OBJEKTORIENTERAD VERKSAMHETSANALYS

Carl-Johan Westin

TRIAD-PROJEKTET DRIVS GEMENSAMT AV TELIA, POSTEN, STATSKONTORET, ERICSSON OCH SISU, SVENSKA INSTITUTET FÖR SYSTEMUTVECKLING. FÖR VIDARE INFORMATION KONTAKTA SISU, ELECTRUM 212, 164 40 KISTA, TEL 08-752 16 00, FAX 08-752 68 00.

Kort om Modelleringshandboken

Inom Triad-projektets ram har parterna, d v s Ericsson, Telia, Posten, Statskontoret och SISU, beslutat sig för att satsa på ett generellt modellspråk för att analysera och beskriva verksamheter i generella konceptuella modeller. Resultatet av denna satsning utgörs av Modelleringshandboken.

Följande personer har deltagit i arbetet:

Agneta Hagberg, Posten GK-Data	Ann Rehbinder, Posten GK-Data
Malte Nordström, Telia Data	Margareta Pettersson, Ericsson Data
Claes-Göran Lindström, IT Plan	Hans Willars, SISU

Parterna bidrar successivt till Modelleringshandboken genom att producera separat utgivna avsnitt som ingår i en överordnad gemensam handbokstruktur. Som framgår av nedan är handboken indelad i ett antal block med delvis olika syften och målgrupper. De delar som är markerade med * har getts ut i en första utgåva.

Referenser inne i en text till andra handboksdelar markeras med titel i fet kursiv stil. Referenser till avsnitt i den här handboken markeras med med fet stil.

Handboksstrukturen

Block A: Översikter

Målgrupp: Ni som vill veta vad modellering är för att kunna var med.
Handboksöversikt*
Grundkunskap för modelleringsdeltagare

Block B: Handledningar

Målgrupp: Ni som har kommit i kontakt med modellering och vill kunna arbeta på egen hand eller leda ett modelleringsarbete.
Modelleringsledarens bashandledning*
Modelleringsteknik
Referensramar, angreppssätt
Modermodeller
Integration
Informatikövergång (Start: Modellbaserad datautformning)
Modellering i grupp*
Kommunikation*
Arbetsgångar* (Första utgåvan innehåller endast Verksamhetsanalys för informatikutveckling.)
Hjälpmedel (Start: Modelleringsväskan*)

Block C: Teorier

Målgrupp: Ni som vill ha djupare kunskap i modellering.
Referenslitteratur
Vardagsteori, teoretisk baskunskap
Teoriavsnitt efter behov

Block D: Hjälpmedel för kunskapsspridning

Målgrupp: Ni som vill visa, lära ut och sprida information om modellering.
Informationsmaterial (Start: Grundinformation)
Kursmaterial
Lärarhandledning (Start: Lärarhandledning Grundkurs)
Praktikfall: (Start: "Bilverkstaden")

Rapporterna beställs från
SISU • Electrum 212 • 164 40 Kista • Fax 08-752 68 0
Rapporterna är endast tillgängliga för TRIAD-parterna och är avgiftsfria

Innehåll

1. Inledning

Användningen av objektorientering har under de senaste decennierna ökat drastiskt inom systemutvecklingens alla faser. Framförallt har objektorienteringen anammats inom programmeringsvärlden. Här råder det idag inget tvivel om dess fördelar. Objektorienteringen har dessutom börjat användas i de tidiga skedena i systemutvecklingsprocessen.

Många objektorienterade analysmetoder gör anspråk på att täcka de tidiga skedena i systemutvecklingsprocessen. Dessa skeden modellerar verksamheten och producerar design som är oberoende av realisering. I detta sammanhang modellerar de objektorienterade metoderna och de rena verksamhetsanalysmetoderna delvis samma sak. Utifrån detta faktum har behovet av en belysande rapport uppstått.

Vilka arbetssätt och begrepp i en objektorienterad metod motsvarar arbetssätt och begrepp i en verksamhetsmodellering? Finns det likheter mellan verksamhetsmodeller och objektorienterade modeller? Vilka är då dessa och hur väl stämmer de överens?

Denna rapport beskriver tre objektorienterade analysmetoder och ställer därefter dessa mot den metod som används i modelleringshandboken.

Idag finns det ungefär tio välkända metoder för objektorienterad analys. Efter intern diskussion på SISU samt diskussion med parterna inom Triad avgränsades rapporten att omfatta tre metoder:

- *Object Oriented Analysis* (OOA)
- *Object Modeling Technique* (OMT)
- *ObjectOry* (version 3.3.0)

OOA, utvecklad av Peter Coad och Ed Yourdon, valdes därför att den anses vara den enklaste metoden att arbeta med och den lättaste att förstå. Metoden är dessutom en av de mest kända på marknaden.

OMT, utvecklad av James Rumbaugh m fl, valdes därför att partsföretagen och många internationella analytiker anser att den är den mest nyanserade metoden.

ObjectOry, utvecklad av Ivar Jacobsson och Objective Systems AB, valdes därför att den är välkänd i Sverige i allmänhet och av Triadparterna i synnerhet.

I rapporten görs en sammanställning av respektive metod. Sammanställningen beskriver de olika metodernas begrepp och i viss mån även deras notation. Rapporten strävar efter att beskriva de olika metoderna på ett objektivt och opartiskt sätt. Eventuella tolkningar av metodernas praktiska fördelar får läsaren själv göra.

De fördelar som anges för respektive metod är de som metodens utvecklare själva framhäver. Dessa argument är således inga objektiva sanningar, utan utvecklarnas egna utsagor.

Rapporten är uppdelad i fyra huvudkapitel:

Befintliga metoder, kapitel fyra, sammanfattar först kort de metoder som idag finns på marknaden. Sedan beskrivs de tre utvalda metoderna mer ingående.

Metodlikhet, kapitel fem, beskriver de egenskaper som förenar de tre metoderna.

Metodskillnader, kapitel 6, beskriver de egenskaper som är unika för respektive metod.

Paralleller till modelleringshandboken, kapitel 7, beskriver dels de egenskaper som har direkt motsvarighet i modelleringshandboken, dels objektorienterings-begrepp som kan uttryckas i modelleringshandbokens termer, dels begrepp som inte har någon motsvarighet i modelleringshandboken. I detta kapitel ges också förslag på hur objektorienterad teknik kan utnyttjas vid verksamhetsmodellering enligt Modelleringshandboken.

Förutom den litteratur som använts har många personer uttryckt åsikter och haft synpunkter under arbetets gång. Utrymme ges inte att nämna alla, men ett speciellt tack vill jag rikta till Christer Nelborn och Hans Willars för värdefull hjälp, som givit mig djupare förståelse för modelleringshandbokens termer och arbetssätt. Mer ingående förklaringar om ObjectOrys begreppsapparat och arbetssätt har Fredrik Lindström på Objective Systems givit.

I rapporten har översättningar av de engelska termerna gjorts för att underlätta läsningen. I de fall då inga motsvarande svenska ord förekommit, har de engelska originalbegreppen använts. Dessa begrepp har citattecken.

Innehållet i rapporten bygger helt på min tolkning av böckerna. Eventuella felaktigheter kan inte belasta någon annan än mig själv.

Jag hoppas Ni får en intressant läsning och jag tar tacksamt emot alla synpunkter gällande rapporten.

2. Syfte

Syftet med denna rapport är att visa vilka begrepp och arbetssätt som är gemensamma för verksamhetsmodellering och objektorienterad analys.

För att visa vad det finns för likheter mellan objektorientering och verksamhetsmodellering har tre objektorienterade analysmetoder undersökts. I dessa tre metoder har vissa gemensamma begrepp och arbetssätt identifierats. Dessa begrepp kan betraktas som kärnan i den objektorienterade ansatsen. Kärnan utgörs av ett antal grundbegrepp som definieras och används på liknande sätt i de olika metoderna. Utöver dessa grundbegrepp använder respektive metod unika begrepp. Även dessa beskrivs i denna rapport.

Rapporten visar vilka paralleller som finns mellan den objektorienterade analysen och den analys som beskrivs i modelleringshandboken. Detta görs genom att beskriva de objektorienteringsbegrepp vars motsvarighet återfinns i modelleringshandboken.

I rapportens avslutande kapitel beskrivs egenskaper som finns i de objektorienterade metoderna men som inte kan uttryckas i modelleringshandbokens termer. I sista avsnittet ges förslag på hur Modelleringshandboken kan förbättras genom att anamma vissa begrepp och arbetssätt som används i de objektorienterade metoderna. Dessa begrepp syftar vanligtvis på datorrelaterade företeelser, men kan användas för att uttrycka företeelser i en verksamhetsanalys.

Jag värderar inte de metoder som finns på marknaden. Rapporten beskriver inte heller vilka tillämpningsområden respektive metod lämpar sig för. Dessa slutsatser får läsaren själv dra.

3. Bakgrund

Bakgrunden till denna rapport är det intresse som på senare år ägnats den objektorienterade ansatsen. Från att ha varit ett sätt att programmera har detta synsätt utvecklats mot att i allt större utsträckning även behandla de tidiga skedena i utvecklingsprocessen. Detta har gjort att verksamhetsanalyser och objektorienterade analyser av informationssystem mer och mer överlappar.

I arbetet med att utveckla en metod för verksamhetsanalyser tas det fram en modelleringshandbok i Triad-projektet. Som en del i denna modelleringshandbok beskriver denna rapport vilka kopplingar som finns mellan verksamhetsmodellering och objektorienterad analys.

Läsaren bör vara förtrogen med de grundläggande begreppen inom informationsmodellering, funktionell nedbrytning och verksamhetsmodellering. Metoderna som behandlas är metoder för att analysera informationssystem, vilket innebär att de fokuserar aspekter som syftar till datoriserade system. Denna inriktning gör att modellerna bitvis färgas av datortermer och datortänkande.

Tidigare rapporter och undersökningar inom detta problemområde har studerats. "A Comparison of Six Object-Oriented Analysis and Design Methods", av Geert van den Goor, Shuguang Hong och Sjaak Brinkkemper, analyserar och jämför sex välkända objektorienterade analysmetoder. Denna rapport har producerats i samarbete mellan Georgia State University i USA, och Twente-universitetet i Nederländerna.

Sveriges Verkstadsindustrier har producerat två rapporter som båda berör den objektorienterade ansatsen: "Objektorientering, metoder & problemområden" av Stefan Sigfried och "Erfarenheter av Objektorienterad Systemutveckling" av Lars Wiktorin.

Förutom dessa rapporter har en rad föreläsningar givits inom området. På OOPSLA-konferensen 1992 redogjorde Martin Fowler för sina rön. Dessa publicerades i dokumentet "A Comparison of Object-Oriented Analysis and Design Methods".

4. Befintliga metoder

I detta kapitel beskrivs de olika objektorienterade metoder som finns idag. Första avsnittet ger en kort presentation av de mest kända metoderna. I de tre följande avsnitten beskrivs de utvalda metoderna mer ingående.

4.1 Översikt över befintliga metoder

Utbudet av metoder för objektorienterad analys ökar för var dag. Nedan följer ett försök till en sammanställning av de för närvarande mest kända objektorienterade analys- och design-metoderna.

- **BON, Business Object Notation**

BON har utvecklats av SOL i Frankrike tillsammans med Enea Data i Sverige. Utvecklingen har fått stöd av det europeiska forsknings- och utvecklingsprogrammet ESPRIT II.

BON är uppbyggd runt två modeller: en statisk och en dynamisk. Den statiska modellen visar systemets struktur. Denna struktur presenteras med klasser som sammanlänkas genom arvs- och klientrelationer. Klasserna grupperas i kluster. Den dynamiska modellen visar scenarier med objekt som kommunicerar. Dessutom visas systemets beteende i vissa typiska användningsfall.

- **DOOS, Designing Object Oriented Software**

DOOS har utvecklats av Wirfs-Brock m fl. Metoden fokuserar möjligheten att hantera den komplexa verkligheten genom abstraktion. Kunskap om problemområdet abstraheras och inkapslas i objekt. Detta är den stora skillnaden mellan de objektorienterade och de traditionella ansatserna. Den objektorienterade ansatsen fokuserar *vad* som sker, de traditionella *hur* det sker.
DOOS kan delas in i två delar:

1. Den initiala utforskande fasen. Här fastställer man objekt och deras uppgifter samt samarbetet mellan objekten och de roller de spelar i verkligheten.

2. Den detaljerade analysfasen. I denna fas förfinas och strömlinjeformas resultaten från den initiala fasen. Den fullständiga specifikationen av systemet slutförs även.

- **OODA, Object Oriented Design with Applications**

Grady Booch har utvecklat denna designmetod. För analysstegen hänvisar Booch till någon av de etablerade objektorienterade analysmetoderna. Resultatet från en sådan metod kan användas direkt för objektorienterad design. Booch nämner OOSA och OOA som möjliga analysmetoder.

Booch anser att de tre största fördelarna med objektorienterad design är att man utnyttjar de objektorienterade programspråken fullt ut, att man kan återanvända

klasser samt att objektorienterade designresultat är mer stabila och mindre känsliga för ändringar än traditionella designresultat.

- **OOSA, Object Oriented Systems Analysis**

Shlaer & Mellor har utvecklat denna metod, som fokuserar informationsmodellering. OOSA är en metod för att formalisera kunskap. Shlaer & Mellor beskriver inte objektorientering generellt, utan beskriver snarare hur en informationsmodell kan tas fram.

Arbetsstegen i OOSA är: analysera problemet, ange externa specifikationer, göra systemdesign och realisera. Tyngdpunkten i OOSA ligger på problemanalysen. De tre andra stegen beskrivs endast övergripande.

Analysfasen delas in i tre steg, där vart och ett bygger upp en modell. De tre modellerna är informationsmodell, tillståndsmodell och processmodell. Av dessa tre modeller beskriver utvecklarna endast informationsmodellen ingående. Denna modell är en sorts utökad Entity Relationship-modell.

- **OOAD, Object Oriented Analysis and Design**

Martins och Odells metod bygger, som de flesta andra objektorienterade analysmetoder, på att hantera den komplexa verkligheten med hjälp av abstraktion, generalisering och aggregering. Metoden har sina rötter i de teoretiska mängd- och logikteorierna.

OOAD försöker integrera de två aspekterna struktur och beteende i en processdel. Beteendeaspekten är dock övervägande, eftersom objektorienterad analys bygger på objekts olika beteenden. Strukturaspekterna identifieras och beskrivs som en härledning av beteendeaspekten.

OOADs första aktivitet är att identifiera målen inom problemområdet. Därefter definieras händelser och objekt i ett cykliskt mönster. En ny händelse leder till ett nytt objekt. När alla händelser identifierats och definierats är processen klar. Denna process kan vara antingen en ansats uppifrån och ner eller nerifrån och upp.

- **OOA, Object Oriented Analysis**

OOA har utvecklats av Peter Coad och Ed Yourdon. Metoden utmärker sig genom att använda ett fåtal begrepp och symboler. Detta leder till att metoden är lätt att sätta sig in i, men saknar i vissa situationer möjligheter att beskriva komplicerade företeelser.

Analysen byggs upp runt en modell, OOA-modellen, som beskriver den statiska strukturen i systemet. Denna modell kompletteras med diagram för att visa de dynamiska aspekterna.

Denna metod beskrivs ytterligare i kapitel 4.2.

- **OMT, Object Modeling Technique**

OMT har utvecklats av James Rumbaugh m fl, på General Electrics. Metoden påminner mycket om informationsmodellering. Många begrepp och arbetssätt har ärvts från Entity Relationship-scheman och -modellering.

Metoden byggs upp runt tre modeller: objektmodeller, dynamiska modeller och funktionsmodeller. Objektmodellen visar den statiska strukturen i systemet, den dynamiska modellen visar de tillstånd systemet kan befinna sig i och funktionsmodellen visar de bearbetningar som sker i systemet.

Denna metod beskrivs ytterligare i kapitel 4.3.

- **ObjectOry**

ObjectOry har utvecklats av Ivar Jacobsson. Analysen drivs av ett, eller flera, användningsfall. Dessa användningsfall specificerar de funktioner som systemet ska utföra. Utifrån dessa användningsfall byggs domänobjekt- och analysobjektmodeller upp. Dessutom utgår testnings- och verifieringsarbetet från dessa användningsfall.

ObjectOry utnyttjar en betydligt större arsenal av begrepp för att specificera ett system. Detta innebär att mer komplicerade företeelser kan beskrivas, men är därmed också svårare att lära sig.

Denna metod beskrivs ytterligare i kapitel 4.4.

4.2 Object Oriented Analysis

4.2.1 Bakgrund

Object Oriented Analysis har utvecklats av Peter Coad och Ed Yourdon. Som grund för metoden använder författarna människans sätt att varsebli verkligheten. För att belysa detta hänvisas kontinuerligt till Encyclopedia Britannica och Webster's:

För att varsebli verkligheten använder människan tre metoder att organisera sina intryck, vilka genomsyrar hennes sätt att tänka:

1. differentiering av upplevelser i speciella objekt och deras attribut – d v s när vi gör skillnad mellan ett träd och dess storlek eller dess rumsliga relation till andra objekt,

2. distinktionen mellan hela objekt och deras beståndsdelar – d v s när vi jämför ett träd och dess grenar samt

3. skapandet av och distinktionen mellan olika klasser av objekt – d v s när man skapar en klass av alla träd och en klass av alla stenar, och skiljer mellan dem.

Det objektorienterade paradigmet härstammar från 1960-talet. Under detta decennium utvecklades programspråket SIMULA, därefter introducerade Xerox

PARC Smalltalk på 1970-talet. Under dessa decennier förekom ingen diskussion om objektorienterad design, än mindre om objektorienterad analys.

Enligt utvecklarna av OOA har fyra förändringar under det senaste decenniet skapat förutsättningar för att anamma objektorienterad design och analys under 1990-talet.

- Det grundläggande konceptet objektorientering har mognat under det senaste decenniet. Uppmärksamheten har sakta skiftat från objektorienterad programmering via objektorienterad design mot objektorienterad analys. Funktionell nedbrytning utvecklades på samma sätt. Från början rörde den programmering och utvecklades successivt via design mot analys.

- Teknologin har gett oss möjlighet att bygga större och mer avancerade system. Däremot har inte vårt sätt att tänka och utveckla system utvecklats i samma takt. Än idag tänker många analytiker främst i termer av hur systemet ska realiseras och kodas.

- System som byggs idag är annorlunda än de som byggdes på 70-talet. I dag bygger vi betydligt större och mer komplexa system. Dessa är mer instabila och känsliga för ändringar, vilket till stor del beror på deras beroende av användargränssnitt och extern kommunikation. Objektorientering ger möjlighet att bygga stabilare system.

- Många system som byggs idag är dataorienterade. Funktionell komplexitet innebär inte samma svårighet som tidigare. Att modellera data har därmed givits högre prioritet.

Underlätta analysarbete
Författarna hävdar att fyra svårigheter alltid inträffar i analysarbetet, oavsett projektform. Dessa svårigheter är förståelse för *problemområdet*, *kommunikation* med problemområdesexperter, *ständiga förändringar* i problemområde och förutsättningar samt svårigheter att *återanvända* tidigare resultat. Genom att använda en objektorienterad analysmetod kan man behärska eller kringgå dessa svårigheter.

Problemområde och krav på systemet
En av de svåraste uppgifter en analytiker möter är att förstå problemområdet. Analytikern måste undersöka problemområdet och de krav som ställs på systemet. Ett av huvudsyftena med den objektorienterade ansatsen är att öka förståelsen för problemområdet och fokusera de problem som finns inom detta.

Kommunikation
För att kunna nå resultat måste analytikern kommunicera med problemområdesexperter. I och med att metoden underlättar en sådan dialog får man fördjupad förståelse för problemområdet och kan formulera de krav kunden ställer på systemet.

Ständiga förändringar
Den värld vi lever i är i ständig förändring. Därför måste analytiker och kund gemensamt avgöra när kraven ska "frysas", d v s i vilken given situation problemområdet ska avbildas. Problemområdet fortsätter att förändras, trots att kravspecifikationen förblir oförändrad.

För att kravspecifikationen ska kunna användas trots att verkligheten förändras delas den upp. De delar som är stabila läggs för sig och de delar som är instabila läggs för sig.

Återanvändning
Återanvändning är en av de största fördelarna vid objektorienterad programmering. Att kunna återanvända tidigare utfört arbete ges också i objektorienterad analys, vilket innebär att man använder tidigare modeller och analysresultat i det pågående arbetet. Återanvändning av analysresultat ger den största möjligheten till förbättrad systemutveckling.

Behärska komplexitet
Att använda en objektorienterad analysmetod ger stora möjligheter att behärska komplexitet. Dessa möjligheter anges av Coad & Yourdon i åtta punkter; *abstraktion, inkapsling, arv, association, kommunikation via meddelandeförbindelser, att organisera modeller, skalstorlek* och *att kategorisera beteende.*

Abstraktion
Abstraktion innebär att man bortser från vissa aspekter i problemområdet för att kunna koncentrera sig fullt på andra. Det finns två typer av abstraktioner:

Procedurell abstraktion innebär att en operation som utför en väldefinierad tjänst kan betraktas som en enhet av dess användare, trots att den i sig består av en sekvens mindre operationer. Att bryta ner en bearbetning i flera mindre bearbetningar och fortfarande betrakta den som en enhet är ett sätt att hantera komplexitet.

Dataabstraktion betyder att en datatyp definieras utifrån de operationer som ska utföras på dess attribut. Dessa tjänster kan bara manipulera attributvärden som tillhör det egna objektet. I praktiken menas att attribut definieras för objektetet utifrån de operationer som kan utföras på det. Dess värden kan endast manipuleras av tjänsterna i objektet. Attribut och tjänster ska betraktas som en inre helhet.

Inkapsling
Detta är en princip för att gömma information. Varje komponent i ett program ska gömma sina interna designbeslut. Fördelen med inkapsling är att det minimerar omarbetet vid utveckling av nya system. Analytikern placerar de delar i systemet som är mest föränderliga tillsammans. Detta minskar kommunikationen mellan olika delar av systemet.

Arv

Arv är en mekanism för att uttrycka likheter mellan klasser. Detta förenklar definitionen av klasser som är lika redan tidigare definierade klasser.
Arv möjliggör generaliseringar och specialiseringar, vilket gör det möjligt att specificera attribut och tjänster en gång per struktur, istället för en gång per klass.

Association

Association används för att knyta saker till varandra som händer vid ett visst tillfälle eller under liknande omständigheter.

Kommunikation via meddelandeförbindelser

Kommunikation via meddelandeförbindelser motsvarar imperativformen i det vardagliga språket, d v s är en uppmaning att utföra något.

Metoder för att organisera modeller

1. differentiering av upplevelser i objekt och deras attribut
2. distinktionen mellan hela objekt och deras beståndsdelar
3. skapandet av och distinktionen mellan olika klasser av objekt

Skalstorlek

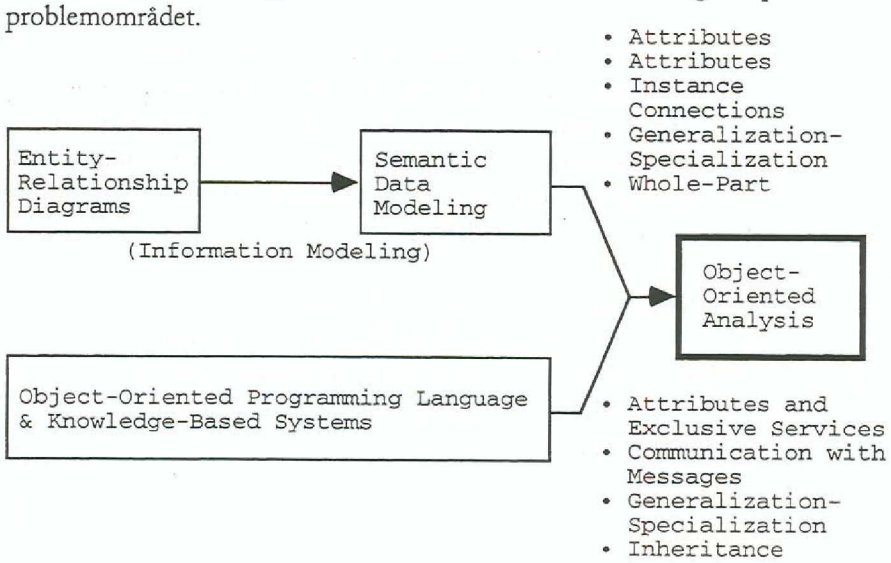
Med hjälp av skalstorlek skapas mönster för att lotsa läsaren genom stora modeller. Det är möjligt att se olika delar av modellen i

Beteendekategorier

Encyclopedia Britannica klassificerar beteende utifrån tre aspekter:

- utifrån dess omedelbara orsak
- utifrån likheter i dess historiska utveckling
- utifrån likheter i funktion

Analytikern måste förstå problemområdet. Genom att använda en objektorienterad ansats ökar analytikerns förståelse för problemområdet. Varje objekt ses som en abstraktion av verkligheten som fokuserar de väsentliga aspekterna i problemområdet.



Coad och Yourdons syn på framväxten av objektorienterad anlys
äxt

4.2.2 Fördelar med Object Oriented Analysis

I föregående avsnitt listades fyra förändringar som har skapat förutsättningar för objektorienterad analys att ge ett bättre resultat än de traditionella metoderna. Dessa förutsättningar gäller naturligtvis också för Object Oriented Analysis.

Utöver dessa generella punkter beskriver Coad och Yourdon vad deras metod erbjuder, nämligen möjligheten att:

1. Tackla mer komplicerade problemområden. OOA fokuserar förståelse för problemområdet.

2. Utveckla kommunikationen mellan analytiker och problemområdesexpert. OOA använder metoder som bygger på människans inneboende sätt att tänka.

3. Öka den interna stabiliteten i analysresultatet. OOA minskar avståndet mellan olika analysaktiviteter, genom att behandla attribut och tjänster som inre helheter.

4. Explicit representera likhet. OOA använder arvsmekanismer för att identifiera och visa likheter mellan attribut och tjänster, genom olika klasser i strukturen.

5. Skapa specifikationer som är töjbara vid förändringar.

6. Återanvända analysresultat.

7. Skapa en konsekvent representation för analys och design. OOA lägger grunden för fortsatt användning av analysrepresentationen genom att systematiskt utveckla denna vidare till specifik design.

4.2.3 Begrepp

I detta avsnitt beskrivs de begrepp som används i Object Oriented Analysis. Begreppen förklaras dels genom de definitioner som ges i boken, dels utifrån det sammanhang i vilket de används.

I sin bok hänvisar Coad och Yourdon vid definitioner till Webster's och Encyclopedia Britannica. Dessa definitioner kommer också att anges här där så är lämpligt.

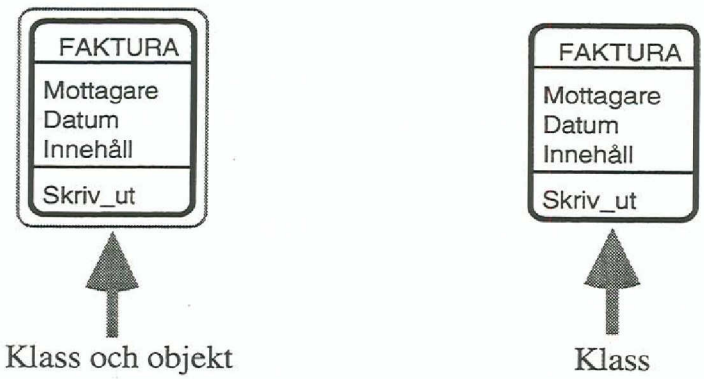
Klasser och Objekt

Enligt Webster's definition är objekt en person eller en sa, för vilken handlingar, tankar eller känslor är tydliga. Någonting synligt eller varaktigt; en materiell produkt eller substans.

Webster's definierar klass som en grupp människor eller saker, grupperade tillsammans utifrån en speciell likhet eller ett gemensamt drag.

I OOA tar författarna fasta på dessa definitioner och gör ett tillägg: Problemområdet och kraven som ställs på systemet ska beaktas.

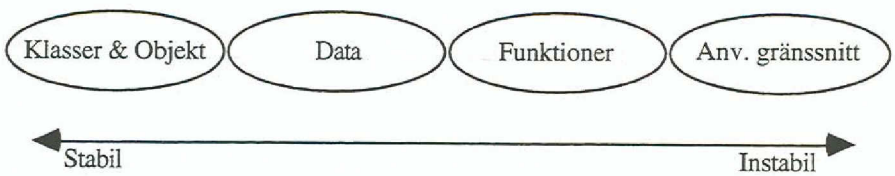
Objekt	En abstraktion av någonting inom problemområdet som speglar möjligheten hos systemet att hålla information om det, påverka det eller bägge delar; inkapsling av attributvärden och deras exklusiva tjänster. Synonym: instans
Klass	En beskrivning av en eller flera objekt med en enhetlig uppsättning attribut och tjänster, inklusive beskrivning av hur nya objekt skapas i klassen.
Klass&Objekt	En klass och objekten i den klassen.



Notation för klasser inklusive objekt och rena klasser

Anledningen till att använda klasser och objekt är att skapa en starkare koppling mellan den tekniska representationen av systemet och den konceptuella bilden av verkligheten.

Genom att urskilja klasser och objekt ur problemområdet skapar man förståelse och en grund för kommunikation mellan analytiker och problemområdesexpert. Ytterligare en anledning att använda klasser och objekt är att det leder till en stabil grund i analysarbetet. De är stabila genom hela utvecklingsprocessen, vilket leder till stabila analysresultat och återanvändbara enheter.



Oavsett vilket system man bygger så är klasser och objekt de mest stabila företeelserna.

Exempel
Bygger man ett kundregister kommer, med all sannolikhet, ett av de centrala begreppen att vara kund. Begreppet kund kommer att vara stabilt under systemets livstid, medan dess egenskaper och beteende kommer att förändras. Datatyper som lagras i klassen kommer under utvecklingens gång att ändras, man kommer att lägga till och ta bort attributtyper. De funktioner som utförs på kundobjektet kommer att utvecklas, mer avancerade operationer kommer att efterfrågas. De tjänster som förändras i snabbast takt är de som är ansvariga för interaktionen mellan system och användare, t ex gränssnittsoperationer.

Inkapsling av attribut och tjänster i objekt ger möjlighet att betrakta dem som inre helheter. Analytikern fokuserar således objekts tillstånd och beteende tillsammans, istället för vart för sig.

Hur hittar man klasser och objekt? Författarna ger tips om var man ska leta efter klasser och objekt. Exempel ges också på vad som kan betraktas som objekt. Arbetet med att hitta adekvata objekt är kärnpunkten i analysarbetet. De resultat som skapas i denna fas, formar den vidare analysen. Författarna hävdar att man ska sätta sig in i användarens situation och uppleva användarens arbetsmönster. Problemområdesexperter ska intervjuas, tidigare OOA-resultat ska kontrolleras, jämförelser ska göras med andra system och prototyper ska produceras.

Författarna ger också tips på vad man ska leta efter. Vid en undersökning av problemområdet ska man leta efter strukturer, vilket innebär att om man hittar ett objekt ska man se om detta har något strukturellt samband med andra objekt. Man ska dessutom undersöka vilka andra system systemet kommunicerar med, vilka saker och händelser man måste spara information om, användare som systemet kommunicerar med samt organisationsenheter som systemet håller information om.

När man har hittat ett objekt ska kontrollfrågor ställas för att fastställa om objektet verkligen är kvalificerat. Dessa frågor kan röra huruvida information om objektet ska sparas, om objektet har något eget beteende, om objektet har flera attribut, om värden kan lagras för varje attribut o s v.

Attribut
Webster's definierar attribut som egenskaper, kvaliteter eller karaktäristika som beskriver en person eller sak.

OOA-begreppet attribut har definierats så att det dessutom ska spegla problemområdet och ta hänsyn till de krav som ställs på systemet.

Attribut	Ett attribut är data (tillståndsinformation) för vilket varje objekt i en klass har sitt eget värde.

Att bestämma vilka attribut ett objekt ska ha är en fråga om urval och begränsning. Objekten ska bara spegla de tillstånd som är relevanta i problemområdet. Det uppstår här en svårighet att avgränsa vad som hör till problemområdet och vad som ligger utanför det. De krav som ställs på systemet ska också beaktas.

Attribut beskriver de värden (tillstånd) som ett objektet har. Dessa värden får endast ändras av objektets egna tjänster. Om en annan del av systemet behöver ändra objektets värden, d v s tillstånd, ska operationen utföras av en tjänst specificerad i objektet som innehåller attributet. Denna begäran skickas i en meddelandeförbindelse.

För att hitta attribut ska man granska hur ett objekt beskrivs generellt, hur det beskrivs specifikt för problemområdet och hur det beskrivs med hänsyn till de krav som ställs på systemet. Därefter granskar man vad objektet behöver lagra för information, vilka tillstånd det ska kunna befinna sig i och vilken tillstånds-information som ska lagras. Dessutom bör tidigare analysresultat granskas för att se om några attribut kan återanvändas.

När attribut definieras ska de endast lagra atomära värden, d v s ett enkelt värde eller en tätt sammanknuten grupp värden. Detta skapar enklare modeller.

Attribut som endast har till syfte att identifiera objekt ska ej tas med. Objekt är i sig unika, varför identitet ej behöver anges.

Kalkylerbara attribut behöver inte tas med i analysfasen. Dessa kan identifieras och anges vid senare aktiviteter i utvecklingsprocessen.

Attributen ska placeras i det objekt där de bäst hör hemma. Om det är oklart vilket objekt attributet hör till är det bra att studera problemområdet och de krav som ställs på systemet. I en specialiseringsstruktur ska attributen placeras så högt som möjligt i strukturen.

Instansrelation

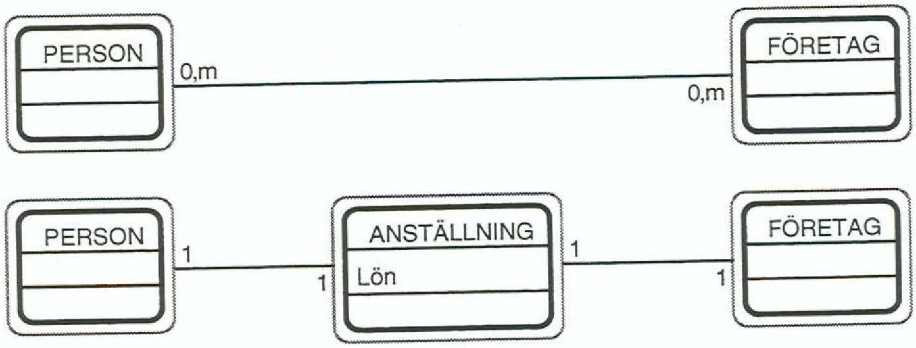
Instansrelationer visar att vissa objekt kräver relationer till andra objekt för att utföra sina uppgifter.

Instansrelation	En instansrelation är en modell över problemområdet som visar att ett objekt kräver information om ett annat objekt för att fullfölja sina skyldigheter.

Vid instansrelationer anges kardinalitet, d v s hur många objekt som deltar i relationen. Vid respektive objekt anges hur många instanser av den motstående klassen som måste förekomma.

Relationer mellan objekt kan vara frivilliga (minsta antal deltagare är noll) eller obligatoriska (minsta antal deltagare är en).

Vid många-till-många-relationer ska den som analyserar vara aktsam. I dessa situationer kan det vara aktuellt att introducera ett objekt som exklusivt knyter ett objekt i en många-relation till ett annat objekt i en annan många-relation.



Instanser i en klass ses som en mängd, d v s ingen inbördes ordning förekommer dem emellan. Möjlighet finns dock att ange restriktioner för relationer vilket visar att instanserna i klassen ordnas efter ett visst kriterium. Instanserna blir då en del i en lista.

Generaliserings/specialiserings-struktur

Webster's definierar struktur som ett sätt ett organisera. För att anpassa denna definition till OOA-termer måste problemområdet beaktas och hänsyn tas till de krav som ställs på systemet.

Struktur Struktur är ett uttryck för problemområdets komplexitet, vilket genomsyrar kraven som ställs på systemet. Termen struktur används som ett generellt begrepp som beskriver både generaliserings/specialiserings-struktur och hel/del-struktur.

Gen/spec-strukturen är en klasstruktur, d v s en klass är en specialisering av en annan klass. Den generella klassen kan antingen vara ett klass&objekt, d v s innehålla instanser, eller bara vara en klass, d v s inte innehålla några instanser. Ett objekt som tillhör en specialiserad klass har sina attribut och tjänster, samt alla de attribut och tjänster som de generaliserade klasserna har. En specialiserad klass kan förfina de ärvda attributen och tjänsterna, däremot ej radera dem.

Gen/spec-strukturen kan antingen forma en hierarki eller ett nätverk. De flesta strukturer är hierarkier vilket innebär att varje klass endast har en "far". Vid nätverk introducerar man multipla arv, d v s en nod kan ha två eller flera "fäder". Nätverk förstärker specialiseringen och visar explicit likheter. Vid användning av nätverksrelationer ökar dock modellens komplexitet.

Om samma namn används för attribut eller tjänster i flera klasser i en nätverksrelation, kan svårigheter förekomma för den klass som ärver dessa egenskaper. Den klass som ärver två attribut med samma namn från olika klasser upplever då en konfliktsituation om vilken av egenskaperna som ska gälla.

Hel/del-struktur

Hel/delar är en aggregeringsstruktur. Denna struktur är en av människans grundläggande metoder för att organisera intryck och detta genomsyrar hela vårt sätt att tänka.

Hel/del-strukturen relaterar objekt till varandra, d v s är en objektstruktur. Med detta menas att ett objekt består av ett eller flera andra delobjekt.

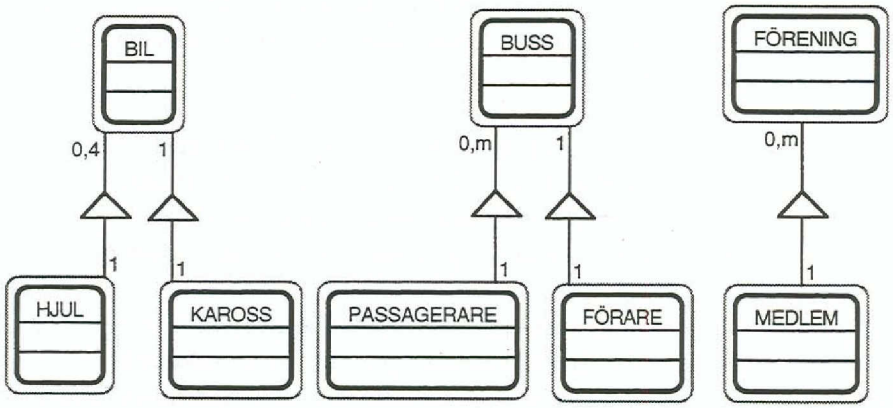
I hel/del-strukturen anges kardinalitet, d v s hur många objekt som deltar i varje relation. Antalet delobjekt anges vid objektet.

Hel/del-strukturer kan vara av tre olika slag. Denna distinktion är dock endast tankemässig. Notationen är den samma.

Tre olika slag av hel/del-strukturen:

- Sammansättning – del: En bil är sammansatt av hjul, kaross m m.
- Innehåller – innehåll: En buss innehåller chaufför och eventuellt passagerare.
- Samling – medlemmar: En taxichaufför kan vara medlem i en fackförening.

Samling – medlem är endast en mental avbildning av verkligheten. Denna används för att behärska komplexiteten i problemområdet, vilket ger analytikern ökad förståelse och främjar kommunikationen med problemområdesexperter.



Tjänst

Webster's definierar en tjänst som en aktivitet som utförs för att förse människor med nyttan av någonting. För att anpassa detta till OOA-termer måste också hänsyn tas till problemområdet och de krav som ställs på systemet.

Tjänst	En tjänst är ett specifikt beteende som ett objekt är ansvarigt för att utföra.

Författarna refererar också till Encyclopedia Britannica, där man gör en distinktion mellan olika beteenden. Det finns tre kriterier för att klassificera beteende:

- utifrån dess omedelbara orsak
- utifrån likheter i dess historiska utveckling
- utifrån likheter i funktion

"Utifrån dess omedelbara orsak". Detta är ett sätt att gruppera operationer som initieras av en viss gemensam anledning.

"Utifrån likheter i dess historiska utveckling". På detta sätt kan operationer grupperas utifrån tidigare händelser. En operation kan bete sig olika om objekten den opererar på har utvecklats olika. Den historiska utvecklingen dokumenteras i objektets attribut. De värden som lagras där avgör hur operationerna ska bete sig.

"Utifrån likheter i funktion". Operationer som utför liknande funktioner kan grupperas tillsammans.

Objekts tillstånd kan vara beroende av hur objektet utvecklats. De tillstånd som objektet genomgått kan styra det beteende objektet kan utföra.

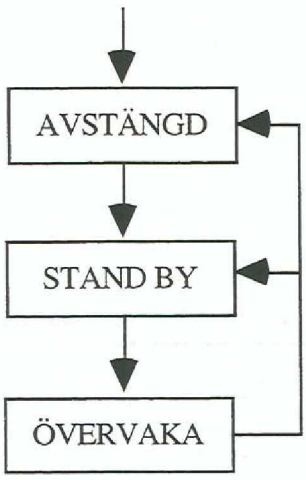
De tjänster ett objekt är ansvarigt för att utföra bygger på principerna om liknande funktioner och orsaken till deras utförande.

För att identifiera de tjänster som respektive objekt ska utföra anger författarna fyra arbetssteg. Dessa steg bildar en iterativ process.

- Identifiera objektets möjliga tillstånd.
- Identifiera efterfrågade tjänster.
- Identifiera meddelandeförbindelser.
- Specificera tjänster.

Varje objekt befinner sig under sin livstid i flera olika tillstånd. Tillståndet representeras av de värden som finns i objektets attribut. Varje förändring av ett attributvärde speglar en förändring i objektets tillstånd. För att identifiera ett objekts tillstånd bör tänkbara värden för varje attribut undersökas. Därefter avgörs om dessa värden ligger inom problemområdet och om de uppfyller de krav som ställs på systemet.

Tillståndsdiagram
Under en inventering av de tillstånd ett objekt kan anta, underlättar det att rita tillståndsdiagram. Dessa diagram visar de tillstånd ett objekt kan befinna sig i under sin livstid. Diagrammet visar endast tillstånd och accepterade händelser.



Tillståndsdiagram för klassen radar

De tjänster som objekten ska utföra kan delas upp i två kategorier: algoritmiskt enkla och algoritmiskt komplicerade. OOA definierar fyra algoritmiskt enkla tjänster. Dessa är:

- *Create*, som skapar ett nytt objekt i en klass.
- *Connect*, som knyter eller lösgör ett objekt till/från ett annat objekt.
- *Access*, som hämtar eller lagrar värden i ett objekts attribut.
- *Release*, som tar bort ett objekt från en klass.

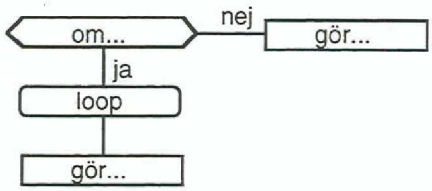
Dessa fyra tjänster står för merparten av de operationer som utförs i systemet. Enligt beräkningar som författarna gjort utgör de enkla tjänsterna 80–95% av det totala antalet utförda tjänster. De algoritmiskt enkla tjänsterna visas inte i modellen utan anses vara implicita för varje objekt.

De algoritmiskt komplicerade tjänsterna är sådana som beräkning av värden (calculate) och övervakning av externa enheter (monitor).

När tjänster ska namnsättas föreslår författarna att man använder domänspecifika namn. Detta ger fördjupad förståelse för problemområdet och underlättar kommunikationen mellan analytiker och problemområdesexpert.

Tjänstediagram

När tjänsterna identifierats och knutits till objekten ska de specificeras. OOA använder tjänstediagram för att specificera detta. Ett tjänstediagram liknar mycket ett flödesschema. För att stega sig igenom en algoritm anser författarna att tjänstediagram, till skillnad från dataflödesscheman, är idealiska. I tjänstediagrammet kan villkor, loopar, kopplingar och instruktioner uttryckas. Tjänstediagrammet kan uttrycka tillståndsberoende beteende genom att använda villkor. Dessa kan vara förvillkor, utlösare och avslutare.



Schema för tjänstediagram

För att sammanfatta de tillståndsberoende tjänsterna finns möjlighet att upprätta en tabell som omfattar tjänster och tillstånd. I tjänste/tillståndstabellen visas de tjänster som ett objekt kan utföra när det befinner sig i ett visst tillstånd.

	Tillstånd 1	Tillstånd 2	Tillstånd 3
Tjänst A	●		
Tjänst B			●
Tjänst C	●		
Tjänst D		●	
Tjänst E	●	●	●

OOAs tjänste-/tillståndsdiagram

Subjekt
En av de viktigaste egenskaperna hos en metod är dess förmåga att underlätta kommunikation och undvika informationsöverflöd. Stora modeller bör delas in i flera mindre delmodeller när de innehåller mer än 35 klasser. Coad och Yourdon använder begreppet subjekt för delmodeller.

Författarna vänder sig även här till Webster's för en generell definition av begreppet. Webster's definierar subjekt som det som behandlas eller handhas i en diskussion, studie, skrift, målning, et c; ett tema eller ett ämne.

OOA-termen subjekt speglar dessutom problemområdet och de krav som ställs på systemet. Coad och Yourdon definierar begreppet enligt följande:

> Subjekt Ett subjekt är en mekanism för att guida läsaren genom en stor, komplex modell. Subjekt är också till hjälp när man organiserar paket i stora projekt, vilka baseras på tidiga OOA-undersökningar.

Genom att använda skalstorlek, aggregering och subjekt lotsas läsaren genom stora modeller.

I OOA skapas subjekt genom att låta den "översta" klassen i varje hierarki forma ett subjekt. Det är viktigt att sträva efter minimalt beroende och minimal interaktion mellan subjekten. För att modellera subjekt ritar man en rektangel runt klasserna som ska bilda subjektet.

4.2.4 Arbetsgång
OOA är uppdelat i fem huvudaktiviteter. Dessa är:

- Hitta klasser och objekt.
- Identifiera strukturer (gen/spec och hel/delar).
- Identifiera subjekt.
- Definiera attribut (attribut och instansrelationer).
- Definiera tjänster.

Dessa är aktiviteter och inte sekventiella steg. De utförs inte i någon uttalad ordning.

Vissa analytiker anser att det är smidigast att först identifiera klasser och objekt och därefter identifiera och definiera attribut för varje enskild klass. När klasserna, inklusive attribut, har identifierats undersöks problemområdet för att finna strukturer mellan klasserna och tjänster hos dem.

Andra analytiker kan arbeta på ett helt annat sätt. De identifierar ett objekt, definierar dess tjänster och struktur och till sist anges vilka attribut som är nödvändiga. Arbetssätten är individuella och varierar från analytiker till analytiker.

Vid mer komplicerade problemområden kan det vara aktuellt att definiera subjekt för att vägleda läsaren genom stora modeller.

Det är upp till analytikern att bestämma arbetets gång. Generellt karakteriseras arbetet av att man går från en hög abstraktionsnivå, med klasser och objekt, till en mer detaljerad låg abstraktionsnivå med attribut och tjänster.

Författarna framhåller att analytikern ska sätta sig in i och förstå problemområdet. Det viktigaste är att känna till användarens situation och därmed kunna sätta sig in i vilken roll systemet kommer att ha i användarens miljö.

4.3 Object Modeling Technique

4.3.1 Bakgrund

Object Modeling Technique, OMT, har utvecklats av James Rumbaugh m fl. De utvecklade metoden under sin tid på General Electric Research and Development Center. General Electric har investerat tid och pengar i utvecklingsprojektet.

Författarna har arbetat med objektorienterad analys, design och programmering under flera år, i en rad olika projekt. De anser sig vara förtrogna med såväl de teoretiska aspekterna som de praktiska.

I de flesta projekt där de använt den objektorienterade ansatsen har den visat sig framgångsrik. Med hjälp av objektorienterad systemutveckling anser författarna att man kan höja kvaliteten, flexibiliteten och förståelsen för de system som byggs.

Fördelarna med den objektorienterade ansatsen är många enligt författarna. Den viktigaste fördelen är att de objekt som fokuseras är tagna ur verkligheten. Dessa objekt kan därefter användas för att bygga språkoberoende design.

Författarna anser att objektorienterad analys skapar bättre förståelse för kraven som ställs på systemet, ger renare design och leder till system som är lättare att vidmakthålla. Målsättningen är att de objektorienterade begreppen ska användas genom hela systemets livscykel, från analys via design till realisation.

Författarna understryker att objektorientering är betydligt mer än ett sätt att programmera. Det viktigaste i den objektorienterade ansatsen är att analytikern hjälps till att tänka abstrakt om problemet, samtidigt som begrepp från verkligheten används. På detta sätt kan man undvika datorrelaterade begrepp i analysfasen.

Den grundläggande egenskapen i en objektorienterad ansats är att objekten innesluter både data och beteende i en enhet. Objektorienterade analysmodeller är användbara för att förstå problemområdet, kommunicera med problemområdes-experter, modellera verksamheter samt skapa realisationsmodeller och databaser. En analysmodell skapas för att abstrahera väsentliga aspekter i problemområdet, utan att ta hänsyn till hur de ska realiseras. Modellen innehåller objekt som hämtas från problemområdet.

För att karakterisera objektorientering krävs fyra egenskaper: *identitet*, *klassificering*, *polymorfism* och *arv*.

Identitet
Identitet innebär att varje objekt är unikt och har sin egen inneboende identitet. Med andra ord, två objekt är två olika identiteter även om deras attributvärden är identiska.

Klassificering
Klassificering innebär att objekt med samma datastruktur (attribut) och samma beteende (operationer) grupperas i klasser. En klass är en abstraktion som beskriver egenskaper som är viktiga inom problemområdet, och undviker dem som är oväsentliga. Varje klass beskriver ett oändligt antal objekt. Ett objekt kallas en instans i sin klass och har egna värden i sina attribut, men delar attributnamn och operationer med de övriga instanserna i klassen.

Polymorfism
Polymorfism innebär att samma operation kan bete sig olika för olika klasser. En operation är en handling eller förändring som ett objekt utför eller utsätts för. En realisering av en operation för en speciell klass kallas en metod. Eftersom en objektorienterad operation ska vara polymorf kan den realiseras som mer än en metod. Resultatet av operationen ska alltid vara det samma för de olika klasserna.

I verkligheten är en operation en abstraktion av ett motsvarande beteende för olika sorters objekt. Varje objekt vet hur det ska utföra sina operationer.

Arv
Arv låter klasser dela attribut och operationer, baserat på deras hierarkiska relation. Klasser kan definieras allmänt och därefter förfinas i underklasser. Varje underklass ärver alla egenskaper hos "föräldraklasserna" och lägger till dem sina egna attribut och operationer. Möjligheten att sortera likheter ur flera klasser och lägga dessa i en gemensam föräldraklass minskar redundansen i modellen.

Den största fördelen med objektorientering erhålls när man använder den redan i analysarbetet. Brister i design som upptäcks under realisering är betydligt mer kostsamma att korrigera än om de upptäcks i ett tidigare skede.

Att koncentrera sig på realiseringsfrågor alltför tidigt i utvecklingsprocessen begränsar designmöjligheterna och leder ofta till undermåliga produkter. En objektorienterad ansats uppmuntrar analytikern att arbeta och tänka i termer som används inom den domän där tillämpningen ska användas. Dessa termer och detta tankesätt utnyttjas genom hela utvecklingsprocessen. Det är först när begreppen inom tillämpningsområdet identifieras, organiseras och förstås som datastrukturen och funktionerna kan detaljeras och användas effektivt.

Objektorienterad utveckling är en konceptuell process, oberoende av programmeringsspråk fram till de sista arbetsstegen. Objektorientering är framförallt ett nytt sätt att tänka, inte ett sätt att programmera. Dess största nytta

framträder när den används för att hjälpa utvecklare, analytiker och problemområdesexperter att uttrycka de abstrakta begreppen tydligt och förmedla dem till andra.

Det finns ytterligare möjligheter som den objektorienterade ansatsen ger. Dessa är *abstraktion, inkapsling, kombination av data och beteende, möjlighet att dela strukturer* samt *fokusera objektstrukturer i stället för procedurstrukturer.*

Abstraktion
Abstraktion innebär att man koncentrerar sig på det väsentliga inom problemområdet och utelämnar det som ligger utanför de krav som ställs på systemet. Om man använder abstraktion på rätt sätt ges möjlighet att använda samma modell i analysfasen, vid design av programstruktur och realisation samt vid produktion av dokumentation.

Inkapsling
Inkapsling innebär att man gömmer information. De externa och interna aspekterna hos objektet skiljs åt. De externa egenskaperna är tillgängliga för alla objekt, de interna är endast tillgängliga för intern bearbetning inom objektet.

Inkapsling förhindrar att små ändringar får följdeffekter i stora delar av systemet. Möjligheten att kombinera data och beteende gör att inkapslingen blir klarare och kraftfullare.

Kombination av data och beteende
Kombination av data och beteende innebär att det endast är ett objekts egna operationer som kan påverka värdena i objektets attribut. Det objekt som vill utföra en operation på ett annat objekt behöver bara begära att operationen ska utföras.

Delad struktur
Att dela strukturer gör det möjligt att ärva redan tidigare utfört arbete. Detta görs genom att skapa en generaliseringsstruktur, där specialiseringarna ärver attribut och beteende från sina föräldrar. Förutom att det spar arbete, så minskar det mängden redundanta data i modellen.

Objektstruktur kontra procedurstruktur
Objektorientering fokuserar vad objekt är, snarare än hur de används. Ett objekts egenskaper är betydligt mer stabila än de sätt på vilket det används. Detta får till följd att system som byggs utifrån den objektorienterade ansatsen ofta är betydligt mer stabila än system som byggs på traditionellt sätt.

4.3.2 Fördelar

OMT bygger på tre modeller. Dessa tre modeller är *objektmodell, dynamisk modell* och *funktionsmodell.* Varje modell avspeglar en viktig del av systemet. Objektmodellen visar objekten i tillämpningsdomänen, deras attribut och relationer till varandra. Dynamiska modellen beskriver interaktionen mellan objekten i systemet och funktionsmodellen beskriver hur data förändras i systemet.

De tre modellerna delar systemet i olika vyer. Dessa representeras och manipuleras med en enhetlig notation. Modellerna är fristående från varandra. Bra design uppnås när man isolerar de olika aspekterna hos systemet och begränsar kopplingen mellan dem. Varje modell kan granskas och förstås för sig.

Det finns dock ett visst beroende mellan modellerna – ett system är trots allt mer än en samling oberoende delar. Varje modell innehåller referenser till de övriga modellerna. Alla tre modeller behövs för att ge en fullständig bild av systemet. De kopplingar som finns mellan modellerna är emellertid begränsade och explicita.

I OMT används samma begrepp och notation genom hela utvecklingsprocessen vilket ger en stabil grund för fortsatt arbete och en jämn övergång mellan de olika utvecklingsstegen.

Författarna poängterar slutligen att modeller är abstraktioner som byggs för att förstå ett problemområde.

Objektmodell
Objektmodellen beskriver den statiska strukturen i systemet. Objekt visas med relationer, attribut, identitet och operationer. Objektmodellen skapar en ram för systemet. Inom denna ram utformas den dynamiska modellen och funktionsmodellen. Objekten är de enheter som utsätts för förändringar och transformationer.

Objektmodellen fångar begrepp från verkligheten som är väsentliga för tillämpningen. Denna modell ska inte innehålla datorlösningar eller realiserings-specifikationer.

Det grafiska resultatet är ett objektdiagram.

Dynamisk modell
I den dynamiska modellen beskrivs de aspekter i systemet som är beroende av tiden. Tidsberoende gäller bl a sekvensiering av operationer, händelser som markerar förändring, sekvenser av händelser, tillstånd som definierar sammanhanget för händelser samt organisering av tillstånd och händelser. Dynamiska modellen beskriver sekvenser av operationer som inträffar, utan att ta hänsyn till vad operationerna gör, vad de opererar på eller hur de realiseras.

Grafiskt representeras detta i tillståndsdiagram. Dessa tillståndsdiagram visar tillståndet och händelsesekvenser som tillåts för en klass. Tillståndsdiagrammet refererar till de övriga OMT-modellerna.

Handlingar (actions) i tillståndsdiagrammet motsvarar funktioner i funktions-modellen. Händelser (events) i tillståndsdiagrammet motsvarar operationer på objekt i objektmodellen.

Funktionsmodell
Denna modell beskriver de aspekter i systemet som behandlar förändring av värden i objekten – funktioner, restriktioner och funktionellt beroende. Den funktionella modellen visar vad systemet gör, utan att ta fasta på hur eller när det görs.

Modellen representeras grafiskt med dataflödesdiagram. Dataflödesdiagrammet visar beroendet mellan värden och beräkningar, utan att ta hänsyn till när funktionerna utförs.

Funktioner representeras som handlingar i den dynamiska modellen och som operationer på objekt i objektmodellen.

4.3.3 Objektmodellering

I detta avsnitt förklaras de olika begrepp som används i objektmodellen och under objektmodellering. Definitionerna är hämtade ur texterna eller ur textens sammanhang i Rumbaughs bok.

Syftet med objektmodellering är att visa den statiska strukturen i systemet. Detta görs genom att visa objekt som finns i problemområdet och deras relationer till varandra. Varje klass av objekt har en unik uppsättning attribut och operationer.

Objektmodellen är den viktigaste av de tre OMT-modellerna, eftersom den är nära knuten till verkligheten och visar stabila företeelser.

Objekt

Objekt	Ett begrepp, en abstraktion eller en sak med klara gränser och innebörd för det aktuella problemet.

Det finns två anledningar att använda objekt som grund för en analys. Dels för att öka förståelsen för problemområdet, dels för att objekt utgör en stabil grund inför realiseringen. Det finns inget korrekt sätt att dela upp ett problemområde i objekt. Det är en fråga om omdöme och problemområdets slag. Ingen lösning är den bästa. Man ska sträva efter en tillfredsställande lösning.

Alla objekt är unika. Objekt urskiljs genom sin inneboende existens och inte av de egenskaper som beskriver dem.

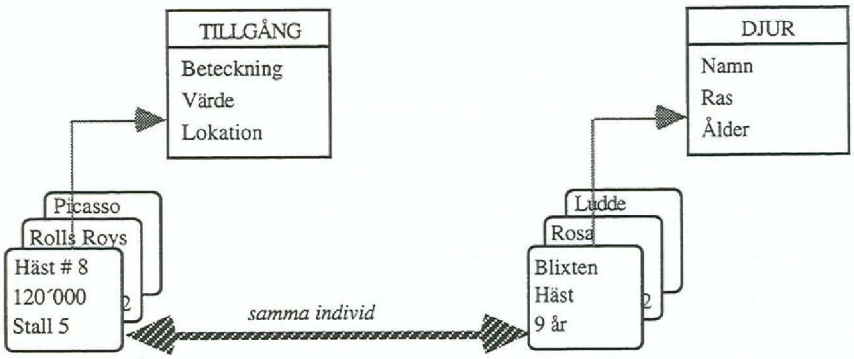
Ordet objekt används i olika sammanhang. Ibland betyder objekt en enskild sak, vid andra tillfällen en grupp liknande företeelser. För att definiera detta använder OMT-begreppet objektinstans när enskilda företeelser refereras, och klass när en grupp liknande objekt åsyftas.

Klass
En objektklass är en grupp objekt med liknande egenskaper.

Klass	En klass är en grupp objekt med gemensamma egenskaper (attribut), gemensamma beteenden (operationer), gemensamma relationer och gemensam semantik.

Alla objekt i en klass har samma attributtyper och beteendemönster. De flesta objekt i en klass har olika värden i sina attribut. Det kan dock förekomma objekt med identiska attributvärden och identiska beteendemönster. Dessa, identiskt lika, objekt är trots detta olika identiteter.

Objekten i en klass delar samma semantiska syfte. Tolkningen av den semantiska betydelsen beror på det sammanhang och det problemområde objektet befinner sig i.



Klass och dess instanser inom problemområdet ekonomiska tillgångar

Klass och dess instanser inom problemområdet husdjur

Genom att gruppera objekt i klasser, bortser vi från det oväsentliga i problemområdet och visar det väsentliga. Detta ger möjlighet till generaliseringar, så att gemensamma definitioner kan lagras en gång per struktur istället för en gång per klass.

Klasser och objekt visas i objektdiagram. Objektdiagram möjliggör en formell notation för att modellera klasser, objekt, deras relation till varandra, attribut och operationer. Dessa diagram är användbara både för abstrakt modellering och för att "designa" program.

Abstrakt klass

Den abstrakta klassen samlar egenskaper som är gemensamma för flera olika klasser. Det är bekvämt att kapsla in klasser som deltar i samma association eller aggregering i en superklass (se Aggregering s. xnågra sidor fram). Vissa abstrakta klasser uppstår naturligt utifrån problemområdet, andra introduceras som mekanismer för att främja återanvändning.

Abstrakt klass En klass som inte har några instanser.

En abstrakt klass måste således bestå av klasser som är instansierade eftersom den inte kan vara det själv. Den abstrakta klassen kan inte bestå av konkreta bilar, däremot av olika sorters fordon, t ex personbil och lastbil. Dessa två klasser består i sin tur av konkreta bilar.

Attribut
Attribut används för att beskriva egenskaper hos objekt i en klass.

Attribut	Ett attribut är ett datavärde som hålls av ett objekt i en klass.

Olika objektinstanser i en klass kan ha lika eller olika datavärden i ett av klassens attribut. Varje attributnamn är unikt inom klassen.

Operationer

Operation	En operation är en funktion eller transformation av ett datavärde som kan appliceras på ett objekt i en klass.

Samma operation kan gälla för flera olika klasser. Denna typ av operation kallas polymorf, vilket innebär att den antar olika former utifrån den klass den tillhör. Realiseringen av en operation kallas metod. En metod kan bara appliceras på en klass.

När en operation realiseras som flera olika metoder är det viktigt att alla metoder har samma signatur. Med signatur menas det antal direktiv, av olika typer, som måste anges när en begäran görs om att en operation ska utföras. Beteendet hos alla metoder, för en viss operation, ska ge samma resultat.

Under en modellering kan det vara bra att skilja mellan operationer som har bieffekter och operationer som endast räknar fram ett värde utan att modifiera objektets tillstånd. Operationer utan bieffekt kallas förfrågningar (query). Dessa förfrågningar kan betraktas som framräknade attribut (derived attribute).

Operationer med bieffekter tas upp i avsnitt 4.3.4.

Länkar och associationer

Länk	En länk är en fysisk eller konceptuell koppling mellan objektinstanser.
Association	En association beskriver en grupp länkar med gemensam struktur och gemensam semantik.

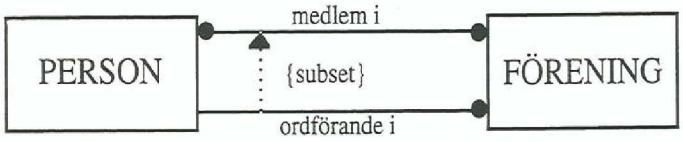
En länk knyter objektinstanser till varandra. En association knyter klasser till varandra. En association beskriver ett antal länkar på samma sätt som en klass beskriver ett antal objekt. Det kan t ex finnas en association mellan klasserna företag och person. Länkarna i denna association går mellan instanserna i dessa klasser, t ex Kalle jobbar på SISU.

Associationer är dubbelriktade. Namnet på associationen läses från det ena hållet. Läsriktning kan dock ändras och samma semantiska betydelse kvarstår: Kalle är anställd hos SISU och SISU anställer Kalle.

En länk visar relationen mellan två eller flera objekt. All kommunikation mellan objekt ska modelleras som länkar.

Associationer knyter två eller flera klasser till varandra. Det stora flertalet av alla associationer knyter två klasser till varandra. Vid analysarbete ska man sträva efter binära kopplingar. Trinärkopplingar, och kopplingar med fler än tre deltagare, är ofta mycket svåra att förstå, modellera och realisera. Dessa ska om möjligt undvikas.

Generella restriktioner
Generella restriktioner används för att göra den semantiska innebörden i modellen tydligare. Restriktioner beskrivs i naturligt språk eller i ekvationer. I modellen visas restriktioner på associationer med en streckad linje mellan associationerna.



Restriktion på en association

Exempel: En förening består av många medlemmar. En person kan vara medlem i många föreningar. En förenings ordförande måste vara medlem i föreningen.

Kardinalitet
Kardinalitet (multiplicity) specificerar hur många instanser av en klass som kan relateras till en instans i en annan klass. Antalet deltagande instanser i en association begränsas utifrån problemområdet.

Det viktigaste när kardinalitet definieras är att tänka på distinktionen mellan förhållanden som är "ett-till-många" och sådana som är "många-till-många". Om det möjliga antalet relationer underskattas leder detta ofta till kostsamma ändringar senare i design- och realisationsarbetet.

Objekt i en klass betraktas som en mängd. I OMT finns möjlighet att visa att objekt i en klass är ordnade utifrån ett kriterium, d v s en lista. Detta görs genom att ange (orderd) vid associationen mellan klasserna.

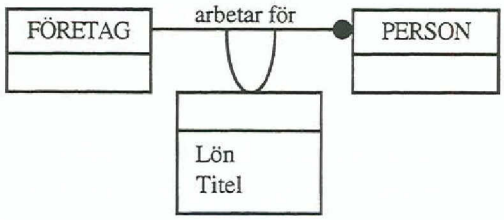
Länkattribut
Attribut är en egenskap hos en klass. Ett länkattribut är en egenskap som tillhör en länk mellan två, eller flera, objekt.

Länkattribut	En egenskap hos länken i en association.

Länkattribut används när en egenskap inte hör till någotdera av objekten som ingår i länken. Denna situation inträffar främst vid många-till-många-relationer.

Exempel: Ett företag anställer många personer. En person kan vara anställd i många företag. För att identifiera en persons lön i ett visst företag, knyts attributet lön till länken mellan företag och person.

Länkattribut kan också användas när en klass är associerad till sig själv. En person är chef över en annan person.



Länkattribut

Association som klass

Ibland kan det vara användbart att modellera en association som en klass. Varje länk blir då ett objekt i klassen. Associationsklassen kan på samma sätt som övriga klasser ha egna attribut och operationer. Denna typ av klasser kommer till användning framförallt när länkar kan delta i associationer med andra objekt eller när länkar i sig är föremål för operationer.

Länkattribut modellerat med klass

Rollnamn

I ändarna av en association kan rollnamn anges. Detta anger objektens förhållande till det/de andra objektet/en i associationen. Exempel på rollnamn kan vara chef för objektet person.

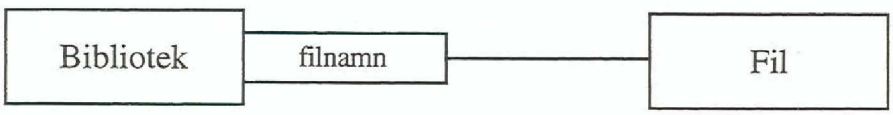
Rollnamn	Ett rollnamn är ett namn som unikt identifierar en ände av en association.

Rollnamn är mycket användbara när två objekt i samma klass associeras till varandra. Rollnamn kan vara chef – underställd.

Qualifier (bestämning)

"Qualifier" skiljer mellan de olika objekten i en klass vid många-sidan i associationen. Med hjälp av denna notation ökar den semantiska korrektheten och förståelsen för modellen. Objekt + Qualifier ger det associerade objektet.

Qualifier	Qualifier är en typ av attribut som minskar antalet deltagande objekt i en association.



Om ett filnamn och ett bibliotek kombineras, pekas en specifik fil ut. Detta reducerar ett-till-många-relationen till en ett-till-ett-relation.

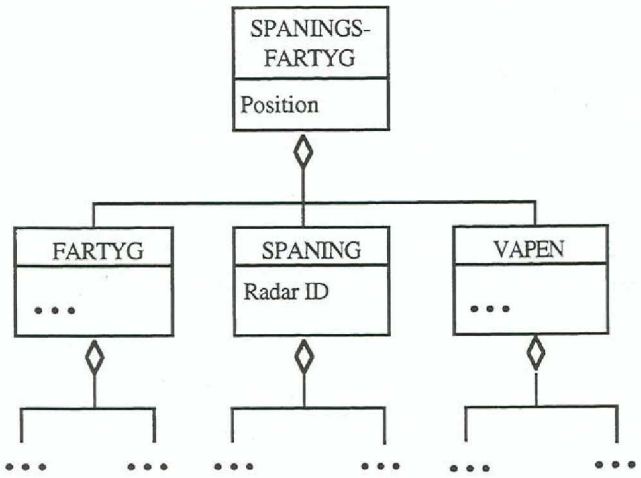
Aggregering

Aggregering är en semantisk metod för att se ett objekt som en enhet, trots att det består av flera andra objekt. Aggregering relaterar objektinstanser till varandra, inte klasser. Aggregering är således en objektstruktur.

Aggregering	Aggregering är en hel-delar- eller en del-av-relation, i vilken objekten representerar komponenter av ett objekt som representerar helheten.

Den mest typiska egenskapen för aggregering är dess transitivitet. Med detta menas att om A är en del av B, och B är en del av C, så är A en del av C. Aggregering är dessutom asymmetrisk, d v s om A är en del av B, så är inte B en del av A.

Egenskaper som tillhör helheten kan ärvas av delarna. Om inte delarna har någon gemensam egenskap med helheten bör aggregering inte användas. I dessa fall bör associationer användas för att koppla samman objekten.



Exempel

På ett spaningsfartyg finns det en radar. Fartyget har en viss position angiven i longitud och lattitud. Radarn ombord ärver vissa av fartygets attribut, bl a dess position.

Generalisering

Generalisering och arv är kraftfulla abstraktioner för att dela gemensamma attribut och operationer mellan klasser, samtidigt som varje klass unika egenskaper bevaras.

Generalisering	Generalisering är en relation mellan en klass och en eller flera förfinade klasser.

Klassen som förfinas kallas superklass och de förfinade klasserna subklasser. Gemensamma attribut och operationer för subklasserna återfinns i superklassen. Varje subklass sägs ärva egenskaperna hos superklassen. Generalisering kallas ibland "är-en-relation" ("is-a" relation).

Generalisering och arv är transitiva genom ett godtyckligt antal nivåer i en hierarkisk struktur.

I den hierarkiska strukturen används begrepp som "anfader" och "avkomma" för att visa den inbördes relationen mellan klasserna. En instans i en subklass är samtidigt en instans i alla sina anfäders klasser. Objektet innehåller värden för sina attribut och för alla ärvda attribut från anfäderna.

Ett objekt innehåller inte bara ärvda attribut och operationer, det kan också utöka sin uppsättningen genom att lägga till egna attribut och operationer.

Arv
Begreppen arv, generalisering och specialisering refererar alla till samma idé. Generalisering används för relationer mellan klasser. Arv är en mekanism för att dela attribut och operationer, genom att använda generaliseringsrelationer.

Generalisering och specialisering är olika synvinklar av samma relation. Generalisering innebär att flera subklasser generaliseras till en superklass. Specialisering innebär att en superklass specialiseras i flera subklasser.

Multipla arv
En klass kan tillåtas ha flera superklasser ("join class"). Detta innebär att klassen ärver egenskaper från två eller flera superklasser, vilket medför att hierarkin blir ett nätverk.

Fördelen med multipla arv är att det känns naturligt för människan att tänka så. Nackdelen är att modellerna ofta blir mer komplexa och realiseringen mer komplicerad.

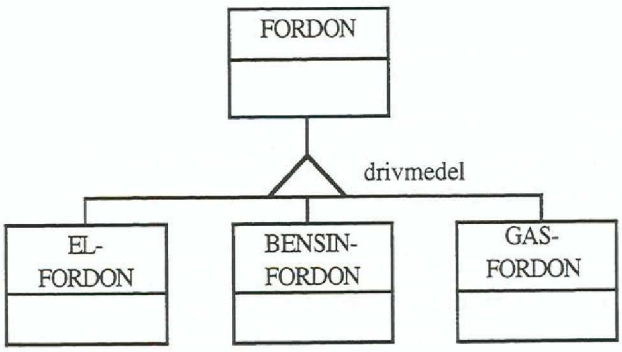
Join class	En klass med flera superklass.

När multipla arv introduceras kan konflikter uppstå om två superklasser använder samma attribut- eller operationsnamn.

Discriminator
"Discriminator" är enkelt uttryckt ett kriterium utifrån vilket man specialiserar klasser. T ex kan klassen fordon specialiseras utifrån kriteriet drivmedel. Detta ger specialiseringarna eldriven, bensindriven och gasdriven.

Discriminator	En discriminator är ett attribut av förteckningstyp[1], som visar vilken egenskap hos ett objekt som abstraherar en speciell generaliserings-relation.

[1] Förteckningstyp är en översättning av "Enumerationtype" . Förteckningstyp är ett attribut som endast kan innehålla ett av en mängd fördefinierade värden. I modelleringshandboken kallas förteckning värdeförråd (domän).



Discriminator är endast ett kriterium som ligger till grund för generaliseringen. Värdet ger ett ett-till-ett-förhållande till subklassen i generaliseringen.

Gruppering
Med hjälp av att gruppera en modell i moduler kan en stor modell delas upp i hanterbara delar. Det blir möjligt att fånga olika aspekter av systemet. En objektmodell innehåller en eller flera moduler.

Modul	En modul är en logisk konstruktion för att gruppera klasser, associationer och generaliseringar.

Man ska sträva efter att skilja de olika modulerna åt. Genom att begränsa kopplingar mellan moduler blir systemet mer stabilt och tåligare mot förändringar.

Med hjälp av en s k arkfunktion (sheet) kan man fysiskt dela upp en modell på flera blad. Ark är ingen logisk konstruktion, utan ett sätt att underlätta överblicken och förståelsen för systemet. Generaliseringar och aggregeringar ska i möjligaste mån placeras på samma ark. Det ska aldrig vara mer än en modul per ark.

En klass kan finnas med på flera ark, vilket visar att innehållen på respektive ark är beroende av varandra.

4.3.4 Dynamisk modellering
Ett system är lättast att förstå om man granskar den statiska strukturen. Klasser, attribut, relationer och operationer visar den statiska strukturen i systemet, vilket beskrivs i objektmodellen. Avbildningen sker vid ett givet ögonblick.

Att modellera de temporära relationerna, inom och mellan objekt, är svårt. Den dynamiska modellen innehåller dessa temporära relationer. Modellen beskriver hur externa stimuli initierar operationer och sekvensierar dem. Beskrivningen visar inte vad operationerna gör, vad de opererar på eller hur de realiseras.

De huvudsakliga begreppen i den dynamiska modellen är händelser (event) och tillstånd (state). Dessa visas i tillståndsdiagram. Den dynamiska modellen består av flera tillståndsdiagram, ett för varje klass, som tillsammans visar aktivitetsmönstret för systemet.

Händelse

Händelse Någonting som inträffar vid en viss tidpunkt.

Varje händelse är en unik företeelse. Dessa företeelser kan grupperas i klasser, händelseklasser, om de har en gemensam struktur och ett gemensamt beteende. Strukturen är hierarkisk. Händelser är ögonblickliga, d v s har ingen varaktighet.

Händelser som inte är relaterade till varandra kan inträffa samtidigt, d v s de har ingen inverkan på varandra. En händelse är en envägssändning av information från ett objekt till ett annat. Det objekt som sänder kan begära ett svar, men detta svar betraktas som en fristående händelse. Om svaret skickas eller inte, avgörs av det mottagande objektet.

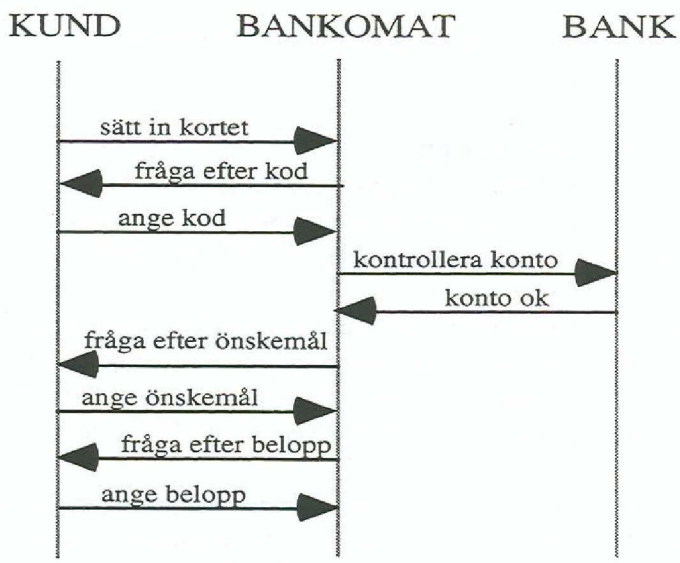
Vissa händelser är endast signaler från ett objekt till ett annat. Det vanligaste är dock att händelseklasser har attribut, i vilka data lagras om den information som överförs.

Scenario

Ett scenario är en sekvens av händelser som inträffar under en specifik körning av systemet. Innehållet i ett scenario kan variera mycket. Det kan inkludera alla händelser i systemet eller endast dem som genereras av ett visst objekt.

Event trace

För att kartlägga hur händelser sänder information mellan objekt kan man konstruera diagram för "event trace" (händelsespårning). Diagrammet visar varje objekt som en vertikal linje och varje händelse som en horisontell pil. Pilen går från det sändande objektet till det mottagande. Tid passerar från topp till botten. Ökningen av tiden är inte proportionell mot avståndet mellan händelserna.



Event trace för bankomatsystem

Tillstånd
Värdena i ett objekts attribut och dess länkar bildar objektets tillstånd. Tillståndet påverkar objektets beteende och specificerar dess respons vid en viss händelse.

Tillstånd	Ett tillstånd är en abstraktion av de attributvärden och länkar som ett objekt har.

Responsen vid en händelse varierar beroende på de värden som attributen har, men är exakt samma för alla objekt som befinner sig i samma tillstånd. Objektets respons vid en händelse innehåller en handling (action) eller en förändring av tillstånd.

Tillståndet existerar i intervallen mellan två händelser. Händelsen representerar ett givet ögonblick som separerar två tillstånd. Tillstånd är varaktiga, händelser ögonblickliga.

När tillstånd definieras, bortser man från de attribut som inte påverkar beteendet hos objektet. Därefter sammanförs alla kombinationer av attributvärden och länkar i ett tillstånd som har samma respons vid en händelse.

Tillståndsdiagram
När en händelse inträffar beror nästa tillstånd på det nuvarande tillståndet såväl som på händelsen. Tillståndsdiagram visar denna relation.

Tillståndsdiagram	Ett diagram i vilket händelser relateras till tillstånd.

Om ett tillstånd ändras till följd av en händelse kallas detta "transmission". Ett tillståndsdiagram är en graf vars noder är tillstånd och pilar är transmissioner som etiketteras med händelsenamn. Etiketten på pilen är namnet på händelsen som orsakar transmissionen. Alla transmissioner utgår från att ett tillstånd måste motsvara en händelse.

Ett tillståndsdiagram beskriver beteendet för en enskild klass. Eftersom alla instanser i en klass kan utföra samma beteende, delar de samma tillståndsdiagram. Varje objekt innehåller sina egna värden, vilket innebär att varje objekt befinner sig i ett eget tillstånd. Objekt kan vara oberoende av andra objekt och kan därför utvecklas och förändras i egen takt.

Den dynamiska modellen är en samling tillståndsdiagram som interagerar med varandra via delade händelser.

Villkor
Ett tillstånd kan definieras som ett villkor, d v s en funktion som antingen är sann eller falsk. En lampa är antingen tänd eller släckt. Ett villkor är giltigt inom en viss tidsintervall. Det är viktigt att skilja händelser från villkor. Händelser har ingen varaktighet. Villkor existerar så länge det är sant.

Villkor kan användas som vakter. En vaktad transmission inträffar först när händelsen inträffar och villkoret är sant.

Operationer
Tillståndsdiagrammet beskriver när en viss operation kan utföras. Operationer utförs antingen när ett objekt befinner sig i ett visst tillstånd eller när en viss händelse inträffar. Operationer knyts till klasser i objektmodellen och deras beräkningar specificeras i funktionsmodellen.

Som nämndes ovan kan en operation antingen utföras när ett objekt befinner sig i ett visst tillstånd eller när objektet utsätts för en viss händelse. För att ytterligare definiera olika operationer, delas de in i *aktiviteter* och *handlingar*.

Aktiviteter
Aktiviteter är operationer som består av flera instruktioner. Dessa operationer pågår så länge objektet befinner sig i ett visst tillstånd. När objektet ändrar tillstånd avbryts operationen.

Aktivitet En operation som består av flera instruktioner. Aktiviteten utförs i sekvens och tar tid att utföra. Aktiviteten är knuten till ett tillstånd.

Handlingar
Handlingar är operationer som utförs omedelbart. Operationer består endast av en instruktion, d v s de kan betraktas som atomära. Instruktionen utförs antingen helt eller inte alls. Halva handlingen kan alltså inte utföras.

Handlingar utförs när en viss händelse inträffar. Händelser är det som skiljer två tillstånd åt. Handlingen kan t ex utföra operationer som att lagra värden i attribut. Handlingar som anges i den dynamiska modellen kan vara både verkliga och artificiella. De verkliga händelserna är sådana som inträffar i problemområdet och påverkar systemet, de artificiella är sådana som används för att visa datoroperationer. I den dynamiska modellen modelleras således både verkligheten och informationssystemet.

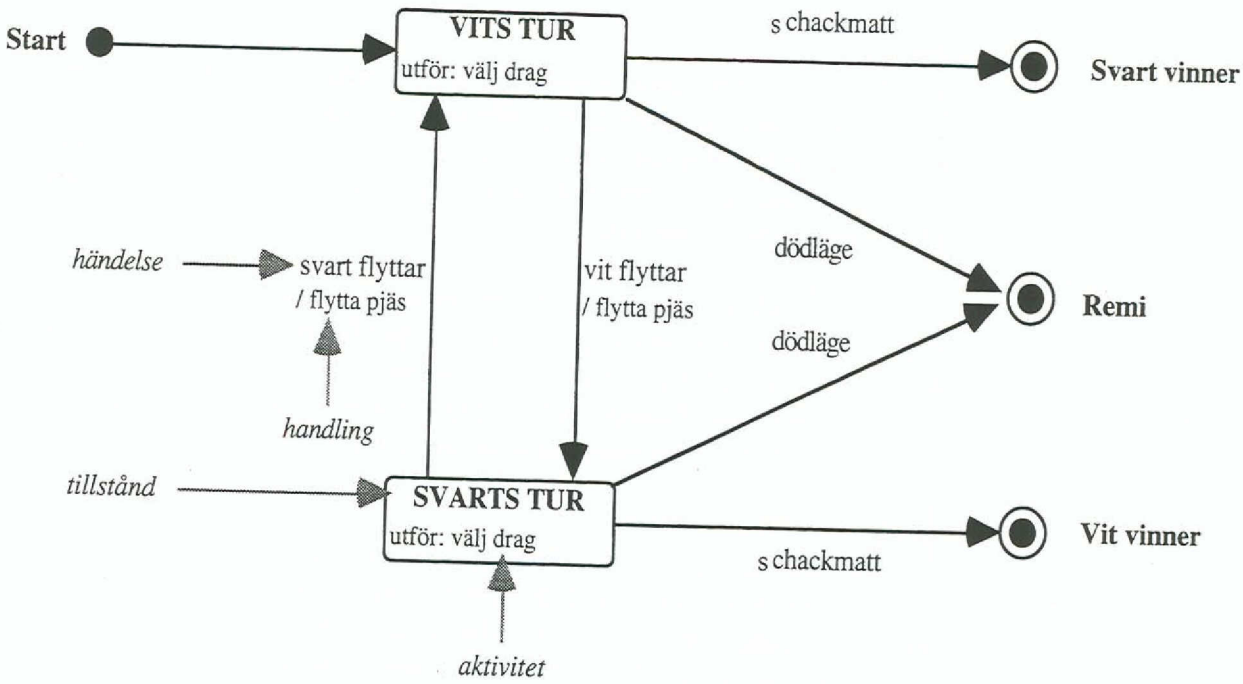
Exempel
Tillstånd
Schackpartiet befinner sig antingen i tillståndet svarts tur att spela eller vits tur. Detta innebär att antingen svart eller vit spelare ska göra nästa drag.

Aktivitet
Så länge partiet befinner sig i tillståndet "vits tur" utförs aktiviteten "välj drag" för de vita pjäserna. Denna operation pågår under en tidsintervall. Operationen består av att olika förflyttningar analyseras och ställs mot varandra.

Händelse
När händelsen "vit flyttar" inträffar, ändrar partiet tillstånd. Från att ha befunnit sig i tillståndet "vits tur" ändrar nu spelet tillstånd till "svarts tur".

Handling
När spelaren har bestämt sig för vilken pjäs som ska flyttas ändras tillståndet. Som svar på händelsen utförs handlingen "flytta pjäs", t ex kungen från C1 till C2.

Handlingar är atomära, d v s kungen står antingen på C1 eller C2, aldrig mitt emellan.



Tillståndsdiagram med handlingar och aktiviteter för ett schackparti

4.3.5 Funktionsmodellering

Funktionsmodellen beskriver beräkningar i systemet. I modellen specificeras vad som händer i systemet. Den dynamiska modellen specificerar när det händer och objektmodellen specificerar vilka objekt som berörs.

Den dynamiska modellen visar hur indata bearbetas och beräknas till utdata, utan att ange i vilken ordning värdena beräknas. Funktionsmodellen består av olika dataflödesscheman, vilka visar dataflödena från externa indata via operationer och intern datalagring, till externa utdata. I dynamiska modellen visas också restriktioner för attributvärden i objektmodellen.

I funktionsmodellen specificeras resultatet av beräkningar. Hur eller när de utförs specificeras inte. I modellen specificeras syftet med de operationer som anges i objektmodellen. Den visar även när och hur händelser inträffar som återfinns i den dynamiska modellen.

Dataflödesdiagram

Dataflödesdiagrammet visar de funktionella relationerna mellan indata och utdata samt interna datalager och de värden som beräknas av systemet. Ett dataflödesdiagram är en graf som visar flöden av datavärden, från deras ursprung i objekt, via processer som förändrar dem, till deras destinationsobjekt. I diagrammet visas ej kontrollfunktioner så som tidpunkten då de körs eller val mellan möjliga beräkningar. Dessa aspekter visas i den dynamiska modellen.

Ett dataflödesdiagram innehåller *processer* som förändrar data, *dataflöden* som flyttar data, *aktörer* som producerar och konsumerar data samt *datalager* som lagrar data.

Process
Den lägsta nivån av processer är rena funktioner utan bieffekter. Dessa funktioner kan t ex summera två tal. Ett helt dataflödesdiagram är en högnivå-process. En process kan ha bieffekter om den innehåller komponenter som inte är funktioner, så som datalager eller externa objekt. Operationer som har bieffekter ändrar värden som lagras i attributen.

I funktionsmodellen specificeras inte resultatet av processer som har bieffekter, istället anges aktuella valmöjligheter. Vilken specifik väg som väljs anges inte. Resultatet av en sådan process beror på systemets beteende, specificerat i den dynamiska modellen. Exempel på händelser som inte är funktioner är att läsa från eller skriva till en fil. Processer realiseras som metoder.

Dataflöden
Ett dataflöde representerar ett datavärde någonstans i en beräkning. Datavärden ändras inte av dataflöden. Dataflöden som innehåller aggregerade värden kan, om det behövs, delas upp i sina element. Vice versa, att flera element aggregeras till ett dataflöde, är också tillåtet. Dataflöden behöver inte ha någon motsvarighet i verkligheten eller i problemområdet.

Aktörer
En aktör är ett aktivt objekt som driver dataflödesdiagrammet genom att producera eller konsumera datavärden. Aktörer knyts till indata och/eller utdata i diagrammet. Exempel på aktörer kan vara användare, enheter som styrs av systemet eller utrustning som lämnar data till systemet.

Datalager
Ett datalager är ett passivt objekt som lagrar data för senare åtkomst. Datalager skapar inte nya data, utan svarar endast för lagring och åtkomst. Datalager gör det möjligt att komma åt datavärden i annan ordning än den de skapats i. Både aktörer och datalager är objekt i objektmodellen. Man gör skillnad mellan dem i funktions-modellen eftersom deras beteende och användningsområde ofta är olika.

Operationer
Processer i dataflödesdiagrammet specificeras som operationer. Varje atomär process är en operation. Operationer kan bland annat specificeras som matematiska funktioner, ekvationer, beslutstabeller, pseudokod eller som naturligt språk.

Operationer kan delas upp i enkla och komplicerade operationer. En enkel operation är åtkomst ("access") av data. Den behöver inte listas eller specificeras i analysen, utan anses vara implicit. De komplicerade operationerna är *förfrågning* ("query"), *handling* (action) och *aktivitet* ("activity").

Förfrågning är en operation utan bieffekt, d v s den påverkar inte tillståndet hos något objekt. Förfrågningar är rena funktioner och kan därmed betraktas som framräknade attribut. Framräknade attribut skapar inte data.

Handling är en transformation som har bieffekter. Den ändrar tillståndet hos objektet den opererar på. En handling är atomär, d v s handlingen anses ur logiskt perspektiv som ögonblicklig. Den tar ingen tid att utföra. Syftet med en handling kan vara att uppdatera ett objekts attribut eller länk. Handlingar kan beskrivas på många sätt.

Algoritmer som specificerar en handling syftar till att definiera resultatet av operationen, inte att realiseras. Dessa algoritmer kan uttryckas som kod, pseudokod eller naturligt språk. En handling i funktionsmodellen motsvarar en process i dataflödesdiagrammet.

En *aktivitet* är en operation som utförs av eller på ett objekt. Aktiviteter består av flera delmoment och tar således tid att utföra. Aktiviteter är endast relevanta för aktörer, d v s objekt som skapar operationer på egen hand. Detaljerna i en aktivitet specificeras i den dynamiska modellen såväl som i funktionsmodellen.

Restriktioner
Restriktioner kan införas i modellerna. Objektrestriktioner specificerar att vissa objekt är beroende, helt eller delvis, av andra objekt. Dynamiska restriktioner specificerar relationer mellan tillstånd och händelser för olika objekt. Funktionsrestriktioner specificerar restriktioner för vissa operationer.

4.3.6 Kopplingar mellan modellerna

De olika modellerna beskriver olika aspekter av samma system. För att visa systemet som en helhet finns kopplingar mellan de olika företeelserna i modellerna. En process i funktionsmodellen är en operation i objektmodellen. I den dynamiska modellen visas i vilken ordning de olika operationerna kan utföras. Aktörer i funktionsmodellen representerar explicita objekt i objektmodellen. Dataflöden, till och från aktörer, visar att operationer utförs på eller av objektet. Datalager är objekt, eller attribut i objekt, i objektmodellen. Flöde av data till ett datalager visar en uppdateringsoperation. Flöde av data från ett datalager symboliserar en förfrågning, d v s en operation utan bieffekter. Dataflöden i dataflödesdiagrammet symboliserar attributvärden i objektmodellen.

4.3.7 Arbetsgång

Analysen är det första arbetssteget i OMT-metoden. Syftet med analysen är främst att få djupare förståelse för problemområdet. Detta görs genom att skapa en koncis, förståelig och korrekt modell av verkligheten.

Analysmodellen fokuserar tre aspekter i verkligheten: systemets statiska struktur, sekvensieringen av interaktionen (i vilken ordning människa och dator kommunicerar) samt transformationen av data. Dessa tre aspekter fångas i respektive delmodell: objektmodellen, dynamiska modellen och funktionsmodellen.

Analysarbetet kan inte utföras i strikt sekventsiell ordning. Arbetet måste ske iterativt. Först skapas en del av modellen, sedan utökas denna och därefter går man tillbaka och skapar en annan del av modellen, o s v.

Det första steget i analysarbetet är att göra en objektmodell. Objektmodellen innehåller den statiska strukturen i systemet. I modellen visas klasser och objekt. Dessutom beskrivs klassernas egenskaper och relationer till varandra. Objekt kan grupperas i aggregeringar, d v s ett objekt består av flera andra objekt, eller generaliseringar, d v s gemensamma egenskaper samlas i en superklass från vilken subklasser ärver egenskaper.

Arbetssteg för att skapa en objektmodell:
- Identifiera klasser och objekt.
- Identifiera associationer mellan klasser (inklusive aggregeringar).
- Identifiera attribut och länkar hos objekt.
- Organisera och förenkla klasserna genom att använda arv (generaliseringar).

Det andra steget i analysarbetet består av att göra en dynamisk modell. Den dynamiska modellen visar de tidsberoende beteendena i systemet.

Man börjar med att förbereda scenarier för typiska dialoger. Även om de scenarier som upprättas inte täcker alla tänkbara situationer är det ett bra sätt att inventera systemets skyldigheter.

Därefter söker man händelser som inträffar i dessa scenarier. Det är ofta bäst att inventera händelserna först och därefter knyta dem till specifika klasser. Händelserna och tillstånden organiseras sedan för de olika klasserna i tillståndsdiagram.

Arbetssteg för att skapa en dynamisk modell;
- Förbered scenarier för typiska interaktionssekvenser
- Identifiera händelser hos olika klasser
- Skapa en händelsespårning (*event trace*) för varje scenario
- Skapa ett tillståndsdiagram
- Jämför händelser mellan olika klasser för att verifiera konsistensen

Det tredje, och sista, steget i analysen är att göra funktionsmodellen. Funktionsmodellen visar hur värden beräknas. I modellen visas vilka värden som är beroende av andra värden, och vilka funktioner som knyter samman dem.

Processer i ett dataflödesdiagram motsvarar handlingar och aktiviteter i tillståndsdiagrammet. Flöden i dataflödesdiagrammet motsvarar objekt, eller attributvärden, i objektmodellen. Det är därför effektivare att skapa funktionsmodellen sist, efter objekt- och de dynamiska modellerna.

Arbetssteg för att skapa en dynamisk modell:
- Identifiera in- och utdata.
- Skapa dataflödesdiagram.
- Identifiera restriktioner för operationer, attributvärden och objekt.

4.4 ObjectOry

4.4.1 Bakgrund

ObjectOry har utvecklats av Ivar Jacobsson m fl. Jacobssons vision är att skapa en metod (produkt) för att förbättra systemutvecklingsprocessen, mot en "industriell systemutvecklingsprocess". Utifrån denna vision har produkten namngivits; ObjectOry står för "Object Factory".

Jacobssons vision är att skapa en metod för att utveckla dagens och morgondagens system snabbare, billigare, med bättre kontroll och högre kvalitet. Medlet för att nå dit är att utnyttja koncept som objektorientering, återanvändning och Case-hjälpmedel. Dessa koncept kan inte själva lösa problemen, utan måste sättas in i ett sammanhang. För att lyckas krävs att man har en helhetssyn på systemutvecklingen.

För att se dessa koncept som en helhet används begreppet "industriell systemutveckling", vilket

- innebär att man har en helhetssyn på utvecklingsprocessen. På så vis skapas en ram för integrering av metoder, teknologi och utvecklingsverktyg.

- är ett sätt att lösa problemen vid systemutveckling med objektorientering som utgångspunkt. Dock utgör objektorientering endast en del av lösningen.

- betyder, i generella termer, att man ska arbeta på ett moget och professionellt sätt, liknande det som företag använder i branscher som är mer mogna än mjukvaruindustrin.

För att systemutvecklingsteorin ska kunna kallas industriell, måste vissa krav uppfyllas. Industriella system måste innehålla en externt definierad standardarkitektur. Samtidigt ska de förbli "öppna", d v s de ska vara lätta att integrera med system från andra oberoende leverantörer. Systemen måste utvecklas enligt användarnas krav, med hänsyn till användardialog och användargränssnitt.

Industriell systemutveckling innehåller tekniker som, jämfört med traditionella systemutvecklingstekniker, är mer inriktade mot de inledande arbetsstegen i utvecklingsprocessen. De medför mer arbete kring analysarbete och prototyper.

Objektorientering är en ny och viktig grundläggande teknik. Den erbjuder systemutvecklare möjlighet att utveckla stabila och användarvänliga system på ett effektivare sätt än de traditionella utvecklingsteknikerna.

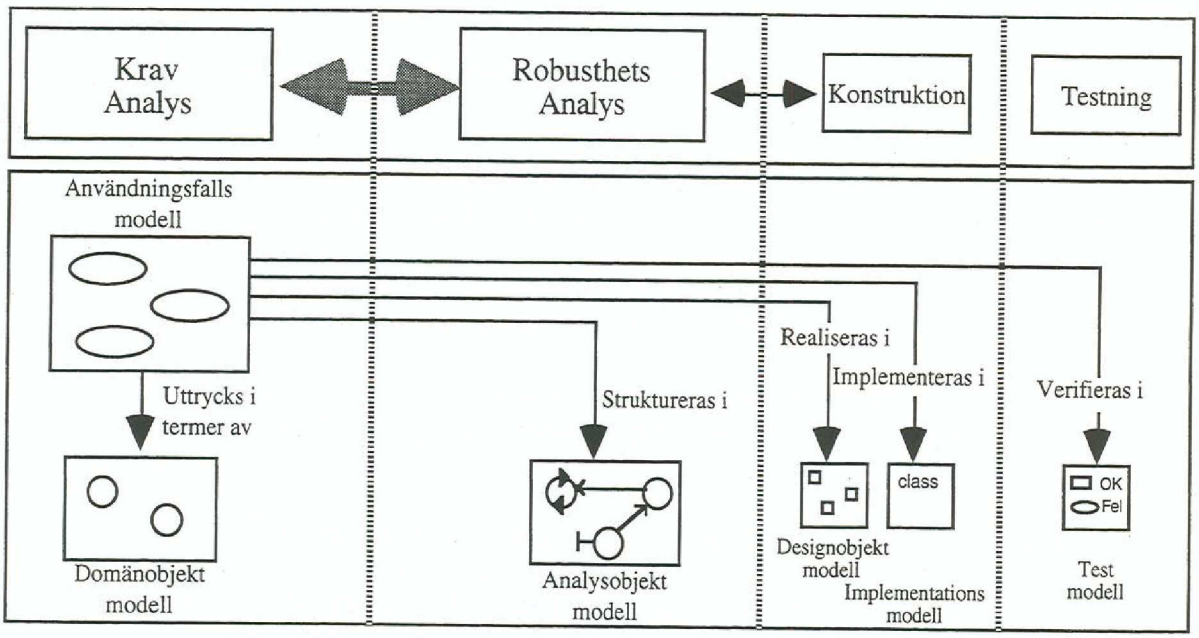
ObjectOry har utvecklats under de senaste 20 åren. Till grund för metoden ligger Jacobssons avhandling "Concepts for modeling large realtime systems" som berör telekommunikation. Därefter har flera år ägnats åt att anpassa metoden till det objektorienterade paradigmet och att generalisera den så att metoden även passar andra områden än telekommunikation. Metoden har utökats för att omfatta hela processen vid systemutveckling, från analys till testning.

4.4.2 Fördelar

ObjectOry omfattar hela livscykeln för ett system. Alla utvecklingsfaser stöds av metoden, från kravspecifikation, via analys och design till test av det färdiga systemet. Med hjälp av ObjectOry binds faserna samman i en jämn utvecklingskedja.

ObjectOry baseras på den grundläggande arkitekturen av objekttyper och relationer dem emellan. Dessa objekttyper används för att beskriva, definiera och utveckla systemet.

Resultatet vid användning av ObjectOry är ett antal verbala modeller som, var och en, representerar en aspekt av systemet. Modellerna skapas i, och representerar, olika faser av utvecklingsarbetet: *analys*, *konstruktion* och *testning*.



Den introducerande *analysfasen* består av *kravanalys* och *robusthetsanalys*. *Kravanalysen* resulterar i en kravmodell som består av tre delar: användningsfallsmodell (use-case model), domänobjektmodell och gränssnittsbeskrivning. Användningsfallen är beskrivningar av de funktioner som systemet ska utföra. Dessa användningsfall definieras ur användarens synvinkel. Domänobjektmodellen innehåller objekt som identifierats i problemområdet. Denna modell görs för att få bättre förståelse för problemområdet. Gränssnitten beskrivs explicit, med hjälp av skärmbilder. Detta görs på ett tidigt stadium för att ge användaren en bild av hur systemet kommer att fungera.

Robusthetsanalysen resulterar i en analysmodell. Analysmodellen definierar systemets struktur och beteende. Denna modell görs som en idealisering av systemet, utan att ta hänsyn till realisationsaspekter. Med analysmodellen vill man skapa en robust modell som tål ändringar under systemets livstid. Detta görs genom att dela upp domänobjekten i tre sorters objekt; entitets-, gränssnitts- och kontrollobjekt. Genom att använda olika typer av objekt påverkar eventuella ändringar endast en begränsad del av systemet.

I denna fas är det lämpligt att bygga prototyper, vilket främjar kommunikationen mellan analytiker och problemområdesexperter.

Efter dessa analyser följer *konstruktionsfasen* som består av design och realisering av analysmodellen.

Den sista fasen i ObjectOry är *testfasen*. Denna fas verifierar att systemet verkligen realiserar det som definierats i användningsfallen. Testfasen delas upp i flera mindre delar där de olika delarna i systemet testas.

4.4.3 Användningsfallsmodell

I detta avsnitt förklaras de begrepp som används inom användningsfallsmodellen. Dessa begrepp är framför allt hämtade från definitionerna i processboken. Vid de tillfällen definitioner inte förekommit har definitioner hämtats ur textens sammanhang.

Användningsfallsmodellen syftar till att definiera vad systemet ska göra och hur systemet interagerar med användaren.

När systemets funktioner definieras ska hänsyn tas till vem som använder systemet. Användningsfallet ska visa vad användarna kräver av systemet och definiera väsentliga begrepp som används i det. Att beskriva varje användningsfall i detalj betyder att klargöra hur systemet interagerar med användarna och vad systemet utför i respektive användningsfall.

En användningsfallsmodell innehåller ett antal fall och de aktörer som använder respektive användningsfall.

Användningsfallsmodellens ändamål:

- Komma överens med kunden och användaren om vad systemet ska göra.
- Ge systemutvecklarna bättre förståelse för de krav som ställs på systemet.
- Utveckla en modell som beskriver vad systemet ska göra.
- Avgränsa systemet.

Användningsfall

Den främsta anledningen till att modellera användningsfall är att beskriva vad systemet ska göra, d v s på vilket sätt användaren vill utnyttja systemet.

Användningsfall	Ett användningsfall är ett flöde av händelser i systemet, inklusive interaktionen med aktören. Flödet av händelser initieras av en aktör. Ett användningsfall har ett namn.

För att fullständigt förstå varför systemet ska utvecklas måste användarna, de som ska utnyttja systemet, identifieras. Användare visas som aktörer i användningsfallet.

Funktionerna som utförs av systemet definieras av ett antal användningsfall. Användningsfallet visar de funktioner systemet erbjuder aktörerna och dokumenteras i löpande text.

Användningsfallen har en central roll i ObjectOry eftersom de bildar den röda tråden genom hela systemets utvecklingscykel. Samma användningsfall används i analys, design och vid testning.

Det kan finnas oändligt många användningsfall i ett system. Beroende på den aktuella händelsen väljer användningsfallet handlingsalternativ. Händelser som avgör vilket handlingsalternativ som väljs kan vara:

- indata från en aktör (En aktör kan välja mellan många olika operationer.)
- kontroll av värden (Om villkoret är sant ska användningsfallet utföra en viss operation, om det är falskt en annan.)

Instanser av användningsfall kan inträffa samtidigt. Dessa sägs vara konkurrerande. I analysfasen behöver man inte ta ställning till om konkurrerande (samtidiga) operationer ska tillåtas eller inte. I användningsfallens modellering förutsätts att olika instanser av användningsfall kan vara aktiva samtidigt.

De användningsfall som utför en hel operation som aktören efterfrågar kallas "konkreta". Konkreta användningsfall initieras av aktören och bildar ett komplett flöde av händelser i systemet. Med detta menas att en instans av ett användningsfall utför hela operationen som aktören efterfrågar.

Ett "abstrakt" användningsfall är någonting som används för att hjälpa ett konkret användningsfall att utföra sin uppgift. Det absktrakta fallet kan inte existera på egen hand, utan utgör en del av det konkreta fallet.

När ett konkret användningsfall initieras, skapas en instans av det. Denna instans kan utföra de beteenden som specificeras i de associerade "abstrakta användningsfallen", vilket innebär att inga separata instanser skapas av de "abstrakta användningsfallen". Dess beteende utgör en del av instansen för det konkreta användningsfallet.

Aktörer

En aktör är någonting som interagerar med systemet. Aktörer kan vara människor som använder systemet, människor som administrerar systemet, extern hårdvara som används av systemet eller andra system som systemet interagerar med.

Aktör	En aktör representerar någonting som interagerar med systemet, vanligtvis olika användartyper. En aktör har ett namn.

Varje typ av externt fenomen som systemet interagerar med ska visas som en aktör. För att lättare identifiera aktörer kan följande kontrollfrågor ställas:

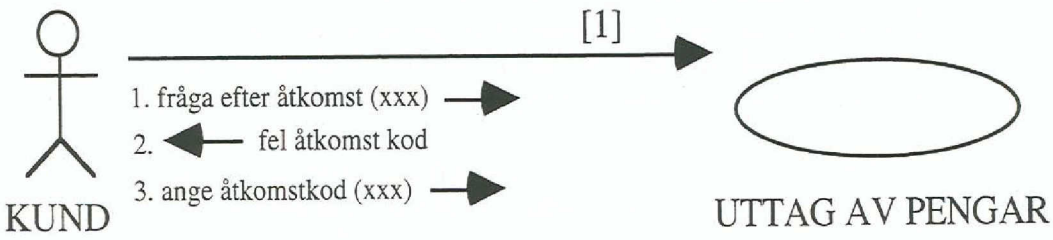
- Vilka användargrupper kräver hjälp från systemet för att utföra sina uppgifter?
- Vilka användargrupper efterfrågar systemets funktioner?
- Vilka användargrupper underhåller systemet?
- Vilka yttre enheter och system kommunicerar systemet med?

Kommunikation
Aktörer kommunicerar med systemet genom att sända och ta emot stimuli. För att fullt förstå aktörens roll måste vi veta vilka användningsfall han deltar i. Dessa relationer, mellan aktör och användningsfall, visas genom kommunikations-associationer.

Kommunikation	En kommunikation mellan en instans av ett användningsfall och en instans av en aktör indikerar att de interagerar. Riktningen på associationen är samma som stimuli-sändningen.

Om både aktören och användningsfallet skickar stimuli till varandra, måste kommunikationen anges i båda riktningarna. För att ange hur många instanser av ett användningsfall en aktör kan kommunicera med, anges kardinalitet vid associationen.

Exempel
I användningsfallet "uttag av pengar" från en bankomat, skickar kunden ett stimuli till systemet med sitt bankomatkort och sin personliga kod. Användnings-fallet kontrollerar att kort och kod stämmer överens. Om koden är felaktig, informeras kunden och ombeds ange koden igen. Användningsfallet väntar tills den riktiga koden angivits. En kund kan endast kommunicera med en instans av användningsfallet "uttag av pengar" samtidigt. Kardinaliteten för användningsfallet är [1].



Kommunikation mellan aktör och användningsfall

Arv
I en användningsfallsmodell kan flera aktörer spela samma roll i ett användningsfall. För att förenkla modellen och skapa bättre förståelse för den kan man introducera arvsfunktionen mellan aktörer.

Arv	En arvs-association från en aktörklass (ättling) till en annan aktörklass (stamfader) indikerar att ättlingen ärver den roll stamfadern kan spela i ett användningsfall.

Användaren kan spela flera olika roller i relation till systemet vilket betyder att användaren motsvarar flera olika aktörer. För att göra modellen klarare kan användaren representeras av en aktör som ärver olika roller från flera aktörer. Varje ärvd aktör representerar en användarroll i systemet.

Extension

Extension används för att strukturera användningsfallet. Detta innebär att det blir möjligt att modellera att ett användningsfall använder funktioner som specificeras i ett annat användningsfall.

Extension	En extension från användningsfall A till användningsfall B indikerar att en instans som lyder under användningsfall A vid något tillfälle slutar lyda denna klass och istället börjar lyda användningsfall Bs klass.

Alla beteenden som läggs till det ursprungliga användningsfallet utökar det. Extensions-funktionen används för att skapa förhållanden mellan konkreta och abstrakta användningsfall (se avsnitt Användningsfall s. x några sidor tidigare).

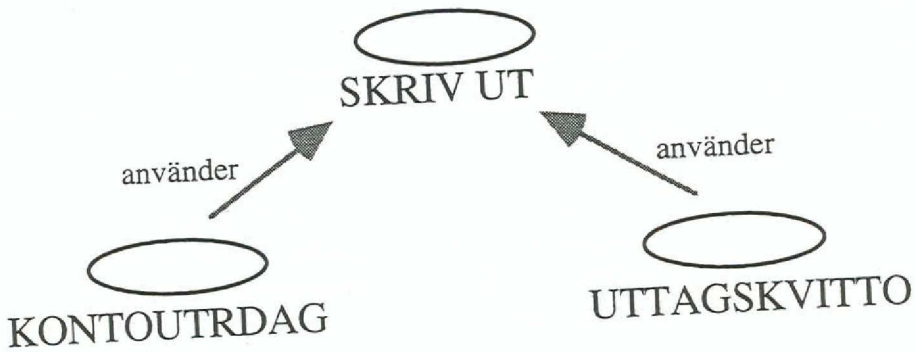
Extension kan användas av flera olika anledningar:

- Modellera valfria beteenden.
- Modellera komplicerade flöden som sällan inträffar.
- Modellera fristående subhändelser som endast körs vid särskilda tillfällen.
- Bygga modeller som visar att flera olika operationer kan utnyttjas av ett användningsfall vid en viss tidpunkt. Aktören avgör vilken operation som körs.

Använder-association

Använder-associationen är en form av arvsfunktion. Om två eller flera användningsfall innehåller ett gemensamt beteende, kan detta placeras i ett nytt användningsfall. Att användningsfall utnyttjar detta gemensamma beteende uttrycks med använder-associationen.

Använder	Använder-association från ett användningsfall A till ett användningsfall B indikerar att en instans av användningsfall A kan utföra alla beteenden beskrivna för användningsfall B.



Utskrift av kontoutdrag och uttagskvitton använder samma användningsfall, "utskrift".

4.4.4 Domänobjektmodell

I detta avsnitt definieras de begrepp som används vid domänobjektmodellering.

Domänobjektmodellen visar en konceptuell bild av verkligheten. I denna modell visas objekt som har motsvarigheter i verkligheten, deras egenskaper och operationer samt deras relationer till varandra.

I domänobjektmodellen användas begrepp som hämtas från problemområdet. Detta ökar förståelsen för tillämpningen som ska utvecklas och underlättar kommunikationen mellan analytiker och problemområdesexpert.

Domänobjektmodellen innehåller ofta begrepp som listas och används i användningsfallsmodellen. I användningsfallsmodellen kan domänobjekt identifieras. Associationer läggs till objekten för att visa relationer mellan dem. Objektens egenskaper beskrivs med hjälp av attribut. De beteenden som beskrivs i användningsfallen kan därefter knytas till de olika domänobjekten.

Målet med domänobjektmodelleringen är att komma överens med kunder och användare om vad systemet ska göra.

Domänobjekt

Domänobjekt representerar företeelser i problemområdet. Många domänobjekt kan identifieras från de begrepp som behövs för att beskriva och förstå användningsfallen.

Domänobjekt	Ett domänobjekt representerar ett "verkligt" objekt och används för att modellera systemet. En domänobjektklass har ett namn.

Domänobjekt kan vara:

- Begrepp som används i miljön som systemet ska finnas i.
- Saker och företeelser som systemet måste känna till.
- Händelser som systemet måste känna till.

Många domänobjekt motsvarar begrepp som används i kravspecifikationen. Begrepp som är lämpliga används i den dagliga verksamheten av personalen som ska utnyttja systemet.

Attribut

Ett domänobjekt har viktiga egenskaper som systemet måste känna till. Dessa egenskaper definieras som attribut i objektet. Attributen är en del av objektets definition.

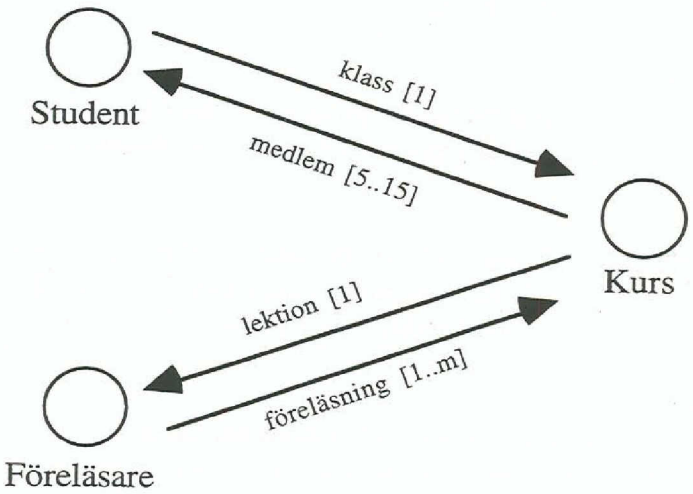
Attribut	Ett attribut för ett domänobjekt representerar en egenskap för objektet. Ett attribut har ett namn.

Attribut för domänobjekt ska hållas på en abstrakt nivå. Syftet med dessa är att göra domänobjekten mer förståeliga, inte att i detalj specificera objektets alla egenskaper.

Acquaintance-association

"Acquaintance-association" (bekantskaps-association) beskriver fenomenet att olika objekt har relationer till varandra. Dessa relationer existerar i verkligheten och ska därför även beskrivas i domänobjektmodellen.

Acquaintance-association	En acquaintance-association från domänobjekt X till domänobjekt Y indikerar att objekt X har en relation till objekt Y. Som regel är acquaintance-associationen en association mellan instanser av objekt. En acquaintance-association har ett namn och en kardinalitet. Kardinaliteten definierar hur många instanser det associerade objektet kan associera.



Acquaintance-association mellan olika domänobjekt

Aggregering

För att visa aggregeringsstrukturer använder ObjectOry "består-av-relationer". Varje relation har en kardinalitet för att visa hur många delar som krävs för att skapa en helhet.

Det är inte nödvändigt att använda består-av-relationen i domänobjektmodellen. Den ska endast användas om det gör modellen mer förståelig.

Består-av-association	En består-av-association från domänobjekt X till domänobjekt Y indikerar att X är uppbyggt av objekt av typen Y. Som regel är består-av-association en association mellan instanser av objekt. En består-av-association har ett namn och en kardinalitet. Kardinaliteten definierar hur många instanser av det associerade objektet som kan associeras.

Operationer

De beteenden som beskrivs i användningsfallen knyts till objekten. Dessa objekt innehåller och utför operationerna.

Operation	En operation representerar ett specifikt beteende som utförs av en domänobjektinstans (eller klass). Varje operation har en signatur som definierar vilka specifika typer av stimuli som startar körningen av operationen. Signaturen inkluderar operationens namn och parametrar.

Kommunikation

Interaktionen mellan objekten visas med kommunikations-associationer. Dessa associationer visar vilka, och hur många, objekt som kommunicerar med varandra.

Kommunikations-association	En kommunikations-association mellan domänobjekt indikerar att de interagerar. Riktningen på associationen är samma som riktningen på stimuli-sändningen. Kommunikations-associationen är nästan alltid en instansassociation. En kommunikations-association har en kardinalitet som definierar hur många instanser av det associerade objektet som kan kommuniceras med.

Arv

Arvs-associationen visar att en objektklass ärver attribut, operationer och associationer från en annan klass.

Arvs-association	En arvs-association från en domänobjekt-klass (ättling) till en annan domänobjekt-klass (stamfader) indikerar att instansen av ättlingen ärver alla attribut, operationer och associationer som tillhör stamfadern.

4.4.5 Analysobjektmodell

Analysobjektmodellen skapar en komplett specifikation över funktionerna i systemet. Avbildningen görs utifrån ett idealtillstånd, utan att hänsyn tas till eventuella realiseringsfrågor. Detta ger möjlighet att diskutera systemet på en logisk nivå, utan att hänsyn tas till begränsningar i utvecklingsmiljön.

Ytterligare en fördel med att modellera ett system utifrån ett idealtillstånd är att det analyserade systemet lätt kan realiseras i olika dator- och systemmiljöer. Målet med analysobjektmodellen är att utveckla en modell som gör systemet tåligt mot förändringar och att ge utvecklarna ökad förståelse för problemområdet.

Analysobjekt

Analysobjekten ökar utvecklarens förståelse för vad systemet ska göra. Förståelsen för de olika funktionerna i systemet ökas genom att objekten delas upp i tre olika kategorier: *gränssnitts-*, *entitets-* och *kontrollobjekt*. Varje objekttyp har en naturlig koppling till problemområdet. Uppdelningen av analysobjekt ökar möjligheten att diskutera systemet på en logisk nivå.

Analysobjekt	Ett analysobjekt representerar ett verkligt objekt i en idealmodell. Analysobjektet skapar användbara koncept när man diskuterar systemets struktur på en logisk nivå. En analysobjektklass har ett namn. Analysobjekten utför gemensamt användningsfallen.

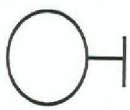
Varje typ av analysobjekt har en viss betydelse i analysobjektmodellen.

Gränssnittsobjekt

Gränssnittsobjekt används för att visa kommunikationen mellan systemet och dess omgivning. De definierar hur systemet presenteras för, och kommunicerar med, omgivningen. Detta omfattar transformation och översättning av händelser från omgivningen till de interna funktionerna i systemet och vice versa.

Gränssnittsobjekt är beroende av omgivningen medan entitets- och kontrollobjekt är oberoende. Förändringar i systemets presentation eller externa kommunikation påverkar endast gränssnittsobjekten. Systemet byggs därmed av fristående delar, vilket gör systemet stabilare.

Gränssnittsobjekten gör modellen klarare och enklare att förstå. Genom dessa visas systemets gränser.



Gränssnittobjekt

Entitetsobjekt

Entitetsobjekt används för att paketera attribut, associationer och beteenden som representerar fenomen i verkligheten. Dessa fenomen kan t ex vara personer, saker eller händelser. Objekten knyts inte till ett specifikt användningsfall, utan kan användas i flera olika användningsfall.

Entitetsobjekt är långlivade objekt i vilka data lagras. Deras attributvärden och associationer används ofta under hela systemets livstid. De innehåller dessutom operationer som används av andra objekt för att ändra objektets attributvärden och associationer.

Entitetsobjekt

Kontrollobjekt

Kontrollobjekt representerar ett specifikt användningsfalls beteende. Syftet med att använda kontrollobjekt är att kontrollera andra objekts beteende. Kontrollobjektens beteenden är, till skillnad från entitetsobjektens, inte relaterade till några speciella attribut eller associationer.

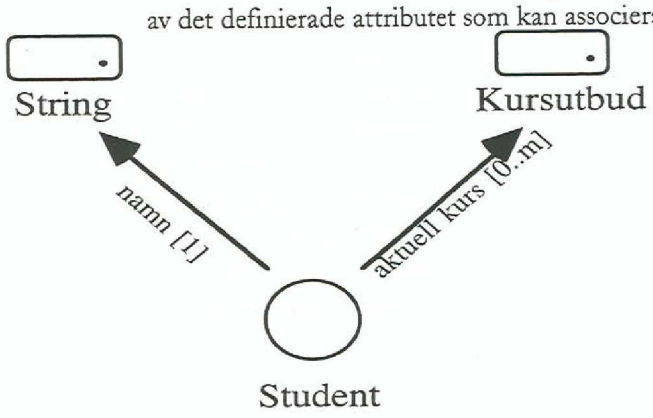
Beteendet för ett kontrollobjekt är oberoende av omgivningen. Hur systemet erhåller stimuli från aktörerna är oväsentligt. Transformationen av dessa signaler sköts av gränssnittsobjekten. Kontrollobjekten är ansvariga för att användningsfallen utförs. De instansieras när ett handlingsalternativ i ett användningsfall efterfrågas. Kontrollobjekten är därmed kortlivade, de upphör att existera när användningsfallet avslutas. De kan ses som logiska processer i systemet.

Kontrollobjekt

Attribut

Ett objekts egenskaper modelleras som attribut. Ett attribut består av en attributs-association från objektet till attributtypen. Attributtypen anger vilken sorts värde attributet innehåller. Attributs-associationen är en instansrelation, d v s en objektinstans knyts till en instans i attributklassen.

Attribut	Ett attribut i ett objekt representerar en lagrad mängd information om objektet. Attributet består av en attributs-association till en attributtyp. Varje attributs-association och attributtyp har ett namn. Associationen har också en kardinalitet som definierar hur många instanser av det definierade attributet som kan associeras.



Entitetsobjektets relation till sina attribut

Attribut för gränssnittsobjekt används ofta för att definiera presentationen av objektet. Dessa presentationer kan vara textmeddelande till användare eller fönsterstorlek i ett grafiskt användargränssnitt.

Kontrollobjekten lagrar kortlivade data. Därför är attribut som beskriver objektets egenskaper ofta överflödiga. Det underlättar däremot att infoga attribut till kontrollobjekten för att förklara dem och deras innebörd närmare.

Acquaintance-association

"Acquaintance-association" beskrevs tidigare i avsnittet om domänobjektmodellen. I detta avsnitt ges ytterligare beskrivningar av vad som gäller denna association mellan entitetsobjekt och gränssnittsobjekt.

Acquaintance-association	En acquaintance-association från analysobjekt X till analysobjekt Y indikerar att objekt X refererar till objekt Y. Som regel är acquaintance-associationen en association mellan instanser av objekt. En acquaintance-association har ett namn och en kardinalitet. När man associerar instanser med acquaintance, definierar kardinaliteten hur många instanser det associerade objektet kan associera.

Acquaintance-association mellan gränssnittsobjekt visar att ett gränssnittsobjekt känner till ett annat objekt och har en relation till det för att kunna utföra sin uppgift. Denna relation kan vara att ett fönster känner till ett annat fönsters position och beteende. Samma förhållande gäller för entitetsobjekten.

Aggregering

Likt föregående association är även denna beskriven i kapitlet om domänobjektmodellen. Aggregering syftar till att visa hur ett objekt kan vara uppbyggt av flera andra objekt.

Består-av-association	En består-av-association från analysobjekt X till analysobjekt Y indikerar att X är uppbyggt av objekt av typen Y. Som regel är består-av-association en association mellan instanser av objekt. En består-av-association har ett namn och en kardinalitet. Kardinaliteten definierar hur många instanser av det associerande objektet som kan associeras med associerade instanser.

Operation
Den grundläggande definitionen för operation finns i föregående avsnitt om domänobjektmodeller. Här beskrivs hur operationer kan appliceras på de olika analysobjekten.

Operation	En operation representerar ett specifikt beteende som utförs av en objektinstans. Varje operation har en signatur, som definierar vilka specifika typer av stimuli som startar exekveringen av operationen. Signaturen inkluderar operationens namn och parametrar.

Operationer på gränssnittsobjekt
Ett stimuli från en aktör till systemet kan vara av många olika slag. Ett mottaget stimuli startar en operation, vilken t ex kan ha till uppgift att ändra utseendet på skärmen.

Ibland är operationer enkla och då behöver inte objektet känna till vilka aktiviteter som föregått dem. De flesta operationer är dock komplexa. Dessa kan bete sig olika, beroende på vilket tillstånd objektet befinner sig i då operationen begärs. I dessa situationer är det förklarande att konstruera tillståndsdiagram. Dessa diagram visar de olika tillstånd ett objekt kan anta, vilka stimuli som kan tas emot och vilka beräkningar som kan utföras.

Operationer på entitetsobjekt
Operationer som utförs på entitetsobjekt ska vara generella. De ska inte bara gälla för ett objekt, utan för alla objekt i klassen.
Ett objekts operation kan till exempel utföra beräkningar och ändra värden i sina attribut.

Operationer på kontrollobjekt
Kontrollobjektets operationer liknar gränssnittsobjektets. Operationerna utför kommunikation och koordinering mellan objekten. Kontrollobjektens uppgift är att se till att användningsfallen utförs.

Prenumerera
Vid många tillfällen behöver ett objekt veta när någonting inträffar i ett annat objekt. Detta modelleras med hjälp av prenumerera-associationen ("subscribeTo-association").

Prenumerera	En prenumerera-association mellan två analysobjekt indikerar att det prenumererande objektet kommer att informeras när en speciell händelse inträffar i det associerade objektet. Varje prenumerera-association har en kardinalitet och ett villkor. Villkoret definierar vilken händelse som ska inträffa i det associerade objektet för att meddela sin prenumerant.

Vanligtvis är det gränssnitts- och kontrollobjekt som prenumererar på händelser som inträffar i entitetsobjekt.

Kommunikation
Användningsfall kommunicerar med aktörer. Detta representeras av kommunikations-associationer mellan aktören och användningsfallet. Under analysmodellering bestäms hur dessa användningsfall utförs, d v s vilka analysobjekt som utför vilka operationer och vilka analysobjekt som kommunicerar med varandra.

Det är väsentligt att definiera vilka objekt som ska sköta kommunikationen med omvärlden, aktörerna. Denna kommunikation bör skötas av gränssnittsobjekten.

Kommunikation	En kommunikations-association mellan analysobjekt eller mellan analysobjekt och en aktör indikerar att de interagerar. Riktningen på associationen är samma som riktningen på stimuli-sändningen. Som regel är en kommunikations-association en association mellan instanser av objekt och/eller aktörer. En kommunikations-association har en kardinalitet. När man associerar instanser med kommunikations-associationer definieras hur många instanser av det associerade objektet som kan kommuniceras med.

Kommunikations-associationer ska ritas från det sändande till det mottagande objektet. Riktningen på pilen uttrycker den riktning stimuli sänds.

Extension
En extensions-association från ett analysobjekt till ett annat ökar antalet beteenden och attribut hos det associerande objektet. En instans av det utökade objektet kan också utföra de beteenden som återfinns i det associerade objektet.

Extension	En extension mellan två analysobjekt indikerar att den associerade objektklassen kan bidra med att utöka beteende hos och informationen om den associerade objektsinstansen. Varje extentions-association har ett villkor.

Arv
Ett syfte med att använda arvsfunktioner i analysmodellen är att göra modellen tåligare mot förändringar. Arv används både för att specificera supertyper och göra specialiseringar.

Arv	Ett arv från en analysobjektklass (ättling) till en annan analysobjektklass (stamfader) indikerar att instansen av ättlingen ärver alla attribut, operationer och associationer som tillhör stamfadern.

Arv mellan gränssnittsobjekt gör det möjligt att återanvända och specialisera vanliga gränssnitt. Detta kan till exempel underlätta presentationer av uniforma skärmbilder, vilket gör systemet mer användarvänligt.

Arv mellan kontrollobjekt kan introduceras om flera användningsfall associerar ett abstrakt användningsfall. Flera kontrollobjekt kan dessutom ärva ett gemensamt kommunikationsprotokoll från en anfadersklass.

4.4.6 Arbetsgång

Att arbeta med ObjectOry kan delas upp i flera olika delmoment. Arbetet börjar med en kravanalys som resulterar i en användningsfallsmodell och en domänobjektsmodell. Efter kravanalysen gör man en robusthetsanalys. Den resulterar i en analysmodell som innehåller de tre olika typerna av analysobjekt.

Efter dessa två analyssteg går man vidare till design, realisering och testning. Dessa steg i utvecklingsprocessen faller utanför ramen för denna rapport.

Kravanalys

Kravanalysen syftar till att:

- Komma överens med kunden och användaren om vad systemet ska göra.
- Ge systemutvecklarna bättre förståelse för de krav som ställs på systemet.
- Utveckla en modell som beskriver vad systemet ska göra.
- Avgränsa systemet.

Huvudsyftet med kravanalysen är att beskriva vad systemet ska göra. För att klarlägga och beskriva detta utvecklas en användningsfallsmodell och en domänobjektsmodell.

Användningsfallsmodell

I denna modell visas systemets miljö genom att ange aktörer som interagerar med systemet. Denna beskrivning görs i löpande text. Det underlättar att göra en lista över den terminologi och de begrepp som används för att definiera användningsfallen. Kommunikations-association är den vanligaste associationen i en användningsfallsmodell.

Domänobjektsmodell

Modellen innehåller objekt hämtade från problemområdet. För att göra en modell tydligare bör "acquaintance"-, består-av- och arvs-associationer användas. Objektets alla attribut behöver inte beskrivas i detalj. De ska endast beskriva objektets semantiska mening, inte ligga till grund för realiseringen. Realiseringsattribut definieras senare i designmodellen.

Syftet med domänobjektmodellen är att öka analytikerns förståelse för problemområdet. Man bör därför inte ange operationer, kommunikationsassociationer eller detaljerade attribut i denna modell.

Arbetssteg för kravanalysen:

1. Definiera möjliga aktörer.

2. Hitta och definiera typiska användningsfall. Detta görs utifrån de kravspecifikationer som gjorts.

3. Strukturera användningsfallen. Skapa extensioner och använder-associationer. I detta arbete kan ytterligare aktörer identifieras.

4. Studera aktörerna för att hitta gemensamma egenskaper och gemensamma roller. Vid dessa tillfällen skapas arvsrelationer mellan aktörerna.

5. Utveckla en domänobjektmodell. Domänobjekten finns ofta som begrepp i användningsfallen.

Kravanalysen är en iterativ process. I punkt tre identifieras ytterligare aktörer, och arbetet börjar om från punkt ett för dessa.

Dokumentation
Kravmodellen dokumenteras i användningsfallsöversikt, användningsfallsbeskrivning, grafiska modeller över användningsfall och deras aktörer, kataloger som innehåller begrepp, kataloger som innehåller domänobjekt och grafiska modeller över dem samt en förfinad kravspecifikation.

Robusthetsanalys
Robusthetsanalysen syftar till att:

- Utveckla en analysobjektmodell som gör systemet tåligt mot förändringar.
- Öka förståelsen för systemet.

I robusthetsanalysen utvecklas en analysobjektmodell. Denna modell visar de krav som ställs på systemet, med andra ord identifieras de objekt som utför vad användningsfallen beskriver. Detta betyder att beteendet för varje användningsfall utförs av analysobjekt som interagerar.

Analysmodellen består av entitetsobjekt, gränssnittsobjekt och kontrollobjekt. Dessa objekt har attribut och relationer till varandra. De grupperas därefter i paket och subsystem.

Gränssnittsobjektet representerar fenomen som gör det möjligt för systemet att kommunicera med omgivningen. Entitetsobjekt representerar data som lagras i systemet. Dessa motsvarar ofta domänobjekt. Kontrollobjekten representerar ett processrelaterat beteende i systemet.

Analysobjekt kan struktureras i subsystem och servicepaket. Analysobjekt som är relaterade till varandra grupperas utifrån ett visst kriterium. För att skapa ett tåligt system är det viktigt att gruppera objekten så att effekten av förändringar minimeras.

Arbetssteg för robusthetsanalysen:

1. Definiera systemets arkitektur genom att skapa subsystem för generella funktioner.

2. Studera domänobjekten för att identifiera analysobjekt.

3. Identifiera analysobjekt som krävs för att utföra användningsfallen. Skapa grafiska modeller över de objekt som deltar i varje användningsfall. Distribuera användningsfallens beteende till objekten.

4. Sammanfoga de olika grafiska modellerna för att betrakta hela systemet.

5. När användningsfallens beteende har distribuerats mellan objekten, identifieras kraven för varje objekt och beteenden definieras.

6. Gruppera objekten i paket.

Dokumentation
Robusthetsmodellen dokumenteras i analysöversikt, analysobjektsbeskrivning, grafiska modeller över analysobjekten, analyssubsystem samt anlyspaketsbeskrivning.

5. Metodlikheter

Detta kapitel behandlar de begrepp och den dokumentation som är liknande för de olika metoderna. Metoderna fokuserar olika aspekter i problemområdet. Detta har i vissa fall försvårat jämförelsen eftersom syftet med respektive begrepp speglar olika stadier i utvecklingsprocessen.

För att nyansera överensstämmelsen mellan begreppen har de delats upp i begreppssynonymer och begreppsparalleller. För varje begrepp ges en generell definition som sammanfattar de tre metoderna.

5.1 Begreppssynonymer

Begreppen som listas i detta avsnitt anses vara synonyma, d v s definitionen i en metod ska passa in i en annan metods sammanhang.

Klasser

En klass är en mall som anger hur ett objekts egenskaper, operationer och realtioner definieras.

I en klass finns inga instanser (objekt). Detta gör att klassen inte kan betraktas som en mängd, utan som en typ. Begreppet klass kan användas synonymt med begreppet typ.

Generell definition: Beskrivning av gemensamma egenskaper, operationer och relationer för objekt. Klassen anger begreppsmässig betydelse för instanserna. Klassen är en typ.

OOA	OMT	ObjectOry
Klass	Abstrakt klass	Objektklass

Objekt

Ett objekt är någonting systemet sparar information om och/eller påverekar. Objekt ska ha klara gränser och ha betydelse för systemet. Objekt kapslar in data och utför de beteenden som definieras i klassen. Begreppet objekt kan användas synomymt med begreppen instans och förekomst.

Generell definition: En enhet av någonting som systemet känner till, som har vissa egenskaper och utför vissa beteenden.

OOA	OMT	ObjectOry
Objekt	Objekt	Objekt

Tillstånd

Ett objekts tillstånd är de sammantagna attributvärdena och de relationer objektet deltar i för ögonblicket. Om ett attributvärde ändras, ändras objektets tillstånd.

Generell definition: Tillstånd är en kombination av aktuella attributvärden och relationer för ett objekt."

OOA	OMT	ObjectOry
Tillstånd	Tillstånd	Tillstånd

Generalisering

Generalisering är ett sätt att strukturera en modell. Gemensamma egenskaper kan identifieras hos klasser. Dessa gemensamma egenskaper samlas i en klass, s k superklass. Specialiserade klasser, subklasser, ärver attribut, operationer och relationer från sin/sina superklasser. Alla instanser i subklassen är samtidigt instanser i sina superklasser.

Super- och subklasser minskar redundans och gör modellen klarare. Generalisering är en klasstruktur.

Generell definition: Ett sätt att strukturera en modell. Gemensamma egenskaper samlas i en superklass, från vilken subklasser ärver egenskaper.

OOA	OMT	ObjectOry
Gen/spec-struktur	Generalisering	Arvs-association

Aggregering

För att visa hur ett objekt kan bestå av andra objekt används aggregering. Aggregering är en struktur mellan instanser.
Aggregering är ett sätt att höja abstraktionsnivån. En enhet betraktas som en helhet, trots att den är en sammansättning av flera enheter.

Generell definition: Aggregering används för att åskådliggöra att ett objekt som betraktas som en helhet består av flera enheter.

OOA	OMT	ObjectOry
Hel/delar-struktur	Aggregering	Består-av-association

Operation

Operationer representerar beteenden som objektet kan utföra. Beteendet utförs endast på attribut och länkar som tillhör den klass operationen tillhör.

Generell definition: En operation utför ett visst beteende på den egna klassen.

OOA	OMT	ObjectOry
Tjänst	Operation	Operation

Paket
En modell kan delas upp i mindre enheter. Dessa enheter innehåller klasser och egenskaper samt deras inbördes relationer. Ett paket ska sträva efter oberoende från andra paket, vilket leder till stabilare system.

Generell definition: Ett paket är en logisk konstruktion som grupperar klasser utifrån något kriterium.

OOA	OMT	ObjectOry
Subjekt	Modul	Paket

5.2 Begreppsparalleller
I detta kapitel listas de begrepp som är snarlika varandra. Försök till generella definitioner görs även.

Klass&Objekt
För att ange klasser som innehåller objekt använder OOA-begreppet klass&objekt. Detta är en klassdefinition som innehåller instanser, d v s det är samtidigt en mängd. OMT använder termen klass för att ange en klassdefinition som innehåller instanser.

Generell definition: Klass och Objekt är en klassdefinition som innehåller instanser.

OOA	OMT	ObjectOry
Klass&Objekt	Klass	-

Attribut
Attribut används i metoderna för att beskriva ett objekts egenskaper. Termen attribut syftar däremot på skilda företeelser. Detta beror på att metoderna refererar till olika företeelser. I OMT refereras till instansnivå, medan OOA och ObjectOry refererar till klassnivå.

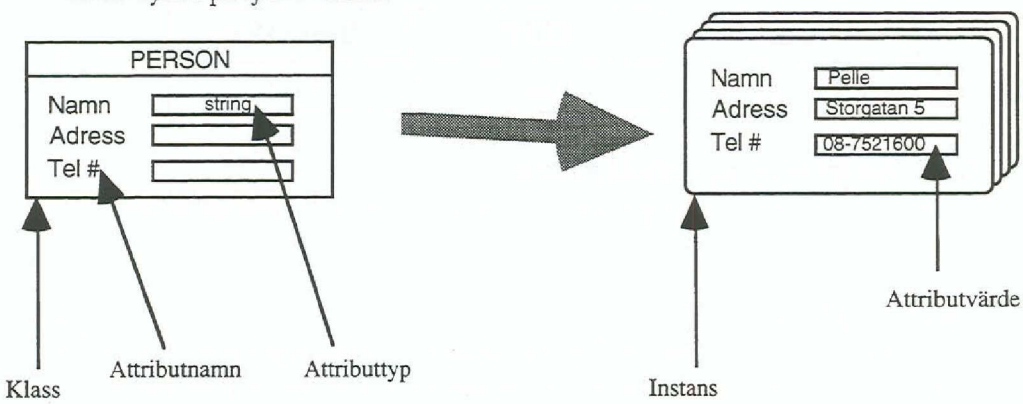
För att göra en klar distinktion beskrivs nedan de olika betydelser som finns för begreppet attribut.

Attributnamn
OOA och ObjectOry syftar med attribut på attributnamn. Attributnamn är det namn som anges för egenskapen i en klassbeskrivningen. OOA och ObjectOry syftar på attributnamn när de använder begreppet attribut. Detta är en beskrivning på klassnivå.

Attributtyp
Med attributtyp menas i alla metoderna den typ av värde som lagras i attributnamnet. Attributtyper kan t ex vara string, integer eller en användar definierad mängd (värdeförråd). Attributtyp används på klassnivå.

Attributvärde
Attributvärdet är det värde som gäller för respektive instans. Ett attributvärde gäller för en instans. OMT syftar på attributvärde när begreppet attribut används. Attributvärde används på instansnivå.

Alla tre metoder syftar dock med begreppet attribut på en egenskap som gäller för objektet. Skillnaden ligger i att OOA och ObjectOry syftar på beskrivningen och OMT syftar på själva värdet.



Skillnader mellan olika attributbegrepp på klass och instansnivå

OOA	OMT	ObjectOry
Attribut (-namn)	Attribut (-värde)	Attribut (-namn)

Relationer
Relationer visar att två eller flera objekt är relaterade till varandra. I OOA modelleras detta med instansrelationer, i OMT via länkar och i ObjectOry via "acquaintance"-associationer. Skillnaden mellan OMTs länkar och OOAs och ObjectOrys instansrelationer respektive associationer är att OMT kan modellera länkar med attribut. Om en instansrelation eller acquaintance-association ska ha ett attribut måste detta modelleras som en egen klass.

Generell definition: Relationer mellan objekt innebär att ett objekt känner till ett annat, eller flera andra, objekt.

OOA	OMT	ObjectOry
Instansrelation	Länk	Acquaintance-association

Kommunikation
OOA modellerar kommunikation via meddelandeförbindelser. Meddelandeförbindelser är en motsvarighet till den språkliga imperativformen. Med detta menas att ett objekt kan skicka en uppmaning till ett annat objekt att utföra en viss operation.

ObjectOry anger kommunikations-association som ett sätt att visa att två objekt interagerar. Kommunikationen är enkelriktad.

OMT har inget uttryckssätt för att visa kommunikation explicit. Kommunikation modelleras som associationer.

Generell definition: Ett sätt att modellera att två objekt interagerar. Ett meddelande skickas i en riktning.

OOA	OMT	ObjectOry
Meddelande-förbindelse	–	Kommunikations-association

5.3 Dokumentation

I detta avsnitt presenteras de modeller som metoderna producerar. Respektive metods modeller kan inte, så som begreppssynonymer, ersätta varandra. Det finns dock vissa likheter mellan de resulterande dokumenten. Framförallt liknar OOAs och OMTs dokumentation varandra. ObjectOrys dokumentation är däremot mer avvikande.

5.3.1 Modeller

En OOA-analys resulterar i en OOA-modell och en uppsättning tjänste- och tillståndsdiagram.

OMT-modellering leder till tre olika modeller: objektmodell, dynamisk modell och funktionsmodell.

ObjectOry producerar en rad olika dokument. De flesta är av verbal karaktär. Till den verbala dokumentationen kan grafer av olika slag knytas.

Objektmodell

I objektmodellen beskrivs de objekt som identifieras, deras attribut och operationer. I modellen visas dessutom hur de olika objekten är relaterade till varandra. Relationerna mellan objekten är associationer, generaliseringar och aggregeringar. Detta medför att både objekt och klasser visas i samma modell.

OOA-modellen överensstämmer till största delen med OMTs objektmodell. ObjectOry har ingen heltäckande objektmodell, utan visar objekt och deras egenskaper i enskilda grafer. I dessa grafer fokuseras, t ex objekt och deras associationer, objekt och deras attribut eller objekt och deras beteende.

Generell definition: Objektmodell innehåller klasser och objekt, deras attribut, tjänster och relationer. Klasserna/objekten visas i strukturer för att öka förståelsen för modellen. Större modeller kan delas upp i mindre delmodeller.

OOA	OMT	ObjectOry
OOA-modell	Objektmodell	–

5.3.2 Diagram

För att ytterligare förklara skeendet i systemet använder metoderna diagram. Dessa diagram förklarar det operationella flödet. Diagrammen utgår från samma princip. Dock kan de både vara fristående och en del av en modell.

Tillståndsdiagram
OOA använder tillståndsdiagram som en förklarande beskrivning av de tillstånd klasserna i OOA-modellen kan anta.

OMT placerar sina tillståndsdiagram i en fristående modell som kallas dynamisk modell. Tillståndsdiagrammet visar möjliga tillstånd och händelser för respektive klass.

ObjectOry preciserar inte när tillståndsdiagram ska användas, men de introduceras ofta som komplement till analysobjektmodellen.

Generell definition: Tillståndsdiagram visar relationer och ordningsföljd mellan tillstånd och händelser för en klass.

OOA	OMT	ObjectOry
Tillståndsdiagram	Tillståndsdiagram	Tillståndsdiagram

Interaktionsdiagram
För att precisera hur objekten interagerar skapas interaktionsdiagram. OMT kallar detta händelsespårning (event trace) och ObjectOry interaktionsdiagram.

Generell definition: Interaktionsdiagram är en graf som visar sekvenser av kommunikationen mellan objekt.

OOA	OMT	ObjectOry
–	Händelsespårning (Event trace)	Interaktionsdiagram

5.3.3 Verbala beskrivningar

Objektbeskrivningar
OOA beskriver framför allt sina modeller i grafer. Det stora analysresultatet är OOA-modellen. Denna kompletteras med tjänstediagram och verbala beskrivningar. Klass-&-objekten beskrivs med klass&objekt-specifikationer. I dessa anges klassens namn, attribut, externa in- och utdata och operationer. Denna beskrivning görs i punktform, inte som en beskrivning i löpande text.

OMT bygger sina modeller på grafer. De objekt som används i objektmodellen beskrivs kortfattat i ett databibliotek. I detta bibliotek listas respektive objekt och en kort förklaring av objektet. Beskrivningen specificerar objektens innebörd i systemet i löpande text.

ObjectOry bygger en stor del av sin dokumentation på verbala beskrivningar. Som nämndes ovan förekommer dock en del grafer, som bl a beskriver tillståndsförändringar och objektrelationer.

OOA	OMT	ObjectOry
Klass&Objekt-specifikation	Databibliotek (Data dictionary)	Objektbeskrivning

Scenarier
ObjectOry bygger hela sin metod runt användningsfallen. Ett användningsfall är en funktion som en användare (aktör) kan begära att systemet ska utföra. Utifrån dessa användningsfall byggs analysmodellen, designmodellen och testningsarbetet upp.

OMT skapar ett liknade schema. Detta schema används för att hitta händelser och operationer som kan användas i den dynamiska modellen.

Scenariots roll i analysarbetet skiljer sig avsevärt mellan de två metoderna. Enligt ObjectOry är det användningsfallen som driver hela utvecklingsprocessen. Dessa initierar och styr utvecklingsprocessen. OMT använder scenarier endast för att identifiera händelser och operationer i den dynamiska modellen.
Skillnaden i användning är avsevärd, men texterna liknar varandra till formen.

Generell definition: Ett scenario är ett antal handlingar som systemet ska utföra. Scenarier initieras ofta av användare (aktörer).

OOA	OMT	ObjectOry
–	Scenario	Användningsfall

6. Metodskillnader

I detta kapitel beskrivs de olikheter som identifierats i metoderna. Metoderna är skrivna av olika personer, med delvis olika fokus och med olika syn på systemutvecklingsprocessen. Detta gör att metoderna i sig är olika.

Det är svårt att dra en gräns mellan vad som är skillnader i metoderna och vad som är nyanser och varianter av liknande tankar.

Kapitlet är upplagt så att varje metod beskrivs för sig. I dessa beskrivningar presenteras de egenskaper, funktioner och idéer som metoden framhåller, som inte finns i de övriga metoderna.

6.1 Object Oriented Analysis (OOA)

OOA är en enkel metod. Den använder en enkel notation och endast ett fåtal begrepp. Många av de koncept OOA använder återfinns i de övriga metoderna. Det finns dock några företeelser som OOA är ensam om.

Uppdelning av aggregering

För att underlätta arbetet med att hitta aggregeringsstrukturer delar OOA upp aggregering i tre typer. Dessa olika typer av aggregering behandlas inte på något speciellt sätt i dokumentationen och kräver ingen egen notation. Uppdelningen är endast ett sätt att hjälpa analytikern att hitta aktuella strukturer.

Sammansättning – del

Denna struktur är den vanligaste aggregeringen. Den betyder att ett objekt är fysiskt uppbyggt av flera andra objekt. Det kan till exempel vara en cykel som är uppbygd av hjul, ram och sadel.

Innehåller – innehåll

Denna struktur visar att någonting innehåller föremål. Helheten är inte uppbyggd av de mindre föremålen. Flygplan innehåller piloter, flygvärdinnor och passagerare.

Samling – medlemmar

Detta är en abstrakt aggregering. Den går inte att granska som en fysisk företeelse. Det handlar snarare om att en företeelse består av en rad andra företeelser. En flygresa kan bestå av flera delflygningar, som åtskiljs av mellanlandningar. Aggregeringen går inte att ta på, utan är en mental modell över den tänkta verkligheten.

Tjänstediagram

OOA använder tjänstediagram för att definiera de algoritmer som tjänsterna ska arbeta utifrån. Tjänstediagrammet påminner mycket om strukturdiagram. Tjänstediagrammet innehåller processer, loopar, villkor och kopplingar. Villkoren som används gör att en operation kan vara tillståndsberoende. Exempel: När villkoret är sant utförs operationen.

Här skiljer sig OOA markant från OMT. OMT bygger en av sina tre modeller, funktionsmodellen, på dataflödesscheman. Coad och Yourdon skriver att ett dataflödesdiagram är ett dåligt val för att stega sig igenom en algoritm; ett flödesschema är idealiskt.

Tjänste-/tillståndstabell
Tjänste-/tillståndstabellen är en tabell som knyter samman tjänstediagram och tillståndsdiagram. I tabellen anges tillstånd i kolumnerna och tjänster i raderna. I cellerna markeras vilka tjänster som kan utföras när objektet befinner sig i ett visst tillstånd.

6.2 Object Modeling Technique (OMT)

OMT är en mer komplicerad metod än OOA. Metoden använder dock samma grundkoncept som OOA. Dessa koncept har sedan utökats och förfinats.

OMT har är inte speciellt många nya begrepp jämfört med OOA, utan det rör sig företrädesvis om nyanser och tolkningar av grundbegreppen. OMT är mer dataorienterad och bygger mycket på informationsmodeller. Detta lyser ibland igenom som t ex när begrepp som "candidate key" används för att identifiera objekt.

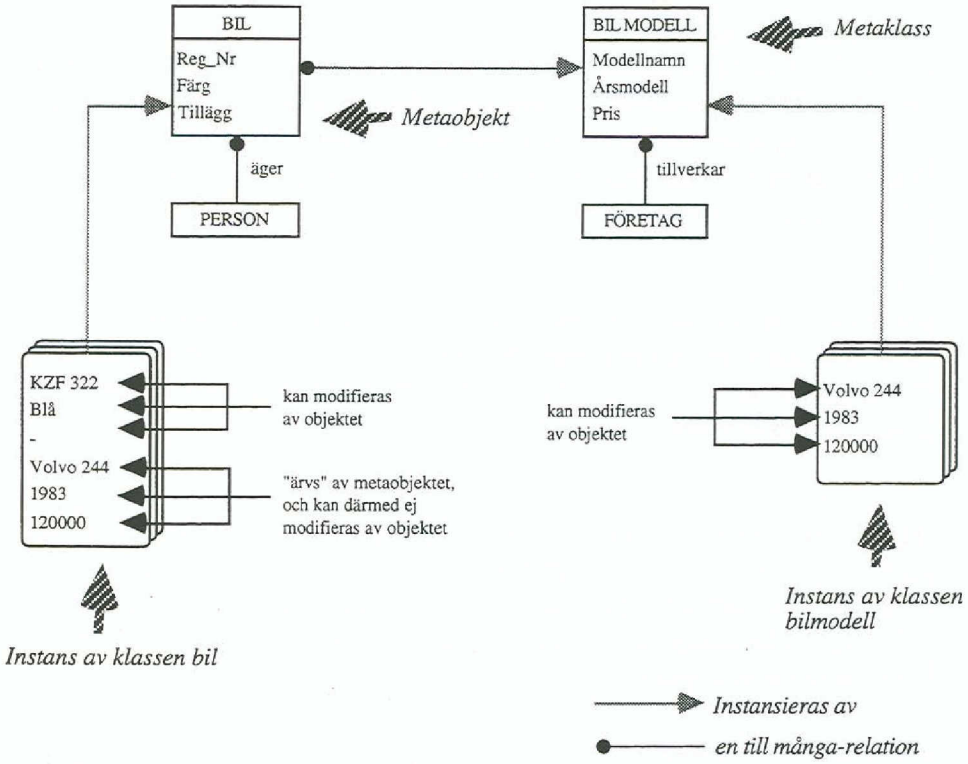
Join class
Metoden har ett eget begrepp, "join class", för att visa att en subklass ärver egenskaper av två eller flera superklasser. Notationen i grafen är den samma som för vanliga subklasser.

Metaklass
I OMT finns möjlighet att betrakta klasser som instanser. Detta gör det möjligt att definiera en klass (typ), vars instanser i sig är klasser (typer). Klasser, vars instanser är typer, kallas metaklasser. Instanser som är klasser (typer) kallas metaobjekt.

Exempel: Bilmodell är en klass som definierar en bilmodells egenskaper. I klassen skapas en instans för varje ny bilmodell. Bilmodellen Volvo 244 är således en instans av klassen bilmodell.

Av en bilmodell kan det förekomma många enskilda fordon. Dessa fordon instansieras i klassen bil. Bil är en instans av klassen bilmodell och samtidigt en klass för instanser av enskilda fordon. Fordon är således av en modell och har dessutom vissa unika egenskaper.

Egenskaperna för respektive bilmodell ärvs av de fordon som är av den typen. Dessa egenskaper kan inte modifieras. Dessutom har varje fordon unika egenskaper,så som registreringsnummer.En instans av bil kan modifiera sina unika egenskaper, så som tillbehör och färg, men kan inte ändra modellens tillverkningsår eller försäljningspris.



Beskrivning av metaklasser och metobjekt

Attributtyper

OMT definierar olika sorters attribut. Dessa attribut är avsedda för en viss uppgift.

Länkattribut

I OMT kan man knyta ett attribut till en relation mellan två, eller flera, objekt. Detta arv från Entity Relationship-modeller finns inte i OOA. OOA gör istället nya klasser när länkattribut behövs. Länkattribut kan också modelleras som egna klasser i OMT. Detta görs om operationer ska utföras på relationen.

Qualifier

"Qualifier" är ett attribut som kan liknas vid en nyckel. Genom att använda denna nyckel kan en ett-till-många-relation reduceras till en ett-till-ett-relation. Genom att ange objektet i ett-klassen samt ett attributvärde för qualifier-attributet kan ett specifikt objekt i många-klassen pekas ut.

Discriminator

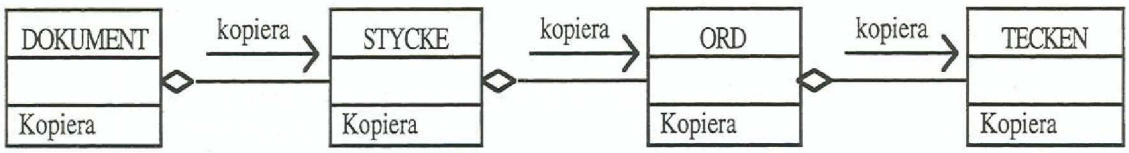
"Discriminator" är den attributtyp som avgör vilken egenskap som styr en specialisering. Endast en egenskap hos klassen kan specialiseras. Discriminatorn är enkelt uttryckt ett kriterium, utifrån vilket man specialiserar klasser. T ex kan klassen fordon specialiseras utifrån kriteriet drivmedel. Detta ger specialiseringarna eldriven, gasdriven, bensindriven eller dieseldriven.

Uppdelning av operationer
Händelser kan utlösa operationer. När en viss händelse inträffar körs en viss operation. Dessa operationer kan vara av två olika slag, aktiviteter och handlingar. Dessa två slag av operationer ska inte förväxlas med klassificeringen av operationer. En operation kan klassificeras som enkel eller komplicerad. Denna typ av klassificering finns även i OOA.

Aktivitet
Aktivitet är en operation som tar tid att utföra. Den är ofta uppbyggd av flera underfunktioner, som tillsammans utför operationen. Aktiviteter knyts till tillstånd. När objektet befinner sig i ett visst tillstånd utförs aktiviteten. Denna aktivitet pågår till dess objektet ändrar tillstånd igen.

Handling
Handlingar är ögonblickliga. De tar naturligtvis en viss tid att utföra, men den är så försvinnande liten att systemet betraktar dem som ögonblickliga. En handling är atomär, d v s antingen utförs den eller utförs den inte. Den kan aldrig utföras "halvvägs". Handlingar utförs som svar på händelser. När en viss händelsen inträffar utförs handlingen.

Propagation of operations
Operationer ärvs av alla subklasser i en generaliseringsstruktur. För att sprida en klassoperation till delklasserna i en aggregeringsstruktur använder OMT "propagation of operations". Att sprida operationerna till delarna innebär att polymorfism utnyttjas. Delarna utför inte exakt samma funktion, men resultatet blir det samma.



Propagation of operations

Exempel
Ett dokument består av stycken som består av ord som består av tecken. Varje objekt kan utföra operationen kopiera. Kopieras dokument, kopieras alla tecken, ord, stycken. Kopierar man ett tecken gäller inte det omvända. Då kopieras endast tecknet, inte de övriga objekten i relationen.

Associationer
OMT kan modellera relationer mellan klasser. Övriga metoder relaterar endast objektinstanser till varandra. Relationer mellan instanser modelleras i OMT med hjälp av länkar. En grupp länkar som knyter samman objekt från samma klasser bildar en association. Notationen är gemensam för länkar och associationer.

Restriktioner på associationer
I OMT kan restriktioner på en association uttryckas. Dessa restriktioner begränsar antalet deltagande instanser i en länk. I associationen anges vilken annan association som definierar dess möjliga deltagare.

Rollnamn
Rollnamn används i OMT för att göra modellen tydligare. Rollnamnen anger vilken roll objekten spelar i relationen. Motsvarande notation används i informationsmodellering. Exempel på rollnamn kan vara anställd för personklassen och arbetsgivare för företagsklassen.

Dataflödesdiagram
Till skillnad från de övriga modellerna använder OMT dataflödesdiagram. Dessa används för att visa hur resultat beräknas i operationerna. På denna punkt skiljer sig OOA och OMT åt. OOA använder tjänstediagram för att specificera hur algoritmer styr operationerna.

6.3 ObjectOry

Den största skillnaden mellan ObjectOry och de två andra metoderna ligger i dokumentationen. OOA och OMT arbetar främst med grafiska modeller och diagram. ObjectOry använder sig däremot främst av verbala dokument. I dessa dokument beskrivs, i löpande text, t ex vad användningsfallen utför eller aktörernas egenskaper. ObjectOry kräver inte någon dokumentation i grafisk form. Denna typ av dokumentation är frivillig och kan användas för att förtydliga delar av systemet.

Uppdelning av objekt
ObjectOry delar upp objekten i domänobjekt och analysobjekt. Domänobjekten liknar de objekt som presenteras i de övriga metoderna. Dessa objekt speglar företeelser inom problemområdet. Domänobjekt används i domänobjektmodellen.

Analysobjekten är förfiningar av domänobjekten. De används i analysobjektsmodellen och delas upp i tre olika kategorier: gränssnitts-, entitets- och kontrollobjekt. Gränssnitts- och kontrollobjekt är mer beroende av en dator än entitetsobjekt. Entitetsobjekten kan ofta identifieras i domänobjekten.

Gränssnittsobjekt	Gränssnittsobjekt används för att ta hand om kommunikation mellan systemet och dess omgivning. De klargör var systemets gräns går. Vanliga gränssnittsobjekt är fönster, kommunikationsprotokoll, printerinterface, sensorer o s v.
Entitetsobjekt	Entitetsobjekt reflekterar verkliga fenomen och är ofta långlivade. De används för att paketera attribut, associationer och beteenden relaterade till vissa fenomen. Entitetsobjekt behöver inte vara relaterade till ett speciellt användningsfall.

Kontrollobjekt Kontrollobjekt används för att paketera processrelaterade beteenden, med andra ord användningsfallsspecifika beteenden. Dessa objekt kontrollerar och koordinerar de andra objekten.

Aktör

Användningsfall är en mycket viktig del av ObjectOry. Det är dessa användningsfall som driver utvecklingsprocessen. Användningsfallen i sig liknar de scenarier som används i OMT.

För att identifiera användningsfall identifieras aktörer. Aktörer är ofta användare av systemet som begär att få funktioner utförda. Aktörer återfinns också i OMTs dataflödesschema som producenter och konsumenter av data.

Extension

ObjectOry använder en rad olika associationer för att visa vilka relationer olika objekt har till varandra.

Extensions-association betyder att ett objekt eller användningsfall kan utföra vissa operationer åt en instans som tillhör ett annat objekt eller användningsfall. Detta är som en programmässig konstruktion där en procedur anropar en annan procedur för att utföra en viss funktion.

Prenumerera

Prenumerera är en form av kommunikations-association. Denna association meddelar det prenumererande objektet när någonting händer i det avsändande objektet. Ofta är det gränssnitts- och kontrollobjekt som prenumererar på händelser som inträffar i entitetsobjekt.

7. Paralleller till modelleringshandboken

De likheter som identifierats i de objektorienterade (OO) metoderna ställs i detta kapitel mot begrepp och arbetssätt i modelleringshandboken (MHB). Förutom de likheter som presenterats i kapitel 5 kommer även vissa unika egenskaper för respektive metod att belysas i skenet av modelleringshandbokens termer.

7.1 MHBs motsvarighet till vissa generella OO-begrepp

I detta avsnitt kopplas de egenskaper som är gemensamma för de tre metoderna till begrepp och termer i modelleringshandboken. OO-begreppen och MHB-begreppen stämmer sällan exakt överens, men de gemensamma dragen är ofta så tydliga att det är mer som förenar än skiljer dem åt.

Klasser

I modelleringshandboken beskrivs prototyper (allmänbegrepp) som ett mönster, som gäller för "alla typiska förekomster". Detta mönster anger vilka egenskaper som gäller för de individuella förekomsterna. De individuella förekomsterna, instanserna, modelleras ej i modelleringshandboken.

MHB-begreppet individualbegrepp motsvarar OO-begreppet objekt, med den skillnad att inget beteende anges för individualbegreppet.

I prototypen anges namn och egenskaper. Prototyper definierar däremot inte beteenden. Detta skiljer dem från klasser, eftersom klasser även definierar de operationer som deras instanser kan utföra. Klass i de objektorienterade metoderna motsvarar de prototyper som används för att definiera MHBs resursmodell.

Objektorientering	Modelleringshandboken
Klass*	Prototyp (resurs)

* Klass visar dessutom beteende, vilket prototyper inte gör.

Attributtyp

I modelleringshandboken anges en prototyps egenskaper som attribut. Attributen ska vara lokala, d v s objektegna företeelser. Dessa egenskaper är möjliga att förstå var för sig. Till varje attribut hör ett värdeförråd som definierar vilka värden attributet får innehålla.

Instanser beaktas inte i modelleringshandboken. Därför motsvarar attribut attributtyper i de objektorienterade metoderna. (se avsnitt 5.2 Attribut sid 63)

Objektorientering	Modelleringshandboken
Attributtyp	Attribut

Relationer

På samma sätt som relationer mellan instanser modelleras i de objektorienterade metoderna, modelleras i modelleringshandboken samband mellan prototyper i resursmodellen. Ett samband uttrycker att två prototyper känner till och kräver kunskap om varandra.

Samband uttrycker den statiska strukturen i modellen. Vid sambanden anges kardinalitet, d v s hur många instanser av respektive prototyp som krävs för att sambandet ska vara giltigt.

Objektorientering	Modelleringshandboken
Relation	Samband

Generalisering

För att uttrycka superklasser och subklasser använder modelleringshandboken och de objektorienterade metoderna samma struktur, generalisering-specialisering. Denna struktur visar att en klass eller prototyp samlar gemensamma egenskaper, som därefter ärvs av flera olika specialiseringar.

Prototyperna uttrycker, som tidigare nämnts, inte beteende, varför detta ej heller kan uttryckas i generaliseringsstrukturen.

Objektorientering	Modelleringshandboken
Generalisering	Generalisering/specialisering

Aggregering

Aggregering uttrycker att en enhet är uppbyggd av flera komponenter. Detta är en abstraktionsmekanism för att fokusera det väsentliga i problemområdet.

Det som skiljer objektorienterad aggregering från modelleringshandbokens generalisering är att de två ansatserna relaterar olika företeelser. Objektorienteringen relaterar instanser medan modelleringshandboken relaterar prototyper.

Objektorientering	Modelleringshandboken
Aggregering	Aggregering/dekomponering

Operationer

I modelleringshandbokens statiska modell, resursmodellen, anges inte de olika prototypernas beteende. Detta uttrycks i en annan modell, flödesmodellen.

I de objektorienterade metoderna anges de operationer objekten kan utföra i den statiska modellen. Hur de utförs definieras därefter i andra modeller och diagram.

I flödesmodellen anges operationer som arbetsuppgifter i arbetspunkter. Dessa arbetspunkter kan inte sägas motsvara klasser eftersom de inte har några egenskaper. De objekt som arbetsuppgifterna opererar på presenteras som sak- och informationsflöden. Dessa flöden visas som in- respektive utdata i arbetspunkterna.

På samma sätt som operationer anges i klasser, anges arbetsuppgifter i arbetspunkter.

Objektorientering	Modelleringshandboken
Operation	Arbetsuppgift

Kommunikation
Kommunikation uttrycker att klasser kommunicerar. Denna kommunikation är enkelriktad, d v s går från en sändare till en mottagare. Kommunikationen bygger på att en sändare skickar ett meddelande till en mottagare. Den mottagande klassen tar emot meddelandet och bestämmer därefter själv hur den ska agera.

I de objektorienterade metoderna modelleras detta som kommunikationsassociationer. Hur den dynamiska kommunikationen sekvensieras uttrycks inte i dessa associationer. Kommunikation modelleras i de objektorienterade ansatserna i den statiska objektmodellen.

Modelleringshandboken visar hur olika arbetspunkter kommunicerar. Detta görs med informationsflöden i flödesmodellen.

Objektorientering	Modelleringshandboken
Kommunikation	Informationsflöde

Objektmodell
Den statiska strukturen visas i OOA och OMT i objektmodellen. I denna modell visas klasser, deras egenskaper och relationer. I klasserna anges attribut och operationer. Objektmodellen liknar modelleringshandbokens resursmodell. Det som saknas är möjligheten att uttrycka vilka beteenden som prototyperna kan utföra.

I modelleringshandboken ges en möjlighet att göra en generell begreppsmodell. Denna modell innehåller prototyper, attribut, relationer och beteenden. Hänsyn tas till både resursaspekten och till agerande/flödesaspekten. En prototyp kan vara både en resurs med attribut och en aktör (arbetspunkt) med beteende. I denna modell uttrycks transportpilar som vanliga relationer.

Objektorientering	Modelleringshandboken
Objektmodell	Generell begreppsmodell

7.2 OO-begrepp som går att modellera i MHB

Det finns begrepp i de enskilda metoderna som inte har någon direkt motsvarighet i modelleringshandboken. Den innebörd begreppen har kan däremot beskrivas på andra sätt i modelleringshandboken. I detta avsnitt beskrivs dessa alternativa modelleringstekniker .

Rollnamn
I OMT finns det möjlighet att ge respektive klass som deltar i en association ett rollnamn. Detta rollnamn anger vilken roll klassen spelar i relationen. Syftet med att använda rollnamn är att göra modellen klarare.

I modelleringshandboken anges ej prototypernas respektive roll i relationen. Relationens namn i kombination med dess riktning (visas med pil) ger samma betydelse och förståelse.

Tillståndsdiagram

I de objektorienterade metoderna kan en instans olika tillstånd uttryckas i ett tillståndsdiagram. I diagrammet visas klassens alla möjliga tillstånd och hur dessa står i relation till varandra, d v s i vilken sekvens de kan inträffa. Här anges t ex att ett tillstånd alltid måste föregå ett annat tillstånd. I vissa tillståndsdiagram kan dessutom anges vilka händelser som föranleder ett tillstånd och vilka operationer som då initieras och utförs.

Det som tillståndsdiagrammet uttrycker kan visas med det regelverk som finns i modelleringshandboken.

MHBs motsvarande regel är: Om ett läge (tillstånd) inträffar så utförs en viss operation. Denna regler motsvaras av OMTs tillståndsdiagram.

I avsnitt 4.3.4 (sid 39) visades ett tillståndsdiagram för ett schackparti. Nedan följer exempel på hur dessa tillstånd och händelser kan uttryckas med modelleringshandbokens regelverk.

När	det är vits tur att göra sitt drag och
om	vit kan flytta någon pjäs
så	flytta den pjäs som ger gynsammast läge
enligt	de regler som gäller för den aktuella pjäsen.
När	det är svarts tur att göra sitt drag och
om	svart inte kan flytta någon pjäs
så	är svart schackmatt

Schackpartiets tillståndsdiagram uttryckt enligt MHBs regelverk

Tillstånds-/tjänstetabell

Tillstånds-/tjänstetabellen uttrycker de operationer som kan utföras när ett objekt befinner sig i ett visst tillstånd. Tabellen listar endast möjliga operationer och möjliga tillstånd, inte de attribut som definierar tillståndet.

I modelleringshandboken kan motsvarande aspekt uttryckas i beslutstabeller. Dessa tabeller är mer omfattande. De innehåller inte tillstånd utan de enskilda villkor som definierar tillståndet.

	Tillstånd 1	Tillstånd 2	Tillstånd 3
Tjänst A	●		
Tjänst B			●
Tjänst C	●		
Tjänst D		●	
Tjänst E	●	●	●

OOAs tjänst-/tillståndstabell

Villkor 1	J	J	J	-	N	N	-	N
Villkor 2	-	N	J	N	J	N	J	-
Tjänst A	●			●				
Tjänst B			●			●		
Tjänst C	●						●	●
Tjänst D		●		●				

Beslutstabell

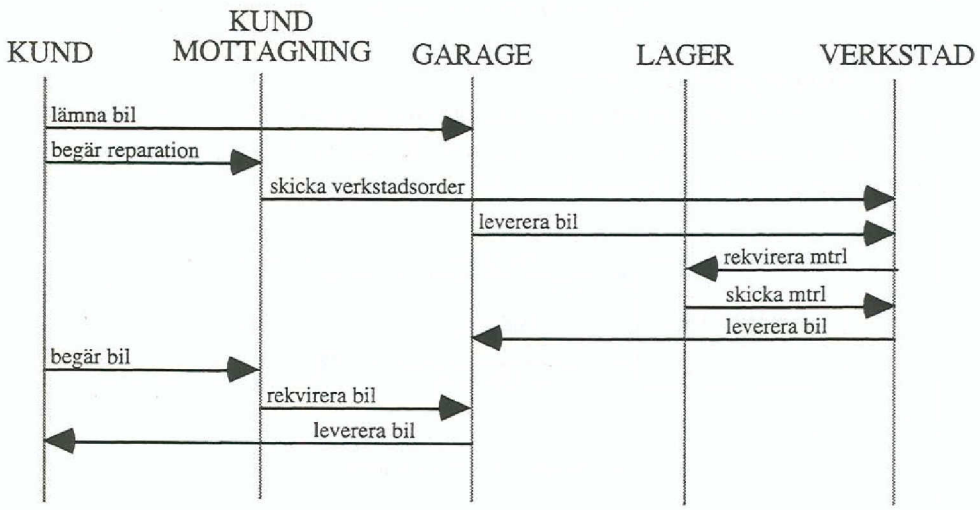
7.3 OO-begrepp som inte går att modellera i MHB

I detta avsnitt presenteras de begrepp och funktioner som finns i OO-ansatserna men inte i MHB. Vissa av dessa begrepp riktar sig mot datorvärlden, varför de inte har någon funktion i MHB. Det finns dock begrepp som skulle kunna användas i en verksamhetsanalys, trots att de i OO-metoderna uttrycker datorrelaterade företeelser.

Interaktionsdiagram

I OMT respektive ObjectOry kan sekvenser av kommunikation mellan klasser visas. I interaktionsdiagrammet visas händelser som inträffar i en klass. Tiden anges inte exakt, utan visar endast att en händelse inträffar före eller efter en annan. Med hjälp av dessa diagram kan en sekvens av kommunikation beskrivas.

På detta sätt kan den dynamiska kommunikationsstrukturen mellan arbetspunkter beskrivas. Ett sakflöde skulle kunna sättas i sitt sammanhang och därigenom beskriva tidsaspekten.

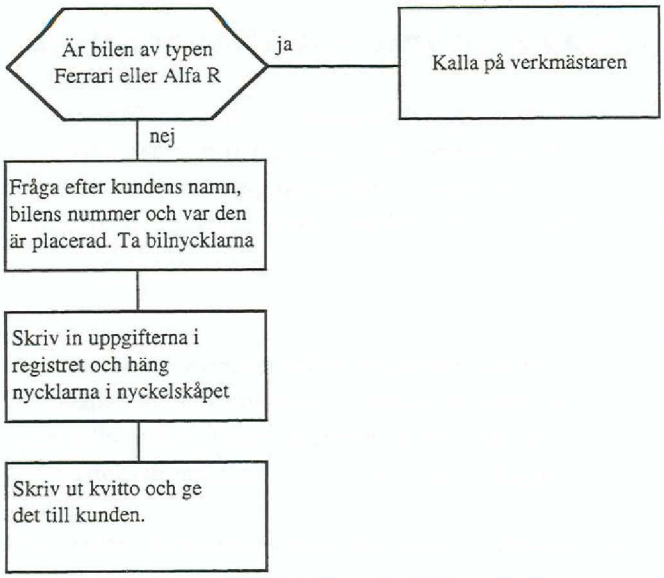


Interaktionsdiagram för reparation av bil

Tjänstediagram

Tjänstediagram används i OOA för att definiera tjänster. Detta diagram visar hur en operation ska utföras. I modelleringshandboken skulle arbetsuppgifterna i arbetspunkterna detaljerat kunna specificeras med tjänstediagram. Denna notation ökar detaljeringsgraden och preciserar i punktform vad som ska göras.

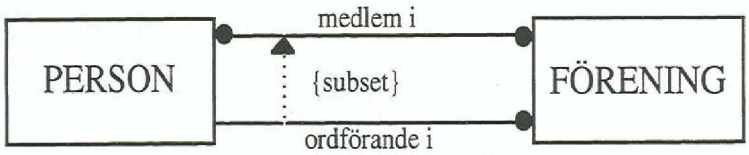
Flödesmodellen kan ses som ett tjänstediagram på en mycket hög abstraktionsnivå. Tjänstediagrammet tar inte hänsyn till den dynamiska aspekten, utan visar alla möjliga handlingsalternativ. En explicit körning modelleras inte, utan tjänstediagrammet visar mönstret för hur en operation kan köras.



Tjänstediagram för arbetsuppgiften "ta emot bil"

Restriktioner på associationer

I OMT kan restriktioner för en viss association uttryckas. Denna restriktion visar de möjliga deltagarna i en association. I modelleringshandboken finns idag ingen teknik för att uttrycka begränsningar på associationer.



Restriktion på en association

8. Litteraturförteckning

Avison, D E , Fitzgerald, G ?Information Systems Development", Blackwell Scientific Publications, 1988

Berild, S :"Objektorientering – de vanligaste begreppen", SISU, 1991

Coad, P, Yourdon, E :"Object Oriented Analysis" (2nd Edition), Yourdon Press, 1991

Elmasri, R, Shamkant, N :"Fundamentals of Database Systems", The Benjamin/Cummings Publishing Co., 1989

Fowler, M :"A Comparison of Object-Oriented Analysis and Design Methods", dokumentation från OOPSLA´92, 1992

Fowler, M :"A Comparative overview", Object Magazine, juli-augusti 1993

van den Goor, G, Hong, S, Brinkkemper, S :"A Comparison of Six Object-Oriented Analysis and Design Methods", University of Twente, 1992

Jacobsson, I, Christersson, M, Jonsson, P, Övergaard, G :"Object-Oriented Software Engineering", Addison-Wesley, 1992

Maleki, T, Hashemi-Nejad, N :"Evaluation of two major object-oriented methodologies and related tools", AVEMCO Corp., 1993

Objekctive Systems AB, "ObjectOry – Process" (version 3.3.0), 1992

Objekctive Systems AB, "ObjectOry – Guide" (version 3.3.0), 1992

Rumbaugh, J, Blaha, M, Premerlani, W, Eddy, F, Lorensen, W :"Object-Oriented Modeling and Design", Prentice-Hall, 1991

Shelton, R :"An Object-Oriented Method for Enterprise Modeling", Open Engineering, 1992

Sigfried, S :"Objektorientering, metoder & problemområden", Sveriges Verkstadsindustrier, 1992

Triad, "Modelleringshandboken", Rapport N 10, 1993

Wiktorin, L :"Erfarenheter av Objektorienterad Systemutveckling", Sveriges Verkstadsindustrier, 1992

Willars, H :"Arbetspapper", SISU, 1993

Ye, P, Hughes, J, McChesney, I, Glass, D :"An object-oriented conceptual model for requirements analysis", University of Ulster, 1993

TIDIGARE UTGIVNA PUBLIKATIONER AV TRIADGRUPPEN

Verksamhetskrav på informationsadministration

V 1: IA och verksamhetens krav – erfarenheter från offentlig förvaltning
V 2: Fallstudie av IA-projektet vidTeleverket
V 3: IA-erfarenheter från företag och myndigheter
V 4: Den gemensamma informationsmarknaden – en referensram för handlingsfrihet och konkurrenskraft

Modellering

N 1: Modelleringsansatser för begrepps- och datamodellering – Beskrivning och försök till jämförelse
N 2: Generering av konceptuella modeller från policydokument
N 3: Espritprojektet Tempora
N 4: Prövning av regelbaserad metodik inom Posten
N 5: En kokbok i remodellering – utkast
N 6: Datorstöd för modellintegration
N 7: Modellbaserad kunskapsinsamling
N 8: Modellkvalitet
N 9: Samband mellan dokument och modeller
N 10: Modelleringshandboken
1 – Översikt
2 – Modelleringsledarens bashandledning
3 – Modellering i grupp
4 – Kommunikation
5 – Arbetsgångar
6 – Modelleringsväskan
N 11: Ett+Ett=Ett – Två praktikers erfarenheter av modellintegrering

Kunskapsförmedling

H 1: Handledarutbildning för modelleringsledare, avancerad
H 2: Slutrapport HUMLA prototyp
H 3: Utbildning i Informationsadministration
H 4: Spridning av Hybris – en fallstudie vid Telia

Uttagssystem

U 1: Hybris i Unix-miljö
U 2: DEBRIS
U 3: Hybris DOS/PimWin på Posten
U 4: Program för sökning i databaser – en marknadsöversikt
U 5: Att nå och förstå data – möjligheter och begränsningar

Katalogprinciper

K 1: IRDS
K 2: IRDS Modeller och modellnivåer
K 3: Koppling begreppsmodell – relationsmodell
K 4: IBM:s Repository Manager – en Introduktion
K 5: IBM:s Repository Manager: Datamodelleringsbegreppen
K 6: IBM:s Repository Manager: Begreppsmodellering i Information Model
K 7: IBM Repository Manager: Attribut- och värdemodellering i Enterprise Submodel
K 8: Navigering i Repository
K 9: TRIAD Newsletter – IRDS inom ISO. Dagsläget
K 10: TRIAD Newsletter – ISO/IRDS. Händelseutvecklingen 91/92
K 11: Samverkan mellan resurskataloger – visioner eller behov
K 12: AD/Cycle 1 Information Model – Processer och och informationsflöden mellan processer
K 13: AD/Cycle 1 Information Model – Info Flows inom Processmodellen
K 14: AD/Cycle 1 Information Model – Relationsdatabasmodellering
K 15: AD/Cycle 1 Information Model – Härledningsspecifikationer i begreppsmodellen
K 16: IA-prototyp
K 17: Repository AD/Cycle – International Users Group
K 18: RAD-konferensen i Chicago, 1992
K 19: Vad händer inom ANSI-IRDS?
K 20: Information Warehouse – vad är det?
K 21: CDIF – en översikt
K 22: PCTE – en översikt
K 23: XLII – en öppen och flexibel utvecklingsmiljö

KORT OM TRIAD

Triad är namnet på ett treårigt samarbetsprojekt kring informations-administration och dataadministration, IA/DA, som Telia, Posten, Ericsson, Statskontoret och SISU bedriver. Syftet är att utveckla parternas synsätt, metoder och hjälpmedel inom detta område. Arbetet inom Triad är uppdelat i delprojekt som är sammanförda i tre block.

Beställarblocket vänder sig dels till dem som är verksamhetsansvariga och måste ta ställning till IA-/DA-satsningar, dels till dem som har ansvaret för IA/DA inom en organisation. Delprojekten inom detta block arbetar med att formulera verksamhetens krav på IA/DA samt studerar och beskriver roller, organisation och arbetsformer för IA-/DA-arbete.

Utförarblocket vänder sig till dem som arbetar med IA/DA. Delprojekten arbetar med modellering, data- och resurskataloger samt uttagssystem.

Kunskapsförmedling är det block som ser till att resultaten kommer Triad-parterna till godo. Detta sker bland annat genom kurser, seminarier samt genom att rapporter som denna ges ut.