo et selvstændigt system LabInfoAUH i år 2001. LabInfoAUH blev oprettet med henblik på at dække AUH´s ekstraordinære behov indenfor Kompendium 2000.

LabInfoAUH blevet dannet ved at modificere en version af det daværende LabInfo system. Unødvendig funktionalitet for AUH, samt funktionalitet forbeholdt Sundhedsstyrelsen, blev ikke aktiveret i LabInfoAUH. Dette var dels AUH´s eget ønske samt et krav fra Sundhedsstyrelsens side.

Ansvaret for informationen bag IFCC-IUPAC kodeskemaet er alene pålagt Sundhedsstyrelsen. Funktioner i LabInfo, som varetager kodeskemaet, må af den grund kun være tilgængelig for Sundhedsstyrelsen.

Landets hospitaler kan have lokal information, som afviger fra LabInfo´s Kompendium 2000. AUH udviklede en funktionalitet til varetagelse af

[1] International Federation of Clinical Chemistry and Laboratory Medicine - International Union of Pure and Applied Chemistry

[2] Århus Universitets Hospital vil fremover blive betegnet som AUH.

sådanne områdespecifik information for Kompendium 2000 under LabInfoAUH. LabInfoAUH fungerer derfor som et supplement til LabInfo.

2.3 Fremtidens LabInfo

Når LabInfoAUH officielt tages i brug formodes det at skabe en øget interesse for LabInfo systemet hos landets øvrige hospitaler. Interessen forventes drevet af et ønske om ligeledes at få dækket deres individuelle informationsbehov inden for laboratorieinformation.

Bliver dette en realitet, er det ikke hensigtsmæssigt at fortsætte den videre distribuering på baggrund af LabInfo eller LabInfoAUH.

Årsagen er, at LabInfo ikke er designet med henblik på at håndtere områder som deling af information mellem flere systemer.

EDB Gruppen har derfor interesse i at få fastlagt funktionaliteten af LabInfo systemet. For derved at kunne vurdere behovet og metoden til en øget kommunikation mellem det centrale og de decentrale systemer.

Dette projekt gennemfører en programmel analyse af det centrale LabInfo system. Derved bliver systemets funktionalitet præciseret og dokumenteret.

På baggrund af analysen vil det være muligt at vurdere og opstille nødvendige krav til de decentrale systemer. Efterfølgende vil det være muligt at opstille velbegrundende metoder for den videre kommunikation mellem de enkelte systemer. Resultatet vil være et nyt og forbedret systemdesign for LabInfo systemet.

Denne rapport giver i efterfølgende afsnit en kort gennemgang af procesforløbet i forbindelse med programmelkonstruktion. På baggrund af denne teori gennemføres i de efterfølgende afsnit den egentlige analyse af LabInfo systemet. Analysen opstilles for både LabInfo og LabInfoAUH. Afsnit 12 summerer op på analyse og giver en vurdering af de to systemer. Efterfølgende præsenteres en forbedret udgave af systemarkitekturen for LabInfo systemet. Endelig gøres der i afsnit 14 rede for den nødvendige kommunikation mellem systemerne.

Kapitel 1

3 Baggrund

Nedenfor beskrives disciplinen programmelkonstruktion og programmelproces ifølge Sommerville (2001). Disciplinerne omfatter teorier, metoder og værktøjer til udvikling af et professionelt programmel.

3.1 Programmelkonstruktion

Der eksisterer ikke noget ideelt procesforløb for programmelkonstruktion. Hver proces skal i teorien præciseres individuelt med øje for formålet med konstruktionen.

Mange firmaer har i dag defineret deres egne procesforløb for programmelkonstruktion, som er tilpasset firmaets øvrige standarder. Det er med til at sikre et bedre overblik og samarbejde med udvikling eller vedligeholdelse af firmaets programmer.

Selvom procesforløbet kan variere er der aktiviteter, som vil forblive obligatoriske for programmelprocesser. Disse er ifølge Sommerville (2001):

- *Programspecifikation* – Definering af programmets funktionalitet og indskrænkelse af dets operationer.
- *Programdesign og implementering* – Producerer programmet til at opfylde programspecifikationerne.
- *Programvalidering* – Programmet skal valideres for at bekræfte, at kundens krav er opfyldt.
- *Programevolution* – Programmet skal fortsætte sin udvikling for at imødekomme nye ønsker fra kundens side.

Ovenstående aktiviteter er karakteristiske for professionelle programmelkonstruktioner. Aktiviteterne bliver gennemgået enkeltvis i næste afsnit.

3.2 Programmelproces

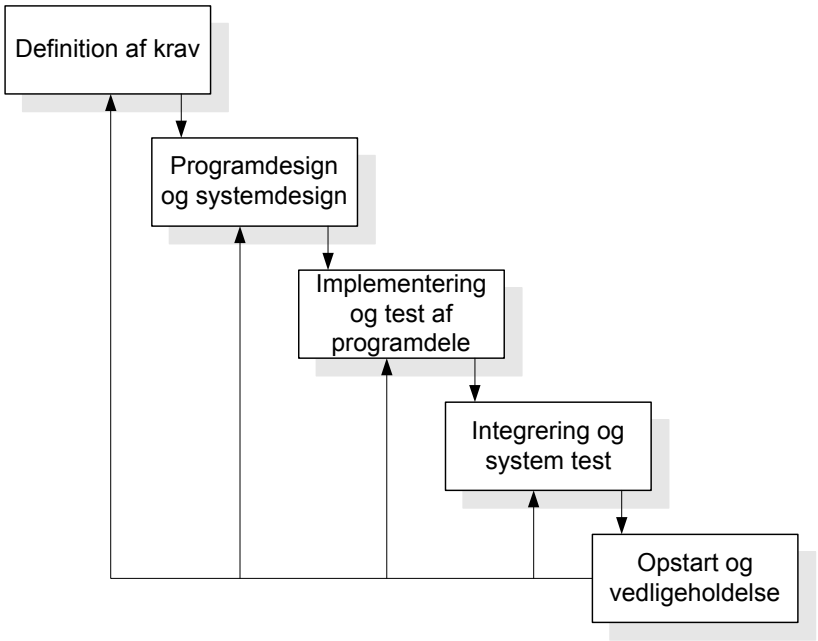
Ved programmelkonstruktion er det fordelagtigt at følge en bestemt programmelproces. Alle involverede personer kan således lettere følge med i programmets udvikling. Det er med til at øge kommunikationsniveauet mellem projektets personer og grupper.

Figur 1 viser en programmelproces. Figuren fokuserer på de overordnede aktiviteter i forbindelse med udvikling af et programmel.

Aktiviteterne i figur 1 dækker over følgende:

- *Definition af krav* – Ud fra programmets funktionalitet og formål laves en analyse for at konkretisere og definere programmets krav. Herved fastlægges programmets specifikation.
- *Programdesign og systemdesign* – Processen for systemdesign opdeler kravene til enten at være maskinel eller programmel. Derved dannes systemets overordnede arkitektur. Programdesign omfatter identificering og beskrivelse af de fundamentale programdele og deres indbyrdes relation.
- *Implementering og test af programdele* – Realisering af programdesignet for et eller flere programdele. Via test undersøges om programdelene lever op til deres specifikationer.
- *Integrering og systemtest* – Individuelle programdele integreres til et komplet system. Systemet testes for at sikre, at programmets krav er opfyldt. Efter endt test kan programmet leveres til kunden.
- *Opstart og vedligeholdelse* – Systemet installeres og tages i brug. Vedligeholdelse omfatter udbedring af fejl, der ikke blev fundet ved de forudgående test. Derudover skal systemet løbende forbedres til at håndtere nye funktioner, efterhånden som de opstår.

Ovenstående aktiviteterne betragtes som selvstændige. Hver aktivitet skal afsluttes med et tilfredsstillende resultat før påbegyndelsen af næste.



Figur 1: "Waterfall" model. Viser udviklingsforløbet for et programmel. [Sommerville, side 45].

Informationen fra forudgående aktivitet(er) bruges i det efterfølgende. Følgefejl kan derfor let opstå, hvis ikke den eller de forudgående aktiviteter er udført korrekt. Dette er tydeligvis en svaghed for modellen.

Modsat er den skarpe opdeling af procesforløbet fordelagtigt med henblik på budgettering og tidsplanlægning. Hver aktivitet kan enkeltvis tildeles et budget og en tidsramme med en endelig deadline. Ved deadline skal den pågældende aktivitet være afsluttet således, at det videre forløb kan påbegyndes.

I tilfælde af en følgefejl er det nødvendigt at vende tilbage til aktiviteten, hvor fejlen kan rettes. Denne aktivitet laves helt eller delvis om for at rette fejlen eller manglen. Procesforløbet fortsætter derefter fra denne aktivitet.

I dag eksisterer flere forskellige programværktøjer, der kan håndtere forskellige programmelprocesser. Sådanne programmer betegnes CASE af det engelske betegnelse: "Computer-Aided Software Engineering". Brug af disse CASE værktøjer er med til at forbedre planlægningen og dokumentationen af det ønskede programmel.

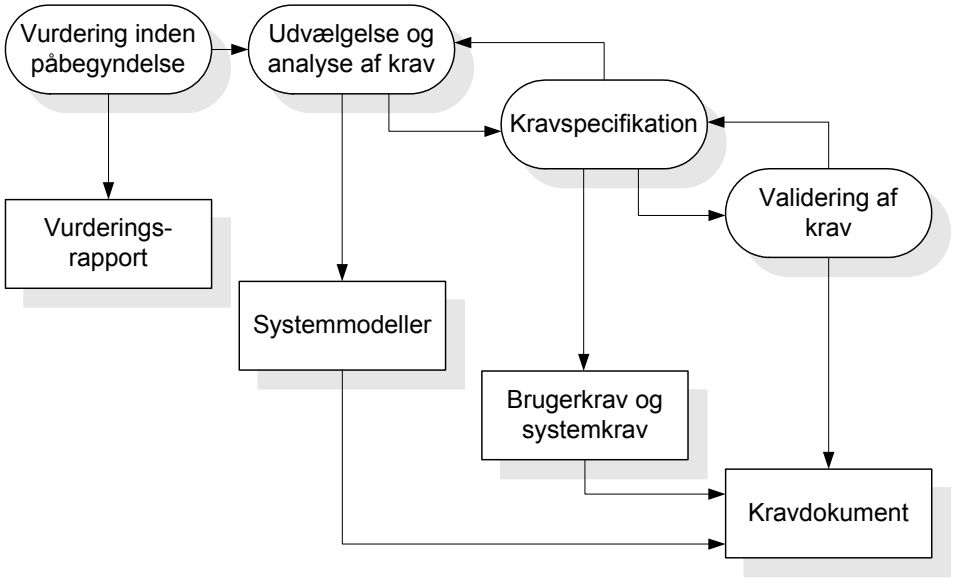
3.2.1 Programspecifikation

Formålet med denne aktivitet er at få specificeret det ønskede programmel. Det sker ved at undersøge den funktionalitet som forventes, at det færdige program skal udføre. En funktionalitet beskrives bedst ved hjælp af en mængde krav, som er forbundet med den pågældende funktionalitet.

Kravene konkretiseres ved hjælp af en analyse og en specificering. Dette resulterer i et dokument med de krav, som er fundamentet for det videre procesforløb. Der er derfor en stor interesse i at få dokumenteret kravene bedst muligt. Det har ført til udvikling af metoder og teknikker udelukkende til konstruering af kravdokumenter. Resultatet heraf er eksempelvis dokumentet "IEEE[3] Recommended Practice for Software Requirements Specifications". Dokumentet præsenterer en standard for dokumentationsførelse af programspecifikation.

Kravdokumentet består typisk af tre dele. En introduktion der giver indblik i programmets formål og domæne. En overordnet beskrivelse af kravene henvendt til slutbrugeren. Samt en detaljeret specificering af kravene henvendt til programmelkonstruktører. Dette medfører en minimering af misforståelser mellem slutbrugere og programmelkonstruktører omkring programmets funktionalitet.

[3] Institute of Electrical and Electronics Engineers



Figur 2: Proces for programspecifikation / kravkonstruktion. [Sommerville 2001 side 55].

Programspecifikation er koncentreret omkring bearbejdningen af det pågældende programs krav. Af den grund benævnes programspecifikation ofte som kravkonstruktion. Figur 2 giver et billede af forløbet for kravkonstruktion. Figuren viser fire faser og deres opnået produkt. Faserne er:

- *Vurdering inden påbegyndelse* – Der foretages et skøn på om de nødvendige teknologier og ressourcer er tilgængelig for udførelsen af det ønskede programmel.
- *Udvælgelse og analyse af krav* – Opstilling og udvælgelse af krav for systemet. Dette sker i tæt samarbejde med eventuelle slutbrugere. Eventuelle systemmodeller kan udvikles med henblik på at skabe bedre forståelse for uklare funktioner.
- *Kravspecifikation* – Informationen fra analysen brugs til udarbejdning af et dokument med specifikation over systemets krav. Dokumentationen opdeles i to typer. Brugerdokumentation med abstrakt notation af systemets funktionalitet. Samt systemdokumentation med en detaljeret beskrivelse af systemet.
- *Validering af krav* – Det undersøges om systemets krav er komplet udført. Opdages der fejl i kravdokumentet skal disse rettes i denne fase.

Udbyttet af programspecifikationen er ifølge figur 2: Vurderingsrapport, systemmodeller, brugerkrav og systemkrav samt et afsluttende kravdokument.

I de efterfølende afsnit gives et mere nuanceret billede af processen bag programmelspecifikationen.

Vurdering inden påbegyndelse

Når et selskab, der er specialiseret i programudvikling, får en henvendelse fra en kunde om at gennemføre udviklingen af et programmel. Da vil selskabet som en naturlig reaktion foretage en vurdering af opgaven, inden der tages fat på den egentlige løsning og modellering.

Det kan være svært at afgøre, hvor grundig vurderingen skal være. Ofte er vurderingen begrænset til en tidsperiode. Ved afslutningen skal selskabet præsentere resultatet til kunden, der ofte har haft flere selskaber til at vurdere opgaven.

Vurderingen går på om selskabet skal tage opgaven til sig eller opgive den. I begge tilfælde er det vigtigt at selskabet har dokumenteret udfaldet. Af dokumentet skal baggrunden for afgørelsen klart fremgå. Afgørelsen træffes ved at selskabet besvarer centrale spørgsmål som:

- Er virksomheden i besiddelse af de kvalifikationer og resurser, som opgaven kræver for at kunne blive gennemført?
- Er et muligt at gennemføre opgaven med nuværende teknologi?
- Kan det færdige programmel integreres med andre eksisterende programmer?

Er en virksomhed ikke kvalificeret til en given opgave, må opgaven betragtes som værdiløs for virksomheden. Til tider nedprioriterer enkelte virksomheder dette kritiske punkt. Eksempelvis kan en virksomhed, i forbindelse med en udvidelse, satse på at opnå de nødvendige kvalifikationer.

Hvis ikke der eksisterer nok information omkring opgaven, kan det være vanskeligt at vurdere om virksomheden dækker det teknologiske niveau for opgaven. For at indsamle supplerende viden kan det være nødvendigt at foretage undersøgelser i de omgivelser, som systemet skal fungere i. Derved er det muligt at danne et mere nuanceret billede af den nødvendige teknologi for systemet. Afgørelsen for om virksomheden skal påtage sig opgaven kan dermed træffes på et mere sikkert grundlag.

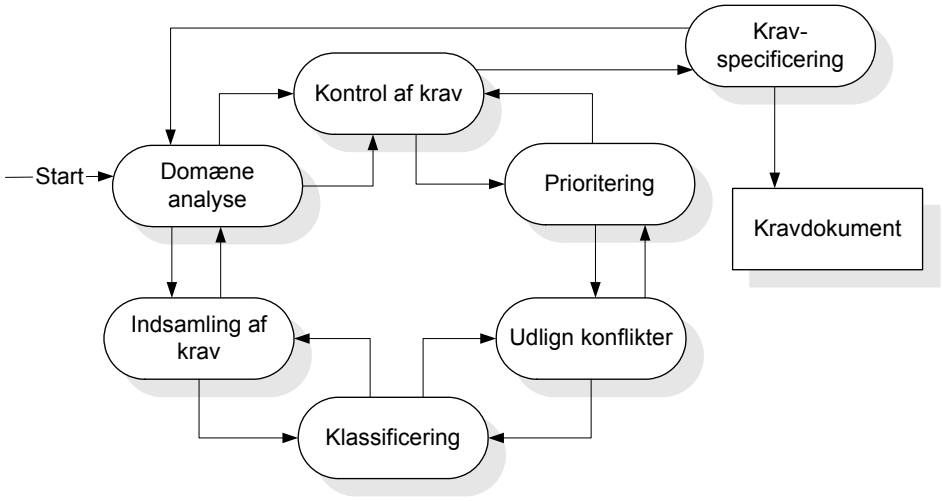
Udvælgelse og analyse af krav

Næste trin er at skabe forståelse for opgavens totale omfang og indhold. Det sker ved et tæt samarbejde mellem alle de personer, der er berørt af systemet. Typisk omfatter det programmelkonstruktører, kunden samt slutbrugerne for systemet. Under et kaldet stakeholders. Disse personer danner tilsammen en projektgruppe. Hvert fagområde i projektgruppen har hver deres forståelse og indblik i systemet. Formålet med denne proces er, at give disse personer samme forståelse for systemet.

Det kan være en vanskelig opgave. Dels fordi der skal opnås enighed omkring, hvad der er ønsket, og hvad der er teknisk eller realistisk ansvarligt. Dels vil hver gruppe have deres eget sprog til at beskrive systemet med.

Problemet løses bedst ved i samarbejde at opstille en række krav for systemet. Programmelkonstruktørerne foretager derefter en udvælgelse af relevante krav, der kan beskrive systemet. For disse krav foretages der en analyse. Analysen er med til yderligere at præcisere systemet overfor både kunden og programmelkonstruktøren. Analysen skal via et fælles sprog beskrive det samlede system med al dets funktionalitet. Modeller og figurer er meget anvendelig i den sammenhæng.

At opnå enighed omkring systemets krav kan være vanskelig og kræver ofte en revurdering af kravene. Et eksempel på et procesforløb for "Udvælgelse og analyse af krav" er opstillet i figur 3.



Figur 3: Proces for udvælgelse og analyse af krav. [Sommerville 2001 side 125].

Forløbet er inddelt i seks aktiviteter startende med en analyse af domænet. Derefter forløber processen mod uret.

1. *Domæne analyse* – Analytikere skal skabe forståelse for systemets domæne.
2. *Indsamling af krav* – Projektgruppen samarbejder om at finde systemets krav. Som et resultat øges forståelsen for domænet.
3. *Klassificering* – De indsamlede krav organiseres i relevante grupperinger.
4. *Udlign konflikter* – Når der indsamles krav fra flere forskellige grupper, vil der uundgåeligt opstå konflikter mellem kravene. Disse skal udlignes i denne aktivitet.

5. *Prioritering* – Projektgruppen vurderer kravene op mod hinanden for at få en prioritering af deres vigtighed.
6. *Kontrol af krav* – Kravene kontrolleres for at de stemmer overens med projektgruppens faktiske ønske af systemet.

Det fremgår af figur 3, at det fra alle aktiviteter er muligt at vende tilbage til den foregående.

Der eksisterer mange forskellige metoder til indsamling af krav. For hvert projekt er det nødvendigt at vurdere hvilken eller hvilke metoder, der er bedst anvendelige. Ofte er den bedste løsning en kombination af flere metoder.

Der vil ikke blive gennemgået nogen specifik metoder her i rapporten. Der henvises til Sommerville (2001).

Kravspecifikation

En udbredt måde at anskueliggøre krav for et system, er ved hjælp af UML[4]. Årsagen er, at notationen for UML henvender sig til alle stakeholders, der er involveret i projektet. Ved hjælp af UML er det muligt at opstille generelle og simple modeller for systemet. Men samtidig kan der også udarbejdes detaljeret modeller.

I forbindelse med kravspecificering bruges ifølge Bennett (1999) en eller flere typer UML modeller. Disse er beskrevet kort nedenfor.

Use case diagram – Disse er illustrationer af den funktionalitet programmet vil udføre. Det vil sige, den måde brugerne vil kommunikere med systemet på for at udføre den enkelte funktion. Diagrammerne kan være abstrakte eller meget detaljeret illustrationer. Diagrammerne opdeles i brugerniveauer.

Informationsgrundlaget for "Use case diagrams" opstår i forbindelsen med indsamling af systemkrav. På grund af deres overordnede informations niveau, er det muligt at opstillede dem tidligt i forløbet. Udover at opstille diagrammerne skal der laves use case beskrivelser. Her dokumenteres handlingsforløbet for de enkelte use cases med hensyn til aktøren og systemet. På den måde bliver kravene set og beskrevet i en helhed, som hjælper til at få dem præciseret. Samtidig medvirker diagrammerne og deres beskrivelser til forståelsen og overblikket for de næste diagrammer.

Eksempelvis er de til stor hjælp ved udarbejdningen af et klassediagram. Via use case beskrivelserne opstilles de nødvendige klasser. Hver klasse består af attributter og operationer. Samtidig kan der for hver klasse være knyttet flere funktionelle som ikke funktionelle krav. Disse beskrives eksempelvis ved hjælp af OCL[5]. Målet med klassediagrammet er at illustrere, hvilke klasser der skal kommunikere med hinanden. Mens beskrivelserne i OCL giver en uddybet billede af den enkelte klasse.

[4] Unifeid modelling language.
[5] Object Constraint Language.

Den næste type er "Interaktionsdiagram", der udarbejdes for at skabe forståelse for systemets funktionalitet inden for de enkelte brugerniveauer. Med disse diagrammer kan analysen bevæge sig ned på et lavere og mere teknisk plan i systemets design. "Use case diagrams" og deres beskrivelse tages i brug for at opnå den nødvendige viden. Det er muligt at opstille to typer af diagrammer:

Tilstandsdiagram – Viser interaktionen af beskeder mellem objekter for den pågældende funktionalitet. Interaktionen angives i forhold til en tidslinje. Rækkefølgen for kald af objekter bliver dermed fastlagt.

Collaboration diagram – Fokusere ligesom "tilstandsdiagram" på objekterne, men indeholder ikke nogen tidslinje. Der lægges her vægt på, hvordan de enkelte objekter eller klasser er forbundet til hinanden inden for den enkelte funktionalitet.

Begge diagrammer kan oprettes til at være mere eller mindre detaljerede.

På baggrund af "interaktionsdiagrammerne" er det muligt at knytte yderlige information til systemets funktionalitet. Det gøres via to former for "behaviour diagrams", som fokuserer på de enkelte objekter:

Statechart diagram – Beskriver de tilstande et objekt kan befinde sig i. Hver tilstand kan være forbundet med en betingelse, som skal være opfyldt før objektet kan bringes til en ny tilstand. En betingelse kan være at udføre en handling eller at vente på et kald udefra. Et objekt venter i en tilstand i et endeligt tidsrum. Til hvert objekt er knyttet en start og en slut tilstand.

Activity diagram – En variant til "statecharts". Koncentrerer sig omkring det interne aktivitets flow, der er forbundet med det enkelte objekt. Bruges ofte til at beskrive en "use case" mere detaljeret.

Den sidste slags diagrammer bruges i typisk i forbindelse med implementering og kaldes af den grund for "implementeringsdiagram".

Component diagram – Illustrerer relationen mellem eventuelle komponenter/filer i systemet.

Deployment diagram – Bruges til at illustrerer hvordan individuelle komponenter distribueres mellem flere systemer og deres indbyrdes interaktion.

Validering af krav

Denne fase har til formål at undersøge om kravene som en helhed beskriver systemet bestilt af kunden. Det er vigtigt ikke at fortsætte procesforløbet før valideringen er acceptabel. Valideringen er den sidste fase til at opfange problemer med kravene. Efterfølgende vil kravene blive brugt som udgangspunkt. En eventuel ændring eller tilføjelse af krav kan have

uoverkommelige konsekvenser. Generelt gælder at jo senere en fejl opdages, des vanskeligere er den at rette.

Med tiden er der blevet udviklet programværktøjer, der kan lette valideringen. Disse værktøjer er i stand til at kontrollere relationerne mellem systemets krav. Derved kan konflikter og mangler for kravene hurtigt bestemmes. Programmerne er imidlertid ikke i stand til at vurdere, om mængden af krav er tilstrækkelig for det pågældende system. Det forbliver en menneskelig vurdering.

Det vanskelige for denne fase er at vurdere, hvornår mængden af krav er tilstrækkelig til at beskrive det pågældende system.

Styring af krav

Fasen har til formål at styre processen i forbindelse med ændring af et systemkrav. Ændringer vil forekomme. Enten under systemets konstruktion eller efterfølgende, når systemet er taget i brug. At foretage ændringer i systemets krav kan som tidligere nævnt være en vanskelig proces. Derfor er det vigtigt så tidligt så muligt at planlægge en fast procedure for ændring af systemets krav. Med denne procedure bevares overblikket og strukturen for systemet og dets dokumentation.

Ved et ønske om en ændring bør følgende overvejes:

1. *Problemanalyse og specificering af ændring* – Der foretages en analyse af problemet. Erkendes det, at der eksisterer et problem, da udarbejdes en specificering for gennemførelse af ændringen.
2. *Analyse og budgettering af ændring* – Ud fra specificeringen gennemføres en analyse for konsekvensen af ændringen. Det vurderes om handlingen er økonomisk realistisk.
3. *Implementering af ændring* – Godkendes ændringen foretages den nødvendige implementering. Kravdokumentet skal ligeledes opdateres.

3.2.2 Programdesign og systemdesign

Programspecifikationen danner grundlag for de beslutninger, der træffes i denne aktivitet. Målet er at opstille og beskrive modeller for programmets struktur med henblik på den endelige implementering. Præciseringen af designet sker i trin, fordi designet typisk er omfattet af flere former for design. Dels kan programdesign og systemdesign opfattes som to forskellige typer. Programdesign omfatter den visuelle del, mens systemdesign er den bagvedliggende opstilling.

Specielt systemdesignet er interessant og opdeles typisk i nedenstående aktiviteter [Sommerville, 2001].

1. *Arkitekturmæssig design* – Delsystemer og deres relationer identificeres og dokumenteres.

2. *Abstrakt specifikation* – For hvert delsystem opstilles abstrakte specifikationer for deres funktion og de rammer, som de skal fungere under.
3. *Design af interface* – Til hvert delsystem specificeres og dokumenteres dets interface til øvrige delsystemer.
4. *Design af komponenter* – Servicer tillægges forskellige komponenter og deres interface specificeres.
5. *Design af datastruktur* – Datastrukturen, der bruges i implementeringen, opstilles og beskrives detaljeret.
6. *Design af algoritmer* – De nødvendige algoritmer specificeres og beskrives detaljeret.

Fremgangsmetoden er at få defineret de enkelte delsystemer og efterfølgende deres indbyrdes relationer. På det grundlag kan der arbejdes mere detaljeret med de enkelte delsystemer. Med kendskab til delsystemernes operationer kan de nødvendige interfaces lanceres. De afsluttende trin lægger meget tæt op ad aktiviteten om implementering og kan til tider med fordel opfattes som en del af denne aktivitet.

3.2.3 Implementering

Bruges ikke i rapporten. Der henvises til Sommerville (2001).

3.2.4 Program validering

Bruges ikke i rapporten. Der henvises til Sommerville (2001).

3.2.5 Program evolution

Bruges ikke i rapporten. Der henvises til Sommerville (2001).

Kapitel 2

4 Analyse af LabInfo systemet

Analysen foretages på baggrund af teorien for programmelkonstruktion præsenteret i afsnit 3. Nedenfor argumenteres for udfaldet af procesforløbet.

Der indledes med en vurdering af systemet, hvor årsagen til den ønskede analyse belyses. Efterfølgende vil den egentlige kravspecificering blive gennemgået og den nødvendige UML notation tages i brug.

4.1 Vurdering af LabInfo

EDB Gruppen overtog i 2002 alle aktiver fra en IT-virksomhed, her i blandt var programmet LabInfo. LabInfo var på det tidspunkt tæt på at blive færdigudviklet. Ved overdragelsen af programmet fulgte der ikke nævneværdig information med omkring programmet. Men programmet blev færdigudviklet med et tilfredsstillende resultat.

Sideløbende var der blevet skabt interesse omkring brugen af LabInfo systemet. Det decentrale system LabInfoAUH blev udviklet. Denne udvikling bar stærkt præg af den manglede dokumentation for LabInfo. Af økonomiske årsager og usikkerhed om programmets videre udbredelse blev der ikke gennemført nogen dokumentationsproces for LabInfoAUH eller LabInfo.

EDB Gruppen ønsker nu at komme på forhånd med en eventuel udbredelse af decentrale LabInfo systemer. Problemet med LabInfo og LabInfoAUH er følgende:

- Manglende systemdokumentation og brugerdokumentation.
- Ustruktureret kildekode, der vanskeliggør programmel-konstruktørernes overblik af programmet. Dette er med til at vanskeliggøre vedligeholdelsen af systemet.
- LabInfo er ikke udviklet med henblik på udbredelse af decentrale systemer. Det vanskeliggør kommunikationen mellem det centrale og de decentrale systemer.
- Manglende fleksibilitet i forbindelse med distribuering af systemet.

På det grundlag ønsker EDB Gruppen at foretage en analyse. Denne analyse skal være første led i at få skabt bedre forståelse og overblik over systemet. Ved hjælp af analysen gøres der forhåbninger om bedre at kunne imødekomme følgende punkter:

- Få styr på programmelstrukturen.
- Indføre objekt orienteret programmering, for derved at kunne tildele de decentrale systemer de nødvendige komponenter og kun disse.

- Få skabt systemdokumentation og brugerdokumentation for de enkelte programmer.
- Skabe bedre kommunikation mellem det centrale og de decentrale systemer. Det skal ske ud fra en overvejelse af mulige teknikker for gennemførelse af den nødvendige datakommunikation. Dermed forventes at undgå eller begrænse redundante data.
- Systemet skal være mest muligt fleksibelt med hensyn til det bagvedliggende operativsystem.
- Opnå økonomisk og tidsmæssig besparelse ved udvikling af nye decentrale systemer.

4.1.1 Informationsgrundlag

Fasen for "udvælgelse og analyse af krav" kan gøres på baggrund af eksisterende program og dokumentation. Mængden af dokumentationen er som tidligere nævnt meget begrænset. Men det tilgængelige og det eksisterende system er brugbart i forbindelse med udvælgelsen af systemets krav.

Mængden af dokumentation for LabInfo og LabInfoAUH er begrænset til at omfatte:

- Kildekode for begge systemer. Formatet er HTML, ASP, css[6].
- En SQL database for hvert system
- En programbeskrivelse af LabInfoAUH, der er henvendt til slutbrugerne, [LabInfoAUH, 2003].
- En beskrivelse af LabInfo og dets domæne, [LabInfo, 2003].

I appendiks afsnit 18.1 forligger en liste med en forklaring af essentielle termer inden for domænet.

[6] Cascading Style sheet er en fil, der indeholder definitioner for et specifikt design.

5 Kravspecifikation

Det er nødvendig at foretage en indsamling af såvel eksisterende krav som nye. Indsamlingen af eksisterende krav sker på baggrund af kildekoden fra LabInfo og LabInfoAUH samt tilhørende dokumentation. Mens de nye krav opstår i forbindelse med Edb Gruppens ønske om at forbedre systemets struktur, præsenteres i afsnit 4.1.

Ud fra programdesignet og dokumentationen for de to systemer beskrives deres indbyrdes funktionalitet. Hver funktion er forbundet med et eller flere systemkrav. Disse systemkrav fremgår af kildekoden og kan af den grund være vanskelige at præcisere.

Funktionaliteten bestemmes ved systematisk at gennemgå systemets hjemmesider. For hver side noteres den tilgængelige funktionalitet. På den måde kortlægges funktionaliteten.

Den opnåede information bruges til opbygning af "use cases". Samt til dokumentation af specifikationerne for brugerkravene.

Nedenfor opstilles først simple use cases for Labinfo systemet. Hver use case beskrives med henblik på at identificere brugen af include og extend. De endelige use cases kan derefter opstilles og de enkelte aktører af systemet beskrives.

Selvom funktionaliteten for LabInfoAUH er meget lig den for LabInfo, bliver der i det efterfølgende kun fokuseret på det centrale LabInfo system. Funktionalitet for det decentrale system LabInfoAUH bliver gennemgået i afsnit 11.

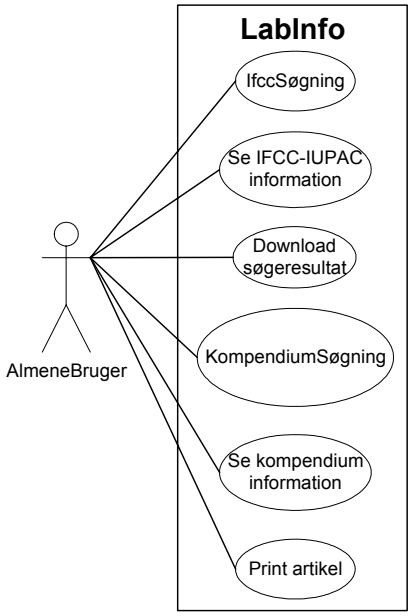
5.1 Simple use cases

Der eksisterer ikke noget dokument omkring brugerniveauet på LabInfo. Men ved hjælp af brugergrænsefladen for programmet defineres der tre brugerniveauer:

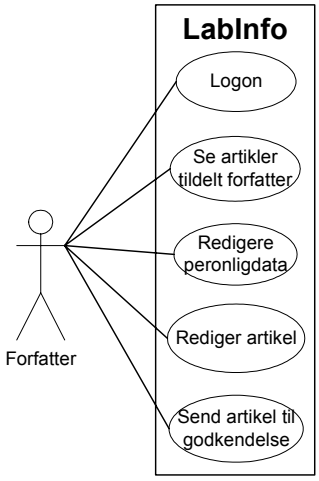
- AlmenBruger
- Forfatter
- Administrator

De tre nævnte er "actors" eller aktører på systemet.

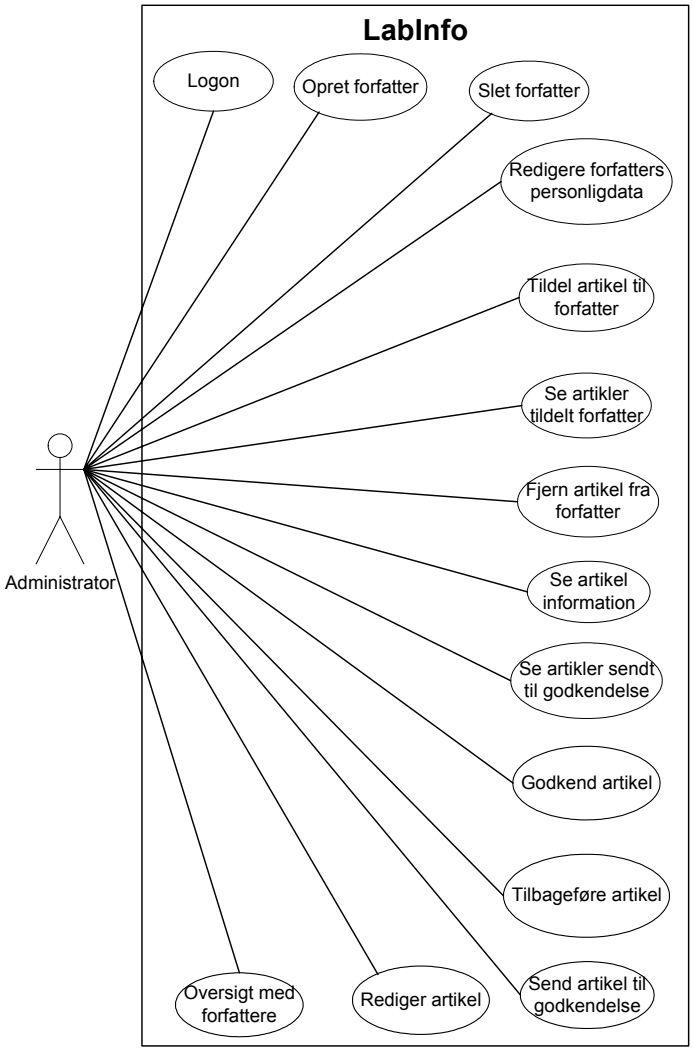
For hvert af systemets aktører opstilles en simpel use case for at illustrere den tilhørende funktionalitet. Funktionaliteten er udvundet fra programmets grænseflade og den tilhørende programbeskrivelse [LabInfo, 2003].



Figur 4: Simpel use case for aktøren almenBruger.



Figur 5: Simpel use case for aktøren forfatter.



Figur 6: Simpel use case for aktøren administrator.

6 Use case beskrivelser

For hver use case opstilles den tilhørende beskrivelse. Beskrivelsen er opdelt i en aktør handling og det efterfølgende svar fra systemet. De krav, der er forbundet med en funktionalitet, bliver hermed beskrevet.

6.1 AlmeneBruger

6.1.1 IFCCSøgning

Aktør handling		System svar	
1	Brugeren trykker *Søg* under IFCC-IUPAC menupunktet.	2	Viser en formular over mulige søgekriterier inden for IFCC-IUPAC.
3	Brugeren præciserer sin søgning og trykker *Søg*.	4	Søger i databasen efter mulige godkendte artikler og viser en oversigt med navnet på disse. Artikler fundet via deres synonym vises separat.

6.1.2 Se IFCC-IUPAC information

Aktør handling		System svar	
1	Brugeren trykker *Søg* under IFCC-IUPAC menupunktet.	2	Viser en formular over mulige søgekriterier inden for IFCC-IUPAC.
3	Brugeren præciserer sin søgning og trykker *Søg*.	4	Søger i databasen efter mulige godkendte artikler og viser en oversigt med navnet på disse. Artikler fundet via deres synonym vises separat.
5	Brugeren vælger en artikel fra oversigten ved at klikke på artiklens navn.	6	Viser artiklens navn og dens IFCC-IUPAC information.

6.1.3 Download søgeresultat

Aktør handling		System svar	
1	Brugeren trykker *Søg* under IFCC-IUPAC menupunktet.	2	Viser en formular over mulige søgekriterier inden for IFCC-IUPAC.
3	Brugeren præciserer sin søgning og trykker *Tekstfil*.	4	Søger i databasen efter mulige godkendte artikler.

			Listen skrives til en fil. Viser et link til filen.
5	Brugeren henter listen ned ved at klikke på linket.	6	Når filen er hentet slettes den fra systemet.

6.1.4 KompendiumSøgning

Aktør handling		System svar	
1	Brugeren trykker *Søg* under menupunktet Kompendium 2000.	2	Viser en formular over mulige søgekriterier inden for Kompendium 2000.
3	Brugeren præciserer sin søgning og trykker *Søg*.	4	Søger i databasen efter mulige godkendte artikler og viser en oversigt med navnet på disse. Artikler fundet via deres synonym vises separat.

6.1.5 Se kompendium information

Aktør handling		System svar	
1	Brugeren trykker *Søg* under menupunktet Kompendium 2000.	2	Viser en formular over mulige søgekriterier inden for Kompendium 2000.
3	Brugeren præciserer sin søgning og trykker *Søg*.	4	Søger i databasen efter mulige godkendte artikler og viser en oversigt med navnet på disse. Artikler fundet via deres synonym vises separat.
5	Brugeren vælger en artikel fra oversigten ved at klikke på artiklens navn.	6	Viser artiklens navn, IFCC-IUPAC information og liste information. Kompendium informationen opdeles i tre kategorier: medicinsk, reference intervaller og laboratorium information.
5	Brugeren vælger *Med. Iinfo.*	6	Viser artiklens navn, medicinsk information fra kompendium 2000 og liste informationen.
7	Brugeren vælger *Ref. Interval.*	8	Viser artiklens navn, information om reference intervaller fra kompendium 2000.
9	Brugeren vælger *Lab. Info.*	10	Viser artiklens navn, Laboratorium information fra kompendium 2000.

6.1.6 Print artikel

Aktør handling		System svar	
1	Brugeren trykker *Søg* under menupunktet Kompendium 2000.	2	Viser en formular over mulige søgekriterier inden for Kompendium 2000.
3	Brugeren præciserer sin søgning og trykker *Søg*.	4	Søger i databasen efter mulige godkendte artikler og viser en oversigt med navnet på disse. Artikler fundet via deres synonym vises separat.
5	Brugeren vælger en artikel fra oversigten ved at klikke på artiklens navn.	6	Viser artiklens navn, IFCC-IUPAC information og liste information. Kompendium informationen opdeles i tre: medicinsk, reference intervaller og laboratorium information.
7	Brugeren vælger *Print artikel*.	8	Viser en printvenlig side med al information for artiklen.

6.2 Forfattere

6.2.1 Logon

Aktør handling		System svar	
1	Forfatteren indtaster brugernavn og adgangskode. Trykker *Login*.	2	Systemet validerer og accepterer brugeren. Viser forfatterens startside med personlig information og tildelte artikler. Ud for hver tildelt artikel er deres status illustreret.

Alternativ handling:
Punkt 2. Brugeren afvises og sendes tilbage til punkt 1 med besked om fejl.

6.2.2 Se artikler tildelt forfatter

Aktør handling		System svar	
1	Forfatteren logger på systemet.	2	Viser forfatterens startside med personlige information, en liste over tildelte artikler og deres status.

6.2.3 Redigere personligdata

Aktør handling	System svar

1	Fra forfatterens startside vælges *Ret oplysninger*.	2	Viser en formular med forfatterens personlige data.
3	Forfatteren opdaterer sine oplysninger og trykker *Ok*.	4	Systemet gemmer formularen og viser de personlige data.
5	Forfatteren trykker *Ok*.	6	Systemet vender tilbage til forfatterens startside.

6.2.4 Redigere artikel

Aktør handling		System svar	
1	Fra forfatterens startside med tildelte artikler vælges den artikel, der skal ændres. Artikler sendt til godkendelse kan ikke vælges.	2	Systemet viser artiklens navn og IFCC-IUPAC information. Informationen for Kompendium 2000 vises i en formular.
3	Forfatteren opdaterer formularen og trykker *Gem artikel*.	4	Systemet beder forfatteren om at bekræfte valget. Hvis artiklen er godkendt informeres forfatteren om, at artiklen ændrer status til ej godkendt efter endt handling.
5	Forfatteren trykker *Ok*.	6	Systemet gemmer kompendium data fra formularen og ændre artiklens status til ikke at være godkendt. Viser informationen for artiklen.
7	Forfatteren trykker Tilbage til oversigten.	8	Systemet vender tilbage til forfatterens startside.

Alternativ handling:

Punkt 7. Forfatteren kan vælge *fortsæt redigeringen*. Systemet vender dermed tilbage til punkt 2.

6.2.5 Send artikel til godkendelse

Aktør handling		System svar	
1	Fra startsiden med tildelte artikler vælges den artikel, der skal ændres. Artikler sendt til godkendelse kan ikke vælges.	2	Systemet viser artiklens navn og IFCC-IUPAC information. Informationen for Kompendium 2000 vises i en formular.
3	Forfatteren trykker *Godkend artikel*.	4	Systemet beder forfatteren om at bekræfte valget.
5	Forfatteren trykker *Ok*.	6	Gemmer kompendium data fra formularen, ændre artiklens status til ikke at være godkendt

			og sender den til redaktøren. Viser informationen for artiklen.
7	Forfatteren trykker Tilbage til oversigten.	8	Systemet vender tilbage til forfatterens startside.

Alternativ handling:

Punkt 7. Forfatteren kan vælge *fortsæt redigeringen*. Systemet vender dermed tilbage til punkt 2.

6.3 Administrator

6.3.1 Logon

Aktør handling		System svar	
1	Administratoren indtaster brugernavn og adgangskode og trykker *Ok*..	2	Systemet validerer og accepterer bruger. Viser startside med mulighed for at *se tilsendte artikler*, *opret forfatter* og *ret forfatter*.

Alternativ handling:

Punkt 2. Brugeren afvises og sendes tilbage til punkt 1 med besked om fejl.

6.3.2 Opret forfatter

Aktør handling		System svar	
1	Administratoren vælger *Opret forfattere*.	2	Viser en formular til personlig data for en forfatter.
3	Administratoren udfylder formularen og trykker *opret ny forfatter*.	4	Systemet gemmer data fra formularen og viser den indtastede data.
5	Administratoren trykker *Ok*.	6	Systemet vender tilbage til administratorens startside.

6.3.3 Oversigt med forfattere

Aktør handling		System svar	
1	Administratoren vælger *Ret forfatter*.	2	Viser en oversigt med forfattere sorteret efter forbogstavet i fornavn eller efternavn.

6.3.4 Slet forfatter

Aktør handling		System svar	
1	Administratoren vælger *Ret*	2	Viser en oversigt med

	forfattere.		forfattere sorteret efter forbogstavet i fornavn eller efternavn.
3	Administratoren vælger *Slet forfatter* udfor den pågældende forfatter.	4	Viser en dialog for at bekræfte valget.
5	Administratoren vælger *Ok*.	6	Slettet forfatteren fra systemet. Og vender tilbage til oversigten med forfattere fra punkt 2.

6.3.5 Redigere personligdata

Aktør handling		System svar	
1	Administratoren vælger *Ret forfatter*.	2	Viser en oversigt med forfattere sorteret efter forbogstavet i fornavn eller efternavn.
3	Administratoren finder forfatteren i oversigten og vælger *Ret forfatters oplysninger*.	4	Systemet viser en formular med forfatterens personlige oplysninger.
5	Administratoren opdaterer formularen og trykker *Ok*.	6	Data fra formularen gemmes og indholdet vises.
7	Administratoren trykker *Ok*.	8	Systemet vender tilbage til forfatteroversigten i punkt 2.

6.3.6 Tildel artikel til forfatter

Aktør handling		System svar	
1	Administratoren vælger *Ret forfatter*.	2	Viser en oversigt med forfattere sorteret efter forbogstavet i fornavn eller efternavn.
3	Administratoren finder forfatteren i oversigten og vælger *Se/ret artikelliste*.	4	Viser en liste over forfatterens tildelte artikler.
5	Administratoren indskriver koden på en artikel og trykker *Tilføj*.	6	Viser om artiklen eventuelt er tildelt anden forfatter og dialog om at bekræfte valg.
7	Administratoren vælger *Ok*.	8	Tildeler artiklen til forfatteren og vender tilbage til oversigt med tildelte artikler, punkt 4.

6.3.7 Se artikler tildelt forfatter

Aktør handling	System svar

1	Administratoren vælger *Ret forfatter*.	2	Viser en oversigt med forfattere sorteret efter forbogstavet i fornavn eller efternavn.
3	Administratoren finder forfatteren i oversigten og vælger *Se/ret artikelliste*.	4	Viser en liste over forfatterens tildelte artikler.

6.3.8 Fjern artikel fra forfatter

Aktør handling		System svar	
1	Administratoren vælger *Ret forfatter*.	2	Viser en oversigt med forfattere sorteret efter forbogstavet i fornavn eller efternavn.
3	Administratoren finder forfatteren i oversigten og vælger *Se/ret artikelliste*.	4	Viser en liste over forfatterens tildelte artikler.
5	Administratoren vælger *Slet* ud for artiklen i listen.	6	Fjerner artiklen fra forfatteren og vender tilbage til oversigten med tildelte artikler, punkt 4.

6.3.9 Se artikel information

Aktør handling		System svar	
1	Administratoren vælger *Ret forfatter*.	2	Viser en oversigt med forfattere sorteret efter forbogstavet i fornavn eller efternavn.
3	Administratoren finder forfatteren i oversigten og vælger *Se/ret artikelliste*.	4	Viser en liste over forfatterens tildelte artikler.
5	Administratoren vælger *Vis* ud for den pågældende artikel.	6	Viser IFCC-IUPAC og kompendium informationen for artiklen.

6.3.10 Se artikler sendt til godkendelse

Aktør handling		System svar	
1	Administratoren vælger *Nye artikler*.	2	Systemet viser en oversigt med navnet på de artikler, der er sendt til godkendelse.

6.3.11 Godkend artikel

Aktør handling	System svar

1	Administratoren vælger *Nye artikler*.	2	Systemet viser en oversigt med navnet på de artikler, der er sendt til godkendelse.
3	Administratoren vælger *Ændre status* ud for den pågældende artikel.	4	Viser dialog med mulighed for at godkende, tilbageføre eller annullere.
5	Administratoren vælger *Godkend*.	6	Ændrer artiklens status til godkendt og sender den tilbage til forfatteren. Vender tilbage til oversigten med nye artikler, punkt 2.

6.3.12 Tilbageføre artikel

Aktør handling		System svar	
1	Administratoren vælger *Nye artikler*.	2	Systemet viser en oversigt med navnet på de artikler, der er sendt til godkendelse.
3	Administratoren vælger *Ændre status* ud for den pågældende artikel.	4	Viser dialog med mulighed for at godkende, tilbageføre eller annullere.
5	Administratoren vælger Tilbageføre til forfatteren.	6	Sender artiklen tilbage til forfatteren. Vender tilbage til oversigten med nye artikler, punkt 2.

6.3.13 Send artikel til godkendelse

Aktør handling		System svar	
1	Administratoren vælger *Ret forfatter*.	2	Viser en oversigt med forfattere sorteret efter forbogstavet i fornavn eller efternavn.
3	Administratoren finder forfatteren i oversigten og vælger *Se/ret artikelliste*.	4	Viser en liste over forfatterens tildelte artikler.
3	Administratoren vælger *Ændre* artiklens *status* ud for den pågældende artikel.	4	Viser dialog med mulighed for at godkende, tilbageføre, indsende til redaktøren eller annullere.
5	Administratoren vælger *Indsend til redaktøren*.	6	Ændrer artiklens status og vender tilbage til oversigten med tildelte artikler, punkt 4.

6.3.14 Redigere artikel

Aktør handling		System svar	
1	Administratoren vælger *Nye artikler*.	2	Systemet viser en oversigt med navnet på de artikler, der er sendt til godkendelse.
3	Administratoren trykker *Ret*.	4	Viser artiklens navn og IFCC-IUPAC information. Informationen for Kompendium 2000 vises i en formular.
5	Administratoren opdaterer informationen i formularen og trykker *Gem artikel*.	6	Formularen gemmes og informationen for artiklen vises.
7	Administratoren trykker *Tilbage til oversigten*.	8	Vender tilbage til oversigten med nye artikler, punkt 2.

Alternativ handling:

Punkt 5. Administratoren kan vælge *Godkend artikel*. I punkt 6 vil systemet dermed også ændre artiklens status til godkendt og fjerne den fra oversigten over nye artikler.

Punkt 7. Administratoren kan vælge *Fortsæt redigering* og systemet vender tilbage til punkt 4.

6.4 Brug af <<uses>>

På baggrund af ovenstående beskrivelser udvides use case diagrammer i figur 4, figur 5 og figur 6 til også at indeholde <<uses>> relationer. Resultatet kan ses i afsnit 7 Endelige use case diagrammer.

For hver aktør er der funktioner, som kræver at systemet et i stand til at finde navnet på en eller flere artikler. Eksempelvis til visning af en oversigt som 6.1.1. Derfor indføres *Find artikel*, til at finde og pege på en bestemt artikel. *Find artikel* bliver brugt af følgende funktioner:

- IFCCSøgning
- KompendiumSøgning
- Startside med tildelte artikler
- Se artikler tildelt forfatter
- Se artikler sendt til godkendelse

Oversigterne fra ovenstående funktioner viser kun artiklens navn og ikke dens samlede information. Vælges en artikel som i 6.2.4, bliver artiklens IFCC- IUPAC og kompendium information også vist. For hver aktør

eksisterer et behov for at få præsenteret disse to slags informationer enten samtidig eller enkeltvis.

Find IFCC-IUPAC information indføres til brug for:

- Vis IFCC-IUPAC information
- Vis Kompendium information
- Redigere artikel
- Se artikel

Find kompendium information indføres for følgende:

- Vis Kompendium information
- Redigere artikel
- Se artikel

En forfatter har brug for at blive genkendt af systemet, når han logger på. Administratoren har brug for at få præsenteret samtlige forfattere på systemet. Derfor indføres *Find forfatter* til det formål at finde en forfatter i systemet. *Find forfatter* bruges af:

- Logon
- Redigere personligdata
- Oversigt med forfattere

Administratoren har også brug for at blive genkendt af systemet. Derfor indføres *Find administrator* til brug for funktionen:

- LogonAdmin

Både administratoren og forfatteren har mulighed for at redigere en artikels information. Efter endt redigering kan forfatteren sende artiklen til godkendelse og administratoren kan godkende artiklen. Ved disse handlinger gemmes den ændrede artikel også. Derfor indføres en relation til funktionen *Gem artikel* fra følgende:

- Godkend artikel
- Send artikel til godkendelse

6.5 Brug af <<extends>>

Use case diagrammerne fra figur 4, figur 5 og figur 6 udstyres med <<extends>> relationer. Aktørens tilgang til de enkelte funktioner præsenteres dermed visuelt.

Af afsnit 6.1.2 fremgår det, at brugeren kun kan få vist IFCC-IUPAC informationen for en artikel ved at vælge artiklen fra en liste. Derfor indføres en <<extends>> relation fra *IFCCSøgning* til *Vis IFCC-IUPAC information*.

Funktionen til at hente søgeresultatet fra en søgning ned til en fil kræver også at *IFCCSøgning* er udført. Se 6.1.3. En relation fra *IFCCSøgning* til D*ownload søgeresultat* er derfor nødvendig.

Samme situation opstår ved præsentationen af kompendium informationen for en artikel. Der oprettes en relation fra *kompendiumSøgning* til *Vis kompendium information*. Printningen af artiklens information kan ifølge afsnit 6.1.6 kun gøres når artiklens kompendium information er vist. En relation fra *Vis kompendium information* til *print artikel* er derfor nødvendig.

Når en forfatter oprettes af administratoren bliver funktionen *Gem forfatter* brugt. Samme funktion bruges senere af administratoren eller forfatteren for at foretage ændringer i de personlige data for forfatteren. Der skabes derfor en relation fra følgende funktioner til funktionen *Gem forfatter*:

- Opret forfatter
- Redigere personligdata

Afsnit 6.2.4 og 6.3.14 beskriver forløbet for redigering af en artikel foretaget af forfatteren eller administratoren. I begge tilfælde afsluttes med at informationen for den opdaterede artikel gemmes. Men samtidig er det også muligt at påvirke artiklens status med hensyn til at være godkendt eller ej. Fra *Rediger artikel* indføres en relation til følgende funktioner:

- Gem artikel
- Send artikel til godkendelse
- Godkend artikel

Oversigten med forfattere beskrevet i afsnit 6.3.3 bliver brugt i forbindelse med udførelsen af følgende tre funktioner:

- Slet forfatter
- Redigere personligdata
- Se artikler tildelt forfatter

Af den grund oprettes en <<extends>> relation fra *Oversigt med forfattere* til ovenstående tre funktioner.

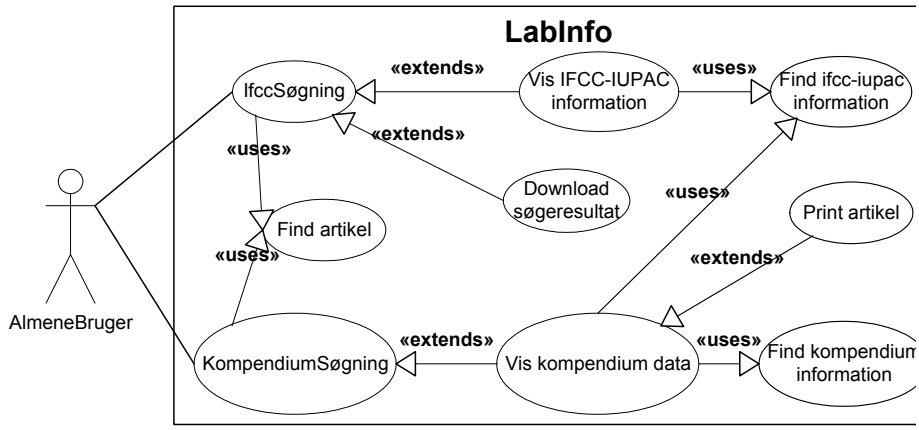
Se artikler tildelt forfatter, beskrevet i afsnit 6.3.7, danner udgangspunkt for seks andre funktioner tillagt administratoren. Disse funktioner kræver alle, at administratoren har valgt en forfatter, førend den pågældende funktion kan udføres. En relation oprettes til disse seks funktioner, som er:

- Fjern artikel fra forfatter
- Tildel artikel til forfatter
- Send artikel til godkendelse
- Se artikel information
- Godkend artikel
- Tilbageføre artikel

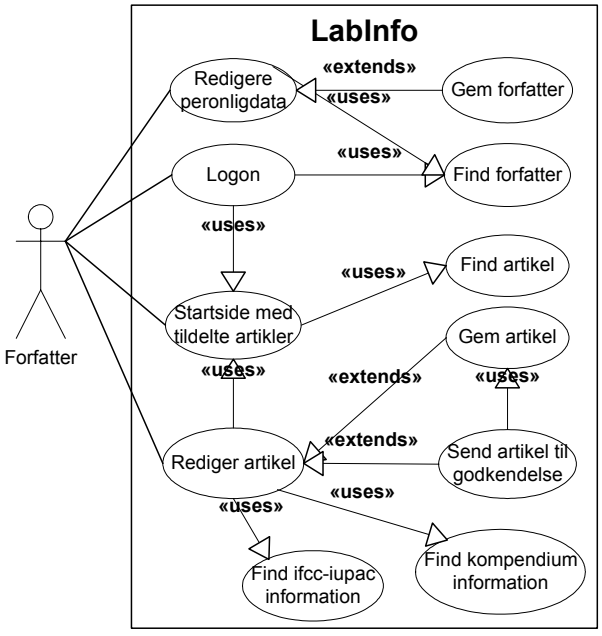
Som administrator er der mulighed for at udføre den samme funktion flere steder i systemet. Eksempelvis kan *Godkend artikel* og *Tilbageføre artikel* også aktiveres fra *Se artikler sendt til godkendelse*, afsnit 6.3.10. Relationerne fra denne funktion er:

- Godkend artikel
- Tilbageføre artikel
- Redigere artikel

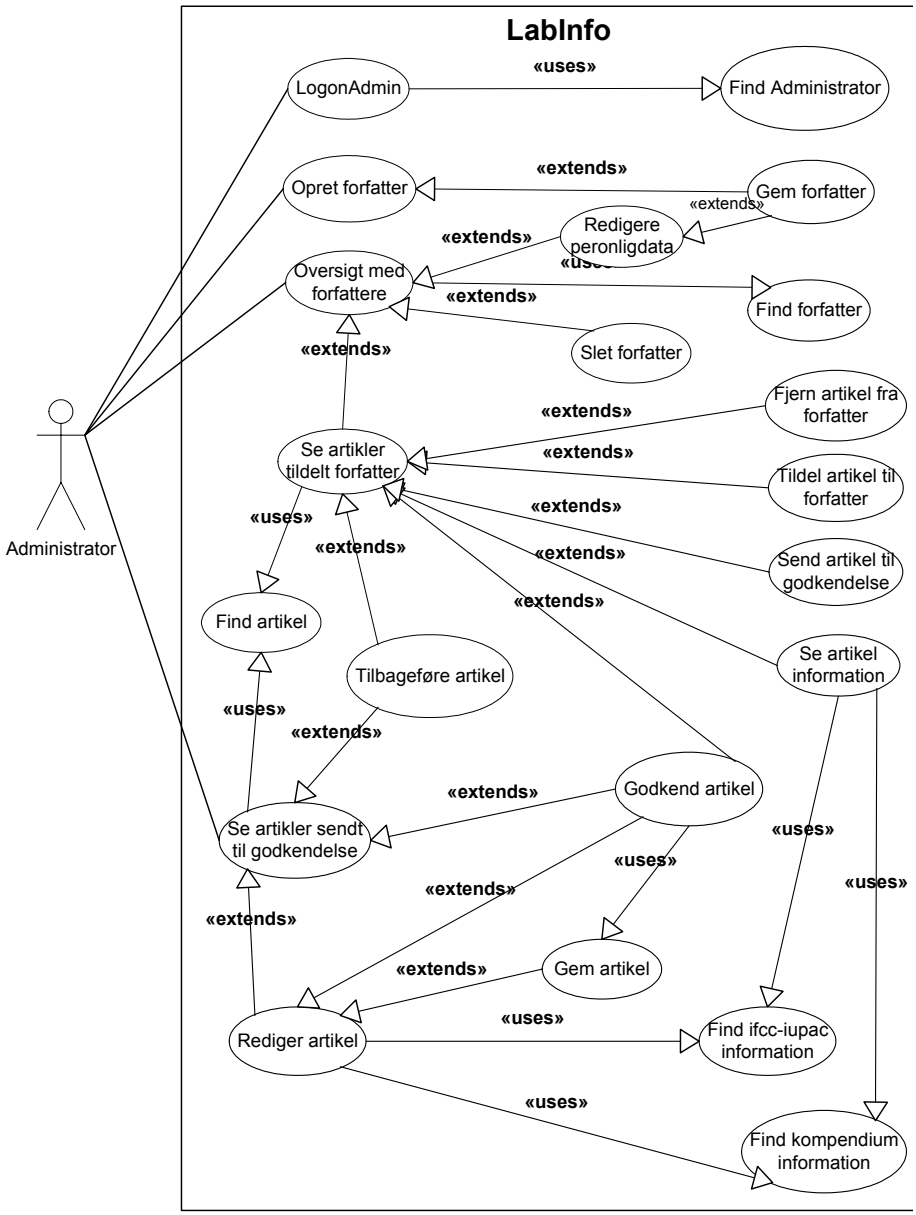
7 Endelige use case diagrammer



Figur 7: Endelige use case for aktøren almeneBruger.



Figur 8: Endelige use case for aktøren forfatter.



Figur 9: Endelige use case for aktøren administrator.

8 Aktørbeskrivelse

8.1 Administrator

En administrator er det administrerende organ for programmet. Programmet kan have en eller flere administratorer. De er ansvarlige for oprettelse og redigering af forfattere til programmet og har ansvaret for programmets artikler. De står for distribueringen af ansvaret for programmets artikler blandt de registrerede forfattere. De skal foretage en vurdering og godkendelse af artikler, hvis indholdet af disse ændres af en forfatter. Det endelige ansvar for artiklernes information er tillagt administratoren. Administratoren afgør hvorvidt en artikel er udgået, offentligt tilgængelig eller ej.

8.2 Forfatter

En forfatter er ansvarlig for de artikler, han eller hun er blevet tildelt af administratoren. Ansvaret indebærer vedligeholdelse af artiklens kompendium information. Ændrer forfatteren artiklens indhold har vedkomne ansvar for, at den bliver sendt til godkendelse hos administratoren. Forfatteren har ansvaret for opdatering af personlige data.

8.3 Almen bruger

Den almene bruger har ikke noget brugermæssigt ansvar. Almene brugere omfatter administratorer, forfattere samt andre bruger uden tilknytningen til programmet. Almene brugere har adgang til de artikler, som administratoren har gjort offentlig tilgængelig.

9 Klassespecifikation

Fra informationen præsenteret i use case beskrivelserne findes følgende klasser:

- Administrator
- AlmenBruger
- Artikel
- ArtikelIFCCData
- ArtikelKompendiumData
- ArtikelListe
- ArtikelSpecialeKobling
- ArtikelSynonym
- Forfatter

9.1 Beskrivelse af klasserne

For hver af ovenstående klasser følger her en detaljeret beskrivelse af deres attributter og operationer. Beskrivelsen af operationerne udføres i OCL. Til hver klasse er vist en figur med UML repræsentation over klassens indhold. Det samlede klassediagram med indbyrdes relationer er opstillet i afsnit 9.2.

9.1.1 Administrator

Figur 10: Administrator klassen

Klassebeskrivelse: **Administrator**

Der kan være en eller flere administratorer til at håndtere administration på systemet. En administrator er beskrevet ved denne klasse.

Specifikation af attribut: **adgangskode**

Beskrivelse: Hver administrator har en adgangskode, der giver adgang til funktioner forbeholdt administratorer.
Format: En tekststreng på maksimal 5 tegn.

Specifikation af attribut: **brugernavn**

Beskrivelse: Hver administrator er identificeret ved et brugernavn. Brugernavnet er et unika.
Format: En tekststreng på maksimalt 8 tegn.

Specifikation af operation: **fjernTildeling**

Hensigt: Fjerner en forfatter fra en artikel, forudsat at artiklen ikke er godkendt eller sendt til godkendelse.
Signatur: Administrator::fjernTildeling(enArtikelKode : Integer, etBrugernavn : String)
Logisk beskrivelse:
Pre: enArtikelKode->notEmpty
etBrugernavn->notEmpty
self.tildelteArtikler->exists(kode=enArtikelKode and forfatterkode=etBrugernavn and godkendt=False and redklar=false)
Post: not self.tildelteArtikler->exists(kode=enArtikelKode and forfatterkode=etBrugernavn)
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: raises InvalidArtikelStatusException
Non-functional requirements: none

Specifikation af operation: **gemArtikel**

Hensigt: Overskriver en eksisterende artikel med ny data.
Signatur: Administrator::gemArtikel(enArtikel : Artikel, enKompendiumData : ArtikelKompendiumData)
Logisk beskrivelse:
Pre: self.enArtikel->notEmpty
Post: none
Andre operationskald: Artikel.gemArtikelData
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **gemNyForfatter**

Hensigt: Gemmer en ny forfatter i systemet.
Signatur: Administrator::gemNyForfatter(enForfatter : Forfatter)
Logisk beskrivelse:
Pre: not bruger->includes(enForfatter)
Post: bruger->includes(enForfatter)
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: raises InvalidForfatterException
Non-functional requirements: none

Specifikation af operation: **godkendArtikel**

Hensigt: Godkender en artikel ved at ændre artiklens status til godkendt. Artiklen tilbageføres samtidig til forfatteren.
Signatur: Administrator::godkendArtikel(enKode : String)
Logisk beskrivelse:
Pre: self.enKode->notEmpty
Artikel->exists(kode=enKode)
Post: Artikel->select(a :Artikel | kode=enKode)->a.godkendt=true
Artikel->select(a :Artikel | kode=enKode)->a.redKlar=false
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: raises none
Non-functional requirements: none

Specifikation af operation: **listArtiklerTilGodkendelse**

Hensigt: Opstiller en liste med de artikler der er sendt til godkendelse.
Signatur: Administrator::listArtiklerTilGodkendelse()
Logisk beskrivelse:
Pre: none
Post: none
Andre operationskald: Artikel.erTilGodkendelse
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **listForfattere**

Hensigt: Henter alle forfattere, hvis begyndelsesbogstavet i enten fornavnet eller efternavnet er lig et givent bogstav.
Signatur: Administrator::listForfattere(etBogstav : String, sortering : String)
Logisk beskrivelse:

Pre: none
Post: none
Andre operationskald: Forfatter.hentForfatternavn
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **listTildelteArtikler**

Hensigt: Henter alle tildelte artikler for en enkelt forfatter.
Signatur: Administrator::listTildelteArtikler(etBrugernavn : String)
Logisk beskrivelse:
Pre: self.tildelteArtikler.kode->select(a:Artikel | a.brugernavn=etBrugernavn)
Post: none
Andre operationskald: Artikel.hentArtikelNavn
Events transmitteret til andre objekter:
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **logon**

Hensigt: Give adgang til administrator delen, hvis brugeren eksisterer i systemet. Ellers vises en fejlbesked.
Signatur: Administrator::logon(enAdmin : Administrator)
Logisk beskrivelse:
Pre: self.enAdmin->notEmpty
Post: result = self.Administrator->includes(enAdmin)
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: raises InvalidAdministratorException
Non-functional requirements: none

Specifikation af operation: **opretForfatter**

Hensigt: Muliggør oprettelsen af en ny forfatter.
Signatur: Administrator::opretForfatter()
Logisk beskrivelse:
Pre: none
Post: none
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **redigereForfattersData**

Hensigt: Viser en forfatters personlige data, hvori der kan redigeres.
Signatur: Administrator::redigereForfattersData()
Logisk beskrivelse:
Pre: none
Post: none
Andre operationskald: Forfatter.hentPersonligdata
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **sendTilGodkendelse**

Hensigt: Sender en artikel til godkendelse ved at ændre dens status.
Signatur: Administrator::sendTilGodkendelse(enArtikelKode : Integer)
Logisk beskrivelse:
Pre: self.artikel->exists (kode=enArtikelKode)
Post: self.artikel->exists(kode=enArtikelKode and redKlar=True and godkendt=False)
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **sletForfatter**

Hensigt: Fjerner en forfatter fra systemet. Forfatteren kan kun fjernes hvis personen ikke har tildelt nogen artikler.
Signatur: Administrator::sletForfatter(enForfatterkode : String)
Logisk beskrivelse:
Pre: not self.artikel->exists (forfatterkode=enForfatterkode)
Post: not self.bruger->exists (brugernavn=enForfatterkode)
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: raises InvalidForfatterException
Non-functional requirements: none

Specifikation af operation: **tilbageførArtikel**

Hensigt: Sender en artikel tilbage til en forfatter ved at ændre artiklens status.
Signatur: Administrator::tilbageførArtikel(enArtikelKode : Integer)
Logisk beskrivelse:
Pre: self.artikel->exists (kode=enArtikelKode and not godkendt and redKlar)

Post: self.artikel->exists (kode=enArtikelKode and not redKlar and not godkendt)

Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: raises InvalidArtikelStatusException
Non-functional requirements: none

Specifikation af operation: **tildelArtikelTil**

Hensigt: Tildeler en artikel til en forfatter. Det kontrolleres om artiklen i forvejen er tildelt en anden forfatter. Er det tilfældet returneres navnet på denne forfatter.

Signatur: Administrator::tildelArtikelTil(enForfatterkode : String, enArtikelkode : String) : String

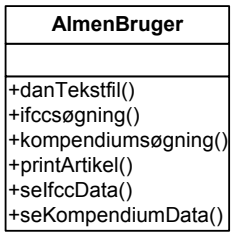
Logisk beskrivelse:

Pre: self.bruger->exists(brugernavn=enForfatterkode)
if(self.artikel->select(a: Artikel | a.kode=enArtikelkode and a.forfatterkode <> null) then
(result = a.forfatterkode)
else (result="")
endif

Post: self.artikel->exists(kode=enArtikelkode and forfatterkode=enForfatterkode)

Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

9.1.2 AlmenBruger



Figur 11: AlmenBruger klassen.

Klassebeskrivelse: **AlmenBruger**

Repræsenterer den funktionalitet, der er tillagt den almene bruger af systemet.

Specifikation af operation: **danTekstfil**

Hensigt: At udføre en søgning på artikler indenfor IFCC kodeskemaet og udskrive søgeresultatet til en tekstfil.
Signatur: AlmenBruger::danTekstfil(enSøgeStreng : String)
Logisk beskrivelse:
Pre: none
Post: none
Andre operationskald: Artikel.artikelsøgning
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: Det kræves at systemet kan udføre søgningen på få sekunder.

Specifikation af operation: **ifccsøgning**

Hensigt: At udføre en søgning på artikler indenfor IFCC kodeskemaet og opstille søgeresultatet med navnene på de fundne artikler.
Signatur: AlmenBruger::ifccsøgning(enSøgeStreng : String)
Logisk beskrivelse:
Pre: none
Post: none
Andre operationskald: Artikel.artikelsøgning
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: Det kræves at systemet kan udføre søgningen på få sekunder.

Specifikation af operation: **kompendiumsøgning**

Hensigt: At udføre en søgning indenfor Kompendium informationen og opstille søgeresultatet med navnene på de fundne artikler.
Signatur: AlmenBruger::kompendiumsøgning(enSøgeStreng : String)
Logisk beskrivelse:
Pre: none
Post: none
Andre operationskald: Artikel.artikelsøgning
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: Det kræves at systemet kan udføre søgningen på få sekunder.

Specifikation af operation: **printArtikel**

Hensigt: Præsenterer Kompendium informationen på en printvenligmåde.
Signatur: AlmenBruger::printArtikel(a : Artikel, i : ArtikelIFCCData, k : ArtikelKompendiumData)

Logisk beskrivelse:
Pre: self.a->notEmpty
Post:
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **seIFCCData**

Hensigt: At præsentere en artikels IFCC information og liste information.
Signatur: AlmenBruger::seIFCCData(enArtikel : Artikel)
Logisk beskrivelse:
Pre: self.artikel->includes(enArtikel)
Post: none
Andre operationskald: ArtikelIFCCData.hentIFCCData, ArtikelListe.hentListe
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **seKompendiumData**

Hensigt: At præsentere en artikels IFCC, Kompendium og liste information.
Signatur: AlmenBruger::seKompendiumData(enArtikel : Artikel)
Logisk beskrivelse:
Pre: self.artikel->includes(enArtikel)
Post: none
Andre operationskald: ArtikelIFCCData.hentIFCCData, ArtikelKompendiumData.hentKompendiumData, ArtikelListe.hentListe
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional reqsuirements: none

9.1.3 Artikel

Figur 12: Artikel klassen.

Klassebeskrivelse: **Artikel**

Klassen repræsenterer basis informationen for en artikel i systemet. Det omfatter information for artiklens forkortede navn og dens status.

Specifikation af attribut: **adminDato**

Beskrivelse:	Angiver datoen, hvor artikel oprettes i systemet.
Format:	Dato format af typen dd-mm-yyyy.

Specifikation af attribut: **adminInfo**

Beskrivelse:	Eventuel administrator kommentar til artiklen noteres her.
Format:	Tekststreng af maksimalt 50 tegn.

Specifikation af attribut: **aktiv**

Beskrivelse:	Angiver om artiklen er udgået eller ej.
Format:	Boolean. Hvis falsk er artiklen udgået.

Specifikation af attribut: **enhed**

Beskrivelse:	Angiver eventuelle enhed for artiklen. Værdien er en del af artiklens forkortede navn.
Format:	Tekststreng på maksimalt 10 tegn.

Specifikation af attribut: **forfatterkode**

Beskrivelse:	Er artiklen tildelt en forfatter angives forfatterkoden på den pågældende forfatter her.

Format:	Tekststreng på maksimalt 8 tegn.

Specifikation af attribut: **godkendt**

Beskrivelse:	Angiver om artiklen er godkendt eller ej.
Format:	Boolean. Hvis sandt er artiklen godkendt.

Specifikation af attribut: **kode**

Beskrivelse:	Angiver IFCC-IUPAC nummeret på artiklen.
Format:	Et naturligt tal på maksimalt 5 cifre.

Specifikation af attribut: **kodetekst**

Beskrivelse:	Angiver IFCC-IUPAC nummeret sammen med en kort forklarende tekst. Værdien er en del af artiklens forkortede navn.
Format:	Tekststreng på maksimalt 8 tegn.

Specifikation af attribut: **komponent**

Beskrivelse:	Indeholder komponent beskrivelsen for artiklen. Værdien er en del af artiklens forkortede navn.
Format:	Tekststreng på maksimalt 50 tegn.

Specifikation af attribut: **molmasse**

Beskrivelse:	Indeholder molmassen for artiklen. Værdien er en del af artiklens forkortede navn.
Format:	Et tal på maksimalt 10 tegn.

Specifikation af attribut: **præfiks**

Beskrivelse:	Angiver det eventuelle præfiks for artiklen. Værdien er en del af artiklens forkortede navn.
Format:	Tekststreng på maksimalt 10 tegn.

Specifikation af attribut: **redKlar**

Beskrivelse:	Angiver om artiklen er hos redaktøren for at blive godkendt eller hos en forfatter.
Format:	Boolean. Hvis sandt er artiklen til godkendelse hos redaktøren/administratoren.

Specifikation af attribut: **revideret**

Beskrivelse:	Angiver datoen, hvor artiklens kompendium information blev ændret.
Format:	Dato format af typen dd-mm-yyyy.

Specifikation af attribut: **revideretAf**

Beskrivelse:	Angiver forfatterkoden på den forfatter der sidst har foretaget ændringer i artiklens kompendium information.

Format: Tekststreng på maksimalt 8 tegn.

Specifikation af attribut: **system**

Beskrivelse: Angiver artiklens system betegnelse. Betegnelsen er en del af artiklens forkortede navn.
Format: Tekststreng på maksimalt 50 tegn.

Specifikation af operation: **artikelsøgning**

Hensigt: Returnerer koden på godkendte artikler, der er omfattet af søgestrengen.
Signatur: Artikel::artikelsøgning(enSøgeStreng : String)
Logisk beskrivelse:
Pre: self.godkendt
Post:
Andre operationskald: ArtikelSynomym.artikelsøgning, ArtikelSpecialeKobling.artikelsøgning, Artikel.hentArtikelnavn
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements:

Specifikation af operation: **erTilGodkendelse**

Hensigt: Returnerer navnene på de artikler der er sendt til godkendelse.
Signatur: Artikel::erTilGodkendelse() liste : Set
Logisk beskrivelse:
Pre: none
Post: Set = self.Artikel->select(redKlar and not godkendt)
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **gemArtikelData**

Hensigt: Gemmer informationen for en artikel.
Signatur: Artikel::gemArtikelData(enArtikel : Artikel, aKD : ArtikelKompendiumData)
Logisk beskrivelse:
Pre: self.Artikel->exists(kode = enArtikel.kode)
Post: self.Artikel->includes(enArtikel)
Andre operationskald: ArtikelKompendiumData.gemKompendiumData
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

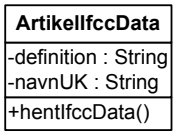
Specifikation af operation: **hentArtikelData**

Hensigt: Returnerer al data for en bestemt artikel.
Signatur: Artikel::hentArtikelData(enKode : Integer)
Logisk beskrivelse:
Pre: enKode -> notEmpty
Post: self.Artikel->select(kode=enKode)
Andre operationskald: ArtikelIFCCData.hentIFCCData, ArtikelKompendiumData.hentKompendiumData, ArtikelListe.hentListe
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **hentArtikelnavn**

Hensigt: Henter navnet på en bestemt artikel.
Signatur: Artikel::hentArtikelnavn(kode : Integer) artikelnavn : String
Logisk beskrivelse:
Pre: self.kode -> notEmpty
Artikel->exists(Artikel.kode=kode)
Post: artikelnavn = self.kodeTekst + self.system + self.præfiks + self.komponent + self.enhed
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

9.1.4 ArtikelIFCCData



Figur 13: ArtikelIFCCData klassen.

Klassebeskrivelse: **ArtikelIFCCData**

Klassen indeholder al IFCC-IUPAC information for en artikel som ikke indgår i artiklens navn og derfor ikke er indeholdt i klassen *Artikel*.

Specifikation af attribut: **definition**

Beskrivelse: Systematisk dansk navn uden forkortelser.
Format: Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **navnUK**

Beskrivelse:	Engelsk udgave af det forkortede navn sammensat i *Artikel* klassen.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af operation: **hentIFCCData**

Hensigt: Returnerer IFCC-IUPAC informationen for en bestemt artikel.

Signatur: ArtikelIFCCData::hentIFCCData(kode : Integer) ifccData : ArtikelIFCCData

Logisk beskrivelse:

Pre: self.kode -> notEmpty

Post: none

Andre operationskald: none

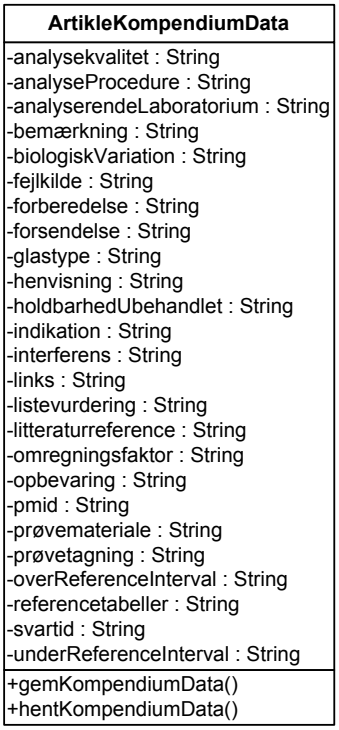
Events transmitteret til andre objekter: none

Attributes set: none

Response to exceptions: none

Non-functional requirements: none

9.1.5 ArtikelKompendiumData



Figur 14: ArtikelKompendiumData klassen.

Klassebeskrivelse: **ArtikelKompendiumData**

Indeholder al kompendium information, der er tilknyttet en artikel. Der er nødvendigvis ikke knyttet kompendium information til en artikel.

Specifikation af attribut: **analysekvalitet**

Beskrivelse:	For den angivne analyseprocedure: Teknisk impræcision og/eller inakkuratesse. Rekommandation eller standard.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **analyseProcedure**

Beskrivelse:	Hvor en bestemt procedure eller metode generelt anvendes til bestemmelse af artiklen. Er en del af artiklens laboratorieinformation.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **analyserendeLaboratorium**

Beskrivelse:	Analyserende laboratorium angives hvis undersøgelsen kun udføres af få (1-3) laboratorier. Er en del af informationen for artiklens prøvetagning.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **bemærkning**

Beskrivelse:	Her placeres information, der ikke logisk kan indpasses i nogle af basens felter.
Format:	Tekststreng på maksimalt 2000 tegn.

Specifikation af attribut: **biologiskVariation**

Beskrivelse:	Indtraindividuel variation som døgn- eller cyklusvariation. Er en del af artiklens medicinske information.
Format:	Tekststreng på maksimalt 2000 tegn.

Specifikation af attribut: **fejlkilde**

Beskrivelse:	Egentlige fejl, der opstår så hyppigt at der er grund til at advare mod dem. Er en del af artiklens laboratorieinformation.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **forberedelse**

Beskrivelse:	Patientforberedelse, rekvirering af speciel emballage, aftale med analyserende laboratorium etc., som skal være udført inden selve prøvetagningen.
Format:	Tekststreng på maksimalt 2000 tegn.

Specifikation af attribut: **forsendelse**

Beskrivelse:	Tidsfrister og transportforskrift. Er en del af informationen for artiklens prøvetagning.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **glastype**

Beskrivelse:	Art at prøveemballage og tilsætningsstoffer til brug ved udtagning af materialet. Er en del af informationen for artiklens prøvetagning.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **henvisning**

Beskrivelse:	Henvisninger til andre artikler med relevant information. Er en del af artiklens medicinske information.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **holdbarhedUbehandlet**

Beskrivelse:	Holdbarhed af det udtagne materiale ved stuetemperatur unden yderligere behandling. Er en del af information for artiklens prøvetagning.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **indikation**

Beskrivelse:	Klinisk indikation for at rekvirere undersøgelse af artiklen eller listen. Er en del af artiklens medicinske information.
Format:	Tekststreng på maksimalt 2000 tegn.

Specifikation af attribut: **interferens**

Beskrivelse:	Stoffer i prøvematerialet der ved korrekt analysering kan påvirke resultatværdien. Er en del af artiklens medicinske information.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **links**

Beskrivelse:	Ét hyperlink til yderligere information om artiklen. Er en del af artiklens medicinske information.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **listevurdering**

Beskrivelse:	Samlet vurdering af en listes resultatværdier. Anvendes for eksempel ved toleranceundersøgelser, hvor det ikke er tilstrækkeligt at beskrive tolkning og referenceintervaller for de indgående artikler hver for sig. Er en del af artiklens medicinske information og optræder kun for lister.

Format:	Tekststreng på maksimalt 2000 tegn.

Specifikation af attribut: **litteraturreference**

Beskrivelse:	Henvisning til relevant litteratur om ariklen. Er en del af artiklens medicinske information.
Format:	Tekststreng på maksimalt 2000 tegn.

Specifikation af attribut: **omregningsfaktor**

Beskrivelse:	Faktorer for omregning mellem artiklens enhed og andre enheder. Er en del af artiklens tekniske information.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **opbevaring**

Beskrivelse:	Behandling ved opbevaring i længere tid. Er en del af informationen for artiklen prøvetagning.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **pmid**

Beskrivelse:	En PubMed indikator danner sammen med litteraturhenvisning fra *litteraturreference* et direkte links til MEDLINE. Er en del af artiklens medicinske information.
Format:	Et naturligt tal på 7 cifre for PMID eller 8 cifre for IU.

Specifikation af attribut: **prøvemateriale**

Beskrivelse:	Materiale der udtages ved prøvetagningen. Er en del af informationen for artiklens prøvetagning.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **prøvetagning**

Beskrivelse:	Særlige forhold omkring selve prøvetagningen. Er en del af informationen for artiklens prøvetagning.
Format:	Tekststreng på maksimalt 2000 tegn.

Specifikation af attribut: **overReferenceInterval**

Beskrivelse:	Tolkning af værdier over referenceintervallet. Er en del af artiklens medicinske information.
Format:	Tekststreng på maksimalt 2000 tegn.

Specifikation af attribut: **referencetabeller**

Beskrivelse:	Inddeles i fire intervaller. Referenceintervaller er 0,95 intervaller eller andre angivelser af forventede værdier hos raske personer. Der angives intervaller i en eller to kolonner. Terapeutiske intervaller er tilstræbte værdier ved behandling. Beslutningsgrænser er grænser for

	beslutning om klassifikation, diagnose, yderligere undersøgelse eller behandling. Sidste interval er alarmgrænser, som er værdier der kræver akut indsats. Er en del af artiklens medicinske information.
Format:	Tekststreng på maksimalt 2000 tegn.

Specifikation af attribut: **svartid**

Beskrivelse:	Forventet tid på et svar for artikler, som ikke besvares samme eller næste dag. Er en del af artiklens laboratorieinformation.
Format:	Tekststreng på maksimalt 255 tegn.

Specifikation af attribut: **underReferenceInterval**

Beskrivelse:	Tolkning af værdier under referenceintervallet. Er en del af artiklens medicinske information.
Format:	Tekststreng på maksimalt 2000 tegn.

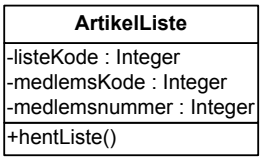
Specifikation af operation: **gemKompendiumData**

Hensigt: Lagrer kompendium Informationen for en bestemt artikel.
Signatur: ArtikelKompendiumData:gemKompendiumData(aKD : ArtikelKompendiumData)
Logisk beskrivelse:
Pre: self.aKD -> notEmpty
ArtikelKompendiumData -> includes(aKD)
Post:
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **hentKompendiumData**

Hensigt: Returnerer kompendium information for en bestemt artikel.
Signatur: ArtikelKompendiumData::hentKompendiumData(kode : Integer) kD : ArtikelKompendiumData
Logisk beskrivelse:
Pre: self.kode -> notEmpty
Post: none
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

9.1.6 ArtikelListe



Figur 15: ArtikelListe klassen.

Klassebeskrivelse: **ArtikelListe**

Angiver om en artikel er en liste og dermed listens medlemmer. Eller om artiklen er et medlem af en liste og dermed af hvilken.

Specifikation af attribut: **listeKode**

Beskrivelse:	Angiver koden for den artikel, der er en liste..
Format:	Naturligt tal på maksimalt 5 cifre.

Specifikation af attribut: **medlemsKode**

Beskrivelse:	Angiver koden på de artikler, der indgår i listen.
Format:	Naturligt tal på maksimalt 5 cifre.

Specifikation af attribut: **medlemssnummer**

Beskrivelse:	Angiver rækkefølgen for listens medlemmer.
Format:	Naturligt tal.

Specifikation af operation: **hentListe**

Hensigt: Henter liste informationen for en artikel. Hvis artiklen er en liste returneres alle medlemmerne og en sandhedsværdi, der indikerer at artiklen er en liste. Er artiklen solitær returneres en tom liste og sandhedsværdien returneres falsk. Hvis artiklen indgår i en liste returneres listeKoden for denne liste og sandhedsværdien returneres falsk.

Signatur: ArtikelListe::hentListe(enKode : Integer) enListe : Set, erListe: Boolean

Logic description:

Pre: enKode->notEmpty

Post: if (self.ArtikelListe->select(ArtikelListe.listeKode=enKode)->notEmpty) then
(erListe=true
enListe=self.liste)
else if (self.ArtikelListe ->
select(ArtikelListe.medlemsKode=enKode)->notEmpty)
then
(erListe=false
enListe=Artikel.listeKode) else

(erListe = false
enListe="")
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

9.1.7 ArtikelSpecialeKobling

Figur 16: ArtikelSpecialeKobling.

Klassebeskrivelse: **ArtikelSpecialeKobling**

Præsenterer de specialer som en artikel er tilknyttet.

Specifikation af attribut: **speciale**

Beskrivelse:	Forkortelse for betegnelsen af det laboratoriespeciale artiklen hører under.
Format:	Tekststreng på 5 tegn.

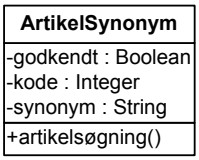
Specifikation af attribut: **prioritet**

Beskrivelse:	En artikel kan være tilknyttet mere end et laboratoriespeciale. Hvert speciale tillægges derfor en prioritering for den pågældende artikel.
Format:	Naturligt tal.

Specifikation af operation: **artikelsøgning**

Hensigt: Returnerer de godkendte artikler, der er omfattet et speciale.
Signatur: ArtikelSpecialeKobling::artikelsøgning()
Logisk beskrivelse:
Pre: kompendiuminfo.godkendt
Post: none
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

9.1.8 ArtikleSynonym



Figur 17: ArtikelSynonym klassen.

Klassebeskrivelse: **ArtikelSynonym**

Indeholder artikler der er forbundet med et givent synonym.

Specifikation af attribut: **godkendt**

Beskrivelse:	Angiver om synonymet er godkendt eller ej.
Format:	Boolean. Hvis sandt er synonymet godkendt.

Specifikation af attribut: **kode**

Beskrivelse:	Angiver koden på den artikel, som skal forbindes med det pågældende synonym.
Format:	Naturligt tal på maksimalt 5 cifre.

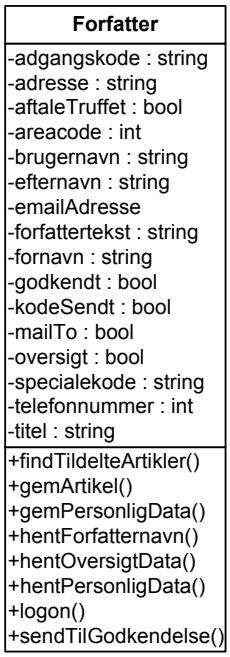
Specifikation af attribut: **synonym**

Beskrivelse:	Omfatter et synonym, der beskriver en eller flere artikler.
Format:	Tekststreng på maksimalt 50 tegn.

Specifikation af operation: **artikelsøgning**

Hensigt: At finde de godkendte artikler, der er repræsenteret ved et givent synonym.
Signatur: ArtikelSynonym::artikelsøgning()
Logisk beskrivelse:
Pre: none
Post: none
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

9.1.9 Forfatter



Figur 18: Forfatter klassen.

Klassebeskrivelse: **Forfatter**	
Klassen beskriver en forfatter på systemet.	
Specifikation af attribut: **adgangskode**	
Beskrivelse:	Sammen med brugernavnet identificerer forfatteren sig med en adgangskode.
Format:	Tekststreng på maksimalt 10 tegn.
Specifikation af attribut: **adresse**	
Beskrivelse:	Angiver forfatterens arbejdsadresse og location.
Format:	Tekststreng på maksimalt 100 tegn.
Specifikation af attribut: **aftaletruffet**	
Beskrivelse:	Angiver dagen hvor der er modtaget tilsagn om at blive oprettet i systemet.
Format:	Dato format af typen dd-mm-yyyy.
Specifikation af attribut: **areacode**	
Beskrivelse:	Angiver forkortelsen for det område forfatteren arbejder indenfor.
Format:	Tekststreng på maksimalt 20 tegn.

Specifikation af attribut: **brugernavn**

Beskrivelse:	En forfatter identificeres af systemet ved et brugernavn.
Format:	Tekststreng på maksimalt 8 tegn.

Specifikation af attribut: **efternavn**

Beskrivelse:	Angiver forfatterens efternavn og eventuelle mellemnavne.
Format:	Tekststreng på maksimalt 50 tegn.

Specifikation af attribut: **emailAdresse**

Beskrivelse:	Angiver forfatterens e-mail adresse.
Format:	Tekststreng på maksimalt 50 tegn.

Specifikation af attribut: **forfattertekst**

Beskrivelse:	Betegner tekst som forfatteren bruger som signatur.
Format:	Tekststreng på maksimalt 50 tegn.

Specifikation af attribut: **fornavn**

Beskrivelse:	Angiver forfatterens fornavn.
Format:	Tekststreng på maksimalt 20 tegn.

Specifikation af attribut: **godkendt**

Beskrivelse:	Angiver om alle forfatterens tildelte koder er godkendte.
Format:	Boolean. Hvis sandt er alle artikler godkendt.

Specifikation af attribut: **kodeSendt**

Beskrivelse:	Angiver dagen hvor adgangskoden er sendt til forfatteren.
Format:	Dato format af typen dd-mm-yyyy.

Specifikation af attribut: **mailTo**

Beskrivelse:	Angiver om forfatterens e-mail må offentliggøres i systemet.
Format:	Boolean. Hvis sandt må adressen offentliggøres.

Specifikation af attribut: **oversigt**

Beskrivelse:	Angiver om forfatteren skal vises på forfatteroversigten i systemet.
Format:	Boolean. Hvis sandt indgår forfatteren i oversigten.

Specifikation af attribut: **specialekode**

Beskrivelse: Note med uddybende information omkring forfatteren.
Format: Tekststreng på maksimalt 50 tegn.

Specifikation af attribut: **telefonnummer**

Beskrivelse: Angiver forfatterens arbejdsnummer.
Format: Naturligt tal på maksimalt 8 cifre.

Specifikation af attribut: **titel**

Beskrivelse: Forfatterens stillingsbetegnelse.
Format: Tekststreng på maksimalt 50 tegn.

Specifikation af operation: **findTildelteArtikler**

Hensigt: Returnerer koden på de artikler, der er tildelt en given forfatter.
Signatur: Forfatter::findTildelteArtikler(enForfatterkode: String) artikler: Set
Logic description:
Pre: enForfatterkode->notEmpty
Post: artikler = self.tildelteArtikler.kode->
select(forfatterkode=enForfatterkode)
Andre operationskald: hentArtikelnavn
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **gemArtikel**

Hensigt: Gemmer artikel og kompendium informationen for en artikel. Ændre samtidig artiklens status til ikke længere at være godkendt. Datoen og forfatteren for revideringen ændres ligeledes.
Signatur: Forfatter::gemArtikel(a : Artikel, aKD : ArtikelKompendiumData)
Logisk beskrivelse:
Pre: self.a->notEmpty
a.godkendt = false
a.revideret = today.oclIsTypeOf(Date)
a.revideretAf=a.forfatterkode
Post: none
Andre operationskald: Artikel.gemArtikelData
Events transmitteret til andre objekter: none
Attributes set: godkendt, revideret, revideretAf
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **gemPersonligData**

Hensigt: Gemmer en eksisterende forfatters personlige data.
Signatur: Forfatter::gemPersonligData(forfatterData:Forfatter)
Logic description:
Pre: forfatterData.brugernavn->notEmpty
forfatterData.adgangskode->notEmpty
Forfatter->exists(brugernavn = forfatterData.brugernavn)
Post: Forfatter->includes(forfatterData)
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: raises InvalidForfatterException
Non-functional requirements: none

Specifikation af operation: **hentForfatternavn**

Hensigt: Returnerer fornavn og efternavn på en eksisterende forfatter.
Signatur: Forfatter::hentForfatternavn(etBrugernavn : String)
forfatternavn : String
Logic description:
Pre: self.etBrugernavn->notEmpty
Forfatter->exists(brugernavn=etBrugernavn)
Post: forfatternavn = Forfatter.fornavn + Forfatter.efternavn
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **hentOversigtsdata**

Hensigt: Returnerer dele af forfatterens data til brug på oversigten.
Signatur: Forfatter::hentOversigtsdata(forfatterkode: String)
fornavn : String, efternavn : String, signatur : String,
publicEmail : Boolean
Logic description:
Pre: forfatterkode->notEmpty
Post: fornavn=Forfatter.fornavn->select(brugernavn
=forfatterkode)
efternavn = Forfatter.efternavn->select(brugernavn
=forfatterkode)
signatur = Forfatter.forfattertekst->select(brugernavn
=forfatterkode)
publicEmail = Forfatter.mailTo->select(brugernavn
=forfatterkode)
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none

Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **hentPersonligData**

Hensigt: Returnerer data knyttet til en eksisterende forfatter.
Signatur: Forfatter::hentPeronligData(etBrugernavn : String) enForfatter : Forfatter
Logic description:
Pre: etBrugernavn->notEmpty
Forfatter->exists(brugernavn=etBrugernavn)
Post: enForfatter = Forfatter->select(brugernavn=etBrugernavn)
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **logon**

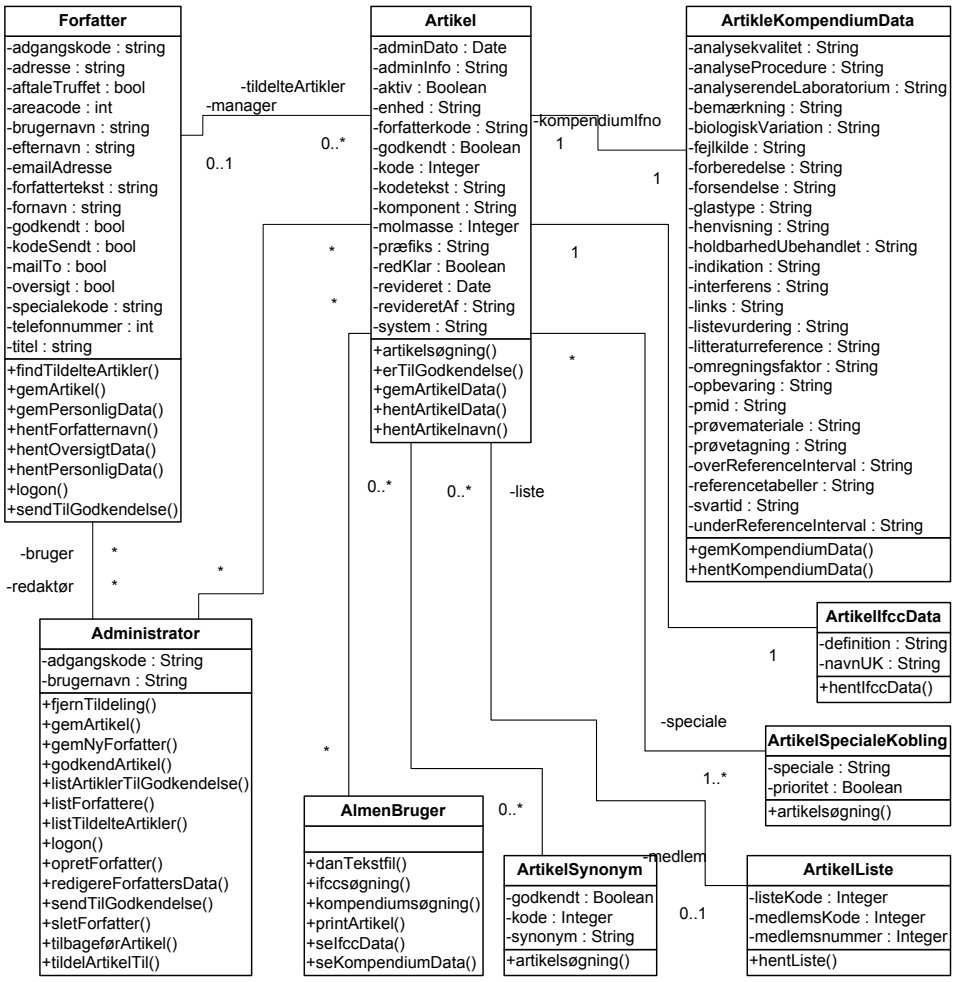
Hensigt: Kontrollerer bruger ud fra et bruger navn og en adgangskode.
Signatur: Forfatter::logon(etBrugernavn : String, enAdgangskode : String) kontrolOk : Boolean
Logic description:
Pre: etBrugernavn->notEmpty
enAdgangskode->notEmpty
Post: kontrolOk = Forfatter.allInstances->forAll(f | f.brugernavn = etBrugernavn and f.adgangskode = enAdgangskode)
Andre operationskald: self.hentOversigtsdata, self.findTildelteArtikler
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **sendTilGodkendelse**

Hensigt: Håndterer at artiklen ændrer status så den er sendt til godkendelse og at dens information samtidig bliver gemt.
Signatur: Forfatter::sendTilGodkendelse(a : Artkel, aKD : ArtikelKompendiumData)
Logisk beskrivelse:
Pre: self.a->notEmpty
self.aKD->notEmpty
self.a.godkendt = false
self.a.redKlar = true
Post: none
Andre operationskald: Artikel.gemArtikelData
Events transmitteret til andre objekter: none
Attributes set: godkendt, redKlar

Response to exceptions: none
Non-functional requirements: none

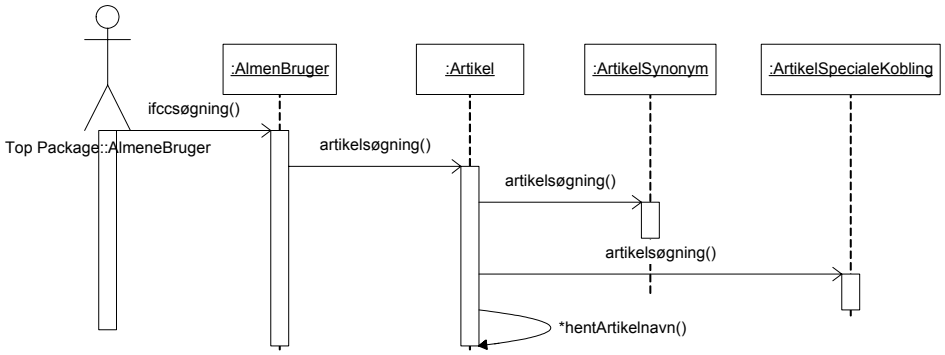
9.2 Endelig klassediagram for LabInfo



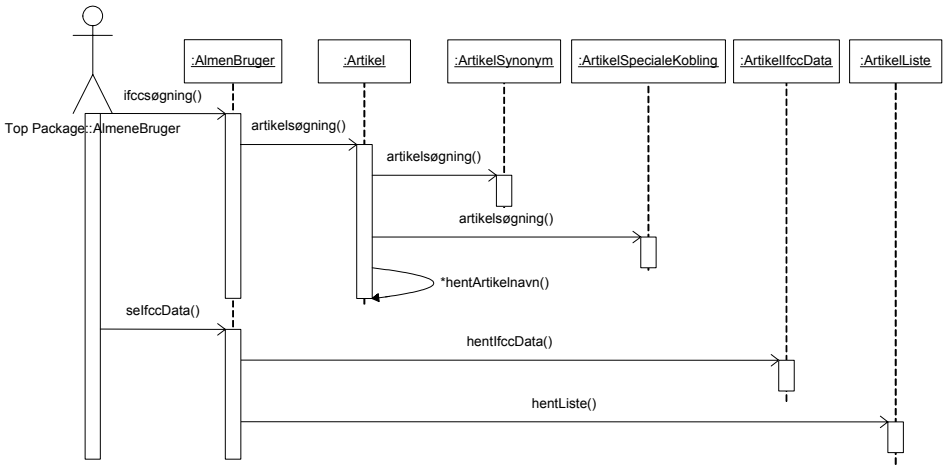
Figur 19: Det endelige klasse diagram for LabInfo med relation mellem klasserne.

10 Sekvens diagrammer

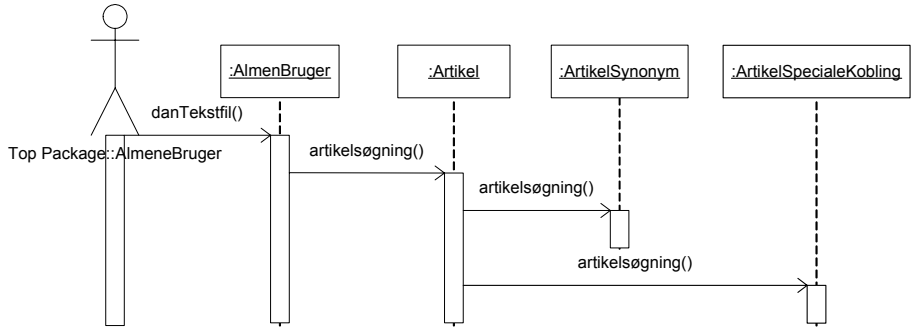
10.1 AlmeneBruger



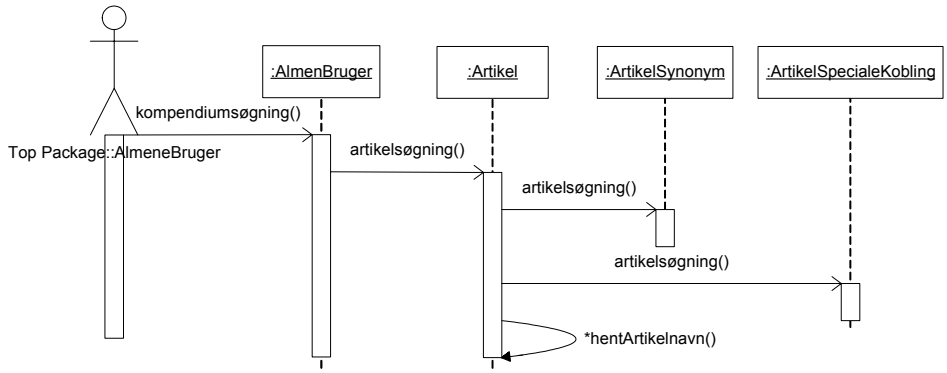
Figur 20: Sekvensdiagram for IFCCSøgning



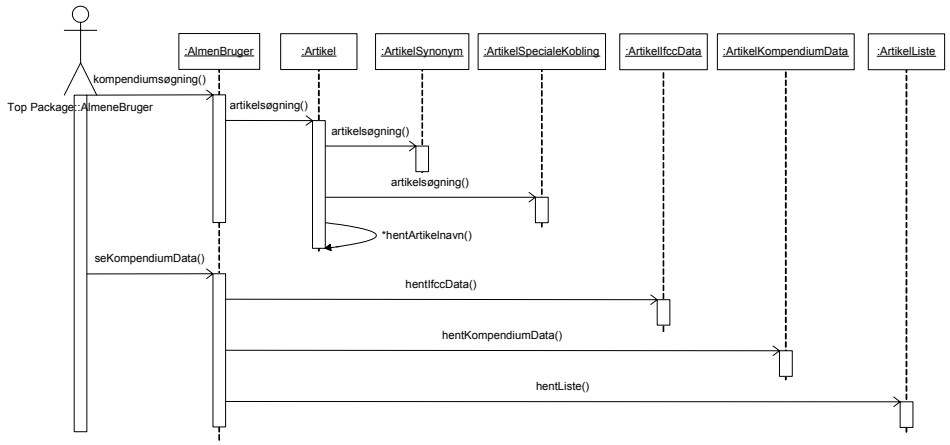
Figur 21: Sekvensdiagram for Se IFCC-IUPAC information.



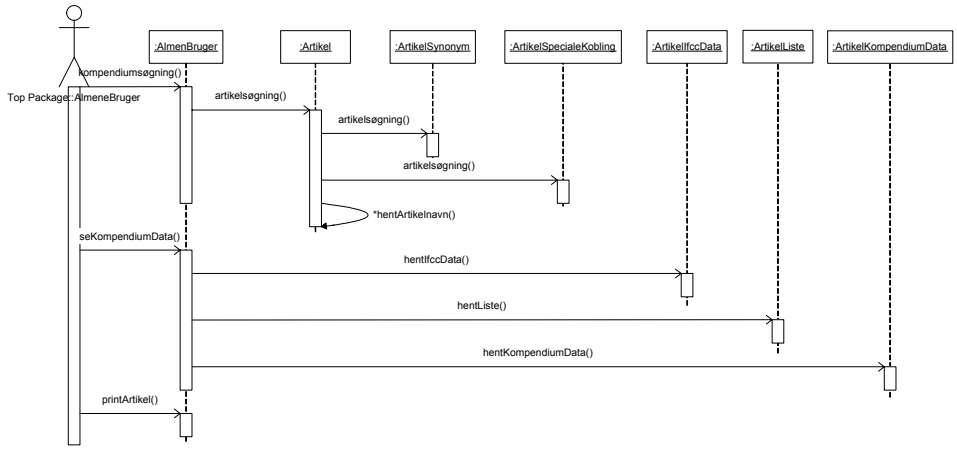
Figur 22: Sekvensdiagram for Download søgeresultat.



Figur 23: Sekvensdiagram for KompendiumSøgning.

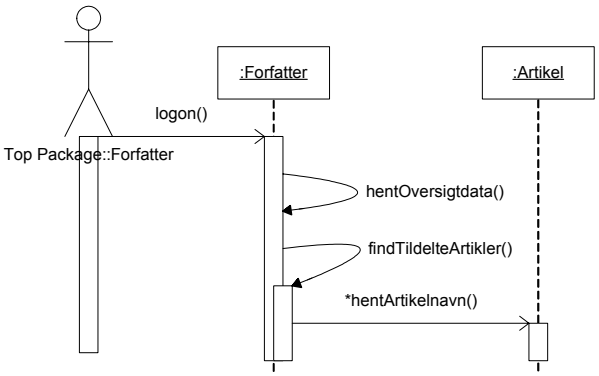


Figur 24: Sekvensdiagram for Se Kompendium information.

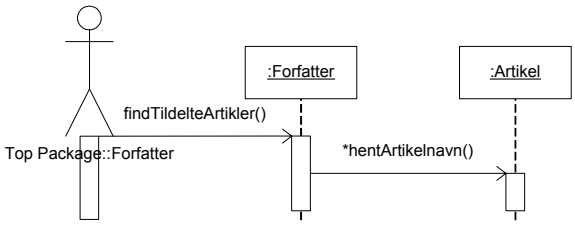


Figur 25: Sekvensdiagram for Print artikel.

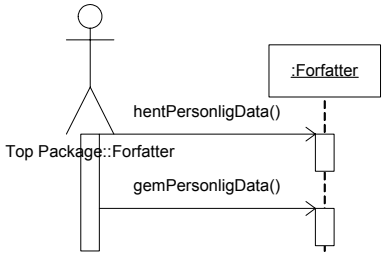
10.2 Forfatter



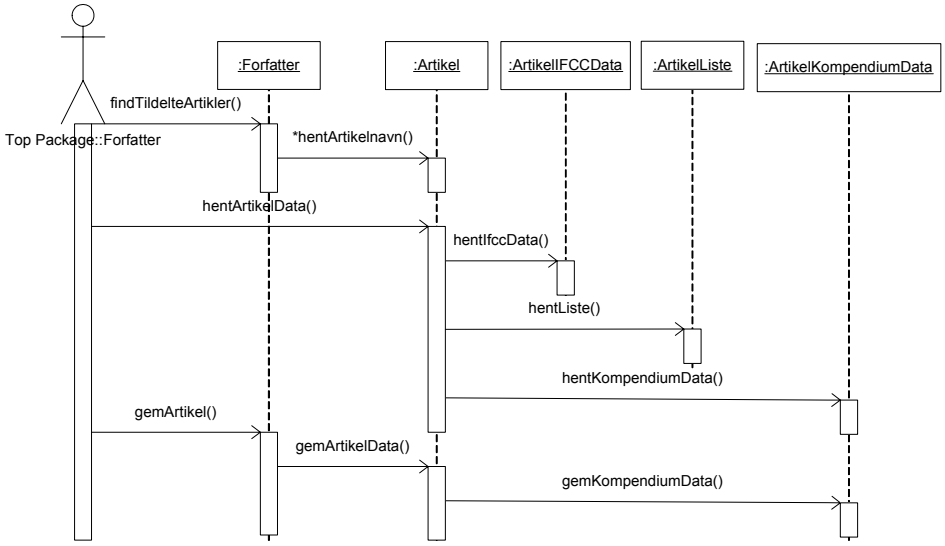
Figur 26: Sekvensdiagram for Logon.



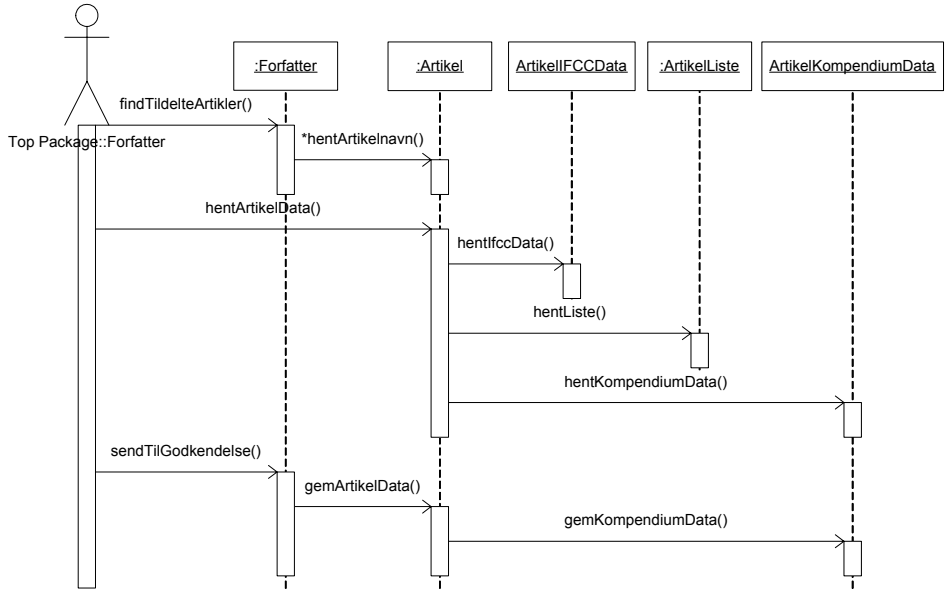
Figur 27: Sekvensdiagram for Se artikler tildelt forfatter.



Figur 28: Sekvensdiagram for Redigere personligdata.

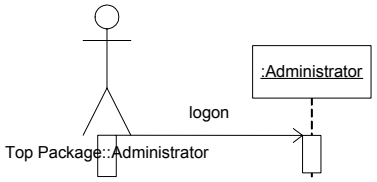


Figur 29: Sekvensdiagram for Redigere artikel.

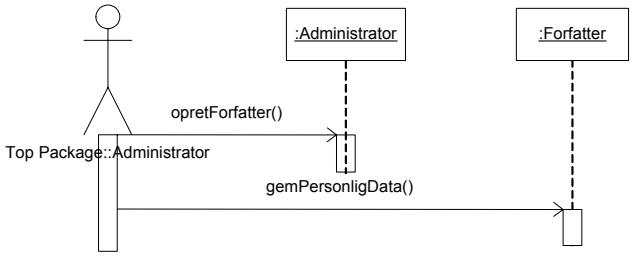


Figur 30: Sekvensdiagram for Send artikel til godkendelse.

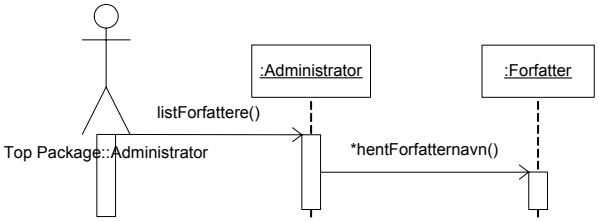
10.3 Administrator



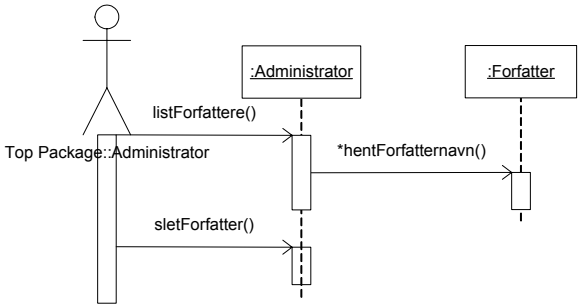
Figur 31: Sekvensdiagram for Logon.



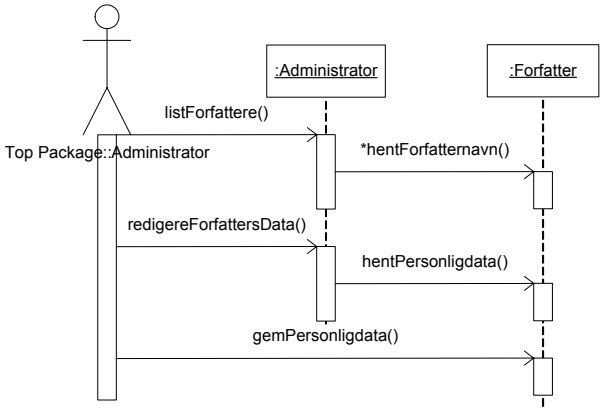
Figur 32: Sekvensdiagram for Opret forfatter.



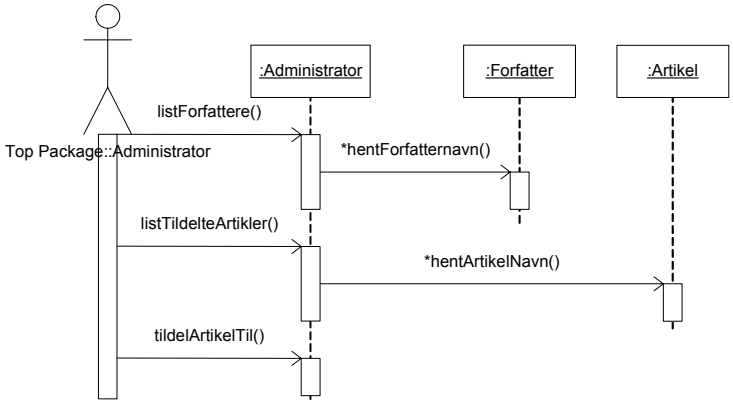
Figur 33: Sekvensdiagram for Oversigt med forfattere.



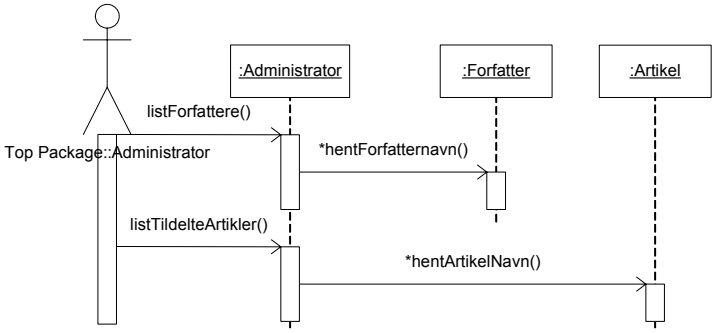
Figur 34: Sekvensdiagram for Slet forfatter.



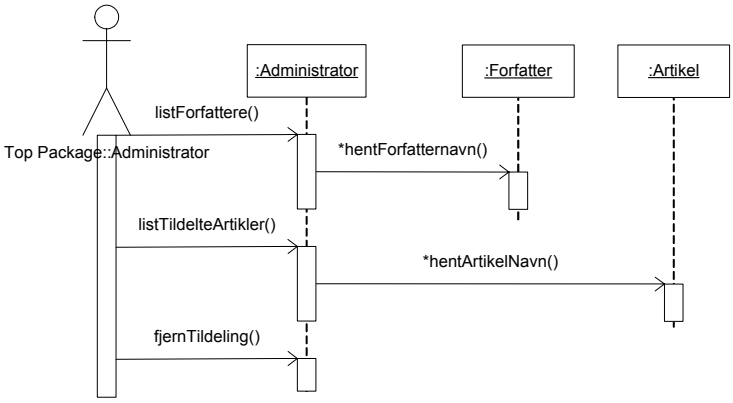
Figur 35: Sekvensdiagram for Redigere personligdata.



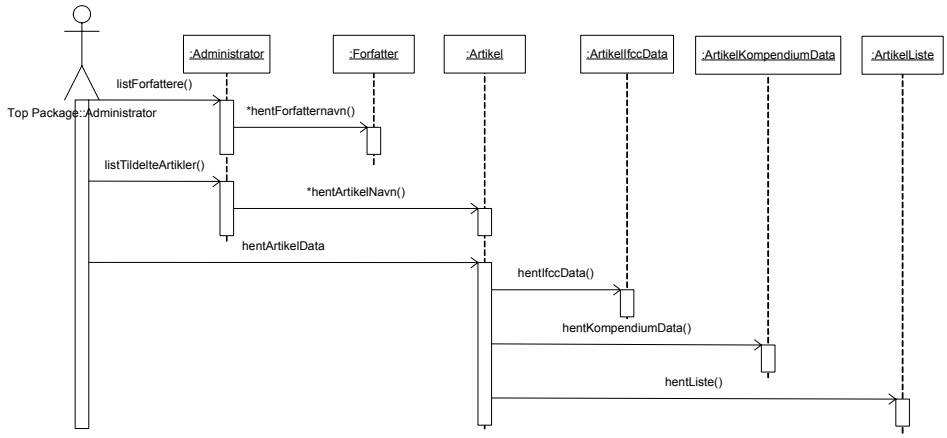
Figur 36: Sekvensdiagram for Tildel artikel til forfatter.



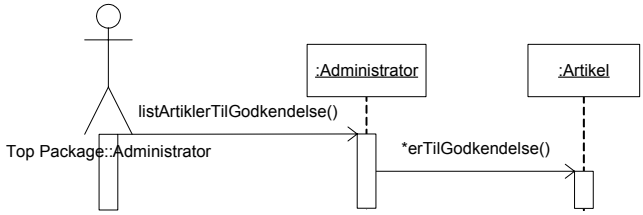
Figur 37: Sekvensdiagram for Se artikler tildelt forfatter.



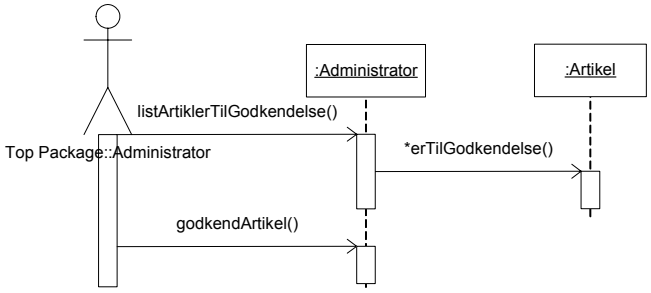
Figur 38: Sekvensdiagram for Fjern artikel fra forfatter.



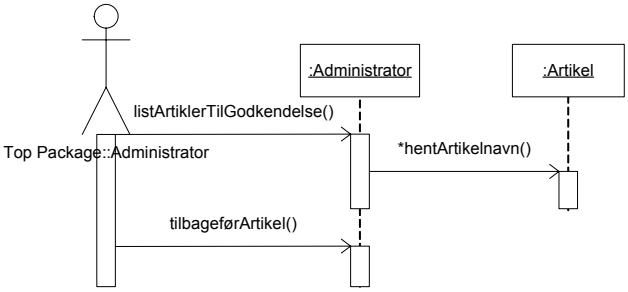
Figur 39: Sekvensdiagram for Se artikel information.



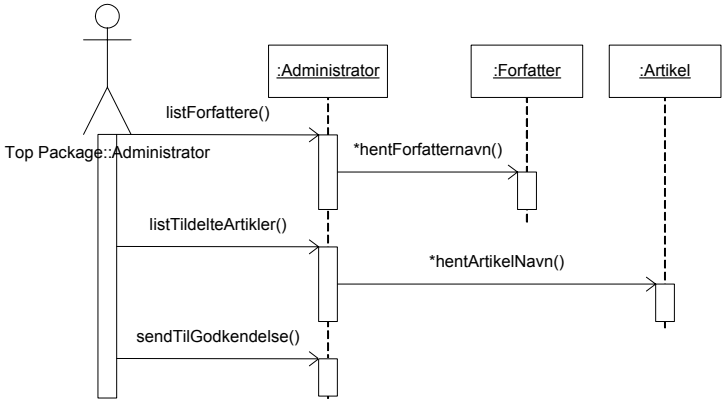
Figur 40: Sekvensdiagram for Se artikler sendt til godkendelse.



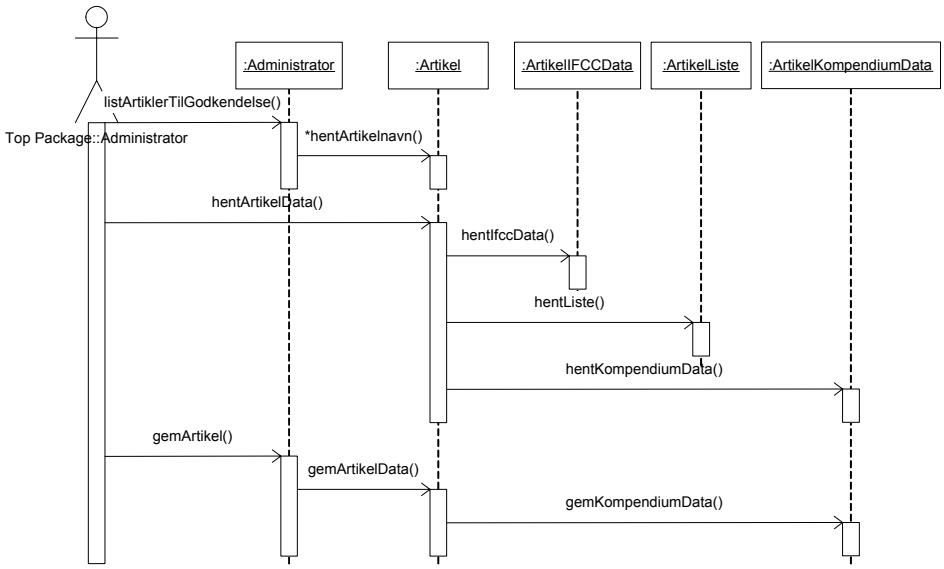
Figur 41: Sekvensdiagram for Godkend artikel.



Figur 42: Sekvensdiagram for Tilbageføre artikel.



Figur 43: Sekvensdiagram for Send artikel til godkendelse.



Figur 44: Sekvensdiagram for Redigere artikel.

Kapitel 3

11 Decentrale LabInfo systemer

Som tidligere beskrevet eksisterer der i dag ét decentralt system, LabInfoAUH. I de næste to afsnit vil LabInfoAUH blive analyseret med henblik på at opstille generelle krav for decentrale systemer. Her tænkes på de krav, som de decentrale systemer må opfylde for at kunne kommunikere med det centrale system.

11.1 Analyse

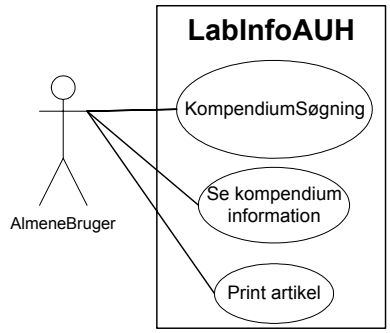
Af programbeskrivelsen LabInfoAUH (2003) fremgår det, at systemets funktion er meget lig det centrale system. Systemet skal være i stand til at håndtere og formidle kompendium information for en række artikler. Dette skal ske på tilsvarende måde som for det centrale system. LabInfo indeholder kun artikler som er af specifik interesse. Denne mængde vil altid være en ægte delmængde af artiklerne fra det centrale system.

For at kunne ændre og administrere ændringerne for disse artikler er der knyttet forskellige brugerniveauer til LabInfoAUH. Disse er ifølge LabInfoAUH (2003) henholdsvis en AlmenBruger, en Forfatter og en Admininstraor, som for det centrale system.

Af figur 1 i LabInfoAUH (2003) fremgår det at funktionaliteten for brugerne afviger i forhold til det centrale system. Kravspecifikationen for de enkelte aktører er beskrevet i næste afsnit.

11.2 Kravspecifikation

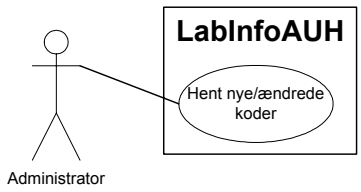
11.2.1 Simple use cases



Figur 45: Simpel use case for aktøren almenBruger

Aktøren forfatter har samme funktionalitet som ved det centrale system, se figur 5. Dog er enkelte funktioner blevet udbygget, hvilket vil fremgå af senere use case beskrivelser.

Det samme gør sig gældende for aktøren administrator. Denne har dog en funktionalitet udover dem i figur 6. Følgende figur viser denne funktionalitet.



Figur 46: Simpel use case for aktøren administrator.

11.3 Use case beskrivelser

Ovenstående use cases beskrives her ved hjælp af aktørens handlinger og systemets svar. Enkelte funktioner afviger ikke i forhold til det centrale system. For disse vil der blive henvist til beskrivelsen for det centrale system.

Beskrivelserne opstilles på baggrund af informationen fra LabInfoAUH (2003).

11.3.1 AlmenBruger

KompendiumSøgning

Afviger ikke i forhold til det centrale system, se afsnit 6.1.4.

Se kompendium information

Aktør handling		System svar	
1	Brugeren trykker *Søg* under menupunktet Laboratoriehåndbog.	2	Viser en formular over mulige søgekriterier inden for Kompendium 2000.
3	Brugeren præciserer sin søgning og trykker *Søg*.	4	Søger i databasen efter mulige godkendte artikler og viser en oversigt med navnet på disse. Artikler fundet via deres synonym vises separat.
5	Brugeren vælger en artikel fra oversigten ved at klikke på artiklens navn.	6	Viser artiklens navn, IFCC-IUPAC information og liste information. Kompendium informationen opdeles i tre kategorier: medicinsk, reference

			intervaller og laboratorium information. Alle navne på tilknyttede hospitaler vises.
5	Brugeren vælger *Med. Iinfo.*	6	Viser artiklens navn, medicinsk information fra kompendium 2000 og liste informationen.
7	Brugeren vælger *Ref. Interval.*	8	Viser artiklens navn, information om reference intervaller fra kompendium.
9	Brugeren vælger *Lab. Info.*	10	Viser artiklens navn, Laboratorium information fra kompendium 2000.
11	Brugeren vælger navnet på et tilknyttet hospital.	12	Viser kompendium information fra det eller de valgte hospitaler i stedet for kompendium 2000 informationen.

Print artikel

Aktør handling		System svar	
1	Brugeren trykker *Søg* under menupunktet Laboratoriehåndbog.	2	Viser en formular over mulige søgekriterier inden for Kompendium 2000.
3	Brugeren præciserer sin søgning og trykker *Søg.*	4	Søger i databasen efter mulige godkendte artikler og viser en oversigt med navnet på disse. Artikler fundet via deres synonym vises separat.
5	Brugeren vælger en artikel fra oversigten ved at klikke på artiklens navn.	6	Viser artiklens navn, IFCC-IUPAC information og liste information. Kompendium informationen opdeles i tre: medicinsk, reference intervaller og laboratorium information.
7	Brugeren vælger *Print artikel.*	8	Viser en printvenlig side med al information for artiklen. Hvis brugeren har valgt et eller flere hospitaler inden punkt 7 vil informationen fra de enkelte hospitaler blive vist i stedet for kompendium 2000 informationen.

11.3.2 Forfatter

Forfatterens funktioner svare til dem beskrevet for det centrale system i afsnit 6.2. Enkelte af disse afviger lidt og vil af den grund blive beskrevet i dette afsnit. For de øvrige henvises til afsnit 6.2.

Se artikler tildelt forfatter

Aktør handling		System svar	
1	Forfatteren logger på systemet.	2	Viser startsiden med en liste over tildelte artikler og deres status. Artiklerne opdeles efter om forfatteren er ansvarlig for artiklens generelle eller lokale kompendium information.

Redigere artikel

Aktør handling		System svar	
1	Fra startsiden med tildelte artikler vælges den artikel, der skal ændres. Artikler sendt til godkendelse kan ikke vælges.	2	Systemet viser artiklens navn og IFCC-IUPAC information. Informationen for Kompendium 2000 vises i en formular. Hvis forfatteren ikke er ansvarlig for den generelle kompendium 2000 kan der ikke ændres i denne formular. Er forfatteren ansvarlig for lokalt kompendium data vises også formular til udfyldning af dette data.
3	Forfatteren opdaterer formularen og trykker *Gem artikel.*	4	Systemet beder forfatteren om at bekræfte valget. Hvis artiklen er godkendt informeres forfatteren om at artiklen ikke er godkendt efter endt handling.
5	Forfatteren trykker *Ok*.	6	Systemet gemmer kompendium data fra formularen og ændre artiklens status til ikke at være godkendt. Viser informationen for artiklen.

7	Forfatteren trykker Tilbage til oversigten.	8	Systemet vender tilbage til forfatterens startside.

Send til godkendelse

Aktør handling		System svar	
1	Fra startsiden med tildelte artikler vælges den artikel, der skal ændres. Artikler sendt til godkendelse kan ikke vælges.	2	Systemet viser artiklens navn og IFCC-IUPAC information. Informationen for Kompendium 2000 vises i en formular. Hvis forfatteren ikke er ansvarlig for den generelle kompendium 2000 kan der ikke ændres i denne formular. Er forfatteren ansvarlig for lokalt kompendium data vises også formular til udfyldning af dette data.
3	Forfatteren opdaterer formularen og trykker *Godkend artikel.*	4	Systemet beder forfatteren om at bekræfte valget.
5	Forfatteren trykker *Ok*.	6	Gemmer kompendium data fra formularen, ændre artiklens status til ikke at være godkendt og sender den til redaktøren. Viser informationen for artiklen.
7	Forfatteren trykker *Tilbage til oversigten*.	8	Systemet vender tilbage til forfatterens startside.

11.3.3 Administrator

Funktionaliteten svarer til den beskrevet i afsnit 6.3 for det centrale system, hvorfor der henvises til dette afsnit. Enkelte funktioner afviger dog og vil blive gennemgået nedenfor sammen med den ekstra funktionalitet vist i 6.3.

Hent nye/ændrede koder

Aktør handling		System svar	
1	Administratoren vælger *Hent koder*.	2	Systemet henter koderne på alle nye eller ændrede artikler fra Sundhedsstyrelsen. Disse koder vises i en liste sammen med tidligere hentede koder.

			Denne liste dækker over alle ubehandlet koder. Koderne er opdelt efter om de dækker over nye eller ændrede artikler.

Opret artikel

Aktør handling		System svar	
1	Administratoren vælger *Hent koder*.	2	Systemet henter koderne på alle nye eller ændrede artikler fra Sundhedsstyrelsen. Disse koder vises i en liste sammen med tidligere hentede koder. Denne liste dækker over alle ubehandlet koder. Koderne er opdelt efter om de dækker over nye eller ændrede artikler.
3	Fra listen med ubehandlede koder vælges en kode over nye artikler.	4	Systemet henter og viser artiklens tilhørende artikelnavn og ifcc-iupac information fra Sundhedsstyrelsen.
5	Administratoren vælger *gem artikel*.	6	Systemet gemmer artiklen og den ifcc-iupac information i databasen. Vender tilbage til punkt 2. Koden er nu behandlet og optræder ikke i listen over ubehandlede koder.

Alternativ handling:

Punk 5. Administratoren vælger *Afvis kode*. I punkt 6 vil systemet fjerne koden fra listen over ubehandlede koder og vende tilbage til punkt 2.

Gem ifcc-iupac information

Aktør handling		System svar	
1	Administratoren vælger *Hent koder*.	2	Systemet henter koderne på alle nye eller ændrede artikler fra Sundhedsstyrelsen. Disse koder vises i en liste sammen med tidligere hentede koder. Denne liste dækker over alle ubehandlet koder. Koderne er opdelt efter om de dækker over nye eller ændrede artikler.
3	Fra listen med ubehandlede koder vælges en kode over ændrede artikler.	4	Systemet viser et link til Sundhedsstyrelsen for information på den

			pågældende artikel. Samtidig bliver administratoren bedt om at bekræfte at han har set denne information.
5	Administratoren vælger *koden er opdateret*.	6	Systemet fjerner koden fra listen over ubehandlede koder. Og vender tilbage til punkt 2.

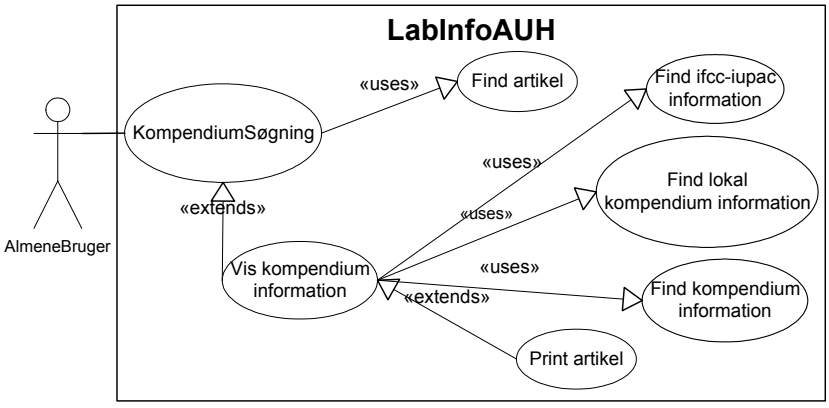
Tildel artikel til forfatter

Aktør handling		System svar	
1	Administratoren vælger *Ret forfatter*.	2	Viser en oversigt med forfattere sorteret efter forbogstavet i fornavn eller efternavn.
3	Administratoren finder forfatteren i oversigten og vælger *Se/ret artikelliste*.	4	Viser en liste over forfatterens tildelte artikler.
5	Administratoren indskriver koden på en artikel, vælger ansvarsniveau og trykker *Tilføj*.	6	Viser om artiklen eventuelt er tildelt anden forfatter og dialog om at bekræfte valg.
7	Administratoren vælger *Ok*.	8	Tildeler artiklen til forfatteren og vender tilbage til oversigt med tildelte artikler, punkt 4.

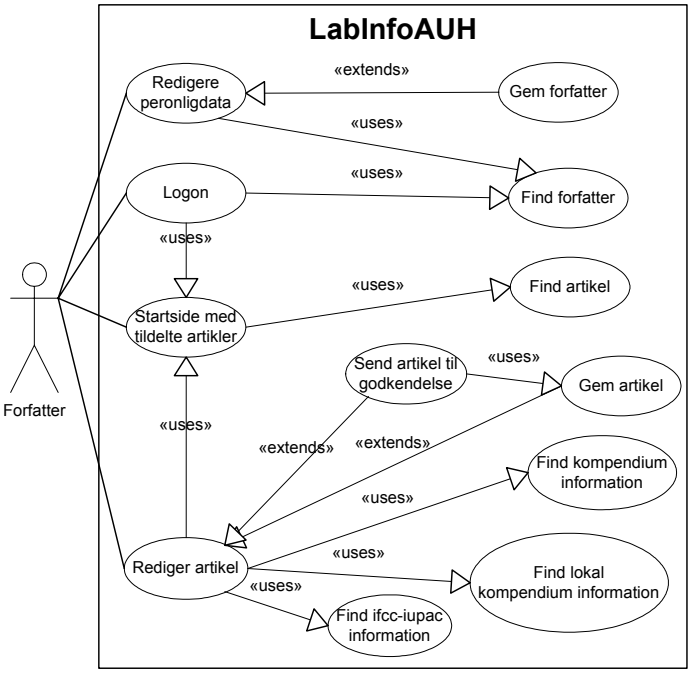
Se artikler tildelt forfatter

Aktør handling		System svar	
1	Administratoren vælger *Ret forfatter*.	2	Viser en oversigt med forfattere sorteret efter forbogstavet i fornavn eller efternavn.
3	Administratoren finder forfatteren i oversigten og vælger *Se/ret artikelliste*.	4	Viser en liste over forfatterens tildelte artikler. Artiklerne er opdelt efter om forfatteren er ansvarlig for artiklens generelle eller lokale data.

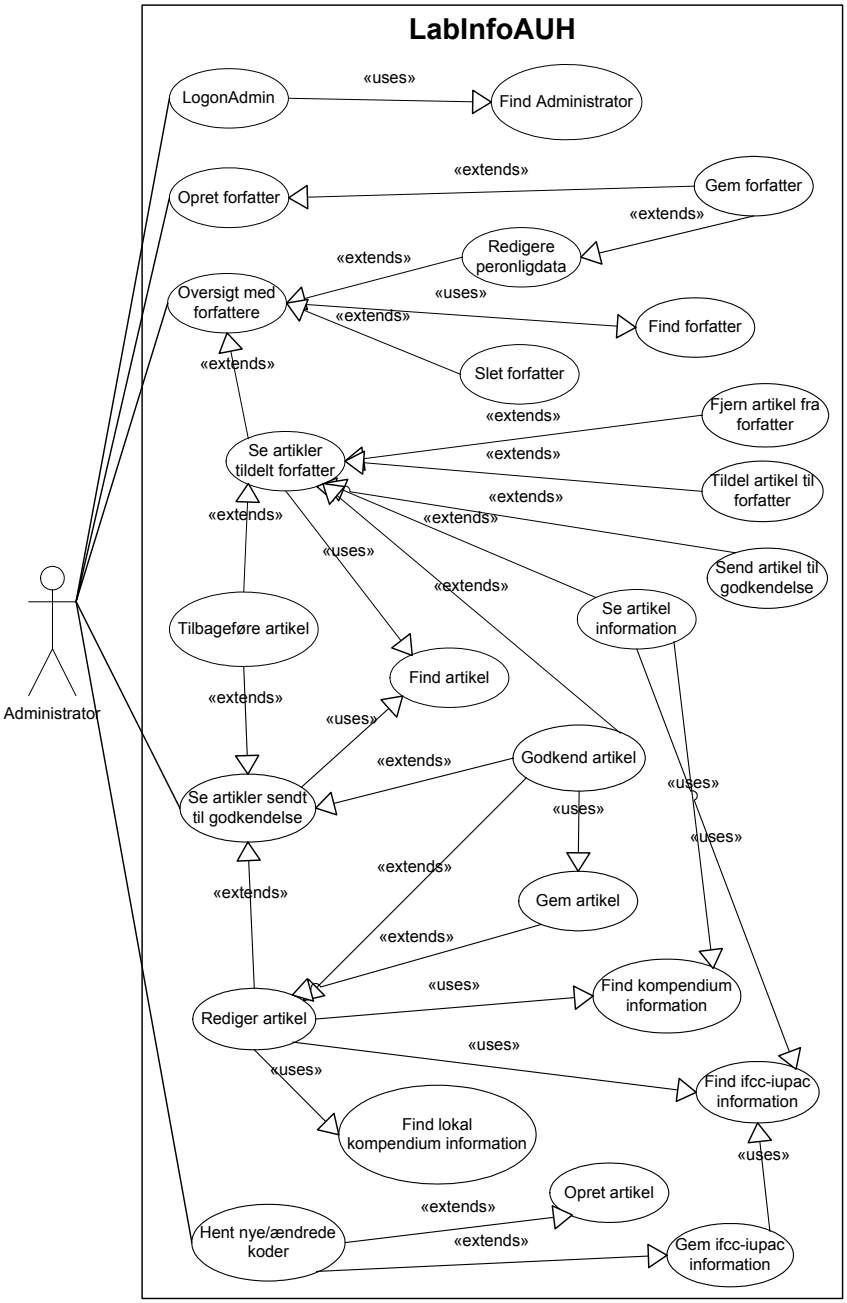
11.4 Endelig use case diagrammer



Figur 47: Endelig use case diagram for almenBruger.



Figur 48: Endelig use case diagram for aktøren forfatter.



Figur 49: Endelig use case diagram for aktøren administrator.

11.5 Aktørbeskrivelse

11.5.1 Administrator

En administrator er ansvarlig for registrerede forfattere og artikler i det decentrale system. Det indebærer oprettelse og vedligeholdelse af såvel forfattere som artikler. De står for tildelingen af artikler til de enkelte forfattere. Herunder om den pågældende forfatter skal være ansvarlig for den generelle Kompendium 2000 information eller den lokale kompendium information, for det hospital forfatteren er tilknyttet. Artikler, der er sendt til godkendelse af forfatterne, er administratoren ansvarlig for at godkende.

Administratoren er ansvarlig for at hente og dermed oprette nye artikler publiceret af Sundhedsstyrelsen. Ligeledes skal ændrede artikler hos Sundhedsstyrelsen gennemgås for nødvendig opdatering af den tilhørende artikel i det decentrale system.

Administratoren afgør om en artikel er udgået, offentligt tilgængelig eller ej.

11.5.2 Forfatter

En forfatter er ansvarlig for de artikler, der er ham tildelt. Ansvaret kan være tillagt en artikels generelle kompendium 2000 information, lokale kompendium information eller begge informationsområder. Forfatteren kan kun tillægges det lokale ansvar for det hospital, han er blevet tilknyttet af administratoren. Ændrer forfatteren i en tildelt artikels indhold er vedkomne ansvarlig for, at den bliver sendt til godkendelse hos administratoren.

Forfatteren er ansvarlig for sine personlige data.

11.5.3 AlmenBruger

Denne aktør er ikke tillagt systemmæssigt ansvar. Almene brugere har adgang til de artikler, der er offentliggjorte af administratoren. Hvis den generelle kompendium information ikke er offentliggjort vil al øvrig lokal kompendium information ikke være tilgængelig for den almene bruger.

11.6 Klassespecifikation

På baggrund af informationen fra use case beskrivelserne over LabInfoAUH opstilles følgende klasser og efterfølgende en beskrivelse af hver enkelt. De klasser, der stemmer overens med dem præsenteret for det centrale system, vil ikke blive nævnt i det efterfølgende. Der henvises i de tilfælde til afsnit 9.

- Administrator
- AlmenBruger

- Artikel
- AtikelIFCCData
- ArtikelKompendiumData
- ArtikelKomplokalData
- ArtikelListe
- ArtikelSpecialeKobling
- ArtikelSynonym
- Forfatter
- UbehandletKoder

11.6.1 Administrator

Figur 50: Administrator klassen for LabInfoAUH.

Klassebeskrivelse: **Administrator**

Der kan være en eller flere administratorer til at håndtere administration på systemet. En administrator er beskrevet ved denne klasse.

Specifikation af operation: **opretArtikel**

Hensigt: Sikre at artikel bliver registret som afvis.
Signatur: Administrator::afvisArtikel(enKode: Integer)
Logisk beskrivelse:
Pre: none
Post: not self.Artikel->exists(kode=enKode)

Andre operationskald: UbehandletKoder.artikelAfvist
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **fjernTildeling**

Hensigt: Fjerner en forfatter fra en artikel, hvis artiklen ikke er godkendt eller til godkendelse.
Signatur: Administrator::fjernTildeling(enKode : Integer, enForfatter: String, lokal: String)
Logic description:
Pre: enKode->notEmpty
self.tildelteArtikler->exists(kode=enKode and forfatterkode=enForfatter and areaName=lokal and godkendt=false and redKlar=false)
Post: not self.tildelteArtikler->exists(kode=enKode and forfatterkode=enForfatter and areaName=lokal)
Andre operationskald: none
Events transmitted to other objects: none
Attributes set: none
Response to exceptions: raises InvalidArtikelStatusException
Non-functional requirements: none

Specifikation af operation: **gemArtikel**

Hensigt: Overskriver en eksisterende artikel med ny data.
Signatur: Administrator::gemArtikel(enArtikel : Artikel, enKompendiumData : ArtikelKompendiumData, lokalKompData : ArtikelLokalKompData)
Logisk beskrivelse:
Pre: self.enArtikel->notEmpty
Post: none
Andre operationskald: Artikel.gemArtikelData
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **godkendArtikel**

Hensigt: Godkender en artikel ved at ændre artiklens status til godkendt. Artiklen tilbageføres samtidig til forfatteren.
Signatur: Administrator::godkendArtikel(enKode : String, lokal: String)
Logisk beskrivelse:
Pre: enKode->notEmpty
Artikel->exists(kode=enKode and areaName=lokal)

Post: Artikel->select(a :Artikel | kode=enKode and areaName=lokal)->a.godkendt=true
Artikel->select(a :Artikel | kode=enKode and areaName=lokal)->a.redKlar=false

Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **hentKoder**

Hensigt: Henter koderne på nye eller ændrede artikler fra Sundhedsstyrelsen. Samt koder på endnu ikke behandlede artikler.
Signatur: Administrator::hentKoder()
Logisk beskrivelse:
Pre: none
Post: none
Andre operationskald: UbehandletKoder.hentUbehandlede
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **listTildelteArtikler**

Hensigt: Henter koder for alle tildelte artikler for en enkelt forfatter.
Signatur: Administrator::listTildelteArtikler(etBrugernavn : String, lokal: String)
Logisk beskrivelse:
Pre: self.tildelteArtikler.kode->select(a:Artikel | a.brugernavn=etBrugernavn and a.areaName=lokal)
Post: none
Andre operationskald: Artikel.hentArtikelNavn
Events transmitteret til andre objekter:
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **opretArtikel**

Hensigt: At hente artikelnavn, artikelSpecialeKobling og artikelListe fra Sundhedsstyrelsen. Denne datamængde gemmes.
Signatur: Administrator::opretArtikel(enKode: Integer)
Logisk beskrivelse:
Pre: not self.Artikel->exists(kode=enKode)
Post: self.Artikel->exists(kode=enKode)
Andre operationskald: UbehandletKoder.artikelOprettet

Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **opdatereArtikel**

Hensigt: Sikre at artikel bliver registret som opdateret.
Signatur: Administrator::opdatereArtikel()
Logisk beskrivelse:
Pre: none
Post: none
Andre operationskald: UbehandletKoder.artikelOpdateret
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **sendTilGodkendelse**

Hensigt: Sender en artikel til godkendelse ved at ændre dens status.
Signatur: Administrator::sendTilGodkendelse(enArtikelKode : Integer, lokal: String)
Logisk beskrivelse:
Pre: self.artikel->exists (kode=enArtikelKode and areaName=lokal)
Post: self.artikel->select(kode=enArtikelKode and areaName=lokal and redKlar=True and godkendt=False)
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **tilbageførArtikel**

Hensigt: Sender en artikel tilbage til en forfatter ved at ændre artiklens status.
Signatur: Administrator::tilbageførArtikel(enArtikelKode : Integer, lokal:String)
Logisk beskrivelse:
Pre: self.artikel->exists (kode=enArtikelKode and areaName=lokal and not godkendt and redKlar)
Post: self.artikel->exists (kode=enArtikelKode areaName=lokal and not redKlar and not godkendt)
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: raises InvalidArtikelStatusException

Non-functional requirements: none

Specifikation af operation: **tildelArtikelTil**

Hensigt: Tildeler en artikel til en forfatter. Det kontrolleres om artiklen i forvejen er tildelt en anden forfatter. Er det tilfældet returneres navnet på denne forfatter.

Signatur: Administrator::tildelArtikelTil(enForfatterkode : String, enArtikelkode : String, lokal: String) : String

Logisk beskrivelse:

Pre: self.bruger->exists(brugernavn=enForfatterkode
areaName=lokal)
if(self.artikel->select(a: Artikel | a.kode=enArtikelkode and a.forfatterkode <> null and areaName=lokal) then
(result = a.forfatterkode)
else (result="")
endif

Post: self.artikel->exists(kode=enArtikelkode and forfatterkode=enForfatterkode and areaName=lokal)

Andre operationskald: none

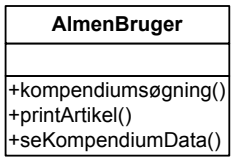
Events transmitteret til andre objekter: none

Attributes set: none

Response to exceptions: none

Non-functional requirements: none

11.6.2 AlmenBruger



Figur 51: AlmenBruger klassen for LabInfoAUH

Den eneste operation, der afviger er følgende.

Specifikation af operation: **seKompendiumData**

Hensigt: At præsentere en artikels IFCC, Kompendium og liste information.

Signatur: AlmenBruger::seKompendiumData(enArtikel : Artikel)

Logisk beskrivelse:

Pre: self.artikel->includes(enArtikel)

Post: none

Andre operationskald: ArtikelIFCCData.hentIFCCData, ArtikelKompendiumData.hentKompendiumData, ArtikelLokalKompData.hentLokalKompData, ArtikelListe.hentListe

Events transmitteret til andre objekter: none

Attributes set: none
Response to exceptions: none
Non-functional reqsuirements: none

11.6.3 Artikel

Figur 52: Artikel klassen for LabInfoAUH.

Specifikation af attribut: **areaName**

Beskrivelse:	Angiver det lokalenavn for det hospital som artiklens kompendium information hører under.
Format:	Tekststreng på maksimalt 50 tegn.

Specifikation af operation: **gemArtikelData**

Hensigt: Gemmer informationen for en artikel.
Signatur: Artikel::gemArtikelData(enArtikel : Artikel, aKD : ArtikelKompendiumData, aLKD: ArtikelLokalKompData)
Logisk beskrivelse:
Pre: self.Artikel->exists(kode = enArtikel.kode and areaName=enArtikel.areaName)
Post: self.Artikel->includes(enArtikel)
Andre operationskald: ArtikelKompendiumData.gemKompendiumData, ArtikelLokalKompData.gemLokalKompData
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **hentArtikelData**

Hensigt: Returnerer al data for en bestemt artikel.

Signatur: Artikel::hentArtikelData(enKode : Integer, lokal: String)
Logisk beskrivelse:
Pre: enKode -> notEmpty
lokal->notEmpty
Post: self.Artikel->select(kode=enKode and areaName=lokal)
Andre operationskald: ArtikelIFCCData.hentIFCCData,
ArtikelKompendiumData.hentKompendiumData,
ArtikelLokalKompData.hentLokalKompData,
ArtikelListe.hentListe
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

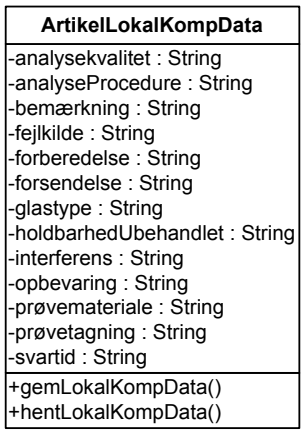
11.6.4 ArtikelIFCCData

Klassen er helt identisk med den beskrevet for det centrale system.

11.6.5 ArtikelKompendiumData

Klassen er helt identisk med den beskrevet for det centrale system.

11.6.6 ArtikelLokalKompData



Figur 53: ArtikelLokalKompData klassen for LabInfoAUH.

Klassebeskrivelse: **ArtikelLokalKompData**

Den lokale kompendium information, som kan være tilknyttet en artikel er beskrevet ved denne klasse. Informationen er givet ved klassen attributter, som er en ægte delmængde af attributterne fra klassen ArtikelKompendiumData.

Specifikation af operation: **gemLokalKompData**

Hensigt: Lagrer lokale kompendium Informationen for en bestemt artikel.
Signatur: ArtikelLokalKompData:gemLokalKompData(aLKD : ArtikelLokalKompData)
Logisk beskrivelse:
Pre: self.aLKD -> notEmpty
ArtikelLokalKompData -> includes(aLKD)
Post: none
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **hentLokalKompData**

Hensigt: Returnerer kompendium information for en bestemt artikel.
Signatur: ArtikelLokalKompData::hentLokalKompData(enKode : Integer, lokalNavn: String) kD : ArtikelLokalKompData
Logisk beskrivelse:
Pre: enKode -> notEmpty
lokalNavn->notEmpty
Post: kD = self.Artikel->select(kode=enKode and areaName=lokalNavn)->lokalInfo
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

11.6.7 ArtikelListe

Klassen er helt identisk med den beskrevet for det centrale system.

11.6.8 ArtikelSpecialeKobling

Klassen er helt identisk med den beskrevet for det centrale system.

11.6.9 ArtikelSynonym

Klassen er helt identisk med den beskrevet for det centrale system.

11.6.10 Forfatter

Figur 54: Forfatter klassen for LabInfoAUH.

Specifikation af attribut: **lokalnavn**

Beskrivelse:	Angiver det lokalenavn for det hospital som forfatteren er blevet tilknyttet af administratoren.
Format:	Tekststreng på maksimalt 50 tegn.

Specifikation af operation: **gemArtikel**

Hensigt: Gemmer artikel og kompendium informationen for en artikel. Ændre samtidig artiklens status til ikke længere at være godkendt. Datoen og forfatteren for revideringen ændres ligeledes.

Signatur: Forfatter::gemArtikel(a : Artikel, aKD : ArtikelKompendiumData, aKL: ArtikelKompLokalData)

Logisk beskrivelse:

Pre: self.a->notEmpty
self.Artikler->exists(kode=a.kode and areaName=a.areaName)
a.godkendt = false
a.revideret = today.oclIsTypeOf(date)
a.revideretAf=a.forfatterkode

Post: self.Artikel->includes(a)

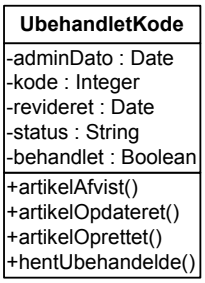
Andre operationskald: Artikel.gemArtikelData

Events transmitteret til andre objekter: none

Attributes set: areaName, godkendt, revideret, revideretAf

Response to exceptions: none
Non-functional requirements: none

11.6.11 UbehandletKoder



Figur 55: UbehandletKode klassen for LabInfoAUH.

Klassebeskrivelse: **UbehandletKode**

Klassen indeholder information på de artikler, der er relevante for LabInfoAUH. Kun artiklens kode er nævnt sammen med dens sidste revideret eller adminDato. Hvorvidt artiklens relevans er behandlet eller ej registreres ligeledes. Endelig registreres resultatet af behandlingen som dens status.

Specifikation af attribut: **adminData**

Beskrivelse: Datoen for oprettelsen af artiklen hos Sundhedsstyrelsen.
Format: Dato format af typen dd-mm-yyyy.

Specifikation af attribut: **kode**

Beskrivelse: Angiver IFCC-IUPAC nummeret på en artikel.
Format: Et naturligt tal på maksimalt 5 cifre.

Specifikation af attribut: **revideret**

Beskrivelse: Angiver datoen, hvor artiklens kompendium information sidst blev ændret.
Format: Dato format af typen dd-mm-yyyy.

Specifikation af attribut: **status**

Beskrivelse: Angiver om artiklen er afvist, opdateret eller oprettet i LabInfoAUH.
Format: Enumeration (Oprettet, Opdateret og Afvist).

Specifikation af attribut: **behandlet**

Beskrivelse: Angiver om artiklen er blevet behandlet eller ej.
Format: Boolean. Hvis true er artiklen blevet behandlet.

Specifikation af operation: **artikelAfvist**

Hensigt: At registrere at en artikel er blevet afvist. Det gøres ved at ændre dens status.
Signatur: UbehandletKode::artikelAfvist(enKode: Integer)
Logisk beskrivelse:
Pre: self.UbehandletKode->exists(kode=enKode and not behandlet)
Post: self.UbehandletKode->exists(kode=enKode and behandlet and status='Afvist')
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **artikelOpdateret**

Hensigt: At registrere at artiklen er opdateret. Det gøres ved at ændre artiklens status.
Signatur: UbehandletKode::artikelOpdateret(enKode: Integer)
Logisk beskrivelse:
Pre: self.UbehandletKode->exists(kode=enKode and not behandlet)
self.Artikel->exists(kode=enKode)
Post: self.UbehandletKode->exists(kode=enKode and behandlet and status='Opdateret')
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **artikelOprettet**

Hensigt: At oprette en artikel i LabInfoAUH og registrere at artiklen er oprettet ved at ændres dens status.
Signatur: UbehandletKode::artikelOpretttet(enKode: Integer)
Logisk beskrivelse:
Pre: self.UbehandletKode->exists(kode=enKode and not behandlet)
not self.Artikel->exists(kode=enKode)
Post: self.Artikel->exists(kode=enKode)
self.UbehandletKode->exists(kode=enKode and behandlet and status='Oprettet')
Andre operationskald: none
Events transmitteret til andre objekter: none
Attributes set: none
Response to exceptions: none
Non-functional requirements: none

Specifikation af operation: **hentUbehandlede**

Hensigt: At returnere koder på alle de artikler, der ikke er blevet behandlet.

Signatur: UbehandletKode::hentUbehandlede() uKoder : Set

Logisk beskrivelse:

Pre: none

Post: uKoder = UbehandletKode.kode->select(behandlet=false)

Andre operationskald: none

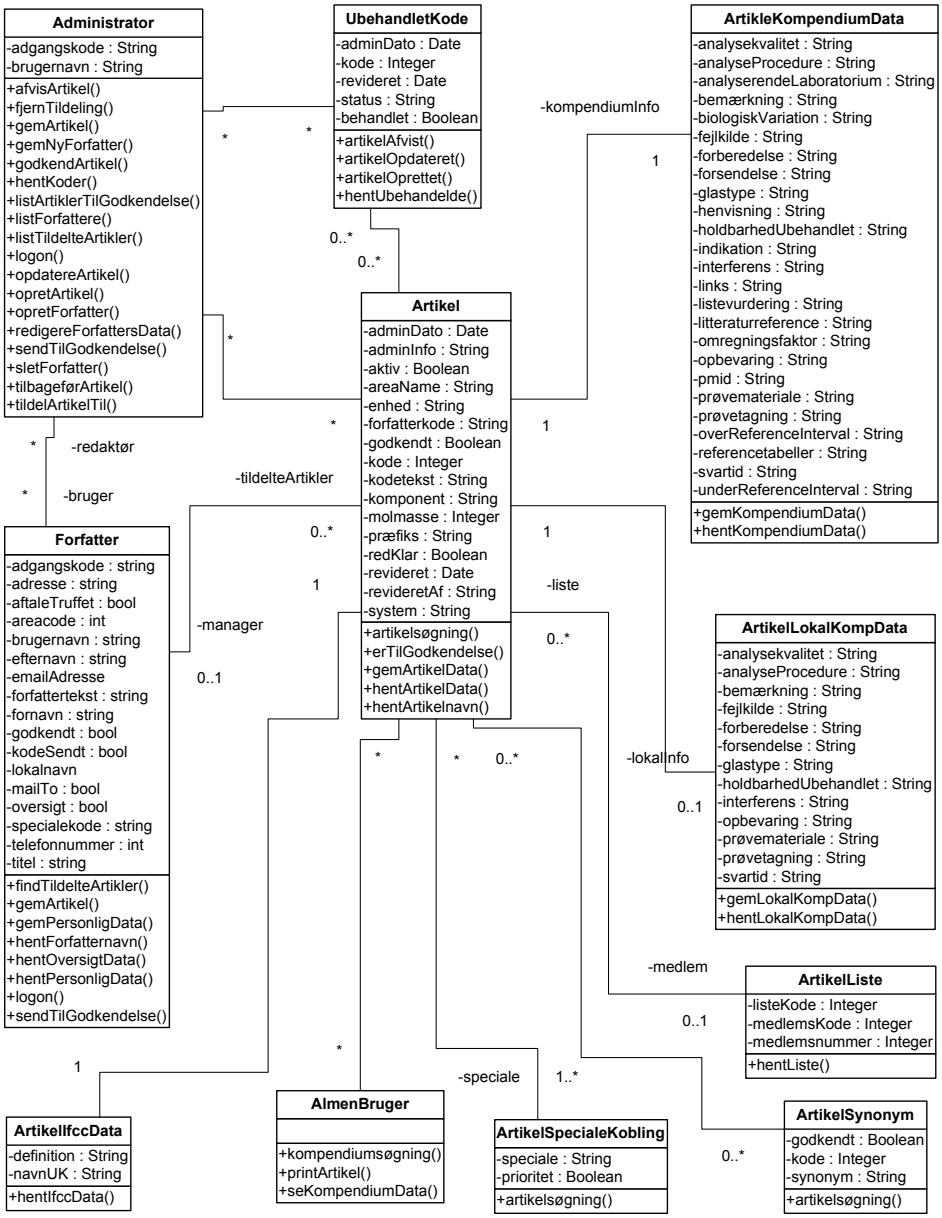
Events transmitteret til andre objekter: none

Attributes set: none

Response to exceptions: none

Non-functional requirements: none

11.7 Endelige klassediagram for LabInfoAUH

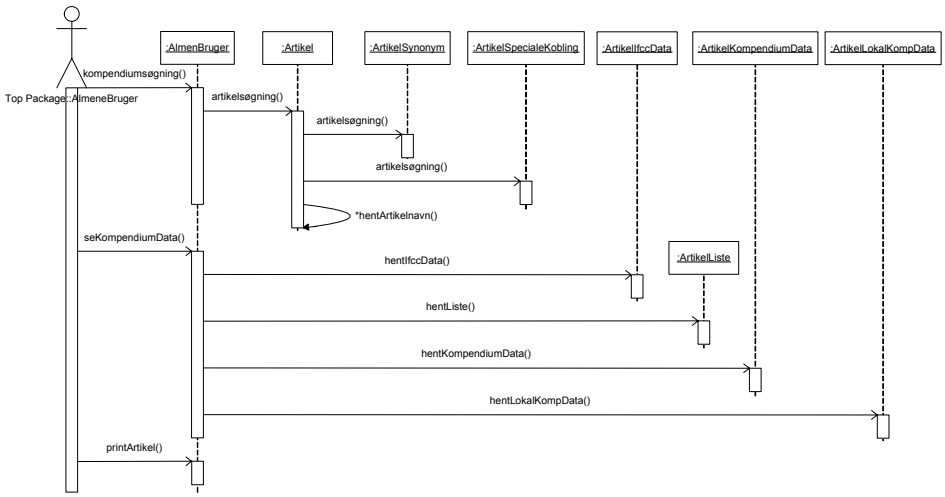


Figur 56: Det endelige klassediagram for LabInfoAUH med relationer mellem klasserne.

11.8 Sekvensdiagrammer

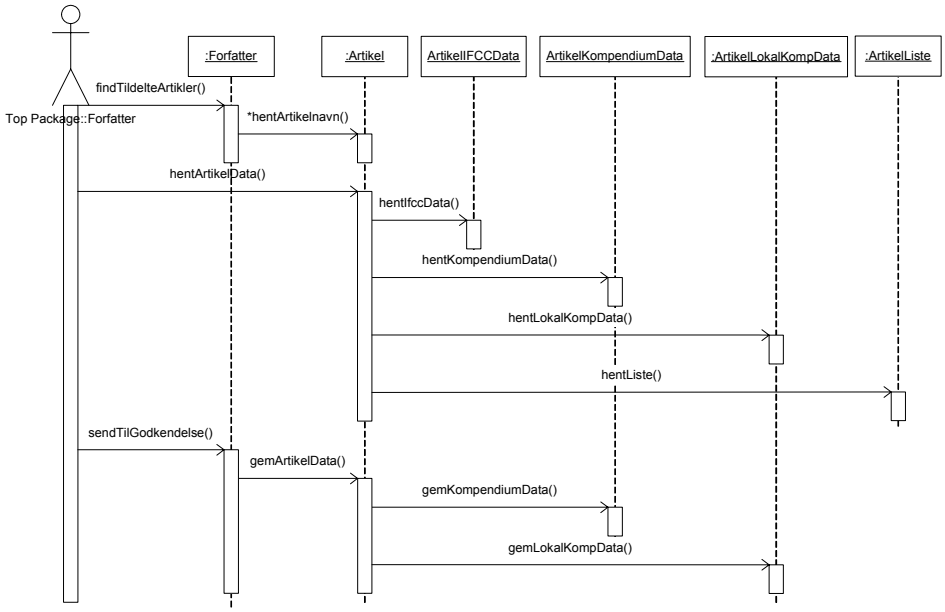
Nedenfor er opstillet de diagrammer for LabInfoAUH, som afviger fra LabInfo.

11.8.1 AlmeneBruger



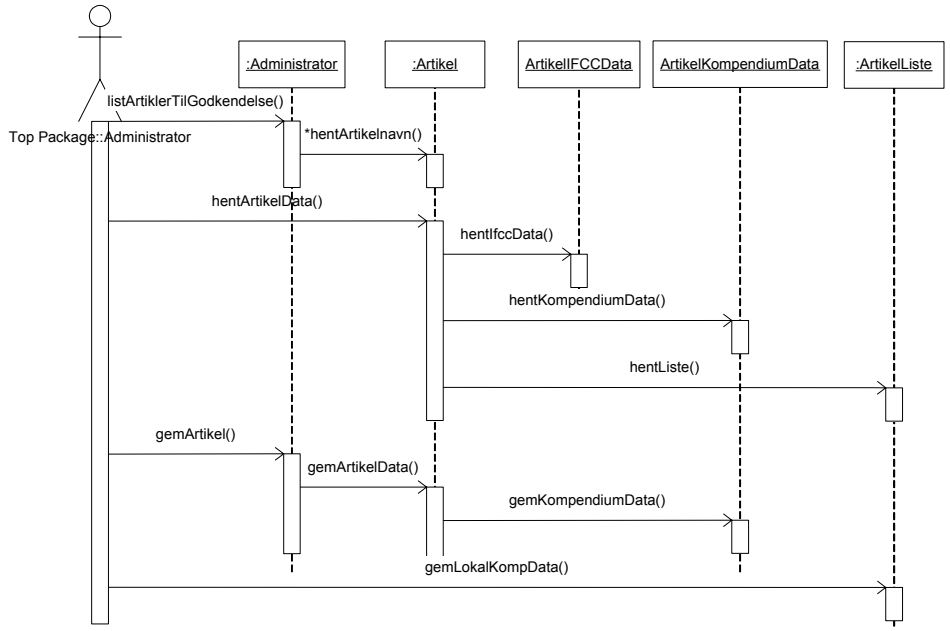
Figur 57: Sekvensdiagram for Kompendium søgning.

11.8.2 Forfatter

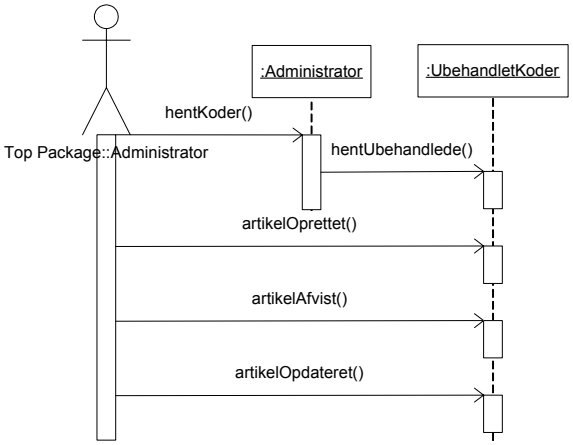


Figur 58: Sekvensdiagram for Send artikel til godkendelse.

11.8.3 Administrator



Figur 59: Sekvensdiagram for Redigere artikel.



Figur 60: Sekvensdiagram for Hent nye/ændrede koder.

Kapitel 4

12 Vurdering af LabInfo og LabInfoAUH

På grundlag af krav- og klassespecifikationerne for LabInfo og LabInfoAUH er det muligt at foretage en kritisk vurdering. Vurderingen vil omhandle hvordan systemdesignet for det fremtidige LabInfo system, LabInfo2003, skal se ud. På den måde vil det være muligt at komme med velbegrundede forslag til centrale problematikker.

Dette afsnit sætter de to systemer op mod hinanden for at identificere, hvor systemerne er lig eller afviger fra hinanden. På baggrund af dette giver efterfølgende afsnit forslag til, hvordan systemarkitekturen for LabInfo2003 bør være.

12.1 Fællestræk

Begge systemer er konstrueret til at håndtere og formidle information knyttet til kodeskemaet IFCC-IUPAC. Samtidig er de begge konstrueret til kun at kunne foretage ændringer i en bestemt del af informationen fra dette kodeskema. Måden hvorpå ændringerne foretages er derimod ikke helt identisk.

Kendetegnene for de to systemer er, at de begge har tre slags aktører. Disse tre aktører besidder egenskaber til at håndtere kodeskemaet og dets information. Egenskaberne varierer aktørerne imellem, men også mellem de to systemer. Denne variation er bedst præsenteret ved systemernes use case diagrammer. Klassediagrammerne giver det bedste billede af, hvordan håndteringen og opbygningen af informationen varierer.

Derfor vil det via systemernes use case diagrammer være muligt at konkludere på fællestræk indenfor systemernes funktionalitet. Konklusionen vil imidlertid kun være overordnet. Forstået på den måde, at en fælles funktionalitet godt kan variere systemerne imellem. Af den grund er det nødvendigt også at betragte klassediagrammerne. Via dem er det muligt præcist at bestemme lighederne mellem systemerne.

Variationen mellem de to systemer er ikke stor, hvilket fremgår af tabel 1 og tabel 2. Tabellerne sammenligner de to systemer med hensyn til deres attributter, som giver et billede af deres mere stationære struktur. Samt deres operationer, som er et udtryk for deres mere fleksible udseende. Indholdet i begge tabeller er opdelt efter systemernes klasser. Navnet på en klasse er angivet i kursiv. Efter klassens navn følger dens attributter eller operationer. Er en attribut eller operation tilstede i begge programmer angives det, om der forekommer en afvigelse, eller om de er identisk i begge systemer.

Tabel 1 illustrerer forekomsten og omfanget af variationen mellem klassernes attributter.

Tabel 1: Sammenhæng mellem klassernes attributter for LabInfo og LabInfoAUH.

Attribut	LabInfo	LabInfoAUH	Afvigelse
Administrator			Nej
adgangskode	√	√	
brugernavn	√	√	
AlmenBruger	√	√	Nej
Artikel	√	√	Ja
adminDato	√	√	
adminInfo	√	√	
aktiv	√	√	
areaName	÷	√	
enhed	√	√	
forfatterkode	√	√	
godkendt	√	√	
kode	√	√	
kodetekst	√	√	
komponent	√	√	
molmasse	√	√	
præfiks	√	√	
redklar	√	√	
revideret	√	√	
revideretAf	√	√	
system	√	√	
ArtikelIFCCData			Nej
definition	√	√	
navnUK	√	√	
ArtikelKompendiumData			Nej
analysekvalitet	√	√	
analyseProcedure	√	√	
analyserendeLaboratorium	√	√	
bemærkning	√	√	
biologiskVariation	√	√	
fejlkilde	√	√	
forberedelse	√	√	
forsendelse	√	√	
glastype	√	√	
henvisning	√	√	
holdbarhedUbehandlet	√	√	
indikation	√	√	

interferens	√	√	
links	√	√	
listevurdering	√	√	
litteraturreference	√	√	
omregningsfaktor	√	√	
opbevaring	√	√	
pmid	√	√	
prøvemateriale	√	√	
prøvetagning	√	√	
overReferenceInterval	√	√	
referencetabeller	√	√	
svartid	√	√	
UnderReferenceInterval	√	√	
ArtikelListe			Nej
listeKode	√	√	
medlemsKode	√	√	
medlemsnummer	√	√	
ArtiklLokalKompData			Ja
analysekvalitet	÷	√	
analyseProcedure	÷	√	
bemærkning	÷	√	
fejlkilde	÷	√	
forberedelse	÷	√	
forsendelse	÷	√	
glastype	÷	√	
holdbarhedUbehandlet	÷	√	
interferens	÷	√	
opbevaring	÷	√	
prøvemateriale	÷	√	
prøvetagning	÷	√	
svartid	÷	√	
ArtikelSpecialeKobling			Nej
speciale	√	√	
prioritet	√	√	
ArtikelSynonym			Nej
godkendt	√	√	
kode	√	√	
Synonym	√	√	
Forfatter			Ja

adgangskode	√	√	
adresse	√	√	
aftaleTruffet	√	√	
areacode	√	√	
brugernavn	√	√	
emailAdresse	√	√	
forfattertekst	√	√	
fornavn	√	√	
godkendt	√	√	
kodeSendt	√	√	
lokalnavn	÷	√	
mailTo	√	√	
oversigt	√	√	
specialekode	√	√	
telefonnummer	√	√	
titel	√	√	
UbehandletKode			Ja
adminDato	÷	√	
kode	÷	√	
revideret	÷	√	
status	÷	√	
behandlet	÷	√	

Tabellen viser hvor begrænset afvigelserne af klassernes attributter er mellem de to programmer. Største parten af afvigelserne er for klasser, der kun er repræsenteret i det ene program. Her tænkes på klasserne: *UbehandletKode* og *ArtikelLokalKompData*. De resterende afvigelser er for klasserne: *Artikel* og *Forfatter*. Her er afvigelserne begrænset til kun at omfatte en ekstra attribut for LabInfoAUH.

Tabel 2 viser variationen af klassernes operationer for de to systemer.

Tabel 2 Operationsoversigt for LabInfo og LabInfoAUH. Opstillet på baggrund af klassespecifikationerne.

Operation	**LabInfo**	**LabInfoAUH**	**Afvigelse**
Administrator			
afvisArtikel	÷	√	-
fjernTildeling	√	√	Ja
gemArtikel	√	√	Ja
gemNyForfatter	√	√	Nej
godkendArtikel	√	√	Ja
hentKoder	÷	√	-
listArtikelTilGodkendelse	√	√	Nej

listForfattere	√	√	Nej
listTildelteArtikler	√	√	Ja
Logon	√	√	Nej
opretArtikel	÷	√	-
opdatereArtikel	÷	√	-
opretForfatter	√	√	Nej
redigereForfattersData	√	√	Nej
sendTilGodkendelse	√	√	Ja
sletForfatter	√	√	Nej
tilbageførArtikel	√	√	Ja
tildelArtikel	√	√	Ja
Almenbruger			
danTekstfil	√	÷	-
IFCCsøgning	√	÷	-
kompendiumsøgning	√	√	Nej
printArtikel	√	√	Nej
seIFCCData	√	÷	-
seKompediumData	√	√	Ja
Artikel			
artikelsøgning	√	√	Nej
erTilGodkendelse	√	√	Nej
gemArtikelData	√	√	Ja
hentArtikelData	√	√	Ja
hentArtikelnavn	√	√	Nej
ArtikelIFCCData			
hentIFCCData	√	√	Nej
ArtikelKompendiumData			
gemKompendiumData	√	√	Nej
hentKompendiumData	√	√	Nej
ArtikelLokalKompData			
gemLokalKompData	÷	√	-
hentLokalKompData	÷	√	-
ArtikelListe			
hentListe	√	√	Nej
ArtikelSpecialeKobling			
artikelsøgning	√	√	Nej

ArtikelSynonym			
artikelsøgning	√	√	Nej
Forfatter			
findTildelteArtikler	√	√	Nej
gemArtikel	√	√	Ja
gemPersonligData	√	√	Nej
hentForfatternavn	√	√	Nej
hentOversigtsData	√	√	Nej
hentPersonligData	√	√	Nej
logon	√	√	Nej
sendTilGodkendelse	√	√	Nej
Ubehandlet			
artikelAfvist	÷	√	-
artikelOpdateret	÷	√	-
artikelOprettet	÷	√	-
hentUbehandlede	÷	√	-

Tabellen viser, at der stort set er den samme funktionalitet i begge systemer. Dog afviger operationen i enkelte tilfælde, men i overvejende grad er de ens. Det er også værd at bemærke, at LabInfoAUH i større grad har operationer, som ikke er repræsenteret i LabInfo. Disse operationer er et resultat af, at LabInfoAUH er udstyret med en funktionalitet, som ikke er tilgængelig i LabInfo. Funktionaliteten er at tillægge lokalt kompendium information til en artikel.

Det bemærkes at operationens afvigelse er begrænset. Operationen er blevet tilpasset de krav, der er opstået i forbindelse med den nye funktionalitet.

De typiske afvigelser er gengivet i disse tre operationer:

Operation:	fjernTildeling
Hensigt:	Afviger ikke i de to systemer.
Signatur:	Operationen er tildelt samme klasse. LabInfoAUH tager 3 parameter mod to for LabInfo. Den tredje parameter er en tekststreng med et lokalnavn.
Logic:	Har samme pre og post betingelser. Men for LabInfoAUH bruges det lokalenavn til yderligere at præciserer betingelserne.

Resten af operationen er identisk for de to systemer.

Operation:	gemArtikel
Hensigt:	Afviger ikke i de to systemer.
Signatur:	Operationen er tildelt samme klasse. Men LabInfoAUH tager 3 parameter med to for LabInfo. Den tredje er et objekt over lokalKompendiumData.

Logic:	Ingen afvigelser

Resten af operationen er identisk for de to systemer.

Operation:	gemArtikelData
Hensigt:	Afviger ikke i de to systemer.
Signatur:	Operationen er tildelt samme klasse. Men LabInfoAUH tager 3 parameter med to for LabInfo. Den tredje er et objekt over lokalKompendiumData.
Logic:	Ingen afvigelser.
Andre kald:	LabInfoAUH laver et ekstra kald til ArtikelLokalKompData.

Resten af operationen er identisk for de to systemer.

Hensigten for operationerne ændrer sig altså ikke. Det typiske er, at signaturen får en ekstra parameter for LabInfoAUH. Men resten af signaturen forbliver den samme. Den ekstra parameter kan medføre ændringer af de logiske betingelser, hvilket hænger sammen med at klassens attributter varierer mellem de to programmer. De ekstra parametre kan også være resultatet af en ny klasse. Hvilket kommer til udtryk ved et ekstra operationskald til denne klasse.

Konklusionen af ovenstående tabeller er:

- Brugen af klasser er tilnærmelsesvis ens for de to programmer.
- Afvigelsen af klasserne mellem de to programmer er hovedsagelig indenfor operationerne. Der er stort set ikke variation på attributterne.

Dette giver et billede af hvor i systemarkitekturen, der optræder ændringer. Den lille afvigelse på klassernes attributter indikerer, at programmerne formidler den samme slags information. Måden informationen formidles og præsenteres på er derimod ikke helt ens. Derfor den større afvigelse på klassernes operationer. Set ud fra systemarkitekturen indvirker klassernes operationer på den øvre del, mens attributterne er koncentreret om den nedre del af arkitekturen.

Ud fra klasserne alene er det ikke muligt at konkludere på forekomsten af redundans i programmernes information. Få at danne et billede af den fælles information betragtes use case diagrammerne og deres beskrivelser. Heraf fremgår det, at der kun foretages ændringer på dele af informationen for en artikel. Den resterende del forbliver konstant i begge programmer. Den konstante del for en artikel er kendetegnet ved følgende attributter:

- Kode
- Kodetekst
- System
- Præfiks

- Komponent
- Definition
- Engelsknavn
- Enhed
- Molmasse
- Liste sammenhæng

Denne mængde er i bilag 19.1 benævnt som basisinformation for en artikel[7]. En artikel, der eksisterer i begge programmer, har således sammen basisinformation uden undtagelser. Udover at informationen i de to programmer følger samme struktur, er der med basisinformationen fastlagt et behov for at dele information på tværs af systemerne. En håndtering af dette behov vil finde sted på det nedre plan i systemarkitekturen.

12.2 Opsamling

De fællestræk for programmerne viser, at informationen for en artikel følger samme struktur i begge tilfælde. Den eneste undtagelse er den ekstra lokale kompendium information præsenteret i det decentrale system. Den information kan betragtes som en mulig modificering af kompendium informationen for en artikel, der er repræsenteret i det centrale system. Strukturen for den lokale information vil derfor altid være en ægte delmængde af en artikels kompendium information.

De fleste afvigelser registreres på det øvre niveau af systemernes arkitektur. Det er nævnt at begge systemer har samme typer og antal aktører, hvis funktionalitet tilnærmelsesvis er ens.

Aktørerne "forfatter" og "administrator" er begge beskrevet ved hjælp af et sæt attributter. Mængden af disse attributter er ikke præcis defineret skriftligt nogen steder. Det eneste præcise krav er, at begge aktører skal have et brugernavn og en adgangskode. Det er tidligere vist, at attributterne for "administratoren" ikke variere mellem de to systemer. Mens "forfatteren" i det decentrale system er tillagt en ekstra attribut.

Den "almeneBruger" er ikke tillagt nogen attributter. Aktøren er beskrevet ved at kunne udføre en række funktioner. Funktionerne for det decentrale system er også repræsenteret i det centrale system. Men som vist i tabel 2 er de ikke helt identiske. Dette fremgår blandt andet af use case diagrammerne, hvor brugen af "uses" og "extends" varierer.

Use case diagrammerne viser også, at "administrator" for det decentrale system kan udføre en ekstra funktionalitet. Denne er en udbygning, der muliggør sammenligning og kommunikation med det centrale system.

[7] I bilag omtales en artikel som en egenskab. Dette er blot en anden betegnelse men dækker over det samme.

Funktionaliteten bevirker, at kodeskemaet for det decentrale system forbliver en ægte delmængde af det centrale.

Det er tydeligt at IFCC-IUPAC kodeskemaet danner ramme om den fællesdel af de to systemer. Samtidig består afvigelserne for det decentrale system i, at fravælge eller udbygge mulighederne for at behandle dette kodeskema. I og med at kodeskemaet er grundlæggende for begge systemer, vil deres nedre del af systemarkitekturen også være det.

Systemarkitekturen bliver gennemgået i næste afsnit. Her tages der stilling til, hvordan udformningen bør være set ud fra en sammenlægning af de to systemer

13 Systemdesign

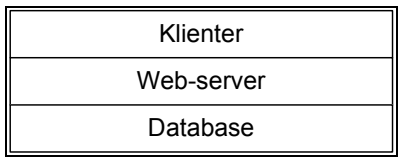
Informationen, fra den forudgående analyse, danner baggrund for ændringerne af systemdesignet. En decideret ændring af systemdesignet er nødvendig, hvis der skal gøres forhåbninger om at sikre LabInfo systemets fremtidige vedligeholdelse og administrering. Dette afsnit tager udgangspunkt i det eksisterende systemdesign og forklarer, hvorfor en ændring er nødvendig. Efterfølgende bliver der taget stilling til udformningen af et nyt og mere hensigtsmæssigt systemdesign.

13.1 Nuværende systemdesign

Fra dokumentationen over LabInfo og LabInfoAUH, se bilag, er nedenstående modeller opstillet for systemernes nuværende systemdesign.

LabInfo

LabInfo er en web-løsning, der er opbygget via HTML og ASP filer. Den stationære del af programmet er skrevet i HTML, som bliver fortolket af en web-server. Den dynamiske del af programmet er skrevet i sproget ASP og sætter web-serveren i forbindelse en SQL-server. Både web-serveren og SQL-serveren er placeret på samme maskine. Tilgangen til web-serveren sker via en PC med Internet forbindelse. Udover at have adgang til Internettet stilles der ikke yderligere krav til klienterne. Arkitekturen mellem klienten og web-serveren er af typen "thin two-tier client-server". Men på grund af forbindelsen til databasen betragtes den samlede arkitektur som en "three-tier" model. Modellen med de tre niveauer er vist i figur 61.



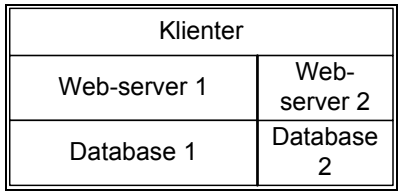
Figur 61: Nuværende arkitektur for LabInfo.

Det kan diskuteres om niveauet med klienten bør være med i modellen, fordi klienten rent programmæssigt ikke er adskilt fra web-serveren. Men den er taget med for at vise kommunikationen inden for LabInfo. Arkitekturen i modellen er lukket, hvilket vil sige, at der kun kan kommunikeres med naboniveauet.

LabInfoAUH

I LabInfoAUH er der som for LabInfo tale om en web-løsning, der er skrevet ved hjælp af to programmeringssprog henholdsvis HTML og ASP. På tilsvarende vis er der tale om en web-server, hvor programmet afvikles. Mens de ekstraordinære data hentes fra en SQL-server. Dog afviger LabInfoAUH ved at have kontakt med to web-servere og to SQL-servere. Henholdsvis en lokal web-server og SQL-server samt de to servere præsenteret under LabInfo. Klienterne er ikke noget specielt, men stadig en del af en web-løsning.

Forholdet mellem web-serveren og klienten er derfor også af typen "thin client-server". Databasen bevirker igen, at der er tale om en "three-tier model". Kommunikationen mellem de tre niveauer bliver som vist i figur 62.



Figur 62: Nuværende arkitektur for LabInfoAUH.

Arkitekturen er lukket, og der kan kun kommunikeres vertikalt. Rent fysisk er systemet kun opdelt i to dele henholdsvis en webapplikation og en database. På grund af klientens type og programmets opbygning kan klienten betragtes som en uadskillelig del af webapplikationen.

Næste afsnit tager udgangspunkt i de nuværende arkitekturer, og præsenterer en arkitektur, hvor systemet er delt bedre op.

13.2 Nyt systemdesign – en nødvendighed

Af ovenstående modeller fremgår det, at systemerne består af en database og en webapplikation. Både databasen og webapplikationen kan betragtes som delsystemer for det samlede LabInfo system. Den simple opdeling kan for større systemer bevirke, at procesforløbet vanskeliggøres ved vedligeholdelse og eller indførelse af nye systemkrav.

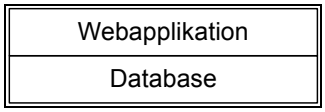
Ligeledes vil det for LabInfo være en klar fordel at opdele systemet i delsystemer. Specielt med udsigt til en antalsmæssig forøgelse af de decentrale systemer. Samtidig vil det forbedre overblikket i forbindelse med vedligeholdelsen. Generelt vil flere af de i afsnit 4.1 opstillede punkter blive opfyldt ved gennemførelse en opdeling af det eksisterende system.

13.3 Systemarkitekturen opstilles

Formålet med en systemarkitektur er at præcisere de rammer, som et programmet opererer indenfor. Det kræver, at essentielle komponenter og

deres indbyrdes kommunikation bliver kortlagt. Flere komponenter kan lægges sammen til et modul. Modulet vil, set fra systemet, være i stand til at udføre en bestemt rolle. Disse roller er repræsenteret i arkitekturen som de enkelte niveauer.

En systemopdeling af LabInfo foretages ved først at se på den nuværende arkitektur, figur 63.

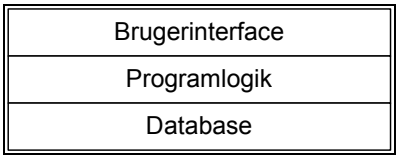


Figur 63: Arkitektur af eksisterende LabInfo.

Det nederste niveau består af en database, hvor al informationen for systemet er lagret. I første omgang ses der bort fra selve informationen. Derimod er strukturen, hvorpå informationen er lagret, interessant.

Webapplikationen er det øverste niveau og er omfattet af diverse bruger- og systeminterfaces. En opdeling af disse vil forbedre systemet på en række punkter.

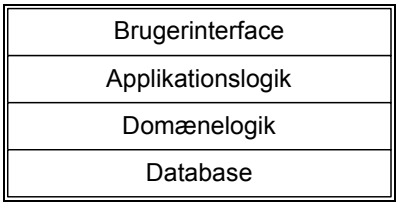
Webapplikationen kan med fordel opdeles i to niveauer [Bennett, 1999]. Det øverste vil håndtere brugerinterfaces. Mens det midterste vil præcisere programspecifikke begreber. Det tredje og nederste lag vil stadig være databasen. Inddelingen er illustreret i figur 64.



Figur 64: Arkitektur inddelt i tre niveauer.

Niveauet med brugerinterface kan for LabInfo ikke opdeles i flere niveauer. Selvom der eksisterer tre forskellige brugerniveauer, så er de indbyrdes uafhængige af hinanden. Hver aktør på systemet har en selvstændig indgang. Derimod kan niveauet med programlogik opdeles i to niveauer.

Det er tidligere nævnt at IFCC-IUPAC kodeskemaet er repræsenteret i både LabInfo og LabInfoAUH. Begge systemer opererer altså inden for samme domæne. Det ene af niveauerne vil derfor præcisere logikken bag domænet. Mens det andet vil beskrive logikken for den enkelte applikation af LabInfo systemet. Resultatet bliver som vist i figur 65.



Figur 65: Arkitektur inddelt i fire niveauer.

En opdeling i 4 niveauer vil være tilstrækkelig for LabInfo. Hvert niveau har en specifik rolle at udfylde for systemet. Nedenfor er indholdet af hvert niveau beskrevet.

Database – Indeholder strukturen og informationen for basisinformationen fra kodeskemaet. Strukturen og informationen til lagring af kompendium og aktør information er ligeledes inkluderet i databasen.

Domænelogik – Omfatter et interface, der er i stand til at udføre al den domænespecifikke funktionalitet. Dette indebærer al kommunikation til databasen. På den måde bliver mulige operationer og handlinger udført på informationen i databaserne kun udført et sted. Dermed sikres troværdigheden af informationen.

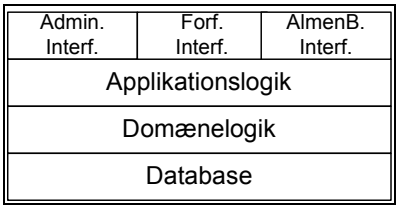
Applikationslogik – Et interface, der muliggør udførelsen af den funktionalitet, som kun er pålagt den enkelte applikation. Interfacet får sin information fra domænelogikken. Informationen fra domænelogikken er entydig, men kan behandles forskelligt i det enkelte system.

Brugerinterface – Præciserer interface med grænseflade for hver enkelt bruger af systemet. Interfacet kommunikerer med applikationslogikken. Informationen modtaget fra applikationslogikken er nødvendigvis ikke længere entydig. Den præsenterede information kan derfor variere mellem de enkelte systemer.

En opdeling af LabInfo i fire delsystemer giver en klar adskillelse af systemet bestanddele. Hvert delsystem har sin egen rolle i det samlede system. Specielt databasen og brugerinterfacet udgør en klar adskillelse i forhold til systemet. At opdele resten i et domæne og en applikationsdel kan forekomme unødvendigt, hvis kun LabInfo betragtes. Men inddrages det decentrale system LabInfoAUH bliver opdelingen hensigtsmæssig.

LabInfoAUH råder over funktionalitet, der ikke er dækket af det centrale system. Men funktionaliteten opererer stadig inden for domænets rammer. Et eksempel er, at det decentrale system kan koble lokalt kompendium information til kodeskemaet. Selvom denne funktion ikke er en del af det centrale system, er strukturen af kompendium information fastlagt af domænet. En komponent til håndtering af lokalt kompendium information er derfor bedst placeret som et delsystem for den pågældende applikation.

At fjerne eller oprette nye komponenter i decentrale systemer vil have indflydelse på delsystemet "applikationslogik". Men det vil også have indflydelse på "brugerinterface". For LabInfoAUH kan det have betydning for en eller flere at systemets aktører. For lettere at kunne identificere de aktører, der berøres af en ændring, vil en horisontal inddeling være fordelagtig. Delsystemet "brugerinterface" inddeles derfor i tre, som i figur 66.

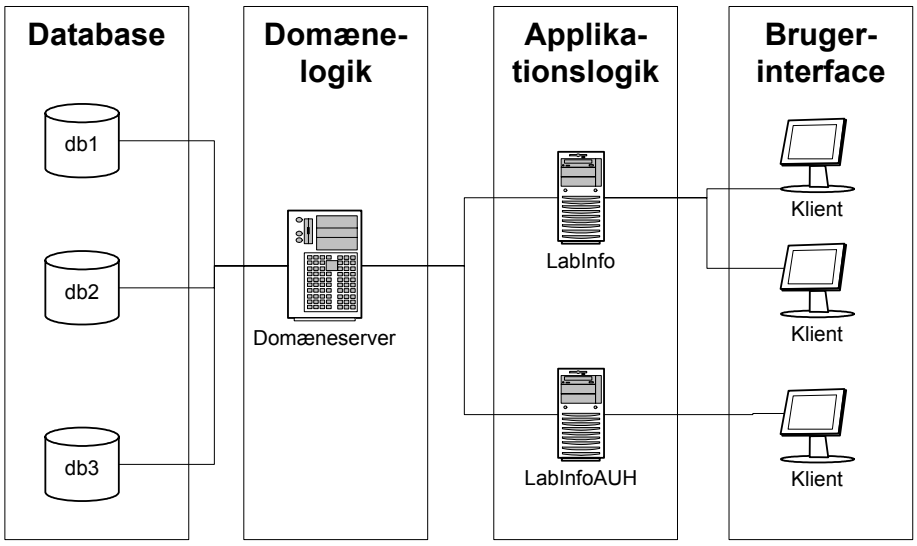


Figur 66: Systemarkitektur for centralt og decentralt system.

Denne arkitektur betragtes som endelig for LabInfo og LabInfoAUH.

13.3.1 Placering af delsystemer

Med den opstillede systemarkitektur i figur 66 er det relevant at vurdere mulighederne for, at kunne betragte LabInfo som et distribueret system. Den store lighed mellem systemerne giver mulighed for at slå dem sammen til et fælles system. Systemarkitekturen for dette fælles system skal være som vist i figur 66. Systemet benævnes LabInfo2003 og tænkes distribueret på følgende måde.



Figur 67: Distribuering af LabInfo2003

Ved en sammenlægning vil der være flere databaser tilknyttet systemet. I afsnit 12 blev det gjort klart, at LabInfo og LabInfoAUH råder over hver

deres information. Men samtidig eksisterer der også en hvis mængde information, som er fælles for dem begge. Informationen fordeles derfor på hver deres database. Dermed minimeres mængden af redundante data. Ansvaret for vedligeholdelsen af databasen kan nemmere placeres når informationen forbliver afgrænset.

Informationen omkring kodeskemaet er prædefineret for alle LabInfo systemer. Derfor er håndteringen af kodeskemaet og den øvrige datamængde placeret på en domæneserver. Denne server er alene i stand til at finde og kommunikere med databaserne. Al behandling af data for LabInfo2003 er dermed samlet på et centralt sted. Selve domænelogikken tænkes opdelt i komponenter, der er i stand til kun at kommunikere med en enkelt database. Dermed sikres, at behandlingen af informationen forbliver entydig. En komponent vil eksempelvis kun skulle håndtere kodeskemaet, som jo er prædefineret.

De enkelte systemer bliver placeret i applikationslogikken som selvstændige applikationsservere. Her tænkes på systemerne LabInfo, LabInfoAUH og eventuelle øvrige tilkommere. Det er vist, at applikationsserverne er ens på det punkt, at de præsenterer information fra det samme kodeskema. Men samtidig er de forskellige i måden at håndtere og præsentere informationen på. Eksempelvis vil to applikationer, der hver især viser informationen for en artikel forskellig, lave samme funktionskald til domæneserveren. Ønsket om at få tilsendt informationen for en artikel behandles af domæneserveren, og resultatet sendes tilbage. Den enkelte applikation er nu selv ansvarlig for at sortere i den tilsendte data. Samtidig kan en applikation være nød til at lave et ekstraordinært funktionskald til domæneserveren for at få den nødvendige datamængde.

Klienterne hver især skræddersyet til en bestemt web-server. Men da der er tale om en web-løsning på Internettet, vil de være i stand til at komme i kontakt med øvrige applikationsservere. Men som tidligere beskrevet giver modellen mulighed for at tillægge klienten større eller mindre arbejdsbelastninger. Dermed forbliver LabInfo2003 fleksibel.

En distribuering af delsystemerne kræver en forbedret måde for systemerne at kommunikere på. En simple "two-tier client-server" løsning er ikke længere tilstrækkelig. I næste afsnit lægges der vægt på at bestemme en mulig løsningsmetode.

14 Kommunikation mellem LabInfo systemerne

I dag er det almindeligt, at informationsprogrammer bliver programmeret som distribuerede systemer. Det giver mulighed for, at programmet kan afvikles ved hjælp af flere computere frem for kun en. I og med at LabInfo er en web-baseret program, kan programmet betragtes som et distribueret system. Web-baserede programmer hører typisk ind under kategorien "Thin-client model". Hvilket vil sige, at klienten udføre et minimalt arbejde. Al funktionaliteten og databehandlingen afvikles på serveren. Klienten er kun i stand til at præsentere data sendt fra serveren.

Der er en række fordele ved den type af systemer, men også svagheder. Svaghederne kommer typisk til udtryk i forbindelse med udvidelse af systemet, som for LabInfo. Frem for at udbygge serveren til at håndtere nye funktioner udbygges klienten til at håndtere opgaven. Denne løsning er typisk, når en funktionalitet kun skal være tilgængelig for en del af systemets klienter. Ulempen bliver derimod, at klienterne med tiden kan blive tunge at afvikle. Hvilket er i modstrid med tankegangen bag "Thin-client model". Resultatet kan være, at programmet blive af typen "Fat-client model". Her bliver programmets funktionalitet primært udført af klienten, mens serveren kun står for håndteringen af data. Løsningen er problematisk, hvis det ikke er muligt at opgradere klienterne til at kunne udføre denne øgede arbejdsbelastning. Løsningen kan være, at udarbejde en ny systemarkitektur for programmet.

Denne løsning er forsøgt brugt på LabInfo, og resultatet blev præsenteret i forrige afsnit. De enkelte lag i arkitekturen kan dermed distribueres, som vist i figur 67. Efterfølgende vurderes hvilken "client-server architecture", der er tænkt brugt til afvikling af LabInfo2003.

Et andet vigtigt aspekt i forbindelse med indførelse af et distribueret "client-server" system er brugen af et objektorienteret programmeringssprog. Specielt JAVA er velkendt i denne sammenhæng, hvor systemet skal præsentere behandlet data via en hjemmeside. Årsagen er, at både "client-server" og et objektorienteret sprog følger samme princip om at opdele en helhed i mindre overskuelige dele.

14.1 Klient-server arkitektur

Inden en bestemt "client-server" metode kan fastsættes, er det vigtigt at have et vist overblik over systemets omfang. Overblik kan opnås ved at vurdere systemet ud fra punkter som:

- Antallet af klienter
- Klienternes maskinelle begrænsninger

- Klientens belastning af serveren
- Eventuelle begrænsninger for størrelsen af datastrømmen mellem delsystemerne
- Hvem der skal være ansvarlige for de enkelte delsystemer

Overvejelser, som punkt 2 og 3, er af stor betydning for arkitekturen. Er klienterne kun i stand til at udføre en meget lille arbejdsbelastning, vil klienternes server være pålagt en stor arbejdsbelastning. Denne belastning er stærkt voksende for et øget antal af klienter. Samtidig kan det medvirke til en øget trafik på netværksforbindelsen mellem klient og server, fordi al behandling af data afvikles på serveren.

En mulig løsning kan være at investere i nogle kraftigere servere. Eller ved at påtvinge større arbejdsbelastning til klienterne og dermed risikere for høje krav til nogle af klienterne. Beslutningen vanskeliggøres hvis systemets fremtid er uvis og eventuelle udvidelser ikke kan forudsiges. En mere holdbar, men også mere tidskrævende løsning er at forbedre systemdesignet.

Netop denne situation har LabInfo systemet været i. Første løsning var et eksternt næsten identisk system LabInfoAUH. For at komme problematikken til livs er der i denne rapport taget hul på et nyt systemdesign for LabInfo2003.

Begge systemer bruger i deres eksisterende udgave en simpel "client-server" model. En vurdering af omfanget for modellen for LabInfo er som følgende:

Tabel 3: Vurdering af "client-server" for LabInfo.

Antal af brugere på LabInfo	Samlet antal af faste brugere, det vil sige administratorer og forfattere, er under 200. Dertil kommer et varierende antal af almene brugere. Denne mængde er på nuværende tidspunkt håndterbar af en almindelig web-server.
Klient typer	Der er ikke fastlagt nogen krav til operativsystemet for en klient. De skal være i stand til at vise Html-sider. Men der er ikke fastlagt nogen krav for, hvor avanceret siderne må være.

Informationen fra tabel 3 er vejledende, eftersom det ikke var muligt at skaffe statistiske data for systemet. Situationen er tilsvarende for LabInfoAUH.

LabInfo2003 vil, som LabInfo og LabInfoAUH, være omfattet af en eller flere web-servere. Ud fra vurderingen forventes det ikke, at antallet af klienter vil skabe problemer for web-serverne under LabInfo2003. På grund af det

nye systemdesign vil dele af web-servernes tidligere arbejdsbelastning være flyttet til domæneserveren. De må derfor også kunne forventes at modstå en vis forøgelse af klienter.

Den store variation eller mangel på præcisering af klienttyperne kan derimod give anledning til bekymring. Det kan mellem klient og web-server være nødvendigt at holde fast i en "thin-client" model. Variationen af klienternes operativsystem kan afhjælpes ved at bruge et programmeringssprog, der ikke er afhængig af operativsystemet. Eksempelvis Java.

Med Java er det også muligt at flytte databehandlingen fra serveren og ud til klienten. For systemer med store klienter kan det være en fordel. Det kræver imidlertid, at systemet kan håndtere to typer af klienter. Af figur 67 fremgår det, hvordan dette kan løses. Applikationslogikken betragtes som en selvstændig del i det fælles system. På den måde kan LabInfo og LabInfoAUH have individuelle krav til deres klienter. I den sidste ende er det et spørgsmål om, hvor stor en arbejdsbelastning deres servere pålægges. Inden for Java vil det være et spørgsmål om i overvejende grad at bruge Servlet eller udbygge en Applet til klienten.

Essensen er i større grad at betragte klienten som et delsystem frem for en integreret del af web-applikationen. Dermed indgår klienten som en selvstændig del af systemarkitekturen.

14.2 Four-tier client-server model

Arkitekturen fra figur figur 67 er af typen "four-tier client-server". Det vil sige, at systemet er fordelt på fire delsystemer. Fordelingen er som følgende:

- Databaseservere
- Domæneserver
- Applikations web-servere
- Klienter

Systemarkitekturen, se figur 66, for LabInfo2003 er også inddelt i fire niveauer. Det er netop disse fire niveauer, der placeres som hver deres delsystem.

Som nævnt giver denne opdeling den eftersøgte fleksibilitet i programmet og ikke mindst på kommunikationsområdet. Opdelingen giver en øget mulighed for en mere synlig og ensrettet kommunikation. Synlig fordi det klart fremgår, hvor de enkelte delsystemer kan hente deres information. Ensrettet eftersom det ikke længere er muligt for klienter at hente information fra andre web-applikationer. Al information kommer fra domæneserveren, som igen får den fra en specifik database. Derved bliver web-applikationerne også uafhængige af hinanden. I det eksisterende system er LabInfoAUH afhængig af LabInfo, hvilket fremgår af figur 62.

Web-applikationerne henvender sig i større grad kun til en gruppe af klienter. Fordelen er som før nævnt, at LabInfo2003 er åben over for nye web-applikationer, som måtte have et andet operativsystem end eksisterende web-applikationer. På grund af opdelingen vil den eneste begrænsning ligge i kommunikationen mellem domæneserveren og web-applikationen. Så længe at domæneserveren er i stand til at modtage "request" fra en bred vifte af web-servere, vil denne begrænsning være ubetydelig.

Samme fleksibilitet skal være gældende for kommunikationen mellem domæneserveren og databaserne. I og med at databaserne kan være placeret forskellige steder i landet, vil de højst sandsynlig også være af forskellige typer. Denne typeforskel skal kunne håndteres af domæneserveren.

Det sidste delsystem er klienten, som er skræddersyet til en bestemt web-server. Derfor er klienten ikke pålagt nogen krav af LabInfo2003, men kun af den tilhørende web-server.

Nedenfor betragtes hvert delsystem nærmere med henblik på at lave en abstrakt specificering af deres servicer. Specificeringen består i at inddrage de to klassediagrammer for LabInfo og LabInfoAUH i det fælles system LabInfo2003.

Klient

Den tillagte service kan variere fra ingen til at omfatte det meste af klasserne for systemets aktører. Variationen bestemmes af klientens type. Er klienten en simpel "thin-client" vil dens web-applikation stå for al servicen. Klienten er med andre ord et præsentationsmedie for web-serveren. Den modsatte klient vil være udføre services frem for web-applikationen. De mulige services er afgrænset af klasserne: "Administrator", "Forfatter", "AlmenBruger".

Applikation web-server

Delsystemet udfører funktionaliteten for alle dets aktører. Denne mængde kan mindskes ved at flytte dele af funktionalitet til klienten. Servicen omfatter behandlingen af den information, der modtages fra domæneserveren.

Domæneserver

Delsystemet skal står for håndteringen af informationen under LabInfo2003. Det vil først og fremmest sige informationen bag kodeskemaet. Klasserne "Artikel", "ArtikelIFCCData", "ArtikelSpecialeKobling", "ArtikelListe", "ArtikelSynonym" og "ArtikelKompendiumData" dækker over denne funktionalitet. Men også den service, som er givet ved klassen "ArtikelLokalKompData" og "UbehandletKode" håndteres her. Ud over kodeskemaet er der lagret information for systemet aktører. Delsystemet besidder altså overblikket over, hvor informationen er lagret i LabInfo2003. Det vil sige hvilken database der kommunikeres med.

Database

Dette delsystem lagrer informationen for LabInfo2003. Systemet er omfattet af mindst en database, som indeholder informationen for kodeskemaet. Denne information er entydig og øvrig ekstraordinær data for en web-applikation lagres i en separat database.

14.2.1 Interface og komponenter

Domæneserveren er det eneste delsystem, der kommunikerer med databaserne. De nødvendige drivere til databaserne behøver dermed kun at blive installeret på domæneserveren. Sidstnævnte skal have et interface til hver database.

Web-applikationerne skal ligeledes have et interface, der sætter dem i stand til at hente information fra domæneserveren. Som nævnt vil domæneserveren have standardiserede komponenter. Det er disse komponenter interfacet skal kunne kommunikere med. De standardiserede komponenter dækker over den funktionalitet, som er tillagt kodeskemaet under LabInfo. Øvrige komponenter vil blive oprettet efter behov fra nye web-applikationer. Eksempelvis vil LabInfoaAUH kræve oprettet en komponent på domæneserveren, der kan håndtere informationen for deres lokale kompendium information. Web-applikationen må derfor have et interface, der kan kommunikere med det standardiserede og den nye komponent.

Komponenterne for web-applikationerne vil være inddelt efter dets aktører. Dermed sikres overblikket af delsystemet.

Netop overblikket er essentielt for delsystemerne. Derfor er det vigtigt at have en fast opdeling af systemet i komponenter. Netop overblikket er en vigtig faktor for næste afsnit om implementering.

15 Implementering – det næste trin

Der vil i denne opgave ikke blive foretaget nogen implementering. Men den forudgående dokumentation skal danne ramme for implementering af LabInfo2003. Emnet vil blive brugt som afrunding, hvor hovedpunkter og overvejelser vil blive belyst.

I gennem opgaven er der ikke gjort nogen overvejelser med hensyn til valg af programmeringssprog. Denne overvejelse er overladt til EDB Gruppen. På grund af systemdesignet for LabInfo2003 vil det være hensigtsmæssigt at valget falder på et objektorienteret sprog. Hvilket dog også er et ønske fra EDB Gruppens side. Sproget JAVA er i rapporten nævnt som et muligt valg. JAVA har den fordel, at det kan bruges på alle delsystemerne. En af kvaliteterne ved JAVA er, at det er specielt udviklet til distribuerede systemer på Internettet. Samtidig er sproget uafhængig af delsystemernes platform. En anden overvejelse ville være at bruge Microsoft .Net som løsning. Løsningen er ligeledes fleksibel eftersom Microsoft .Net blandt andet gør brug af sproget XML[8]. En af grundideerne bag Microsoft .Net er at skabe fleksibilitet for deling af information på Internettet.

Uanset valget af programmeringssprog vil det have en vis indvirkning på interfaces og komponenterne under Labinfo2003. Deres overordnede udseende blev gennemgået afsnit 14.2.1. I forbindelse med implementeringen kan det derfor blive nødvendigt at vende tilbage til forrige aktivitet inden for programmelkonstruktion, se 3.1. Den komplette opdeling af de enkelte delsystemer kan dermed lettere præciseres og eventuelt visualiseres ved hjælp af UML diagrammer.

Eftersom LabInfo systemet eksisterer på nuværende tidspunkt bør aktiviteten for implementering også omfatte overvejelser omkring indførelsen af LabInfo2003. Efter endt test af LabInfo2003 skal det besluttes, om det nuværende LabInfo system skal udfases eller erstattes øjeblikkelig af LabInfo2003. Overvejelsen er vigtig ikke mindst fordi systemet er omfattet af en database, hvis data der løbende foretages ændringer på.

15.1 Generering af decentrale systemer

Problemet med at få indflettet nye decentrale system som LabInfoAUH er blevet forenklet med LabInfo2003. De fælles krav som er til stede for det centrale og de decentrale system er samlet under ét delsystem på domæneserveren. Håndteringen af informationen i kodeskemaet er derfor blevet standardiseret og er ikke længere pålagt de enkelte web-applikationer. LabInfo2003 skal betragtes som et fælles system, og derfor er det mere

[8] Extensible Markup Language

korrekt at tale om web-applikationer frem for decentrale systemer. LabInfoAUH bør således betragtes som web-applikationen LabInfoAUH under LabInfo2003.

Oprettelsen af nye web-applikationer bør beskæftige sig med at fastlægge deres specifikke behov samt håndteringen af disse. Nye applikationer vil bidrage med nye måder til at anskue informationen for kodeskemaet på. Men der vil kun forekomme ændringer på domæneserveren i det omfang at informationen ikke allerede at tilgængelig for serveren. Ellers vil det være et spørgsmål om at tilpasse interfacet på web-applikationen, således at den ønskede information kan hentes fra domæneserveren.

Det kendetegnende for nye web-applikationer er, at de ikke kan ændre på eksisterende datastruktur i LabInfo2003. En eventuel manipulering af data kan kun forekomme på deres egen web-applikation.

16 Konklusion

På trods af manglende dokumentation for LabInfo systemet er det lykkedes at gennemføre en programmelanalyse. Resultatet er præsenteret i form af case diagrammer, kravspecifikationer og sekvensdiagrammer for både LabInfo og LabInfoAUH. Grundlaget for at udarbejde et kravdokument efter en given standard er dermed givet.

Formålet med at skabe bedre kommunikation for LabInfo systemet opnås ved fremover at betragte de to systemer som ét fælles system frem for ét centralt og flere decentrale systemer. Derved opnås den på forhånd ønskede fleksibilitet i forbindelse med udbygning af systemet. Den anvendte metode er en systemarkitektur, der gør brug af en "four-tier client-server" metode. Det bevirker, at LabInfo2003 er inddelt i fire delsystemer. Sidstnævnte bidrager samtidig til det ønskede overblik og forbedret mulighed for systemets videre vedligeholdelse. Sammenlægning i det nye systemdesign har åbnet op for muligheden for at skrive systemet i et objektorienteret programmeringssprog.

Den præcise specificering af de enkelte delsystemer er ikke bragt til ende. Muligheden for valg af implementeringsteknik og -sprog er dermed holdt åben. Det videre forløb med implementering bør derfor indledningsvis fastlægge de præcise systemarkitekturer for delsystemerne.

Sammenlagt har det resulteret i et forbedret systemdesign for LabInfo systemet i form af LabInfo2003.

17 Litteratur liste

[Sommerville, 2001] *Software Engineering.*
af Ian Sommerville
6th Edition, 2001
Addison Wesley

[Bennett, 1999] *Object-Oriented Systems Analysis and Design using UML*
af Simon Bennett, Steve McRobb and Ray Farmer
The McGraw-Hill companies

[IEEE, 1998] *IEEE Recommended Practice for Software Requirements Specifications*
af Software Engineering Standards Committee of the IEEE Computer Society
IEEE Std 830-1998

[LabInfoAUH, 2003] *Programbeskrivelse af LabInfoAUH, Århus Universitetshospital*
af EDB Gruppen A/S
Vedlagt som bilag.

[LabInfo, 2003] *Dokumentation af LabInfo*
af EDB Gruppen A/S
Vedlagt som bilag.

[OCL, 1997] *Object Constraint Language specifikation*
IBM Corporation, Rational Software Corporation
Version 1.1 1. september 1997

18 Appendiks

18.1 Elementoversigt

1	Administrator	En aktør på LabInfo systemet som administrerer mængden og indhold af forfattere og artikler
2	AlmeneBruger	En aktør på Labinfo systemet, der har adgang til al offentlig information.
3	Artikel	En artikel består af flere informationsfelter med information for egenskaber ved en patient.
4	Artikel status	Giver udtryk for den tilstand en artikel er i. Der er tre mulige tilstande: offentliggjort, hos redaktør eller hos forfatter.
5	Artikels navn	Omfatter følgende informationsfelter for artiklen: kodetekst, system, præfiks, komponent og enhed.
6	En liste	Benævnelsen for en artikel, der angiver bestemmelser for andre artikler.
7	Forfatter	En aktør på LabInfo systemet som har adgang til at redigere i artikler.
8	IFCC-IUPAC information	En del af informationen for en artikel og omfatter følgende informationsfelter: definition, engelsk navn, enhed og molmasse.
9	Indgår i liste	En artikel der er medlem af en liste.
10	Informationsfelt	Et felt med information for en artikel.
11	Kompendium information	En del af informationen for en artikel. Omfatter følgende informationsfelter: analysekvalitet, analyseprocedure, analyserendeLaboratorium, bemærkning, biologiskVariation, fejlkilde, forberedelse, forsendelse, glastype, henvisning, holdbarhedUbehandlet, indikation, interferens, links, listevurdering, litteraturreference, opbevaring, omregningsfaktor, overReferenceInterval, pmid, prøvemateriale, prøvetagning, referenceInterval, svartid og underReferenceInterval.
12	LabInfo system	Omfatter LabInfo, LabInfoAUH og LabInfo2003
13	Laboratorium information	En del af kompendium informationen. Omfatter analyserendeLaboratorium, forberedelse, prøvetagning, prøvemateriale, glastype, holdbarhedUbehandlet, opbevaring, forsendelse, enhed, molmasse, omregningsfaktor, analyseprocedure, analysekvalitet, svartid og fejlkilder.

14	Liste information	Viser om en artikel er en liste eller indgår i en liste.
15	Medicinsk information	En del af kompendium informationen. Omfatter indikation, biologisk variation, værdier over referenceinterval, værdier under referenceinterval, interferens, bemærkninger, henvisning, links, litteraturreference og pmid.
16	Redaktør	Andet brugt ord for administrator i systemet.
17	Reference intervaller	En del af kompendium informationen. Omfatter referenceinterval, terapeutisk interval, beslutningsintervaller og alarmgrænser.
18	Solitær artikel	Artiklen er alene om at bestemme én egenskab for en patient.

19 Bilag

19.1 Bilag 1 – Dokumentation af LabInfo

19.2 Bilag 2 – Programbeskrivelse for LabInfoAUH

Bilag 1

Bilag 2

Dokumentation af LabInfo.

1 IFCC-IUPAC kodesystemet

Sundhedsstyrelsens enhed for Sundhedsinformatik forestår videreudvikling og implementering af det internationale "IFCC - IUPAC - kodeskema" til kommunikation af laboratorierekvisitioner og -svar på nationalt plan, og understøtter den lokale implementering praktisk.

Databasen indeholder kodesystemets basisinformationer:

- Egenskabernes kode og forkortede navn på dansk
- Egenskabernes fulde navn (definition)
- Egenskabernes forkortede navn på engelsk
- Egenskabernes strukturelle sammenhæng (lister og deres indhold af egenskaber)
- Enhed og molmasse, hvor det er relevant

Der er mulighed for download af hele kodesystemet i form af en excel-fil, eller af et specificeret udtræk af informationerne i form af en tab-separeret tekstfil. Der kan genereres udtræk på op til 1300 poster, ud fra de samme kriterier som i søgefunktionen.

Søgefunktionen er i øvrigt analog til funktionen for Kompendium 2000.

Den litteratur, der beskriver den teoretiske og tekniske baggrund for systemet kan læses/downloades fra den engelsksprogede del af siten (klik på det lille flag i menuen for at skifte mellem engelsk og dansk version)

1.1 Kodesystemets struktur og funktion

1.1.1 Nomenklaturen

IUPAC-nomenklaturen er en patientrelateret nomenklatur til beskrivelse af laboratorieundersøgelser og deres resultater; alle undersøgelser er defineret og navngivet som egenskaber ved patienten eller patientens delsystem. Det adskiller det fra traditionel navngivning, hvor undersøgelser ofte beskrives ved de anvendte teknikker. (f.eks. 'mikroskopi', 'elektroforese', 'tælling'). I stedet for f.eks. 'Plasmaelektroforese' tales om 'Plasma—Proteintype' som den egenskab ved patienten, der undersøges.

Den mest synlige konsekvens af dette er, at systemet 'Serum' ikke findes i IUPAC systemet. Undersøgelser af serum udtrykker hvordan forholdene var i patientens plasma på prøvetagningstidspunktet, og de er derfor konsekvent defineret som plasmaundersøgelser.

En egenskab (undersøgelse) beskrives med en kode (f.eks. NPU02319) og en definition.

Egenskaber defineres ved deres

- System – hvilken del af patienten (blod, biopsi, urin, sekret…) der undersøges
- Komponent – hvad der undersøges for i systemet (glucose, erythrocytter, virus, antistoffer…)
- Egenskabsart – hvilken slags egenskab der undersøges hos komponenten (koncentration, volumen, følsomhed, udskillelseshastighed…..)
- Enhed – den enhed, egenskaben måles med (g/l, nmol/d, mm, arbitrære enheder….)

For nøjere at beskrive, hvad der undersøges, kan der i definitionen være tilføjet specifikationer til både system, komponent og egenskabsart.

1.1.2 Gruppestrukturer (lister)

En undersøgelse kan være solitær, dvs bestemme én egenskab.

Eller den kan omfatte bestemmelse af flere egenskaber ved patienten - en liste (f.eks. Syre-base status).

Listens navn kan betragtes som en overskrift for en gruppe af egenskaber, der rekvireres og/eller besvares samlet.

Listerne beskrevet i IUPAC kodesystemet er tænkt som eksempler, de omfatter principielt alle de egenskaber, der kan tænkes placeret under den pågældende overskrift.

Det analyserende laboratorium bestemmer så hvilket udvalg ydelsen omfatter lokalt.

De indgående egenskaber i en liste kan være komplette eller inkomplette.

En komplet egenskab beskriver selvstændigt en dimension af patientens tilstand

Eks.: B—Hæmoglobin(Fe); stofk. = ? mmol/l

Komplette egenskaber kan rapporteres og genbruges under en hvilken som helst overskrift, eller uden overskrift, uden at deres betydning ændres.

Hovedparten af egenskaber i kodesystemet er komplette.

En inkomplet egenskab definerer ikke hele dimensionen der undersøges, en del af definitionen fremgår af den liste, den indgår i (overskriften).

Under en anden liste beskriver samme egenskab noget andet vedrørende patienten.

Inkomplette egenskaber uden den tilhørende listeoverskrift har ingen mening.

Eksempler:

B—Glucose; stofk.(15 min) = 7,3 mmol/l
kan høre til i mange forskellige 'Belastningsundersøgelser', f.eks..
Tyndtarm—Lactose-tolerance; egensk.(lactose p.o.; liste)
eller
Hypofyse—Somatotropin-sekretion; stofhast.(liste; insulin i.v.)
Overskriften besvarer spørgsmålet '15 min. efter hvad?'

Syst—Penicillin; følsomhed = Sensitiv
kan tilhøre f.eks. overskriften
Sekr(Ørekanal; venstre)—Bacterium(spec.); følsomhed(liste; R I S)
eller
Sekr(Bronchus; spec.)—Actinomyces; følsomhed(liste; R I S)
Overskriften besvarer spørgsmålet 'hvilken mikroorganisme hvorfra?'

2 Kompendium i Laboratoriemedicin 2000

Ca. 6000 af IFCC-IUPAC kodesystemets (omkring 16.000) egenskaber (undersøgelser) anvendes almindeligt i Danmark.

Indsamling af klinisk og praktisk information om egenskaberne er påbegyndt år 2000 og foregår løbende.

Forfatterne - udpeget af de relevante speciallægeselskaber - skriver deres bidrag direkte til Kompendium 2000 basen via Internettet.

- Søg i basen
- Hjælp til brug af søgefunktionen
- Oversigt over forfattere

2.1.1 Formål

Formålet med Kompendium i Laboratoriemedicin 2000 er at formidle information om laboratorieundersøgelser til brugere inden for sundhedssektoren; sygehuse, laboratorier, praksis, systemhuse...

2.1.2 Indhold

Databasen indeholder information om egenskaber (laboratorieundersøgelser) inden for en række laboratoriespecialer:

- Klinisk Allergologi
- Klinisk Biokemi
- Klinisk Farmakologi
- Klinisk Genetik
- Klinisk Immunologi

- Klinisk Mikrobiologi
- Reproduktion og fertilitet
- Thrombose og hæmostase

2.1.3 Informationen omfatter:

Navne og koder ifølge IUPAC-IFCC.
Information om rekvisition, prøvetagning, referenceintervaller og vurdering af resultater, mv.
Indsamling af informationen er påbegyndt foråret 2000.

Redaktion:
Henrik Olesen

3 Brug af LabInfo's database

3.1 Søgning efter en bestemt undersøgelse

Søgefunktionen fungerer ens for IFCC-IUPAC kodesystemet og for Kompendium 2000. Der kan søges på (en kombination af) tekststrenge i egenskabens kode, navn og eventuelle synonymer.
Bemærk at navnet er forkortet for system og egenskabsart ('Csv' for Cerebrospinalvæske, 'stofk.' for stofkoncentration). Se hjælpen til de enkelte søgefelter for anvendte forkortelser.
Der kan søges på en del af et navn, og der kan yderligere indsættes en * i stedet for en del af strengen (ex. moglobin*stofk, faktor*rel.)

Som default vises i IFCC-IUPAC søgning kun aktive poster, og i Kompendium 2000 kun poster med kompendieinformation.
I IFCC-IUPAC kodesystemet kan der derudover søges efter egenskaber opdateret indenfor et bestemt datointerval, og i Kompendium 2000 efter artikler, hvor den medicinsk/tekniske information er opdateret efter en given dato.

Der er mulighed for at forfine søgningen yderligere ved at angive flere søgekriterier. Der vises kun poster, der tilfredsstiller ALLE angivne kriterier samtidig.
Specifikation af speciale kan medføre at nogle poster vises mere end én gang, hvis de er tilknyttet mere end ét af de søgte specialer.

Det er muligt specifikt at angive at der skal søges efter en indtastet streng i egenskabens uforkortede navn ('Søg også i..'). Det giver mulighed for at udsøge på f.eks. 'spinalvæske' eller 'reciprok', hvis man er i tvivl om forkortelsen. Bemærk at det skal angives hvilket felt (System eller Komponent) den søgte 'uforkortede' streng står i.

De fundne poster sorteres alfabetisk efter
1) Komponent etc.,
2) præfix og
3) System.
De vises i grupper á 10 poster (ved over 100 hits i større grupper) ad gangen. Poster fundet via synonymer vises i grupper for sig, sammen med det aktuelle synonym. Klik på de enkelte poster for at se yderligere information.

3.1.1 IFCC-IUPAC Kodesystemet

Her vises kodesystemets information om undersøgelsen: Kode, Navn, Definition, Engelsk navn, evt. Enhed og Molmasse. Ved udsøgning på Opdateringsdato vises denne i oversigten over hits. For lister vises listens indhold af egenskaber, og for egenskaber vises, hvilke lister de indgår i, med links til pågældende poster.

3.1.2 Download af IFCC-IUPAC information i tabelform

I IFCC-IUPAC søgefunktionen kan basisinformation om udsøgte poster samles i en tab-separeret tekstfil, til download og viderebehandling i egne systemer. Søgefunktionen er identisk med den almindelige IFCC-IUPAC søgefunktion, blot trykkes 'Tekstfil' i stedet for 'Søg'.

Der vises en side med et link til den dannede tekstfil. Den downloades ved at højreklikke på linket og vælge 'Save target as'/ 'Gem destination som'.

Der kan dannes downloadfiler med op til 1300 poster. Ved behov for større datamængder eller mere information om de enkelte poster og deres sammenhæng kan regneark (MS Excel)med hele datasættet downloades fra siden Download .

3.1.3 Kompendium i Laboratoriemedicin 2000

Ved klik på en undersøgelses navn vises altid IFCC-IUPAC kodesystemets generelle information om undersøgelsen: Kode, Navn, Definition, evt. Enhed og Molmasse.

Derudover kan vises medicinsk og teknisk information og referenceintervaller for egenskaberne, ved klik på de forskellige 'knapper'.

For lister, der omfatter flere egenskaber, f.eks. Syre-base-status egenskaber, vil der nederst på siden være et link for hver egenskab der indgår i listen. Ved klik på dette link vises den enkelte egenskabs information.

Tilsvarende vises for de enkelte egenskaber links til eventuelle lister, de indgår i.

Nederst på siden vises den seneste revisionsdato for den kliniske og tekniske information, og navnet på den forfatter, der er ansvarlig for den.

3.1.4 Tips og tricks

De hyppigste problemer er enten for mange eller slet ingen hits. Nedenstående er en samling forslag til at forbedre resultatet.

Hvis der ikke findes nogen poster, så kontroller søgekriterierne nøje. De forventede poster kan være fravalgt, for eksempel ved en speciale- eller datospecifikation. husk at Kompendium 2000 som default kun viser poster, der har medicinsk/teknisk information.

Ved at sætte DNK i feltet 'Kode' kan man begrænse søgningen til 'danske' enheder som U/l og g/l<

Det er svært at gætte hvor der er mellemrum og bindestreger, eller om et ord er med 'f' eller 'ph'.

Brug altid den kortest mulige søgestreng:

(Adeninphosphoribo)syltrans(ferase)

Hepatitis (B virus c-antistof(IgM); arb.k.(0 1))

<

Kopier evt. en god delstreng fra visningen af en post der ligner, med Crtrl+C og sæt den ind i Komponent el Synonym feltet med Crtrl+V. Ret den derefter til.

Et semikolon efter en komponent udelukker en del sammensatte navne:

glucose; i stedet for 'glucose' udelukker f.eks. 'UTP-glucose-1-phosphaturidyltransferase; kat.indh.'

Hvis systemet er Plasma, så forsøg med 'to bindestreger': P-- . Det udelukker systemer som 'Syst(spec.)—' og Dialysev(perit.)— (men også P(vB)—!).

Programbeskrivelse af

Labinfo

Århus Universitetshospital

Udarbejdet af EDB Gruppen A/S

Sommer 2003

Indholdsfortegnelse

Indledning

Programbeskrivelsen er tænkt som et hjælpemiddel til brug af vedligeholdelse af hjemmesiden. Dokumentation koncentrerer sig om de centrale funktioner på Labinfo, samt opbygning og brug af den bagvedliggende databases indhold. Hjemmesiden indeholder en mængde oplysende information, som er tilgængelig via almindelig navigation og af den grund ikke bliver gennemgået her.

Hjemmesiden muliggør søgning, præsentation og administration af et kodesystem med tilknyttet information. En kode og dens information vil i det følgende blive omtalt som en artikel. Yderligere information omkring kodesystem kan finde på Labinfo´s hjemmeside.

Systemkrav

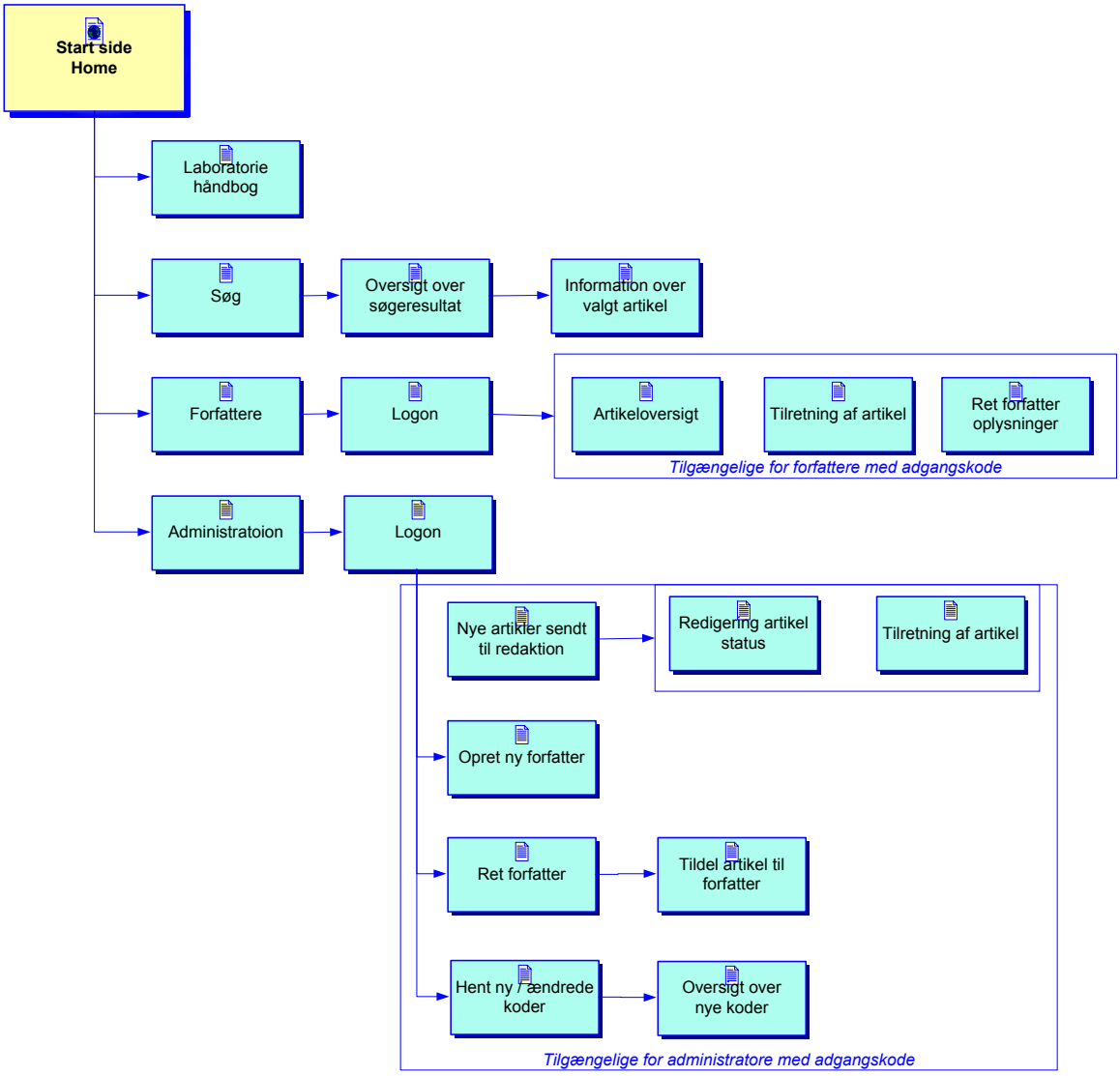
Labinfo skal afvikles på en webserver, der kan fortolke ASP[1]-sider. Hjemmesiderne er skrevet primært til Microsoft Internet Explorer serien, brug af andre browsere kan medføre ændringer i layout, men ikke i den bagved liggende funktionalitet. Informationen gemmes og hentes fra en Microsoft SQL database.

Adgangsniveauer

Labinfo er et program, der via en web-løsning præsenterer information hentet fra en database. Denne information er offentliggjort for en mængde bruger, som kan søge og lave opslag i databasen efter ønsket information. For at informationen kan forblive opdateret vil, der til programmet være tilknyttet en administrator(e), som igen kan tilknytte forfattere til dele af informationen. På den måde er det op til den enkelte forfatter at ændre og opdatere den information, som han/hun har ansvaret for. Men i den sidste ende er det administratoren, som skal bestemme om de foretaget ændringer kan godkendes og dermed blive offentliggjort.

Af ovenstående fremgår det at brugerne kan opdeles i tre forskellige grupper, en almindelig, en forfatter og en administrator. Sidst nævnte har adgang til en speciel administrations del på hjemmesiden. Er man forfatter på hjemmesiden giver det adgang til sider, hvorfra den enkelte forfatter kan administrere den information, som er ham tilknyttet. Som almindelig bruger kan man bevæge sig frit på alle resterende sider på hjemmesiden. En skematisk inddeling af brugerniveauet kan ses i Figur 1.

[1] Active Server Page

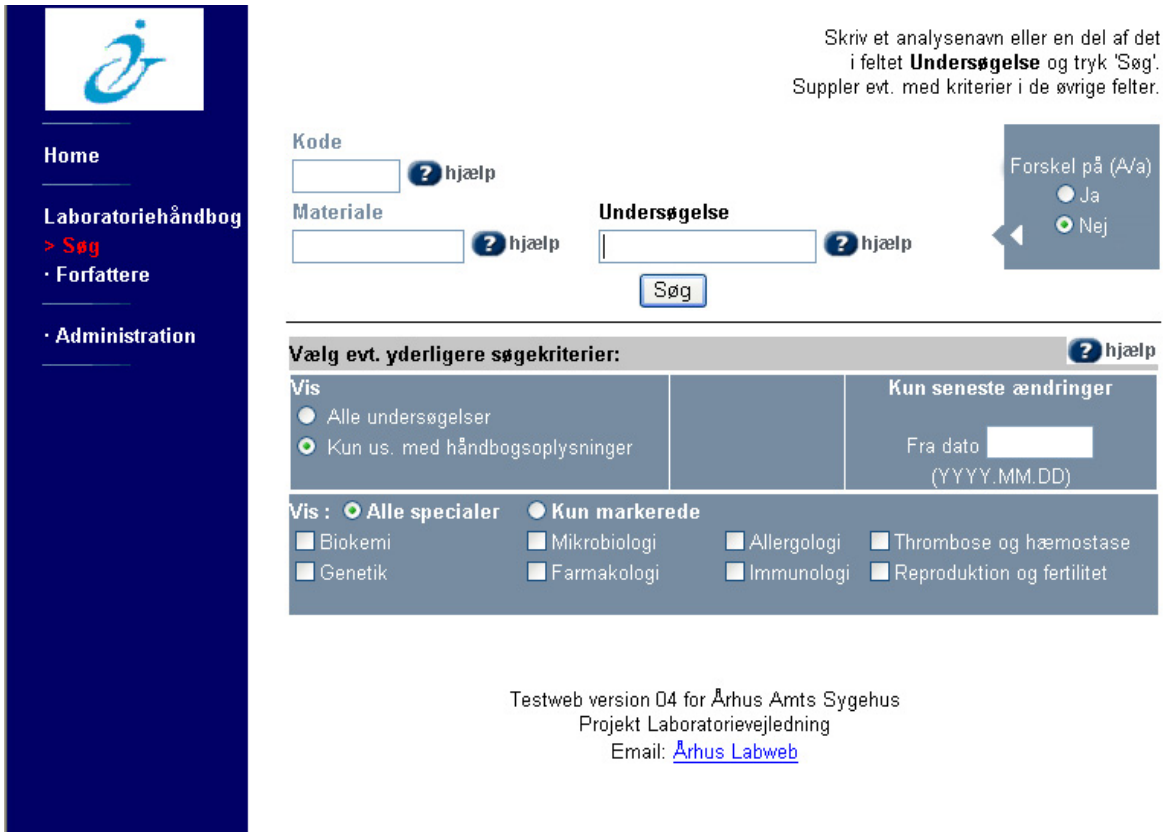


Figur 1 Træstruktur af funktionaliteten på Labinfo opdelt efter brugerniveau.

Almindelig bruger

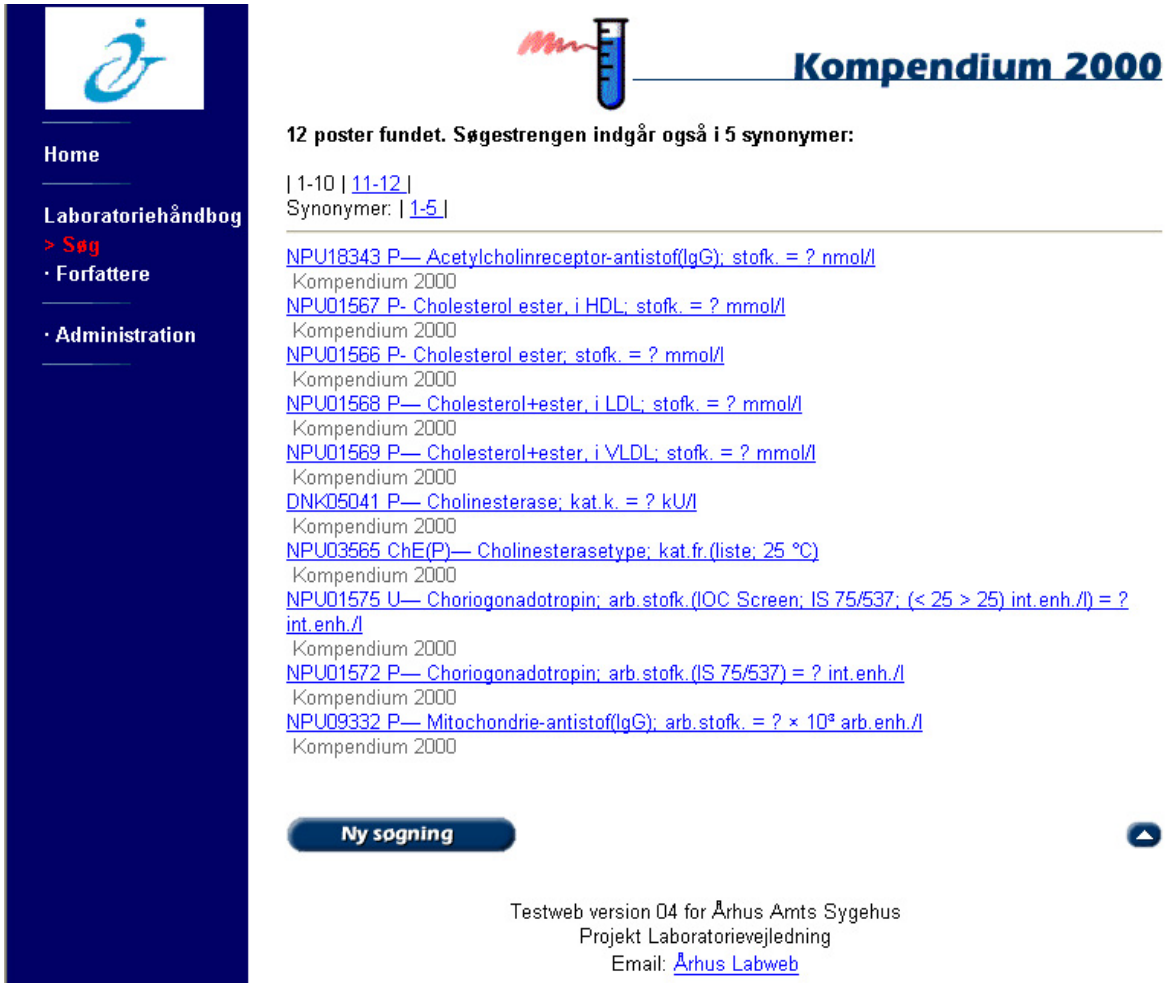
Denne betegnelse omfatter alle brugere, som har netadgang til hjemmesiden og giver mulighed for at finde generel information omkring Labinfo, samt information på alle godkendte artikler i databasen. Den generelle information omkring Labinfo findes via menupunktet ”Laboratoriehåndbog”, hvorfra der kan navigeres mellem flere sider.

Ønsker man at finde informationen for artiklerne i databasen sker det via menupunktet ”Søg”. På denne søgeside har man mulighed for at specificere sin søgning i større eller mindre grad, se Figur 2 for overblik over søgekriterierne.



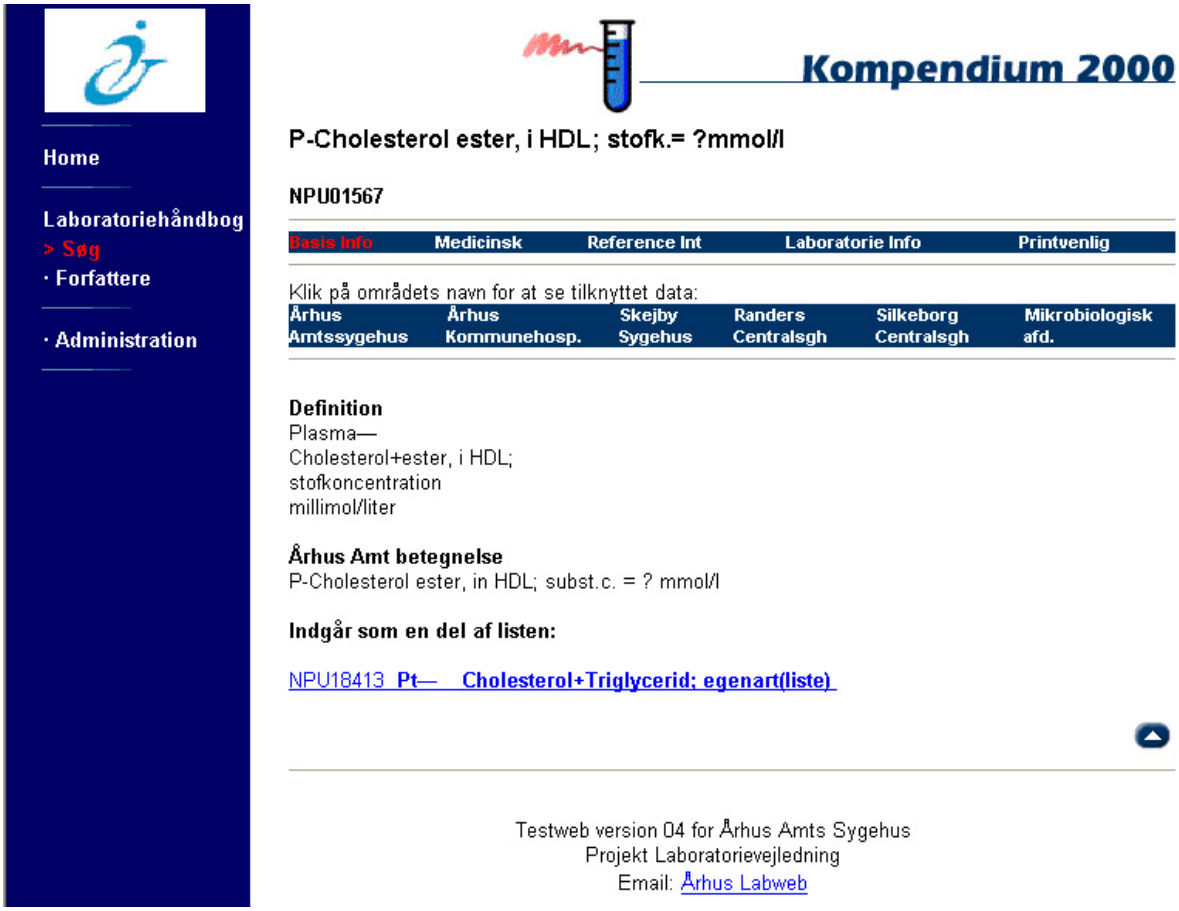
Figur 2 Søgesiden med de mulige søge -kriterier og -specifikationer.

Resultatet af søgningen er afhængig af indtastet data samt eventuelle tilvalgte specifikationer. Men fælles for alle søgninger er, at der kun søges på godkendte artikler i databasen og at eventuelle hits på synonymer vises separat, se Figur 3.



Figur 3 Resultatoversigt for en søgning, hvor synonymer også indgår.

Vælges en artikel fra listen bliver dens information vist som i Figur 4..



Figur 4 Som udgangspunkt vises basis information for en artikel, her for NPU01567.

Informationen er opdelt i passende kategorier, vis indhold vises ved at klikke på benævnelsen. På Figur 4 er basis-informationen for artiklen vist. Grundet en artikels information kan variere fra sygehus til sygehus, er det også muligt at slå sygehusspecifik information til eller fra. Dette sker ved at klikke på sygehusets navn, hvorefter eventuel information vises. Klikkes på samme sygehus igen vil dets tilhørende information forsvinde for visningen.

Forfatter

Er man oprettet som forfatter på hjemmesiden, har man fået tildelt et brugernavn og en adgangskode, som via menupunktet ”Forfattere” giver adgang til følgende:

- Ændre personlig data

- Ændre indholdet af en tildelt artikel
- Sende en redigeret artikel til godkendelse hos administratoren

Når en forfatter logger sig på systemet bliver han præcenteret for en oversigt med de artikler, som er ham tildelt af administratoren. Herfra vælges den artikel, som der skal foretages ændringer i. Forfatteren kan på efterfølgende sider ændre artiklens indhold og som afslutning må han beslutte om ændringerne skal gemmes og om artiklen skal sendes til godkendelse hos administratoren. Status for en artikel vil fremgå af oversigten over tildelte. Hvis der foretages ændringer på en godkendt artikel, vil den ændre status til ”ikke godkendt” og man vil ikke kunne søge på den fra søgesiden førend den igen har ændret status til ”godkendt”.

Administrator

Denne rolle giver adgang til følgende administrationspunkter:

- Opret forfatter
- Slet forfatter
- Tilknytte artikler til forfattere
- Fjerne artikler fra forfattere
- Godkende artikler sendt til redaktion
- Afvise artikler sendt til redaktion
- Ændre artikel-information for artikler sendt til redaktion
- Hente og tilføje eller afvise ændrede og nye artikler fra Sundhedsstyrelsens database til lokal database

Derudover har administratoren en række oversigter som letter overblikket af sammenkædningen mellem forfattere og artikler.
Som det fremgår af ovenstående kan en administrator, som det eneste, oprette en ny forfatter til database. Det er altså ikke muligt at oprette en ny artikel eller andet via hjemmesiden. Al den

information er stationær for hjemmesiden, men kan ændres manuelt ved hjælp af et database editeringsprogram[2].
På Figur 1 er administratorens funktioner illustreret, samt hvordan der navigeres frem til de enkelte funktionaliteter.

Udfra nedenstående afsnit vil man, som administrator eller ansvarlig for Labinfo, kunne danne sig et billede af strukturen bag Labinfo med henblik på at foretage ændringer uden brug af hjemmesiden.

Database opbygning

Databasen bag Labinfo indeholder en række tabeller. Disse er opstillet nedenfor med en kort beskrivelse:

- tblAdmin – Indeholder brugernavn og adgangskode for administratorer.
- tblAreas – Indeholder navn og placeringsnummer for de sygehuse som er tilknyttet Labinfo på Århus Amtssygehus.
- tblForfattere – Indeholder information på de forfattere, der er tilknyttet Labinfo.
- tblGenerelt – Indeholder information for hver enkelt kode i systemet.
- tblGeneretlSubArea – Indeholder områdespecifik information for koder, der er tildelt et sygehus.
- tlbLitteraturreferencer – Indeholder litteraturreferencer for koderne.
- tblNewKoder – Indeholder alle kodernumre, som er blevet hentet fra Sundhedsstyrelsen. Til hver kode er der tilknyttet en værdi, som indikerer kodenummerets status. Værdierne kan være 1 for at koden er ny, 2 for at koden er ændret hos Sundhedsstyrelsen, 3 for at koden er blevet gemt i lokal database og 4 for at koden er afvist.
- tblRefIntervaller – Indeholder eventuelle information for en kodes referenceintervaller.
- tblSpeciale – Indeholder forkortelse og navn på de mulige speciale områder, som informationen for en kode kan høre under.
- tblSpecialekobling – Hver kode skal eksistere her, hvor de bliver knyttet til et eller flere specialeområder.

[2] Her tænkes på programmer som Microsoft SQL Server, men det er naturligvis afhængigt af hvilken database der bruges.

- tblSvar – Hver kode skal være oprettet her, hvor det er muligt at knytte information, udfra de i tabellen eksisterende punkter, til hvert enkelt kodenummer.
- tblSvarkobling – Hver kode skal være repræsenteret her. Kodens nummer bliver knyttet til sit eget nummer eller til at andet, hvorved den bliver en del af en liste. Om en kode er medlem af en liste er altså muligt at se udfra denne tabel.
- tblSynonymer – Indeholder eventuelle synonymer for koderne.

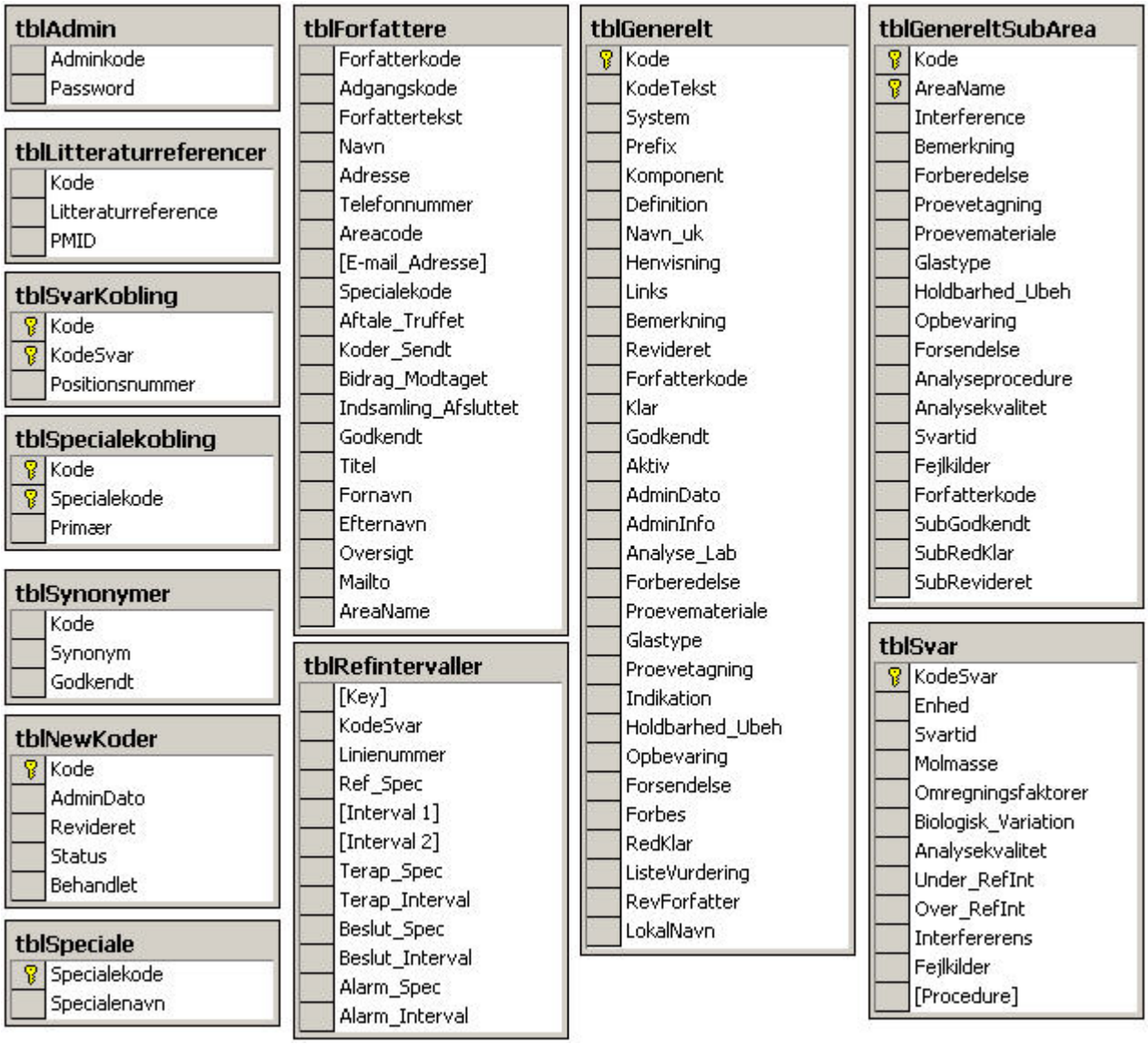
Ved hjælp af et program til redigering af Microsoft SQL Databaser kan indholdet af ovenstående tabeller modificeres efter ønske. Men det anbefales at bruge de indgange, der er tilgængelige via hjemmesiden, og på den måde undgå indtastningsfejl eller lignende.
Se Bilag 1 for et diagram over ovenstående tabellers opbygning.

Ændringer på databaseniveau

Enkelte funktionaliteter er kun tilgængelig ved at ændre/tilføje i databasen. Disse funktioner er følgende:

1. *Administrering af Administratorer* – Dette gøres ved at tilføje / ændre i tblAdmin, hvor et brugernavn og adgangskode er påkrævet.
2. *Administrere sygehuse, hvortil ekstraordinær information kan tilknyttes* – tlbAreas indeholder navne på de sygehuse, som forfatterne kan til dels og som der dermed kan tilknyttes sygehusspecifik information til. Indføres et navn på et nyt område vil det automatisk være muligt at tilføje information til det indtastede område.
3. *Indførelse af artikel kode* – Dette kræver at artiklens kode bliver oprettet i flere af databasens tabeller, hvor ekstra information kan være påkrævet. Kodenummeret skal indføres i følgende tabeller tblGenerelt, tblLitteraturreferencer, tblSvar, tblSvarKobling og tblSpecialekobling. Generelt for disse tabeller er at alle nøglefelter skal være udfyldt, dette vil brugeren dog automatisk blive gjort opmærksom på af det brugte redigeringsprogram. Disse nøglefelter fremgår på Bilag 1.

Bilag



Bilag 1 Diagram over databasen bag Labinfo.