SISU rapport

nr 13

Objektorientering– de vanligaste begreppen

En introduktion för den oinsatte

Stig Berild

SISU
Svenska Institutet för Systemutveckling
Box 1250, 164 28 KISTA

Objektorientering– de vanligaste begreppen

En introduktion för den oinsatte

ISSN: 0282-9924

Innehåll

1. Inledning

Systemutveckling är idag ofta en långdragen och dyrbar process. Trots detta är det inte säkert att resultatet svarar upp mot berättigade förväntningar, snarare är motsatsen en regel. Systemspecifikationen återspeglar föråldrade förutsättningar eller saknar tillräcklig förankring bland de olika kategorier användare som kommer att bli berörda av systemets funktion eller den informationshantering som produceras. Implementering görs i konventionella språk och enligt en metodik som gör systemen svåra att ändra eller vidareutveckla. Förvaltning av system har blivit en dyrbar huvudvärk.

Både systemutvecklingsprocessen och dess resultat präglas av hög ålder och fårat ansikte - inte av den väderbitna, sunda typen - utan snarare av mycket bekymmer och nattvak i rökfyllda rum. Behovet av ett så kallat paradigmskifte, ett livselexir, är trängande. Inte undra på att metoder och verktyg som, likt en undergörande hudkräm, påstår sig ha djupverkande effekter, får ett omedelbart intresse. Förhoppningar och förväntningar övergår inte sällan i besvikelser.

Begreppet objektorientering (OO) har för många blivit liktydigt med en förhoppning om koncisare och mer användarnära systemutveckling, ökad produktivitet under implementering och ökad underhållsvänlighet. Är detta återigen en dagslända eller äntligen något långsiktigt trovärdigt? Det undergörande kanske snarare är att se som en vägledning i ett sunt levnadssätt (metod) tillsammans med några förebyggande näringspreparat (verktyg), gärna från naturens egen örtagård (verklighetsnära modeller)?

Inom området objektorienterad programmering finns mycket och övervägande positiva erfarenheter redovisade från både små- och storskaliga projekt. Där finns ett flertal tillgängliga och populära objektorienterade språk, metodik, årliga konferenser m m.

Det är dock för tidigt att bedöma om objektorientering passar alla faser eller bara en viss fas i systemutvecklingen, om det passar alla typer av tillämpningar eller bara en viss kategori, om det alltid är ett steg i rätt riktning eller om det kan ge upphov till nya typer av problem osv. Objektorientering inom de tidiga faserna av systemutvecklingsarbetet är ännu till stora delar ett oskrivet kapitel, vilket skapar grogrund för många åsikter och idéer. Erfarenheterna är begränsade.

Anhängare tenderar att se alla problems slutliga lösning, medan skeptikerna bara ser luftslott eller återuppfinning av redan etablerad kunskap inom andra områden (t ex databaser, datamodellering och AI). Ofta brukar så småningom den successivt tillägnade erfarenheten visa att "sanningen" ligger någonstans mittemellan. Att behovet av en intensiv spridning av kunskap och erfarenheter tillsammans med en aktiv debatt i hösta grad är önskvärd, är däremot lättare att komma överens om.

Syfte med denna rapport

Ett antal begrepp och mekanismer har kommit att kopplas till området objektorientering. I allmänhet förklaras de med hjälp av några "kodsnuttar" i ett befintligt eller för ändamålet påhittat programmeringsspråk, ett språk som läsaren utan pardon förutsätts bekant med. Vi tror att det finns ett behov av att förklara dessa begrepp även för personer som inte är bevandrade i något programmeringsspråk, speciellt om objektorientering ska komma att influera även de tidiga faserna av systemutvecklingen.

Syftet med denna rapport är att för den oinsatte läsaren introducera några av de grundläggande begreppen och att förklara bakomliggande mekanismer med hjälp av ett enkelt exempel. Innan vi tar itu med exemplet i avsnitt 3 beskriver vi ett par "upprinnelser". Först kommer en kort förklaring till uppkomsten av denna rapport. Därefter, i avsnitt 2, ges en kort sammanfattning av objektorienteringens korta historia. Rapportens sista avsnitt, avsnitt 4, diskuterar något om tillämpningsområden för objektorientering. Tanken är att avsnittet ska vara en intresseväckare för en diskussion kring i vad mån den administrativa tillämpningsvärlden har något behov av objektorientering. Diskussionen kommer att redovisas som ett debattinlägg i en efterföljande rapport. Bl a kommer där ett mer stringent resonemang kring begreppet "objekt" och "objektmodell" att efterlysas.

Kompetensnät för objektorienterad systemutveckling

För att få ett bättre grepp om området objektorienterad systemutveckling startade och drev SISU våren 1989 ett Kompetensnät kring Objektorienterad Systemutveckling (KOS) tillsammans med ett 15-tal deltagare från intressentföretag. Ett kompetensnät är en studiegrupp där varje deltagare deltar på lika villkor genom att redovisa referat från lästa artiklar och böcker, utbyta erfarenheter, diskutera problem osv. KOS gästades även av ett antal inbjudna experter.

Någon djupare erfarenhet fanns av naturliga skäl inte hos KOS-deltagarna. De flesta var med för att lära mer. Konkreta problem att diskutera förekom av samma anledning sparsamt. Gruppen enades om att mellan sig dela upp den existerande floran av böcker, rapporter, konferenspapper, tidskriftsartiklar för enskild genomläsning och därpå

följande redovisning och diskussion inom gruppen. Efter en inledande genomgång av objektorienteringens grundläggande idé och etablerade begreppsapparat skulle fokus ligga på objektorienterad *systemutveckling*.

Lika överraskande som omgående visade det sig vara synnerligen svårt att få fram material att läsa inom det valda temat. Material om objektorienterade språk, objektorienterad programmering, objektorienterade databaser fanns i övermått. I den mån de tidigare utvecklingsfaserna fanns belysta visade det sig inte sällan vara en snabb användning av det nya "innebegreppet" *objektorientering* på gammal etablerad kunskap inom konventionell systemutveckling och datamodellering. Litteraturlistan fick lov att kompletteras med visst centralt material kring objektorienterade språk och objektorienterade databaser. På så vis kom KOS att delvis ändra inriktning till att stå för en mer allsidig genomlysning av området.

Debatt

En försiktig, tyst men ändå spirande förhoppning fanns nog hos de flesta av oss deltagare att ett objektorienterat synsätt skulle kunna bli det länge efterfrågade kittet, den röda tråden mellan systemutvecklingens olika faser, som hittills saknats. Med objektorienteringsglasögonen på näsan skulle man få ett enhetligt sätt att se på problemområdet och ett enhetligt språk. Det visade sig dock snart att verksamhetsutvecklarens objekt är något annat än systemutvecklarens. Databaskonstruktörens objekt överenstämmer inte med programmerarens osv. Behöver det vara så eller tillhör framtiden objektet?

Till synes likadana glasögon gav ändå inte samma bild av verkligheten.

Tiden är alltså ännu inte mogen för att heltäckande beskriva en objektorienterad systemutvecklingskedja. Däremot är tiden mogen för en intensifierad debatt.

Brasklapp

Observera att vad som skrivs i denna rapport endast kommit från författarens penna och inte på något sätt representerar KOS-gruppens åsikter. Däremot bär deltagarna det fulla ansvaret för att ha inspirerat mig till ett aktivt intresse inom ett turbulent område och för att ha bidragit med kunskaper i denna min, för inte så länge sedan, totalt vita fläck.

2. Återblick

Traditionell systemutveckling har länge präglats av den så kallade "vattenfallsprincipen", vilket innebär att utvecklingen sker i steg som inordnas efter varandra. Resultatet från ett steg blir förutsättningar för nästa steg osv. Numer strävar man efter flexiblare disposition av arbetet, med experimentella inslag, parallella steg, modifieringar av tidigare specifikationer m m. Dock gäller fortfarande att administrativa system är dyra och tidsödande att utveckla samt svåra att underhålla. Ofta är användaracceptansen låg. Förhoppningar ställs på

- bättre modeller och metoder
- moderna utvecklingsmiljöer, typ 4G-system och Case-verktyg
- idén kring öppna system
- enhetliga utvecklingsplattformar, typ så kallade repositories mm

Problemen med dessa förhoppningar kan vara att de

- ännu är på idéstadiet eller åtminstone oprövade.
- ännu inte fått ett markant genomslag på marknaden.
- inte följer trenden inom internationellt standardiseringsarbete.
- inte klarar realistiskt stora applikationer (där behovet är störst).

I ett sådant perspektiv är det inte underligt att nya trender tilldrar sig stort intresse. Märkligt nog är inte det objektorienterade synsättet något nytt. Det nu snart 25-åriga programmeringsspråket Simula anses vara den egentliga upprinnelsen. Simula användes under sina första år som ett rent simuleringsspråk, vilket kan förklara dess ringa spridning. Antagligen var programmerarvärlden vid den tidpunkten inte heller mogen att ta till sig en helt ny modell för att bygga system.

Vid samma tid började behovet av implementeringsoberoende specifikationer av data och dess hantering att göra sig gällande. Området data- och begreppsmodellering började ta form, inte minst genom ett tidigt svenskt intresse för området (Langefors, Bubenko, Sundgren). Objektbegreppet har i dessa sammanhang sin centrala plats. Det vid samma tid uppflammande intresset för relationsmodellen gav under en tid tyvärr inte nämnvärt utrymme för de mer objektorienterade ansatserna.

Programmeringssidan och modelleringssidan levde länge sina parallella liv utan att veta om eller ta speciellt mycket intryck av varandras existens. Flera programmeringsspråk utvecklades, t ex Smalltalk och LISP-baserade språk som Flavors. På senare tid har bla C++ och Eiffel blivit väl mottagna av marknaden. Modelleringssidan fick sin ER-modell, sin NIAM-modell, mfl snarlika.

Först vid mitten av 1980-talet började områdena närma sig varandra, dock inte som en ömsesidig handling. Intressenter för objektorienterad programmering började vidga vyerna mot de faser som föregår programmering. I visionen låg en mer integrerad syn på systemutveckling. Objektmodeller skulle skapas redan i analysskedet och ges successivt mer detaljerad form under designskedet, för att sedan mer eller mindre omärkligt övergå i en implementering i ett objektorienterat språk. Tyvärr verkar ännu inte den existerande kompetensen kring metoder och modeller för systemanalys och design ha mött kompetensen inom objektorienterad programmering, vilket gjort att onödigt många "hjul uppfunnits på nytt". En viss begreppsförvirring har dessutom uppstått. Gamla företeelser har fått nya beteckningar, gamla beteckningar har fått ny betydelse osv.

Vi befinner oss nu i ett läge där det är hög tid att precisera objektorienteringens innebörd och mekanismer. Kommer förväntningarna att kunna uppfyllas och i så fall till vilken grad?

3. Grundläggande begrepp

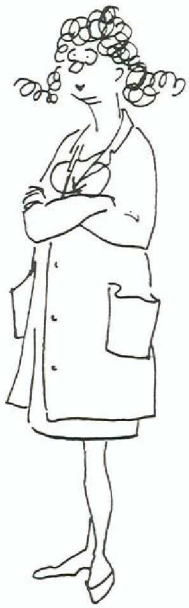
De flesta papper om objektorientering förutsätter en grundläggande kompetens inom programmeringsområdet. Det har därför ansetts naturligt att även hämta exempel från samma område. Vi föreställer oss att det bland de objektorienterings-intresserades skara även finns de som inte har någon, har mycket liten eller ganska gammal sådan kunskap. För dessa har vi valt att introducera begreppen med hjälp av ett högst vardagligt exempel, som till att börja med ligger långt ifrån systemutvecklingens bistra verklighet. Den datakunnige läsaren uppmanas ha överseende. Till att börja med är diskussionen förvillande lik en introduktion till datamodellering, vilket inte är förvånande eftersom det finns stora likheter och överlappningar mellan områdena. Så småningom blir diskussionen mer specifikt inriktad på objektorientering och mer datanära.

3.1 Verklighet, modell, objekt

Antag följande scen ur "verkliga livet".

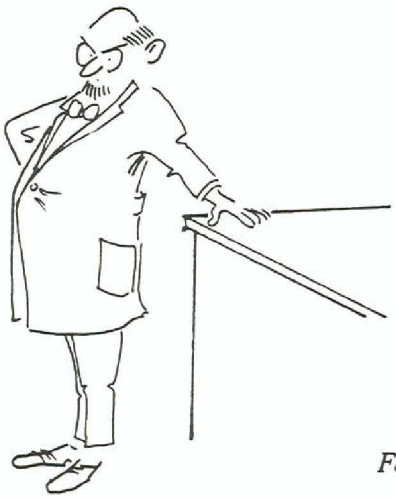
> ***När lille Pelle hjälpte mormor köpa mat i handelsboden***
>
> *Mormor var gammal och hade svårt både att skriva och att läsa. En dag bad mormor Pelle om hjälp med att skriva en lista över de varor som behövde köpas in. Mormor tittade i skafferiet och Pelle skrev. Pelle var stark för sin ålder (12 år), men hade hål i byxfickorna och var allmänt lite slarvig. Därför gjorde de sällskap till handelsboden, Pelle för att läsa listan och bära hem varorna, mormor för att kontrollera kvalitet på färskvarorna och för att betala.*

Handelsboden hade tre expediter (fru Lamm, herr Spik och fröken Ny), förutom föreståndaren. Fru Lamm hade huvudansvaret för kött och fisk, herr Spik för järn och färg. Fröken Ny hade just börjat och fick hugga in där det behövdes. Alla fyra expeditierade kunder, men när det kom till att fru Lamms kund skulle ha "tvåtums förzinkad dyckert" så var det bara att ropa på herr Spik. Han var den ende som visste var dessa fanns. På samma sätt var fru Lamm den enda som kunde stycka köttbitarna snyggt. Handel på krita beviljades bara av föreståndaren och bara om förra månadens hela skuld var reglerad.

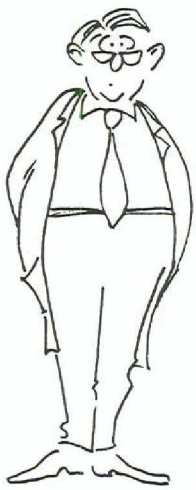


Fru Lamm

Fröken Ny



Föreståndaren



Herr Spik

Idag blev herr Spik mormors expedit. Utan att gå in på detaljer fick mormor sina varor (bl a en bit rökt skinka) och betalade, inklusive förra månadens skuld. Pelle släpade och bar, inte utan tanke på godispåsen, som låg och prasslade bland varorna.

Antag att vi vill beskriva eller avbilda denna verklighet i något syfte. Kanske funderar en arbetsgrupp inom Handelsbodsföretagareförbundet på att ge ut en ny upplaga av "Du och din kund". Inför detta behöver det bl a kartläggas hur vissa typiska handelsbodskunder tas om hand idag. Arbetsgruppen tittar för sitt syfte bara på de aspekter som är intressanta för kundrelationen. Förbundets rationaliseringsexpert, å andra sidan, skulle sannolikt aldrig ens notera mormors existens i sin iver att lokalisera sammanfallande beteende vid fisk- och köttdiskarna inför en möjlig sammanslagning av dem.

Syftet kan ju även vara att åstadkomma en rimligt enkel beskrivning av en situation som kan ligga till underlag för att introducera några centrala objektorienteringsbegrepp. Scenen ur "det verkliga livet" har med andra ord utgjort underlag för tre olika modeller, var och en med sitt syfte och sina avgränsningar.

Att avgränsa det intressanta är ofta betydligt svårare än denna beskrivning låter påskina. Det är viktigt att ta med allt av betydelse men utelämna allt annat, dvs det som inte tillför bilden något för det aktuella syftet. Annars blir modellen för rörig eller oöverskådlig, och kanske inte användbar. Det gäller också att de personer som ska jobba med modellen är någotsånär överens om hur den ska tolkas för att undvika missförstånd. Här gäller det att nyttja ett språk som är stringent och begripligt.

Modellen som används för att introducera objektorienteringsbegrepp är ovan uttryckt på vanlig svenska, ett språk som varje läsare antas vara bekant med. Stycke ett och tre beskriver händelseförloppet medan det andra stycket berättar lite om gällande regler mellan butikens personal. Arbetsgruppen kanske istället skulle gjort ett stillbildsscenario om 36 teckningar. Rationaliseringsexperten kan välja att beskriva handgreppen i form av ett flödesdiagram i form av boxar och pilar.

> **modell:** En modell är en avbildning i vissa former av någonting (verkligheten) där vissa aspekter har valts ut i något syfte så att modellen blir användbar på åtminstone ett sätt. (Formulerat av Hans Willars, SISU)
>
> Motsvarande engelska begrepp: *model.*

I modellen finns ett antal utskiljbara företeelser som gör saker och som tar kontakt med andra företeelser. Det hela pågår över en begränsad tidsperiod. Företeelser av olika slag kallar vi i fortsättningen för **objekt**. Den första kontakten med objektorienteringsansatsen är upprättad.

> **objekt**: Ett objekt är någonting i en modell som upplevs ha en motsvarighet i verkligheten, det kan vara en förenklad avbild av något avgränsbart i verkligheten. Ett objekt har egenskaper, kan göra saker och befinna sig i vissa tillstånd. Ett objekt svarar mot en unik punkt i modellen, dvs har en unik identitet.
>
> Motsvarande engelska begrepp: *object, entity.*

Pelle är ett typiskt objekt, någonting vi iakttar i samband med ett inköp. Pelle är stark och 12 år (egenskaper). Han skriver listor, bär och tänker på godis (beteenden). Just nu kanske han väntar (tillstånd) medan mormor betalar. Att Pelle tycker om att cykla och att han är allergisk mot maskrosor, är ointressant i denna modell. *Vår* Pelle är en annan än granngårdens busfrö, som också heter Pelle, de har olika identitet. Om det inte finns risk för missuppfattningar räcker det att ropa på Pelle och rätt Pelle svarar. Skulle de båda samtidigt vara i affären, vilket ju ibland händer, måste man ropa på *bus-Pelle* eller *Pelle-stark* .

Förutom Pelle finns i modellen även objekten mormor, herr Spik och varorna som mormor köper. Lite vid sidan om i det här läget finns föreståndaren, fru Lamm, fröken Ny, men även listan på varor, mormors pengar och förra månadens skuld. Av detta kan vi se att objekt inte bara behöver vara sådant man kan ta på (konkret). Skuld, t ex är ju en slags överenskommelse (får man hoppas) och som sådan ganska abstrakt. Skulden kan även uppfattas som ett permanent konto med ett givet dagssaldo (tillstånd). Båda uppfattningarna är rimliga. Vilken man väljer beror på vad iakttagarna av situationen uppfattar som naturligt och vettigt, hur det i praktiken fungerade för det behov modellen ska tillgodose. Listan kan ju finnas på en papperslapp, men skulle lika gärna kunna befinna sig i Pelles huvud. En annan skiljelinje är att Pelle, mormor m fl företar sig olika saker, medan både listan och skulden bara är passiva databärare (om man inte ser skulden som en hårdför indrivare).

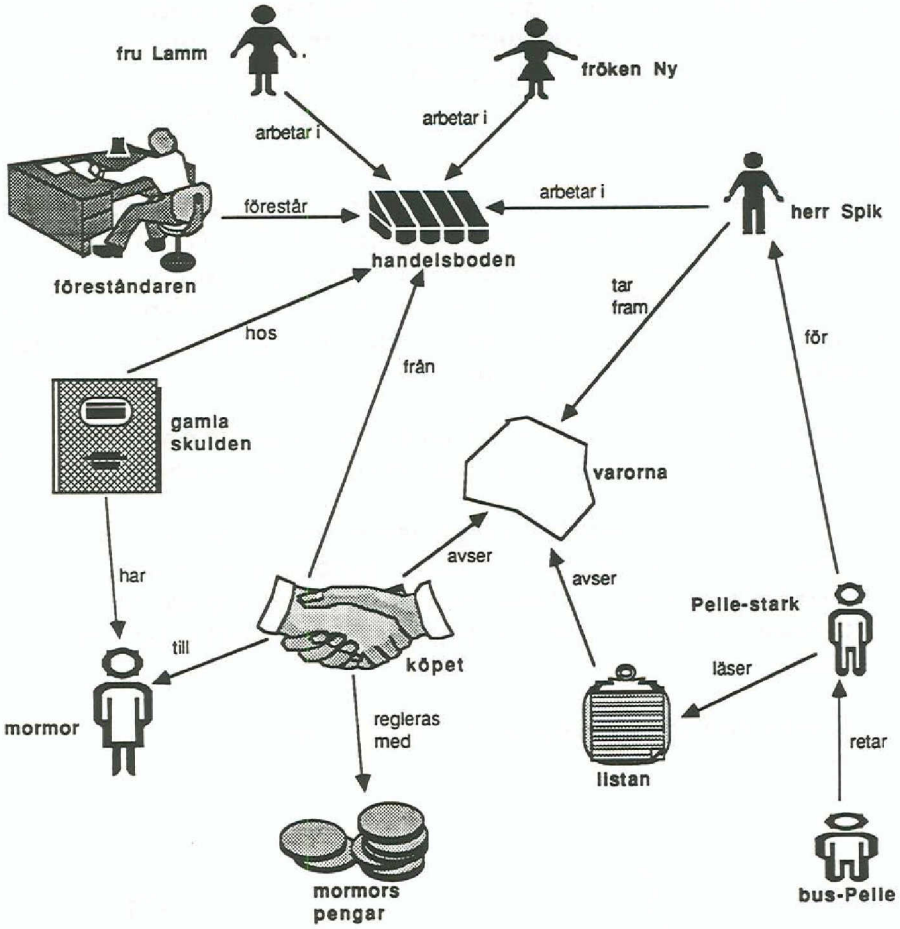
Vi vet nu lite mer om vad ett objekt är och kan därför precisera dess definition till

> **objekt**: Se definition på föregående sida med tillägget:
>
> Förutom att objekt kan avbilda såväl konkreta som abstrakta företeelser ur verkligheten kan de också uppfattas som passiva och aktiva.

Följande objekt har hittills påträffats:

Pelle, mormor, föreståndaren, herr Spik, fru Lamm, fröken Ny, lista, varor, mormors pengar och gammal skuld.

Varför inte åskådliggöra dessa objekt i en grafisk modell? Se figur 3.1. En grafisk presentationsteknik kommer att underlätta förståelsen av vad de objektorienterade begreppen står för i den fortsatta diskussionen. Den uttrycker i stort sett samma sak som den tidigare verbala texten (förutom en del regler ur stycke två, som vi återkommer till) men kanske på ett mer lättillgängligt sätt. Utan någon mer omfattande specifikation av det grafiska språket kan man förhoppningsvis uppleva att de grafiska symbolerna står för objekt medan pilarna står för någon typ av namngivet intressant samband eller förhållande mellan objekt.



Figur 3.1

3.2 Skillnaden mellan identitet och referens

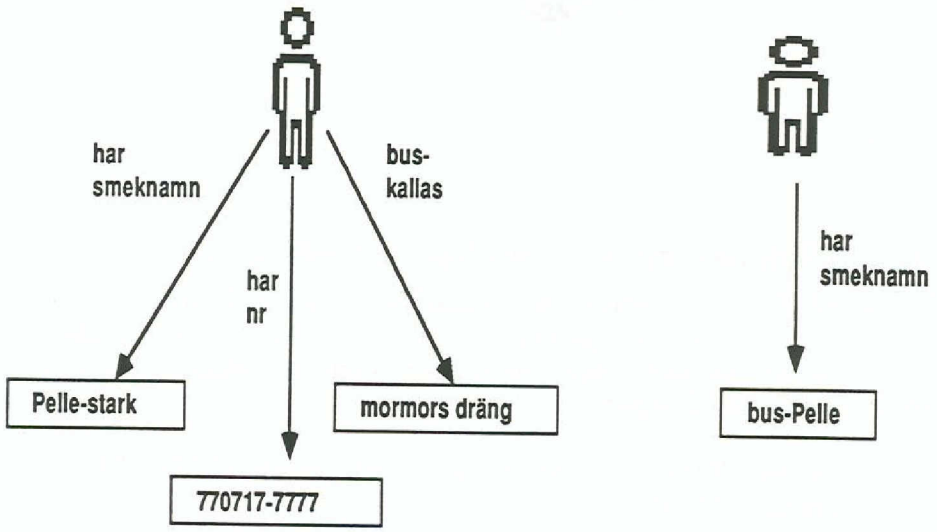
Vi har också noterat behovet av att uppleva varje objekt som unikt, dvs åtskilt varje annat objekt (se t ex bus-Pelle och Pelle-stark ovan). Denna åtskillnad är lika viktig både i verkligheten och i modellen. Varje objekt sägs ha en unik identitet.

> **objektidentitet:** Objektidentitet är det som särskiljer ett objekt från varje annat objekt.
>
> I ett informationssystem förverkligas detta ofta genom en av ett stödsystem (t ex en databashanterare) internt genererad kod. Koden behövs för korrekt intern hantering och har i allmänhet ingen giltighet för systemanvändare, dvs objektidentiteten är oberoende av objektets egenskaper.
>
> Motsvarande engelska begrepp: *identity, surrogate.*

I verkligheten är det inte så svårt, så länge vi tittar på fysiska företeelser (bilar, personer, artiklar). Det blir genast mer besvärligt när det gäller de abstrakta objekten. De senare är inte lika påtagliga eller påtagligt avgränsade, inte lika enkelt iakttagbara. I en modell brukar man ge objektet en kod, symbol eller liknande som följer objektet under hela dess levnad och som inte har något annat syfte än att vara unikt i modellen.

Däremot är det fullt möjligt att peka på eller referera till ett och samma objekt på flera sätt. Pelle har personnumret 770717-7777. Vi känner alltså till åtminstone två olika sätt att referera till just honom: *Pelle-stark* och *770717-7777*. *Pelle* räcker inte eftersom den referensen även kan avse *bus-Pelle*. För övrigt kan avslöjas att *bus-Pelle* för att retas använder en tredje referens på vår Pelle, nämligen *mormors dräng*.

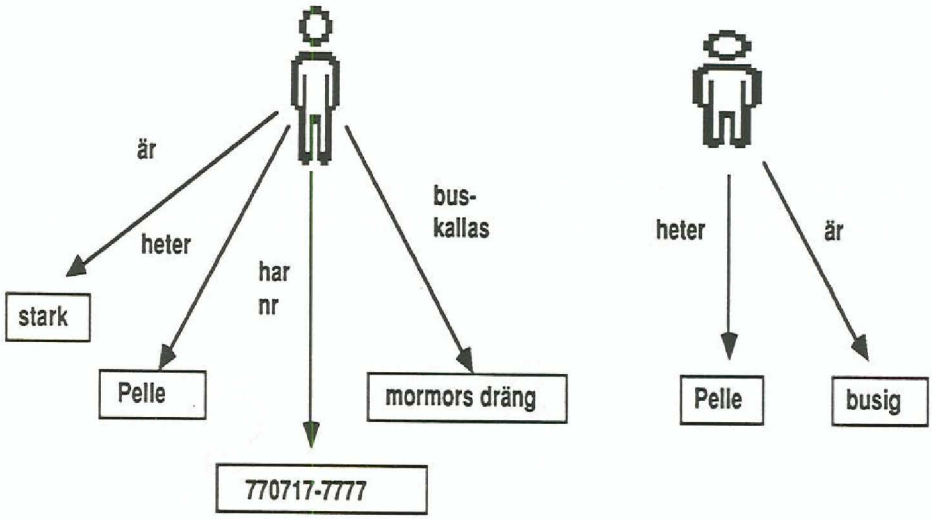
> **objektreferens:** Varje kombination av registrerbara förnimmelser eller egenskaper som ger en unik association till ett visst objekt.
>
> I informationssystemsammanhang är referensen vanligtvis en överenskommen symbol eller kombination av symboler som pekar ut ett visst objekt. Symbolerna består traditionellt av text (ord eller tal). I framtiden kan vi förvänta oss även grafiska symboler, bilder, ljud eller video.
>
> Motsvarande engelska begrepp: *object reference*.



Figur 3.2

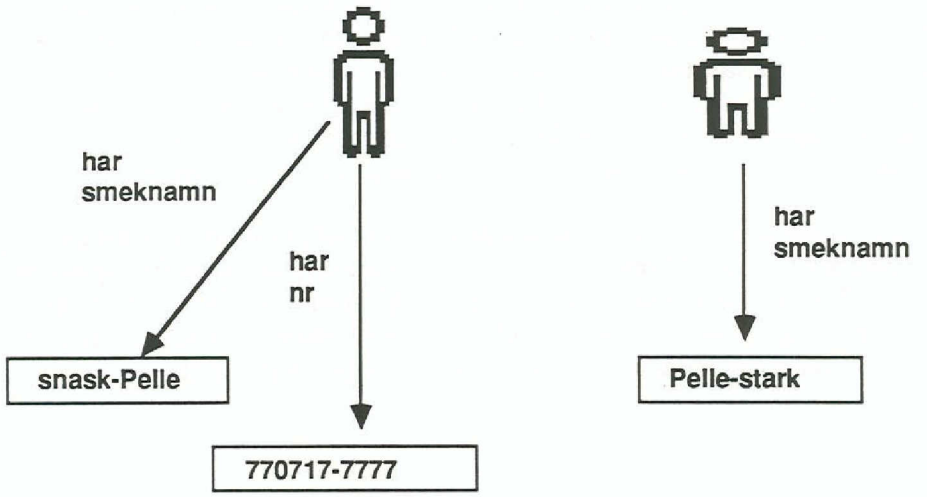
I figuren kan vi uppleva den grafiska symbolen som identiteten, medan texterna i rutorna är referenser. Ett alternativt men lika acceptabelt sätt att se på samma sak visas i figur 3.3. Smeknamnet är borta, båda heter rätt och slätt Pelle. Att de är starka eller busiga ligger som en separat egenskap. För att referera till en unik pojke behövs är- och heter egenskaperna tillsammans. Vi har en sammansatt referens.

Den vaksamme läsaren invänder genast med påpekandet att det egentligen i det här fallet borde räcka med att bara använda *stark* respektive *busig*. Vad som dock inte visas i det här urklippet av modellen är att även *herr Spik* beskrivs som *stark*. Med andra ord räcker inte enbart *stark* och inte heller bara *Pelle*, däremot *stark* och *Pelle* i kombination.



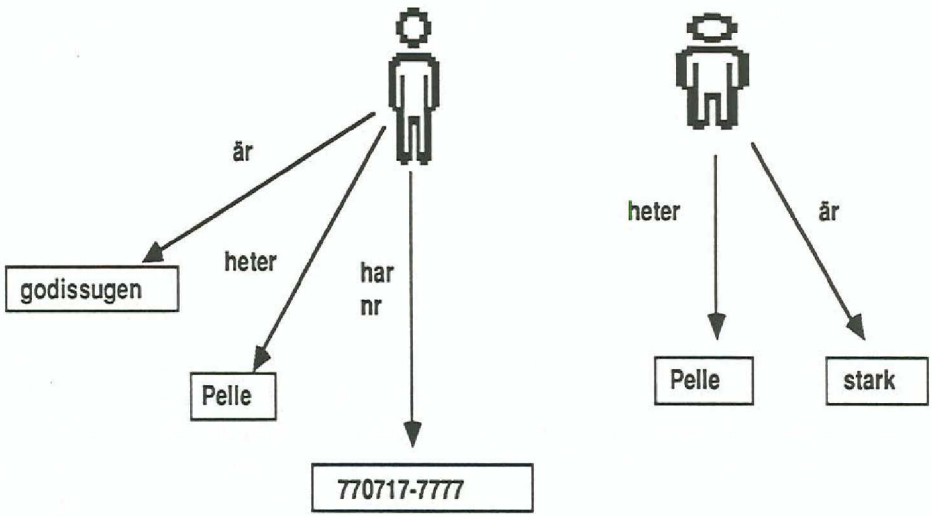
Figur 3.3

En av finesserna med att skilja mellan identitet och referens är att man kan ha flera olika referenser (för olika passande förutsättningar) till samma objekt. Referensen har en innebörd, en betydelse för modellens användare vilket inte identiteten har. En annan finess är möjligheten att kunna komma överens om nya eller ändrade referenser till samma objekt. T ex kan Pelle om något år ha fått sådan smak på godis att han nu kallas *snask-Pelle*, medan *bus-Pelle* blivit en trevlig prick som lagt av att retas med *mormors dräng* och vuxit till sig så till den grad att han fått överta smeknamnet *Pelle-stark*. *Bus-Pelle* används överhuvudtaget inte längre. Se fig 3.4 resp 3.5.



Figur 3.4

En unik referens är ju egentligen inget annat än en tillräcklig uppsättning egenskaper som tillsammans återfinns hos bara ett enda objekt.



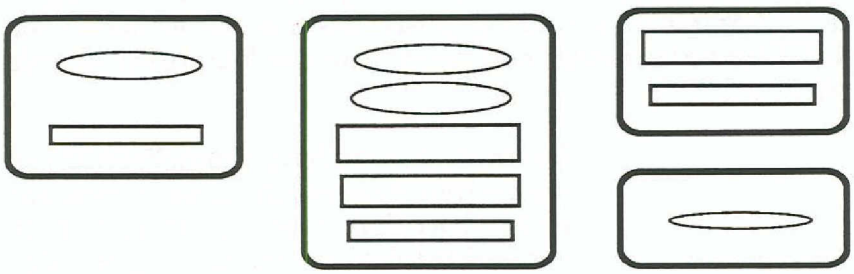
Figur 3.5

Tills vidare kan vi sammanfatta: En referens (och även en del av en referens om den är sammansatt) är en egenskap hos ett objekt. Det kan även finnas andra egenskaper av intresse även om de inte används i referenser. Till exempel kan det vara bra att känna till pojkarnas ålder, att de båda är 12 år. Vi har också konstaterat att objekt kan ha

beteenden, t ex herr Spiks sätt att plocka fram varor. Därutöver har detta beteende oftast någon typ av påverkan på omgivningen, det finns samband mellan objekten.

3.3 Enhetliga grafiska symboler

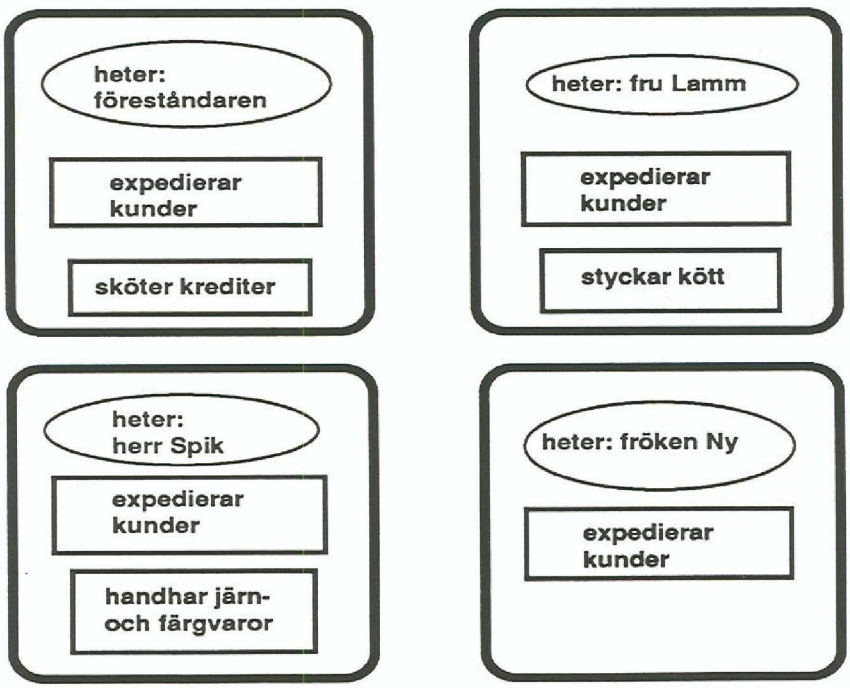
Låt oss med detta i bakhuvudet rationalisera bland floran av grafiska symboler till endast ett standardiserat principutseende för objekt. Se figur 3.6.



Figur 3.6 Exempel på grafiska symboler för objekt.

Ett objekt har en ram med rundade hörn. Inom ramen kan det sedan finnas egenskaper i form av ovaler och beteenden i form av rektanglar.

Den nya tekniken använd på de anställda i affären blir enligt figur 3.7.



Figur 3.7

3.4 Beteenden och dess inkapsling

Herr Spik bär på en besvärande hemlighet. Antalet varor inom järn har expanderat så till den grad att han omöjligen orkar hålla all information i huvudet. När mormor önskar ett dörrhandtag av typen 'Skultuna, allmoge, typ C', harklar sig herr Spik och säger att "hörrödu, jag ska ta och se efter på lagret, ett ögonblick". I själva verket rusar han in på kontoret för att slå upp i Skultuna-katalogen hur de ser ut, därefter ut på lagret för att leta på troliga ställen, förhoppningsvis hitta handtaget, tillbaka till liggaren för att slå upp priset, pusta ut bakom dörren och sedan lungt lämna över varan med orden "jaha, den här brukar jag vilja ha 72,50 för".

Hur modellerar vi detta? Antagligen är varken mormor eller någon annan intresserad av hur herr Spik beter sig för att få fram varan, bara det inte tar för lång tid. Den ende som måste känna till detta är herr Spik själv, annars hittar han ju inte varorna alls. "Rusa hit och dit i lagret", "slå i liggare" m m är interna egenskaper hos herr Spik (inkapslat i objektet), medan "överlämna vara, meddela dess pris" i högsta grad är av intresse för kunden. De senare är externa egenskaper. Figur 3.8 visar en kompletterad herr Spik.

Figur 3.8

Inkapsling: En teknik där ett objekts alla egenskaper och allt som ett objekt känner till, tillsammans med allt objektet kan göra, samlas *inom* objektet. Endast de egenskaper och beteenden som ligger i objektets *skal* är kända för omgivande objekt. Interna egenskaper och beteenden är helt objektets ensak och utanför omvärldens påverkansmöjlighet.

Det interna kan objektet fritt förändra vid behov så länge inte skalet påverkas. Förändringar av egenskaper och beteenden i skalet måste alltid ske i samförstånd med de objekt som påverkas av förändringen.

Objektets skal går normalt under benämningen specifikation eller gränssnitt medan det interna brukar benämnas realisering. Inom programmering innebär det att ett objekts data sammanförs med sina relaterade procedurer.

Motsvarande engelska begrepp: *encapsulation, information hiding.*

Tack och lov håller just ett lagerhanteringssystem på att installeras. Därifrån kommer man direkt att få uppgift om ifall varan finns i lager, dess pris och på vilken hylla den finns. När mormor nästa månad vill ha samma typ av handtag på andra sidan dörren hemma märker hon ingen som helst skillnad hos herr Spik. Han försvinner bakom dörren och dyker fram igen efter ett tag med handtaget i handen. Det enda synliga är att han är mindre rödblommig trots att det gått fortare den här gången.

Det interna beteendet har kunnat ändras utan att de externa egenskaperna och därmed omgivande objekt påverkats. Ändringen sker i realiseringen. Det synliga beteendet "handhar järn- och färgvaror" gäller fortfarande. En av de främsta egenskaperna hos inkapsling!

I strikt mening består ett objekts skal (gränssnitt mot omvärlden) endast av ett antal beteenden eller operationer. Varje kontakt med ett objekt består av en begäran om utförande av något tillgängligt beteende. I denna rapport har vi för åskådlighetens skull en något modifierad ansats. Efterfrågas en egenskap anser vi den direkt tillgänglig i skalet istället för via ett beteende som tar fram och levererar egenskapen. Därav behovet av både ovaler och rektanglar.

> **beteende:** En till visst objekt hörande service eller operation som är tillgänglig för andra objekt. Normalt utförs all kommunikation mellan objekt via beteenden, även sådana som i princip bara hämtar värden på objektets egenskaper.
>
> Motsvarande engelska begrepp: *method, behaviour, operation, action, service.*

3.5 Hur objekten umgås

Varje objekt har sitt skal i form av tillgängliga egenskaper och beteenden. När mormor vill köpa sitt handtag begär hon att herr Spik ska utföra ett beteende, nämligen *överlämna vara, meddela dess pris*. Skulle fröken Ny vilja veta Pelles ålder (egenskap), hänger svaret *13 år* bildligt som en lapp på hans mage att titta på. Uppdelningen i egenskaper och beteenden har för övrigt gruvligt irriterat Pelle när han försökt gå på barnförbjudna filmer. Tänk vad bra en situationsanpassad åldersvararprocess skulle ha varit istället. "Om biokassören frågar så svarar jag 15 år annars svarar jag 13 år".

Samverkan mellan objekt i form av *vilja veta, titta på, begära en service, osv* är en slags begäran som i objektorienteringssammanhang går under beteckningen **meddelande**. Vill föreståndaren att fru Lamm ska stycka kött, skickar han ett meddelande till fru Lamm om att *stycka kött*. Fru Lamm styckar den leverans som kommit från slakteriet idag. Sedan var den saken avklarad. Vill mormor få sin köttbit styckad skickar hon samma typ av meddelande till fru Lamm. Eftersom det inte är föreståndaren som begär servicen behöver fru Lamm veta vilken köttbit som ska styckas. Mormor måste lämna en kompletterande uppgift, en parameter till fru Lamms *stycka kött*. Tillbaka får mormor ett resultat i form av det styckade köttet. Resultat kallas i objektorienteringssammanhang för returvärde. Eftersom det sannolikt innebär delvis olika handgrepp att stycka dagsleveransen och att stycka lite kött till en kund vore det i realiteten bättre att dela upp i två fristående beteenden *stycka dagsleverans* och *stycka kundkött*.

När mormor skickar ett meddelande till herr Spik att han ska utföra sitt *överlämna vara, meddela dess pris*, överlämnas två parametrar för att jobbet ska bli rätt utfört, nämligen typen av vara och vilket antal som önskas. Tillbaka kommer varorna samt priset (får man hoppas). Resultatet av en operation behöver inte alltid vara ett returvärde. En tillståndsförändring hos ett objekt är också en form av resultat.

I det generella fallet kan en begäran ge upphov till att ett antal objekt blir kontaktade i tur och ordning, samtidigt eller enligt något annat mönster. Herr Spik skickar ju meddelande till objektet Skultuna-katalogen med begäran om en bild på handtaget. Det skulle inte förvåna om han efter avslutad kontakt med mormor också skickar ett meddelande till föreståndaren där han begär kafferast. Om bus-Pelle skulle skjuta en pappersmärla med slangbågen på fröken Nys smalben kan man bara ana vilken meddelandesändning som skulle bli följden.

> **meddelande:** En signal från ett objekt (**sändare**) till ett annat objekt (**mottagare**) med begäran om att ett beteende ska utföras. Ett meddelande består av tre delar: Vem som är mottagande objekt, beteckningen på aktuellt beteende samt de uppgifter (**parametrar**) beteendet behöver för att kunna utföra sitt arbete.
>
> Ofta resulterar en begäran i ett svar (**returvärde**). Svaret kan vara ett objekt eller något annat värde. I ett generellare resonemang kan man tänka sig svar omfattande flera värden(i strikt objektorienterade termer ≈ komplext objekt).
>
> Motsvarande engelska begrepp: *message.*

3.6 Klassificering

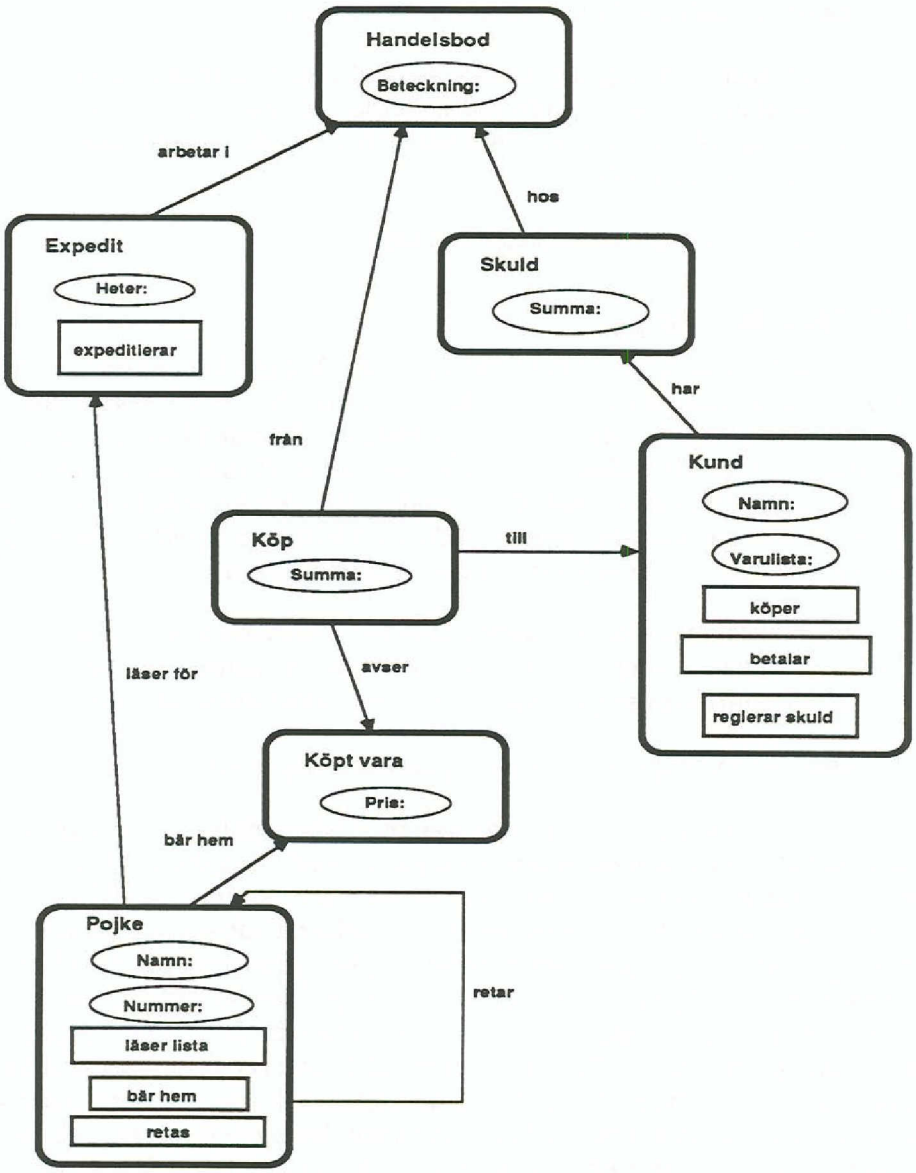
Figuren 3.7 presenterar de fyra expediterna. De expedierar kunder och gör lite annat. Redan i avsnittets första mening har vi grupperat dem under samma rubrik, nämligen som *expediter*. Därmed har vi gjort en abstraktion eller klassificering. Vi har fokuserat på väsentliga likheter och bortsett från ovidkommande olikheter. Var och en som svarar upp mot *mallen* anses tillhöra typen *expedit* eller **klassen** *expedit*, som man föredrar att kalla det i objektorienteringssammanhang.

> **objektklass:** Karakteristik (egenskaper och beteenden) som i något sammanhang överensstämmer för ett antal objekt så påtagligt att man hänför dem under en gemensam beteckning, under viss objektklass.
>
> Motsvarande engelska begrepp: *object class, object type.*

Ett närmare skärskådande av vår handelsbodsmodell ger vid handen att vi ser ett antal objektklasser, t ex vara, kund, expedit och köp. Det är en naturlig mekanism hos människan att abstrahera. Vi talar obehindrat om personer, bilar, tankar, modeller. Problemet är snarare att vi abstraherar olika och därför inte kan kommunicera om det så att andra förstår. Någon tycker att alla är expediter, medan en annan tycker att föreståndaren så sällan är ute i butiken att han får bilda en egen klass *föreståndare*. Mormor tycker att fröken Ny är en liten snärta som inte kan så mycket och föredrar uppfattningen att personerna bakom disken är av klassen *senior* och *junior*. En modell ska ses som en överenskommen kompromissuppfattning av verkligheten. En tänkbar kompromiss skulle kunna se ut som figur 3.10. Den grafiska symbolen

har kompletterats med objektklassens namn. Egenskapsvärden har av naturliga skäl utelämnats eftersom de varierar från objekt till objekt inom klassen.

Ibland talar man i objektorienteringssammanhang om *extension* och *intension*. *Extension* av en modell är en uppsättning objekt och deras samband. *Intensionen* är beskrivningsnivån av extensionen, dvs objektklasser och deras samband eller det vi i denna rapport normalt förknippar med beteckningen *modell*.

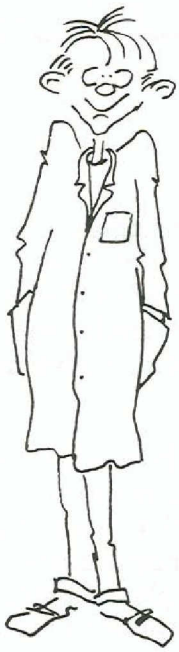


Figur 3.9

3.7 Specialisering och generalisering

Av figur 3.9 framgår att handelsbodar har expediter som anställda och att de kan expediera kunder. I själva verket vet vi mer om deras olika kvalifikationer. Låt oss nu skärskåda expediterna ytterligare. Helt klart är att alla fyra kan fungera i rollen som expedit. Föreståndaren ordnar därutöver krediter, herr Spik plockar fram varor från järnavdelningen medan fru Lamm gör motsvarande i köttdisken. Fröken Ny har ännu ingen specialuppgift. Se vidare figur 3.7.

Antag först att handelsboden expanderar. Ännu en expedit anställs, herr Torsk.



Herr Torsk är liksom fru Lamm duktig på kött och fisk. Båda antas i stort kunna samma saker. Beskrivningen av fru Lamms kvalifikationer och ansvar gäller inte längre enbart henne utan båda. Vi har fått två objekt som kan betecknas som *kött- och fiskexpert*. Aha, en ny objektklass! Varför inte löpa linan ut och tänka oss samma resonemang för de övriga även om de än så länge är ensamma i sina klasser. Föreståndaren tillhör kategorin *föreståndare*. Herr Spik tillhör kategorin *järn- och färgexpert*. Fröken Ny är *expedit*. Se figur 3.10.

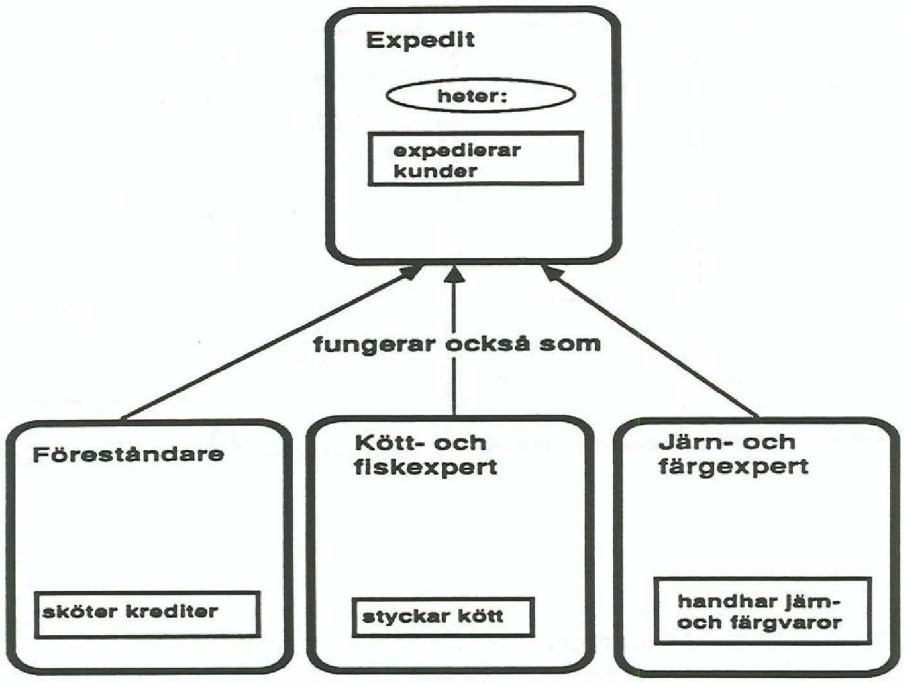
Figur 3.10

Fröken Ny känner en omedelbar glädje inför detta. De övriga har sina specialitéer, hon har liksom på samma nivå nu specialitén att vara expedit. Glädjen grumlas lika snabbt när föreståndaren myndigt förklarar att "även de övriga kan figurera i rollen som expedit. Det är inte fråga om att vara antingen det ena eller det andra, man kan ha mer än en roll i handelsbon".

Herr Torsk kan enligt figurens beteenderutor, beroende på aktuella omständigheter, antingen expediera kunder (rollen som *expedit*)" eller stycka kött (rollen som *kött- och fiskexpert*). Kunderna expeditieras på i stort sett samma sätt. Istället för att beskriva detta på fyra ställen kan man samla beskrivningen på en plats, lämpligen inom en objektklass kallad expedit. Övriga specialiteter blir kvar där de varit. Att en och samma fysiska företeelse i verkligheten kan utföra flera roller är ingenting konstigt. Varje person är t ex mantalsskriven, men även konsument. Några är bostadssökande, andra är röstberättigade. En bil kan befinna sig i rollen som oregistrerad, försäkringsobjekt, gods, veteranbil osv. Centralt i objektorienteringssammanhang är just att precisera väl avgränsade och naturliga roller i klassbeskrivningar. (Inom programmeringsområdet talar man om modularisering.) Den mer renodlade klassindelningen återfinns i figur 3.11. Varje fysiskt objekt får, beroende på kvalifikationer, ikläda sig skepnaden av en eller flera av dessa klasser.

Figur 3.11

En väsentlig aspekt eller regel har försvunnit från vad som framgick av figuren 3.10, nämligen att den som är t ex *kött- och fiskexpert* även alltid kan fungera som expedit. Motsatsen är däremot inte säkert sann. Se bara på fröken Ny och även på herr Spik. Inte heller sägs att den som är *kött- och fiskexpert* inte är *järn- och färgexpert*. Nämnda aspekter är väsentliga att ta med i modellen för att den ska anses svara mot verkligheten på ett rimligt sätt. Modifiering i enlighet med figur 3.12 reder ut situationen.



Figur 3.12

Detta, att bryta ut gemensamma egenskaper och beteenden ur klassbeskrivningar och föra dessa till en separat klass, som man istället refererar till, kallas generalisering. Referensen kan uttryckas enligt *fungerar också som* eller *är en*. Den senare har sin engelska motsvarighet i *is-a* vilken är den vedertagna beteckningen i litteraturen.

> **generalisering:** Att bryta ut och gruppera överensstämmande egenskaper och beteenden hos objektklasser under en separat, överordnad, mer generell klass.
>
> Man upplever likheterna mellan objekten på en mer övergripande nivå. På denna nivå upplevs objekten från de ursprungliga objektklasserna ha en likhet som motiverar dem att tillhöra en ny gemensam objektklass.
>
> Obs, att de särskiljande dragen fortfarande representeras i de ursprungliga objektklasserna.
>
> Motsvarande engelska begrepp: *generalization.*

Antag att vi istället hade börjat enligt figur 3.7, ur vilken vi nöjde oss med att särskilja och klassificera det grova beteendet *expediera kunder* under klassen *expedit*. Så småningom skulle de olika specialiteterna uppdagas och beskrivas separat under egna klasser. De specifika

expeditegenskaperna skulle preciseras. Resultatet blir, förhoppningsvis, i slutänden detsamma. Skillnaden är att vi här arbetar oss igenom modelleringen så att säga "top-down" mot tidigare "bottom-up".

> **specialisering**: Precisering och klassificering av särskiljande egenskaper och beteenden under en eller flera nya objektklasser. Obs, de gemensamma dragen representeras fortfarande inom den ursprungliga objektklassen.
>
> Motsvarande engelska begrepp: *specialization.*
>
> Objektklassernas position till varandra i en *är en*-struktur brukar vanligen uttryckas med begreppen subklass och superklass.
>
> **subklass**: En precisering av innebörden av en överordnad klass (superklass). Preciseringen är ofta i form av en kategoriindelning med hänsyn till någon specifik aspekt. Indelningen kan ge upphov till ett antal subklasser för en och samma överordnade klass. Varje objekt i subklassen förekommer även i den överordnade klassen (medan det omvända inte nödvändigtvis gäller, se nedan).
>
> Resultatet av en specialisering.
>
> Motsvarande engelska begrepp: *subclass.*
>
> **superklass**: En gruppering och överföring av gemensamma egenskaper och beteenden ur en eller flera underliggande klasser (subklasser) under samma "hatt".
>
> Resultatet av en generalisering.
>
> Motsvarande engelska begrepp: *superclass.*

Ibland används begreppet **uttömmande specialisering** för att indikera att samtliga objekt i superklassen även har en placering i någon av de nya subklasserna. En specialisering av objektklassen person till subklasserna man och kvinna måste anses uttömmande eftersom det knappast finns någon tredje kategori som inte täcks in av de två övriga.

Specialiseringen i figur 3.12 är däremot exempel på en **ej uttömmande specialisering**. Vi har ju fröken Ny. Hon är expedit men platsar för närvarande inte i någon av de specialiserade objektklasserna. Det enda vi kan säga är att *vissa* expediter även kan tillhöra någon av de tre subklasserna. De övriga har ingen specialitet, ännu okänd specialitet eller ingen specialitet värd att skapa objektklass kring.

För båda fallen gäller att specialiseringen normalt placerar ett objekt från en superklass i högst en av subklasserna. Vi talar då om icke överlappande subklasser. Skulle inte denna förutsättning gälla har vi istället överlappande subklasser.

3.8 Arv

Därmed är vi framme vid en av de mer centrala objektorienteringsmekanismerna. Istället för att renodlat uppleva att fru Lamm ibland figurerar som expedit, ibland som *kött- och fiskexpert* placeras fru Lamm i *kött- och fiskexpert*rollen. Den rollen antas, genom "är en"-kopplingen, innefatta kompetensen att vara *expedit*. Eftersom expedit beskrivs separat, säger man att fru Lamm ärver egenskaper och beteende så som de beskrivs under *expedit*. Generellare uttryckt, ett objekt av klassen *kött- och fiskexpert* ärver egenskaper och beteende från klassen *expedit*.

> **arv:** En tolkning av super/subklassförhållandet som tillför en subklass egenskaperna hos dess superklass. En något mer realiseringsnära formulering:
>
> En mekanism med vars hjälp objekt i en subklass får tillgång till (ärver) egenskaper och beteenden från sin superklass.
>
> Motsvarande engelska begrepp: *inheritance.*

Med objektorienteringsglasögon kan modellen sägas presentera följande fyra objektprofiler.

föreståndare + expedit

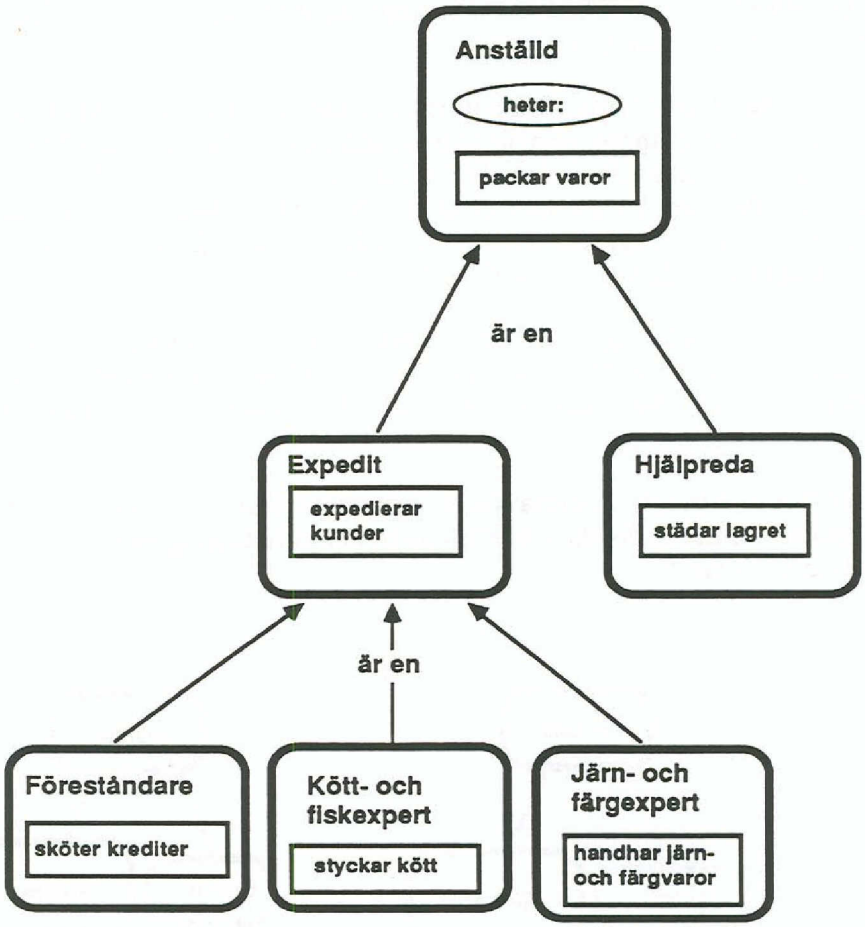
kött- och fiskexpert + expedit

järn- och färgexpert + expedit

expedit

Pelle, ja faktiskt båda två, får sommarjobba med att städa i lagret och hjälpa till att packa varor i kassar. Under den perioden är de anställda och får lön som alla andra. Den expedit som har en Pelle bredvid sig behöver med andra ord inte för ögonblicket packa kassar men får vara beredd att även fortsättningsvis då och då göra det när båda pojkarna är upptagna på annat håll. Pojkarna får alltså inte expediera men väl göra en delsyssla. Den trista städningen i lagret sköter de själva.

Generaliseringsgrafen behöver kompletteras.



Figur 3.13

Som vi ser kan generalisering och arv omfatta valfritt antal nivåer allteftersom behoven uppstår. Man talar ibland om **generaliserings- och arvshierarkier**, eng *IS-A hierarchies*. Fru Lamm hittar nu sina egenskaper och beteenden under tre objektklasser (*tillhör tre objektklasser* alternativt *kan figurera i tre roller*), nämligen kött- och fiskexpert, expedit och anställd.

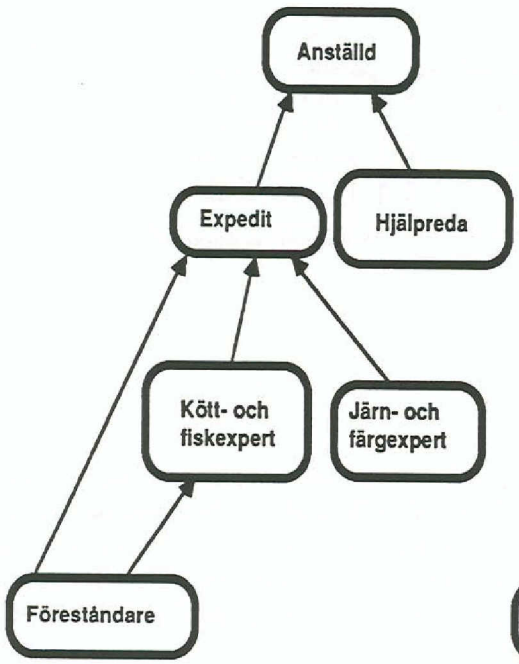
3.9 Multipla arv

Det där med generaliseringshierarkier och arv är bara skenbart enkelt. Flera centrala objektorienteringsbegrepp behöver preciseras. Kontroversiella uppfattningar finns. Vi kommer inte att lägga in några värderingar utan försöker endast redovisa begreppens innebörd. Till att börja med ökar vi komplexiteten kring arv.

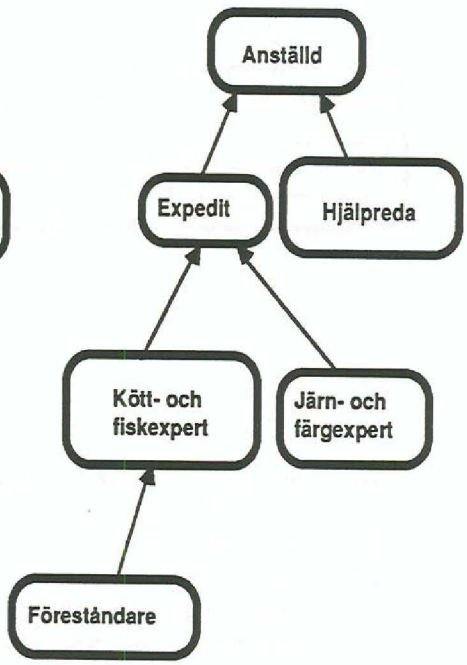
Föreståndaren avslöjade vid en lunch att han faktiskt varit styckmästare på ett slakteri innan han blev föreståndare i handelsboden. För sent insåg han misstaget. Framför allt fru Lamm och herr Torsk gladdes åt nyheten eftersom de nu såg chansen till avlastning vid kött- och fiskdisken. Förutom föreståndare och expedit kan hädanefter föreståndaren kalla sig *kött- och fiskexpert*.

Närmast till hands ligger en komplettering av grafen enligt figur 3.14 a. Vid närmare påseende är ju varje kött- och fiskexpert även expedit varför bilden kan förenklas enligt figur 3.14 b. Resultatet är en något mer komplex hierarki.

Observera att vi i de kommande exemplen utlämnat egenskaper och beteenden för att underlätta överblicken. I en fullständig modell ska de givetvis vara med.



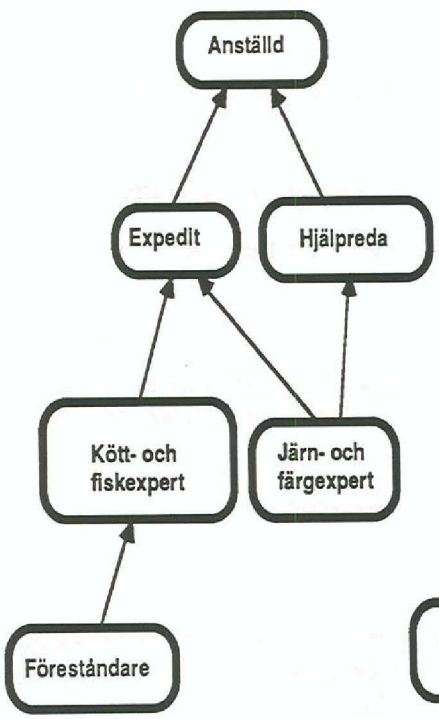
Figur 3.14 a

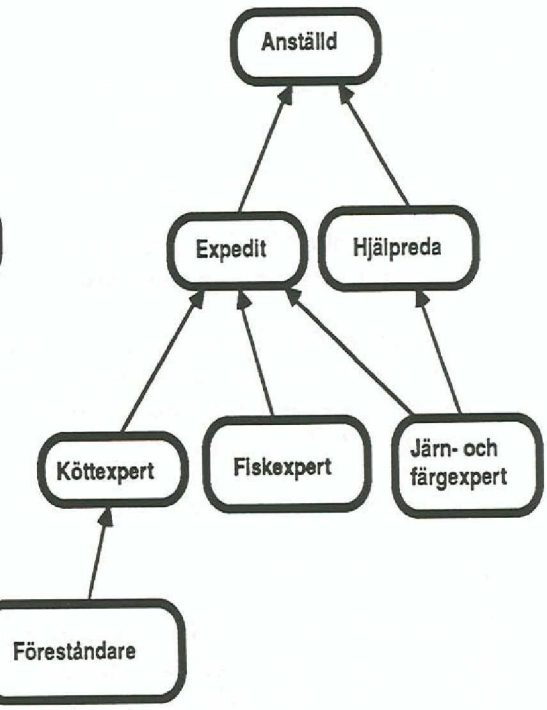


Figur 3.14 b

När nu föreståndaren varit så bussig, ville inte herr Spik vara sämre. Han erbjöd sig att hjälpa till med lagerstädningen. Han tyckte till och med att det borde vara varje järn- och färgexperts skyldighet att hjälpa till med detta eftersom det i stor utsträckning är dennes varor som finns i lagret. Nya ändringar i hierarkin. Vilken hierarki, förresten? Grafen har blivit ett nätverk. Se figur 3.15 a. Järn- och färgexperten ärver egenskaper från såväl expedit som hjälpreda. Vi har ett **multipelt arv**.

> **multipelt arv:** En subklass kan vara relaterad till flera superklasser. Ett objekt i subklassen tillhör var och en av superklasserna. Objektet ärver i konsekvens med detta egenskaper och beteenden från var och en av superklasserna.
>
> Motsvarande engelska begrepp: *multiple inheritance.*





Figur 3.15 a *Figur 3.15 b*

Om nätverk och därmed multipla arv ska anses acceptabla företeelser för *är en*-strukturer diskuteras i objektorienteringskretsar. Båda sidor har företrädare. Förespråkarna hävdar att omständigheterna i verkligheten naturligt modelleras i form av nätverk, att man får en renare struktur med var sak (egenskaper och beteenden) på rätt plats. Figur 3.15 a fyller ju ganska väl sitt syfte. Hur skulle motsvarande situation beskrivas genom enbart enkla arv?

Motståndarna pekar på otydligheter i tolkning ibland. Hjälpreda har beteendet *städar lagret*, vilket betyder rejäl rengöring. Antag att även expediter har ett beteende som kallas *städar lagret,* men att det står för att vika ihop tomlådor och kasta dem i containern. Skulle föreståndaren ropa "Spiken, tar du och städar lagret ett tag" skulle herr Spik direkt undra om han menade rejäl rengöring eller plock med tomkartonger eller eventuellt både och. Herr Spik ställer givetvis ingen kontrollfråga. Han gör sin egen bekväma tolkning och går ut och pysslar lite med tomlådorna. I allvarligare situationer kan feltolkningar få förödande konsekvenser. Den beskrivna typsituationen måste kunna hanteras entydigt.

Trenden inom objektorienteringsområdet verkar för närvarande vara *för* multipla arv, dvs att se enkelt arv enbart som ett specialfall av multipelt arv.

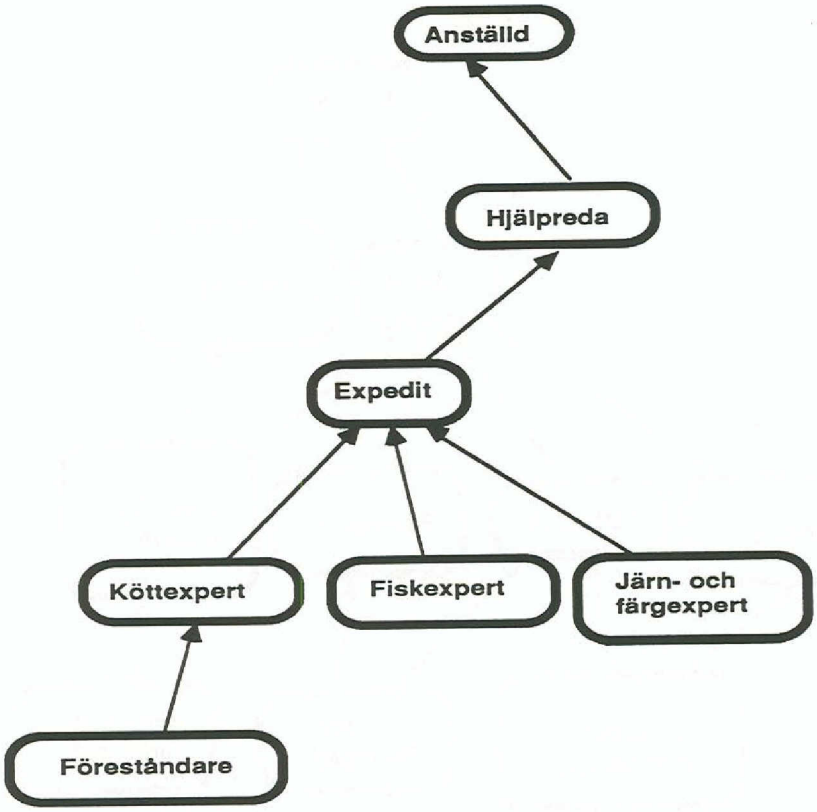
Olika konventioner för att hantera namnkonflikter har vuxit fram. Några olika ansatser är:

- Välj egenskaper och beteenden i första hand från lägsta nivå, därefter från närmast över osv.
- Om flera alternativ, ta det vänstra. (En i sanning tveksam strategi.)
- Ange explicit i subklassen att egenskapen ersätter motsvarande med samma namn på en övre nivå.
- Notera endast beteckning på övre nivå för att klara generell referens men med innebörd att varje subklass har sin version av egenskapen eller beteendet (se vidare under *kontrollerad dynamisk bindning*, avsnitt 3.12 nedan).
- Ge egenskaperna nya namn på den lägre nivån.

Debatten fortsätter.

Tillbaka till lunchen. Vilket lägligt, turbulent tillfälle för herr Torsk att berätta om sin tid på en västkusttrålare. Baktanken om ett eget ansvar för en separat fiskdisk hägrade. Föreståndaren gillar idén under förutsättning att han själv får inskränka sig till köttsidan. Vi delar upp kött och fisk i två klasser. Uppsplittringen av expertisen bjuder inga problem, se figur 3.15 b.

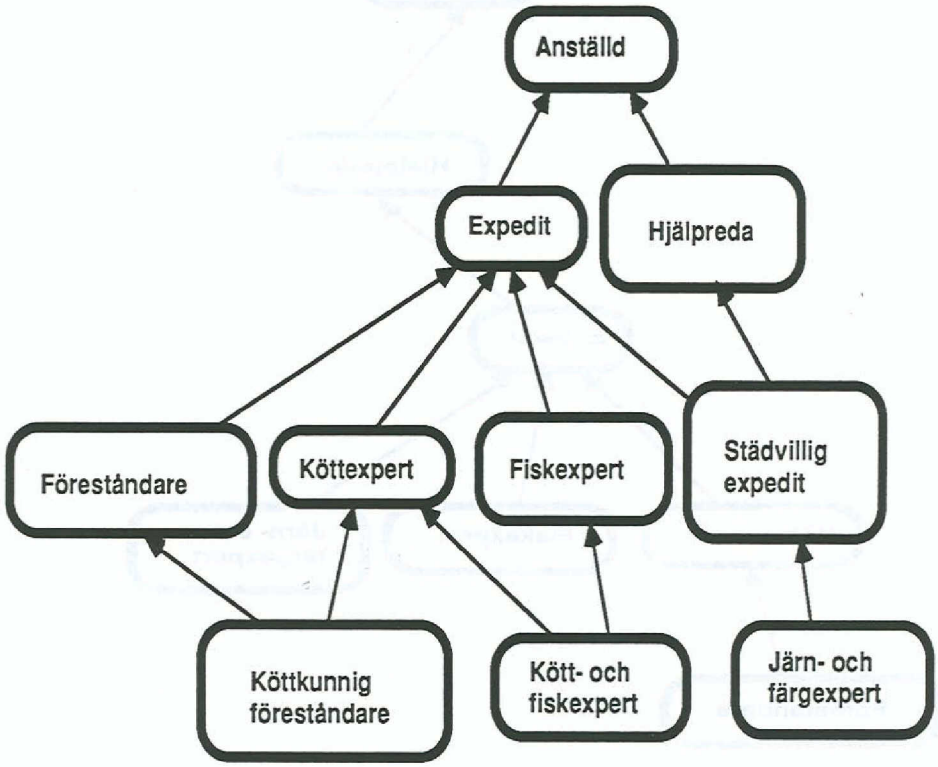
Fröken Ny känner stundens tryck och erbjuder sig plötsligt att vid behov hjälpa till med städningen i lagret. Sagt och gjort. En expedit kan i och med detta även figurera som hjälpreda. Följden blir att herr Spiks hjälpsamhet nu kan kanaliseras via expedit. Expeditens roll som anställd kan gå via hjälpreda. Vips tillbaka till en rejäl hierarki, figur 3.16.



Figur 3.16

Den observante läsaren har redan upptäckt ett antal oförsiktigheter i resonemanget. Vi borde inte längre prata om herr och fru den och den utan om objektklasser och egenskaper/beteenden som gäller varje objekt som tillhör klassen. Helt klart var att varje *järn- och färgexpert* skulle kunna sättas in på städning, vara hjälpreda. Sedan finns det åtminstone en expedit (fröken Ny) som är villig att göra samma sak. Därmed inte sagt att det skulle gälla alla. Fru Lamm, till exempel, är allergisk mot damm. Fru Lamm förresten, det har inget annat sagts än att hon fortfarande ska ägna sig åt både kött och fisk. Var finns den kompetensen objektifierad nu?

Till sist skänker vi kommande föreståndare en tanke. Vad säger att de samtliga har varit styckmästare som den nuvarande? Det enda vi kan säga är att det finns föreståndare som kan kött men sannolikt även sådana som helt avser gå upp i sitt föreståndarskap. Nu är inte strukturen lika enkel längre. Förhoppningsvis speglar den verkligheten bättre. Se figur 3.17.



Figur 3.17

3.10 Arvsaspekter

Varför alla dessa exempel på *är en*-strukturer och arv? Dels har vi kommit in på centrala objektsorienteringsegenskaper vars precisering fortfarande är föremål för debatt. Olika aspekter behöver belysas. Dels vill vi med resonemanget peka på att det till synes uppenbara inte alltid är så enkelt, inte ens med kraftfulla mekanismer. Litteraturen inom området är fullt av små prydliga exempel på arvshierarkier. Här har en, vid första påseende, liten enkel verklighet expanderat till en ganska omfattande modell som innehåller ett antal fallgropar och bedömningslägen. Hur komplexa kan inte modellerna bli om man utgår från realistiska förutsättningar? Man kommer aldrig ifrån svårigheten och utmaningen av att tolka och modellera en komplex verklighet.

Kan det vara så att arvsmekanismerna överutnyttjats i exemplet? Det kan vara farligt att dra för stora växlar på kunskapen om en liten specifik situation. En mer distanserad syn på en handelsbods verksamhet skulle kanske ge vid handen att vad expediterna gör och inte gör, är och kommer att vara mycket personberoende och dynamiskt. En expedit kan vara villig att hjälpa till med allt, en annan håller sig strikt till sin expertis. Även om målet inte var att modellera handelsbodar i allmän-

het utan enbart "vår" handelsbod, kan expediternas dagshumör och föreståndarens intresse för att styra och ställa påverka modellen med en frekvens som känns oacceptabel. Att lägga sig på en stabilare modellnivå kanske skulle vara en lösning.

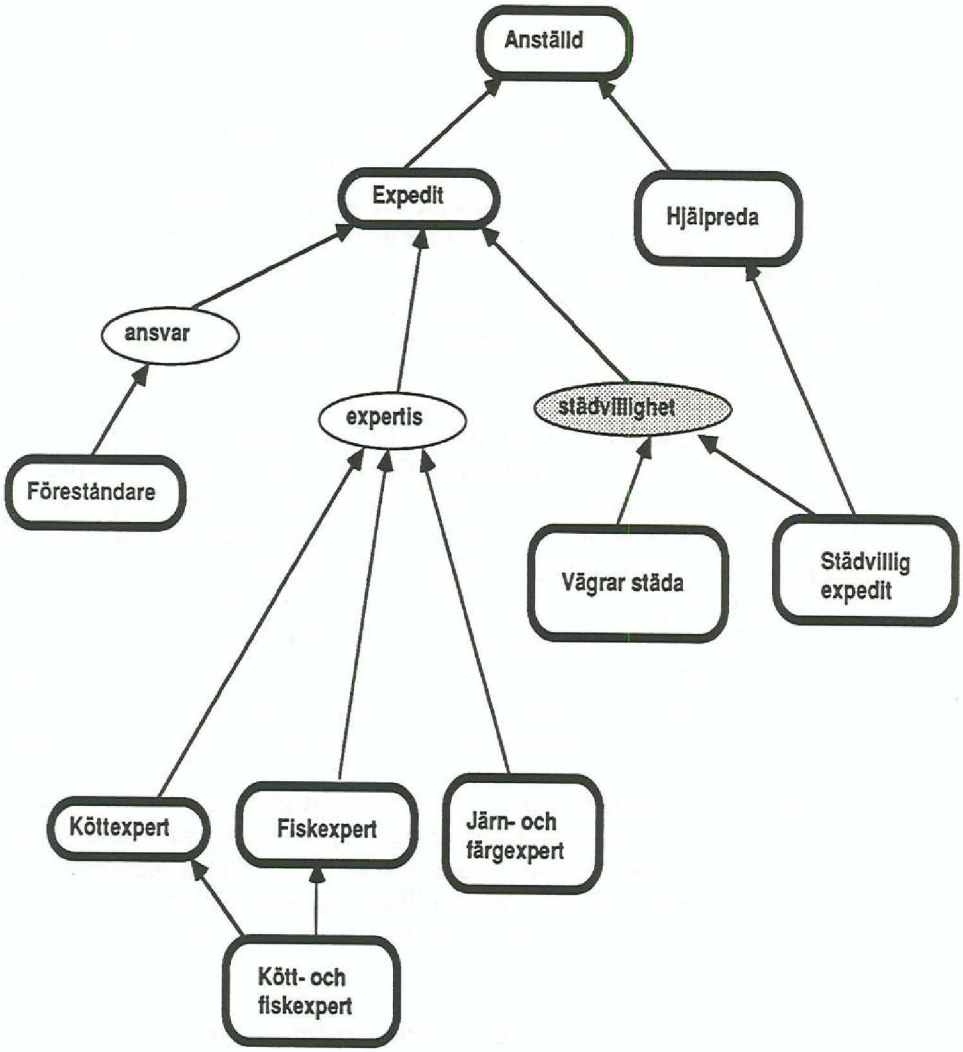
Kanske finns det en infallsvinkel som gör strukturen klarare. Å ena sidan är vi intresserade av att indela expediter med hänsyn till deras *expertis*. Å andra sidan vill vi skapa en specialisering som belyser aspekten *städvillighet*. Vi har hittills blandat dessa aspekter, vilket har gått bra eftersom handelsboden är liten. I ett större perpektiv blir det ohållbart att inte titta på varje aspekt för sig. Det finns t ex ingenting som säger att just järn- och färgexperten skulle vara städvillig och att köttexperten inte skulle vara det.

Antag nu att städvillig inte längre betyder städförpliktelse och än mindre bara för järn- och färgexperter. Det betyder helt enkelt villighet att utföra städjobb. Expertinriktningen har inte någonting att göra med inställningen till städning. Visserligen sade herr Spik att han gärna städade men det är ju ingenting som säger att herr Skruv, handelsbodens näste järn- och färgexpert, känner något alls för städning. Samtidigt kanske herr Torsk går och bär på en hemlig last i form av städdille, och så vidare. Att vara föreståndare är snarast att betrakta som ett slags *ansvar*. Vår föreståndares *expertis* låg ju på köttsidan. Hans *städvillighet* inskränker sig till ett kategoriskt nekande med hänsyftning till allergi.

Figur 3.18 visar att specialiseringen av expedit görs utifrån tre fristående aspekter (ovaler). Detta innebär en gruppering av subklasserna i tre grupper. En aspekt anses vara en uttömmande specialisering, städvilligheten (gråtonad). Antingen gillar man eller gillar man inte att städa. Expertisen har vi redan tidigare utrett vara ej uttömmande. Samma sak gäller ansvar.

När nu *järn- och färgexpert* är en typ av expertis, varför inte placera även *kött- och fiskexpert* som en fjärde expertis på samma nivå som de övriga? Men, skulle fru Lamm då befinna sig i tre expertisroller eller bara den kombinerade? Behövs överhuvudtaget den kombinerade rollen när den täcks in av de båda separata *köttexpert* och *fiskexpert*? Nej, inte under dessa förutsättningar. Vad vi ännu inte avslöjat är att under beteckningen *kött- och fiskexpert* ligger en specifik kompetens, nämligen hur kött och fisk ska hanteras tillsammans i kyldisken, utan att de ska ta smak av varandra. En kompetens bara den som kan de specifika gebiten kan tillägna sig. Figur 3.18 får fortsatt förtroende.

Tyvärr uppmärksammas sällan detta med aspektindelad specialisering i objektorienteringslitteraturen. Överhuvudtaget återstår en hel del teoribildning och erfarenhetsinsamling runt specialiseringshierarkier och arvsmekanismer.

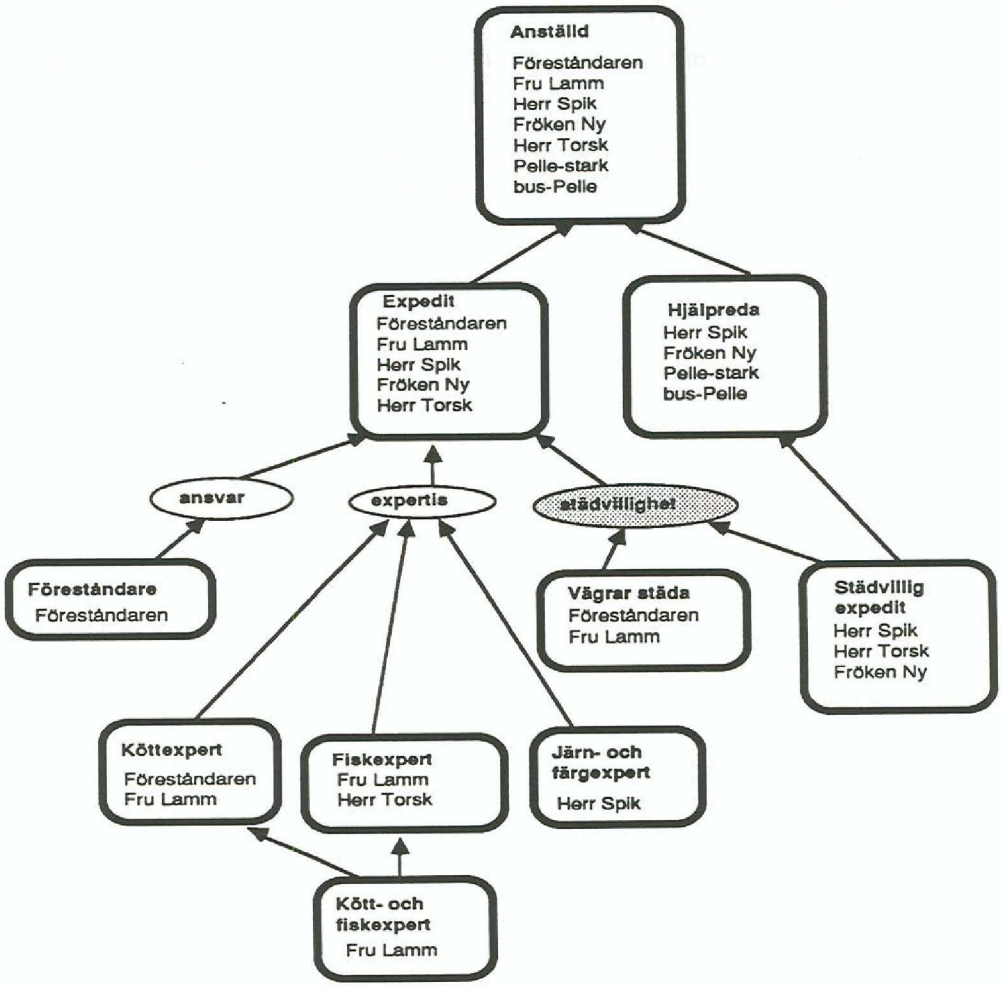


Figur 3.18

I rena trotset mot all specialisering skulle någon kunna tycka att det räcker med objektklassen anställd som, förutom *namn*, ges egenskapen *kompetensprofil*. Beteendet skulle formuleras som beroende av aktuell kompetens. Den anställde utför det som efterfrågas av kunden (och föreståndaren), och begär hjälp när han inte kan.

I alla händelser har vi för närvarande uppfattat ett antal personprofiler samt formulerat deras roller i en *är en*- eller arvsstruktur av nätverkstyp. Vilka personer i handelsboden vi haft i bakhuvudet framgår av figur 3.19. Enligt grafens grundvillkor måste en person på *lägre* nivå även återfinnas på närmast utpekad högre nivå. Följ till exempel Fru Lamm från *kött- och fiskexpert* uppåt. Nya personer kan tillkomma efterhand, exempelvis herr Torsk som fiskexpert.

Ofta utgår man från att ett objekt i superklassen uppträder i högst en av subklasserna, dvs objektet inordnas antingen i klass x eller klass y eller..., om någon. Vår nya aspektindelning modifierar detta till *ett objekt i superklassen uppträder i högst en av subklasserna inom viss gruppering*. Tittar vi in i figur 3.19 ser vi att inte heller det stämmer. Fru Lamm har dubbla expertiser. Förmodligen har vi gjort något fel, brukat våld på det vedertagna. Å ena sidan, om modellen känns rimlig är den ju bra i något avseende (vi kommer sannolikt framöver att göra om samma typ av modellering, även när vi fått kunskap om vad som var galet). Å andra sidan inträder ett gnagande bekymmer: Vilka konsekvenser kan detta få längre fram? Är vi föremål för en enbart tekniskt motiverad formalism eller blir modellen tvetydig, omöjlig att utföra entydiga operationer på? Suck!



Figur 3.19

Sammanfattningsvis:

- Ett verklighetsobjekt kan uppträda i många roller.
- Varje sådan roll motsvaras av ett modellobjekt.
- Varje objektklass svarar mot en roll, en karakteristik som en eller flera objekt i verkligheten uppfattas svara mot.
- Uppdelningen i roller syftar till precisering, enklare underhåll och återanvändning av beskrivningar
- Rollernas förhållanden till varandra beskrivs i en specialiseringsstruktur. Överordnad objektklass kallas superklass, underordnad objektklass kallas subklass.
- Specialiseringsstrukturen utormas som gruppering kring en eller flera specialiseringaspekter.
- Egenskaperna i specialiseringsstrukturen medger arv av egenskaper och beteenden.
- Arvsmekanismen är en central egenskap inom objektorientering.

Vi nöjer oss med att konstatera: Omdöme och klokskap i modellering måste vara givna följeslagare med kunskap om den modellerade verkligheten och svaret på frågan "Varför gör vi modellen?".

3.11 Andra arvsrelaterade begrepp: polymorfism

(Den läsare som fått nog av arv rekommenderas ta språnget direkt till avsnitt 4.)

Vi har sett hur beskrivningen av en och samma fysiska person delas upp i ett antal fristående, avgränsade delbeskrivningar. Dessa hänger ihop i en beroendestruktur (*är en*-struktur). Vill vi referera till ett *visst* objekt, för att t ex titta på egenskaper, använder vi oss av någon unik referens. Ber mormor att få tala med fru Lamm så är det för att mormor vill få något ordnat som hon vet att bara fru Lamm kan. (Vi bortser ifrån andra aspekter än de som finns beskrivna. Att fru Lamm t ex är glad och snabb har under modellerandet bedömts irrelevant.) Det unika för fru Lamm enligt modellen är de samlade egenskaperna som kött- och fiskexpert. Mormor kunde i vår handelsbod lika gärna ropa på en *kött- och fiskexpert*.

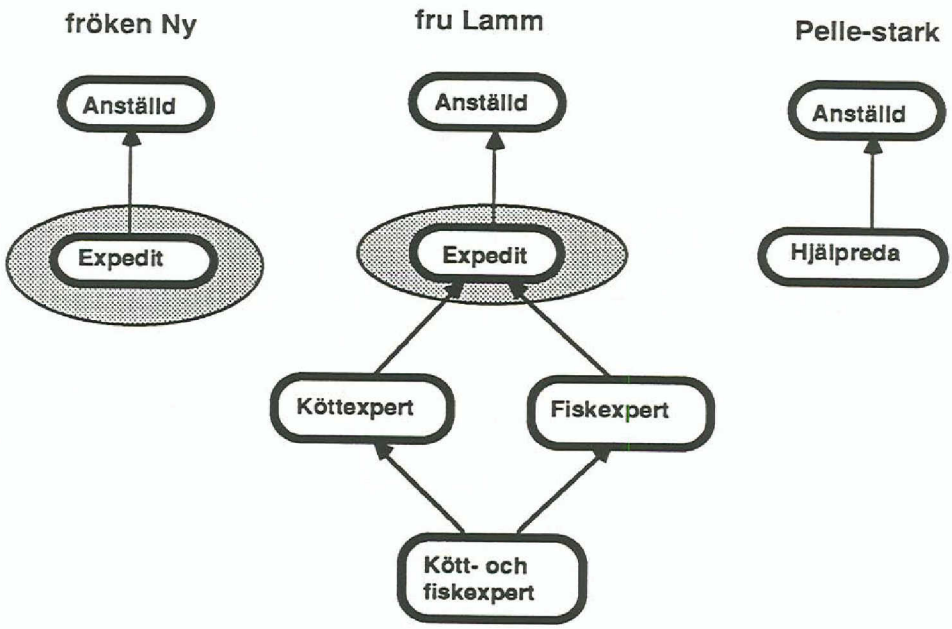
I normalfallet ringer dock mormor med mässingsklockan på disken, vilket för alla parter betyder att det är en kund som vill prata med en *expedit*.

Pelle-pojkarna noterar inte ens signalen, men det gör samtliga andra. Vem som reagerar och kommer till mormors undsättning är ur mormors perspektiv ointressant, huvudsaken det är någon som kan expediera. Exempelvis har fröken Ny exakt erforderliga kvalifikationer. Fru Lamm kan detta och mer därtill, medan Pelle-stark inte är att lita på som expedit. Närsynte mormor tilltalar fortsättningsvis tillskyndande an-

ställd med *expediten*. Fröken Ny tycker det låter trevligt medan fru Lamm bistert accepterar faktum trots att hon egentligen har den betydligt finare titeln som *kött- och fiskexpert*. Det väsentliga är att båda har expeditegenskaper.

Att *expediten* kan syfta på eller referera till objekt ur olika klasser kallas inom objektorienteringsområdet för **polymorfism** (något kan anta flera former). Referensen *expediten* är polymorfistisk eftersom den kan peka på objekt som åtminstone har expeditegenskaper, dvs objekt ur samtliga klasser förutom *hjälpreda* och *anställd*. Se de skuggade ovalerna i figur 3.20. Detta är ju praktiskt för mormor, som nu inte behöver bekymra sig om att alltid titulera vederbörande exakt rätt. (Som extremt närsynt skulle det för övrigt vara en omöjlighet.) Om personen kan mer än det efterfrågade eller inte är i nuvarande stund ointressant så länge åtminstone de efterfrågade egenskaperna finns.

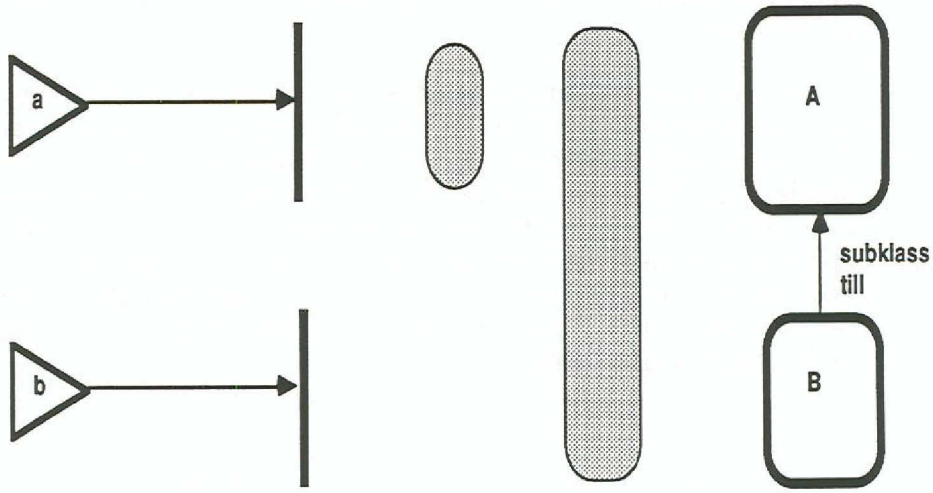
Arvshierarkier spelar, som vi ser, en central roll för att styra och realisera polymorfism i ett system.



Figur 3.20

Polymorfism: Något som kan anta flera former. En referens (eller beteckning) som används för att peka ut objekt av en viss klass kan även användas för att peka ut objekt ur en subklass. Se figur 3.21.

Motsvarande engelska begrepp: *polymorphism*.



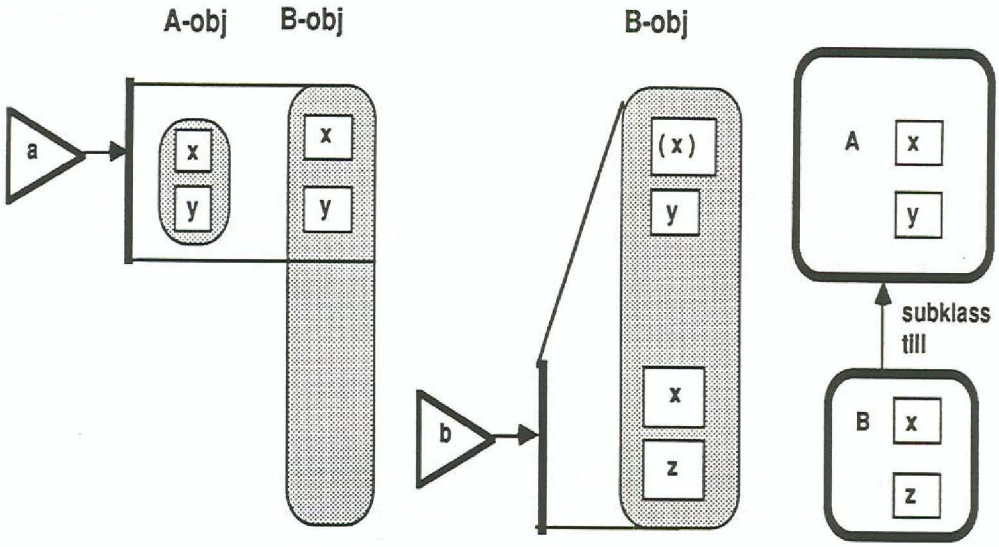
referens (t ex a) för objekt tillhörigt viss objektklass (A)	den nivå i strukturen som måste ge "träff"	A:s täckning	B:s täckning	Specialiserings-strukturen

Figur 3.21

När mormor en annan dag efter lång väntan vid fiskdisken ilsket ropar ut över handelsboden "Finns det ingen som kan filéa en makrill här" och effekten blir att fröken Ny (som inte hörde så noga) kommer framtrippande, blir mormor knappast lycklig. Knappast fröken Ny heller, hon kan ju inget om fisk. När mormor ber *fiskexperten* ordna till fyra hekto filé varseblir fröken Ny misstaget, smiter snabbt in i kafferummet, tar tag i herr Torsk och trycker ut honom mot fiskdisken. Den något vilsne herr Torsk, väl medveten om mormors närsynthet, uttrycker ett erfaret "Såja, nu är filékniven vass igen. Kanske mormor ville upprepa sitt önskemål?" och kommersen kan fortsätta. Även fru Lamm skulle kunnat komma ifråga som alternativ till herr Torsk, men inga andra. Fröken Ny har visserligen vissa gemensamma egenskaper med en fiskexpert (anställd och expedit), men det räcker inte. *Fiskexperten* används av mormor som referens på någon som förväntas vara *åtminstone* fiskexpert (inklusive alla överordnade roller i *är en*-strukturen). Därav kan man lära att aldrig låta *fiskexperten* peka på någon expedit. Ibland kan kompetensen finnas, ibland inte.

Inom objektorienteringsvärlden skiftar principerna för hantering av polymorfism.

Den ena ansatsen säger att man före start kontrollerar att de egenskaper och beteenden som kommer att efterfrågas finns definierade på åtminstone referensens nivå. Att åberopa fiskkompetens hos en expedit skulle inte tillåtas även om det ibland fungerar (som när expediten är herr Torsk). Väsentligt för en korrekt referens är att de efterfrågade (klass-) egenskaperna återfinns inom klassen eller någonstans i den överordnade arvshierarkin. Full säkerhet eftersträvas. Polymorfistisk referens är ok, men med ett begränsat synfält. Se figur 3.22. Ansatsen kallas **statisk bindning**.



Figur 3.22 Referensens synfält (sin nivå och uppåt i strukturen). Vid namnkonflikt tas normalt den lägsta nivån. När b begärt att x ska utföras, blir det B:s x som startas, inte A:s.

3.12 Andra arvsrelaterade begrepp: dynamisk bindning

De andra två ansatserna utnyttjar något som kallas **dynamisk bindning**. Dynamisk bindning utgör det naturliga komplementet till polymorfism.

Fri dynamisk bindning

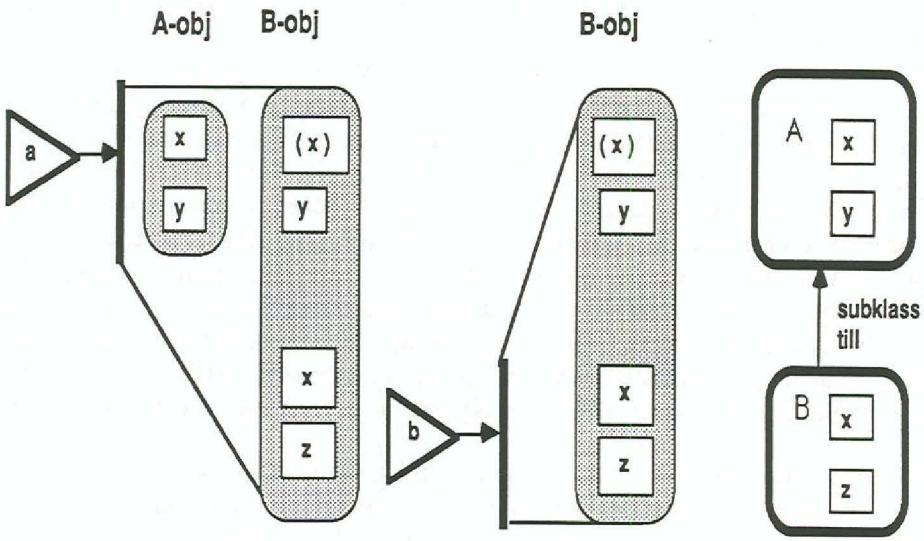
En av dem är av typen "blunda och ge". Vid fel, som när *expediten* fröken Ny visar sig sakna fiskkunskaper, avbryts exekveringen. Ansvaret läggs på implementatörens förmåga att förutse och hantera alla situationer.

Åter till situationen med expeditreferensen och under förutsättning att fru Lamm expedierar. Bullar, mjölk, konserver m m inhandlas. Längst ner på inköpslistan står "4 hg makrillfilé". Mormor kan inte utan vidare förutsätta att hennes expedit besitter erforderlig kompetens. Det kunde ju vara fröken Ny igen. Listiga mormor frågar därför först "Är månne *expediten* fiskexpert?", får ett jakande svar och levererar beställningen "Kan då *expediten* ge mig 4 hg makrillfilé"? eftersom hon nu vet att den som kallas *expediten* i det aktuella fallet även är fiskexpert. Normalt sett finns med säkerhet bara expedit- och anställdegenskaperna tillgängliga, men vet man att den aktuella personen tillhör en subklass är det ju rimligt att även subklassegenskaperna finns tillgängliga. Efter förfrågan kan mormor utan problem besvära expediten (fru Lamm) med filéer. Samma begäran till fröken Ny skulle, som vi redan vet, bli helt fel.

Referensen *expediten* är i strikt mening avsedd att peka på objekt av klassen expedit (för att kunna nå egenskaper tillgängliga på denna eller högre nivå). Pekar referensen för tillfället på ett objekt som tillhör en subklass sätts referensen "dynamiskt" att peka på subklassnivån, dvs objektets samtliga egenskaper och beteenden hamnar i synfältet. Detta alldeles naturliga förhållande i ett verklighetsperspektiv kallas inom objektorienteringsområdet för **dynamisk bindning**.

Fört till sin extrem skulle man i utgångsläget kunna tillåta referensen att vara frikopplad från varje form av klassberoende. Den dynamiska bindningen utförs villkorslöst när objektet uppenbarar sig. Även kravet på subklasstillhörighet utgår. Man får helt enkelt se vad det blir. Begreppet "fri dynamisk bindning" ligger nära till hands när man inte förrän vid utförandetidpunkten vet om operationen finns tillgänglig (kan utföras) eller inte. Ibland säger man här att objektreferensen har en **svag typning**.

> **Dynamisk bindning:** Det synfält som görs tillgängligt för en referens (pekare) avgörs i samma stund som det kan konstateras vilken objektklass objektet tillhör. När objektreferensen kopplats till en klass möjliggör arvsmekanismerna att egenskaper och beteenden inom klassen och i dess superstruktur blir tillgängliga. Se figur 3.23.
>
> Motsvarande engelska begrepp: *dynamic binding.*



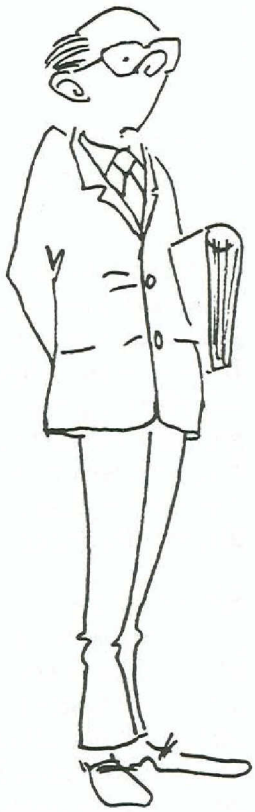
Figur 3.23 Referensens synfält (allt som finns tillgängligt hos referatobjekt).

Kontrollerad dynamisk bindning

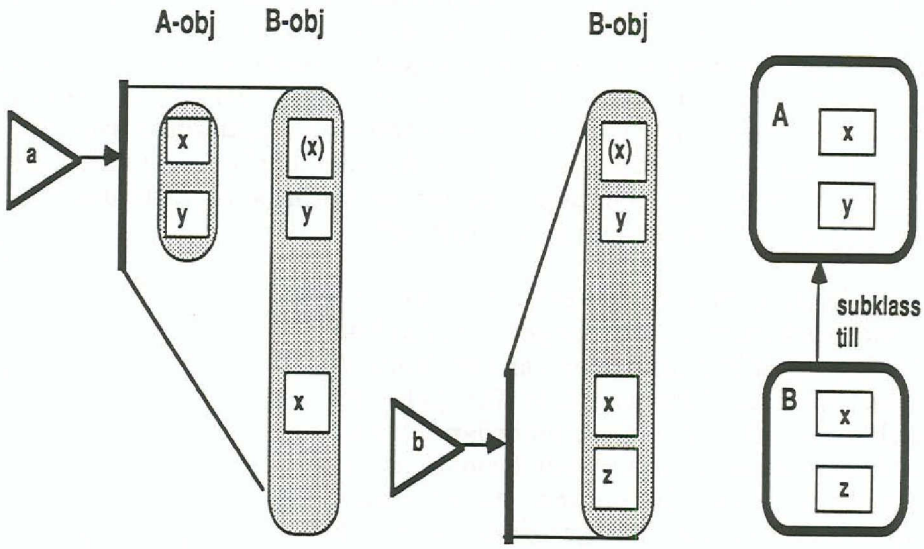
En viss formalism måste upprättas om man ska kunna uttrycka operationer på en modell på ett strikt sätt, för att korrekt kunna hantera modellsituationer. Polymorfism, dynamisk bindning och arvsmekanismer är viktiga förutsättningar. Förutom vid exekvering spelar de en mycket viktig roll under arbetet med att bygga upp en modell. Antag att vi till att börja med endast såg personerna som anställda. Anställda har, genom anställningsavtal, i det läget av modellarbetet bl a rätt till vissa anställningsförmåner. Vi etablerar objektklassen *anställd* med egenskapen *förmån*. Redan nu kan den som beskriver egenskaper och beteende hos objektklassen *handelsbodens revisor* i dennes beteendebeskrivning införa syn av t ex rimligheten i ingångna avtal genom att referera till *viss anställds förmån*. Antag att den generellt är 15% rabatt. Vid det fortsatta modelleringsarbetet preciseras så småningom uppdelningen i expedit och hjälpreda. Expediter visar sig av hävd, utöver 15% i rabatt, även ha rätt att äta så mycket godis de önskar. Detta beskrivs i egenskapen *förmån* inom klassen expedit.

Rimligtvis borde objektklassen *handelsbodens revisor* anpassas till den nya situationen genom en omformulering av förmånsreferensen till något i stil med: "Om den anställde är en expedit så hämta förmånsbeskrivning från klassen expedit, i annat fall från klassen anställd". Finns det andra klasser som utnyttjar dessa förmånsbeskrivningar måste motsvarande justeringar göras även hos dem. Mycket merarbete och risk för fel eller ofullständigheter blir följden. Vem hittar och talar om för de berörda objektklassernas specificerare att ändringen måste införas?

Den eleganta objektorienteringlösningen på förmånsbekymret heter återigen polymorfism och dynamisk bindning, men i en mer kontrollerad version. *Viss anställd* får peka på subklasser till anställd (polymorfism). De av subklassens egenskaper, som även finns uttryckta i generellare form högre upp i superstrukturen, görs tillgängliga (dynamisk bindning). Om viss anställd pekar ut fru Lamm tas förmånsbeskrivningen från klassen expedit. Tittar revisorn på Pelle-stark finns ingen förmånsbeskrivning inom klassen hjälpreda, varför den gamla under anställd får duga, eller "om inget annat sagts gäller 15% rabatt"! Finessen blir att revisorsbeskrivningen inte alls behöver röras! Kontrollen utförs alltid på den närmast objektet beskrivna versionen av *förmån*.



Kontrollerad är denna version eftersom efterfrågad egenskap eller beteende alltid kan återfinnas om inte annat så åtminstone på referensens objektklassnivå. Alltså föreligger ingen risk att hamna i en odefinierad situation. Genomförbarhet kan kontrolleras i förväg. Fördelen med dynamisk bindning kvarstår. Finns en mer preciserad eller anpassad version inom det aktuella objektets synfält, tas denna. Se figur 3.24 där B:s beteende x kommer att utföras när det objekt som a pekar på tillhör objektklassen B. Pekar a på ett A-objekt utförs A:s version av x. Den frihet som går förlorad ligger i att z-beteendet inte kan nås även om a skulle peka på ett B-objekt. Kanske ett rimligt pris för ökad säkerhet!?



Figur 3.24 Referensens synfält (sin nivå och uppåt i strukturen, men i versioner på lägre nivåer, som sådana finns).

Revideringar kan förväntas ett stort antal gånger under en modells framväxt. Rationaliseringsvinsten med en objektorienterad ansats måste i detta perspektiv bli avsevärd. Vår modell växer t ex snabbt ut till sin fulla storlek enligt tidigare figur 3.19. Det kanske uppdagas att föreståndare i gemen unnar sig en rabatt på 20 % på samtliga varor. I gengäld får köttexperter ta hem köttben till hunden och fiskexperter dagens överblivna strömming. Förmånsbeskrivningen preciseras inom respektive klass i enlighet med denna vetskap. Revisorn kan utan åtgärd oförtrutet jobba vidare som förut.

Självklart har även solen sina fläckar, feltolkning t ex. När revisorn "studerar" kött- och fiskexperten fru Lamm finns förmånsbeskrivningar på tre olika övernivåer. Den under anställd slipper han, men vilken av kött- eller fiskexpertförmånerna ska gälla? Fru Lamm själv har sannolikt tolkat det till att båda gäller. Tänk om modellbyggaren lite slarvigt under fiskexperts förmån bara lagt in den extra strömmingsförmånen med förhoppningen att 15%-rabatten ändå beskrevs under anställd? Även en objektorienterad ansats kräver eftertanke!

3.13 Avrundning av avsnitt 3

Hela avsnitt 3 har varit en lång resa genom ett antal begrepp som anses vara förknippade med den objektorienterade ansatsen. En tröst kan vara att det finns en uppsjö fler, de flesta från programmeringssidan.

Några begrepp har en klar och entydig definition. Andra är skenbart enkla och tilltalande men löses upp i alltmer tilltagande förvirring ju mer man försöker fördjupa förståelsen. Mycket av vad som skrivs inom området handlar om teknik medan modell- och metodintresset verkar svalt. Var finns rekommendationerna om hur man hittar och avgränsar objekt? Vad blir konsekvenserna om man gör så eller så? Vad kan konsekvenserna bli av ändringar i en specifikation? Vad är en objektmodell? En del anser det vara ungefär liktydigt med datamodell, andra inkluderar lite beteenden i datamodellens objekt medan ytterligare andra ser objektmodellen som en helt ny typ av "levande" företeelse. Undringar och frågetecken dyker upp.

4. Tillämpningsområden för objektorientering

Ett par frågor som säkert trängt sig på läsaren efter att ha behövt leva med i uppbyggnaden av en modell över handelsbodens värld och verksamhet är "Varför överhuvudtaget göra en modell av någonting?" och "Varför över handelsboden?".

Den sista frågan kan lätt besvaras med att det bara blev så. Den första däremot kräver en mer seriös kommentar.

Åtminstone tre användningsområden kan urskiljas:

a. För att komma underfund med vissa egenskaper hos verkligheten, förmodligen under ett antal olika förutsättningar. Det är omöjligt eller för dyrt att göra det på verkligheten själv. Verkligheten kanske ännu inte ens finns.

b. För att hantera information om verkligheten.

c. Den huvudsakliga användningen hittills av objektorientering har dock varit i situationer där det inte gällt att skapa en modell av något utan att skapa en verklighet i sig utan tanke på att efterlikna något annat.

För denna rapport har syftet bara varit att ha ett exempelunderlag.

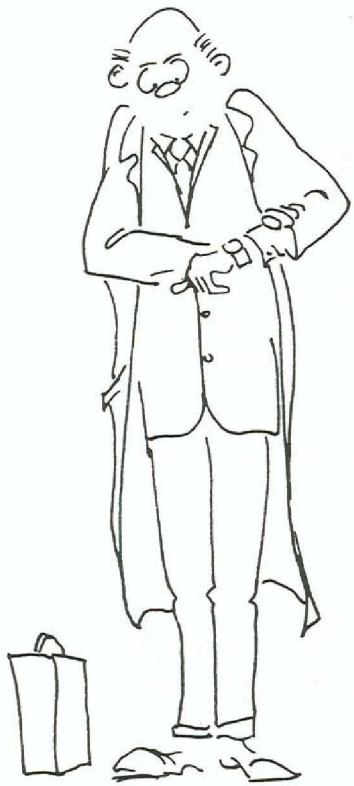
Låt oss kort beröra de tre olika användningsområdena utifrån exemplet ovan.

4.1 Användningsområde a

Antag att handelns rationaliseringsnämnd vill studera funktion och resursutnyttjande inom handelsbodar i syfte att kunna lägga fram förslag till nya, kostnadsbesparande rutiner för kundservice och lagerhantering. Man behöver testa olika strategier och deras konsekvenser. Verksamheten i en handelsbod är för komplex för att man utan tester skulle kunna ana konsekvenserna av de olika strategierna.

Det skulle bli ytterligt kostsamt om man behövde stänga en handelsbod några dagar eller veckor för att utföra testerna. Expediter, kunder, lagerpersonal m fl skulle behöva lära in och genomföra sitt beteende i

enlighet med varje testad strategi. Lagret skulle behöva omorganiseras för varje ny strategi. Ett antal av nämndens anställda skulle samtidigt springa omkring och iaktta och tidmäta allt som händer.



Istället har man valt att beskriva de olika objektklasser som är av intresse för denna studie. Deras egenskaper, beteendemönster och samspel med omgivningen formuleras i en simuleringsmodell, förslagsvis skriven i Simula. När modellen är färdig spelas dess beteende upp, den exekveras. Relevant statistik samlas in och jämförs. Den bästa strategin formuleras i form av en rutinbeskrivning som skickas ut till samtliga handelsbodar för efterlevnad. Skulle modellen vara bristfälligt utformad kan man utgå ifrån att även statistiken har dålig verklighetsöverensstämmelse och att det är en tidsfråga innan de införda rutinerna ute i handelsbodarna ger upphov till ett mindre kaos.

En annan variant på samma tema får vi hos Arkitekterna Rit&Bygg som fått i uppdrag att rita den nya handelsboden i byns centrum. För att få bekräftat att ritningens intentioner kommer att hålla i den färdigbyggda boden bygger man en modell i plast. Till på köpet har man gjort de viktigaste objekten rörliga och fjärrstyrda. Modellen "körs" under ett antal olika förutsättningar för att man ska kunna konstatera att allt verkar fungera. Bland annat upptäcktes att vändplan för ICAs lastbilar var för snålt tilltagen samt att det snabbt blev trängsel framför köttdisken med aktuell placering av diskarna.

God verklighetsöverensstämmelse är vital för ett bra resultat! Egenskaper och beteende i modellen bygger på en befintlig eller tänkt verklighet. När modellen aktiveras arbetar den efter sina egna förutsättningar. Är dessa verklighetsnära beter sig modellen i nära överensstämmelse med verkligheten, annars inte. Objekten i modellen lever sitt eget liv, men förhoppningsvis verklighetstroget, annars är nyttan med modellen tveksam.

4.2 Användningsområde b

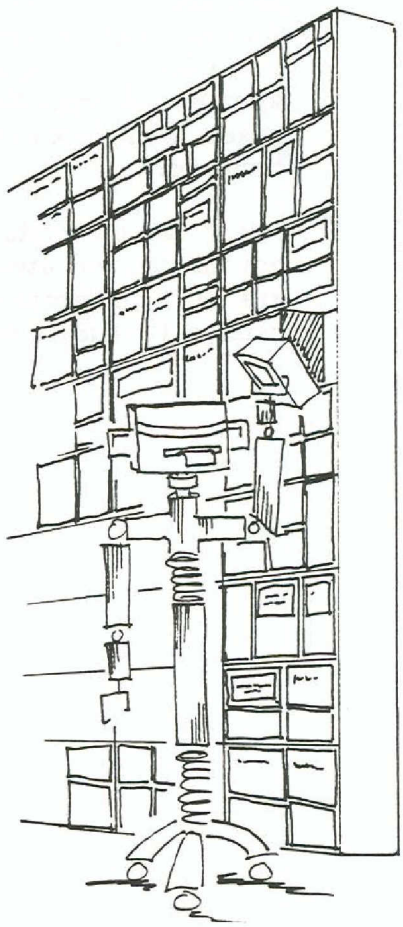
Antag nu att vi ska bygga ett informationssystem som bl a ska hålla reda på vilka kunder som köpt vad, när och av vilken expedit. Återigen uppträder objektklasser som kund, expedit och vara. Skillnaden mot tidigare ligger i att vi nu, istället för en modell som ska testas för sig, har en verklighet att hålla reda på. Ett objekt i modellen ska hela tiden vara en god bild av ett objekt i verkligheten. Då blir det genast svårare. Hur kan vi ge modellobjektet ett eget beteende när det i själva verket hela

tiden ska vara en aktuell avbildning av en annan verklighet? Visserligen vet säkert vår gamle föreståndare ganska väl vad som försiggår i handelsboden, men säker kan han ju inte vara. Någon kund kanske väljer att plocka ihop sina egna varor. En annan betalar för varor han inte orkar bära med sig just nu. När det är köer vid köttdisken kan det tänkas att Fröken Ny väljer att filéa fisken själv även om hon borde ha väntat på herr Torsk. Många liknande fall kan uppstå.

Om informationsmodellen ska kunna spela rollen som en god avbildning av verkligheten måste någon rapportera vad som händer i själva verket. Det räcker inte med att spela upp vad som borde hända.

Ett administrativt informationssystem har först och främst funktionen att kunna ta emot information från registratorn/sensorn om vad som hänt eller om aktuellt tillstånd i verkligheten. Objekten i modellen är avbildningar av de företeelser man vill hantera information om. De har olika typer av egenskaper, de är närmast att betrakta som statiska objekt som omgivningen utför operationer på. Man brukar tala om ett kameraperspektiv på verkligheten. När objektorienteringslitteraturen ger exempel på objektbeteenden inom detta applikationsområde får dessa snarare karaktären av beteenden/funktioner hos olika delar av informationssystemet eller hos det databashanteringssystem som utnyttjas än hos de objekt vars information hanteras.

Frågan är om inte ett par av de centrala mekanismerna inom objektorienteringen, nämligen att objekten kan härbergera och utföra beteenden samt kan kommunicera via meddelanden, tappar i attraktion i samband med utveckling av så kallade administrativa tillämpningar. Frågan kommer att diskuteras vidare i en separat SISU-rapport.



4.3 Användningsområde c

Antag att mormor i själva verket var en produktspecifikation. Antag vidare att expediterna var tillverkningsprocesser som då och då behövde komponenter för sitt arbete. Pelle var antingen en höglagerrobot eller annan internleverantör av komponenter.

Företaget har att hantera ett antal sådana processer där varje process i princip har överensstämmande funktion, nämligen att sammansätta komponenter till en produkt. Varje typ av process producerar däremot sin del av produkten uppbyggd av sin uppsättning komponenter och i överensstämmelse med produktspecifikationen. Verkstaden är höggradigt automatiserad. Konstruktörerna av den

nya verkstaden hämtade inspiration och kunnande från ett motsvarande företag i Japan men har anpassat processen efter svenska förhållanden och det delvis egna produktsortimentet.

Verkstadsprocessen har växt fram på konstruktörens ritbord. Det finns ingen bakomliggande verklighet som den ska efterlikna, den är en egen verklighet. I denna verklighet händer det saker. En del av de ingående beståndsdelarna utför arbetsuppgifter 24 timmar om dygnet. Andra företeelser ingår i processen som passiva objekt. Dit hör kanske komponenterna (om de inte av egen kraft hoppar på plats). Varje objekt arbetar efter sina förutsättningar men i samklang med de övriga för att helhetsresultatet ska bli det avsedda. Förutsättningarna för en objektorienterad ansats är excellenta!

Ta ett annat fall. Varorna är i själva verket data i en databas. Expediterna är processer i ett fleranvändar-dbms. Mormor är en formulärhanterare. Pelle stoppar inte varorna i påsar. Han är en grafisk editor för grafisk presentation av data. Hela paketet är tillverkat av företaget Komplett AB. Företaget har givetvis sneglat på andra liknande produkter men i stort utvecklat efter sin egen bärande idé. Produkten är sin egen verklighet.

I denna programvara händer det också saker. De olika funktionerna eller objekten utför var och en sina arbetsuppgifter. Hur de väljer att lösa uppgiften är vars och ens ensak så länge den utförs på ett fullgott sätt. Objekten anropar varandra för att få hjälp, tala om att arbetet är klart, m m. Återigen förstklassiga förutsättningar för en objektorienterad ansats. Skilj dock noga på dessa processobjekt och de dataobjekt som hanteras.

Det är lätt att förstå att objektorientering i första hand kommit till användning vid utveckling av system i denna kategori. Här finns kraften och erfarenheterna. Kanske är det här objektorientering hör hemma. Framtiden får utvisa.

5. Var kan jag läsa mera?

För den som är nyfiken på objektorienterad programmering och objektorienterade programmeringsspråk rekommenderas

Bertrand Meyer: *Object-oriented Software Construction*, Prentice Hall, 1988, ISBN 0-13-629049-3. Boken är mycket informativ och utförlig. Betraktas som ett standardverk inom området.

Den nyligen utkomna boken **Grady Booch**: *Object Oriented Design with Applications*, Benjamin/Cummings, 1990, ISBN 0-8053-0091-0, inkluderar en hel del om modeller och metoder. Den redovisar även ett antal praktikfall. Är skriven av en av förgrundsfigurerna inom området.

Rumbaugh, m fl: *Object-Oriented Modeling and Design*, Prentice Hall, 1991, ISBN 0-13-629841-9, har rykte om sig att vara välskriven och heltäckande.

Välkänd för många är Edward Yourdon och "structured analysis". Nu är han en förespråkare av objektorienterad analys tillsammans med den flitige föredragshållaren Peter Coad.

Coad/Yourdon: *Object-Oriented Analysis*, Yourdon Press, 1990, ISBN 0-13-629122-8.

En nyligen utkommen, informativ och lättläst bok är **Winblad, Edwards, King**: *Object-Oriented Software*, Addison-Wesley, 1990, ISBN 0-201-50736-6. Den behandlar det mesta från analys till implementering.

Varje år anordnas flera konferenser om objektorientering. En av de mer välkända är **OOPSLA** (Object-Oriented Programming: Systems, Languages and Applications). Proceedings utges av ACM Press. Andra konferenser är **ECOOP** och **TOOLS**.

Den som inte vill läsa allt som står i varje års uppsättning proceedings kan istället överlåta urvalsarbetet till andra kloka personer. Det har blivit vanligt att kändisar på detta sätt fungerar som editorer. Sammanställningar som gjorts inom objektorientering är bl a:

Kim, Lochovsky: *Object-Oriented Concepts, databases, and applications*, ACM Press, 1989, ISBN 0-201-14410-7.

Zdonik, Maier: *Readings in Object-Oriented Database Systems*, Morgan Kaufmann, 1990, ISBN 1-55860-000-0.

Cárdenas, McLeod: *Research Foundations in Object-Oriented and Semantic Database Systems*, Prentice Hall, 1990, ISBN 0-13-806340-0.

Sveriges Tekniska Attachéer har givit ut två rapporter:

Järvinen: *Objektorienterad programmering - översikt och erfarenheter i USA*, USA U3-8803, 1988, ISSN 0283-0531.

Wretling, Karlquist: *ODBMS Objektorienterade databaser - utvecklingen i USA*, USA 9010, 1990, ISSN 1100-2999.

Den som är medlem i ACM får månadsmagasinet Communications. Septembernumret 1990 var ett intressant temanummer om Object-Oriented Design. Annars kan man löpande följa utvecklingen i kvartalsskriften *Journal of Object-Oriented Programming*.

För utförligare bibliografier hänvisas till respektive bok. Boochs bok t ex har en fullmatad och välstrukturerad lista.

6. Till sist

Ett stort tack till Anna Resare och Matts Ahlsén på SISU för alla goda råd och synpunkter och till Helena Persson för förträffligt redigeringsarbete.