Handelshögskolan vid
Göteborgs universitet
Institutionen för informatik

OBJEKTORIENTERAD DESIGN
- en studie om problem och lösningar

Per-Erik Gustafsson

Abstract
Rapportens utgångspunkt var att moderna informationssystems höga komplexitet och svårigheter med att identifiera objektklasser gör att metodutveckling inom konceptuell design är viktigt. Fokus i rapporten var frågan om hur en relevant uppsättning objektklasser kan identifieras. Behovet av metoder inom konceptuell design belystes i ett resonemang om en central fråga inom västerländsk filosofi, den om hur vi identifierar universella begrepp ur individuella objekt. Rapporten redogjorde för metoder ur litteratur och ur aktuella forskningsrapporter. En jämförelse av rapporterna visade på behovet av att ha ISO-modellens distinktion mellan objektsystem och informationssystem som utgångspunkt. Med ett par enkäter till personal på konsultbolag som arbetar med systemutveckling gjordes en avstämning av hur erfarna yrkesmänniskor uppfattar rapportens problemställningar. Rapporten drog slutsatsen att metodutveckling krävs för att möjliggöra bättre kvalitetssäkring av objektmodeller. De ansatser som beskrivs i de forskningsrapporter som rapporten tar upp utgör inte några nya metoder. Men dessa metoder har en viktig funktion att fylla genom att ge en vetenskaplig grund och vetenskapliga förklarningsmodeller för de mest utvecklade metoder som används idag. Det bidrar till att dessa metoder kan befästa sin position och bli mer allmänt använda. Detta kan leda till ökad möjlighet att värdera kvalitet i modelleringsarbetet, utvärdera arbetssätt och fortsätta ett metodutvecklingsarbete med sikte på att möjliggöra utveckling av informationssystem som är flexibla och av hög kvalitet.

Examensarbete, 10p
ADB-programmet 80p
Vårterminen 2001

Handledare
Roy Corneliusson

Innehållsförteckning

1. Introduktion

1.1 Inledning

”Det finns inga andra paradis än de förlorade”
(José Louis Borges)

Under de laborationer och studier i informationssystemutveckling jag har genomfört på institutionen för Informatik har objekt- och databasmodelleringen alltid inneburit stora utmaningar. Att ta fram eller tyda en kravspecifikation har varit jämförelsevis enkelt och även arbetet med att skriva källkoden för gränssnitt och logiken i systemet har varit en jämförelsevis lätt uppgift när väl designarbetet är klart. En fråga som därför funnits i mina tankar är huruvida metodutveckling inom området modellering kommer att underlätta arbetet med att identifiera en lämplig uppsättning objektklasser. Tyvärr är jag övertygad om att så är inte fallet (därav det inledande citatet). Objektmodelleringen kommer alltjämt att vara den svåraste fasen av systemutvecklingen. Just därför är metodutveckling inom konceptuell modellering ett angeläget arbete, inte minst mot bakgrund av de allt mer komplexa informationssystem som byggs idag. Kraven på systemutvecklingsarbetet ökar och nya arbetssätt är nödvändiga i arbetet med att skapa stabila system. Det positiva i sammanhanget är att det alltid kommer att behövas kompetens inom systemmodellering. Därför är detta en angelägen rapport.

1.2 Målsättning

Objektmodellering är ett av de mest avgörande momenten i systemutveckling[1]. Arbetet med att ta fram en relevant objektmodell har direkt betydelse för kvaliteten på det färdiga systemet samt och på hur väl systemet kan såväl integreras med andra system som utvecklas med förändringar i verksamheten. Dessutom är det en av de största utmaningarna. Att med kravspecifikationen, annan dokumentation, samtal med användare/beställare och andra hjälpmedel identifiera och definiera alla relevanta objekt samt relationerna dem emellan innehåller en mängd svårigheter varav de viktigaste handlar om;

[1] Se t.ex. Sommerville, 161

- *kategorisering av tingen omkring oss i objektklasser*
- *förståelse för tingens identitet*
- *kommunikation mellan utvecklare och användare/beställare*
- *kunskap om att bilden av verksamheten är tillräckligt bra*
- *kunskap om att vi identifierat en relevant uppsättning objektklasser*

Den kanske viktigaste anledningen till svårigheterna med objektmodelleringen är dynamiken mellan objekt och objektklasser. Vi uppfattar verkligheten som enskilda ting och att skapa generella klasser av mängden enskilda ting är en uppgift som kräver mycket tankemöda och tid i utvecklingsarbetet. Min målsättning med denna rapport är att bidra till en förståelse för kvalitetsaspekter vid konceptuell modellering i samband med utveckling av IT-system för verksamheters informationsförsörjning, både vad gäller att uppnå och att förvalta kvalitet. Ämnet är aktuellt då moderna IT-system är mer komplexa än någonsin och frågan om identifiering av en relevant objektmodell ställs på sin spets.

Uppsatsen har med andra ord två utgångspunkter för studiet av systemdesign;

Allt mer komplexa system får konsekvenser för den konceptuella designen
Moderna system präglas av hög komplexitet och att krav på nya sätt att hantera information växer fram samtidigt som datamängderna ökar lavinartat. Mathiassen skriver att tidigare utvecklades datasystem för att automatisera stora arbetskrävande informationsbehandlingsuppgifter, medan moderna informationssystem ofta handlar om att understödja sakbehandling, kommunikation och samordning[2]. Kraven på de metoder som används i systemutvecklingsarbetet ökar således och metodutveckling är därför nödvändigt[3]. En annan bidragande orsak är att många stora organisationer och verksamheter baserar sin verksamhet på ett antal olika system, vart och ett med sin egen databas, som har mer eller mindre grad av koppling till de övriga systemen. Även när olika aktörer (företag, organisationer) går samman uppstår samma situation, ett antal ”informationsöar” som behöver integreras för att uppnå ett effektivt informationsutnyttjande. Verksamhet är idag nästan synonymt med förändring. I alla sådana sammanhang finns problem bland annat med objektidentifiering och inkonsistens mellan databaserna[4].

Finns det några naturliga objektklasser?
Frågan om naturliga objektklasser har rent filosofiska kopplingar. Det handlar i grund och botten om vad vi egentligen kan veta om omvärlden och om hur vi skapar en bild av universella objektklasser ur den mängd individuella ting som vi ser omkring oss. Den filosofiska diskussionen sedan Platon har presenterat olika förklaringsmodeller över verklighetens beskaffenhet och om hur vi uppfattar ting och klasser. Denna diskussion ger en bra förståelse för de problem vi ställs inför under den konceptuella designen. Eftersom det inte finns några naturliga objektklasser så finns det inte heller någon objektmodell för ett givet IT-system som är den ”rätta”. Det går att modellera varje system på en rad olika sätt. Därför är det viktigt att kunna veta att modellen är relevant och tillräckligt bra för att möjliggöra ett stabilt system som dessutom går att förändra vid till exempel verksamhetsförändringar eller nya önskemål om funktionalitet.

[2] Mathiassen, 18
[3] Wang & Madnick; Mylopoulos
[4] Wang & Madnick

1.3 Rapportens frågeställningar och avgränsningar

Objektmodelleringen är allmänt beskriven och finns i olika versioner i litteraturen. Den är med andra ord en vedertagen arbetsmetod. Jag kommer hädanefter att använda den vokabulär som Connolly[5] använder;

”Konceptuell design” beskriver den fas som jag inledningsvis kallade objektmodellering. Under denna fas utvecklas en modell över de ingående objektklasserna och relationerna mellan dem i den del av verkligheten som det tänkta datasystemet skall administrera eller styra. En ”konceptuell modell” är den modell vi får som resultat.

”Logisk design” beskriver på motsvarande sätt den efterföljande fasen, där den konceptuella modellen skall översättas till de tabeller som skall ingå i databasen. I modellen skall det finnas detaljerad information om alla de i databasen ingående tabellerna. I princip kommer varje objektklass att bli en tabell, men då den konceptuella modellen också skall normaliseras för att skapa en stabil och konsistent databasmodell så tillförs en del nya tabeller och en del objektklasser styckas upp för att representeras i flera tabeller. Målet är en modell utan redundans. Vilken normaliseringsgrad som är eftersträvansvärd beror på de krav som ställs på systemet, men tredje normalformen[6] är en lämplig nivå för de allra flesta system. En ”logisk modell” är resultatet av den här fasen.

”Fysisk design” som är den fas då den logiska modellen skall implementeras i en fysisk miljö. Här fattas beslut om den specifika implementeringen av databasen. Det gäller att välja lagringsstruktur för de ingående tabellerna, vilket i hög grad bestäms av den databasprodukt som används, hur och när olika index skall skapas samt om och hur modellen skall denormaliseras för att införa redundans[7].

Arbetet med den konceptuella designen är i fokus för denna rapport. Dels för att denna fas utgör en så avgörande del i systemutvecklingen men också för att de metoder och teorier som används i det arbetet utsätts för påfrestningar i takt med att de krav som ställs på informationssystemen ökar.

Jag har medvetet valt ett brett perspektiv för att kunna studera problemområdet i sin helhet, få en samlad bild och urskilja de viktiga dragen i de överväganden och arbetssätt som krävs för att utveckla informationssystem med kvalitet. Då denna kvalitet i hög grad avgörs av kvaliteten på objektmodellen är det nödvändigt som systemutvecklare att ha bred kompetens på det området.

---

[5] Connolly, 227

[6] Normalisering är en metod för att arbeta bort redundans ur moellen. Normaliseringen görs i olika steg för att gradvis öka kvaliteten på modellen. De första och andra normalformerna säger att varje enskilt attribut skall vara odelbart (skilj på namn och efternamn t.ex.) och att varje attribut skall vara beroende av den primära identifieraren. Den tredje normalformen att inget attribut får vara beroende av identifieraren via något annat attribut.

[7] Denormalisering kan till exempel övervägas om systemet inte klarar av uppställda kapacitetskrav. I sådana fall kan tabeller som uppdateras sällan men efterfrågas ofta denormaliseras. Connolly, 287.

Huvudfrågan är kort och gott;

> **Hur vet vi att den objektmodell vi designar är relevant** och leder till ett informationssystem som både är stabilt och förändringsbart vid framtida förändringar i verksamheten?

Bakom huvudfrågan finns frågeställningar som;

Vari ligger svårigheterna med det konceptuella designarbetet?
Vilka metoder och verktyg behövs för att designa stabila och effektiva system?

1.4 Metod, material

Detta är en teoretisk studie. De kurser jag deltagit i på Institutionen för informatik har tagit upp system- och databasteori och praktiska laborationer har inneburit att jag ställts inför svårigheterna med modelleringen.

Litteraturstudier skall här fördjupa och konkretisera några problemställningar samt visa på ansatser för förbättring av utvecklingsmetoder.

Jag har också för avsikt att genomföra ett par intervjuer/enkäter med personal i näringslivet som arbetar med systemdesign för att få jämförande material. Jag har inte för avsikt att genomföra intervjuer i en sådan omfattning att resultatet blir statistiskt säkerställt eftersom tiden är för kort. För att göra en sådan intervjuinsats skulle uppsatsen i stort sett ägnas uteslutande åt intervjuer på bekostnad av teori. Jag är övertygad om att den förlusten av teoriinnehåll skulle ge ett sämre resultat. Därför inriktar jag mig på några få intervjuer för att ”stämma av” teorierna.

Min arbetsmetod mer specifikt är att ha en tydlig avgränsning för att i litteraturen kunna välja ut det som har relevans. Jag har försökt skapa mig en klar bild av problem och delproblem för att kunna arbeta på ett systematiskt sätt.

Förutom de avgränsningar jag beskrivit ovan avseende själva designprocessen så avgränsas rapporten också enligt följande.

Utmaningarna i arbetet med att identifiera en relevant uppsättning objekt är rapportens fokus. Studier av rena notationstekniker som till exempel UML[8] eller hela projektmetoder som RUP[9] ingår inte i rapporten.

Jag är alltså i första hand ute efter att förstå och fördjupa mig i konceptuell design och de utmaningar som finns i detta i en tid med allt mer komplexa system och med nya krav på informationshantering. Inom konceptuell design så är det också den första frågan om att hitta en relevant uppsättning objektklasser som är rapportens fokus.

[8] Unified Modeling Language – en notationsteknik för objektmodellering, framtagen av företaget Rational Rose.
[9] Rational Unified Process – en systemutvecklingsmetod framtagen av företaget Rational Rose.

Frågor om att identifiera beteendemönster, attribut och relationer mellan klasserna är därmed inte i rapportens fokus.

1.5 Rapportens disposition

Kapitel 1 ger en bakgrund till rapporten. Jag beskriver mina motiv för att behandla det valda ämnet, beskriver rapportens målsättning, avgränsningar och frågeställningar. Metodval och material beskrivs också här.

Kapitel 2 behandlar den del av litteraturstudiera som behandlar det övergripande teoretiska ramverket för rapporten. Här beskrivs en bakgrund till svårigheterna med objektmodellering där en filsofisk diskssion sedan Platon är utgångspunkten. I detta kapitel redogörs också för de båda ansatserna ANSI-SPARC och ISO-modellen.

Kapitel 3 är del två av rapportens teoretiska genomgång. Här redogörs i detalj för objektorienteringens grundläggande begrepp och principer. Frågan om vad ett objekt är och hur det avgränsas samt frågan om metoder för objektmodellering behandlas.

Kapitel 4 presenterar en enkät som skickades till tre konsuler som arbetar med systemutveckling. Enkätsvaren presenteras i sin helhet och dessutom finns en bearbetning av svaren i tabellform. Denna tabell utgör rapportens empiriska resultat.

Kapitel 5 utgör rapportens sammanfattande kapitel med diskussion och slutsatser.

2. Teoretisering: grundläggande ramverk

”Å gubevars för dumt fôlk!
Att en annan, sôm inga lärdom har, int allti ä sôm en ska, d´ä inga unner,
men når di, som ska var för mer, int ä klar i bokstaveringskônsta en gang,
d´ä lett. Te äxämpel dä va en söndagsmôra, sôm ja geck å spanklér på
landsvägen. Da kommer dä en harrkär gånass mä glasyger för yga
å smal å vesen va´n, tocken sôm tocker ä.
”Gudagen, min goda man”, sa´n. Vet han, vart ja ska gå för te komma te
prästgårn”, töckte´n.
”Håja, nock vet ja dä”, sa ja. ”D´ä inga kônst te komm dit. Når han gått
ett litt stöck länger, komer han te en väg te vänster, men den ska´n
int gå, utta bar fortsätt tess han kommer te näst vägskel, där dä ta å te höger,
där ska´n gå.”
”Mä se”, sa harrkärsillänne å såg fundersam ut änna sôm dä int
skull var klar grejer, ”Mä se, inte te vänster, men te höger.
Nå vidare da.”
”Jo, når´n da ha gått ett stöck te, så kommer´n te e å, å i åa ä e ö.”
”Vasa”, sa´n.
”Å i åa ä e ö”, sa ja.
”Men va i all ti ä dä ni säjer, a, o?” sa´n.
”D´ä e å, vett ja”, skrek ja, för ja ble rasen, ”å i åa ä e ö, hörer han lite
d´ä e å, å i åa ä e ö.”
”A, o, ö”, sa´n å dämme geckén.
Jo, den va nôe te dum den.
(Dumt fôlk, Gustav Fröding)

Konceptuell design utförs till största delen i en dialog mellan utvecklare och användare/beställare. Så, det gäller ju att parterna förstår varandra.

2.1 Definitioner och utgångspunkter

Rapporten föreutsätter att läsaren har grundläggande kunskaper om systemutveckling och objektorientering och alltså förstår vad systemvetenskapen menar med informationssystem och kan det viktigaste angående klasser, objekt och systemdesign. Men det viktigaste förklaras ändå i texten.

Några utgångspunkter förklaras här.

Vad betyder begreppet ”metod”?
Jag utgår ifrån de resonemang om metoder, processer och notationer som Mathiassen[10] för. En metod definieras som en samling föreskrifter som instruerar i hur arbetet skall utföras. Därmed dras en skiljelinje mellan metod och notation. Exempelvis är UML, som har blivit något av en standard, en notationsteknik och ingen metod för objektmodellering.

Filosofi
De filosofiska resonemangens relevans för ämnet utgår ifrån en stor diskussion inom filosofin, den om hur vi kategoriserar universella begrepp ur individuella ting. Den diskussionen är en direkt spegel av den konceptuella designens uppgift att identifiera objektklasser ur den mängd objekt som finns i ett tänkt systems problemdomän.

Vad är en modell?
En förenklad bild av verkligheten, ett mönster eller en abstraktion. Modellen skall vara fokuserad på det väsentliga. Ett sätt att skapa begriplighet och enighet om systemet som skall utvecklas.

2.2 Objektmodellering

2.2.1 Filosofisk bakgrund

Artz[11] för en filosofisk diskussion om klassbegreppet. Frågan om huruvida objektklasser existerar eller inte bearbetar han med hjälp av de tankar om hur vår kunskap om omvärlden ser ut som formulerats av ett par av de centrala europeiska filosofiska tänkarna;

Platons (429-347 f.kr) tankar om att det finns en så kallad ”sinnevärld” är utgångspunkten. Här finns idén att det finns en sinnevärld oberoende av människans tankar. I sinnevärlden existerar en form av ”mallar” för de ting människor uppfattar i verkligheten. Sinnevärldens mallar är vad vi skulle kalla objektklasser. Utifrån Platons filosofi finns det alltså objektklasser i verkligheten.

Aristoteles (384-322 f.kr) menade att en sådan sinnevärld inte existerade. Det enda som existerar är det vi kan uppfatta omkring oss. Varje objekt har en uppsättning egenskaper som vi kan uppfatta. Objektklasser existerar som samlingar av egenskaper. Vi kan alltså se objekt och gruppera dem i klasser.

Både Platon och Artistoteles var övertygade om att det vi kallar objektklasser existerade oberoende av människans sätt att uppfatta världen. De hade däremot olika uppfattning om deras natur.

[10] Mathiassen, 14-15
[11] Artz, 25-30

Med de brittiska empiristerna formulerades långt senare nya insikter i det mänskliga sinnets betydelse när det gäller att skapa objektklasser och idén om naturligt existerande objektklasser övergavs.

Locke (1632-1704) förkastade den aristoteliska bilden av att objektklasser faktiskt existerar. Hen menade att det är människan som, genom en abstraktionsprocess, avgör hur verkligheten skall kategoriseras. Likheter och skillnader finns där, det är vi själva som använder detta för att gruppera objekten i klasser. Enligt Locke var det med andra ord människans aktiva tänkande som formulerade klasserna.

Hume (1711-1776) formulerade ytterligare insikter i hur människans omvärldsuppfattning fungerar. Enligt honom så formuleras klasserna inte bara genom människans aktiva och medvetna tankeverksamhet. Det är också en undermedveten process, som påverkas av bland annat kulturell förförståelse, uppfostran, kunskaper. Det är med andra ord en process som i hög grad är ganska grumlig för oss.

Den Platonska filosofin är inte någon bra utgångspunkt för konceptuell design. Med Aristoteles synsätt har vi kommit bra mycket närmare modern design. Vi betraktar verkligheten och bestämmer objektklasser utifrån gemensamma attribut. Detta går förhållandevis bra så länge det handlar om välkända objekt. Men när det handlar om en mer konstgjord värld av information där klasser behöver definieras utifrån en organisations verksamhet (ofta stadd i förändring) så blir det svårare att acceptera utgångspunkten att objektklasser existerar i verkligheten. Det ligger närmare till hands att utgå från att objekt av mer eller mindre abstrakt karaktär och som kanske inte existerar i fysisk mening utan representerar händelser eller överenskommelser kan definieras på en oändlig mängd sätt. Dessutom är en klass inte relevant enbart på grund av en uppsättning väldefinierade attribut utan också beroende av vilken relation till organisationen och till det tänkta informationssystemet klassen har.

Om **Lockes** tankar får vara utgångspunkten för systemutveckling så innebär det att det finns **många sätt att modellera ett system**. Om det är vi själva som avgör hur objekten skall delas upp i klasser så måste modelleringen vara starkt beroende av vad systemet skall användas till. De egenskaper och attribut vi tillskriver objektklasserna är de som är viktiga för tillfället.

Humes tankar för oss till en annan svårighet. I hans filosofi är varje definition av en klass beroende av en **undermedveten kategoriseringsprocess**. Om klasser existerar i användarnas sinnen utan att vara explicit definierade så blir **kommunikationen mellan användare och utvecklare oerhört viktig**. Det är då oerhört viktigt att användare och utvecklare arbetar fram en gemensam syn på begrepp och definitioner.

Connolly[12] tar upp problemet med att deltagare i designsamtal definierar sin omgivning i allmänna termer. Till exempel så är begreppet PERSONAL inte något som används till vardags. I stället är det vanligt att antingen prata om uppgifter som skall utföras eller specifika namn på dem som utför dessa uppgifter. Dessutom används gärna synonymer och homonymer[13]. Mathiassen nämner problemet med att organisationer ofta präglas av existerande traditioner för hur begrepp används[14]. I ett

[12] Connolly, 231
[13] Synonym = två ord som betyder samma sak. Homonym = ett ord som har olika betydelse i olika sammanhang.
[14] Mathiassen, 75

utbildningsföretag används kanske begreppet ”kurs” för att beskriva olika aktiviteter. Men om ett datasystem för administration av detta skall utvecklas måste vi vara mer exakta i definitionen för att se om det t.ex. är skillnad på kurser och seminarier. Det kan ju finnas viktiga skillnader i examinationsformer och villkor för deltagande. Ibland är det också svårt att veta vad som är objekt, relationer eller attribut. Det kan också vara svårt att veta vilka klasser som är relevanta.

Det är då **viktigt att utveckla en gemensam syn på verksamheten och objektklasserna**. Den konceptuella designen måste då till stor del innehålla rent språkliga överväganden för att säkerställa så långt det är möjligt att alla definitioner och begrepp har samma innebörd för de som deltar i utvecklingsarbetet. Ett enkelt exempel är ordet ’äktenskap’[15]. Det kan symbolisera både en objektklass, en objektegenskap eller en relation. Situationen avgör vad som är rätt.

I den konceptuella designen har vi ett antal verktyg till förfogande. Mekanismer som attribut, metoder och arv är grundläggande element i objektorienteringen och hjälper oss att i detalj beskriva de objektklasser vi identifierat. Men, menar Wang[16], de erbjuder ingen hjälp i den mest komplicerade frågan, den att identifiera en relevant uppsättning objektklasser. Wang skriver att i detta avseende är **objektorienteringen än så länge outvecklad**, det **finns ingen riktig metod för att lösa den största utmaningen** i systemutvecklingen. Med det menas inte att en universell metod efterlyses, en kombination av metoder är nödvändigt i de allra flesta fall. Men de metoder som finns att arbeta med räcker inte till utan förbättringar krävs.

2.2.2 Nya krav

Svårigheterna accentueras också av att IT-systemen blir allt mer komplexa och att nya krav på systemens funktionalitet växer fram, på grund av att IT används i allt fler sammanhang i samhället. Idag är IT-stöd ett självklart inslag i de flesta verksamheter. Det blir allt mer vanligt att olika system, självständigt utvecklade skall integreras. Där finns problem med objektidentifiering över skilda databaser. Vad är identiteten, hur garanterar vi konsistens och integritet? I skilda databaser kan finnas objekt som gäller samma verkliga objekt men definierade och identifierade på olika sätt.

Att bedriva verksamhet idag är synonymt med förändringar. I modern systemutveckling är det viktigt att inse att det inte finns någon självklart ”korrekt” objektmodell. Om dessutom ett nytt system byggs för att stödja nya arbetssätt och rutiner så är det uppenbart att kraven på objektmodellen framför allt är att den skall vara relevant, möjliggöra ett stabilt system och inte hindra framtida eventuella kompletteringar och förändringar. Ett misslyckat modelleringsarbete på de punkterna innebär framför allt en stor risk att den databas som implementeras med modellen som grund blir svår att anpassa till en förändrad verksamhet.

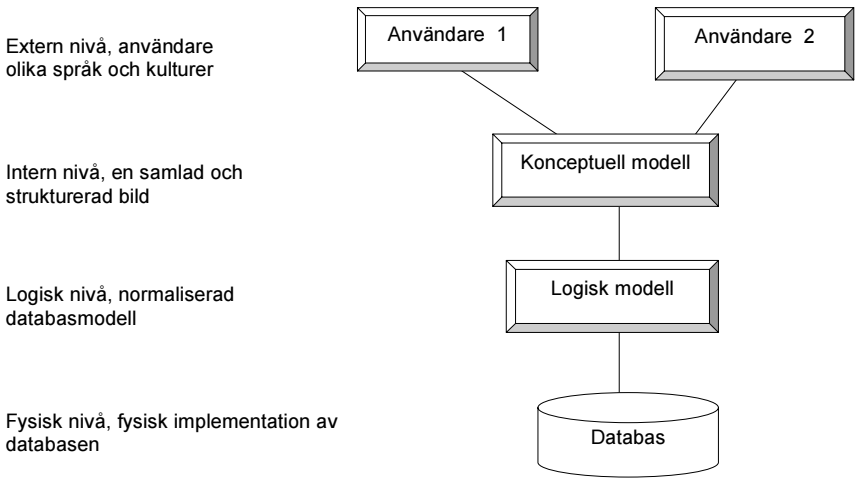
[15] Connolly, 232
[16] Wang, 305

2.2.3 Ansatser för att möta kraven : ANSI-SPARC och ISO-modellen

Hur den konceptuella designen förhåller sig till logisk och fysisk design beskrivs tidigare i rapporten (sid 3). Bild 1 visar detta på ett mer schematiskt sätt med hjälp av en förenklad bild av ANSI-SPARC:s[17] trelagersmodell. Det är avsett framför allt för tydlighetens skull. Denna rapports huvudfråga berör designarbetet som utförs på den konceptuella nivån.

Bilden visar att på den externa nivån finns olika användargruppers perspektiv på verksamheten. Under det designarbetet på den konceptuella nivån som görs av i samarbete med användarna skapas den konceptuella modellen. Designarbetet på den logiska nivån har den konceptuella modellen som ”indata” och det som skapas är en logisk modell, som visar alla tabeller som skall ingå i databasen, dock helt oberoende av fysiska faktorer som vald databasprodukt eller serverkonfiguration. På den fysiska nivån slutligen byggs en fysisk modell som visar exakt hur databasen skall implementeras i den fysiska miljön där den skall installeras. För den här rapporten är det framför allt dynamiken mellan den externa den interna nivån som är intressant. De verklighetsuppfattningar och begrepp som olika användargrupper har måste översättas till en gemensam och strukturerad modell.

Bild 1. ANSI_SPARC:s trelagersmodell



ANSI-SPARC:s modell i detta sammanhang utgör framför allt ett sätt att beskriva frågan om språkets betydelse, om att samordna en mängd verklighetsuppfattningar. Modellen ISO –82 (Bild 2), speglar på ett mer dynamiskt sätt vikten av en tydlig struktur i modelleringsarbetet. Modellen beskriver sambanden och skillnaderna

[17] ANSI-SPARC (ANSI = American National Standards Institute SPARC = Standards Planning and Requirements Committee) tog i mitten av 70-talet fram en trelagersmodell för databasscheman. ANSI-SPARC blev aldrig någon internationell standard, men fungerar utmärkt för att visa modellernas inbördes plats.

mellan ett tänkt informationssystem, dess användare och den verklighet systemet avser. ISO-modellens struktur är ett bra sätt att visa var denna rapport fokuserar på.

Bild 2. ISO -modellen

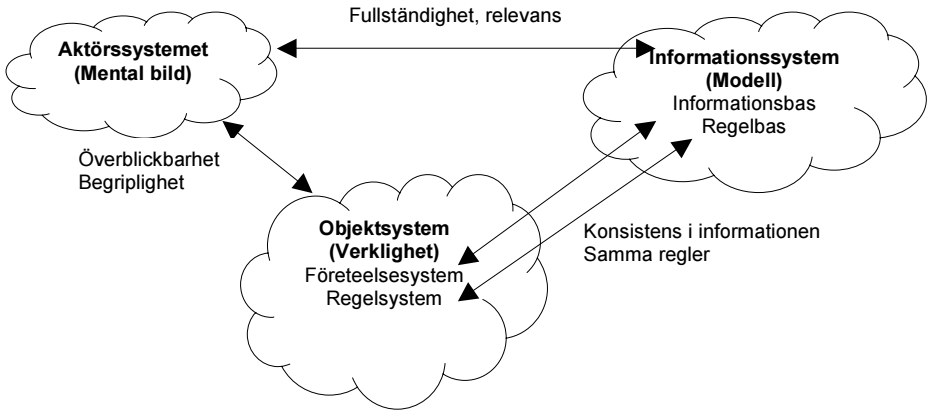
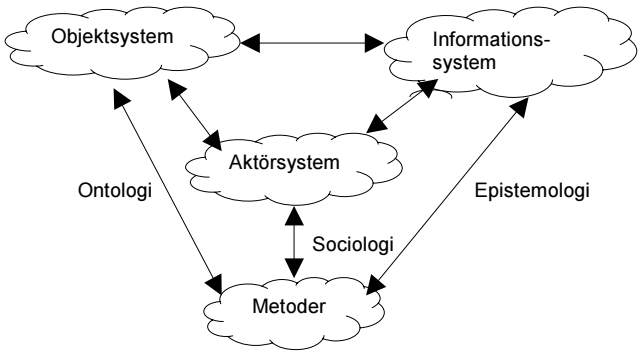


Bild 2 visar hur aktörssystemet, objektsystemet och informationssystemet förhåller sig till varandra. Aktörssystemet utgörs av både systemägare, användare och systemutvecklare. Tillsammans (i olika konstellationer) skapar dessa en mental bild av objektsystemet. Denna mentala bild måste vara gemensam för aktörerna, dvs att både användare, ägare och utvecklare skall ha samma bild av objektsystemet. Objektsystemet i sig självt är den del av verkligheten som informationssystemet skall betjäna. Informationssystemet skall präglas av relevans och fullständighet. Det vill säga att all relevant information om objektsystemet skall finnas med. Det betyder i sin tur att i förhållandet mellan informationssystem och objektsystem har vi konsistens (dvs all nödvändig data, utformad i överensstämmelse med önskvärda kvalitetskrav), samt samma regler vilket innebär att de regler som gäller i objektsystemet också återfinns i informationssystemet.

Den här rapporten fokuserar på behovet av metodutveckling inom objektmodellering, det vill säga den del av systemutvecklingen där aktörerna skapar en gemensam mental bild av verkligheten. Även inom den ramen fokuserar rapporten på en del av metodarbetet. Bild 3 visar ISO-modellen kompletterad med en ”dimension” där de olika aspekterna på ’metod’ förtydligas.

Bild 3. Kompletterad ISO-modell



I förhållande till objektsystemet är metoderna i hög grad frågan om ontologiska överväganden på så sätt att i objektmodelleringen görs olika antaganden om verkligheten. Dessa antaganden måste valideras på ett metodiskt sätt. I denna dynamik är ISO-modellen en grund för att förstå distinktionen mellan objekten i objektsystemet och de objekt vi modellerar i informationssystemet. I förhållande till aktörssystemet är metoderna mer av sociologisk karaktär. Det innebär att sociologiska metoder ligger nära till hands när samspelet mellan beställare, användare och systemutvecklare är i fokus. I förhållande till informationssystemet, till sist, är det aktuellt med metoder av mer tekniskt slag. Det är här som ANSI-SPARC positionerar sig. Det är de två första förhållandena som denna rapport fokuserar på, det vill säga metoder i förhållande till aktörssystemet och till objektsystemet.

2.2.4 Utveckling av ansatserna : Shouhong Wang och ANSI-SPARC

Wang[18] är ute efter att strukturera upp arbetet med konceptuell modellering utifrån olika perspektiv på den verksamhet informationssystemet skall handla om. Syftet är att på ett metodiskt sätt få en modell som täcker många aspekter på de ingående processerna och som därmed skall bli ett stabilt system. Han delar upp arbetet i fyra delar;

- *Verksamhetens processer*
- *Verksamhetens mål*
- *Verksamhetens aktörer*
- *Klient/server-beskrivningar*

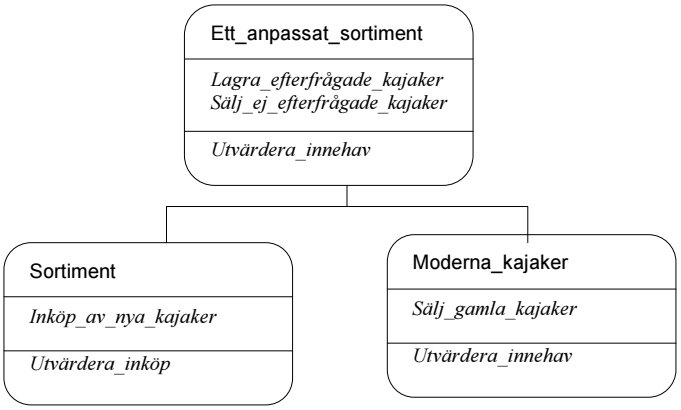
Processbeskrivningarna
Kan enligt Wang göras med tre fundamentala typer av objekt; **physiomorfiska** (fysiskt existerande) objekt, objekt som representerar **händelser** och objekt som representerar **dokument.** Med denna uppdelning kan processerna beskrivas utifrån deras ”uppförande”, funktioner och informationsstruktur. Tyvärr visar Wang i ett exempel på hur han modellerar att det är oerhört lätt att missa distinktionen mellan information och objekt. Hans objekt av dokumenttyp är i hans exempel snarast att hänföra till information snarare än till verkliga objekt. Därmed arbetar Wang mer enligt ANSI-SPARC än ISO-modellen. I avsnitt 2.3 och 2.4 visar jag konsekvenserna av att modellera enligt Wangs sätt jämfört med en mer objektorienterad ansats.

Målbeskrivningar
När det gäller målbeskrivningar har Wangs beskrivning ett mer gediget innehåll. Objektorienteringen utgår ifrån en systembeskrivning där mål och delmål ofta finns med. Därför skall dessa mål vara objektklasser. Attribut till en målsättning kan vara strategier för att uppnå målet, metoder kan vara sätt att utvärdera måluppfyllelsen. Ett enkelt exempel är följande modell (Bild 4) över ett par målsättningar för en kajakuthyrare. Uthyrningens övergripande mål består av att ha kajaker i sortimentet av de typer som faktiskt efterfrågas av kunderna. Strategier för detta är att sälja kajaker som inte varit efterfrågade och att ha efterfrågade i lager. Som delmål finns också att ha relativt nya och säkra kajakmodeller för uthyrningen samt att regelbundet

[18] Wang, 303

uppdatera sortimentet. Strategin för att uppdatera sortimentet är att beställa nya kajaker och strategin för att ha nya och säkra kajaker är att sälja alla som är äldre än 2 år.

Bild4. Modellering av målsättningar



Aktörsbeskrivningar
Aktörer är de människor som agerar mot systemet. Utifrån ett organisatoriskt perspektiv kan attributen vara information om ansvar och befogenheter. Ur ett mer tekniskt perspektiv finns information om kompetens. Till sist finns också attribut som visar hur aktören agerar i användningen av systemet.
Aktörsperspektivet skiljer sig en del ifrån en physiomorfisk beskrivning av människor, vilket också låter sig göras. Aktörsperspektivet ser människan mer utifrån dennes roller i organisationen medan det physiomorfiska perspektivet betonar information om människan. Wang ger ett exempel;

Om ANSTÄLLD är ett physiomorfiskt objekt i ett system så ger det information om varje anställd. Om dessutom TJÄNSTEMAN är ett aktörsobjekt så beskriver det människans roller och ansvar i det dagliga arbetet. Det är alltså två olika uppsättningar av information.

Detta aktörsperspektiv, menar Wang, ger möjlighet att bygga system som passar dagens organisationer. De traditionella hierarkiska organisationsbeskrivningarna speglar inte riktigt de arbetssätt som utvecklas idag. Det blir allt mer vanligt med tillfälliga eller permanenta nätverk av personer kring delar av processflödena snarare än utifrån avdelningsstrukturer. Det är ett mer flexibelt och föränderligt arbetssätt som aktörsperspektivet klarar att fånga upp.

Klient- och serverbeskrivningar
I moderna system blir det mer och mer angeläget att identifiera dessa typer av objekt. Klient-serverlösningar handlar mycket om automatisering och nätverkslösningar. En konceptuell modell för ett nätverkssystem måste därför innehålla beskrivningar av klienter och servrar. Både klienter och servrar kan beskrivas utifrån adress, hårdvara och mjukvara. Gränssnittsklasser kan rymmas inom detta också.

2.2.5 Utveckling av ansatserna : John Mylopoulos och ISO-modellen

Mylopoulos har liksom Wang tagit ett större grepp om metodtänkandet. Till skillnad mot Wang är Mylopoulos väldigt tydlig med distinktionen mellan objektsystem och informationssystem. Hans utgångspunkt är mer de filosofiska aspekterna på konceptuell modellering. Hans teorier utgår ifrån att människans språk inte riktigt räcker till för att vi skall kunna identifiera en relevant uppsättning objektklasser i sammanhang där vi rör oss i den konstgjorda världen som en organisations komplexa verksamhets- och informationsstrukturer består av. Orsaken till detta är att språket inte är tillräckligt exakt och rymmer många tillfällen till tvetydiga definitioner och missförstånd. Därför behövs någon form av metod för att värdera en konceptuell modell. Mylopoulos drar upp en struktur som utgår ifrån tre dimensioner;

Ontologier	Varje konceptuell modell gör antaganden om verkligheten (ontologier) som skall modelleras.
Abstraktionsmekanismer	Avgör organiseringen av informationsbasen.
Verktyg	Ett system som skall finnas länge måste kunna underhålla själva informationsstrukturen.

Med en liknande strategi som Wang skall en sådan struktur syfta till att den konceptuella modellen kan testas så att de verkligen täcker in alla relevanta aspekter på en verksamhet. Det som Mylopoulos hänför till verktyg berör jag inte i denna rapport då det handlar om dokument- och kodgeneratorer och liknande instrument som ligger utanför denna rapports avgränsning.

2.2.5.1 Ontologier - antaganden om verkligheten

Ontologi handlar om sådant som vi antar existerar, antaganden om verkligheten. Mylopoulos identifierar fyra ontologiska perspektiv som en modell måste stödja;

Statisk ontologi
Detta är ting som existerar. Till skillnad från Wang så visar Mylopoulos att detta inte är helt självklart. Exempel är objekt som kan blandas med varandra och där blandningen resulterar i ett helt annat objekt (t.ex. inom kemin). Det är också möjligt att skilja på fysiskt existerande ting, icke-existerande objekt, abstrakta objekt samt objekt som omöjligt kan ha en existens.

Dynamisk ontologi
Handlar om dynamisk information om systemet i termer av objekttillstånd, transformationer och processer.

Syftebeskrivande ontologi
Beskriver aktörer, uppgifter och intentioner. Det handlar alltså om agenter som kan ha uppfattningar, övertygelser och är kapabla att genomföra aktiviteter. I

kravmodellering har man länge modellerat agenter, speciellt i situationer som innehåller samtidiga aktiviteter.

Social ontologi
Den här ontologin omfattar social miljö, organisationsstruktur, allianser och beroenden. När en objektmodell granskas utifrån detta perspektiv skall den kunna svara på frågor om hur organisationens sociala miljöer ser ut. På ett liknande sätt som Wangs aktörsperspektiv kan detta bidra till att kartlägga tillfälliga nätverk och organisationsformer kring olika projekts arbetsflöden.

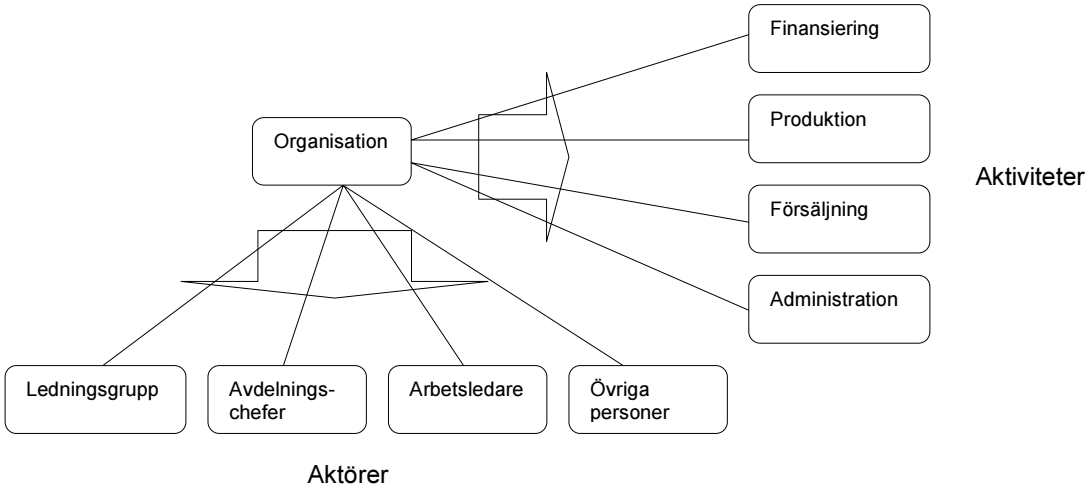
2.2.5.2 Abstraktionsmekanismer – grundläggande objektorientering

Detta handlar väldigt mycket om de grundläggande mekanismerna i objektorientering. **Klassificering, generalisering, aggregering** bör vara kända begrepp för läsarna av denna rapport. Några aspekter som kan var värda att notera är;

Det kan vara viktigt att när en objektklass bryts ner till aggregerade delar kan det göras på många sätt. Det räcker inte med att göra en första sådan nedbrytning och tro att det viktiga är infångat med det. Bild 5 visar ett exempel som Mylopoulos ger.

Här är det alltså möjligt att se organisationen antingen ur ett ledningsperspektiv (den nedåtriktade pilen) eller ett administrativt perspektiv. Kanske behövs båda i systemet. Dels skall IT-systemet fungera som beslutstödssystem där t.ex. ekonomiska data för de olika avdelningarna ’Finansiering’, ’Produktion’, ’Försäljning’ och ’Administration’ skall kunna presenteras. Dels skall strategier kring personalfrågor kunna hanteras genom information om personer och kompetens inom de olika yrkeskategorierna ’Verkställande ledning’, ’Avdelningschefer’, ’Arbetsledare’ och ’Övrig personal’.

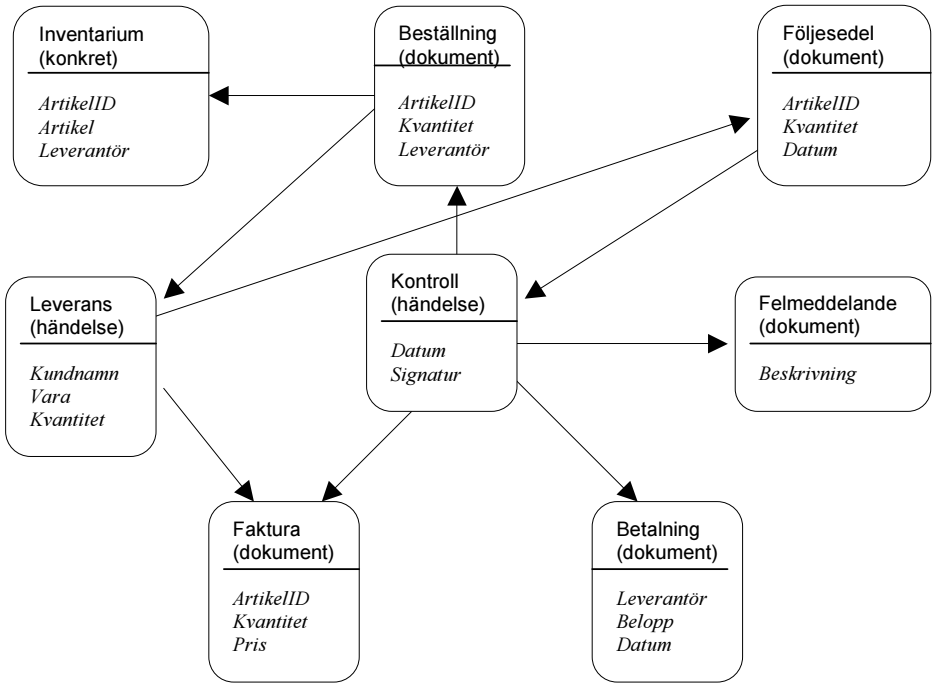
Bild 5. Aggregering i två dimensioner



2.3 Exemplifiering : modellering enligt ANSI-SPARC

Wang ger ett enkelt exempel (Bild 6), som är hämtat från en del av ett betalningssystem för Ford[19]. Exemplet visar en inköpsprocess som går till så här;

Bild 6. Delbeskrivning av betalningssystem hos Ford (från Wang).



Vid inköp tas en kopia på ”Beställningen” och när leveransen kommer tas en kopia på ”Följesedeln”. Dessa dokument sänds till den som ansvarar för att göra utbetalningen. Den personen får också ”Fakturan” från leverantören. Då görs en ”Kontroll” av att de tre dokumenten stämmer med varandra. Om allt är OK så görs en ”Betalning”, annars görs ett ”Felmeddelande”.

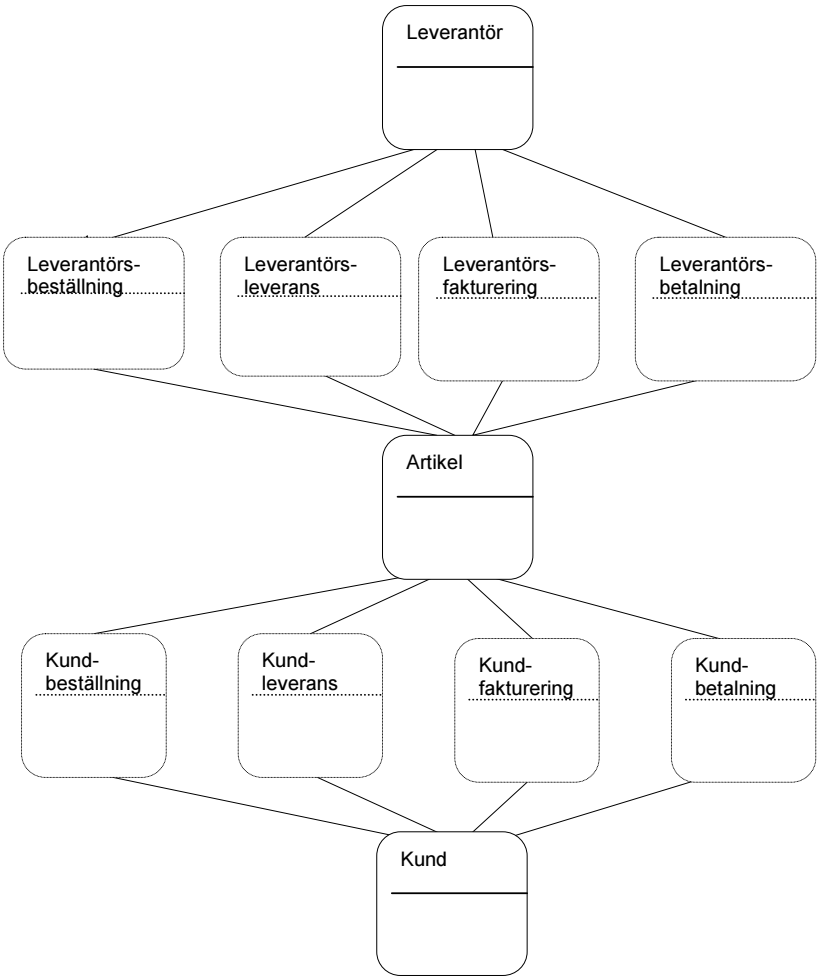
Egentligen är detta exempel snarare ett mycket bra exempel på svårigheterna med konceptuell design än ett exempel på en bra modelleringsteknik. Wangs arbete skapar en hel del förvirring kring några av de mest grundläggande aspekterna på objektens natur. För det första är det inte egentliga objekt Wang modellerar utan den information som de egentliga objektens aktiviteter ger upphov till. De egentliga objekten finns inte med i Wangs modell. De dokumentobjekt som modellen innehåller är alltså resultat av händelser i objektsystemet, inte objekt i sig själva. Denna dynamik mellan objekt och de processer som pågår i objektsystemet måste speglas mer tydligt och oberoende av den information dessa processer ger upphov till.

[19] Wang, 306-307

2.4 Exemplifiering : modellering enligt ISO-modellen

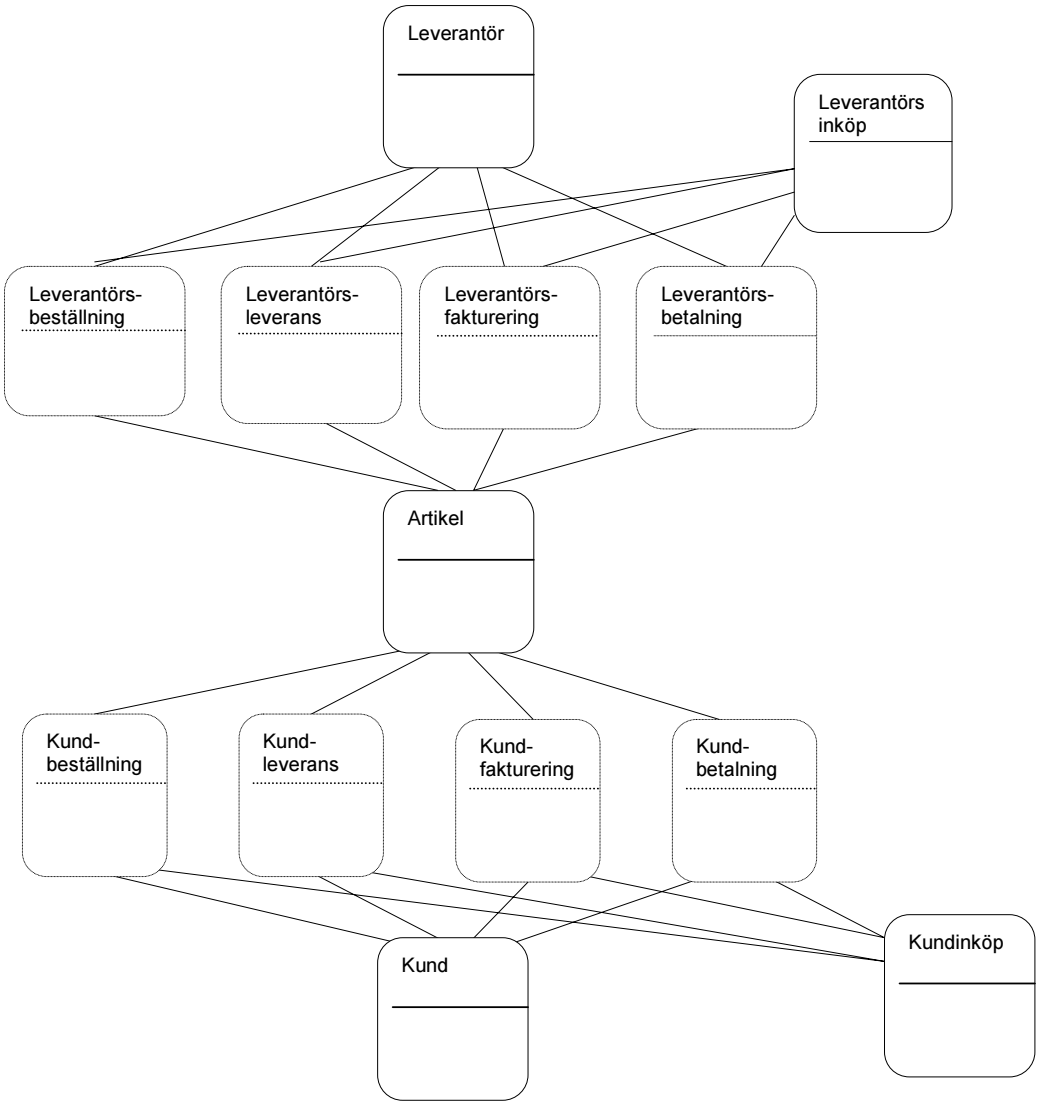
Bild 7 visar ett alternativt och mer objektorienterat sätt att modellera en liknande situation. Modellen kretsar kring de centrala objekten ARTIKEL, LEVERANTÖR och KUND. I de olika händelseklasserna (streckad linje) har vi information om de händelser som sker i objektsystemet och de dokument som Wang modellerar kan vi här skapa med hjälp av den information som finns i systemet.

Bild 7. Ett objektorienterat exempel, 1



För att verkligen spegla situationen skulle ett objekt KUNDINKÖP vara med också (med relationer till de relevanta händelseklasserna) med syfte att identifiera varje inköpsärende som en kund utför och ur händelseklasserna hitta den information som varje dokument behöver. På samma sätt skall ett objekt LEVERANTÖRSKÖP finnas för att identifiera varje inköp som organisationen gör från sina leverantörer. En komplett modell ser då ut enligt bild 8.

Bild 8. Ett objektorienterat exempel, 2



En av de stora fördelarna med detta exempel jämfört med Wang är att systemet blir mycket lättare att förvalta. Dokumentstandard i en verksamhet förändras oftare än de grundläggande objekten, så ett system byggt enligt Wangs modell skulle kräva oerhört stora ingrepp vid varje sådan förändring. I ett system enligt bild 6 är den grundläggande objektmodellen bestående och de dokument som behövs kan formas i gränssnitt som låter sig förändras på ett mycket smidigare sätt.

3. Teoretisering : Grundläggande begrepp och principer

3.1 Objekt eller information[20]?

Inledningsvis vill jag för tydlighetens skull återkomma till den distinktion mellan objekt och information som är så oerhört viktig för att förstå svårigheterna med det konceptuella designarbetet. I jämförelsen mellan Wang och Mylopoulos beskrev jag hur Wangs otydlighet på den punkten innebar ett sätt att modellera som leder till bristande kvalitet.

Tabell 1

Förnamn	Efternamn	Yrke	Ort
Nisse	Nilsson	Telereparatör	Varberg
Nisse	Nilsson	Teleoperatör	Varberg
Nisse	Nilsson	Teleoperatör	Varberg
…	…	…	…

Tabell 1 visar en tabell som skulle kunna vara hämtad ur en organisations personaldatabas. De som arbetar på avdelningen i Varberg kan givetvis utan problem skilja på de tre Nisse Nilsson, men för någon annan som enbart ser dessa uppgifter i databasen är det omöjligt att avgöra om det är tre olika personer eller en och samma. Databasen måste innehålla någon information som unikt identifierar var och en. Detta enkla exempel visar att den information som databasen innehåller inte är samma sak som det verkliga objektet. Det är denna åtskillnad som ISO-modellen vill lyfta fram. Det är oerhört viktigt att vara medveten om skillnaden mellan objekten i objektsystemet och informationen i informationssystemet.

[20] Inom informatiken åtskiljs ibland mellan information och data. Data representerar då de enkla meddelandena i sin grundform. När en uppsättning data har betydelse för någon användare har datan blivit till information. I det här stycket distinguerar jag inte så hårt emellan detta. Med information menar jag data/information i en enkel text eller sifferform, dvs i den form som data vanligtsvis lagras i databaser.

3.2 Vad är ett objekt?

Innan det objektorienterade arbetssättet slog igenom talades det om entiteter. Det är i princip ting, det vill säga fysiskt existerande objekt som definierades utifrån sina attribut och relationer till andra entiteter. Med objektorienteringen började begreppet objekt användas istället. Objekt kan sägas vara entiteter med beteende. Det innebär att förutom attribut och relationer skall också beteenden hos objekten beskrivas. Dessutom inleds i och med detta också en tid då även mera abstrakta typer av objekt tas med i modellen (om det behövs för systemets funktion). I litteraturen används objekt/entiteter ibland som om man de har samma betydelse, vilket det ofta också är. Connolly talar om starka och svaga entiteter[21], där svaga entiteter är sådana som för sin existens är beroende av ett starkt objekt.

Ett exempel på svaga entiteter är om vi har en klass PERSONAL med personaluppgifter samt en klass NÄRMASTE ANHÖRIG som har en relation till klassen PERSON. Klassen NÄRMASTE ANHÖRIG är intressant för organisationen för att ha någon att kontakta till exempel vid personskada. När någon ur gruppen PERSON slutar sin anställning så måste också alla relaterade entiteter i klassen NÄRMASTE ANHÖRIG tas bort, då objekten i klassen NÄRMASTE ANHÖRIG inte har någon relevant existens för sin egen skull (detta gäller om modellen avser ett system för hantering av information om anställda, i andra sammanhang kan givetvis klassen NÄRMASTE ANHÖRIG ha ett eget existensberättigande).

I Brown används begreppet objekt genomgående. Han skriver om olika typer av objekt[22]. Konkreta (ting vi har omkring oss), konceptuella, händelser och tillstånd.

Inom objektorienteringen gäller att objekten skall stödja funktionerna specialisering, generalisering, arv, polymorphism, aggregat och association;

Specialisering	I en modell över ett studentsystem skulle objektklassen STUDENT kunna uttrycka de grundläggande egenskaperna och beteendena för varje student. Det är också möjligt att ”specialisera” klassen och lägga till subklasser som till exempel HELTIDSSTUDENT, FORSKARSTUDENT, och/eller DELTIDSSTUDENT. Dessa utgör då specialiseringar eller subklasser av STUDENT och objekten i de klasserna har både specifika egenskaper och beteenden och de allmänna som de har genom att samtidigt tillhöra klassen STUDENT.
Generalisering	Samma sak som specialisering, fast betraktat från andra hållet. Klassen STUDENT generaliserar sina subklasser och är alltså deras superklass.
Arv	Subklasser ärver beteenden och egenskaper från sin superklass. I studentexemplet kan vi tänka oss att superklassen STUDENT identifieras av ett för skolan internt identitetsnummer. Objekten i

[21] Connolly, 152
[22] Brown, 112

	subklasserna till klassen STUDENT ärver då sitt identitetsnummer därifrån. Subklasserna kan givetvis också ha egna, specifika, egenskaper och beteenden.
Polymorphism	Är en mekanism som är nära förbunden med arvsmekanismen. En subklass ärver sin superklass men kan ju se annorlunda ut och kan ha ett annorlunda beteenden. Ett exempel kan vara om vi har en superklass FORDON och subklasserna BIL och CYKEL. I superklassen har vi en metod (beteende) skriv_ut() som gör att objektet presenteras t.ex. på bildskärmen. I subklasserna finns samma metod skriv_ut(), men implementerad på olika sätt eftersom BIL och CYKEL har olika egenskaper (attribut). Om vi i informationssystemet hämtar alla existerande fordon ur en databas och presenterar dem på skärmen så vet systemet vilken subklass varje objekt tillhör och använder sig av rätt skriv_ut()-metod. Dessutom skulle det kunna finnas en subklass (t.ex. TRAKTOR) som inte har några egenskaper utöver de allmänna som ärvs via FORDON. En sådan subklass behöver ingen egen skriv_ut()-metod utan klarar sig med den som ärvs via FORDON. Polymorphism betyder mångformighet och innebär alltså att ett beteende (en metod) i en sub- superklasshierarki kan ha samma namn men vara implementerad på olika sätt.
Aggregat	Ett förhållande mellan klasser som innebär att en eller flera klasser ingår som delar i en annan, som då är den/de ingående klassen/klassernas container. Hur detta modelleras är inte på något sätt självklart och beror på syftet med informationssystemet, men ett exempel kan vara att klassen BIL är container för klasser som MOTOR, KAMAXEL, INSTRUMENTPANEL och så vidare. En bil består alltså av ett antal komponenter.
Association	En association är en relation mellan två klasser. Förutom de som beskrivs här ovan kan det handla om relationer där en klass behöver utnyttja metoder i en annan klass för att lösa en uppgift eller där flera klasser är inblandade i gemensamma händelser. I ett informationssystem implementeras sådana relationer med metodanrop.

Aspekter på objekt (ontologi) rör deras natur, form, egenskaper, verksamhet, uppkomst, tillstånd, processer, förmågor, rörelser, relationer. För att bedöma ett objekt måste vi med andra ord kunna svara på frågor om hur objektet blir till, hur det upphör att existera, hur det utvecklas över tiden, objektets identitet och hur identiteten säkerställs när objektet utvecklas över tiden, om och i så fall när och hur ett objekts identitet förändras.

3.3 Metoder för objektmodellering

Mycket talar för att det centrala problemet i systemutvecklingen är just att hitta den ”rätta” uppsättningen objekt[23]. En av svårigheterna i detta ligger i människans perceptionsförmåga. Det vi egentligen skall modellera är ju objektklasser, dvs samlingar av objekt. Men människan ser inte primärt klasser i sin omvärld, vi uppfattar individuella objekt. Av olika skäl grupperar vi dessa objekt i klasser. Problemet är då att denna gruppering/klassificering kan göras på många sätt, till exempel;

- *liknande värden för ett eller flera attribut*
- *värdet på något attribut ligger inom ett intervall*
- *gemensam uppsättning attribut*
- *gemensamt beteende*
- *liknande betydelse/mening*

Hur vet vi att vi väljer den rätta uppdelningen? Finns det en riktig modell, eller finns det flera riktiga modeller? Kärnfrågan i detta är hur vi definierar en uppsättning ”universella” klasser ur en mängd individuella objekt. Existerar objektklasser i verkligheten och väntar på att bli upptäckta av modelleraren, eller är objektklasserna modellerarens egna tankekonstruktioner baserade på abstraktioner av den del av verkligheten som skall modelleras? Hur kan vi veta att ett träd är ett träd när det existerar så stor variation mellan olika sorters träd? Om vi med fågel menar ett djur som flyger, vad är då en struts? Egentligen är en fågel ett ryggradsdjur som härstammar från kräldjuren där strutsen som saknar flygförmåga är den största. I vardagligt tal är vi inte så exakta i våra definitioner, utan tar en hel del kategoriseringar för givna.

Det finns en rad vedertagna metoder beskrivna när det gäller att arbeta med den konceptuella designen. I framför allt Brown, Sommerville och Connolly finns detaljerade beskrivningar på de viktigaste. Även om de olika metoderna skiljer sig åt i detaljerna så är huvuddragen desamma. I korthet gäller det att;

- *identifiera alla relevanta objekt*
- *identifiera relationerna mellan objekten*
- *hitta objektens attribut*
- *identifiera attributens domäner*
- *bestämma kandidat- och primärnycklar*
- *identifiera sub- och superklasser*

Gemensamt är också betoningen på att denna fasen bör göras i nära samarbete med beställarna/användarna. Genom att i samtalsform analysera verksamhetens struktur skall objekten med dess attribut och relationer kunna identifieras. Den konceptuella modell som detta resulterar i beskriver ”vilka data som skall lagras i databasen och vilka relationer som gäller mellan dem.”[24]

[23] Se t.ex. Wang, 305,
[24] Connolly, 41

De redskap man har att arbeta med i dessa samtal är dels de skriftliga kravspecifikationer och andra dokument som tagits fram tidigare i arbetet och dels arbetsmetoder för själva samtalsmötena. Objektmodellens kvalitet är avgörande för kvaliteten på det färdiga systemet.

I det material jag läst går det att urskilja ett antal stora grupper av metoder. Dels är de några som för oss informatiker representerar de ”gamla kända”, som vi berört under utbildningen. Då dessa utgör en basuppsättning i den ”verktygslåda” en utvecklare behöver så kan en genomgång försvara sin plats här. Dels är det ett par intressanta ansatser till metodutveckling som har siktet inställt på kommande utmaningar för den konceptuella designen.

3.4 Traditionella metoder

Här finns ett antal beskrivningar i litteraturen som delvis överlappar varandra. Med rubriken ”Traditionella metoder” menar jag enbart att de är hämtade ur publicerad litteratur och inte ur aktuella forskningsrapporter. Jag betraktar de med andra ord inte som traditionella i något värderande syfte i förhållande till de tankar som Wang och Mylopoulos presenterar. Gemensamt är användarmedverkan och att i flera steg arbeta sig fram till en modell.

Genomföra **intervjuer** med användare, med penna och formulär, för att fånga in arbetssätt och informationsbehov[25].

Studera verksamhetens processer, rapporter och formulär ”på plats”. Det vill säga att direkt på plats i verksamheten studera processerna och notera hur arbetet utförs[26].

Göra en **grammatisk analys av kravspecifikationen**. Här blir substantiv eventuella objektklasser, verb eventuella relationer[27].

Identifiera generella objektklasser eller objektklasser som har ett berättigande i sig själva. För till exempel ett företag som hyr ut segelbåtar går det att anta att BÅT är en klass som kommer att behövas[28].

User case scenarios (användarfallsbeskrivningar). Här sitter en grupp bestående av flera användare och utvecklarna och tar fram beskrivningar på alla sätt som användarna skall kunna interagera med systemet. Det görs grafiska bilder av gränssnitt som visar informationsbehov och vilka uppgifter systemet skall kunna utföra med informationen[29].

Brainstorming. Ett kreativt kaosmöte där deltagare från användarsidan enbart skall kasta fram idéer om vilka objektklasser som skulle kunna vara tänkbara. Detta mötet är då en kreativ fas där inget förslag får kritiseras eller förkastas. Allt skall upp på

[25] Brown, 294
[26] Lewis, 162
[27] Connolly, 231, Brown, 294
[28] Connolly, 231, Mathiassen, 83
[29] Brown, 294

bordet och noteras. Efter detta tar en kontrollfas vid då en utvärdering av kandidatklasserna kan göras[30].

Delphi-metoden. Cirkulationsbrev eller e-post med en lista över tänkbara objektklasser går runt bland personalen. Var och en fyller på efter sina idéer. När listan gått runt är det utvecklarens uppgift att bearbeta den i möten med några användarrepresentanter[31].

SSM (Soft Systems Methodology) är en något annorlunda metod, där avstampet tas i en mycket vid omvärldsformulering innehållande beskrivningar av systemets intressenter och syfte. Intressenterna kan finnas långt utanför den organisation där systemet skall finnas, till exempel samhälleliga institutioner eller befolkningsgrupper. Med denna beskrivning som utgångspunkt arbetas en objektmodell fram[32].

[30] Brown, 294
[31] Brown, 294
[32] Lewis, 155 -

4 Enkät, empiriska resultat

För att få en bild av hur metodarbetet inom objektmodellering ser ut i professionella sammanhang har jag gjort en enkät som några personer svarat på. I detta kapitel presenteras enkäten. Jag har skickat den till tre konsulter, en på RKS Data och två på Sigma nBit.

Då det är vanligt med hög arbetsbelastning på IT-företag har jag utformat enkäten på ett enkelt och kortfattat sätt. Risken är att frågeställningar inte blir presenterade på ett riktigt komplett sätt, men vinsten är att det överhuvudtaget går att få svar. Enkäten ger en så pass tydlig bild av problemområdet att jag bedömer den som relevant.

4.1 Enkätformuläret

Bakgrund

Mitt examensarbete handlar om objektmodellering / konceptuell design. Fokus ligger på frågan om hur vi kan veta att den objektmodell vi tar fram är tillräckligt bra. Grunden till problemet är att vi modellerar objektklasser snarare än objekt. Då några naturliga objektklasser inte existerar så är det upp till oss själva att kategorisera objekten i de klasser vi finner lämpliga. Det innebär dels att ett givet system kan modelleras på många olika sätt och dels att kommunikationen mellan användare/beställare och utvecklare är kritisk eftersom missförstånd och olika uppfattningar kan innebära att systemet modelleras fel.

I den litteratur jag studerat beskrivs arbetet med att identifiera en riktig uppsättning objektklasser som den svåraste fasen i systemutvecklingen och dessutom som ett område där metoder än så länge är relativt outvecklade. Ett antal metoder/arbetssätt som är tänkta att säkerställa kvalitet på objektmodellen. Syftet med den här enkäten är att "stämma av" de teorierna mot praktiska erfarenheter. Jag är alltså inte ute efter speciella notationstekniker som t.ex. UML eller hela utvecklingsmetoder som t.ex. RUP.

Fråga 1 - Problemområdet

Hur ser Du på problemområdet? Är arbetet med att ta fram en bra objekt(klass)modell den svåraste fasen? I så fall, vari ligger de största svårigheterna? Är metoder inom detta området överlag outvecklade?

Fråga 2 - Metoder/arbetssätt

Jag nämner ett antal metoder/arbetssätt här nedan. Ge några kommentarer om de Du känner till och har en uppfattning om.

Användarintervjuer. Genomförs för att fånga in arbetssätt och informationsbehov.

Studera verksamhetens processer. Studera rapporter, formulär, informationsstruktur och arbetssätt ”på plats”.

User Case Scenarios. Användare/beställare och utvecklare går tillsammans igenom varje tänkt sätt som användarna skall kunna interagera med systemet. Analysen görs både i bild och text.

Grammatisk analys. Kravspecifikationen och annan dokumentation (t.ex. från Use Case scenarios) studeras. Substantiv är potentiella klasser, verb potentiella relationer mellan klasser.

Brainstorming. ”Kreativitetsmöten” med syfte att enbart vaska fram idéer om tänkbara klasser. Inget får kritiseras, utvärderingen görs senare.

Delphi-metoden. Ungefär som brainstorming, men cirkulationsbrev eller e-post går runt bland deltagarna. Var och en fyller på med tänkbara klasser. Efteråt bearbetas den i möten med utvecklare och användare.

SSM (Soft Systems Methodology). Avstampet tas i en mycket vid omvärldsformulering där systemets intressenter och dess syfte beskrivs. Det kan t.ex. handla om vilka grupper som tjänar respektive förlorar på om systemet införs. Med dessa beskrivningar som utgångspunkt börjar klasser formuleras.

Fråga 3 - Er organisations metodutvecklingsarbete

Hur arbetar Ni med dessa frågor? Någon speciell metod? Vad är bra resp. dåligt? Arbetar Ni med egen metodutveckling?

Fråga 4 - Forskningen

I den mer akademiska forskningen finns ett par ansatser till metodutveckling. Jag redogör (mycket kort) för två forskares arbete. Ge några kommentarer till detta.

Shouhong Wang, (University of New Brunswick, Canada)

Angriper problemet genom att strukturera upp arbetet. Genom att steg för steg arbeta med olika perspektiv på den verksamhet som skall modelleras så menar han att kvaliteten på modellen skall bli bättre. Utvecklarna försäkrar sig om att täcka in alla aspekter och att inte missa något väsentligt. Enligt Wang består en verksamhet av 4 delar;

Processer	*innehåller objekt av typerna fysiska objekt, händelser och dokument*
Målsättningar	*innehåller objekt av typerna mål och delmål*
Aktörer	*människor, men inte utifrån personuppgifter utan mer utifrån ansvar, befogenheter och uppgifter*
Klient/server-lösningar	*viktiga i automatiserade system. Även gränssnittsklasser*

John Mylopoulos, (University of Toronto, Canada)

Tar ett mer filosofiskt grepp på problematiken. Det mänskliga språket räcker inte till för att kommunikationen mellan utvecklare och användare skall räcka som utgångspunkt för en bra objektmodell. Liksom Wang vill Mylopoulos strukturera modelleringen men gör det utifrån ett antal ontologier (antaganden om verkligheten).

Statisk ontologi.	*Existerande objekt, (kan vara konkreta eller abstrakta).*
Dynamisk ontologi.	*Förändringar i objekttillstånd, transformationer, processer*
Syftebeskrivande ontologi.	*Modellen skall kunna beskriva Aktörer, målsättningar och liknande.*
Social ontologi.	*Modellen måste kunna spegla en organisatorisk miljö, struktur, allianser och beroenden.*

4.2 Svar nr 1, Sigma nBit

Fråga 1

Det svåra med design är i starten då mycket information om hur systemet skall fungera kanske saknas samt att en del förutsättningar förändras.

Design av ett system är en iterativ process, där man jobbat med en grunddesign av systemet, som sedan förfinas allt eftersom use case blir klara. Grunddesignen delar upp systemet i olika delar (objekt) med ansvar för att lösa ett eller flera problem. Utifrån det tänkta ansvarsområdet för ett objekt, kan interfacet för objektet skapas.

Iteration av use case genom systemet förfinar och/eller förändrar objekten, svaren från iterationen kan ge en indikation på om grunddesignen fungerar.

Fråga 2

Användarintervjuer
Bra sätt att få information från tänkta slutanvändare om gränssnitt och arbetssätt som kan påverka designen.

Studera verksamhetens processer

User Case Scenarios
I produktutveckling är det ofta så att det finns en eller flera kravställare, kraven överlämnas i Use Case form med tillhörande beskrivningar.

Grammatisk analys

Brainstorming
En klassiker som jag inte är särskilt förtjust i, finns ett stress moment att komma på en massa idéer.

Delphi-metoden

SSM (Soft Systems Methodology)

Fråga 3

Har hittills arbetet med brainstorming som jag inte tycker om, fungerade inte särskilt bra men med Use Case modellen, som fungerar väldigt bra, ger en överskådlig bild av ett problem eller del av ett problem, minskar risken för feltolkningar osv.

Fråga 4

Angående Shouhong Wang, (University of New Brunswick, Canada)
Som jag tolkar Wang så vill han arbeta med en iterativ process genom modellen som allteftersom förfinas

Angående John Mylopoulos, (University of Toronto, Canada)
John Mylopoulos, vet inte om jag tolkar texten fel men design handlar inte så mycket språket mellan utvecklare och användare, design ska spegla systemets interna uppbyggnad, interface osv man kan säga språket mellan designern och utvecklare.

4.3 Svar nr 2, Sigma nBit

Fråga 1

Har inga djupare erfarenheter av objektmodellering men med den lilla erfarenhet jag har så tror jag ofta att det är svårt att ta fram en modell som håller i det långa loppet. Det är svårt att i tid fånga upp alla krav som modellen bör leva upp till. När dessa krav klargörs är det inte ovanligt att man har en modell som kräver större förändringar än vad den borde haft för att hantera de nya kraven.

Jag tror nog att metoderna inom detta område idag börjar bli ganska mogna. Problemet är ofta att för lite tid och resurser läggs på krav hantering. Man börjar implementera systemet för tidigt.

Fråga 2

Användarintervjuer
Har man tillgång till en lämplig användargrupp är detta ett utmärkt sätt att fånga krav. Lätt att missa något om man inte har en definierad struktur för intervjuerna.

Studera verksamhetens processer

User Case Scenarios
Ett bra sätt att fånga krav. Har dock enbart använt det för att beskriva ett redan existerande system. Dvs. vi gjorde systemet först och när systemet sedan skulle objektifieras använde vi Use Case Scenarios för att beskriva systemet. I det här läget kändes det dock ganska onödigt då vi redan hade alla krav definierade i det redan befintliga systemet.

Grammatisk analys

Brainstorming
Bra sätt för att få igång processen och samla ideér men jag tror det är viktigt att utvärderingen sedan är kritisk till resultatet.

Delphi-metoden.

SSM (Soft Systems Methodology)

Fråga 3

Har använt RUP men i väldigt liten utsträckning och då med betoning på "Use Case Scenarios". Tycker "Use Case modellering" är ett väldigt bra sätt att fånga krav men det är mycket viktigt att man har möjlighet till en bra dialog med beställaren för att det skall ge något.

Fråga 4

Kommentarer
Med den erfarenhet av objektmodellering känner jag att jag inte kan ge några direkt intressanta kommentarer kring dessa ansatser. Rent spontant känner jag dock att ansats 1 känns lite mer jordnära vilket är viktigt. Det får inte bli för teoretiskt. Det blir då svårhanterligt när vi sedan skall använda det i praktiken. Det känns också som om ansats 1 lägger stor vikt vid att analysera kraven utifrån flera perspektiv för att inte missa något. Detta är bra och något jag tror man ofta missar.

4.4 Svar nr 3, RKS Data

Fråga 1

Frågan hur man kan bedöma en modells kvalitet har diskuterats länge och mig veterligen finns det inget bra svar, ännu. Frågan är ju inte heller specifik för objektorienterade modeller utan är, i mitt tycke, applicerbar i de flesta sammanhang där modeller används. Det finns förvisso en del metriker som kan användas för att utvärdera sina modeller. Problemet med dessa är att de oftast mäter storlek, komplexitet eller koherens, inte kvalitet eller korrekthet. Vad man egentligen skulle vilja mäta är väl egentligen hur användbar en modell är givet en problemdomän.

Svårt att ge något generellt svar på om just arbetet med att ta fram klassmodellen är det svåraste. Detta hänger ihop med att det arbetas på väldigt olika sätt på olika företag. En del går verkligen in i detalj vid designen och lämnar väldigt få frihetsgrader till programmerarna. Andra nöjer sig med en ganska grovhuggen design och överlåter en stor del av tänkandet åt programmerarna. Om vi utgår från att det

mesta tänkandet ska göras vid design så blir dock svaret att arbetet med att ta fram en bra klassmodell är det svåraste.

De största svårigheterna när det gäller modelleringsarbetet har visat sig vara att hitta bra klasser, särskilt för mindre erfarna utvecklare. Har man väl hittat bra klasser faller många delar på plats, mer eller mindre av sig själva. Det finns ju, som ni nämner i fråga 2, ett antal olika arbetssätt när det gäller att hitta klasskandidater. Många av dessa arbetssätt är väl inarbetade och fungerar ganska bra men även här saknas metriker. Det är alltså svårt att objektivt ”visa” att en klasskandidat kommer att bli en bra klass. Min uppfattning är att det säkerligen går att hitta bättre metoder än de som finns idag men att det alltid måste till en rejäl portion erfarenhet för att skapa bra modeller.

Fråga 2

Användarintervjuer.
Jätteviktigt. Finns det möjlighet att träffa verkliga användare bör man inte missa detta. Det handlar ju till stor del också om införsäljning av systemet hos de framtida användarna. Förbises tyvärr alltför ofta, man pratar enbart med köparen istället.

Studera verksamhetens processer.
Mycket viktigt. Alla system är ju till för att stödja verksamheten på ett eller annat sätt så naturligtvis måste man ha förståelse för verksamheten. Ofta ett mycket bra sätt att hitta klasskandidater.

User Case Scenarios.
Bra och enkelt sätt att kommunicera med icke-tekniker.

Grammatisk analys.
Snabbt och enkelt sätt att hitta klass- och relationskandidater.

Brainstorming
Kräver att företagsklimatet och gruppen är sådan att alla vågar och får chansen att komma till tals. Tyvärr är min erfarenhet att dessa möten sällan blir så effektiva som de borde kunna vara.

Delphi-metoden.

SSM (Soft Systems Methodology).

Fråga 3

Eftersom RKS är ett renodlat konsultföretag utan egen produktutveckling använder vi inte någon speciell metod. Vi använder helt enkelt den metod som kunden använder. I de fall kunden inte har någon egen metod och vi ombeds komma med förslag så föreslår vi RUP eller DSDM. Vi arbetar inte med egen metodutveckling. Däremot kräver ju RUP anpassningar till den specifika organisationen och sådant gör vi. Fördelarna med RUP är att den är väldigt flexibel och därmed användbar i de flesta sammanhang. RUP använder UML som notation vilket är bra eftersom UML är en

standard. Nackdelarna med RUP är att den inte är gratis och att den är väldigt stor. Det är svårt att få överblick och den bör alltid anpassas till den tänkta miljön. DSDM har jag tyvärr endast teoretiska kunskaper om så jag avstår från att kommentera den metoden.

Fråga 4

Angående Shouhong Wang, (University of New Brunswick, Canada)
Svårt att skapa sig någon riktig bild av metoden utifrån ovanstående, korta beskrivning. Jag tycker att det ser ut som en beskrivning av vilken verksamhetsmodellering som helst och kan inte riktigt se vad det är som är nytt. Betrakta verksamheten ur olika vyer har man väl alltid gjort och utgår då från verksamhetens mål, processer och resurser. Jag vet inte riktigt vad han avser med klient-server-lösningar men om inte detta begrepp används i sin absolut vidaste definition så kan jag se en risk med att binda upp verksamheten kring denna typ av lösning. Trenden idag är väl snarare att man går ifrån klient-server lösningar till förmån för ”terminalliknande” system.

Angåene John Mylopoulos, (University of Toronto, Canada)
För det första tycker jag ansatsen att naturligt språk inte räcker till är direkt felaktig. Problemet är väl tvärtom att naturligt språk är alltför stort och ger alltför många möjligheter. Jag kan dock hålla med om att detta skapar problem när det gäller att uttrycka sig formellt. Utan att veta mer om metoden än vad ni beskriver ovan så har jag svårt att se hur hans ansats kan lösa kommunikationsproblemet. Tvärtom har jag svårt att se något annat än just naturligt språk när det gäller kommunikationen mellan utvecklare och användare (om inte användarna är tekniker själva). Jag tror att de flesta användare inte har en aning om vad aktörer, objekt etc. innebär vid systemutveckling. Vad händer om något av antagandena om verkligheten visar sig vara felaktig?

Kommentarer
När det gäller båda de ovanstående metoderna kan jag inte se att de skulle innebära något stort steg framåt när det gäller metoder. De verkar dessutom enbart behandla verksamhetsmodellering, som visserligen är en viktig del av systemmodellering, men metoderna säger inget om hur man går från verksamhetsmodell till systemmodell. Tyvärr är det nog så att det fortfarande återstår mycket arbete innan vi på allvar kan prata om modellkvalitet och särskilt hur man kan mäta denna.

4.5 Sammanställning av enkätsvar

Frågor	Sigma nBit (1)	Sigma nBit (2)	RKS Data	Mina kommentarer
1. Är rapportens problemområde relevant? Är metodutveckling viktigt?	Föränderliga förutsättningar försvårar.	Svårt att skapa en modell som håller i långa loppet. Implementationen börjar ofta för tidigt.	Viktig frågeställning, än finns inga riktigt bra metoder för arr mäta kvalitet i modellen. Svårt att hitta bra klasser.	En tydlig bild av att modellering är en svår fas i systemutveckling.
2. Dina åsikter om 'Användar-intervjuer' som metod?	Bra sätt att få information om gränssnitt.	Utmärkt sätt att fånga krav. Viktigt med struktur i samtalen.	Jätteviktigt,. Man bör inte missa detta, om tillfälle ges.	---
3. Dina åsikter om att 'Studera processerna' som metod?	---	---	Mycket viktigt. Bra sätt att hitta kandidatklasser.	Bara ett svar, vilket tyder på att många inte ger sig tid (eller är med-vetna om behovet av) att skapa stor förståelse för den aktuella verksamheten. Dåligt gensvar på en metod som är lämplig för att hitta kandidatklasser.
4. Dina åsikter om 'User Case Scenarios' som metod?	Kravspecifika-tionen överlämnas ofta i USC-form.	Bra sätt att fånga krav.	Bra och enkelt sätt att kommunicera med icketekniker.	---
5. Dina åsikter om 'Gramatisk analys' som metod?	---	---	Snabbt och enkelt sätt att hitta kandidatklasser.	Återigen ett dåligt gensvar på en metod som är lämplig för att hitta kandidatklasser.
6. Dina åsikter om 'Brain-storming' som metod?	Dåligt på grund av stressmomentet.	Bra metod men viktigt med en kritisk utvärdering.	Kräver bra relationer. Dåliga erfarenheter.	Det ligger säkert en hel del i att stressmomentet gör metoden ineffektiv.
7. Dina åsikter om 'Delphi' som metod?	---	---	---	Ännu ett dåligt gensvar på en metod som är lämplig för att hitta kandidatklasser.
8. Dina åsikter om 'Soft Systems Methodology' som metod?	---	---	---	Okänd metod för alla. Jämför med fråga 3. Tyder på att helhetssyn är bristvara. System designas mer utifrån interna överväganden.

9. Hur arbetar din organisation med metoder och metod-utveckling?	USC fungerar bäst. Minskar risken för feltolkningar.	USC är en bra metod om dialogen med beställaren är bra.	Konsultföretag som använder kundens metoder.	---
10. Ge några kommentarer till min beskrivning av Wangs ansatser.	Iterativ ansats.	Praktiskt inriktad metod. Bra att analysera kraven utifrån olika perspektiv.	Svårt att se vad som är nytt.	---
11. Ge några kommentarer till min beskrivninga av Mylopoulos ansatser.	Design handlar inte om språket mellan utvecklare och användare utan om systemets interna uppbyggnad.	---	Håller med om att naturligt språk skapar problem när det gäller att uttrycka sig formellt, men är det enda som går att använda.	---

5. Diskussion och slutsatser

Inom den konceptuella designen finns det en rad verktyg som är viktiga att känna till och kunna använda. Den konceptuella designen syftar till att ta fram en objektmodell komplett med objektklasser, deras attribut, relationer mellan objektklasser och beteendemönster för varje objektklass. Rapportens har avgränsats till att undersöka svårigheter med och metoder för att identifiera de objektklasser som skall ingå i en konceptuell modell. Frågorna om attribut, beteenden och relationer har inte behandlats lika ingående. Genom att föra samman metoder från olika håll i litteraturen går det att identifiera en ”verktygslåda” med olika arbetssätt för att identifiera relevanta objektklasser. Att döma av enkätsvaren så används de på ett blandat sätt i olika organisationer och i olika projekt, vilket säkert också är rimligt med tanke på olika förutsättningar i form av kravspecifikationer, tillgänglig tid och kompetens både hos beställare och utförare. Frågan är dock om inte ett metodutvecklingsarbete för att konsolidera existerande och utveckla nya arbetssätt (givetvis utan att eliminera möjligheterna till anpassning) kan ge möjligheter till en mer systematisk kvalitetssäkring av modellerna? Det kanske kan tänkas att någon form av Balanced Scorecard[33] för att bedöma en modell går att ta fram (med parametrar för de svårigheter jag beskrivit)? I det följande besvaras de frågeställningar som rapporten definierade i inledningen. Dessutom följer ett sammanfattande stycke.

Hur vet vi att den objektmodell vi designar är relevant och leder till ett informationssystem som är både stabilt och förändringsbart?

I svaret från RKS Data påpekas att de metoder som finns ofta mäter andra saker än just modellens kvalitet. Istället mäts till exempel storlek eller komplexitet som det säkert är lättare att hitta mätetal för men som inte är vad vi är ute efter.
När det gäller frågan om hur vi kan bedöma en modells kvalitet har jag identifierat en rad kritiska faktorer. Det som krävs är dels att inte gå i samma fälla som Wang delvis fastnar i när han blandar ihop objekt och information. ISO-modellens åtskillnad av

[33] Balanced Scoracard har (inom organisationsutvecklingstänkandet) blivit ett allmänt accepterat sätt att bedöma kvalitet i verksamheter. Det går ut på att identifiera ett antal parametrar som är mätbara enligt någon skala (t.ex. i form av en vanlig betygsskala som ”1-5” eller någon mer värderande skala som ”Starkt – Tillfredsställande – Svagt – Hotfullt”). Syftet är att kartlägga starka och svaga sidor för att få en tydlig bild av hur till exempel ett kavalitetsarbete bör priortitera tillgängliga resurser.

objektsystem och informationssystem måste följas, så att vi modellerar de riktiga objekten och inte den information deras beteendemönster ger upphov till. Det är med andra ord oerhört viktigt att ha en hög grad av förståelse för ISO-modellens budskap. I grund och botten handlar jämförelsen mellan Wang och Mylopoulos om detta, där Wang inte har tillräcklig insikt i, eller respekt för, de grundläggande objektorienterade synsättet på objekt och åtskillnaden objektsystem/informationssystem.

- Detta innebär att arbeta med en stor **förståelse för skillnaden mellan objekt och den information som lagras** i informationssystemet. Den konceptuella modellen skall innehålla de riktiga objekten, hämtade ur objektsystemet. ISO-modellen utgör en gemensam plattform för hur det grundläggande objektorienterade synsättet skall användas i det konceptuella designarbetet.
- Dessutom är det viktigt att **utveckla en metod där objektsystemet granskas på ett strukturerat sätt utifrån olika perspektiv**. Mylopoulos fyra ontologier representerar en möjlig metodutveckling här och även Wang har inslag som är användbara. Att arbeta på ett strukturerat sätt enligt de riktlinjer som Wang och Mylopoulos förespråkar innebär att modellen förses med fler ”mätpunkter”.
- Detta kan i sin tur leda till att en **bättre kvalitetskontroll** av modellen är möjlig.
- Det innebär också **att arbetet kan standardiseras** och utföras på ungefär samma sätt över flera projekt, **vilket innebär möjlighet till utvärdering, förbättring och fortsatt metodutveckling** och därmed en efterhand allt säkrare kvalitetsvärdering.

Vi kan egentligen aldrig veta om en modell är rätt. Däremot kan vi säkerställa att den är relevant nog för sin uppgift. Vi bör också se till att den är begriplig (för framtida underhåll) och flexibel (så att önskemål om ändringar och kompletteringar är möjliga). Det kan förefalla självklart, men det får konsekvenser för hur modellen skall se ut som kan vara svåra att se från början. Det handlar om att alltid utgå ifrån att verksamheten kan, och kommer att, förändras. Ställ hela tiden ytterligare frågor om delar i modellen även om de verkar vara färdiganalyserade och klara. Vilket förhållande kommer att gälla mellan olika klasser? Kommer en kund i ett kundregister alltid att vara en människa? Kommer en lärare i ett utbildningsföretag aldrig att gå kurser som deltagare i det egna företaget?

Var ligger svårigheterna med objektmodellering?

Mina utgångspunkter i rapporten var att den konceptuella designen är ett svårt område och att allt mer komplexa system och svårigheter med att identifiera objektklasser innebär att kraven på arbetet med konceptuell design ökar. I litteraturen påpekas att det är det kanske svåraste momentet i systemutvecklingen och att metoderna behöver bli bättre. Även enkätsvaren bekräftar överlag detta. Det stämmer också överens med de argument Wang för fram, att den objektorienterade metoden erbjuder en uppsättning verktyg för att specificera objekt som identifierats (identitet, attribut, beteenden, strukturer) men ingen riktig hjälp i arbetet med själva kärnan i arbetet; att identifiera relevanta objektklasser. Det var lite förvånande att enkätsvaren bekräftade att modellering är en svår fas samtidigt som svaren på flera av de frågor som rörde metoder ägnade åt att identifiera kandidatklasser fick blandade svar. Jag drar utifrån det slutsatsen att modelleringsarbetet inte alltid görs så noggrant som det borde.

Min undersökning visar att de största svårigheterna i hög grad vilar på filosofisk grund. I ett systemutvecklingsarbete är det oerhört viktigt att utvecklare och användare utvecklar en gemensam bild av objektsystemet. Då det inte finns några naturliga objektklasser så finns det alltid olika sätt att modellera ett system. Vår verklighetsuppfattning och de sätt på vilka vi kategoriserar ting i olika klasser låter sig dock inte utan vidare kommuniceras på ett tillräckligt exakt sätt. Ibland tar vi också själva en hel del objektklasser för givna. Naturligt språk är alltför rikt så de samtal som förs i utvecklingsarbetet måste vara mycket strukturerade. Här ligger en konflikt i att den konceptuella modellen måste å ena sidan vara begriplig för användarna/beställarna och å andra sidan representera en tydlig professionell nivå när det gäller exakthet och kvalitet.

Vilka metoder och verktyg behövs för att designa stabila och effektiva system?

När det gäller de existerande metoderna så visade det sig att ingen av de som svarade på enkäten hade någon erfarenhet av SSM eller Delphi-metoden. Personligen tror jag att Delphi-grupper kan vara ett bra sätt att hålla kommunikationen igång och att ge alla inblandade möjlighet att reflektera över problemställningarna och därigenom få fram en bra lista med kandidater till klasser. Både användarintervjuer och User Case Scenarios rekommenderas i enkätsvaren, medan bara RKS Data rekommenderar at studera verksamhetens processer. Jag anser i och för sig att det är ett viktigt arbetssätt inte minst med tanke på de svårigheter som ligger i att vi i naturligt språk lätt använder begrepp och definitioner på olika sätt. Det är då viktigt att utvecklarna själva kan studera processerna i verksamheten för att få en bild av hur olika uppgifter utförs. Grammatisk analys får inget lysande gensvar i enkäterna. Jag uppfattar det själv som ett mycket bra sätt att hitta kandidatklasser, eftersom den skriftliga dokumentationen utgör ett kontrakt över det system som skall tas fram och utgör en omfattande beskrivning över både information och processer som systemet skall hantera. Det är bra om User Case Scenarios kan göras i samband med detta och dokumentationen kring dessa användarfall ingår i den dokumentation som den grammatiska analysen görs på. Brainstorming är det ingen av de svarande som har någon riktigt positiv upplevelse av. Det ligger antagligen en hel del i att stressmomentet gör att det inte blir så effektivt.

Angående frågan om metoder och verktyg så drar jag slutsatsen att metoderna ’Användarintervjuer’, Processtudier’, ’User Case Scenarios’, ’Grammatisk analys’ och ’Delphi-metoden’ är arbetssätt som bör ingå i en ”verktygslåda” för systemutveckling.

En mångfald av metoder behövs således. Jag har identifierat de mest användbara ur litteraturen. Dessa skall användas i kombination och hur de kombineras avgörs i hög grad av de unika villkor som gäller i varje projekt. Till dessa kommer de mer strukturerade arbetssätt som framför allt Mylopoulos men även Wang argumenterar för. Det råder inget antingen/eller-förhållande mellan de som jag kallat för ’traditionella’ metoderna och Wang/Mylopoulos. Tvärtom kan både Wangs och Mylopoulos metoder användas ihop med i stort sett var och en av de andra.

Slutord

Sammanfattningsvis går det att dra slutsatsen att konceptuell design otvetydigt är en av de största utmaningarna i arbete med systemutveckling. I litteraturen beskrivs en rad metoder för hur systemutvecklare kan gå till väga för att arbeta fram en objektmodell. Dessa metoder är dock inte tillräckligt exakta för att möjliggöra kvalitetssäkring av objektmodellen. De utgör dock en nödvändig uppsättning verktyg då de innehåller arbetssätt anpassade för olika situationer; i möten med användare och beställare, under egen utvärdering och i direkta studier av verksamheten. **Rapportens huvudfråga, hur vet vi att objektmodellen är relevant, kan anses besvarad** med att medvetenhet om ISO-modellen som en gemensam nämnare för hur en objektmodell skall se ut är en nödvändig grund för kvalitetssäkring. Genom att kombinera detta med metodutveckling på ett sätt där mer förfinade valideringsmöjligheter ges kan modelleringsarbetet bli säkrare. Det kan då också bli lättare att utvärdera och förmedla erfarenheter. Frågan om var **svårigheterna med objektmodellering ligger** besvaras med att de största svårigheterna vilar på filosofisk grund. Då det inte finns några naturliga objektklasser finns det alltid en mängd olika sätt att modellera ett system. Då dessutom naturligt språk ofta är för oexakt för att erbjuda de formella beskrivningar som måste göras i en objektmodell så är kommunikationen mellan utvecklare och användare/beställare ett kritiskt moment. Den sista frågan, **vilka metoder och verktyg behövs**, besvaras med att en rad olika metoder och verktyg krävs. En bra systemutvecklare måste dessutom ha förmåga att förstå helheten, den sociala, kulturella och tekniska miljö där systemet skall användas.

Det är min starka övertygelse att personer med god kunskap om objektmodellering kommer att vara en mycket efterfrågad grupp i dagens IT-samhälle och under överskådlig tid framöver. Siffror som att bara 15 % av IT-investeringarna leder till lyckade lösningar talar sitt tydliga språk. Självklart är en viktig orsak till detta att perspektivet i systemutvecklingsprojekt ofta betonar de inre aspekterna av systemet och de tillfälliga lösningarna. Det brister i att se helheten och i att designa systemet för en framtida verksamhet och förändringar, vilket idag är synonymt med verksamhet.

6. Referenser

Litteratur

Brown, D. (1997) *An Introduction to Object-Oriented Analysis. Objects in Plain English* . Wiley

Connolly, Th., Begg, C. (1999). *Database Systems. A Practical approach to Design, Implementation and Management* (2nd ed.). Addison-Wesley.

Lewis, P., (1994) *Information-Systems Development* Pitman Publishing.

Mathiassen L, Munk-Madsen A, Nielsen P A, Stage J (1998*) Objektorienterad analys och design*, Studentlitteratur

Somerville (2001) *Software Engineering* (6th ed.) Addisson-Wesley

Tidskriftsartiklar

Artz, J.M., (1997) A Crash Course in Metaphysics for the Database Designer. *Journal of Database Management* 8(4): 25-30.

Mylopoulos, J., Information Modelling in the Time of the Revolution (1998) *Information Systems* Vol. 23, No. ¾, 127-155.

Wang, R., Madnick, S.E., Facilitating Connectivity in Composite Information Systems (1989) *Database*, Fall 1989, 38-46.

Wang, S., Modelling Information Architecture for the Organization (1997) *Information & Management* 32, 303-315.