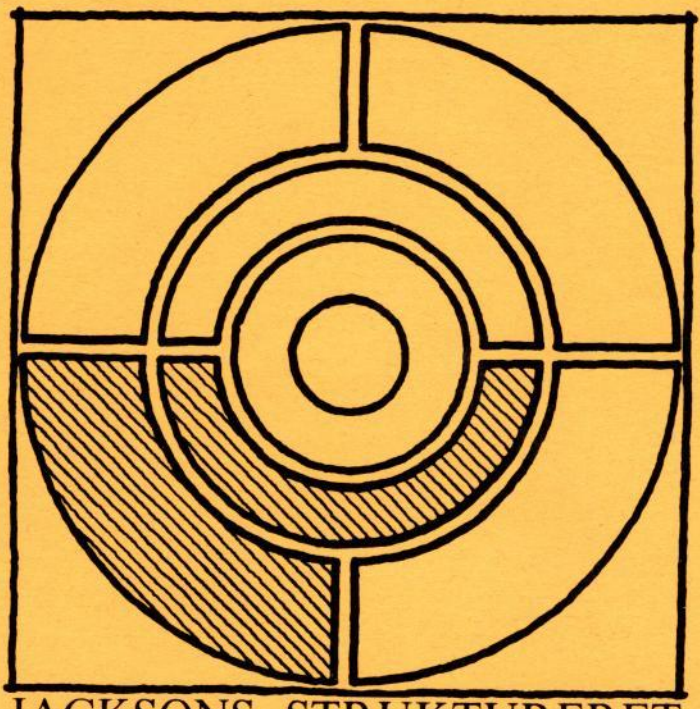
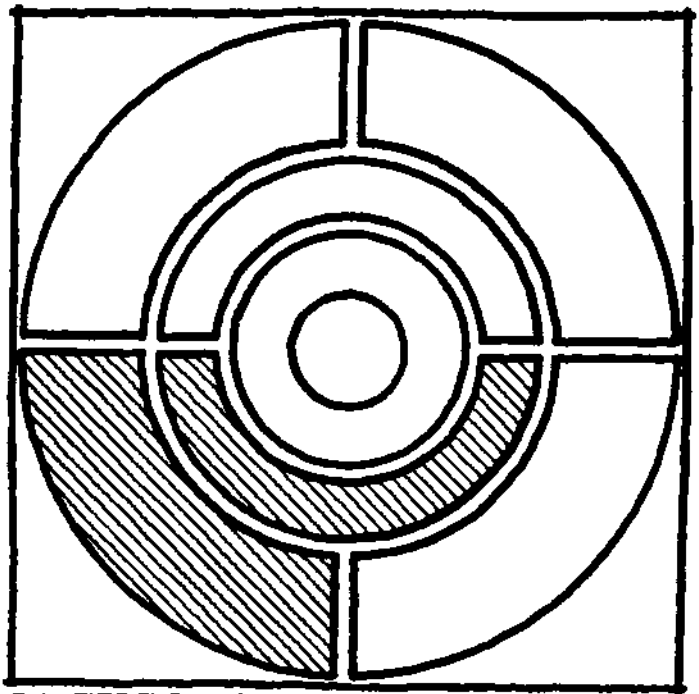
Metodesektionen

ARBEJDSKOPI



JACKSONS STRUKTURERET PROGRAMMERING

Metodesektionen



JACKSONS STRUKTURERET PROGRAMMERING

Jacksons Struktureret Programmering

Første udgave juni 1989

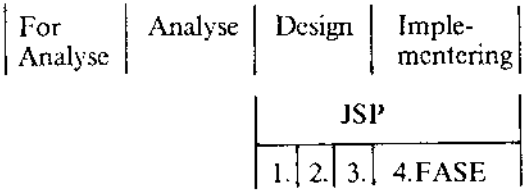
udgiver og vedligeholder:

Metodesektionen i Teknikstaben

fordelingsnøgle: **MHJSP**

JACKSONS STRUKTURERET PROGRAMMERING

JSP anvendes til design og kodning:



FASE 1-3

JSP-Design bygger på følgende principper:

- Programmer er en kombination af strukturer, der kun har en ind- og en udgang (f.eks. if - else - endif)
- Der anvendes kun følgende tre grundstrukturer (sekvens, selektion og iteration)
- Hop "(go to)" anvendes ikke
- Statuskoder (switches) anvendes så lidt som muligt

FASE 4

Ved Implementering tilpasses et JSP-Design til den/det maskine/sprog der ønskes benyttet, uden at ændre i designets struktur.

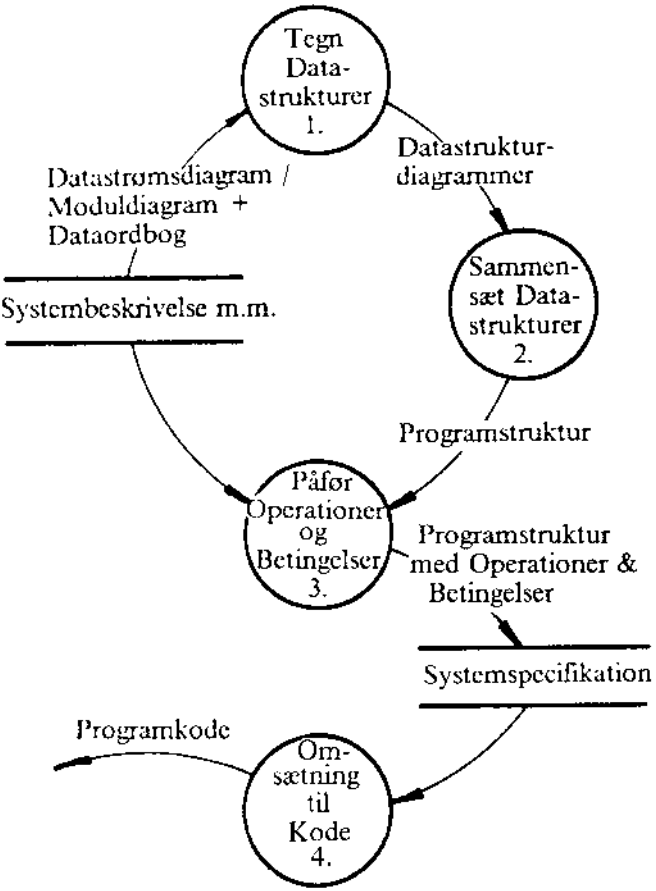
LITTERATUR

Materiale fra kursus i JSP
Leif Ingevaldson 'JSP i batch-, on-line- och DB-miljø
Michael Jackson 'Principles of program Design'
Bo Sanden 'Systemprogrammering med JSP'

JACKSONS STRUKTURERET PROGRAMMERING

Jacksons Struktureret Programmering (JSP) består af 4 Faser, der altid udføres i samme rækkefølge. Og hver fase resulterer i et slutprodukt, der anvendes i næste fase.

DE 4 FASER



I det følgende vil Design og Implementering blive behandlet hver for sig.

JSP GRUNDSTRUKTUR

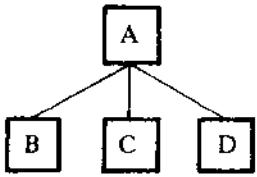
NOTATION

Princippet i beskrivelsen hviler på 3 grundlæggende strukturer, af hvilke alle tilstande og aktiviteter kan opbygges. I en struktur er alle elementer af samme type.

Disse strukturer er:

SEKVENS

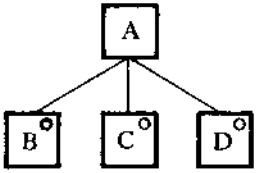
Dataelementerne kommer i en bestemt rækkefølge.



Dette læses som : A består af B,C og D i nævnte rækkefølge. De er alle tilstede altid

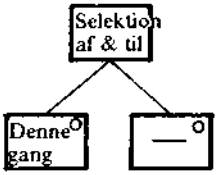
SELEKTION

Vælges een og kun een af 2 el. flere muligheder.



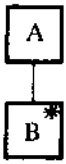
Dette læses som : A består af enten B, C, eller D. Der findes ikke flere alternativer.

Ind imellem har man brug for at beskrive et evt. manglende data.



ITERATION

Gentages noget 0,1 eller flere gange.



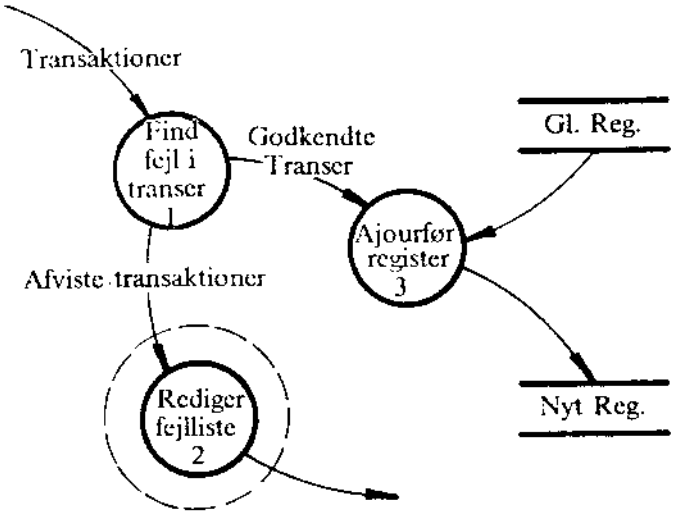
Dette læses som: A består af et antal B'er.

Grundstrukturen består af to niveauer. Det øverste niveau begrebet, med elementerne på det underliggende niveau. Hvert element kan være en struktur i sig selv.

D E S I G N (Datastruktur)

1. FASE

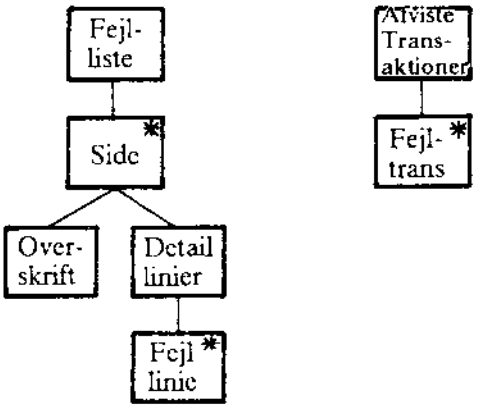
Tag udgangspunkt i et datastrømsdiagram. Man kan også tage udgangspunkt i et moduldiagram.



PROCES 2

I det følgende gennemgås de 4 faser på processen rediger fejlliste.

Tegn datastruktur over hver datastrøm (for sig). Diagrammerne skal vise den tidsmæssige rækkefølge, som data bliver anvendt eller produceret i.



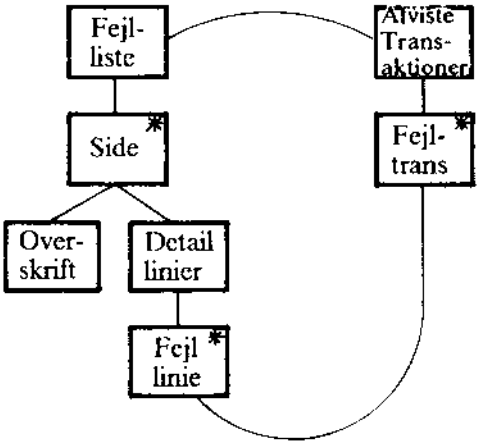
D E S I G N (Programstruktur)

2. FASE

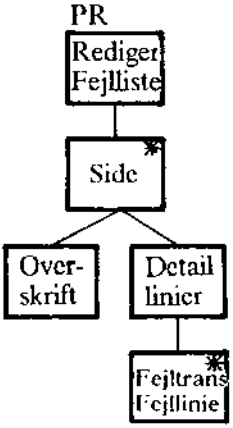
Sammensæt datastrukturerne, så de danner en programstruktur. Det sker ved at kombinere korresponderende kasser i datastrukturerne.

KORRESPON-
DANCER

Korrespondance vil sige, at der i 2 kasser er det samme antal forekomster af data, og at disse findes i samme rækkefølge.



SAMMENSAT
STRUKTUR



Der skal tegnes datastrukturer over alle de datastrømme, der har betydning for programmets struktur.

Man kan undlade at tegne datastrukturer over:
- registre, der accesser direkte.
- enkeltdata.

Hvis 2 datastrømme er identiske eller den ene er helt indeholdt i den anden, kan man se bort fra den ene.

D E S I G N (Operationer og Betingelser)

3. FASE

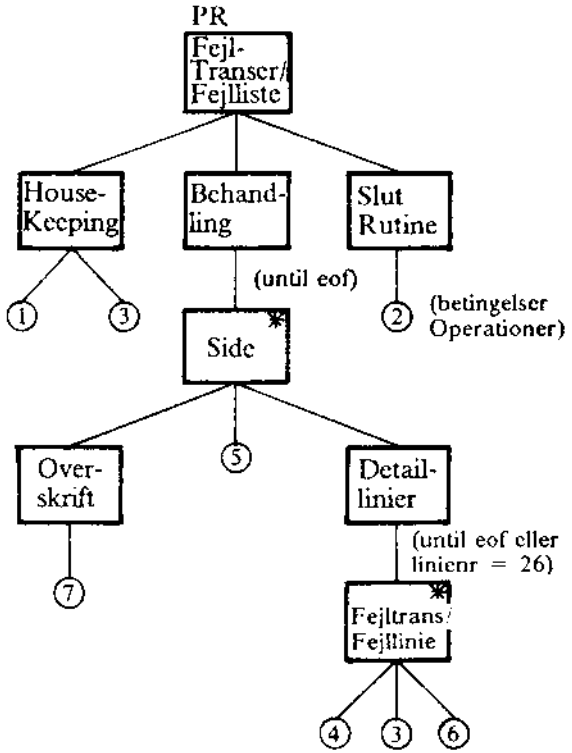
Når programstrukturen er færdig, skal den påføres de nødvendige operationer og betingelser.

- Operationer listes.
- Operationer placeres i programstrukturen, enten under eksisterende kasser, eller ved at udvide programstrukturen.
- Betingelser påføres programstrukturen.

OPERATIONSLISTE

1 Åbne alle filer
2 lukke alle filer
3 læs fejltrans
4 skriv fejltrans på fejlliste
5 Sæt linienr = 1
6 Læg 1 til linienr
7 Skriv overskrift

PROGRAM-STRUKTUR



DESIGN (Matching)

Den grundlæggende fremgangsmåde (på de foregående sider) er ofte ikke tilstrækkelig. JSP beskriver 3 specielle problemer: Matching, Strukturkonflikter og Erkendelsesproblem.

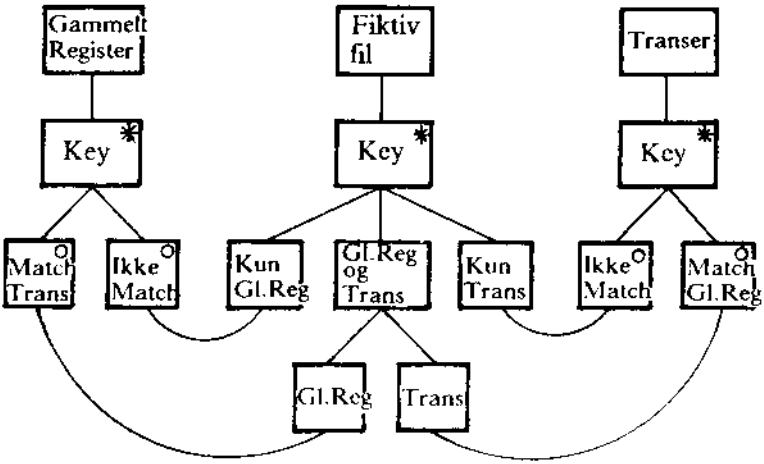
MATCHING

Man skal bruge Match - Teknik, hvis forekomster i to kasser er i samme rækkefølge, men der ikke er lige mange af hver.

Løsningen er i så fald en bagved liggende generel struktur: Foreningsmængden af de to strukturer.

Foreningsmængde strukturen bruges som programstruktur ved behandling af de to strukturer.

I match tilfælde bruges korrespondance reglerne til at kontrollere, om foreningsmængde strukturen indeholder alle elementer i de to delmængde strukturer.



Typisk anvendelsesområde:

- Batch ajourføring af serielle filer.

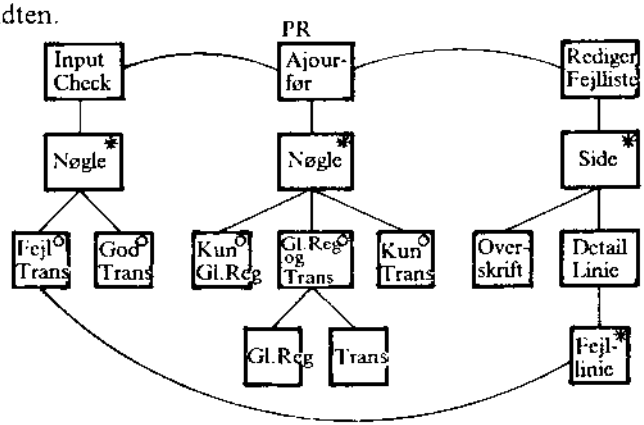
STRUKTUR-KONFLIKTER

Strukturkonflikter opstår når datastrukturer med kasser, hvori der er lige mange forekomster, alligevel ikke kan kombineres. Fordi de ikke kommer i samme rækkefølge i midten og evt. heller ikke i bunden.

Der findes 3 typer:

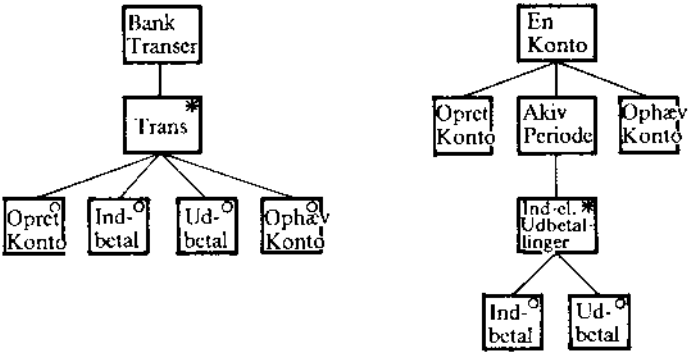
BOUNDARY CLASH

Strukturerne korresponderer i toppen og i bunden, men ikke i midten.



INTERLEAVING CLASH

Strukturerne korresponderer kun i toppen, men i bunden kommer kasserne i rigtig rækkefølge, hvis de opdeles i selvstændige strukturer (multi-threading).



SORTERINGS-KONFLIKT

To strukturer består af de samme data, men den ene må sorteres for at kunne kombineres med den anden.

Bemærk at boundary og interleaving clash indebærer at strukturen ikke kan kombineres dvs. de forbliver selvstændige med et behov for at kommunikere de korresponderende elementer (f.eks. fejltrans).

D E S I G N (Erkendelsesproblem)

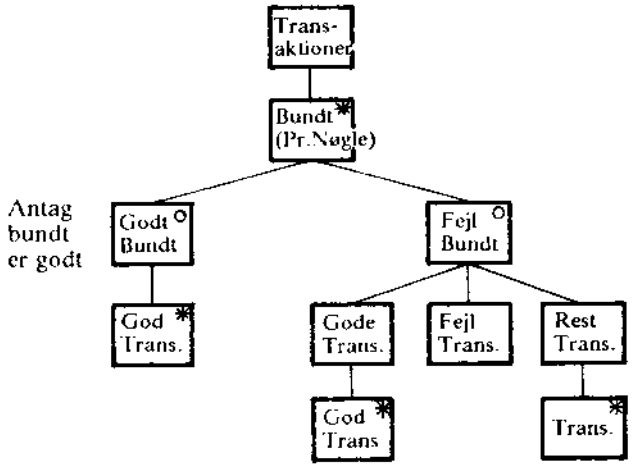
ERKENDELSES-PROBLEM

Man bliver nødt til at gøre en antagelse og så gardere sig imod konsekvenserne af antagelsen, hvis man senere kan se at antagelsen var forkert. Derfor kaldes løsning på erkendelsesproblemet backtracking: Der er fulgt en forkert sti (track) og man følger nu stien tilbage (back) til det sted, hvor den delte sig.

DE 4 TRIN

Backtracking foretages i 4 trin:

1. Lad som om valget kan foretages, når der skrives betingelser på strukturen.
2. SEL og ALT ændres til posit og admit, senest når man koder.
3. Indsæt QUIT i posit-delen, de steder hvor du kan konstatere, at antagelsen var forkert.
4. Find bivirkninger af antagelsen:
   - uacceptabel tag højde for dem
   - neutral lad dem være
   - udbytterig spar overflødig kode.



Backtracking kan næsten tilsvarende foretages i en iteration, hvor antagelsen er, at iterationen gennemløbes endnu en gang.

Typiske anvendelsesområder for backtracking:

- online dialoger, hvor bruger skal have fortrydelsesret
- batchkørsler hvor et register kun må ajourføres hvis et andet register også må (kaldet rollback/commit m.m.).

4. FASE

JSP's 4. fase går ud på at implementere programstrukturen i et sprog.

Der er to fremgangsmåder:

1. Sproget understøtter de 3 grundstrukturer og programstrukturen kan kodes direkte:

```
PROGRAM
   OPEN - - -
   READ FEJLTRANS
   LOOP UNTIL E-O-FEJLTRANS
      WRITE OVERSKRIFT
      LINIENR = 1
      LOOP UNTIL (E-O-FEJLTRANS OR
                  LINIENR = 26)
         IF FEJLTYPE = 1
            WRITE FEJLLINIE1 FROM FEJLTRANS1
         ELSE
            WRITE FEJLLINIE2 FROM FEJLTRANS2
         END IF
         LINIENR + 1
         READ FEJLTRANS
      END LOOP
   END LOOP
   CLOSE - - -
END PROGRAM
```

2. Sproget understøtter ikke eller kun delvis de 3 grundstrukturer. I så fald skrives først strukturtekst, der bagefter omsættes til nestfree kode:

STRUKTURTEKST:

```
PROCES FEJLTRANSER FEJLLISTE SEQ
   ÅBN - - -
   LÆS FEJLTRANS
   SIDER ITR TIL E-O-FEJLTRANS
      SKRIV OVERSKRIFT
      LINIENR = 1
      DETAILLINIER ITR TIL (E-O-FEJLTRANS OR
                            LINIENR = 26)
         DETAILLINIE SEL, HVIS FEJLTYPE = 1
            WRITE FEJLLINIE1 FROM FEJLTRANS1
         DETAILLINIE ALT
            WRITE FEJLLINIE2 FROM FEJLTRANS2
         DETAILLINIE END
         LINIENR + 1
         LÆS FEJLTRANS
      DETAILLINIER END
   SIDER END
   LUK - - -
PROCES FEJLTRANSER FEJLLISTE END
```

NESTFREE KODE:

```
PROCES-FEJLTRANSER-FEJLLISTE SEQ.
   OPEN - - -
   READ FEJLTRANS
SIDER-ITR.
      IF E-O-FEJLTRANS
         GO TO SIDER-END.
      WRITE OVERSKRIFT
      LINIENR = 1
DETAILLINIER-ITR.
         IF (E-O-FEJLTRANS OR
               LINIENR = 26)
            GO TO DETAILLINIER-END.
DETAILLINIE-SEL.
            IF FEJLTYPE NOT = 1
               GO TO DETAILLINIE-ALT.
            WRITE FEJLLINIE1 FROM FEJLTRANS1
            GO TO DETAILLINIE-END.
DETAILLINIE-ALT.
            WRITE FEJLLINIE2 FROM FEJLTRANS2
DETAILLINIE-END.
         LINIENR + 1
         READ FEJLTRANS
         GO TO DETAILLINIER-ITR.
DETAILLINIER-END.
      GO TO SIDER-ITR.
SIDER-END.
   CLOSE - - -
PROCES-FEJLTRANSER-FEJLLISTE-END
```

Implementering af specialteknikker

Matching implementeres som ovenfor.

Backtracking kan implementeres ved hjælp af goto, hvis sproget ikke indeholder quit.

Strukturkonflikter kan implementeres på flere forskellige måder:

1. som selvstændige programmer i en batchkørsel med en mellemfil imellem.
2. som parallelle processer / corutiner, hvis sproget tillader det.
3. ved et hovedmodul - submodul arrangement, hvor submodulet inverteres:

Det kan være nødvendigt at kode (i hvert fald det øverste niveau af) et submodul nestfree, nemlig hvis det er tilladt at hoppe ind i loops / selektioner.

DataCentralen