Danske Infra–strukturer: Menneskeskabte Systemer og deres IT

Fokusering på Professionelt Udviklet Informatik

Professor Dines Bjørner, Informatik og Matematisk Modellering, DTU
db@imm.dtu.dk, http://www.imm.dtu.dk/~db

- **At få programmellet rigtigt !**

  Læseren er sikkert fortrolig, og — hvad der i forbindelse med dette notat er mere relevant — føler sig rimelig tryg ved at anvende programmel, der betinger elektronisk post og muliggør brev– og rapport–skrivning. Men ville læseren på samme måde vide sig sikker, føle sig tryg, ved at flyve i et passagerfly, eller bo tæt ved et atomkraftværk, der blev delvist styret udfra programmel–forskrifter ? Enhver bygnings–ingeniør, der deltager i 'designingen' af en bro, foretager til stadighed matematiske beregninger med henblik på at "afprøve" om broen eventuelt styrter sammen. Tilsvarende for ingeniører og deres konstruktions–opgaver indenfor medicinsk elektronik, skibs–, fly– oma. konstruktioner. Og så videre. Men sligt — matematiske beregninger med henblik på at sikre programmel's korrekthed — er desværre oftere snarere undtagelsen end reglen indenfor programmel.

  Dette notat forsøger introducere læseren til metoder indenfor programmel–udvikling, der leder frem til mindst samme professionelle stade for data–ingeniører, som indenfor andre, mere klassiske ingeniør–discipliner. Det er ikke gjort alene med objekt–orienteret design, UML og strammere discipliner indenfor krav–definition. Der skal en hel ny måde til at anskue programmel på. Et anskuelses–synspunkt, der fokuserer på programmel som matematiskt bestemte — såvelsom uformelle, dog stringent Dansk– (eller Engelsk–sprogligt) beskrevne — størrelser. Dog, i dette notat skal vi alene fokusere på den uformelle, men stringente måde, i mange trin af udvikling, at anskue programmel.

- **At få det rigtige programmel !**

  Læseren er sikkert bekendt med ibrugtagning og foreløbig drift af sådanne programmel–systemer hvis funktionalitet "spænder" over flere "oktaver": De forventes tilbyde ikke bare almene funktioner indenfor hvert af flere, tilsyneladende kun løst relaterede funktions–områder, men forventes også at koble disse, oprindeligt set ikke–relaterede, funktions–områder sammen og at tilbyde funktioner der "går på tværs" og derved at tilbyde et reelt, begrebs–mæssigt "løft".

  I dette notat skal vi plædere for en tilgangsvinkel i programmel–udviklingen, der tilgodeser en fordybelse i forståelsen af genstands–området (også kaldet anvendelses–domænet, eller, som oftest, her, blot 'domænet'). Vi skal mao. plædere for udvikling af omfattende (både uformelle, dog rigorøst sådanne, såvelsom formelle) beskrivelser af sådanne infra–struktur–komponent–domæner som f.eks. transport–, sundheds–, (E–)handels–, og finans–sektoren.

- **Forandrings–ledelse**

  Læseren er sikkert bekendt med fatalt fejlslagne, t.eks., offentlige programmel–systemer (`Vue`, `Amanda`, oa.). I flere tilfælde kan store dele af "ansvaret" for "katastroferne" føres tilbage til topledelsen i klient–institutionerne: De har simpelthen ikke forstået, at mange programmel–systemer, typisk for infra–struktur–komponenter, kræver dybt–gående og bredt favnende forandringer i den måde hele institutionen i fremtiden, med det nye, omfattende programmel–system, skal ledes og drives på.

  Dette notat vil koble forandrings–ledelses–problematikken til problematikken omkring udviklingen af infra–struktur–komponent–domæne-beskrivelser.

Dette dokument blev første gang udleveret Onsdag 14 November 2001. Da blev en første net version installeret:

- HTML: `http://www.imm.dtu.dk/~db/infra/`
- Postscript: `http://www.imm.dtu.dk/~db/infra.ps`
- Portable Document Format:`http://www.imm.dtu.dk/~db/infra.pdf`

•••

Idéen til at skrive dette notat opstod i begyndelsen af Oktober 2001. Dels er det en første Dansk–sproget rapport fra min hånd indenfor mit forsknings– og undervisnings–felt; dels er det et første forsøg, på populariserende form, på at henvende sig til en basalt ikke–IT–teknisk gruppe "beslutnings–tagere" med henblik på at forklare denne gruppe hvorledes ihvertfald nærværende forfatter anskuer problematikken: At indforskrive, at udvikle og at tage ibrug endog meget store programmel–baserede IT systemer.

Når jeg således "henvender" mig er det jo forståeligvis for at bidrage mit til at ændre en situation sådan som jeg ser den.

Og denne situation er at jeg ser dagens indforskrevne informatik–systemer som langt fra gode nok, som langt fra ambitiøse nok, og som slet ikke udnyttende det store potentiale der dog ligger nedlagt i flertallet af de mange professionelle datalog–kandidater de seks datalogiske universitets–institutter i Danmark årligt genererer.

Mao. jeg vil gerne advare mod visse tendenser.

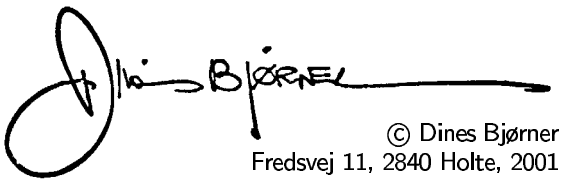
Jeg ser også meget gerne at det offentlige Danmark gav sig i kast med at indforskrive, at udvikle og at tage ibrug langt mere omfattende informatik–systemer, både i omfang og i dybde, systemer med langt større 'semantisk' indhold, end jeg i dag ser. Eette medfører langt mere eksperimentel udvikling, herunder forskning, end man idag er vant til. Dette, igen, kræver største åbenhed, og deltagelse af den samlede Danske datalogiske forsknings–verden, i snævert gensidigt arbejde med Danske IT udviklere, IT leverandører, og de offentlige institutioner — der skal bruge de således eksperimentalt udviklede informatik–systemer.

•••

Nu, hvor der foreligger en første "komplet" udgave af dokumentets hoveddel, dvs. undtagen Appendiks B (startende på side 79), er det på tide at forsøge formulere en ultra–kort version af hoveddelen. F.eks. i form af en "power–point"–lignende 30–45 minutters foredrags–lignende præsentation.

•••

Notatet benytter sig af, for mange læsere sikkert, fremmedartede sætnings–konstruktioner, og fremmedartet bogstavering (herunder en systematisk, men udbredt, brug af binde–streger i forbindelse med lange ord) og tegn–sætning.



Fredsvej 11, 2840 Holte, 2001

Danske Infra–strukturer: Menneskeskabte Systemer og deres IT
Fokusering på Professionelt Udviklet Informatik

Professor Dines Bjørner, Informatik og Matematisk Modellering, DTU
db@imm.dtu.dk, http://www.imm.dtu.dk/~db

Hensigten med dette notat er at meddele en gruppe af ikke–tekniske personer, sådanne som er beskæftigede f.eks. i den offentlige sektor, i ministerier, styrelser, mv., [— for at meddele dem —] en række nutidige forhold omkring nyere, teknisk–videnskabelige metoder som tilbyder muligheder for at man mere sikkert kan "tackle" udvikling af endog meget store programmel (cum IT) systemer, dvs. for at man med større troværdighed kan indforskrive og ibrugtage sådanne systemer.

Sagen er ikke helt så enkel som: *"at programmere"*, det kan man bare sådan lære, enten på en teknisk erhvervs–, eller på en teknisk–videnskabelig akademisk uddannelse. Og det på to år, om man tidligere har en Bachelor grad fra et andet dertil ikke relateret område. Derfor er dette notat lidt langt — og går i detaljer for at vise *"what IT takes to do IT !"*. Før man, stort set alene på dette notats grundlag, får en fornemmelse for dette, kan man ikke, mener jeg, troværdigt gennemskue hvilke herlige muligheder, der dog er, for at rekvirere endog meget store, meget komplekse programmel–systemer. Systemer, der kan sikre klart bedre sundheds-sektor service–systemer, klart bedre arbejds–markeds-anvisnings service–systemer, endnu bedre Told & Skat systemer, endnu bedre logistik systemer, klart bedre transport systemer, klart bedre finans–systemer, klart bedre I–handel[1], mmm.

Med afsæt i den senere tids omtalte IT "skandaler" (jvf. VUE, Amanda, Told & Skat, mfl. infra–struktur–komponenter) gives først, i Afsnit 3 (startende på side 13), en afgrænsning af og eksempler på begrebet 'infra–struktur'; dernæst i Afsnit 4 (startende på side 15) berettes om konventionelle måder, såvelsom om en nyere metode, for udvikling af programmel. I Afsnit 5 (startende på side 25) gennemgås et bestemt trin i udvikling af programmel efter denne nyere metode: Den såkaldte 'domæne–beskrivelses–metodik'. Afsnit 6 (startende på side 35) detaljerer hvorledes man kan bygge videre på domæne–beskrivelser i arbejdet med krav–definitioner og programmel–design (specifikation). Summen af Afsnittene 4—6 peger på programmel–udvikling som en uhyre findelt process, med mange principper, teknikker og værktøj. Virkeligheden idag lever desværre ikke op til den krævende modenhed — og vi står derfor i den noget tvivlsomme situation at man, i Danmark, arbejder langt mere seriøst i f.eks. skibs– og bro–bygnings–branchen end i programmel–konstruktions–branchen.

Afsnit 7 (startende på side 57) indsætter de tidligere afsnit i en større sammenhæng — idet afsnittet samtidig indfører og præciserer begrebet 'informatik'. Dette informatik–begreb kontrasteres til det komplementære 'IT' begreb. Vigtigheden af at fokusere kraftigere på informatik, end på det komplementære 'IT' begreb, understreges.

Afsnit 8 (startende på side 59) diskuterer domæne ingeniør–arbejdet i relation til *"Business Process [Re–]Engineering"*. Vi mener at den klare hovedvægt, der bliver lagt på domæne–beskrivelser dels medfører et afklaret forhold til *"Business Process [Re–]Engineering"*, dels medfører at man langt mere systematisk kan "tackle" problematikken: 'Forandrings–ledelse'. Som bemærket i anbefalinger (til undgåelse af visse negative aspekter ved indførelse af nye, store, infra–struktur–informatik–systemer), er det alt mere bydende nødvendigt at rekvirenter — de institutioner, der indfører disse systemer, til afløsning af tidligere arbejds–gange — i langt højere grad end tidligere akcepteret, forbereder sig på de forandringer sådanne, "om–sig–gribende" informatik–systemer med usvigelig sikkerhed medfører.

Et appendiks giver eksempler på beskrivelser af infra–struktur–komponenter.

[1] Det burde hedde I, for informatik, snarere end E, for elektronik, handel, da det jo ikke er elektronikken (alene), men især programmellet, og det bagved liggende informatik–begreb, sådan som dette notat vil beskrive det, der tilbyder de nye muligheder. Hvis ikke man tilfulde udnytter informatikkens spilleregler får man ikke meget nyt ud af sit programmel.

Indhold

1 Sammenfatning

Dette afsnit er formuleret som et *"Executive Summary"*.
Det opsummerer:

- (α) indholdet af (forholdene omkring, egenskaberne ved), og
- (ω) den lære vi mener der bør drages

af

- den videnskab der hedder datalogi,
- de metoder, principper, teknikker og værktøj datalogien stiller til rådighed for den praktiserende datalog (cum informatik–ingeniør),
- de muligheder denne datalogi — koblet med matematisk modellering i informatikken — afstedkommer for sikre måder at udvikle fleksible programmel–systemer for endog meget store anvendelser, og
- de infra–struktur–komponenter for hvilke det nu er muligt — i langt højere grad end det før var troværdigt — at udvikle sådanne programmel–systemer.[2]

1.1 Datalogi

Datalogi er læren og viden om hvilke størrelser der kan eksistere inde i datamaten, og hvorledes de konstrueres: Processer og data. Datalogi, som fag, kan, alt efter datalogi–forskerens eller datalogens indstilling (holdning), opdeles i en lang række fag–discipliner:

- Automat–teori, formelle sprog og beregnelighed
- Algoritmer og data strukturer, og disses kompleksitet
- Funktions–, logik–, imperativ–, og parallel–programmering
- Algebraisk–, denotations–, aksiomatisk–, og operational semantik
- Bevis–lære
- Type–teori
- Programmel–konstruktion *f.eks. sådan som dette notat vil præsentere dette emne*
- Programmerings–sprog, fortolkere og oversættere
- Operativ–systemer: Centrale og distribuerede
- Database–styre–systemer og databaser

[2] Appendiks B giver eksempler på beskrivelser af flere infra–struktur–komponenter.

- Data–kommunikations–systemer og data–protokoller
- Videns–baseret metode–lære og systemer (inkl. AI)
- Sandtids–systemer: Indlejrede mv.
- Kryptografi, sikre systemer, mv.
- *&c.*

Datalogien bygger på matematikken, i særdeleshed:

- Matematisk logik og meta–matematik
- Mængde–, funktions–, og relations–lære
- Algebra
- *&c.*

Og datalogien inddrager, i sit virke, andre matematiske discipliner:

- Operations–analyse
- Statistik
- Numerik
- *&c.*

1.2 Metode–lære

Fag–disciplinerne som er optalt først i Afsnit 1.1 kan enten studeres (udforskes, tilegnes (læres)) udfra en teoretisk vinkel: Hvilke teorier udgør de (hvad er muligt), eller udfra et metode–lære synspunkt. Typisk vil en rimelig moden, akademisk datalogi–uddannelse beskæftige sig med disse to områder i tids–forholdet 50/50 %: Lidt mere ingeniør–orienterede uddanelser vil sikkert ændre forholdet fra 50/50 % til 40/60 % eller endog 30/70 %. Men alene at fokusere, 100 %, på metode–lære er som at lære at regne uden at forstå den underliggende matematik. Dog, mener jeg, må man med skam melde at forholdet meget ofte er 10/90 %–0/100 % !

I dette notat skal vi fokusere på sådanne metode–lærer, der bygger meget direkte på et matematisk (dvs. teoretisk) fundament. Mao.: Der findes såkaldte metode–lærer, der ikke bygger på en (eksplicit erkendt) matematisk teori.

Ved en metode forstås her en række principper for udvælgelse og anvendelse, til både analyse og syntese, af teknikker og værktøj således at man, ingeniøren, effektivt kan udvikle et effektivt "apparat" (herunder programmel).

Den i dette notat skitserede metode–lære bygger på et matematisk fundament, udspringer fra cirka 30 års forskning og anvendelse i Europa, USA, Canada, Australien, m.fl.

Metode–lære–situationen idag kan bl.a. karakteriseres ved at der fra universiteterne udgår en stadig strøm af forsknings–resultater og kandidater der helst skulle bruges, hhv. bliver ansat

i en industri (herunder programmel–huse), der i endog meget stor udstrækning bruger "andre" metoder, i særdelsehed sådanne, som ikke bygger på et matematisk fundament, men som er udviklede af een–mands konsulent–firmaer og få andre sådanne (f.eks. er `UML` "udviklet" af det kommercielle firma `Rational`).

Dette notat anbefaler, forståeligvis, metoder, der, som alle andre ingeniør–grenes metoder (kalkyler), bygger på et teoretisk (for programmel: Matematisk) fundament.

1.3 Troværdig Programmel–udvikling

Problemet, sådan som det fremgår indirekte af de to sidste paragraffer i det foregående afsnit (Afsnit 1.2), er evnen til at sikre troværdigt programmel: Sådant som leverer den funktionalitet kunden forventer, som leveres til tid og indenfor oprindeligt budget, og som det er en fornøjelse at bruge, at udvikle, og at vedligeholde. Og meget mere.

Problement er, på troværdig vis, at påvise at programmel er korrekt mht. en lang række kriterier.

Dette notat udgår fra en "skole", der vedkender sig at mange former for programmel kan og derfor bør udvikles vha. matematiskt baserede metoder; fra en "skole", der objektivt kan påvise at programmel, der er udviklet vha. sådanne metoder, kan sikres en langt større troværdighed; og fra en "skole" der idag, efter godt 30 års virke, alt mere synes at "gribe om sig", at udbredes, mere–og–mere.

1.4 Infra–strukturer og deres Komponenter

Det er en hoved–tese i dette notat at sådanne troværdige programmel–udviklings–metoder idag muliggør udvikling af endog meget komplekse programmel–systemer til understøttelse af endog meget store infra–struktur–komponenter. Der bruges allerede idag programmel til hjælp i de daglige handlinger i sådanne infra–struktur–komponenter som f.eks. finans–, transport– (inklusive logistik–), sundheds–, handels– (herunder elektronisk handels–), fremstillings– (produktions–), service–, oa. sektorer. Men, påstås det nu, slet ikke (måske med finans–sektoren som en positiv undtagelse) med den konsekvens, dvs. i den udstrækning, som faktisk er muligt.

Mao., pga. "angst" for nok en "EDB–skandale", og stort set uvidende om hvad der faktisk idag er muligt, dog kun med altfor få programmel–leverandører, undgår ledelsen i potentielle informatik–forbrugende institutioner (og firmaer) at indforskrive sådant programmel — medens andre firmaer, meget naïvt (institutioner), "rask væk" indforskriver programmel uden at sikre sig at dets udvikling er troværdig — samtidig med at man ikke selv, til fulde, er forberedt.

Der er, mao., tale om et toægget sværd: På den ene side skal leverandøren kunne yde troærdigt programmel, og, på den anden side skal klienten (modtageren) være vel forberedt — især når det drejer sig om ("store") programmel–systemer til infra–struktur–komponenter.

1.5 Fra Domæne via Krav til Programmel

Den generelle (meta–) metode, vi her skal plædere for, foreskriver at klient og programmel– (i bredere forstand maskinel– plus programmel–) leverandør sammen sikrer at det leverede

system bygger på vel–gennemtænkte krav–definitioner — hvad der, i og for sig, muligvis ikke er så meget nyt, endsige noget revolutionerende, i — og (eller snarere: men) at disse krav–definitioner bygger på tilsvarende vel–gennemtænkte beskrivelser af anvendelses–domænet (herefter blot benævnt: 'domænet'). Det nye, som dette notat primært ønsker at berette om, er netop domæne–beskrivelses–konceptet. Visse vil påstå at *"det gør vi allerede"*, men jeg vil forlange, at en sådan viden om domænet skal være nøje dokumenteret, og det på en sådan vis at domæne–beskrivelsen overhovedet ikke nævner (dvs. inddrager) krav til evt. programmel — endsige antydninger af hvad dette programmel "gør" (for slet ikke at tale om hvorledes det er struktureret).

Notatet går i en vis dybde hermed.

1.6 Forandrings–ledelse: *"Business Process Re–engineering"*

Indførelse, tilstedeværelse, brug, af store programmel–systemer har det med at indvirke på, ja faktisk at forudsætte ofte drastiske ændringer i ledelsen og det daglige, øvrige operationelle beteende af den institution, den virksomhed, hvori programmel–systemet virker.

Det synes som om stort set alle, på nær måske de "aller–laveste", ledelses–niveauer ikke forstår dette tilfulde: De reagerer ihvertfald ikke korrekt på problematikken.

Vi vil, i dette notat, påstå, at én, blandt sikkert flere, forudsætninger for fornuftig forandrings–ledelse udgøres netop af det ovenfor omtalte par af dokumenter: Domæne–beskrivelse "plus" krav–definition: At med en sådan beskrivelse af hvorledes domænet er idag, og med en definition af hvorledes man ønsker det "i morgen", at dér er grundlaget tilstede for omhyggeligt at gennemtænke og at planlægge og siden at gennemføre de nødvendige ledelses og operationelle forandringer firmaet, institutionen, har behov for.

At en sådan forandrings–ledelse går hånd–i–hånd med både udviklingen af firmaet's, institutionen's, domæne–beskrivelse og udviklingen af krav–definition turde være en selvfølge: De to virker ind på hverandre.

2 Problemet

Store Danske programmel (og i bredere forstand: IT) systemer lever funktionelt ikke op til forventningerne (dvs. yder ikke de funktioner man troede de ville yde), deres udviklings–omkostninger overskrider oprindelige budgetter, udviser altfor mange såkaldte ikke–funktionelle problemer (fejl, lange reaktions– ("respons") tider, mm.), og/eller medfører/afslører store tilpasnings–problemer i de organisationer i hvilke de indføres.

I dette notat vil jeg forsøge skitsere én af de (sikkert flere nutidige) måder ved hvis brug man givetvis kunne undgå de værste excesser ved ovenstående problematik.

3 Infra–Strukturer

Appendiks B giver eksempler på beskrivelser af infra–struktur–komponenter.

3.1 En Socio–Økonomisk Indfaldsvinkel

Ifølge Verdens Banken er termen 'infra–struktur'[3] ≫en paraply–term for mange former for aktiviteter der, af visse udviklings–økonomer, forstås som *"social overhead capital"*. Begrebet infra–struktur omfatter aktiviteter som deler teknologiske og økonomiske egenskaber — såsom *"economies of scale and 'spill-overs' from users to non-users"*.≪

Følgende er eksempler på komponenter i et land's infra–struktur — nævnt i aldeles tilfældig rækkefølge:[4]

- Sundheds–væsenet (herefter: Sundheds–sektoren)[5]
- Transport– & Trafik–nettet [6]
- Telekommunikation[7]
- El/kraft–varme (inkl. naturgas) forsyning[8]
- Postomdeling[9]
- Undervisning[10]

3.2 En Bredere Forretnings og Aktivitetsvinkel

Almindelige Danske erhvervsvirksomheder, som f.eks. hoteller, restauranter, detail–forretninger, fremstillings–virksomheder, aviser, reklamebureauer, mmfl., opfatter f.eks. de ovenstående infra–struktur–komponenter som dele af Danmarks infra–struktur. Mere generelt, så synes én bestemt erhvervsgruppe at opfatte "alt andet, som omslutter det," på nær direkte leverandører og kunder, som tilhørende infra–strukturen.

[3] Winston Churchill citeres for i Underhuset, i 1946, at have sagt: ... *Den unge Labour–taler, som vi netop har lyttet til, ønsker klart at imponere sin valgkreds med det faktum at han har gået på Eton og Oxford siden han nu bruger sådanne moderne termer som 'infra–struktur'* ...

[4] Appendiks B giver eksempler på beskrivelser af flere af disse infra–struktur–komponenter.

[5] Med sådanne komponenter som f.eks. privat–læge–klinikker, tandlæge–klinikker, fysioterapeut– og kiropraktiker–klinikker, klinisk– oa. analyse–laboratorier, sundheds–sygeplejersker, hospitaler, apoteker, den farmaceutiske industri, syge–forsikrings–anstalter, sundheds–styrelsen, sundheds–ministeriet, oa.

[6] Med sådanne komponenter som f.eks. veje (inkl. broer, tuneller), jernbaner, havne og lufthavne — og med sådanne under–komponenter som f.eks. betalings–veje, –broer og –tuneller, bus–firmaer, DSB, Banestyrelsen, Metro, Taxi, mmm.

[7] Med sådanne komponenter som f.eks. Telestyrelsen, TDC, Sonofon, Telia, Orange, mmfl.

[8] Med sådanne komponenter som f.eks. Nesa, Energi 1–2, Vindmølleparker, DUC, DONG (med dets rørnet — hvor det nu ellers hører "hjemme"), mfl.

[9] Med sådanne komponenter som f.eks. Post Danmark [med dets mange under–komponenter], DHL, DPD, FedEx, UPS, m.fl.

[10] Med sådanne komponenter som f.eks. Børnehaver, Folke– og Friskoler, HT, Efterskoler, Gymnasier, Folke–højskoler, Handelsskoler, Seminarier, Universiteter, m.fl.

Vi skal anlægge, i vores betragtning af samspillet mellem begreberne infra–struktur og store IT, cum data & kommunikations–systemer, det bredere syn på begrebet infra–struktur: som omfattende, foruden ovenstående, også følgende "størrelser":

- Finans–sektoren[11]
- Distributions–sektoren[12]
- Turist–erhvervet[13]
- Underholdnings–industrien
- *&c.*

Vi skal, mao., anlægge et mere teknisk syn, og ser begrebet 'infra–strukturer' (det hele, komponenter, under– eller del–komponenter, etc.) som havende til formål at understøtte andre systemer og aktiviteter.[14]

3.3 Programmel (IT) til 'Infra–Strukturer'

Programmel, og i bredere forstand IT, til infra–strukturer er som regel distribueret (hhv. distribuerede), og er i vid udstrækning "beskæftiget" med at understøtte kommunikation af mennesker, materialer og information. Derfor er begreber som 'åbne systemer'[15], tidmæssig påpasselighed[16], sikkerhed[17], ubestikkelighed[18], og modstands–dygtighed[19] ofte meget vigtige "størrelser".

3.4 Diskussion

Jeg var i årene 1991–1997 FN Direktør. Da opbyggede jeg et Forsknings & 'Post–Doctoral' "Trænings" Center, UNU/IIST: UN University's International Institute for Software Technology, i Macau. Centret lever i bedste velgående. En vigtig ingrediens i centrets virke var at hjælpe regeringer, universiteter og ledende IT brugere og IT industrier, herunder programmel–huse — altsammen i udviklings–lande — "self–sufficience" og "self–reliance" mht. programmel til infra–strukturer. Dette er dokumenteret i bl.a. *UNU/IIST Annual Reports* [4, 9, 10, 15, 53], i mere generelle "brochurer" [3, 6, 7, 12, 17, 18], samt i rapporter, der specifikt omhandler infra–struktur emnet: [13, 11, 43] (sidste reference skal skrives til UNU/IIST's 10 års jubilæum's Symposium, Lissabon, 18–22 Marts 2002).

[11] Med sådanne komponenter som f.eks. banker, forsikring, børs, mæglere, finanstilsyn, mmfl.

[12] Med sådanne komponenter som f.eks. Logistik–firmaer, transport–firmaer (lastbil, godstog, air cargo, rederier, mv.), oplagrings–pladser, mv.

[13] Med sådanne komponenter som f.eks. hoteller, restauranter, mmfl.

[14] Vi minder om at Appendiks B giver eksempler på beskrivelser af flere infra–struktur–komponenter.

[15] *openness*

[16] rettidighed, *timeliness*

[17] *security*

[18] *lack of corruption*, det at f.eks. information "transporteres" uden at andre kan korrumpere denne information

[19] *resilience*

4 IT System Udvikling

Ved IT system udvikling forstås planlægning og konstruktion af et samlet system af maskinel[20] og programmel. Vi skal her alene fokusere på den mest styr– & kontrollér–bare af disse to størrelser: Programmellet.

4.0.0.1 Dogme I: Før man kan programmere, må man kende til, hvad det er programmellet skal kunne, dvs. man er nødt til først at udvikle en definition af kravene.

Visse dele af en krav–definition sikres (dvs. "implementeres") vha. maskinel og indkøbt programmel. Andre, dvs. stort set tilbageværende, dele programmeres. Dog:

4.0.0.2 Dogme II: Før man kan definere kravene, må man kende til anvendelses–området, dvs. man er nødt til først at udvikle en beskrivelse — en forståelse — af domænet.

Data–, cum informatik–ingeniøren, har, eller bør, i det mindste have en special–uddannelse indenfor programmering og almen krav–definition. Men vedkommende har, og kan, som regel ikke, som nybagte ingeniører, have en professionel baggrund i ethvert domæne. Men de bør have en general uddannelse indenfor almene principper og teknikker for indkredsning, forståelse og beskrivelse af stort set vilkårlige anvendelses–domæner.

- Der henvises allerede her til Afsnit 5 (startende på side 25). Dette senere afsnit er skrevet således at det kan, dvs. burde kunne, forstås "fritstående" fra det øvrige notat !
- Appendiks B giver, tilsvarende, en lang række eksempler på synopsis information omkring og råskitse–beskrivelser af adskillige anvendelses–domæner.

4.0.0.3 Dogme III: Dogmerne I og II skal ikke tages altfor dogmatisk, men ej heller altfor letfærdigt.

Hermed menes blot: Det kan iblandt være akceptabelt at påbegynde programmel–konstruktions– eller krav–definitions–arbejdet inden krav–definitions–, hhv. domæne–beskrivelses–arbejdet er fuldt afsluttet.[21]

Dog menes der også at når programmel–konstruktions–arbejdet er afsluttet da forventes også domæne–beskrivelses– og krav–definitions–arbejdet at være fuldt afsluttet.

4.0.0.4 Caveat: Dog så enkelt opfører dagens virkelighed sig ej. Derfor Afsnittene 4.1 (startende på side 15), 4.2 (startende på side 16) og 8.3 (startende på side 61).

4.1 Traditionel Programmel–udvikling

Vi analyserer tre facetter, α, β og γ, ved det vi vil forstå som den traditionelle programmel–udvikling.

[20] inklusive datamater og data kommunikations–udstyr
[21] Se Afsnit 8.3.

4.1.1 Programmerings–fasen (Kodning) — α

I traditionel programmel–udvikling kan det nok være at selve programmerings–, dvs. kodnings–fasen tages rimeligt alvorligt: Programmel–huse (*software houses*, generelt *IT + Software leverandører*) følger ofte rimeligt velordnede procedurer med henblik på at sikre en rimelig kvalitet — hvad der så ellers skal forstås herved, set i lyset af hvad vi siden påstår — herunder tilgængelig og sikker dokumentation af programmel–produktet og dets konstruktion. Programmerings–fasen leder til programmel, der opfylder en lang række tekniske normer: (i) Er korrekte mht. syntaksen af det (formelle) sprog (f.eks. `C++` eller `Java`), som programmerne er skrevne i; (ii) er vel–dokumenterede; (iii) har været underkastet en række afprøvninger, "tests", der kan reproducerers; mv.

4.1.2 Krav–definitions Fasen — β

Men det skorter ofte på samme 'stringens' når det kommer til krav–definitioner: Selvom der findes internationale standarder herfor opfylder kommercielle krav–definitioner ikke altid — for ikke at påstå: Sjældent — disse ! Situationen skyldes bl.a. at krav–definitioner nødvendigvis ikke kan danne et direkte grundlag for datamaskiners arbejde (*"execution"*) sådan som programmel gør det. Derfor er der ikke de samme tekniske værktøjer, der — som program–oversættere (*compilers*) med samt alle deres mulige kode–analysatorer — kan delvis afprøve de således, som ofte uformelt, formulerede krav–definitioner. Disse afprøvninger kunne forestilles udført med henblik på f.eks. relativ komplethed, konsistens, mv. Samtidig må det erkendes at den teknisk–videnskabelige viden om hvorledes man kan sikre rimeligt udtømmende krav–definitioner, at denne teknisk–videnskabelige viden, ikke endnu idag er forholdsvis så omfattende som den teknisk–videnskabelige viden om program–kodning.

4.1.3 Domæne Beskrivelse — γ

Og endelig må man nok ret så konsekvent erkende at selv eksisterende, gode krav–definitioner (i) ikke nuancerer mellem hvad der er krav og hvad der er mere eller mindre uforanderlige egenskaber ved anvendelses–domænet, (ii) og dermed også på hvilke antagelser krav–definitionerne bygger.

Mao. dagens programmel afspejler ofte en beklagelig mangel på affinitet til den aktuelle verden i hvilket det skal tjene. Samtidig må det være en internationalt kendt forsker tilladt at ytre den yderligere beklagelse at megen god Dansk forskning, som det selvfølgelig for det meste er tilfældet, ikke endnu er trængt ind de nødvendige steder. Herom mere i Afsnit 9 (startende på side 63).

4.2 Moderne Programmel–udvikling

Der undervises idag på universiteterne i Nord-europa, i Danmark, og på DTU, i sådanne metoder: Principper, teknikker og værktøj, der, når de anvendes efter deres hensigt, fra systematisk til rigorøst, har vist sig — og dette er dokumenteret gennem mange års publikationer fra især Europæiske forskere og praktiserende data–ingeniører — (har vist sig) at medføre klart mindre problematisk programmel: (i) Programmel, der klart bedre leverer de

funktioner brugerne og øvrige interessenter ønsker sig; (ii) programmel, der indeholder klart færre fejl; og (iii) programmel, der er udviklet indenfor oprindeligt estimerede budgetter.

Jeg skal i det følgende, i dette afsnit, Afsnit 4.2, og i følgende afsnit, Afsnittene 4.3, 5 (startende på side 25), og 6 (startende på side 35), alene beskæftige mig med de metoder, teknikker og værktøj, jeg dels forsker i, dels underviser i, og som mange af vore kandidater har fortrolighed med fra realistiske studie–tids–projekter: Midtvejs–, for– og eksamens–projekter.

4.2.1 Faser, Afsnit, og Trin af Udvikling

Princippet om *"del og hersk"*, på Engelsk: *"separation of concerns"* — et princip der er nødvendigt at følge om man skal kunne beherske "systemernes" kompleksitet — ligger bag opdelingen af programmel–udvikling i faser, disse i afsnit, og disse igen i trin.

4.2.1.1 Udviklings–Faser: Vi opfatter programmel–udvikling som bestående af følgende — hvad vi vil kalde — faser: Domæne–forståelse, krav–definition, og programmel–specifikation.

Om de bliver udført i streng, eller mindre streng rækkefølge, er — indtil videre, for denne fremstillings hovedformål, mindre væsentligt. Dog er det væsentlig, mener vi, at når et programmel–system er fuldt udviklet og kan leveres til rekvirenten, at da er al dokumentation vedrørende de tre faser ikke alene fuldt gennemført, men også, på rimelig overbevisende facon, kryds–relateret. Hvad der forstås ved kryds–relateret fremgår af det følgende.

4.2.1.2 Udviklings–Afsnit: Indenfor hver fase kan der være blot eet, det eneste, eller flere, udviklings–afsnit.

Typisk vil, f.eks., programmel– specifikations–fasen opdeles i f.eks. følgende afsnit: Design af system–, siden programmel–arkitektur; dernæst design, udfra disse, af programmel–komponent–organisation & –struktur; dernæst, typisk, et afsnit i hvilket en mere detaljeret modularisering finder sted; etcetera; afsluttende, hvor det er nødvendigt, i kodning (i et programmerings–sprog som f.eks. `Java` eller `C++`).

Afsnittene 5.2 (startende på side 25) og 5.3 (startende på side 26) detaljerer væsentlige afsnit indenfor domæne–forståelse, ligesom Afsnit 6 (startende på side 35) detaljerer væsentlige afsnit indenfor krav–definition, hhv. programmel–specifikation.

4.2.1.3 Udviklings–Trin: Indenfor hvert udviklings–afsnit kan der så, af mere programmerings–tekniske årsager, foretages en yderligere findeling af udviklingen i trin.

Typisk kan sådanne årsager være: Fra en abstrakt specifikation af hvad en programmel–procedure skal producere af resultater til en "snedig" algoritme og/eller data–struktur, der aktuelt realiserer en sådan procedure, kan der være indeholdt et *"heureka"*, der bedst kan forklares for, forstås af, andre, ved at "omsætte" dette *"heureka"* i ét eller flere udviklings–trin. Tilsvarende kan det, i overgange fra egenskabs–orientede domæne–beskrivelser og/eller krav–definitioner til følgende fase, eller afsnit, være formålstjenligt med ét eller flere mellem–trin.

4.2.2 Dokument–produktion

Fælles for alle faser, fælles for alle hovedafsnit indenfor disse faser, og fælles for de ofte flere trin indenfor visse af disse hovedafsnit, er, i de metoder jeg kan anbefale, at de kræver en omhyggelig, påpasselig og gennem–checket udvikling af dokumenter, og at disse faser, afsnit og trin af udvikling relateres nøje (herunder fra systematisk, via rigorøst til matematisk formelt — det sidste hvor det bedømmes nødvendigt at foretage en sådan, eventuel matematisk bevisbar, relatering).

Typisk omfatter dokumentationen, for hver fase, for hvert afsnit indenfor enhver fase, og for ethvert trin indenfor ethvert afsnit, en buket af komplementerende — herunder konkordante — dokumenter:

4.2.2.1 Informerende dokumenter: Information indfører læseren i problemstillingen for den enkelte fase, det enkelte afsnit eller trin.

Informationen er ofte af pragmatisk art.

Den hjælper ikke til at forstå domænet. Den meddeler ikke bestemte krav (eller design), men bringer en kontekst i hvilken relevansen af senere fremførte krav måske bedre kan fornemmes.

Blandt informative dokumenter kan man forestille sig flg.:

- `Problem-oplæg / Synopsis` — *Design Briefs*
- `Indledning til Udbuds-materiale` — *Introduction to Tender*
- `Indledning til Udbuds-svar-materiale` — *Introduction to Bid*
- `Kontrakt`
- *&c.*

4.2.2.2 Beskrivende dokumenter:

- Uformelle dokumenter — dvs. national/fag–sproglige dokumenter:
  - `Råskitser` (*Rough Sketches*) — som ikke forestilles at udgøre en del af kunde–leverancer, men som er vigtige, for leverandøren (ie. udviklerne), at udtrykke, for derigennem at indkredse problematikken for den pågældende fase, det pågældende afsnit eller trin. Udfra råskitser, vha. analyse, se nedenfor, identificeres begreber, "afsløres" eventuelle inkonsistenser, mmm.

    De i Appendiks B bragte beskrivelser har for det meste karakter af at være råskitser.
  - `Fortælling:` (*"narrative"*) En regelret, systematisk beskrivelse af genstandsområdet (domænet, eller definition af kravene, eller programmel–specifikationen).

    Vi skal have meget mere at sige om *"narratives"* (fortællinger) siden.

    Eksempel 4 på side 27 illusterer en "stram" fortælling.

- `Terminologi:` En systematisk optælling og forkalring af alle de termer, der er specielle for genstands–området, dvs. som ikke umiddelbart kan forventes forstået af alle: Både alle interessent–, herunder bruger–, grupper og af leverandørens informatik–ingeniører.[22]

- Formelle dokumenter: Dette er det springende punkt i "den nye metode": Matematisk præcise domæne–beskrivelser, krav–definitioner og programmel–specifikationer.

  Man kan ikke forvente at kunden, rekvirenten, kan forstå disse "formler". Og det skal kunden da ej heller. Derfor *Narratives* og Terminologi. Det er dem kunden "skriver under på".

4.2.2.2.1 Opmands–problematikken: For visse projekter, f.eks. særligt store, "avancerede", komplekse, sikkerheds–kritiske, mmm., kan det dog være nødvendigt for kunden at kunne følge med i de formelle dokumenter: Deres forvandling fra domæne–beskrivelser via krav–definition til programmel–specifikation. I sådanne tilfælde skal rekvirenten kunne alliere sig med et uafhængigt firma der "så–at–sige" kan "det med det formelle" ! Sådanne firmaer kender vi, f.eks., fra skibsbygnings–enterpriser oa. Dvs. fra kontraktlige forhold vedrørende planlægning, design og konstruktion af skibe, fly, nukleare kraft–værker, mv. Her betinger det forhold at den færdige konstruktion (skibet, flyet, atom kraft–værket), skal forsikres, at opmands–firmaer nøje efterprøver planlægning, design og konstruktion. Dette medfører allerede i dag en meget nøje ekstern vurdering også af de formelle, de matematiske, dokumenter. Forsikrings–selskaberne ønsker derfor at bringe den "allerhøjeste" pågældende udviklings–ekspertise ind "på deres side" — som opmand.

4.2.2.2.2 Opmands–firmaer: Eksempler på firmaer der leverer den ønskede opmands ekspertise er Lloyd's Register of Shipping, Bureau Veritas, Norwegian Veritas, TÜV, oa. Disse respekterede selskaber udbygger i disse år deres ekspertise til bredt og dybt, dvs. på højeste professionelle niveau, at beherske netop de formelle metoder, der omtales som primære i dette notat.

I Danmark repræsenterer IFAD (Institut for Anvendt Datalogi, Odense) et bedste Dansk eksempel på et seriøst opmands–firma.

4.2.2.3 Analyserende dokumenter:

- `Validering:` Validering er en process — og et dokument — der relaterer beskrivelses–dokumenter til rekvirenter, dvs. til almindelige mennesker. En validering kan således kun uformelt sikre at en beskrivelse udsiger det man ønsker den skal udsige. Der er således typisk to former for validering:

  - En indre, intern validering — at en formel beskrivelse "svarer" til en fortællende beskrivelse (en *narrative*). En intern validering finder, i det mindste, sted internt hos leverandøren, og, i visse tilfælde imellem leverandør (udvikler) og rekvirentens opmand.

[22] Dette notat bringer, i nuværende November 2001 version, ikke en sådan, illustrerende, Terminologi.

  - En ydre, ekstern validering — at en uformel, dvs. en fortællende beskrivelse (en *narrative*) modsvarer rekvirentens forestillinger.

  Validerings–processen er en uhyre vigtig process.

- `Egenskabs-verifikation:` Mange domæne–, krav–, og programmel–egenskaber er af en sådan natur, at det kan blive nødvendigt at sikre sig, hvad der kan svare til et matematisk bevis for, at den foreliggende beskrivelse opfylder den pågældende egenskab. Hertil tjener egenskabs–verifikations–processen (og det resulterende dokument).

- `Verifikation af udviklings-trin:` Tilsvarende for fase–, afsnit– eller trin–relaterede beskrivelser: Korrekthed af udvikling kan tilsvarende lede til et matematisk bevis.

- `Slagord:`[23]

  - Validering tjener til at sikre at rekvirenten får det rigtige produkt.
  - Verifikation tjener til at sikre at produktet er rigtigt.

Man kan også, istedetfor at tale om dokumenter, som for de tre klasser ovenfor, og disses under–klasser, tale om dokument–dele: Disse er så mere indflettede med hverandre i hvad der kan opfattes som enkelt–dokumenter. Dog bør der klart sondres mellem arten (de tre klasser og de flere under–klasser) af de enkelte dokument–dele.

4.2.3 Diskussion

Eet af de "nye" aspekter, ved den her fremførte, og i mange henseender vidt udbredte metode, er således sikringen af at udviklingen både har et uformelt, men præcist, såvelsom et formelt grundlag – og at disse hviler på en "finkornet" opfattelse af nødvendig dokumentation — herunder en tilsvarende finkornet udviklings–process.

4.3 En System–udviklings "Triptych"

Eet andet af de "nye" aspekter ved den her fremførte, og i stedse stigende grad anerkendte metode er sikringen af at udviklingen bygger på en klar opfattelse af domænet.

Jeg gentager fra indledende afsnit i Afsnit 4 (startende på side 15):

4.3.0.1 Dogme I — Gentaget: Før man kan programmere, må man kende til, hvad det er programmellet skal kunne, dvs. man er nødt til først at udvikle en definition af kravene.

4.3.0.2 Dogme II — Gentaget: Før man kan definere kravene, må man kende til anvendelses–området, dvs. man er nødt til først at udvikle en beskrivelse af domænet.

Konsekvensen heraf er "nedlagt" i flg. Triptych:

- **Programmel–udvikling =**

  Domæne–forståelse Se Afsnit 4.3.2 (startende på side 21)

[23] Barry W. Boehm: Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ., USA (1981)

⊕

Krav–definition Se Afsnit 4.3.3 (startende på side 22)

⊕

Program–specifikation Se Afsnit 4.3.4 (startende på side 23)

Mao.: Hvor der "før" var to hoved–faser i det data–ingeniørmæssige arbejde omkring udvikling af programmel (dvs. system), er der nu tre i (det vi nu vil kalde) det informatik–ingeniørmæssige arbejde.

4.3.1 Leverandør og Rekvirent: Informatik–ingeniør–firma, hhv. Kunde

Vi antager i det følgende, at der i alle trin af udvikling, fra fase til fase, kan identificeres et klart kontraktligt forhold: En kontrakt og kontraktens partnere: Leverandør og rekvirent. Førstnævnte er her primært repræsenteret ved informatik–ingeniører.

De kontraktlige forhold ytrer sig, typisk, som følger:

- *Domæne–beskrivelse:* Mellem rekvirent og leverandør
- *Krav–definition:* Mellem rekvirent og leverandør
- *Program–specifikation:* Mellem leverandørens krav–definitions–udarbejdere og leverandørens program–konstruktører (dvs. to, eventuelt sammenfaldende grupper informatik–ingeniører).

4.3.2 Domæne–forståelse

Ved domæne–forståelse forstår vi her [a] tilegnelse af anvendelses–domæne–viden, [b] en analyse af den indhentede information: Klargøring, identifikation og afklaring af domænets begrebs–dannelser, [c] en terminologisering af domænets begreber, [d] en domæne–beskrivelse, og [e] validering (godkendelse) af domæne–terminologi og domæne–beskrivelse.

- Der sigtes mao. her til at det er informatik–ingeniørerne, der skal sikre sig at de har en rimelig forståelse af kundens, hhv. kundernes domæne.
- Forståelsen sikres gennem et snævert samarbejde mellem informatik–ingeniørerne og domænets interessenter, inklusive de brugere for hvem et programmel–system er påtænkt — dvs. de, der i dagligdagen skal benytte sig af et eventuelt leveret system.
- Og dette før de samme informatik–ingeniører og domæne–interessenter primært repræsenteret ved kunden: Den, eller de, der rekvirerer det påtænkte programmel–system — går igang med krav–definitions–fasen.

4.3.2.1 [a] Tilegnelse af anvendelses–domæne–viden: De forskellige interesse–grupper (*"stake–holders"*), der på den ene eller anden måde er involverede i domænet og som kan have en interesse i effekten af de programmel–systemer der måtte påtænkes — disse interesse–grupper — samt aktuelle, påtænkte brugere, *identificeres*. De *udspørges* om hvorledes de opfatter deres eget forhold til domænet: Dets "fænomener", deres arbejds–opgaver, de "størrelser (de mennesker, de materialer og den information mv.) de arbejder med, og de ønskelige og ikke–ønskelige effekter arbejde hermed adstedkommer — mmm.

4.3.2.2 [b] Analyse — Klargøring, Identifikation og Afklaring af et Domæne's Begrebs–dannelser: Den indsamlede viden bearbejdes: Mere eller mindre abstrakte *begrebsdannelser* — f.eks. også sådanne, der måtte være fælles for ellers ikke–erkendte forskellige situationer — *identificeres*. *Inkonsistenser* i indsamlet viden afdækkes og "fjernes" (resolveres).

4.3.2.3 [c] Terminologisering: Udfra afklarede begreber opstilles en terminologi. Denne dækker alle de termer, der betragtes som værende fag–termer — sådanne som ikke har en fag–specifik tyd i den almindelige borger's, herunder informatik–ingeniør's, dagligdag. Terminologien "definerer": indkredser, beskriver, kommenterer, og diskuterer, disse termer.

En således tidligt i programmel–udviklingen etableret terminologi skal hele tiden, i hele programmel–udviklings–processen bruges og vedligeholdes (dvs. opdateres). Brug af eventuelt ændrede termer spores til alle relevante dokumenter og deres brug bør så rimeligvis revideres.

4.3.2.4 [d] Domæne–beskrivelse: En domæne–beskrivelse består i det mindste af en naturligt–sproget (Dansk, Engelsk, ...) beskrivelse af enhver af alle de fænomener der kan tillægges domænet: "størrelser", disses sammensætning, deres observérbare egenskaber, begivenheder, forandringer som flg. af ude– eller indefra kommende "operationer" mv., "opførsler", mm.

En beskrivelse beskriver noget. En god beskrivelse blander ikke tingene sammen.

Man kan tale om det syntaktiske og om det semantiske i en beskrivelse: Hvorledes manifesterer "størrelserne" sig (syntaktisk) og hvad står de egentlig for (hensigt, mål — i snæver, semantisk, forstand).

Dertil kommer et vigtigt, ikke et beskrivelses–, men et informations–, forklarings–mæssigt forhold: Det pragmatiske. Hvorfor, i bredere forstand, fokuserer vi, i beskrivelserne, på de udvalgte "størrelser", hvorfor udelades andre ?

4.3.2.5 [e] Validering cum Godkendelse af Terminologi + Domæne–beskrivelse: Endelig, som led i den itererede domæne–forståelses–process, sikres det, at de forskellige interessent– og bruger–grupper forstår, og at de godkender, den opstillede terminologi og den opstillede domæne–beskrivelse.

Eventuelle indsigelser medfører, principielt, en total iteration over alle områderne [a–e].

• • •

Afsnit 5 (startende på side 25) går i teknisk detalje.

4.3.3 Krav Definition

Med en rimeligt udtømmende domæne–beskrivelse "bag sig", kan man nu påbegynde arbejdet med at opstille krav–definitionerne. Igen, som ved opstillingen af en domæne–forståelse, er der tale om et kontraktligt forhold mellem rekvirent og leverandør.

Ved krav–definition, som process, forstår vi her (a) tilegnelse (indhentning, "hjemtagning") af krav til ønsket programmel, (b) en analyse af den indhentede information: klargøring, identifikation og afklaring af kravenes eventuelle nye begrebs–dannelser, (c) en terminologisering af disse begreber, (d) en krav–definition (som dokument), og (e) validering (godkendelse) af krav terminologi og krav–definitions dokumentet.

Forholdene (a–e) svarer "mangt og meget" til emnerne [a–e] i Afsnit 4.3.2 (startende på side 21).

• • •

Afsnit 6.1 (startende på side 35) går i teknisk detalje.

4.3.4 Program Specifikation

Fra krav–definition udvikles programmellet. Vi skal ikke her, men i Afsnit 6.2 (startende på side 54), komme ind på program–specifikations–metodikker.

5 Domæne–beskrivelser

Dette afsnit kan, og bør vel, læses før læsning af flere tidligere afsnit — sådan som det også tidligere er antydet.

Ved en domæne–beskrivelse forstås her en beskrivelse af et praktisk, ikke nødvendigvis IT–relateret, genstands–område. Eksempler herpå kunne være

- Sundheds–sektoren, "det hele", eller store dele heraf
- Finans–sektoren, "det hele", eller store dele heraf
- Transport–sektoren, "det hele", eller store dele heraf
- Handels–sektoren, "det hele", eller store dele heraf

Disses domæne–beskrivelse, som process, kan foregå uden relation, og som resulterende dokumentation forblive ikke–relateret, til eventuelle ønsker om krav til muligt programmel der kan understøtte, herunder "erstatte", funktioner (nuværende arbejds–processer) i domænet. Der henvises til Appendiks B.

5.1 Generelt

En domæne–beskrivelse er således, idéelt set, en beskrivelse af domænet sådan som det er, ikke som det ønskes, og ihvertfald uden referencer til eventuelt ønsket programmel.

5.2 Bruger– og Interessent–grupper

Som den første altivitet ved opstillingen af en domæne–beskrivelse, er identifikationen af alle de bruger– og, mere generelt, de interessent–grupper, mennesker, institutioner, mv., der på den ene eller anden måde *"har et ord at skulle have sagt"*, og som kan forventes, før eller siden at sige dette eller disse ord !

∘ Eksempel 1 ∘

1. Sundhedssektor Interessenter: *(0) Raske og syge borgere, (1) syge borgeres evt. pårørende, (2) familie–, dvs. (privat) praktiserende læger (inkl. disses stab), (3) sundheds-sygeplejersker, (4) klinisk analyse laboratorier (inkl. disses stab, udstyr, mm.), (5) fysio–, kiropraktor– og andre revaliderings klinikker (inkl. disses stab, udstyr, mm.), (6) apoteker (inkl. disses stab, varelager, mm.), (7) hospitaler (inkl. disses stab, afdelinger, laboratorier, mmm.), (8) sygeforsikrings–institutioner (inkl. disses stab), (9) den farmaceutiske industri (inkl. disses stab mv.), (10) lægemiddel–styrelsen (inkl. dennes stab), (11) Statens Serum–institut (inkl. dennes stab), (12) Sundhedsstyrelsen (inkl. dennes stab), (13) Sundhedsministeriet (inkl. dettes stab), (14) pressen (inkl. disses stab), (15) politikerne, (16) "the public at large".*

[Slut på Eksempel 1]

Listen i eksemplet ovenfor er ikke nødvendigvis udtømmende.

Ved opstilling af listen over bruger– og interessent–grupper identificeres yderligere (ved navn og adresse, tlf. nr., mm.) de bestemte personer og institutioner man ønsker — man måtte ønske — at inddrage i domæne–beskrivelses–processen og i valideringen af det beskrevne.

Men hvad de enkelte bruger– og interessent–grupper har at meddele nedskrives først senere.

5.3 Domæne–facetter

Der er mange måder at "tackle" domæne–beskrivelses–processen på, og at opdele, strukturere domæne–beskrivelsen. Vi skal blot antyde én af koordinaterne i sådanne. Vi vil kalde denne koordinat 'domæne–facet–koordinaten'[24].

Vi fremhæver blot fem facetter ved domæne–facet–koordinaten:

- det basale,
- understøttende "teknologier",
- ledelse & organisation,
- almene & retlige forskrifter, og
- menneskelig opførsel.

Vi skal kort karakterisere disse facetter for at læseren derigennem kan fornemme hvad det er der forstås ved en domæne–beskrivelse.

5.3.1 Det Basale

Der er som regel mange basale facetter i ethvert domæne. En egenskab ved et domæne udgør et basalt facet hvis denne egenskab går igen, "bruges" i to eller flere af de andre facetter.

∘ EKSEMPEL 2 ∘

2. Patienters / Patienten's Sundhedstilstand: *Det må være klart at borgeren qua potentiel eller aktuel patient udgør en basal egenskab ved sundheds-sektoren. Følgende regnes for basale egenskaber ved en borger: (i) den øjeblikkelige sundhedstilstand, (ii) den potentielt livstids–akkumulerede viden om borgerens tidligere sundhedstilstande, (iii) den tilsvarende sygdoms–historie: En beskrivelse af "alle" de interaktioner patienten har haft, eller pt. har, med autoriserede, såvelsom ikke–autoriserede sundheds–arbejdere, hvilke "annamneser" der blev "udtalt" (eller antaget), (lægelige oa.) undersøgelser (inkl. prøver), diagnostikker, behandlingsforløb og disses effekt, mmm., (iv) de aktuelle interaktioner, nul, én eller flere, med sundheds–sektoren: Besøg hos lægen, indkøb af medicin, gennemførsel af diverse terapier, mmm.* [SLUT PÅ EKSEMPEL 2]

[24] Andre koordinater omhandler domæne–attributter som f.eks. (i) statiske og dynamiske [og under dynamiske, de passive (*inert*), de aktive, og de reaktive], (ii) tidslige og rumlige [hhv. temporale og spatielle], (iii) hierarkiske og kompositionelle [*"top down"* og *"bottom up"*], (iv) konfigurationelle [kontekster og tilstande], (v) dimensionalitet, (vi) manifestérbarhed [(a) sanselige: (berøring, lugt, smag, syn, lyd), kontra (b) ikke–sanselige ((b′ apparat–sanselige [elektriske, kemiske, mv.], b″ begrebs–modellérbare (matematiske og logiske), hhv. b‴ emotionelle oa.)]. Vi vil ikke nærmere berøre disse beskrivelses–koordinater udover at påpege at det blotte spektrum af disse, deres formålstjenlige anvendelse, og de mange teknikker deres korrekte anvendelse kræver, at alt dette klart viser at der skal professionelt, og ret så seriøst langvarigt uddannede informatik–ingeniører til troværdigt at bestride jobbet !

Andre basale forhold kan beskrives, sikkert (mindst) én for hver bruger– og, mere generelt, interessent-gruppe.

Det er vigtigt at den i Eksempel 2 udarbejdede beskrivelse — hvis elementer blot blev antydet i eksemplet — er basalt udtømmende. Dvs., det er nok begrebs–mæssigt abstrakt, men alligevel præcist, tilladende en række "lige gode" tilsigtede fortolkninger.

∘ EKSEMPEL 3 ∘

3. Patient–Journaler / Journalen: *Ved en patient–journal skal der her forstås, ikke blot de journaler, de dokumenter, der udarbejdes af sundheds–fagligt personale (læger, mmfl.), men også "al" anden viden, udtalt eller uudtalt, umiddelbart tilgængeligt såvelsom ikke (umiddelbart) tilgængelig, om patientens sundheds–tilstand, historisk og aktuel (hvad patienten måske godt ved, men ikke siger; hvad tante Agathe husker [at man havde mæslinger som barn], mmm.). Mao. en patient–journal er enten registreret eller ikke–registreret. Registrerede patient–journaler består af en stamdel og en lang række forskellige oplysnings-dele (–ark). Disse indeholder eventuelt særskilte oplysninger om annamneser, analyser (undersøgelser [muskel-reflektioner, pupil-reaktioner, blod–, urin–, cholesterol–, oma. –prøver]), diagnostik, behandlings-planer, disse planers iværksættelser og reaktion herpå (observationer heraf), overførsler til anden behandling, udskrivning, mv.*[25]*. Dernæst beskrives "alle" de forhold i hvilket en sådan registreret patient–journal indgår: Oprettelse, udfyldning, læsning, som "generator" for henvisninger (til undersøgelse, terapi, operation, mmm.), osv. Mao. patient–journalens "vandring" i systemet beskrives: Den mere eller mindre fuldstændige (dvs. partielle) udlevering af oplysninger fra én afdeling's patient–journal til en anden afdeling, eller til anden myndighed. Osv.* Dernæst beskrives de ikke–registrerede former for patient–oplysninger af sundheds–faglig art. Vi udelader dog dette i nærværende eksempel. [SLUT PÅ EKSEMPEL 3]

Grunden til at man, som i Eksempel 3, ønsker også at beskrive sådanne egenskaber ved domænet som ikke umiddelbart dækker over alment erkendte forhold, er at man før eller siden måtte ønske at inddrage sådanne forhold i fremtidige informatik–systemer. Man kan jo tale om dem, om de så er nok så flygtige — og de kan være flygtige fordi vi ikke gør os den umage at "nedskrive" det, f.eks. fordi vi ikke mener at have en passende, understøttende teknologi til at nedskrive — optegne, registrere — disse flygtige forhold.

Mao. vi ser at allerede i den basale domæne–beskrivelse stilles beskriverne, herunder interessent– og bruger–grupperne overfor et valg: At inddrage visse forhold, og at udelade andre forhold fra domæne–beskrivelserne.

Aht. senere eksempler anfører vi et eksempel fra transport–infrastrukturen:

∘ EKSEMPEL 4 ∘

4. Jernbane–net: *(1) Et 'jernbane net' består af to eller flere 'stationer' og én eller flere 'linjer'. (2–3) Stationer og linjer består af én eller flere 'spor–enheder'. (4–7) En spor–enhed er enten en 'lineær' spor–enhed (et par af parallelt forløbende lige, eller kurvede, skinner), eller er et 'simpelt sporskifte', eller er et 'simpelt sporkryds', eller er et 'sporskifte sporkryds',*

[25] Herefter følger så en mere detaljeret beskrivele af de ovenfor nævnte "størrelser": Hvori en annamnese, en analyse, en diagnostik, en behandlings-plan, mv., består, etc. Men beskrivelsen bør, for Guds skyld, ikke gå i syntaktiske detaljer — såsom de forskellige felters struktur og placering, for sådanne felter der måtte være i et ark der skal afkrydses og udfyldes, f.eks., ved en blodprøve.

etc. (8) Enhver spor–enhed består af to eller flere 'konnektorer' (forbindelser). (9–12) En lineær spor–enhed består af to 'unikke' konnektorer. Et simpelt sporskifte består af tre unikke konnektorer. Både simple sporkryds og sporskifte sporkryds består af fire unikke konnektorer. (13) For enhver bestemt, dvs. unik, konnektor er der højst to spor–enheder som "deler" denne konnektor (dvs. er "samlede" (forbundne) vha. denne konnektor). (14) En 'vej' gennem en spor–enhed kan opfattes som et ordnet par af konnektorer. Lad k, k' stå for en lineær spor–enheds' to distinkte konnektorer, dvs. k er forskellig fra k'. Da er parret (k, k') én af vejene i den lineære spor–enhed – og (k', k) den anden af de mulige veje. Lad k, k', k'' stå for et sporskifte's tre distinkte konnektorer, hvor vi opfatter k som den konnektor udfra hvilken enten k' eller k'' kan "nås" afhængig af sporskiftets indstilling, da er vejene (k, k'), (k, k''), (k', k), og (k'', k) de fire veje der findes i et sporskifte, dvs. som et sporskifte kan "give anledning" til. *(15) En 'rute' er en følge af veje gennem forbundne spor–enheder, hvor hver enkelt vej er vej i en (pågældende) spor-enhed, og vejens to konnektorer indgår i deres "forbundethed". (16) En spor–enheds øjeblikkelige 'tilstand' er en mængde af nul, én eller flere veje sådan som den pågældende spor-enhed definerer disse veje. (17) En 'åben rute' er en rute hvis enkelte veje er i de pågældende spor–enheders' øjeblikkelige tilstand. (18) En spor–enheds tilstands–rum er mængden af alle de tilstande enheden kan antage.* For et sporskifte kan et tilstandsrum maksimalt "udspænde" flg. tilstande (hvor vi til illustration antager at vejen (k, k'') er en lige vej): $\{\}$, $\{(k, k')\}$, $\{(k', k)\}$, $\{(k, k'')\}$, $\{(k'', k)\}$, $\{(k, k'), (k', k)\}$, $\{(k, k''), (k'', k)\}$, $\{(k, k''), (k'', k), (k', k)\}$, $\{(k', k), (k'', k)\}$. *(19) En linje bestå alene af forbundne lineære enheder. (20) En linje forbinder eksakt to forskellige stationer. (21) To vilkårligt forskellige stationer, to vilkårligt forskellige linjer, og vilkårligt udvalgte par af én station og én linje deler ikke spor–enheder.*

[Slut på Eksempel 4]

Bemærk at vi, i den ovenstående beskrivelse af det basale i et(hvert) jernbane–net, har abstraheret, også i den Dansk–sprogede beskrivelse, fra spor–enheders konkrete topologi: Deres terræn–koordinater, svaj i kurver, stigning, fald, om de går over broer, langs perroner, gennem tuneller. Slige forhold skal der blive rig anledning at beskæftige sig med siden.

De første og de fleste, om ikke alle følgende eksempler vil ikke være udtrykt med den stringens vi virkelig fordrer. Det just overståede jernbane—net eksempel (Eksempel 4) illustrerer den nødvendige stringens. Vi henviser til diskussionen i Afsnit 5.4 (startende på side 32).

5.3.2 Understøttende Teknologier

Med understøttende teknologi forstås ikke den informations teknologi (IT) og den deri indeholdte informatik, der som regel er målet for den domæne–forståelse hvoraf domæne–beskrivelsen er et led. Istedet forstås ethvert sådant menneskeligt, eller mekanisk, eller elektronisk, herunder datamat og data–kommunikations udstyr (inkl. fax, telefon, E-post, EDI–Fact, mv.), som allerede i et nuværende domæne understøtter dets funktioner.

∘ Eksempel 5 ∘

5. Sporskifter i et Jernbane–net: *Forskellige teknologier understøtter skiftning af et sporskifte: (1) En stærk sporskifte–arbejder trækker i diverse, evt. med lodder forsynede stænger indenfor en lille meters afstand af spor–skiftet, (2) Spor–skiftet er forbundet, over en kortere afstand, af typisk maksimalt 200 meter, vha. stålkabler, til en moment–forstærkende mekanisk*

arm. (3) Fra et station–centralt placeret tårn styres spor–skifterne: I tårnet aktiveres, vha. knapper forbundne til elektriske kabler, elektro–mekaniske motorer, der kan være tæt placeret ved spor-skifterne, og som så afstedkommer skiftningen. (4) Endelig kan en udvidet sådan anordning vha. ét knaptryk udløse skiftning af større ruter af spor-skifter (såkaldt `interlock`*).* Beskrivning af udvalgte, relevante af disse teknologisk understøttede spor--skifter følger: Minimal og maksimal mekanisk påvirkning, reaktions– og skifte-tider, arten af elektroniske signaler, pålideligheds–faktorer (snevejr, kulde, mv.), statistiske fejlsikkerheds–tal, mmm. [Slut på Eksempel 5]

Vi så i ovenstående eksempel hvorledes en "understøttende teknologi" kunne udgøres af et menneske, i én version, af noget mekanisk, i en anden version, osv. I Eksempel 6 skal vi se at fortrykte formularer kan siges at repræsentere en understøttende teknologi.

∘ Eksempel 6 ∘

6. Sundheds–faglige Formularer mv.: Baggrunden for nærværende eksempel er den situation hvor en patient får taget en blodprøve, egentligt en række sådanne, med henblik på disses analyse mht. sukker indhold, hæmoglobin, cholesterol (af den ene eller anden art), mmm. *Til hjælp for lægen i henvisninger til blodprøve, til hjælp for sygeplejersken, der skal tage blodprøver (hvor mange), og til hjælp for analyse–laboratoriet, der skal foretage de egentlige analyser på grundlag af indleveret blod, findes der én eller flere relaterede, fortrykte formularer (A, B, ..., C). Disse formularer forsyner lægen (eller en lægesekretær eller en sygeplejerske) med den adspurgte, obligatoriske information (patientens navn, dato og tid, lægens navn, adresse, tlf.nr., etc.), hvorefter lægen krydser af — og evt. forsyner med kort-fattede meddelelser — én eller flere af de fortrykte analyse–felter. En strimmel–skriver bruges til at generere et passende antal éntydigt identificerende og evt. patient–navngive etiketter. På grundlag heraf tager lægen selv, eller dennes sygeplejerske, én eller flere blod–prøver (ned) i reagensglas–lignende beholdere, som hver forsynes med én af de strimmel–skriver–genererede etiketter. Analyse–laboratoriet foretager nu de udbedte analyser og tilbagesender de afmærkede formularer, nu forsynede med adspurgte måleresultater.* Herefter beskrives hver af formularerne A, B, ..., C, i detalje — abstrakt syntaktisk, snarere end konkret syntaktisk (hvad angår milimeter–mæssig formatering). [Slut på Eksempel 6]

Nok et eksempel:

∘ Eksempel 7 ∘

7. Klinisk–analyse Instrumenter: *En elektro–kardiogram maskine fungerer som følger: Patienten forsynes med elektroder (som er forbundne til maskinen):* Herefter flgr. en beskrivelse af den tilnærmelsesvist korrekte placering af typisk en 8–10 elektroder efter en forudgående klargøring (vask, "spritning") af hvert målepunkt. *Når patienten skønnes at være "faldet til ro" startes maskinen, og én eller flere serie målinger foretages, hver serie med forskellige følsomheds–indstillinger (evt. justeringer). De udskrivne EKG strimler hæftes sammen idet dog hver forsynes med én på forhånd fortrykt etikette med ...* Flere detaljer (mht. målingskriterier, indstillinger, og heraf flg. fortolkning af EKG strimler) beskrives ligeledes. [Slut på Eksempel 7]

Beskrivelser af understøttende teknologi–aspekter tjener, dels til at komplettere forståelsen af domænet, dels til at sikre at sikre at eventuelle datamatiske "forbedringer" (udvidelser, "erstatninger") af de teknologiske understøttelser tilpasses disses omgivelser.

5.3.3 Ledelse & Organisation

Ledelse er defineret i forhold til de, der bliver ledet, og til dem til hvilke lederne rapporterer. En leder udstikker generelle retnings–linjer for sædvanlige arbejds–processer, sådan som de ledede forventes at udføre dem, samt reagerer på henvendelser fra de ledede. De sidste henvendelser rapporterer typisk om sådanne forhold hvor baggrunden for de generelle retnings–linjer for sædvanlige arbejds–processer ikke er tilstede. Mao.: Ledere dikterer ("direkterer") normal–forhold "nedad" i firma–"pyramiden", hvor de ledede rapporterer unormale forhold "op ad" i samme pyramide. Veje for 'direktion', henholdsvis 'rapportering', fastlægges ved organisations–strukturen. Eksempler på sådanne er f.eks. de hierarkiske og matrix–strukturerne.

En beskrivelse af et domænes ledelses– & organisations–struktur har til formål at sikre, dels domæne–forståelsen, dels at eventuelle krav til understøttelsen af ledelses–, hhv. tilbage––rapporterings–processen, formuleres med reel affinitet til virkeligheden.

∘ EKSEMPEL 8 ∘

8. Patient, Familielæge, Apotek, Hospital, mv.: *For patienten er der flg. organisatoriske struktur at indrette sig efter: Når man er syg går man til sin private, praktiserende læge. Lægen kan udstede en recept, og det betinger at patienten selv henter den foreskrevne medicin (og at man tager denne efter forskriften). Skulle lægen henvise til visitation på et nærmere angivet hospital, da forventes patienten at følge en skrivelse herom fra hospitalet: Møde op på et meddelt tidspunkt, osv. ...* [SLUT PÅ EKSEMPEL 8]

I domænet er alt muligt — som vi skal se i Afsnit 5.3.5 (startende på side 32). Enhver ledelses– & organisations–mæssig struktur kan omgås. Derfor skal man i første omgang ikke, i domæne–beskrivelserne, fastlægge altfor "stramme" bekrivelser af ledelses– & organisations––strukturen. Dog skal det være muligt at beskrive visse leder/arbejder–interaktioner som værende ønskelige, andre som værende uhensigtsmæssige, etc. Det bliver mao. først, når man skal beskrive krav til en datamatiseret understøttelse af leder/arbejder–interaktioner, at man kan pålægge at systemet detekterer og så vidt muligt sikrer 'korrekt' brug af ledelses– & organisations–strukturen.

5.3.4 Almene og "Retlige" Forskrifter

De basale (m.fl.) egenskaber ved et domæne danner grundlaget for menneskelig aktivitet. Men disses basale egenskaber, de understøttende teknologier, den "herskende" ledelses– & organisations–struktur er, set i isolation, ikke tilstrækkelige til — blot nødvendige for — at sikre en forventet, korrekt (hensigtsmæssig) "opførsel" af systemet, herunder dets stab, brugere, kunder. Hertil tjener almene og disciplinære forskrifter: Regler og regulativer: Love, ministerielle cirkulærer, styrelses–forordninger, kontor–procedurer, mmm., — og tilsvarende for ikke–offentlige institutioner: Private virksomheder. Alle har de behov for at udtrykke regler for hvorledes "man" forventer at systemet skal "opføre" sig.

Almene forskrifter (*"rules"*) angiver hvorledes man ønsker at systemet skal fungere. Retlige forskrifter (*"regulations"*) angiver disciplinære forholdsregler som "systemet" har tænkt sig at iværksætte om ikke de almene forskrifter følges efter hensigten.

∘ EKSEMPEL 9 ∘

9. Jernbane–drift: Der kan gives to eksempler. Først de almene regler: *(1) Ethvert tog opfattes som optagende en rute på et jernbane–net. To vilkårlige, men forskellige, tog, der trafikerer enhver bestemt, men vilkårligt valgt linje, antages at trafikere denne linje ved bevægelse i samme retning, fra én station henimod "den anden" station. For enhver sådan tog–trafik skal det gælde, at der skal være mindst én spor–enhed, som ikke er inkluderet i de to togs øjeblikkelige ruter, mellem de to tog. (2) Til ethvert tidspunkt og i ethvert tidsinterval af længde* m *minutter* (hvor m typisk, i Kina, er 2) *omsluttende dette tidspunkt, må der, fra enhver af nærmere angivne stationer, højst ankomme og/eller forlade ét tog.*[26] Nu til de disciplinære forskrifter: *(1) Togfører og/eller andet personale, og/eller andre personer, der er skyld i forseelse mod denne regel [(1)] disciplineres som flgr.: ... (2) Stations–personale, togførere samt andet personale, og/eller andre personer, der er skyld i forseelse mod denne regel [(2)] disciplineres som flgr.: ...* [Slut på Eksempel 9]

∘ Eksempel 10 ∘

10. Bank–væsen: En almen forskrift lyder: *I USA skal en almindelig check udskrevet i én stat, stilet til en person eller et firma i samme stat forvaltes ("cleares") indenfor 24 timer: Pengene skal være fratrukket balancen på udsteders relevante konto og tilskrevet modtagers tilsvarende konto — fra det øjeblik modtager præsenterer checken i en vilkårlig af sin egen bank's afdelinger. Hvis udsteder– og modtager–banker ligger i forskellige stater, men indenfor samme region (f.eks. vest for "The Rocky Mountain Divide", eller mellem denne og Missisippi, eller mellem denne flod og Atlanterhavet), da skal checken 'cleares' indenfor 48 timer, ellers, hvis iøvrigt indenfor kontinentale USA, da 72 timer.* Dertil hørende retlige forskrifter lyder: *Hvis ikke en check 'cleares' som foreskrevet træder flg. erstatnings–, bøde–, oa. forhold i kraft: ...* hvorefter disse beskrives ... [Slut på Eksempel 10]

∘ Eksempel 11 ∘

11. Behandlings–garantier: To almene forskrifter lyder: *(1) Enhver Dansk borger kan frit vælge hospital efter flg. retnings–linjer: ...* hvorefter disse beskrives ... *(2) Henvist til (kirurgisk) operation for en række nærmere bestemte forhold skal borgeren have mulighed for at få denne operation, på et nærmere angivet Dansk hospital (klinik), indenfor et* nærmere bestemt antal måneder. Dertil hørende retlige forskrifter lyder: *(1) Hvis en borger ikke kan vælge hospital efter forskriften tilbydes flg.* "ersatz" (erstatning, o.lign.). *(2) Hvis en borger ikke kan tilbydes den pågældende operation indenfor de anførte tidsfrister da tilbydes borgeren et valg mellem flg. muligheder: Øjeblikkelig operation på et udenlandsk hospital (efter nærmere retnings–linjer), finansiel erstatning på K tusind kroner per måned operationen forsinkes. Etc.* [Slut på Eksempel 11]

Man kan beskrive disse forskrifter, men man kan, i domænet, ikke sikre sig at det opfylder dem. Mao., man kan tale herom, man kan tale om at detektere brud på regler, og man kan tale om eventuel 'disciplinering' heraf. Men det er først ved opstilling af krav–definitionerne at man kan beslutte sig for eventuelt at "gøre noget ved sagen !" vha. datamatisering !

[26] Reglen, der gælder for mange stationer i Kina, skyldes at der engang i 1970'rne var en tog–ulykke hvor to tog kørte ind i hverandre på en station hvorved mange passager, i tog og på perroner, omkom.

5.3.5 Menneskelig Opførsel

Forskrifter eller ej, ledelses– & organisations–struktur eller ej, mennesker og understøttende teknologi kan fejle. Nogle mennesker forsøger til stadighed at udføre deres arbejde efter bedste vilje: Korrekt, omhyggeligt, "med rette omhu", og blandt disse er der folk der altid undgår fejltagelser. Andre mennesker er mindre agtsomme, nogle direkte sløsede i udførelsen af deres daglige arbejde. Og atter andre mennesker er direkte kriminelle: Snyder og bedrager.

∘ EKSEMPEL 12 ∘

12. Rente-tilskrivning og Bank–fagligt Ansatte: *Der er i regler og forskrifter, for tilskrivning af renter til indestående beløb på spare–konti, angivet meget præcist hvorledes afrundings–rester statistisk skal udjævnes over enhver konti, eventuelt tilskrives en særlig bank–konto. Men, i gamle dage, før datamatens brug i bank–verdenen, og i de aller–første år efter datamatens brug ved beregning af rentetilskrivning, var der bank–ansatte, der kriminelt tilegnede sig selv en akkumuleret sum af sådanne afrundede brøkdele som ofte fremkommer ved rente–beregning.* [SLUT PÅ EKSEMPEL 12]

I enhver fyldestgørende beskrivelse af et domæne er det bydende nødvendigt at beskrive alle tænkelig fejl og misbrugs–situationer.

∘ EKSEMPEL 13 ∘

13. Patient–journaler og Sundheds–fagligt Ansatte: *Nok skal en patient–journals diverse ark, stamdel, mv., indeholde alle foreskrevne oplysninger, men man må være forberedt på undladelses–synder. Nok kan man tale om en rimelig komplet og konsistent patient–journal, men læger, sygeplejersker, oa. sundheds–fagligt personale kan gøre fejl: Én patient's journal ark "falder ubemærket på gulvet", eller bliver indsat i en anden patient's journal, etc.* [SLUT PÅ EKSEMPEL 13]

Så selvom man udmærket godt véd, hvad de korrekte tilstande af systemet er, må man i beskrivelser tage højde for snart sagt enhver mulig fejl–tilstand. Uden at gøre dette, systematisk og ("kedsommeligt") udtømmende, kan man ikke, siden, gøre sig forhåbninger om at opstille et konsistent og relativt komplet sæt krav–definitioner til et "fejl–detekterende" system.

Hvad der ovenfor blev sagt om mennesker's opførsel (*"behaviour"*) gælder, `inter alia,` også for understøttende teknologi: Før eller siden vil maskinelt, mekaniskt, elektriskt, og elektroniskt udstyr fejle.

5.4 Diskussion

En række spørgsmål melder sig:

- *Hvor mange interessent– og bruger–grupper skal medtages ?*

  Flere end man sædvanligvis tager hensyn til; flere end man normalt adspørger.

  I beskrivelsen af én interessent– eller, mere specifikt, én bruger–gruppe, kommer man ofte over interaktions–grænseflader mellem dem (og det) man har valgt at beskrive, og forhold omkring interessent– eller bruger–grupper som man har udeladt. Sker det, er det på tide at overveje om man også burde tage disse sidste i betragtning.

– Man kan ikke forvente at det blir' billigt. Dagens og især gårsdagens datamatik blev udviklet altfor billigt, under altfor optimistiske antagelser. Morgendagens datamatik kan forventes at blive ganske betragteligt dyrere *"up front"* — mod så, som vi siden skal fremføre (indtil videre, som en påstand), at ende op med de samme eller lavere livtids–omkostninger, nok snarere det sidste !

- *Hvornår er et domæne–facet tilstrækkeligt beskrevet ?*

  Svaret beror på hvilket formål beskrivelsen skal tjene. Hvis formålet er opstilling af nødvendige og tilstrækkelige krav, ja så vides det egentligt først, om en herfor tilgrund–liggende domæne–beskrivelse er "komplet", når man mener at krav–definitionen har berørt alle relevante aspekter. Mao.: Kombinationen af domæne–beskrivelses og krav–definitions processerne itererer "hen imod" en situation hvor man mener at "nu er alt beskrevet" !

- *Hvor stringent skal en uformel beskrivelse være?*

  Der henvises til bemærkning Side 28, lige før Afsnit 5.3.2.

  Svaret her er enkelt, en uformel beskrivelse skal mindst være så præcis, jvf. jernbane–net eksemplet, Eksempel 4 på side 27, at man alene på dens grundlag kan opstille en matematisk model, dvs. en formel beskrivelse der kan siges at dække "det samme" som den uformelle beskrivelse.

  Vi skal ikke vise udtømmende eksempler på sådanne formelle beskrivelser. Istedet henvises til en lang række offentliggjorte artikler, en længere række rapporter, og en rimelig stor samling studenter–projekt–rapporter. Rerencer hertil gives siden.

  For hver af de ovenfor omtalte facetter findes der specifikke formelle teknikker ved hvis hjælp man hjælpes til effektivt formelt at modellere de uformelt beskrevne forhold. Ja: Man kan mao. beskrive domænets "ubeslutsomhed", dets internt og eksternt bestemte ikke–determinisme (*"skal jeg nu gøre det eller det eller ... eller det"*, hhv. *"stillet overfor flere muligheder for næste aktion, hvorledes udtrykke den tilfældige aktion der vælges, selekteres ?"*).

- *Blot "Antydende" Eksempler.*

  Jeg har, mao., valgt, i de fleste eksempler, blot at antyde uformelle beskrivelser. At bringe helt realistiske eksempler — når citerede dokumenter udmærket illusterer sådanne — vil være urealistisk i nærværnede notat's sammenhæng. De ville simpelthen fylde for meget.

Mange andre forhold vedrørende domæne–beskrivelser kan anføres. Vi vil slutte af med:

5.5 Fra Domæne–beskrivelse til Domæne Teori

5.5.1 Klassiske Naturvidenskabelige Teori–dannelser

Fysikerne, siden Nikolaj Kopernikus', Isaac Newton's, etc.'s dage, har beskrevet de mekanisk–fysiske, specielt de planetær–mekaniske forhold (*celestial mechanics*), og har på grundlag af disse beskrivelser skabt teorier, formuleret i den af disse fysikere (og siden af andre) skabte

matematik. Det var deres primære hensigt[27]. Disse og mange fysikere siden dem, har ikke haft andet behov end at forstå. De har ikke alle haft det ønske også ingeniørmæssigt at kunne udnytte eventuelt opdagede lovmæssigheder. Således har vi idag altruistiske teorier indenfor elektro–fysik, atom–fysik, kemi, biologi, mmm.

5.5.2 Infra–Struktur–teorier

Og sligt kunne vi også forholde os. For at uddanne folk indenfor sundheds–sektoren, finans––verdenen, transport–væsenet, mmm., kunne det dog være rart med stramme, konsistente, og relativt komplette beskrivelser — også gerne sådanne ved hvis hjælp man bedre, intuitivt kan forstå en matematisk teori, sådan som f.eks. gymnasiaster lærer at forklare, forudsige mekaniske forhold's opførsel vha. den simple Newton'ske fysik.

Så vi kunne stoppe her. Ved domæne–beskrivelser — for hvert område. Og vi bør vel engagere os i et stort anlagt 50–100 års projekt, der kunne have som mål at opstille omfattende teori–dannelser for bl.a. de infra–struktur–komponenter der er omtalt tidligere i dette notat.[28] Vi skal vende tilbage hertil i Afsnit 7 (startende på side 57).

Men vi vil mere end det. Hvad dette er omtales i Afsnit 6 (startende på side 35).

5.5.3 Hvem skal Udforske disse Infra–Struktur–teorier ?

Hvem skal udforske sådanne infra–struktur–teorier ? Vort svar er: Det for øjeblikket mest rimelige udgangspunkt for en sådan infra–struktur–forskning synes at ligge i datalogien i bredere forstand. Det er i, eller fra datalogien vi finder, hhv. kan hente de metoder: Principper, teknikker og værktøj der idag er bedst egnede til sådanne studier. Det er i den datalogiske forskning at de nutidige studier heri finder sted. Som bredere beskrevet i Afsnit 6 (startende på side 35), tilkommer klart metoder fra matematikken, fra det vi ville kalde den anvendte matematik. Men alene den, den anvendte matematik, er idag ikke nok: Den klassiske matematik kan ikke beskrive de forhold — såsom 'samtidighed', 'typer', oma. — sådan som den nutidige datalogi kan det.

I takt med at samarbejdet med de akademiske discipliner, der tradtionelt "ligger nærmest" det enkelte, udforskede genstands–område udbygges — den medicinske videnskab mhp. studier af sundshedssektoren, osv. — i samme eller stigende takt, kan denne genstands–område–videnskab "selv" overtage hoved–arbejdet.

Ligesom den anvendte matematik ikke kun dyrkes på akademiske, universitets–institutter af sligt navn, men også, f.eks. i de fleste ingeniør–teknisk/videnskabelige institutter: Vandbygning (Hydraulik), Bærende Konstruktioner, *Aeronautics,* Elektro–magnetisk Feltteori, mmfl., på samme måde kan man forestille sig at forsknings–institutter indenfor sådanne genstands–områder som Trafik & Transport, Byggeri, Produktion, Økonomi, Handels–"videnskab", oma., "overtog" deres del af infra–struktur–forskningen.

[27] Dog siges det om Newton at han interesserede sig for at formulere de mekaniske love for at *"bevise/modbevise Gud's eksistens !*

[28] Appendiks B giver eksempler på beskrivelser af flere infra–struktur–komponenter.

6 Fra Domæne via Krav til Programmel

At opstille domæne–beskrivelser kan retfærdiggøres alene udfra ønsket om "til fulde" at forstå domænet, eventuelt herunder, som videnskabs–person at ville udvikle en teori for domænet. Men man kan, som vi skal komme ind på i Afsnit 8 (startende på side 59), også ønske at bruge domæne–analyse– og beskrivelses–arbejdet som grundlag for en institution's, et firma's, *"business process (re–)engineering"*.

Endelig er der en tredie grund til at engagere sig i det ofte omfattende arbejde med at opstille en domæne–beskrivelse, nemlig som grundlag for krav–definitions–arbejdet — henimod udvikling, installation og brug af et større programmel–system. Dette sidste vil vi beskæftige os med i dette afsnit.

6.1 Krav–definition

Hvor en domæne–beskrivelse alene beskriver domænet, infra–struktur–komponenten ("institutionen", firmaet, o.lign.) — uden at nævne, overhovedet, krav til eventuelt programmel — mao.: Hvor domæne–beskrivelsen alene beskriver "verden som den er", tjener en krav–definition til at beskrive "verden som man ønsker den" (efter udvikling og installation af ønsket, dvs. krav–defineret, programmel). En krav–definition fastlægger, mao., *hvad* det er man forventer af det ønskede programmel (ikke *hvorledes* det skal implementeres).

Ved 'maskine' forstår vi "summen" af det maskinel og programmel, der skal til for at løse en krav–definition, dvs.: Maskinen er det brugeren "ser".[29]

Vi opdeler en(hver) krav–definition i tre dele:

- *Domæne–krav–definition* udtrykkes alene med reference til domæne–termer. Der bruges ikke termer, der har med maskinen at gøre. Domæne–krav–definition er som regel de "vigtigste" i den forstand at de er hovedårsagen til at man ønsker et programmel–system, dvs. en maskine.
- *Grænseflade–krav–definition* udtrykkes ved at identificere de fænomener i domænet som "deles" med maskinen i flg. forstand: (1) Fænomenernes øjeblikkelige tilstand i domænet skal, til passende tidspunkter, eller i forbindelse med specifikke begivenheder, "afbildes ind i" maskinen, dvs. "indlæses"; eller (2) de ydre fænomener (for hvilke der i maskinen findes modsvarende tilstands–komponenter) er som regel genstand for forudsigende oa. beregninger — og kan da ønskes "udskrevet til omverdenen" (f.eks. via data–skærmen); eller (3) begge dele. Der er mao. tale, dels om alm. ind/uddatering, dels om visuel, tekstuel og anden visualisering vha. en grafisk grænseflade.
- *Maskin–krav–definition* udtrykkes alene med reference til maskinens terminologi. Maskin–krav–definition er således (som regel) udtrykt med meget generelle henvisninger, ikke til domæne–specifikke, men blot til almene "funktioner" sådan som ethvert programmel–system "realiserer".

I det flg. skal vi nærmere præcisere ovenstående sikkert lidt for abstrakte formuleringer.

[29] Vi forventer at læseren ikke forveksler vor brug af begrebet 'maskine' alene med datamaskinen som maskinel.

6.1.1 Domæne–krav ≡ Funktions–krav

Domæne–beskrivelser har identificeret en lang række forhold i deres genstands–område: Basale, teknologi–understøttende, ledelse– & organisation, almene (retledende) og legale (disciplinære) regler mv., menneskelig opførsel, oa.

Indførelse af datamatik skyldes ønsket om at understøtte, eventuelt at automatisere, visse af disse forhold. Disse ønsker kan være begrundet, f.eks., i en kombination af ét eller flere flg. "meta–krav":

- At udskifte lønmæssigt omkostnings–belastende (menneske–) arbejde med billigere maskin–arbejde.
- At udskifte monotont, trættende eller farligt menneskeligt arbejde med maskinelt, datamat–styret arbejde.
- At sikre en størst mulig pålidelighed i udførelsen af sådanne arbejds–processer som ofte giver anledning til fejl (upålidelighed).
- At få visse arbejds–processer udført hurtigere.
- At få udført arbejde det forhen ikke var muligt at få udført.
- *&c.*

Vi har listet ovenstående "trivialitet", dels for at introducere det pragmatiske "meta–krav" begreb, dels for at begrunde begrebet domæne–krav.

- *Meta–krav:* Disse er sådanne som — direkte formuleret — ikke i sig selv fortæller datalogen præcist hvilke funktioner programmellet skal tilbyde.

  Vi skal, i Afsnit 6.1.5 (startende på side 52), fundere nærmere over begrebet meta–krav.

- *Domæne–krav:* Disse er krav, der er direkte begrundet, herunder eksplicit formuleret, mht. bestemte. umiddelbare, domæne–fænomener.

  Vi skal i de flg. mange afsnit give eksempler herpå.

Der er en række domæne–krav–definitions–teknikker under hvis brug man mere systematisk kan komme frem til domæne–kravene. Disse inkluderer:

- Projektion,
- instantiering,
- udvidelse, og
- initialisering.

I det flg. skal vi se nærmere på disse teknikker.

6.1.1.1 Projektion: Ved 'projektion' forstås en afbildning af domænet, en mulig indsnævring af dette, overpå krav: I projektionen fastlægges det hvilke dele, hvilke facetter, af domænet der er mål for krav—definitionen.

Specifikt startes der ud med en præcis afgrænsning af hvem der er at betragte som brugere, og hvem der er at betragte som interessenter iøvrigt.

∘ EKSEMPEL 14 ∘

14. Sundhedssektor: Tidligere eksempler (1 på side 25, 2 på side 26, 3 på side 27, 6 på side 29, 7 på side 29, 8 på side 30, 11 på side 31, og 13 på side 32) har illustreret aspekter ved, facetter af et sundheds–sektor–domæne (sådan som vi, sikkert noget amatøragtigt, opfatter det). Vi skal nu illustrere krav til programmel til understøttelse af funktioner, aktiviteter, i et sådant sundheds–sektor–domæne. *Istedet for den "bredeste kreds" af interessenter der er listet i Sundhedssektor Interessenter eksemplet på Side 25, projiceres til blot: (0) Raske og syge borgere, (2) familie–, dvs. praktiserende læger, (3) sundheds--sygeplejersker, (4) klinisk analyse laboratorier, (6) apoteker, og (7) hospitaler (inkl. disses stab, afdelinger, laboratorier, mmm.). Desuden projiceres, dvs. overføres til videre bearbejdelse, det eksplicitte, mere–eller–mindre formular–baserede patient–journal begreb. (Eksempel 3 på side 27)* [SLUT PÅ EKSEMPEL 14]

'Projektion', som princip og som teknik, med baggrund i de stringente metoder for domæne–beskrivelse, hvad enten de er uformelle eller formelle — men som ikke kan vises i dette notat — præciseres vha. af en række anvisninger, uformelle såvelsom formelle.

6.1.1.2 Instantiering: Ved 'instantiering' forstås en præcisering af de projicerede fænomener. Nu ikke som egenskaber ved et domæne, men som krav til maskinen (programmellet).

Hvor det basale i et domæne måtte tillade mange forskellige manifestationer af et fænomen kan det tænkes at krav reducerer disse til nogle få, f.eks. netop én. Hvor den understøttende teknologi tilsvarende måtte tillade mange forskellige manifestationer kan tilsvarende reduktion tænkes. Hvis der stilles krav til understøttelse af ledelses– & organisations–funktioner, som måske "før", i domæne–beskrivelsen, var løst skitserede, og/eller tillod mange mindre formaliserede interaktioner mellem ledere og de ledede, kan man forestille sig en "stramning" til at bestemte daglige, rutine–mæssige ledelses–forhold underlettes vha. præcise spille–regler som maskinen da skal overvåge (*monitor*) om de finder sted, og i givet fald, gennem styring (*control*), sikrer at de finder sted. Vejledende og disciplinære forskrifter (regler) er klart et område hvor datamatisering kan understøtte, subsidiært automatisere — bl.a. gennem indføring af ét eller flere domæne–specifikke "regel– & rutine–skripts" (sprog). Og endelig udgør menneskelig opførsel, specifikt vores mere eller mindre manglende, stadige/ustadige evne til regel–ret at følge intentioner, forskrifter, et område hvor krav til lettelse, kontrol (overvågning) og styring vha. datamaten, kan sikre smidigere afvikling af typisk monotone, og/eller fejl–typiske, og/eller sikkerheds–kritiske arbejds–opgaver.

I Eksempel 15 skal vi alene set på et simpelt, oplagt krav–emne.

∘ EKSEMPEL 15 ∘

15. Den Elektroniske Patient–Journal (EPJ): *Til enhver (nulevende) borger er der knyttet nul, én eller flere, men et endeligt antal elektroniske patient–journaler (herefter: EPJ). Nogle er aktive, dvs. endnu ikke afsluttede. Andre, "ældre", er afsluttede. Til enhver tid kan*

der eksistere nul, én eller flere aktive EPJ. Disse forstås (antages), hver for sig, knyttet til af hinanden ikke–relaterede sygdoms–forløb. *Enhver EPJ har en stamdel og en problem–del. Stamdelen indeholder flg.* nærmere angivne ..., *administrativ og "cave" information. Problem–delen er rekursivt defineret, dvs. kan karakteriseres vha. eventuelt "ægte indeholdte" problem–dele: En (hoved–)problem–del har (i) et problem–navn; (ii) en problem–beskrivelse; (iii) en mål–beskrivelse; (iv) nul, én eller flere, men et endeligt antal under-problem–dele; og (v) eventuelt en tidligere, nu forkastet problem–del. Et under–problem er, i og for sig, ikke direkte relateret til sin hoved–problem, men er antruffet i forbindelse med behandling af et hoved–problem. En forkastet problem–del er en tidligere problem–del, dvs. en problem–del, men som "dengang" var en hoved–problem–del som engang blev erstattet af en ny (hoved–)problem–del.*

• • •

En problem–beskrivelse består af flg. "arkivalier": Annamnese, analyser, diagnostik, behandlings–plan (medikation, operativt indgreb, mv.), observationer (i forb. med behandlings–planens gennemførelse [eller manglende sådanne]), ...samt andre "arkivalier". Herefter flgr. nærmere beskrivelser af arten, formen, indholdet af og de aktioner der kan udføres af det sundheds–faglige personale på de forskellige former for "arkivalier", hvem disse måtte tilflyde, mmm.

• • •

Enhver problem-del's problem–beskrivelse (hvad enten denne direkte tilhører et "øverste" hoved–problem, eller tilhører et deraf "afledet" under–problem) kan betragtes som knyttet til en tilsvarende, "fritstående" process. Denne process svarer til at en række sundheds–faglige personer varetager denne problem–del, og dermed at eventuelt andre, derfra forskellige, men gerne overlappende grupper af sundheds–faglige personer varetager hver sin under-problem–del.

• • •

En EPJ skal således tillade ikke–delt, dvs. samtidig læse–tilgang til hele EPJ'en, men med egen, beskyttet, skrive–, dvs. opdaterings–tilgang til den bestemte under–del. Overordnet kan der udføres flg. aktioner på en EPJ: (1) Oprettelse af en hel ny EPJ, (2) erstatning af én hoved–problem–del med en ny (evt. en modifikation af en tidligere erstattet, og dermed "hobet") problem–del, (3) oprettelse af en (ny, frisk) under–problem–del (som derfor, til en begyndelse, hverken vil have under–problemer eller "erstatninger"), osv. Ved erstatning "fryses" et billede af hele EPJ'en, og dette samt en angivelse af hvor i denne EPJ "man kom fra", dvs. en "identifikation" af rette plads i hierarkiet af eventuelle under–problemer, udgør del (v): Den komponent i en problem–del, der repræsenterer en forkastet, dvs. "tidligere" problem–del.

• • •

Og meget mere. Ovenstående EPJ beskrivelse er sikkert altfor stram til at mange kan forstå den. Beskrivelse skal derfor understøttes vha. abstrakte, generiske, såvelsom konkrete, "øjebliks–billed"–tegninger, diagrammer, mv.

• • •

Eksemplet er dog illustrativet også for behovet for en mere stringent, formel beskrivelse. Vi skal kort illustrere en sådan:

type
Stamme
EPJ = *Stamme* × *(Pn* × *ProbI)*
ProbI = *ProbI_Beskriv* × *Maal* × *under:(Pn* $\overrightarrow{m}$ *ProbI)* × *gammel:ProbI*
ProbI_Beskriv = *(Anam|Anal|Diag|Plan|...)*-**set**
Anam, Anal, Diag, Plan, ...

De to forekomster af rekursionen mht. Problem har hvert sit formål: (I) I under–delen opnås en vilkårlige dybde af under–under–· · ·–dele (selvom måske det i praksis højest er relevant at tale om f.eks. maksimalt 2 dybder). Afbildningen Pn $\overrightarrow{m}$ ProbI (der definerer at der til hvert af forskellige problem-navne, nul, én eller flere, entydigt kan knyttes et problem) tillader således et vilkårligt, endeligt antal under–problemer; rekursionen en vilkårlig dybde. (II) I gammel–delen ligger der "gemt" en "stak", dvs. en vilkårlig liste af, fra seneste erstatning, til tidligste, mest oprindelige erstatning. At det er en stak fremgå dels af selve definitionen ovenfor, dels af definitionen af de "erstatnings"-–aktioner der "stakker" en tidligere hoved–del (af en vilkårlig problem–del). Vi viser ikke definitionen af disse aktioner.

• • •

Ovenstående ligninger —mener vi — er kortere, "rammer" mere præcist, efterlader knap så mange ubesvarede spørgsmål, når vi sammenligner med (bl.a. UML diagrammer i) en nyere Sundhedsstyrelses rapport om EPJ emnet. [Slut på Eksempel 15]

'Instantiering', som princip og som teknik, med baggrund i de stringente metoder for domæne–beskrivelse, hvad enten de er uformelle eller formelle — men som ikke kan vises i dette notat — præciseres vha. af en række anvisninger, uformelle såvelsom formelle.

6.1.1.3 Udvidelse: Ved 'udvidelse' forstås indførelse af, typisk, sådanne funktioner på instantierede størrelser (entiteter) som ikke var realistisk mulige i domænet, men som nu muliggøres vha. datamaten.

De faciliteter, der således repræsenterer en 'udvidelse' i forhold til domænet, var typisk ikke mulige i domænet. Der kan væ forskellige grunder hertil. (1) Enten (for de tilfælde hvor de repræsenterer funktioner), fordi det ikke var menneskeligt eller på anden vis (mekaniskt) muligt at beregne (det ville tage for lang tid, år, for et menneske at beregne). (2) Eller (for de tilfælde de repræsenterer søgning i endog meget store informantions–mængder), det er fordi sådanne mængder ikke er menneskeligt overskuelige. (3) Eller (for de tilfælde de repræsenterer visualisering, typisk af multi–variabel information [dvs. information med mange, 10–20 dimensioner]), det er fordi det ikke er menneskeligt muligt at "skrue" på 10–20 indstillinger "samtidigt".

∘ Eksempel 16 ∘

16. Rejseplanlægning: Dette eksempel er taget fra bus–, jernbane–, skib– og fly–rejse domænet. Antag at der findes time–tabeller for samtlige verdens bus–, tog–, skibs–, og fly–forbindelser. Lad et punkt repræsentere enten et bus–stoppested, en tog station, en havn eller en lufthavn. Lad et knude–punkt repræsentere et punkt hvortil der ankommer og (antages det yderligere), derfor også afgår indtil flere, dvs. mere end én, bus–, tog–, skibs–, eller

fly–forbindelse — et vilkårligt "mix". Lad os ved en rejse forstå en bus-, tog–, skibs–, eller fly–forbindelse mellem to punkter — en forbindelse der gerne må passere via andre, mellemliggende punkter, og hvor det gerne må ske at rejsen skifter fra éet befordings–middel til et andet. *Givet to vilkårlige punkter, a og b, ønskes der nu genereret, af det krævede programmel–system, den endelige mængde af alle de rejser der forbinder a med b.* For et menneske at forsøge generere alle mulige rejser mellem en lille by i Siberien med en tilsvarende lille by i USA's midtvest er ganske enkelt ikke overkommeligt. Vedkommende person skulle have titusindvis af fingre, skulle kunne huske alle de "træstrukturerede" søgeveje, etc., etc. Men at skrive et `Prolog` program herfor er en smal sag — selvom kørsels-tiden sikkert vil være lang, måske et par dage, i hvilke maskinen til stadighed ville generere nye rejsemuligheder. [Slut på Eksempel 16]

Problemet med Eksempel 16 er, at det faktisk, i realiteten, er et polynomisk "svært" problem: At den ønskede beregning — om der f.eks. yderligere bedes om hurtigste vej med færrest skift mellem befordrings–midler således at rejse–strækninger kortere end 6 timer altid ligger i dagtimerne 8 morgen til 20 aften — at beregnings–tiden for dette problem vil vokse så kraftigt at man i realiteten ofte nøjes med sådanne rejseruter der ligger indenfor en bred "avenue" omkring den direkte, sfærisk korteste vej mellem a og b.

Det næste eksempel (Eksempel 17) "snerter" i samme retning.

∘ Eksempel 17 ∘

17. EPJ "Data Mining": *Givet en virkårlig stor mængde personers samlede EPJ'er, f.eks. hele den således (engang i fremtiden) registrerede del af Danmarks befolkning, da at indføre ("kræve") en søge-funktion, der, givet en række sygdoms–symptomer, finder alle de deraf afledte sygdoms–symptomer således at der er mindst 5% af befolkningen der både har, eller har haft eller har de første af, de givne symptomer, og (både og) som før eller nu, har de afledte.* Vi henviser til eksemplerne startende med Eksempel 27 på side 46 hvori vi omtaler krav til eventuelle kørsels–tider for sådanne søgefunktioner. Snævre, dvs. krav om korte, beregnings–tider kan medføre at man "ændrer" ("korter af") af på det med de 5%, etc. [Slut på Eksempel 17]

'Udvidelse', som princip og som teknik, med baggrund i de stringente metoder for domæne–beskrivelse, hvad enten de er uformelle eller formelle — men som ikke kan vises i dette notat — præciseres vha. af en række anvisninger, uformelle såvelsom formelle.

6.1.1.4 Initialisering: Ved 'initialisering' forstås opbygning og vedligehold af de maskin–tilstands–komponenter, der afspejler tilsvarende tilstande, dvs. entiteter, i det aktuelle domæne (*"the real world"*).

Der er mao. tale om to tilsyneladende forskellige klasser af funktioner: De der oprindeligt initialiserer tilstande, og der der "vedligeholder" dem, "opdaterer" dem i takt med ændringer i deres "modpart ude i domænet" !

∘ Eksempel 18 ∘

18. EPJ: Initialisering og Løbende Ajour–føring: *Det ønskede system skal, ved leverance, undergå en initialiserings–proces hvorved et nærmere bestemt antal hidtidige patient–journaler "overføres" (fra den form de måtte eksistere på) til det leverede system. Systemet skal*

ved interaktiv hjælp fra yngre læger og rimeligt erfarne sygeplejersker, vha. diverse skannere, f.eks. råtekst–genkendelses–delsystemer, oa., på en gennemsnitstid af mindre en x minutter per journal, kunne omdanne denne til det ny system's simpleste form for EPJ–repræsentation efter retningslinjer der vil blive redegjort herefter: Herefter flgr. nærmere retningslinjer mmm. *Løbende opdatering af en EPJ, "ind i" dens vilkårligt udpegede hoved– eller under–problemer's beskrivelser, skal ske efter flg. retningslinjer. (i) En lang række instrument–registrerede analyse–resultater skal overføres direkte fra instrument til journal.* Herefter flgr. hvilke disse instrumenter er. *(ii) Lyd (tale) indtalte annamneser kan, til en begyndelse, indlemmes som en réel del af en EPJ.* Osv. ... Herefter flgr. nærmere retningslinjer mmm. [Slut på Eksempel 18]

'Initialisering', som princip og som teknik, med baggrund i de stringente metoder for domæne–beskrivelse, hvad enten de er uformelle eller formelle — men som ikke kan vises i dette notat — præciseres vha. af en række anvisninger, uformelle såvelsom formelle.

• • •

Ovenfor er blot fremdraget fire hoved–teknikker for en systematiseret afledning, under interaktion med projicerede interesse–grupper, af domæne–krav. Andre "afledning"–teknikker kan identificeres.

6.1.2 Grænseflade Krav

6.1.2.1 Grafisk Brugerflade: Ved 'grafisk brugerflade' forstås basalt (typisk) den visuelle dataskærm der tillader visning af stort set vilkårlig farve–grafik (evt med pinger–peger taktilitet).[30]

∘ Eksempel 19 ∘

19. EPJ Vinduer: *Alle menneske/maskin grænseflader skal ske via sådanne 'vinduer' som er udviklede vha. det system der krav–defineres i Eksempel 20 ("Formular–baserede EPJ Journaler").* [Slut på Eksempel 19]

∘ Eksempel 20 ∘

20. Formular–baserede EPJ Journal–"Vinduer": Vi forestiller os at al kommunikation mellem bruger og EPJ–System finder sted vha. såkaldte "vinduer". I stedet for, én gang for alle, at konstruere alle de mange forskellige "vinduer", der typisk vil blive brugt i et administrativt informations–system (som det til hvilket EPJ knyttes) — opstilles flg. krav om afdelings–bestemt "vindues"–design. *EPJ Systemets stamdel, problem–beskrivelses arkivalier, målbeskrivelses–del, og dets mulighed for de sundhedsfagligt ansatte at navigere rundt i én eller flere (patienters) EPJ'er, (design af dette) skal baseres på (dvs. finder sted under brug af) et del–system,* `"EPJ Formular--Konstruktøren"`. *Vha. dette del–system kan hvert enkelt hospital, eller hver enkelt afdeling indenfor et hospital, konstruere sine egne "vinduer" — idet dog en fast forskrift, som ikke vil blive beskrevet her, skal følges aht. sikring af nem udveksling af EPJ'er mellem afdelinger, mellem hospitaler, og mellem hospitaler og andre sundheds–sektor–institutioner.*

[30] At denne grafiske skærm synes at være knyttet sammen med et sædvanligt tastatur, en "mus" og evt. andet grænseflade udstyr, er i og for sig sagen uvedkommende — men tillader dog at man logisk kan koble tekstuel eller "klikkende" inddatering med grafisk uddatering.

• • •

*Det generelle EPJ vindue består af [0,1][31] et unikt vindues–navn (Wn), [0,1] en basis tekst (***Text***), [1] én eller flere éntydigt navngivne (Nm) ikoner, som [2] enten er simple ikoner ([4] IVAL), eller vindues–ikoner ([1] WVAL), eller "rullegardiner" ([5] CVAL), og [2,3] nul, én eller flere éntydigt navngivne tabeller. [4] En simpel ikon består af et ikon–navn og en ikon–værdi. (Ikon–navnet er unikt indenfor et bestemt vindue.) [1] Et vindues–ikon består af et vindues–ikon–navn og en vindues–ikon–værdi, som er et generelt EPJ vindue. [5] Et rullegardin består, i sin grund–status, af et enkelt rullegardin–ikon, som så består af et rullegardin–navn og en rullegardin–værdi. [3] En tabel består af et tabel–navn (der er unikt indenfor et vindue), en tekst (***Text***), og to eller flere felter. Et felt består af et feltnavn (Fn) der er unikt indenfor tabellen, en felt–tekst (***Text***), og en felt–værdi. En felt–værdi kan enten være "tom", dvs. uudfyldt, eller (kan blive udfyldt med) en felt–værdi (FVAL). [4] En ikon–værdi er enten "slukket" eller "tændt" ("off" eller "on", ikke–"klikket" eller "klikket"). Ikon–navnet skal, af vindues–designere vælges således at det antyder denne's to–værdi status. Eksempler kunne være:* `han/hun-køn`, *eller* `ambulant/indlagt`. *Når ikonen er "slukket" er ikon–navnet illumineret med sin "default", som designeren bestemmer. [1] en vindues–ikon–værdi er i den ikke–"klikkede" tilstand det ikke–illuminerede vindues–ikon–navn. I den "klikkede" tilstand ændres status (værdi) til at åbne et nyt vindue (mao. rekursion) med samme navn som vindues–ikon–navnet. [5] Et rullegardin–ikon–værdi er — i den ikke–"klikkede" tilstand — det ikke–illuminerede rullegardin–ikon–navn ([5] nil). I den "klikkede" tilstand ændres status til at vise en liste: "rullegardinet" ([5] mC(sC(Nm* $\overrightarrow{m}$ *Icon))). Denne liste "fjernes" ved et dobbelt-klik på rullegardin–ikonet. Listens elementer er [5,2] enten simple–ikoner, eller vindues–ikoner, eller er yderligere rullegardin–ikoner. Disse kan klikkes som øvrige ikoner.*

• • •

Vindues–designeren starter med en "tom" vindues–design skabelon. Denne er indrammet med meta–ikoner, én for hver type (ikke meta–) ikon og for tabeller og for disses felter. Ved at klikke disse meta–ikoner "åbnes" en grafisk størrelse af den benævnte art. Denne kan flyttes til en ønsket, eller en temporær vindues–position. Designeren skal nu navngive den, forsyne den med tekst, og kan nu vælge, før eller siden, at videre definere den (som for ikke–simple ikoner). Designeren kan om–placere ("flytte") ikoner, tabeller, rullegardin lister, felter, eller ikon–åbnede vinduer ved at flytte disse. Designeren kan forsyne et felt med en type angivelse: Tekst, heltal, naturligt tal, rationelt tal, oa. Bemærk at en rullegardin–listes elementer kan være ikoner og tabeller af forskellig art.

type
[0] *Nm, Fn*
[1] $WVAL = Nm \times \mathbf{Text} \times (Nm \xrightarrow{m} (\mathbf{Text} \times (Icon \mid Tbl)))$
[2] $Icon = IVAL \mid WVAL \mid CVAL$
[3] $Tbl = Fn \xrightarrow{m} (\mathbf{Text} \times FVAL)$
[4] $IVAL == on \mid off$
[5] $CVAL == off \mid mC(sC(Nm \xrightarrow{m} Icon))$

[31] De kantet–omklammede cifre refererer til formel–linjer i formler på Side 42. Læseren kan udelade læsning af disse formler. Vi angiver dem blot for at vise hvor "kompakt" en formel model kan udtrykkes jævnført med den uformelle, dog stringent præcise beskrivelse i ovennævnte paragraf.

[6] *FVAL* == *nil* | **Bool** | **Int** | **Rat** | **Char** | **Text**

I det ovenstående har vi overhovedet ikke nævnt de hermed konstruerede vinduers relation til EPJ'er ! Det kommer nu. *Alle navne, al tekst og de feltværdier med hvilke brugeren kan udfylde tabeller skal være udvalgt fra en liste af sådanne navne og felt–værdi–typer som er designet mere generelt, f.eks. ved dels at være en del af et lands' sundhedsfaglige vokabular, og dels at være separat krav–definerederede. Hermed "forbindes" de pågældende ikoners og felters status direkte til størrelser, dvs. variable, i de programmel–procedurer til hvilket et vindue siden knyttes.* Dette sidste anvises siden. ... [SLUT PÅ EKSEMPEL 20]

Eksempel 20 på side 41 (netop afsluttet ovenfor) illustrerer, gennem sin både uformelle og formelle fremstilling — vil vi gerne fremhæve — nytten af den formelle del: Det blir' li'som "nemmere" at sikre det vi kalder *reference–mæssig gennemskuelighed*, her illustreret ved at *"ethvert sted en ikon kan forekomme kan enhver sådan type ikon–værdi forekomme"*.

6.1.2.2 Bruger/Maskine Dialog: Ved en 'bruger/maskine–dialog' forstås retnings–linjer for den sproglige, konventionelt typisk den syntaktiske, gensidige interaktion mellem bruger og maskine: Hvilke protokoller man bør eller skal følge, fastlægning af et, til domænet svarende 'sessions'–begreb, hvorledes man åbner en ny session, "lukker" en nuværende session, mm.

○ EKSEMPEL 21 ○

21. EPJ Dialog–Maskiner: *Det rekvirerede EPJ informatik–system skal understøtte indtil flere maskinelle former for menneske/maskin kommunikation:*
    - *Konventionelle, stationære, PC–baserede datamater tilkoblet alm. Inter– og Intra–net (med E–Mail) ... både gennem fysiske og gennem trådløse (f.eks. infra–røde) data–kommunikations–forbindelser.*
    - *Konventionelle, bærbare, men stadig PC–baserede datamater, men trådløst tilkoblet alm. Inter– og Intra–net (med E–Mail) ... (f.eks. gennem infra–røde data–kommunikations–forbindelser).*
    - *Bærbare (håndholdte), special–designede Journal–maskiner — som separat beskrives andetsteds,* dog ikke i dette notat.

    Man kan, med nogen ret, hævde, at ovenstående egentligt er et platforms–krav. Se siden (Afsnit 6.1.3.4.) [SLUT PÅ EKSEMPEL 21]

Tilsvarende maskinel kan tænkes i forbindelse med medicinering oma.

○ EKSEMPEL 22 ○

22. Patient/Læge–Maskin Dialog: Nærværende eksempel på en krav–definition kan siges både at repræsentere en 'domæne–udvidelses'– og et 'dialog'–aspekt. Eksemplet bygger på nutidig, lovende forskning indenfor *Speech Act Theories, Agent Communication Languages* og *Modal Logikker* (det senere såsom *"Belief/Knowledge"*, *"Intention/Commitment"*, *Nu/Engang/Altid* [det sidste repræsenterer såkaldte temporal logikker] og andre modaliteter). *Der ønskes en delvis automatisering af Annamnese–delen af journal–skrivning og (om end "foreløbig") diagnostiserende fortolkning. Det skal sikres, vha. en*

stærkt begrænset natursproglig mulighed for patienten, eller for lægen, skriftligt at kommunikere med maskinen således at de ønskede programmel udspørger patient/læge efter flg. 'Speech Act' og Agent–sprogs konventioner: Herefter flgr. så en nærmere præcisering — typisk ved henvisning af f.eks. flg. art: *Man ønsker at patient/læge–maskin dialogen skal bygge på de teorier og de forslag der er publieceret i flg. artikler [...], osv.* [SLUT PÅ EKSEMPEL 22]

6.1.2.3 Anden Ind/Ud–datering: Ved 'anden ind/uddatering' forstås ind– og/eller udlæsning, som "dirigeret" fra menneske/maskin grænsefladen, fra ét del–system til et andet. Eksempler kunne være: Kopiering, "back–up", generel informations–kommunikation, mmm.

◦ EKSEMPEL 23 ◦

23. Indlæsning af EPJ Information: *Det udviklede system skal understøtte — ikke her nærmere grænseflade– og dialog–definerede — indlæsning af EPJ information (data) fra flg. udstyrs–typer:*
    - *Overføring af Röntgen billeder fra flg. typer Röntgen–udstyr direkte til det sted i en EPJ hvortil det frit kan refereres til: ....*
    - *Overføring af MR skanninger fra flg. typer MR skanner–udstyr direkte til det sted i en EPJ hvortil det frit kan refereres til: ....*
    - *Overføring af EKG strimler fra flg. typer EKG–udstyr direkte til det sted i en EPJ hvortil det frit kan refereres til: ....*
    - *Overføring af alle former for blod–analyser fra flg. typer blod–analyse–udstyr direkte til det sted i en EPJ hvortil det frit kan refereres til: ....*
    - &c.

    Det leverede EPJ System skal, derudover, sikre en fælles menneske–maskine grafisk– og dialog–grænseflade "tværs over" ovenstående ind–daterings–udstyr. Flg. retningslinjer bør så vidt muligt efterleves: ... [SLUT PÅ EKSEMPEL 23]

◦ EKSEMPEL 24 ◦

24. Særlig Visualisering (Udskrivning) af EPJ Information: *EPJ Information af grafisk karakter, sådan som tjener til visuel analyse med henblik på bestemmelse af diagnostik og behandlings–planer, skal kunne udskrives som følger:*
    - *Röntgen billeder til flg. typer Röntgen–billed–display–udstyr: ....*
    - *MR skanninger til flg. typer MR–billed–skanner–udstyr: ....*
    - *EKG strimler til flg. typer EKG–strimmel–display–udstyr: ....*
    - &c.

    Det leverede EPJ System skal, derudover, sikre en fælles menneske–maskine grafisk– og dialog–grænseflade "tværs over" ovenstående ud–daterings–udstyr. Flg. retningslinjer bør så vidt muligt efterleves: ... [SLUT PÅ EKSEMPEL 24]

◦ EKSEMPEL 25 ◦

25. Udveksling af EPJ Information: Med udveksling af EPJ information forstås overførsel af elektroniske kopier af vilkårlige, inklusive alle grafiske dele af en EPJ ("original"). *Det leverede system skal tillade udveksling af principielt vilkårlige dele af en EPJ.* Dele (1) af dette krav–element, kan med rette også, eller rettere, henregnes til domæne––krav kategorien af krav–definitioner. Andre dele (2) til maskin–krav kategorien af krav–definitioner. Mht. (1) tænkes her på det forhold at udvekslingen er direkte adspurgt gennem en specifik identificérbar forespørgsel, "noget der er en almen hændelse i domænet". Mht. (2) tænkes her, dels på autorisations–forhold (tilgængelighed og sikkerhed), dels på implementerings– (ie. platforms–) forhold såsom f.eks. at krav–definere at udvekslet data "indkapsles" i `XML`. *Det leverede EPJ System skal, derudover, sikre en fælles menneske–maskine grafisk– og dialog–grænseflade "tværs over" de ovenfor antydede udveklings–former. Flg. retningslinjer bør så vidt muligt efterleves: ...* [Slut på Eksempel 25]

∘ Eksempel 26 ∘

26. Kopiering i Almindelighed: *Ved en EPJ original forstås en EPJ, der er mærket, der — per edict — er at betragte som værende en, eller den, "første" EPJ mht. et bestemt sygdomsforløb. En EPJ, eller dele af en EPJ, kan være en kopi. Med en kopi af en del af en "eksisterende" EPJ original eller anden kopi, forstås en version der ikke er (en del af) den kopierede original eller den kopierede anden kopi, men som i alle andre henseender indeholder nøjagtig den samme information som den kopierede EPJ, vel at mærke på det unikke tidspunkt kopien blev skabt. En kopi skal mærkes som værende en kopi. Hvis kopien er på papir skal papir–kopien med andre ord dels være en kopi, dels derudover klart vise at det er en kopi, samt henvise til den originale elektroniske original (eller kopi) (på basis af, med reference til, hvilken, den er en kopi), hvornår kopien blev fabrikeret, samt af hvem (person eller institution). Tilsvarende for elektroniske kopier. Man skal også kunne kopiere elektroniske kopier. Der kan ikke institueres datamat–understøttelse af overvågning af ikke–elektroniske kopier. Originaler er med det EPJ System, der ønskes udviklet, alle elektroniske. Mere specifikt skal det rekvirerede EPJ System understøtte følgende aktuelle kopierings–funktioner: ...* [Slut på Eksempel 26]

6.1.3 Maskin Krav

Et krav er et maskin–krav hvis det alene omhandler forhold omkring den programmelle og/eller maskinelle realisering af domæne– og/eller grænseflade–krav.[32]

Vi kan identificere flg. kategorier og under–kategorier af maskin–krav:

- effektivitet:
  - afviklings–tid,
  - svar–tid,
  - lagerforbrug, og
  - andet ressource forbrug;

[32]Det er ikke altid nyttigt at forsøge 'definere' ethvert begreb; At reducere det éntydigt til en passende meningsfyldt kombination af andre begreber. I stedet for en sådan "definition" tjener en passende række under–definitioner på hvad der så siges at være eksempler på under–klasser af maskin–krav.

- afhængigheder:
  - pålidelighed,
  - fejl–tolerance,
  - nærværenhed,
  - tilgængelighed,
  - sikkerhed, og
  - robusthed;
- vedligehold:
  - forbedrende,
  - tilpassende, og
  - fejlrettende vedligehold;
- platforme:
  - udviklings–,
  - drifts–, og
  - vedligeholds–platform; og
- indlæring / tilegnelse;
- dokumentation:
  - udviklings–,
  - installations–,
  - bruger–, og
  - vedligeholds dokumenter.

Både domæne– og grænseflade–krav kategorierne var mindre nuancerede. Kategoriseringerne dér er nyere og repræsenterer nyere indsigt. Maskin–kravs nuanceringen er, i den henseende, bedre etableret.

6.1.3.1 Effektivitet: Med effektivitet forstås generelt forbruget af ressourcer som følge af udførsel, eksekvering, af domæne–krav–definerede funktioner (inklusive lagring af information).

6.1.3.1.1 Afviklingstid: Ved 'afviklings–tid' forstås den totale tid (*"the elapsed time"*) der udspænder sig fra det tidspunkt forespørgslen om løsningen af et veldefineret problem er afsendt fra brugeren (en person eller en enhed der ligger udenfor det betragtede system) til et fuldt fyldestgørende svar, dvs. reaktion, herpå foreligger.

∘ EKSEMPEL 27 ∘

27. EPJ "Data Mining": Vi henviser til Eksempel 17 på side 40. *Ved søgning over store EPJ informations–mængder skal seneste svar (resultater) være præsenteret (ud mod data–skærmen (eller tilsvarende respons–medium)) inden 60 sekunder (ét minut).* [SLUT PÅ EKSEMPEL 27]

6.1.3.1.2 Svar–tid: Ved 'svar–tid' forstås den tid der udspænder sig fra det tidspunkt forespørgslen om løsningen af et veldefineret problem er afsendt fra brugeren (en person eller en enhed der ligger udenfor det betragtede system) til en "aller–første" reaktion foreligger.

○ EKSEMPEL 28 ○

28. EPJ "Data Mining": Vi henviser til Eksempel 17 på side 40. *Ved søgning over store EPJ informations–mængder skal første svar (resultater) "strømme" ud mod data–skærmen (eller tilsvarende respons–medium) indenfor 1,5 sekund.* [SLUT PÅ EKSEMPEL 28]

6.1.3.1.3 Lagerforbrug: Man bør forestille sig at nogle former for, visse dele af, information konceptuelt foreligger i mange kopier (i et domæne, og derfor potentielt i et data–system).

Ved lagerforbrug forstås der her den samlede mængde lagerplads som sådan (eventuelt, ie. konceptuelt, fler–kopieret) information optager.

○ EKSEMPEL 29 ○

29. EPJ Hierarkier: I vores model af en nutidig EPJ, med dens under–journaler og deres "kopier" af EPJ–"øjebliks–billeder" mmm. er der konceptuelt lagt op til redundans, dvs. til — i en vis forstand "overflødig" duplikering (mm.) af information. *I et aktuelt EPJ System skal enhver duplikering af information "så vidt muligt" undgås. Leverandøren skal godtgøre at enhver sådan duplikering af information, ressource–mæssigt, modsvares af formindsket kørsels– (afviklings og/eller svar) tid.* [SLUT PÅ EKSEMPEL 29]

6.1.3.1.4 Andet Ressource–forbrug: Ved 'andet ressource–forbrug' forstås her forbrug af sådanne afviklings–ressourcer som data–kommunikations–tid, data–kommunikations–båndbredde, energi–forbrug, oma.

○ EKSEMPEL 30 ○

30. EPJ System: *Det ønskede EPJ System, der skal dække hele Danmark, skal ikke unødigt beslaglægge for store dele af data—kommunikations–kanalerne mellem de enkelte sundheds––sektor institutioner.* [SLUT PÅ EKSEMPEL 30]

Man bemærker den stedse stigende "vaghed" i ovenstående maskin–krav–definitioner.

6.1.3.2 Afhængigheder: Mine afgrænsninger af nedenstående seks aspekter af begrebet afhængighed er illustrative. Mere præcise — og ud i "alle led" mere "korrekte, rammende" — definitioner kræver en mere teknisk og teknisk/videnskabelig forberedelse, noget der ligger uden for dette notats øjeblikkelige sigte.

De to første begreber: Pålidelighed og fejl–tolerance hører, som vi skal se, tildels sammen.

6.1.3.2.1 Pålidelighed: Ved 'pålidelighed' forstås et systems' evne til, over tid, at udvise en bestemt, forventet opførsel. Pålidelighed angives således, f.eks., i *antal fejl per udførte operationer* eller *timer,* eller *middel–tid mellem fejl,* eller *"oppe–tid" målt over (på anden vis) angivne perioder,* eller lign.

○ EKSEMPEL 31 ○

31. Anæstesi Udstyr: *Et bestemt Anæstesi–apparat skal sikres ekstrem pålidelighed: Én fejl per* 10^9 *fuldførte kirurgiske operationer hvor dette apparat blev brugt.* [SLUT PÅ EKSEMPEL 31]

Pålideligheds–problematikken er en kompliceret størrelse: Der er et stort spektrum af kilder til upålidelighed. Det er både en kunst, et håndværk, en disciplin, en logik, og en videnskab at kunne mestre et sligt spektrum.

6.1.3.2.2 Fejl–tolerance / Fejl—sikkerhed: Ved 'fejl–tolerance' ('fejl—sikkerhed') forstås et systems' evne til at "modvirke manglende" pålidelighed: Til — trods fejl i udstyr eller programmel, eller, og dette er udover karakteristikken af pålidelighed, i de data der præsenteres via menneske/-maskin grænsefladen — at udvise en forventet opførsel, der, om den ikke netop er den mest ønskede, dog lever op til visse minimum–krav.

Her antages pålidelighed f.eks. (typisk) udtrykt i form af *middel–tid mellem fejl.* Når fejlen så indtræder antages det da at denne, eller disse, fejl detekteres hvorefter en fejl–rettende funktion træder i kraft for at sikre at systemet stadig "fungerer", omend, for visse typer fejl, med eventuel nedsat ydeevne (mere begrænsede afhængigheds–mål).

◦ EKSEMPEL 32 ◦

32. EPJ: Patient–journaler: *Afvigelser i på hverandre flg. indførelser (registreringer) af måle–data fra kliniske analyser skal fejl–sikres mod "for store", fejl–registrerede målinger: Dvs., ikke–forventede afvigelser (udsving) skal lede til multipelt gentagne måleforsøg.* ... Herefter flgr. en nærmere præcisering af hvad der menes hermed. [SLUT PÅ EKSEMPEL 32]

Fejl–sikkerheds–problematikken er en kompliceret størrelse: Det kæmpe store spektrum af kilder til upålidelighed og de mange mere eller mindre ad–hoc'e, mere eller mindre systematiske måder man sikre sig imod fejl, betinger dette. Det er således både en kunst, et håndværk, en disciplin, en logik, og en videnskab at kunne mestre et sligt spektrum.

6.1.3.2.3 Nærværenhed: Ved *'nærværenhed'* forstås at et system er "oppe & kørende", tilgængeligt for passende autoriserede brugere.

En "nærværenheds"–krav–definition kan således formuleres i form af det antal timer per døgn systemet skal være "tilstede". Bemærk sondringen mellem pålidelighed og nær– dvs. tilstede–værenhed: Sidst nævnte skal gælde per døgn, før nævnte skal gælde akkumuleret over alle døgn for de tidsperioder hvor systemer skal være nærværende.

◦ EKSEMPEL 33 ◦

33. EPJ Systemet: *Et EPJ System skal ubetinget være nærværende (hele tiden "oppe") i perioden 5 am til 11 pm, hvorimod det gerne må være "nede" i perioden 11 pm til 5 am i op til to perioder på allerhøjst 10 minutter hver.* [SLUT PÅ EKSEMPEL 33]

6.1.3.2.4 Tilgængelighed: Ved *'tilgængelighed'* forstås at ingen funktion iøvrigt under udførelse af det ønskede programmel, dvs. den ønskede maskine, skal hindre andre, retteligt autoriserede brugere at benytte systemet

Dog kan man specificere grænser for akceptabel degradering af systemets "svartider" mv. i tilfælde af at særlige funktioner udføres. I denne karakteristik ligger også at den funktion, der eventuelt kan afholde andre brugere fra rimeligt stadigt at benytte systemet, kan være den at systemet "overbelastes".

◦ EKSEMPEL 34 ◦

34. EPJ Systemer: *Et spidsbelastet EPJ System skal ikke degradere svar–, hhv. afviklings–tider med mere end højst 100%.* [SLUT PÅ EKSEMPEL 34]

6.1.3.2.5 Sikkerhed: I forbindelse med sikkerhed antages det at der først er defineret et net af autorisations–nøgler, sådanne som tillader nøgle–ejeren at tilgå visse, eller alle, former for information og visse, eller alle, former for operationer på sådan information.

Et *system* siges nu at være *totalt sikkert* hvis ikke–autoriserede brugere, under "forsøg" på at tilgå ikke–autoriserede dele af systemet, (i) ikke kan finde ud af hvad systemet tilbyder af funktionalitet mv., (ii) ikke kan finde ud af hvorledes systemet iøvrigt "virker" (hvorledes det er implementeret, kodet), og (iii) endelig ikke véd om han véd !

Mao.: Systemet skal "fingere" en vis troværdighed, skal "lade som om", osv., men alligevel ej !

Som definition er den OK; som grundlag for kriterier for afprøvning om et system er mere eller mindre sikkert, et det OK; men giver i realiteten ingen retningslinier for implementering. Det er desuden tvivlsomt om man kan implementere *totalt sikre systemer*. Men som "rettesnor" kan ovenstående "idealiserede" karakteristik være nyttig. Bemærk forskellen (dvs. sondringen) mellem tilgængelighed og sikkerhed.

∘ EKSEMPEL 35 ∘

35. EPJ System: *EPJ Systemet skal muliggøre en (såkaldt fleksibel) tildeling af tilgangs–læse– såvelsom skrive–rettigheder til vilkårlige grupper af sundheds–fagligt personale, berørte patienter og disses pårørende (inkl. nære bekendte), og til vilkårlige dele af en patient's EPJ. (Med vilkårlige grupper, hhv. dele, forstås dog sådanne som aftales i samråd mellem mindst to mere generelt autoriserede sundheds–faglige personer og evt. patienten (etc.).) Disse tilgangs–rettigheder skal bestemme systemets sikkerhed overfor ikke–autoriseret indtrængning som flgr.:* ... Herefter flgr. en nærmere detaljering ... [SLUT PÅ EKSEMPEL 35]

6.1.3.2.6 Robusthed: Ved 'robusthed' forstås et systems' evne til at modstå også sådanne fejl som er eksplicit påtvunget systemet, f.eks. som er forårsaget af direkte fysiske eller elektroniske anslag mod systemet.

6.1.3.3 Vedligehold: Der kan tales om fire former for vedligehold af programmel. Vi giver ingen eksempler herpå.

6.1.3.3.1 Forbedrende Vedligehold: (*"perfective maintenance"*) Ved *'forbedrende vedligehold'* forstås sådanne ændringer til, eller udvidelser af allerede leveret programmel, der har til hensigt at effektivisere det modsvarende system's opfyldelse af (eksisterende, dvs. nu nye) maskin–krav.

6.1.3.3.2 Tilpassende Vedligehold: (*"adaptive maintenance"*) Ved *'tilpassende vedligehold'* forstås sådanne ændringer til, eller udvidelser af allerede leveret programmel, der har til hensigt at sikre at det eksisterende system kan tilsluttes nye former for eksternt udstyr, kan afvikles på andre platforme, etc.

6.1.3.3.3 Fejlrettende Vedligehold: (*"corrective maintenance"*) Ved *'fejlrettende vedligehold'* forstås sådanne ændringer til, eller udvidelser af allerede leveret programmel, der har til hensigt at rette fejl i det eksisterende system — hvor sådanne fejl ytrer sig ved system–opførsler, der ikke er foreskrevet mht., eller er i modstrid med, foreskrevne krav.

6.1.3.3.4 Forebyggende Vedligehold: (*"preventive maintenance"*) Ved *'forebyggende vedligehold'* forstås regelmæssig analyse af "log–optegnelser" — med deraf eventuelt følgende andet vedligehold.

Der er mao. indført et begreb: "en log–bog". Under brug af programmel–systemet skal eksekvering af en række "nøgle"–procedurer føre til optegnelser i denne log–bog. Disse "nøgle"–procedurer og disse "log–optegnelser" skal da være af en sådan art, at de berører, hhv. udsiger noget om, sådanne system–opførsler, der så igen, afhængigt af "log–optegnelsernes" værdi (de specifikke data som bliver optegnet), indikerer god eller dårlig system–opførsel, forventede eller ikke forventede system–reaktioner (indikatorer såsom: svartiders længde, lager–forbrug, kø–længde for access til visse serielt brugbare ressourcer, resultat af "på skrømt" udførte procedurer — hvor disse resultater kendes på forhånd — men hvor "afvigende" resultater kan forventes som følge af mulige system–problemer, etc.). Antagelsen er nu at man kan identificere sådanne "nøgle"–procedurer og sådanne tilhørende indikatorer at deres regelmæssige overvågning kan hjælpe med til, "før ulykken sker", at forudsige mulige fejl.

Forebyggende vedligehold kendes, typisk fra maskinel, hvis opførsel kan ændres som følge af materiel "nedslidning". Programmel "nedslides" ikke. Men, typisk aht. forbedrende vedlighold kan man måle hvorledes system–belastning ("brug") muligvis måtte indvirke på svar–tider, lager–forbrug, mv. Det er en kunst og et håndværk at identificere og at instrumentere passende log–bog procedurer etcetera !

6.1.3.4 Platforme: Ved *'platform'* forstås maskinel og sådant programmel der ikke er programmel, der skal udvikles særskilt som flg. af de krav der iøvrigt stilles, dvs. som typisk kunne være *"commercial, off–the–shelf"* programmel.

Man kan tale om tre former for platforme:

- Udviklings–platform,
- drifts–platform, og
- vedligeholds–platform.

For visse systemer, typisk alment kommercielle, såsom `MS Word`, kan der være sammenfald mellem de tre platforme. For andre systemer, som f.eks. programmel til satellitter, kan de tre platforme være tre vidt forskellige: Til eksempel et `Sun` arbejds–stations–baseret `Linux` system, den rum–baserede mikro–datamat, hhv. de i f.eks. i Houston, Texas, baserede satellit–sporende datamater.

Et generelt platforms–krav, der i en vis forstand også (dvs. istedet) kunne rubriceres under "anden ind/ud–datering", nævnes før de tre specikke system–platfome.

6.1.3.4.1 Installerings– og Opdaterings–Platform: Herved forstås (i) hvilket lager–medie der bruges ved levering af programmel–systemer samt af opdatering til sådanne, og (ii) hvilken læse–, dvs. ind–daterings enhed der skal bruges i forbindelse hermed.

◦ EKSEMPEL 36 ◦

36. EPJ System: *EPJ Systemet, samt opdateringer hertil, skal leveres på CD–ROM, med tilsvarende CD–ROM drev, af flg. typer: ...* [SLUT PÅ EKSEMPEL 36]

6.1.3.4.2 Udviklings–platform: Ved *'udviklings–platform'* forstås den platform, subsidiært de platforme, på hvilken, hhv. hvilke, det ønskede programmel skal udvikles.

6.1.3.4.3 Drifts–platform: Ved *'drifts–platform'* forstås den platform, subsidiært de platforme, på hvilken, hhv. hvilke, det ønskede programmel skal danne grundlag for afvikling (dvs. "kørsel").

◦ EKSEMPEL 37 ◦

37. EPJ System: *EPJ Systemet skal afvikles på `IBM PC`–kompatible datamater med `Microsoft Windows 1998` kompatibelt programmel.* Etcetera. [SLUT PÅ EKSEMPEL 37]

6.1.3.4.4 Vedligeholds–platform: Ved *'vedligeholds–platform'* forstås den platform, subsidiært de platforme, på hvilken, hhv. hvilke, det ønskede programmel skal kunne vedligeholdes.

6.1.3.5 Indlæring: Herved forstås krav vedrørende hvilke forudsætninger personale og "tilfældige" brugere ("*casual users*") skal have for at kunne indlæres i brugen af systemet, hhv. hvor meget instruktion, i tid, der maksimalt skal bruges i en sådan i ndlæring.

6.1.3.6 Dokumentation: Ved *'dokumentation'* forstås sådanne skrifter, på papir og/eller elektronisk form, der informerer om, beskriver og analyserer et bestemt genstands–område.

Eksempler på genstands–område omfatter bl.a. de nedenfor beskrevne.

6.1.3.6.1 Udviklings–dokumenter: Ved *'udviklings–dokumenter'* forstås sådan dokumentation, der omhandler udvikling af én eller flere af flg.: domæne–beskrivelse, krav–definition, og/eller programmel–specifikation.

◦ EKSEMPEL 38 ◦

38. EPJ System: Det ønskede EPJ System er og forbliver det Danske Sundhedsvæsens ejendom, også selvom det udvikles, på kontrakt, af private virksomheder, og/eller, i en vis, mindre udstrækning af amtlige sundhedsvæsener. *Den fulde, komplette og konsistente (informative, beskrivende og analyserende) dokumentation af alle tre trin af programmel–udviklingen af et EPJ System: Domæne forståelse, krav–definition, og programmel–specifikation skal leveres som en del af et samlet udviklings–forløb, også selvom dette måtte "stykkes" ud i mange separate og ikke nødvendigvis kontrakt–relaterede udviklinger.* [SLUT PÅ EKSEMPEL 38]

6.1.3.6.2 Installations– & Drifts–dokumenter og Kurser: Ved *'installations– & drifts–dokumenter og kurser'* forstås sådan dokumentation og eventuelle kurser der omhandler installation og drift på bestemte platforme af leveret programmel.

∘ EKSEMPEL 39 ∘

39. EPJ System: *Der skal med det leverede EPJ System følge et komplet og konsistent sæt installations– & drifts–dokumenter, der muliggør at EDB operatører med almindelig erfaring i installation og drift af de platforme hvortil EPJ Systemet er krav–defineret, selv kan installere og dagligt kan varetage dettes drift. Flg. dele af installations– & drifts–dokumenterne skal findes på flg. elektroniske ("help") form:* ... Herefter flgr. nærmere specifikation af sådanne specifikke former ... [SLUT PÅ EKSEMPEL 39]

6.1.3.6.3 Bruger–manualer og Kurser: Ved *'bruger–manualer og kurser'* forstås sådan dokumentation og eventuelle kurser der indsætter "almindelige brugere" (indenfor den i krav–definitionerne definerede bruger–skare) i den daglige brug (benyttelse) af systemet.

∘ EKSEMPEL 40 ∘

40. EPJ System: *Samtlige potentielle brugere af det leverede EPJ System skal tilbydes tre halv–dags–kurser i brug af systemet: Et generelt (sundheds–sektor–orienteret), et specifikt (arbejds–gruppe–orienteret), og et (person–ansvars–områder) detaljeret kursus.* ... Disse beskrives nærmere .... *Respektive kursus–mål er som flgr.:* ... Disse beskrives nærmere .... [SLUT PÅ EKSEMPEL 40]

6.1.3.6.4 Vedligeholds–dokumenter: Ved *'vedligeholds–dokumenter'* forstås sådan dokumentation der tillader (nærmere beskrevet) kvalificerede programmel–udviklings–ingeniører — f.eks. folk der ikke nødvendigvis har været med i den oprindelige udviklings–gruppe for et pågældende programmel — selv at vedligeholde dette programmel.

∘ EKSEMPEL 41 ∘

41. EPJ System: *Det leverede EPJ System skal leveres med et komplet og konsistent sæt dokumenter der sammenholdt med udviklings–dokumentationen (nævnt ovenfor) skal* ... OSV. ... [SLUT PÅ EKSEMPEL 41]

6.1.4 Diskussion

I hele dette afsnit vedr. krav skal vi primært berøre sådanne krav som klienten direkte måtte ønske at fremsætte. Derudover kan komme krav der mere er udsprunget af udviklernes ønsker, eller som er "anden ordens, afledede" krav. Vedligeholds–krav kunne siges eventuelt at medføre krav om modularitet, og korrektheds–krav kunne siges eventuelt at medføre krav om afprøvning (*"test"*).

6.1.5 Meta–krav

Standard eksemplet på et meta–krav er forventningen om at systemet er lyde– og fejlfrit: At det opfylder krav–definitionen og at det passer ind i domænet. Dette krav burde egentligt være så selvfølgeligt at det "bare" kan antages. Men virkeligheden er desværre den at man

kan stille mærkværdige afhængigheds–krav som at der højst må være én fejl per 10.000 linjers kode (!), eller der højst optræder én fejl per 12.000 person–timers brug !

Klienter indforskriver programmel ofte som et led i deres ønske om bedre lønsomhed for deres firma. At udtrykke, som et krav til det ønskede programmel, at det skal gøre firmaet mere lønsomt betragter vi her som et meta–krav. Det er ikke et krav der direkte kan omsættes til programmel–specifikation (design). Det er "for luftigt", for generelt. Det har intet med de dele af domænet som vi normalt kan beskrive. Det er et grundlæggende problem: At udtrykke de forhold hvori, hvorpå et firma's lønsomhed består, hhv. beror. Vi har i begyndelsen af Afsnit 6.1.1 (startende på side 36) listet nogle af de meta–krav der sædvanligvis formuleres.

Vi kan ikke, idag, præcist indkredse alle aspekter ved meta–kravs–begrebet, men gør her opmærksom på at klienter ofte blander de to forhold sammen: Intentionelt opfyldelige domæne–krav med kun meget indirekte opfyldelige meta–krav.

Meta–krav er mao. krav, der dels antager en beskrivelse af domænet der som regel ikke foreligger, og dels antager at emnet (de refererer til) er beregneligt.

Vi gentager, i forkortet udgave, de meta–krav der er anført i Afsnit 6.1.1 (startende på side 36), samt kommenterer disse:

- Lavere lønmæssige omkostninger:
  - At formulere dette som et almindeligt krav, dvs. at rykke det fra meta–krav til domæne–krav, vil kræve at man nøje modellerer alle personale omkostninger ved alle arbejds–processer, også de der ikke datamatiseres, mmm.
- Fra monotont eller trættende menneskeligt arbejde til datamat–styret arbejde:
  - At formulere dette som et almindeligt krav, dvs. at rykke det fra meta–krav til domæne–krav, vil kræve at man kan modellere hvad det vil sige at en arbejds–process er monoton eller trættende — hvad man idag ikke kan gøre på en sådan måde at en sådan model kan danne grundlag for system design, der kan siges at opfylde et sådant krav.
- Størst mulig pålidelighed i udførelsen af arbejds–processer:
  - At formulere dette som et almindeligt krav, dvs. at rykke det fra meta–krav til domæne–krav, vil kræve at man kan modellere hvad det vil sige at en menneskelig eller ikke–datamatiseret (mekanisk) arbejds–process er upålidelig, hhv. at en datamatisert sådan er (mere) pålidelig. Meget gøres sikkert (dvs. muligvis) i denne retning for at indkredse problemet, men ikke idag til et sådant niveau at vi kan opstille beregnelige modeller herfor.
- Hurtigere arbejds–processer:
  - Her er sikkert noget man generelt kan gøre mere ved — men det tages normalt for givet — og deri "ligger humlen begravet".
- Nye arbejds–processer, der førhen var "umulige":
  - Også dette synes opnåeligt, men måske det er trivielt, dvs. en selvfølge.

- *&c.*

Det er et meta–krav om man definerer produktvitets–forøgelse, f.eks. i form af afviklede kunde–transaktioner per sags–behandler.

6.1.6 Diskussion

Her slutter vi vores lange krav–definitions Odyssé. Meget mere kunne siges. Det som er sagt (skrevet) illustrerer til fulde kompleksiteten af krav–definitions–processen og dens resultater.

6.2 Fra Krav til Programmel

Der findes nu en lang række metoder, principper, teknikker og tildels værktøj, der assisterer i den videre udvikling, fra krav–definitioner via programmel–specifikationer til kode.

Vi skal blot illustrere nogle nutidige begreber: Arkitektur, Program Organisation & Struktur, Modularitet, Objekter, UML og OMG.

Mht. sondringen mellem Arkitektur og Program Organisation & Struktur blot dette. For typisk sikkerheds–ukritiske, dvs. typisk sædvanlige administrative EDB systemer "tackler" udvikleren først arkitektur–aspektet, siden organisations– & struktur–aspektet. For kritisk fejl–tolererende systemer "tackler" man ofte disse to i omvendt følge !

6.2.1 System– og Programmel–Arkitekturer

Vi har forsømt i dette notat (men ikke i vores sædvanlige skrifter og forelæsnings–noter) at sondre mellem udvikling af systemer — hvorved forstås maskinel + programmel, og programmel alene. Hvor vi i det tidligere alene skrev programmel, kan læseren, stille og roligt indsætte: "maskine plus programmel" !

Ved 'arkitektur' skal vi her forstå den del af en (system cum) programmel–specifikation der varetager implementeringen af alle domæne– og visse (om ikke alle) grænseflade–krav.

6.2.2 Program Organisation & Strukturer

Ved 'program organisation & struktur' skal vi her forstå den del af en (system cum) programmel–specifikation der varetager implementeringen af visse (om nogen) grænesflade– og alle maskin–krav.

6.2.3 Modularitet, Objekter, UML, OMG, mmm.

Modularitet, som begreb, har forhåbentligt været ledende i alle beskrivelser, fra domæne, via krav, til (system cum) programmel.

Ved modularitet forstås en tekstuel afgræsning af en beskrivelse således at denne, på op til maksimalt 2–3 sider A4 ark, beskriver en "sammenhørende" mængde fænomener: information indeholdt i, funktioner over, begivenheder involverende og opførsler af processer for disse.

Ved et objekt forstås et instantieret modul, en størrelse der, til illustration, kan forestilles at være en process, med egen tilstand, under udførelse samtidigt med andre processer (dvs. objekter) med hvilke det eventuelt kan udveksle information (f.eks. gennem meddelelser).

Ikke alt i en aktuel dagligdag er objekter, men noget er. Når vi fra domæne–beskrivelser via krav–definitioner og programmel–arkitektur når frem til program organisation & struktur, da er det at det virkelig har vist sig, siden `Simula'67` (1967), at modularisering og objekter kan være endda overordentligt nyttige programmel–størrelser.

`UML` er en idag, 2001, meget "fashionabel" måde at anskue programmel–udvikling på. Dog besidder, réelt set, `UML` ikke nye revolutionerende tanker udover f.eks. sådanne som allerede var gængse i lang tid før fremkomsten af `UML`. Statechart har været brugt nogen tid før `UML`, hvori det også er "indlejret". Tilsvarende for *"Message Sequence Charts", Petri Net,* oa.

Når det er sagt at modularisering og objekter er nyttige ihvertfald i mellem–trin af programmel–specifikation (–design), så skal der også her "advares" mod visse "ny–religiøse" tendenser omkring `UML`. Årtiers dybe viden om konstruktion af og teorier for endda overordentligt nyttige, abstrakte specifikations– såvelsom konkrete programmerings–sprog synes i stor grad at være gået "hen over hovedet" på `UML`: Mange, de fleste, abstrakte specifikations– og konkrete programmerings–sprog har i årtier "tilbudt" objekt–orientering på et langt sikrere grundlag end `UML` gør det idag. `UML` savner derudover flg. endda overordentligt vigtige egenskaber: (i) Det har ingen sikker semantik, (ii) det har ikke noget reelt abstraktions–begreb, (iii) man kan ikke rimeligt logisk ræssonere over `UML` diagrammer (mm.), og (iv) `UML` har ingen implementations–relation: Dvs., man kan ikke vide om et stykke Java kode repræsenterer en korrekt implementering (realisation) af en `UML` specifikation. `UML` specifikationer har det med at specificere design i ét med krav–definition. Mao., `UML` specifikationer sammenblander domæne–beskrivelse, krav–definition og programmel–specifikation.

OMG ("Object Management Group") leverer en seriøst udarbejdet og nyttig samling definitioner af en lang række "services" der forventes, og/eller som har vist sig, at være nyttige i mange forskellige sammenhænge. *OMG*'s definitioner bygger, som navnet antyder, på objekt-orientering.

6.2.4 Diskussion

Trods de måske mere kritiske bemærkninger omkring `UML`, skal man ikke blankt forkaste objekt–orientering. Men man skal ej heller tro at objekt–orientering tilbyder et *"cure all"*, eller at det bliver sidste, endelige, definitive svar på programmel–ingeniørens trængsler.

6.3 Diskussion

6.3.1 Generelt

Der kommer nye "fashions", nye tider. Og vi kan forvente, som det dog har vist sig, at mere professionelt uddannede programmel–ingeniører, i stedse stigende takt vil optage de mere formelle metoder, principper, teknikker og værktøj.

Vi har i dette stor–afsnit skitseret overgang fra domæne–beskrivelse til krav–definition, og fra disse til programmel–specifikation. En stor "mundfuld" !

6.3.2 Vedr. Påstanden: Krav ændrer sig til stadighed !

Det fremføres ofte at krav til stadighed ændrer sig — så hvorfor være så nøje, som her anført, med krav–definitions–fasen ? Svaret herpå er selvflg. ganske enkelt: Det er en illusion at tro

at krav til stadighed ændrer sig — ihvertfald ikke domæne–krav. Domæne–krav udtrykkes, som antydet, vha. domæne–termer, og disse er utroligt stabile. De ændrer sig meget lidt, *kun en gang hvert andet årti !* Det er teknologien, ved hvis hjælp vi implementerer (realiserer) krav, der ændrer sig, og dermed er det i bedste fald højst grænseflade– og maskin–krav der ændrer sig. Bemærk dog vores måde at definere de forskellige facetter af maskin–krav på. Deres definition gør, om man, i sine oprindelige, specifikke maskin–krav, følger definitionerne (karakteristikkerne), at selv disse kun ændres kvantitativt, ikke kvalitativt.

Men "nye" domæne–funktioner, der førhen ikke var reelt beregnelige, bliver nu tilgængelige, er mulige, via datamaten: Banker, forsikring og kredit–institutioner tilbyder nye services, og alle disse er udtrykt vha. gængse domæne–termer.

Mao. det kan betale sig nøje at forstå domænet, for derved har man garderet sig mod at nye domæne–krav eventuelt skulle "vælte båden" ved signifikant at kræve fundamentalt nye programmel–arkitekturer, organisationer & strukturer. Ved at sikre at de basale programmel–arkitekturer, organisationer & strukturer reflekterer basale egenskaber ved domænet, hhv. de fundamentale maskin–kravs kategorier (effektivitet, afhængighed, vedligehold, etc.), sikrer man sig mod en eventuel, dvs. ellers nødvendig ændring i programmel–arkitekturer, organisationer & strukturer. Lignende betragtninger gør sig gældende for grænseflade–krav og deres deraf afledte programmel–arkitekturer, organisationer & strukturer.

7 Informatik = Matematik ⊕ Datalogi ⊕ Anvendelser

Ved informatik forstås en konfluens, en "sammensmeltning", af matematiske og datalogiske discipliner sammen med deres brug ved udvikling af datamatik til bestemte anvendelser — såsom f.eks. datamatik til infra–struktur–komponenter.

Ligesom bio–teknik er teknologi, der løser problemer af biologisk art, og som derfor bygger på biologi (som en lære), således er informations–teknologien (IT) informatikkens teknologiske modsvarighed. I Danmark bruger man til stadighed begrebet IT, hvor man som oftest mener informatik; men man sondrer klart mellem biologi og bio–teknologi. Dette skyldes sikkert mediernes "dims"–begejstring: Fascinationen med informations–teknologiens materielle univers i hvilket "fremskridt" måles kvantitativt: Større ydeevne (hukommelse, lagerplads, "Giga–bytes"), mindre pris, mindre volumen (plads), hurtigere udførelse. Istedet burde man snarere beskæftige sig med informatikkens kvalitative og intellektuelle univers: Bedre EDB løsninger: Venligere tilgang til data–systemerne, større affinitet til genstands–området, korrekt (fejlfri) data behandling, mv.

Sammensmeltningen, der nævnes ovenfor, beror (i) bl.a. på informatik–ingeniørens benyttelse af lingvistiske elementer: Beskrivelse af fænomener; (ii) disses "nær–filosofiske" udskillelse (identifikation og afgrænsning; "hvad kan beskrives" [hvad kan erkendes]) fra et større genstands–område; (iii) sondring mellem pragmatik, semantik og syntaks — samt (iv) på at dette sker indenfor en ellers stringent, matematisk verden, samtidig med at det, som kan krav–defineres, skal kunne beregnes. Domæne–beskrivelser er ikke bundet af 'beregnelighed'. Så der er mao. lagt op til en "spændende" overgang fra det bredere spektrum af domæne–beskrivelser (inkluderende ikke–beregnelige såvelsom beregnelige modeller af fænomener) til et smallere spektrum af nødvendigvis beregnelige krav–definitioner. Overgangen er ikke, i traditionel forstand, ingeniørmæssig.

Hvor elektronik–, maskin–, bygnings– og kemi–ingeniørens fag–discipliner eksisterer i kraft af, fokuserer og bygger på, natur–videnskabelige fænomener, er dette ikke tilfældet for informatik–ingeniøren: Dennes verden bygger på, er begrænset alene af matematikkens og meta–matematikkens verden.

Begrebet 'beregnelig' er tidligere i dette notat fremført og anvendt. Det er ikke alt der kan beregnes. Og selvom en ting kan beregnes, selvom denne ting kan formuleres klart, logisk, er det ikke sikkert at en beregning er praktisk mulig.

Det er således ikke muligt, for et vilkårligt generelt anvendeligt programmerings–sprog, at konstruere en programmerings–sprogs—oversætter (*compiler*) der, givet et vilkårligt, syntaktisk velformet program, kan afgøre om dette program går ind i en uendelig løkke når det ligger til grund for datamat–eksekveringer, mao. om det aldrig stopper.

Et andet aspekt "skiller" programmel–teknologi fra anden teknologi. Vi tager som eksempel på en sådan anden teknologi den elektroniske (mm.) teknologi, der indgår i konstruktion af datamaskiner. Lidt slag–ords–agtigt kunne man mene: Der opstår hvert 5te eller ca. år en ny datamat–teknologi og denne nye teknologi bruges på at løse det samme problem: At konstruere (fremstille) datamaskiner. I kontrast hertil bruges – i informatikken — den samme "intellektuelle 'tænke' teknologi": Matematik, til, hele tiden, at løse nye problemer !

"Moralen" af dette (og de foregående) afsnit er at informatik, læren om og praksis af

informatik, udgør en hel ny ingeniørmæssig disciplin: At man langt fra har forstået at kommunikere en lang række budskaber til klienterne: Hvad der kan beregnes, hvorledes problemer (domæne– og krav–problematikken) kommunikeres mellem klient og informatik–leverandører. Mao., at vi langt fra har forstået hvorledes klienter og informatik–leverandører bedst kan arbejde sammen med henblik på en rimelig (optimal) udnyttelse af informatikkens muligheder.

8 Forandrings–ledelse

Der henvises, til indledning, til Afsnit 1.6 på side 10. Visse forhold, der er nævnt i Afsnit 1.6, vil ikke blive gentaget her.

Det er fremført at en væsentlig årsag til de mange såkaldte "EDB–skandaler" vi jævnligt læser om skyldes klientens top–ledelses' manglende "opbakning", herunder forståelse for de videre konsekvenser af at indføre informatik.

8.1 Problemet

Opstilling, det "lange", nøjagtige, omhyggelige, detaljerede, tilbundsgående arbejde med at opstille de multi–perspektiverede, fler–facetterede domæne–beskrivelser, medfører, helt naturligt, et ret så nyt, afklaret syn på den virksomhed, den offentlige administration, det private firma, som domæne–beskrivelsen omhandler. Dette ny–syn, processen hermed, medfører ofte, resulterer ofte i, et presserende ønske at omgøre, at redefinere virksomhedens måde at arbejde på. Domæne–beskrivelserne resulterer som oftest i en skarpere forståelse af virksomhedens mangfoldige fænomener, i dets "opførsler", "begivenheder". Sådanne, mere nuancerede, mere "objektive" forståelser medfører naturligt ønsket at omgøre institutionens arbejdsgange, som ofte ret så radikalt. Datalogiens, informatikkens, begrebs–verden, dens iboende trang til abstraktion, konceptualisering, overføres på sædvanlige genstands–områders som oftest ret så konkrete begrebs–verden. Alt dette åbner muligheden for at tilføre sædvanlige, "klassiske" genstands–områder's begrebs–verden nye, mere abstrakte, som oftest dermed mere generelt anvendelige, begreber, og dermed at lade disse "styre" basale forhold i et klassisk genstands–område.

Vort næste eksempel viser hvorledes telekommunikation og dermed fordelte (distribuerede) og samtidige beregninger muliggør at et "ældgammelt" begreb: Godstogs–rangering (*marshalling*) indenfor dertil særligt indrettede godstogs rangér–terræner (*marshalling yards*), muligvis bør, dvs. potentielt helt kan, "erstattes" af distribueret rangering (*shunting*) indenfor de allerede eksisterende banegårde.

∘ Eksempel 42 ∘

42. Abstrakt Konceptualisering: Godstogs–rangering: *Idag, i domænet, "rejser" en typisk godsvogn via mange forskellige godstog fra en oprindelig udgangs–station til en endelig slut–station. Skift, af en godsvogn, fra et godstog til et andet godstog sker, idag, ved at godstoget, hvori godsvognen befinder sig, "omrangeres" på et rangér–terræn.*

• • •

Omrangeringen kan beskrives, abstrakt, som følger: En 'ind'–sekvens of godsvogns–stammer omdannes til en 'ud'–mængde af godsvogns–stammer således der eksisterer en én–til–én korrelation mellem godsvogne i 'ind'– og 'ud'–stammer.

• • •

Konkret forestiller man sig rangér–terrænet som bestående af et indgangs–spor, en "pukkel", og en udfletning fra denne "pukkel" til en vifte af ud–rangérings–spor, der så igen samles (ind–flettes) til et udgangs–spor. Enhver 'ind'–godsvogns–stamme føres henover "puklen" hvorfra de enkelte, afkoblede vogne "falder" ned mod én af 'ud'–godsvogns–stammerne — dirigeret imod en sådan vha. tilsvarende foreskrevet indstilling af de mange spor–skifter

der muliggør viften af ud–rangérings–spor. 'Ud'–stammerne køres siden "væk" fra rangér–terrænet — hvorefter enhver godsvogn nu potentielt er på vej til næste destination.

• • •

I ovenstående traditionelle realisering af enhver bestemt godsvogns' sammensatte rejse, over ofte meget store afstande mellem virkårlige to stationer, bestemmer, implementerer, opholdet på et rangér–terræn (under nøje overvågning og styring af rangér–terræn–sporskifter, rangér–planer og personale), det som passagerer selv sørger for, når de skifter tog på en station: Ved, ved egen kraft, drevet af en "indre, privat" plan, at vandre (haste, eller løbe) fra én perron til en anden. De fleste stationer er passager-stationer. Meget få af disse har eget rangér–terræn. Regions-centrale rangér–terræner "deles" mellem mange lokale stationer. Traditionen har medført at godsvogns–stammer og dermed godsvogne har opholdt sig i endda overordentligt lange perioder på de fleste rangér–terræner. Man har taget den med ro: Der har ikke, i de nationale jernbane–selskabers monopol–periode, været nogen særlig motivation for at fremskynde om–rangering.

• • •

Idéen er nu, i en vis forstand, at lade lokal-stationer udføre al form for godsvogns–rangering: Afkobling, shunting, midlertidigt, kortere ophold på et godsvogns-sidespor, fulgt af shunting og sluttelig tilkobling til et "næste" godstog. Istedet for en central, "off–line" omrangering sker sådanne nu distribueret, med mulighed for meget snævre tids–begrænsninger for afkobling, shunting, midlertidigt ophold, shunting, og tilkobling.

• • •

For at muliggøre dette, herunder specielt at muliggøre hurtigere om–rangering, kræves det nu at hvor der før var et vist, mindre antal rangér–terræner, at enhver almindelig station foretager om–rangering baseret på meddelelser og planer der "løbende indløber". Der kræves ikke nogen særlig "pukkel", ingen "vifte", etc., blot at der er et rimeligt enkelt to–vejs–net af side—spor, et rangér–lokomotiv, samt passende lokal ledelse: Overvågning (monitoring) og styring (control). De, til stadighed "indløbende" meddelelser og planer, kræver passende telekommunikations–systemer. Ledelse af, og selve om–rangérings–arbejdet kræver datamatiseret understøttelse, og dette (det datamatiserede arbejde) genererer siden nye meddelelser og planer — og disse sendes "videre": "Op og ned" ad linjen.

• • •

Måske de store godstogs–rangér–terræner's tid er måske forbi, men omstillingen kræver stor ledelses–forståelse, forberedelse og opbakning. [Slut på Eksempel 42]

Eksemplet ovenfor viser at nye begrebs–dannelser kan lede til endda overordentligt signifikant nedsatte omkostninger såvelsom tilsvarende nedsatte gods–transport & ekspeditions–tider.

Vort næste eksempel viser, dog, hvorledes ny teknologi, medfører relativt nye koncepter.

∘ Eksempel 43 ∘

43. Ny, "Revolutionerende" Teknologi: EPJ + Medicinerings System: *Vi forestiller os et system hvori en aktuel patient's aktuelle indtagning af ordineret medicin registreres automatisk i denne patient's EPJ. Dette kunne gøres vha. f.eks. én eller flere af flg. måder. På den ene side indeholder hver af de medicin–afgivende midler: Insulin–sprøjten, pille–dåsen,*

oa., en sensor–og–sender/modtager der registrerer at medicin bliver "fjernet", og, på den anden side findes der i patientens krop en tilsvarende elektronisk/kemisk–biologisk sensor–og–sender/modtager–mikro–maskine, der "sejler" rundt i blodårene og som er istand til, i kontakt med lever, nyrer eller andre organer, eller uafhængigt af disse, at registrere at den afgivne medicin aktuelt er blevet modtaget. Disse to former for mekanisk/elektronisk, hhv. elektronisk/kemisk–biologisk sensor–og–sender/modtager–mikro–maskiner, kommunikerer, trådløst, med et distribueret opsamlings– og fordelings–system, der så igen står i forbindelse med EPJ Systemet. Med et sådant system kan f.eks. kritisk syge patienter overvåges på en helt ny måde, og med store forandringer i sundheds-sektorens arbejds–gange til følge.

[Slut på Eksempel 43]

Indførelse af nye, basale begreber i et klassisk genstands–område kræver, som oftest, omstilling: Fra top–ledelse helt ud i de yderste led af virksomheden.

Og her er det at "tingene" begynder at gå galt: Som oftest rekvirerer (akcepterer) top–ledelsen (at) nye informatik–systemer (indføres). Men, som oftests er samme top–ledelse ikke indstillet på at sådanne informatik–systemer medfører *"omstilling: Fra top–ledelse helt ud i de yderste led af virksomheden."* Og når en sådan "forberedthed, villighed, åbenhed" ikke "sanses", i dybden, af top–ledelsen, så skal det gå galt.

Mange, om ikke alle de nutidige, seneste, såkaldte EDB–skandaler, kan henføres, i det væsentligste, til denne omfattende, basale 'uforberedthed'. Mao.: Ledelsen, der reelt set har beordret en omfattende domæne–beskrivelse, kan ikke "leve op til dennes konsekvenser".

8.2 Mod en Generel Løsning

Vi ser ingen anden løsning på dette basale problem end at man, informatik–leverandøren i særdeleshed, fra en tidligste begyndelse — fra før domæne–beskrivelse — sikrer sig at rekvirentens top–ledelse genuint, ærligt, er forberedt på og er moden til "forandring"; at den har viljen og magten til at gennemføre nødvendige forandringer.

Hensigten med dette notat er at påpege at en omhyggeligt og detaljeret gennemført domæne–beskrivelse kan danne grundlag for en sober, troværdig *"business process re–engineering"* — og dermed danne et teknisk grundlag for forandrings–ledelse. Kun en troværdigt udtømmende virksomheds–, dvs. domæne–beskrivelse kan sikre rimelige estimater af hvad det vil kræve at omgøre en virksomheds' organisation og ledelse ud i alle led.

8.3 Mod en Speciel Løsning

I forbindelse med udvikling af visse typer store programmel–systemer for visse typer infra–struktur–komponenter (eller dele deraf) kan det være enten nødvendigt eller formålstjenligt at klient og udvikler indgår i et snævert, dog klient–betalt samarbejde således at domæne–beskrivelse, krav–definition og programmel–specifikation (inklusive aktuel kode) udvikles samtidigt, måske følgende én eller anden form for "spiral"–process.[33]

[33] Forfatteren takker Gert Mikkelsen, *PriceWaterhouseCoopers*, for idéen til dette afsnit: Den her beskrevne måde at udvikle programmel synes anvendt, med stor sukcess, af Dansk *PriceWaterhouseCoopers'* DPS afdeling i udvikling af en bestemt klasse logistik–systemer — ikke netop den der er beskrevet i Appendiks B.2 (startende på side 87).

Mao.: Der udvikles ikke først en mere eller mindre komplet domæne–beskrivelse, dernæst en mere eller mindre komplet krav–definition før programmellet (specifikation mm.) udvikles. Istedet kan man forestille sig at et "råt–skitseret prototype–programmel–system" hurtigt etableres på grundlag af tilsvarende "råt–skitserede" domæne–bekrivelser og krav–definitioner. Klient og udvikler (kunde og leverandør) "afprøver" nu sammen denne "prototype" (der sikkert blot har nogle meget få, måske blot "antydede" funktioner). Denne "afprøvning" kan f.eks. have til hensigt, at validere (i) "mest betydende" dele af den gensidige domæne–forståelse, (ii) tilvarende "mest betydende" dele af krav–definitionen, samt (iii) "mest betydende" dele af programmellet, herunder dets grænsefladen til brugerne. Dernæst itereres domæne–beskrivelse, krav–definition, og programmel–specifikation (inklusive aktuel kode) med yderligere trinvise validerende afprøvninger.

Bemærk at vi egentligt ikke har opgivet vore triptych, blot omfortolket dens "udførelse" !

Typisk kunne man ønske at anvende den ovenfor beskrevne "spiral"–model i situationer hvor klienten er villig til f.eks. fast–pris–aftaler for hver iteration uden på forhånd fastsatte mål, men med akcept af at visse forventningers ambitions–niveau eventuelt "neddrosles" (eller "opskrues"). Sådanne situationer kunne forestilles relevante for udvikling af sådanne informatik–systemer hvor klienten, dels har stor tillid til programmel–udviklings–firmaet, dels ønsker, gennem et snævert samarbejde i alle faser, at afprøve egne forestillinger om domænet, og af krav, samt eventuelt at eksperimentere med forskellige *"business re–engineering"* modeller, eller måske blot at indføre disse, dvs. forandrings–ledelsen, sukcessivt for derved at sikre en bedre, mere smidig institutionel akcept.

9 Nogle Sociologiske og Psykologiske Betragtninger

9.1 Et Vedvarende Problem: Programmel–Amatører

Fordi man, i `Lego` klodser, kan bygge en legetøjsbro, er det langt fra sikkert at man egner sig til at blive bygnings–ingeniør, endsige brobygger. Fordi man tilsyneladende kan "hakke" et `Visual Basic` program sammen, for slet ikke at tale om blot at lave web sider eller programmere `Linux` system–funktioner, er det slet ikke spor sikkert at man egner sig til at forestå udvikling af administrative systemer til arbejds–anvisning, elektroniske patient–journaler (EPJ), mmm. Man har ihvertfald ikke godtgjort det. Der skal mere til. Dette notat antyder at den professionelle data–ingeniør cum programmør cum informatik–ingeniør skal beherske mange discipliner. Vi kan nævne en række discipliner indenfor datalogi som ikke har været berørt før, men som der er bred enighed om at en professionel datalog (cum data–ingeniør cum programmør cum informatik–ingeniør) skal beherske:

- Formelle sprog og automat–teori (regulære–, kontekst-fri–, kontekst–følsomme— oa. sprog; endelige tilstands automater og maskiner, stak automater og maskiner)
- Beregnelighed: Lambda kalkyle, Turing maskiner, m.fl.
- Algoritmer og data—strukturers kompleksitet
- Funktionelle programmerings–sprog og funktions programmering (Standard ML) "med samt" rekursiv funktions–teori og fikspunkter
- Imperative programmerings–sprog og programmering "med samt" Hoare bevis–kalkyler
- Logiske programmerings–sprog og logik programmering (Prolog) "med samt" matematisk logik: Udsagns–, første og højere ordens prædikat–kalkyler
- Parallelle programmerings–sprog og parallel programmering "med samt" Petri Net, CSP, ccs, π–kalkyle, Statecharts, Process algebraer
- Bevis–systemer, beviser, bevisførelse, bevis–taktikker & –strategier, og bevis–værktøjer
- Model–checking
- Type teori: Rekursive typer, afhængige typer, parameteriserede typer, multipel nedarvning, kontravarians
- Algebraisk, denotations, axiomatisk og strukturel operationel semantik
- Program–analyse, abstrakt fortolkning, partiel evaluering
- Videnbaserede systemer, videns–repræsentation, modal logikker, *"speech acts"*, multi–agent systemer, etc.
- *&c.*

Ovenstående er blot et mindre udpluk. Og så har jeg slet ikke nævnt de kalkyler og teori–dannelser der ligger nærmest dette notats emne !

Disse og mange andre del–emner indenfor datalogi udgør den professionelle datalogs' teknisk/videnskabelige ballast. Dertil kommer flere andre emner der er vigtige for projekt– og produkt–ledelse.

Man bemærker at vi ikke har nævnt sådanne "populære", og for mange nyttige emner som f.eks. objekt–orienteret analyse og modellering, herunder UML. OO er en del, udspringer, af de før nævnte, mere formelle discipliner — og UML er blot en nutidig teknologi. Den vil sikkert hedde noget andet om 10 år — ligesom man for 10 år siden fylkedes omkring andre etiketter og slagord.

Moralen af dette under–afsnit er at langt de fleste af hverdagens programmel–huse (i Danmark og andetsteds) idag ikke besidder den fornødne viden om de ovenfor opregnede emner, endsige behersker dem til den grad det her skønnes nødvendigt.

9.2 Cheferne er ikke Professionelle Dataloger / Informatikere

Dagens kandidater fra kemi–, elektronik–, bygnings– og maskin-retningerne ansættes, i den udstrækning de ansættes indenfor deres klassiske fag, i virksomheder hvis ledelse tilsvarende har en klassisk dannelse indenfor samme disciplin: Stort set læst ("tyret over") de samme lærebøger — ihvertfald indenfor enkelt–fagenes grund-discipliner.

Men indenfor programmering, i programmel–huse, og især typisk for de virksomheder der engang startede som elektronik–virksomheder, men som (oftest umærkeligt, og især for ledelsen ikke til fulde forstået), gled over i programmering, datamatik, indenfor disse virksomheder (og det er stort set langt, langt de fleste, de berømmelige 99%), dér er ledelsen ikke datalogisk disciplineret, har ikke den fornødne pli ! Katastrofen udebliver som regel ej heller.

9.3 Laveste Fællesnævner

Resultatet af ovenstående: Fra dilletantisk til, i bedste fald, amatøragtig programmering, udebliver ej. Store programmel–udviklings–projekter savner den fornødne professionalisme. Det går, for en tid. Og så indhenter fortidens synder firmaet. Vedligehold af arkaisk programmel "koster det hvide ud af øjnene"; man evner ikke at "dreje om på en tallerken" og kan ikke hurtigt reagere på konkurrence fra nyere firmaer med inkrementalt bedre (læs: yngre, nyere) uddannede folk; de bedste folk "skrider" ("siver") til de bedre, mere lovende firmaer, og en altfor ømtålelig del af firmaets ekspertise forsvinder sammen med disse folk. Det hele ender med lavest fællesnævner — istedet for største fællesfold.

De dog ikke helt så få idag vel–uddannede dataloger, som kan de mere formelle, de mere stringente metoder, som forventer at programmel–udvikling f.eks. flgr. det, der er skitseret i nærværende notat, disse dataloger "drukner" i de større IT firmaers stab: Deres viden bliver på langt nær udnyttet. Et stort potentiale går tabt.

9.4 Akademisk Uddannelse

Imens er de seriøse datalogiske uddannelses–steder nødt til at fastholde kvaliteten, til ikke at bøje sig for *"popular demand"*, men at fortsætte med at uddanne den næste generation af

kandidater til højeste, faglige niveau. Alt andet ville være uansvarligt. Det er ikke svært at lære de nye metoder. Store grupper kandiderer hvert år med en fast forankring heri.

9.5 Største Fællesfold

Så hvordan kan vi opnå stabilitet, sikre de lange linjer — trods eventuel "udsivning" af hvad man før betragtede som nøglepersonale ? Hvordan kan vi sikre en professionel, troværdig programmel–industri ?

9.5.1 Forankring i et Domæne (Genstandsområde)

Foruden at sikre at alle ledende, dvs. at 80% af de professionelle ansatte er akademisk–trænede på og til dagens højeste niveau, bør man også sikre at firmaet, institutionen, programmel–huset:

- Udnytter de ansattes evner og kunnen, og tilbyder dem et intellektuelt udfordrende miljø — se Afsnittene 9.5.2–9.5.3;
- og at det (firmaet, institutionen, programmel–huset) har en fast forankring i et erkendt genstands–område: Ét af, eller et par nærbeslægtede domæner for hvilket firmaet har en særlig affinitet. Denne affinitet kan f.eks., vil typisk, bestå deri at firmaet har en dyb indsigt i domænet: At det har videreudviklet særlig indsigt i, og, mere konsekvent, har udviklet *"proprietary tools"*, egne værktøjer, teknologier, til hjælp i forståelsen, og til den videre udvikling, af systemer (nærmere bestemt: programmel) indenfor dette genstandsområde.

Det vi oftest ser idag er firmaer, der forvirret "jagter" de seneste "trends", uden dybere at have reflekteret over deres egen, særlige kvalifikationer.

9.5.2 Projekt Kvalitet

Medvirkende til at sikre medarbejdernes loyalitet er at ledelsen sikrer at udviklings–projekterne har kvalitet. Til projekt–kvalitet medregner vi flg.:

- at projektet's udførelse er fast forankret i en, på forhånd, eksplicit erkendt metode, og at det derigennem bliver understøttet af relevante værktøj;
- at projektet's ressource–forbrug og risici[34], på forhånd, kan estimeres på troværdig vis, og at det løbende overvåges og styres;
- at projektet's ledelse har de øvrige ansatte's respekt, og omvendt;
- at der gennem projektet's udførelses–forløb, til stadighed er snævre, vel–identificerede og fokuserede menneskelige og faglige relationer til projekt–produktets genstands–område (kunder, klienter, interessenter — "sågar" konkurrenter);

[34] Trods rimeligt brugbare, enkle procedurer for planlægning, estimering, overvågning og opfølgning mht. risici er det foruroligende at observere hvor lidt disse procedurer er kendte, endsige brugte.

- samt at alle projekt–medarbejdere "har det sjovt": At man, gennem projektet, til stadighed udfordres, videre–dygtiggører sig, samt at ens evner og kunnen tilfulde udnyttes, mm.

Programmel–udvikling kunne, med stor fordel, "kopiere" andre ingeniør–discipliners "studie/-eksperiment/anvendelses" paradigme i hvert afsnit af udvikling:

- *Studie:* Et literatur–studie undersøger hvorledes "best practices", teknisk og videnskabeligt, "bedrives" andetsteds — dvs. hvorledes andre tilsynelandende har løst problemer nært beslægtede med, eller lig, det man ønsker selv at løse.
- *Eksperiment:* Et efterflg. eksperiment anvender én eller to af de studerede løsnings–tilgangsvinkler.
- *Anvendelse:* Eksperimentet resulterer i valg af én bestemt tilgangsvinkel, én løsnings–model. Og den endelige udvikling, for dette afsnit, sættes igang.

Ressource–, f.eks. person–måned–mæssigt, udsiger erfaringen, fordeler de tre paradigme–komponenter sig som én–til–tre–til–ni: 1:3:9 !

9.5.3 Produkt Kvalitet

Medvirkende til at sikre medarbejdernes loyalitet er at ledelsen sikrer at de udviklede produkter har kvalitet. Til produkt–kvalitet medregner vi flg.:

- at produktet er seriøst: Troværdig højest mulige affinitet til genstands–området: at det er det rigtige produkt for dette genstands–område, at det er korrekt mht. påståede egenskaber (dvs. krav: At produktet er rigtigt);
- at produktet er bakket op af en seriøs, troværdig salgs–, service– og markeds–funktion — der sikrer kontinuitet (*"CRM: Customer Relation(ship) Management"*);
- at produktet har den rette, en seriøs, troværdig pris, og at det, af bruger, klient og interessenter, ses som det bedste produkt på markedet;
- samt at produktet følges op af nye (del–) produkter, der på seriøs, troværdig vis "vokser" i takt med klientens egen virksomhed og klientens ansattes og kunders kunnen og behov.

9.5.4 Diskussion

Det siger sig selv at vores insistering på en dyb forankring i domænet — intim forståelse, klart erkendt "ekspert"–kunnen — er med til at sikre ovenstående seriøsitet og troværdighed, til en begyndelse, samt at leverandør–firmaets løbende videre–udforskning af genstands–området er med til at vedligeholde projekt– og produkt–kvaliteterne.

10 Afslutning

Vi har fremstillet arbejdsgange og mellem– og slut–resultater af en meget omfattende, detaljeret og "nøjeregnende" programmel–udviklings–process. Vi har antydet at denne, i den form vi selv bruger den, dvs. også med det indhold hvori vi underviser deri, udover de mange uformelle, dog stringente og præcise udviklings–faser, –afsnit og –trin, også indeholder formelle, matematisk baserede sådanne komponenter. Vi har faktisk antydet at det netop er denne side af den fremstillede, nyere, klasse af programmel–udviklings–metoder, der betinger deres nyttigste brug. Vi har mao. antydet at det er muligt — og literaturen godtgør, at der er belæg for denne påstand — at udvikle programmel til en langt højere standard end det idag almindeligvis kendes: At vi kan komme langt tættere på at indforskrive programmel vi kan have tillid til, som i langt højere grad end det idag erfares, opfylder kundens og brugernes behov og forventninger; at de ikke er "fyldt med fejl"; at de "stille og roligt" kan udvides til også at rumme nye, domæne–beslægtede funktioner; at de udvikles til tid og kost; oma.

Vi har også kommenteret denne metodes relation til begrebet forandrings–ledelse, herunder *"business process (re–)engineering"*. Og vi har endelig kommenteret visse sociologiske og psykologiske aspekter ved sådanne, nyere metoders mere udbredte akcept.

11 Selv–Bibliografiske Noter

Egne[35] publikationer og rapporter, der dækker diverse infra–strukturers' domæne–modellering.

- Jernbaner: [5, 49, 44, 8, 45, 46, 50, 51, 25, 39, 39]
- Finansvæsen: [52, 31]
- Lufttrafik: [14]
- Logistik: [35]
- Sundheds–sektor: [33]
- E–Handel: [48, 41, 30]
- Projekt og Produktions–planlægning, Kontrol & Styring: [16, 21, 22, 38]
- Bærbar Udvikling: [20]

Publikationer der dækker de metode–mæssige forhold ved udvikling af store programmel–systemer:

- Generelt: [1, 2, 23, 36, 37]
- Domæne–beskrivelse: [29, 42, 34]
- Krav–definition: [19, 40, 32]
- Programmel–specifikation: [27]

Referencer

[1] Dines Bjørner. The VDM Principles of Software Specification and Program Design. In *TC2 Work.Conf. on Formalisation of Programming Concepts, Peniscola, Spain*, pages 44–74, LNCS Vol. 107, 1981. IFIP, Springer-Verlag.

[2] Dines Bjørner. Stepwise Transformation of Software Architectures. In *[47]*, chapter 11, pages 353–378. Prentice-Hall, 1982.

[3] Dines Bjørner. A Rôle for UNU/IIST: Developing Countries' Access to New Information Technologies. In *Access to Science and Technology — The Rôle of Information Technology*. Kyoto University & UNU Press, May 12–14 1992.

[35] Jeg tillader mig alene at henvise til egen produktion. Der er, selvflg., en langt videre literatur — som de nedenfor refererede publikationer og rapporter henviser til. Listen af mine egne publikationer mm. skal således tjene til at de læsere af dette notat, som ikke er bekendt hermed, kan erfare at der bag dette notat ligger snart 30 års seriøst arbejde. I Appendiks Afsnit A bringer jeg mit CV. Heraf fremgår yderligere "tillids–skabende" forhold.

[4] Dines Bjørner. 1992 Annual Report. Administrative Report 2, UNU/IIST, P.O.Box 3058, Macau, January 1993. Reports on the status of UNU/IIST and its activities after the first 6 months of operation.

[5] Dines Bjørner. Prospects for a Viable Software Industry — Enterprise Models, Design Calculi, and Reusable Modules. Technical Report 12, UNU/IIST, P.O.Box 3058, Macau, 7 November 1993. Appendix — on a railway domain model — by Søren Prehn and Dong Yulin, Published in *Proceedings from first ACM Japan Chapter Conference*, March 7–9, 1994: World Scientific Publ., Singapore, 1994.

[6] Dines Bjørner. UNU/IIST Programme. Administrative Report, 1, UNU/IIST, P.O.Box 3058, Macau, February 19 1993. Presents the co-ordinates along, and principles according to which UNU/IIST will select and pursue its Programmatic Activities the next couple of years.

[7] Dines Bjørner. Accreditation, Licensing and Certification; Curricula, Engineers and Software. Technical Report 14, UNU/IIST, P.O.Box 3058, Macau, 7 November 1993 1993, Revised 8 December 1993. Presented at the *UNIDO/COGIT Meeting*, Vienna, Austria, November 23, 1993.

[8] Dines Bjørner. An Architecture for Running Map Systems. Technical Report db/arch/01, UNU/IIST, the UN University's International Institute for Software Technology, P.O.Box 3058, Macau; E-Mail: `library@iist.unu.edu`, February 1994.

[9] Dines Bjørner. Annual Report: 1993. Administrative Report 18, UNU/IIST, P.O.Box 3058, Macau, 1. January 1994.

[10] Dines Bjørner. Annual Report, 1994. Research Report 22, UNU/IIST, P.O.Box 3058, Macau, 1. January 1995.

[11] Dines Bjørner. Infrastructure Software Systems. Technical Report 58, UNU/IIST, P.O.Box 3058, Macau, Dec 1995. Presentation solicited for the Academia Europae (AE/CWI/SMC) Symposium, Amsterdam 11–12 April, 1996. .

[12] Dines Bjørner. New Software Technology Development. Technical Report 46, UNU/IIST, P.O.Box 3058, Macau, November 1995. International Symposium: New IT for Governance and Publication Administration, Beijing, China; organized by UNDDSMS, June 1996.

[13] Dines Bjørner. Software Support for Infrastructure Systems. Technical Report 47, UNU/IIST, P.O.Box 3058, Macau, November 1995. Position statement for the *First Malaysia Information Technology Days: 1–3 November 1995* .

[14] Dines Bjørner. Software Systems Engineering — From Domain Analysis to Requirements Capture [— an Air Traffic Control Example]. Technical Report 48, UNU/IIST, P.O.Box 3058, Macau, November 1995. Keynote paper for the *Asia Pacific Software Engineering Conference*, APSEC'95, Brisbane, Australia, 6–9 December 1995. .

[15] Dines Bjørner. 1995 Annual Report. Administrative Report 53, UNU/IIST, P.O.Box 3058, Macau, January 1996. Reports on the status of UNU/IIST and its activities during 1995.

[16] Dines Bjørner. Models of Enterprise Management: Strategy, Tactics & Operations — Case Study Applied to Airlines and Manufacturing. Technical Report 60, UNU/IIST, P.O.Box 3058, Macau, January – April 1996. .

[17] Dines Bjørner. New Software Development. Administrative/Technical Report 59, UNU/IIST, P.O.Box 3058, Macau, January 1996. Special *Theme* paper: *New Software Technology Development*. Paper was first prepared in September 1995 for an International Symposium: *New IT Applications for Governance and Public Administration*, convened by the UN's Department for Development Support and Management Service: UNDDSMS, Beijing, November 9–14, 1995. This symposium was subsequently moved (tentatively) to June 1996, same venue. .

[18] Dines Bjørner. UN's Software Technology R&D in Africa, Asia, Eastern Europe, and Latin America — A Personal View. Research Report 90, UNU/IIST, P.O.Box 3058, Macau, May 1997.

[19] Dines Bjørner. Domains as Prerequisites for Requirements and Software *&c.* In M. Broy and B. Rumpe, editors, *RTSE'97: Requirements Targeted Software and Systems Engineering*, volume 1526 of *Lecture Notes in Computer Science*, pages 1–41. Springer-Verlag, Berlin Heidelberg, 1998.

[20] Dines Bjørner. A Triptych Software Development Paradigm: Domain, Requirements and Software. Towards a Model Development of A Decision Support System for Sustainable Development. In ErnstRüdiger Olderog, editor, *Festschrift to Hans Langmaack*. University of Kiel, Germany, October 1999.

[21] Dines Bjørner. Domain Modelling: Resource Management Strategics, Tactics & Operations, Decision Support and Algorithmic Software. In J.C.P. Woodcock, editor, *Festschrift to Tony Hoare*. Oxford University and Microsoft, September 13–14 1999.

[22] Dines Bjørner. Project Information, Monitoring and Control Systems — A Domain Analysis. Technical report, Dept. of Informatics and Mathematical Modelling, Technical University of Denmark, Bldg. 322, DK–2800 Lyngby, Denmark, 1999.

[23] Dines Bjørner. Where do Software Architectures come from ? Systematic Development from Domains and Requirements. A Re–assessment of Software Engneering ? *South African Journal of Computer Science*, 1999. Editor: Chris Brink.

[24] Dines Bjørner. Domain Engineering, Elements of a Software Engineering Methodology — Towards Principles, Techniques and Tools — A Study in Methodology. Research report, Dept. of Computer Science & Technology, Technical University of Denmark, Bldg. 343, DK–2800 Lyngby, Denmark, 2000. One in a series of summarising research reports [26, 27].

[25] Dines Bjørner. Formal Software Techniques in Railway Systems. In Eckehard Schnieder, editor, *9th IFAC Symposium on Control in Transportation Systems*, pages 1–12, Technical University, Braunschweig, Germany, 13–15 June 2000. VDI/VDE-Gesellschaft Mess– und Automatisieringstechnik, VDI-Gesellschaft für Fahrzeug– und Verkehrstechnik. Invited plenum lecture.

[26] Dines Bjørner. Requirements Engineering, Elements of a Software Engineering Methodology — Towards Principles, Techniques and Tools — A Study in Methodology. Research report, Dept. of Computer Science & Technology, Technical University of Denmark, Bldg. 343, DK–2800 Lyngby, Denmark, 2000. Not yet released. Meanwhile refer to [28]. One in a series of summarising research reports [24, 27].

[27] Dines Bjørner. Software Design: Architectures and Program Organisation, Elements of a Software Engineering Methodology — Towards Principles, Techniques and Tools — A Study in Methodology. Research report, Dept. of Computer Science & Technology, Technical University of Denmark, Bldg. 343, DK–2800 Lyngby, Denmark, 2000. Not yet released. Meanwhile refer to [28]. One in a series of summarising research reports [24, 26].

[28] Dines Bjørner. *Software Engineeering: A New Approach. From domains via requirements to software. Formal specification and design calculi.* Dept. of Informatics and Mathematical Modelling, Technical University of Denmark, Richard Petersens Plads, Building 322, DK–2800 Lyngby, Denmark, 2000. Presently this document is a rather extensive (approx. 900 page) set of lecture notes. Postscript versions of the document are accessible over the web: `http://www.it.dtu.dk/~db/s2000/notes.ps`.

[29] Dines Bjørner. Domain Engineering — A Prerequisite for Requirements Engineering — Principles and Techniques. Technical report, Informatics and Mathematical Modelling, Building 322, Richard Petersens Plads, Technical University of Denmark, DK–2800 Kgs.Lyngby, Denmark, 2001. This paper is one of a series of papers currently being submitted for publication: [36, 37, 40, 33, 30, 35, 38, 39, 31].

[30] Dines Bjørner. E–Business. Towards a Domain Theory for Work Flow Systems. Technical report, Informatics and Mathematical Modelling, Building 322, Richard Petersens Plads, Technical University of Denmark, DK–2800 Kgs.Lyngby, Denmark, 2001. This paper is one of a series of papers currently being submitted for publication: [36, 37, 29, 40, 33, 35, 38, 39, 31].

[31] Dines Bjørner. Financial Service Institutions: Banks, Securities Trading, Insurance, *&c.* Towards a Domain Theory for Work Flow Systems. Technical report, Informatics and Mathematical Modelling, Building 322, Richard Petersens Plads, Technical University of Denmark, DK–2800 Kgs.Lyngby, Denmark, 2001. This paper is one of a series of papers currently being submitted for publication: [36, 37, 29, 40, 33, 30, 35, 39, 38].

[32] Dines Bjørner. From Domains to Requirements — Some Protocol Challenges. In *FORTE: Formal Protocol Description and Verification Techniques.* IFIP WG6.1, Kluwer Press, 2001.

[33] Dines Bjørner. Healthcare Systems. Towards a Domain Theory for Work Flow Systems. Technical report, Informatics and Mathematical Modelling, Building 322, Richard Petersens Plads, Technical University of Denmark, DK–2800 Kgs.Lyngby, Denmark, 2001. This paper is one of a series of papers currently being submitted for publication: [36, 37, 29, 40, 30, 35, 38, 39, 31].

[34] Dines Bjørner. Informatics Models of Infrastructure Domains. In *Computer Science and Information Technologies*, pages 13–73. National Academy of Science of Armenia, Institute for Informatics and Automation Problems, September 2001.

[35] Dines Bjørner. Logistics. Towards a Domain Theory for Work Flow Systems. Technical report, Informatics and Mathematical Modelling, Building 322, Richard Petersens Plads, Technical University of Denmark, DK–2800 Kgs.Lyngby, Denmark, 2001. This paper is one of a series of papers currently being submitted for publication: [36, 37, 29, 40, 33, 30, 38, 39, 31].

[36] Dines Bjørner. Models, Semiotics, Documents and Descriptions — Towards Software Engineering Literacy. Technical report, Informatics and Mathematical Modelling, Building 322, Richard Petersens Plads, Technical University of Denmark, DK–2800 Kgs.Lyngby, Denmark, 2001. This paper is one of a series of papers currently being submitted for publication: [37, 29, 40, 33, 30, 35, 38, 39, 31].

[37] Dines Bjørner. Principles and Techniques of Abstract Modelling — Some Basic Classifications. — Towards a Methodology of Software Engineering. Technical report, Informatics and Mathematical Modelling, Building 322, Richard Petersens Plads, Technical University of Denmark, DK–2800 Kgs.Lyngby, Denmark, 2001. This paper is one of a series of papers currently being submitted for publication: [36, 29, 40, 33, 30, 35, 38, 39, 31].

[38] Dines Bjørner. Projects & Production: Planning, Plans & Execution. Towards a Domain Theory for Work Flow Systems. Technical report, Informatics and Mathematical Modelling, Building 322, Richard Petersens Plads, Technical University of Denmark, DK–2800 Kgs.Lyngby, Denmark, 2001. This paper is one of a series of papers currently being submitted for publication: [36, 37, 29, 40, 33, 30, 35, 39, 31].

[39] Dines Bjørner. Railways Systems: Towards a Domain Theory. Technical report, Informatics and Mathematical Modelling, Building 322, Richard Petersens Plads, Technical University of Denmark, DK–2800 Kgs.Lyngby, Denmark, 2001. This paper is one of a series of papers currently being submitted for publication: [36, 37, 29, 40, 33, 30, 35, 38, 31].

[40] Dines Bjørner. Requirements Engineering — Some Principles and Techniques — Bridging Domain Engineering and Software Design. Technical report, Informatics and Mathematical Modelling, Building 322, Richard Petersens Plads, Technical University of Denmark, DK–2800 Kgs.Lyngby, Denmark, 2001. This paper is one of a series of papers currently being submitted for publication: [36, 37, 29, 33, 30, 35, 38, 39, 31].

[41] Dines Bjørner. Towards the E–Market: To understand the E–Market we must first understand "The Market". In *Government E–Commerce Development*. Ningbo Science & Technology Commission, Ningbo, Zhejian Province, China, 23++24 April 2001.

[42] Dines Bjørner. *"What is a Method ?" — A Study of Some Aspects of Software Engineering*. IFIP WG2.3. MacMillan, Oxford, UK, 2001. Programming Methodology: Recent Work by Members of IFIP Working Group 2.3. Eds.: Annabelle McIver and Carrol Morgan. To be published.

[43] Dines Bjørner. Towards a Theory of Infrastructures. In *10 Years of Software Technology Research and Transfer*. UNU/IIST, 18–21 March 2002.

[44] Dines Bjørner, C.W. George, B.Stig Hansen, H. Laustrup, and S. Prehn. A Railway System, Coordination'97, Case Study Workshop Example. Research Report 93, UNU/IIST, P.O.Box 3058, Macau, January 1997. .

[45] Dines Bjørner, C.W. George, and S. Prehn. Domain Analysis — a Prerequisite for Requirements Capture. Technical Report 37, UNU/IIST, P.O.Box 3058, Macau, February 1995. .

[46] Dines Bjørner, C.W. George, and S. Prehn. *Scheduling and Rescheduling of Trains*, chapter ???, pages ???–??? *Industrial Strength Formal Methods,* Eds.: M. Hinchey and J.P. Bowen. FACIT, Springer–Verlag, London, England, 1999.

[47] Dines Bjørner and C.B. Jones, editors. *Formal Specification and Software Development.* Prentice-Hall, 1982.

[48] Dines Bjørner, Zhu Li, and Nikolaj Duva. Towards a Flow, Action and Information Semantics of E–Commerce. Technical, Informatics and Mathematical Modelling, Technical University of Denmark, Building 322, Richard Petersens Plads, DK-2800 Lyngby, Denmark, December 2000.

[49] Dines Bjørner, Dong Yu Lin, and S. Prehn. Domain Analyses: A Case Study of Station Management. Research Report 23, UNU/IIST, P.O.Box 3058, Macau, 9 November 1994. Presented at the *1994 Kunming International CASE Symposium*: KICS'94, Yunnan Province, P.R.of China, 16–20 November 1994. .

[50] Dines Bjørner, Søren Prehn, et al. Formal Models of Railway Systems: Domains. Technical report, Dept. of IT, Technical University of Denmark, Bldg. 344, DK–2800 Lyngby, Denmark, September 23 1999. Presented at the FME Rail Workshop on Formal Methods in Railway Systems, FM'99 World Congress on Formal Methods, Toulouse, France. Avaliable on CD ROM.

[51] Dines Bjørner, Søren Prehn, et al. Formal Models of Railway Systems: Requirements. Technical report, Dept. of IT, Technical University of Denmark, Bldg. 344, DK–2800 Lyngby, Denmark, September 23 1999. Presented at the FME Rail Workshop on Formal Methods in Railway Systems, FM'99 World Congress on Formal Methods, Toulouse, France. Avaliable on CD ROM.

[52] Dines Bjørner, Vasco Rosario, and M. Helder. A Normative Model of Concrete Banking Operations — Banking Rules & Regulations and Staff/Client Behaviours. Research, Department of Information Technology, Software Systems Section, Technical University of Denmark, DK–2800 Lyngby, Denmark, June 1998. (Need be revised: Some typos etc. !).

[53] Dines Bjørner and M. Stuart. UNU/IIST Annual Report 1996. Administrative Report 85, UNU/IIST, P.O.Box 3058, Macau, October 1996.

A Dines Bjørner (DB) CV

- Born in Odense, Denmark, 4 October 1937, DB, grew up in Århus where he attended the Århus Cathedral School (founded in 1142). DB graduated in January 1962 with an M.Sc. in Electronics Engineering and with a Ph.D. in Computer Science in January 1969 from the Technical University of Denmark (founded by Hans Christian Ørsted in 1828).
- DB joined IBM in March 1962 at their Nordic Laboratories (founded by Cai Kinberg) in Stockholm, Sweden (where DB also first met Jean Paul Jacob and Gunnar Wedell). DB was transferred to the IBM Systems Development Division (IBM SDD) at San Jose, California, USA, in December 1963. While doing his Ph.D. (September 1965–January 1969) DB was a lecturing consultant to IBM's European Systems Research Institute (ESRI) at Geneva, Switzerland (where DB acquainted with Carlo Santacroce and where DB's friendship with Gerald Weinberg started) (1967–1968). In 1969 DB worked at IBM's Advanced Computing Systems (IBM ACS) laboratory, Menlo Park, California, and, later that year until early 1973 at IBM Research, San Jose (again Jean Paul Jacob became a colleague). Transferred to the IBM Vienna Laboratory (directed then by Heinz Zemanek), Austria, DB resigned from IBM in August 1975 to return to Denmark after basically 13 years abroad.
- During his stay at IBM Research DB was a visiting lecturer, for several quarters, at University of California at Berkeley (1971–1972), instigated by Lotfi Zadeh whom DB considers his main mentor and for whom DB has the fondest regards. DB was a visiting guest professor at Copenhagen University in the academic year 1975–1976, before taking up his present chair in September 1976 at the Technical University of Denmark (DTU). During the summer semester of 1980 DB was the Danish Chair Professor at the Christian-Albrechts University of Kiel, Germany. DB was the founding and first UN Director of UNU/IIST, the United Nations University's International Institute for Software Technology, located in Macau. During the 1980s DB was chief scientist at Dansk Datamatik Center (DDC). (DB was a co-founder of DDC in 1979 and interim managing director in 1979.) In 1982–1984 DB was chairman of a Danish Government (Ministry of Education) Commission on Informatics.
- DB has lectured and regularly lectures on six continents in almost 50 countries and territories and has graduated more than 80 MSc's and a dozen PhDs.
- At IBM DB first worked in the hardware (logic and systems) design of such equipment as the IBM 1070 (Sweden), the IBM 1800 and IBM 1130 computers (San Jose), and, finally, with Gene Amdahl and Ed Sussenguth, the IBM ACS/1 supercomputer (Menlo Park). At Research DB worked with John W. Backus and Ted Codd on Functional Languages, resp. Relational Data Base Systems. At Vienna, DB, together with such colleagues as Peter Lucas, the late Hans Bekič, Kurt Walk, and Cliff B. Jones, worked on a Denotational (–like) Semantics Description of PL/I while, with his colleagues conceiving, researching, developing and using VDM (the Vienna software Development Method). At DTU and at DDC, supported by the European Community, DB initiated several advanced research & development projects: (1) Formal Semantics Description of

and (2) full language compiler for CHILL (the Intl. Telecommunications Unions Communications [C.C.I.T.T.] High Level Language), (4) Formal Semantics Description of and (3) the first European US DoD officially validated compiler for the US DoD Ada embedded systems programming language, (5) RAISE (Rigorous Approach to Industrial Software Engineering), (6) Formal Semantics Definition of VDM–SL (the VDM Specification Language), (7) ProCoS (Provably Correct Systems) with, amongst others Profs. Sir Tony Hoare (then Oxford, now Microsoft Research, Cambridge, UK), Hans Langmaack (Kiel) and Ernst-Rüdiger Olderog (Oldenburg), etc.

- At UNU/IIST DB had a rather free hand, and was able, with a small team of excellent colleagues (Prof. Zhou Chaochen (Academician, the Chinese Academy of Science), Søren Prehn (CRI, now Terma Electronics), Chris W. George, Richard Moore, Tomasz Janowski, Dang Van Hung, Xu Qi Wen and Kees Middelburg), to further explore the research issues currently (still) occupying DB's interest, and to apply them (ie. test them out) in a number of joint R&D projects with institutions in developing and newly industrialised countries [including newly independent states] (Argentina, Belarus, Brasil, Cameroun, China, Gabon, India, Indonesia, Mongolia, North Korea, Pakistan, Philippines, Poland, Romania, Russia, South Africa, South Korea, Thailand, Vietnam, Ukraine, Uruguay, etc.).

- DB was a co–founder of VDM-Europe in 1987 and moved VDM-Europe onto FME: Formal Methods Europe in 1991. DB co-chaired two of the VDM Symposia (1987, 1990), and the International Conference on Software Engineering (ICSE) in 1989 in Pittsburgh, Pennsylvania, USA. DB was chairman of the IFIP World Congress in Dublin, Ireland in 1986, and was the instigator and General Chairman of the first World Congress on Formal Methods, FM'99, in Toulouse, France, September 20–24, 1999.

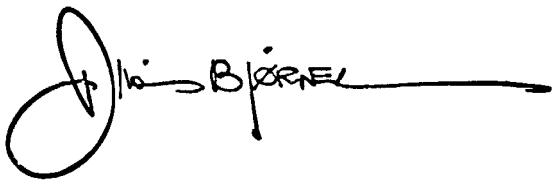
- DB is a Knight of The Danish Flag; is a Member of Academia Europaea (MAE), of The Russian Academy of Natural Sciences (MRANS [AB]), and of IFIP Working Groups 2.2 and 2.3. DB has received the John von Neumann Medal of the JvN Society of Hungary and the Mazaryk Medal from the Mazaryk University, Brno, The Czech Republic. DB received the Danish Engineering Society's (IDA) Informatics Division's (IDA–IT) first BIT prize, March 1999.

- DB has authored around 80 published papers and co-authored and co-edited some 15 books.

- DB's research interests, since his Vienna days, have centered on programming methodology: Methods as sets of principles for selecting and applying mathematics–based analysis and construction techniques and tools in order efficiently to construct efficient artifacts — software. DB sees his main contributions to be in the research, development and propagation of formal specification principles and techniques. Currently DB focuses on the triptych of domain engineering, requirements engineering and the software architecture and program organisation methods — emphasising such that relate these in mathematical as well as technical ways: (1) Intrinsic, support technology, management & organisation, rules & regulation, and human behaviour facets of domains; (2) projection, instantiation, extension and initialisation of domain requirements, etc.; (3)

software architectures as refinements of domain requirements, and program organisation as refinements of machine requirements — with interface requirements (currently) being refinements of either and both! A series of reports are currently (Fall 2001 and Winter 2001/2002) being written as is a rather comprehensive set of lecture notes. These summarise DB's research since 1973. DB hopes that the lecture notes will someday appear in book form.

- Among the very many people who have been kind to DB and helped DB in his professional career, he wishes to bear tribute, in approximate chronological order, to Cai Kinberg, Gunnar Wedell, Jean Paul Jacob, Gerald Weinberg, Carlo Santacroce, Gene Amdahl, Ed Sussenguth, Tien Chi (T.C.) Chen, Lotfi Zadeh, Ted Codd, John W. Backus, Peter Lucas, Cliff Jones, (the late) Hans Bekič, Kurt Walk, Ole N. Oest, Erich Neuhold, Søren Prehn, Sir Tony Hoare, Hans Langmaack, Zhou Chao Chen, and Chris George.



B Eksempler på Beskrivelser af Infrastruktur–komponenter

Det kan synes særegent at bringe eksempler på tilsyneladende "ganske almindelige beskrivelser" af "ganske almindelige anvendelses–domæner". Men måske disse beskrivelser ikke er så almindelige endda: Dels mangler de som regel, for ikke at sige altid, i udvikling af programmel og dermed fra dets dokumentation, dels er der faktisk ikke særlige mange, for ikke at sige ingen sådanne, tilgænglige beskrivelser. Mao.: Man udvikler, rask væk, store programmel–systemer uden at have godgjort for andre, til eksempel klienten, endsige sig selv, at man véd noget om domænet. *Man ville vel ikke udvikle mekanisk udstyr om man ikke kunne godtgøre professionalisme, f.eks. gennem en akkrediteret uddannelse i de mekaniske fag. Tilsvarende for elektro–teknik, elektronik, kemi, bygnings–væsen: At man kender fagets basale lovmæssigheder oma.* Etablering og dokumentation af domæne–beskrivelser skal derfor sikre denne tillid.

De anvendelses–domæner vi vil illustrere er flg.:

- *Jernbane–domænet og krav dertil* 81–85
- *Logistik–domænet og krav dertil* 87–90
- *Handel og E–handels—domænet og krav dertil* 91–95
- *Finans–sektor–domænet og krav dertil* 97–100
- *Sundheds–sektor–domænet og krav dertil* 105–106
- *"Flow"–systemer: Planlægnings og udførelses–domænet og krav dertil* 101–104

Nu kan vi jo ikke sådan lige beskrive "alt" i hvert af disse domæner. Istedet fokuserer vi på blot at antyde hvad det er der skal beskrives. Dvs. vi bringer blot en kort synopsis og råskitser (jvf. omtale af disse begreber Side 18). Andetsteds bringer vi større sådanne fortællende beskrivelser. Dog, Eksempel 4 på side 27 antyder en mere tilfredsstillende, detaljeret (udtømmende) beskrivelse.

• • •

Der er andre eksempler for hvilke vi ikke bringer særskilte afsnit:

- **Luft–trafik:** Vi har modelleret, i [14] og i et deraf afledt eksamens–projekt, domæne–begrebet *"air traffic"* og det deraf afledte krav–definitions–begreb *"air traffic control"*.
- **Luft–havne:** I ikke–publicerede noter, og i et deraf afledt kursus–projekt, har vi modelleret domæne–begrebet 'en lufthavn': En "størrelse" med alle dets strømme af mennesker, (øvrige) ressourcer, information og styrings–signaler: Passagers "vej" gennem en lufthavn, udefra til et fly, i transit fra fly til fly, og fra fly "hjem"; baggagens vej fra check–in til fly via diverse transport–bånd og vogne, i transit fra fly til fly, og fra fly via transport–bånd til baggage–udlevering (inkl. evt. told); fly–brændstofs–distribution: Fra indgående tank–biler til lokalt lager, derfra til fly; planlægning, produktion og distribution af fly–fortæring (*"catering"*); flys' ankomst til en lufthavn: Deres rute over *"tarmac'en"* til *gate*, *gate*–fordeling, dets grænseflader til passagerer, baggage, brændstof, *"catering"*, rengøring, mekanisk service; oma.

- **Et Universitet:** Vi har modelleret hvad det vil sige at være et universitet: Set fra de studerendes side, fra underviser/forsker–stabens side, fra administrationens side; etc. Vi har modelleret studentens kursus–planlægning, kursus–deltagelse, aflevering af opgaver, eksamen, mm. Vi har modelleret underviserens udvikling af et nyt kursus, forberedelse af et planlagt kursus, afvikling af dette, forelæsning–for–forelæsning, opgave–evaluering, eksamens–forberedelse (mundtlig, skriftlig) og afholdelse. Vi har modelleret de administrative forhold omkring forskning: Planlægning, herunder budgetering, afvikling og rapportering. Og vi har modelleret aspekter af hvorledes de tre størrelser griber ind i hverandre: Studerende, underviser/forskere og administration. Vi har ikke, endnu, modelleret domæne–forhold omkring et universitet's interaktion med f.eks. resort–ministerium, lokalt erhverv og lokal industri, eller andre universiteter — selvom vi selvflg. har erfaring hermed.
- **Turisme mm.:** Vi har, i forbindelse med vort arbejde ved FN, modelleret hvad det vil sige at være "turisme" i en region som f.eks. Macau (eller Storkøbenhavn): Strømme af turister, deres forbrug af rejser, hoteller, restauranter, underholdning, sport oa. rekreation, muséer, *"sightseeing, shopping"*, oma., herunder rejse–forberedelse, start, "eksekvering", og afslutning.

Og så fremdeles: *"The sky's the limit"*.

Fælles for alle eksempler, ovenfor, og i de flg. afsnit, er at vi starter med at stille spørgsmålet:

Hvad er x ?

hvor x er een blandt: en jernbane (DSB, f.eks.), logistik, handel, en finans-verden, en sundheds–sektor, et *"flow system"*, fly trafik, en lufthavn, et universitet, turisme.

Andre eksempler kunne nævnes.

B.1 Jernbane–domæne og Krav

Vort jernbane–domæne består af flg. entiteter: Et spor–net, tog, disses trafik, køreplaner, passager, og gods. Disse, deres aktiviteter og interaktion, beskrives nedenfor.

B.1.1 Synopsis

Essensen i vort jernbane–domæne er at passagerer (med billetter) og gods (iflg. fragtbreve) transporteres af tog, ad linjer, mellem stationer, efter mere eller mindre faste køreplaner, og til pris.

En flg. essens af vort jernbane–domæne er at tog afsendes til tid, at tog–trafik overvåges med henblik på sikkerhed, passager–komfort, fragt–sikkerhed, og punktlighed, og at det hele sker rimeligt optimalt, indenfor budget og uden altfor mange drifts–uregelmæssigheder.

Andre flg. essensser af vort jernbane–domæne er at dets ansatte trives: Regelmæssigt modtager deres rettelige løn, udnyttes efter evne, kunnen, uden "hast–&–jag"–induceret "stress", og under hensyn til hvile–, fridags– og ferie–bestemmelser.

B.1.2 Spor–net

Et jernbane–spor–net består af stationer og linjer: Mindst to stationer og mindst een linje. Stationer og linjer består af spor–enheder. En linje forbinder eksakt to distinkte stationer. Enhver linje har et unikt navn. Enhver spor–enhed besidder sammenføjnings–punkter (konnektorer). Spor–enheder er enten lineære (og har da to distinkte konnektorer), eller er sporskifter (og har da tre distinkte konnektorer), eller er almindelige, simple spor–kryds (og har da fire distinkte konnektorer), eller er skiftelige spor–kryds (og har da fire distinkte konnektorer), eller er .... En linje består af en sammenføjet sekvens af een eller flere lineære spor–enheder (således at to på hverandre følgende spor–enheder i en linje deler eet sammenføjnings–punkt. For ethvert sammenføjnings–punkt er der højst to spor–enheder der deler dette sammenføjnings–punkt.

En spor–enhed tillader bevægelse af et tog i een eller flere retninger. Lad k, k' angive en lineær spor–enhed's to konnektorer. Da er det principielt muligt for et tog (en togvogn) at bevæge sig i enten retning (k, k') eller i retning (k', k). For et sporskifte, med tre konnektorer: k, k', k'' (hvor vi antager at k angiver det sammenføjnings–punkt fra hvilke man kan bevæge enten i retning mod k' eller mod k''), da er flg. bevægelses–retninger principielt mulige: $(k, k'), (k, k''), (k', k)$ og (k'', k). Ved en spor–enheds tilstand forstås mængden af alle de bevægelses–retninger der er aktuelt mulige. For en lineær spor–enhed kan en sådan tilstand enten være $\{\}$ (spor-enheden er lukket for trafik, eller $\{(k, k')\}$, eller $\{(k', k)\}$, eller $\{(k, k'), (k', k)\}$ (spor–enheden er åben for trafik i begge retninger). For et spor–skifte kan flg. tilstande være mulige: $\{\}$, eller $\{(k, k')\}$, eller $\{(k', k)\}$, eller $\{(k, k'')\}$, eller $\{(k'', k)\}$, eller $\{(k, k'), (k', k)\}$, eller $\{(k, k''), (k'', k)\}$, eller (for særlige sporskifter) $\{(k, k'), (k', k), (k'', k)\}$, eller $\{(k, k'), (k'', k)\}$, eller $\{(k', k), (k'', k)\}$. Vi siger at mængden af alle en spor–enhed's mulige tilstand angiver dets tilstands–rum. Man kan, mao., tale om en spor-enhed's øjeblikkelige tilstand, og om at spor-enheden kan skifte tilstand. Hvad det er der bevirker en spor-enhed's tilstands–skift siges der her ikke noget om !

Ovenstående beskriver alene et spor–nets basale forhold.

Vha. disse begreber definerer vi nu yderlige begreber.

En rute er en sekvens af forbundne (ie. sammenføjede) spor-enheder "med samt", for hver spor-enhed, en bevægelses–retning således at disse "hænger sammen" i samme retning. En åben rute er en rute som har alle sine spor-enheder's bevægelses–retning i den pågældende spor-enhed's øjeblikkelige, dvs. nuværende tilstand.

En station definerer nul, een eller flere, som regel mindst een perron– og nul, een eller flere side–linjer (*"tracks"* og *"sidings"*, hvor disse '–linjer' ikke må forveksles med linjer mellem stationer). Perron–linjer muliggør på– og af–stigning af passagerer

Det er nu muligt at definere de linjer (ud fra en station) der kan nås (dvs. for hvilket der er en intern stations–rute) fra en perron– eller en side–linje, hhv. de perron– og side–linjer der kan nås fra en vilkårlig linje ind til den pågældende station.

Og så fremdeles.

B.1.3 Tog og Trafik

Vi abstraherer et tog ved en rute og en unik tog–identifikation. Senere tilkommer der evt. flere tog–attributter. Vi kan nu tale om en tog–bevægelse. Der er syv mulige tog–bevægelser, idet beskrivelserne alene henholder sig til spor-enheder, og disse betragtes her som diskrete størrelser. Vi abstraherer således fra positioner relativt til et sammenføjnings–punkt. De syv tog–bevægelser er som følger: (i) Toget optager samme rute i begyndelsen som i slutningen af sin bevægelse. (ii–iii) Toget "kaprer" en ny spor-enhed; dvs. føjer en sådan til sin rute, enten i den ene eller den anden retning. (iv-v) Toget "gir' slip" på, forlader, en spor-enhed, dvs. "fjerner" en sådan fra sin rute, enten i den ene eller den anden retning. (vi–vii) Eller toget både "kaprer" og forlader en spor-enhed, enten i den ene eller den anden retning.

Til et hvert tidspunkt opererer nul, eet eller flere, men et endeligt antal tog på et net. Mellem to, på hinanden "tætte", tidspunkter er det muligt, dels at nogle eller alle togene bevæger sig, at "gamle" tog går ud af drift, og/eller at "nye" tog sættes i drift. Samtidig hermed er det muligt at spor–nettet ændrer tilstand.

Trafik er nu en funktion fra tid over i kombinationer af spor–net og nul, eet eller flere togs' positioner (dvs. ruter).

Vi har ikke i ovenstående, basale, beskrivelse taget hensyn til (i) om to eller flere tog optager overlappende ruter (herunder er stødt ind i hverandre); (ii) om deres "underliggende" rute er åben eller lukket, eller om visse af dens spor–enheder er tilsvarende; (iii) osv.

Men vi foreskriver, som muligvis refutérbare påstande, at *"Vorherre kaster ikke terninger:"* (I) At et tog der til to tidspunkter er registreret i respektive trafikker ikke til et tidspunkt mellem disse to ikke er således registreret. Mao. der er ingen *spøgelses–tog*. (II) At to, eller flere, tog der, til et tidspunkt, bevæger sig langs en rute, ikke til "næste" tidspunkt er "ombyttede". (Her skal, forståeligvis begreberne *næste tidspunkt*, *ombyttede*, oa. defineres.)

Og så fremdeles.

B.1.4 Køreplaner

Ved et stations–besøg forstås et begreb der indeholder flg. attributter: Tog–identifikation, ankomst-tid, stations–navn, og afgangs–tid. (Ved en uden–stop–rejse forstås, foruden en tog–identifikation, en simpel sekvens der starter og slutter med respektive stations–besøg og som "ind–imellem" består af et navn på en linje der forbinder de to stationer for respektive stations–besøg, således at ankomst–tid til den anden station står i et rimeligt forhold til den

mellem–liggende linje's længde og øvrige [ikke her berørte] attributter.) Ved en rejse forstås, foruden en tog–identifikation, en sekvens der starter og slutter med respektive stations–besøg og som "ind–imellem" består af en alternerende sekvens af linje–navne og stations–besøg, begyndende og afsluttende med et linje-navn.

Ved en almindelig, passagér–orienteret køreplan forstås, foruden en topologisk beskrivelse af et spor–net, her en beskrivelse, struktureret som ovenfor angivet, af nul, een eller flere rejser, således at alle tog–navne er unikke og iøvrigt begrænset af sådanne begrænsninger som f.eks. de at rejse–tider er kommensurable med afstande, at visse tog's ankomster "synkroniserer" med andre tog's afgange, etc.

B.1.5 Passagerer

Passagerer forespørger om rejse–muligheder og forespørger om, eller stiller betingelser (tider, priser, mv., korteste, hurtigste rute, mmm.), køber og afbestiller billetter og plads–reservationer, stiger på tog, billeteres eller rejser "sort", rejser forkert, for langt, ændrer tute undervejs, oma.

B.1.6 Gods

Mennesker og firmaer sender fragt med godstoge: Forespørger om fragt–muligheder og forespørger om, eller stiller betingelser (tider, priser, mv., korteste, hurtigste rute, mmm.), køber, afbestiller og ændrer fragt–plads, indleverer godset, får fragt–brev, forespørger om igangværende fragt–transporter ("sporer" indleveret fragt), modtager meddelelse om ankommet gods, afhenter gods, og betaler eventuelle gebyrer.

B.1.7 Planlægning, Drift og Vedligehold

Jernbane–personale planlægger fremtidig drift, langsigtet (næste år, næste sæson), "mellemsigtet" (allokering af stab og vogne til næste uges eller dags' trafik), og kortsigtet (sikrer afsendelse af tog til tiden, overvåger trafikken og foretager eventuelt "sidste–minut" justeringer, etc.). Jernbane–personale sikrer at materiel (rullende og stationært [signaler mv.]) er drifts–sikkert. Og jernbane–personale står for oprettelse (nedlæggelse) af nye (gamle) linjer, stationer, og services, herunder disses planlægning og konstruktion.

B.1.8 Diskussion

Meget mere kunne siges, og meget af det er beskrevet i [5, 49, 44, 8, 45, 46, 50, 51, 25, 39, 39].

Ovenstående beskrivelser er blot råskitser.

B.1.9 Mulig Jernbane–informatik

Der er givet er et specifikt jernbane–domæne, dvs. et jernbane–system. Det er givet i form af et spornet, en stab, rullende materiel (vogne af forskellig art), og en fortid — i form af tidligere, inklusive seneste, års statisitk.

Forfatteren til nærværende notat deltager i en fire års periode i årene 2000–2005 i et EU projekt: AMORE: Algorithmic Methods for Optimising Railways in Europe. I projektet undersøges en meget lang række operations–analytiske planlægnings–problemer, svarende til

nedenstående krav: 1, 3, 5, 6, 8, 10, 12, og 14. Jeg skal derfor, med henvisning til AMORE–projektets web side, ***, undlade at detaljere de specifikke optimalitets–kriterier.

Der optælles 17 råskitse eksempler på krav til programmel der understøtter planlægning og drift af fragmenter af dette specifikke jernbane–system.

1. Køreplans–planlægning:

   Der er specifikt givet det seneste kalender–års passager–statistik: Hvor mange der aktuelt rejste, hvorfra og hvortil, og til hvilke tider, og på hvilken klasse og pris. Der er også givet spørgsmål og svar i forbindelse med en statistisk udvalgt rundspørge der giver troværdige indikationer af hvorledes "det seneste kalender–års passager–statistik" ville (dvs. kunne) have været såfremt der netop havde været tog til de tider de adspurgte (evt. ellers) ville have rejst til, hhv. for.

   Kravet der nu stilles er at udvikle et køreplan–planlægning–system. Som ind–data skal dette system have information om spornettet, rullende stab, hypotetisk (idéelt set) tilgængelig stab, den ovennævnte passager–togs statistik, samt rundspørge. Som ud–data skal systemet levere en køreplan der skal opfylde de forventninger om passager–trafik som ind–data indikerer. Køreplanen skal være optimal mht. flg. kriterier: ... osv. ...

2. Passagértogs–drift: * * * ...

   MERE TILKOMMER SIDEN

3. Tog– og Spor–drifts–planlægning: * * * ...

   MERE TILKOMMER SIDEN

4. Tog– og Spor–drift: * * * ...

   MERE TILKOMMER SIDEN

5. Passager Rejse–planlægning: * * * ...

   MERE TILKOMMER SIDEN

6. Tog–personale–planlægning: * * * ...

   MERE TILKOMMER SIDEN

7. Tog–personale–drift: * * * ...

   MERE TILKOMMER SIDEN

8. Vedligeholds–planlægning: * * * ...

   MERE TILKOMMER SIDEN

9. Vedligeholds—-drift: * * * ...

   MERE TILKOMMER SIDEN

10. Net–planlægning: * * * ...

    MERE TILKOMMER SIDEN

11. Passager–billet–salg og billetering: * * * ...

    MERE TILKOMMER SIDEN

12. Fragtgodstogs–planlægning: * * * ...

    MERE TILKOMMER SIDEN

13. Fragtgods–reservation og Transport: * * * ...

    MERE TILKOMMER SIDEN

14. Almindelig Ranger–planlægning: * * * ...

    MERE TILKOMMER SIDEN

15. Almindelig Ranger–drift: * * * ...

    MERE TILKOMMER SIDEN

16. Godstogs–rangering: Planlægning: * * * ...

    MERE TILKOMMER SIDEN

17. Godstogs–rangering: Drift: * * * ...

    MERE TILKOMMER SIDEN

Én af pointerne, ved at bringe eksempler på råskitse–domæne–krav–definitioner, er at vise at krav–definitionerne "stort set" alene forholder sig til den tidligere anførte domæne–beskrivelse — og at, hvor den ikke gør så, dvs. bruger domæne–termer, der ikke er nævnt tidligere, at dér er det en "smal sag" at rette op på disse mangler, disse undladelser, i domæne–beskrivelsen.

En anden af pointerne er at vise hvorledes domæne–egenskaber der optræder i definitioner for ét sæt krav også optræder i definitioner andre, derfra tilsyneladende "meget forskellige" krav–definitions–sæt.

- *"Før vi har fået set os om har vi, tværs over de ønskede programmel–pakker, fået repræsenteret hele et jernbane–system's domæne !"*

B.2 Logistik–domæne og Krav

Vort logistik–domæne består af flg. entiteter: Et transport–netværk, eet eller flere transport–firmaer, een eller flere sendere og modtagere af fragt, samt af eet eller flere logistik–firmaer. Disse, deres aktiviteter og interaktion, beskrives separat nedenfor.

B.2.1 Synopsis

Essensen i vort logistik–domæne er at kunder sender og modtager fragt, at denne fragt transporteres af transport–firmaers fartøjer, fra et udgangs–knudepunkt i et transport–netværk via ruter mellem knudepunkter og via sådanne knudepunkter — hvor sådan fragt eventuelt omlades — til et slut–knudepunkt, at fartøjer rejser efter faste fartplaner, og at logistik–firmaer arrangerer, for kunder, og i samarbejde med transport–firmaer, sådan transport af fragt — symboliseret ved såkaldte fragt–breve (Engelsk: *"Bill-of-Ladings"*).

B.2.2 Transport–netværk: Ruter og Knudepunkter

Transport–netværk er sammensat af eet eller flere separate transport–del–netværk, eet eller flere for hver af flg. transport–medier: Biler, tog, skibe og fly. Vi abstraherer fra typen af transport–medie. Ved et transport–medie forstås således enten en bil (bus, taxi, lastbil, mfl.), eller et tog (passagertog, godstog, eller blandet tog), eller et skib (passager-skib, fragt–skib, eller blandet), eller et fly (passagerfly, fragtfly, eller blandet). Istedet for det lange ord 'transport–medie' skal vi i det flg. alene bruge termen fartøj. Vi abstraherer også fra bilers mulige stoppesteder (bus–stoppested, taxi holdeplads, lastbils–omlade–plads, etc.), stationer, havne og lufthavne, og bruger istedet fælles–termen: knudepunkt. Herved kan vi også enklere lade to eller flere fartøj typers' knudepunkter være sammenfaldende. Og endelig kan vi ved en simpel rute forstå enten en vej (mellem to stoppe–steder), en jernbane–linje (mellem to stationer), en søvej (mellem to havne), og en luft–korridor (mellem to lufthavne). Knudepunkter og simple ruter har distinkte navne. Simple ruter har alle længde–attributten.

Et transport–netværk består således af to eller flere knudepunkter og een eller flere simple ruter som forbinder disse knudepunkter. Mao., vi "ligner" et transport–netværk ved en simpel ikke–orienteret graf. Dog tillader vi, aht. senere brug, at vi kan analysere et knudepunkt til, foruden sit unikke navn, at have een eller flere af attributterne: Stoppe–sted, station, havn, og lufthavn; og en simpel rute til, foruden sit unikke navn, at have eksakt een af attributterne: Vej, linje, søvej, og luft–korridor. Vi tillader således at par knudepunkter kan være forbundet af nul, een eller flere, distinkt navngivne simple ruter.

B.2.2.1 Simple Ruter Enhver simpel rute er forankret i to knudepunkter, har en længde, og er ellers geografisk positioneret. Enhver simpel rute tillader eet eller flere fartøjer at befare denne simple rute under iagttagelse af en række færdsels–regler, som så enten følges eller ikke omgås.[36]

B.2.2.2 Knudepunkter logistik–firmaer, (ii) opbevarer dette indtil det i det pågældende fragt–brev angivne første rute's fartøj er ankommet, (iii) ser til det bliver lastet, (iv) modtager

[36] I den udstrækning disse regler har direkte betydning for logistik aspektet beskrives disse færdsels–regler.

fragt fra fartøjer til midlertidig opbevaring (v) før videre–befordring af andre fartøjer, og (vi) afleverer fragt til logistik–firmaer. Indbefattet i ovenstående er mao. at knudepunkter modtager og afsender fartøjer, og giver dem midlertidigt ophold.

B.2.3 Transport–virksomheder

Til et fartøj knyttes dels statiske attributter: Transport–kapacitet, f.eks. udtrykt i antal 21 fods *containere* fartøjet kan transportere, dels dets dynamiske attributter: Dets last, dvs. hvilket antal det, til ethvert givet tidspunkt aktuelt transporterer, og, for hver sådan *container*, en sådan kontainers yderligere attributter: fragtbrev, position på fartøjet, mm. (Se nedenfor.)

En transport–virksomhed abstraheres ved flg. to forhold: Et antal eentydigt navngivne (og attributerede) fartøjer og en fartplan. En fartplan angiver, for hvert (operativt) fartøj (dvs. fartøjer fra en delmænge, eventuel hele transport–virksomhedens "flåde", af fartøjer) deres ruter, ankomst– og afgangs–tider, samt fartøjets statiske attributter. En fartplan "strækker sig ud i tid": Dette ("udstræknings"–begreb) kan være udtrykt f.eks. ved at angive et modul, en rejse frem og tilbage, samt at modulet gentages (modulært), f.eks. hver anden måned, dag, time, eller tilsv.

Transport–virksomheder korresponderer med logistik–firmaer: Underretter dem om dets fartøjer og disses fartplaner: tider, priser, betingelser, mv. Transport–virksomheder korresponderer med sine fartøjer: Véd hvor de er, hvornår de ankommer til knudepunkter, hvilken fragt der venter på dem i knudepunkter, hvad der skal losses, hvad der skal lastes, osv. Transport–virksomheder korresponderer således også med knudepunkter.

B.2.4 Sendere og Modtagere af Gods: Fragtbreve

Sendere afsender gods. Modtager modtager gods. Sendere henvender sig til et logistik–firma. Informerer om godsets art, modtagers adresse (mv.), og forespørger om mulige forsendelses–vilkår: korteste, eller hurtigste, eller billigste, eller "sikreste" rute, afsendelses–tider, priser, forsikring, og evt. øvrige vilkår. Sendere beslutter sig for afsendelse: Leverer godset, modtager kopi af fragtbrev, og betaler eventuelt for transporten.

Et fragtbrev anskues her meget generelt: Som et papir– (eller elektronisk–) dokument der beskriver godsets art, afsenders og modtagers navne, adresser mv., transport–ruten: Fra og via hvilke knudepunkter og simple ruter, med hvilke transport–virksomheders' fartøjer, til hvilke tider, til hvilken priser (afsenders og modtagers evt. gebyrer, afgifter, mv.), oplysninger vedr. forsikring, hvorledes modtager adviseres, oma.

Modtagere adviseres om en forsendelses modtagelse af og hos et logistik–firma. Modtager henvender sig til dette, får godset leveret (mod behørig legitimation og dokumentation, f.eks. en kopi af godsets fragtbrev som enten afsender eller logistik–firmaet tidligere har fremsendt), og evt. mod en vis betaling (herunder evt. told oa. gebyrer).

B.2.5 Logistik–firmaer

Logistik–firmaer arrangerer forsendelse af fragt, ofte over vilkårlige ruter. Logistik–firmaer kender til transport–virksomhedernes fart–planer, betingelser, priser, mv. Logistik–firmaer modtager forespørgsler fra afsendere og modtagere af fragt, modtager fragt der skal forsendes

fra afsendere, afleverer fragt til knude–punkter, afleverer fragt der er modtaget i knude–punkter til modtagere, og sporer hvor fragt er. Hvor ændringer i aktuel afvikling af fartøjers trafik betinger det, foretager logistik–firmaer ændringer i fragt–breve.

B.2.6 Diskussion

Meget mere kunne siges. Men nok er sagt til at opnå vort mål: At illustrere behovet for at beskrive, og at antyde det mulige omfang af beskrivelser af logistik–domæner.

Vi har ikke, endnu, fokuseret nævneværdigt på arten af og indholdet i det gensidige "samtale–transaktioner" der finder sted mellem sendere af fragt og logistik–firmaer, mellem logistik–firmaer og transport–virksomheder, mellem transport–virksomheder og deres fartøjer, mellem de tre sidste rolle–havere og knudepunkter, etc.

Afsnit B.3 vil komme ind herpå. Vi henviser især til vores betragtninger i Afsnit B.3.6 på side 94 og i begyndelsen af Afsnit B.3.7.

B.2.7 Mulig Logistik–informatik

Der optælles 15 eksempler på krav til programmel er understøtter planlægning og drift af fragmenter af et logistik–system:

- Transport–virksomhed: Planlægning og Drift:
  1. : * * * ...

     MERE TILKOMMER SIDEN
  2. : * * * ...

     MERE TILKOMMER SIDEN
  3. : * * * ...

     MERE TILKOMMER SIDEN
- Logistik–firma: Planlægning og Drift
  4. : * * * ...

     MERE TILKOMMER SIDEN
  5. : * * * ...

     MERE TILKOMMER SIDEN
  6. : * * * ...

     MERE TILKOMMER SIDEN
- Knudepunkts–administration: Planlægning og Drift:
  7. : * * * ...

     MERE TILKOMMER SIDEN

8. : * * * ...

MERE TILKOMMER SIDEN

9. : * * * ...

MERE TILKOMMER SIDEN

- Transport–fartøjer: Planlægning og Drift

10. : * * * ...

MERE TILKOMMER SIDEN

11. : * * * ...

MERE TILKOMMER SIDEN

12. : * * * ...

MERE TILKOMMER SIDEN

- Afsender og Modtager Aktioner:

13. : * * * ...

MERE TILKOMMER SIDEN

14. : * * * ...

MERE TILKOMMER SIDEN

15. : * * * ...

MERE TILKOMMER SIDEN

B.3 Fra Handel til E–Handel: Domæne og Krav

Vort handels–domæne består af flg. entiteter: Varer, tjeneste–ydelser og betalinger, kunder, detail–handlere, grossister, og fremstillings–virksomheder, samt distribution. Disse, deres aktiviteter og interaktion, beskrives separat nedenfor.

B.3.1 Synopsis

Essensen i vort handels–domæne er at varer købes og sælges, at kunder efterspørger, køber og returnerer varer hos, respektive til detail–handlere; at detail–handlere efterspørger, køber og returnerer varer hos, respektive til grossister; og at grossister sluttelig efterspørger, køber og returnerer varer hos, respektive til fremstillings–virksomheder. Vi skal ikke her se nærmere på levering, distribution, af varer. Vi anskuer dette som en funktion (funktions–klynge) der hører til under logistik. Detail–handlere og grossister optræder således både i køber– og sælger–roller, kunder alene som købere, og fremstillings–virksomheder, i denne sammenhæng, alene som sælgere.

Essensen i vort E–handels–domæne er dels at ovenævnte forhold undstøttes vha. informatik: At efterspørgsel kan ske via nettet, understøttet af "elektroniske", typisk web–baserede kataloger; at ordre—afgivning, ordre–akcept (konfirmering), fakturering og betaling ligeledes kan ske via nettet; og så fremdeles.

Essensen i vort E–handels–domæne er desuden at udvide handels–begrebet med både et agent og et mægler begreb, samt at understøtte, eventuelt automatisere disse. Ved en agent forstås en mekanisme (en person eller en datamatiseret process), der agerer på vegne af en kunde, detail–handel, grossist eller fremstillings–virksomhed. Agenter interagerer med enten købere, sælgere, agenter eller mæglere. Ved en mægler forstås en mekanisme (en person eller en datamatiseret process), der bringer potentielle købere, sælgere, agenter, eller andre mæglere sammen med det formål at formidle en handel.

Essensen i det forneden indførte E–forretnings–begreb er (1) at omfortolke 'handel' til 'forretning'; og (2) at udvide og omfortolke kunde, detail–handel, grossist og fremstillings–virksomheds entiteterne til først at indføre "det offentlige" (O): Statlige, amtlige, amts–kommunale og kommunale institutioner; dernæst — måske arbitrært — at gruppere detail–handel, grossist og fremstillings–virksomheder sammen med andre private virksomheder i en såkaldt privat 'forretnings–gruppe' (P), og kunder i gruppen af borgere (B). Mao. at "definere" (at anskue) 'forretning' som udtrykt gennem een eller flere transaktioner mellem enkelt entiteter indenfor hver af de tre grupper, hvor vi, i optællingen der nu flgr., ved ordningen $X2Y$ antyder at X tager initiativet til en følge af een eller flere transaktioner med Y: $O2O$, $O2P$, $O2B$, $P2O$, $P2P$, $P2B$, $B2O$, $B2P$, og $B2B$.

B.3.2 Varer, Tjeneste–ydelser og Betalinger

Vi skal ikke sondre mellem de to former for handels–entiteter: varer og tjeneste–ydelser. Ved en vare forstår vi en fysisk manifestérbar størrelse, af fysisk art, optagende plads i rummet, noget der kan forbruges, enten éngangs–brug (som f.eks. fødevarer eller energi), eller genbrugelig, men typisk med en vis begrænset levetid (kan nedslides, som f.eks. en bog, en bil, et sæt tøj). En vare kan købes og skifter da hænder: I det ene øjeblik er varen hos sælger, i det næste hos køber som netop har betalt for varen. Ved en tjenesteydelse forstår vi en oftest

ikke–manifestérbar størrelse, af ikke–fysisk art, optagende "plads", ikke i rum, men typisk i tid ! Noget der enten bliver forbrugt, i samme tid, som et teater–besøg, eller som efter en første, tids–mæssig periode, et interval, hvori tjeneste–ydelsen leveres, som f.eks. konsulent–bistand, kan "genbruges": Modtageren af tjeneste–ydelsen er blevet mere vidende, og bruger den nye viden, "igen–og–igen". Også her betales der for tjeneste–ydelsen, muligvis i afdrag.

Mao.: Varer og tjeneste–ydelser kommer i forskellige former, der er varer af type A, af type B, etc., type C, og der er tjeneste–ydelser af type H, af type J, etc., type K, Til hver type er der en betalings–pris. Denne kan afhænge af formen for leverance. Som regel køber man kun eet "eksemplar" af en tjeneste–ydelse, og da for en bestemt periode, et bestemt tids–interval. Men man kan købe indtil flere "eksemplarer" af en vare, nogle leveret nu, andre siden. Der kan så ydes mængde–rabat, mv.

Til varer og tjeneste–ydelser kan man således lad svare et katalog over art, leverance–betingelser og priser.

B.3.3 Kunder, Detail–handel, Grossister, og Fremstillings–virksomheder

En "forsynings–kæde" (*"a supply chain"*) er et aggregat af én eller flere kunder (men det er tilstrækkelgt blot at tale om én kunde); nul eller én detail–handel; nul, én eller flere grossister, hvor disse er ordnede i en følge: lokal–grossist der leverer til detail–handel (eller kunde), mellem–handels grossister der leverer til andre sådanne eller lokal–grossister, og én fremstillings–virksomhed. Mao., kunder kan forestilles at købe direkte fra grossist eller fra fremstillings–virksomhed ("fabriks–udsalg").

Relationen mellem på hverandre følgende led i forsynings–kæden er køber/sælger relationen. Køber kan henvende sig til sælger med flg. "transaktioner": (i) forespørgsel, (iii) ordre, (vi) leverings–akcept, (viii) betaling, (ix) reparations–indlevering og (xi) klage (retur). Sælger kan henvende sig til køber med flg. "transaktioner": (ii) tilbud, (iv) konfirmation, (v) leverance, (vii) faktura, (x) reparations–ydelse, og (xii) refundering. Her antyder de romerske numeraler en mulig sekventiering af disse inter–agerende transaktioner. Desuden kan sælger henvende sig til købere med (ii') tilbud, der ikke var en flg. af en (i) forespørgsel: Postkasse–reklame–sager o.lign.

Læseren kan selv modificere ovenstående til "bedre" at passe på tjeneste–ydelses–konceptet.

Ovenstående var formuleret sådan som vi opfatter begrebet "klassisk varehandel".

B.3.4 Agenter og Mæglere (*"Agents and Brokers"*)

Vi gentager: *Ved en agent forstås en mekanisme (en person eller en datamatiseret process), der agerer på vegne af en kunde, detail–handel, grossist eller fremstillings–virksomhed. Agenter interagerer med enten købere, sælgere, agenter eller mæglere.*

I "klassisk varehandel" kan man bede sin nabo om at handle for én. Naboen er da éns agent. Og en detail–handler kan reklamere gennem TV, radio, i aviser, eller gennem *"tele–marketing":* Mål–rettet telefon–salg. TV–stationen, radio–stationen, avisen, hhv. tele–marketing–firmaet (der yder en tjeneste) er da detail–handlerens agent.

Vi gentager: *Ved en mægler forstås en mekanisme (en person eller en datamatiseret process), der bringer potentielle købere, sælgere, agenter, eller andre mæglere sammen med det formål at formidle en handel.*

I "klassisk forretning" kan en konstellation af købere og sælgere i fællesskab, men typisk urelaterede, tids– og steds–uafhængigt af hverandre, engagere en mægler, til eksempel et firma der bringer disse parter (købere og sælgere) sammen med henblik på design og konstruktion af f.eks. et nøgleklart kraft/varme–værk.

Vi opfordrer læseren til selv at eventuelt omfortolke eller revidere den tidligere anførte liste af køber/sælger transaktioner således at den er tilpasset agenter og mæglere.

B.3.5 Offentlige og Private Virksomheder, og Borgere

Hidtil har vi antaget en forsynings–kæde der blot involverede private virksomheder (detail–handlere, grossister, fremstillings–virksomheder (fabrikker), agenter, og mæglere), hhv. borgere. Og vi har set på transaktioner mellem disse to parter (*B2P*, *P2B*, og *P2P*).

Vi skal nu udvide transaktions–begrebet til, foruden 'handel' (med varer og tjeneste–ydelser) også at omfatte 'forretning' i bredeste forstand, samt endelig også at involvere det offentlige: Stat, amt og kommune (*O*). Vi skal nedenstående forsøge antyde hvad vi forstår ved et "udvidet transaktions–begreb".

- *O2O:* Offentlige institutioner henvender sig til andre sådanne: (i) Udbeder sig statistisk materiale, (ii) prognose, (iii) beder én offentlig myndighed overtage en sag fra en anden offentlig myndighed, (iv) oplyser om nye, eller ændrede, eller ikke–længere relevante forordninger, etc.
- *O2P:* Offentlige institutioner henvender sig til prvate virksomheder: Udbeder sig (i) skatte–, (ii) miljø–, (iii) personale–mæssige eller andre oplysninger, (iv) beder om betaling af skatter, moms, og afgifter, etc.
- *O2B:* Offentlige institutioner henvender sig til borgere: Udbeder sig (i) folketællings– eller (ii) ejendoms–oplysninger, (iii) beder om at få selvangivelsen indleveret, (iv) eller skatter betalt, etc.
- *P2O:* Private virksomheder henvender sig til det offentlige: Ansøger om godkendelse af (i) miljø–, eller (ii) bygnings–mæssige eller lign. forhold, (iii) et aktieselskab giver oplysninger til fonds–børsen, etc.
- *P2P:* Private virksomheder henvender sig til andre private virksomheder: (i) Alm. køb og salg, (ii) firma–overtagelse, etc.
- *P2B:* Private virksomheder henvender sig til borgere: (i) Alm. salg, (ii) tilbud om ansættelse, etc.
- *B2O:* Borgere henvender sig til det offentlige: Ansøger om (i) sociale ydelser, (ii) optagelse på en uddannelse, (iii) udsættelse af skat, (iv) stilling i det offentlige, etc.
- *B2P:* Borgere henvender sig til private virksomheder: (i) Alm. køb, (ii) stilling i det private, etc.
- *B2B:* Borgere henvender sig til andre borgere: (i) Græsrods–bevægelse, (ii) fritids–sysler, etc.

Vi lader ovenstående "stå for sig selv", og overlader til læseren at "boble" videre.

B.3.6 Diskussion

Vi har skitseret et bredt spektrum af handels og forretnings–begreber.

Hvad vi ikke har nævnt er de "samtale–transaktioner", de, med et moderne begreb hentet fra lingvistikken, *"speech acts"*, som udøves af partnerne i en handels–transaktion, stort set ligegylding hvilken af de ovenfor antydede. Vi har ej heller nævnt de former for ræssonnementer partnerne hver for sig, men under indflydelse af "samtalerne", foretager — specielt de modal–logikker der her bringes i brug: Temporale, epistemiske, deontiske, oma.

B.3.7 Mulig E–Forretnings Informatik

Med udgangs–punkt i netop de just ovenfor antydede *"speech act"*–teorier og modal–logikker, med udgang i disse, er det så at vi mener at kunne tilføre emnet E–forretning nogle dimensioner som går langt ud over dagens E–handels–trivialiteter. Mao.: Kun veltrænede dataloger cum informatikere kan virkelig "løfte" brugen af IT.

Der optælles 11 eksempler på krav til programmel er understøtter planlægning og drift af fragmenter af et E–forretnings–system:

- Traditionel E–Handel:
  1. Traditionelt Køb og Salg: * * * ...

     MERE TILKOMMER SIDEN
  2. Auktioner: * * * ...

     MERE TILKOMMER SIDEN
  3. Agenter: * * * ...

     MERE TILKOMMER SIDEN
  4. Mæglere: * * * ...

     MERE TILKOMMER SIDEN
  5. "Speech Act Transaktioner": * * * ...

     MERE TILKOMMER SIDEN
- E–Forretning:
  6. E–Forretning — Offentlig–til–Offentlig: * * * ...

     MERE TILKOMMER SIDEN
  7. E–Forretning — Offentlig–til/fra–Firma: * * * ...

     MERE TILKOMMER SIDEN
  8. E–Forretning — Offentlig–til/fra–Borger: * * * ...

     MERE TILKOMMER SIDEN
  9. E–Forretning — Firma–til–Firma: * * * ...

MERE TILKOMMER SIDEN

10. E–Forretning — Firma–til/fra—Borger: * * * ...

MERE TILKOMMER SIDEN

11. E–Forretning — Borger–til–Borger: * * * ...

MERE TILKOMMER SIDEN

B.4 Finans–sektor–domæne og Krav

Vort finans–sektor–domæne består, "udpluks–vist", af flg. entiteter: Kunder, banker, kredit–anstalter, børsmæglere & børser, forsikrings–virksomheder, og kredit–korts–virksomheder. Disse, deres aktiviteter og interaktion, beskrives nedenfor.

B.4.1 Synopsis

Essensen i vort finans–sektor–domæne er at penge (generelt: værdi–papirer cum værdi–instrumenter) "skifter hænder": Indsættes på lønkonti, hæves derfra, enten som rede penge, eller overføres til låne– og spare–konti, eller tilkøb af værdi–papirer, eller til betaling af forsikrings–præmier, eller indsættes på dertil egnede konti fra salg af værdi–papirer, eller modtaget som flg. af udbetalt forsikring, og meget, meget andet.

En afledet essens ved vort finans–sektor–domæne er at kunder og finans–virksomeheder har financiel nytte heraf: Får sikkerhed og rimeligt afkast af indsatte midler, hhv. opnår en rimelig profit, herunder at enhver financiel tjeneste, på, for kunderne rimelig vis, bidrager til firmaernes indtjening (*"value added services"*) — for derved at sikre disse virksomheders stabilitet og dermed tjener kundens sikkerhed.

B.4.2 Værdi–papirer / Værdi–instrumenter

Aht. det flg. skitserer vi her spektret af værdi–papirer, også kaldet værdi–instrumenter: Rede penge (i diverse valutaer), checks, kreditkorts–"slips", skøder, pantebreve, gældsbreve, obligationer, aktier, optioner, etc. Ethvert værdi–papir har en handels–værdi (evt. forsk. købs–, hhv. salgs–værdier). Sådanne værdier ændrer sig løbende, har en kurs.

B.4.3 Banker

Der er forsk. typer banker: De hvor Du og jeg har vores lønkonti, sætter penge ind (på sådanne eller spare eller andre konti), hæver dem, låner penge (får kasse–kredit), etc. Og der er andre typper banker: Investerings–banker, kredit–banker (i forb. med f.eks. huslån oa.), etc.

For den for os almindeligste type bank anføres: En kunde kan have indtil flere konti. Enhver konto har et unikt nummer. Flere kunder kan dele samme konto. En konto kan oprettes, "fryses", og nedlægges; der kan indsættes indskud på, og hæves beløb fra en konti; man kan anmode om et konto–udtog, dvs. en liste over "seneste" bvægelser på en konto. Banken kan tilskrive renter til en positiv, og hæve renter fra en negativ konto. Kombinationer af hævning fra og indskud på konti svarer til overførsel mellem konti. Disse kan inludere "konti" i andre financielle institutioner: Forsikrings–anstalter, børs–mæglere, mv.

Og meget andet.

B.4.4 Kredit–korts–virksomheder

Der er flg. fire "størrelser" at tage hensyn til her: Et aftryk af et kredit–kort, to banker, sælger's, hhv. køber's, og en "clearing"–central.

Et kredit–kort udtrykker at dets ejer (køber) har en "vis kredit" i den financielle virksomhed der står bag kortet: En bank eller tilsv. Vha. et kredit–kort kan man "effektuere"

betaling for en handsel. Købers kredit–kort "aftrykkes" af sælger, salgs–beøb, dato, sælgers og købers identitiet (dvs. underskrifter) registreres på kortet. Sælger's kopi af aftrykket præsenteres i sælgers bank. Sælgers bank videre–sender denne kopi til den "clearing"–central der er agent for kredit–kort–firmaet og de to banker. Registrering (dvs. "clearing") betyder at købers bak påtager sig skyld og derved står inde for at det anførte beløb hæves fra en dertil designeret køber–konto og overføres til en dertil designeret sælger–konto.

Det "bagved–liggende" kredit–kort–firma anfører overførsel, dvs. transaktionen ved næste konto–udtog.

Og meget andet.

B.4.5 Børs–mæglere & Børser

Børs–sektoren handler med værdi–papirer.

Ejere (dvs. sælgere) af disse præsenterer ønsker (tilbud) om salg — til en børs–mægler — ved,f.eks. at angive navn og type på værdi–papiret, antal (styk) der ønskes solgt, salgs/-tilbuds–perioden, samt et "spænd": absolut laveste pris for eventuelt salg, hhv. akceptabel "højeste" pris til hvilken et salg ubetinget kan finde sted — idet sælger forventer at børs–mægler så vidt muligt sælger til denne pris, eller gerne højere, men akcepterer salg blot over minimum pris hvis 'markedet' så betinger.

Købere af værdi–papirer præsenterer ønsker (tilbud) om køb — til en børs–mægler — ved,f.eks. at angive navn og type på værdi–papiret, antal (styk) der ønskes købt, købs/-tilbuds–perioden, samt et "spænd": absolut højeste pris for eventuelt køb, hhv. akceptabel "laveste" pris til hvilken et køb ubetinget kan finde sted — idet køber forventer at børs–mægler så vidt muligt køber til denne pris, eller gerne lavere, men akcepterer køb blot over minimum pris hvis 'markedet' så betinger.

En børs–mægler "bruger" nu børsen som et instrument til at tilgodese dets kunder — der jo typisk vil være en blanding af sælgere og købere — og børsens andre børs–mægleres købende og sælgende kunder.

Og meget andet.

B.4.6 Forsikrings–anstalter

Der er her tale om forskellige former for forsikrings–anstalter: livs–, pensions–, syge–, familie–(hus, brand, indbo (tyveri, røveri), mm.), arbejdsløsheds–, risiko–, oma. forsikrings–anstalter.

Ved en forsikring forstås * * * ...

MERE TILKOMMER SIDEN

En forsikring kan tegnes (oprettes) eller opsiges, jævnlige forsikrings–præmier indbetales, og en forsikring kan nedlægges — hvorefter en evt. "balance", alt efter forsikrings–typen, kan udbetales. Krav på forsikrings–godtgørelse kan rejses, afvises, justeres og evt. honoreres. Forsikrings–præmier kan forhøjes eller nedsættes.

Overførsler af penge–beløb finder, som regel, sted, mellem nærmere designerede bank–konti og forsikrings—konti.

Og meget andet.

B.4.7 Mulig Finans–sektor–informatik

Der optælles 7 eksempler på krav til programmel er understøtter planlægning og drift af fragmenter af et finans–system:

- Banker:
  1. Alm. og Intra–Bank–forretning: * * * ...

     MERE TILKOMMER SIDEN
  2. Inter–Bank–forretning: * * * ...

     MERE TILKOMMER SIDEN
  3. Bank/Forsikrings–forretning: * * * ...

     MERE TILKOMMER SIDEN
  4. Bank/Børs–forretning: * * * ...

     MERE TILKOMMER SIDEN
  5. Bank/Kreditkort–forretning: * * * ...

     MERE TILKOMMER SIDEN
- Kredit–kort:
  6. Sælger–Køber Overføring/"Clearing": * * * ...

     MERE TILKOMMER SIDEN
  7. "Clearing": * * * ...

     MERE TILKOMMER SIDEN
- Børs:
  8. Mægler — Køb og Salg af Aktier og Obligationer: * * * ...

     MERE TILKOMMER SIDEN
  9. Børs — "Clearing" af Aktier og Obligationer: * * * ...

     MERE TILKOMMER SIDEN
  10. IPO: * * * ...

      MERE TILKOMMER SIDEN
- Forsikring:
  11. Alm. og Intra–Forsikrings–forretning: * * * ...

      MERE TILKOMMER SIDEN
  12. Inter–Forsikrings–forretning: * * * ...

MERE TILKOMMER SIDEN

13. Interaktioner: Forsikring / Det Offentlige: * * * ...

MERE TILKOMMER SIDEN

B.5 "Flow"–systemer: Domæne og Krav

Vort "flow"–system–domæne består af flg. entiteter: Produktions–ressourcer (mennesker, maskiner, penge), materialer, planer for brug af (produktions– og materiale–) ressourcer, og disse planers udførelse vha. mennesker, maskiner, penge og materialer.

B.5.1 Synopsis

Én måde at anskue essensen i vort "flow"–system–domæne er at tale om projekter og produktion. Vi skal her anskue disse to begreber som værende identiske: Projekter producerer "resultater", dvs. fremstiller manifestérbare produkter eller yder mere eller mindre manifestérbar service (konsulentbistand).

Produktion, gennemførelse af dele af et projekt, forstås som en process.

Essensen i vort "flow"–system–domæne er at produktion sker ved at diverse produktions–ressourcer bringes sammen og samlet "påtrykkes" materiale–ressourcer, ogat denne "påtrykning" derved omvandler disse materiale–ressourcer til mellem– eller slut–produkter. Produkter er blot en form for ressource.

B.5.2 Ressourcer

Materiale–ressourcer er typisk sådanne ressourcer som kun kan bruges (kun kan "processes") éngang. Tilsvarende for de financielle produktions–ressourcer. Maskin– og menneske–ressourcer, i modsætning hertil, kan genbruges, som oftest serielt, én efter én.

B.5.2.1 Materiale Ressourcer : Ved en materiale–ressource forstås en fysisk sørrelse, der er forarbejdet, eller kan forarbejdes: formet, hhv. formes, vha. produktions–ressourcer. En materiale–ressource har en type.

Eksempler på materiale–ressource–*type*: jern (I–, H–, oa. profil jern–bjælker, stål–plader, mv.), træ (planker, lister, finer–plader, mv.), olie (petroleum, benzin, terpentin, mv.), ilt, etc.

Materiale–ressourcer har *attribut*ter, og til attributter kan der knyttes *værdi*er og indikatorer (f.eks. værdi–intervaller).

Eksempler på materiale–attributter: Stål's jern, krom, nikkel oa. metal–egenskaber. En træbjælke's mål–, vand–, harpiks–, oa. egenskaber. Eksempler på attribut–værdier: 18% krom indhold, 8% nikkel indhold, 3 fod 4 tommer, etc.

B.5.2.2 Produktions Ressourcer : Produktions–ressourcer er enten af typen engangs– eller flergangs–forbrugbar. Ved en

Ved en engangs–forbrugbar produktions–ressource forstås typisk sådanne som tid, energi og klar– og rengørings–ressourcer (vand, ilt, boreolie). Ingen af disse er materiale–ressourcer: De indgår ikke i resultatet af en produktion–process.

Ved en flergangs–forbrugbar produktions–ressource forstås et rum (bygning: kontor, fabrik), maskine eller et menneske.

B.5.2.2.1 Maskiner : Ved en maskine forstås her et instrument der enten automatisk, eller betjent af mennesker, kan bearbejde ("processe") bestemte typer materiale–ressourcer med

henblik på at ændre enten dets type eller visse af dets attribut–værdier, eller en kombinaton heraf.

Eksempler på maskiner: Sav, høvl, bor, mv.; katalysator (tårn), bunsen–brænder, mv.

B.5.2.2.2 Mennesker : Mennesker planlægger projekter (produktion), betjener maskiner, overvåger produktions–processer, styrer rumlig og ressource allokering og tidslig fordeling (*"scheduling"*) af ressourcer, mmm.

B.5.2.2.3 Financielt Instrument : Ved et financielt instrument forstås penge eller sådant som direkte kan bruges som betalings– og løn–middel.

B.5.3 Projekt– cum Produktions–planer

Produktions–ressourcer påtrykkes materiale–ressourcer efter en eller anden "mønster". Nogle serielt, andre parallelt: Materiale–ressourcerne *m, m', ..., m"* bearbejdes først af produktions–ressource *p,* dernæst af produktions–ressource *p',* etcetera, *p"*. Eller produktions–ressourcerne *q, q', ..., q"* bearbejder samtidigt respektive materiale–ressource–mængder *mm, mm', ..., mm"*.

En produktions–plan er at ligne ved en orienteret, acyklisk graf hvis éntydigt navngivne knudepunkter står for hver sin bestemte (atomiske, eller enkelt–) bearbejdnings–process vha. en bestemt produktions–ressource, og hvis annoterede kanter står for materiale–ressourcer: én kant, én bestemt type materiale–ressource af bestemte attribut–værdier.

B.5.4 Projekt– cum Produktions–planlægning

Ved produktions–planlægning forstås konstruktionen af en produktions–plan: En graf. Produktions–planlægning fastlægger, mao., hvilke knudepunkter produktions–grafen skal have, hvilke bestemte produktions–ressourcer der skal knyttes til hvilke knudepunkter, hvilke bestemte materiale–ressourcer der skal knyttes til hvilke kanter.

Til knudepunkter kan man knytte estimater eller aktuel forbrugt process–tid, energi, mmm. Til kanter kan man knytte tid for, og arten af distribution af de til en kant knyttede materiale–ressourcer mellem kantens to knudepunkter.

B.5.5 Projekt– cum Produktions–udførelse

Ved produktions–udførelse, dvs. ved produktion forstås nu en mængde sekventierede og parallelle bearbejdnings–processer således at disse kan siges at svare til en fortolkning af en produktions–plan: Så–at–sige en "eksekvering" baseret på en graf. Enten eksisterende, på forhånd fastlagt, eller ikke–eksisterende, men, som, alt efter produktionen "skrider frem", bliver defineret ved denne produktion.

Da grafen er acyklisk vil der være mindst ét knudepunkt som ingen ind–kanter har. Disse knudepunkter har, som regel, ud–kanter, og disse knudepunkter svarer til initiering af respektive første bearbejdnings–processer. Der vil også være mindst ét knudepunkt som ingen ud–kanter har. Disse knudepunkter har, som regel, indd–kanter, og disse knudepunkter svarer til afslutning af respektive sidste bearbejdnings–processer.

B.5.6 Ressourcer, Produktion og Produkter

En virksomhed besidder typisk flere enheder af hver ressource. Og samme virksomhed engagerer sig typisk i produktion af én eller flere produkter. Til enhver tid er en bestemt mængde ressourcer knyttet ("bundet") til produktion af et bestemt (antal) produkt(er).

B.5.7 Ressource–ledelse

Ved ressource–ledelse forstås nu planlægning af produktions–planer og overvågning og styring af produktion efter sådanne planer således at ressource–forbrug "følger" på forhånd udarbejdede produktions–planer.

B.5.7.1 Strategisk Ressource–Ledelse : Ved strategisk ressource–ledelse forstår vi planlægning og udførelse af en ikke–produktions–bestemt omsætning af én type ressource til en anden resssource: Typisk forvandling af *"good will"* til penge via optagelse af lån eller udbyging af aktie–kapital, eller køb eller leje (hyre) af primære produktions–ressourcer (bygninger, maskiner, mennesker), eller salg eller frigørelse ("fyring") af sådanne primære produktions–ressourcer.

B.5.7.2 Taktisk Ressource–Ledelse : Ved taktisk ressource–ledelse forstår vi allokering og tidslig fordeling af primære produktions–ressourcer: Deres "binding" ("knytning") til produktion af bestemte produkter.

B.5.7.3 Operationel Ressource– Ledelse : Ved operationel ressource–ledelse forstår vi allokering og tidslig fordeling af alle produktions– og materiale–ressourcer samt overvågning og styring af de samlede bearbejdsnings–processer, deres aktuelle forløb mv.

B.5.8 Diskussion

Vi har skitseret væsentlige elementer ved produktion, produktions–planlægning, de dertil knyttede resssourcer, deres planlægning og brug. Vi har fokuseret entydigt på produktion af manifestérbare produkter. Men, mener vi, man kan udskifte ordet produkt med service, produktion med projekt, og — med trivielle andre justeringer af begrebs–apparatet (materiale–ressource, produktions–ressource, etc.) — derved opnå at begrebet en plan kan overføres til også at "gælde" for service–ydende projekter.

Bemærk at vi ikke har taget særlig notits af ressource–distributions– (i betydning transport–) problematikken. Vi anskuer produktions– og transport–problematikkerne for "ortogonale". Og vi mener at en forenklet variant af logistik–problematikken — således som behandlet i Afsnit B.2 — kan tilføres (kan "skrues" på) produktions–problematikken ("mere–eller–mindre") uafhængigt af denne.

B.5.9 Mulig "Flow System"–informatik

Der optælles 10 eksempler på krav til programmel er understøtter planlægning og drift af fragmenter af et projekt– cum produktions–planlægnings– og eksekverings–system:

1. : * * * ...

MERE TILKOMMER SIDEN

2. : * * * ...

MERE TILKOMMER SIDEN

3. : * * * ...

MERE TILKOMMER SIDEN

4. : * * * ...

MERE TILKOMMER SIDEN

5. : * * * ...

MERE TILKOMMER SIDEN

6. : * * * ...

MERE TILKOMMER SIDEN

7. : * * * ...

MERE TILKOMMER SIDEN

8. : * * * ...

MERE TILKOMMER SIDEN

9. : * * * ...

MERE TILKOMMER SIDEN

10. : * * * ...

MERE TILKOMMER SIDEN

B.6 Sundheds–sektor–domæne og Krav

Vor sundheds–sektor–domæne består af flg. entiteter: Borgere (raske eller potentielt syge), privat praktiserende læger, sundheds–sygeplejersker, apoteker, klinikker af forskellig at (fysio–terapeuter, kiropraktikere, m.fl.), klinisk analyse laboratorier, hospitaler, rekonvalescent centre, den farmaceutiske industri, den mediko-tekniske industri, sygeforsikrings–anstalter, samt det offentliges overvågning og styring af denne sektor, i Danmark gennem Sundhedsstyrelsen, Statens Seruminstitut, Lægemiddelstyrelsen, og Sundhedsministeriet. Disse, deres aktiviteter og interaktion, beskrives separat nedenfor.

B.6.1 Synopsis

Essensen i vor sundheds–sektor–domæne er at borgerne bliver syge og skal helbredes: At borgerne "går til lægen", undergår klinisk analyse (får taget blodprøver, Röntgen billeder, MR–skanninger, elektro–kardio–grammer, mm.), henter medicin på apoteket, får besøg af den lokale sundheds–sygeplejerske, kommer på hospitalet, blir' behandlet af fysioterapeuter og kiropraktorer, kommer på rekonvalescens, oa.

Der er, mao., tale om et "flow" (en rute–føring) af mennesker, materialer (medicin, bandage, krykker, mm.), information (recepter, syge–journaler, mm.), og styre–signaler.

B.6.2 Interessent–grupper og Brugere

* * * ...

MATERIALE FORVENTES UDARBEJDET TIL NÆSTE VERSION

B.6.3 Patient–forløb

* * * ...

MATERIALE FORVENTES UDARBEJDET TIL NÆSTE VERSION

B.6.4 Patient–journaler

* * * ...

MATERIALE FORVENTES UDARBEJDET TIL NÆSTE VERSION

B.6.5 Hospitals–forløb

* * * ...

MATERIALE FORVENTES UDARBEJDET TIL NÆSTE VERSION

B.6.6 Diskussion

* * * ...

MATERIALE FORVENTES UDARBEJDET TIL NÆSTE VERSION

B.6.7 Mulig Sundheds–sektor–informatik

Der optælles 10 eksempler på krav til programmel er understøtter planlægning og drift af fragmenter af et sundheds–sektor–system:

1. : * * * ...

   MERE TILKOMMER SIDEN

2. : * * * ...

   MERE TILKOMMER SIDEN

3. : * * * ...

   MERE TILKOMMER SIDEN

4. : * * * ...

   MERE TILKOMMER SIDEN

5. : * * * ...

   MERE TILKOMMER SIDEN

6. : * * * ...

   MERE TILKOMMER SIDEN

7. : * * * ...

   MERE TILKOMMER SIDEN

8. : * * * ...

   MERE TILKOMMER SIDEN

9. : * * * ...

   MERE TILKOMMER SIDEN

10. : * * * ...

    MERE TILKOMMER SIDEN