KUNGL TEKNISKA HÖGSKOLAN
Royal Institute of Technology
Numerical Analysis and Computing Science

TRITA-NA-D0103 • CID-109, KTH, Stockholm, Sweden 2001

Begreppsmodellering och Matematik

Ambjörn Naeve

CID
Centre for
User Oriented IT Design

Ambjörn Naeve
Begreppsmodellering och Matematik
Report number: TRITA-NA-D0103, CID-109
ISSN number: ISSN 1403-0721
Publication date: February 2001
E-mail of author: amb@nada.kth.se
URL of author: http://www.nada.kth.se/~amb

Reports can be ordered from:
CID, Centre for User Oriented IT Design
Nada, Dept. Computing Science
KTH, Royal Institute of Technology
S-100 44 Stockhom, Sweden
telephone: + 46 8 790 91 00
fax: + 46 8 790 90 99
e-mail: cid@nada.kth.se
URL: http://kmr.nada.kth.se

INNEHÅLLSFÖRTECKNING

1 Begreppsmodellering i UML

1.1 Inledning

Den här lilla skriften handlar om begreppsmodellering, vilket är en teknik för att beskriva egenskaper hos och relationer mellan olika begrepp inom ett givet problemområde. Den introducerar grunderna i ett begreppsmodelleringsspråk som kallas UML (Unified Modeling Language), vilket är ett bildspråk där man kan rita upp begreppen och deras relationer i olika typer av diagram[1]. Avsikten med detta är att synliggöra hur man tänker inom ett visst område, dvs vilka aspekter man tycker är viktigt att betona och vilka man anser att man kan utelämna. När man ger en verbal beskrivning av sina tankar, så klingar orden bort så fort som man har uttalat dem. När man däremot ritar upp en bild över begreppen och deras relationer så får man en bakgrund mot vilken man kan diskutera och upptäcka otydligheter och motsägelser i sina resonemang. Bilden stannar ju kvar hela tiden, vilket gör att när man diskuterar begreppen så syns det hela tiden hur man har tänkt på dem hittills. Detta underlättar vidareutveckling och förfining av begreppsmodellen såväl som kommunikation av densamma mellan olika individer och grupper. En viktig avsikt med begreppsmodellering är att uppnå konsensus (= samsyn) på vilka aspekter som är viktiga inom ett visst område, och vilka som kan utelämnas (= bortses ifrån) eftersom de är oväsentliga i sammanhanget.

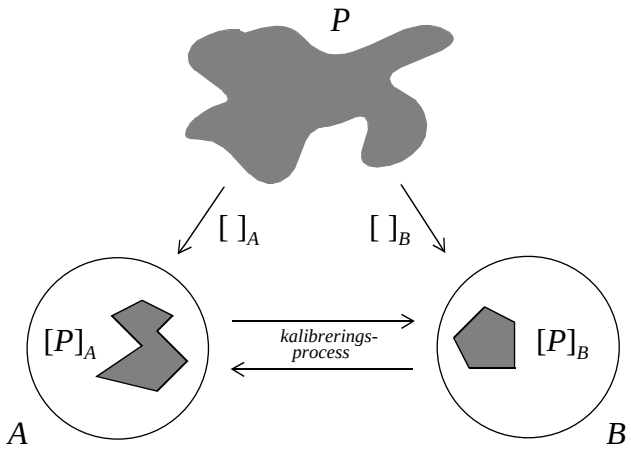

Fig. 1. Kalibreringsprocessen mellan två olika beskrivningar av samma sak.

Fig. (1) illustrerar den kalibreringsprocess (= sammanjämkningsprocess) som uppstår när man har två olika beskrivningar $[P]_A$ respektive $[P]_B$ av samma område P framställda av individerna (eller grupperna) A respektive B.

Begreppsmodellering av ett område handlar alltså om att beskriva hur vi tänker på området, och UML ger oss ett grafiskt språk för att rita upp dessa tankar. För en fördjupande beskrivning av UML se t.ex. [(18)].

1. Se http://www.omg.org/uml

1.2 Syfte

Syftet med att introducera begreppsmodellering i kursen “Intro till IT” är att ge dig som student ett verktyg för att beskriva din egen uppfattning om de matematiska begreppen och deras kopplingar till varandra och till andra ämnen. Det är väl känt att många studenter har svårigheter med detta, vilket antagligen har att göra med det sätt på vilket matematiken presenteras i skolan. Den traditionella matematikundervisningen ägnar väldigt mycket tid åt räkneträning och betydligt mindre tid åt att tydliggöra de bakomliggande begreppen. Naturligtvis måste man räkna, annars missar man poängen, men vad det handlar om är att försöka tydliggöra målet med räkningarna, i vilket sammanhang man räknar och varför man väljer ett visst sätt att räkna etc.

Man skulle kunna definiera en matematiker som en person som tänker i en dag för att slippa räkna i en timme. Trots detta marknadsförs skolmatematiken i huvudsak som räkning. Ett mera önskvärt sätt att presentera matematiken skulle kunna genomsyras av mottot “Tänk först och räkna sedan!” Ju mer man har tänkt, desto mindre måste man nämligen räkna.

Att förändra matematikundervisningen i denna riktning är ingen enkel uppgift, och det måste betonas att användning av begreppsmodellering innebär ett nytt pedagogiskt grepp i detta sammanhang. Förhoppningen är att genom att begreppsmodellera sin egen uppfattning av de matematiska begrepp som man möter kan man som student öka sin egen medvetenhet om de matematiska begreppens struktur, dvs om deras definitioner, egenskaper och relationer till varandra. Detta bör kunna hjälpa till att skapa en bättre överblick över det matematiska sammanhanget bakom räkningarna, samt bidra till en ökad förståelse för hur de matematiska begreppen tillämpas inom olika vetenskaper.

1.3 Begreppsbildning

Begreppsbildning är en förutsättning för all form av mental aktivitet. Poängen med begreppsbildning är följande: *Väl valda begrepp hjälper oss att bortse från det oväsentliga genom att skapa idealiserade strukturer som fokuserar på det väsentliga.*

Exempel på sådana begrepp är “punkt”, “linje”, “plan” och “cirkel” inom geometrin. De är idealiserade, i den meningen att de inte betecknar något som finns i verkligheten, men de uttrycker ett “extremt ideal” som gör att verkliga föremål kan rangordnas med avseende på hur väl de närmar sig detta ideal, dvs hur punktformiga, rätlinjiga, plana eller cirkulära de är i verkligheten. Dessa geometriska begrepp utgör de tidigaste kända historiska exemplen på matematiska idealiseringar. De uppfanns och började utforskas av grekerna på 600-talet före Kristus. Man kan faktiskt säga att detta innebar den abstrakta (= rena) matematikens födelse. Innan dess hade matematiken bestått uteslutande av erfarenhetsbaserade regler för praktiska tillämpningar inom t.ex. ekonomi (handel) och juridik (arvsskifte).

Begreppsbildningsprocessen har diskuterats och problematiserats av de flesta stora filosofer genom historien alltsedan Platon på 300-talet före Kristus. Platon blev så imponerad av de idealiserade matematiska strukturerna att han betraktade dem som mera verkliga än den sinnliga verkligheten själv. Man talar alltsedan dess om Platons idévärld.

Det skulle föra för långt att här diskutera de mer subtila (filosofiska) aspekterna på begreppsbildning och begreppsmodellering. Vi nöjer oss i stället med en starkt förenklad beskrivning som i huvudsak bygger på framställningen i [(16)], dit läsaren hänvisas för en mer ingående diskussion av dessa frågor. Vi börjar med att definiera vad vi menar med ett begrepp:

Definition 1: Ett begrepp är en representation av något som vi har upplevt eller kan föreställa oss, och som vi kan tillämpa på objekten (= föremålen = tingen) i vårt medvetande.

Exempel på begrepp är t.ex. “stol” och “odödlig”. Vi behöver bilda begrepp för att kunna tänka och kommunicera med varandra. Utan begrepp skulle vi inte kunna bygga upp något språk och därför inte kunna utbyta information i mänsklig mening.

De begrepp vi skapar inom ett visst område tjänar syftet att beskriva hur vi uppfattar området i fråga. Vi säger att vi bygger upp en *begreppsmodell* av området. Det är viktigt att betona att en begreppsmodell över ett område inte är en direkt beskrivning av området utan en beskrivning av hur vi väljer att uppfatta området - dvs vad vi tycker är väsentliga aspekter på området.

Ett sätt att skapa ett begrepp är genom att ange konkreta exempel på begreppet. Det är så vi lär oss de flesta begrepp när vi är små. Någon pekar t.ex. på ett föremål och säger “bil”, pekar på ett nytt föremål och upprepar ordet “bil”, etc, och så småningom lär vi oss att förstå begreppet bakom ordet (= beteckningen). Så fort vi har förstått ett begrepp kan vi sedan använda det som ett filter på vår omvärld, och t.ex. dela in alla objekt i världen i två grupper: “bilar” respektive “icke-bilar”. En sådan indelning kan ett förskolebarn i princip klara av att utföra.

Begreppet bil motsvaras alltså i detta fall av mängden av alla bilar.

Definition 2: Mängden av objekt som tillhör ett begrepp kallas för begreppets *extension.*

Definition 3: Att identifiera ett begrepp genom att iaktta likheter och särskiljande egenskaper hos en grupp av objekt kallas för att *klassificera* objekten.

Ett begrepp kan även skapas på ett abstrakt sätt, genom att formulera en s.k. definition:

Definition 4: Ett begrepps *definition* anger dess *intention*, dvs vilka egenskaper det vill uttrycka respektive avgränsa i förhållande till omgivningen.

Definition 5: Vi säger att ett begrepp kan *tillämpas* på ett visst föremål (= objekt) om föremålet uppfyller begreppets intention, dvs villkoren i dess definition.

Man skulle t.ex. kunna definiera begreppet “bil” på följande sätt:

Definition 6: En bil är en låda som har minst 3 hjul, plats för minst en person, samt kan framföras under personlig kontroll längs en väg.

Ett föremål kan alltså (i den införda begreppsmodellen) betraktas som en bil om det uppfyller villkoren i Definition (6).

I detta exempel kan vi notera flera saker som är typiska för begrepp:

• Ett begrepp måste definieras med hjälp av andra begrepp. Här definieras begreppet bil med hjälp av begreppen “låda”, “hjul”, “person”, “framföras” “personlig kontroll” och “väg”.

• Ett begrepp innehåller alltid förenklingar som lyfter fram vissa saker och utelämnar andra. Man säger att begreppen är *idealiserade.* Begreppsbildning är i själva verket nära kopplat till *abstraktion* (= bortseende). “The power of thinking is knowing what not to think about”.

• I den värld som omger oss är gränserna ständigt flytande. Var går t.ex. gränserna mellan dag och natt? Om vi kallar dessa skymning och gryning får vi samma problem med att definiera gränsen mellan t.ex. dag och skymning. Detta problem är typiskt för all form av begreppsbildning. Så fort vi inför ett antal begrepp så lägger vi i själva verket ett skarpt rutnät över den mjukt varierande verkligheten. Det är detta som är själva avsikten - att ge oss ett mentalt koordinatsystem mot vilket vår uppfattning om verkligheten blir lättare att beskriva. Ett väl definierat begrepp ger alltså upphov till (någorlunda) skarpa gränser. Ju bättre definierat ett begrepp är, desto lättare blir det att gå igenom föremålen i omvärlden och avgöra om de uppfyller begreppets intention eller inte.

• Definitionen av ett begrepp är alltid beroende av i vilket sammanhang begreppet ska användas. Det gäller hela tiden att bortse från det som är oväsentligt och fokusera på det väsentliga. Ett begrepp blir därför alltid kopplat till en grupp av personer och ett visst användningsområde (problemdomän). Om vi t.ex. betraktar begreppet “bil” så skulle det säkerligen definieras på olika sätt av t.ex. skattemyndigheterna respektive av ett biltillverkningsföretag. För skattemyndigheterna kanske det skulle räcka med att definiera en bil som en post i bilregistret, dvs ett registreringsnummer, en fabrikstyp, en årsmodell och en ägare, eftersom denna information kanske är allt som myndigheten är intresserad av för att kunna räkna ut hur ägaren ska beskattas för sitt bilinnehav. För ett biltillverkningsföretag är detta uppenbarligen en helt otillräcklig beskrivning av begreppet bil. I detta sammanhang blir andra egenskaper hos begreppet relevanta.

1.4 Att symbolisera begrepp

Till ett begrepp hör enligt ovan dess intention (= definition) och dess extension, dvs de föremål som satisfierar begreppet (= uppfyller villkoren i begreppets definition). Till ett begrepp hör även ett antal *symboler*, som används för att beteckna begreppet. Den viktigaste symbolen för ett begrepp är dess *namn*. Symboler ger oss ett effektivt sätt att referera till olika begrepp. De är speciellt användbara när vi vill kommunicera begreppen utan att använda långa definitioner.

Definition 7: Två symboler kallas *synonyma* om de betecknar samma begrepp.

I en modell av ett affärssystem kan t.ex. symbolerna “kund” och “klient” vara synonymer för samma begrepp.

Definition 8: Två begrepp kallas *homonyma* om de kan betecknas av samma symbol.

För de flesta företag betecknar t.ex. symbolen “avslutat köp” två olika begrepp. För en säljare innebär ett avslutat köp en handskakning med meningsfulla nickar, medan det för administrationsavdelningen inte existerar något avslutat köp förrän ett köpekontrakt har undertecknats och

godkänts. Homonyma begrepp kan ställa till stora problem vid begreppsmodellering, eftersom en symbol i detta fall bara kan få mening genom det sammanhang i vilket symbolen används.

1.5 Objekt eller värde - en fråga om identitet

Den verklighet vi upplever består av de begrepp som vi besitter och de föremål (= instanser = exempel) på vilka begreppen kan tillämpas. Man brukar inom datalogin skilja mellan föremål som har identitet (som t.ex. en bil) och föremål som inte har identitet (t.ex. ett heltal). Om man skapar en kopia av en bil, så blir den ändå en annan bil, men om man skapar en kopia av talet 7 så rör det sig ändå om samma tal.

Definition 9: En instans med identitet kallas en *objekt-instans* (eller ett *objekt*) medan en instans utan identitet kallas en *värde-instans* (eller ett *värde*).

Anledningen till att man inom datalogin betonar skillnaden mellan objekt och värden är att de måste behandlas på helt olika sätt i ett datorprogram. Ett objekt måste t.ex. alltid refereras till (pekas ut) - och får inte kopieras fritt, medan ett värde får kopieras utan inskränkning.

1.6 Typ och klass - två synonyma symboler för begrepp

Definition 10: Det begrepp vars extension är en mängd av instanser - och vars intention beskriver deras gemensamma struktur - kallas inom datalogin för instansernas *typ* eller *klass*[1].

Symbolerna “begrepp” “typ” och “klass” är alltså synonyma, eftersom de alla används för att beteckna begrepp. Anledningen till att vi nämner dessa synonymer är att de är så vanligt förekommande i begreppsmodelleringssamanhang. Vi kommer dock här att hålla oss till det mest intuitiva och låta symbolen “begrepp” beteckna begreppet begrepp, vilket ligger helt i linje med språkets symboliska natur.

1.7 Egenskaper och operationer för ett begrepp

När man beskriver ett begrepp så anger man vissa egenskaper som karaktäriserar de instanser som hör till begreppet. De statiska kännetecknen beskriver instansernas struktur och de dynamiska kännetecknen beskriver deras beteende. Till exempel kan vi välja att beskriva begreppet “penna” genom att säga att en penna kännetecknas av en viss färg och av att den går att skriva med.

Definition 11: De statiska kännetecken som hör till ett begrepp kallas dess *egenskaper* (= *attribut*) och de dynamiska kännetecknen kallas dess *operationer.*

1. Om det rör sig om objekt-instanser kallas typen för *objekttyp* eller *objektklass*, och om det rör sig om värde-instanser kallas typen för *värdetyp* eller *värdeklass.*

I vår modell av begreppet penna har vi alltså valt följande beskrivning: *Begreppsnamn*: **Penna**, *attribut*: **färg**, *operationer*: **skriv**. Begreppet **Penna** har alltså egenskapen **färg**, medan varje *instans* av begreppet **Penna** har ett *värde* på denna egenskap (= en viss färg, t.ex. grön).

Fig. 2. Begreppslåda med *namn, attribut* och *operationer* för begreppet Penna.

Fig.(2) visar hur man kan representera ett begrepp i UML, som en låda med tre avdelningar som representerar begreppets namn (**Penna**), attribut (**färg**) respektive operationer (**skriv**). Det är tillåtet att utelämna attributen och/eller operationerna och representera begreppet med en låda som enbart anger begreppets namn och egenskaper, eller begreppets namn och operationer. Notera att begreppsnamn normalt inleds med stor bokstav.

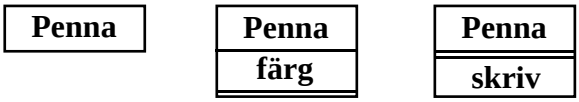

Fig. 3. Tre andra varanter av begreppsbox för begreppet Penna.

1.8 Egenskapsvärden och meddelanden för en instans

En instans av begreppet **Penna**, dvs en viss penna, karaktäriseras alltså av att den har värden på attributen och av att operationerna kan tillämpas på den. En namngiven begreppsinstans kan ritas upp i UML på något av de sätt som visas i Fig. (4) (med eller utan sina attributvärden):

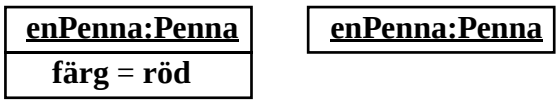

Fig. 4. Två olika sätt att i UML representera en instans av begreppet Penna.

Denna instans har namnet **enPenna** och är en instans av typen (= begreppet) **Penna**. Analogt kan en icke namngiven instans av samma typ representeras i UML i enlighet med Fig. (5):

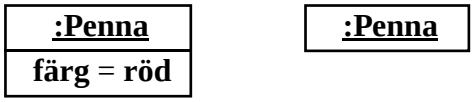

Fig. 5. Två olika sätt att representera en icke namngiven instans av begreppet Penna.

Den allmänna syntaxen för att ange en instans av en viss typ (= begrepp) är alltså

instansNamn:BegreppsNamn

En icke typad instans, dvs en instans av okänd typ kan beskrivas som i Fig. (6).

Fig. 6. En namngiven instans av okänd typ.

Att utföra operationen **skriv** på instansen **enPenna** representeras i UML på följande sätt:

enPenna . skriv( )

Man säger att man skickar meddelandet **skriv** till pennan **enPenna** och den reagerar då genom att utföra den metod (= algoritm) som genomför (= implementerar) operationen. Anledningen till att man skriver en parantes bakom meddelandet är att man där kan skriva in eventuell information (= argument) som måste skickas med för att operationen ska kunna utföras. Vi har i vårt lilla exempel antagit att det inte behövs någon sådan information. Därför är parantesen tom.

Operationerna som kan utföras på en instans brukar normalt inte ritas ut i "instanslådan". Detta beror på att alla instanser av ett visst begrepp kan utföra precis samma operationer. Operationerna hör därför naturligen ihop med själva begreppet. Attributens värden däremot varierar med instanserna och innehåller därför information som hör ihop med varje enskild instans. Attributen kallas därför ofta för *instansvariabler.*

1.9 Klassattribut

Ett begrepp kan även ha attribut vars värden inte varierar med instanserna. Sådana attribut kallas *klassattribut* (eller *klassvariabler*) i motsats till den ovan beskriva typen av attribut, som skulle kunna kallas *instansattribut.* Om vi t.ex. betraktar begreppet **Telefon**, så är det naturligt att betrakta egenskapen **telefonNummer** som ett instansattribut eftersom värdet på **telefonNummer** varierar med de olika instanserna. Om vi håller oss inom Sverige (dvs modellerar svenska telefoner), så har alla instanser samma **landsNummer** = 46. Värdet på **landsNummer** hör därför till själva klassen (= begreppet) **Telefon**. Det är därför ett klassattribut vilket representeras i UML genom understrykning. Man kan även representera ett klassattribut genom att sätta ett $-tecken före attributets namn, vilket är praktiskt i vissa sammanhang. Notera att värdet på ett klassattribut kan anges i begreppslådan, så som visas i Fig. (7), eftersom det ju är gemensamt för alla instanser av begreppet.

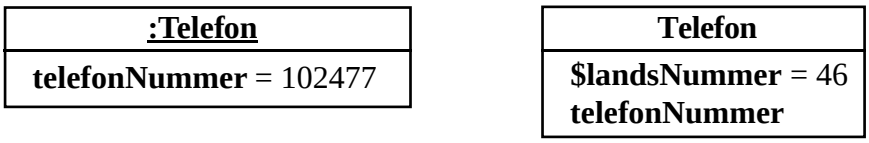

Fig. 7. Instansattribut och klassattribut för begreppet Telefon.

1.10 Typning av attribut och operationer

I en begreppslåda kan man även ange attributens typ, vilket görs på det sätt som visas i Fig. (8). Här har vi (för enkelhets skull) antagit att både **landsNummer** och **telefonNummer** är av typ **Heltal**.

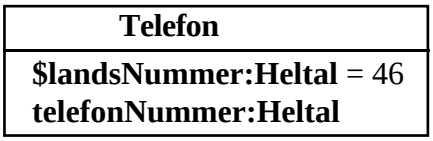

Fig. 8. Typade instans- och klassattribut för begreppet Telefon.

Även operationer kan typas. För att kunna tillämpas på en instans behöver operationer i allmänhet oftast indata av någon typ och lämnar oftast ifrån sig (= returnerar) utdata av någon typ (i form av ett resultat). Operationens namn, typen av dess indata, och typen av dess utdata beskriver tillsammans operationens typ och kallas ofta för dess *signatur*. I UML kan man ange signaturen för en operation i begreppslådan enligt följande syntax:

operationsnamn(**IndataTyp**):**UtdataTyp**

Om vi antar att telefoner kan ringa, så har vi en operation **ring** som hör till begreppet **Telefon**. Låt oss anta att för att utföra en ring-operation i vår telefonmodell behöver man ge telefonen ett nummer (av typ **Heltal**), och låt oss anta att resultatet av ring-operationen är en signal (av typ **Signal**)[1]. Signaturen för ring-operationen blir då **ring(Heltal):Signal**, och begreppslådan för Telefon, med typade attribut och operationer, får utseendet i Fig. (9).

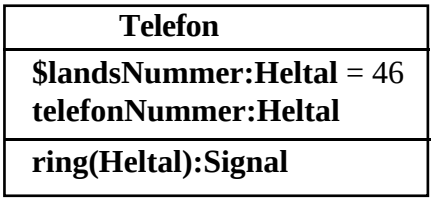

Fig. 9. Typade attribut och operationer för begreppet Telefon.

Om en operation inte behöver några indata eller inte returnerar några utdata så anges detta genom typen **Void**. Ovan antog vi att operationen **skriv** kunde tillämpas på instansen **enPenna** utan att någon ytterligare information behövde tillföras. Om vi antar att skriv-operationen inte heller lämnar ifrån sig några utdata får den alltså signaturen **skriv(Void):Void**, och vi kan alltså ange typinformation om både attribut[2] och operationer för begreppet **Penna** enligt Fig. (10).

1. Av skäl som vi inte ska gå in på här brukar man inte (datalogiskt sett) betrakta signalen som ett returvärde från ring-operationen utan snarare som en *sidoeffekt* av densamma.
2. Notera att **färg** är attributets namn och **Färg** är dess typ.

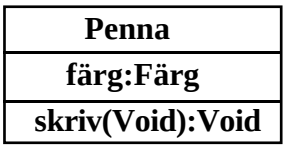

Fig. 10. Typade attribut och operationer för begreppet Penna.

Vi kan även ange antalet attribut i begreppslådan. Antag t.ex. att en penna har plats för 4 olika färger som kan väljas genom att trycka på olika knappar på pennan. Då kan vi ange detta i begreppslådan, antingen genom att explicit skriva upp attributen som t.ex. **färg1**, **färg2**, **färg3**, **färg4**, eller genom att skriva **färg[4]** som i Fig. (11).

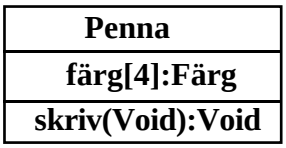

Fig. 11. Fyra färg-attribut för begreppet Penna.

Om en penna kan vara av enfärgs-, tvåfärgs-, trefärgs- eller fyrfärgs-typ så kan vi ange detta som i Fig. (12).

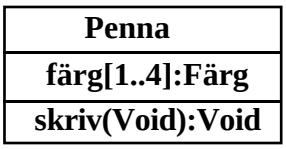

Fig. 12. enPenna kan ha mellan en och fyra olika färger.

1.11 Objektorientering

På 1970-talet skilde man noga mellan funktioner och datatyper. Man konstruerade datorprogram i form av stora funktioner genom vilka sedan man skickade data av olika slag. Funktionen tog reda på vilken typ av data som kom in, och behandlade sedan varje typ på sitt eget speciella sätt genom att skicka det vidare till en lämplig hjälpfunktion (= subrutin). Detta sätt att tänka gav upphov till program som var väldigt hårt kopplade till den specifika problemställningen.

Ett annat sätt att tänka på ett problemområde är i termer av olika föremål som samverkar med varandra för att åstadkomma förändringar av olika slag. Dessa föremål kan enligt ovan betraktas som instanser av olika begrepp som beskriver deras respektive struktur. Vare sig instanserna är objekt eller värden så karaktäriseras de instanser som hör till samma begrepp av vissa gemensamma egenskaper och vissa gemensamma beteendemönster.

Detta objektorienterade synsätt har under de senaste 20 åren fått en stark förankring inom datalogin. Det visar sig att datorprogram som är uppbyggda som en samling av samverkande objekt kan göras mycket mer flexibla och lättare att förändra och bygga ut i jämförelse med datorpro-

gram som är uppbyggda på det mer traditionella sättet, dvs som ett antal övergripande funktioner som utbyter data med varandra. I dagens föränderliga datorvärld med ständigt nya möjligheter och krav på samverkan mellan olika program är detta en oerhört viktig egenskap som har lett till att objektorientering har fått ett mycket stort genomslag inom programvaruindustrin.

Att beskriva ett problemområde i termer av växelverkande objekt kallas att genomföra en *objektorienterad analys* (OOA) av problemet. Att designa en modell av objekten och deras växelverkan som är avsedd att köras på en dator kallas att genomföra en *objektorienterad design* (OOD) av lösningen, och att skriva programkod som genomför denna design i något programspråk kallas för att genomföra en *objektorienterad implementation* (OOI) av densamma. OOA beskrivar alltså *vad* som ska utföras, OOD beskriver *hur* det ska gå till och OOI *utför* det i något lämpligt programspråk.

1.12 Relationer mellan begrepp

Vi har ovan beskrivit hur man kan fånga upp det som är gemensamt för ett antal olika instanser genom att bilda begrepp. Vi ska nu titta lite närmare på hur begreppen kan vara relaterade till varandra.

1.12.1 Klassificering

Att identifiera begreppet (= typen = klassen) som förenar ett antal instanser kallas enligt Definition (3) för att *klassificera* instanserna. När vi beskriver ett begrepp med hjälp av ett substantiv använder vi oftast artikellös form (singularis) av substantivet för att beteckna själva begreppet (t.ex. Bil) medan vi använder obestämd eller bestämd form av substantivet (singularis eller pluralis) för att betecka instanser av begreppet (t.ex en bil, två bilar, bilen, bilarna). Vi har redan använt denna konvention ovan (tillsammans med konventionen att inleda namnet på ett begrepp med stor bokstav) när vi diskuterade begreppet Penna och instansen enPenna.

Klassificering är en sorts *relation* mellan två begrepp - i detta fall en relation mellan begreppets instanser och dess typ (dvs begreppet i sig själv). När vi använder namnet klassificering för denna sorts relation så har vi beskrivit den på ett riktat sätt - från instanserna i riktning mot typen. Ett icke-riktat namn på klassificeringsrelationen är *instans/typ relation.*

Denna relation är ett exempel på en s.k. *beroenderelation* (= dependency) i UML. Ett begrepps instanser är beroende av dess typ Beroenderelationer betecknas i allmänhet med en streckad pil som visas i Fig. (13). Här har vi givit denna typ av beroende namnet <<ärEn>> med hjälp av en s.k. *stereotyp,* vilket är ett sätt att införa nya förtydligande beskrivningar av begrepp och relationer i UML. Se kap. (1.15.1) för vidare diskussion av stereotyper.

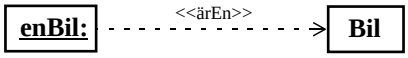

Fig. 13. Klassificering som en stereotypad beroenderelation mellan instans och typ.

Beroendet mellan instansen **enBil** och typen **Bil** utläses alltså: **enBil** <<ärEn>> **Bil**, vilket språkligt sätt är en s.k. *tautologi* (påstående som alltid är sant). Avsikten är här att försöka modellera relationerna mellan begreppen på samma sätt som vi beskriver dem i ord. Mer om detta nedan.

1.12.2 Generalisering/Specialisering

Klassificering innebär alltså att olika instanser samlas ihop under samma begrepp om de kan karaktäriseras av gemensamma egenskaper och ett gemensamt beteende. På liknande sätt kan man iaktta strukturella likheter hos olika begrepp. Låt oss t.ex. betrakta begreppen **Bil**, **Båt** och **Flygplan**, och anta att de kan karaktäriseras enligt Fig. (14). Bilar, båtar och flygplan har i denna beskrivning både **förare** och **ägare**, bilar har dessutom **hjul**, båtar har **köl** och flygplan har **vingar**.[1] Bilar, båtar och flygplan kan dessutom **navigera** och **åka**. Låt oss anta att de navigerar på samma sätt (t.ex. genom att använda positioneringssystemet GPS) men att de åker på olika sätt (vilket är ett ganska rimligt antagande).

Fig. 14. Karaktäristiska egenskaper och beteenden hos tre olika begrepp.

Vi kan då sammanföra den gemensamma strukturen hos de tre begreppen (= klasserna) **Bil**, **Båt** och **Flygplan** och bilda det *generaliserade begreppet* (= *superklassen*) **Fordon**, som karaktäriseras av de egenskaper och operationer som är gemensamma för begreppen **Bil**, **Båt** och **Flygplan**. Notationen för detta i UML framgår av Fig. (15). Triangeln som pekar på begreppet **Fordon** anger alltså att **Fordon** är ett generaliserat begrepp i förhållande till **Bil**, **Båt** och **Flygplan** eller - vilket är ett annat sätt att uttrycka samma sak - att begreppen **Bil**, **Båt** och **Flygplan** är *specialiserade begrepp* (= *subklasser*) i förhållande till begreppet **Fordon**. Man säger att de specialiserade begrepen *ärver* struktur från det generella begreppet. Vi ser hur detta sätt att representera begreppen skapar större tydlighet och ekonomi genom att vi slipper upprepa den gemensamma strukturen. Endast det som avviker (= är speciellt) behöver representeras på den speciella nivån. I de objektorienterade programmeringsspråken finns ett inbyggt stöd för denna arvsmekanism vilket underlättar programmeringsarbetet och minskar riskerna för tvetydigheter. Dessutom blir det lättare att lägga till nya specialiseringar - i vårt exempel nya typer av **Fordon**. Om vi t.ex. inför **Motorcykel** som en specialisering av **Fordon** så är ju egenskaperna **förare** och **ägare** samt beteendet **navigera** redan representerat på fordonsnivån. Om det är så att motorcyklar navigerar på ett annat sätt än fordon, så är det bara att upprepa denna operation i klassen **Motorcykel**. Detta kallas att *överrida* det generella beteendet i den specialiserade klassen.

1. Vi ska i nästa avsnitt se hur vi kan modellera dessa egenskaper som *associationer* med andra begrepp, men låt oss för ögonblicket välja att modellera dem som attribut.

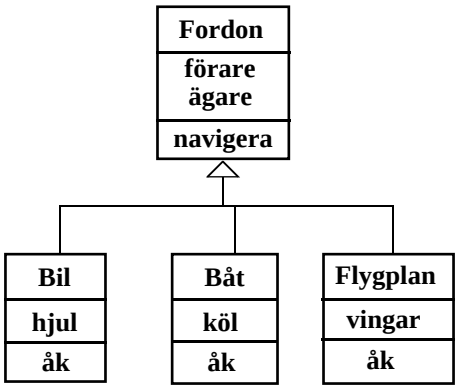

Fig. 15. Det generaliserade begreppet Fordon fångar upp den gemensamma strukturen.

Den begreppsrelation som avbildas i Fig. (15) kallas alltså *generalisering* om man läser den från **Bil**, **Båt** och **Flygplan** i riktning mot **Fordon** och *specialisering* om man läser den åt det motsatta hållet. När man inte vill införa någon riktning i sin beskrivning så brukar man kalla relationen för *gen/spec*. Om vi inför namnet <<Sorts>> som en synonym för generalisering så kan vi koppla ihop klassificering och generalisering till en beskrivning som överensstämmer med hur vi talar om dessa begreppsrelationer i det vanliga språket:

enBil <<ärEn>> **Bil** <<Sorts>> **Fordon**
enBil <<ärEnSorts>> **Fordon**

Denna beskrivning är avbildad i Fig. (16).

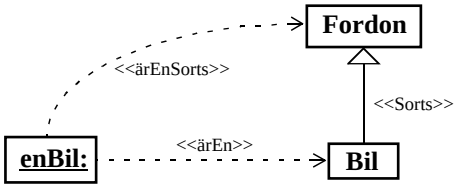

Fig. 16. enBil <<ärEnSorts>> Fordon.

Lägg märke till att namngivningen i Fig. (15) är sådan att vi kan läsa ihop det speciella och det generaliserade begreppet:**BilFordon**, **BåtFordon** och **Flygplan(s)Fordon** är korrekta uttryck - även om de är omständiga. Detta är en avspegling av hur det vanliga språket fungerar. När man begreppsmodellerar är det bra att försöka namnge generaliserade begrepp så att man kan läsa ihop det specialiserade begreppsnamnet med namnet på det generaliserade begreppet. Detta ökar tydligheten och underlättar den mentala navigationen i modellen. Det överensstämmer dessutom med “arvsprincipen” att låta det specialiserade begreppet överta namnet på det generella med ett tillägg som beskriver själva specialiseringen.

1.12.3 Association

I föregående avsnitt modellerade vi det förhållandet att ett fordon har en ägare genom att införa **ägare** som ett attribut i begreppet **Fordon**. Eftersom fordonsägare normalt är personer, vilka både har egna egenskaper och ett eget beteende är det bättre att i stället modellera förhållandet mellan olika fordon och deras ägare som en *association* mellan dessa begrepp.

En association mellan två begrepp uttrycker det faktum att instanser av dessa begrepp är *länkade* (= kopplade) till varandra på något sätt. Låt oss anta att i vårt exempel äger t.ex. **pelle enBåt** medan **anna** äger **enBil**. Detta uttrycks genom en association mellan motsvarande begrepp, dvs mellan **Fordon**[1] och **Person**, och representeras i UML med en heldragen linje mellan dessa begrepp, på det sätt som visas i Fig. (17).

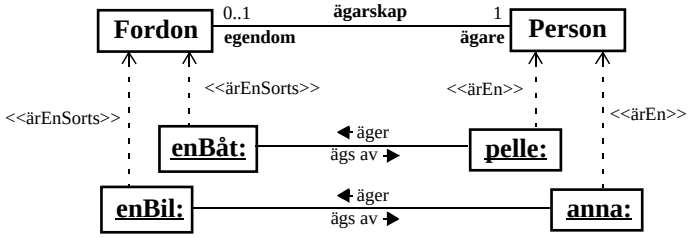

Fig. 17. Ägarskap som en association mellan begreppen Fordon och Person.

Associationen har fått *namnet* **ägarskap** och i dess ändar står *rollnamnen* **ägare** respektive **egendom**. Detta betyder att i denna association spelar en person rollen av ägare relativt ett fordon medan ett fordon spelar rollen av egendom relativt en person. Associationen exemplifieras av länken mellan **pelle** och **enBåt** samt länken mellan **anna** och **enBil**. Dessa länkar är alltså *instanser* av associationen **ägarskap**. Precis som ett begrepp uttrycker den gemensamma strukturen hos sina instanser så uttrycker en association den gemensamma strukturen hos sina länkar. En del av denna uppgift är att uttrycka hur många olika instanser som en given instans kan vara länkad till på den andra sidan. För detta ändamål använder sig associationen av *multiplicitetssymboler*. Siffran 1 bredvid begreppet **Person** innebär att varje instans av typ **Fordon** ägs av exakt en instans av typ **Person**, medan uttrycket 0..1 bredvid begreppet **Fordon** innebär att varje instans av typ **Person** äger 0 eller 1 instanser av typ **Fordon**.

Låt oss anta att **anna** säljer sin bil till **pelle**. Då blir ju **pelle** länkad till två fordon medan **anna** inte blir länkad till något fordon alls. Pelles länkstruktur bryter därmed mot ägarskapsassociationens multiplicitet eftersom han inte är länkad till 0 eller 1 fordon som multiplicitetsangivelsen föreskriver. Om vi vill tillåta en sådan förändring inom ramen för vår modell så måste vi ändra multipliciteten för **egendom** från 0..1 till 0..2 i enlighet med Fig. (18).

Multipliciteter kan anges explicit som en lista (t.ex 1, 3, 6), eller som intervall (t.ex. 0..17). Symbolen ∗ (stjärna) betyder att multipliciteten är oinskränkt, dvs en instans på den andra sidan kan vara länkad till 0, 1 eller flera instanser på den stjärnmärkta sidan utan några begränsningar.

1. Associationen är ju gemensam för både **Bil** och **Båt** och ska därför uttryckas i det generaliserade begreppet **Fordon**.

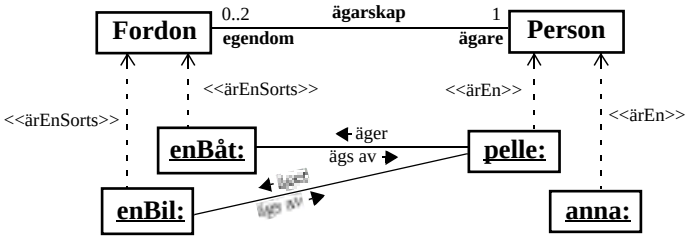

Fig. 18. Ägarskapets multiplicitet tillåter ägande av högst två fordon per person.

1.12.4 Aggregation

Associationer brukar ibland beskrivas som "lösa förbindelser". Ägarskapslänkarna ovan förändras t.ex. av att fordonen köps och säljs. Dessutom är det så att det som inträffar med en instans på den ena sidan av en länk inte behöver påverka instanserna på den andra sidan av länken. Om t.ex. en bil kör i diket händer det inte med nödvändighet något med dess ägare.

Det finns dock en speciell form av association som innebär en fastare koppling mellan de länkade instanserna. Den kallas *aggregation* (eller *helhet/del relation*) och uttrycker det faktum att en instans består av delar som är andra instanser. Så består t.ex. en bil av (bl.a.) fyra hjul och en motor, medan ett hjul (liksom en motor) är en del av en bil. Om t.ex. bilen fattar eld så påverkas även hjulen och motorn av detta. Bilen bildar en *helhet* som omfattar sina *delar* (de 4 hjulen och motorn). En aggregation kan alltså betraktas som en association med *ansvar* (en ansvarsfull förbindelse). Helheten är ansvarig för sina delar. Händer det något med helheten så påverkas även delarna.

En aggregation representeras i UML som en association (heldragna linjer) med en romb som utmärker helheten (= aggregatet) på det sätt som framgår av Fig. (19). Aggregationen utläses **Bil** <<helhet för>> **Motor** och **Bil** <<helhet för>> **Hjul** åt det ena hållet och **Motor** <<del av>> **Bil** respektive **Hjul** <<del av>> **Bil** åt det andra.

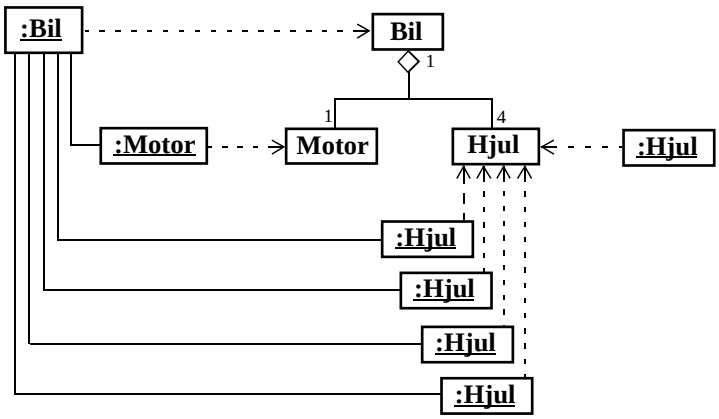

Fig. 19. En bil består av fyra hjul och en motor. En motor och 4 hjul är delar av en bil.

För att se hur vi modellerar aggregationer i språket kan vi införa beteckningen <<delAv>> för en aggregation med riktning från delarna mot helheten, så som visas i Fig. (20). Om vi dessutom låter den obestämda artikeln <<en>> beteckna "instantieringsavbildningen" (som går från ett begrepp till någon av dess instanser) så kan vi ur Fig. (20) utläsa följande relationer:

:Motor <<ärEn>> **Motor** <<delAv>> **Bil** <<en>> **:Bil**

Om vi nu namnger instanserna med obestämd artikel framför typen (i enlighet med vår tidigare införda konvention) så kan ovanstående relationer sammansättas till:

enMotor: <<ärEnDelAv>> **enBil:**

Detta motsvarar vårt sätt att beskriva aggregationer i det vanliga språket.

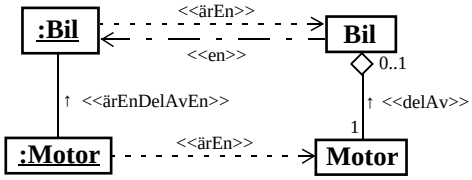

Fig. 20. En motor är en del av en bil.

Det är värt att notera att namngivningen vid aggregation fungerar så att delarnas namn alltid kan inledas med helheten: **Hjul** kan kallas **BilHjul** och **Motor** kan kallas **BilMotor**. Även denna namnkonvention används i det vanliga språket och är värd att eftersträva när man begreppsmodellerar, eftersom tydligheten ökar och det blir lättare att navigera mentalt i begreppsdiagrammet.

1.13 Tre hierarkier och resten anarki

Relationerna mellan de fyra olika sorters begreppsrelationer som vi har infört ovan framgår av Fig. (21), som är ett exempel på ett s.k. *metadiagram* (= diagram om diagrammens struktur). Vi använder generalisering för att generalisera generalisering, klassificering och association till en allmän relation. Även här fungerar namngivningen så att vi kan säga **Generalisering(s)Relation**, **Klassificering(s)Relation**, **Association(s)Relation** och **Aggregation(s)Relation**.

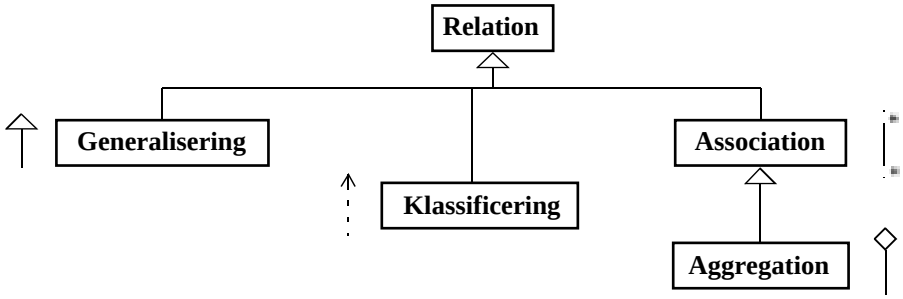

Fig. 21. De grundläggande begreppsrelationerna i UML. Notationen för de olika typerna av relationer är avbildade bredvid sina respektive begreppslådor.

Associationer bildar generella grafer utan några som helst restriktioner. Detta innebär att motsvarande länkstrukturer kan bilda *cykler* (= slutna kedjor), dvs det finns situationer där instansen *a* är länkad till instansen *b*, som är länkad till instansen *c* som i sin tur är länkad tillbaka till instansen *a*, vilket skulle kunna beskrivas som en "relationsanarki". Cykliska (= cirkulära) beroenden är i allmänhet betydligt svårare att hantera datalogiskt än icke-cykliska sådana, som ju alltid tar slut efter ett ändligt antal steg. Det är därför skönt att veta att var och en av de tre andra typerna av relationer som beskrivs i Fig (21), dvs generalisering, klassificering och aggregation, aldrig kan bilda riktade cykler. Till exempel kan en bil betraktas som en sorts fordon och en amfibiebil kan betraktas som en sorts bil, men ett fordon kan aldrig betraktas som en sorts amfibiebil. Analogt är ett hjul en del av en bil och en navkapsel är en del av ett hjul, men en bil kan aldrig vara en del av en navkapsel. Man brukar säga att generalisering, klassificering och aggregation är *hierarkiska* relationer eftersom de var för sig bygger upp en hierarki av riktade beroenden. Detta uttrycks i Fig. (22), där RIG står för Riktad Icke-cyklisk Graf och betyder en graf som saknar riktade cykler.

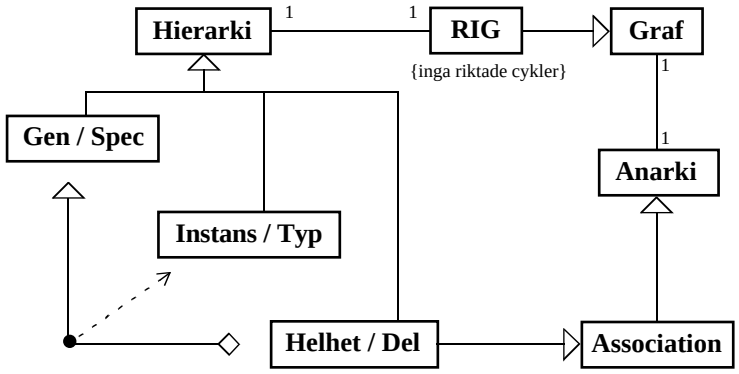

Fig. 22. Tre sorters hierarki och resten anarki.

Betrakta ett begrepp *B* i en viss begreppsmodell. De begrepp som är relaterade till *B* i modellen bildar en *begreppsmässig omgivning* till *B*. Eftersom tre av de begreppsrelationer vi har infört är hierarkiska, kan vi säga att i vår modell av begreppsrelationer finns det tre hierarkiska begreppsdimensioner och sex hierarkiska begreppsriktningar (två i varje begreppsdimension). Dessa kan användas för att bygga upp den hierarkiska delen av varje begreppsmässig omgivning till ett givet begrepp.

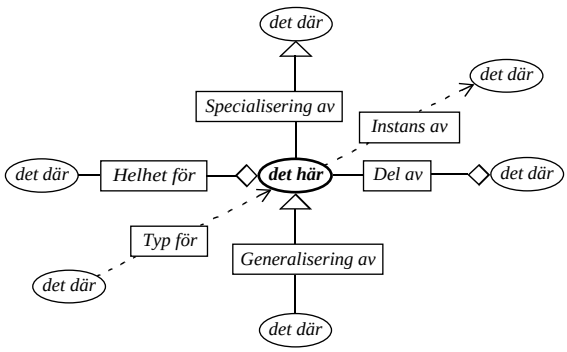

Fig. 23. De sex hierarkiska begreppsriktningarna från *det här* begreppet till *det där* begreppet.

De sex hierarkiska begreppsriktningar som omger varje begrepp är grafiskt beskrivna och namngivna i Fig. (23). Observera att det inte behöver finnas begrepp i varje sådan riktning. Ritktningarna anger endast möjligheterna för hierarkiska begreppsbildningar i relation till ett givet begrepp.

Relationerna i Fig. (23) kan uttryckas: det här är en *specialisering av* det där, det här är en *generalisering* av det där, det här är en *del av* det där, det här är en *helhet för* det där, det här är en *instans av* det där och det här är en *typ för* det där.

Poängen med hierarkier är att de har en början och ett slut. Ett begreppsdiagram som utnyttjar ovanstående hierarkier är därför lättare att överblicka än den allmänna "associationsanarki" som uppstår när begreppen kan kopplas ihop med varandra hur som helst - som i en vanlig mind-map.

1.14 Att modellera aktiviteter

Ett problemområde handlar ofta om någon typ av system som förändras över tiden. Att modellera ett sådant system innebär dels att namnge delarna i systemet samt beskriva deras egenskaper och relationer, dels att beskriva hur dessa delar växelverkar (= interagerar) med varandra över tiden. Den förstnämnda beskrivningen brukar kallas den *statiska* och den sistnämnda den *dynamiska* modellen av systemet.[1] Man kan säga att den statiska modelleringen handlar om att beskriva *vad som förändras* (= bitarna i pusslet) medan den dynamiska modellering handlar om att beskriva *när de förändras.* Den statiska systemmodellen är därför en logisk förutsättning för den dynamiska.

Hittills har vi enbart diskuterat statisk begreppsmodellering i UML. Detta språk erbjuder även ett flertal olika sätt att uttrycka dynamiska aspekter av ett system. Vi ska här begränsa oss till att kortfattat beskriva s.k. *aktivitetsdiagram,* vilka är nära besläktade med de klassiska *flödesdiagrammen* (eng. workflow diagrams) som används för att beskriva tidsförlopp (= processer).

Ett aktivitetsdiagram beskriver ett tidsförlopp i termer av ett antal aktiviteter som är kopplade i ett flöde. Precis som i ett flödesdiagram kan aktivitetsflödet förgrena sig åt olika håll och välja olika vägar genom diagrammet beroende på utfallet av olika tester. Aktivitetsflödet kan även dela upp sig i parallella flöden vilka pågår samtidigt. Detta uttrycks med hjälp av s.k. *synkroniseringslinjer* (eng. *synchronization bars*). Aktiviteter representeras av lådor med runda hörn, tester av snedställda rutor med skarpa hörn och synkroniseringslinjer av tjocka heldragna streck.

Ett exempel på ett aktivitetsdiagram återges i Fig. (24), som beskriver en modell av hur vi kan använda oss av mentala modeller. Vi skapar ständigt förenklade representationer av fenomenen i vår omvärld i form av olika mentala modeller. Dessa modeller möjliggör grova (men effektiva) förutsägelser av hur de bakomliggande fenomenen kommer att uppföra sig. Låt oss t.ex. anta att jag ser en man komma gående mot mig på gatan. Min mentala modell av detta fenomen

1. Om vi modellerar på ett *objektorienterat* sätt så måste vi dessutom beskriva växelverkan mellan systemets olika delar i termer av *operationer* som ägs av (= hör till) de olika objektens respektive typ, och som sätts igång genom att objekten tar emot *meddelanden* av varandra.

leder mig till en mängd olika förväntningar, som t.ex. att mannen kommer att passera på min högra sida. Samtidigt som jag registrerar mannens rörelse via mina sinnesintryck, kontrollerar jag mina modellförväntningar. Så länge som han rör sig i enlighet med förväntningarna kan denna process iterera (= loopa) i bakgrunden av mitt medvetande.

Om jag däremot upplever en skillnad (= diff) mellan mina sinnesintryck och mina modellförväntningar så tvingas jag öka uppmärksamheten och utföra kontrollen i förgrunden av mitt medvetande. Om det nu visar sig att det var "falsk alarm", så kan jag minska uppmärksamheten igen och låta processen återgå till bakgrunden. Om skillnaden däremot överlevde den medvetna kontrollen så måste jag förändra min modell av mannens rörelse för att t.ex. kunna anpassa min egen rörelse så att vi inte kommer att kollidera med varandra. När mina sinnesintryck återigen överensstämmer med mina modellförväntningar kan jag låta processen återgå till min mentala bakgrund, där den inte kräver så mycket uppmärksamhet (= bandbredd) av mitt medvetande.

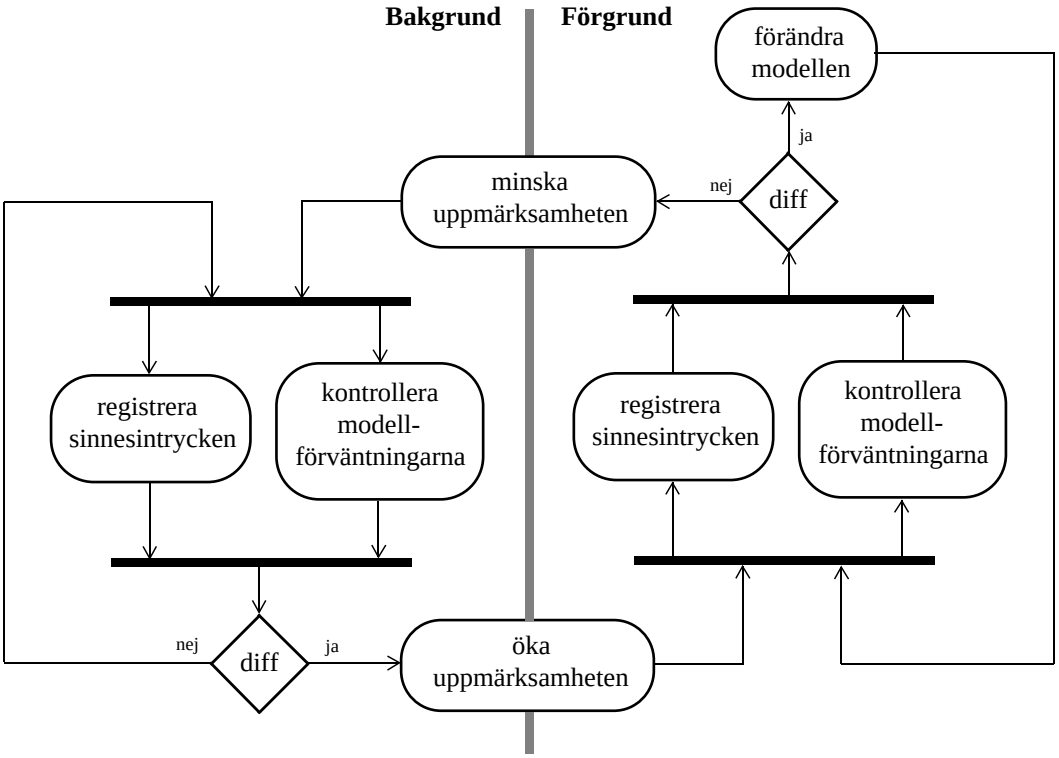

Fig. 24. Ett aktivitetsdiagram som modellerar ett sätt att använda mentala modeller.

Ett aktivitetsdiagram uttrycker de nödvändiga sekvensieringsvillkoren mellan de olika delarna (= aktiviteterna) i en process. Aktivitetsdiagram ger därför tillfälle att upptäcka möjligheter att utföra olika aktiviteter parallellt (= samtidigt) med varandra. De har därför blivit populära vid modellering av affärsprocesser, som traditionellt ofta utförs på ett onödigt sekvensiellt sätt. Att parallellisera delar av en affärsprocess innebär en möjlighet att förkorta exekveringstiden och därigenom öka effektiviteten hos processen.

1.15 Några olika sätt att utvidga UML

UML är ett standardspråk för begreppsmodellering som har vuxit fram ur ett antal olika modelleringsmetoder för utveckling av datorprogram. Det är emellertid inte möjligt för ett slutet språk att vara tillräckligt nyanserat för att kunna uttrycka alla tänkbara modellstrukturer inom alla tänkbara begreppsområden av idag och inom en oöverblickbar framtid. På grund av detta är UML designat som ett öppet språk, med möjlighet för användaren att utvidga det på ett kontrollerat sätt. De utvidgningsmekanisner som tillhandahålles i UML innefattar

• *Stereotyper.*
• *Begränsningar* (eng. *constraints*).
• *Namngivna värden* (eng. *tagged values*).

Dessutom kan man förse varje UML-element med kommentarer i form av *noter* (eng. *notes*).

1.15.1 Stereotyper

En stereotyp är en modifikation av ett existerande modellelement i UML som man själv kan införa. Vi har t.ex. använt stereotyper i Kap. (1.12.1), där vi införde klassificering som en stereotypad beroenderelation (med namnet <<ärEn>>) mellan en instans och dess typ. En stereotyp anges genom att sätta dess namn innanför dubbelhakar (guillemets) som t.ex. <<ärEn>>.

Stereotyper utvidgar vokabulären i UML och gör det möjligt att konstruera nya typer av byggstenar som är skräddarsydda för ett visst problem. När man stereotypar ett element, som t.ex. en klass eller en relation, så skapar man i själva verket en ny typ av element. Detta element kan ha sina egna speciella egenskaper, eftersom varje stereotyp kan tillhandahålla sina egna namngivna värden. Det nya elementet kan även ha sin egen speciella semantik (= mening), eftersom varje stereotyp kan tillhandahålla sina egna begränsningar. Slutligen kan det nya elementet även ha sin egen speciella notation, eftersom varje stereotyp kan tillhandahålla sin egen ikon.

1.15.2 Villkor och begränsningar

I Fig. (22) finns en klammerförsedd text: **{inga riktade cykler}**, vilket betyder att varje instans av typ RIG saknar riktade cykler. Detta är ett exempel på hur man kan uttrycka en s.k. *begränsning* eller *inskränkning* (eng. *constraint*) i UML. En begränsning gäller för alla instanser av det begrepp intill vilket begränsningen förekommer. En begränsning utvidgar semantiken hos ett UML element och möjliggör tillägg av nya regler såväl som modifikation av existerande sådana.

1.15.3 Namngivna värden

Ett *namngivet värde* (eng. *tagged value*) utvidgar egenskaperna hos ett UML element och tillåter oss att tillföra ny information vid specifikationen av elementet. Ett namngivet värde är inte detsamma som ett (instans)attribut, ty dess värde gäller för elementet självt och inte för dess instanser. Ett namngivet värde kan därför närmast liknas vid ett klassattribut. I sin enklaste form

uttrycks ett namngivet värde som en textsträng **{namn = värde}** innesluten i måsvingeparanteser som placeras i närheten av namnet på det element som det tillhör. Strängen innehåller ett namn, en separator (symbolen =) och ett värde på namnet. En av de vanligaste användningarna av namngivna värden i programvarumodellering är för att ange egenskaper som är relevanta för generering, konfigurering och versionshantering av programkod.

1.15.4 Noter

En not är en symbol för att uttrycka begränsningar eller kommentarer som tillhör ett element eller en samling element [se Fig. (25)]. En not som uttrycker en kommentar har ingen semantisk effekt på (dvs förändrar inte meningen av) den modell till vilken den hör. Inom mjukvarumodellering används kommentarsnoter ofta för att specificera saker som systemkrav, observationer, granskningar och förklaringar. En not kan innehålla en godtycklig kombination av text och grafik.

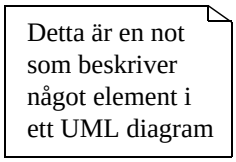

Fig. 25. En not i UML.

2 Matematikens struktur

2.1 Teorier och modeller

Vi har i Kapitel (1.3) ovan beskrivit hur grekerna på 600-talet f.Kr. införde idealiserade begrepp som punkt, linje och plan i matematiken, vilken innan dess hade varit uppbyggd av ett antal erfarenhetsbaserade regler och metoder. Grekerna inledde på detta sätt den utveckling från det konkreta i riktning mot det abstrakta (teoretiska), som har karaktäriserat matematiken ända sedan dess. Idag formuleras de matematiska teorierna på ett mycket allmänt (= generellt = abstrakt) sätt. Man talar inte om vad ett matematiskt begrepp egentligen är för något utan begreppet definieras enbart genom att man beskriver dess egenskaper och/eller hur det uppför sig i förhållande till andra begrepp. Följande beskrivning av matematisk teoribildning är hämtad från kursboken i kursen Matematik I.[1]

> Vid framställningen av teoretiska ting brukar man kräva att varje begrepp man handskas med på något sätt definieras utifrån "enklare" begrepp. Ett tungt skäl härför är att man vill undvika varje form av missförstånd om begreppets innebörd. Det säger sig dock självt att man förr eller senare måste utgå från några "enklaste" begrepp. Det enda som kan anges begräffande dessa är deras *egenskaper* - inte vad de "egentligen" är för någonting. Beskrivningen av dessa egenskaper kallas *postulat* eller *axiom*.
>
> Valet av "enklaste" begrepp och dessas axiom är egentligen mera en konventionsfråga än något som är "av naturen givet". Man låter gärna begreppen anknyta till något, i sinnevärlden eller den fysikaliska verkligheten, som man allmänt anser sig ha grepp om intuitivt, och ser till att postulaten från den synpunkten ter sig odiskutabla.[2]
>
> Angående uppsättningen av postulat kräver man vidare att den är *fullständig*, vilket betyder att man senare inte kommer att använda sig av några andra egenskaper hos begreppen än de som är omnämnda i postulat och definitioner eller är logiska konsekvenser av dessa. Viktigt är också att postulaten är *motsägelsefria*, dvs att man inte ur dem kan härleda två påståenden som strider mot varandra. Att bevisa motsägelsefriheten med enbart logiska metoder är i allmänhet ogörligt. Man får dock en viss garanti för att den föreligger om man knutit postulaten till något i sinnevärlden el dyl - man har då en *modell* för vilken postulaten "gäller", varför dessa inte bör kunna motsäga varandra. Ofta nöjer man sig med att visa en "relativ motsägelsefrihet", nämligen att det axiomsystem man har är motsägelsefritt om något annat, "enklare" eller "allmänt vedertaget" axiomsystem är det.
>
> Begrepp som inte finns i postulaten måste definieras med hjälp av grundbegreppen. De logiska resonemang man gör när man härleder egenskaper på detta sätt kallas en *stringent bevisföring*. Alltsammans bildar en *matematisk teori*.
>
> En tanke bakom denna uppläggning är att man får teorier som uppfyller högt ställda krav på objektivitet - slutsatserna som dras blir desamma vem som än gör dem, förutsatt att man inte gör något logiskt fel.

En matematisk teori består alltså av definitioner och två olika typer av påståenden, axiom (= postulat) och teorem (= satser). Axiomen utgör fundamentet för teorin och teoremen kan bevisas logiskt med utgångspunkt från dessa. En begreppsmodell som uttrycker strukturen hos en matematisk teori i enlighet med denna beskrivning visas i Fig. (26).

---

1. Petermann: *Analytiska Metoder* I, sid 64.
2. Alldeles nödvändigt är detta dock inte - i prinicip kan man välja begrepp som inte knyts till någonting speciellt och välja postulat efter gottfinnande. Det blir då närmast fråga om ett abstrakt spel efter vissa regler (postulaten) vars eventuella "användbarhet" till annat är av underordnad betydelse.

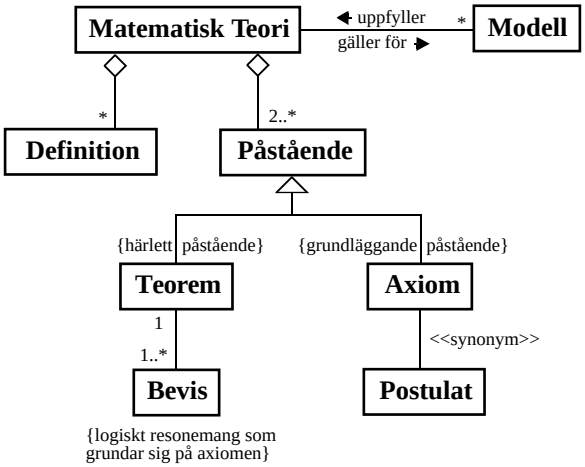

Fig. 26. Begreppsmodell för begreppet Matematisk Teori.

Låt oss t.ex. betrakta en abstrakt framställning av den geometriska teorin om punkter och linjer i planet. Vad som egentligen menas med en punkt eller en linje definieras inte. Man har naturligtvis en intuitiv föreställning om detta, men man är noga med att inte låta denna ingå i definitionen. Vad man istället gör är att ställa upp ett antal spelregler för hur punkter och linjer uppför sig i förhållande till varandra. Några av dessa axiom är t.ex.

(1): Det existerar 3 punkter som inte ligger på samma linje.
(2): Två punkter ligger på exakt en linje.
(3): Två linjer ligger på exakt en punkt[1].

Med utgångspunkt från dessa grundläggande påståenden försöker man sedan resonera sig fram till nya påståenden (teorem) som är logiska konsekvenser av dessa utgångspunkter. Resonemangen genomförs med hjälp av s.k. logiska slutledningsregler, som beskriver på vilka sätt det är tillåtet att resonera. Sådana regler formulerades första gången av Aristoteles på 300-talet före Kristus, och hans modell för tillåtna resonemang brukar kallas aristotelisk logik. Två exempel på resonemangsregler ur den aristoteliska logiken är:

R1: Varje påstående *P* är antingen sant eller falskt[2].

R2: Om vi vet att (*P* sant) och om vi vet att (*P* sant medför *Q* sant) så är även (*Q* sant)[3].

1. Den geometriska teori där detta axiom ingår kallas *projektiv geometri.* Inom den projektiva geometrin skär alla linjer varandra och två s.k. parallella linjer säges skära varandra i en s.k. *oändlighetspunkt*, vilket är en typ av punkt som saknar motsvarighet inom den klassiska (s.k. euklidiska) geometrin. Den projektiva geometrin började utvecklats på 1400-talet och hade sitt ursprung i studiet av hur man skapar ett korrekt perspektiv i en målning.
2. Denna resonemangsregel brukar kallas "*lagen om det uteslutna tredje*". Med detta menar man att det inte finns något tredje alternativ. Med utgångspunkt från denna regel blir det tillåtet att genomföra s.k. *motsägelsebevis.* Om man vill visa att påståendet *A* är sant, så kan man anta att *A* är falskt och försöka resonera sig fram till en motsägelse. Lyckas man med detta kan man dra slutsatsen att eftersom *A* uppenbarligen inte kan vara falskt, så måste *A* vara sant. Det är värt att påpeka att lagen om det utslutna tredje inte accepteras som giltig av alla matematiker, t.ex.inte av dem som sysslar med s.k. *konstruktiv matematik.*
3. Denna resonemangsregel benämnes *modus ponens*, vilket är dess latinska namn.

Påståendet (*P* sant medför *Q* sant) utläses även (om *P* så *Q*) eller (*P* implicerar *Q*). Detta påstående brukar i matematiken betecknas med hjälp av en s.k. *implikationspil*: (*P* *Q*).

2.2 Ett enkelt formellt matematiskt system

Det geometriska axiomsystemet ovan måste kompletteras med flera andra axiom innan det fullständigt kan uttrycka grunderna i den projektiva geometrin. För att förstå hur ett axiomatiskt system fungerar ska vi ta en titt på ett minimalistiskt formellt system som endast innehåller två symboler, ett axiom och fyra logiska slutledningsregler.

Odefinierade symboler: *A*, *B*.

Axiom: *A*.

Odefinierad regelsymbol: (utläses: "får omvandlas till").

Definition: Ett *ord* är en ändlig kombination av *A*:n och *B*:n, t.ex. *ABABAABBAA*.

Logik: För varje par av ord *X*, *Y* gäller följande omvandlingsregler
(= regler för tillåtna resonemang = bevisregler):

R1: *XA* *XAB*

R2: *X* *XX*

R3: *XAAAY* *XBY*

R4: *XBBY* *XY*

Vi har nu allt som behövs för att bedriva matematik inom detta formella matematiska system. Det kan t.ex. vara instruktivt att fundera lite på följande

Övning: Undersök om *B* är ett teorem i den matematiska teori som utgår från *A*, dvs undersök om *B* kan härledas ur *A* med hjälp av de logiska slutledningsreglerna. (Ledning: undersök hur antalet *A*:n ändras när reglerna tillämpas.)

2.3 Matematiken som ett formellt spel i en hypotetisk värld

Det formella system som presenterades ovan innehåller essensen i all matematisk teoribildning. Matematiken kan med detta synsätt betraktas som ett *formellt spel*. Det matematiska spelet har en uppsättning bevisregler (= logik = grammatik) som anger tillåtna drag och ett antal startsentenser (= axiom) som anger utgångspunkterna. Spelet går sedan ut på att konstruera så många tillåtna sentenser (= teorem) som möjligt. Härvidlag låter man sig framförallt styras av en inre matematisk estetik, vilken innebär starka krav på renodling av struktur och naturliga kopplingar mellan olika begrepp. Detta har visat sig leda till matematiska teorier som varit förvånansvärt

effektiva när de tillämpats på olika problem med anknytning till den fysikaliska verligheten.[1]

Det är viktigt att betona att matematiken inte är någon vetenskap. Vetenskapen försöker beskriva verkligheten så som vi upplever den, medan matematiken beskriver en hypotetisk värld där alla påståenden är betingade (= inleds med "om"). Kom ihåg att ($P \quad Q$) betyder "*om* P är sant *så* är även Q sant". Detta säger ingenting om sanningshalten hos vare sig P eller Q utan ger endast en upplysning om *relationen* mellan de förhållanden under vilka de är sanna. Det är därför som Bertrand Russel beskrev matematiken som "den disciplin där vi inte vet vad vi talar om och inte heller huruvida det vi säger är sant eller inte". Matematiska påståenden är alltid betingade av de axiomatiska utgångspunkterna. Med en språklig analogi skulle man kunna säga att matematiska påståenden alltid formuleras i *konjunktiv* form (Om P *vore* sant så *bleve* Q sant) medan vetenskapliga påståenden uttrycks i *presens* (P *är* sant).

2.4 Att tillämpa en matematisk teori

Orsaken till att man vill befria sig från intuitionen när man beskriver de matematiska begreppen är att man vill försöka frilägga exakt hur begreppen logiskt beror av varandra. I våra intuitiva förställningar finns oftast ett antal dolda begrepp och sammanhang, och om man skulle ta med dem i definitionerna så skulle de logiskt relevanta strukturerna bli mindre tydliga. Man strävar alltså efter att synliggöra exakt vad det är man stödjer sig på när man genomför ett matematiskt resonemang. Poängen med detta är att de matematiska resonemangen därigenom blir giltiga och tillämpbara i ett stort antal olika konkreta situationer.

Att använda abstrakt matematik på ett konkret problem brukar kallas att *tillämpa* matematiken. Detta innebär att man till sitt problem associerar en matematisk teori samt en modell som uppfyller axiomen i densamma. När man gjort detta kan man omedelbart sluta sig till att modellen även uppfyller teoremen i den matematiska teorin. Om teorin har ett stort antal teorem, så vet man alltså en massa saker om modellen genom att man redan har bevisat dem "en gång för alla" enbart med utgångspunkt från axiomen.

2.5 Hur matematiken tillämpas på vetenskapen

Ända sedan grekerna införde den axiomatiska metodiken har matematiken kämpat en lång och hård historisk kamp för att befria sig från all underliggande mening (= innehåll = semantik). Den har därvid utvecklats till ett formellt spel av den sort som beskrivits ovan, vilket kännetecknas av en extremt renodlad struktur (= syntax) som i sig själv är meningslös[2].

En matematisk teori får mening först när den tillämpas på något problemområde. Varje modell som uppfyller de matematiska axiomen fyller den matematiska teorin med mening. Det är därför vi har ett så stort behov av konkreta exempel på tillämpningar av matematik. Utan dessa upplevs den formella matematiken med rätta som meningslös. Den frustration som man ofta

1. "On the Unreasonable Effectiveness of Mathematics in the Natural Sciences" är en berömd artikel på detta tema skriven av fysikern Eugene Wigner [(22)].
2. Ett berömt försvarstal för den s.k. "rena" matematiken finns återgivet i [(9)].

kan känna över detta faktum beror på att man förväntar sig en mening där denna omsorgsfullt har abstraherats bort.

Figur (27), illustrerar hur matematiken tillämpas inom vetenskapen. Som beskrivits ovan består matematiken av en mängd villkorliga (= hypotetiska) påståenden. Vetenskapen använder matematikens logiska resonemangsförmåga (*A* *B*) för att transformera ett *antagande* (*A* är sant) till en *logisk slutsats* (då måste även *B* vara sant).

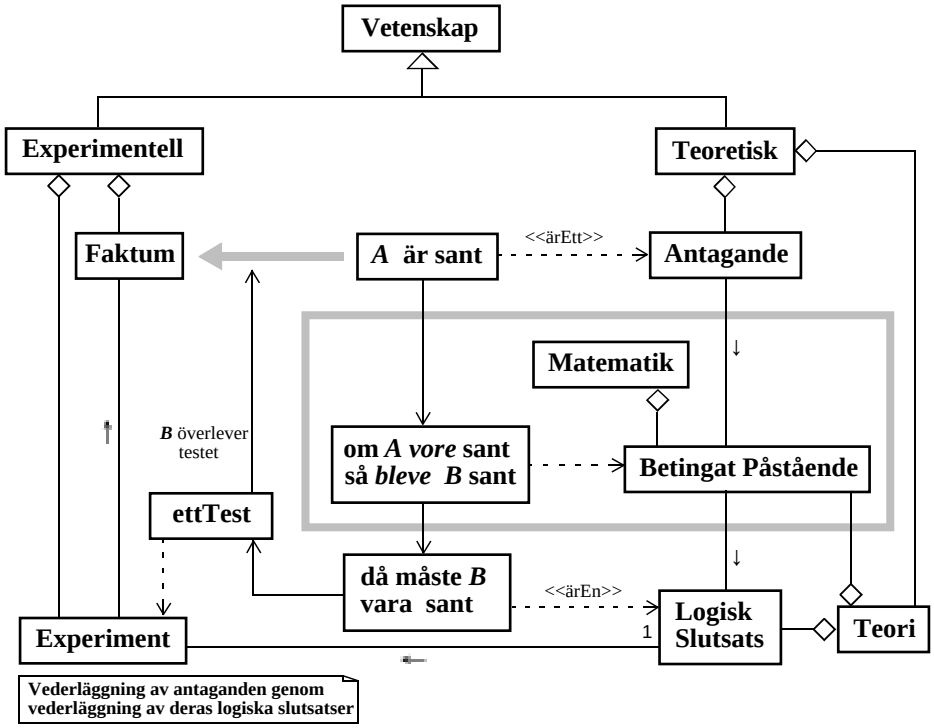

Fig. 27. Samspelet mellan matematiken och vetenskapen.

Varför vill då vetenskapen omvandla antaganden till logiska slutsatser? Orsaken till detta är att vetenskapen vill förvandla *teoretiska antaganden* till *experimentella fakta*, vilket den försöker göra genom att utsätta de *logiska slutsatser* som följer ur antagandena för *falsifikationsförsök* genom *experiment*[1]. Poängen är att de logiska slutsatserna (om de ska vara användbara) måste vara mycket lättare att testa experimentellt än de ursprungliga antaganden från vilka de har härletts med hjälp av logiska (= matematiska) metoder. På detta sätt sluter vetenskapen cirkeln och de antaganden, vars logiska slutsatser överlever de experimentella falsifikationsförsöken transformeras (gradvis) till vetenskapliga fakta.

Den tjocka grå pilen i diagrammet representerar ett egendefinierat UML-element som vi kan kalla en *trendpil.* Den uttrycker en *tendens* - i detta fall tendensen att betrakta antagandet *A* som ett vetenskapligt faktum. Det är genom att överleva många olika falsifikationsförsök - av många olika logiska slutsatser dragna från *A* - som antagandet *A* till slut uppnår status som ett vetenskapligt faktum.

---

1. Eftersom vi bara kan mäta skillnader (och aldrig likheter) så kan vi aldrig experimentellt bevisa en fysikalisk “naturlag” (som formulerats som en matematisk likhet). Vi kan enbart förstärka hypotesen om dess giltighet genom att vi inte lyckades mäta upp några signifikanta skillnader från de teoretiska värden som lagen förskriver.

Fig. (28) visar en begreppsmodell som försöker beskriva hur naturlagar uppstår. Vi bildar begrepp i syfte att ge struktur åt de fenomen vi upplever. Våra begrepp leder till en teori, vilken enligt tidigare består av utsagor klassificerade i två grupper - axiom och teorem. Teorin leder i sin tur till hypoteser (= förväntningar) vilka kan testas genom experiment.

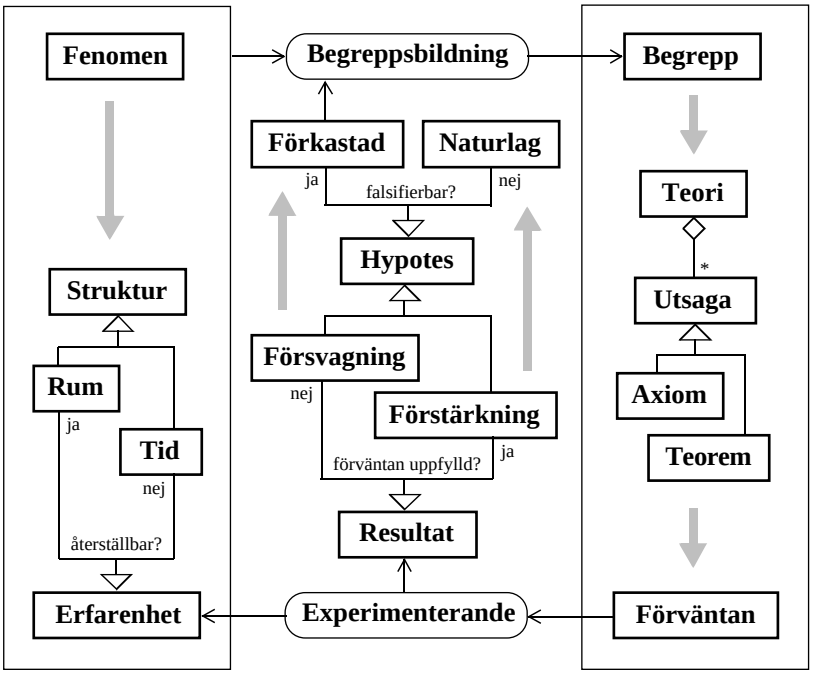

Fig. 28. Begreppsmodell över hur naturlagar uppstår.

Vid genomförandet av ett experiment kan två olika saker inträffa. Antingen uppfylls förväntningarna vilket leder till en förstärkning av den underliggande hypotes som skulle testas, eller också uppfylls inte förväntningarna, vilket leder till en försvagning av motsvarande hypotes. Om hypotesen överlever tillräckligt många experiment så kanoniseras den till sist och uppnår status av naturlag. Om den däremot vederläggs av tillräckligt många (trovärdiga) experiment så måste den förkastas tillsammans med den teori som den byggde på. Detta leder till ny begreppsbildning, omformulering av teorin, nya förväntningar och nya experiment, etc.

De erfarenheter som experimenterandet ger oss - kombinerade med de teorier som vi skapar med hjälp av våra begrepp - leder fram till en struktur på fenomenen som gör våra upplevelser av världen begripliga för oss. I diagrammet i Fig. (28) är denna struktur indelad i två olika typer: **Rum(s)Struktur** respektive **Tid(s)Struktur**. Klassificeringen av ett fenomen med avseende på rum eller tid har att göra med dess grad av återställbarhet. Om det (i princip) är möjligt att återställa effekten av upplevelsen (= "flytta tillbaka det som förändrades") så förlägger vi fenomenet i rummet, dvs vi gör det till ett rumsligt fenomen. Om det däremot inte är möjligt att återställa fenoments effekter så förlägger vi det i tiden.[1] Med modern IT-terminologi skulle vi kunna säga att rumsfenomenen har *undo* medan tidsfenomen saknar detta.

1. Detta sätt att beskriva skillnaderna mellan rum och tid går tillbaka till den franske matematikern och filosofen Henri Poincaré (1848-1912) och finns beskrivet i [(15)].

3 Matematisk begreppsmodellering

3.1 Introduktion

Matematiken är ett exempel på ett hårt kalibrerat begreppsområde. Den största delen av en modern matematisk text utgörs av definitioner, vilket vittnar om att matematikerna lägger ner stor möda på att beskriva begreppens egenskaper och deras förhållande till varandra på ett så preciserat sätt som möjligt.

3.2 Fördelar med en begreppskarta

Som vi berörde i Kapitel (1.1) så har en begreppskarta flera fördelar i jämförelse med en verbal presentation av motsvarande begreppsrelationer. Två av dessa förtjänar att särskilt framhållas i detta sammanhang

För det första: En begreppskarta är att den bryter upp ordningen i den verbala framställningen av begreppsrelationerna. Den visar alla relationer på samma gång, till skillnad från en verbal framställning, som är tvungen att välja att beskriva dem i en viss ordning. En verbal framställning av en relation mellan två begrepp kan betraktas som en resa på kartan - en navigerad väg från det ena begreppet till det andra. En bakomliggande orsak till fördelen med en begreppskarta i förhållande till en verbal presentation är det faktum att vår förmåga att visuellt överblicka en begreppsrelation från olika håll är betydligt större än vår förmåga att verbalt byta riktning på relationen ifråga. Vi kan alltså lättare “visuellt integrera” informationen och med kartans hjälp skapa överblick och ge oss en helhetsbild av relationerna mellan begreppen. Det är ju precis därför som vi kallar det “överblick” och “helhetsbild”.

För det andra: En begreppskarta skapar ett övergripande logiskt sammanhang mellan begreppen. Både begreppen (= noderna) och länkarna (= bågarna) på kartan kan sedan förses med olika typer av innehåll, varefter kartan kan navigeras och innehållet presenteras ur ett antal olika (konfigurerbara) aspekter (= filter). Genom att begreppsinnehållet presenteras separat från begreppsrelationerna blir det möjligt att ta del av innehållet utan att förlora överblicken över relationerna, dvs sammanhanget[1].

3.3 Ett enkelt exempel

Vi har redan sett exempel på matematisk begreppsmodellering, t.ex. begreppsmodellen av en matematisk teori i Fig. (26). Matematiken kan även delas in i olika ämnesområden, svarande mot olika typer av (= sorters) matematik. Fig. (29) visar fyra olika matematiska områden: **Geometri**, **Algebra**, **Analys** och **Kombinatorik**. De tre prickarna anger att uppdelningen inte är fullständig, dvs att det finns många andra typer av matematik som inte visas i diagrammet. Naturligtvis finns det heller inga “vattentäta skott” mellan de olika områdena, utan de överlap-

1. En prototyp till ett överblicksverktyg för navigation av begreppssammanhang och presentation av begreppsinnehåll som bygger på dessa principer är för närvarande under utveckling på CID. Den går under namnet *Conzilla* och finns närmare beskriven på http://cid.nada.kth.se/il. De underliggande principerna för begreppsnavigation finns beskrivna i [(13)].

par med varandra på samma sätt som dag och natt överlappar i skymning. Ändå är det i många sammanhang poängfullt att genomföra en uppdelning på detta sätt. Fig. (29) visar även en uppdelning av geometrin i delområdena (= subtyperna) **Algebraisk Geometri**, **Differential Geometri** och **Projektiv Geometri**.

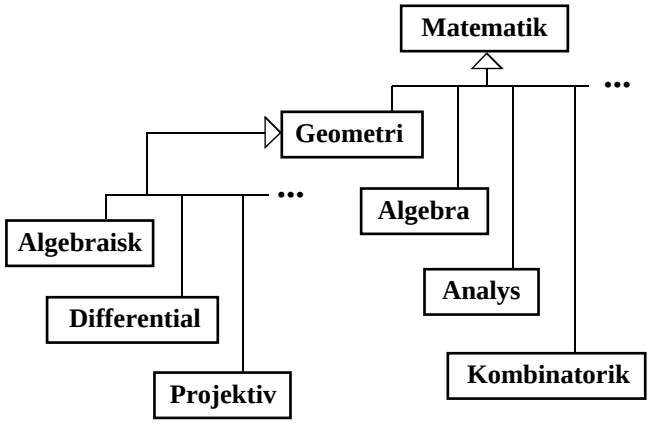

Fig. 29. Matematik och Geometri indelade i några olika ämnesområden.

3.4 Lingvistiskt baserad matematisk begreppsbildning

I sitt berömda Erlangerprogram från 1872 framlade den store geometrikern Felix Klein idén att betrakta en geometri som en samling påståenden om objekt som är invarianta (= förblir oförändrade) under en grupp av transformationer (se t.ex. [(12)]). Detta markerar början på det moderna betraktelsesätt som betraktar varje geometri som en sorts språk, med sin egen samling av verb (= transformationer) och substantiv (= invarianter). Vårt språk är intimt relaterat till våra metoder för begreppsbildning. Verben beskriver operationerna (förändringarna) som vi kan observera – eller föreställa oss – medan substantiven beskriver invarianterna, dvs de ”substanser” som överlever verbens operationer (transformationer). Adjektiven beskriver värdet av aspekter (egenskaper) hos substantiven – som t.ex. i satsen: ”den röda bilen stannade”, där adjektivet ”röd” representerar värdet av aspekten ”färg” hos just den omtalade instansen av substantivet ”bil”.

I det moderna matematiska synsättet uppför sig alltså ett geometriskt system som ett språk. Det har sina verb, som uttrycker de tillåtna transformationerna (= rörelserna), och sina substantiv, som uttrycker dess invarianter, dvs de begrepp som överlever verbens rörelse-transformationer. Så utgör t.ex. begreppet ”kvadrat” ett substantiv i den vanliga *euklidiska geometrin*, eftersom transformationerna i denna geometri består av de vanliga (stela) rörelserna samt likformig förstoring och förminskning, och dessa förändringar förvandlar en kvadrat till en ny kvadrat – ”lämnar kvadrat-egenskapen invariant” som en matematiker skulle uttrycka saken. Begreppet kvadrat är således ett euklidiskt begrepp, eftersom det överlever att transformeras av de euklidiska verben.

I de geometriska skuggornas värld däremot – s.k. *projektiv geometri* – är inte begreppet kvadrat väldefinierat. I projektiv geometri är alla skuggbilder av en plan figur ekvivalenta, eftersom de överförs i varandra via projektion från en punkt mot ett plan – en tillåten rörelse-transformation i projektiv geometri[1]. Eftersom en kvadrat kan projiceras till en ”sned” fyrhörning, så överlever inte kvadraten de projektiva rörelserna, och är därför - i Kleins mening – inget projektivt substantiv. Det relevanta (= invarianta) begreppet är i stället “fyrhörning”.

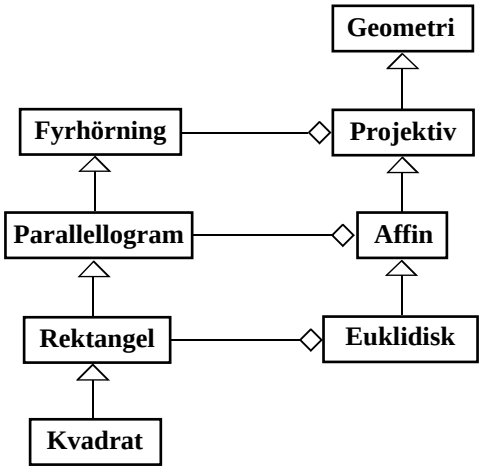

Fig. 30. Invarianta begrepp (= substantiv) i några olika typer av geometri.

Den s.k. *affina geometrin* är en specialisering av den projektiva geometrin (och en generalisering av den euklidiska geometrin) som karaktäriseras av att den bevarar parallellitet (se [(12)]). De affina verben avbildar alltså parallella linjer på parallella linjer, medan vinklar i allmänhet inte bevaras. Begreppen “kvadrat” och “rektangel” saknar därför mening i den affina geometrin, där det relevanta begreppet i stället blir “parallellogram”, eftersom varje parallellogram förblir en parallellogram när den undergår en affin rörelsetransformation. Dessa förhållanden illustreras i Figur (30).

3.5 Begreppet matematisk komposition

En *matematisk komposition* (= sammansättning) innebär ett sätt att sätta ihop saker genom att kombinera dem. Exempel på kompositioner är *addition*, *subtraktion*, *multiplikation* och *division*, vilka samtliga sätter ihop två tal och gör ett nytt tal av dem. Ett sätt att betrakta en matematisk komposition är som en sorts maskin där vi matar in ett antal matematiska instanser (input) som sedan omvandlas (= transformeras) på något sätt och matas ut ur maskinen[2] i form av output. De saker som stoppas in i kompositionsmaskinen kallas *operander* och det som kommer ut ur den kallas *resultat.* När man talar om en matematisk komposition brukar man dessutom kräva att både operanderna och resultatet ska vara av samma matematiska typ (t.ex. heltal eller rationella tal).

1. En översiktlig beskrivning av projektiv geometri finns i [(2)]. En fördjupad framställning av ämnet ges t.ex. i [(3)].
2. En sådan typ av maskin kallas i matematiken allmänt för en *funktion*.

En viktig egenskap hos en komposition är dess s.k. *aritet*, vilket står för antalet operander i kompositionen. En komposition med aritet 1 (dvs med en operand) kallas en *unär* komposition, medan en komposition med aritet 2 kallas en *binär* komposition. Kompositioner med högre aritet än 1 eller 2 är ovanliga i matematiken.

De fyra räknesätten, addition, subtraktion, multiplikation och division är exempel på binära kompositioner. Att ta minus av ett tal är däremot ett exempel på en unär komposition. Minus av ett tal *a* brukar kallas för talets *additiva invers* (betecknad -*a*) och står för det tal som adderat till det ursprungliga talet ger talet 0, som är det additiva enhetstalet, dvs $a + (-a) = 0$. Operationen att "ta minus" är alltså synonym med *additiv invertering.* Ett annat exempel på en unär komposition är *multiplikativ invertering*. Den multiplikativa inversen till ett tal *a* brukar kallas "ett-genom-*a*" (betecknad 1/*a*) och står för det tal som multiplicerat med *a* ger talet 1, som är den multiplikativa enheten, dvs $a * (1/a) = 1$.

En välkänd svårighet i skolmatematiken är att skilja mellan *subtraktion* och *additiv invertering.* Anledningen till den begreppsförvirring som råder på detta område är att symbolen minus (-) används i två helt skilda betydelser, nämligen dels för att beteckna den unära kompositionen additiv invertering (-*a*), dels för att beteckna den binära kompositionen subtraktion (*a*-*b*). Additiv invertering och subtraktion är alltså homonyma begrepp eftersom de betecknas av samma symbol. [se Kapitel (1.4)]. En begreppskarta över olika typer av matematiska kompositioner som sammantattar ovanstående diskussion visas i Fig. (31).

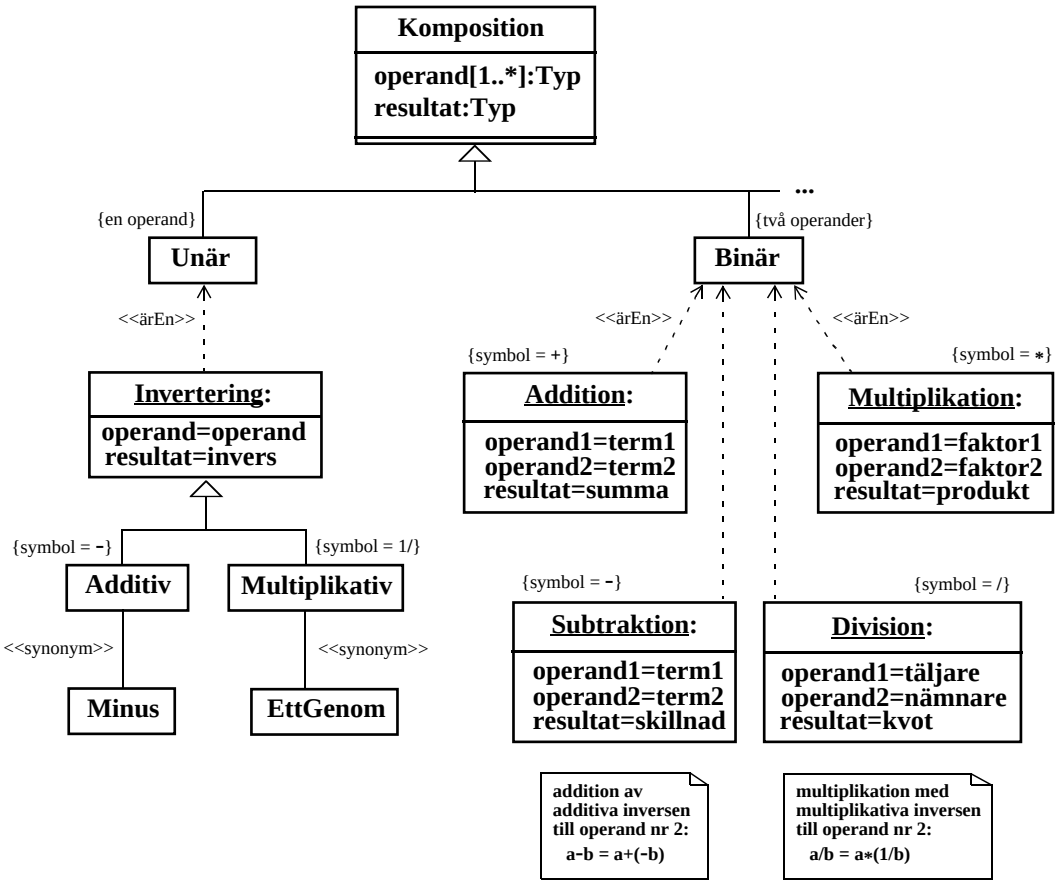

Fig. 31. Begreppsmodell av några vanliga typer av matematiska kompositioner.

Notera: I Fig. (31) har vi använt namngivna värden för att uttrycka symbolerna för de olika aritmetiska räknesätten. Vi hade lika gärna kunnat använda oss av stereotyper på det sätt som visas i Fig. (32).

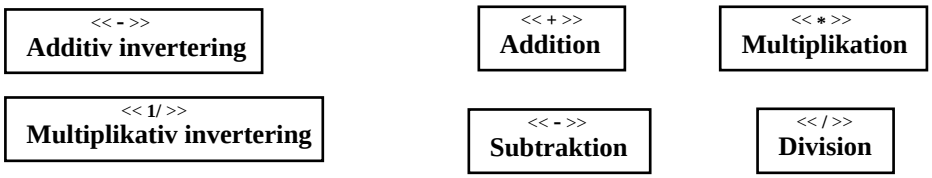

Fig. 32. Aritmetiska symboler som stereotyper.

Det är värt att notera ett par saker i anslutning till Fig. (31). I begreppslådan för **Komposition** framgår att varje komposition har en eller flera operander av samma typ, samt att operandernas typ måste överensstämma med resultatets typ. Vidare är såväl invertering som addition, subtraktion, multiplikation och division beskrivna som *instanser* av typ unär- respektive binärkomposition. Värdet på attributet **operand** är **term** för instanserna **Addition**: och **Subtraktion**: , **faktor** för instansen **Multiplikation:**. och **täljare** respektive **nämnare** för instansen **Division:**.Vad gäller instansen **Invertering:** så har attributet **operand** i allmänhet inget speciellt namn (= värde). Operanden i en inverteringsoperation kallas fortfarande för "operand" medan resultatet av en inverteringsoperation kallas **invers**. Attributet operand kan alltså i detta fall sägas ha sig själv som värde, vilket förklarar utseendet av begreppslådan för **Invertering:**. Detta reflekterar helt enkelt språkbruket inom området, vilket ju är avsikten med en begreppsmodell.

3.6 Metodiska riktlinjer vid matematisk begreppsmodellering

Att begreppsmodellera ett problemområde kan göras på många olika sätt med betoning på olika aspekter. Modelleringen kan dessutom utföras med olika upplösning (= finkornighet). Poängen med en begreppsmodell är ju att fokusera på vissa strukturer inom området och utelämna andra. Det är därför svårt att ange en generell metodik som passar för alla typer av matematisk begreppsmodellering. De metodiska riktlinjer som anges nedan kan dock vara bra att ha som utgångspunkt.

• Upprätta ett *terminologiskt lexikon* över de matematiska begrepp du vill modellera. Beskriv begreppens definition (= intention) så kortfattat som möjligt (ett par meningar) och ange *exempel* (instanser) på varje begrepp.

• För varje matematiskt begrepp *B* som du har noterat, skriv ner olika *egenskaper* (= *attribut*) hos begreppet *B*. Gör en kort beskrivning av vad varje egenskap innebär. Skilj noga mellan *instansattribut* (vars värden varierar med olika instanser av begreppet *B*) och *klassattribut* (vars värden är gemensamma för alla instanser av begreppet *B*). Räkneregler och satser för begreppet *B* är exempel på klassattribut för *B* eftersom de gäller för alla instanser av *B*.

• Finns det några speciella beteckningar (= matematisk notation) för begreppet *B*? Notera i så fall dessa.

• Anteckna eventuella synonymer (= andra namn) för begreppet *B*.

• Skriv ner de operationer som kan utföras på instanser av begreppet *B*. Tänk på att många operationer omfattar flera begrepp. Sådana operationer hör därför inte naturligt ihop med något enstaka av dessa begrepp.

• Består begreppet *B* av några delar som själva är matematiska begrepp? Beskriv i sådana fall delarnas egenskaper och vilka operationer som kan utföras på dessa.

• Kan begreppet *B* generaliseras eller specialiseras till något annat matematiskt begrepp?

• Är begreppet *B* associerat med något annat matematiskt begrepp? I så fall, har associationen något naturligt namn? Finns det några naturliga rollnamn för de olika sidorna av associationen? Hur många instanser på den andra sidan kan en instans på den ena sidan vara länkad till? Uttryck eventuella begränsningar genom att ange multipliciteter.

• Skriv ner olika användningsområden för begreppet *B*, dvs olika sammanhang i vilka begreppet *B* kan förekomma, såväl inom andra områden av matematiken (än det som du för närvarande modellerar) som inom tillämpningar av olika slag. Uttryck kopplingen mellan *B* och varje användningsområde som en association, och ge associationen ett beskrivande namn som t.ex <<förekommer i>> eller <<används inom>>.

Det är viktigt att betona att den matematiska begreppsmodelleringen inte syftar till att framställa programkod, utan till att fokusera på begreppen och deras relationer. Arbeta därför gärna med t.ex. kommentarsnoter om du känner att du vill uttrycka någon struktur som inte riktigt får plats bland de UML begrepp som vi har infört. Du får naturligtvis även använda UML begrepp som inte har införts här - det är dock inte meningen att du ska behöva dem i dina modelleringsuppgifter.

I den här skriften har vi huvudsakligen beskrivit den statiska delen av UML. Dessa element lämpar sig bäst för att modellera begreppens attribut och relationer. Det är därför meningen att du ska koncentrera dig på dessa aspekter i ditt matematiska modelleringsarbete.

3.7 Standardbeteckningar för matematiska talmängder

N = Naturliga talen = {0, 1, 2, 3, ...}.

Z = Hela talen = { .... -2, -1, 0, 1, 2, ...}.

Q = Rationella talen.

R = Reella talen.

C = Komplexa talen.

3.8 Exempel - begreppet reellt polynom i en variabel

Konvention: När vi nedan talar om begreppet **Polynom** menar vi hela tiden begreppet **Reellt polynom i en variabel**.

3.8.1 Terminologiskt lexikon

• *polynom*: summa av ett ändligt antal termer.
  • *exempel* på (= instans av) polynom: $3x^2-4x+7$

• *term*: produkt av en koefficient och ett antal kopior (= förekomster) av en variabel.
  • *exempel*: termerna i polynomet $3x^2-4x+7$ är $3x^2$, $-4x$ och 7.

• *koefficient*: ett reellt tal, dvs en medlem i mängden **R**.
  • *exempel*: koefficienterna i polynomet $3x^2-4x+7$ är 3, -4 och 7.

• *variabel*: en symbol (= namn) som kan representerta (= anta) olika reella talvärden. En symbol består oftast av en bokstav. Man kan tänka på en variabel som en låda med ett namn och eventuellt ett värde (= innehållet i lådan). Att byta värde på variabeln motsvaras då av att ändra innehållet i lådan.
  • *exempel*: polynomet $3x^2-4x+7$ har en variabel med namnet x.

• *värde*: ett element i mängden **R** av reella tal.

• *formell variabel*: symbol som inte har något värde.

• *polynomgrad*: ett polynom har en grad som anger största graden hos någon av dess termer.
  • *exempel*: polynomet $3x^2-4x+7$ har grad 2, eftersom termerna har grad 2, 1, resp. 0.

• *termgrad*: varje term har en grad som anger antalet förekomster av variabeln i termen. Graden av en term är alltså lika med antalet faktorer (= kopior) av variabeln som ingår i termen.
  • *exempel*: termen $4xxx = 4x^3$ (i variabeln x) har graden 3.

• *addition* (av två polynom): summan av två polynom är ett nytt polynom som bildas genom att termer av samma grad i de båda polynomen adderas.

• *addition* (av två termer): suman av två termer av samma grad är en ny term av samma grad vars koefficient är summan av de bägge termernas koefficienter.

• *multiplikation* (av två polynom): produkten av två polynom bildas genom att varje term i det ena polynomet multipliceras med varje term i det andra polynomet och de resulterande termerna adderas.

• *multiplikation* (av två termer): produkten av två termer är en ny term vars grad är summan av graderna hos de båda termerna och vars koefficient är produkten av koefficienterna hos de båda termerna.

3.8.2 Exempel på användningsområden för polynom (inom matematiken)

Polynom används ofta för att bilda *funktioner.* Själva polynomet kallas i detta sammanhang ofta *formellt polynom* och den funktion man bildar med polynomets hjälp kallas en *polynomfunktion*. Det begrepp som beskrivits ovan under namnet polynom är i själva verket liktydigt med formellt polynom.

En *polynomfunktion* p:**R** → **R** använder ett formellt polynom $p(x)$ för att avbilda varje reellt tal a på ett annat reellt tal via utbyte av polynomet p:s formella variabel x mot talet a och uträkning av motsvarande värde, som vi kallar b. Detta brukar kallas insättning av a i polynomet p och sambandet mellan a och b uttryckes $b = p(a)$. Oftast säger man bara att variabeln x *varierar* över talmängden **R** och kallar resultatet av insättningen av x-värdet i polynomet p för y, dvs man skriver $y = p(x)$ för att beteckna sambandet mellan ett godtyckligt x-värde och motsvarande uträknade polynomvärde.

3.8.3 Begreppsmodell av reella polynom i en variabel

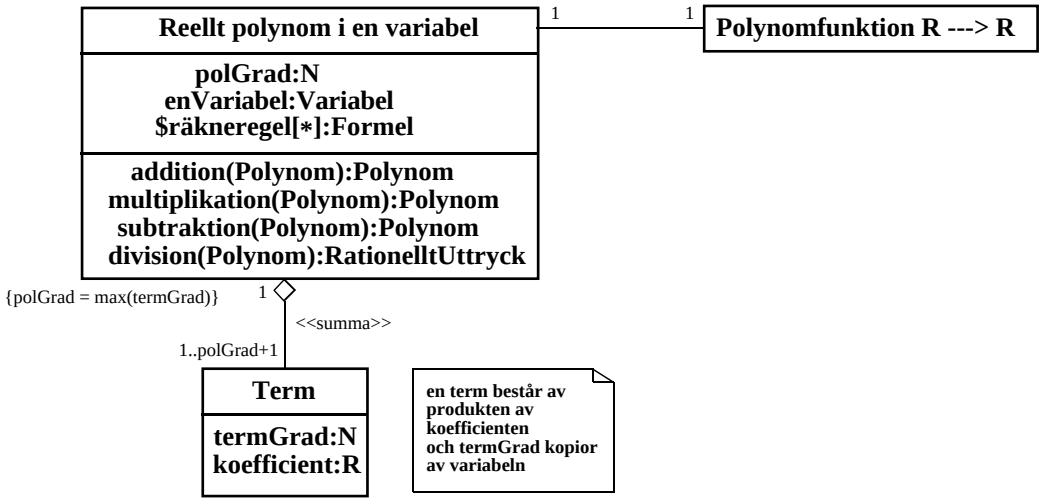

Fig. 33. Begreppet reellt polynom i en variabel.

3.9 Några exempel på blandade begreppsmodeller

Vi ska avsluta med att presentera några begreppsmodeller som är uppbyggda genom att blanda UML med en allmän mind-mapping teknik. Avsikten är att visa att man kan bygga begreppsbeskrivningar på många olika sätt för att passa olika syften. UML utgör en språklig utgångspunkt, men även en begränsning som man är fri att överskrida om man vill.

3.9.1 Konsten att skapa svårigheter i matematikundervisningen

Ett problem i den tidiga matematikundervisningen är de starka förväntningarna att matematiken kommer att bli svår när man kommer upp på högstadiet. Denna attityd är mycket smittsam. vilket blir speciellt tydligt när man ska börjar räkna med symboler - eller “räkna med *x*” som man brukar säga. Den konversation som återges i Fig. (34) är nog tyvärr ganska typisk i detta sammanhang.

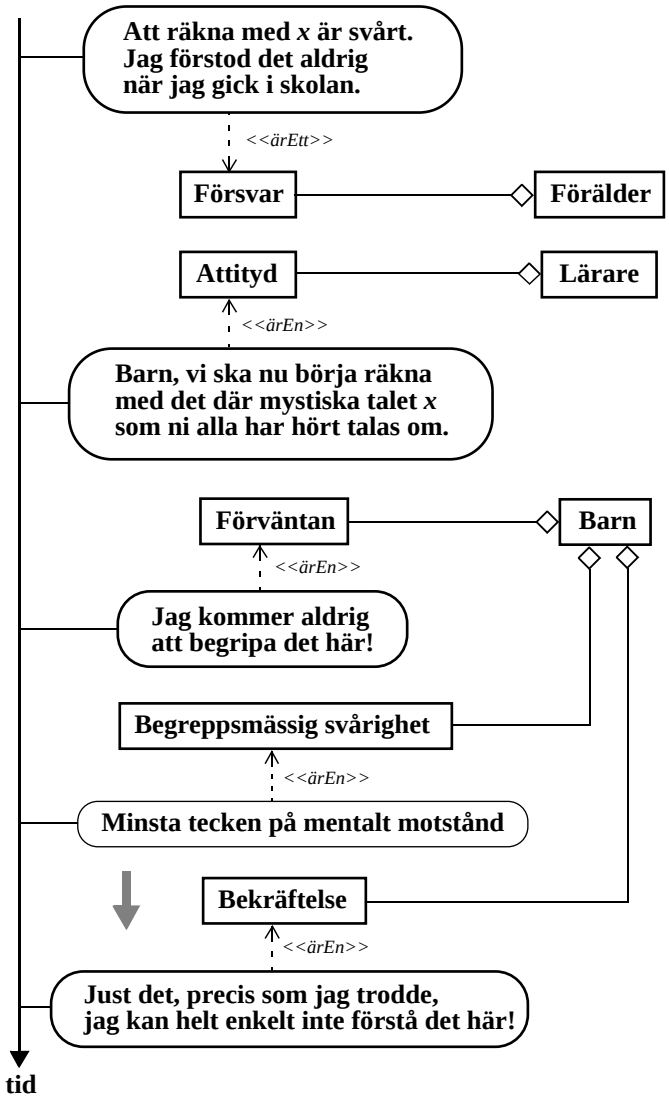

Fig. 34. *x*-istentiella förväntningar inom skolmatematiken[1].

1. I figuren är instanser av begrepp återgivna i rutor med rundade hörn. Denna notation är ofta mycket praktisk och används bl.a. i OMT (Object Modeling Technique [(17)]) som är den mest spridda föregångaren till UML.

3.9.2 En variabel som en låda med ett namn och (eventuellt) ett innehåll

När min dotter Ylva gick i låg-och mellanstadiet hade jag möjlighet att genomföra ett antal olika matematikpedagogiska experiment i hennes klass. Experimenten - som gick under samlingsnamnet *Förstklassig Matematik* - innefattade bl.a. att lära barnen att räkna med *x* när de gick i 4:e klass. Ett ord som jag konsekvent undvek i detta sammanhang var "variabel". Jag använde i stället analogin mellan en *variabel* och en *låda* som utgångspunkt. Båda har ett namn och eventuellt även ett innehåll, ifall vi har givit variabeln ett värde - dvs om vi har lagt något i lådan. Med denna metafor kan man betrakta ekvationer som samband mellan innehållet i ett antal olika lådor. Att lösa ekvationerna blir då liktydigt med att lista ut vad som finns i (vissa av) lådorna.

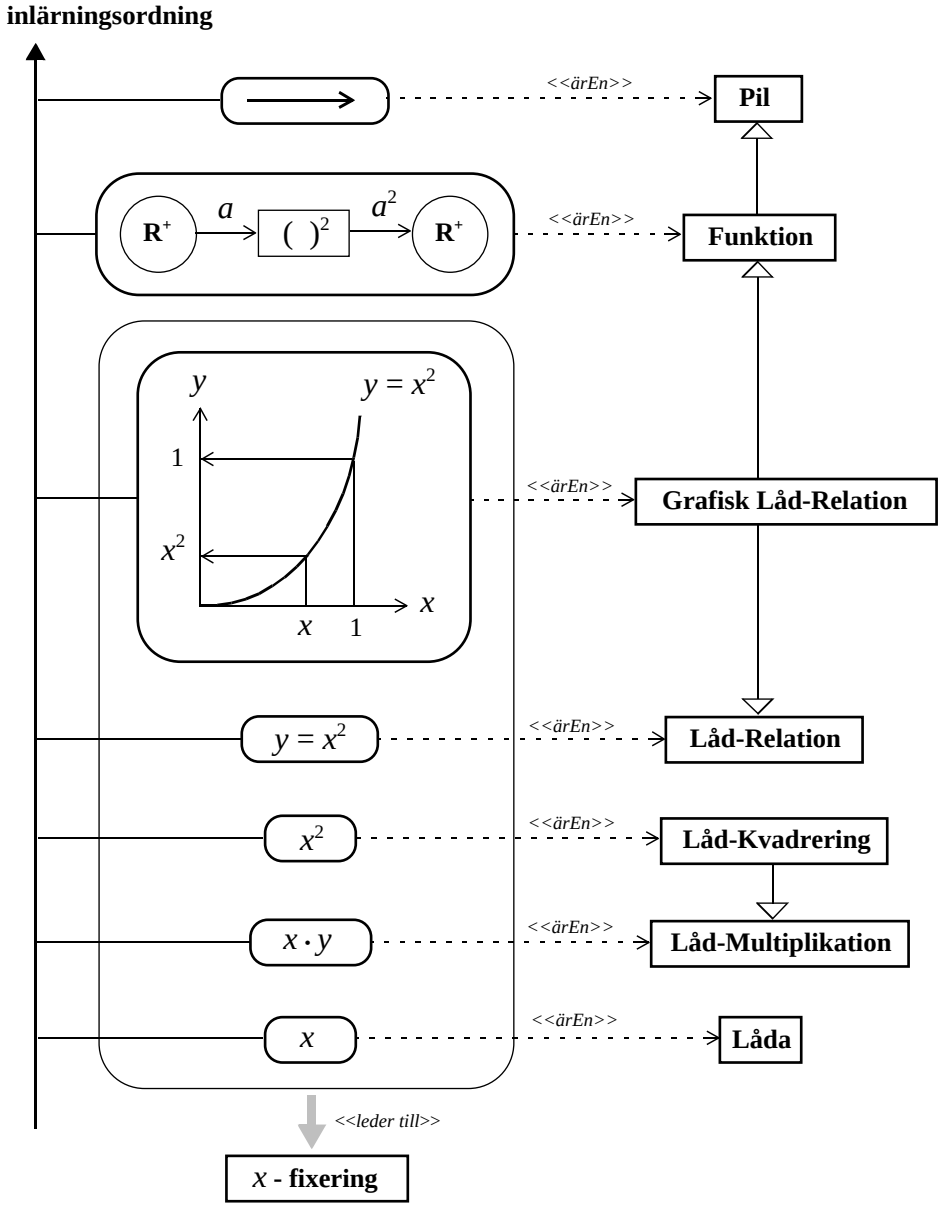

Fig. 35. Variabler och funktioner - traditionell respektive lådmässig approach.

Detta sätt att betrakta variabler är väldigt konkret och visade sig vara lätt att förstå för barnen. Det motsvarar dessutom exakt det sätt på vilket symboliska matematikprogram, som t.ex. Mathematica™ behandlar variabler. När man matar in ett namn på en variabel till Mathematica, svarar programmet med att ange variabelns värde (= lådans innehåll) - ifall något sådant värde har tilldelats variabeln (= lagts i lådan). I annat fall svarar Mathematica med att ange variabelns namn. Genom att demonstrera detta beteende - och förklara det i termer av att "räkna med lådor" - lyckades jag förankra idén om vad en variabel är för något hos var och en av de 25 eleverna - vid en tidpunkt då de gick i 4:e klass, dvs då de var ungefär 10 år gamla.

Figur (35) illustrerar det traditionella respektive det lådbaserade sättet att tänka på variabler och funktioner - två grundläggande begrepp som måste bemästras för att skaffa sig tillträde till matematikens högre abstraktionsnivåer. Den vänstra delen av figuren visar de olika abstraktionsnivåer som uppträder i den traditionella förklaringsmodellen, medan den högra delen visar hur motsvarande begrepp beskrivs i "lådmodellen". Notera att lådor med rundade hörn används för att beteckna instanser, vilket ibland är praktiskt ur tydlighetssynpunkt.

4 Referenser

[1] Clifford, W. K., *The Common Sense of the Exact Sciences*, Dover Publ. Inc., New York, 1955 (1945).

[2] Courant, R. & Robbins, H., *What is Mathematics?*, Oxford University Press, New York, 1978 (1941).

[3] Coxeter, H. S. M., *Projective Geometry*, Springer Verlag (2nd ed.), New York, 1987 (1964).

[4] Davis, P. J. & Hersh, R., *The Mathematical Experience*, Houghton Mifflin Co., Boston, 1981.

[5] Davis, P. J. & Hersh, R., *Descartes' Dream*, Houghton Mifflin Co., Boston, 1987.

[6] Euclid, *The Thirteen Books of the Elements*, Vol I-III, Dover Publ. Inc., New York, 1956 (325 B.C.).

[7] Gamma, E. & Helm, R. & Johnson, R. & Vlissides, J., *Design Patterns - Elements of Reusable Object-Oriented Software*, Addison-Wesley Publ. Co., Reading MA, 1995.

[8] Hadamard, J., *The Psychology of Invention in the Mathematical Field*, Dover Publ. Inc., New York, 1949.

[9] Hardy, G. H., *A Mathematicians Apology*, Cambridge University Press, Cambridge, 1948 (1940).

[10] Heath, T. L., *A History of Greek Mathematics*, Vol I-II, Dover Publications Inc., New York, 1981 (1921).

[11] Hilbert, D., *Grundlagen der Geometrie*, Teubner, Leipzig und Berlin, 1913.

[12] Klein, F., *Vorlesungen über höhere Geometrie*, Chelsea Publ. Co., New York, 1949 (1926).

[13] Naeve, A., *Conceptual Navigation and Multiple scale Narration in a Knowledge Manifold*, CID-52, TRITA-NA-D9910, KTH, 1999.

[14] Newton, I., *Philosophiæ Naturalis Principia Mathematica*, Cambridge, 1687.

[15] Poincaré, H., *Science and Hypothesis*, Dover Publications Inc., New York, 1952 (1905).

[16] Odell & Martin, *Object-Oriented Methods - a Foundation*, Prentice Hall, 1998.

[17] Rumbaugh, J. & Blaha, M. & Premerlani, W. & Eddy, F. & Lorensen, W., *Object-Oriented Modeling and Design*, Prentice Hall, New Jersey, 1991.

[18] Rumbaugh, J. & Jacobson, I. & Booch, G., *The Unified Modeling Language Reference Manual*, Addison Wesley Longman Inc., 1999.

[19] Sankaracarya, B. K. T., *Vedic Mathematics*, Banaras Hindu University Press, Varanasi, 1965.

[20] Schrödinger, E., *Nature and the Greeks*, Cambridge University Press, 1996 (1951).

[21] Schrödinger, E., *Science and Humanism*, Cambridge University Press, 1996 (1951).

[22] Wigner, E., P., *On the Unreasonable Effectiveness of Mathematics in the Natural Sciences*, Comm. in Pure Appl. Math., Vol. 13, 1960, pp. 1-14.

[23] Wittgenstein, L., *Tractatus Logico-Philosophicus*, Suhrkamp Verlag 1963 (1921).

[24] Wittgenstein, L., *Filosofiska Undersökningar*, Bonniers, Stockholm 1978 (1967).