

# Jan Pries-Heje:

## UNI – Et eksempel på anvendelse af teknikker til Struktureret Analyse og Design.

### IT-højskolen i København, Oktober 2001

#### INDHOLD

1.	Præsentation af case: Adgangskontrol hos UNI	4
2.	Dataflow-diagrammer	4
3.	Datastruktur-diagrammer	12
4.	Entitets/Relations-diagrammer	16
5.	Structure Charts	21
6.	Pseudokode	26
7.	Kontrolflow-diagrammer	28
8.	Kontrolspecifikationer	32
9.	Afrunding	

## 1. Præsentation af case: Adgangskontrol hos UNI

Hos UNI har man erkendt at man bruger alt for mange resurser på adgangskontrol. Man hverken kan eller vil dog undvære kontrollen, idet man har ansvar for en række meget fortrolige oplysninger som nødtigt skulle falde i de forkerte hænder.

Efter at have analyseret sagen grundigt har man besluttet at anskaffe et elektronisk system til adgangskontrol, hvor hver bruger får et magnetkort og en 4-cifret talkode.

Den grundlæggende funktion i adgangskontrol-systemet hos UNI skal være at tillade eller afvise adgang gennem en baseret på en sammenligning af magnetkort og indtastet talkode med adgangstilladelser registreret i en database. Denne primære funktion medfører et behov for en række sekundære funktioner, såsom:

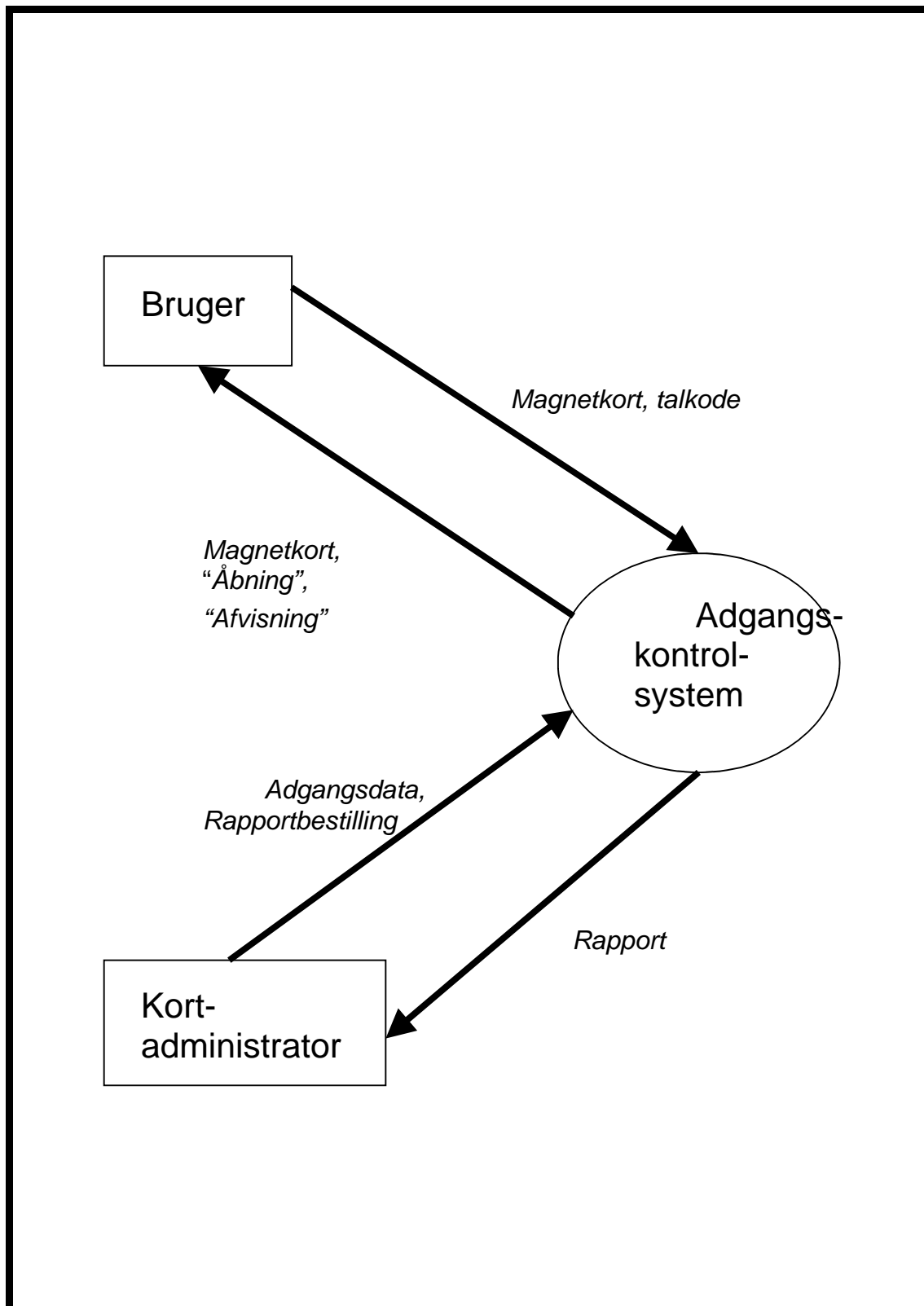
- Udstedelse af et magnetkort
- Tildeling og/eller ændring af en talkode til et magnetkort
- Registrering og/eller ændring af en adgangstilladelse

På markedet findes der selvfølgelig en række standardsystemer (COTS - Commercial Off The Shelf), men for eksemplets skyld vil jeg i de følgende afsnit gennemgå udvalgte dele af designet af et adgangskontrol-system til UNI.

## 2. Dataflow-diagrammer

Formålet med et dataflow diagram (herefter kaldet DFD) er på en let og overskuelig måde at vise:

- Hvilke data der strømmer igennem et system, uanset om systemet er stort, f.eks. et helt firma, eller lille.
- Hvor data kommer fra og hvor de går hen når de forlader systemet.
- Hvor data bliver gemt, dvs. i hvilke kartoteker.
- Hvilke processer der transformerer (ændrer) data.
- Hvordan samspillet er mellem processer og kartoteker.



Figur 1: Data kontekst diagram for UNI's adgangskontrol-system.

Der findes to notationer for DFD, kaldet De Marcos (1978) og Gane & Saons (1979) efter "opfinderne". I første omgang anvender jeg De Marcos notation der indeholder fire slags symboler:

1. En pil med tilhørende tekst angiver data kommende fra pilens start og gående til det pilen peger på.



2. En cirkel bruges til at angive en aktivitet eller en proces, som evt. kan transformere data.



3. To streger med tekst imellem angiver et kartotek og dets navn.



4. En kasse bruges til at angive en (for systemet) ekstern kilde til data.



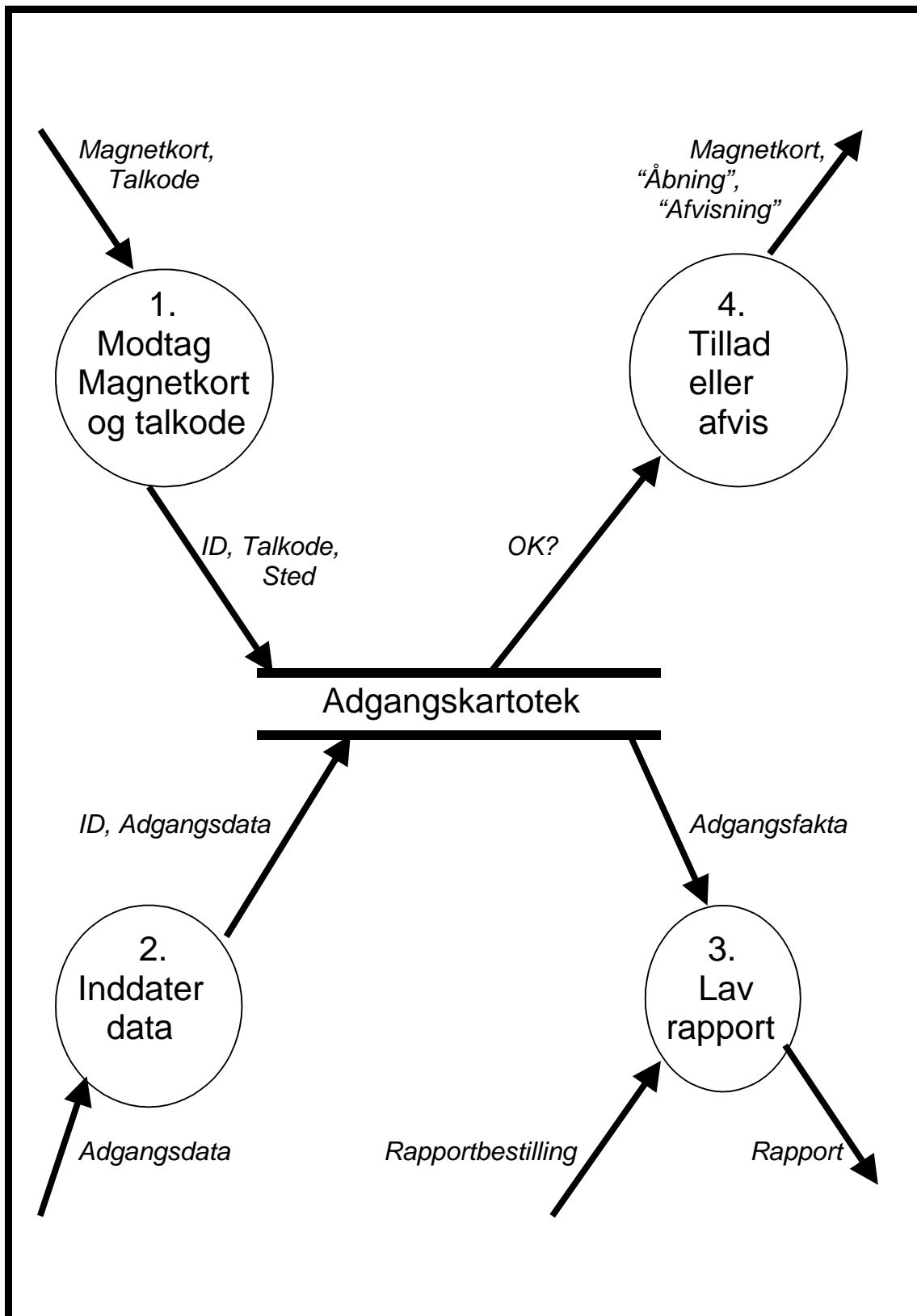
### Kontekst DFD

De Marco foreslår at man starter med at lave et såkaldt "Kontekst DFD" hvor man viser alle de eksterne relationer, input og output, som systemet har. På figur 1 er vist et Kontekst DFD for adgangskontrol-systemet, hvoraf det ses at der to eksterne kilder til data, dels en bruger, dvs. en ansat med behov for adgang det pågældende sted, dels en kartoteks-administratør.

Brugeren kommer med sit magnetkort og en talkode for at få adgang. Dette er illustreret med pilen fra "Bruger" ind til systemet.

Magnetkort eller talkode kan blive afvist af systemet, f.eks. fordi de er forkerte. Eller Brugeren kan få sit magnetkort tilbage samtidig med at der åbnes til det pågældende sted. Dette er illustreret med pilen fra systemet til "Brugeren".

Kartoteks-administratøren er den person der dels holder styr på hvem der har adgang til hvad hvornår, f.eks. hvilke ansatte der har hvilke magnetkort, dels sørger for at udskrevet adgangs-rapporter hvis og når man har brug for at kontrollere hvem der har været hvor.



Figur 2: Dataflow-diagram, niveau nul.

**DFD niveau 0 for adgangskontrol-systemet**

Det næste skridt efter "Kontekst DFD" er at opløse cirklen kaldet "Adgangskontrol-system" i dets hovedbestanddele, dvs. i de fire funktioner:

1. Modtag magnetkort og talkode
2. Inddater data og udsted magnetkort
3. Lav adgangsrapport
4. Tillad eller afvis adgang

Dette er vist i figur 2.

Bemærk at det er nøjagtig de samme pile der går ud og ind af de fire cirkler, som der gik ud og ind af den ene cirkel i kontekst DFD. Det DFD der opløser den ene cirkel der altid er i kontekst DFD, kaldes for niveau nul, svarende til det nul der stod inde i cirklen på kontekst DFD. Cirklerne på DFD niveau nul (DFD 0) nummereres fortløbende startende med nummer et.

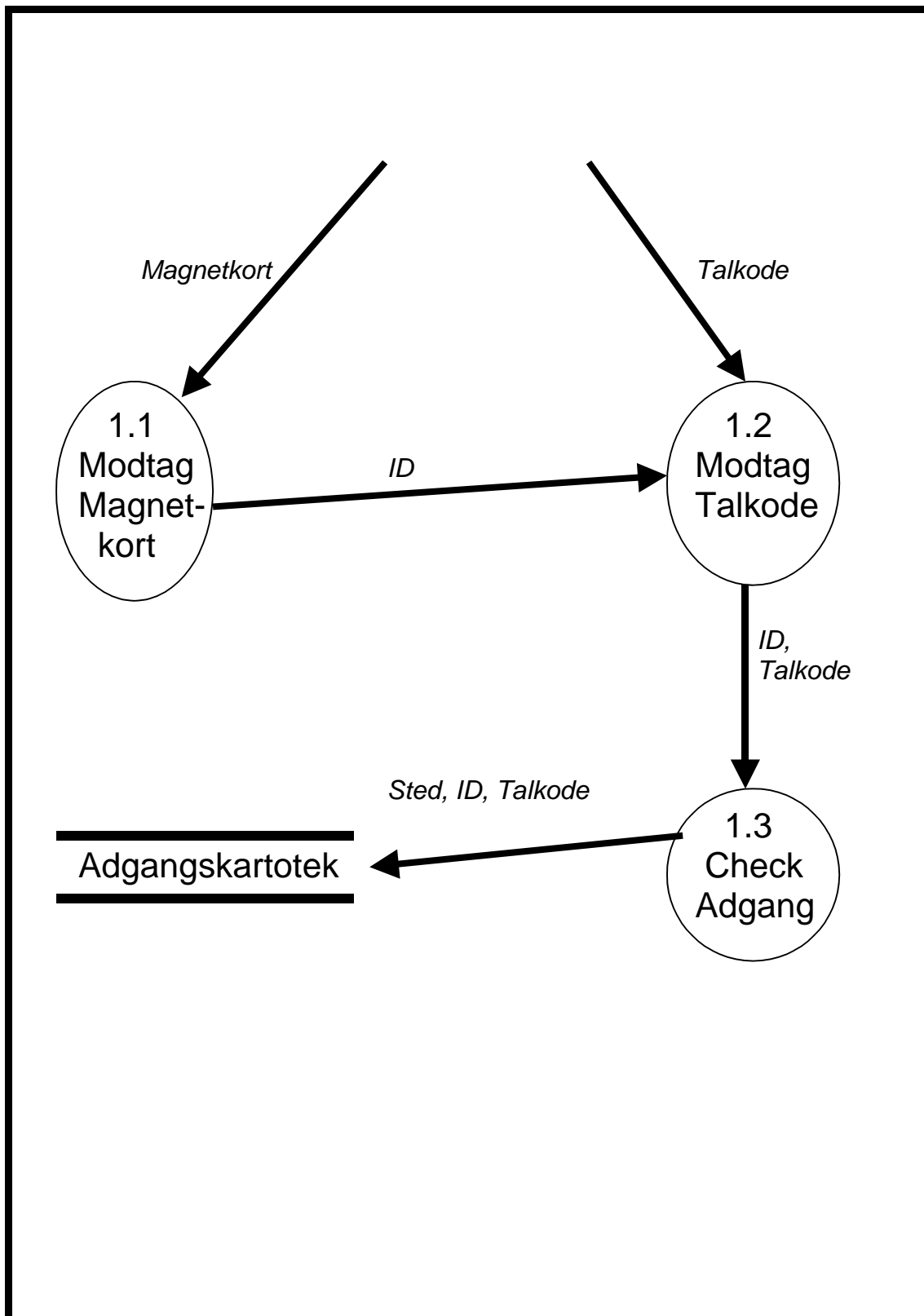
**DFD niveau 1 for adgangskontrol-systemet**

Om nødvendigt kan hver af de fire cirkler i DFD 0 opløses i underaktiviteter. Generelt anbefaler De Marco at man aldrig har mere end syv cirkler i et DFD. Hvornår er det nødvendigt at opløse en cirkel? De Marco's anbefaling er, at man opløser sine DFD til et niveau, hvor hver cirkel kan beskrives på maksimalt en side struktureret dansk.

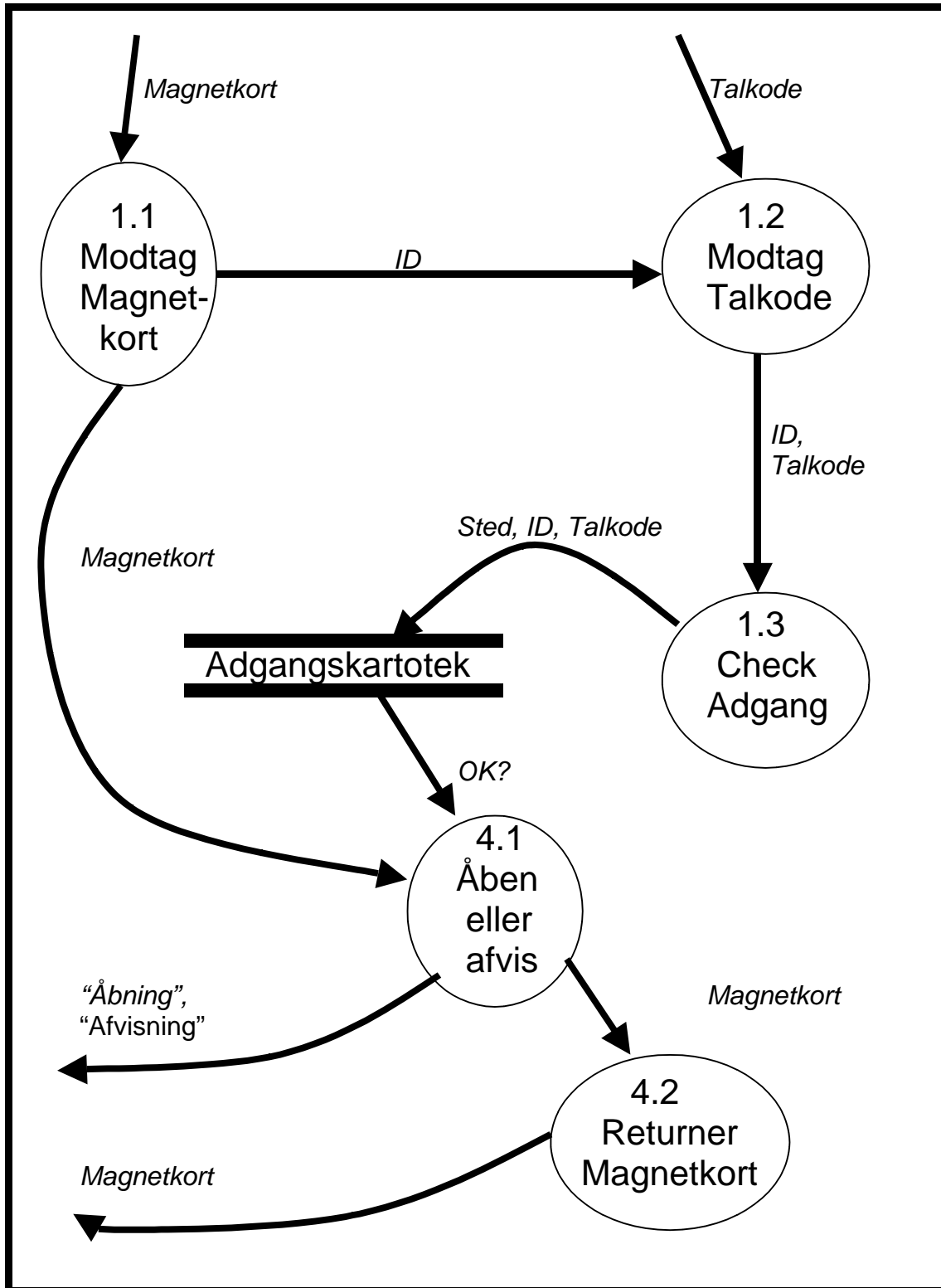
Når man opløser en cirkel så overtager underaktiviteterne den overordnede aktivitets nummer, plus en fortløbende nummerering. På figur 3 er aktivitet nummer 1, "Modtag magnetkort og talkode", opløst i tre aktiviteter nummereret fra 1.1 til 1.3.

På figur 4 er både aktivitet nummer 1 og aktivitet nummer 4 opløst i tilsammen fem delaktiviteter. Bemærk hvordan figur 4 giver et bedre overblik over hele forløbet end figur 3. Måske burde vi aldrig i vores design have delt aktivitet 1 og aktivitet 4 ?

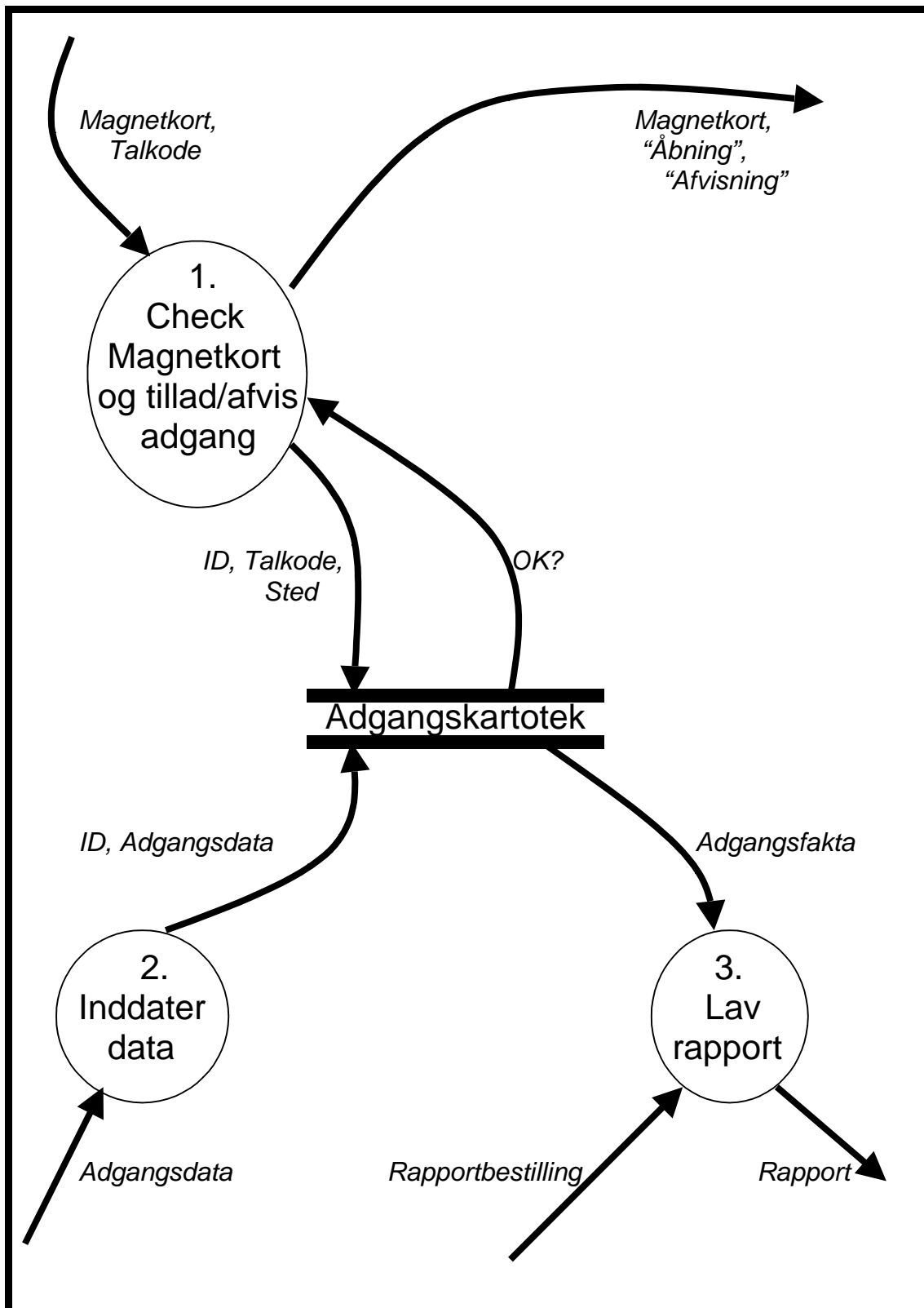
I figur 5 er vist et ny DFD niveau nul ( i stedet for figur 2) med tre i stedet for fire aktiviteter. Ændringen af figur 2 til figur 5 illustrerer bottom-up design, dvs. hvordan vi langt nede i en detail-problemstilling kan nå en erkendelse der får os til at gå tilbage (op) og ændre i noget vi egentlig var færdige med.



Figur 3: Dataflow-diagram niveau 1 af "Modtag magnetkort og talkode".



Figur 4: Dataflow-diagram niveau 1 afaktivitet 1 "Modtag magnetkort og talkode" og aktivitet 4 "Tillad eller afvis".



Figur 5: Nyt Dataflow-diagram, niveau nul, med kun tre aktiviteter.

#### 4. Datastruktur-diagrammer

Formålet med et data struktur diagram (DSD), også kaldet et "Jackson diagram" efter "opfinderen" Michael A. Jackson (1983), kan enten være:

1. For en entitet at vise tilknyttede handlinger og deres rækkefølge (1983: 41-42).
2. At vise strukturen i et dataflow (1983: 376).
3. At vise strukturen i et program (1983: 376).

Notationen i et DSD er følgende:

- Alle datastrukturer repræsenteres ved aflange "kasser".
- Kasser på linie angiver at den første kasse (længst til venstre) kommer før den næste etc.
- En cirkel i højre hjørne af en kasse betyder "exclusive or". Så hvis to kasser står på linie og de begge har en lille cirkel i højre hjørne, så betyder det at kassen på niveauet lige over *enten* kan høre til den ene *eller* den anden kasse.
- En stjerne i højre hjørne af en kasse betyder, at der kan forekomme mange af den pågældende kasse.

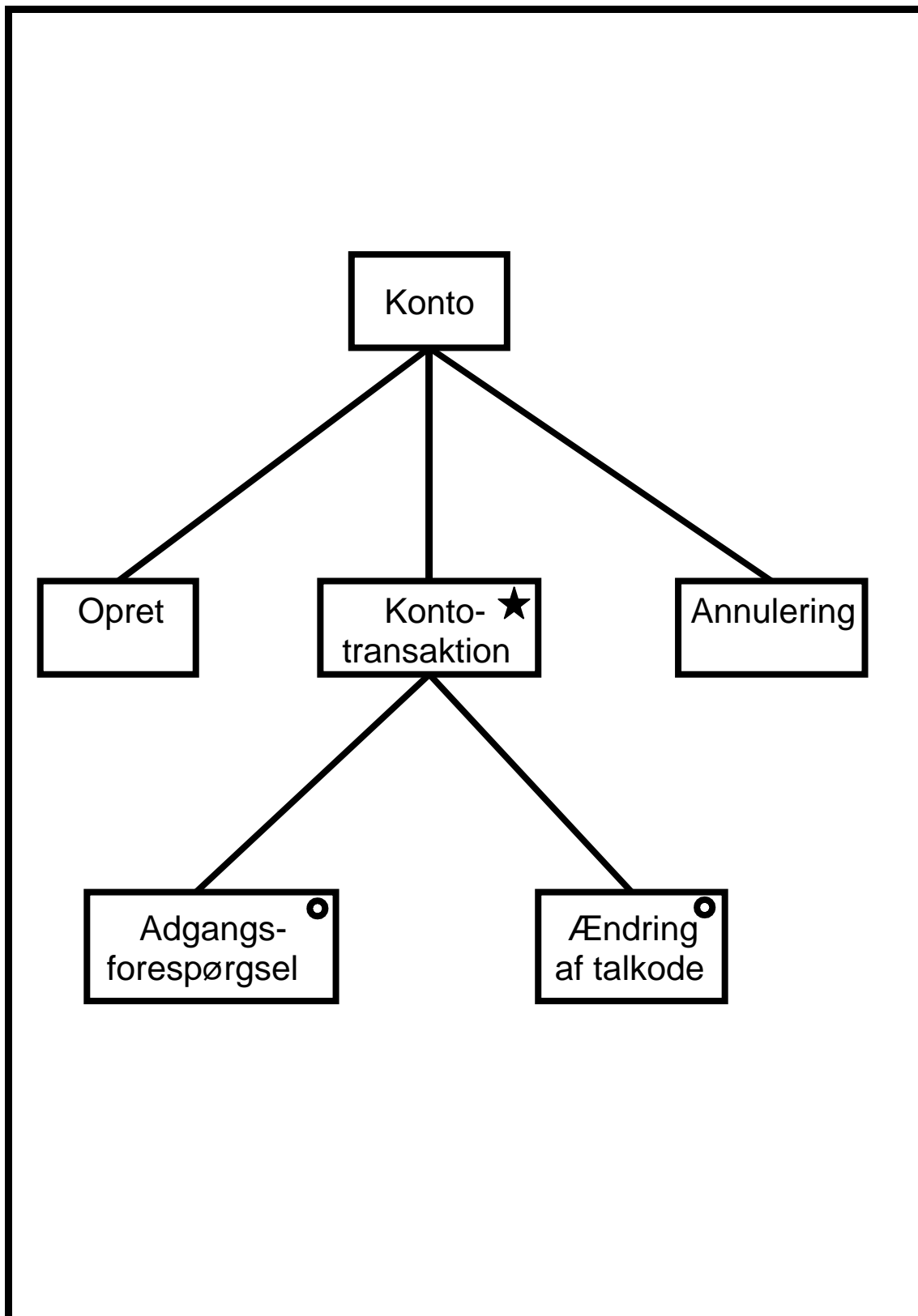
#### Et DSD for entiteten Konto og tilknyttede handlinger

Vi vender nu tilbage til adgangskontrol-systemet. Lad os antage at en person får udstedt et nyt magnetkort. Ved udstedelsen er der tilknyttet en tilfældig fire-cifret talkode. Denne kode kan ændres enten efter ønske eller af sikkerhedsgrunde. Magnetkortet med talkoden kan bruges til at opnå adgang et eller flere steder. Magnetkortet er gyldigt indtil det annulleres f.eks. fordi det er blevet stjålet, slidt op el. lign.

Lad os nu sige at magnetkortet er den fysiske fremtræden for en logisk entitet som vi kunne kalde adgangskonto eller blot konto. Til en konto er der knyttet tre handlinger:

- Den første handling er oprettelsen af en konto, f.eks. når et nyt magnetkort bliver udstedt med en tilfældig talkode tilknyttet.
- Den anden type handling er transaktioner i forbindelse med kontoen. Der kan være tale om to typer transaktioner, enten en adgangsforespørgsel eller en ændring af talkoden.
- Den tredje og sidste handling er annulleringen (nedlæggelsen) af en konto.

I figur 6 er vist et DSD for entiteten Konto og de tre tilknyttede handlinger.



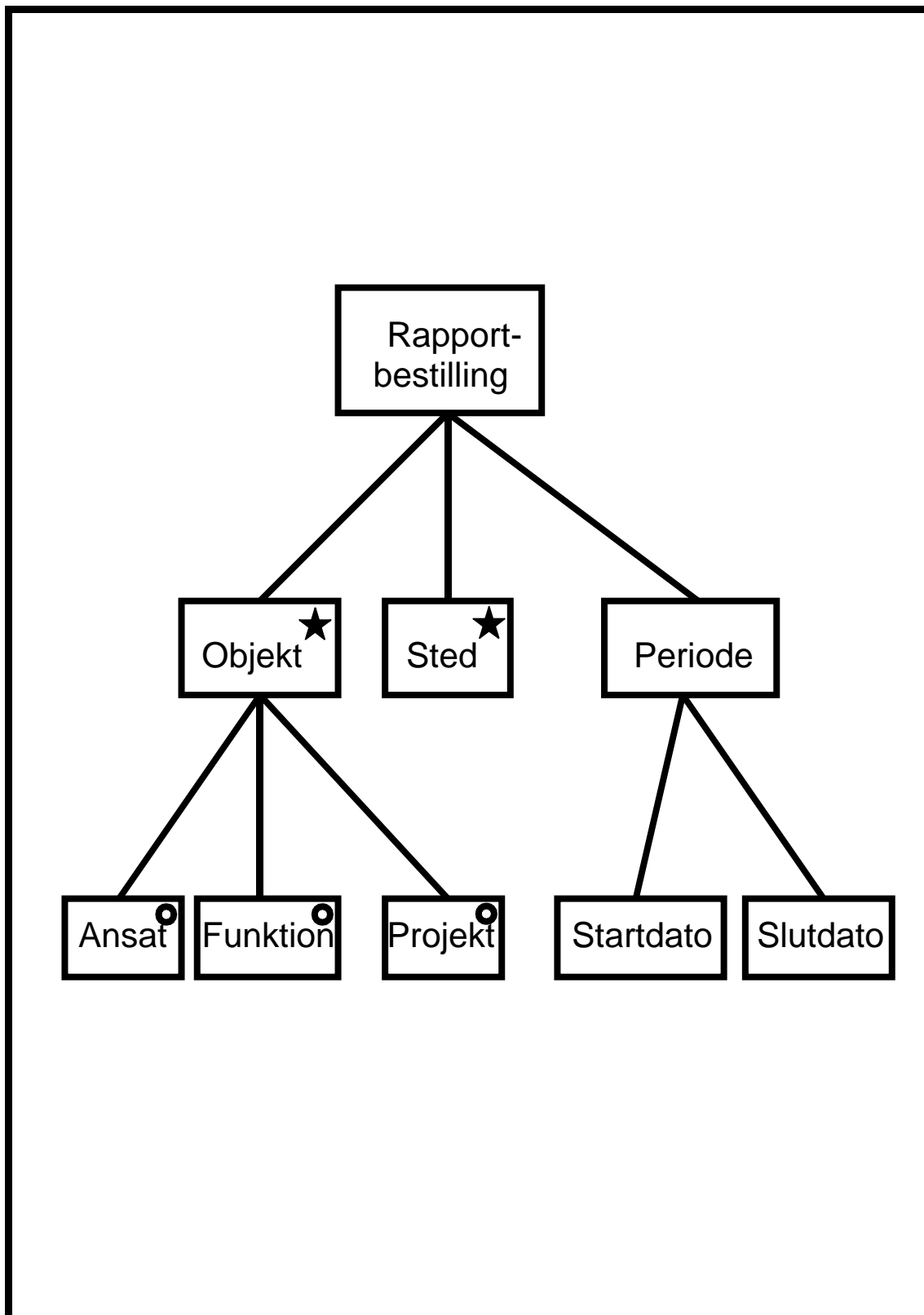
Figur 6: Datastruktur-diagram for entiteten konto.

**Et DSD for datastrømmen Rapportbestilling**

I DFD for adgangskontrol-systemets niveau nul var der en funktion nr. 3, lav rapport. Til denne funktion var der en indkommende datastrøm kaldet rapportbestilling. Lad os sige at UNI ønsker at magnetkort kan udstedes til:

1. En ansat personligt
2. En bestemt job-funktion, f.eks. sikkerhedsansvarlig, kopist, vagthavende osv.
3. Projekter, øvelser eller andre tidsbegrænsede anvendelser.
4. Bestemte steder (= en enkelt adgangskontrol f.eks. en dør) eller områder (= flere adgangskontroller, f.eks. døre på 6. Etage i en bestemt bygning).

Når en rapport bliver bestilt så sker det enten for en ansat, en funktion eller et projekt. Man ønsker en rapport om en eller flere evt. alle adgangskontroller. Og man ønsker måske at studere en nærmere bestemt periode angivet ved en start- og en slutdato. I figur 6.7 er vist et DSD for rapportbestilling.



Figur 7: Datastruktur-diagram for dataflowet rapportbestilling.

## 5. Entitets/Relations-diagrammer

Formålet med et Entitets/reasons diagram (herefter forkortet ERD) er at designe en database, dvs. at finde ud af hvilke entiteter der skal indgå i en database og hvilke relationer der er mellem entiteterne. Til en entitet knytter sig som regel nogle egenskaber også kaldet attributter. F.eks. kunne entiteten Person have attributterne "Navn", "Adresse", "CPR-nummer" "Fødselsdag" etc. Mellem to entiteter kan der være en relation. Denne relation kan kategoriseres som en af følgende tre typer (også kaldet relationens kardinalitet):

- En en-til-en relation (noteres ofte 1:1). F.eks. vil der til en gift mand kun høre én gift kone og omvendt.
- En en-til-mange relation (noteres ofte 1:m). F.eks. kan en mor have mange børn, men til hvert barn vil der kun være en og netop en mor.
- En mange-til-mange relation (noteres ofte n:m). F.eks. kan en person eje mange huse, samtidig med at hvert af husene kan have mange ejere.

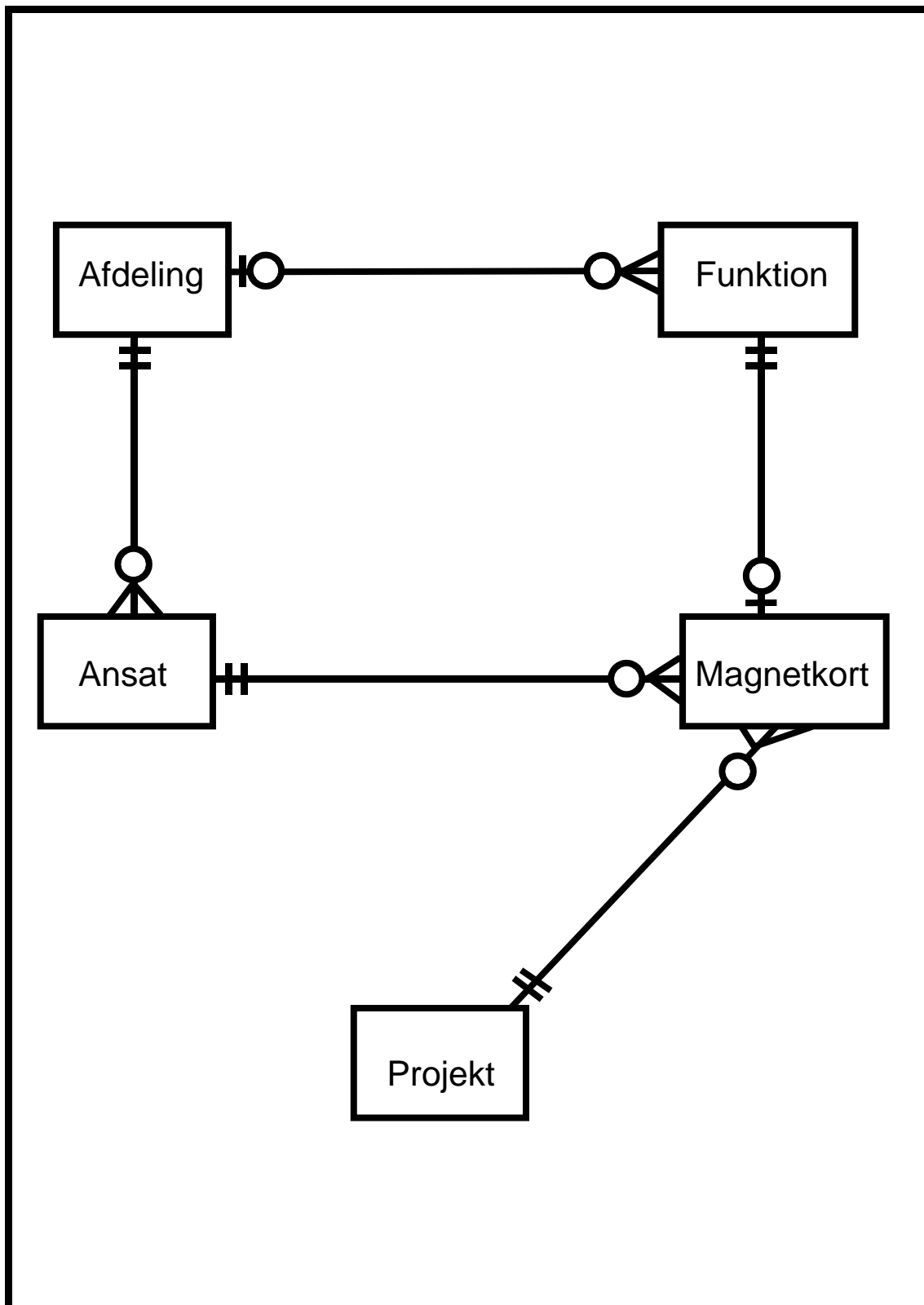
Der findes mange forskellige notationer for ERD. Reglerne i den notation jeg vil anvende her er følgende:

- Entiteter repræsenteres ved aflange kasser.
- Relationer er streger med hønseben til "mange"-siden
- På relationerne kan man med en lille streg på tværs angive, at relationen *altid* eksisterer. Alternativt kan man med en lille cirkel angive at relationen *kan* eksistere men ikke nødvendigvis behøver at gøre det.
- Attributter vises ikke i diagrammet men noteres i en tabel med angivelse af attributterne for hver entitet.

### Entitets-Relations diagram for Adgangskartoteket

I DFD niveau nul for adgangskontrol-systemet indgik der et kartotek kaldet "Adgangskartoteket". En del af dette kartotek er illustreret i figur 8. Det diagrammet viser os er en del af indholdet af kartoteket, nemlig den del der handler om personer, projekter, funktioner og magnetkort:

- Enhver ansat er *altid* knyttet til en og kun en afdeling, mens en afdeling *kan* have mange ansatte (en-til-mange relation).
- En ansat *kan* have mange magnetkort, og et magnetkort *kan* kun høre til én bestemt ansat (en-til-mange relation).
- Et magnetkort *kan* også være knyttet til en bestemt funktion. I så fald *kan* der kun høre ét magnetkort til funktionen (en-til-en relation).
- Et magnetkort *kan* endelig være knyttet til et bestemt projekt, og projektet kan godt have mange magnetkort tilknyttet.
- En job-funktion kan høre til en afdeling, men behøver ikke at gøre det. En afdeling kan have en eller flere job-funktioner tilknyttet.



Figur 8: Entitets/relation-diagram for en del af Adgangskartoteket

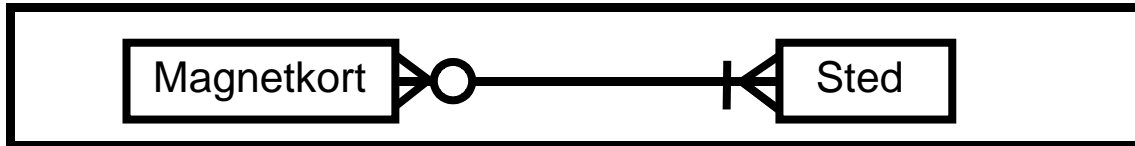
Til hver af entiteterne i figur 8 er der knyttet følgende attributter:

Afdeling	<u>Afdelings-initialer (Max. 4 bogstaver)</u> Afdelings-navn Adresse
Ansæt	<u>CPR-nummer</u> Navn Gade Postnummer By
Funktion	<u>Funktionsnummer (FUN)</u> Funktionsnavn
Projekt	<u>Projektnummer (PN)</u> Projektnavn
Magnetkort	<u>ID</u> Talkode <i>En reference til steder hvor kortet gælder</i>

Den førstnævnte attribut ved hver entitet er en såkaldt *nøgle* der entydigt identificerer en bestemt entitetsforekomst. F.eks. vil et CPR-nummer kunne bruges til at finde én og kun én ansat. Nøglen skrives ofte understreget som i rammen oven for.

### Normalisering

Hvis vi i vores ERD havde haft en mange-til-mange relation så ville det have været nødvendigt at forfinne designet af databasen yderligere. Forestil dig f.eks. at vi havde haft en mange-til-mange relation mellem Magnetkort og entiteten Sted, svarende til at et magnetkort gav adgang til mere end et sted, samtidig med at der til et sted sagtens kunne være mange forskellige magnetkort der gav adgang.



Hvis vi nu skulle lave et kartotekskort for et Magnetkort så ville vi ikke vide hvor mange pladser vi skulle holde fri til at notere Steder der var adgang til.

Magnetkort	<u>ID</u>
	Talkode
	Reference til Sted #1
	Reference til Sted #2
	Reference til Sted #3
	<i>Hvor mange skal vi gøre plads til ???</i>

Tilsvarende ville vi for et Sted ikke vide hvor mange ID (= nøgler til magnetkort) vi skulle gøre plads til for at kunne notere alle der havde lovlig adgang.

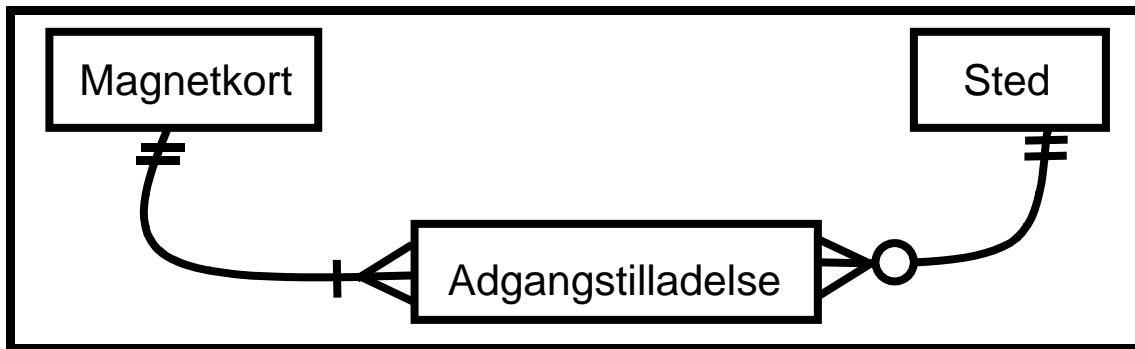
Sted	<u>Sted#</u>
	Gade
	Postnummer
	By
	ID - Reference til Magnetkort #1
	ID - Reference til Magnetkort #2
	ID - Reference til Magnetkort #3
	<i>Hvor mange skal vi gøre plads til ???</i>

Og hvis vi sætter plads af til f.eks. 1.000 Steder for hvert magnetkort og 10.000 magnetkort til hvert sted, så spilder vi en utrolig masse plads i kartoteket, f.eks. for alle meget hemmelige steder som kun har en eller to magnetkort der må give adgang.

Løsningen er at normalisere datafilerne. I praksis taler man om to normalformer der af historiske grunde kaldes 1. og 3. normalform. Når man normaliserer til 1. normalform fjerner man netop det problem vi har set på, ved at repræsentere en mange-til-mange. Problemet løses ved at indføre en hjælpe-entitet mellem de to entiteter der havde en

mange-til-mange, således at man får to en-til-mange relationer i stedet for en mange-til-mange relation.

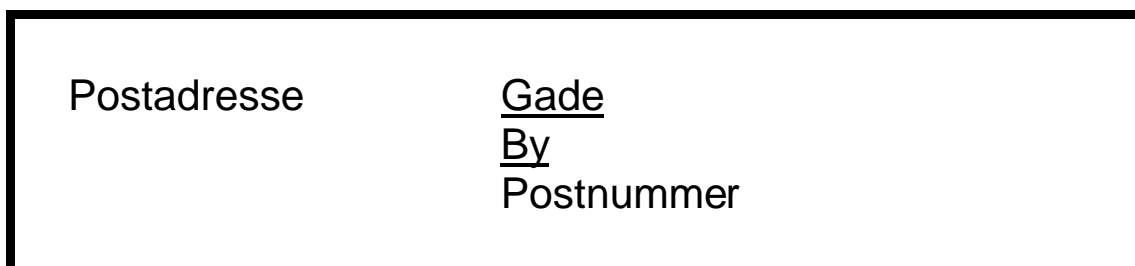
Neden for er vist hvordan man kan indskyde en ny hjælpe-entitet kaldet *adgangstilladelse* således at ét magnetkort kan have mange adgangstilladelser, samtidig med at én adgangstilladelse kun kan dreje sig om ét magnetkort. Og tilsvarende at der til ét sted kan være mange adgangstilladelser, men at én adgangstilladelse kun kan være til ét sted.



### Tredje normalform

Tredje normalform kræver at afhængige oplysninger skal i separate filer (entiteter). Det vil sige at alle attributter tilknyttet en entitet skal være uafhængige af hinanden, således at indholdet af en attribut kan ændres uden at indholdet af andre attributter også skal ændres.

I vores database har vi kun et eksempel på afhængige oplysninger. Feltet Postnummer afhænger nemlig af Gade og By. Så for at vores database kan siges at være på 3. normalform, så skal vi fjerne attributten Postnummer fra entiteterne Ansæt og Sted, og i stedet lave en ny entitet:





## 6. Structure Charts

I forbindelse med designet af et edb-system, når man skal til at finde ud af hvilke moduler der skal indgå i systemet, kan man anvende såkaldte "Structure Charts", herefter kaldet SC.

SC anvendes fortrinsvis som designværktøj i forbindelse med såkaldt "Structured Design" (Page-Jones 1988). Formålet med SC er at:

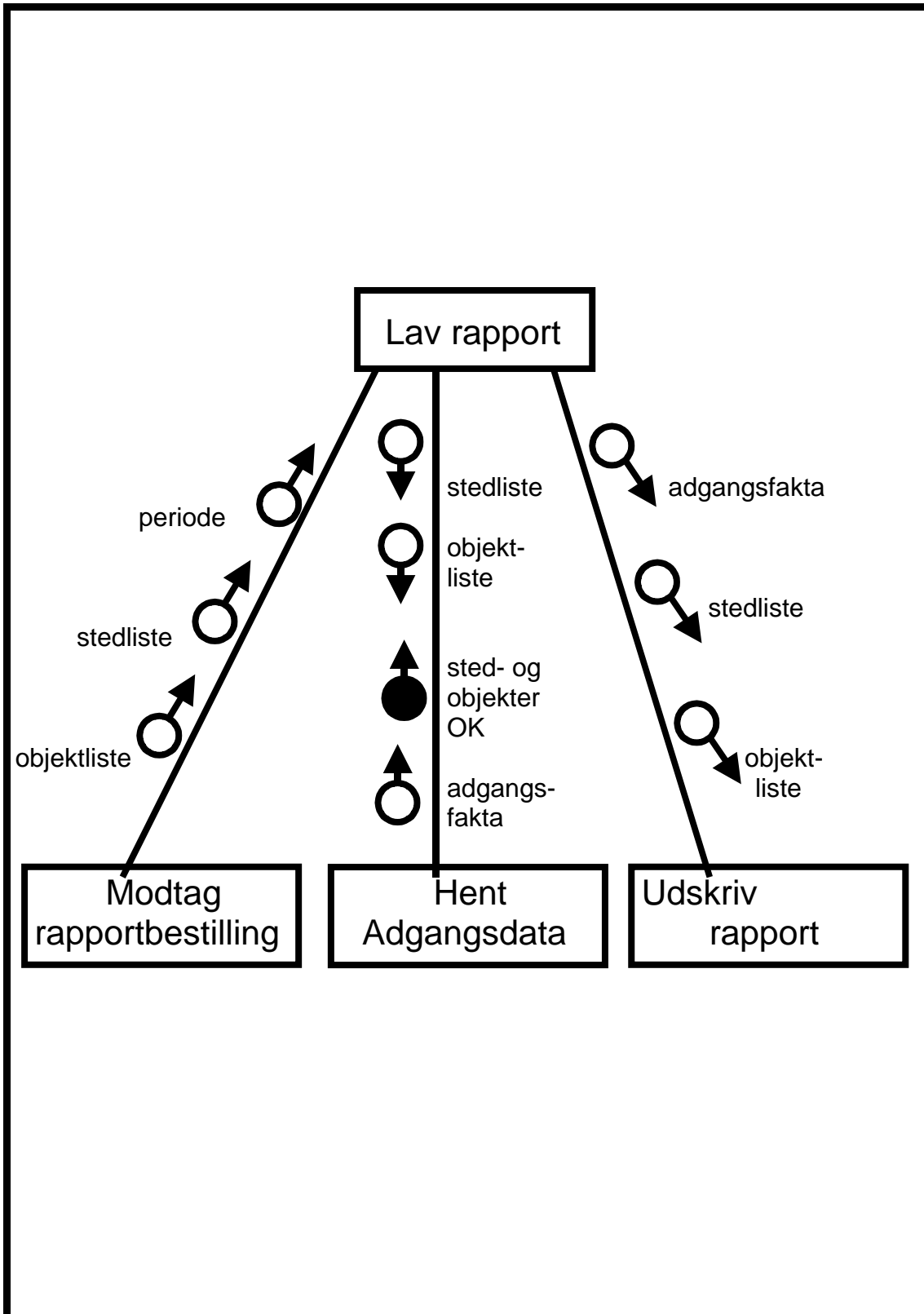
- Vise hvilke moduler der indgår i et edb-system.
- Vise hvordan modulerne er organiseret, dvs. hvilke moduler kalder hvilke moduler?
- Vise hvilken information modulerne videregiver når de kalder hinanden.
- Fungere som specifikation for en programmør (sammen med evt. nødvendige modulspekifikationer, f.eks. i form af pseudokode) der skal udvikle selve programkoden.

Følgende notation anvendes i et SC:

- En aflang kasse symboliserer et modul. Hvis modulet er "standard" eller hentes fra et bibliotek, så vil kassen have dobbeltstreger i begge sider.
- En linie mellem to kasser, angiver at modulet øverst i diagrammet kan kalde modulet nederst.
- En lille pil med *ikke-udfyldt* cirkel, langs med en kalde-linie viser data der sendes i pilens retning fra det ene modul til det andet.
- En lille pil med *udfyldt* cirkel, langs med en kalde-linie viser et flag der sendes i pilens retning fra det ene modul til det andet. Et flag er et symbol, ofte repræsenteret ved en enkelt bit, der benyttes til at angive en tilstand, f.eks. sand/falsk.
- En kurvet (rund) pil på en kalde-linie angiver gentagelse (iteration), typisk et kald inde fra en løkke.
- En lille udfyldt diamant angiver et betinget kald, typisk et kald fra en IF- eller CASE-sætning.

### Structure Chart for bestilling & udskrivning af rapporter

I DFD niveau nul var der en aktivitet nummer tre der hed "Lav rapport". I figur 9 er vist et SC hvor "Lav rapport" kalder tre moduler. Først modtages en rapportbestilling. Vi ved (se evt. figur 7) at en rapportbestilling består af et eller flere objekter (ansat, funktion, projekt), et eller flere steder, samt af netop én periode, så fra modulet "Modtag rapportbestilling" kommer der en objektliste, en stedliste og en periode. Objekt- og stedlisten bruges så i kaldet af modulet "Hent adgangsdata". Og tilbage fra dette modul får vi dels adgangsfakta, dels et flag der viser om alle objekter og steder i listerne var OK eller ej. Til sidst udskriver vi de fundne adgangsfakta i forhold til objekter og steder i det tredje modul.

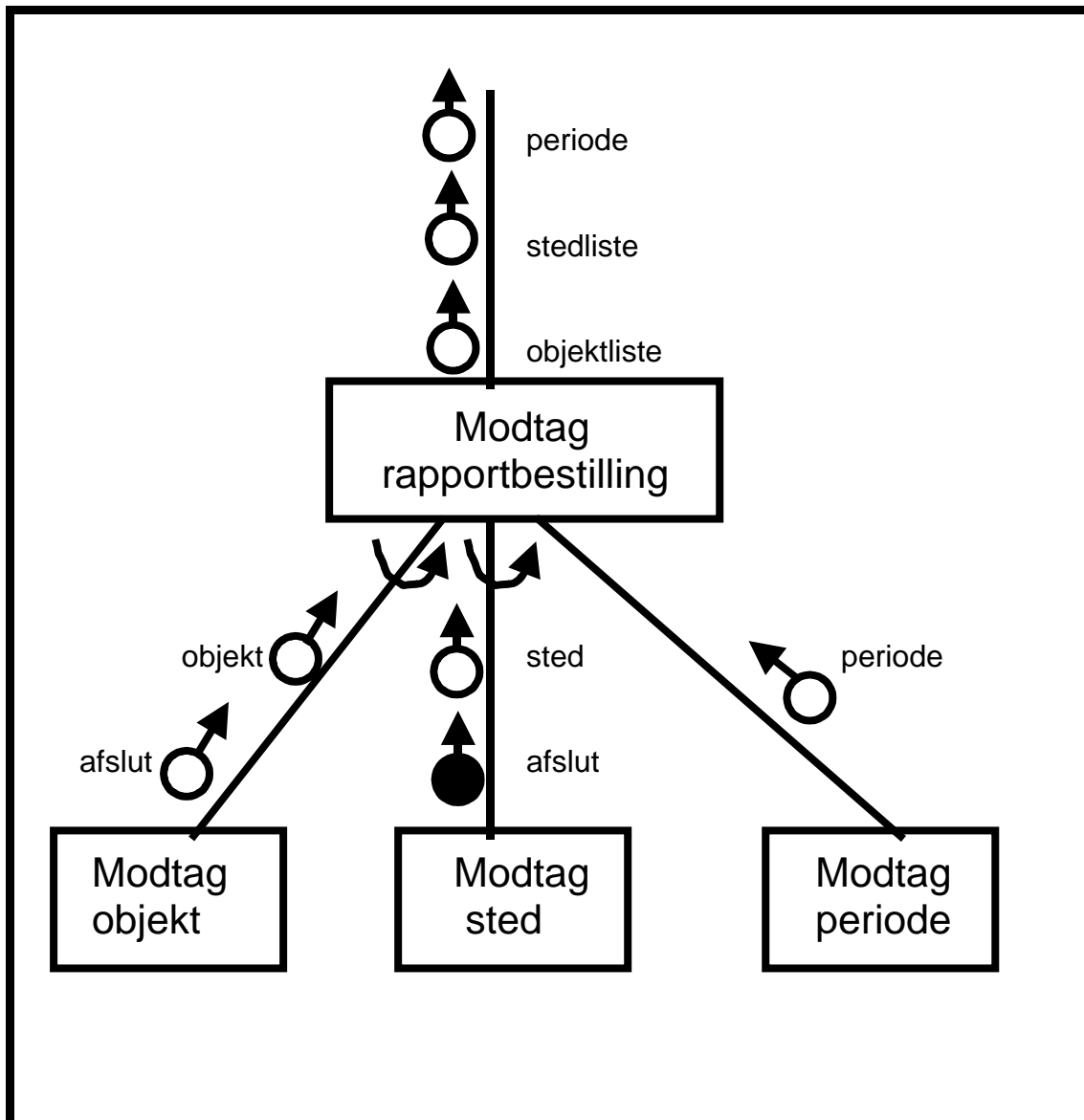


Figur 9: Structure Chart for "Lav rapport" fra DFD niveau nul.

### Structure Charts for at modtage rapportbestilling

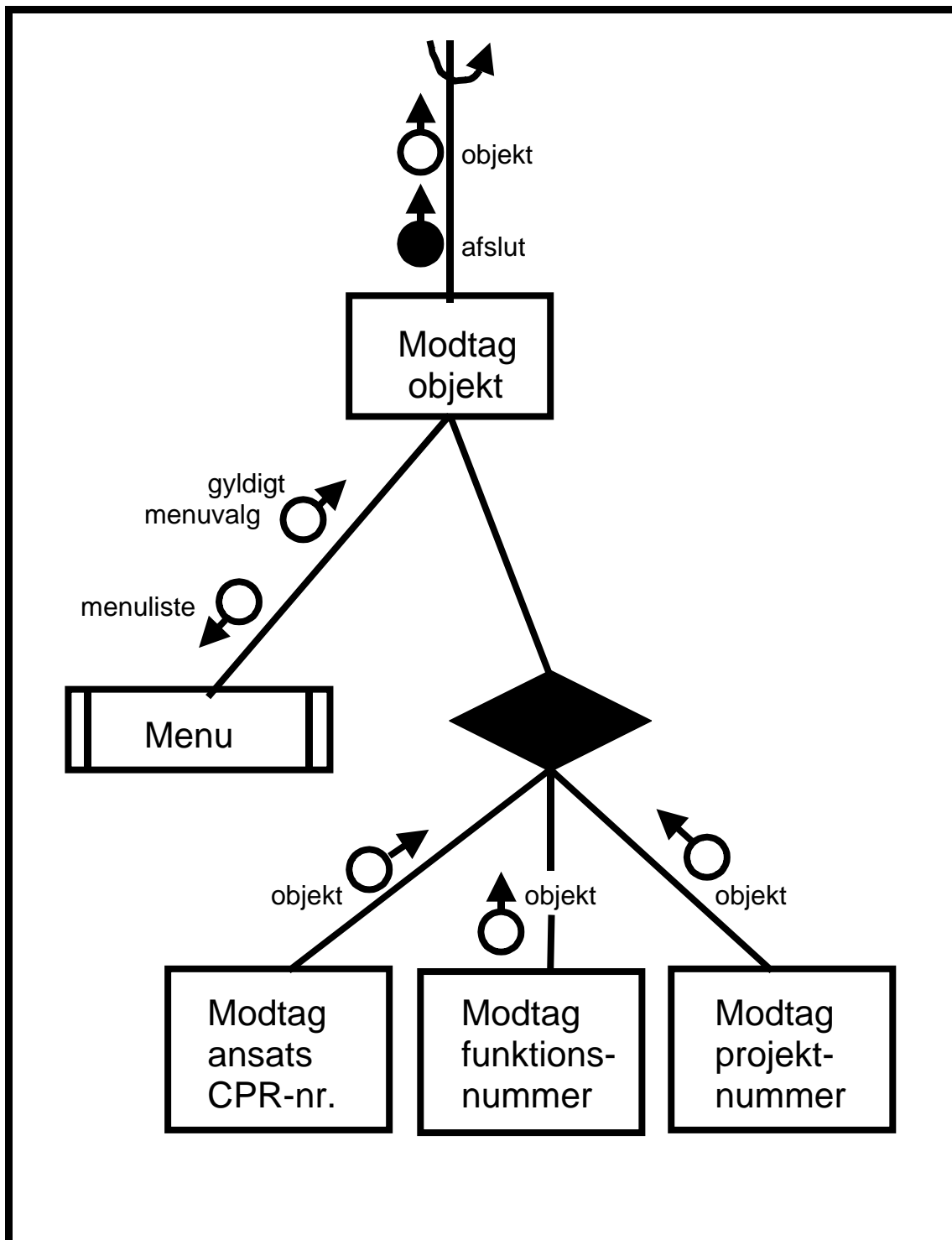
Ligesom DFD kan have flere niveauer, så kan SC også dekomponeres i flere niveauer. I figur 10 har jeg dekomponeret det første modul fra SC i figur 9, modtag rapportbestilling, i to moduler:

1. Modtag objekt. Dette modul gentages, symboliseret ved den kurvede pil, indtil flaget afslut bliver "sandt", dvs. når brugeren af edb-systemet ikke ønsker at indtaste flere objekter til sin rapportbestilling.
2. Modtag periode.



Figur 10: Dekomponering af "Modtag rapportbestilling" fra SC, figur 9

Figur 11 viser endnu en dekomponering, nu blot af "Modtag objekt" fra den figur 10.



Figur 11: Dekomponering af "Modtag objekt" fra SC i figur 10.

Det figur 11 viser er at "Modtag objekt" først kalder et biblioteksmodul der hedder "Menu" med en "Menuliste" som argument. Menu returnerer så et gyldigt menuvalg (gyldigt i forhold til den fremsendte menuliste). Vi kan forestille os at menuen f.eks. ser ud som vist neden for.

Vælg en af nedenstående punkter (indtast tallet):

1. Vælg ansat
2. Vælg funtion
3. Vælg projekt
4. Afslut

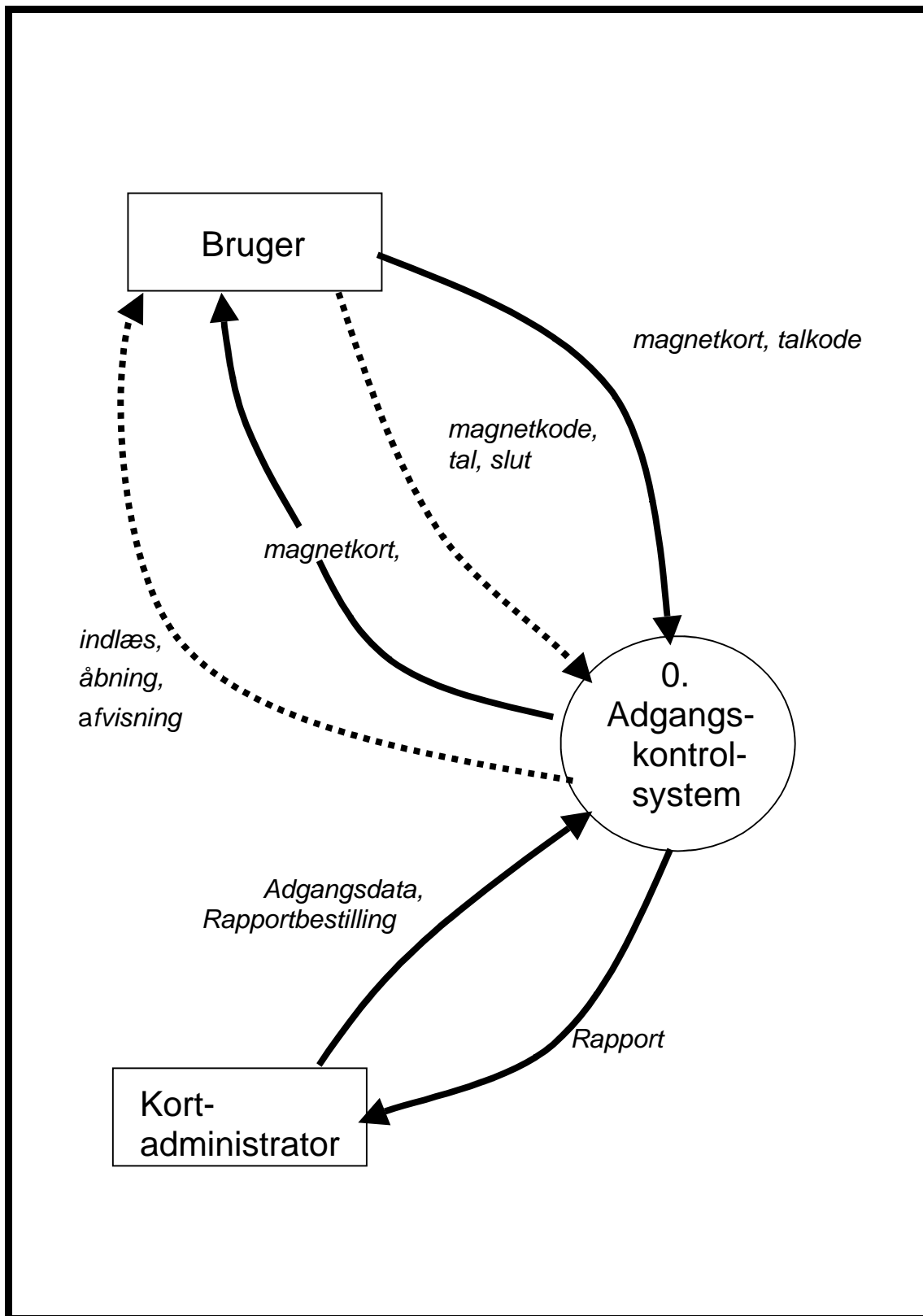
Afhængigt af hvad brugeren vælger vil "Modtag objekt" enten bede brugeren om at indtaste en identifikation (nøglen) af det valgte objekt, f.eks. indtastning af et CPR-nummer for en ansat, eller "Modtag objekt" vil returnere til det kaldende modul med flaget "afslut" sandt.

## 7. Pseudokode

Hvor det er nødvendigt kan de enkelte moduler i et SC uddybes f.eks. ved hjælp af pseudokode. Pseudokode er en grov skitsering af kildeteksten til et program ved hjælp af struktureret dansk. Nedenfor er givet et eksempel på pseudokode for modulet "Modtag periode" i en notation der minder om programmeringssproget Pascal.

```
1:  Modul Modtag periode (Start,slut: string[6]);
2:
3:  Start := "000000";
4:  Slut := "000000";
5:  Bed om indtastning af periodestart;
6:  REPEAT
7:    Læs indtastet periodestart;
8:    IF indtastet periodestart er gyldig dato
9:      THEN start := indtastet periodestart;
10:   ELSE udskriv fejltekst og bed om korrekt indtastning
11: UNTIL start er forskellig fra "000000";
12: Bed om indtastning af periodeslut;
13: REPEAT
14:   Læs indtastet periodeslut;
15:   IF indtastet periodeslut er gyldig dato
16:     og indtastet periodeslut > start
17:     THEN slut := indtastet periodeslut;
18:   ELSE udskriv fejltekst og bed om korrekt indtastning
19: UNTIL slut er forskellig fra "000000";
```

Figur 12: Pseudokode for modulet "Modtag periode" fra figur 10.



Figur 13: Kontrolflow kontekstdiagram

## 8. Kontrolflow-diagrammer

En af bagdelene ved DFD er at de primært er beregnet til at beskrive processer og delvis data, men ikke hændelser. Med andre ord så er DFD i den form vi hidtil har set velegnet til administrative systemer, såsom forretningssystemer og kontorautomatisering.

For at råde bod på dette har Hatley & Pirbhai (1988) udvidet DFD med nogle såkaldte *kontrolflows*, der symboliserer diskrete signaler, dvs. signaler der tilfældigt kan antage givne værdier<sup>1</sup> (1988: 64-65). Hovedpointen ved Hatley & Pirbhais kontrolflows er, at man på de samme diagrammer som DFD indfører kontrolflows som stiplede pile.

I figur 13 og 14 er henholdsvis vist det tidligere data kontekst diagram (figur 1), og DFD niveau nul (figur 2), med kontrolflow-pile indtegnet. Vi ser at brugeren af magnetkortet kan afgive tre signaler:

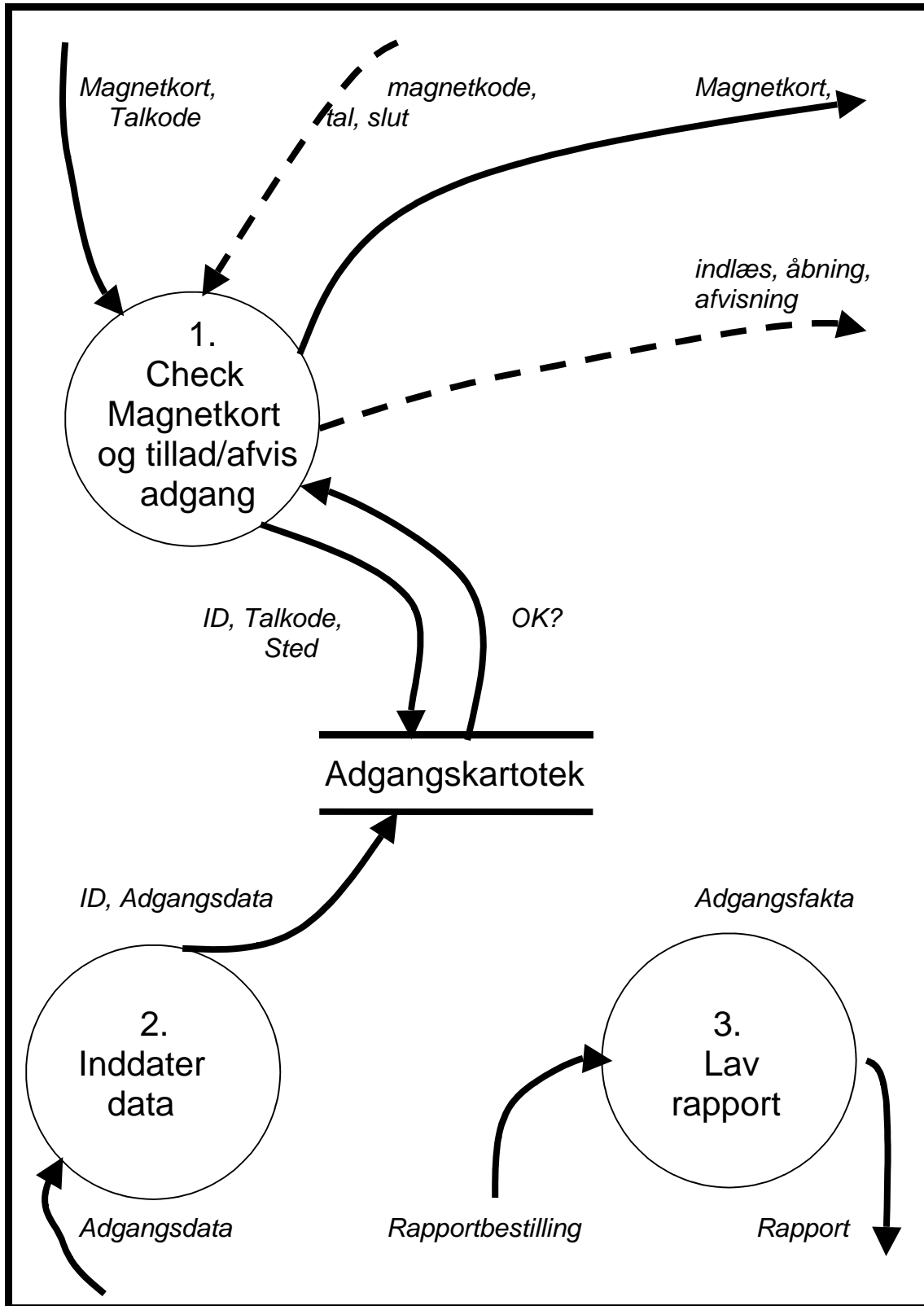
1. "Magnetkode" svarende til at magnetkortet køres igennem kortlæseren på adgangskontrollen.
2. "Tal" svarende til indtastningen af et af tallene i talkoden.
3. "Slut" når vedkommende er færdig med at indtaste talkoden og ønsker at få adgang.

Og vi ser at systemet kan signalere til brugeren at:

4. At magnetkoden ikke blev læst korrekt. "Indlæs" hedder signalet der foranlediger at brugeren skal prøve at indlæse magnetkortet igen.
5. Adgang blev givet ("Åbning") eller nægtet ("Afvisning").

---

<sup>1</sup> I modsætning til kontinuerte signaler, dvs. signaler der kan antage vilkårligt mange numerisk ordnede værdier. Denne type signaler repræsenteres typisk i et system som dataflows (Hatley & Pirbhai 1988: 64).



Figur 14: Kontrolflow-diagram, niveau nul.

## Kontrollflow diagram niveau 1

Der ligger flere vigtige designbeslutninger gemt bag beslutningen om disse fem signaler som laver voldsomt om i DFD niveau 1 (se evt. figur 4). Først og fremmest betyder magnetkode-signalet, at jeg ikke længere vil holde fast på magnetkortet og returnere det til sidst. Så aktivitet 4.2 *Returner Magnetkort* bliver overflødig og bortfalder.

Dernæst beslutter jeg "Åbning" og "Afvisning" ikke er data men hændelser, f.eks. bestående i at en dør åbnes eller forbliver lukket.

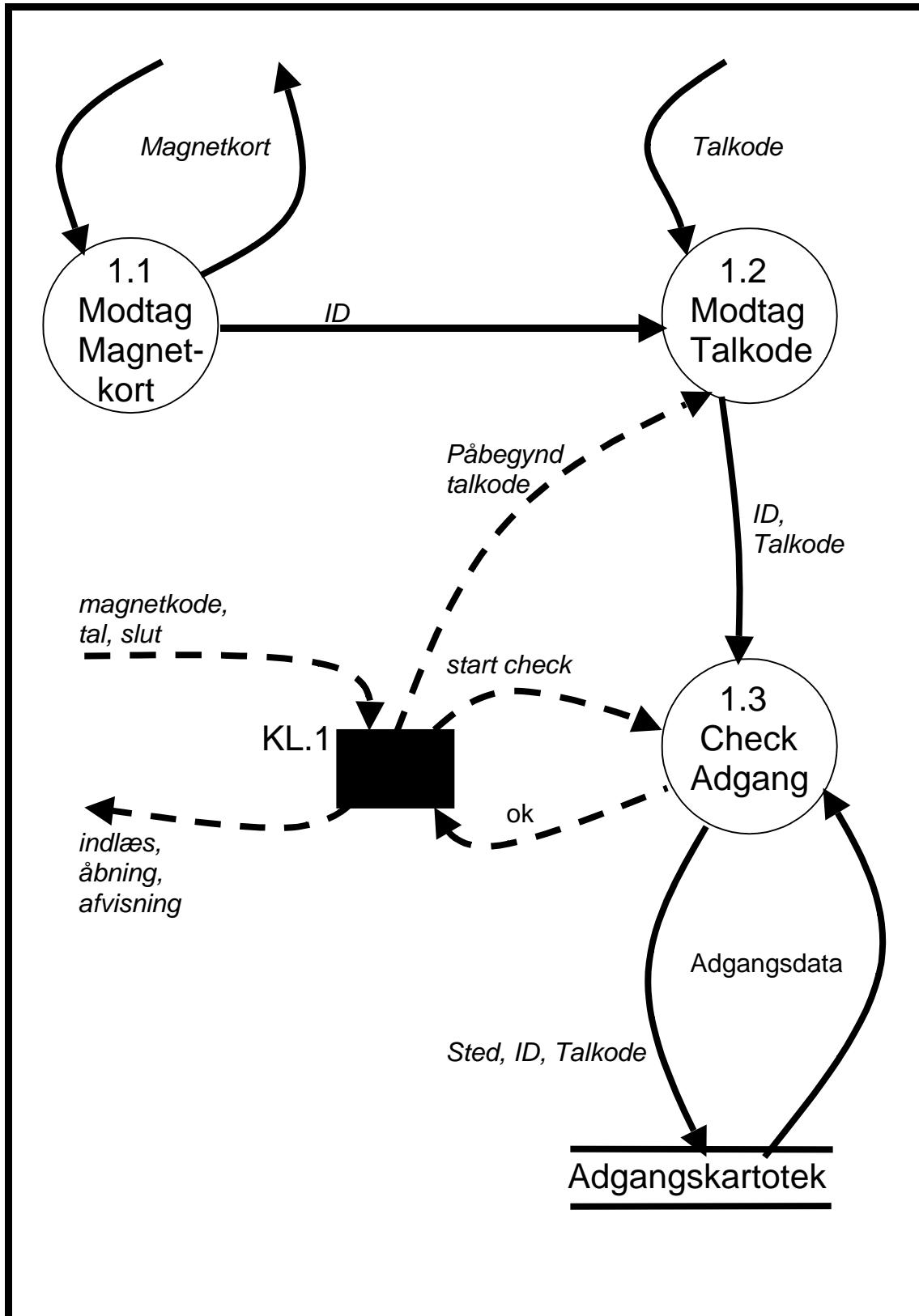
Heller ikke *OK?* er et rigtigt dataflow men snarere et flag der kan være sandt eller falsk - dvs. at *OK?* Også er et signal. Dermed bliver aktivitet 4.1 *Åben eller afvis* også overflødig, thi nu kommer der hverken data ind eller ud.

Til gengæld mangler vi noget der kan håndtere signaler. Derfor udvider vi DFD-notationens fire elementer med et femte, nemlig den tykke sorte streg, kaldet et kontrollager (engelsk: control store), som kan modtage, behandle og afsende signaler.

Vi opløser nu aktiviteten "Modtag magnetkort" i kontrollflow diagrammet niveau nul til et kontrollflow diagram niveau 1. Dette diagram er vist i figur 15.

Som du ser går alle signaler til og fra denne del af adgangskontrol-systemet nu vi kontrollager nummer 1, kaldet KL 1 i figuren.

Et eksempel på et nyt signal afsendt fra kontrollageret er signalet "Påbegynd talkode" fra kontrollageret til aktivitet 1.2 *Modtag Talkode*. Det der sker er, at signalet fra kontrollageret igangsætter aktivitet 1.2.



Figur 15: Kontrolflow-diagram, niveau 1.

## 9. Kontrolspecifikationer

Den opførelse man ønsker et kontrollager skal have, som reaktion på de indkommende signaler, specificeres i en såkaldt kontrol-specifikation, herefter kaldet CSPEC (Hatley & Pirbhai 1988).

I en CSPEC kan man repræsentere en endelig tilstandsmaskine (engelsk: finite state machine), hvadenten maskinen er kombinatorisk, dvs. uden hukommelse, eller sekventiel, dvs. at den husker sin tilstand og kan reagere forskelligt på samme signal afhængig af aktuell tilstand.

Repræsentationen i en CSPEC kan ske vha. forskellige diagrammer og tabeller. Nedenfor vil jeg gennemgå tre af dem anvendt på adgangskontrol-systemet til UNI.

### Tilstandsovergangs diagrammer

En meget anvendt teknik til repræsentation af sekventielle maskiner er tilstandsovergangs-diagrammet (engelsk: state transition diagram). Notationen består af fire elementer (Hatley & Pirbhai 1988: 81-82):

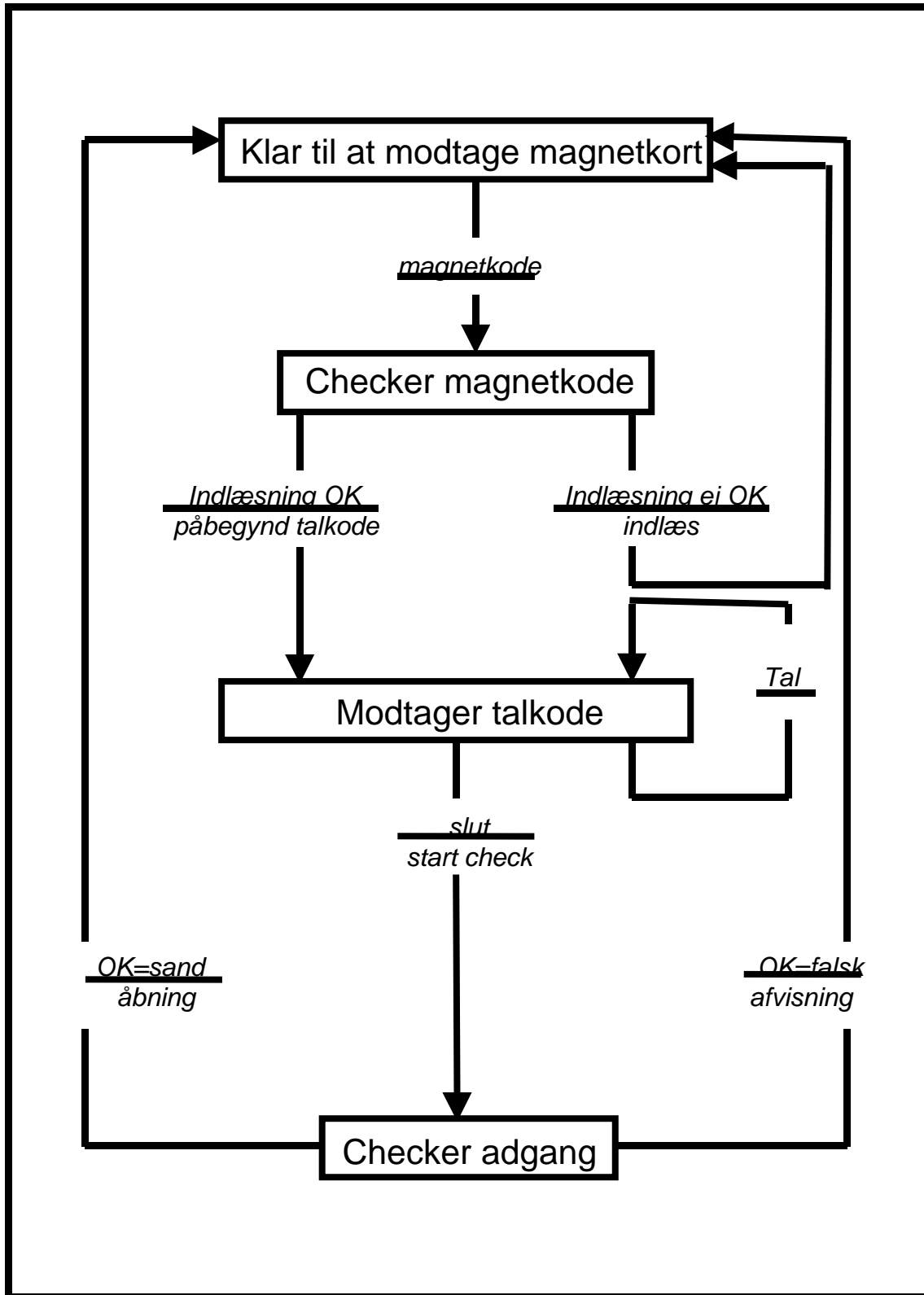
1. *Tilstande*, repræsenteret ved rektangulære kasser indeholdende navnet på tilstanden.
2. *Overgangs-pile*, dvs. pile hvis retning angiver retningen for en tilstandsovergang.
3. *Hændelser*, som forårsager tilstandsovergange, skrives som tekst til overgangspilene.
4. *Aktioner* der udløses af hændelser skrives også som tekst til overgangspilene. Konkret skrives hændelser over en brøkstreg og de tilhørende aktioner under samme brøkstreg.

I figur 16 er vist et tilstandsovergangs-diagram for kontrollager nr. 1 i figur 15.

Det man kan se i figur 16 er egentlig hele den indre logik i den automat der styrer adgangskontrollen, som f.eks. monteres i forbindelse med låsemekanismen på en dør.

Automaten har fire tilstande:

1. Klar til at modtage magnetkort. Når en magnetkode detekteres overgås til:
2. Check magnetkode. Her laves en simpel kontrol af, at vi læser en magnetkode der kunne være korrekt. Der kunne jo være tale om et galt magnetkort eller måske blot et stykke papir der for sjovs skyld blev kørt igennem magnetkortlæseren. Såfremt der er tale om korrekt magnetkode overgås til:
3. Modtager talkode. Her modtager vi et tal indtil der trykkes slut. Ved slut overgås til:
4. Checker adgang, der enten resulterer i en åbning af døren eller en afvisning. Herefter returneres til tilstand 1, klar til at modtage magnetkort, igen.



Figur 16: Tilstandsovergangs-diagram for kontrollager nr. 1 i figur 15.

### Procesaktiverings tabeller

Det vi ikke kunne se i tilstandsovergangs-diagrammet, figur 16, var den præcise sammenhæng mellem de aktioner der stod under brækstrøerne og de processer der skulle igangsættes. Til at illustrere det kan man bruge en såkaldt procesaktiverings tabel.

I figur 17 er vist en procesaktiverings tabel for adgangskontrol-systemets kontrollager nr. 1. Det man kan se i figur 17 er, at når aktionen (kontrolsignalet) "påbegynd talkode" sendes, så skal proces 1.2, "Modtage Talkode" sættes i gang. Og når aktionen "start check" sendes så skal proces 1.3 i gang.

Notation i figuren:

- 1 Betyder : Start proces
- 0 Betyder: Gør ingenting

Kontrol-signal \ Proces der aktiveres	1.2 Modtag Talkode	1.3 Check Adgang
<i>påbegynd talkode</i>	1	0
<i>start check</i>	0	1

Figur 17: Proces-aktiverings-tabel for kontrollager nr. 1.

### Tilstands/aktions tabeller

En anden måde at repræsentere næsten samme information som i tilstands-  
overgangs diagrammet (figur 16) og procesaktiverings-tabellen (figur 17) tilsammen, er  
tilstands/aktions-tabellen.

Denne tabel har 4 kolonner der læses fra venstre mod højre. Første kolonne er  
nuværende tilstand. Anden kolonne er lovlige hændelser der kan indtræffe i nuværende  
tilstand. Tredje kolonne er de processer der skal sættes i gang som følge af pågældende  
hændelse i nuværende tilstand. Og fjerde kolonne er den tilstand systemet skal overgå til  
efter udførelsen af aktionene nævnt i tredje kolonne. I figur 18 er vist et eksempel for  
kontrollager 1 i adgangskontrol-systemet.

De tre diagrammer/tabeller; tilstandsovergangs-diagrammer, procesaktiverings-tabeller  
og tilstands/aktions-tabeller, er eksempler på indholdet af en kontrol-specifikation  
knyttet til et kontrollager.

Andre eksempler er tilstandsovergangs-tabeller (Hatley & Pirbhai 1988: 82-85) eller  
kontrollogik-tabeller (Hatley & Pirbhai 1988: 89-91), som jeg dog ikke vil gennemgå  
her.

## 10. Afrunding

Der er selvfølgelig meget mere at designe før adgangskontrol-systemet er helt klar til at  
programmere. Hvordan skal brugergrænsefladen se ud? Skal et fejlindlæst magnetkort  
medføre at en lille rød knap lyser, eller skal vi lave et minidisply som på en typisk  
mobiltelefon og skrive responsen der? Hvad med DFD niveau 1 og 2 for de andre  
processer i systemet? Osv. osv.

Jeg vil dog stoppe her. Vi har været igennem de centrale principper for design, vi har  
diskuteret hvad et god design er, og alle de vigtigste teknikker, både data-, flow- og  
hændelses-orienterede i det man kalder *struktureret designer* gennemgået, så fra en  
pædagogisk synsvinkel er dette det rigtige tidspunkt at stoppe

Nuværende tilstand	Hændelse	Aktion der igangsættes	Ny tilstand
Klar til at modtage magnetkort	magnetkode	0	Checker magnetkode
Checker magnetkode	indlæsning OK	påbegynd talkode (1.2)	Modtager Talkode
	indlæsning ej OK	indlæs (prøv igen)	Klar til at modtage magnetkort
Modtager talkode	tal	0	Modtager talkode
	slut	start check (1.3)	Checker adgang
Checker adgang	OK=sand	åbning	Klar til at modtage magnetkort
	OK=falsk	afvisning	Klar til at modtage magnetkort

Figur 18: Tilstands/aktions-tabel for kontrollager nr. 1 i figur 15.