Vejledning til kravspecifikation SL-07

Problem-orienterede krav v5

Søren Lauesen 2017

Køb vejledningen som et hæfte på
www.amazon.de eller www.amazon.co.uk

ISBN-13: 978-1523319589

Søren Lauesen
Vejledning til kravspecifikation SL-07 - Skabelon med eksempler v5

Version 5, 2017
Denne version svarer næsten ordret til den engelske version 5

Layout og omslag: Forfatteren

Omslagsbillede: Rob Gonsalves: "The Sun Sets Sail"
Venligt stillet til rådighed af Saper Galleries, East Lansing, Michigan, USA

Billedet symboliserer overgangen fra krav (broen) til produkt (skibet)

Indhold

1. Skabelonens formål 5
1.1. Pas på skabelonblindhed 5
1.2. De største problemer med kravene 6
1.3. Det rigtige kravniveau 6
1.4. Præcise (verificerbare) krav 7
1.5. Dæk kundens formål med systemet 7
1.6. Tidlig afdækning af store risici 8

2. Indsamling af krav 8
2.1. Centraliser arbejdet 8
2.2. Inddrag interessenter og evt. leverandørerne 9
2.3. Tidlig ændringsstyring 9

3. Kontraktspørgsmål 10
3.1. Når løsningen ikke opfylder behovene 10
3.2. Ret til at opsige kontrakten og vælge en anden leverandør 10

4. Indhent tilbud og vurder dem 11
4.1. Alternative løsninger 12
4.2. Optioner 12

5. Test af systemet 13

6. Vejledning til skabelonens dele 14

A. Baggrund og vejledning til leverandøren 16
A1. Baggrund og vision 16
A2. Vejledning til leverandøren 18
A3. Overordnet løsning og alternativer 18

B. Overordnede behov 20
B1. Flow 20
B2. Forretningsmæssige mål 22
B3. Tidligt bevis for gennemførlighed (proof of concept) 24
B4. Minimumskrav og tildelingskriterier 24
B5. Nettogevinst over 5 år 28
B6. Vægtede point pr. DKK 30

C. Task systemet skal støtte 34
Arbejdsområder 34
C1. Regler for task (Indskriv patient inden ankomst) 36
C2. Lignende task (Indskriv akut) 38
C10. Et komplekst task (Udfør klinisk session) 38
C11. Et langt subtask (Ordinér medicin) 40
C20. Andre omgivelser (udfør klinisk session mobilt) 40
Hvorfor ikke user stories eller use-cases 40

D. Data systemet skal opbevare 44
Data model (E/R) 44
D0. Fælles felter 46
D1. Databeskrivelse (Diagnose) 46
D2. En typeklasse (Diagnosetype) 48
D3. Vis eksisterende tabeller og skærmbilleder (Ydelse) 50

E. Andre funktionelle krav 54
E1. System-genererede hændelser 54
E2. Udskrifter og rapporter 54
E3. Forretningsregler og komplekse beregninger 56
E4. Udbygning af systemet 58
59

F. Integration med eksterne systemer 60
SOA eller datareplikering? 62
F0. Fælles integrationskrav 62
F1. Simpel envejs integration (SKS) 64
F2. Tovejs integration (LabSys) 66
F10. Integration med nye eksterne systemer 68

G. Teknisk it-arkitektur 70
G1. Eksisterende hardware og software 70
G2. Nyt hardware og software 70
G3. Leverandøren har driftsansvar 71

H. Sikkerhed 72
H1. Log-in og adgangsret for brugere 72
H2. Sikkerhedsadministration 74
H3. Sikring mod tab af data 76
H4. Sikring mod utilsigtet brugeradfærd 76
H5. Sikring af personlige data (privacy), fx GDPR 78
H6. Sikring mod trusler 80

I. Brugervenlighed og design 82
I1. Indlæring og effektivitet i daglig brug 82
I2. Tilgængelighed og Look-and-Feel 86

J. Andre krav og leverancer 88
J1. Andre standarder der skal følges 88
J2. Uddannelse 88
J3. Dokumentation 90
J4. Datakonvertering 90
J5. Installation 90
J6. Test af systemet 92
J7. Udfasning 92

K. Kundens leverancer 94

L. Drift, support og vedligehold 96
L1. Svartider 96
L2. Tilgængelighed (driftseffektivitet) 100
L3. Datalagring 100
L4. Support 102
L5. Vedligehold 104

7. Litteratur og andre skabeloner 106

Baggrund

Systemudviklere og IT-konsulenter spørger ofte efter en eksemplarisk kravspecifikation som udgangspunkt for deres eget projekt. SL-07 er sådan en kravspecifikation. Det er en skabelon udfyldt med et indviklet eksempel: *Krav til en elektronisk patientjournal (EPJ)*.

Dette hæfte forklarer hvorfor kravene er formuleret som de er, hvad man skal passe på, hvordan kravene hænger sammen med kontrakten, osv.

Du kan hente den udfyldte skabelon her:

http://www.itu.dk/people/slauesen/SorenReqs.html#SL-07

SL-07 er oprindeligt baseret på erfaringer fra offentlige udbud efter EU-reglerne, især når systemet forventes at være et standard/rammesystem, sådan at store dele af det eksisterer allerede. Senere viste SL-07 sig at være nyttig også i andre slags systemanskaffelser, samt for produktudvikling og agile in-house projekter.

I 2007 skrev jeg store dele af skabelonen og vejledningen på opfordring fra VTU (Ministeriet for Videnskab, Teknologi og Udvikling) som led i arbejdet med Statens standardkontrakt for IT anskaffelser (K02).

Tidligere udgaver af skabelonen er blevet brugt med succes i over 100 projekter af meget forskellig art, udbudsforretninger såvel som in-house, agile såvel som vandfald, fx styring af hjemmehjælp i en kommune, inkl. ruteoptimering; en medicinalvirksomheds innovative dokumentstyringssystem; elektronisk patientjournal; lagerstyring for udstyr til filmproduktion; skadeanmeldelser til bilforsikringer med brug af GIS til at dokumentere situationen.

Mit web-site indeholder den fulde kravspecifikation inkl. leverandørens tilbud til Y-Fondens sagsbehandlingssystem. Der er også beslutningsnotatet til bestyrelsen om leverandørvalget, liste af fejl/ændringsønsker, etc.
http://www.itu.dk/people/slauesen/Y-foundation.html

Erfaringerne fra disse 100+ projekter er indarbejdet i denne udgave.

Jeg har erfaret at SL-07 virker rigtig godt i praksis - når man har lært at bruge den. Selvom det ser let ud, får de fleste galt fat i princippet i første omgang, især task (arbejdsopgaver) i kapitel C. Men med lidt hjælp, bliver det rigtigt. Mange bliver meget dygtige og har hjulpet med at forbedre SL-07.

Jeg er interesseret i løbende at forbedre kravskabelonen og hører derfor gerne om både problemer og fordele ved at anvende den. Hvis du prøver SL-07 på egen hånd, er du velkommen til at bede mig om råd.

Søren Lauesen
IT-Universitetet i København, november 2017
slauesen@itu.dk
http://www.itu.dk/people/slauesen

1. Skabelonens formål

Krav til IT-systemer kan formuleres på mange måder. Hovedprincippet i SL-07 er at være *problemorienteret* i stedet for løsningsorienteret: Lad være med at beskrive hvad systemet skal gøre. Beskriv hvad det skal bruges til og hvilke problemer det skal løse.

Det er meget lettere at beskrive hvad systemet skal bruges til, end at finde en mulig løsning. Lad leverandøren finde en løsning, være innovativ og bruge det han har allerede.

Skabelonen opnår dette ved at have kravene i to kolonner. Kolonne 1 viser kundens behov, dvs. hvad systemet skal bruges til. Kolonne 2 bliver til den løsning leverandøren foreslår. I begyndelsen er kolonne 2 enten tom eller også viser den en løsning som kunden forestiller sig. Afhængig af hvad slags projekt der er tale om, kan parterne samarbejde om at forbedre løsningen og/eller ændre behovene. Eller kunden kan vælge en af flere leverandører ud fra hvor godt deres løsning dækker behovene.

Erfaringen er at kolonne 1 (behovene) er meget stabil, mens kolonne 2 (løsningen) ændrer sig når parterne opdager de forskellige muligheder. Dette gør også SL-07 velegnet til agil udvikling.

Når kunde og leverandør er to forskellige virksomheder, vil der som regel også være en kontrakt. Kravspecifikationen vil være et bilag til kontrakten. Der er ingen faste regler for hvad der skal stå i kontrakten og hvad der skal være bilag.

Kravskabelon SL-07 bruger en elektronisk patientjournal (EPJ) som gennemgående eksempel. Eksemplet er forsimplet lidt for at gøre det mere forståeligt for personer uden for hospitalsområdet. EPJ-området er meget komplekst, så eksemplet illustrerer, hvordan man kan håndtere vanskelige krav. Kun nogle få krav måtte illustreres med eksempler uden for EPJ.

Du kan genbruge store dele af eksemplet i andre projekter. Du skal dog ikke blindt genbruge dele i **blåt**. De er meget EPJ specifikke. **Røde dele** er leverandørens tilbud. Dele med gul baggrund er råd til kunden, og ikke relevant for leverandøren. Slet dem i den færdige kravspecifikation.

1.1. Pas på skabelonblindhed

En skabelon kan let give skabelonblindhed: Dit verdenssyn indsnævres til det skabelonen beskæftiger sig med.

Skabelonen dækker ikke alt

Skabelonen viser typiske krav inden for hvert kravområde, men omfatter ikke alt. Tilføj de krav der er nødvendige i dit eget projekt. Lyt omhyggeligt til ledelsen og brugerne, og sørg for at deres behov er dækket af kravene på en eller anden måde. Ofte beder de om en bestemt løsning. Skriv den i kolonne 2 og pas på den ikke bliver til et krav i kolonne 1.

Skabelonen omfatter for meget

Samtidig kan skabelonen omfatte mere end nødvendigt for dit projekt. Det betyder at man let kommer til at medtage unødvendige krav. Resultatet kan blive et alt for dyrt system, eller at ingen leverandører sender et tilbud. Fx indeholder skabelonen

krav om at kunden selv kan udvide systemet. Det er dyrt, men vigtigt i et EPJ-system. I de fleste andre projekter er det unødvendigt.

Kig på hvert krav og spørg: Hvad ville der ske, hvis vi fik et system der ikke opfyldte dette krav? Hvis det ikke gør en forskel, så er kravet overflødigt.

Skabelonen medtager nogle strenge krav

Et krav kan være relevant, men for krævende. Fx kræver skabelonen svartider omkring et sekund for systemer i intens daglig brug. Men hvis systemet er et web-site som sjældent bruges, kan man nøjes med væsentligt længere svartider.

1.2. De største problemer med kravene

Erfaringer fra udbudsforretninger viser at en række problemer går igen fra sag til sag. Denne vejledning hjælper med at undgå følgende problemer.

a. Kravene er på et forkert niveau. De kan være for løsningsorienterede så der højst er en enkelt leverandør der kan opfylde dem. Eller de kan være så forretningsorienterede at leverandøren ikke kan tage ansvaret for dem.

b. Kravene er for upræcise til at man kan verificere dem. Dvs. at man ikke kan teste om de er opfyldt. Eller de kan være så åbne at man ikke kan sammenligne leverandørernes tilbud.

c. Kravene dækker ikke kundens formål med systemet. Dvs. at selvom kunden får opfyldt kravene, bliver de egentlige behov og forretningsmæssige mål ikke nået.

d. De store risici viser sig for sent. Typisk ser man at store dele af funktionaliteten leveres tidligt, og kunden tager dele af systemet i brug. De vanskelige ting udskydes til senere. Det viser sig til sidst at leverandøren ikke kan levere disse vanskelige ting, men på grund af det fremskredne tidspunkt bliver kunden nødt til at acceptere systemet alligevel.

Vi uddyber disse problemer i det følgende.

1.3. Det rigtige kravniveau

Kravspecifikationen skal ikke beskrive systemet for detaljeret, for så risikerer man at højst en enkelt leverandør kan leve op til kravene. På den anden side må den ikke være så overordnet at leverandøren ikke kan tage ansvaret for kravene. Der skal være en balance mellem de to yderpunkter. Vi skelner mellem fire kravniveauer:

> Krav 1 (målsætningsniveau: for forretningsorienteret). *Systemet skal sikre at antallet af fejlmedicineringer reduceres fra de nuværende 10% til 2%.*

Kommentar: Dette krav er på for højt niveau. Det omfatter forretningsmæssige forhold som er kundens ansvar. Leverandøren kan ikke opfylde kravet alene. Kundens indsats er også nødvendig, fx uddannelse af personale og registrering af det nødvendige data.

> Krav 2 (domæne-niveau: tilpas balance). *Systemet skal støtte task C1 til C7.*

Kommentar: Et task beskriver hvad de to parter, bruger og system, tilsammen skal gøre for at få udført en bestemt arbejdsopgave. Task ligner "use cases", men beskriver ikke hvem der gør hvad. I et task kan man også skrive at noget er et problem der skal reduceres. Man behøver ikke beskrive hvordan. Leverandøren kan

tage ansvar for denne slags krav, og de kan opfyldes på flere måder. Skabelonen bruger denne metode.

> Krav 3 (produktniveau: en krævet funktionalitet med uklart formål). *Systemet skal kunne vise en oversigt over patientens diagnoser.*

Kommentar: Vi kan ikke se formålet med denne oversigt. Er det at finde en behandling, forklare et nyt symptom, eller skrive en epikrise (udskrivningsbrev)? Det betyder at vi ikke kan vurdere om leverandørens løsning er god nok. Dette er den traditionelle måde at skrive krav (IEEE 830 standard) og en væsentlig årsag til at kunderne ikke får hvad de har brug for - selvom de får hvad de har bedt om.

> Krav 4 (design-niveau: for løsningsorienteret). *Systemet skal kunne vise patientens diagnoser som en hierarkisk struktur. Ved at klikke på plus og minus skal man kunne se underordnede og overordnede diagnoser.*

Kommentar: Dette krav beskriver en løsning. Det er inspireret af et bestemt system som kunden har set. En leverandører med en anden, måske bedre løsning, må afvises fordi den ikke opfylder dette krav.

1.4. Præcise (verificerbare) krav

Kravene skal være så præcise at de kan *verificeres*, dvs. at vi kan afgøre om de er opfyldt. Præcision har intet at gøre med kravenes niveau. Fx kan krav 3 og 4 ovenfor verificeres når systemet leveres. Krav 1 kan verificeres når systemet har været i brug i nogen tid.

Krav 2 kan også verificeres, men på en gradskala. Nogle systemer kan støtte task godt, andre mindre godt, men dog tilstrækkeligt. Kundens medarbejdere kan vurdere *hvor* godt ved at gennemgå alle task med leverandøren, mens man ser på skærmbillederne - eller skitser af dem - og noterer hvor godt tasket støttes (se kapitel 4). Sådan en vurdering er vigtig for at vælge den bedste leverandør.

Her er et krav der ikke kan verificeres. Det er uklart hvordan man skal måle "let at bruge" og afgøre om det er godt nok:

> Krav 5 (ikke verificerbart). *Systemet skal være let at bruge.*

Et krav kan være verificerbart, men samtidig så vagt at man ikke kan sammenligne de tilbudte løsninger. Her er et eksempel:

> Krav 6 (for åbent: svært at sammenligne leverandørerne). *Leverandøren bedes beskrive sin integrationsstrategi.*

Kommentar: Dette krav kan verificeres allerede når tilbuddet foreligger. Det eneste man skal gøre er at tjekke at leverandøren har beskrevet en strategi. Men det er svært at sammenligne strategierne fordi leverandørerne har skrevet "romaner".

1.5. Dæk kundens formål med systemet

I praksis ser man mange systemer der opfylder alle krav, men alligevel bliver de ikke en succes. Brugernes behov dækkes ikke og de forretningsmæssige mål nås heller ikke.

Vi kan sikre os at brugernes behov er dækket ved at beskrive de task systemet skal støtte og kontrollere at de faktisk støttes. Skriver vi i stedet krav på produktniveau

eller design-niveau, kan vi få et system der gør som vi bad om, men ikke støtter task tilstrækkeligt effektivt.

Det er sværere at dække de forretningsmæssige mål. I mange projekter ser man udmærkede forretningsmæssige målsætninger, men ingen har overvejet hvordan de skal opnås og hvordan det nye system skal bidrage. Resultatet bliver ofte at gevinsterne udebliver. Afsnit B2 i skabelonen viser en simpel måde at forbinde de forretningsmæssige mål med kravene. Brugt rigtigt kan det hjælpe med at identificere de forretningsmæssige mål og finde innovative løsninger.

1.6. Tidlig afdækning af store risici

De vanskelige ting i projektet er ofte svartider når det fulde antal brugere nås, brugervenlighed og integration med andre systemer. Mangler på disse områder kan i praksis ikke udbedres sent i projektet.

Afsnit B3 i skabelonen stiller krav om et tidligt bevis (proof-of-concept, POC), dvs. en kontrol af at disse ting realistisk kan nås. Et sådant bevis er dyrt, og derfor er det ikke rimeligt at leverandøren skal gøre det uden en underskrevet kontrakt. Til gengæld skal han gøre det hurtigt efter underskriften. Kan han ikke levere et tidligt bevis, kan kunden opsige kontrakten.

2. Indsamling af krav

Arbejdet med at skrive kravene kan virke overvældende, specielt i store organisationer. Det er derfor fristende at uddelegere skrivningen til de enkelte afdelinger og lade en central gruppe redigere det hele sammen. Dette må stærkt frarådes af flere grunde:

a. Hver afdeling vil se på sine egne behov og har svært ved at se det hele fra den samlede organisations synspunkt. Kravene vil derfor afspejle de eksisterende arbejdsgange uden plads til nytænkning og forbedringer på tværs af afdelinger.

b. Afdelingerne har sjældent ekspertise til at skrive krav, og kvaliteten af kravspecifikationen bliver ringe.

c. Den centrale gruppe har ikke opnået den fornødne viden til at forstå afdelingens krav, og kan derfor ikke forbedre resultatet - bortset fra sproglige tilretninger. En gruppe udtrykte det således:

   Vi forstod ikke hvad de ville have, men redigerede det sammen og sendte det til leverandørerne. Vi regnede med at de forstod hvad det handlede om. Først langt senere gik det op for os at leverandørerne heller ikke forstod det. De lod bare som om de gjorde, og tænkte "det må vi finde ud af senere".

2.1. Centraliser arbejdet

Lad en lille arbejdsgruppe udføre det meste af arbejdet:

1. Indsaml behov, visioner og ønsker fra de forskellige interessenter (herunder afdelingerne, ekspertbrugere, ledere og kundens klienter).
2. Skriv det om til krav i overensstemmelse med denne vejledning og skabelon.
3. Gennemgå kravene med interessenterne og foretag de nødvendige ændringer.
4. Send kravene i udbud (se kapitel 4 nedenfor).

Arbejdsgruppen bør være på 3-5 medlemmer, sammensat så der er ekspertise om flest mulige arbejdsområder, inkl. IT-funktionen. Mindst et af medlemmerne skal have kompetence i kravspecifikation.

Erfaringer med denne arbejdsform viser at det samlede arbejde kan reduceres til en femtedel. Samtidig går kvaliteten af kravspecifikationen drastisk op.

2.2. Inddrag interessenter og evt. leverandørerne

Selvom arbejdsgruppen har bred ekspertise, kan den ikke vide alt. Det er vigtigt at inddrage interessenterne undervejs. Det kan ske på mange måder, fx:

1. Interview af brugere, såvel ekspertbrugere som menige brugere. Spørg om eksisterende task, problemer i den måde de udføres på i dag, ønsker og visioner om fremtiden.
2. Få brugerne til at vise hvordan de udfører forskellige task, specielt de sjældne, men vanskelige.
3. Indsaml relevante dokumenter, fx rapporter og blanketter der bruges i dag, print af skærmbilleder, dokumentation af den eksisterende database og tekniske grænseflader til systemerne, statistikker og driftsrapporter.
4. Workshops hvor man sammen prøver at kortlægge de tværgående arbejdsgange og de ideelle arbejdsgange.
5. Forskellige former for brainstorm og fokusgrupper hvor man inspirerer hinanden til nye måder at gøre tingene på.
6. Når der skal indføres nye arbejdsgange, så design dem ret præcist. Hvis kundens klienter fx skal bruge elektronisk kontakt med kunden i stedet for personlig kontakt, skal kundens medarbejdere arbejde på en anden måde. Dette er ofte dårligt planlagt. Beskriv de nye task med notationen i kapitel C og udfør disse task som et rollespil for at kontrollere at de fungerer korrekt.
7. Besøg hos potentielle leverandører. Tit ved de hvordan andre udnytter deres produkt, og de kan henvise til kunder der bruger det. De kan også fortælle om muligheder kunden ikke selv har tænkt på, eller nye måder at gøre tingene på.

Undgå at opstille alle disse oplysninger som krav. Det bliver let til en lang ønskeseddel med krav på et alt for løsningsorienteret niveau og risiko for favorisering af enkelte leverandører. Spørg i stedet: Hvorfor er dette ønske interessant? Hvornår skal det bruges? Hvad er formålet? Hvilke task får gavn af det? Resultatet bliver bredere behov der kan formuleres som krav.

2.3. Tidlig ændringsstyring

Under indsamlingen af krav samler man en masse ideer, ønsker, problemer og potentielle krav. Deltagerne kan bruge oceaner af tid på at beslutte hvad der skal med, og det kan blokere arbejdet. I stedet bør man vedligeholde en liste af udestående punkter så gruppen kan komme videre. Nogle kalder listen *fejl og ændringsønsker.*

Med jævne mellemrum gennemgår man listen og beslutter hvad der skal gøres til krav, til mulige løsninger, afvises eller blive stående på listen. Man ser ofte at punkter der oprindeligt så ud til at være umulige at håndtere, senere finder en simpel løsning.

Fortsæt med ændringsstyringen efter kontraktunderskrivelsen. Nu viser det sig at kolonne 1 (behovene) er ret stabile, mens kolonne 2 (løsningerne) ændrer sig når man opdager de forskellige muligheder.

3. Kontraktspørgsmål

Når systemet udvikles in-house er der sjældent en egentlig kontrakt. Kravene angiver hvad der skal leveres. Kravændringer diskuteres undervejs, og der er ikke økonomiske sanktioner mellem parterne.

Men når kunde og leverandør er forskellige virksomheder, er der som regel en kontrakt *og* en kravspecifikation. Kravene specificerer hvad der skal leveres, og kontrakten specificerer hvad der skal ske når det ikke går som forventet. Hvad skal der fx ske hvis leverandøren ikke leverer til tiden; eller hvis kunden har glemt et vigtigt krav.

Jurister med it-kontrakter som speciale er gode til at formulere regler om alle de mange ting der kan gå galt under projekt, på samme måde som programmører er gode til at håndtere alle de mange situationer der kan opstå når systemet kører.

Som regel er kravene et eller flere bilag til kontrakten. Andre bilag indeholder leverandørens lange beskrivelse af løsningen, prisen for leverancerne, tidsplanen, projektledelsen, test, osv.

SL-07 bruger nogle principper der skal afstemmes med kontrakten:

3.1. Når løsningen ikke opfylder behovene

I SL-07 står alle krav i tabeller. Kolonne 1 specificerer kundens behov, fx at et bestemt subtask skal støttes. Kolonne 2 skitserer et eksempel på en løsning og senere - i kontrakten - leverandørens løsning (se eksemplet i afsnit A2). Leverandøren kan give en længere beskrivelse af løsningen i en løsningsnote eller et separat bilag.

Hvad sker der hvis det på leveringstidspunktet viser sig, at leverandørens løsning ikke dækker kundens behov? Hvem skal betale for en forbedret løsning? I mange lande er det kundens problem - han accepterede løsningen ved at underskrive kontrakten. I andre lande er reglen at beskytte den svage part - den der har svært ved at forstå det tekniske - i dette tilfælde kunden.

Danske standardkontrakter undgår tvivlen ved at skrive at kundens behov har *forrang* (højere prioritet). Leverandøren er ansvarlig for at opfylde kundens behov. Han er ansvarlig for at hans løsning er god nok.

3.2. Ret til at opsige kontrakten og vælge en anden leverandør

De fleste krav er ikke risikable. Hvis de er blevet "glemt", kan man let opfylde dem sent i projektet, fx hvis man har glemt et par felter i databasen. Andre er meget risikable. De er så afhængige af systemets arkitektur at man ikke kan håndtere dem sent i projektet.

For at reducere risikoen bruger SL-07 et tidligt proof-of-concept (POC, afsnit B3). Kunden - og evt. også leverandøren - har ret til at opsige kontrakten hvis det tidlige bevis ikke er tilfredsstillende. Dette skal præciseres i kontrakten.

Kunderne har ofte svært ved at bruge denne ret og opsige kontrakten, selv hvis det er tydeligt at forventningerne ikke er opfyldt. Kunden har allerede investeret tid og anstrengelser, og desuden skal han gentage hele udbudsprocessen. Gør det lettere for kunden: Anfør i udbudsmaterialet at leverandørens tilbud skal være gyldigt i en passende periode efter at vinderen er udpeget. Forklar at dette gør det muligt for

kunden at vælge det næstbedste tilbud, hvis det viser sig at det bedste tilbud ikke lever op til det tidlige bevis.

4. Indhent tilbud og vurder dem

Hvis man bare sender kravspecifikationen til nogle udvalgte leverandører og beder om et tilbud, er der stor chance for at kun få leverandører svarer, og det de svarer forstår man ikke. Vi har gode erfaringer med denne metode:

a. Send kravspecifikationen til de udvalgte leverandører og bed om et møde med hver af dem (højst 3 timer). Ved mødet skal leverandøren vise hvordan hans system kan opfylde kravene. Udpeg evt. enkelte krav eller task der er særlig interessante. Bed ham pege på krav der er uhensigtsmæssige eller mangler. Han forventes ikke at skrive noget. Leverandører kan holde sådan et møde med en beskeden indsats.

b. Tillad mindst en dag mellem to leverandørmøder. Din hjerne skal komme sig.

c. Brug resultaterne til at revidere kravspecifikationen. Send den til de leverandører der er interessante (kan være flere end dem man mødte i første runde). Bed om et formelt tilbud, hvor de skriver deres løsning i højre side, angiver priser, mv.

d. Hold afklarende møder med hver tilbudsgiver. Vælg så en af dem.

Ved et EU udbud skal kunden vurdere tilbuddene på en numerisk skala og vælge tilbuddet med højest score. I andre slags projekter kan det også være en god idé at vurdere på en numerisk skala, selvom det ikke er krævet.

Det grundliggende princip er at kunden ser på hvert krav og vurderer hvor godt løsningen opfylder det. Det er bedst at se beviser for det i stedet for subjektive meninger. Lad de relevante interessenter deltage i vurderingen af de forskellige kravområder.

Lad os fx se på kravet om at støtte et bestemt task. Sammen med medarbejdere der kender dette arbejdsområde, udfører vi tasket ved hjælp af leverandørens tilbudte system. Undervejs noterer vi hvor godt tasket støttes. Vi kan prøve selv, eller lade leverandøren vise os hvordan. Hvis dette ikke er muligt fordi de nødvendige dele af systemet ikke findes endnu, må vi basere vurderingen på leverandørens skitse af skærmbillederne eller andre forklaringer af løsningen. I dette tilfælde bør vi også notere at der er en risiko for at det ikke vil virke i praksis.

Baseret på notaterne kan vi give en samlet score for støtten til dette task. Afsnit B5 af skabelonen viser hvordan man kan beregne score i beløb, B6 hvordan man kan bruge point.

For andre slags krav kan man bruge samme princip. For integrationskrav kan leverandøren fx vise hvordan eksisterende integrationer virker, eller forklare hvordan de vil virke. For dokumentationskrav kan kunden se på leverandørens eksisterende dokumentation. For brugervenlighedskrav kan kunden udføre usability-tests eller tale med eksisterende brugere af et lignende system, som leverandøren har leveret.

Afsnit B5 til B6 af skabelonen viser hvordan man kan kombinere de mange scores til en enkelt score for hele tilbuddet. Afsnittene viser også hvordan man garderer sig

mod at tilsyneladende uvæsentlige kravområder støttes så dårligt at hele systemet bliver en fiasko.

4.1. Alternative løsninger

En leverandør kan sende et tilbud med alternative løsninger. Det er fx nyttigt hvis han kan levere en dyr løsning der fuldt ud opfylder kundens krav, og en alternativ, meget billigere løsning, der ikke opfylder alle krav, men formentlig er god nok. Han kan tilbyde alternativer for flere krav eller kravområder.

Dette kan blive en stor belastning for kunden, som skal vurdere alt dette, måske i flere kombinationer. Af denne grund forbydes alternative løsninger i mange udbudsforretninger.

På den anden side er det risikabelt at forbyde alternativer. Kravet i afsnit A2 viser et virkeligt eksempel, hvor kunden havde krævet en tilgængelighed på 99,5%. Leverandøren havde to driftsmuligheder (99,0% til 3 mio. om året og 99,8% til 15 mio. om året), men måtte ikke tilbyde alternativer. Så han tilbød 99,8%, som kunden accepterede. Kunden kunne nemt have klaret sig med 99,0% og kom således til at spilde 12 millioner DKK om året, fordi han ikke tillod alternativer.

Hvis kunden vurderer tilbuddene med et beskedent antal del-kriterier (fx som afsnit B5 og B6), er det ret let at vurdere den marginale forskel mellem to alternativer og den marginale forskel på den samlede score. Man kan bruge dette princip:

1. For et sæt af alternativer, brug den første som udgangspunkt og beregn den samlede score. For hver af de andre, vurder den marginale effekt på den samlede score (inklusiv omkostningerne).
2. Hvis der er en klar forskel, så vælg det bedste alternativ. Hvis ikke, så foretag ikke noget valg nu. Det kan gøres når kontrakten er underskrevet.
3. Brug samme metode for de andre sæt af alternativer. Resultatet bliver den bedste kombination af alternativerne og en samlet score for hele tilbuddet.

For eksemplet i A2 bliver resultatet at kunden vælger det billige alternativ, med mindre der er en væsentlig forretningsmæssig fordel ved det dyre alternativ.

For at princippet ovenfor kan give gode resultater, skal sættene af alternativer være uafhængige af hinanden. Det må leverandøren sørge for.

4.2. Optioner

Ovenfor var det leverandøren der definerede alternativerne. Kunden kan også definere nogle; de kaldes optioner. Fx vil kunden måske have et data warehouse som en del af leverancen og beder om en separat pris på det. Når han får prisen, skal han så sige ja eller nej til optionen? Og har det indflydelse på hvilken leverandør han skal vælge?

Han kan træffe dette svære valg på samme måde som for alternativerne: Beregn den marginale effekt på den samlede score og sig ja til optionen hvis det giver den højeste score.

5. Test af systemet

Før kunden accepterer systemet, skal han teste det - eller få en anden til at teste det. Hvis han ikke gør det, kan han tabe sin ret til at reklamere over fejl der opdages senere. Han skal bevise at rimelige test på leveringstidspunktet ikke ville have fundet fejlen.

Som minimum skal kunden verificere alle krav, dvs. teste at de er opfyldt. Men mange fejl har ikke noget med konkrete krav at gøre, bortset fra en rimelig forventning om at systemet ikke bryder sammen når brugeren gør mærkelige ting, når kommunikationslinjerne fejler, osv. For at teste for den slags ting skal man kigge på detaljer der ikke er nævnt som krav. Her er en kort liste af ting man skal teste (se mere i Patton, 2006).

1. Test at alle krav er opfyldt, herunder krav til svartider, brugervenlighed, mv.
2. For hvert skærmbillede, test at hver "knap" gør det rigtige i forskellige tilfælde. Test også at usædvanlige værdier og ikke-tilladte værdier i hvert input-felt håndteres korrekt.
3. Test at usædvanlige hændelser i omgivelserne håndteres korrekt, fx tab af datakommunikation og sammenbrud af eksterne systemer.
4. Kontroller at hver forgrening i programmet er blevet udført.

I mellemstore systemer er der behov for tusinder af test-cases for at dække det hele, og testen kan tage uger. Hvis systemet er et standard-ramme system, findes store dele af det allerede. Det er normalt unødvendigt at udføre en detaljeret test af disse dele (dvs. punkterne 2,3 og 4 ovenfor).

Som regel udfører man testen i flere faser:

Installationstest: Leveringen af systemet starter ofte med installation af det nye hardware, software, etc. Formålet med installationstesten er at sikre at komponenterne arbejder sammen og har grundlæggende funktionalitet til de følgende test.

Systemtest: Formålet med systemtesten er at teste at kravene er opfyldt, skærmbillederne fungerer korrekt, osv. svarende til punkterne 1-4 ovenfor. Man bruger specielt data og database-indhold for at komme igennem alle de specielle tilfælde.

Anvendelsestest: Formålet med anvendelsestesten er at sikre at systemet virker tilfredsstillende i rigtige driftsomgivelser med rigtigt data og rigtige brugere.

Accepttest: En accepttest består af en systemtest plus en anvendelsestest. De to tests kan udføres på forskellige tidspunkter eller sammen.

Pilot test: Ved en pilot test udfører man rigtigt arbejde, men med få brugere. Observer hvor meget støtte der er brug for og skalér op til det fuld drift. Man kan bedre overkomme at hjælpe nogle få brugere og så forbedre processen for de mange.

Driftsprøve: Formålet med driftsprøven er at teste de krav der først kan verificeres efter en periode med daglig drift. Det kan være svartider under den faktiske belastning, tilgængelighed (antal driftsstop), brugernes arbejdshastighed, leverandørens hotline-kvalitet, mv.

6. Vejledning til skabelonens dele

Vejledningen i resten af hæftet kommenterer kravskabelonen afsnit for afsnit. De grå tekster er dele af skabelonen. Side 15 er fx skabelonens forside. Bemærk at kapitelnumrene A, B, C … i vejledningen stemmer med kapitelnumrene A, B, C … i skabelonen.

Man kan gratis downloade og bruge skabelonen til sine krav når blot man klart angiver kilden og copyrighten, fx som i fodteksten på skabelonens forside.

Skabelonen nummererer kapitlerne med A, B, C i stedet for 1, 2, 3. Det er for at undgå konflikt med kontraktens bilagsnumre, som normalt er 1, 2, 3. Bilag 2 kan fx være kravspecifikationen med kapitlerne A, B, C …

Lad være med at ændre skabelonens kapiteloverskrifter. Mange mennesker er blevet vant til SL-07 strukturen og husker fx at kapitel C handler om task og kapitel H om sikkerhed.

Skabelonen starter med en indledningside som skal fjernes i dit eget dokument. Den næste side er forsiden til den færdige kravspecifikation (vist på side 15). Den angiver navnet på det system der skal leveres. Det er hensigtsmæssigt også at have et kort navn for systemet, idet der flere steder i kravene skal refereres til systemet ved navn, fx for at skelne det fra andre systemer som det skal samarbejde med.

Der angives også kundens navn, leverandørens navn og en kort beskrivelse af hvad leverancen omfatter. Det hjælper læseren til straks at forstå om der også er tale om hardware, drift, mv. Hvis kravspecifikationen er et bilag til en kontrakt, vil systemets navn, kundens navn, etc. stå på kontrakten og er unødvendige på kravene.

Nogle dele er **blå**. De skal erstattes med noget andet i den færdige specifikation - eller slettes. Dele med gul baggrund er advarsler eller alternativer. Fjern dem. Andre dele kan ofte genbruges. Røde dele er leverandørens tilbud.

Forsidens toptekst viser hvornår dokumentet sidst er ændret og af hvem. Det er dokumentfelter som MS Word selv opdaterer når dokumentet printes eller gemmes. Topteksten viser også versionsnummeret. Tilpas den til jeres egen standard.

Siden efter forsiden er dokumentets versionshistorie. Den viser hvad der blev ændret hvornår. Tilpas den til jeres eget projekt.

Kapitel A er projektets baggrund og en vejledning til leverandøren om kravenes form og hvordan han skal svare. Kapitel B forklarer projektets forretningsmæssige formål, hvad der skal bevises tidligt i projektet og hvordan kunden udpeger vinderen.

Kapitel C til J specificerer hvad leverandøren skal levere ved overtagelsen, dvs. når accepttesten er godkendt. Kapitel K specificerer hvad kunden skal levere. Kapitel L specificerer leverandørens forpligtelser efter overtagelsen.

Kapitel K og L er ofte separate bilag til kontrakten i stedet for kapitler i kravspecifikationen. Det er ikke afgørende når bare det står et eller andet sted.

Version 5.5

16-11-2017, 13:38
Sidst rettet af: slauesen

Kravspecifikation til
Elektronisk Patientjournal System
(i det følgende kaldet EPJ-systemet)

Kunde
Midtland Hospital

Leverandør
…

Leverancen omfatter
Levering, drift og vedligehold af EPJ-system

Indhold

A. Baggrund og vejledning til leverandøren 4
A1. Baggrund og vision 4
A2. Vejledning til leverandøren 5
A3. Overordnet løsning og alternativer 6
B. Overordnede behov 7
B1. Flow 7
B2. Forretningsmæssige mål 8
B3. Tidligt bevis for gennemførlighed (proof of concept) 9
B4. Minimumskrav 10
B5. Tildelingskriterie: Nettogevinst over 5 år 11
B6. Tildelingskriterie: Vægtede point pr. DKK 12
C. Task systemet skal støtte 13
Arbejdsområde 1: Patientadministration 13
C1. Indskriv patient inden ankomst 13
C2. Indskriv akut 13
Arbejdsområde 2: Patientbehandling 14
C10. Udfør klinisk session 14
C11. Ordiner medicin til patient 15
C20. Udfør klinisk session mobilt 15
D. Data systemet skal opbevare 16
D0. Fælles felter 17
D1. Diagnose 17
D2. Diagnosetype 18
D3. Ydelse 18
E. Andre funktionelle krav 21
E1. System-genererede hændelser 21
E2. Udskrifter og rapporter 21
E3. Forretningsregler og komplekse beregninger 22
E4. Udbygning af systemet 23
F. Integration med eksterne systemer 24
F0. Fælles integrationskrav 25
F1. SKS 26
F2. LabSys 27
F10. Integration med nye systemer 28
G. Teknisk it-arkitektur 29
G1. Eksisterende hardware og software Alternativ 1: Brug hvad vi har 29
G2. Nyt hardware og software Alternativ 2: Leverandøren foreslår 29
G3. Leverandøren har driftsansvar Alternativ 3: Leverandørens problem 29
H. Sikkerhed 30
H1. Log-in og adgangsret for brugere 30
H2. Sikkerhedsadministration 31
H3. Sikring mod tab af data 32
H4. Sikring mod utilsigtet brugeradfærd 32
H5. Fortrolighedskrav 32
H6. Sikring mod trusler 33
I. Brugervenlighed og design 34
I1. Indlæring og effektivitet i daglig brug 34
I2. Tilgængelighed og Look-and-Feel 35
J. Andre krav og leverancer 36
J1. Andre standarder der skal følges 36
J2. Uddannelse 36
J3. Dokumentation 37
J4. Datakonvertering 37
J5. Installation 37
J6. Test af systemet 38
J7. Udfasning 38
K. Kundens leverancer 39
L. Drift, support og vedligehold 40
L1. Svartider 40
L2. Tilgængelighed (driftseffektivitet) 41
L3. Datalagring 41
L4. Support 42
L5. Vedligehold 43

A. Baggrund og vejledning til leverandøren

A1. Baggrund og vision

Dette afsnit giver læseren et hurtigt overblik over systemet og dets formål. Forklar de vigtigste forretningsmæssige mål (hvorfor kunden vil bruge penge på systemet), men gå ikke i detaljer (målene uddybes i afsnit B2). Forklar kort kundens nuværende situation og hans visioner om fremtiden.

Leverandørerne vil gerne have nogle tal om kunden, så han har en forestilling om hvor "stort" projektet er. Hvor mange brugere, hvor meget data, etc. Skriv nogle få nøgletal.

Kontekstdiagrammer for den nuværende og fremtidige situation er en god illustration. Pilene viser datas vandring. I forbløffende mange kravspecifikationer er det uklart for såvel kunde som leverandør hvor meget der skal leveres og hvem der skal integrere med andre systemer. Marker det system der skal leveres som én kasse med dobbelt væg. Vis de integrationer leverandøren skal udføre som dobbeltlinje pile.

I eksemplet skal leverandøren levere et EPJ-system inklusiv et medicinsystem. Det er vist ved at kassen med dobbelt væg (leverancen) indeholder et medicinsystem. Det kan være at leverandørens eget EPJ-system allerede indeholder et medicinsystem eller at han vælger et (måske kundens nuværende) og integrerer med det.

Han skal desuden integrere med de eksisterende SKS tabeller og LabSys. Diagrammet viser at han ikke skal integrere med nye eksterne systemer. Som beskrevet i afsnit F10 skal tredjepart kunne udføre disse integrationer.

Ofte ser man at kunden skriver en lang redegørelse for sin it-strategi, den historiske udvikling, osv. Det kan være udmærket som orientering for leverandøren hvis det kan holdes på et par sider og er relevant for det han skal levere. Men ofte afspejler redegørelsen mere kundens interne overvejelser og politiske udsagn, som ikke er relevante for leverandøren.

Der kan være behov for at kunden - eller hans konsulent - forklarer de interne overvejelser grundigt, fx de møder man har holdt, de valg man har truffet, og kilden til kravene, men gør det i et andet dokument; ikke i kravspecifikationen.

Pas også på at der ikke kommer til at stå egentlige krav i afsnittet om *baggrund og vision*. Krav skal stå i tabeller som forklaret i næste afsnit.

A. Baggrund og vejledning til leverandøren

A1. Baggrund og vision

Kunden har i øjeblikket flere EPJ-systemer af ældre dato, som han ønsker at erstatte med ét for at opnå:

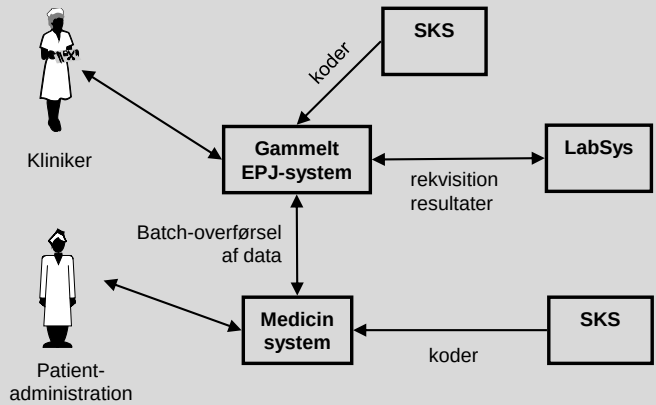
1. Mere effektiv støtte til det kliniske arbejde
2. Bedre mulighed for integration med fremtidige systemer
3. Lavere driftsomkostninger

Midtland Hospital har ca. 5.000 ansatte, hvoraf ca. 800 er læger. Hospitalet har ca. 50.000 indlæggelser om året og ca. 250.000 ambulante behandlinger.

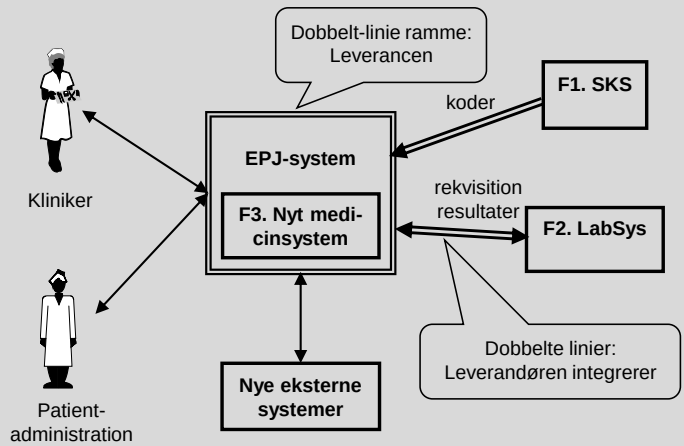
Kunden forventer at leverandøren allerede har et standardsystem der kan opfylde mange af kravene. Kunden er til gengæld villig til i rimelig udstrækning at ændre sine arbejdsgange så de passer til standardsystemet, så længe de overordnede mål nås (se afsnit B2).

Den nuværende og fremtidige situation er anskueliggjort med disse kontekstdiagrammer. Leverandørens ansvar er vist: Kassen med dobbelt-linje ramme er det system der skal leveres. Dobbelt-linje pile er de integrationer der skal leveres. I dag er der fx dårlig integration mellem EPJ-systemet og medicin-systemet. Kunden ønsker et EPJ-system der inkluderer et medicinsystem. Det kan være kundens nuværende medicinsystem eller et nyt leveret som del af EHR-systemet.

Figur 1: Eksisterende system



Figur 2: Vision om nyt system



A2. Vejledning til leverandøren

Dette afsnit forklarer kort hvordan kravene er opstillet og hvordan leverandørens svar skal struktureres. Specielt forklares hvordan tabellerne bruges, hvad der er krav, hvad der er løsninger og hvad der er forudsætninger leverandøren kan gøre.

Hensigten er at leverandøren ikke behøver anden vejledning end dette afsnit af skabelonen. Fx behøver han ikke læse vejledningshæftet.

Det anbefales at leverandørens svar skal være med **rødt**. Efter flere forsøg med forskellige markeringer, fx kursiv eller en farve for hver forfatter, blev det klart at rødt for tilbuddet var det tydeligste og mest læselige. Det er også langt lettere at læse end traditionelle tabeller med en søjle til hver af parterne eller - endnu værre - et bilag for kravene og et andet bilag for løsningen.

Kolonne 3 (kode) kan bruges til mange formål. Den har fx været brugt til:

1. Kravprioriteter.
2. Leverandørens angivelse af om den tilbudte løsning er en del af standardsystemet, en udbygning, en senere leverance, etc.
3. Kundens point for den tilbudte løsning.
4. Senere i projektet en henvisning til en test case der tester løsningen til dette krav. Så sikrer man at alle krav bliver testet et eller andet sted.

Ret evt. i vejledningen så den afspejler hvad kodekolonnen skal bruges til i dette projekt.

Alternativer, open target og andre kraveksempler

Begreberne alternativer og "open target" illustreres med et krav om system-tilgængelighed. Der kan være behov for flere eksempler på krav for at hjælpe leverandøren med at forstå hvad han forventes at skrive.

Anbefaling: Udbyg vejledningen med et par krav fra jeres eget projekt, og vis hvordan leverandøren i princippet kunne besvare dem.

A3. Overordnet løsning og alternativer

Dette afsnit er beregnet til leverandørens korte beskrivelse af sin løsning. Han kan tilbyde alternative løsninger (se afsnit 4.1), og dette er et godt sted at skrive dem.

A2. Vejledning til leverandøren

Dette afsnit forklarer kravenes form. Alt hvad leverandøren skriver skal være med rødt.

Alle krav står i tabeller:
- Kolonne 1 er kravene (kundens behov - hvad han ønsker systemet skal støtte).
- Kolonne 2 kan indeholde kundens eksempel på en løsning. I leverandørens svar er kolonne 2 en kort beskrivelse af den tilbudte løsning. Den skal være med rød skrift.
- Kolonne 3 (Kode) er til kundens vurdering af den tilbudte løsning, testreferencer, etc.

Kravene er opdelt i kapitler efter deres art, fx kapitel C om arbejdsopgaver systemet skal støtte, kapitel H om sikkerhed. Indenfor hvert kapitel er kravene skrevet i tabeller, fx en tabel med krav der har med en bestemt arbejdsopgave at gøre. En reference til krav 3 i afsnit L2 ser sådan ud: L2-3.

Kundens eksempler på løsning er kun til inspiration. Leverandøren er velkommen til at foreslå helt andre løsninger. Leverandørens foreslåede løsninger bliver til kontraktmæssige krav når begge parter har godkendt dem. Hvis den godkendte løsning alligevel ikke opfylder behovene i kolonne 1 i rimelig grad, har kolonne 1 dog forrang (højere prioritet).

Tekst udenfor tabellerne
Tekst der står uden for tabellerne kan have flere formål:

A. *Forudsætninger* for kravene, fx at arbejdsopgaven skal støttes for den slags brugere, den hyppighed, mv.
B. *Kravnoter* som uddyber kravsøjlen i tabellen. I princippet kunne de være i tabellen, men formatet er ikke egnet. Eksempel: en liste af adgangsrettigheder.
C. *Løsningsnoter* som uddyber løsningssøjlen i tabellen. Eksempel: forskellige måder brugeren kan finde en kode i en tabel.
D. Eksempler og andre oplysninger der hjælper læseren med at forstå kravene.

Alternativer
Det sker ofte at en kunde uafvidende kommer til at stille krav der vil være meget dyre at få opfyldt. I sådanne tilfælde er leverandøren velkommen til at tilbyde alternative løsninger: en dyr der fuldt ud opfylder kundens krav og en billigere der kun delvis opfylder kravene. Kravet i rammen nedenfor er et eksempel.

Når tilbuddet indeholder alternativer på flere områder er det vigtigt at hvert sæt alternativer kan vurderes separat af kunden.

Open target
Der er i kapitel L mange eksempler på krav udtrykt ved "open target". Kunden beder fx om en høj systemtilgængelighed, men er usikker på hvad det koster. Han angiver derfor hvad han forventer og overlader det til leverandøren at angive hvad han vil foreslå. I tilbuddet bliver det krav L2-2:

Krav:	~~Eksempel på løsning~~ Tilbudt løsning:	Kode:
2. I tidsrummet 8:00 til 17:00 på hverdage skal systemet have høj tilgængelighed.	I disse tidsrum er tilgængeligheden mindst ___ %. (Kunden forventer 99.5% eller bedre). Alternativ A: 99,0%, ca. 3 mio./år, se bilag . . . Alternativ B: 99,8%, ca. 15 mio./år, se bilag . . .	

Bemærk at kunden har skrevet "*99.5% eller bedre"*. Det betyder at leverandøren får ekstra point for alternativ B. Havde kunden udeladt "*eller bedre"*, ville alternativ B ikke give flere point end de 99.5%, kun ekstra udgifter.

Om formatet
Kravspecifikationen er skrevet i MS-Word. Dokumentet bruger *Heading 1*, *Heading 2* og undertiden *Heading 3*, samt *Heading no number*. De vi[illegible] tvungent sideskift for nogle overskrifter for a[illegible]
Afsnit → S[illegible]
Tabellerne har rammer på ½ punkt. Cellern[illegible] formatet *Column1*. Det har hængende indry[illegible] med Ctrl+Tab, idet Tab alene rykker markø[illegible]

A3. Overordnet løsning og alternativer

Leverandøren tilbyder …

Leverandøren tilbyder følgende alternativer:
Alt A.

B. Overordnede behov

Dette kapitel indeholder ikke egentlige krav, men giver sammenhængen mellem kravene, kundens forretningsmæssige mål og anskaffelsesprocessen.

B1. Flow

Det man kan observere brugerne gøre, er ofte små stumper arbejde (task), der er del af et større flow som giver de egentlige resultater kunden er interesseret i.

EPJ-eksemplet har kun ét flow: Patientens behandling fra indskrivning til helbredelse. Undervejs undersøger man patienten, stiller diagnoser (hvad patienten fejler), planlægger og udfører behandlingen, kontrollerer resultatet og udskriver patienten. Hele forløbet kan tage dage eller måneder.

Et flow kaldes også et forløb, en proces, en forretningsproces, en livscyklus eller et højniveau task eller højniveau use case. Et flow kan gå fra aktør til aktør, endda fra organisation til organisation. I EPJ-eksemplet er det dog kun hospitalet vi ser på.

I sundhedssektoren er der andre flows end patientbehandling, fx livscyklus af en behandlingsmetode fra den i sin tid blev anbefalet af en kommission til den mange år senere bliver nedlagt af en anden kommission. Det er ikke meningen at EPJ-systemet skal støtte dette flow, men det kan godt være det skal levere data til det.

Flow som tabel: I afsnit B1 beskriver man de flows systemet skal støtte. Vi anbefaler at de beskrives som tekst i en tabel, men det kunne også være grafisk. I eksemplet består et behandlings-flow af 12 trin, men det er ikke sikkert de alle bliver udført for en given patient. Nogle af dem kan gentages flere gange, fx kontrolbesøg.

I tabellens kolonne 2 skriver man hvilke task og subtask der udfører dette trin. Fx er der to forskellige task der kan udføre trin 1: Indskrivning inden patienten ankommer (C1, typisk efter henvisning fra egen læge) og akut indskrivning (C2, fx ved en færdselsulykke). Til gengæld håndteres trin 3 til 4 og 6 til 9 af den samme task beskrivelse, den kliniske session (C10).

Der er altså en mange-til-mange sammenhæng mellem flowets trin og de fysiske, observerbare task. Det er ikke et hierarki.

Når man beskriver et flow, opdager man ofte nye behov for it-støtte. I dette tilfælde opdager vi behovet for at aftale kontrolbesøg (trin 8) og samspil med hjemmeplejen (trin 10). Det er vist som røde spørgsmålstegn i tabellen.

Flow som graf: En meget udbredt grafisk notation er BPMN (Business Process Modeling Notation), der viser hvert trin i processen som en kasse og forbinder kasserne med pile der viser hvad der kommer efter hvad, og med romber ("diamanter") for at vise valg der skal træffes om rækkefølgen. Det kan give et fint overblik - med mindre man går for meget i detaljer og prøver at specificere hvad der skal ske når noget går galt.

I praksis ser man at der bruges meget tid på flowdiagrammer og de fylder mange sider. Det er ofte svært at se om flowet handler om de logiske trin eller de fysiske task, og endnu vanskeligere at stemme dem af mod hinanden, som vi har gjort i kolonne 2 i tabellen.

B. Overordnede behov

Dette kapitel forklarer hvordan kravene tænkes at tilgodese kundens forretningsmæssige mål, hvordan risikable krav ønskes afklaret tidligt og hvordan tilbud sammenlignes. Kapitlet indeholder ikke egentlige krav.

B1. Flow

Systemet skal kun støtte én slags flow: Behandling af en patient. Nedenstående er det generelle, logiske flow for en behandling. Mange af trinene kan udelades (fx trin 2 og 8) eller kan gentages flere gange under behandlingen (fx trin 3 til 9).

Det logiske flow udføres i fysiske arbejdsopgaver (task), hvor en medarbejder en kort periode arbejder med patienten uden væsentlige afbrydelser. Kolonne 2 viser de relaterede task og subtask for hvert trin. Detaljerne fremgår af kapitel C.

Trin i patientbehandling	Task og subtask
1. Indskriv patienten enten via egen læge, ved henvendelse fra patienten selv eller akut (fx trafikuheld med bevidstløs patient).	C1, C2
2. Indkald patienten og aftal tid.	C1-4
3. Patienten ankommer til afdelingen. Undersøg hvad patienten fejler, herunder foretag diverse test på stedet eller via laboratorier. Stil diagnoser.	C10-1, 2, 3 C12
4. Planlæg behandling, herunder ordinér medicin, book tider, bestil implantater, etc.	C10-6, C11, C13
5. Overfør evt. patienten til en anden afdeling, fx hvis der er flere diagnoser.	C3
6. Udfør behandling.	C10-3, C14
7. Vurder resultatet. Udfør evt. yderligere test og udfør supplerende behandling.	C10
8. Aftal kontrolbesøg.	C10-6 ?
9. Patienten ankommer til kontrolbesøg. Udfør diverse test og evt. supplerende behandling. Aftal evt. næste kontrolbesøg og evt. behandling.	C10
10. Aftal evt. hjemmepleje.	?
11. Udskriv patienten og orienter relevante parter, fx egen læge eller sociale myndigheder. Patienten kan også være død.	C6, C7
12. Afregn tilskud eller egenbetaling.	C8

I det generelle flow har vi ikke anført at der kan være tidsovervågning af de forskellige trin. Det beskrives i task og subtask.

Flow-beskrivelsen er ikke krav, men et krydstjek mellem det logiske flow og task. I dette tilfælde har det afsløret nogle mangler i task, noteret som spørgsmålstegn ovenfor.

B2. Forretningsmæssige mål

Dette afsnit af skabelonen indeholder systemets forretningsmæssige mål, opstillet i en tabel så man kan se hvordan de tænkes nået. Kolonne 1 indeholder målene; kolonne 2 visionen - løsningen i store træk; kolonne 3 de krav der skal muliggøre visionen. Det præciseres at der ikke er tale om krav til leverandøren, men en hjælp til at vurdere hvad der er vigtigt for kunden. Kolonne 4 giver kunden mulighed for at angive en tidsfrist for målet. Hvis der er en frist, er den for kundens og leverandørens fælles anstrengelser. Leverandøren skal tænke på at kunden skal bruge tid til den organisatoriske implementering.

De forretningsmæssige mål tjener flere formål:

a. De fortæller leverandøren hvad kunden vil opnå.
b. De er væsentlige kriterier for kundens valg af løsning.
c. De hjælper kunden med at kontrollere at han har de afgørende krav med.

I eksemplet er mål 1 (effektiv støtte til alle task) et meget bredt mål der afhænger af mange krav. Kunden kan fravælge løsninger der giver dårlig støtte til et eller flere task. Fx kan lægen have behov for en god oversigt over patientens situation for at kunne træffe den rigtige beslutning. Et system med et meget dårligt oversigtsbillede skal derfor kunne fravælges selvom oversigtsbilledet blot er én blandt 1000 detaljer i systemet. Afsnit B4 forklarer hvordan dette kan blive en del af tildelingskriterierne.

I eksemplet havde kunden tidligt formuleret mål 3, *løbende forbedring af arbejdsgangene*, men havde ikke gjort sig klart at det krævede at man let kunne definere nye standardplaner og nye skærmbilleder. Først da man skulle udfylde tabellen blev dette behov klart og kravene i E4 blev defineret.

Hold antallet af mål på et rimeligt niveau. Bliver der mere end 10 mål bør man kontrollere at det ikke bare er krav man skriver. Fx ser man tit "målsætninger" af denne art: *Der skal let kunne udskrives forbrugsopgørelser.* Selvom det var vigtigt for en af interessenterne, er det et simpelt systemkrav, ikke et forretningsmæssigt mål. Et forretningsmæssigt mål handler om hele organisationens resultater, ikke blot noget som it-systemet gør.

Hvis man ikke kan skrive noget fornuftigt i kolonne 2, kan det være et tegn på at målet ikke er et rigtigt forretningsmæssigt mål, men et krav. Hvis målet fx er: *Der skal let kunne udskrives forbrugsopgørelser,* vil det være svært at skrive en overordnet løsning. Hvis man *skal* have et mål som ikke er et rigtigt forretningsmæssigt mål, så lad kolonne 2 være tom.

Måle forretningsmålene: Et godt forretningsmæssigt mål kan måles og sammenlignes med den eksisterende situation. Mål 2 er klart af denne art. Mål 1 kan måles på en subjektiv skala, fx 1 til 5, men det er svært at se den forretningsmæssige værdi. Det kunne også måles som antallet af task udført pro persona pr. dag, eller som den tid der bruges ved computeren pr. patient. Det er hårde data der hænger tæt sammen med en forretningsmæssig værdi. Mål 4 kan måles som driftsudgifterne før og efter indførelsen af det nye system.

Det er vigtigt at have tal for disse mål for at kunne vælge det økonomisk mest fordelagtige tilbud, som beskrevet afsnit B5.

B2. Forretningsmæssige mål

Kundens formål med at anskaffe systemet er at opfylde en række forretningsmæssige mål. Kunden forventer at systemet bidrager til målene som anført nedenfor. Leverandøren kan sjældent opfylde målene alene, idet kundens medvirken også er afgørende. Målene er altså *ikke krav* til leverandøren, selvom de står i en tabel for overskuelighedens skyld.

Alle mål er vigtige og jo før de kan nås, desto bedre. For nogle mål er det kritisk at de nås på et bestemt tidspunkt, fx af forretningsmæssige eller lovgivningsmæssige grunde. Sådanne tidsfrister er anført i tabellen.

Formål med det nye EPJ-system	**Løsningsvision**	**Relaterede krav**	**Evt. tidsfrist**
1. Effektiv støtte til alle task.	Alt nødvendigt data er til rådighed ved arbejdet uden at skifte mellem flere systemer. Alle parter kan se journalen.	Støtte til alle task i kapitel C. Systemintegration, specielt F2. Svartider i afsnit L1.	
2. Reducer fejlmedicineringer fra 10% til 2%.	Undgå manuelle mellemled - indtast ordinationen straks. Systemet kontrollerer for rimelighed, interaktion af præparater, etc.	Støtte til task C10 (klinisk session), især problem 2p (vurder patientens tilstand) og 6q (fejl når data skifter hånd). Støtte til task C11 (ordination), stort set alle subtask.	
3. Løbende forbedring af arbejdsgange.	Let at opbygge og ændre standardplaner for behandling. Let at integrere systemet med nye laboratoriesystemer, etc.	Krav i afsnit E4 og F10 (udbygning af systemet og integration med nye systemer).	
4. Lavere drifts-omkostninger.	Anskaffe et nyt, formentlig billigere, system.	Alle kravene og tildelingskriterierne i afsnit B5.	
5. Overholde de nye EU-regler om ...	…	…	1-1- 2019

Selvom målene kan måles, er det ikke sikkert at kunden vil afsløre de forventede resultater. De kan fortælle leverandøren hvor meget kunden er villig til at betale. Afsnit B6 viser et eksempel på hvordan man kan undgå det.

B3. Tidligt bevis for gennemførlighed (proof of concept)

Dette afsnit beskriver de dele af projektet som anses for mest risikable, fordi de ikke kan udbedres sent i projektet. For at reducere risikoen, skal leverandøren tidligt sandsynliggøre at han kan levere hvad der kræves.

De fleste funktionelle krav har lav risiko. Det er fx let at tilføje nogle felter eller tabeller til databasen, eller nogle simple skærmbilleder til brugergrænsefladen. De fleste høj-risiko områder handler om kvalitetskrav for, som amerikanerne siger, *quality is not an add-on feature*.

Afsnittet nævner at kontrakten giver begge parter ret til at opsige kontrakten hvis de tidlige beviser ikke er tilfredsstillende. Sørg for at det er tilfældet (se også 3.2).

Punkt B3-1 til B3-5 præciserer hvad der skal testes tidligt. Kolonne 2 giver et eksempel på hvordan det kan testes. Leverandøren kan ændre det til sit eget forslag om test. Desuden angiver han hvornår beviset kan foreligge. (Undertiden kan en leverandør have et bevis allerede inden han underskriver kontrakten).

Disse tests kan være dyre, og derfor er det ikke rimeligt at leverandøren skal have gjort det inden han giver tilbud. Han kan inkludere omkostningen i sit tilbud og vil således få betaling når han leverer hvad han lovede.

B4. Minimumskrav og tildelingskriterier

Hvordan vælger vi det bedste tilbud? Tildelingskriterierne er som regel en del af udbudsmaterialet, men ikke af kravene. SL-07 skabelonen har dem med for at vise hvordan tildelingskriterierne kan vælges så de passer til de forretningsmæssige mål og kravene. Flyt dem til andre dele af udbudsmaterialet efter behov.

I en udbudsforretning skal kunden vælge leverandør ud fra nogle kriterier der er objektive og kendt af leverandøren. Som regel er der kun disse overordnede kriterier:

1. Den forretningsmæssige værdi af løsningen.
2. Kundens risiko.
3. Leveringstiden.
4. Kundens samlede omkostninger.

Hvordan kan disse kriterier kombineres? En pragmatisk metode der egner sig til et lille antal tilbud er at se på dem ud fra den konkrete situation, kassere de tilbud der er klart dårligere end de andre, og finde de argumenter der kan udpege vinderen blandt resten.

Desværre er det ikke tilladt efter EU-reglerne, fordi det giver kunden mulighed for at manipulere med argumenterne, så yndlingsleverandøren kan fremstå som vinderen. EU-reglerne siger at man skal vælge to sæt kriterier og fortælle leverandørerne om dem på forhånd:

A. Minimumskrav. Tilbud der ikke overholder alle disse krav skal afvises.
B. Tildelingskriterier. Hvert kriterium har en foruddefineret vægt. Hvert tilbud får point for hvert tildelingskriterium. Tilbuddet med den største vægtede sum af point, får kontrakten.

Dette er svært i praksis og kan tvinge kunden til at vælge et dårligt tilbud.

B3. Tidligt bevis for gennemførlighed (proof of concept)

Nogle krav er meget risikable og leverandøren kan være ude af stand til at levere hvad han lovede i sit tilbud. Hvis det opdages sent i projektet, kan kunden hæve kontrakten, men det er en katastrofe for begge parter. Som regel ender det med at kunden vælger at leve med det uegnede system, evt. med afslag i prisen. For at reducere risikoen, kræver kunden et tidligt bevis for at de risikable krav er gennemførlige (proof of concept).

Ifølge kontrakten kan begge parter opsige aftalen hvis de tidlige beviser ikke er tilfredsstillende.

Følgende krav betragtes som de mest risikable. Væsentlige mangler her kan næppe udbedres sent i projektet. Leverandøren skal i sit svar angive hvordan han vil udføre disse tidlige beviser, og hvornår det kan ske. Tidspunktet angives som antal arbejdsdage efter kontraktunderskrift. Kunden forventer 40 arbejdsdage eller mindre.

Områder hvor tidligt bevis kræves:	Eksempel på bevis:	Kode:
1. Effektiv støtte til klinisk session (task C10).	En prototype af de nødvendige skærmbilleder (gerne på papir) vurderes af ekspertbrugere. Kan ske inden __ arbejdsdage. (Se også område 5 nedenfor).	N/A
2. Brugervenlighed (alle krav i afsnit I1).	En prototype (gerne på papir) usability-testes med typiske brugere. Kan ske inden __ arbejdsdage.	N/A
3. Svartider med det planlagte antal brugere (alle krav i L1).	Et testsystem sættes op med simulation af det forventede antal brugere. Svartiderne måles. Kan ske inden __ arbejdsdage.	N/A
4. Mulighed for tredjeparts udbygning (E4 og F10).	Dokumentation af dele af systemet og dele af de tekniske grænseflader vurderes af et uafhængigt software-hus med henblik på egnethed til udbygningsformålet. Kan ske inden __ arbejdsdage.	N/A
5. Integration med andre systemer.	En forsøgsopstilling hvor dataudvekslingen demonstreres. Kan ske inden __ arbejdsdage.	N/A

Problem A: Der kan være over 1000 krav. Hvilke af dem er minimumskrav (ufravigelige)? Hvis de alle er ufravigelige, kan man blive nødt til at afvise gode tilbud der ikke opfylder et tilfældigt, mindre vigtigt krav. Hvis kun nogle er ufravigelige, kan man blive tvunget til at tage et der slet ikke opfylder resten.

SL-07 løser problemet ved at definere minimum for **kravområder** i stedet for enkeltkrav (se eksemplet nedenfor). Der kan være omkring 40 kravområder og det giver mening at tjekke at hvert område støttes tilstrækkeligt godt. Et område kan støttes tilstrækkeligt godt selvom enkelte krav på området ikke støttes. Typisk vil nogle leverandører fejle for visse krav på området, andre for andre krav. De kan stadig komme i betragtning når blot de støtter området som helhed godt nok.

Problem B: Hvordan definerer man point og vægte i praksis? Nogle bruger vægte der tilsammen giver 100%, men hvordan kan de begrundes? Som regel har vægtene ingen sammenhæng med den forretningsmæssige værdi, så kunden kan tvinges til at vælge et tilbud der ikke har den højeste værdi.

SL-07 viser to løsninger: En hvor tildelingskriteriet er den forretningsmæssige nettoværdi (B5), og en hvor det er flest point pr. krone (B6).

Minimumskrav og minimumspoint

I udbudsmaterialet har kunden delt kravene op i områder og angivet minimum-point for hvert område. På den måde kan man dække alle krav uden at de hver for sig skal være minimumskrav. Der er ca. 20 kravområder i EPJ eksemplet.

Skabelonen bruger disse point: -2 (ikke støttet eller meget ubekvemt), -1 (ube-kvemt), 0 (som i dag eller lige godt nok), 1 (effektivt), 2 (meget effektivt).

Her er begrundelsen for minimumspoint i EPJ-eksemplet:

Område		Minimum point
C1-C4	Indskriv patient (ét område). Der er egentlig ikke behov for støtte. Kunden kan bare beholde sit eksisterende indskrivningssystem.	-2
C10	Udfør klinisk session. For at undgå at skulle vælge en leverandør der scorer højt alle andre steder, men dårligt for kliniske sessioner, kræver vi at kliniske sessioner støttes mindst så godt som med det nuværende system.	0
C11-C…	Medicinering (ét område). Skal også være mindst så godt som i dag.	0
…	…	…
D	Data. Vurderes ikke separat. Det sker indirekte gennem støtten til task.	N/A
…	…	…
F10	Integration med nye systemer. Det skal være bedre end i dag - et af de forretningsmæssige mål..	1
H1	Login og adgangsret for brugere. Det skal være mindst så godt som i dag.	0
H2-5	Anden sikkerhed (ét område). Vi kan acceptere at det er lidt værre end i dag.	-1
I	Brugervenlighed. Det skal være mindst så godt som i dag.	0
J2	Uddannelse. Skal være mindst så godt som i dag. Det får også en indirekte score fordi det indgår i de samlede omkostninger.	0
J4	Datakonvertering. Det skal bare være godt nok. Det er en engangs-affære.	0
L1.	Svartider. Det skal være mindst så godt som i dag.	0
…	…	…

Når leverandørerne har afgivet deres tilbud, skal kunden vurdere hvert tilbud. Kapitel 4 forklarer hvordan man sammen med leverandøren vurderer hvert task, hver integration, etc. og skriver noter om det. Giv et samlet point for hvert **krav-område** baseret på vurderingerne og noterne. Et enkelt task eller krav der støttes meget dårligt, kan give et lavt point for hele området. Skriv en note om hvorfor området som helhed ikke er støttet godt nok.

Minimumskrav: Alle kravområder skal mindst opnå områdets minimumspoint.

Noterne er egnede til interne diskussioner hos kunden. De er også nyttige hvis en leverandør finder bedømmelsen uretfærdig og går rettens vej. Så kan noterne bevise at kunden faktisk lavede en fair vurdering.

Princippet med at give et samlet point for et kravområde i stedet for point for enkeltkrav, er vigtigt. Det er stort set umuligt at vurdere et krav isoleret. Krav påvirker hinanden og tilsammen giver de en mere eller mindre god støtte til kravområdet. God støtte til krav A plus dårlig støtte til krav B kan give samme forretningsmæssige værdi som dårlig støtte til A plus god støtte til B. Derfor bør de have samme point.

Lad ikke leverandøren vurdere sit eget tilbud. Forbløffende mange kunder har en tabel med alle udbudsmaterialets krav og beder leverandøren udfylde den med graden af kravopfyldelse, fx opfyldt/ikke-opfyldt. *Vi har ikke selv tid til det,* siger kunden, *lad leverandøren gøre det.* Hvad gør leverandøren? Han lader sin salgsafdeling udfylde tabellen. Ikke overraskende får alle tilbud top point for alting. Resultatet er at kunden kun kan sammenligne tilbuddene på deres pris og derfor let kommer til vælge et dårligt system.

Kunden har nok sparet nogle arbejdstimer nu, men til gengæld spilder han tusinder af timer senere fordi medarbejderne skal bruge et dårligt system.

I EPJ-eksemplet bliver nogle kravområder vurderet to gange: Når vinderen skal vælges og under det tidlige bevis. Det er OK. På den måde kan kunden gradvis reducere risikoen for at leverandøren ikke kan leve op til sine løfter. Hvis løsningen ikke opfylder det tidlige bevis, er det en god grund til at opsige kontrakten (se afsnit 3.2).

B4. Minimumskrav

Afsnit B4 til B6 er vigtige i offentlige udbud efter EU-reglerne. Leverandørerne skal kende tildelingskriterierne og deres vægte inden de afgiver tilbud. Ved andre anskaffelser behøver kunden ikke at angive nogen kriterier.

Point: Kunden giver hvert tilbud point for de kravområder der er vist i tabellen nedenfor. For at give bedre oversigt, har tabellerne plads til flere tilbud (kolonne A, B og C). Pointene gives efter denne skala: -2 (ikke støttet eller meget ubekvemt), -1 (ubekvemt), 0 (som i dag eller lige tilstrækkeligt), 1 (effektivt), 2 (meget effektivt).

Minimumspoint: For hvert kravområde har kunden fastsat nedenstående minimumspoint. Et system der ikke opnår minimumspoint, vil være uanvendeligt i praksis.

Minimumskrav: Systemet skal på alle kravområder opnå nedenstående minimumspoint.

Bemærk at minimumspoint kan være -2 eller -1. Det betyder at et tilbud kan være acceptabelt selvom det er dårligere end i dag på dette område. Fx får område C1-C7 minimumspoint -2 fordi man kan klare sig med det eksisterende indskrivningssystem. Tabellen viser et fiktivt eksempel hvor tilbud A får -1 på område H2-H5, og det er acceptabelt fordi minimumskravet er -1.

Kravområde		Minimum point	Point		
			A	B	C
C1-C7.	Indskriv og udskriv patient (ét område).	-2	1		
…					
C10.	Udfør klinisk session.	0	1		
C11-C…	Medicinering (ét område).	0	2		
…					
D.	Data. Vurderes gennem støtten til task.	N/A	N/A		
…					
F10.	Integration med nye systemer.	1	1		
H1.	Log-in og adgangsret for brugere.	0	0		
H2-H5.	Anden sikkerhed (ét område).	-1	-1		
I.	Brugervenlighed.	0	1		
J2.	Uddannelse.	0	0		
J4.	Datakonvertering.	0	1		
L1.	Svartid.	0	0		
…					

B5. Nettogevinst over 5 år

Minimumskravene afviste tilbud der ikke var acceptable. Bland de resterende tilbud skal vi vælge vinderen.

Metode B5 beregner nettogevinsten i kroner for hvert tilbud. Kunden vælger tilbuddet med højest nettogevinst.

Først beregner kunden den *mulige gevinst* for hvert forretningsmæssigt mål. I eksemplet har kunden beregnet den mulige gevinst for en periode på 5 år. Effektiv task-støtte kan fx spare hver medarbejder en time om dagen. Dette skøn er baseret på observationer af hvad klinikerne gør. I dag skal de logge ind på flere systemer for hver patient og skrive noter på papir for at få overblik. De bruger omkring en time på det hver dag. Det kan undgås med mere integrerede systemer og oversigter på skærmen. For 4000 kliniske medarbejdere betyder det en besparelse på 1000 millioner over 5 år.

Opfyldelsesgrad: Et tilbud kan have svagheder der reducerer den faktiske gevinst til en brøkdel af det mulige. Kunden vurderer denne brøkdel for hvert tilbud og hvert forretningsmæssigt mål. Hvis det tilbudte system fx kun kan reducere tiden med 0,5 time om dagen, bliver brøkdelen 0,5. I princippet kan brøken blive større end 1. Det sker hvis tilbuddet overstiger kundens forventninger.

Risiko: Et tilbud kan være risikabelt, fx fordi løsningen ikke er afprøvet andetsteds, eller løsningen er meget skitseagtig, eller leverandøren har brug for meget lang tid til det tidlige bevis. For hvert tilbud og hvert forretningsmæssigt mål vurderer kunden risikoen for ikke at opnå gevinsten.

Ud fra den potentielle gevinst og de tilbudsspecifikke brøkdele og risici, beregner man bruttogevinsten over 5 år.

Samlede omkostninger: De samlede omkostninger i eksemplet består af produktets pris i tilbuddet, udgifter til hardware og andet udstyr som kunden skal købe, udgifter til uddannelse af medarbejderne, og driftsudgifter i 5 år.

Bemærk at alle disse udgifter kan variere fra tilbud til tilbud. Nogle tilbud har brug for mere hardware end andre; nogle har brug for mere uddannelse end andre, osv.

Nettogevinst: Nettogevinsten (bundlinjen) består af bruttogevinsten over 5 år minus de samlede omkostninger over 5 år.

Metode B5 vil nu tvinge kunden til at afvise tilbud der ikke overholder minimumskravene, og blandt resten vælge tilbuddet med størst nettogevinst.

B5. Tildelingskriterie: Nettogevinst over 5 år

Brug enten afsnit B5 eller B6 som tildelingskriterie.

Bruttogevinsten for et tilbud beregnes ud fra den finansielle værdi af de forretningsmæssige mål. Tabellen viser et eksempel med fiktive tal for tilbud A.

Potentiale: Kundens vurdering af den mulige gevinst over en 5-årig periode. Måles i millioner kr.

Opfyldelsesgrad: For hvert tilbud vurderer kunden i hvor høj grad den mulige gevinst kan opnås hvis leverandøren leverer hvad han lover. Det angives som et tal med én decimal, normalt mellem 0,0 og 1,0. Eksempel: Den mulige omkostningsbesparelse ved effektiv støtte af klinikerne er vurderet til en time pr. dag pr. kliniker. Tilbud A ser ud til kun at kunne spare ½ time pr. dag og får derfor opfyldelsesgraden 0,5.

Risiko: For hvert tilbud vurderer kunden risikoen for at opfyldelsesgraden ikke nås. Risikoen vurderes ud fra hvor detaljeret løsningen er, hvor meget af den der eksisterer allerede, om den bliver brugt andetsteds, leverandørens viden om anvendelsesområdet, og hvor lang tid leverandøren skal have til det tidlige bevis. Eksempel: Leverandør A har skitseret en detaljeret løsning, men den findes endnu ikke. Leverandør A har dog godt kendskab til anvendelsesområdet. Risikoen vurderes til 30%.

5-års værdi: Beregnes som potentiale * opfyldelsesgrad * (1-risiko)

Forretningsmæssigt mål	5-års poten-tiale	Opf.grad			Risiko			5-års værdi		
		A	B	C	A	B	C	A	B	C
1. Effektiv støtte af klinikerne	1000	0,5			30%			350		
2. Reducer medicineringsfejl	250	1,0			10%			225		
3. Løbende procesforbedring	250	1,0			40%			150		
4. Mindre driftsomk. (medregnet nedenfor)										
Bruttogevinst for 5 år (millioner DKK)	**1500**							**725**		

Kunden vurderer nettoværdien for hvert tilbud. Bruttogevinsten over 5 år er beregnet ovenfor. Omkostningerne til anskaffelse og drift trækkes fra. Resultatet er nettoværdien over 5 år. Bemærk at alle tallene kan være forskellige fra tilbud til tilbud.

Kunden vælger det tilbud der har den højeste nettoværdi over 5 år.

Nettogevinst over 5 år, millioner DKK	A	B	C
Bruttogevinst over 5 år	**725**		
Produktets pris	100		
Kundens hardware-udgifter	50		
Medarbejderuddannelse	28		
Driftsomkostninger over 5 år	100		
Samlede omkostninger over 5 år	**278**		
Nettogevinst over 5 år	**447**		

B6. Vægtede point pr. DKK

Metode B6 afviser også tilbud der ikke overholder minimumskravene. Men den beregner ikke gevinsten i kroner, men som en sum af vægtede point.

Samlede vægtede point: Vi starter med en kopi af tabellen for minimumskravene og erstatter minimumskravene med en vægt for hvert område. Vi beholder de point vi allerede har givet for hvert tilbud. Vi tilføjer kolonner hvor vi beregner de vægtede point for hvert kravområde og hvert tilbud. De samlede vægtede point for et tilbud afspejler værdien af tilbuddet.

Vægte: Hvordan fastlægger vi vægtene? En mulighed er at give hvert område en prioritet, fx mellem 1 og 5. Nu er prioriteten vægten, men det er svært at forsvare ud fra et forretningsmæssigt synspunkt. Desuden kan det være meget svært at overbevise en interessent om at hans område har prioritet 1 og et andet prioritet 5.

I stedet finder vi vægte der afspejler områdets størrelse, fx antal medarbejdere der er berørt, betydningen for kvaliteten, eller effekten på omkostningerne. I eksemplet startede vi med den mulige 5-års gevinst beregnet som i B5. Vi delte gevinsten ud på kravområderne. Fx kom gevinsten for *løbende procesforbedring* hovedsagelig fra F10, *integration med nye systemer.* Til sidst forklædte vi gevinsterne som vægte ved at dividere dem med en faktor så summen blev 100. Så passer det med traditionen om at bruge vægte der er procent.

Nogle kravområder har ikke direkte noget med de forretningsmæssige mål at gøre. Alligevel har de fået en vægt der afspejler en subjektiv eller politisk værdi. Bemærk at mange områder har vægten nul. Bedre støtte til dem har lille effekt - så længe minimumskravene er opfyldt.

I eksemplet har C10 en meget høj vægt fordi næsten halvdelen af gevinsten høstes her. Det gør resultatet meget følsomt for om en leverandør får 1 eller 2 point her. Derfor har vi givet point med en decimal for C10. Decimalerne kan fx beregnes ud fra point for hvert task eller krav der indgår i området.

Samlede omkostninger: Omkostningerne beregnes præcist som i metode B5.

Bundlinjen: Med B5 trak vi omkostningerne fra gevinsten for at få nettogevinsten. Det kan man ikke gøre for B6. Det giver ingen mening at trække omkostninger i kroner fra gevinst målt i point. Men det giver mening at dividere de to. Det giver os antal point pr. million kroner.

Metode B6 vil afvise tilbud der ikke overholder minimumkravene og blandt resten vælge tilbuddet med flest vægtede point pr. million kroner.

Sammenligning: Den vigtigste fordel ved B6 er at vi ikke afslører den forretningsmæssige værdi overfor leverandørerne eller den offentlige myndighed der finansierer os. B6 giver os desuden mulighed for at give en vægt til kvalitetsaspekter som ikke kan vurderes i kroner. Endelig er processen lidt enklere fordi vi kan genbruge de point vi allerede har givet ved minimumskravene.

Selvom vi ikke tildeler ud fra den forretningsmæssige værdi i kroner, er det stadig nyttigt vurdere den forretningsmæssige værdi og omsætte den til vægte.

B6. Tildelingskriterie: Vægtede point pr. DKK

Med dette alternativ behøver kunden ikke at angive værdien i DKK, og han behøver ikke afsløre overfor leverandøren hvor meget han tjener på projektet. Vi har ikke medtaget risikovurderingen nedenfor, men det kunne gøres.

Point: Pointene nedenfor er dem som kunden allerede har givet ved minimumskriterierne i B4. Da et af områderne har en meget høj vægt, er resultatet meget følsomt for om tilbuddet får 1 eller 2 point her. Af denne grund har vi givet point med en decimal.

Vægt: Hvert kravområde har en vægt som afspejler områdets betydning for det samlede resultat. Fx antal medarbejdere der er berørt af det, effekten på kundens service-kvalitet, eller bidraget til den forretningsmæssige værdi. Vægtene giver 100 tilsammen.

Kravområde		Vægt	Point			Vægtede point		
			A	B	C	A	B	C
C1-C7.	Indskriv patient (ét område).	5	1			5		
…								
C10.	Udfør klinisk session.	50	1,5			75		
C11-C…	Medicinering (ét område).	15	2			30		
…								
D. til task.	Data. Vurderes gennem støtten	N/A	N/A					
…								
F10.	Integration med nye systemer.	15	1			15		
H1. brugere.	Login og adgangsret for	0	0					
H2-H5.	Anden sikkerhed (ét område).	0	-1					
I.	Brugervenlighed.	10	1			10		
J2. omkostningerne	Uddannelse (indgår i nedenfor).	0	0					
J4.	Datakonvertering.	0	1					
L1.	Svartid.	5	0					
…								
Samlet vægt og samlede vægtede point		100				135		

Kunden beregner de samlede vægtede point for hvert tilbud og de samlede omkostninger til anskaffelse og drift over 5 år. Til sidst beregnes de vægtede point pr. million DKK.

Kunden vælger det tilbud der giver flest vægtede point pr. million DKK.

Vægtede point pr. million kr.	Total A	Total B	Total C
Samlede vægtede point	**135**		
Produktets pris	100		
Kundens hardware-udgifter	50		
Medarbejderuddannelse	28		
Driftsomkostninger over 5 år	100		
Samlede omkostninger over 5 år	**278**		
Vægtede point pr. million DKK.	**0,48**		

Varianter

Der er mange varianter af temaerne ovenfor.

For minimumskravene er der mange kravområder hvor kunden kan acceptere en løsning der er værre end i dag. Det ville være dumt at afvise en ellers god løsning hvis den er værre end i dag på et par områder. Men det bør ikke være værre på *for* mange områder. Det kan vi undgå ved et ekstra minimumskrav: Det må ikke være værre end i dag på mere end 3 ud af de 30 områder. (Det skal stadig være mindst så godt som minimumspoint for området).

Vi kan tilføje maximumskrav på omkostningerne, fx *vores budget tillader ikke en investering på mere end 150 millioner kr.* Og et minimumskrav på gevinsten, fx *vi vil ikke investere i noget med mindre vi får 20% i afkast*.

I B5 kunne vi erstatte 5-års perioden med fx 10 år. Det vil gøre tildelingen mindre afhængig af udviklingstiden og de initielle omkostninger.

I B5 kunne vi vælge vinderen ud fra nettogevinsten pr. investeret krone. Det svarer til en ledelsessituation hvor vi har et begrænset beløb at investere og derfor vælger de projekter der giver det største afkast pr. krone.

Vi kan også være mere præcise og beregne den interne rente (IRR) ved at indregne de skiftende gevinster og omkostninger over en årrække.

For B6 kan vi inkludere risikoen for ikke at høste de fulde point og trække "manglende" point fra for den periode hvor systemet udvikles.

Vi kan variere point-skalaen, fx fra (-2 -1, 0, 1, 2) til (1, 2, 3, 4, 5). Det vil gøre B6-metoden mere følsom for omkostningsforskelle og mindre følsom for kvalitetsforskelle, som vist i eksempet nedenfor. B5-metoden har ikke denne svaghed. Generelt er det en god ide at teste vægte og skalaer med tænkte eksempler på tilbud med forskellige point og omkostninger og tjekke at tildelingskriterierne giver mening.

Endelig skal man huske på at der er høj usikkerhed og risici ved store it-projekter. Justering af detaljer i beregningerne har kun ringe betydning i sammenligning med disse risici. Heldigvis er valget af vinderen ofte *robust*: Selv hvis vi varierer vægte og skøn ganske meget, vil der stadig komme den samme vinder ud af det.

Konsekvensen af at skifte skala

Betyder det noget om man bruger den ene eller anden pointskala? Ovenfor brugte vi skala -2 til 2 og A var bedst. Hvad sker der hvis vi bruger skala 1 til 5 i stedet? Man skulle tro at A stadig er vinder, men det er forkert. Det kan blive en anden leverandør.

Tabellen nedenfor viser et eksempel. A har 100 vægtede point når vi bruger en skala fra -2 til 2. B har nul. Men A er også dyrere. Hvis vi ændrer skalaen til 1 til 5, bliver -2 point til 1, og 2 point til 5. Med andre ord: Vi lægger 3 til alle point. Fordi vægtene tilsammen er 100, øges de samlede vægtede point med 300, som vist i tabellen nedenfor.

Omkostningerne for A og B er de samme som før. A fik 0,5 point per million, B fik nul. A var vinderen. Nu får A 2,0 point per million, B får 3,0. B bliver vinderen fordi han er meget billigere og med denne skala får han næsten lige så mange point som A.

Hvordan kan det ske og hvad er rigtigt?

Årsagen er at skala 1 til 5 ikke afspejler den forretningsmæssige værdi. Det ser ud som om B giver værdi. Det gør han ikke. Med skala -2 til 2 kan vi se at B er "som i dag". Hvorfor betale 100 millioner for det?

Konklusionen er at når man bruger point, skal de afspejle den forretnings-mæssige værdi. Positive point skal afspejle værdi, negative tabt værdi.

Hvad gør de fleste kunder mon? De bruger kun positive point!

Konsekvens af skalaen	Skala -2, -1, 0, 1, 2		Skala 1, 2, 3, 4, 5	
	A	B	A	B
Samlede vægtede point	100	0	400	300
Omkostninger, millioner	200	100	200	100
Point per million	0,5	0	2,0	3,0

C. Task systemet skal støtte

Dette kapitel beskriver de task systemet skal støtte. Det engelske begreb *task* svarer nogenlunde til det danske *arbejdsopgave*. Kravet er at alle task skal støttes i en eller anden grad.

Et task er noget en bruger og systemet udfører sammen fra start til slut uden væsentlige afbrydelser. Et godt startpunkt er noget der sker i brugerens verden, fx at en klient henvender sig. Et godt slutpunkt er at man ikke kan gøre mere for klienten lige nu - brugeren fortjener en "kaffepause" (engelsk: *task closure*).

Det første task i skabelonen er C1. Det starter når sekretæren modtager en henvendelse om en patient. Det ender når patienten er blevet indskrevet og har fået et mødetidspunkt - eller er sat på ventelisten - eller når sagen er blevet parkeret fordi der mangler oplysninger.

Tabellen lister de subtask der indgår. Så vidt muligt bestemmer brugeren selv hvilke subtask der skal udføres og i hvilken rækkefølge.

Subtask 1 registrerer patienten. Vi skriver ikke om det er bruger eller computer der gør det. På dette tidspunkt ved vi ikke hvor meget computeren kan gøre. Det afhænger af leverandørens løsning. God støtte er at computeren gør det meste, fx automatisk overfører patientdata når henvendelsen er elektronisk.

Subtask 1a er en variant, dvs. en anden måde at udføre subtask 1 på. Enten udføres 1 eller også 1a.

I et task kan man angive at noget er et **problem** der bør fjernes. Man behøver ikke skrive hvordan. I eksemplet er det et problem at nogle elektroniske meddelelser ikke overholder MedCom formatet.

Strengt taget skal man skelne mellem **task beskrivelse** og **task udførelse.** Lægesekretæren udfører C1 mange gange hver dag. Første gang vedrører det patient A, næste gang patient B som måske mangler oplysninger og derfor parkeres. Sidst på dagen modtager sekretæren endnu en meddelelse om patient B, denne gang med de manglende oplysninger. Nu kan sekretæren gøre noget andet for B end første gang. Hver gang er det en ny task *udførelse* men den følger samme task *beskrivelse*. Programmører ville sige at C1 er en *klasse* og udførelsen af C1 er en *instans*.

Arbejdsområder

For at kunne vurdere hvor godt et task støttes, skal man vide hvad for nogle brugere der er tale om, hvilke omgivelser det foregår i, etc. Vi kan angive det for hvert task, men skal ofte gentage det samme. Derfor er det bekvemt at gruppere task efter brugertype og omgivelser. Sådan en gruppe task kaldes et *arbejdsområde*.

I skabelonen beskriver man arbejdsområdet som indledning til gruppen af task. Man beskriver brugerprofilerne (rollerne) og måske omgivelserne. Brugerprofilen angiver brugernes IT baggrund, anvendelseserfaring, motivation, mv. Nogle brugere kan arbejde i flere arbejdsområder, evt. med forskellig rolle i hvert område.

Brugerprofilerne er en kort udgave af det der er blevet populært som **personas**. Task minder om **use cases** og **user stories**, men er mindre løsningsorienterede. Se mere nedenfor.

C. Task systemet skal støtte

Systemet skal støtte alle task i dette kapitel, herunder alle subtask og varianter, samt reducere problemerne. Kolonne 1 af tabellerne beskriver hvad bruger og system skal udføre tilsammen. Hvem der konkret gør hvad afhænger af den valgte løsning.

Når et task udføres sker det fra start til slut uden væsentlige afbrydelser. Om nødvendigt må sagen parkeres og genoptages senere. Selvom subtask er nummereret, skal de ikke nødvendigvis udføres i den rækkefølge og mange subtask er valgfrie. Brugeren bestemmer hvad der skal gøres og i hvilken rækkefølge. Et subtask kan også udføres flere gange inden for samme task.

Et subtask kan undertiden udføres på flere alternative måder. Det vises med a, b, osv. Bogstaverne p, q, osv. angiver noget der i dag er problematisk med dette subtask.

Arbejdsområde 1: Patientadministration
Området omfatter indkaldelse af patienter, overvågning af ventelister …

Brugerprofil: **Lægesekretærer**. De fleste lægesekretærer er erfarne it- brugere og har et godt kendskab til deres arbejdsområde. De er gode til at kommunikere med klinikere.
Brugerprofil: **Administrative medarbejdere**. …

C1. Indskriv patient inden ankomst

Dette task opretter en indskrivning eller fortsætter en parkeret indskrivning. De fleste indskrivninger kan behandles færdigt i én arbejdsgang. Resten må parkeres, fx fordi der mangler oplysninger. Det er vigtigt at systemet sikrer at de parkerede sager ikke glemmes (se E1-1)

Start: Meddelelse fra egen læge, fra andet sygehus … meddelelse med manglende oplysninger eller en påmindelse om en parkeret indskrivning.
Slut: Når patienten er indkaldt eller på venteliste, eller når sagen må parkeres mens der ventes på manglende oplysninger.
Hyppighed: Totalt: Ca. 600 henvendelser pr. døgn. Pr. bruger: Op til 40 pr. dag.
Vanskeligt: (aldrig)
Brugere: I første omgang lægesekretær, men opgaven kan overdrages til andre.

Subtask og varianter:	Eksempel på løsning:	Kode:
1. Registrér patient. (Se databeskrivelse D5).		
1a. Patient findes i systemet. Opdater data.		
2. Indskriv også en rask ledsager.		
3. Opret indskrivningen inkl. den foreløbige diagnose. (Se databeskrivelse D1 og D6).		
3a. Overfør data elektronisk fra egen læge, etc.	Systemet bruger MedCom-formaterne.	
3p. **Problem**: Nogle elektroniske meddelelser overholder ikke MedCom formatet.	Systemet tillader manuel redigering af den overførte meddelelse.	
3q. **Problem**: Patienten kan være indskrevet på flere hospitaler og afdelinger. Det kan være svært at overskue hvem der har plejeansvar og hvem der stiller seng til rådighed.		
4. Find et mødetidspunkt for patienten og send et indkaldelsesbrev eller en fortrolig e-mail.		
4a. Overfør patienten til venteliste.		
4b. Der mangler oplysninger. Parker sagen med tidsovervågning.		
4c. Overdrag sagen til en anden, evt. med tidsovervågning.		
4d. Afslå evt. henvendelsen.		
5. Rekvirér tolk til indkaldelsestidspunktet.		

C1. Regler for task (Indskriv patient inden ankomst)

Task beskrivelsen består af disse dele:

ID: Task nummereres C1, C2, osv. For at undgå for meget omnummerering under kravarbejdet, grupperer vi task efter arbejdsområde og starter hvert arbejdsområde med et rundt tal. I skabelonen er C10 første task i næste arbejdsområde, *patientbehandling*.

Navn: Et task skal have et navn i bydemåde, fx *indskriv patient.* Navne som *brugeren indskriver patienten* eller *patientindskrivning* er ikke bydemåde (imperativ). Bydemåde skjuler hvem der gør hvad - bruger eller computer. Under kravarbejdet ved vi endnu ikke hvem der gør hvad.

Indledning: Lidt hjælp til læseren om hvad tasket drejer sig om.

Start: Et task skal være noget der udføres af én person fra start (trigger) til slut (kaffepause) uden væsentlige afbrydelser. Bemærk at et task kan starte af mere end en grund og slutte på mere end en måde.

Startsignalet (*trigger*) bør være noget der sker i brugerens verden. I eksemplet er det at sekretæren modtager en meddelelse om en patient. Undgå hovsa-triggere som *brugeren vil registrere en patient*. Det viser at vi ikke har forstået hvornår det her sker, og om systemet kunne støtte det bedre, fx gennem automatisk modtagelse af MedCom beskeder.

Slut: Et task er slut når brugeren har fortjent en "kaffepause", enten fordi brugeren har gjort det nødvendige, eller fordi han ikke kan gøre mere ved sagen lige nu. Task C1 kan parkeres fordi der mangler data (subtask 4b). Selvom tasket ikke er logisk afsluttet endnu, er det fysisk afsluttet i denne omgang. Brugeren giver sig til at lave noget andet. Dette mønster er meget almindeligt og det er vigtigt at systemet støtter det godt, fx med påmindelser om parkerede sager.

Hvorfor definerer vi et task på den måde? Det er fordi det er den observerbare periode hvor systemet skal støtte brugeren - uden væsentlige afbrydelser. Vi skal tjekke at systemet giver effektiv støtte i hele perioden.

Hyppighed: Hvor hyppigt tasket udføres for hele organisationen og for den enkelte bruger. Hyppigheden for hele organisationen hjælper leverandøren med at estimere computer-kapaciteten. Hyppigheden for brugeren siger noget om vigtigheden af en effektiv brugergrænseflade. Tallene står uden for tabellen, så de er ikke krav, men forudsætninger leverandøren kan gøre. De tilsvarende kvalitetskrav står i andre afsnit: svartider (L1) og brugervenlighed (I1).

Vanskeligt: Situationer hvor tasket er særlig vanskelig at udføre, fx fordi det sker under stress eller kræver særlig akkuratesse. Bemærk at C1 ikke har nogle vanskelige situationer, mens C10 har en.

Man kan ikke umiddelbart observere vanskelige situationer, men må spørge brugerne om dem. *Vanskeligt* er uden for tabellen og derfor ikke et krav. Tidligt i kravprocessen kan man skrive *vanskeligt*, men prøv at flytte det til andre afsnit senere. Tit kan man gøre det til et *problem* i et subtask. Så er det let at tjekke om leverandøren har en god løsning. Vi kan også beskrive en vanskelig situation som et separat task. Så kan vi også tjekke leverandørens løsning.

C1. Indskriv patient inden ankomst

Dette task opretter en indskrivning eller fortsætter en parkeret indskrivning. De fleste indskrivninger kan behandles færdigt i én arbejdsgang. Resten må parkeres, fx fordi der mangler oplysninger. Det er vigtigt at systemet sikrer at de parkerede sager ikke glemmes (se E1-1)

Start: Meddelelse fra egen læge, fra andet sygehus … meddelelse med manglende oplysninger eller en påmindelse om en parkeret indskrivning.
Slut: Når patienten er indkaldt eller på venteliste, eller når sagen må parkeres mens der ventes på manglende oplysninger.
Hyppighed: Totalt: Ca. 600 henvendelser pr. døgn. Pr. bruger: Op til 40 pr. dag.
Vanskeligt: (aldrig)
Brugere: I første omgang lægesekretær, men opgaven kan overdrages til andre.

Subtask og varianter:	Eksempel på løsning:	Kode:
1. Registrér patient. (Se databeskrivelse D5).		
1a. Patient findes i systemet. Opdater data.		
2. Indskriv også en rask ledsager.		
3. Opret indskrivningen inkl. den foreløbige diagnose. (Se databeskrivelse D1 og D6).		
3a. Overfør data elektronisk fra egen læge, etc.	Systemet bruger MedCom-formaterne.	
3p. **Problem**: Nogle elektroniske meddelelser overholder ikke MedCom formatet.	Systemet tillader manuel redigering af den overførte meddelelse.	
3q. **Problem**: Patienten kan være indskrevet på flere hospitaler og afdelinger. Det kan være svært at overskue hvem der har plejeansvar og hvem der stiller seng til rådighed.		
4. Find et mødetidspunkt for patienten og send et indkaldelsesbrev eller en fortrolig e-mail.		
4a. Overfør patienten til venteliste.		
4b. Der mangler oplysninger. Parker sagen med tidsovervågning.		
4c. Overdrag sagen til en anden, evt. med tidsovervågning.		
4d. Afslå evt. henvendelsen.		
5. Rekvirér tolk til indkaldelsestidspunktet.		

Brugere: De brugere der udfører dette task. Udelad denne angivelse hvis der kun er én brugerprofil i arbejdsområdet.

Subtask og varianter: Kravene står i tabellen. Kolonne 1 er en liste af subtask, varianter og problemer. Det er dem systemet skal støtte. Subtask er nummererede i en logisk rækkefølge, men det er mest for at vi kan referere til dem. De er valgfri (brugeren behøver ikke udføre dem alle), nogle af dem kan gentages, og de kan udføres i mange rækkefølger. Brugeren bestemmer så vidt muligt hvad der skal ske.

Bemærk at vi også bruger bydemåde for subtask (*registrér patient*). Skriv gerne flere liner for at forklare et subtask eller et problem. C1-3q and C10-2 er gode eksempler. Er der brug for mere plads, så skriv en kravnote under tabellen.

Varianter af et subtask angives med a, b, etc. En variant betyder at subtasket kan udføres på mere end en måde. Fx kan vi enten registrere patienten (subtask 1) eller finde patienten i systemet (1a). Problemer der har med et subtask at gøre, anføres med p, q, etc.

Mange subtask går ud på at registrere eller bruge noget data, men nogle omfatter mere, fx at advisere andre (subtask 5), hælde medicin op, betale et beløb. Det er vigtigt at medtage dette selvom det sker manuelt i dag. Måske har leverandøren en løsning som kunden ikke har forestillet sig.

Problem = problem i dag: Kolonne 1 lister også problemer. Et problem skal være noget der volder kunden problemer med den nuværende løsning. Problem 3q er et godt eksempel. Kunden ønsker at leverandøren skal løse det problem. Vi ser desværre ofte *problem* brugt om noget man mener vil give problemer i fremtiden, fx at det vil være svært at give oversigt over data. Det er ikke tanken med *problem*. Vil man nævne den slags, hører det hjemme i kolonne 2, som handler om fremtiden.

Løsninger: Kunden kan skrive eksempler på løsninger i kolonne 2. Senere skriver leverandøren sit eget løsningsforslag der.

Som kunde skal man spare på sine løsningsforslag. Man skal ikke tvinge sig selv til at skrive noget "genialt" her. Skriv kun noget hvis det er en ikke-triviel løsning. Løsninger er ikke i bydemåde. De bør eksplicit angive hvem der gør hvad, fx *Systemet viser* eller *Brugeren vælger*.

Hvad er kravene? Lige efter overskrift C i skabelonen står der at kravet er at støtte alle task, herunder alle subtask, og reducere problemerne så vidt muligt. Det betyder at tabellens kolonne 1 er kravene (kundens behov). Kolonne 2 kan indeholde et løsningseksempel, men løsningen er ikke et krav. Ting uden for tabellen er antagelser leverandøren kan gøre, eller hjælp til læseren. Krav eller løsningseksempler der er for lange til at passe ind i tabellen, kan skrives udenfor tabellen, men skal have overskriften *kravnote* eller *løsningsnote*.

C2. Lignende task (Indskriv akut)

Task C2 handler om patienter der ankommer til skadestuen uden advarsel. Selv om tasket ligner C1 er der forskelle, og C2 kan have behov for en anden støtte.

Vær ikke bekymret over at de samme subtask optræder i både C1 og C2. Vi skal tjekke at de støttes godt i alle sammenhænge. En programmør vil prøve at genbruge kode - fint, men kravarbejde er ikke programmering. I kravarbejdet skal vi sikre os at alle brugssituationer støttes godt.

C10. Et komplekst task (Udfør klinisk session)

Den vigtigste aktivitet i et hospital er at undersøge og behandle patienter. Hvor mange task taler vi om? Er undersøgelse ét task og behandling et andet? Hvis vi studerer hvad der faktisk foregår, ser vi at undersøgelse, behandling og andre aktiviteter ofte udføres indenfor det samme korte tidsrum uden væsentlige afbrydelser. Dette tidsrum kalder vi en *klinisk session*. Det er vigtigt at computeren understøtter hele sessionen godt.

C2. Indskriv akut

Dette task opretter en indskrivning for en patient der ankommer til en skadestue uden henvisning …

Arbejdsområde 2: Patientbehandling

Området omfatter …

C10. Udfør klinisk session

En klinisk session kan omfatte diagnose, planlægning, udførelse, vurdering, mv. Som regel udføres der lidt af det hele, men det kan også ske at der fx kun udføres planlægning.

Start: Kontakt med patienten eller konference om patienten.
Slut: Når der ikke skal gøres mere med patienten lige nu.
Hyppighed: Totalt: Ca. 15.000 pr. døgn. Pr. bruger: Op til 20 pr. døgn.
Vanskeligt: Katastrofe med mange tilskadekomne. (Beskriv det hellere som et selvstændigt task. Se vejledningshæftet).
Brugere: …

Subtask og varianter:	Eksempel på løsning:	Kode:
1. Identificér patient.	Systemet kan læse et elektronisk armbånd, fx til bevidstløse patienter.	
2. Vurdér patientens tilstand. Se åbne diagnoser og tilhørende indikationer. Se notater. Se resultater fra ydelser der tidligere er rekvireret og sammenhold dem med operationelle mål. Det data der skal overskues omfatter D1 …	Systemet viser en oversigt på ét skærmbillede, fx med en Gantt-agtig tidsdimension. Brugeren kan vælge detaljer fra oversigtsbilledet.	
2p. **Problem**: I dag skal klinikerne logge ind og ud af flere systemer for at se alt relevant data.		
3. Giv ydelser der kan gives med det samme, fx blodtryk og SAT.	Systemet gør det let at registrere resultatet straks.	
4. Følg op på planlagte ydelser og resultater. Kontrollér om tidsfrister er udløbet.	Oversigten viser bestilte ydelser og deres tilstand, fx udløbne frister.	
5. Justér diagnoser (ret, tilføj, slet, prioritér). Afstem med vejledninger. Skriv notater.		
5p. **Problem**: Besværligt at få adgang til vejledninger.	Systemet kan vise vejledninger og checklister ud fra en valgt diagnose.	
6. Planlæg og bestil nye ydelser. Afstem med ledige tidspunkter hos alle parter - også patienten. (Se de lange subtask C11, C12 … om medicinordinering, booking …).	For bookinger viser systemet ledige tider for alle parter.	
6p. **Problem**: Man glemmer dele af bestillingen.	Systemet kan booke standardpakker af ydelser.	
6q. **Problem**: Fejl når data først noteres på papir, senere tastes ind.	Systemet gør det let at registrere alting straks.	
7. Evt. afslutning af indskrivning. (Se task C6).		

Mange patienter har flere diagnoser (sygdomme), og under den kliniske session prøver klinikerne at gøre noget ved dem alle. Det kan fx være opfølgning på behandlingen af én diagnose og planlægning af behandlingen af en anden.

Så tasket starter når klinikeren begynder at beskæftige sig med patienten, og slutter når man ikke kan gøre mere for patienten lige nu. Tasket indeholder mange slags subtask. Klinikeren afgør hvilke der skal udføres nu og i hvilken rækkefølge.

C11. Et langt subtask (Ordinér medicin)

Undertiden er der så mange subtask i et task at det bliver svært at overskue.

En løsning er at gruppere subtask i logiske grupper der hver har en overskrift. Vi har set eksempler hvor det virker fint med 50 subtask delt i ti grupper. Formålet med grupperingen er kun at hjælpe læseren. Subtask kan stadig udføres i næsten vilkårlig rækkefølge.

En anden løsning er at gøre hver gruppe til et *langt subtask* med sit eget C-nummer. Fx henviser C10-6 (planlæg og bestil nye ydelser) til flere lange subtask: C11 (ordinér medicin), C12 (booking), osv.

C11 er vist i detaljer. Bemærk at et langt subtask ikke har sin egen start- og slutbeskrivelse. Det er bare en del af hovedopgaven, C10. Til gengæld har det mening at angive hyppigheden fordi kun nogle af de kliniske sessioner har ordinationer eller bookinger.

Subtask 6 (beregn dosis) viser hvordan en **forretningsregel** kan indgå i et task. Vi kunne splitte subtask 6 i flere subtask, men som det står, overlader vi det til leverandøren.

C20. Andre omgivelser (udfør klinisk session mobilt)

Det sker at et task udføres i forskellige omgivelser med forskelligt behov for itstøtte. Et eksempel er den kliniske session (C10) når klinikeren bevæger sig rundt fra patient til patient. Her vil kunden gerne have støtte via PDA'er eller Smartphones. I teorien behøver vi ikke gøre andet ved kravspecifikationen end at skrive i indledningen til C10, at omgivelserne også kan være mobile.

Men hvor skulle leverandøren så beskrive sin løsning, som formentlig er anderledes end den normale PC støtte til C10? Og hvordan skulle kunden vurdere løsningen? Vi anbefaler derfor at gentage tasket for hver omgivelse:

C10: Udfør klinisk session, stationært.
C20: Udfør klinisk session, mobilt.

Hvad med det lange subtask C11 (ordinér medicin)? For at være sikker på at det også støttes i begge omgivelser, bør vi gentage det.

Ligesom for C2 (indskriv akut) skal vi ikke bekymre os om at samme subtask optræder i flere task. Vi skal kunne tjekke støtten til task i alle omgivelser.

Hvorfor ikke user stories eller use-cases

User stories

Ligesom task, prøver user stories at beskrive hvad brugeren skal med systemet. Vi ser User Stories blive brugt mere og mere i kravspecifikationer. De findes i mange udgaver. Her er tre typiske eksempler:

A. Som patientens læge vil jeg gerne se en oversigt over patientens diagnoser.

B. Som patientens læge vil jeg gerne se en oversigt over patientens diagnoser. (Vedhængt et skærmbillede med skitsen af oversigten – en ”wireframe”).

C. For at se patientens diagnoser højre-klikker jeg på patientens navn og vælger *Se diagnoser*.

C11. Ordiner medicin til patient

Dette er ikke et selvstændigt task, men et længere subtask der indgår i en klinisk session. Derfor er der ingen start-, slut- og brugerangivelse.

Hyppighed: Totalt: Ca. 30.000 pr. døgn. Pr. bruger: Op til 20 pr. døgn.

Subtask og varianter:	Eksempel på løsning:	Kode:
1. Vurdér patientens samlede medicinerings-situation, både i denne indskrivning og andre indskrivninger.	Systemet viser en oversigt over alle medicineringer, CAVE og diagnoser.	
1p. **Problem**: Besværligt at få adgang til vejledninger.	Systemet kan vise vejledninger og tjeklister ud fra diagnose og medicintype.	
...		
6. Beregn dosis. Kontrollér at den er rimelig. Kontrollér interaktion med anden medicin.	Systemet tilbyder en beregning baseret på legemsvægt hentet i journalen.	
6p. **Problem**: Omregning mellem enheder. Der kan være forskel på den enhed man ordinerer i (fx mg) og den enhed der doseres i (fx antal tabletter).	Systemet viser dosis i både ordinationsenheder og dosisenheder.	
...		

...

C20. Udfør klinisk session mobilt

Nogle kliniske sessioner kan udføres mens klinikeren bevæger sig fra patient til patient, fx med PDA eller mobiltelefon. I princippet er der samme subtask som i C10, men de kan ikke støttes på samme måde. Derfor gentages hele tasket her, således at leverandøren kan anføre den løsning han har til den mobile situation.

Start: Når …
Slut: Når …
Hyppighed: …

Alle tre eksempler er uegnede som krav. Man kan ikke se hvad oversigten skal bruges til. Er det for at finde en behandling, for at forklare et nyt symptom, eller for at skrive en epikrise (en rapport om patienten)? Selv om man får støtte til disse user stories, er det ikke sikkert at der er god støtte til det fulde task fra trigger til kaffepause. I værste fald kan man risikere at brugeren må skrive diagnoserne på papir for at kunne udføre næste trin af det fulde task.

Eksempel B og C er på et forkert niveau (design niveau) og foreskriver en bestemt løsning. Det er ikke hensigtsmæssigt hvis man vil have et næsten færdigt system.

Use cases

Use cases prøver også at beskrive hvad brugeren skal med systemet, men er også meget løsningsorienterede. De udføres over kort tid og strækker sig sjældent fra den egentlige trigger til den fortjente kaffepause.

I use cases nævner man ikke problemer. Nogle use-case forfattere bliver ophidsede hvis man gør det: *Så har man ikke gjort sit arbejde ordentligt*. (Jamen det er ikke mit job! Jeg forventer at leverandøren løser problemerne).

Hvis man bruger task-begrebet rigtigt, bliver der kun få task at beskrive. Mange store systemer kan beskrives med bare 10-30 task. Det er en fordel fordi man både får bedre oversigt og meget mindre at skrive.

Vi ser ofte kravspecifikationer hvor 10 task er svulmet op til 100 use cases, og hver af dem fylder en eller flere sider selvom der kun sker lidt i dem. Årsagen er som regel at det der burde være et subtask, er beskrevet som om det var et separat task med start og slut, hyppighed, mv. I virkeligheden er disse use cases ikke adskilte, men udføres sammen med andre use cases indtil vi ikke kan gøre mere ved sagen lige nu ("kaffepause"). Ud fra use cases får leverandøren ingen fornemmelse af hvordan use casene hænger sammen, og resultatet kan blive at han støtter dem dårligt.

Her er et skræmmende eksempel fra hospitalsverdenen:

En skadelig specifikation fra et virkeligt EPJ projekt - 3 sider i alt

Use case 2.1. Vis diagnoser
Den sundhedsfaglige person ønsker at opnå et overblik over patientens samlede diagnoser og disses indbyrdes relationer i diagnosehierarkiet.

Start: Brugeren ønsker at orientere sig om udviklingen i patientens helbredstilstand.
Slut: Brugeren har orienteret sig.
Precondition: Brugeren er logget ind. Patienten er registreret og valgt.

Trin:	Eksempel på løsning:
1. Vis diagnosehierarki.	
2. Vælg visning.	Fx et hierarki, Gannt-diagram …
3. Vælg detaljeringsniveau.	Fx udvid og indskrænk med plus og minus.
4. Vis notater om en udpeget diagnose.	
5. Vis dato og forfatter for notatet.	
6. Vis eventuelle eksterne årsager for diagnosen.	
. . . (osv. i alt 8 trin)	

Dette er ikke et korrekt task fordi det ikke er afsluttet i "kaffepause" forstand. Det vil være et subtask i et rigtigt task, fx en klinisk session. Use casen er en detaljeret løsning der beskriver en dialog mellem bruger og computer. Bemærk hovsa-triggeren: *Brugeren ønsker at …* Det er tegn på at vi ikke ved hvornår dette sker.

Ligesom for user stories, kan man ikke se formålet med sådan en use case.

Undgå at beskrive data som subtask

Use casen ovenfor er på tre sider i alt. En af grundene er at man beskriver data som trin. Notater, datoer og eksterne årsager håndteres som separate trin. Den oprindelige specifikation havde også use cases der hed *Opret diagnose* (4 sider) og *Ret diagnose* (3 sider). De henviste til næsten det samme data. Det var selvfølgelig svært at bruge datanavnene konsistent i alle use cases. Den oprindelige specifikation havde også en log-in use case og en vælg-patient use case.

Med SL-07 beskriver man data separat i kapitel D. Fra subtask kan man kort henvise til det data der er relevant i dette subtask. Skabelonen viser eksempler i C1-3 og C10-2.

Undertiden er det nyttigt at liste det nødvendige data mere præcist, fx i et enkelt subtask eller som en kravnote under tabellen.

Task har ingen preconditions

Use casen ovenfor har to "preconditions": Brugeren skal være logget ind og patienten registreret og valgt. Dette fremtvinger et flow mellem use cases. Brugeren skal først udføre *login* use casen, så *vælg patient* use casen, så *vis diagnose* use casen.

Rigtige task har ingen preconditions, men subtask kan have det, selvom vi sjældent har brug for at skrive dem. Brugeren kan starte en klinisk session når som helst uden preconditions. Det er en del af selve tasket at identificere og vælge patienten (subtask 1). Det er en implicit precondition for resten af tasket at det er sket. Fordi konteksten er let at se, er der ingen grund til eksplicit at skrive preconditions for alle disse subtask.

Hvad med login use casen? Ud fra et task-synspunkt er det ikke et behov, men en løsning på et problem: Hvem er brugeren og hvad har han ret til at gøre? Login er bare en besværlig måde at støtte det. SL-07 håndterer det i kapitel H, sikkerhed, og nævner det ikke i task.

Lauesen & Kuhail, 2012, har en detaljeret sammenligning af use cases og task. Den er baseret på kravspecifikationer fra 15 professionelle teams i 5 forskellige lande.

D. Data systemet skal opbevare

Dette kapitel beskriver det data systemet skal opbevare. Data kan beskrives på mange måder. Skabelonen viser 5 måder:

1. En kort forklaring af hver af dataklasserne (tabellerne).
2. En datamodel, også kaldet et E/R diagram eller Entitets/Relations diagram.
3. En databeskrivelse med detaljer om hvert felt, data volumen, etc.
4. Indholdet af eksisterende tabeller.
5. Indholdet af eksisterende skærmbilleder.

Desværre er der ingen ideel måde at beskrive data. Nogle er lette at forstå for interessenterne, men svære at gøre præcise og konsistente. Andre er omvendt.

Kapitel D af skabelonen starter med en kort forklaring af dataklasserne efterfulgt af en datamodel. Så er der detaljerede databeskrivelser og til sidst eksempler på tabeller og skærmbilleder som krav.

Data model (E/R)

Datamodeller er gode til at give overblik og konsistens. Domæneeksperter kan ofte forstå dem, men menige brugere har svært ved det.

Figur 3 i eksemplet er en datamodel (E/R diagram). Hver kasse er en klasse af data. Forestil dig at der er en stabel kartotekskort bag hver kasse. Bag Personkassen er der fx et kartotekskort for hver person systemet skal registrere. Kassen selv symboliserer kortet for en bestemt person. Derfor skal navnet på kassen være ental, fx *Person* og ikke *Personer*.

Ved siden af kassen er der en liste af felterne på kortet. Et Personkort indeholder personnummeret, navnet og andre simple felter. Der må ikke være repeterende felter, fx en liste af personens indskrivninger (i database-terminologi hedder det *første normalform*). Data om en indskrivning skal være på ét kort i Indskrivningskassen og må ikke gentages på Personkortet. På et skærmbillede kan vi godt vise en person plus alle hans indskrivninger, men ikke på et kort i datamodellen.

Der er relationer mellem kasserne vist som "hønseben". Et hønseben viser at et kort har en relation til et eller flere kort i en anden kasse. Fx kan en persons kartotekskort være relateret til flere indskrivningskort (strengt taget til nul, et eller flere indskrivningskort). Læser man hønsebenet den anden vej, er én indskrivning relateret til præcist ét personkort.

En punkteret kasse viser at data i denne klasse helt eller delvis deles med et eksternt it-system.

Når data er i en relationsdatabase, vil en kasse svare til en database-tabel. Hvert kartotekskort bag kassen svarer til en række eller "record" i tabellen. Men E/R diagrammer er også yderst nyttige når data ikke er i en database.

Ved siden af kassen viser diagrammet en liste af kartotekskortets felter. I mange tilfælde viser man kun de vigtigste felter og antyder med … at der er flere. Bemærk at vi ikke viser tabellernes fremmednøgler. Det er database-teknik og forvirrer brugerne. Hønsebenene viser allerede det væsentlige.

UML klassemodeller bruges meget. En UML model ligner en E/R-model, men listen af felter står inden i kassen, så kasserne bliver meget store. Relationerne mellem

kasserne er vist som rette eller knækkede linjer med 0:1, 1:*, etc. i enderne. Disse små forskelle gør en stor forskel når man prøver at få oversigt over et stort diagram. Vores hjerne kan meget lettere overskue et E/R-diagram end et UML-diagram. Desuden fylder et UML-diagram typisk fem gange mere.

D. Data systemet skal opbevare

Systemet skal registrere de data der er beskrevet i dette kapitel. Data skal kunne oprettes, ses og ændres gennem de relevante task i kapitel C. Data skal i mange tilfælde kunne udveksles med eksterne systemer som beskrevet i kapitel F.

Figur 3 er en datamodel (et Entitets/Relations diagram, E/R) der giver en oversigt over data. Hver kasse er en dataklasse. Bag kassen er der en stabel kartotekskort. Kassen selv symboliserer et af kortene. Fx indeholder D5 et kort for hver person systemet kender. Ved siden af kassen er der en liste af de felter som kortet indeholder.

Der er relationer mellem kasserne vist som "hønseben". Et hønseben viser at et kort har en relation til et eller flere kort i en anden kasse. Fx kan en person have flere indskrivninger, mens én indskrivning kun vedrører én person. Data skal ikke nødvendigvis struktureres på denne måde i systemet, men det skal håndteres på en eller anden måde.

De punkterede kasser viser data der helt eller delvis er delt med eksterne systemer.

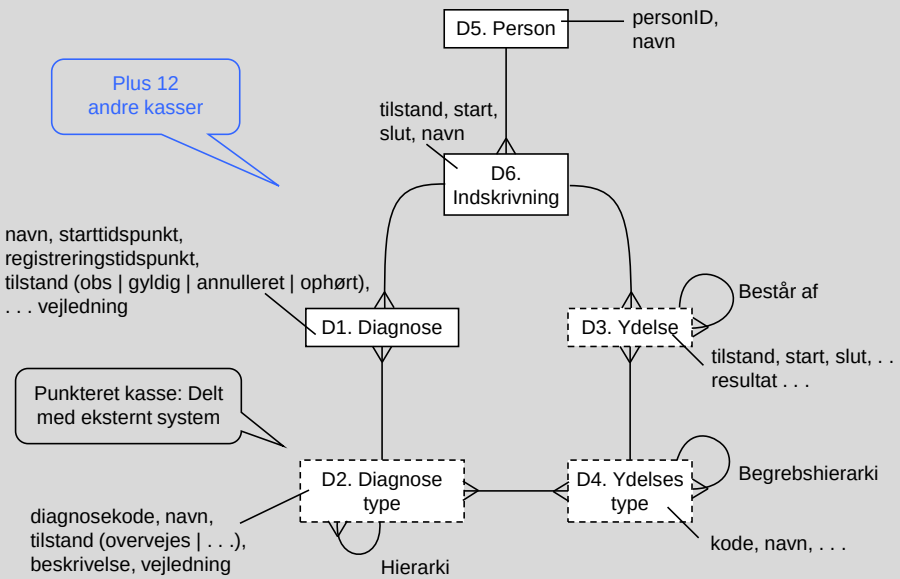
D1. Diagnose: Hver record indeholder data om en af patientens sygdomme. Den svarer til SKS-klassifikationen af sygdomme, men der skal også være mulighed for at registrere sygdomme der ikke er i SKS eller som først senere klassificeres.

D2. Diagnosetype: Hver record indeholder data om en bestemt type diagnose - uafhængigt af patienten: diagnosens navn og SKS-kode (hvor muligt), vejledning, standard behandlingsplaner (gennem relationen til ydelsestyper) … Klinikerne vil vælge diagnoser fra dette katalog af diagnosetyper.
…

D5. Person: Hver record indeholder data om en person: navn, adresse … En person kan være en kliniker, en patient, eller en pårørende.

D6. Indskrivning: Hver record indeholder data om et behandlingsforløb: start tidspunkt, relateret person, …

Figur 3. Datamodel for EPJ



D0. Fælles felter

I mange systemer skal man holde styr på historikken, dvs. hvem har oprettet og ændret data og hvornår. Det er felter som alle tabeller skal have. Gamle udgaver af "kartotekskortene" skal gemmes. Rent teknisk kan det gøres på flere måder, men det er løsninger som kunden ikke behøver at interessere sig for. Det væsentlige er de krav der står i D0.

D1. Databeskrivelse (Diagnose)

Dette afsnit er en databeskrivelse for diagnoser. En databeskrivelse kaldes også en *data dictionary*. Den består af disse dele:

1. Nummer og navn på dataklassen. Dataklasser nummereres D1, D2, osv. For at undgå for meget omnummerering, kan man dele klasserne i bundter og starte hvert bundt med et rundt tal.

2. Eksempler på hvad et kartotekskort kan indeholde. Giv både typiske og usædvanlige eksempler. For diagnoseklassen svarer et kartotekskort til diagnosen for en bestemt patient. Diagnosen kan fx være kolera eller hoste.

3. Datakilden. Hvor kommer data fra? Data kan indtastes under et task, opsamles automatisk af systemet, eller importeres fra et andet system. I mange tilfælde kommer alle felter fra samme kilde, i andre tilfælde har nogle felter en særlig kilde. I eksemplet hentes diagnosenavnet fra diagnosetype klassen, men det kan også indtastes af klinikeren.

4. Databrug. Hvad bruges data til? Det kan bruges i et task eller eksporteres til andre systemer. Også her kan der være en fælles beskrivelse for alle felter eller særlige beskrivelser for nogle af dem.

5. Datavolumen. Dette står i tabellen og er derfor et krav. Systemet skal kunne opbevare så meget data. Afsnit L3 angiver hvor lang tid data skal opbevares og hvor hurtigt arkiveret data skal kunne hentes.

I eksemplet gives antallet af diagnoser pr år. Det antyder hvor mange der oprettes pr dag og hvor mange der oprettes ved spidsbelastning. Dette er vigtigt når man angiver svartider i afsnit L1. Hvis der er særlige spidsbelastninger, bør det angives under datavolumen.

6. En tabel med detaljer om hvert felt. Felter nummereres fortløbende. Problemer med et felt nummereres p, q, r. Tabellen har tre kolonner ligesom task. Bemærk at vi beskriver hønseben (relationer) som en slags felter, fx D1-2 og 3.

Eksemplet er skrevet uden detaljer om dataformatet (fx om det er tekst eller tal). I nogle tilfælde er detaljer om fx datoformater og tekstlængder dog nyttige, fx som i løsningskolonnen for D1-4. Hvis et specielt format er nødvendigt, skal det være et krav og stå i kolonne 1. Brug det sparsomt. Ellers bliver det svært at finde et standard/ramme-system der opfylder kravet.

Bemærk at problemer, kravnoter og løsningsnoter kan bruges på samme måde som i andre dele af kravspecifikationen.

D0. Fælles felter

Alle dataklasser har historik, dvs. at enhver ændring giver en ny udgave af "kartotekskortet" og den gamle bevares. Det registreres i disse felter:

Felter og relationer:	Eksempel på løsning:	Code:
1. Ændringstidspunkt: Tidspunkt hvor "kartotekskortet" blev oprettet eller ændret.		
2. Kilde: Den person der oprettede eller ændrede kortet.		
3. Historik: Relation til tidligere udgaver af "kartotekskortet" (ikke vist på E/R-diagrammet).		

D1. Diagnose

En diagnose er en sygdom eller et symptom hos en konkret patient.

Eksempler: Der er en flydende overgang mellem sygdomme og symptomer. Fx er både kolera og hoste "diagnoser".
Datakilde: Diagnoser registreres under en klinisk session (C10) og ofte under indskrivningen (C1).
Databrug: Diagnoser vises i oversigten for en patient, ved afregning og ved indberetning til myndighederne.

Datavolumen:	Eksempel på løsning:	Kode:
1. Der registreres op til 800.000 diagnoser årligt.		

Felter og relationer:	Eksempel på løsning:	Kode:
2. Diagnosekode: Relation til Diagnosetypen. Patientens primære diagnose kan ændres under behandlingen. Den primære diagnosetype bruges ved afregning og indberetning.		
2p. Problem: Meget svært at finde den rigtige SKS-kode ud af de 20.000 mulige.	Se løsningsnoten nedenfor.	
3. Indskrivning: Relation til Indskrivning, som viser videre til patienten (Person).	Systemet registrerer det automatisk ud fra den valgte patient.	
4. Navn: Normalt navnet fra Diagnosetypen, men kan også være et navn indtastet specielt for denne patient.	Feltlængde: 100 tegn.	
5. Tilstand: En diagnose kan være i følgende tilstande: Obs, gyldig, annulleret, ophørt.		
6. Starttidspunkt: Det tidspunkt hvorfra diagnosen er i denne tilstand. Er ofte det samme som Ændringstidspunkt, men ikke altid, fx hvis man registrerer at patienten havde hoste i går.	Systemet gør det let at vælge registreringstidspunktet som Starttidspunkt.	
...		
17. Vejledning: Den vejledning der var gældende på det tidspunkt hvor diagnosen blev oprettet.		

Løsningsnote

Brugeren kan fx vælge en diagnose på disse måder:

a. Mulighed for at søge i et begrebshierarki (svarende til SKS over- og underklasser).
b. Reduceret hierarki tilpasset den enkelte afdeling så de normalt kun ser diagnosetyper relevante for dem.
c. "Live search" hvor brugeren indtaster dele af diagnosens navn og systemet viser mulige match for hvert tastetryk.

D2. En typeklasse (Diagnosetype)

Diagnoseklassen D1 indeholder patienternes faktiske diagnoser. D2 er derimod et eksempel på en typeklasse. Kartotekskortene bagved D2-kassen udgør et katalog over mulige diagnoser.

Det er som regel vigtigt også at angive typeklasser, især hvis man skal kunne tilføje typer, ændre dem, og måske have historik om hver type. EPJ-systemet henter diagnosetyperne fra Sundhedsstyrelsens katalog (SKS), se F1.

Bemærk hvordan D2-5 angiver et eksempel i kolonne 1 (Kolera DA00). Det er en god måde at forklare hvad feltet kan indeholde. Vi ser tit at man skriver sådanne eksempler i kolonne 2. Det er forkert, kolonne 2 er til eksempler på *løsninger*. DA00 er ikke en løsning.

Bemærk hvordan D2-6 håndterer længden af beskrivelsesfeltet. Det bør være ca. to linjer, men det præcise tal er ikke vigtigt. Kunden har derfor foreslået en længde i løsningskolonnen. Leverandøren kan ændre den til noget der er bekvemt for ham, fx 255 tegn.

I EPJ-eksemplet er der også en ydelsestype (D4 i modellen). Den svarer til et katalog af alle de mulige ydelser en patient kan få, fx *blodtryksmåling* og *bypass-operation*.

I nogle tilfælde kan der være flere type-niveauer. Fx ordinerer læger ikke bare Aspirin. De angiver ydelsestypen "Aspirin, 12 tablets pakning". Denne ydelsestype hører til en medicintype der er "Aspirin, 500 mg tabletter". Denne medicintype hører til en præparattype der er "Aspirin, tabletter", som hører til aktiv-ingredienstypen "Acetylsalicylsyre".

Medicintype, præparattype og aktiv-ingredienstype er separate dataklasser (kasser) som ikke er vist i skabelonens eksempel.

D2. Diagnosetype

Samlingen af diagnosetyper udgør diagnosekataloget.

Eksempler: DA009: Kolera vibrio eltor; DR059: Hoste.
Datakilde: Importeres fra SKS.
Databrug: Brugeren vælger en diagnosetype når han registrerer en patientdiagnose.

Datavolumen:	Eksempel på løsning:	Kode:
1. Der vil være op til 30.000 diagnosetyper. SKS har i dag 20.000 typer.		

Felter og relationer:	Eksempel på løsning:	Kode:
2. Diagnosekode: SKS-kode eller en midlertidig kode.		
3. Navn: Det fulde navn på diagnosen, fx "kolera uden specifikation".		
4. Tilstand: En diagnosetype kan være i følgende tilstande: Overvejes, gyldig, forældet.		
5. Hierarki: Relation til en mere generel diagnosetype, fx "Kolera, DA00".	Eksempel: "Kolera, DA00". FORKERT - ikke **løsnings**eksempel	
6. Beskrivelse: En længere beskrivelse af diagnosen, men ikke mere end et par linjer. Endnu længere beskrivelser kan evt. findes i "Vejledning".	Feltlængde 160 tegn.	
7. Ydelsestyper: En relation til ydelsestyper der kan bruges i behandlingen af diagnosen.	Systemet udtrækker dem selv fra oplysningerne i "Vejledning".	
...		
10. Vejledning: En lang tekst der beskriver indikationer, praksis, etc.	Kan fx være en URL.	

D3. Vis eksisterende tabeller og skærmbilleder (Ydelse)

I dette afsnit viser vi nogle andre måder at kravspecificere data: Eksisterende tabeller og eksisterende skærmbilleder.

Der er mange slags ydelser i et EPJ-system. Det er svært for kunden at angive dem alle. I den første del af D3 har kunden angivet de felter og relationer der er fælles for alle ydelser.

D3-4 er ydelsens tilstand. Når klinikeren bestiller en ydelse, starter den i tilstanden *bestilt*. Så bliver den *bekræftet* af den der leverer ydelsen, derefter *startet* og *færdig*. Den bør ende med at være *vurderet* af lægen. At holde styr på ydelsens tilstand og hvornår den skifter, er vigtigt når tingene ikke går som forventet og man skal korrigere. Det gør det også muligt for systemet at advare når noget er blevet glemt. Reglerne for tilstandsskift og advarsler kan være indviklede. Man kan skrive reglerne som kravnoter under tabellen eller som forretningsregler i kapitel E.

Afsnit D3.1 angiver de ydelser der er kliniske målinger. I princippet burde man beskrive de specielle felter for målinger i tabel D3.1, men kunden har ikke den nødvendige viden. I stedet referer han til et skærmbillede der stammer fra det eksisterende database-system. Skærmbilledet lister de eksisterende felter. Det giver leverandøren en forestilling om hvad der er brug for, men man må evt. afklare detaljerne under udviklingen.

Afsnit D3.2 angiver de ydelser der er kirurgiske operationer. De er specificeret på samme måde som kliniske målinger.

D3. Ydelse

En ydelse er noget der måles eller gives til en patient. Der er mange undergrupper af ydelser, fx målinger, operation og medicin. I dag er de lagret i forskellige tabeller eller endda i eksterne systemer der skal integreres.

Felter og relationer fælles for alle ydelser:	Eksempel på løsning:	Kode:
1. Ydelseskode: Relation til Ydelsestype.		
2. Indskrivning: Relation til Indskrivning, som viser videre til patienten (Person).	Systemet registrerer det automatisk ud fra den valgte patient.	
3. Start: Det tidspunkt ydelsen blev givet.		
4. Tilstand: I et normalt forløb gennemløber en ydelse disse tilstande: Bestilt, bekræftet (af ydelsesleverandøren), startet (fx prøve taget), færdig, vurderet (af klinikeren). Undtagelsesvis kan tilstanden være: Afbrudt, ændret.		
5. Består af: En relation til ydelser der er en del af denne ydelse, fx "operation" der består af flere ydelser.		

D3.1. Patientmåling

Eksempler: Blodtryk; Vægt; B-glukose; Gamma globulin; Røntgen.
Datakilde: Nogle registreres under en klinisk session, andre importeres fra et eksternt system, fx laboratoriesvar.
Databrug: Vises i oversigten for en patient og i detaljebilleder for at støtte diagnosticering og behandling.

Datavolumen:	Eksempel på løsning:	Kode:
1. Der registreres omkring 100.000 målinger om dagen. Af dem er 5.000 billeder.		

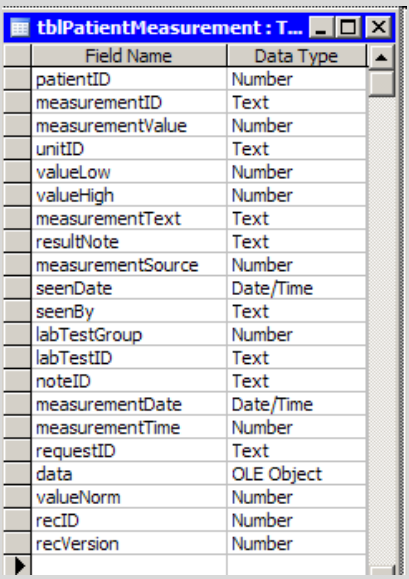
Felter:	Eksempel på løsning:	Kode:
2. En patientmåling bør indeholde data fra den eksisterende tabel, se figur 4, tblPatient-Measurement. Bemærk at denne tabel ikke indeholder disse fælles felter for ydelser: indskrivningsidentifikation og tilstand.		

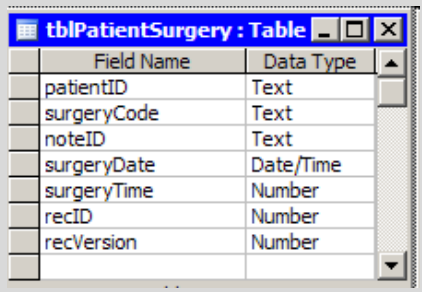
D3.2. Patientoperation

Eksempler: Bypass-operation; Fotodynamisk terapi (PDT).
Datakilde: Registreres før og efter operationen.
Databrug: Vises i oversigten for en patient og i detaljebilleder til diagnosticering og behandling.

Datavolumen:	Eksempel på løsning:	Kode:
1. Der registreres omkring 100 operationer om dagen.		

Felter:	Eksempel på løsning:	Kode:
2. En patientoperation bør indeholde data fra den eksisterende tabel, se figur 4, tblPatientSurgery. Bemærk at denne tabel ikke indeholder disse fælles felter for ydelser: Indskrivnings-identifikation og tilstand.		





Afsnit D3.3 angiver de ydelser der er medicin. I dette tilfælde havde kunden ikke tabelformatet, men brugte et skærmbillede fra det eksisterende medicinsystem. Det giver også leverandøren en forestilling om hvad der er brug for, men også her må man afklare detaljerne under udviklingen.

D3.3. Medicinering

Eksempler: Ibumetin, 400 mg*3; Furix, 40 mg*2.
Datakilde: Registreret som ordinationer under en klinisk session.
Databrug: Vises i oversigten for en patient og i detaljebilleder til diagnosticering og behandling.

Datavolumen:	Eksempel på løsning:	Kode:
1. Der registreres omkring 30.000 ordinationer om dagen.		

Felter:	Eksempel på løsning:	Kode:
2. En medicin-ordination bør indeholde de data det eksisterende system viser. Se figur 5, som er et skærmbillede fra det eksisterende system.		

Figur 5

Fast

Start ▲	Slut	Type	Lægemiddel	Vareform	Dosis	Døgndosis	Adm.vej	Info
13.07.06		Fast.Inf.	Cefuroxim 1500 ...	15 mg/ml inf.v...	1500 mg x 3	4500 mg	IV	
17.08.06		Fast	Furix	10 mg/ml inj.v...	40 mg x 2	80 mg	IV	
20.10.06		Fast.Inf.	Metronidazol "Bax...	5 mg/ml inf.v...	500 mg x 3	1500 mg	IV	
13.07.06		Fast	Selo-zok	100 mg depot...	100 mg x 1	100 mg	OR	
13.07.06		Fast	Laktulose SAD	667 mg/ml mi...	13340 mg x 1	13340 mg	OR	
13.07.06		Fast	Multivitamin mine...	tabletter	1 stk x 1	1 stk	OR	
03.10.06		Fast	Ibumetin	400 mg tablet...	400 mg x 3	1200 mg	OR	
03.10.06		Fast	Picolon	7,5 mg/ml ora...	10 drb x 1	10 drb	OR	
01.02.08		Fast	Pinex	500 mg filmov...	1000 mg x 3	3000 mg	OR	

(Blank med vilje)

E. Andre funktionelle krav

De fleste systemfunktioner er simple oprettelser, sletninger, editeringer og forespørgsler der implicit fremgår af arbejdsopgaverne og datakravene. Dette kapitel beskriver funktionalitet der er mere kompleks.

E1. System-genererede hændelser

Systemet kan gøre ting på egen hånd, fx hente data fra omgivelserne, eller sende påmindelser til brugere når tidsfrister er overskredet.

Krav E1-1 beder om en påmindelse når en indskrivning er blevet "glemt". Der skal være et task der kan tage sig af denne påmindelse. I eksemplet tager task C1, *Indskriv patient inden ankomst*, sig af sagen som en af de mulige triggere.

Krav E1-2 beder om en påmindelse når et laboratoriesvar udebliver. Også her skal der være et task der kan tage sig af påmindelsen. Dette task er ikke med i skabelonen. Det udføres af en afdelingssekretær eller ledende sygeplejerske.

E2. Udskrifter og rapporter

Ofte er man i den situation at det eksisterende system kan udskrive et hav af rapporter, men for de fleste ved man ikke om de bruges og til hvad. Skabelonen viser flere muligheder for at omsætte det til krav.

Udskrift 1 har en veldefineret brug og man kan beskrive formatet præcist, fx ved at vedlægge et eksempel.

Udskrift 2 forklarer behovet, men giver ikke et præcist format. Det kan være en fordel at henvise til det eller de task hvor rapporten skal bruges, så leverandøren bedre kan vurdere hvad der er hensigtsmæssigt.

Udskriftskrav 3 omgår problemet ved at bede leverandøren give en fast stykpris for at lave en rapport. På den måde kan kunden udskyde beslutningen om hvilke rapporter der er brug for. Den faste pris forhindrer leverandøren i at misbruge det monopol han har efter indgåelse af kontrakten. Leverandøren må angive en pris, som evt. kan afhænge af rapportens kompleksitet. Han kan fx angive en pris pr. Function Point (se L5-7).

Krav 4 omgår problemet på en anden måde, nemlig ved at bede om en rapportgenerator. Så kan kunden selv udvikle rapporterne. I eksemplet beder man leverandøren specificere hvor let det er at udvikle rapporterne, fx ved at angive hvor mange superbrugere der kan gøre det.

Krav 5 præciserer at alle rapporter skal kunne vises både på skærm og print.

Krav 6 handler om et beslægtet problem: Brugeren har egentlig ikke brug for en rapport, men vil udforske data.

E. Andre funktionelle krav

De fleste systemfunktioner er simple oprettelser, sletninger, editeringer og forespørgsler, som ikke er specificeret yderligere. Det fremgår implicit af task (kapitel C), systemintegrationer (kapitel F) og databeskrivelserne (kapitel D). Desuden skal systemet kunne udføre funktionerne i dette kapitel.

E1. System-genererede hændelser

Systemet skal kunne generere disse hændelser:	Eksempel på løsning:	Kode:
1. Hvis en indskrivning har været parkeret i X dage, skal lægesekretæren have en påmindelse. IT-afdelingen skal kunne definere X.	X er typisk 4 dage, men kan variere fra afdeling til afdeling.	
2. Hvis et laboratoriesvar er bestilt men ikke færdigt indenfor 24 timer, skal klinikerne have en påmindelse.		

E2. Udskrifter og rapporter

Nogle udskrifter og rapporter skal bruges i forbindelse med de konkrete task beskrevet i kapitel C. Udskriftsformaterne er ikke afgørende, når blot task støttes effektivt. Disse rapporter beskrives ikke i dette kapitel. Der er dog også behov for udskrifter der bruges til ad hoc formål, på tværs af task, og udskrifter med et præcist format. De beskrives her.

Udskriftskrav:	Eksempel på løsning:	Kode:
1. Checks skal udskrives på fortrykte blanketter med formatet vist i …		
2. Systemet skal kunne vise en oversigt og prognose over sengebelægningen (bruges bl.a. i task …).	Figur … viser et eksempel på en sådan udskrift.	
3. Leverandøren kan levere op til 100 nye rapporter til en fast stykpris som led i vedligeholdelsen.	Prisen pr. rapport er ____. (Prisen kan afhænge af kompleksiteten).	
4. Systemet har en rapportgenerator der er let at bruge.	__ % af superbrugerne kan udvikle rapporter som dem i bilag X. Kunden forventer 50%.	
5. Alle udskrifter og rapporter skal kunne vises på skærm såvel som i print.		
6. Superbrugere kan analysere data efter behov.	Systemet kan overføre data til regneark.	
…		

E3. Forretningsregler og komplekse beregninger

Beregninger og regler kan beskrives på flere måder. Nogle passer fint ind i task-beskrivelser, fx dette subtask i C11, *Ordiner medicin*:

> *Kontrollér interaktion med anden medicin.*

Leverandøren kan specificere at hans system automatisk gør det og hvordan.

Andre regler er del af datakravene (fx mulige tilstande af en ydelse) eller del af sikkerhedsreglerne (fx hvem har ret til at gøre hvad?). Dette afsnit specificerer andre slags komplekse regler.

E3-1 i eksemplet kræver en beregning der er beskrevet i et separat bilag (ventelisteberegning). Bilaget kan fx indeholde en algoritme skrevet som et lille program, et *flow chart*, eller en tabel over mulighederne.

E3-2 henviser til et offentligt dokument hvor reglerne er beskrevet (lønoverenskomster). For at kunne omsætte dette til en løsning, skal leverandøren have stor ekspertise på lønområdet.

Man kan også indirekte beskrive en funktion ved et nøjagtighedskrav, fx at systemet skal kunne genkende menneskelig tale med en baggrundsstøj på 30 dB. Eller at systemet skal kunne beregne en vagtplan der højst er 3% dyrere end den optimale plan.

E3-3 viser en regel der er udtrykt som et tilstandsdiagram (figur 6). En patientdiagnose kan være i en af disse tilstande: *Obs, Gyldig, Annulleret, Ophørt.* Officielt kan den kun skifte tilstand som vist med pilene. Brugerens handlinger kan bevirke alle disse tilstandsændringer undtagen sletning af diagnosen. Sletning sker automatisk efter 20 år. Men som krav E3-3 forklarer, skal brugeren kunne lave en hvilken som helst tilstandsændring alligevel.

E3-4 viser en mere kompleks regel som et tilstandsdiagram (figur 7). Det viser tilstandene for en LabSys ydelse, som den er registreret i EPJ-systemet. De mulige tilstande er vist som kasser med runde hjørner. Det er de ydelsestilstande som er nævnt i D3-4. Diagrammet viser hvordan tilstanden skifter.

Klinikeren opretter ydelsen via EPJ-systemet, som sætter den i tilstanden *Bestilt*. Samtidig sender EPJ-systemet en Rekvisition til LabSys. Labsys sender en LabBekræft meddelelse til EPJ-systemet, der sætter ydelsestilstanden til *Bekræftet*. Klinikeren sender nu den fysiske prøve og markerer det i EPJ-systemet, som sætter tilstanden til *Startet*. Senere sender LabSys en besked med resultatet til EPJ-systemet, som sætter tilstanden til *Færdig*. Når lægen senere ser resultatet i EPJ-systemet, sætter systemet tilstanden til *Vurderet*.

Den slags diagrammer kan detaljeres yderligere med aktivitetsdiagrammer (fra UML) eller SDL (fra teleindustrien). Nogle gange er sådan et detaljeringsniveau vigtigt, men i de fleste tilfælde er det at specificere en løsning fremfor et behov. I eksemplet er brugeren ligeglad med disse LabSys detaljer, men det er vigtigt for ham at kunne se hvor langt LabSys-ydelsen er kommet. Man kunne i stedet skrive dette som det egentlige krav.

E3. Forretningsregler og komplekse beregninger

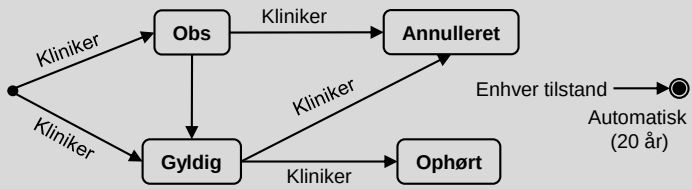
Nogle forretningsregler er specificeret i subtask, fx *Kontroller at …* Se eksemplet i C11-6. Andre forretningsregler er specificeret i databeskrivelserne Se eksemplet i D3-4. og nogle er specificeret som adgangsrettigheder (afsnit H1). Her er yderligere forretningsregler og komplekse funktioner:

Funktion:	Eksempel på løsning:	Kode:
1. Ventelisteprioritering skal beregnes som beskrevet i …		
2. Lønberegningen skal følge de til enhver tid gældende overenskomster (se også vedligeholdskravene i …).		
3. En diagnose må normalt kun skifte tilstand som beskrevet i figur 6. I tilfælde af fejltagelser skal brugeren dog have mulighed for at afvige fra reglerne (se også H4-2).	En bruger der forsøger at afvige fra reglerne, spørges om det er hensigten. I så fald udføres ændringen og den logges i …	
4. Inde i systemet skal en laboratorieydelse skifte tilstand som beskrevet i figur 7.		

Kravnote: Tilstandsdiagram

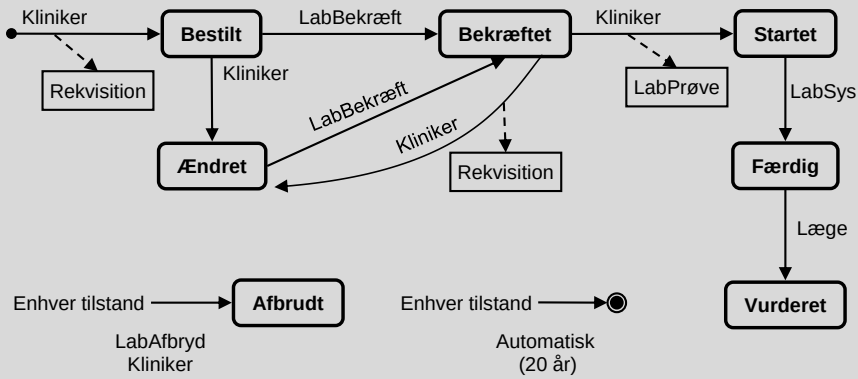
Figur 6 viser at en kliniker opretter diagnosen. Den skabes i tilstanden *Obs* eller *Gyldig*. Klinikere kan ændre tilstanden ifølge diagrammet. Diagnosen forsvinder når systemet rydder op efter 20 år.

Figur 6. Diagnosetilstande



Figur 7 viser hvordan tilstanden af en laboratorieydelse skifter i systemet. En kliniker opretter en LabSys service i tilstanden *Bestilt*. Under oprettelsen sender systemet en rekvisition til LabSys. Når LabSys sender en LabBekræft-meddelelse til systemet, skifter ydelsens tilstand til *Bekræftet*. En kliniker tager en prøve på patienten, sender den til laboratoriet og fortæller det til systemet, som ændrer ydelsens tilstand til *Startet*. Ydelsen kan skifte tilstand på andre måder, som angivet i diagrammet.

Figur 7. LabSys ydelse: Tilstande og meddelelser



E4. Udbygning af systemet

I nogle tilfælde er der behov for at kunden selv kan udbygge systemet på et eller andet område. Han kan fx ønske at eksperimentere med nye skærmbilleder for at forbedre brugervenligheden, eller han frygter at leverandøren vil kræve en urimelig pris for at udbygge systemet.

Dette afsnit stiller krav om funktionalitet der gør det muligt at foretage visse udbygninger uden at indblande leverandøren. Tidligere var leverandører meget utilbøjelige til at åbne for den slags, fordi de frygtede for systemets korrekthed og stabilitet. Dette har dog ændret sig, og selv ERP-systemer som SAP og Axapta giver bedre og bedre muligheder for at udbygge systemet.

I EPJ-eksemplet er behovet meget udtalt fordi der er over 20.000 slags ydelser med hver deres datafelter, og nye kommer til hele tiden. Man vil ikke kunne leve med at leverandøren skal rette i systemet hver gang en ny slags ydelse kommer til. Desuden er der mange lægefaglige specialer som har brug for oversigter på hver deres facon.

Der er også behov for fremtidig integration med eksterne systemer. Det beskrives i afsnit F10.

Bemærk at skabelonen ikke alene stiller krav om funktionalitet, men også om ret til at bruge den (E4-8). Det skyldes tidligere dårlige erfaringer med leverandører der leverer funktionaliteten, men beholder retten til at bruge den og til at udtrække de data systemet opbevarer.

E4. Udbygning af systemet

Systemet viser og vedligeholder data gennem brugerens skærmbilleder. I dette afsnit betyder "kunden" hans egen IT-stab eller tredjepart bemyndiget af ham. Kunden forventer at kunne modificere skærmbillederne og tilføje nye, sådan at der kan skabes overblik for lægefaglige specialer, nye arbejdsgange, etc.

Systemet håndterer mange slags ydelser, mange af dem med særlige kombinationer af data. Kunden forventer at kunne tilføje nye typer ydelser. Nedenstående krav præciserer behovene.

Krav til udbygningsmuligheder:	Eksempel på løsning:	Kode:
1. Kunden kan definere nye ydelsestyper baseret på data i kapitel D.		
2. Kunden kan definere skærmbilleder der kombinerer data fra hele datamodellen i kapitel D (vilkårlige views af data).		
3. Skærmbillederne kan aktivere funktionalitet i EPJ-systemet og i eksterne systemer forbundet med EPJ-systemet. Fx bestilling af en service, notifikation, udskrift af en rapport.		
4. Skærmbillederne kan sammensættes af mange typer komponenter og deres farve kan afspejle datas værdi. Fx tekstfelter, tabeller, knapper, grafer, billeder.		
5. Kunden kan tilføje nye typer komponenter til brug i skærmbillederne.		
6. Der kan defineres skærmbilleder for flere slags udstyr. Fx PC, PDA, Smartphone.		

Dokumentation og rettigheder:	Eksempel på løsning:	Kode:
7. Værktøjerne til opbygning af billeder, tilføjelse af komponenttyper, mv. skal dokumenteres så kundens IT-stab eller tredjepart kan forstå dem og bruge dem til formålet.	Et kursus på ___ dage er nødvendigt for at kunne bruge værktøjerne.	
8. Kunden skal have ret til at bruge værktøjerne og systemets data.		

F. Integration med eksterne systemer

Integration betyder at to systemer skal kommunikere. Som regel drejer det sig om at data skal overføres fra det ene system til det andet. Tendensen er at nye systemer skal integrere med flere og flere andre systemer - såkaldte **eksterne systemer**. Mere end 10 eksterne systemer er almindeligt.

I nogle tilfælde kan vi undgå eksplicitte integrationskrav fordi fuld støtte til task kræver integration. Det gjorde vi i C1-3a (brug af MedCom til dataoverførsel). Men som regel er integration en indviklet affære, og det vil være svært at vurdere en leverandørs integrationsløsning ved at udføre nogle task. Det er især svært hvis vi ønsker et tidligt bevis (B3). Så vi har normalt brug for eksplicitte integrationskrav.

Skabelonen starter med en verbal oversigt over de eksterne systemer samt en grafisk oversigt i form af et kontekstdiagram. Diagrammet svarer til kontekst-diagrammet i afsnit A1, men vil som regel have flere detaljer.

Vis systemet der skal leveres som en kasse med dobbelt-linje ramme. Vis de integrationer leverandøren skal udføre som dobbelt-linje pile. Lad pilene pege den vej data løber. Sæt et navn på hver pil for at antyde det data der løber.

I eksemplet skal leverandøren integrere med det eksisterende SKS-system og LabSys. Han skal levere et nyt medicinsystem eller integrere med det eksisterende. Bemærk at han ikke skal integrere med nye eksterne systemer. Det skal andre kunne gøre.

Hvilket system skal starte dataoverførslen? Det afhænger af hvad der er muligt i de eksisterende systemer - og egentlig er det ligegyldigt for kunden. Han skal kun interessere sig for at hans behov er opfyldt. Så hvad er de egentlige behov? Undersøgelse af mange integrationer viser at der er flere aspekter:

A. Adgangsret til data. Hvem må overføre hvad?
B. Beskyttelse af data: Undgå datatab, dublering og forvanskning af data.
C. Dokumentation og rettigheder: Hvad skal dokumenteres? Hvem må bruge det og til hvad?
D. Ansvar: Hvem skal udvikle og teste integrationen og hvordan skal den "anden ende" hjælpe?
E. Task støtte: Kan brugernes task støttes godt med denne integration?
F. Dataimport fra det eksterne system: Hvilket data?
G. Dataaktualitet: Hvor gammelt er det data kundens system viser? Dette er det vigtigste aspekt af integration. Med den ideale SOA-arkitektur (se nedenfor) er data på skærmen kun nogle få sekunder gammelt. Med en batchvis overførsel kan det være timer eller uger gammelt, men det kan være godt nok.
H. Svartid ved import: Når systemet beder om import af data, hvor hurtigt skal det så overføres?
I. Dataeksport til det eksterne system: Hvilket data og hvornår?
J. Svartid ved eksport: Når systemet beder om eksport af data, hvor hurtigt skal det så overføres?
K. Anden funktionalitet: Kan systemet beordre andre funktioner i det eksterne system, fx sende en påmindelse til brugere eller printe data? Tilbyder det selv funktioner?

Skabelonen har afsnit og eksempler for hvert af disse aspekter.

F. Integration med eksterne systemer

Systemet skal i større eller mindre grad integreres med de eksterne systemer der er vist i figur 8 (kontekst diagram). Integrationen består dels i datadeling eller replikering, dels i at brugeren via systemet kan aktivere funktionalitet i andre systemer (eksterne systemer).

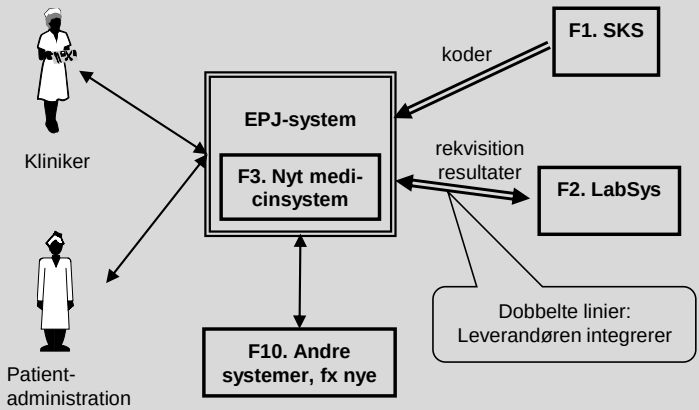
I dette kapitel betyder "kunden" kundens IT-afdeling eller tredjepart bemyndiget af ham.

S-data (Systemets data) er de integrerede data der befinder sig i EPJ-systemet, S.
E-data (Eksterne data) er de integrerede data der befinder sig i det eksterne system, E.

Her er en kort forklaring af de eksterne systemer:

F1. SKS: Sundhedsstyrelsens klassifikationssystem. Opdateres regelmæssigt af Sundhedsstyrelsen.
F2. LabSys: Kundens eksisterende laboratoriesystem til …
F3. Et medicinsystem. Det kan være kundens nuværende medicinsystem eller et nyt leveret som en del af EPJ-systemet.
F10. Et eksternt system som kunden senere vil indkøbe og integrere.

Figur 8. Kontekstdiagram



Kravnote: Svartider
De svartider der specificeres i dette kapitel skal fortolkes på samme måde som i L1, dvs. med L1's fraktil, målt i spidsbelastningsperioder, etc.

Integrationsaspekter
For hver integration er der mange aspekter der skal overvejes:

A. Adgangsret til data.
B. Sikring mod tab af data.
C. Rettigheder og midler til at integrere systemet med andre systemer eller migrere data.

D. Integrationsansvar, fx leverandøren, eller kunden med støtte fra leverandøren.
E. Task som integrationen skal støtte.
F. Data-import fra E (det eksterne system). Hvilket data skal importeres.
G. Data-aktualitet (hvor gammel må den lokale kopi af data være).
H. Svartider ved import.
I. Data-eksport til E. Hvilket data skal eksporteres.
J. Svartider ved eksport.
K. Andre funktioner, fx advarsler til bruger eller til E.

Af praktiske grunde er kravene under punkt A, B og C opstillet som fælles integrationskrav, dvs. at de gælder alle integrationer hvor det er relevant.

Hvad skal kravene sige om de eksterne systemer som leverandøren skal integrere med? Systemerne findes og leverandøren har brug for viden om deres tekniske grænseflader (API'er eller XML services) for at kunne vurdere sine egne integrationsomkostninger. Men kunden har sjældent disse oplysninger.

Kunden må henvise til leverandøren af det eksterne system, men ofte er han ikke villig til at hjælpe. Han ser den nye leverandør som en konkurrent. Kunden bør derfor sikre at den gamle leverandør vil hjælpe, fx ved at frikøbe de nødvendige rettigheder. Dette er en vigtig forudsætning for leverandøren. Den skal stå over kravtabellerne, sammen med de andre forudsætninger leverandøren kan gøre.

For at undgå at den nye leverandør senere skaber lignende problemer, skal han opfylde krav F0-5 til 9. Så behøver kunden ikke forhandle med ham næste gang noget skal integreres.

SOA eller datareplikering?

Nogle kunder lytter til IT-guruerne og beder om en service-orienteret arkitektur (SOA), hvor systemerne er forbundet med XML-services og data kun opbevares i kildesystemet. Andre systemer skal hente det der. I princippet er det en god idé, men kunden gør sig ikke klart at det kræver 10-50 gange mere computer-tid end traditionelle metoder. Det gør det også umuligt for leverandøren at love hurtige svar og høj tilgængelighed, fordi hans system afhænger af andre systemers svartid og tilgængelighed.

Når leverandøren tilbyder et standard/ramme-system (COTS-baseret), kan SOA blive meget dyrt også af andre grunde. Standard/ramme-systemet henter data fra sin egen database, men nu skal data hentes med SOA fra et andet system. Leverandøren skal ændre sit system hundredvis af steder - selv hvis det er pænt opbygget med en fler-niveau arkitektur. Et system der er ændret så mange steder, kan ikke vedligeholdes som en del af standard/ramme-systemet. Så vedligehold bliver også meget dyrt.

En alternativ løsning er at replikere data mellem systemerne og synkronisere data periodisk (batchvis overførsel). Det er som regel meget enklere at tilføje til et standard/ramme-system.

F0. Fælles integrationskrav

Dette afsnit indeholder krav der gælder for alle integrationerne hvor det er relevant, med mindre andet er angivet.

F0-1 kræver at data kun kan overføres til brugerens PC, hvis han har ret til at se dem. Så fortroligheden afhænger ikke af at kun specielle PC-programmer viser de tilladte dele af det data der er overført. Det ville være for let at installere et spionprogram der lader brugeren kigge i de forbudte data. Dette krav kunne også betragtes som et sikkerhedskrav og placeres i afsnit H1.

F0-2 til 3 kræver at systemet skal beskytte mod tekniske problemer med tabte eller dublerede data. Dette kunne også betragtes som et sikkerhedskrav og placeres i afsnit H3.

F0-4 anerkender at det kan blive nødvendigt at undersøge de faktiske dataoverførsler, og beder om støtte til det.

F0. Fælles integrationskrav

Kravene i dette afsnit gælder alle integrationer hvor det er relevant.

A. Adgangsret til data: Kan flyttes til H1	Eksempel på løsning:	Kode:
1. Systemet må kun importere E-data til PC'er og mobile enheder hvor brugeren har ret til at se data i henhold til H1.		

B. Sikring af data: Kan flyttes til H3	Eksempel på løsning:	Kode:
2. Systemet skal beskytte mod tab eller dublering af data ved overførsel mellem to systemer, fx i forbindelse med at ét eller begge systemer har været off-line eller går ned.		
3. Systemet skal beskytte mod udeleligheds-problemer, fx at bruger A ser og dernæst opdaterer E-data, mens bruger B gør det samme. Hverken bruger A eller B vil opdage konflikten.		
4. Systemet skal hjælpe med at finde fejl ved data-overførsler.	Logning af alle fejl ved overførslerne.	

C. Dokumentation og rettigheder:	Eksempel på løsning:	Kode:
5. Det skal være let at tilføje nye grænseflader, fx SOA services, database queries eller API'er.	Systemet tilbyder en OData grænseflade så klienten selv kan definere services. Eller: Leverandøren kan gøre det til en fast pris pr. service.	
6. Kunden skal have mulighed for og ret til at udtrække og bruge alt data beskrevet i kapitel D, fx for at konvertere det til et andet system.		
7. Den tekniske grænseflade til S skal dokumenteres. Dokumentationen skal kunne forstås af et typisk tredjeparts software-hus og findes egnet til integrationsformål og udtræk.	Et kursus på ___ dage er nødvendigt for at kunne anvende dokumentationen og udføre integrationen. Et uddrag af dokumentationen skal leveres tidligt (se B3-4).	
8. Kunden skal have ret til at bruge dokumentationen og selve grænsefladerne.		
9. Leverandøren skal loyalt støtte kunden i disse integrationer og konverteringer med kvalificerede medarbejdere til en fair pris.		

F0-5 kræver at det skal være let at tilføje nye tekniske grænseflader til systemet, fx SOA-services. Selvom nogle kunder tror de kan definere de nødvendige services i kravene, viser erfaringen at der alligevel bliver brug for nye services. Skal man bruge en ny service, er det meget dyrt fordi leverandørerne af de to systemer skal blive enige og afprøve deres dele sammen. Et alternativ er at bruge en OData grænseflade (Open Data Protocol) hvor den der skal bruge data, i stor udstrækning selv kan definere hvad han vil udtrække (som en slags SQL-sætning).

F0-6 kræver at kunden (eller tredjepart) skal kunne migrere (overføre) alt data til et andet system. Dette er et afgørende krav for at kunne skifte leverandør senere. Forbavsende mange kunder glemmer det, og derfor får leverandøren et monopol.

F0-7 til 9 kræver at kunden skal kunne integrere systemet med andre systemer. Han skal have værktøjer, dokumentation og rettigheder til at gøre det, og leverandøren er forpligtet til at støtte ham. Hvis alle eksterne systemer havde opfyldt lignende krav, ville integration være meget lettere.

Bemærk hvordan det er muligt at verificere dokumentationens kvalitet ved at bede et typisk tredjeparts software-hus om at afprøve dokumentationen. Dette bør gøres tidligt for at sandsynliggøre at leverandørens måde at dokumentere på vil være tilstrækkelig for at tredjepart kan integrere med EPJ-systemet (se afsnit B3).

F1. Simpel envejs integration (SKS)

Dette afsnit er et eksempel på en meget løs integration med et eksisterende system, Sundhedsstyrelsens klassifikationssystem, SKS. SKS består af nogle tabeller som enhver kan downloade.

Indledningen over tabellerne giver forudsætninger for kravene, på samme måde som forudsætningerne for task-beskrivelserne.

Task: Hvilke task bruger integrationen?
E-support: Hvem har rettigheder til at integrere? Hvor får man dokumentationen af de eksterne grænseflader? Hvem kan give support?
E-opdateringer: Hvor hyppigt bliver SKS-koder opdateret inde i SKS-systemet?
Datavolumen: Hvor meget data skal overføres?

Der er vist to udgaver af kravtabellen for F1. En hvor man omhyggeligt overvejer alle punkterne D til K, og en hvor man kun skriver det strengt nødvendige.

Alle punkter overvejet

F1-1 kræver at leverandøren integrerer med det eksterne system. Det antages at der ikke er brug for støtte (en rimelig antagelse i dette tilfælde).

Der er ingen særlige krav om task-støtte. Indledningen siger at data bliver brugt i de fleste task. Det er tilstrækkeligt i dette tilfælde.

F1-2 angiver hvilke data der skal hentes fra SKS.

F1-3 viser at data-aktualiteten ikke er vigtig. Hvis systemet har nye data en uge efter at de er frigivet af Sundhedsstyrelsen, er alt i orden. Løsningseksemplet nævner at en periodisk overførsel er nok. En anden løsning er at IT-afdelingen manuelt starter overførslen når Sundhedsstyrelsen annoncerer ændringerne.

F1-3p nævner et eksisterende problem med konflikter mellem lokale koder og nye officielle koder og foreslår to løsninger.

F1-4 nævner at der undertiden er brug for bedre data-aktualitet.

Der er ingen krav om en bestemt svartid (tid for dataoverførslen). Der er heller ingen krav om at overføre data til SKS eller bruge særlige funktioner.

Den korte udgave

Her kan vi nøjes med to krav: (1) Leverandøren har ansvaret. (2) De nye SKS-tabeller skal bruges af systemet kort tid efter at de er frigivet.

F1. SKS

E-data (eksterne data)**:** SKS-tabellerne omfatter koder og tilsvarende navne for diagnoser, ydelser, behandlingsafdelinger, mv.

Task: Koder og navne bruges i de fleste task. Dog hentes oplysningerne om behandlingsafdelinger fra et andet system.

E-support: Tabellerne er offentligt tilgængelige fra Sundhedsstyrelsens web-site. Det er zip-tekstfiler med fast feltlængde. De er nærmere beskrevet på web-sitet.

E-opdateringer: Afdelingsoplysningerne opdateres månedligt, de andre oplysninger ca. hver tredje måned.

Data-volumen: SKS-tabellerne består af ca. 100.000 rækker, hver på omkring 100 tegn.

Alternativ 1: Alle punkter overvejet

D. Integrationsansvar:	Eksempel på løsning:	Kode:
1. Leverandøren skal integrere med de relevante SKS-tabeller.		

E. Task der skal støttes: Ingen specielle.	Eksempel på løsning:	Kode:

F. Data-import:	Eksempel på løsning:	Kode:
2. Alle koder og navne skal overføres undtagen afdelingsoplysningerne.		

G. Data-aktualitet:	Eksempel på løsning:	Kode:
3. S-data bør ikke være ældre end en uge.	Systemet importerer E-data hver __ dag. Eller: Driftsafdelingen starter overførslen når Sundhedsstyrelsen annoncerer at data er tilgængelige.	
3p. Det sker at nye SKS-data er i konflikt med lokalt definerede koder.	Driftsafdelingen kan rulle SKS-data tilbage til forrige version. Eller: Lokale koder har et mærke så de ikke giver konflikt.	
4. I særlige tilfælde kan der være behov for bedre data-aktualitet.	Driftsafdelingen kan starte en overførsel.	

H. Svartider ved import: Ingen krav.	Eksempel på løsning:	Kode:

I. Data-eksport: Ingen.	Eksempel på løsning:	Kode:

J. Svartider ved eksport: Ikke aktuelt.	Eksempel på løsning:	Kode:

K. Andre funktioner: Ingen krav.	Eksempel på løsning:	Kode:

Alternativ 2: Den korte udgave

Integrationskrav:	Eksempel på løsning:	Kode:
1. Leverandøren skal integrere med de relevante SKS-tabeller.		
2. Systemet skal anvende de nye tabeller kort tid efter at de er frigivet.	System eller supportmedarbejdere overfører data senest en uge efter at de er frigivet.	

F2. Tovejs integration (LabSys)

Dette afsnit er et eksempel på en tæt integration med et eksisterende system. Data overføres begge veje: rekvisitioner sendes til LabSys og laboratoriesvar sendes den anden vej. Indledningen forklarer hvad LabSys gør fra et brugersynspunkt.

Task: Kun task C10 bruger LabSys.
E-support: Kunden henviser til et teknisk dokument og lover at et bestemt firma, MediData, kan yde support. Kunden har aftalt de nødvendige rettigheder med MediData.
E-opdateringer: En opdatering svarer til at LabSys genererer et svar.
S-opdateringer: S er EPJ-systemet. S-data er rekvisitionerne. En opdatering svarer til at sende en rekvisition.
Datavolumen: Et svar består af et eller flere resultater, hver på ca. 500 tegn.

F2-1 kræver at EPJ-leverandøren integrerer med det eksterne system. Han kan forudsætte support fra MediData, som lovet under E-support.

F2-2 kræver effektiv støtte til task C10. Dette krav kan se overflødigt ud fordi indledningen også nævner C10. Men når vi skriver det som et krav, er det lettere at huske at vurdere løsningen. Det giver også leverandøren mulighed for at forklare hvad han anser for en god løsning.

F2-3 angiver hvilke data der skal hentes fra LabSys. Data svarer til Ydelses-data i datamodellen (afsnit D3).

F2-4 og 5 kræver at LabSys-resultater skal være i EPJ-systemet (S) inden 3 timer, men at der undertiden er behov for bedre data-aktualitet. Kunden nævner et par løsninger, men de forudsætter at leverandøren kan aftale dette med MediData, da elektronisk data i dag kun overføres om natten.

F2-6 angiver svartiden for data-import (hente svaret). Eksemplet giver LabSys tid til at sende svaret. Generelt har en leverandør problemer med at opfylde svartids-krav der inkluderer det eksterne systems svartid. Så et fair krav giver det eksterne system den tid det har brug for.

Som nævnt i kravnoten i begyndelsen af kapitel F, skal svartider forstås ligesom i afsnit L1, dvs. med fraktiler og spidsbelastning.

F2-7 kræver at brugeren kan sende LabSys rekvisitioner ved hjælp af EPJ-systemet (S). Det betragtes som en slags data-eksport, men kunne også betragtes som en funktion og specificeres under "andre funktioner" (F2-K).

F2-8 specificerer svartider for data-eksport (sende en rekvisition). Der er to tider i spil: Tiden indtil brugeren kan fortsætte med at taste eller klikke, og tiden til brugeren kan se kvitteringen fra LabSys.

F2-9 og 10 kræver at EPJ-systemet adviserer sine egne brugere og LabSys om nye og udeblevne svar.

F2. LabSys

E-data (eksterne data**):** LabSys version yyy. Brugere kan rekvirere laboratoriesvar fra LabSys. Selve prøven leveres som … og svaret kommer både på fax og elektronisk. Hvert svar kan omfatte mange resultater.

Task: LabSys bruges ved task C10, Udfør klinisk session.

E-support: Den tekniske grænseflade til LabSys er beskrevet i … MediData supporter LabSys i Danmark og kan også levere bistand til integrationen. Kunden har aftalt rettighederne med MediData.

E-opdateringer: LabSys leverer svar løbende via fax, men i øjeblikket leveres de elektroniske svar kun som batch i løbet af natten.

S-opdateringer: Hele hospitalet rekvirerer omkring 8000 svar om dagen, hovedsagelig mellem 8:00 og 16:30.

Data-volumen: Hvert svar består af et eller flere resultater, hver på ca. 500 tegn.

D. Integrationsansvar:	Eksempel på løsning:	Kode:
1. Leverandøren skal integrere med LabSys.		

E. Task der skal støttes:	Eksempel på løsning:	Kode:
2. Integrationen skal give effektiv støtte til C10.	Rekvisitioner og svar ser for brugeren ud som andre ydelser - uden at indtaste patientnummer igen.	

F. Data-import:	Eksempel på løsning:	Kode:
3. Alt E-data der svarer til data i afsnit D3.		

G. Data-aktualitet:	Eksempel på løsning:	Kode:
4. S-data bør ikke være ældre end 3 timer.	Systemet importerer E-data hver __ time. Eller: Data importeres på E's initiativ når de er tilgængelige. Eller: Data hentes altid fra E.	
5. Undertiden er der behov for de seneste resultater for en specifik patient, fx under en operation.	Systemet kan hente data på brugerens initiativ. Eller: Data hentes altid fra E.	

H. Svartider ved import:	Eksempel på løsning:	Kode:
6. Når brugeren rekvirerer et svar, skal det ske så hurtigt at brugeren ikke taber tålmodigheden.	Resultatet er synligt inden __ s plus den tid LabSys bruger på at sende svaret. (Kunden forventer 3 s).	

I. Data-eksport:	Eksempel på løsning:	Kode:
7. Brugeren kan sende LabSys rekvisitioner gennem EPJ-systemet (S).		

J. Svartider ved eksport:	Eksempel på løsning:	Kode:
8. En rekvisition kan afsendes og brugeren fortsætte med at taste inden for den mentale skiftetid (ca. 1.3 s). Kvitteringen fra LabSys bør vises kort tid efter.	Tastning kan ske inden __ s. (Kunden forventer 1.3 s). Kvitteringen fra LabSys kan ses __ s efter at LabSys har sendt den. (Kunden forventer 3 s).	

K. Andre funktioner:	Eksempel på løsning:	Kode:
9. S kan advisere brugeren om nye eller udeblevne LabSys-svar.		
10. S kan advisere Labsys (E) om udeblevne LabSys-svar (rykkere).		

F10. Integration med nye eksterne systemer

Når man først har anskaffet systemet, kan det blive meget dyrt at få det integreret med nye eksterne systemer. Leverandøren vil nemlig normalt have monopol på at udføre den slags ændringer. Afsnit F0 (krav 5-9) forhindrer et monopol ved at kræve at kunden (eller tredjepart) kan udføre sådanne integrationer. I afsnit F10 prøver kunden at få oplysninger om hvilke slags integrationer han selv kan udføre.

E-support siger at det (selvfølgelig) er kundens ansvar at skaffe dokumentationen for det eksterne system.

F10-1 siger at kunden (eller tredjepart) er ansvarlig for integrationen, men leverandøren skal støtte ham som krævet i F0-9.

F10-2 siger at EPJ-systemet (S) skal muliggøre at det nye eksterne system kan bruges offline i en periode og synkronisere data når systemerne igen forbindes. (Som for andre krav kan leverandørens "løsning" være at han ikke støtter det).

F10-3 til 6 specificerer hvordan EPJ systemet skal kunne importere data fra det eksterne system: Overfør på S eller E's initiativ, overfør periodisk, overfør data der er nyere end et bestemt tidspunkt, eller overfør data om en bestemt patient. Kunden forestiller sig at han kan konfigurere EPJ-systemet til at gøre den slags ting.

F10-7 specificerer svartider, men det er ikke muligt for en EPJ-leverandør at love noget om svartider, med mindre han kender belastningen af EPJ-systemet og arten af dataoverførsler. Hvis kunden overfører mange data til/fra EPJ-systemet, kan det blive overbelastet og svare langsomt.

Hvordan kan vi angive et fair krav om dette? En måde er at bede om yderligere kapacitet til fremtidige dataoverførsler. Det er hvad F10-7 beder om.

F10-7p nævner et kendt problem med den slags integrationer: En usædvanlig lang dataoverførsel kan blokere systemet og de almindelige korte overførsler.

F10-8 til 11 svarer til F10-3 til 6, men handler om dataeksport til det eksterne system.

F10-12 og 13 beder om en liste af funktioner som EPJ-systemet kan bruge i et eksternt system og en liste af de funktioner det selv stiller til rådighed.

F10. Integration med nye systemer

Kunden forventer at kunne integrere nye eksterne systemer med S - med lidt eller ingen hjælp fra leverandøren af S. Dette afsnit lister de behov sådanne integrationer kunne have og beder om leverandørens forslag til at opfylde dem.

Eksternt system: I princippet et hvilket som helst system. Eksempler: Røntgen-system, mobile anvendelser, specialsystem for intensivafdeling.
Task: Defineres senere.
E-support: Kundens ansvar.
E-opdateringer: Defineres senere.
Data-volumen: Defineres senere.

D. Integrationsansvar:	Eksempel på løsning:	Kode:
1. Kunden udfører integrationen. Leverandøren skal støtte som beskrevet i F0-9.		

E. Task der skal støttes:	Eksempel på løsning:	Kode:
2. For mobile anvendelser kan E i perioder være off-line. Når E igen forbindes med S, skal data synkroniseres.	Kunden kan konfigurere S til automatisk at synkronisere data når systemerne igen forbindes.	

F+G. Data-import og dataaktualitet:	Eksempel på løsning:	Kode:
3. S kan importere data fra E, forudsat at de passer til S's eksisterende datatabeller.	Kunden kan konfigurere S til at importere på S's initiativ eller E's.	
4. S kan periodisk importere data fra E.	Kunden kan konfigurere S til at gøre det.	
5. S kan optimere importen ved kun at bede om data der er yngre end et bestemt tidspunkt.	Kunden kan konfigurere S til at gøre det.	
6. S kan optimere importen ved kun at bede om data for en bestemt patient.		

H+J. Svartider ved import og eksport:	Eksempel på løsning:	Kode:
7. S kan skalere op til en væsentlig højere belastning end angivet i L1. Kunden kan bruge denne ekstra belastning til dataoverførsler.	Systemet kan skalere op til at håndtere en belastning på __ gange så meget som krævet i L1. (Kunden forventer 2 gange).	
7p. Når en lang overførsel er i gang, kan den blokere for kortere overførsler så de giver en meget lang svartid.	Systemet kan håndtere mange parallelle overførsler.	

I. Data-eksport:	Eksempel på løsning:	Kode:
8. S kan eksportere data til E når det er data der findes i S's tabeller.	Kunden kan konfigurere S til at eksportere på S's initiativ eller E's.	
9. S kan periodisk eksportere data til E.	Kunden kan konfigurere S til at gøre det.	
10. S kan optimere eksporten ved kun at sende data der er yngre end et bestemt tidspunkt.	Kunden kan konfigurere S til at gøre det.	
11. S kan optimere eksporten ved kun at sende data for en bestemt patient.		

K. Andre funktioner:	Eksempel på løsning:	Kode:
12. S kan bruge funktioner i E, fx rekvirere ydelser eller advisere om udeblevne svar.	Leverandøren bedes specificere de funktioner S vil kunne bruge.	
13. E kan bruge funktioner i S, fx advisere brugere eller printe på printere der styres af S.	Leverandøren bedes specificere de funktioner S stiller til rådighed.	

G. Teknisk it-arkitektur

Ordet *it-arkitektur* har efterhånden fået to helt forskellige betydninger. Den klassiske er konfigurationen af hardware, software, datatransmission, mv. Det er den *tekniske it-arkitektur.* Den nye betydning omfatter både den tekniske arkitektur, datamodel, brugervenlighed, drift, support, mv. Kort sagt det meste af hvad denne kravskabelon allerede handler om.

Krav til den tekniske it-arkitektur afhænger af situationen. Har kunden allerede udstyr han vil bruge? Eller vil han købe det? Eller overlader han det til leverandøren fordi leverandøren alligevel skal drive systemet?

Skabelonen viser et eksempel for hver af disse tre situationer. Vælg den der passer bedst til din egen situation, tilpas den efter behov og fjern de to andre situationer.

G1. Eksisterende hardware og software

Dette afsnit beskriver kundens eksisterende udstyr. Det forklarer at andre anvendelser kan køre på udstyret samtidig, men at leverandøren kan forudsætte at der er en vis mængde ressourcer igen til det nye system. Bemærk at de frie ressourcer er til rådighed i enhver periode på et sekund. Uden denne forudsætning kan leverandøren ikke garantere svartider omkring 1 sekund.

Leverandøren har brug for disse oplysninger for at vurdere om hans system har brug for flere ressourcer.

G1-1 beder leverandøren angive hvor mange brugere systemet kan betjene med det eksisterende udstyr. "Betjene" betyder at svartider, tilgængelighed og lagerkrav overholder kravene i kapitel L.

G1-2 beder leverandøren angive hvor meget ekstra udstyr der er behov for til at dække den fulde belastning i L1.

Ofte udføres nogle dele af et system i en internet browser, fx dele beregnet til offentligheden. G1-3 kræver at disse dele kan køre på gængse browsere, og nævner i løsningssiden de browsere kunden tænker på.

Mange IT-guruer reklamerer med at alt skal være web-baseret, for så kan det bruges alle steder. Det er desværre ikke korrekt. Simple web-sider, ja, men så snart det bliver mere komplekst, er det browser-afhængigt. Sådan noget som genvejstaster, kobling til databaser eller sikkerhedsindstillinger varierer fra browser til browser. Så i praksis må leverandøren indbygge test i programmet der undersøger om tingene skal gøres på den ene eller anden måde. Og det skal testes på alle browserne - også når der kommer en ny release af browseren.

G2. Nyt hardware og software

Dette afsnit beder leverandøren angive hvilket udstyr kunden skal anskaffe, og hvor meget ekstra der skal anskaffes når der bliver dobbelt så stor belastning.

G2-3 angiver at der kun bør bruges udstyr fra kundens liste af favoritter. Det kan være vigtigt hvis kunden har ekspertise med netop dette udstyr eller har en købsaftale med bestemte leverandører.

Også her er der brug for krav til browser støtten.

G3. Leverandøren har driftsansvar

Dette afsnit angiver at da leverandøren skal drive systemet, bestemmer han også hvilket udstyr der skal bruges.

Også her er der brug for krav til browser støtten.

G. Teknisk it-arkitektur

G1. Eksisterende hardware og software Alternativ 1: Brug hvad vi har

Kunden har i øjeblikket følgende it-udstyr der forventes brugt til drift af det nye system:

1. 20 servere af typen …
2. 300 PC'er med Windows XP og mindst 100 GB disk.
3. Optisk fibernet …
4. Oracle database …

Udstyret bliver brugt til andre opgaver på samme tid men inden for disse grænser:

5. Indenfor enhver 1 sekunds periode er 50% af servernes hastighedskapacitet til rådighed for EPJ systemet.
6. Indenfor enhver 1 sekunds periode er 50% af fibernettets kapacitet til rådighed for EPJ systemet.
7. Der kører ikke andre opgaver på PC'erne når de kører EPJ-systemet.

Krav til platformen:	Eksempel på løsning:	Kode:
1. Systemet skal i starten køre på det eksisterende udstyr og overholde kravene i L1, L2 og L3 for et begrænset antal brugere.	Systemet kan under disse omstændigheder trække ___ brugere. Kunden forventer 20 brugere.	
2. For at klare den fulde spidsbelastning (se L1) skal systemet udbygges for at overholde kravene i L1, L2 og L3.	Kunden skal udbygge med dette udstyr ____.	
3. De browser-baserede dele skal kunne køre på gængse browsere.	MS Internet Explorer, Chrome, Safari.	

G2. Nyt hardware og software Alternativ 2: Leverandøren foreslår

Kunden har tænkt sig at anskaffe nyt udstyr for at drive systemet.

Krav til platformen:	Eksempel på løsning:	Kode:
1. Kunden skal bruge nyt udstyr for at systemet kan overholde kravene i L1, L2 og L3.	Kunden skal bruge dette udstyr _____.	
2. Når spidsbelastningen fordobles i forhold til L1, skal systemet udbygges for at overholde kravene i L1, L2 og L3.	Kunden skal udbygges med dette udstyr ____.	
3. Der skal så vidt muligt kun bruges hardware og software fra positivlisterne i bilag x.		
4. De browser-baserede dele skal kunne køre på gængse browsere.	MS Internet Explorer, Chrome, Safari.	

G3. Leverandøren har driftsansvar Alternativ 3: Leverandørens problem

Krav til platformen:	Eksempel på løsning:	Kode:
1. Leverandøren driver systemet og bruger det nødvendige udstyr for at overholde kravene i L1, L2 og L3.		
2. De browser-baserede dele skal kunne køre på gængse browsere.	MS Internet Explorer, Chrome, Safari.	

H. Sikkerhed

Formålet med sikkerhedskrav er at beskytte disse 4 sikkerhedsfaktorer: (1) fortrolighed af data, (2) korrekthed af data, (3) tilgængelighed af data og funktionalitet, (4) ægthed (at brugerne er dem de udgiver sig for).

På engelsk kaldes faktorerne CIA+A: Confidentiality, Integrity, Availability, Authenticity.

H1. Log-in og adgangsret for brugere

Dette afsnit beskriver de situationer hvor brugerens adgangsrettigheder skal kontrolleres. Systemet skal beskytte fortrolighed, korrekthed og ægthed. Kravene er udtrykt som subtask der skal støttes og problemer der skal undgås. Skabelonen viser to alternativer: (1) Det nye system skal gøre som vore andre systemer. (2) Det nye system skal give bedre eller mere bekvem sikkerhed.

Alternativ 1: Log-in som i dag

H1-1 angiver at brugeren skal identificeres med den eksisterende metode og hvad denne metode er.

H1-2 siger at der kun skal gives adgang til brugere der har de nødvendige rettigheder. Løsningseksemplet viser to måder at gøre det på.

Alternativ 2: Kunden ønsker bedre sikkerhed

H1-1 siger også her at brugeren skal identificeres. Løsningseksemplet foreslår den traditionelle metode, men også en alternativ.

Kravet siger intet om længden af password. Længden giver beskyttelse mod hacker-angreb og behandles i afsnit H5.

H1-2 beder om støtte til den situation hvor bruger 1 har været væk fra systemet et stykke tid og en anden bruger anvender systemet med bruger 1's rettigheder. Den traditionelle løsning er timeout, men det giver problemer der kræver støtte.

H1-3 siger at brugerens rettigheder skal kontrolleres og nævner de kendte problemer med at have et password for hvert system. Der nævnes en løsning: Single sign-on. (Dette er kun en del af løsningen, fordi kundens andre systemer skal ændres så de bruger samme metode. Dette er ikke EPJ-leverandørens ansvar).

H1-4 nævner en trussel der skal beskyttes mod, fx ved at skifte password.

Mulige rettigheder og granularitet

Både for alternativ 1 og 2 er det vigtigt at angive de mulige adgangsrettigheder. De står i en kravnote under kravtabellen. I EPJ-systemet er der forskellige rettigheder for at ordinere medicin og for at se patientdata. Et vigtigt punkt er granulariteten af rettigheden, dvs. hvor små "områder" skal der være rettigheder for. Får brugeren ret til at ordinere medicin generelt eller kun for en bestemt afdeling? I eksemplet er granulariteten en afdeling og i nogle tilfælde en patient. Bemærk at en person kan have flere rettigheder.

Mange kunder ignorerer listen af rettigheder, selvom den er vigtig for leverandørens vurdering af løsningens kompleksitet. Teknisk set er det let at tildele rettigheder til en bruger, men at tjekke rettighederne med den rigtige granularitet er ofte komplekst og skal håndteres dybt nede i systemet.

H. Sikkerhed

H1. Log-in og adgangsret for brugere

Log-in, mv. er ikke selvstændige task for brugerne, men subtask der optræder i mange forskellige task. Systemet skal støtte følgende subtask i forbindelse med brugernes adgang til systemet.

Alternativ 1: Log-in som i dag

Subtask vedr. brugeres adgang:	Eksempel på løsning:	Kode:
1. Identificér brugeren med den eksisterende brugeridentifikation, login metode og timeout-metode som er LDAP og LA …		
2. Kontrollér at kun autoriserede brugere har adgang til system og data. (Se kravnoten nedenfor).	Database-systemet kontrollerer rettighederne. Eller: Brugerens skærmbilleder viser kun de funktioner og data han har ret til at bruge.	

Alternativ 2: Kunden ønsker bedre og mere bekvem sikkerhed

Subtask vedr. brugeres adgang:	Eksempel på løsning:	Kode:
1. Identificér brugeren. (Se afsnit H5-3 om længden af password).	Brugeren identificerer sig selv med brugernavn og password. Gerne også en alternativ identifikation, fx stemme-genkendelse eller fingeraftryk.	
2. Brugeren har været væk fra systemet i et stykke tid.		
2p. Problem: En anden bruger kan anvende systemet med den førstes rettigheder.	Systemet timer ud efter 10 min.	
2q. Problem: Hvis systemet selv logger ud, er det besværligt at logge på igen.	Systemet kræver kun password. Timeout perioden er sted-afhængig, fx en lang timeout på operationsstuen.	
2r. Problem: Hvis systemet selv logger ud, kan indtastet data gå tabt.		
3. Kontrollér at kun autoriserede brugere har adgang til system og data. (Se kravnoten nedenfor).	Database-systemet kontrollerer rettighederne. Eller: Brugerens skærmbilleder viser kun de funktioner og data han har ret til at bruge.	
3p. Problem: I dag har brugerne password for hvert system. Det er tungt at skifte mellem systemerne og at skifte password regelmæssigt. Derfor opstår der vaner med at anbringe passwords hvor alle kan se dem.	Hver bruger har kun ét brugernavn og ét password (single sign-on).	
4. Stjålne password handles ofte af kriminelle. Begræns mulighederne.	Brugerne skal regelmæssigt skifte password. Når et brud er konstateret, skal alle password hurtigt kunne blokeres.	

Kravnote: Mulige rettigheder

1. Ret til at ordinere medicin på afdeling M.
2. Ret til at se patientdata på afdeling M.
3. Ret til at registrere klinisk data (ydelser og diagnoser) på afdeling M.
4. Ret til at se data med patientens tilladelse (se H5-3).

…

En læge på afdeling M kan fx have rettighederne 1, 2 og 3, mens en tilsynsførende læge på afdeling M kun har rettighed 2 og 3.

H2. Sikkerhedsadministration

Sikkerhedsadministrationen tildeler og fjerner brugerrettigheder, definerer nye roller, osv. En organisation kan have en central sikkerhedsadministration eller delegere det til afdelinger. Det angives som en forudsætning lige inden tabellen.

Skabelonen beskriver sikkerhedsadministrationen som nogle subtask der skal støttes og problemer der skal fjernes.

Alternativ 1 forventer at sikkerhedsreglerne håndteres af kundens eksisterende sikkerhedsadministration. EPJ-systemet skal blot spørge i det eksisterende system for at kontrollere brugeres password og deres rettigheder.

Alternativ 2 forventer at det nye system har funktionalitet til at oprette brugere, ændre rettigheder, mv.

Et af problemerne er at tildele rettigheder til et stort antal brugere når de starter i begyndelsen af måneden.

Nogle af løsningerne er velkendte teknikker, fx rollebaseret tildeling af rettigheder og tidsbegrænsede rettigheder. De er eksempler på løsninger og ikke krav.

H2. Sikkerhedsadministration

Hver afdeling har sin egen sikkerhedsadministration.
Eller: Sikkerhedsadministrationen er central for hele hospitalet.
Sikkerhedsadministrationens arbejde omfatter nedenstående subtask.

Alternativ 1: Benyt den eksisterende sikkerhedsadministration

Kunden bruger i dag LDAP og AD og ønsker at administrere alle rettigheder på den måde.

Subtask vedr. sikkerhedsadministration:	Eksempel på løsning:	Kode:
1. Opret og fjern brugere.	Overlad det til den eksisterende sikkerhedsadministration.	
2. Tildel eller fjern rettigheder for en bruger.	Overlad det til den eksisterende sikkerhedsadministration.	
3. Kontroller at brugeren har de fornødne rettigheder. (Egentlig et subtask under H1).	Systemet henter rettighedsoplysninger i kundens eksisterende system.	

Alternativ 2: Det nye system har egen sikkerhedsadministration

Subtask vedr. sikkerhedsadministration:	Eksempel på løsning:	Kode:
1. Tildel eller fjern rettigheder for en bruger.		
1a. Opret brugeren først.		
1p. Problem: En lang række brugere skal oprettes når de tiltræder ved månedsstart.	Systemet overfører data fra personalesystemet hver måned.	
1q. Problem: En hasteindkaldt vikar er endnu ikke kommet ind i personalesystemet, men skal have adgangsret.	Mulighed for midlertidig oprettelse på afdelingen udenom sikkerhedsadministrationen.	
1r. Problem: Sikkerhedsadministrationen skal holde styr på sammenhængen mellem 4000 brugere og 300 rettigheder.	Hver bruger tildeles en eller flere roller, fx læge i afdeling M og tilsynsførende i afdeling N. Hver rolle har en eller flere rettigheder, fx ordination og diagnostik.	
1s. Problem: Sikkerhedsadministrationen glemmer at oprette og fjerne rettigheder på de rigtige tidspunkter, fx i forbindelse med ansættelse og fratræden.	Rettigheder og roller kan defineres i god tid så de gælder for en periode, fx fra den dag personen ansættes.	
2. Opret nye roller med egne kombinationer af rettigheder.		
3. Få oversigt over hvem der har ret til hvad og om der fx er en rettighed som ingen har.		

H3. Sikring mod tab af data

Skabelonen nævner nogle typiske risici for tab af data, og leverandøren bedes angive sin løsning. For disknedbrud og brand nævner skabelonen de traditionelle løsninger. F02 og F03 nævner risici for at tabe data under systemintegration.

Disse krav beskytter korrekthed og tilgængelighed af data.

Med hjælp fra en sikkerhedsekspert, kan kunden bede om beskyttelse mod mange andre kilder til datatab. Skabelonen viser et eksempel hvor leverandøren lod en underleverandør køre systemet (i "skyen"). Underleverandøren opbevarede ikke data forsvarligt. Han lagde fx backup-versionen at databasen på samme disk som selve databasen. Den dag disken brød sammen, forsvandt både databasen og dens backup kopi.

I skabelonen er denne erfaring behandlet som en trussel på linje med andre trusler. Kunden foreslår et par løsninger.

H4. Sikring mod utilsigtet brugeradfærd

Dette afsnit nævner typiske risici der skyldes at brugere utilsigtet kommer til at gøre noget med uventede resultater.

H4-1 siger at ingen brugerhandling må få systemet til at lukke ned. Det er et underforstået krav til alle systemer, og selv om det ikke er skrevet ned vil det formentlig få medhold i retten. Men skriver man det ned, er der ingen tvivl. Eksemplet nævner en måde kunden kan overbevises om at systemet opfylder kravet.

H4-2 og 3 kræver tjek mod simple fejltagelser og brug af undo ved uventede systemreaktioner.

H4-4 anerkender at ikke alle funktioner er fortryd-bare og beder om forebyggelse mod at de bruges af en fejltagelse.

H4-5 kræver mulighed for at stoppe en funktion der viser sig at tage meget lang tid.

Disse krav beskytter korrekthed af data og for H4-5 også tilgængelighed.

H3. Sikring mod tab af data

Data kan utilsigtet tabes eller fejltydes på flere måder.

Systemet skal beskytte mod:	Eksempel på løsning:	Kode:
1. (Se F0-2 om beskyttelse mod tab eller replikering af data under overførsel mellem to systemer).		
2. (Se F0-3 om beskyttelse mod udelelighedsproblemer med eksterne systemer).		
3. Lokale udelelighedsproblemer, fx at bruger A ordinerer medicin, men inden systemet registrerer beslutningen, har bruger B ordineret en medicin der interagerer. Hverken bruger A eller B vil opdage konflikten.		
4. Disknedbrud	Periodisk backup eller RAID diske.	
5. Brand og sabotage	Backup på steder med mindst 10 km afstand …	
5p. **Problem**. Driftsoperatøren opbevarer ikke data forsvarligt, lægger fx backup data på samme drev som selve databasen. Forekommer især hos underleverandører.	Hovedleverandøren kontrollerer jævnligt at det foregår forsvarligt. Eller: Kunden får ugentlig en backup af alt data til egen opbevaring.	

H4. Sikring mod utilsigtet brugeradfærd

Ved utilsigtet brugeradfærd forstås at brugeren kommer til at gøre noget han ikke havde til hensigt, fx ved at ramme den forkerte tast eller bruge en kommando der har en anden virkning end han troede.

Krav:	Eksempel på løsning:	Kode:
1. Utilsigtede brugerhandlinger må ikke kunne få systemet til at bryde sammen, hverken på klienten eller serveren.	Kan være svært at teste ved leverancen, men leverandørens problemlog og en beskrivelse af leverandørens testmetoder kan hjælpe.	
2. Alt indtastet data skal kontrolleres for format, konsistens og rimelighed. I tvivlstilfælde skal brugeren advares og tage stilling.		
3. Brugeren skal let kunne rette fejltagelser.	Systemet bruger undo i stor udstrækning.	
4. Forebyg fejlagtig brug af funktioner der ikke kan fortrydes.	Anbring knappen så den ikke bliver ramt ved et tilfælde - eller bed om bekræftelse.	
5. Brugeren skal kunne afbryde langvarige funktioner, fx store dataoverførsler, uden at ødelægge dataintegriteten.		

H5. Sikring af personlige data (privacy), fx GDPR

Dette afsnit giver eksempler på krav der beskytter mod misbrug af personlige data. Misbrug af personlige data er en af truslerne mod fortrolighed.

GDPR (General Data Protection Regulation) er EU's regler om beskyttelse af personlige data. EU kræver at alle europæiske virksomheder og offentlige organisationer følger reglerne (med visse undtagelser). EU kan give meget store bøder hvis man bryder reglerne.

Reglerne omfatter organisatoriske forhold, fx at udnævne en ansvarlig for data-beskyttelse (DPO, Data Protection Officer) og at informere om eventuelle brud på fortroligheden. De omfatter også regler der kræver IT-støtte, fx en persons ret til at få slettet alle data om sig selv, og retten til at få at vide hvad ens data bliver brugt til. For små organisationer er det et mareridt at håndtere alt dette.

H5-1 og 2 dækker behovene for små virksomheder der vil have en erfaren leverandør til at hjælpe sig. H5-1 beder om IT-støtten, H5-2 om administrative råd.

Nogle organisationer har specielle regler for beskyttelse af personlige data på deres eget område. Her er nogle af kravene for et hospital.

H5-3 og 4 kræver beskyttelse af patientdata på et hospital. Under behandlingen, kan lægen ønske at se data fra patientens praktiserende læge eller andre hospitalsafdelinger. Systemet skal støtte at lægen kun kan se det, når patienten har givet tilladelse til det.

H5. Fortrolighedskrav

Kunden skal overholde EU's fortrolighedsregler (GDPR, General Data Protection Regulation). Kunden ved kun lidt om hvad det er, men forventer at leverandøren ved det og leverer den nødvendige funktionalitet og rådgivning.

Krav:	Eksempel på løsning:	Kode:
1. Systemet har funktioner der gør kunden i stand til at overholde GDPR, fx på klientens anmodning slette alt data om klienten, sende data om klienten på elektronisk form, og automatisk slette personlige oplysninger når de har tjent deres formål.		
2. Leverandøren vejleder kunden om de nødvendige administrative tiltag.		

Patienter har ret til at bestemme hvem der må se deres diagnoser og andre kliniske data. Det sker under de kliniske sessioner C10 og C20.

Subtask i C10 og C20:	Eksempel på løsning:	Kode:
3. Bed patienten om tilladelse til at se hans kliniske data fra andre organisationer, fx praktiserende læge. Registrer tilladelsen.		
4. Systemet viser kun data som patienten har givet tilladelse til. Se også kravnoten om rettigheder under H1.		

H6. Sikring mod trusler

Dette afsnit vedrører forsætlige trusler med virus, hacking, SQL injection, Denial of Service (DoS), Trojanske heste, mv. De kan true alle sikkerhedsfaktorer (fortrolighed, osv.). For at identificere de vigtigste, bør man lave en risikovurdering.

Under en risikovurdering ser man på de mulige trusler en efter en, vurderer deres hyppighed og konsekvensen hvis de optræder (helst i kroner og øre). Så beregner man den "gennemsnitlige" skade pr. år for hver trussel. På grundlag af det, beder man om beskyttelse mod de mest alvorlige trusler.

I praksis er beskyttelsen mod trusler den svageste del af sikkerhedskravene, og gode risikovurderinger bliver sjældent udført. Både kunde og leverandør tror at det er nok at følge sikkerhedsstandarder (fx som i H6-7).

For at gøre sagen værre, vokser listen af mulige trusler i takt med at angriberne bliver snedigere. At forudsige alle trusler er ligeså umuligt som at forudsige menneskehedens opfindelser. En god leverandør følger udviklingen (se H6-8).

Alternativ 1: Kunden kender risici

Kunden har lavet en risikovurdering og har listet de alvorligste trusler. Så beder han leverandøren om forslag til beskyttelse. Skabelonen viser et par eksempler på trusler.

Man ser ofte sikkerhedskrav der beskriver en løsning i stedet for et behov. Fx ser man krav som dette:

> *Password skal være på mindst 9 tegn med mindst ét stort bogstav.*

Det er besværligt for brugerne, så lad os spørge en sikkerhedsekspert hvorfor det er nødvendigt. *Det er fordi,* siger han, *at hackere prøver alle mulige password med et særligt program. Hvis systemet arbejder for fuld fart med log-in, er det muligt at bryde koden på en måneds tid.*

H6-3 håndterer dette som en trussel. Nu kan vi se at der er andre løsninger. Løsningskolonnen nævner to der er langt mere bekvemme for brugerne.

H6-6, *forhindre uvedkommende i at få adgang til personoplysninger,* lyder let, men omfatter en lang række trusler, fx wire-tapping (kopiere data under transport i netværket) og it-medarbejdere der kigger i data på disken. Leverandørens forslag kan blive en lang roman - og det er svært at sammenligne to leverandørers romaner. Vi foreslår at man udelader kravet og sikrer sig at risikovurderingen dækker alle trusler på området og inkluderer de alvorlige som krav i H6.

H6-7 prøver at løse problemet ved at henvise til en lov, *Lov om behandling af personoplysninger*. Her opstår ofte et interessant spil. Kunden har ikke læst den pågældende lov, men forestiller sig at den sikrer mod problemet (det gør den kun delvis). Hvis han nu giver leverandøren besked på at overholde loven, så har leverandøren vel overtaget ansvaret?

Leverandøren kender muligvis loven og ved at den ikke dækker tilstrækkeligt. Han ved også at kundens formål med kravet bare er at fraskrive sig ansvaret, og at overholdelse af loven ikke bliver verificeret ved afleveringsprøven. Hvorfor skulle han dog gøre opmærksom på det? Resultat? Det egentlige behov bliver ikke dækket.

H6-7 kan være en nyttig tilføjelse til sikkerhedskravene, men den bør ikke opfattes som en erstatning for risikovurdering og trusselskrav.

H6-8 beder leverandøren følge udviklingen af nye trusler og gøre noget ved dem.

Alternativ 2: Kunden har ikke lavet en risikovurdering

H6-1 beder leverandøren liste de vigtigste risici og foreslå beskyttelse. Bemærk at vi ikke beder ham lave en risikovurdering, men kun liste trusler for denne type projekt. Taler vi om simple anvendelser som internet-butikker, og har leverandøren ekspertise her, bør dette krav være tilstrækkeligt.

Har kunden et usædvanligt projekt, bør han bede leverandøren lave en risikovurdering med kundens profil. Dette er dyrt for begge parter, så det bør laves under projektet, fx som del af det tidlige bevis (B3).

H6. Sikring mod trusler

Alternativ 1:

En risikovurdering har vist at følgende trusler er alvorlige. Systemet skal beskytte mod dem.

Systemet skal beskytte mod:	Eksempel på løsning:	Kode:
1. At uvedkommende skaffer sig administrator-rettigheder via internettet (hacking).	Rettighederne kan kun benyttes internt.	
2. Wire-tapping af passwords eller data.	Kryptering.	
3. En angriber prøver alle mulige passwords med et særligt program.	Passwords skal være mindst 9 tegn med både store og små … (besværligt). Eller: mindst 5 sekunder mellem to log-in forsøg. Eller: Blokér adgang efter 3 forsøg.	
4. SQL injection (angriberen indtaster en database-kommando hvor systemet forventer fx et personnavn. Resultatet er at systemet udfører database-kommandoen).		
5. DoS angreb (Denial of Service). En angriber sender så mange forespørgsler til systemet at det lammes.		
6. At uvedkommende får adgang til personoplysninger. For åbent krav, se vejledningshæftet.		
7. Systemet skal overholde Lov om behandling af personoplysninger (Lov nr. 429 af 31. maj 2000). OK, men kontroller at alle risici er dækket. Se vejledningshæftet).		
8. Leverandøren skal løbende følge udviklingen på sikkerhedsområdet og levere de nødvendige beskyttelser.		
. . .		

Alternativ 2:

Kunden har ikke lavet en risikovurdering.

Beskyttelse mod trusler:	Eksempel på løsning:	Kode:
1. Leverandøren skal liste de trusler der er mest alvorlige for denne slags systemer og angive de beskyttelser han anbefaler.		

I. Brugervenlighed og design

Brugervenlighed betyder at systemet er let at lære, effektivt i daglig brug, let at huske for lejlighedsvise brugere, let at forstå - også i usædvanlige situationer, og behageligt at bruge. Disse såkaldte *usability-faktorer* er ikke alle lige vigtige - det afhænger af hvad det er for et system vi specificerer.

Når vi taler om manglende brugervenlighed, forudsætter vi at systemet teknisk set virker korrekt og svarer hurtigt, og at man faktisk kan udføre de opgaver der er behov for. Alligevel har brugerne problemer med at udføre opgaverne.

Mange udviklere, designere og ekspertbrugere tror de kan granske skærmbillederne og se om systemet er tilstrækkeligt brugervenligt. Det er bevist mange gange at dette er umuligt. Det er heller ikke nok at følge gode råd (guidelines). Brugervenlighed skal testes og måles med rigtige, potentielle brugere.

Brugervenlighed kan måles på mange måder. En af de vigtigste er at observere brugere udføre nogle realistiske arbejdsopgaver med systemet eller en primitiv prototype af det. Man noterer ned når brugeren behøver hjælp udefra, er lang tid om at finde løsningen, mv. Dette kaldes en **usability-test**. De problemer man noterer ned, kaldes *usability-problemer*.

Første gang man usability-tester et mellemstort system, finder man 20-50 usabilty-problemer. Der skal 1-3 re-designs til for at få et acceptabelt resultat.

Vi kan bede brugeren *tænke højt* undervejs. Det giver langt bedre muligheder for at forstå hvorfor brugeren løb ind i problemet, og dermed bedre muligheder for at udviklerne kan fjerne problemerne.

Vi kan rimeligt objektivt klassificere problemerne som kritiske og mindre kritiske. Skabelonen viser detaljerne som en kravnote under tabellerne. Man kan så formulere usability-krav som det tilladte antal kritiske usability problemer. Bemærk at et problem kun kaldes kritisk hvis to eller flere brugere har oplevet det. Det er fordi der normalt er mange usability problemer der kun observeres én gang (*singulære* problemer). Som regel kan det ikke betale sig at gøre noget ved dem.

I1. Indlæring og effektivitet i daglig brug

Ofte er det nye system stort set færdigt og det er meget begrænset hvad man kan ændre i brugergrænsefladen. Klager kunden over den besværlige grænseflade, får han at vide at det er et standardsystem der ikke kan ændres, eller at han jo har set det inden han købte det (underforstået at brugerne bare er for dumme). Men tekniske fejl vil leverandøren gerne rette, fx forkert opdatering af databasen.

Når brugergrænsefladen er udviklet specielt til dette projekt, er kunden i næsten samme situation. Leverandøren afviser problemerne.

Krav I1-1 håndterer denne situation. Kritiske usability-problemer skal behandles som andre fejl, dvs. prioriteres og udbedres svarende til hvor alvorlige de er for kunden (se L5).

I. Brugervenlighed og design

I1. Indlæring og effektivitet i daglig brug

Selvom systemet har en færdig brugergrænseflade, kan den på nogle punkter give brugerne væsentlige ulemper. Kunden vil gerne undgå den situation at leverandøren afviser problemerne med henvisning til at kunden har købt et system "som beset".

Krav til håndtering af usability-problemer:	Eksempler på løsning:	Kode:
1. Kritiske usability-problemer (se definitionen i kravnoten nedenfor) skal behandles som systemfejl på linje med andre fejl i systemet.	Fejlen behandles af support-organisationen og overdrages om nødvendigt til vedligehold.	

Kravnote: Alvorlige og kritiske usability-problemer
Et **alvorligt** problem er en situation hvor brugeren:
a. ikke kan gennemføre opgaven på egen hånd,
b. eller tror den er udført selvom den ikke er,
c. eller klager over at det her *er virkelig besværligt*,
d. eller forsøgslederen kan se at brugeren ikke anvender systemet effektivt.

Et **kritisk** usability-problem er et alvorligt usability-problem som er observeret for mere end én bruger.

Dele af brugergrænsefladen skal udvikles

Når hele eller dele af brugergrænsefladen skal udvikles, er det vigtigt at det sker på en succesfuld måde.

Erfaringen viser at usability problemer skal opdages og rettes inden der er programmeret noget. Senere er det alt for dyrt at rette problemer der kræver programændringer. Derfor laver man tidlige prototyper af skærmbillederne med papir og blyant eller simple IT-værktøjer. Man bruger så prototypen til tænke-højt forsøg. De fleste usability problemer kan faktisk afsløres på den måde. Bagefter ændrer vi prototypen, tester igen, osv. Disse erfaringer er baggrunden for de tidlige beviser i B3-2.

I1-2 stiller krav om brugervenlighed på sådan en måde, at man tidligt kan vurdere om systemet kan blive tilstrækkeligt brugervenligt. Vi definerer også de detaljerede brugervenlighedskrav der skal bruges senere. Samtidig designer vi dele af brugergrænsefladen og tester dens brugervenlighed. Erfaringen viser at systemudviklingen går hurtigere når en detaljeret, afprøvet brugergrænseflade foreligger tidligt.

Før det tidlige bevis kan det være svært at specificere den præcise måde vi skal måle brugervenligheden, og kunden kan let stille urealistiske krav. Hvad ville der fx ske hvis vi fjernede krav I1-2 og beholdt I1-3 til 7? Vi ville så kræve at brugerne var i stand til at udføre alle task med få kritiske usability-problemer, var i stand til at forstå alle fejlmeddelelser, osv.

Leverandøren skulle så i tilbuddet specificere hvor mange usability problemer der måtte være, misforståelser, osv. Det er stort set umuligt for systemdele der slet ikke findes endnu. Formålet med I1-2 er at finde nogle rimelige usability-krav tidligt i projektet.

I1-3 til 7 er skitser af usability-krav der skal defineres i detaljer under det tidlige bevis. Fx skal de nøjagtige test task og tallene i kolonne 2 fastsættes.

I1-3 tester at brugerne efter den planlagte introduktion kan udføre deres daglige arbejdsopgaver med minimal hjælp fra andre.

I1-4 tester at fejlmeddelelserne er brugbare. Hvorfor er denne test nødvendig når vi allerede har tjekket at brugerne kan udføre deres task? Fordi brugerne under en usability test kun kommer i berøring med en lille del af systemets fejlmeddelelser. Med I1-4 kan man teste en større del af meddelelserne, også de der kun vil forekomme sjældent.

I1-5 siger at systemet skal kunne betjenes uden mus, og at brugerne selv skal kunne lære det. Dette krav er selvfølgelig ikke altid relevant.

I1-6 handler om store systemer som en typisk bruger ikke på egen hånd kan finde ud af. Kunder stiller ofte krav om kurser som alle brugere skal tage, men det er tit en dyr og dårlig mulighed. I stedet beder vi om måder superbrugerne kan lære systemet og selv uddanne andre. En måde er at give superbrugerne kurser. I krav J2-1 beder vi leverandøren om sådanne kurser.

I1-7 handler om effektivitet for den hyppige bruger. Under den tidlige usability-test kan man kun få en fornemmelse af hvor hurtigt opgaven burde kunne udføres, men det kan reelt først måles når systemet er næsten færdigt.

Web systemer der kun bruges lejlighedsvis

Skabelonen viser krav der er egnet til produktionssystemer som bruges dagligt. Men I1-3, 6 og 7 er som regel irrelevante for web-sites der kun bruges lejlighedsvis af offentligheden. Der er ingen superbrugere i nærheden og effektivitet er ikke vigtig.

Test task

Princippet i afsnit I1 er at udføre usability-test for at finde og fjerne usability-problemer. Et afgørende punkt er hvordan man definerer de test-task som brugerne skal udføre. Skabelonen foreslår at kunden skriver nogle test-task i en kravnote inden man beder leverandørerne om tilbud. Det giver leverandørerne en ide om hvad kunden forventer. Parterne kan justere test-task under det tidlige bevis.

Skabelonen giver et eksempel på et godt test task og et med "skjult hjælp", men der er andre ting man skal overveje, fx hvordan testopgaverne dækker de vigtigste aspekter af systemet. Se fx Lauesen (2005), kapitel 13.

Relevant når væsentlige dele af brugergrænsefladen skal udvikles:

Det skal sikres at systemet opnår den fornødne brugervenlighed. Det bør ske ved tidlige usability-tests. Efter de tidlige tests, aftaler parterne de detaljerede krav der skal verificeres ved overtagelsen. Det kan fx være præcisering af de testopgaver der skal udføres (se kravnote nedenfor) og de tal der skal indsættes i kolonne 2 nedenfor.

Kan parterne ikke enes om de detaljerede krav, kan de opsige aftalen (jvf. afsnit B3-2).

Krav til tidligt bevis:	Eksempler på løsning:	Kode:
2. Parterne skal usability-teste brugergrænsefladen snarest efter at kontrakten er underskrevet. De kritiske usability-problemer (se kravnoten ovenfor) skal rettes indtil usability-testen giver tilfredsstillende resultat. Desuden aftales de detaljerede usability-krav.	For dele af systemet der allerede findes, udføres der tænke-højt test i en passende systemopsætning. For dele der ikke findes endnu, udføres der tænke-højt test med papirprototyper. Der testes med tre nye brugere i hver runde.	

Krav der skal aftales i detaljer under det tidlige bevis, og verificeres ved overtagelsen:		
3. Efter en kort instruktion fra superbrugerne, skal brugerne kunne udføre alle task i kapitel C, inden for deres respektive arbejdsområder med få kritiske usability-problemer.	Inden for hvert arbejdsområde udføres tænke-højt test med 3 tilfældigt valgte medarbejdere. Der må højst observeres __ kritiske usability-problemer.	
4. Fejlmeddelelserne skal være forståelige og hjælpsomme.	Under tænke-højt testen vises et udvalg af fejlmeddelelser for brugeren, som skal forklare hvad meddelelsen betyder og hvad han skal gøre. __% af forklaringerne skal være acceptable.	
5. Systemet skal kunne betjenes med tastaturet alene. Brugerne skal selv kunne lære det.	Sent i tænke-højt testen bliver brugerne bedt om kun at bruge tastaturet. __% af brugerne skal kunne gøre det.	
6. Superbrugere skal hurtigt kunne lære at betjene systemet så de kan instruere andre brugere i det (jvf. J2-1).	Uddannelsen af superbrugere tager ___ dage. (Kunden forventer 3 dage).	
7. En bruger der har brugt systemet i en uge skal hurtigt kunne bestille 5 ydelser til en patient, fx lab test, scanning …	En typisk bruger kan bestille sådanne 5 ydelser på ___ minutter.	

Kravnote: Testopgaver
En god testopgave svarer til et task en rigtig bruger kunne udføre. Den skal præsenteres på sådan en måde at den ikke vejleder brugeren. Her er et eksempel på en god og en dårlig testopgave:

Testopgave 1 (god): Ordinér medicin: Patienten klager over smerter. Brug systemet til at behandle problemet.
(Når brugeren udfører opgaven, notér om han først ser på den eksisterende medicinering inden han ordinerer noget.)

Testopgave 2 (dårlig - vejleder brugeren): Ordinér medicin: Patienten klager over smerter. Indtast CPR-nummeret og vælg medicineringsskærmen. Kig på de andre medicineringer og beslut hvad du vil ordinere. Luk medicineringsskærmen og vælg ordineringsskærmen …

I2. Tilgængelighed og Look-and-Feel

Nogle aspekter af brugervenlighed er svære at udtrykke gennem usability-test. Regler og standarder kan være bedre.

I2-1 siger at brugergrænsefladen skal følge MS-Windows standarden. Bemærk begrundelsen: De fleste brugere er vant til Windows, så standarden vil gøre systemet lettere at lære. Har man ikke en god begrundelse, er der ingen grund til at kræve en standard. Mange tror at en standard sikrer brugervenlighed. Det gør den ikke. Den kan højst bidrage lidt og undertiden kan den endda være skadelig. Det er ikke gratis af følge en standard. Det er forbløffende svært at tjekke om en sådan standard er overholdt - og rette fejlene.

I2-2 siger at brugergrænsefladen skal være egnet for blinde, svagsynede, mv. En mulig løsning er at følge HTML principperne, der netop er udviklet til det formål (og mange andre formål). De siger fx at man skal bruge de standardiserede overskriftsmarkeringer i stedet for selvdefinerede, mere imponerende formater. Det giver højtlæsningsprogrammerne mulighed for at bruge intonation til at angive hvad der er overskrifter. På samme måde skal man undgå faste kolonnebredder og bogstavstørrelser, så svagsynede kan forstørre teksten kraftigt.

Nogle kravspecifikationer erstatter hele krav I2-2 med et krav om at web-siderne skal kunne passere en W3C Markup Validation test (http://validator.w3.org). Testen analyserer en web-side og påpeger fejl. Her er atter et eksempel på at man henviser til en standard i den tro at så er man dækket. Testen finder kun formelle fejl, fx at der mangler anførselstegn eller slutelementer. Den siger intet om fx egnethed til blinde eller svagsynede. Det gør derimod de andre retningslinjer i WCAG10, men de kan ikke tjekkes af en computer.

I2-3 er et eksempel på at man har brug for at præcisere hvilket sprog der skal bruges på grænsefladen.

I2-4 er en simpel måde at reparere nogle af de usability-problemer der først opdages når systemet er i drift. Mange usability-problemer, fx en besværlig brugergrænseflade, kan ikke repareres på den måde.

I2. Tilgængelighed og Look-and-Feel

Krav:	Eksempel på løsning:	Kode:
1. Brugergrænsefladen skal følge MS-Windows standarden, som de fleste brugere er vant til.		
2. Websiderne skal kunne læses højt, skalere til svagsynede, samt udnytte den fulde skærmstørrelse på såvel små som store skærme.	Siderne følger HTML-principperne for Accessability (WCAG10 fra W3C).	
3. Brugergrænsefladen skal være på dansk. Web-delen om åbningstider, telefonnumre og adresser skal dog være tilgængelig på både dansk, engelsk, tyrkisk og urdu.		
4. Kunden skal selv kunne tilføje hjælpetekster – også efter leveringen.	Simple pop-up tekster som kunden kan definere.	

J. Andre krav og leverancer

Dette kapitel samler en række krav der ikke passer ind i de andre kapitler.

J1. Andre standarder der skal følges

De fleste standarder hører hjemme i et af de andre kapitler, fx om sikkerhed eller brugervenlighed. Resten kan skrives her.

I praksis ser man at kunden lister den ene standard efter den anden, ofte uden at vide hvad den egentlig omfatter og hvorfor den er relevant. Det er som regel meget besværligt at kontrollere om en standard er overholdt. For en ansvarsbevidst leverandør betyder det en forhøjelse af prisen, mens en mindre omhyggelig leverandør regner med at kunden alligevel ikke vil kontrollere om standarden er opfyldt (se eksemplerne i H5 og I2).

Skabelonen viser kun et enkelt eksempel på en standard (af den bløde slags). Det præciseres at leverandøren skal sørge for certificering eller godkendelse, dvs. en uafhængig kontrol af at systemet overholder standarden. Det fritager kunden for selv at tjekke.

J2. Uddannelse

Ofte glemmes uddannelse af brugerne, eller der kræves et urealistisk omfang. Man ser også at uddannelsen kommer på et forkert tidspunkt, fx så tidligt at brugerne har glemt det hele når systemet endelig kommer i drift.

J2-1 er et eksempel hvor man har indset at leverandøren kun behøver at uddanne superbrugere. Vi beder leverandøren uddanne 50 superbrugere. Uddannelsen skal gøre dem i stand til at uddanne de andre brugere. Det er en erkendelse af at de fleste leverandørkurser er for fjernt fra brugernes egentlige arbejdsopgaver. Man bruger derfor superbrugere som mellemled. Det præciseres hvad superbrugerne skal kunne efter uddannelsen (se også I1-5).

J2-2 stiller tilsvarende krav for uddannelse af kundens it-medarbejdere.

J2-3 præciserer hvornår uddannelsen skal finde sted i forhold til overtagelsen af systemet.

J. Andre krav og leverancer

J1. Andre standarder der skal følges

Krav:	Eksempler på løsning:	Kode:
1. Systemet skal overholde god regnskabsskik.	Leverandøren skal sørge for den nødvendige revisorgodkendelse eller certificering.	
2. …		

J2. Uddannelse

Kunden ønsker at varetage en stor del af uddannelsen selv. Det tænkes gjort ved først at uddanne superbrugere, der så kan uddanne andre.

Krav:	Eksempler på løsning:	Kode:
1. Leverandøren skal uddanne 50 superbrugere, så de kan varetage uddannelsen af andre medarbejdere. Uddannelsen skal gøre superbrugerne i stand til at udføre alle task i kapitel C, inklusive alle varianter, inden for deres respektive arbejdsområder.	Uddannelsen af en superbruger kan ske på ___ dage. (Kunden forventer at 3 dage er nok).	
2. Leverandøren skal uddanne 10 it-medarbejdere så de kan varetage kundens del af drift og support.	Uddannelsen af en it-medarbejder kan ske på ___ dage. (Kunden forventer at 10 dage er nok).	
3. Uddannelserne skal gennemføres inden for den sidste måned før overtagelsen, således at medarbejderne straks kan bruge systemet og ikke allerede har glemt hvordan. Om nødvendigt må uddannelsen gentages og overtagelsen udskydes.		
4. …		

J3. Dokumentation

Bruger- og system-dokumentation glemmes ofte. I eksemplet præciseres det at der ikke behøver være fuld dokumentation for alle. Det er i erkendelse af at meget få brugere anvender dokumentation eller online hjælp, selv hvis den er til rådighed og i rimelig brugbar form. Denne erkendelse kan spare mange omkostninger og frustrationer hos begge parter.

J3-1 og 5 angiver at der skal være kursusmateriale som superbrugerne kan anvende når de underviser andre brugere, fx inden systemovertagelsen. Det skal foreligge i sådan en form at superbrugerne selv kan tilpasse det, fx med eksempler fra brugerens verden. J3-2 kræver at der er fuld dokumentation til superbrugerne kort efter systemovertagelsen.

J3-3 kræver på samme måde dokumentation til kundens it-medarbejdere.

J3-4 kræver dokumentation af specielt udviklet software og tekniske grænseflader. Kriteriet er at dokumentationen skal gøre tredjepart i stand til at vedligeholde disse dele og overføre data til andre systemer. For at sikre at leverandøren rent faktisk kan levere den nødvendige dokumentationskvalitet, kan man i B3 bede om et tidligt bevis.

J4. Datakonvertering

Datakonvertering fra tidligere systemer til det nye system er ofte en meget betydelig del af leverancens pris. Dette afsnit angiver hvad der skal konverteres. Det er vigtigt at kunden dokumenterer dataformaterne, da leverandøren ellers må skaffe oplysningerne selv inden han kan give et forsvarligt tilbud. Det kan afholde gode leverandører fra at byde.

Validering af datakonverteringen er et stort problem, som leverandøren normalt har langt mere erfaring med end kunden. Derfor beder J4-3 leverandøren beskrive hvordan han vil validere konverteringen.

J5. Installation

Dette afsnit præciserer hvem der installerer hvad. Hvis kunden ønsker at installere selv, kan han bede om den nødvendige dokumentation og et overslag over hvor lang tid det vil tage.

J3. Dokumentation

Det forventes at kun superbrugere, driftsfolk og systemudviklere vil læse dokumentationen. Der er derfor ikke behov for begynderdokumentation, bortset fra kursusmateriale.

Krav:	Eksempler på løsning:	Kode:
1. Der skal senest ved overtagelsen være kursusmateriale til rådighed som superbrugerne kan anvende når de underviser andre brugere. (Kunden bidrager selv med vejledning om hvordan de eksisterende arbejdsgange skal udføres i fremtiden, se K-10).		
2. Der skal senest en måned efter overtagelsen være dokumentation til rådighed af alle systemets funktioner set fra et brugerperspektiv. Dokumentationen skal være anvendelig for superbrugerne.		
3. Der skal senest ved overtagelsen være tilstrækkelig dokumentation til at kunden kan varetage sin del af drift og support.		
4. For specielt udviklet software og tekniske grænseflader som tredjepart kan benytte, skal der senest to måneder efter overtagelsen være dokumentation til rådighed der er egnet til videreudvikling.		
5. Al dokumentation skal leveres på maskinlæsbar form, og kunden skal frit kunne modificere den og kopiere den til eget brug.		
6. …		

J4. Datakonvertering

Leverandøren skal konvertere følgende data fra de eksisterende systemer:	Eksempler på løsning:	Kode:
1. De data fra PAS systemet som det nye EPJ-system skal håndtere i fremtiden. Formatet er beskrevet i …		
2. De data fra det gamle EPJ-system som det nye EPJ-system skal håndtere i fremtiden. Data må overføres ved IBM 3270 emulering. Se skærmbilledformatet i …		
3. Alt konverteret data skal valideres.	Leverandøren bedes beskrive hvordan.	
4. …		

J5. Installation

Krav:	Eksempler på løsning:	Kode:
1. Leverandøren skal installere alt hvad leverancen omfatter, såvel hardware som software.		
2. Leverandøren skal ligeledes installere de konverterede data.		
3. …		

J6. Test af systemet

Leverandøren skal selv udføre systemtest (se kaptitel 5), så i princippet skal kunden ikke interessere sig for det. Erfaringer viser dog at mange leverandører er meget dårlige til at teste, så der kan være god grund til at kigge leverandøren i kortene her. Specielt kan det være vigtigt at kunne genteste systemet når det er blevet ændret (regressionstest).

J6-1 og J6-2 præciserer behovene og skitserer nogle løsninger.

Desuden har kunden behov for selv at teste systemet, fx anvendelsestesten i forbindelse med overtagelsesprøven (se kapitel 5). Mange kunder bliver lokket ud i at teste på en udgave af systemet der er sat i drift med rigtige brugere og rigtigt data. Det kan efterlade det mærkeligste data i databasen og forstyrre driften på anden vis. Bør undgås.

J6-3 til 5 præciserer behovene og skitserer nogle løsninger.

J7. Udfasning

På et eller andet tidspunkt vil kunden gerne af med det gamle system og bruge et nyt. Så kommer der nye problemer. Hvilke data ligger der i det gamle system? Hvordan kan vi konvertere det til et nyt? Vil leverandøren hjælpe os - også når han ved vi vil af med ham?

J7 stiller krav til leverandørens assistance, samt hjælpemidler og dokumentation kunden kan få brug for.

Det er vigtigt at have disse aftaler på plads fra starten mens parterne er på god fod.

J6. Test af systemet

Krav til leverandørens test af systemet:	Eksempler på løsning:	Kode:
1. Kunden har behov for at kontrollere hvilke test leverandøren udfører og hvor godt de dækker.	Leverandøren stiller sine test cases og testprincipper til rådighed for kunden.	
2. Der er behov for at kunne gentage store dele af testen efter ændringer.	Leverandøren bruger regressionstest.	

Krav til kundens egen test af systemet:	Eksempler på løsning:	Kode:
3. Kunden har behov for selv at teste systemet inden overtagelsen.	Leverandøren stiller en testversion af systemet til rådighed for kunden.	
4. Der er behov for data til test af særlige situationer.	Kunden kan selv lægge særlige test data ind.	
5. Der er behov for realistiske data til testen.	Leverandøren konverterer dele af kundens eksisterende data og anbringer dem i testversionen.	

J7. Udfasning

I dette afsnit betyder "kunden" kundens egen stab eller tredjepart bemyndiget af kunden.

Krav:	Eksempler på løsning:	Kode:
1. Leverandøren skal på kundens anmodning udtrække alt data beskrevet i kapitel D, på en form der er egnet til import i andre systemer.		
2. Kunden skal selv kunne udtrække alt data beskrevet i kapitel D på en form der er egnet til import i andre systemer.		
3. Leverandøren skal levere beskrivelser af alle tabeller og felter på elektronisk form.		
3. Leverandøren skal loyalt hjælpe med at udfase systemet eller overdrage det til en anden leverandør.		
4. Leverandøren skal udføre arbejdet til en fair pris der dækker de brugte timer og materialer.		

K. Kundens leverancer

Det meste af kravspecifikationen handler om hvad leverandøren skal levere, men et it-system er ikke bare noget man "ruller ind og sætter i stikkontakten". Kundens medarbejdere skal bidrage på forskellig vis, og leverandørens personale skal måske have kontor og faciliteter under udvikling og idriftsættelse.

Dette kapitel beskriver hvad kunden skal levere. Leverandøren kan evt. i kolonne 2 præcisere hvad han forventer at kunden skal levere. Desuden kan han i sit tilbud tilføje nye punkter på listen.

I mange anskaffelser er systemintegration et stort problem, fordi leverandøren af det eksterne system der skal integreres med, skal hjælpe til. K-11 angiver at kunden skal sørge for de nødvendige rettigheder, fx ved at købe dem fra leverandøren af det eksterne system. Dette burde også være anført i kapitel F som forudsætninger leverandøen kan gøre, men vær sikker på at det står et eller andet sted.

I mange kontrakter er kapitel K erstattet af et separat bilag i kontrakten.

Ligesom i de andre afsnit er kravene kun eksempler og ikke en udtømmende liste. Efter juridisk praksis skal alt hvad kunden skal levere være specificeret. Leverandøren kan ikke efter kontraktindgåelse kræve at få kontor til rådighed eller ekspertise på et eller andet område, med mindre det er specificeret i kontrakten eller dens bilag.

K. Kundens leverancer

Nedenstående liste af kundens leverancer skal være udtømmende, og leverandøren kan ikke forvente yderligere leverancer. Leverandøren må derfor i sit tilbud tilføje til listen om nødvendigt.

Kunden leverer:	Eksempler på løsning:	Kode:
1. Hardware, software og eksterne systemer som det nye system forudsætter (se detaljer i kapitel G). Udstyret skal være til rådighed når installationsprøven starter.		N/A
2. Kontor med tre it-arbejdspladser fra en måned før planlagt installationsprøve til en måned efter overtagelsen.		N/A
3. Uddrag af produktionsdata til testformål og det fulde produktionsdata til konvertering.		N/A
4. Testcases til anvendelsestest.		N/A
5. Ekspertise på anvendelsesområdet, svarende til en halv medarbejder over hele projektets løbetid.		N/A
6. Testpersoner til usability-tests.		N/A
7. En halvtids projektleder og en halvtids sekretær.		N/A
8. Instruktører/superbrugere der selv lærer systemet så de kan uddanne menige brugere.		N/A
9. Ekspertise til validering af konverteret data.		N/A
10. Bidrag til kursusmateriale om fremtidige arbejdsgange (jvf. J3-1).		N/A
11. Frikøbte rettigheder til at integrere med de systemer der er nævnt i kapitel F og få den nævnte support.		N/A

L. Drift, support og vedligehold

Dette kapitel angiver leverandørens ansvar efter at selve produktet er leveret. Kravene kan kun delvis verificeres (testes) ved accepttesten. Vi kan fx simulere 2000 brugere og måle svartiderne, eller vi kan teste at support-organisationen virker, men vi kan ikke teste at det også virker når 2000 rigtige brugere arbejder med systemet.

Den egentlige verifikation sker senere, fx ved en driftsprøve eller ved granskning af logs og driftsstatistikker.

Skabelonen svarer til den situation hvor leverandøren er ansvarlig for drift, support og vedligehold. Hvis leverandøren fx ikke er ansvarlig for support, skal det tilsvarende afsnit være tomt. I dette tilfælde kan kunden have behov for kurser og dokumentation der gør det muligt for ham selv at supportere systemet. Kravene for det står i kapitel J.

Hvis leverandøren ikke er ansvarlig for drift, så kan vi ikke bare fjerne afsnit L1 (svartid) og afsnit L2 (tilgængelighed). Leverandøren er stadig ansvarlig for svartiderne hvis systemet kører på det udstyr der er beskrevet i kapitel G. På samme måde er leverandøren ansvarlig for tilgængeligheden. Hvis systemet bryder sammen pga. fejl i hans software, er han ansvarlig for den resulterende mangel på tilgængelighed. Dette præciseres i afsnit L2.

I mange kontrakter er dette kapitel flyttet ud i separate kontraktbilag.

L1. Svartider

Den indledende del angiver den **nominelle belastning** af systemet. Den nominelle belastning er det antal transaktioner systemet skal kunne håndtere per sekund med den angivne svartid. Det faktiske antal transaktioner per sekund bør ligge væsentligt under den nominelle belastning. Hvis det faktiske antal transaktioner er større end den nominelle belastning, behøver systemet ikke svare som angivet.

Ud fra den nominelle belastning kan leverandøren vurdere det nødvendige hardware.

I eksemplet er den nominelle belastning angivet som antallet af transaktioner af forskellig slags. Erfaringen viser at det ofte giver konflikter sent i projektet, fordi transaktioner af en bestemt slags kan have mange størrelser. Leverandøren kan fx have antaget at alle transaktioner er små, men i virkeligheden er nogle af dem enorme, men sjældne. Kunden insisterer på at måle dem også. Det anbefales at specificere den nominelle belastning mere præcist, fx under det tidlige bevis.

Systemet forventes at have mest travlt i visse perioder, **spidsbelastningerne**. De er ikke vigtige for kravene, fordi systemet skal kunne håndtere den nominelle belastning når som helst, men kunden vil måle den faktiske belastning og svartid i disse spidsbelastningsperioder.

Løsningsnoten beskriver en måde at måle svartiderne i praksis. Det er ikke et krav, og leverandøren kan angive sin egen målemetode i L1-2.

L. Drift, support og vedligehold

Dette kapitel beskriver leverandørens ansvar efter at selve produktet er leveret. Kravene kan kun delvis verificeres (testes) ved overtagelsen. Den egentlige verifikation sker senere, ved driftsprøven. Nogle af kravene er kun relevante hvis leverandøren har driftansvar, andre kun hvis han har support-ansvar, etc. Se vejledningshæftet.

L1. Svartider

Det er vigtigt at systemet svarer så hurtigt at brugerne ikke sinkes. Svartiderne er især vigtige i de travleste timer, **spidsbelastningsperioderne**, som er morgen 9-11 og ….

Når systemet er i drift skal det håndtere det antal transaktioner der er specificeret nedenfor, med den angivne svartid. Tallene er estimeret ud fra hyppigheden af task (kapitel C), datavolumen (kapitel D) og driftsstatistikker om spidsbelastning. Tallene er den **nominelle belastning**, dvs. at leverandøren ikke er ansvarlig for svartiderne hvis den aktuelle belastning er højere end den nominelle belastning.

Nominel belastning

1. Simple forespørgsler i klinisk session (C10): 10 pr. sekund i gennemsnit.
2. Opdateringer i klinisk session (C10): 2 pr. sekund i gennemsnit.
3. Simple forespørgsler i administrative task (C1 til C4): 3 pr. sekund i gennemsnit.
4. Den offentlige web-del: 5 sider hentes pr. sekund i gennemsnit.
5. …

Løsningsnote: Måling af svartid
Svartiden er tidsintervallet fra brugeren afgiver sin kommando til resultatet er synligt og brugeren har mulighed for at afgive en ny kommando. Ved en kommando forstås et tastetryk eller et museklik. Alle målinger udføres i perioder med spidsbelastning og med det aktuelle antal brugere, forudsat at den aktuelle belastning er indenfor den nominelle belastning ovenfor.

Arbejdsbrug: Målingerne udføres med en konfiguration svarende til kapitel G..

Den offentlige web-del: Målingerne udføres på en PC forbundet til Internettet med en 20 Mbit-ned/2 Mbit-op forbindelse og lav trafik på vejen til serverne, men spidsbelastning af serverne selv.

L1-1 angiver at den krævede svartid skal gælde for en bestemt **fraktil** af tilfældene. Løsningseksemplet siger at kunden forventer 98%, men leverandøren kan tilbyde en anden fraktil. Kunden kunne også forvente 99%. Hvorfor ikke bede om 100%? Fordi det er urealistisk i et flerbruger system. Transaktioner ankommer tilfældigt, og ved et tilfælde kan der komme en masse indenfor samme sekund. I det tilfælde vil de sidste få en meget lang svartid. Selvom dette er sjældent, kan vi ikke garantere en god svartid i 100% af tilfældene. Se mere i Lauesen (2002), afsnit 6.5.

L1-2 siger at der er behov for regelmæssige målinger i spidsbelastningsperioderne. I kolonne 2 har kunden givet eksempler på hvordan det kunne gøres. Leverandøren kan tilbyde en løsning der er bekvem for ham.

L1-3 til 9 angiver de krævede svartider. De er baseret på ergonomiske målinger af hvordan folk arbejder med computere (se Card et al.: The keystroke-level model, 1980). En hurtig bruger taster 5-10 tegn i sekundet, så 0,2 sekunder for at gå fra ét felt på skærmen til det næste, vil ikke sinke brugeren væsentligt.

En bruger anvender ca. 1,3 sekunder på mentalt at skifte fokus fra én delaktivitet til en anden, fx fra at indtaste klientdata til at indtaste klientens ordre. Hvis skærmbillederne er struktureret tilsvarende, vil 1,3 sekunder for at skifte skærmbillede ikke sinke brugeren. Dette princip bruges i L1-4, 5 og 6.

Der vil i praksis være tilfælde hvor systemet skal bruge lang tid til at svare, og hvor brugeren er forberedt på det. Her optræder en ergonomisk konstant på 20 sekunder. Selvom brugeren ved at det vil tage tid, vil han ubevidst vente omkring 20 sekunder og så begynde at lave noget andet. Skift fra én arbejdsopgave til en anden tager tid - tid som er spildt. For komplekse arbejdsopgaver kan den mentale skiftetid til en anden opgave være så lang som 10-20 minutter. L1-7 er et eksempel hvor 20 sekunder er acceptabelt.

Endelig kan der være funktioner hvor man af tekniske grunde må forvente svartider større end idealerne ovenfor. L1-8 og 9 (log-in) er eksempler på det. Ideelt bør log-in ske indenfor 1,3 sekunder, men nuværende erfaringer viser at vi kan blive nødt til at acceptere et langsommere svar.

Leverandøren kan i kolonne 2 anføre særlige funktioner der ikke overholder de fælles regler om svartid, fx at et bestemt oversigtsbillede kan tage 3 minutter at vise.

Web systemer der bruges lejlighedsvis

Svartiderne i eksemplet er beregnet til produktionssystemer i daglig brug. For websites der kun bruges lejlighedsvis, er det alt for strenge krav, som kan være urimeligt dyre at opfylde.

Krav til svartider:	Eksempel på løsning:	Kode:
1. **Fraktil.** Svartiderne nedenfor skal gælde i næsten alle tilfælde.	Indenfor hver time skal __ % af svartiderne være indenfor grænserne. (Kunden forventer 98%).	
2. Målingerne skal udføres regelmæssigt i spidsbelastningsperioder.	Der måles en gang ugentligt med stopur. Eller: Systemet måler selv løbende.	
3. Når man går fra et felt til det næste, skal brugerens tastehastighed ikke nedsættes.	Man kan taste inden ___s. (Kunden forventer 0,2 s).	
4. Når man går fra et skærmbillede til det næste, skal man kunne se billedets data og taste inden for den mentale skiftetid (ca. 1,3 s).	Man kan se data og taste inden ___s. (Kunden forventer 1,3 s).	
5. Opslag i dropdown-lister skal tillade valg fra listen inden for den mentale skiftetid.	Man kan starte valget inden ___s. (Kunden forventer 1,3 s).	
6. Visning af rapporter der bruges hyppigt skal ske inden for den mentale skiftetid.	Man kan se rapporten inden ___s. (Kunden forventer 1,3 s).	
7. Visning af rapporter der kun bruges lejlighedsvis, skal ske inden brugeren taber tålmodigheden.	Rapporten skal være synlig inden ___s. (Kunden forventer 20 s).	
8. Log-in skal kunne gennemføres inden brugeren taber tålmodigheden.	Brugeren kan starte arbejdet inden ___ s udover den tid det tager ham at taste navn og password. (Kunden forventer 10 s).	
9. Gentagen log-in når brugeren midlertidigt har været væk fra systemet, skal kunne ske uden væsentlig ventetid.	Man kan starte arbejdet inden ___s udover den tid det tager brugeren at taste sin identifikation. (Kunden forventer 4 s).	

L2. Tilgængelighed (driftseffektivitet)

Tilgængelighed angiver i hvor stor en del af tiden systemet er i drift set fra brugernes synspunkt. Hvad det præcis betyder at systemet er ude af drift må præciseres, og der må også tages stilling til hvordan man håndterer at nogle brugere oplever et driftsstop mens andre ikke gør. Fx vil man næppe regne det for et driftsstop hvis en enkelt bruger ikke kan få adgang til systemet.

Et driftsstop kan have forskellige årsager, og skabelonen nævner 5. Det er ikke dem alle der kan tillægges leverandøren. Hvis leverandøren ikke har driftsansvar, vil han alligevel være ansvarlig for driftsstop med årsag 3 (fejl i software eller konfiguration). Hvis leverandøren har driftsansvar, kan også strømsvigt, hardware-nedbrud, kapacitetsproblemer, mv. tillægges ham.

I princippet kan kunden stille alle mulige krav om måling af tilgængeligheden, men i praksis bør han acceptere de muligheder driftsoperatøren kan tilbyde - så længe de dækker kundens egentlige behov.

Løsningsnoten foreslår en måde at beregne længden af et driftsstop: Et driftsstop regnes altid for mindst 20 minutter. En driftsperiode skal vare mindst 60 minutter. Det skyldes at brugerne ikke får genoptaget deres afbrudte opgaver førend der er gået 20 minutter efter et driftsstop, og at de har svært ved at få noget ud af en driftsperiode på mindre end en time.

Skabelonen giver også forslag til at beregne tilgængelighed når kun nogle af brugerne er berørt af driftsstoppet.

L2-1 siger at tilgængeligheden skal opgøres periodisk. Der kan således ikke spares tilgængelighed op fra én periode til den næste. I kolonne 2 har kunden foreslået at tilgængeligheden beregnes som i løsningsnoten. Leverandøren kan foreslå sin egen beregningsmåde, evt. ved at henvise til et bilag.

L2-2 og 3 angiver den krævede tilgængelighed i to forskellige driftsperioder. Pas på ikke at kræve for meget. Det kan blive meget dyrt. Fx kan det koste 3 mio kr. om året at drive et stort system med 99% tilgængelighed, mens 99,8% kan koste 15 mio kr. om året. Er det pengene værd? En tilgængelighed på 99% i normal arbejdstid, betyder at systemet kan være ude af drift i 16 arbejdstimer om året. En tilgængelighed på 99,8% betyder 3,2 arbejdstimer om året.

Bemærk den måde L2-2 og 3 er formuleret. Den giver leverandøren mulighed for at foreslå andre tal end kundens. Se mere i afsnit A2.

L3. Datalagring

Dette afsnit beskriver hvor store datamængder der skal opbevares. Eksemplet skelner mellem data med hurtig tilgang og arkiveret data med langsommere tilgang. Visse pladskrævende billeder skal kun opbevares i kort tid.

Eksemplet henviser til de detaljerede datamængder i kapitel D, hvor der for hver tabel er angivet en samlet størrelse, og evt. en årlig tilvækst. Vi kunne også have angivet alle tabelstørrelserne her i afsnit L3 og fjernet dem fra kapitel D. Det ville være bekvemt at have dem begge steder, men det giver let inkonsistens.

L2. Tilgængelighed (driftseffektivitet)

Systemet er ude af drift når en del af brugerne ikke kan få støttet deres task som normalt. Årsagen til driftsstoppet kan være:

1. Kundens forhold, fx fejl i kundens udstyr.
2. Udefra kommende fejl, fx strømsvigt.
3. Leverandørens forhold, fx fejl i software eller konfiguration.
4. Planlagt vedligehold.
5. Utilstrækkelig hardware kapacitet.

Løsningsnote: Måling af tilgængelighed
Et **driftsstop** regnes altid for at vare i mindst 20 minutter, selv hvis driften reelt genetableres inden. Hvis en driftsperiode varer mindre end 60 minutter, regnes den for en del af driftsstoppet.

Hvis leverandøren ikke har driftsansvar, medregnes kun driftsstop af årsag 3. Hvis leverandøren har driftsansvar, er han også ansvarlig for årsag 2, 4 og 5.

Driftstiden i en periode beregnes som periodens samlede længde minus den samlede længde af de driftsstop som leverandøren er ansvarlig for. **Tilgængeligheden** beregnes som driftstiden i forhold til periodens samlede længde. Der kan kompenseres for at kun nogle brugere er ramt af et driftsstop, fx ved at tilgængeligheden udregnes pr. bruger hvorefter den gennemsnitlige tilgængelighed beregnes.

Krav til driftstid:	Eksempel på løsning:	Kode:
1. Tilgængeligheden skal opgøres periodisk. Der bør kompenseres for hvor stor en del af brugerne der oplever driftsstoppene.	Tilgængeligheden opgøres månedligt og beregnes som beskrevet ovenfor.	
2. I tidsrummet 8:00 til 18:00 på alle hverdage skal systemet have høj tilgængelighed.	I disse tidsrum er tilgængeligheden mindst ___ %. (Kunden forventer 99,5%).	
3. I andre tidsrum behøver tilgængeligheden ikke være så høj.	I disse tidsrum er tilgængeligheden mindst __ %. (Kunden forventer 99%).	

L3. Datalagring

Datavolumen fremgår af datakravene i kapitel D. Data skal opbevares som følger:

Krav til datalagring:	Eksempel på løsning:	Kode:
1. Systemet skal give adgang til de sidste 5 fulde kalenderårs data med de svartider der er anført i L1.		
2. Røntgenbilleder og … opbevares kun i __ dage.		
3. Systemet skal give adgang til arkiveret data for de sidste 20 kalenderår med svartider som rapporter der bruges lejlighedsvis (L1-7).		

L4. Support

Dette afsnit specificerer leverandørens support-ydelser, fx at hjælpe brugere (hotline), ændre systemkonfiguration, og overvåge driften. (ITIL har særlige navne for disse ting. Hotline hedder fx *Service Desk*. Se Bon, 2004).

Indledningen angiver som en forudsætning at superbrugere er de almindelige brugeres første kontakt. Hvis superbrugeren ikke kan løse problemet, kan enten superbruger eller almindelig bruger kontakte hotline. Vi kunne tillade almindelige brugere at kontakte hotline direkte, men i de fleste organisationer ville det blive meget dyrere - og mindre effektivt.

L4-1 angiver at den krævede svartid for hotline skal gælde for en bestemt **fraktil** af henvendelserne. Løsningseksemplet siger at kunden forventer 95%. Lad være med at kræve en maksimal svartid (gyldig for 100%). Det vil blive meget dyrt at overholde, fx ved hændelser hvor alle beder om hjælp samtidig.

L4-3 angiver i hvilke perioder brugerne kan kontakte hotline telefonisk eller personligt (direkte kontakt).

L4-4 kræver at supporteren ved direkte kontakt skal prøve at løse problemet på stedet. Mange SLA'er (Service Level Agreements) kræver at en hvis procentdel af henvendelserne skal løses på stedet. Erfaringen viser at det gør leverandøren interesseret i at få en masse trivielle henvendelser. Han er ikke motiveret til at forebygge dem, fx ved at annoncere hvordan visse problemer undgås.

Derfor beder L4-4 kun leverandøren bruge nogle få minutter på stedet. Hvorvidt supportens kvalitet generelt er tilstrækkelig, er svært at måle. L4-11 foreslår at parterne jævnligt mødes og drøfter det.

L4-5 angiver at for indirekte kontakter, skal brugeren have et første svar indenfor nogle få timer.

L4-6 og 7 beder om specielle ydelser såsom fjerndiagnose og at sende en supportperson til kundens arbejdsplads. Som for andre krav, kan leverandøren svare at han ikke tilbyder denne service. I mange projekter er der ingen behov for det, fordi kunden allerede gør det selv.

Kravnoten efter tabellen forklarer hvad det betyder at håndtere en henvendelse om hjælp (en *incident* i ITIL terminologi). Det beskrives som en liste af valgfri subtask. Efter de fleste subtask får brugeren et første svar. Et svar betyder at brugeren har fået hjælp til at løse eller omgå problemet, at et teknisk problem er blevet fjernet, eller at problemet er blevet overført til en anden organisation. Det er *ikke* et gyldigt svar at henvendelsen er blevet modtaget af hotline eller overført til en anden supporter i organisationen. Brugeren får ofte et foreløbigt svar og senere yderligere svar efterhånden som supporterne undersøger sagen.

Ligesom i andre dele af skabelonen er support-kravene eksempler og ikke en fuldstændig liste. ITIL standarden kan bruges til skabe længere lister af support-ydelser. Ligesom med andre standarder skal man passe på ikke at bruge dem blindt. Man kan ende med at betale for mere end man har brug for, eller komme til at bede om uhensigtsmæssige processer, fx *send altid svaret tilbage gennem brugerens første kontakt.*

L4. Support

Support omfatter hjælp til brugerne, overvågning af driftssituation, samt ændring af den tekniske konfiguration. I dette afsnit skal "leverandør" forstås som leverandørens driftsorganisation. En "supporter" betyder en kvalificeret medarbejder fra leverandøren. Hjælpen omfatter alt udstyr og programmel leveret under denne kontrakt.

Superbrugerne er de almindelige brugeres første kontakt om hjælp. Leverandøren skal derfor kun yde hjælp når superbrugerne ikke kan løse problemet.

Krav til support:	Eksempler på løsning:	Kode:
1. **Fraktil.** Svartiderne nedenfor skal gælde i næsten alle tilfælde.	__ % af svartiderne skal være indenfor grænserne. (Kunden forventer 95%).	
2. Leverandøren skal håndtere henvendelser om hjælp. Se kravnoten nedenfor.		
2p. Problem: Selv superbrugere har svært ved at afgøre hvilket produkt et problem hører under. Det er endnu sværere at fungere som bindeled mellem flere leverandører.	Leverandøren inddrager på egen hånd andre parter for at løse problemerne.	
3. Direkte kontakt: I perioden 8:00 til 18:00 på alle hverdage kan brugerne hurtigt få telefonkontakt eller personlig kontakt med en supporter.	I denne periode kan man få kontakt inden for ___ minutter. (Kunden forventer 10 minutter).	
4. Ved direkte kontakt giver supporteren så vidt muligt svar på stedet.	*På stedet* betyder hvad der kan gøres indenfor 5 minutter.	
5. Indirekte kontakt: Henvendelsen er sendt med e-mail, via web, eller eskaleret fra direkte kontakt. Brugeren får svar indenfor få timer.	Leverandøren svarer indenfor __ arbejdstimer (8:00 til 18:00 på hverdage). Kunden forventer 3 timer.	
6. Leverandøren udsender en supporter hvis det er nødvendigt for at løse problemet.		
7. Leverandøren kan udføre fjerndiagnose for at løse problemet.		
8. Leverandøren overvåger henvendelserne for at kontrollere at de lukkes og at svartiderne overholdes.		
9. Leverandøren registrerer data til beregning af svartider for support, og identifikation og forebyggelse af hyppige problemer.	Leverandøren fører en log over alle trin i behandlingen af henvendelserne med angivelse af problemets årsag.	
10. Leverandøren overvåger driftssituationen for at forudse problemer med tilgængelighed, og tager skridt til at ændre den tekniske konfiguration så tilgængeligheden sikres.		
11. Kunde og leverandør mødes regelmæssigt for at vurdere svartiderne og diskutere forebyggelse af problemerne.	Parterne mødes hver ___ måned. (Kunden forventer månedlige møder).	

Kravnote: Håndtér en henvendelse

Når en supporter modtager en henvendelse, kan han udføre et eller flere af de følgende subtask. Alle subtask undtagen e (eskalering) slutter med et **svar** til brugeren. Henvendelsen **lukkes** når der ikke kan gøres mere ved sagen (subtask f).

a. Hjælp brugeren: Assister brugeren med at løse problemet eller omgå det. Om nødvendigt, kontakt brugeren for at afklare problemet. Assistance er et gyldigt svar.
b. Ændr konfiguration: Fx start servere, ret indstillinger, udskift printer-patron, installer software. Svar brugeren når det er gjort.
c. Bestil udstyr eller hjælp fra en ekstern organisation. Svar brugeren om det forventede forløb.
d. Fejl: Support-organisationen kan ikke løse problemet. Rapporter det til vedligeholdelsesorganisationen. Svar brugeren om at det gjort.
e. Eskaler henvendelsen: Supporteren kan ikke løse problemet fuldt ud selv. Send henvendelsen videre til en anden supporter, som igen kan udføre et eller flere subtask.
f. Luk henvendelsen: Der kan ikke gøres mere ved sagen. Det kan ske ved første kontakt med support. Henvendelsen kan også eskalere flere gange, vente på en ekstern leverance eller på svar fra vedligeholdelsesorganisationen førend den kan lukkes. Svar brugeren når henvendelsen lukkes.

L5. Vedligehold

Dette afsnit viser typiske eksempler på krav til vedligehold. Der er både eksempler på afhjælpning af fejl, installation af nye versioner og forbedring af systemet.

L5-1 angiver at den krævede svartid skal gælde for en bestemt **fraktil** af henvendelserne. Løsningseksemplet siger at kunden forventer 95%.

L5-2 kræver at leverandøren fører en log over henvendelserne om vedligehold.

L5-4 siger at forretningskritiske fejl afhjælpes hurtigt, fx indenfor 24 timer. Men hvem afgør om en henvendelse er kritisk? Er det brugeren der rapporterer den, eller leverandøren? Det afhænger af hvilken slags system og kunde vi har med at gøre. Som regel skal brugere ikke afgøre det, fordi almindelige brugere ofte synes alt er kritisk. På den anden side vil leverandøren foretrække at intet er kritisk.

L5-3 foreslår at leverandøren afgør det og at hans beslutninger vurderes regelmæssigt (L5-5). Alternativ 1 er at den lokale superbruger afgør det og alternativ 2 at kundens it-afdeling afgør det.

Når et system skal ændres eller udbygges, har leverandøren monopol på at gøre det, og kan tage sig betalt derefter. L5-7 viser hvordan det kan undgås: Ændringens størrelse vurderes i antal *Function Points*, og leverandøren har angivet en fast pris pr. Function Point.

Function Points (FP) er en teknologi-uafhængig måde at måle størrelsen af en udviklingsopgave. Den er baseret på erfaringsdata fra tusindvis af projekter over hele verden.

Målingen kan fx baseres på antallet af klasser i E/R-modellen og deres kompleksitet, samt antallet af skærmbillede og deres kompleksitet. FP eksperter kan bruge task til at give et godt skøn over antallet af skærmbilleder. En medium kompleks klasse kræver 10 FP, et medium komplekst skærmbillede med bruger interaktion kræver også ca. 10 FP. Desuden er der en korrektionsfaktor på 0,3 til 1,6 for projektets organisation, mv. Ændringer til et system kan estimeres på tilsvarende vis.

Uden noget der svarer til E/R og task/use cases kan man ikke estimere projektets størrelse.

Afhængig af leverandørens ekspertise og teknologi, kan han tilbyde en højere eller lavere pris per Function Point. Typisk koster et FP i Danmark 10.000 til 20.000 DKK.

Det kræver ekspertise at bruge Function Points. FP eksperter hævder de er meget enige når de uafhængigt estimerer det samme projekt. Bliver der uenighed om antallet af FP, kan man få den danske Function Point gruppe til at afgøre sagen. L5-8 kunne angive at den skal bruges til at afgøre konflikter.

L5. Vedligehold

Vedligehold omfatter både afhjælpning af fejl, system opdatering og systemændringer.

Krav til afhjælpning af fejl:	Eksempler på løsning:	Kode:
1. **Fraktil.** Svartiderne nedenfor skal gælde i næsten alle tilfælde.	__ % af svartiderne være indenfor grænserne. (Kunden forventer 95%).	
2. Leverandøren fører en log over såvel rapporterede fejl som ændringsønsker.		
3. For alle rapporterede fejl afgør leverandøren hurtigt om fejlen er forretningskritisk, om den midlertidigt kan omgås, eller om den permanent kan omgås (afvises). Alternativ 1: Den lokale superbruger afgør det. Alternativ 2: Kundens it-afdeling afgør det.	Leverandøren udfører vurderingen inden for ___ timer i perioden 8:00-18:00, alle hverdage. (Kunden forventer 3 timer).	
4. Forretningskritiske fejl afhjælpes hurtigt.	Kritiske fejl afhjælpes inden for … timer. (Kunden forventer 24 timer).	
5. Kunde og leverandør mødes regelmæssigt for dels at kontrollere vurderingerne af fejlene, dels at afgøre hvad der skal rettes eller forbedres og hvad det skal koste.	Parterne mødes hver __ måned. (Kunden forventer månedlige møder).	

Krav til forbedring af systemet:	Eksempler på løsning:	Kode:
6. Leverandøren skal installere nye versioner og releases af det leverede software uden unødig forsinkelse.	Installationen sker inden for ___ dage efter at det er frigivet til brug i Danmark. (Kunden forventer 30 dage).	
7. Inden for en periode af 3 år skal leverandøren tilbyde ændringer til en fast pris pr. Function Point.	Prisen pr. Function Point er ____ DKK.	
8. Uenighed om beregningen af Function Points skal afgøres af …		

7. Litteratur og andre skabeloner

Alexander, Ian & Beus-Dukic, Ljerka: Discovering Requirements - How to Specify Products and Services. Wiley, 2009, ISBN 978-0-470-71240-5. Giver gode råd og eksempler på mange metoder og formuleringer. Indeholder eksempler fra mange anvendelsesområder.

Bon, Jan v., et al. (eds. 2004): IT Service Management - an Introduction based on ITIL. Van Haren Publishing, ISBN 90-77212-28-0. Beskriver på en overskuelig måde de processer der indgår i at drive og supportere et system (240 sider).

Card, Stuart K. et al. (1980): The keystroke-level model for user performance time with interactive systems. *Communications of the ACM,* 23 (7), pp. 396-410. Opdeler brugerens arbejde i grundbestanddele og måler tiden for hver slags bestanddele.

Constantine, Larry & Lockwood, Lucy A.D. (1999) Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design, Addison-Wesley. Beskriver en systematisk metode til at designe brugergrænseflader, fra analysen af behovene til prototype og usability test.

International Function Point Users Group IFPUG. http://www.ifpug.org/
Den traditionelle metode til at måle størrelsen af IT projekter. Den er baseret på erfaringsdata fra tusindvis projekter over hele verden.

Lauesen, Soren (2002): Software Requirements - Styles and Techniques. Addison-Wesley, ISBN 0-201-74570-4. En lærebog om hvordan man kan formulere krav, og hvordan man finder og kontrollerer dem. I alt forklarer den omkring 100 teknikker med realistiske eksempler. Der er også råd om hvordan man kan verificere krav (kontrollere om de er opfyldt) og hvordan man som leverandør overbeviser kunden. Indeholder store uddrag af virkelige kravspecifikationer der er formuleret på forskellige måder. Se:
http://www.itu.dk/people/slauesen/SorenReqs.html

Lauesen, Soren (2005): User Interface Design - A Software Engineering Perspective. Addison-Wesley, 0-321-18143-3. Viser hvordan man på en systematisk måde kommer fra task beskrivelse og datamodel, til en brugergrænseflade der opfylder kravene til brugervenlighed. Besvarer også det svære spørgsmål: Hvor mange skærmbilleder er nødvendige og hvad skal de indeholde? Se:
http://www.itu.dk/people/slauesen/SorenUID.html

Lauesen, Soren (2017): Problem-orienterede krav i praksis: Y-Fonden. Den fulde kravspecifikation inkl. leverandørens tilbud til Y-Fondens sagsbehandlingssystem. Der er også beslutningsnotatet til bestyrelsen om leverandørvalget, liste af fejl/ændringsønsker, etc. Se:
http://www.itu.dk/people/slauesen/Y-foundation.html

Lauesen, Soren & Kuhail, Mohammad (2012): Task descriptions versus use cases. In Requirements Engineering (a Springer Journal): ISSN 0947-3602 Requirements Eng (2012) 17:3-18, DOI 10.1007/s00766-011-0140-1. Viser med eksperimentelle resultater hvorfor use-cases ikke egner sig til krav, og hvordan task-metoden løser problemerne. Se også: http://www.itu.dk/people/slauesen/SorenReqs.html#UseCases

Patton, Ron (2006): Software testing. Sams Publishing, Indiana. ISBN 0-672-32798-8. Dækker mange slags test, fx white-box test, black-box test, compatibility test, fremmedsprogstest, og sikkerhedstest.

Robertson, Suzanne & Robertson, James (2012): Mastering the Requirements Process. Addison-Wesley. Bruger et konkret eksempel, et system til at styre saltning og snerydning af veje, til at forklare forfatternes Volere-metode. Dækker primært systemer der udvikles fra grunden. Medtager mange flere typer kvalitetskrav end skabelon SL-07.

Technology Group International: Software Selection Requirements Template (accessed May 2011). En skabelon til at sammenligne forretningssystemer (ERP) ud fra 1250 funktionelle krav på "produktniveau". Man skal registrere sig, men så er skabelonen gratis. http://www.tgiltd.com/erp-software-selection/erp-requirements-template.html

Wiegers, Karl E. (2003): Software Requirements, 2nd Edition. Microsoft Press, ISBN 0-7356-1879-8. Dækker mange aspekter af krav lige fra rettigheder og forpligtelser til processer og formuleringer. Illustreret med gode og dårlige krav, samt dialoger fra analysefasen.

Withall, Stephen (2007): Software Requirement Patterns. Microsoft Press, ISBN-0-7356-2398-8. En stor samling punkter man skal overveje, og eksempler på krav fra mange anvendelsesområder. Alle krav er på "produktniveau", dvs. løsninger snarere end egentlige behov. Der er fx ingen brugervenlighedskrav.