CONSTRUCT

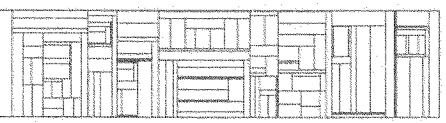
Et værktøj til systemudvikling

af

Ole Jacobsen
og
Claus Qvistgaard

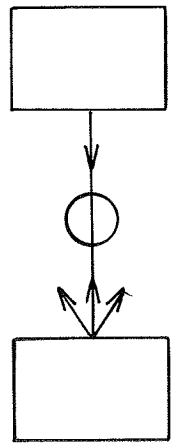
DAIMI PB - 76
Maj 1977

Matematisk Institut Aarhus Universitet
DATALOGISK AFDELING
Ny Munkegade - 8000 Aarhus C - Danmark
Tlf. 06-12 83 55



C O N S T R U C T

RAPPORT NR. 1



af
Ole Jacobsen
og
Claus Qvistgaard,

Århus, maj 1977.

FORORD.

Denne første CONSTRUCT-rapport er resultatet af et samarbejde mellem Ole Jacobsen og Claus Qvistgaard. Vi er begge studerende ved Datalogisk Afdeling Matematisk Institut Århus Universitet, og vi har i vores studium deltaget aktivt i praktisk systemarbejde henholdsvis på B og O i Struer og FDB i Viby.

Vi har begge i vort arbejde savnet en konstruktiv metode til brug i de indledende faser af systemarbejdet.

Denne rapport er en præsentation af en ny metode til brug i systemarbejdets første faser og giver et værktøj til design og konstruktion af systemets datamodel.

Rapporten indgår i begge vore specialer.

Til slut ønsker vi at rette en særlig tak til Kristen Nygaard, Jens Peder Vium og Børge Sand Kirk, fordi de - hver på sin måde - har ydet en uvurderlig hjælp ved udarbejdelsen af denne rapport.

Århus den 5. maj 1977
Ole Jacobsen
Claus Qvistgaard

INDHOLDSFORTEGNELSE.

1. INDLEDNING.

1.1 CONSTRUCT.

I denne rapport introduceres en samling værktøjer til brug i systemarbejdet: CONSTRUCT.

CONSTRUCT (CONceptual STRUCTure) er en samling beskrivelsesteknikker med særlige faciliteter til analyse og beskrivelse af begrebsmæssige informationsstrukturer.

1.2 Traditionelle faser i systemarbejde.

I takt med EDB-systemernes stigende kompleksitet og omfang, har man udviklet metoder og teknikker til brug i arbejdet med udviklingen af disse EDB-systemer. Dette arbejde opdeles sædvanligvis i 6 faser:

1: Systemanalysefasen,
hvor man analyserer forretningsgangene i det bestående administrative system, og klarlægger ønsker og krav til det EDB-system, man overvejer at gennemføre. Disse ønsker og krav formaliseres i en beskrivelse af input-, output- og registerdata.

2: Systemkonstruktionsfasen,
hvor man specificerer systemets konstruktion, omfatter:
Definition af sammenhænge mellem inddatatransaktioner og registerdata.
Definition af systemets moduler.
Definition af regelsæt for opdatering og produktion af uddata.

Valg af lagringsstruktur for registre.
Valg af databærende medier for registre.
Dimensionering af datalagre.

3: Programmeringsfasen,
hvor det konstruerede system beskrives i et programmeringssprog (programmering).

4: Testfasen,
hvor programmerne aftestes, dels modulvis (modultest), dels samlet (systemtest).

5: Konverteringsfasen,
hvor det nye EDB-system integreres i det omkringliggende system, kartoteker oprettes og systemets validitet afprøves.

6: Driftsfasen,
hvor EDB-systemet kører som en integreret del af det omkringliggende administrative system.

1.3 Faseopdeling ved brug af CONSTRUCT.

Begreber.
VIRKELIGHEDEN = den materielle og mercantile virkelighed, omfatter såvel personer som materialer og maskiner. Personerne i virkeligheden styrer eller administrerer virkeligheden eller dele af denne på grundlag af viden om virkelighedens tilstand. Denne videns omfang og struktur er afhængig af det behov for viden, den pågældende person har, for at kunne operere. Disse behov skifter fra person til person afhængig af hvilke funktioner personen udfører. Vi har altså flere forskellige VIRKELIGHEDSOPFATTELSEr, som hver

for sig omfatter begreber eller dele af begreber knyttet til virkeligheden samt sammenhænge mellem disse.

Tilsammen giver disse partielle virkelighedsopfattelser en total DATAMODEL, som omfatter alle begreber og delbegreber knyttet til virkeligheden og sammenhænge mellem (struktur på) disse. I datamodellen kan samtlige partielle virkelighedsopfattelser genfindes.

Personer, der opererer i virkeligheden udgør tilsammen VIRKELIGHEDSADMINISTRATIONEN. Denne virkelighedsadministration udfører sine handlinger på grundlag af viden fra den totale datamodel. Disse handlinger forårsager ændringer i virkelighedens tilstand og derigennem på indholdet i datamodellen. Virkelighedsadministrationens handlinger kan beskrives i en SYSTEMBESKRIVELSE, der altså bygger på datamodellen.

CONSTRUCT-faserne.

Analysefasen.
Analysefasen omfatter analyse af datamodel og analyse af virkelighedsadministration. Den dokumenteres i en beskrivelse af datamodellen og en systembeskrivelse byggende på datamodellen. De to delanalyser kan ikke holdes skarpt adskilt, hvorfor de er knyttet sammen i én fase.

Analyse af datamodel.
Her analyseres først de partielle virkelighedsopfattelser. Det er her vigtigt at få isoleret de begreber eller delbegreber, virkelighedsopfattelsen omfatter, samt at få specificeret hvordan sammenhængene mellem dem er. Disse beskrives som partielle

datamodeller, som samarbejdes og kontrolleres for konsistens.

Efter denne konsistenskontrol følger en eventuel modificering af de partielle datamodeller, så konsistens opnåes. Vi har nu en total struktur, hvori alle partielle strukturer genfindes. Den totale datamodel.

Analyse af virkelighedsadministrationen.
På baggrund af de partielle datamodeller analyseres de handlinger, der udføres på dem. Disse handlinger fastholdes i partielle systembeskrivelser.

De partielle systembeskrivelser samarbejdes i en total systembeskrivelse, og denne beskrivelse skal danne grundlag for programmeringsfasen.

Programmeringsfasen.
Idéen er nu, at det programmeringssprog, der benyttes skal ligge meget tæt op ad systembeskrivelsessproget. Under denne forudsætning omfatter programmeringsfasen, foruden programmeringen, udvælgelse af de dele af systembeskrivelsen, der skal overføres til EDB. Er forudsætningen ikke opfyldt, kræves yderligere visse dele af den traditionelle konstruktionsfase gennemført før programmeringen.

Testfasen,
Konverteringsfasen og
Driftsfasen
gennemføres traditionelt.

Kommentarer.
Som systembeskrivelsesværktøj benyttes DELTA.
Til beskrivelse af datamodellen mangler DELTA

imidlertid de faciliteter, der gør det nemt at beskrive sammenhænge mellem begreber. Vi mangler altså et stykke værktøj, der dels har faciliteter til beskrivelse af begreber, begrebsindhold og begrebssammenhænge, og dels kan forbindes til DELTA-systembeskrivelsen.

2. ANALYSE AF BEHOV FOR ET VÆRKTØJ.

For at kunne arbejde med systembeskrivelse som skitseret i foregående afsnit må vi have et redskab, der kan bruges ved analyse og beskrivelse af datamodellen.

I første del af analysefasen, hvor man skal kortlægge de partielle strukturer, er det vigtigt at kunne fastholde den virkelighedsopfattelse, man er i færd med at analysere. Dette skal kunne gøres på en overskuelig måde, og der må derfor ikke indgå information, der på dette tidspunkt er uvæsentlig.

I de første partielle analyser af virkelighedsopfattelserne er det vigtigt at få isoleret begreberne, samt at få kortlagt sammenhængene mellem dem. Det er imidlertid uvæsentligt her at specificere hvilke atributter, der i den endelige datamodel skal knyttes til begreberne. Disse atributter kan være en hjælp til at isolere begreberne, men behøver ikke direkte at indgå i beskrivelsen.

Værktøjet skal derfor her kunne bruges til at give en overskuelig beskrivelse af begreber og begrebssammenhænge.

Beskrivelsen skal kunne bruges til at kontrollere, at man har fået virkelighedsopfattelsen fastlagt rigtigt. Alle personer, der opererer i virkeligheden, skal altså kunne forstå beskrivelsen, og den skal derfor være intuitiv klar.

Når man skal lave totalbeskrivelsen ud fra de partielle beskrivelser skal man igen have et redskab til hjælp. Her stilles lidt andre krav,

idet den totale beskrivelse indeholder flere begreber og sammenhænge end de partielle beskrivelser gør. Man kan derfor blive nødt til at sløje af på kravet om, at det skal være intuitivt og overskueligt.

En forudsætning for at kunne lave en total beskrivelse, er at de partielle beskrivelser er indbyrdes konsistente. Redskabet skal derfor også være velegnet til at finde de steder, hvor kravet til denne konsistens ikke er opfyldt.

På grundlag af de enkelte virkelighedsopfattelser skal totalbeskrivelsen nu kunne udvides med de atributter, som er knyttet til begreberne.

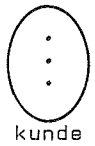
Resultatet bliver en totalbeskrivelse med atributter. Det vil sige, at vi har den samlede datamodel for systemet. Denne datamodel skal bruges i systembeskrivelsen, og skal derfor kunne oversættes til DELTA.

3. KENDTE REDSKABER, DER KUNNE BRUGES VED OPKLARINGSARBEJDET.

3.1 Informationsgrafer.

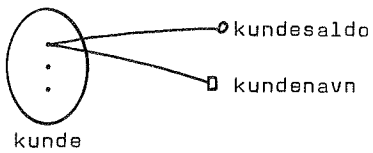
Begreber.
Ved en entitet forståes en ting, et væsen eller en hændelse, der enten fysisk eller begrebsmæssigt kan afgrænses fra det omkringliggende system. Entiteter kan navngives. En entitet karakteriseres ved et sæt af egenskaber, der enten udtrykker atributter ved en entitet eller relationer mellem entiteter. En entitetsklasse er den delmængde af entiteter i det omkringliggende system, der alle er karakteriseret ved det samme sæt af egenskaber.

I en informationsgraf angives en entitetsklasse ved en elipse med tre punkter, de symboliserer entiteterne. Desuden er der et navn på entitetsklassen:



Egenskaber udtrykkes ved buer, der udgår fra en entitetsklasse. Navnet på egenskaben angives i forbindelse med buen.

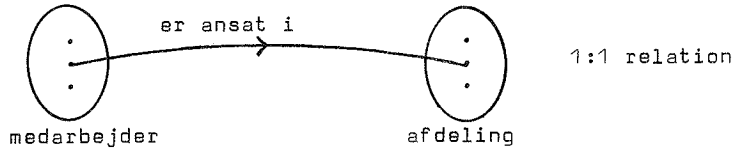
For en atribut vil buen ende i et symbol, der angiver værditypen for atributten: numerisk (o) eller alfanumerisk (□).

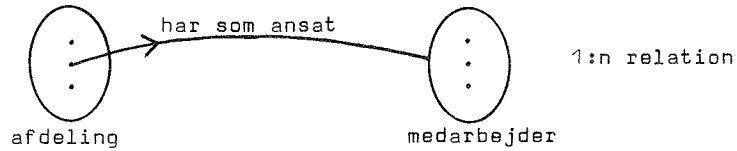


For relationer vil buen ende i en entitetsklasse, og vi skelner mellem:

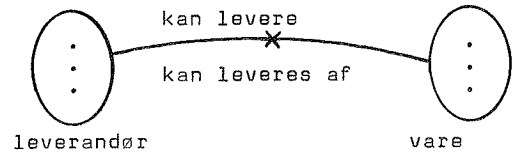
- 1:1 relationer
- 1:n relationer
- n:n relationer

En pil angiver relationens retning.





n:n relationer er sammensætningen af en 1:n relation begge veje:



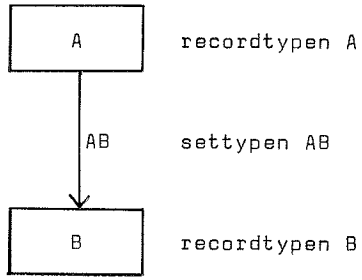
Desuden giver informationsgrafen mulighed for at angive om en egenskab identificerer entiteten, samt om en egenskab er sammensat af andre egenskaber.

3.2 Bachmann-diagrammer.

Bachmann-diagrammet er beregnet til at beskrive begreber og sammenhænge på en måde, der gør det let at lave en CODASYL-database. De muligheder og begrænsninger, der ligger CODASYL, går derfor igen i Bachmann-diagrammet.

I CODASYL arbejder man med records og sets. En record er en samling af data, der begrebsmæssigt hører sammen, og et set er en forbindelse mellem records.

En recordtype tegnes i Bachmann-diagrammet som en kasse, og en settype tegnes som en pil mellem to kasser:

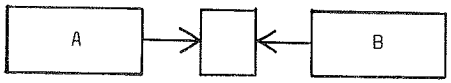


På grund af restriktioner i CODASYL, skal alle forbindelser være "en til mange" forbindelser. Det vil sige, at et set altid knytter en record af én type sammen med mange records af en anden type.

På Bachmann-diagrammet betyder pilen, at der til en record, af den type pilen udgår fra, er knyttet mange records af den recordtype, pilen går til.

Mellem to recordtyper A og B er man tit interesseret i at beskrive en forbindelse, der til hver A-record knytter mange B-records, og til hver B-record knytter mange A-records. Denne "mange til mange" forbindelse må i Bachmann-diagrammet beskrives med to sets og en ekstra recordtype.

Fx kan en "mange til mange" forbindelse udtrykkes således:



3.3 Sammenligning af Bachmann-diagrammer og informationsgrafer.

Når man sammenligner de to beskrivelsesmetoder, er det forskellene mere end lighederne, der falder i øjnene. De to beskrivelsesmåder er da også beregnet til at bruges i to forskellige situationer. Hvor Bachmann-diagrammet er beregnet til at give en oversigt over strukturer i en database, er informationsgrafen beregnet til at bruges til hjælp i analysen af et systems information og struktur. Det er derfor klart, at informationsgrafen indeholder andet end den strukturelle information, og at Bachmann-diagrammet holder sig til strukturen.

Der er dog også visse lighedspunkter. Begge steder kan man symbolisere et begreb, nemlig som en recordtype henholdsvis en entitetsklasse. Forbindelser mellem begreber kan også udtrykkes begge steder: relationer henholdsvis sets. Relationerne kan dog udtrykke en "én til én", en "én til mange" og en "mange til mange" forbindelse, medens settet kun kan udtrykke en "én til mange" forbindelse.

3.4 Vurdering af Bachmann-diagrammer og informationsgrafer til brug i analyse af datamodellen.

Til brug ved analysen af datamodellen havde vi brug for et redskab. Ved hjælp af dette redskab skulle man kunne fastholde begreber og strukturer i de partielle virkelighedsopfattelser på en overskuelig og intuitiv måde.

Man kunne tænke sig, at Bachmann-diagrammer eller informationsgrafer kunne bruges her. Med begge hjælpemidler kan man nemlig beskrive begreber

og sammenhænge.

Informationsgrafen indeholder dog alt for meget information til at en sådan graf kan være overskuelig nok til dette brug. Hvis den skal bruges, skal det være i en skrabet udgave, der kun indeholder entitetsklasser og relationer.

Bachmann-diagrammet indeholder ikke for meget information, men her er det galt med det intuitive. Det fremgår nemlig ikke umidelbart af diagrammet, at et set knytter records af forskellig type sammen, og at informationen om sammenhængen ligger i records af begge typer.

Hvis man vil have en intuitiv fornemmelse af, hvad pilen betyder, er man nødt til at tænke på den som gående fra en record af den ene type, derfra til en record af den anden type, videre til andre records af samme type for til slut at gå tilbage til den første record. Tænker man på den måde, kan man sige, at det kun er den første pil, der er tegnet - resten må man så tænke sig til.

Det er imidlertid uheldigt at tænke på forbindelserne i diagrammet som gående mellem individer (records) og ikke mellem klasser af individer (recordtyper). I denne fase af systemarbejdet er det nemlig vigtigt at tænke i klasser og ikke i individer.

Der er en fejl mere ved Bachmann-diagrammet. Det drejer sig om "mange til mange" forbindelser. Her er man nødt til at oprette en ny recordtype. Denne ekstra kasse modsvares ikke af et begreb fra virkelighedsopfattelsen, og er kun med til at gøre diagrammet mere uoverskueligt.

4. KRAV TIL CONSTRUCT VÆRKTØJET.

Det er klart, at en graf er et velegnet redskab til at fastholde strukturen i de partielle virkelighedsopfattelser.

I arbejdet med at samle disse partielle strukturer til en total struktur er grafen derimod ikke særlig hensigtsmæssig, idet den hurtigt vil blive uoverskuelig.

Vi vil derfor vælge en sproglig beskrivelse til at udtrykke den totale struktur.

Vi har altså brug for en graf og et sprog til at bruge i hver sin del af analysen og beskrivelsen af datamodellen. De skal hver for sig rumme faciliteter, der gør dem velegnede til dette arbejde, og vi vil nu se nærmere på de krav vi må stille til grafen og sproget.

Grafen.

Grafen skal indeholde begreber (komponentklasser) og forbindelser (connections) ligesom både Bachmann-diagrammet og informationsgrafen gør det.

Den skal på intuitiv måde vise hvilken slags connection det drejer sig om ("én til én", "én til mange" eller "mange til mange").

I CONSTRUCT-grafen skal det være muligt at beskrive komponentklasser og connections på en sådan måde, at grafen bliver ækvivalent med strukturen i den partielle virkelighedsopfattelse. Det vil sige, at alt i grafen skal kunne modsvares af elementer i virkelighedsopfattelsen.

CONSTRUCT-grafen skal med andre ord indeholde:

- Komponentklasser med navn.
- Connections med navn.
- Specifikation af connectiontypen.

Sproget.
Ved overgang til en sproglig beskrivelse tabes den intuitivitet grafen indeholder. Til gengæld opnåes mulighed for at beskrive mere komplicerede strukturer og stadig bevare overskueligheden.

Ved overgang til sproglig beskrivelse allerede ved de partielle strukturer, opnåes yderligere den fordel, at man ved hjælp af en tekstanalysator kan få en datamat til at hjælpe i arbejdet med konsistenskontrol og udformning af den totale struktur.

I den sproglige beskrivelse kan vi, stadig uden tab af overskueligheden, påføre atributter på komponentklasser i strukturen, og derigennem beskrive komponentklassernes opdeling i underklasser. Dermed fås en beskrivelse af den totale datamodel, hvori de partielle datamodeller kan genfindes.

CONSTRUCT-sproget skal derfor opfylde følgende krav:

- Man skal direkte kunne udtrykke informationen fra CONSTRUCT-grafen.
- Det skal være muligt i forskellige detaljeringsgrader at beskrive de attributter, som ligger i komponenter og connections.
- Underklasser af komponentklasser skal kunne udtrykkes.
- Connections skal kunne beskrives eksplicit, hvad angår atributter, connectiontype og hvilke komponent- eller underklasser de forbinder.

Desuden skal CONSTRUCT-beskrivelsen af datamodellen danne grundlag for DELTA-systembeskrivelsen. CONSTRUCT-beskrivelsen skal altså kunne oversættes til DELTA.

5. CONSTRUCT-GRAFEN.

5.1 komponentklasser.

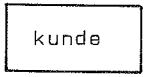
En komponentklasse skal i CONSTRUCT-grafen kunne indeholde navnet på komponentklassen. Begrebet komponentklasse modsvares i informationsgrafen af entitetsklasse og i Bachmann-diagrammet af records.

Symbolet for en entitetsklasse er uheldigt, fordi man i CONSTRUCT-grafen ikke har behov for eksplicit at udtrykke, at en komponentklasse består af enkeltindivider til hvilke der kan knyttes atributter.

De 3 punkter i informationsgrafens elipse er derfor overflødige til dette formål. Endelig vil vi i CONSTRUCT-grafen kunne hæfte et navn på komponentklasserne. Dette navn er det nu naturligt at placere inde i symbolet for komponentklassen, hvorved de 3 punkter i informationsgrafen kommer direkte i vejen.

Her finder vi Bachmann-diagrammets recordsymbol - kassen - langt mere hensigtsmæssigt at benytte.

En komponentklasse udtrykkes derfor i CONSTRUCT-grafen ved en kasse med komponentklassens navn inden i:



5.2 Connections.

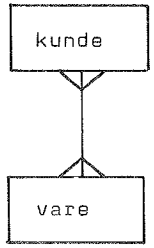
En connection skal i CONSTRUCT-grafen kunne forbinde 2 komponentklasser, udtrykke hvilken type forbindelse det drejer sig om, og endelig skal den kunne navngives.

I Bachmann-diagrammet modsvares connections af sets symboliseret ved pile. Vi har tidligere redegjort for hvorfor pile-notationen er uheldig.

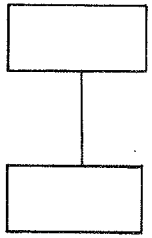
Man kunne forestille sig, at erstatte pilene med "gafler", og så tegne connections:

Dette kan umidelbart tolkes som, at hvert individ fra kunde-komponentklassen har tilknytning til mange individer fra vare-komponentklassen.

I Bachmann-diagrammet udtrykkes "mange til mange" forbindelsen klodset. Dette kunne umidelbart løses ved, at tillade "gafler" i begge ender af forbindelsen. Således:



Tilbage er "én til én" forbindelserne. Disse kunne udtrykkes som en forbindelseslinie uden "gafler". Således:



Dette giver dog problemer. I analysearbejdet vil det være formålstjenligt at kunne tegne en connection af flere omgange: Først blot tegne en forbindelseslinie mellem 2 komponenter for at angive, at der er en forbindelse imellem dem (uspecificeret connection) og dernæst betragte dem en af gangen og se,om hver komponent har tilknytnytning til én eller mange individer i den modsatte klasse.

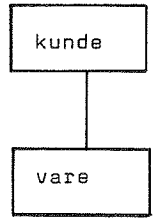
Med "gaffel" notationen kan man imidlertid ikke se, om det drejer sig om en "én til én" connection eller om en uspecificeret connection. Man kan heller ikke se, om det drejer sig om en "én til mange" connection eller en "mange til mange" connection, som blot ikke er tegnet færdig.

En anden ubehagelig ting ved "gaffel" notationen er, at det ved en connection mellem komponentklasserne A og B, er ved A-klassen det er specificeret, om et individ fra B-klassen har forbindelse til ét eller flere individer fra A-klassen. Denne oplysning hører naturligt hjemme i B-klassen.

Vi må derfor forkaste "gaffel" notationen som en egnet måde at symbolisere connections i CONSTRUCT-grafen

Vi vælger i stedet en notation, der har alle "gaffel" notationens fordele og samtidig imødegår de to uheldige egenskaber ved "gaffel" notationen.

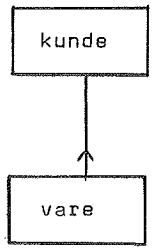
I CONSTRUCT-grafen tegnes en uspecificeret connection således:



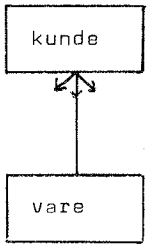
Altså blot en forbindelseslinie mellem 2 komponentklasser.

Ved nøjere specifikation af forbindelsen kan man fra hver af enderne specificere, om den går til én eller til mange.

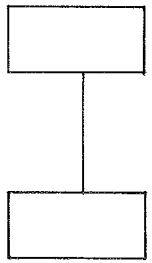
Hvis der til hvert vare-individ er knyttet én kunde, tegnes således:



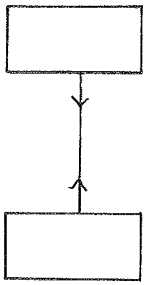
Hvis der til hvert kunde-individ er knyttet mange vare-individer, tegnes således:



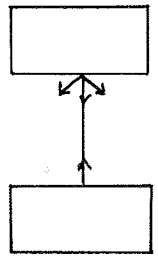
Vi har nu opnået mulighed for grafisk at udtrykke, om en connection mellem komponentklasser er uspecificeret:



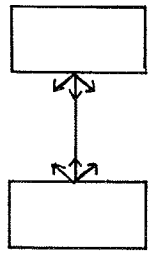
én til én:



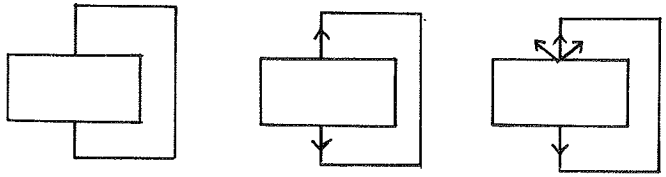
én til mange:

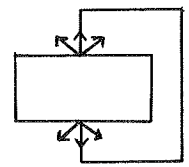


eller mange til mange:



En connection kan også udtrykke forbindelse mellem individer fra samme komponentklasse:



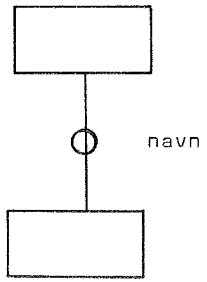


Vi mangler nu blot at kunne knytte et navn til forbindelsen.

I Bachmann-diagrammet gøres dette ved at skrive navnet på eller tæt ved forbindelseslinier.

Dette har den ulempe, at der ved lange forbindelseslinier eller forbindelseslinier, der ligger tæt, kan være svært at afgøre hvilket navn, der hører til hvilken connection.

I CONSTRUCT-grafen tegnes en cirkel på forbindelseslinien, og ud for cirklen skrives navnet. Således:



I CONSTRUCT-grafen er det nu muligt at udtrykke:

Komponentklasser med navn,
connections med navn og
specifikation af connections.

6. CONSTRUCT-SPROGET.

6.1 CONSTRUCT-syntaks.

En CONSTRUCT-beskrivelse skulle kunne oversættes til DELTA, derfor har vi valgt at lade CONSTRUCT-sproget bygge på DELTA, og så lave de nødvendige udvidelser og restriktioner.

Man skulle i sproget direkte kunne udtrykke informationen fra CONSTRUCT-grafen. Her findes to forskellige slags objektklasser, nemlig komponenter og connections. De beskrives som underklasser til klasserne COMPONENT og CONNECTION. Dette kan gøres direkte i DELTA og får udseendet:

```
CLASS VARE: COMPONENT OBJECT;
```

og

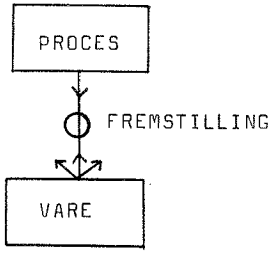
```
CLASS FREMSTILLING: CONNECTION OBJECT;
```

Grafen indeholdt dog også information om, at en connection sammenbinder to komponenter. Den information skal i hvert fald ligge i connection-klassen, men den bør også være tilstede i component-klasserne, idet den siger noget om en komponents omgivelser.

I komponent-klassen skal der specificeres:

Navnet på connection-klassen.
Navnet på den komponent-klasse, forbindelsen går til.
Om forbindelsen går til én eller mange.

Desuden er det nødvendigt at give et navn til forbindelsen betragtet fra denne side. Ellers kan man ikke i den senere DELTA-beskrivelse henvise til forbindelsen.



Lad os betragte denne simple CONSTRUCT-graf, hvor komponenterne PROCES og VARE er forbundet med en connection FREMSTILLING. Det er en "en til mange" connection, fordi hver proces fremstiller én vare, og hver vare kan fremstilles af flere processer.

I PROCES komponent-klassen vil vi beskrive forbindelsen således:

```
CLASS PROCES:
   COMPONENT OBJECT BEGIN
   FREMSTILLER: ONE FREMSTILLING CONNECTION TO VARE
END OBJECT;
```

I VARE komponent-klassen vil samme forbindelse få udseendet:

```
CLASS VARE:
   COMPONENT OBJECT BEGIN
   FREMSTILLES AF: MANY FREMSTILLING CONNECTION TO PROCES
END OBJECT;
```

Vi har nu i hver af komponent-klasserne beskrevet en forbindelse ved: et navn, om den går til én eller mange, connection-navnet og navnet på den komponent, forbindelsen går til.

I connection-klassen skulle de samme oplysninger være til stede. Det beskriver vi således:

```
CLASS FREMSTILLING:
   CONNECTION OBJECT BEGIN
   ONE PROCES TO ONE VARE THROUGH FREMSTILLER;
   ONE VARE TO MANY PROCES THROUGH FREMSTILLES AF
END OBJECT;
```

Vi har nu mulighed for at udtrykke al informationen fra grafen i sproget, og mangler kun atributter og underklasser. Begge dele kan udtrykkes i DELTA, og vi tager notationen derfra.

Vil man fx beskrive, at INDKØBT VARE er en underklasse af VARE, og at den har atributterne HOVEDLAGER og MINIMAL LAGERTID, sker det således:

```
CLASS INDKØBTE VARE:
   VARE OBJECT BEGIN
   HOVEDLAGER: TEXT;
   MINIMAL LAGERTID: INTEGER
END OBJECT;
```

I objectkroppe tillader vi kun connectionspecifikation og simple atributter.

I connection tillader vi altså specifikation af atributter. Men underklasser af connection-klasser har ingen mening, og er derfor ikke tilladt. Derimød kan connections godt forbinde underklasser af komponenter.

6.4 Formel syntaksbeskrivelse af CONSTRUCT-sproget.

Den samlede syntaks for CONSTRUCT-sproget kommer til at se således ud (metasproget er taget fra DELTA-rapporten):

```
<construct description> is
MODEL BEGIN
  list { <component class definition>
  or <connection class definition> } sep { ; }
END MODEL
<component class definition> is
CLASS <component name> : <prefix part> <component object>

<component name> is <identifier>

<prefix part> is COMPONENT or <component name>

<component object> is
OBJECT
or OBJECT BEGIN
opt { list <connection reference specification> sep {;} }
opt { ; list <atribut specification> sep {;} }
END OBJECT

<connection reference specification> is
<reference name>: { ONE or MANY } <connection name>
  CONNECTION TO <component name>

<atribut specification> is
<atribut name> : <type>

<atribut name> is <identifier>

<type> is TEXT
       or INTEGER
       or REAL
```



```
<connection class definition> is
CLASS <connection name>: CONNECTION <connection object>

<connection name> is <identifier>

<connection object> is
OBJECT BEGIN
      <connection specification>;
      opt { < connection  specification> ; }
      opt { list <atribut specification> sep { ; } }
END OBJECT

<connection specification> is
ONE <component name> TO { ONE or MANY }
   <component name> THROUGH <reference name>
```

6.3 CONSTRUCT-oversætter.

CONSTRUCT-beskrivelsen skulle kunne oversættes til DELTA. Vi vil ikke beskrive oversættelsen i detaljer, men kun skitsere, hvordan det kan lade sig gøre.

I DELTA preerklærer vi to klasser:

```
CLASS COMPONENT: OBJECT;

CLASS CONNECTION:
OBJECT BEGIN
  VIRTUEL REF1, REF2: REF COMPONENT CONSTANT;
  VIRTUEL REF1MANY, REF2MANY: BOOLEAN CONSTANT;
END OBJECT;
```

Vi har her tilladt os at udvide DELTA med virtuelle atributter. Betydningen er, at de virtuelle atributter REF1, REF2, REF1MANY og REF2MANY skal specificeres nøjere i hver underklasse. Det er altså en parallel til virtuelle procedurer i DELTA.

Komponentklassen VARE fra før bliver så oversat til:

```
CLASS VARE:
COMPONENT OBJECT BEGIN
  FREMSTILLES AF: LIST REF FREMSTILLING;
END OBJECT;
```

Her er vi nødt til at udvide DELTA med muligheden for en liste. Men da DELTA ingen datastrukturer har, tillader vi os denne frihed.

Connection-klassen FREMSTILLING fra før bliver oversat til:

```
CLASS FREMSTILLING:
CONNECTION OBJECT BEGIN
  REF1: REF VARE CONSTANT;
  REF2: REF PROCES CONSTANT;
  REF1MANY: FALSE;
  REF2MANY: TRUE;
END OBJECT;
```

Atributter og underklasser fra CONSTRUCT-beskrivelsen kan oversættes direkte til DELTA.

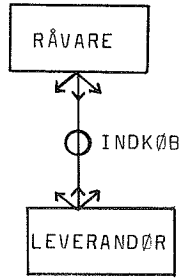
7. BRUG AF SPROGET.

I dette afsnit beskrives, hvorledes man ved anvendelse af CONSTRUCT-sprogets faciliteter kommer fra de partielle strukturer, beskrevet i CONSTRUCT-grafen, til en total struktur beskrevet i CONSTRUCT-sproget.

De CONSTRUCT-grafer, der beskriver de partielle strukturer, indeholder hver for sig information om, hvorledes begreber og delbegreber knyttet til virkeligheden hænger sammen i virkelighedsopfattelsen.

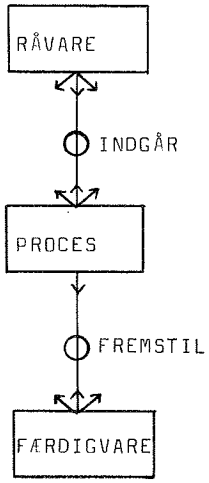
Det er vigtigt at gøre sig klart, hvilke af kasserne, der symboliserer delbegreber. Hvis en kasse symboliserer et delbegreb er det vigtigt, på dette sted, at fastlægge, hvilket begreb den symboliserer et delbegreb af.

Graf nr. 1



I graf nr. 1 indgår delbegreberne RÅVARE, som er et delbegreb af begrebet VARE.

Graf nr. 2



I graf nr. 2 indgår delbegreberne RÅVARE og FÆRDIGVARE, som begge er delbegreber af begrebet VARE.

Disse 2 partille CONSTRUCT-grafer beskrives nu i CONSTRUCT-sproget:

Struktur nr. 1

```
CLASS RÅVARE:
   VARE OBJECT BEGIN
   LEVERINGSMULIGHED: MANY INDKØB CONNECTION TO LEVERANDØR;
   END OBJECT;

CLASS LEVERANDØR:
   COMPONENT OBJECT BEGIN
   KANLEVERE: MANY INDKØB CONNECTION TO RÅVARE;
   END OBJECT;
```


Struktur nr. 2

```
CLASS RÅVARE:
   VARE OBJECT BEGIN
   INDGÅRILISTE: MANY INDGÅR CONNECTION TO PROCES;
   END OBJECT;

CLASS PROCES:
   COMPONENT OBJECT BEGIN
   STYKLISTE: MANY INDGÅR CONNECTION TO RÅVARE;
   FREMSTILLER: ONE FREMSTIL CONNECTION TO FÆRDIGVARE;
   END OBJECT;

CLASS FÆRDIGVARE:
   VARE OBJECT BEGIN
   FREMSTILMULIGHED: MANY FREMSTIL CONNECTION TO PROCES;
   END OBJECT;
```

Denne beskrivelse i CONSTRUCT-sproget indeholder samme information som CONSTRUCT-graferne, men det er stadig ingen CONSTRUCT-beskrivelse. Her mangler eksplicit erklæring af connections og komponentklassen VARE.

Eftersom de to partielle beskrivelser indeholder den information, der skal til for at gøre beskrivelsen til en CONSTRUCT-beskrivelse, kan dette foregå automatisk ved brug af en CONSTRUCT-tekstanalysator og en datamat.

Tekstanalysatoren vil samtidig kunne lave en total

beskrivelse ud fra de to partielle beskrivelser, under forudsætning af at de to partielle beskrivelser er indbyrdes konsistente.

Denne kontrol af indbyrdes konsistens kan også laves af tekstanalysatoren.

Derfor underkaster vi nu de to partielle beskrivelser en CONSTRUCT tekstanalyse, og får kontrolleret konsistensen.

Hvis konsistensen er i orden, resulterer tekstanalysen i en total CONSTRUCT-beskrivelse, hvori de 2 partielle beskrivelser kan genfindes.

Hvis de 2 partielle beskrivelser ikke er indbyrdes konsistente, kan man ikke lave den totale CONSTRUCT-beskrivelse, men må først tilbage i analysen og gøre de to partielle beskrivelser indbyrdes konsistente.

Efter tekstanalysen af de 2 partielle beskrivelser fås nu følgende totale CONSTRUCT-beskrivelse:

```
MODEL BEGIN

CLASS VARE:
   COMPONENT OBJECT;

CLASS RÅVARE:
   VARE OBJECT BEGIN
   LEVERINGSMULIGHED: MANY INDKØB CONNECTION TO LEVERANDØR;
   INDGÅRILISTE: MANY INDGÅR CONNECTION TO PROCES;
   END OBJECT;
```



```
CLASS FÆRDIGVARE:
   VARE OBJECT BEGIN
   FREMSTILMULIGHED: MANY FREMSTIL CONNECTION TO PROCES;
   END OBJECT;

CLASS LEVERANDØR:
   COMPONENT OBJECT BEGIN
   KANLEVERE: MANY INDKØB CONNECTION TO RÅVARE;
   END OBJECT;

CLASS PROCES:
   COMPONENT OBJECT BEGIN
   STYKLISTE: MANY INDGÅR CONNECTION TO RÅVARE;
   FREMSTILLER: ONE FREMSTIL CONNECTION TO FÆRDIGVARE;
   END OBJECT;

CLASS INDGÅR:
   CONNECTION OBJECT BEGIN
   ONE RÅVARE TO MANY PROCES THROUGH INDGÅRILISTE;
   ONE PROCES TO MANY RÅVARE THROUGH STYKLISTE;
   END OBJECT;

CLASS FREMSTIL:
   CONNECTION OBJECT BEGIN
   ONE PROCES TO ONE FÆRDIGVARE THROUGH FREMSTILLER;
   ONE FÆRDIGVARE TO MANY PROCES THROUGH FREMSTILMULIGHED;
   END OBJECT;

CLASS INDKØB:
   CONNECTION OBJECT BEGIN
   ONE RÅVARE TO MANY LEVERANDØR THROUGH LEVERINGSMULIGHED;
   ONE LEVERANDØR TO MANY RÅVARE THROUGH KANLEVERE;
   END OBJECT;

END MODEL;
```

For at opnå en CONSTRUCT-beskrivelse af den totale datamodel, mangler vi nu kun at beskrive de attributter, som benyttes i datamodellen.

I de partielle virkelighedsopfattelser benyttes de attributter knyttet til et begreb eller delbegreb, som er nødvendige for at kunne operere. Alle disse attributter skal kunne genfindes i den totale datamodel, og således at de knytter sig til de samme begreber eller delbegreber, de knyttede sig til i den partielle datamodel.

Struktur nr. 1.

I struktur nr. 1 benyttes VARENUMMER, VARENAVN og LAGERBEHOLDNING i tilknytning til klassen RÅVARE.

I klassen LEVERANDØR benyttes NAVN, ADRESSE og KREDITTID som attributter.

Når der er tale om én bestemt RÅVARE i tilknytning til én bestemt LEVERANDØR, benyttes også oplysninger om PRIS og LEVERINGSTID.

Struktur nr. 2.

I struktur nr. 2 knytter attributterne VARENUMMER, VARENAVN og LAGERLOKATION sig til klassen RÅVARE.

Til klassen PROCES knyttes PROCESNUMMER og PROCESNAVN samt processens VARIGHED.

Når der er tale om én bestemt RÅVARE i forbindelse med én bestemt PROCES, benyttes oplysning om MÆNGDE.

Til FÆRDIGVARE knyttes VARENUMMER og VARENAVN. I forbindelsen mellem én bestemt PROCES og den FÆRDIGVARE den fremstiller, knyttes oplysning om ANTAL.

I CONSTRUCT-beskrivelsen oplyses i forbindelse med attributnavnet også hvilken type attribut, der er tale om.

Den færdige CONSTRUCT-beskrivelse af den totale datamodel ser nu således ud:

```
MODEL BEGIN

CLASS VARE:
   COMPONENT  OBJECT BEGIN
   VARENUMMER: INTEGER;
   VARETEKST: TEXT;
   END OBJECT;

CLASS RÅVARE:
   VARE OBJECT BEGIN
   LEVERINGSMULIGHED: MANY INDKØB CONNECTION TO LEVERANDØR;
   INDGÅRILISTE: MANY INDGÅR CONNECTION TO PROCES;
   LAGERBEHOLDNING: INTEGER;
   LAGERLOKATION: INTEGER;
   END OBJECT;
```

```
CLASS FÆRDIGVARE:
   VARE OBJECT BEGIN
   FREMSTILMULIGHED: MANY FREMSTIL CONNECTION TO PROCES;
   END OBJECT;


CLASS LEVERANDØR:
   COMPONENT OBJECT BEGIN
   KANLEVERE: MANY INDKØB CONNECTION TO RÅVARE;
   NAVN: TEXT;
   ADRESSE: TEXT;
   KREDITTID: INTEGER;
   END OBJECT;


CLASS PROCES:
   COMPONENT OBJECT BEGIN
   STYKLISTE: MANY INDGÅR CONNECTION TO RÅVARE;
   FREMSTILLER: ONE FREMSTIL CONNECTION TO FÆRDIGVARE;
   PROCESNUMMER: INTEGER;
   PROCESNAVN: TEXT;
   VARIGHED: INTEGER;
   END OBJECT;


CLASS INDKØB:
   CONNECTION OBJECT BEGIN
   ONE LEVERANDØR TO MANY RÅVARE THROUGH KANLEVERE;
   ONE RÅVARE TO MANY LEVERANDØR THROUGH LEVERINGSMULIGHED;
   PRIS: INTEGER;
   LEVERINGSTID: INTEGER;
   END OBJECT;


CLASS INDGÅR:
   CONNECTION OBJECT BEGIN
   ONE RÅVARE TO MANY PROCES THROUGH INDGÅRILISTE;
   ONE PROCES TO MANY RÅVARE THROUGH STYKLISTE;
   MÆNGDE: INTEGER;
   END OBJECT;
```



```
CLASS FREMSTIL:
   CONNECTION OBJECT BEGIN
   ONE PROCES TO ONE FÆRDIGVARE THROUGH FREMSTILLER;
   ONE FÆRDIGVARE TO MANY PROCES THROUGH FREMSTILMULIGHED;
   ANTAL: INTEGER;
   END OBJECT;


END MODEL
```

Resumé.

For at opnå en CONSTRUCT-beskrivelse af den totale datamodel med udgangspunkt i de partielle CONSTRUCT-grafer, gøres følgende:

- Oversæt CONSTRUCT-graferne til en beskrivelse af graferne i CONSTRUCT-sproget, således at den sproglige beskrivelse er ækvivalent med graferne.

- Analysér de partielle strukturbeskrivelser, eventuelt ved brug af en CONSTRUCT-tekstanalysator.

  Denne analyse resulterer i en total strukturbeskrivelse, hvis de partielle beskrivelser er indbyrdes konsistente. Hvis de ikke er indbyrdes konsistente, gås tilbage i analysen, hvor de partielle beskrivelser rettes, så de bliver indbyrdes konsistente.

- På den totale struktur indføjes nu attributterne fra de partielle datamodeller, således at disse kan genfindes i den resulterende CONSTRUCT-beskrivelse af den totale datamodel.

LITTERATURLISTE.

Introduktion til Informationsanalyse
Paul Lindgren
Regnecentralen a/s juni 1973

CODASYL june 1973 Report
Data Description Language Committee

System Description and the DELTA Language
Erik Holbæk-Hanssen
Petter Håndlykken
Kristen Nygaard
Norsk Regnesentral september 1975