Handelshögskolan
VID GÖTEBORGS UNIVERSITET
Institutionen för informatik

Publiceringsdatum: 2005-05-27

STUDIE AV SYSTEMUTVECKLINGSMETODER PÅ FÖRETAG OCH UNIVERSITET INOM OBJEKTORIENTERAD ANALYS OCH DESIGN

Abstrakt

Vi ville med denna uppsats kunna se skillnader och likheter med de metoder och arbetssätt som idag undervisas på universitet med fokus på kurser inom informatik. Vi trodde att företag idag vet om att en detaljrik analys av ett projekt ger bättre slutresultat och mindre underhåll, precis som Göteborgs universitet utbildar, men att företag trots detta inte väljer att arbeta efter dessa metoder. Denna uppsats är en kvalitativ empirisk studie som innebär att vi har intervjuat fyra stycken olika företag i Göteborgsregionen. Dessa intervjuer låg till grund för jämförelsen mellan företagen och även jämförelsen mellan de fyra universitet vi valt att undersöka. Vi tar reda på med hjälp av intervjuer, de olika företagens arbetsmetoder inför ett systemutvecklingsprojekt. Om man sätter företagen i förhållande till vad universitet idag undervisar är en av de största skillnaderna att deras förstudier inte ens är hälften så stor som vi har fått lära oss. Att de inte heller tänker ut problemområdet i samma utsträckning och att de inte delar in arbetsmomenten i klasser, objekt, händelser och tillstånd.

Nyckelord

Objektorienterad Analys, UML, Systemutveckling,
RUP, Designdokument, Analysdokument

Författare: Charlotte Carlsson, Therese Olofsson
Handledare: Alan B Carlson
Examensarbete I, 10 poäng

TACK OCH NIG

Vi vill tacka alla som på något sätt varit i kontakt med oss under framtagningen av denna uppsats.

Främst vill vi tacka de företag som så snällt ställt upp och tagit sig tid:
Mikael Skogström på WM-data Raindance
Carol Wittgren och Susanne Källefjärd på WM-data Sverige AB
Camilla Johansson och Andréas Bengtsson på COGZ
Mikael Alveberg, Niklas Rhöse och Anders Wideskott på Republic.

Vår handledare Alan B Carlson för stöd genom hela uppsatsen.

Vänner och familj som under tiden trogen läst igenom uppsatsen.

Vi hoppas ni kommer få en trevlig läsning,

Charlotte Carlsson och Therese Olofsson

INNEHÅLLSFÖRTECKNING

1 INLEDNING

1.1 Bakgrund

Idén till vår uppsats kom genom kontakter i arbetslivet, vi funderade över hur arbetssätt på olika företag skiljer sig emellan och hur dessa sätt skiljer sig med de metoder och teorier universitet idag undervisar. Vi valde att intervjua företag för att få svar på frågan om hur deras arbetsmetoder skiljer sig. För att ta reda på vilka metoder som undervisas på universitet valde vi att studera vilka kurser samt vilken litteratur som används på olika universitet i Sverige.

Vet företag idag att en detaljrik analys av ett projekt ger ett bättre slutresultat, och trots detta väljer att dra ner på den fasen för att kunden inte vill betala eller på grund av tidsbrist. Men varför väljer företag att minska på analysdelen när metoder som undervisas på universitet visar att en väl genomarbetad analysfas är lösningen på en bra slutprodukt som kräver mindre underhåll?

Det kan vara mycket svårt och tidskrävande att välja vilket modelleringsspråk man ska arbeta med för att få fram en bra analys och design. Idag finns det ett modelleringsspråk som ofta används som standard vid systemutveckling, Unified Modeling Language eller som det mest är känt för UML. På universitet är det även där vanligt att denna standard lyfts fram som det bästa modelleringsspråket.

Mathiasen, Munk-Madsen, Nielsen och Stage (2001) menar att objektorienterad analys handlar inte om programmering, det handlar om analysen och dokumentationen innan programmeraren sätter tänderna i självaste utvecklandet.

Diskussionerna om metodologi är den principiella utgångspunkten för objektorienterad analys, medan den praktiska utgångspunkten är det problem som skall lösas. Systemdefinitionen klargör vad systemutvecklarna skall skapa till användarna (Mathiasen et al., 2001)

1.2 Syfte och frågeställning

Syftet med studien är i första hand att vi vill med kunna se skillnader och likheter med de metoder och arbetssätt som idag undervisas på universitet med fokus på kurser inom informatik. Syftet är också vårt intresse för hur företag arbetar inom objektorienterad analys och systemutveckling idag. Uppsatsen vänder sig till de företag vi valt att studera samt studerande och lärare på Göteborgs Universitet. Huvudfrågan för uppsatsen är:

Hur skiljer sig arbetssätt och metoder mellan företag och hur skiljer sig dessa företag med de teorier universitet i Sverige undervisar?

1.3 Avgränsning

Vi har valt att endast använda oss av litteratur från kurser på Göteborgs Universitet, institutionen för Informatik. Mathiasens modeller är det som vi har valt att betona som den korrekta efter en diskussion med vår handledare. Det är även de modeller som undervisas på Institutionen för Informatik. De övriga kurser vi slutligen valde att ta med är återfinns på Umeå universitet, Växjö universitet samt Högskolan Dalarna. Detta val baserades på geografisk placering, storlek på universitet samt vilket kursutbud universiteten har. Vi har också valt att avgränsa oss till att intervjua endast fyra företag i Göteborgsregionen. Detta för att studien endast pågår under tio veckor begränsar tiden oss till att intervjua fler företag. Anledning till att vi valde Göteborgsregionen var för att vi ville träffa personerna på företagen personligen.

2 METOD

Denna uppsats är en kvalitativ empirisk studie som innebär att vi har intervjuat fyra stycken olika företag i Göteborgsregionen. Dessa intervjuer ligger till grund för jämförelsen mellan företagen och även jämförelsen mellan de fyra universitet vi valt att undersöka. Intervjuerna tar reda på de olika företagens arbetsmetoder inför ett systemutvecklingsprojekt.

Vi har även valt att använda oss av den studentlitteratur som har används igenom vår utbildning på Informatik i Göteborg och har därigenom kunnat göra jämförelsen mellan universitet och arbetsliv. Vi har också valt att undersöka vilken typ av litteratur och vilket utbud av kurser som finns på andra universitet

Anledning till att vi valde att intervjua de olika företagen beror på att en intervju ofta leder till följdfrågor och diskussioner, vilket ofta resulterar i väldigt innehållsrikt resultat. Alternativet skulle vara en enkätundersökning, vilket kräver en noggrann studie för att arbeta fram rätt frågor. Även en genomarbetad enkät leder ofta till missad information, om det inte är så att studien och resultatet skall vara kvantitativ.

2.1 Kvalitativ

Vi har valt att använda oss av en kvalitativ metod för att få en större förståelse om hur systemutvecklingsmetoder tillämpas hos olika företag. Anledningen till att vi valde en kvalitativ undersökning före en kvantitativ undersökning beror på att en kvantitativ studie enligt Backman (1998) resulterar oftast i numeriska observationer genom till exempel experiment, prov eller enkäter. En kvalitativ undersökning däremot går ut på att man undersöker ett fenomen i sin realistiska miljö, där fenomen och kontext inte är givna.

2.2 Förberedelser av intervjuer

Det finns en mängd olika sätt att förbereda en intervju. Vårt tillvägagångssätt var enkelt. Vi läste den litteratur som vi använt under våra kurser, med fokus på boken Objektorienterad analys och design (Mathiasens 2001). Anledning till att vi valde Mathiasen var för att han använder sig av de grundläggande metoderna som finns för en analysering med tillhörande dokumentation skall bli lyckad. Mathiasen använder sig även av UML som modeleringsgrund, vilket är en standard som idag växt sig ganska stark. Vi gick igenom de punkter litteraturen tog upp för att uppnå en lyckad modulering och omformulerade dessa punkter till frågor som vi förväntades oss ge informativa svar.

Att använda en pilot-intervju ansåg vi inte skulle ha någon större påverkan på de frågor vi valt att ställa. Vi valde istället att diskutera våra framarbetade frågor med andra elever på universitetet eller med andra närstående personer i vår omgivning och lät dem komma med synpunkter och eventuella svar.

När vi kontaktade företagen var samtliga mycket tillmötesgående och ville, efter att vi förklarat vår frågeställning givetvis hjälpa till. Intervjutiderna bokades ganska omgående efter att kursen startat. Vi valde att inte skicka ut frågorna till företagen innan, utan bad dem endast att fundera på vilka arbetssätt de använde inför, under och efter ett systemutvecklingsprojekt. För att underlätta för oss själva och för att inte gå miste om någon information valde vi att spela in intervjun på band för att sedan kunna transkribera intervjun i efterhand. Genom en

inspelning kunde vi också fullt fokusera oss på att rätt frågor och följdfrågor ställdes till företaget.

2.3 Val av företag

Vi ville göra en studie som innefattade olika sorters företag för att få ett större omfång på informationen. Genom att använda oss av företag inom systemutveckling med något skilda arbetsuppgifter och av olika storlek, med antal anställda räknat hoppades vi få studera olika analysmetoder. Vi gick först igenom vilka olika systemutvecklingsföretag som finns i Göteborgsregionen och kom fram till att många företag idag arbetar mycket med webbaserade system, men vi ville även komma i kontakt med företag som utvecklar egna mjukvaror. Resultatet blev:

- WM-data Sverige – Ett stort företag med många anställda som sysslar främst med webbutveckling, men även mjukvaruutveckling om ett sådant projekt dyker upp. Intervju skedde med Carol Wittgren, projektledare på WM-data Sverige AB inom affärsintegration dvs enterprise information portals, intranät, extranät och webbsidor med kopplingar mot diverse affärssystem. Utöver projektledning arbetar Carol Wittgren med analyser, förstudier, framtagning av kravspecifikationer för kunds räkning. Vid Intervjun medverkade även Susanne Källefjärd, systemutvecklare främst inom dotNet, och har den senaste tiden arbetat med intranät- och internetlösningar i dotNet.
- Republic Factory – En liten webbyrå med stor kunskap som sysslar uteslutande med webbutveckling. Intervju skedde med Mikael Alveberg, Niklas Rhöse och Anders Wideskott på Republic Factorys kontor.
- WM-data Raindance – De sysslar uteslutande med mjukvaruutveckling av produkten Raindance som är ett affärssystem till försäljning. Intervju skedde med Mikael Skogström, projektledare för Raindance användargränssnittsgrupp på Cross Industry Solutions AB ett bolag inom WM-data koncernen.
- COGZ – Ett litet företag bestående av kognitionsvetare. De arbetar tillsammans med utvecklingsföretagen och slutkunden för att rätt produkt skall tas fram. Intervju skedde med användbarhetskonsulterna Camilla Johansson och Andréas Bengtsson.

2.4 Val av universitet

Anledningen till att studera fler universitet än Göteborgs universitet var för att får en uppfattning om vad man undervisar om inom objektorienterad analys och design samt vilken litteratur man använder. Resultatet blev en sammanställning av utvalda universitet där fokus var på att koncentreras på kursinnehåll samt kursernas litteraturlistor. Mathiasen et al. (2001) återfanns på en mängd olika universitet, men dessa universitet hade även annan litteratur inom ämnet, som t ex. Checkland et al (1998) och Fagerström et al (1998). Många av kurserna utbildar i metoderna RUP och UML som ett sätt att lösa olika problemområden. Genom att fokusera på kurser inom ämnet informatik kunde vi lättare jämföra de olika universiteten. Vi studerade först en mängd olika universitet som låg placerade runt om i landet. För att få med ett så brett innehåll som möjligt valde vi att ta med kurser som innehöll annan litteratur än Mathiasen et al. (2001). För en fullständig kursreferens se bilaga C.

2.5 Uppsatsens trovärdighet

Det är viktigt att de företag som svarar på frågorna är insatta i ämnet och förberedda på att vi skall komma. Det vill säga att företagen måste ha avsatt tid och ha en vilja till att dela med sig av sin information och kunskap. Detta var något som vi tyckte att samtliga företag ställde upp på. Under samtliga intervjuer fanns det inga störelsemoment vilket ledde till att svaren blev noga genomtänkta.

3 TEORI

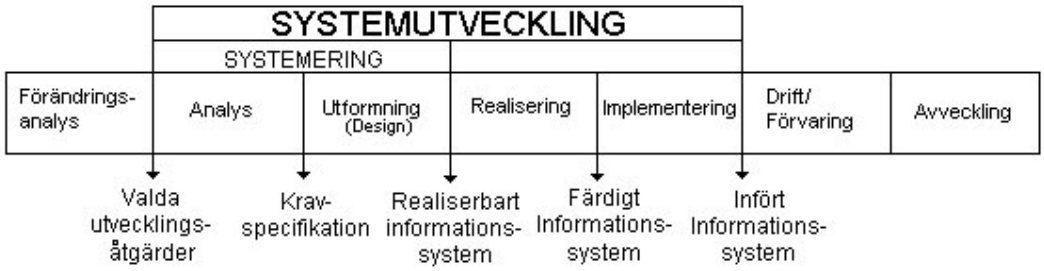
3.1 Modell

En modell är en förenklad representation av en komplex verklighet som vanligtvis används för att få en förståelse av verkligheten och av att få karaktäristiska drag som är nödvändiga för uppgiften eller problemet. (Brown, 2002)

3.1.1 Livscykelmodell

Livscykelmodellen är en typ av utvecklingsmodell som speglar hela livscykeln för ett systemutvecklingsprojekt, från analys till förvaltning till avveckling. Varje fas i livscykelmodellen mynnar i ett resultat som leder vidare till nästa steg. I förändringsanalysen läggs grunden för arbetet för utvecklingen av systemet. I faserna Analys, Utformning (Design) och Realisering är de tre faser som modelleringsspråket UML används. (Andersen, 1994)

Livscykelmodell:



Figur 1. Livscykelmodell (Andersen, 1994, s. 46)

3.2 Rational Unified Process - RUP

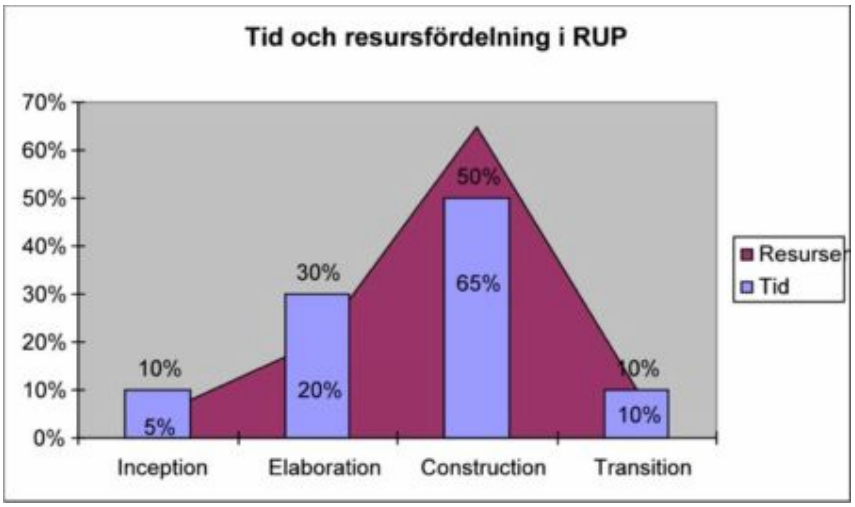
RUP är idag enligt Lunell (2003) en välkänd och etablerad process bland dem som sysslar med programvarokunstruktion. Många stora organisationer använder redan RUP eller ’r på god väg att gå över till detta arbetstätt. RUP grundar sig på det objektorienterade paradigmet och inkorporerar mycket av bästa praxis för systemytveckling enligt Lundell (2003). RUP är en heltäckande process som inkluderar både tekniska och stödjande aspekter på utvecklingen och är numera en referenspunkt för alla diskussioner om systemutvecklingsprocesser (Lunell, 2003). Det är också viktig att komma ihåg enligt Lunell (2003) att RUP är en kommersiell produkt som utvecklas och marknadsförs av Rational Software Corporation. Det är en produkt under ständig utveckling och nya utgåvor kommer regelbundet.

Utvecklingen av det som idag är Rational Unified Process är sprungen ur utveckling och användning av ett flertal metoder och arbetssätt. 1987 släpptes Objectory Process vilken senare utvecklades till Rational Objectory Process 1997 och senare Rational Unified Process 1998. Många källor har influerats av det som idag är RUP, utan att försöka identifiera alla kan några av intresse nämnas: 1987 lämnade svensken Ivar Jacobsson företaget Ericsson för att i egen regi jobba med Objectory, 1995 anslöt sig Ivar till Rational och kom tillsammans med

Grady Booch och James Rumbaugh att leda arbetat med att forma det som senare blev RUP (Fägnell, 2003).

RUP delar in hela projektet i fyra olika faser och kan grovt sammanfattas med följande fyra fasers beskrivning:

1. *Inception* (Förberedelse) - Skapa en gemensam vision
2. *Elaboration* (Etablering) - Skapa en arkitektur
3. *Construction* (Konstruktion/Bygg) - Skapa en produkt
4. *Transition* (Överlämnande) - Installera och leverera



Figur 2. En beskrivning av tid och resursfördelning i ett RUP projekt (Fägnell, 2003).

3.2.1 Inception

Syftet i den första fasen är att alla inblandade ska vara överens om vad produkten ska göra, hur lång tid det tar att utveckla den och vad det kostar. Detta ska utgöra ett tillräckligt underlag för att fatta beslut om projektet ska läggas ned eller om man ska gå vidare (Kruchten, 2002). Syftet är dock att inte göra en fullständig kravspecifikation utan istället få en grov uppfattning om projektets omfång.

Enligt Lunell (2003) ägnas en del av denna fas åt en första ansats till kravanalys. Kravspecifikationen består av bland annat Vision, Use-Case Model, User-Interface Prototype, och Supplementary Specification. För alla dokument gäller att ingenting ska vara med som inte tillför något av vikt. Inget av dokumenten är slutgiltigt, de kommer att ändras i kommande iterationer (Kruchten, 2002). Därförutom bör prototyper avfattas för att försäkra sig om att det man planerar göra är möjligt.

Kravspecifikation:

- Vision
  Detta är den mest övergripande artefakten. En artefakt är ett föremål skapat av en människa. Synonym till apparat inom teknologin. Den innehåller en beskrivning av vilka roller som har intressen i produkten, vilka dessa intressen är samt vad systemet är till för och vad den ska utföra. Tyngden av denna punkt är att alla inblandade ska få en gemensam syn på vad arbetet går ut på.Use-Case Model

På ett program ställs både funktionella krav och ickefunktionella krav så som säkerhet, prestanda, användarvänligt och enkelt underhåll. I use-case:n kompileras alla funktionella krav och eventuellt även ickefunktionella krav som är kopplade till något specifikt use-case (Kruchten, 2002). I fasen Inception, är det lämpligt att identifiera så gott som alla use-case.och aktörer till namn.

Ett use-case är en text. Den viktigaste delen av ett use case är scenariona. Det är viktigt att skriva dem noggrant och punkt för punkt reda ut vad som skickas till och från systemet. Dessa punkter kommer att utgöra grunden för all programmering.

Några definitioner:

Actor - Någonting som kan göra något med produkten, till exempel en person eller ett annat datorsystem.

Scenario - En följd av händelser som leder genom ett use-case. Om det finns ett use-case som man kallar för ”*ta ut pengar*” för en bankomat, är det till exempel ett scenario att allt går bra, ett annat att användaren slår fel kod, ett annat att det inte finns pengar på kontot.

Use-Case - En samling av alla scenarion som beskriver hur produkten används för att utföra en viss operation (Kruchten, 2002).

- Supplementary Specification
  Denna artefakt behandlar de ickefunktionella kraven. Den ska innehålla övergripande krav som inte täcks i use-case-modellen. Supplementary specification ska innehålla problemen, inte lösningarna på dem (Kruchten, 2002). Exempel på frågor som bör beaktas här är säkerhet, prestanda, tillgänglighet, standarder som ska följas och krav på paketering och underhåll.

- User-Interface Prototype
  Med ledning av främst use-case-model men även supplementary specification byggs en user-interface prototype som innehåller detaljerade förslag på användargränssnitt. Det är viktigt att kunden tittar noga på dessa eftersom det även är ett bra sätt att upptäcka brister i systemets funktionalitet. "I värsta fall" ritas de på papper men det är absolut att föredra att de utvecklas i skarp kod (Kruchten, 2002).

Enligt Lunell (2003) är det i inceptionsfasen också viktigt att så många risker som möjligt är kända. Här antecknas och graderas riskerna, sedan gäller det att ta itu med dem så snart som möjligt.

Det är mycket lämpligt att bygga en proof-of-concept protoype. Syftet är att övertyga sig om att de tilltänkta teknologierna är användbara och om man behärskar dem. Prototypen ska försäkra att man klarar att utveckla systemet från användargränssnitt till databas. Den skall vara av typen throw-away, dvs. man ska kunna kasta förslaget utan några vidare komplikationer (Kruchten, 2002).

3.2.2 Elaboration

Huvudsyftet med Elaborationsfasen är att få fram en stabil och trovärdig arkitektur baserad på de arkitekturellt signifikanta användningsfallen (Fägell, 2003). Detta åstadkoms genom att implementera så mycket av produkten i färdig kod att:

- Arkitekturen är klar. För detta krävs att det kodas något i alla "hörn" av produkten.

- Att alla kritiska use-case fungerar. Med kritiska menas både sådana som innehåller tekniska risker och sådana som innehåller nödvändig funktionalitet.
- Det går att göra en bra uppskattning om hur mycket tid och resurser som krävs för resten av projektet.

Dessutom ska man ha identifierat mer eller mindre alla krav på produkten, dvs. alla dokument som påbörjades i inception ska vara i det närmaste färdiga. Den kod som skrivs under elaboration ska inte kastas bort senare, den ska vara med ända till skarp drift (Kruchten, 2002).

Fasen består nästan alltid av flera iterationer. Hur många iterationer och hur långa dessa ska vara beror på projektets karaktär. Med kortare iterationer fås bättre koll på vad som faktiskt är klart och fungerar, man kan lättare ändra i den befintliga koden och snabbare hitta fel (Fägnell, 2003). Alla artefakter som påbörjades under inception uppdateras under elaboration, när fasen är slut är alla artefakter i fas och speglar projektets aktuella status. Vilka funktionaliteter som ska implementeras under en viss iteration anges i iterationsplanen (*iteration plan*) (Kruchten, 2002). Iterationsplanen ska bara innehålla en detaljerad beskrivning av arbetet i aktuell iteration.

Arbetet under en iteration består av dessa aktiviteter:

1. *Kravspecificering*
   En kort genomgång av kravspecifikationen. Här finns möjlighet att uppdatera den med föreslagna ändringarna. Dessa ändringar kan vara både nya förslag från kunden och förändringar tvecklaren kommit på själva under arbetet i föregående iteration.
2. *Analys och design*
   Syftet med analys är att omvandla kravspecifikationerna till en form mer lik mjukvara (klasser, delsystem osv). Analysen fokuserar på de funktionella kraven, dvs på en ideal bild av systemet där den krassa verkligheten inte gör sig påmind. Detta innebär att indata till analysen främst är use-case-modellen. Design tar vid där analysen slutar. Syftet är att omvandla analysresultatet till en modell med konkreta klasser och metoder som ska konstrueras i programmet.
3. *Implementation*
   Här implementeras designresultatet i konkret kod. Färdigtestad kod integreras med de delar av systemet som tagits fram i tidigare iterationer.
4. *Test*
   Tätt sammanvävt med kodning är testning. Här testas allt ifrån delar av systemet och klasser till funktionalitet och prestanda (Kruchten, 2002).

3.2.3 Construction

Huvudsyftet med Constructionfasen är att bygga systemet enligt de kriterier som definierades i de två tidigare faserna. Bland de primära målen bör man ha hittat några av dessa punkter:

- Optimera och planera resurser för att undvika dubbelarbete.
- Arbetet med att uppnå rätt kvalitet tidigt i fasen.
- Att hålla leveransplaner för de releaser som planerats (Fägnell, 2003).

Alla tekniskt komplicerade delar av koden ska vara klara och den för kunden viktigaste funktionaliteten ska vara färdig. Trots att construction tar längre tid än elaboration ska de svåra besluten redan vara fattade (Kruchten, 2002).

Under fasen construction utförs tre uppgifter:

- *Utveckla dokukmentation* - Skriva online-hjälp, användarmanual och eventuellt kursmeterial.
- *Sluttest* - av hela produkten internt. Produkten måste klara detta test innan fasen construction är slut.
- *Skapa betarelease* – Installationsfiler, installationsanvisningar, release notes, licensavtal osv. (Kruchten, 2002).

Innan Constructionfasen kan avslutas bör produkten fungera stabilt, utan att skapa nya problem och risker för projektet (Fägnell, 2003).

3.2.4 Transition

Syftet med transition är just att få ut produkten till kunderna, dvs. att ta steget från att utvecklarna är nöjda till att kunderna är nöjda (Kruchten, 2002). Bland de primära målen bör man enligt Fägnell (2003) ha hittat några av dessa punkter:

- Mottagaren av systemet har en produkt de kan hantera själv utan hjälp av extern support.
- Kravställare är överens om att vision och slutprodukt stämmer överens mot uppsatta ramar.

För att nå dessa primära mål krävs det att i denna fas identifiera och formulera förutsättningarna för projektet och dess eventuella fortsättning. Aktiviteterna kan enligt Fägnell (2003) vara:

- Installera och testa produkten på plats.
- Fixa fel i kod och dokumentation som upptäckts under testning.
- Ändra i koden för att förbättra prestanda.
- Skapa slutgiltig release.
- Utbilda användarna.
- Säljaktiviteter.

3.3 Unified Modeling Language - UML

I början av 90-talet innebar begreppet objektorienterade metoder en mängd variationer. Dessa variationer skapade förvirring och osäkerhet. Ett standardmodelleringsspråk arbetades fram och 1997 lanserades UML (Unified Modeling Language) av OMG (Object Management Group). UML täcker notationsbehoven i den objektorienterade utvecklingsprocessen. En notation är det sätt man representerar tex en matematisk ekvation. En stor fördel med UML är att man kan välja att använda alla konstruktioner eller bara vissa delar av notationen. I denna uppsats har vi valt att inrikta oss på vissa delar av UML, de delar som litteraturen Objektorienterad Analys och Design (Mathiasen, 2001) tar upp. Detta för att litteraturen täcker de relevanta delarna för analys och design.

Eftersom UML är ett grafiskt språk krävs det en standard för vad olika symboler betyder, bestämmelser för hur man tecknar en klass, ett objekt och så vidare. Det finns stereotyper att följa enligt UML-mallen, men det betyder inte att man kan utveckla dem vidare. Detta kan ibland skapa förvirring för framtida bruk, något som man måste ha i åtanke om man väljer att ta ett sidospår.

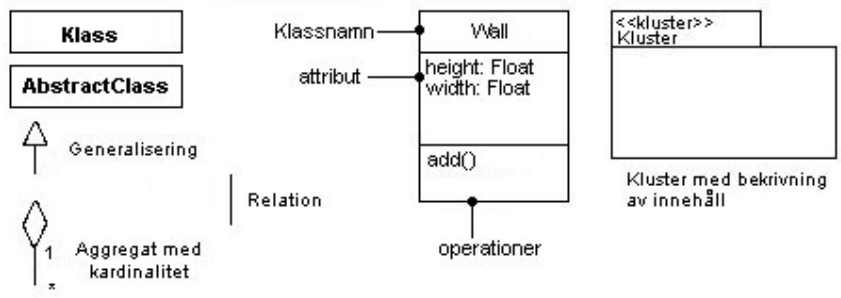
UML består av nio olika komponenter där varje komponent bildar ett diagram.

- *Klassdiagram:* Visar de statiska som finns vid design av ett system.
- *Objektdiagram:* Visar en mängd objekt och deras relationer.
- *Användarfallsdiagram:* Visar en mängd användarfall och deras samspel med externa aktörer.
- *Sekvensdiagram:* Visar interaktionen mellan en mängd objekt
- *Aktivitetsdiagram:* Visar flöden av aktiviteter mellan objekten
- *Samarbetsdiagram:* Visar hur man strukturellt organiserar de objekt som sänder och tar emot meddelanden.
- *Tillståndsdiagram:* Visar vid dynamisk beskrivning av objekt, de olika tillstånd, aktiviteter, tillståndsväxlingar som kan uppträda.
- *Komponentdiagram:* Visar samspelet mellan programkomponenter.
- *Fördelningsdiagram:* Visar hur ett system är konfigurerat över processorer, vilka programkomponenter som placeras var etc.

(Apelkrans och Åbom, 2001)

Ett klassdiagram är en av de centrala delarna i UML. Oftast räcker det i början av analysen att rita ut klasser med endast klassnamn, men i vissa fall kan det vara bra att redan i ett tidigt stadium göra en tydlig klassbeskrivning. En klass är en beskrivning av en samling objekt som delar struktur, beteende, mönster och attribut. Olika klasser sätts i relation till varandra där kardinaliteten beskrivs. En kardinalitet är antalet händelser av varje entitet som är involverad i en relation. (Brown, 1997)

Grundnotation för klassdiagram:

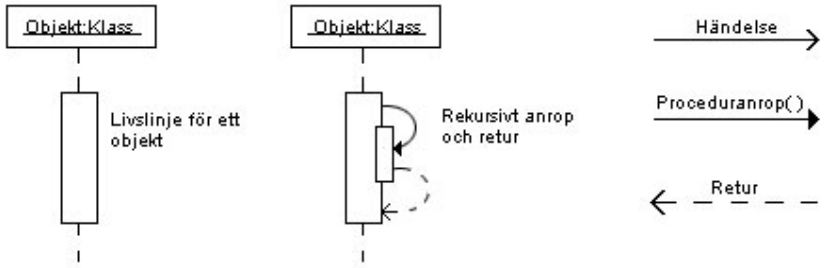


Figur 3. Grundnotation för klassdiagram. (Booch, J., Rumbaugh, J. och Jacobson, I. 1999)

Om man vill representera ett objekt använder man sig av samma symboler som för klasser förutom att man stryker under objektnamnet samt dess klass.

För att komplettera ett klassdiagram kan man använda sig av ett sekvensdiagram. Ett sekvensdiagram består av en beskrivning av den generella, statiska situationen. Syftet med diagrammet är att man visar interaktionen genom meddelanden som skickas mellan objekten.

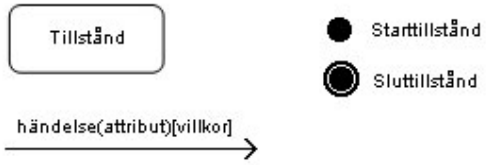
Notation för sekvensdiagram:



Figur 4. Notation för sekvensdiagram. (Mathiasen et al., 2001 s. 387)

Tillståndsdiagrammen är en del av UML som beskriver beteenden hos samtliga objekt.

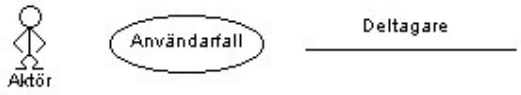
Grundnotation för tillståndsdiagram:



Figur 5. Grundnotation för tillståndsdiagram. (Mathiasen et al., 2001, s. 388)

För att beskriva de olika rollerna i systemet använder man sig av ett användarfallsdiagram. Ett användarfallsdiagram visar de relationer som finns mellan aktörer och användarfall.

Notation för användarfallsdiagram:



Figur 6. Notation för användarfall. (Mathiasen et al., 2001, s. 390)

UML är inte själva grunden till objektmodelleringen utan endast en av alla tekniker för att objektmodellera. Anledning till att vi valt att fokusera på UML är för att den är en av den mest kända, samt att de kurserna på olika universitet vi valt att studera använder sig av böcker med denna standard.

3.4 Metod för framtagning modeller

Den metod som Mathiasen et al. (2001) använder bygger på objekt och klasser som nyckelbegrepp. *Modellera systemets omgivning, betona arkitekturen, återanvänd mönster* och *anpassa metoder* är fyra principer som ligger till grund för framtagning av analys och design. Att *modellera systemets omgivning* handlar om att man modellerar det problemområde som utnyttjas av användningsområdet. Problemområdet är den omgivning som sköts av systemet

och användningsområdet är den organisation som styr problemområdet. En enkel arkitektur som Mathiasen et al. (2001) använder som exempel är:

Figur 7. Enkel beskrivning av en systemarkitektur. Funktionskomponenten styr så att användaren kan uppdatera modellkomponenten. Gränssnittskomponenten är skärm, text och grafik som används för att koppla systemet till användaren och omgivningen (Mathiasen et al., 2001).

Att *betona arkitekturen* innebär att fokusera på lättbegriplighet, flexibilitet och användbarhet *Återanvänd mönster* är en viktig del av framtagningen av systemet. Det är vanligt att man återanvänder mönster som man tidigare använt och som man vet fungerar. Sista och en av de viktigaste delarna, *anpassa metoden* består av att behandla informationsinnehållet, hur systemet skall användas, systemet som helhet och systemets komponenter. Det är viktigt att tänka på dessa delar då man måste anpassa metoderna utifrån vilken typ av organisation man ska arbeta med (Mathiasen et al., 2001).

Objekt är nyckelordet, som kan beskrivas mer ingående med hjälp av en *klass*. Ett objekt är en entitet med identitet, tillstånd och beteende. Detta innebär att ett objekt kan vara en kund, som har ett kundnummer (identitet), vara aktiv (tillstånd) och ha ett beteende. En klass är en plats där man kan samla alla objekt som är av samma struktur, beteendemönster och attribut.
Det finns olika typer av objekt, analys- och designobjekt. Analysobjektet uttrycker beteenden genom de händelser det deltar i, men ett designobjekt utrycker beteenden hos ett objekt genom de operationer det kan utföra och göra tillgängligt för andra objekt (Mathiasen et al., 2001).

Resultatet som man arbetar mot är att ta fram en fullständig dokumentation. Dokumentationen skall kunna användas av utvecklare, slutanvändare och projektledare. Mathiasen et al. (2001) förespråkar att dokumentationen skall bestå av två delar, Analys- och Designdokument, men påpekar också att detta absolut inte är ett måste för en lyckad dokumentation (*Se bilaga A).*

3.5 Analysfasen

Det första steget är att ta fram ett analysdokument. Detta dokument kommer senare att ligga som grund för designdokumentet. I korthet består analysdokumentet av uppgiften, problemområdet, användningsområdet och rekommendationer (Mathiasen et al., 2001). För diagram och beskrivningar se, Bilaga D.

3.5.1 Uppgiften

Systemutvecklingsprocessen startar när man har en idé om vad systemet ska göra.
För att förenkla detta problemområde underlättar det om man förstår strukturen, relationerna och detaljerna i den organisation som systemet skall användas i. Den första delen av dokumentet skall vara max en sida långt och innehålla grundtanken, syftet till systemet samt kort presentera problem- och användningsområde. För att arbeta fram en systemdefinition används begreppet VATOFA.

Villkoren för utveckling och användning av systemet
Användingsområde för systemet
Teknologin för implementering av systemet
Objekt som ingår i systemets modell av problemområdet
Funktionalitet
Ansvar i förhållande till omgivningen
(Mathiasen et al. 2001, s. 58)

Var alltid öppen för nya idéer och arbeta fram dessa tillsammans med användarna. Som grund för idéer kan man använda system som redan finns, metaforer, prototyper och experiment för att studera användarnas vardag (Mathiasen et al., 2001).

När man nått så här långt i framtagningen är det dags att ta beslut om vilket typ av system man skall utveckla. Det naturliga känns kanske att systemutvecklarna tar detta beslut, men de ska endast finnas med som stöd för att underlätta valet (Mathiasen et al., 2001).

3.5.2 Problemområdet

Vilken typ av information skall systemet hantera? Genom att *modellera* svaret på denna fråga behöver inte användaren tänka på de tekniska aspekterna utan kan fokusera på att förstå problemområdet. Själva analysen av problemområdet kan delas in i tre delar: objekt, klass och händelse.

Ett objekt är som vi nämnt tidigare är en entitet med identitet, tillstånd och beteende. Det viktiga med framtagningen är att vi kan identifiera och avgränsa objektet som en oberoende entitet (Mathiasen et al., 2001).

En klass är en beskrivning av en samling objekt som delar struktur, beteende, mönster och attribut. Klassen illustreras som ett substantiv med relation till problemområde och organisation. Det bästa sättet att arbeta fram substantiven är med hjälp av brainstorming där deltagarna fritt får säga ord som förknippas med organisation, problemområde och arbetssituationer. När detta är gjort har man ett antal klasser som går under begreppet kandidatklasser. Alla kandidatklasser diskuteras, övervägs och väljs ut och namnges sedan med ett lättläst och lättförståligt klassnamn. För att klasser och klassnamn i framtiden inte skall skapa några missförstånd bör man skapa en klassbeskrivning som innehåller en kort beskrivning av samtliga klasser (Mathiasen et al., 2001).

En händelse är en momentan tilldragelse i vilken ett eller flera objekt är inblandade. Precis som med framtagning av klasser använder man sig av brainstorming, fast en händelse är inte ett substantiv utan ett verb. Det är även här viktigt att diskutera, överväga och välja ut för att sedan namnge med ett bra namn (Mathiasen et al., 2001).

För att utvärdera både klasser och händelser använder man sig av generella och specifika utvärderingskriterium. En klass eller händelse bör tas med i systemet om och endast om systemfunktionerna använder informationen om den. För att sammanställa klasser och händelser ska man ta fram en händelsetabell. För att komma fram till detta ställer man frågorna: Vilka händelser är denna klass inblandad i? Vilka klasser är inblandade i denna händelse?

När alla klasser och objekt är framtagna är nästa steg att fokusera på relationerna mellan dessa. När hela strukturen är klar finns ett första klassdiagram framtaget.

Klasstruktur
En klasstruktur förbinder klasser med varandra och dess relation förändras inte. För att beskriva de begreppsliga relationerna mellan två klasser använder man sig av generaliseringstruktur, ett antal klasser som specialiserar av en mer generell klass och klusterstruktur som grupperar samling av relaterade klasser (Mathiasen et al., 2001).

Att generalisera en klass innebär att man beskriver egenskaper gemensamma för en grupp av specialiserade klasser. Detta är en så kalla ”är-en” relation. (Taxi ”är-en” Bil).

”Allting som gäller för den generella klassen gäller också för de specialiserade klasserna, medan egenskaperna hos en specialiseringsklass bara gäller för just den klassen.” (Mathiasen et al., 2001, s. 96)

Ett kluster innebär en samling klasser som på något vis har en logisk relation. Det är inte tillåtet att placera samma klass i två olika kluster. Känns klassen rätt på flera ställen bör den ligga i ett nytt kluster (Mathiasen et al., 2001).

Objektstruktur
En objektstruktur förbinder objekt med varandra och dess relations *kan* förändras utan att det sker någon förändring i beskrivningen. Det finns två olika slags objektstrukturer; aggregat- och associationsstruktur (Mathiasen et al., 2001).

Aggregat är en helhet som består av delar och kan beskrivas med en ”har-en”-, ”är-en-del-av”- eller ”ägs-av”-relation. Aggregatstrukturen beskriver relationen mellan ett objekt och dess delobjekt.

Association beskriver hur olika objekt förhåller sig till varandra utan någon rangordning. Relationen skall kunna beskrivas med ”känner” eller ”hör-ihop-med”.

Beteende
Det grundläggande syftet med ett system är att registrera, lagra och producera information om händelser som finns i problemområdet. Målet med att studera olika beteenden är att ta fram ett tillståndsdiagram samtidigt som man utvidgar klassdefinitionerna i klassdiagrammet (Mathiasen et al., 2001).

De händelser som är gemensamma kan man placera i en tabell, en händelsetabell. En händelsetabell skapar en övergripande förståelse för vilka klasser som förhåller sig till olika händelser (se bilaga D).

3.5.3 Användningsområdet

Hur kommer målsystemet att användas? Genom att bestämma användningsområdet med hjälp av användarfall och användare kan man relativt smidigt få fram svaret på den frågan. Resultatet av ett användarfall är ett användarfallsdiagram med en aktörstabell. (se bilaga D) Ett sådant diagram består av aktörer, användarroller och andra system som skall interagera med systemet samt användarfall som består av själva mönstret mellan systemet och aktörerna. För bästa resultat bör någon eller några användaren delta i framtagningen.

En aktör är den roll som finns i systemet. När alla aktörer är framtagna skriver man en aktörsspecifikation som består av syfte, beskrivning och exempel. (se bilaga D)
De användarfall man väljer att plocka fram bör vara av betydelse för de inblandade aktörerna (Mathiasen et al., 2001).

Funktioner
Vad ska systemet göra? En funktion är en del av systemet som gör en modell användbar för olika aktörer. En funktion aktiveras, utförs och ger ett resultat.
För att ta fram vilka funktioner som verkligen behövs i ett system behövs en grundlig analys av området genom att man studerar klasser och händelser. När denna analys är klar skall det finnas en fullkomlig lista på funktioner som överrensstämmer med användarfallen, en så kallad funktionslista (se bilaga D).

Funktionerna måste vara så pass detaljerade att den ger en bra överblick för både användare och utvecklare, men inte så detaljerad att det blir rörigt och oöverskådligt. De funktioner som är större, komplicerade eller som kräver en extra beskrivning bör ha just, en extra beskrivning. Denna beskrivning kan vara i ord, en algoritm eller en mer detaljerad och uppdelad funktionslista (Mathiasen et al., 2001).

Användargrässnitt
Användargränssnittet är det som användaren ser och använder för att komma åt funktionerna i systemet. En användare skall inte behöva tänka på vad som händer bakom gränssnittet, utan skall genom ett lättnavigerat och överskådligt gränssnitt kunna använda systemet till fullo. Framtagningen av gränssnittet är ett viktigt moment, speciellt ur användarens synvinkel. Ett bra användargränssnitt är anpassat till användarens arbetsuppgifter (Mathiasen et al., 2001).

När man arbetar fram användargränssnittet är det viktigt att välja vilken typ av dialogstil man skall använda sig av; menyval, formulärifyllning, kommandospråk och direkt manipulation.

- Menyval ger användaren möjlighet att välja olika alternativ genom en meny. Detta alternativ passa passar tillfälliga användare, men funkar även för regelbundna användare.

- Formulärifyllning är alternativet där man låter användaren fylla i information på skärmen genom ett formulär. Denna typ av stil är anpassat till regelbundna användaren då det krävs förståelse om vilken typ av information som skall fyllas i (Mathiasen et al., 2001).

- Kommandospråk är en stil som inriktar sig mot erfarna användare då användaren bestämmer själv, genom kommandon vad som skall visas på skärmen.

- Direkt manipulering innebär att objekt representeras som ikoner och användaren kan direkt manipulera med objekten. Konsekvenserna av manipulationen blir direkt synliga vilket gör att stilen passar bra till tillfälliga användare. Alla dessa stilar kan kombineras, men vanligast är att menyval och formulär kombineras (Mathiasen et al., 2001).

Gränssnittsanalysen bör bestå av en fullständig beskrivning av användargrässnittens och systemgränssnittens komponenter. Denna beskrivning bör/kan kompletteras med ett navigeringsdiagram för att ge en god överblick. (se bilaga D) Ett navigeringsdiagram visar vilka dialoger som kommer att finnas och hur de förhåller sig till varandra, ofta får alla centrala klasser ett eget användargränssnitt. (Mathiasen et al., 2001).

Teknisk plattform
Den tekniska plattformen skall beskrivas i textform. Detta för att man skall ha tänkt igenom utvecklingsmiljön och vilka kunskaper utvecklarna har (Mathiasen et al., 2001). T ex ”utvecklas för användning på PC i programmeringsspråket Java, systemet kommer att användas med mus och tangentbord”.

3.5.4 Rekommendationer
På den sista punkten av analysdokumentet är det tid för att argumentera för det fortsatta arbetet med utvecklingsarbetet. En genomgång av kraven för att studera om man verkligen klarar att genomföra en fortsättning av arbetet. Det är även aktuellt att uppskatta tid och kostnad för projektet (Mathiasen et al., 2001).

3.6 Designfasen
Designdokumentet är den andra och sista i dokumentationsdelen. Grundstommen i detta dokument kommer ifrån analysdokumentet. Designdokumentet består av uppgifter, teknisk plattform, arkitektur, komponenter och rekommendationer (Mathiasen et al., 2001).

3.6.1 Uppgiften
Uppgiften består av en kort beskrivning av uppgiften och de uppställda kvalitetsmålen som sattes i analysdokumentet. Här har man även tillfälle att göra rättelser och kompletteringar. Man skriver också ner de kvalitetsmål som en sammanfattning av designkriterier och kompletterade mål för arkitekturen (Mathiasen et al., 2001).

3.6.2 Teknisk plattform
Den tekniska plattformen består av framtagning och dokumentation av den tekniska plattform som skall användas under utvecklingen. Denna tekniska plattform skiljer sig från analysdokumentet genom att den är mer djupgående och detaljerad. Den innehåller en kort beskrivning av utrustning, basprogramvara, systemgränssnitt och designspråk (Mathiasen et al., 2001).

3.6.3 Arkitektur
Designen är en viktig del i systemet då en god design inte ska ha några större svagheter. Under flera år har forskare arbetat fram en lista med kriterium för vad som krävs för en god

design. Dessa kriterier kan absolut användas som grund för att ta fram designen: användbart, säkerhet, effektivt, korrekt, tillförlitligt, lättunderhållet, flexibelt, begripligt, återanvändbarhet, portabelt och interoperabelt (Mathiasen et al., 2001, s. 212). Angående användbarheten gäller det att betrakta systemet som en helhet utan att tänka på systemets inre struktur.

Komponentarkitektur
Användargränssnitt – Funktioner – Modell är en vanlig struktur i ett mindre system där varje framtagen komponent måste ha ett tydligt och väldefinierat ansvarsområde. Lite större system bör delar upp i delsystem, vilket innebär att man delar upp modellen, funktionaliteten och gränssnitt i mindre delar, komponenter.

De olika komponenterna:

- *Modellen*: Huvudansvar att inrymma de objekt som representerar problemområde.
- *Funktioner*: För att tillhandahålla modellens funktionalitet.
- *Gränssnitt*: Hantera interaktionen mellan aktörerna och funktionaliteten.

(Mathiasen et al., 2001, s. 224)

Processarkitektur
Att arbeta med processer går ut på att definiera den fysiska struktureringen av ett system. Resultatet blir ett fördelningsdiagram eller aktivitetsdiagram som visar fördelningen av och samarbetet mellan programkomponenter och aktiva objekt på olika processorer (Mathiasen et al., 2001, se bilaga D).

3.6.4 Komponenter

Hur förbinder vi komponenterna med varandra? Här är det mycket viktigt att tänka på, speciellt som utvecklare, att man måste hålla sig till den redan framtagna arkitekturen. För varje komponent skall man arbeta fram ett klassdiagram som beskriver klasserna och deras strukturella relationer samt ger namnen på deras attribut och icke-triviala operationer (se bilaga D).

Om det inte framgår sedan tidigare skall även varje klass beskrivas med klassnamn, en kort beskrivning av klassens ansvar och syfte, attribut, komplicerade operationer, med användning av en operationsspecifikation samt tillståndsdiagram för att beskriva klassens beteendemönster (Mathiasen et al., 2001, se bilaga D).

Modellkomponent
”En modellkomponentdesign är struktur. Modellen bör återspegla problemområdets relevanta begreppsliga relationer, den bör vara klar, och den bör vara både snabb och lätt att arbeta med.” (Mathiasen et al., 2001, s. 271).

En modellkomponent är en del av systemet som implementerar modellen av problemområdet. För att ta fram modellkomponenter använder man sig av modellen som togs fram i analysfasen i form av ett klassdiagram och ett tillståndsdiagram. Framförallt brukar man behöva utveckla tillståndsdiagrammen på grund av att fler händelser framkommer. För att kunna representera dessa händelser måste man lägga till fler attribut och nya klasser (Mathiasen et al., 2001)

För att ta fram en modellkomponentspecifikation arbetar man sig igenom följande steg. Man utgår ifrån de klassdiagram, beteendemönster och komponentspecifikationer som man arbetat fram under analysfasen. Representerar deras gemensamma händelser, privata händelser och

omstrukturerar klasserna. Detta resulterar i att man utvidgar det befintliga klassdiagrammet med nya klasser, attribut och strukturer (Mathiasen et al., 2001).

Funktionskomponent
Funktionskomponenten ser till att användargrässnittet och systemkomponenter får tillgång till modellen. Det är viktigt att *rita* dessa komponenter för att de inte skall bli för likt programmering. De operationer/funktioner som är lite mer avancerade bör specificeras för att skapa en bättre förståelse. Specifikationen består i de flesta fall av text, men i mer komplicerade fall kan man använda sig av sekvensdiagram och/eller tillståndsdiagram (Mathiasen et al., 2001).

3.7 Rekommendationer

Rekommendationer består av en övergripande utvärdering av hur designen förhåller sig till omgivningen, grundad på de uppställda kvalitetsmålen. Rekommenderad plan för hur systemet ska tas i bruk, realiseringen av systemet, med aktiviteter och uppskattningar av resurs- och tidsåtgång. Här skall även en kort text om en planerad implementering skrivas ner. Vanligen sker en implementering med testning, utvärdering, uppdateringar och en ny implementering av uppdateringar (Mathiasen et al., 2001).

4 RESULTAT

4.1 WM-data - Cross Industry Solutions AB, Raindance

Intervju med Mikael Skogström, projektledare för Raindance användargränssnittsgrupp på Cross Industry Solutions AB ett bolag inom WM-data koncernen.

Om företaget
Raindance är ett ekonomisystem som finns installerat hos cirka 600 kunder allt från stora industriföretag till små kommuner. Systemet används bland annat av Claes Olson, Åhléns, Telia, Västra Götalandsregionen, Chalmers, Stockholms läns landsting mm. Omkring 120 personer i Sverige och Finland arbetar med att utveckla, anpassa och ge support på systemet. Raindance verkar i både den offentliga och privata sektorn, men har på sistone vinklat in sig till den offentliga sektorn.

Mikael Skogström berättar att för anställda på Raindance innebär det mycket förvaltning, de får önskemål om nya produkter eller idéer till nuvarande funktioner som de tycker borde skrivas om. Raindance har en användarförening som de bollar idéer med, men även systemförvaltare och kontaktpersoner hos kunder som de pratar med. Raindance frågar dessa personer hur de skulle vilja att de olika funktionerna fungerade, vilka funktioner de saknar samt vilka möjligheter som behöver tillföras till systemet. Detta skrivs sedan ihop till en kravspecifikation som kunden tittar på. Raindance föreslår också en lösning i kravspecifikationen, den lösningen är då dels hur användare ska uppleva den, dels hur programmeraren skall utföra den. Den passerar sen ett antal instanser; iblandade kunder, användarföreningen, och Raindance kundservice. De vet ofta vilka problem kunderna har och vilka följdeffekter detta kan få.

Arbetssätt
Raindance använder mest ord för att utrycka funktionaliteten och beskriva problemen i systemet. Med skärmbilden försöker de visa antingen ett gränssnitt eller så ritar de fejkade skrämbilder. De använder inte UML för detta, utan enbart text. Mikael Skogström skulle kunna tro att det beror på att de flesta som arbetar med utvecklingen har arbetat på sitt sätt väldigt länge, innan UML existerade. Det finns nyare personal som har arbetat med UML och som tyckt att det är en dålig återkoppling mellan vad man har ritat och vad som faktiskt sen blir kod. När man sedan har gjort kod av modellen har det varit svårt att få koden att uppdateras.

Raindance använder sig av användningsfall som är så kallade scenarios och det är olika för olika funktioner. Handlar det om att skriva om en befintlig funktion då har kunden en hel del möjligheter att påverka. En del kunder fungerar annorlunda än andra: När kunden märker att man lyssnar på dem vill de plötslig ha mycket mer att säga till om och då gäller det att lägga det på en normal ambitionsnivå för annars läggs för mycket tid på en viss funktion eller en viss kunds önskemål.

Att lägga ner mer tid på att dokumentera och göra mer utförliga användarfall, mer enligt UML eller andra klassbeskrivningar där man tar fram klasser och databaser, säger Mikael Skogström att det finns vissa tekniska förutsättningar som begränsar dem. Han säger att

överhuvud taget är hans kollegor väldigt medvetna om var det brister och vad man skulle kunna förbättra. Ibland finns det inte gehör för vissa förändringar som skulle kunna behövas.

Problem
Ledningen tycker att idéerna om ett analysarbete är bra, men att de skulle stjäla tid, på så sätt att det skulle vara förändringar som inte syns utåt för kund. Men en förutsättning för att kunna arbeta med dessa metoder är att de måste fungera på deras egen plattform. När produkten startade fanns inte Java och man ville ha något om gick att köra på flera operativsystem. Man utvecklade en egen pattform och applikationerna skrevs i ett eget språk som liknar Pascal eller den typen av språk. Applikationerna innebär väldigt mycket kod som tar lång tid att skriva om. Detta har lett till att man har ett system som t ex inte kan exportera sin systemmodell. Raindance har även stöd för externa databaser och sql-databaser. Systemet levereras idag med en intern databas eftersom det var den bästa lösningen när de en gång startade, vilket betyder att den kan vara knepigt att byta ut. Databasen är väldigt snabb och effektiv men har lite andra förutsättningar än en sql-databas. Det är svårt att underhålla datamodellen som de har i diagram-modell. De löser underhållningen med hjälp av manuella krafter, de har en grafisk beskrivning av datamodellen som ligger i ett verktyg.

Problemet med att arbeta med ett system som är gammalt och där vissa av delarna inte är objektorienterade löser de genom ett underhållsavtal. Raindance hjälper kunderna med deras problem och ser till att kunden får ett antal uppdateringar varje år. Då förväntar sig kunden att Raindance betalar för ny uppdatering och även för nya funktioner. Han säger att detta är en avvägning, vad är *uppdatering* och vad är nya *funktioner*.

Man kan säga det att det finns en konsensus bland många medarbetare, och då menar han både gamla och unga, att viktiga delar av systemet borde skrivas om till objektorienterat. De ser fördelar med att kapsla in problemen, men för att kunna göra det måste man möjligen börja i en annan ände, med att dra ut modellen över hur systemet ser ut idag. Detta har de koll på men han säger att det är lite krångligt. De som arbetar med bas-applikationen i själva operativsystemet håller på med ett jättearbete att skriva om allt till en ny teknik. Denna nya teknik skall göra det möjligt att köra gamla applikationer på plattformen, men likaså att göra nya typer av applikationer som är mer nyanserade. Det är även meningen att de ska kunna kapsla in externa programvaror i deras egen programvara.

Jämförelsen med hur mycket efterarbete det är i förhållande till förarbete säger Mikael Skogström är att analysen av en funktion inte är jämförbar med en installation av ett helt system. Det han påstår gällande analysen är att om man gör den bra behöver man inte göra om lika mycket.

Användargränssnittgruppen
Han berättar om ett exempel där han tillsammans med användargränssnittsgruppen i våras började arbeta fram ett nytt gränssitt för Raindance. Anledningen till det nya gränssnittet var för att skapa en mer användarvänlig miljö med en mer lättnavigerad meny. Raindance gjorde tester med slutanvändare, kunder som sitter på Gislaveds kommun och på Skövde sjukhus. Man ville se om användarna kunde hitta i det nya gränssnittet, eftersom man även hade bytt beteckningar. Detta arbete gjordes inte bara av Raindance och användargränssnittsgruppen, utan även med hjälp av WM-data Sveriges User Focus grupp, för att få ett bollplank att arbeta mot.

Mikael Skogström säger att man inte bryr sig om vad kundens IT-chef tycker utan man vill se vad slutanvändaren tycker. Han säger att testresultaten inte överraskade dem utan man hade tänkt på det mesta tidigare. Däremot fick man stöd för vad som kunde vara problem. Han tror att det är ovanligt att man pratar med slutanvändaren utan att man istället pratar med företagets representant, för dem var det viktigt att undvika det. Resultatet blev trovärdiga svar med ett bättre statistiskt underlag. Detta betyder mycket när det i efterhand dyker upp åsikter då kan motivering ges till varför man har löst det på ett visst sätt.

Systemmodellering
Klassisk systemmodellering handlar om att ta reda på vilka objekt som finns och dessa binds ihop för att se vilka relationer de har och hur information skall flöda samt hur detta skall interagera. Med användargränssitt är det svårare säger Mikael Skogström, för man kan ha ett gränssnitt som innehåller andra gränssitt som kommer från annat håll. Det finns försök att beskriva detta i UML, t ex en variant som handlar om interaktion, men den känns inte riktigt fullvärdig än anser han. Det slutar nog med att Raindance kommer att använda flash från Macromedia för att det går snabbt att bygga ihop något som går att klicka sig igenom. Prototyper är väldigt bra för att hitta problem. Har man gjort prototypen vill man bakom den ha en systemmodell för att få fram ett diagram som visar hur allt flödar. Ett enda verktyg har han stött på och det är ett från Israel, som var alldeles för dyrt. Det han saknar är ett bra sätt beskriva sådana flöden och modeller. Angående vanlig systemutveckling tycker han att UML fungerar mycket bra, han tror inte att man ska bli för detaljerad i UML, för då får man problem senare i uppdateringen.

Arbetsgången fortlöper på så sätt att Raindance har säljare som först skapar relation med blivande kunder. När sedan dessa intressenter har blivit kunder går kommunikationen över till kundtjänst rörande problem och önskemål. Är det en specifik funktion kunden efterfrågar är det något kunden tar upp med konsulten när denne är på plats vid t ex installation eller underhåll av olika slag. Detta kan leda till konsultuppdrag eller att det återkommer till utvecklingsgruppen och blir en funktionsförbättring. När kunden har ett önskemål om en funktion, hamnar den i en lista, Man vet t ex att den tillhör en speciell del av produkten. Den delen av produkten har en projektledare och en styrgrupp, man har sedan en diskussion mellan styrgruppen, projektledaren och användarföreningen. Sedan synkas detta inom styrgruppen och det bestäms vilka funktioner man i år ska satsa på.

Kvalitetsarbete måste på lång sikt vara förebyggande, man ser till att undvika problem och det är därför han tycker att tester och analyser är viktiga. Att rita figurer istället för att beskriva i text är ett tydligt och konkret sätt att visa skisser på vad systemet skall göra. Det är viktigt att hitta problemen innan kunden gör det, eller åtminstone samtidigt. Han påpekar att ingen skulle välja att gå till en läkare med gamla kunskaper, för man kräver att läkaren hela tiden skall veta det senaste. I kunskapsyrken är det viktigt att man hela tiden lär sig det nya och vidareutbildar sig.

Vid en installation av ett system får kunden utbildning som ingår i offerten. Konsulterna håller i utbildningen och i installationen. Ett ekonomisystem är inte bara att installera, det ska konfigureras också, det finns ingen färdig lösning utan det är olika hos olika organisationer.

Det finns ett par olika problem som kan uppstå under utvecklingsfasen. Ett problem är att man oftast ska bygga ihop den nya koden med en befintlig kod och att man inte förstår vad den koden gör. Han säger att det finns alltid olika traditioner mellan systemutvecklare. Det finns bra skrivregler som man kan följa, ”codecomplete” från Microsoft t ex. Mycket av problemen

skulle lösas om man hade mer disciplin och följde regler. Det har allmänt och traditionellt funnits ett motstånd hos utvecklare och programmerare att var disciplinerade, för de vill vara fria konstnärssjälar. Detta gäller de flesta organisationer Mikael Skogström har stött på, men självklart är det inte så att folk är rabiata motståndare. Mycket av detta skulle kunna lösas med intern-utbildningar, men det finns sällan tid till det, anser Mikael Skogström.

Raindance håller på att installera ett nytt ärendehanteringssystem. Det innehåller utvecklingspunkter, eventuella fel och allt som har med utvecklings- och felaktighetsarbete att göra. Det kommer att finnas tillgängligt för alla som har en webbläsare, där kan alla inblandade registrera sig och titta under och efter utvecklingsarbetet. Han tror starkt på den lösningen. Han menar att många utvecklingsföretag är dåliga på att dokumentera idéer, felaktigheter och ärenden.

Privat har Mikael Skogström använt UML och har genom det märkt att koden faller rätt snyggt på plats av sig självt, han anser att det blir färre problem i testningen och de som uppstår har varit lätta att rätta till, t ex saker i koden man inte har sett förut. Som snickare måste man ha en ritning, om man börjar snickra på ett hus utan ritning är man tvungen att ta problemen när de dyker upp och det kan bli mycket dyrt.

4.2 WM-data Sverige AB

Intervju med Carol Wittgren, projektledare på WM-data Sverige AB inom affärsintegration dvs enterprise information portals, intranät, extranät och webbsidor med kopplingar mot diverse affärssystem. Utöver projektledning arbetar Carol Wittgren med analyser, förstudier, framtagning av kravspecifikationer för kunds räkning.

Intervju med Susanne Källefjärd, systemutvecklare främst inom dotNet, och har den senaste tiden arbetat med intranät- och internetlösningar i dotNet.

Om företaget
Den övergripande affärsidén för WM-data är att genom ett brett utbud av design- och IT-relaterade tjänster skapa ökad effektivitet och konkret nytta för valda kundsegment. Affärsidén uttrycker deras fokus på värdet av vad de producerar. Fungerande och värdeskapande lösningar kräver en effektiv samverkan mellan människor, applikationer och teknik. Det anser WM-data åstadkomma genom ett komplett utbud av tjänster, vilka de för närvarande levererar från verksamheter indelade i tre områden: bransch-, specialist- och infrastrukturverksamhet Den primära målgruppen är större företag och organisationer i respektive nordiskt land. WM-data har få färdiga produkter utan man har satsat på att förvalta färdiga system hos olika kunder eller systemutveckling, men med mycket lite produktutveckling.

Analysfas och Dokumentation
Carol Wittgren berättar att efter en beställning av ett system eller projekt inleds det oftast med en utredningsfas eller analysfas där man specificerar vad som skall göras och implementeras. Analysen är en mycket viktig del för att få systemet lyckat hos kunden, men det är inte alltid kunden förstår det. Detta leder till att ofta vill inte kunden betala för analysfasen och då får man antingen paketera in det som en del i själva lösningen eller helt enkelt kalla det för något annat. Nästföljande steg är systemutveckling eller anpassning om det är någon annan produkt man ska arbeta med från t ex Microsoft. Därefter är det installation hos kund, testning och verifiering och efter det lansering. Dokumentationen överlämnar man alltid till kund i form av en rapport.

En anledning till att man gör en dokumentation är för att underlätta att gå tillbaka till ett gammalt projekt vilket det görs många gånger. Det är lätt att gå tillbaka och se vad man gjort tidigare och fortsätta på det om man har gjort en tydlig dokumentation. Det lättaste är om man går tillbaka och tittar på det man själv tidigare har gjort, säger Susanne Källefjärd. Det underlättar också för vidarearbete om t ex någon annan tar över, då är det viktigt att dokumentationen är bra. Det är ganska vanligt att dokumentationen dessvärre brister säger Carol Wittgren.

Under utredningsfasen är kunden väldigt tätt inblandad och tillsammans med dem tar man fram vilka riktlinjer som skall väljas. I vissa fall används UML säger Carol Wittgren men i många fall används det inte. Men det är ett mycket bra sätt att beskriva hur ett ”use-case” ska se ut för att sedan återanvända det vid testfasen för att se om det stämmer. I en del projekt går man tillbaka och tittar, men inte i alla.

I utredningsfasen sker kontakt med kunden via deras projektledare eller slutanvändare, ibland med båda. Oftast finns det en kontaktperson, men för att få användarkraven att fungera måste man prata med användarna och då använder vi oss av workshops, arbetsmöten eller intervjuer, säger Carol Wittgren. När ett lösningsförslag lagts fram efter utredningsfasen, tar projektledare och programmerare tillsammans fram hur man kan lösa problemet på bästa sätt. Här har man redan ändrat det som inte är möjligt, projektledaren är då redan medveten om att vissa planer inte går att genomföra. Programmeraren ser till att projektledaren inte lovar guld och gröna skogar säger Susanne Källefjärd.

Beroende på vilket uppdrag man arbetar med och hur stort projektet är varierar analysfasen, det är alltid ett stöd att ha en tydlig del i början med diagram och liknande. Har man ett mindre projekt på cirka femhundra timmar finns inte tid till en stor analysdel. Men om projektet är på ungefär femtusen timmar är förutsättningarna annorlunda.

Angående förvaltningsfasen och analysfasen i ett projekt innebär det naturligtvis att desto mer man specificerar och förbereder innan ett system utvecklas, desto mindre fel blir det när man ska genomföra arbetet. Det har varit en del projekt som timmar har försvunnit för att förarbetet har varit dåligt. Efterarbetet eller förvaltningen, blir kanske inte större om man minskar analysdelen, men systemutvecklingsfasen är den största delen, programmeringen tar mest tid. Använder man sig av RUP och dess sätt blir analysfasen större menar Carol Wittgren.

Arbetsroller och missförstånd

Carol Wittgren säger att det ofta blir samma arbetsroller, men är det helt nya uppgifter man ska arbeta med diskuteras det emellan de olika projektdeltagarna. Finns det personal som anser att någon viss del verkar speciellt intressant arbetar den personen med det. Detta fungerar bra för gruppen Carol Wittgren arbetar med, men man vet att det absolut inte fungerar så i alla projekt. Kommunikationen fungerar på det sätt att man ofta har möten, men samtidigt sitter gruppen placerade nära varandra vilket gör att det är lätt att hålla en bra och fungerande kommunikation. Men under större projekt, när man är placerade på olika ställen sker oftast kontakten via e-post, telefon eller arbetsmöten.

Carol Wittgren säger att de vanligaste missförstånden under systemutveckling är att man pratar olika språk och att kommunikationen brister. Ibland tas prototyper fram på något exempel, men i övrigt används inte bilder som man t ex gör i UML. För att de skulle

underlätta att använda bilder när man beskriver systemet krävs det att man har insikt i UML, de olika objekten, vad det är osv. Det ställer en del krav på kunden och att de ska har lite förståelse för det. Det är svårt för användarna att se hur ett system ska fungera innan de själva har sett det. När kunden väl får se ett system tittar de bara på bilder, textstorlek färger osv. inte själva funktionaliteten för systemet.

Det är inte alltid WM-data som gör den grafiska designen, men diskussionen kommer alltid upp. Användaren har en subjektiv uppfattning om hur det skall se ut, att det fungerar är något helt annat. Allt är relaterat till vad kunden vill betala, de vill inte se ett stort analysprojekt, med diagram och bilder över det blivande systemet berättar Carol Wittgren. I den bästa av världar skulle man vilja arbeta efter metodens alla regler, men det finns ingen tid och kunden vill inte betala för det.

4.3 COGZ

Intervju med användbarhetskonsulterna Camilla Johansson och Andréas Bengtsson.

Om företaget
COGZ är specialiserade på interaktionen mellan människa och dator. De arbetar med att säkerställa användbarhet hos IT-system och webbapplikationer i samarbete med produktutvecklare, samt genom konsultation och utbildningar. Deras mål är att informationsteknik ska vara enkelt, effektivt och tillfredsställande att använda. Först då uppnås syftet med en produkt. Deras företagsidé är att i samverkan med projektutvecklingsteamet bidra till att utveckla en produkt som i slutänden blir både billigare och mer omtyckt Deras arbetar med användbarhet på bred front, alltid ur användarens perspektiv. De anser att deras metoder är kvalitativa och kostnadseffektiva.

COGZ arbetar mest med webbgränssnitt samt en del PA-gränssnitt. De säger att på webben är det enkelt att peka på vad som är fel, det är en enkel värld, men i affärssystem är det svårare, det krävs mycket mer tid på analysdelen. För COGZ är det väldigt tacksamt att arbeta med webben, det kostar inte lika mycket att göra en analys över ett webbgränssnitt. Arbetet innebär att medla mellan tekniker och användare, och i den bästa av världar skulle en systemvetare vara med för att ha hand om strukturering så att de kan satsa mer på gränssnitt och vad som händer på skärmen.

Arbetssätt
Hur man arbetar på COGZ är mycket olika, man producerar inget, de arbetar antingen med färdiga system eller med en projektgrupp tillsammans med en teknisk leverantör. Andréas Bengtsson säger att man skulle kunna arbeta tillsammans med t ex WM-data eller Republic och ge dem den kunskap som de saknar. Det varierar också i vilket tidpunkt av projektet man träder in, och oftast blir det för sent av olika anledningar. Något kan ha gått snett i ett projekt, en kund som t ex har problem med att få kontroll över sin informationsstruktur eller en kund med en komplex webbplats på tusen sidor. Kunden kanske behöver hjälp med att se om de är på rätt väg och då kan de komma in och bedöma just informationen och strukturupplägg.

COGZ försöker att lära upp leverantörer att analysen är viktig, som med Republic där man har ett samarbete som fungerar. Andréas Bengtsson menar att desto mer man arbetar med en teknisk leverantör desto mer inser de hur viktig deras kunskap är. Andréas Bengtsson säger att man har upplevt det väldigt svårt att komma in sent i ett projekt och samarbeta med en teknisk leverantör som redan har fastlagda rutiner. Leverantören arbetar efter sina egna modeller. När problem och prototyp redan är definierade är det svårt att komma in med sin analys om

kunden redan har sin egen utredning som de följer. COGZ vet att en av deras leverantör känner att de har nytta av deras kompetens, men det är svårt att hitta ett sätt att samarbeta. Leverantören har nämligen svårt att redogöra för slutkunden att ett samarbete med en extern anskaffare kommer att användas, för att lägga mer tid på användbarheten. Det finns inte alltid utrymme för det i budgeten. Slutkunden kan också tycka att leverantören redan skall ha den kunskapen.

COGZ tittar på kundnyttan när de planerar sina projekt, som t ex med ett resebolag. De kanske har en enkel funktion som de lägger 10 000 kr på, och för det får de en analys av deras lösning och som gör att de tjänar 10 miljoner per år. I ett sådant fall där ett företag direkt tjänar pengar på uppdateringar och analyser kan de lägga mycket tid och kraft på en undersökning. Men sen finns det företag med en intranätlösningar som de inte alls lägger ner pengar på, för de anser att deras anställda redan kan redan systemet. Där är det svårare att visa kundnyttan. Mycket beror på leverantörens slutkund, vilka krav som finns och vilka kunskaper som finns runt analysering samt vilken kompetens de själva har. Det anstår på hur de själva funderar runt att arbeta användarcentrerat, sen finns det projekt där det verkligen finns ett krav, t ex att systemet måste bli användarvänligt och enkelt att förstå. COGZ har kommit i kontakt med ett företag som vill arbeta på deras sätt med informationsdesign och användbarhetsanalys i alla projekt från början. Detta känns verkligen roligt, medvetenheten har ökat hos företag, säger Andréas Bengtsson.

COGZ är med i analysfasen, men tittar bara på användarbehoven och där kan de ibland brista säger Andréas Bengtsson, för man har inte det holistiska tänkandet när det gäller affärsnyttan, utan man är egentligen bara medlare mellan tekniker och användare. Oftast har man beställt en databas t ex för en viss funktion och sen får de arbeta med att strukturera gränssnittet för att få det hela mer greppbart och inte tekniken bakom, men man skulle gärna göra det. Oftast har analysen redan inletts och kunden inser att ganska snart att detta kommer att bli en komplex databas. Hur ska de få ihop detta på ett snyggt sätt? Resultatet kan då bli en djungel av information som påstås vara lättförståligt.

COGZ arbetar med något som man kallar ”personas”, detta innebär att ta reda på vilka personer som ska använda systemet. Intervjuer eller enkäter görs för att ringa in olika användargrupper beroende på t ex yrkesgrupp, intressen, fritid osv. Efter det får man ut grupper av människor som har gemensamma egenskaper och då skapar man en fejkad identitet, en så kallad persona. Denna person får ett namn t ex Kalle, han är 40 år och bor i villa, han får representera en av användarna. Detta kan tyckas abstrakt, säger Camilla Johansson men det roliga är att alla i en projektgrupp kan använda sig av detta. Man får in rätt tankegångar, t ex att just den funktionen man arbetar med nu kommer inte att fungera för Kalle. Från detta som kan tyckas vara abstrakt blir det till slut konkret. En del av analysarbetet innebär att arbeta på detta sätt, modellerna blir mycket tydliga.

Andréas Bengtsson, säger att om COGZ får chansen att arbeta som de vill, arbetar man utifrån vad man kallar användarcentrerad systemutveckling. Andréas Bengtsson förklarar att det innebär i princip att systemutvecklare inte sitter ensamma på en kammare och tänker, utan de är ute och gör intervjuer. Det finns en del fördelar med att kontrollera tankar rakt av. När systemutvecklare sitter med en grupp användare och förklarar för dem via ritningar och diagram hur informationen flödar. Andréas Bengtsson menar att visar man ett sådant här flöde med diagram om hur informationen går, kan det missuppfattas väldigt lätt av en person som aldrig har sett detta förut och då använder de oftast mycket enklare metoder. Att arbeta

användarcentrerat, om det ska ske efter boken, ska man göra det från koncept till vilka uppgifter som skall göras och därifrån gå till interaktion, knappar och dialoger.

Metoder
COGZ använder sig av stora A3 papper, spritpennor och post-it-lappar för att visa bilder och planskisser över en organisation. När det gäller webbplatser bygger man upp webbens utseende rent konceptuellt, man visar att det finns en sökfunktion men vart den ligger någonstans spelar ingen roll. Användaren kan sitta bredvid och säga t ex ”den här menyn fungerar inte det måste fixas.” På så sätt kan de få in mycket feedback genom att man diskuterar fram lösningar.

COGZ berättar att de har en erfarenhet från ett företag som arbetade på ett traditionellt reklambolags vis. Där lyssnar man på kundens önskemål och tar fram tre designförslag. Kunden väljer då kanske förslag B, då har man lagt ner tid och pengar på förslag A och C. Egentligen vet man i förväg att massor av timmar kommer att läggas på olika förslag i onödan. Deras tanke är att om du visste vad kunden ville ha från början hade man kunnat strunta i förslag A och C, och tjänat in den tiden för att lägga på kvalité istället. Andréas Bengtsson säger att man vid ett försök att erbjöd workshoppar istället, där de samlades alla som skulle använda sig av systemet tillsammans med teknikerna och cheferna som skulle betala. Man samlade ibland upp till tjugofem personer under två dagar, vilket kan tyckas dyrt, men kunden är helt övertygade efteråt att det här *är* sättet att arbeta på. Ett bra bevis på detta är att dessa kunder vägrar nu att gå in i projekt utan att göra workshops innan.

Andréas Bengtsson har också märkt att om en person sitter med i en workshop och frambringar åsikter, får man aldrig höra två månader senare när systemet är klart att det inte är vad som beställts. Skulle det hända att personen satt och sov under workshopen inser denne att han gick miste om sin chans. COGZ får aldrig klagomål om man arbetar på detta sätt, men man vet att det finns leverantörer vars kunder säger att om man inte arbetar på detta sätt har slutresultatet inte blivit vad man ville ha och man behöver bygga om produkten för hundratusentals kronor. Det viktigaste är att man måste fastställa rutiner som man arbetar efter och att man från början tar reda på vad som behövs och hur man egentligen gör säger Camilla Johansson.

Arbetsroller
Andréas Bengtsson har främst hand om COGZ workshops, då han har mer erfarenhet av det säger Camilla Johansson. Deras kunskap blir aldrig gammal det enda man behöver egentligen är papper, penna och hjärnan, allt kommer tillbaka till hur hjärnan fungerar. Man har knappt några digitala verktyg när man arbetar, möjligtvis arbetas det med webbformulär när undersökningar på distans görs. Det skulle kunna vara ett alldeles utmärkt arbete för COGZ om en kund har ett dataprogram, men kan inte sälja det, kanske för att programmet är svårförklarat. De är inte marknadsförare eller reklammänniskor, men frågan är vad kunden försöker säga och vad han vill kommunicera. Det är i detta stadium man försöker hjälpa till. Slutkunden kanske inte förstår en PowerPoint-presentation för det är för komplext, då kan de se över den, titta på hur man kan strukturera informationen annorlunda. Lyfta fram vissa saker och börja på en övergripande nivå, detta har egentligen inget med tekniken att göra.

Problem
De vanligaste kommunikationsproblemen COGZ säger sig stöta på är framför allt okunskap om vad man gör. De gör sitt bästa för att förstå problem och missförstånd, man säger sig ha ett öga för det. Ibland kan det bli lite konflikter när man träder in i ett projekt, i och med deras

kritiska sysätt. Deras arbete innebär att kritiska till en lösning, man fungerar som de personerna som måste säga att det är fel. Detta kan leda till olika konflikter.

Många säger att det COGZ sysslar med är vanligt sunt förnuft, men det räcker med fem minuter framför ett gränssnitt sen har man pekat ut ett antal fel, säger Andréas Bengtsson. Många av felen kanske är triviala men det räcker att peka ut ett stort grundfel och kunden säger ”ja just det, det tänkte vi inte på”. Man måste alltid komma med konstruktiv kritik och med förslag. I slutändan är det ändå alltid kunden som bestämmer, och ibland vet man inte själv vad som är den bästa lösningen. Ibland händer det att man som konsulter kommer in i ett projekt, där kunden säger sig veta om sina problem och påstår att de genast ska ta tag i dem, men ändå händer inget. Det är frustrerande menar han.

COGZ skulle önska ett närmare arbete med slutkunden, idag verkar man nära leverantören. De vill att högre krav ställs för att många företag idag inte levererar användbarhet. Det finns många dåliga produkter som säljs, flera av dem skulle man inte behöva betala pengar för. Det finns redovisningssystem som är helt oförlikneligt dåliga och på detta läggs mycket pengar, vilket ideligen beror på okunskap och trivialisering av problemen.

Man kan t ex göra en effektanalys för att se om det verkligen ger den effekt man önskar. Man tittar på orsak och värde, och utgår ifrån effekten som man vill uppnå och vad som krävs för det. Det kan visa sig att vissa utav funktionerna som har framkommit ur en analys inte uppnår något mål, och då kan man avlägsna den funktionen.

COGZ säger att man tror många teknikföretag samt kunder tycker att det kan bli dyrare att arbeta på det vis de förespråkar, men man har också hört kunder som har sagt. ”Ja, vi kör, men vi måste ha med en kognitionsvetare först!”

4.4 Republic Factory AB

Intervju med delägarna Mikael Alveberg, Niklas Rhöse och Anders Wideskott på Republic Factorys kontor.

Om företaget
Republic Factory AB har tre delägare som intervjuas. Republic Factory arbetar tillsammans med Republic Consulting AB, som har två anställda. I Republic Consulting AB ligger fokus på att utveckla den verbala och digitala kommunikationen mellan företag och dess kunder på det sättet att företaget blir mer lönsamt. Det innebär att man kan göra allt från att analysera hur affärsmässiga processer ser ut, både mänskligt och tekniskt, till att höja säljkompetensen hos medarbetare eller formulera slagkraftiga texter i den dagliga kommunikationen. Republic Consluting genomför allt från workshops för ledningen till ren kommunikationsutbildning för deras medarbetare. Republic Factory AB bygger sedan de digitala lösningar som kanske behövs för att stödja företags interna eller externa processer.

Republic säger sig vara passionerat engagerade i att hjälpa deras klienter att bli mer lönsamma i relationen till sina kunder, genom att visa hur kundens affärsprocesser kan förenklas genom ny teknik förpackad i god design. Republic deltar aktivt genom att hjälpa medarbetarna att ta till sig nya arbetssätt. Det skall leda till förenkling av vardagen och skapa ökad effektivitet och lönsamhet. Ibland avråder man också från satsningar som man vet inte leder till ett visst mål.

Arbetssätt och dokumentation
Hur Republic arbetar i projekt beror ofta på i vilken del av projektet man infinner sig. Ibland har en reklambyrå startat och gjort en grund och Republic ska producera det och ibland kommer projektet direkt till dem från start. Vissa kunder har en struktur klar och en detaljerad kravspecifikation när de kommer och vet vad de vill ha och vad de inte behöver. Andra kunder har en ide, t ex att de vill ha ett intranät för att skicka lite meddelanden till varandra. Det finns många olika sätt att ta sig an ett projekt. Republic ställer naturligtvis frågor, om det kommer en kund med en luddigt formulerad kravspecifikation. Oftast intervjuas också kunden för att exakt kunna ta reda på vad de är ute efter. Där handlar det enligt Republic mycket om erfarenhet och rutin för att få ut rätt information.

Dokumentationen som görs inför ett projekt är mötesprotokoll, anteckningar från produktionsmöten med kunden, men dessa är inga formella dokument. Det finns färdiga projektmallar med vilka delar som skall vara med, men något sånt använder sig Republic inte av. Ofta stämmer man av via e-post.

Republic kastar sig inte in i systemutvecklingen utan har ändå ett analysiskt moment. I några projekt har man samarbetat med COGZ, de har ingått som en person i projektgruppen. Sen har de själva fört diskussionen, men COGZ har på något sätt satt det på papper. Camilla Johansson på COGZ har Republic använt på det sättet att hon har arbetat fram användarvänlighet och gränssnitt.

Hur detaljerade kravspecifikationer eller offerter är beror på säger Republic. Databasen har man aldrig tagit med i dokumentationen. Oftast kommer man överens om vissa saker på möten och via e-post. Man har sett vissa offerter som väldigt exakta och noggrant specificerade, men deras kravspecifikation anser man oftast inte behöver vara så detaljerad. När Republic arbetade mot ett engelskt företag var det ”sign offer” varannan dag, väldigt byråkratiskt, man fick verkligen inte stiga ifrån den ramen. När man arbetade med länsstyrelsen var det även där mycket offertarbete och alla upphandlingar med extrema mängder text som skulle tas fram.

Vid t ex ett uppdrag som handlar om en hemsida specificerar Republic alla funktioner som skall finnas med, annars är det svårt att sätta pris. På så vis bryter man ner systemet/problemet i bitar för att komma fram till priset. Ibland kan det hända att någon del liknar något arbeta man gjort tidigare då har man redan kännedom om hur lång tid det tar att lösa problemet. Republic säljer inte paket där man inte är säkra på vad det innehåller, man vill inte sälja grisen i säcken. Vid framtagningen av funktioner har ofta kunden redan någonting som de inte är nöjda med, då arbetar Republic fram varför de inte är nöjda

Republic har också arbetat med WM-data Raindance, där man var med att arbeta fram gränssnittet för Raindance nya utseende. Arbetet skedde mot Raindance:s användargrupp och samtliga avstämningar var med dem.

I frågan om tiden med analysfasen blir svaret att det beror det lite på vad uppgiften handlar om. Det känns inte som att det är raketforskning vi håller på med menar Republic. Man specificerar ner vilka delar som kommer att ingå i systemet och sätter pris på detta. Skulle man då sätta en stor del på analysdelen kommer priset att hamna på enorma summor. Naturligtvis, tänker man efter innan på hur systemet skall se ut desto smidigare går det. Men oftast är det så att kunderna ändrar sig under tiden.

Arbetsgång och metoder
När kunder kontaktar Republic har de ofta redan innan internt diskuterat frågan i kanske ett halvår. De kan t ex säga att om två veckor måste vi ha detta löst för då har vi en konferens, mässa eller liknande. Republic menar att man i den situationen inte kan säga; först måste vi analyser detta i några veckor. Republic menar att det är svårt, för det känns som att hela analysmodellen är jättebra på pappret men kunderna börjar tänka på ett nytt sätt när de får ett nytt verktyg. De testar och kommer på att vissa lösningar är smarta, detta föder nya idéer och tankar. Det bästa, ur företagssynpunkt, är ju att ge kunden några smarta lösningar till en början och inte ge dem hela paketet. Oftast är det bra att börja på det viset för då märker kunden själv att den lösningen är bra och vill ha mer.

Oftast låter Republic kunden följa arbetsgången på projektet genom att utveckla projektet skarpt. Det vill säga arbeta online under hela utvecklingen. Detta gör att kunden hela tiden kan kommentera och känna sig delaktiga i projektet. Problemet med att låta kunden se arbetsgången är att de kan kommentera funktioner som inte är klara. Oftast är det ett otacksamt arbetssätt eftersom det under ytan inte syns, utan kunden ser mest om bakgrundsfärgen är bra. Webbutveckling är ändå ett relativt snabbt utvecklingsätt i förhållande till mjukvaruutveckling. Om man jämför med stora IT-konsulter, som från början till slut utför projekt i storlek tre till fyra år, så har vi mindre och kortare projekt.

Att företag och näringslivet är öppna för IT-system och att en vändning är på väg menar Republic att man sagt länge nu, men att det ändå känns lovande. Det var ett tag sedan det fanns många skeptiker där man inte riktigt visste vad som önskades. Republic tror att IT är på gång, men man vet inte riktigt hur det kommer att arta sig för de riktigt stora IT-projekten, om de kanske fortfarande väntar på större satsningar.

På frågan om vilka misstag som kan uppstå säger Republic att det generellt sätt är att man arbetar på olika sätt med olika kunder. Vissa är lite flyktigare och där man inte har någon kravspecifikation och det funkar alldeles utmärkt. Problemet med det är att det är svårt att veta vad som står i offerten, vilket leder till att projekt har dragit ut på tiden, i och med det har kunden förväntat sig mer än vad Republic har tolkat det som. Det tar på krafterna att ha projekt som bara rinner ut i sanden.

Kunder kan ibland kommentera småsaker såsom att de inte vill ha en blå bakgrund eller liknande men Republic säger att sånt har man alltid stämt av innan. Man försöker att leda dem rätt och tillmötesgå dem, ofta har Republic en sådan punkt med i offerten. Efter avstämning av design, är det den som gäller. Det funkar inte att kunden ändrar sig hela tiden.

Arbetsroller och kommunikation
Kommunikationen på Republic är enkelt eftersom de är så få anställda på ett litet kontor, det är bara att öppna munnen, eller springa runt det lilla bordet för att hjälpa varandra. När man har varit på möte eller om det är något på gång är det inte heller något officiellt, utan man har ett litet planeringsmöte. Ofta blir det att man bara drar igenom informationen för varandra. Det är ganska inofficiellt. Missförstånd kan uppstå om man inte är tillräckligt tydlig. Oftast är köpare okunniga. Det sker missförstånd med både skriftligt och muntligt, en del kunder tycker att en mening betyder något underförstått. Men Republic säger att missförstånd sällan uppstår.

Ibland händer det att en del kunder vill ha allting enligt ”skolans regler”, då anser Republic det är bra att anlita någon som verkligen kan det där, som t ex COGZ. Gränssnittsdesign och liknande känns som om de kan bra, men vissa användaranalyser är inget som Republic arbetar

med och det kan de behöva ha hjälp med. Ofta är det väldigt bra argument gentemot kunder att man arbetar med kognitionsvetare. Ibland vill kunder stryka den biten, ibland inte.

4.5 Utvalda kurser

4.5.1 Handelshögskolan vid Göteborgs universitet

Databaser och systemutveckling, 10p
Kursens syfte är att ge teoretiska kunskaper och praktiska färdigheter i analys, design, och realisering av databasbaserade informationssystem. Denna kurs finns dock inte längre idag utan har ersatts med en 5-poängskurs med objektorienterad programmering där dokumentationen skall vara enligt Mathiasen et al. (2001). Kursens huvudsakliga innehåll är objektorienterad systemutveckling i form av metoder och tekniker för analys och design av informationssystem.

4.5.2 Umeå universitet

Systemutveckling och organisationsförändring 10p
Diskuterar systemutveckling i relation till fenomen som organisationsförändring och digitala affärer, samt modellering i allmänhet och objektorientering i synnerhet.

Systemdesign, 5p
Kursen avser att introducera till centrala metoder, tekniker, och verktyg som används vid systemdesign. Kursmomentet syftar till att skapa en förståelse för allmänna teorier om systemutveckling och användandet av formella modeller liksom kunskaper om projektorganisering och dess relation till både datorsystem och organisationer.

4.5.3 Växjö universitet

Design av informationssystem, 10p
Kursen syftar till att ge grundläggande kunskaper inom området design av informationssystem. Genom introduktion av olika modeller, metoder och tillvägagångssätt skall man visa på sambandet mellan en organisations verksamhet och dess informationssystem.

4.5.4 Högskolan Dalarna

Introduktion till informatik och IT, 10p
Kursen ska ge grundläggande kunskaper om programvara, datorer och datornätverk. Efter kursen ska man kunna förstå och redogöra för olika begrepp inom informatik. Dessutom ska man kunna redogöra för informatikens utveckling som ämne.

Databaser och informationssystem, 10p
Kursen skall ge studenten faktakunskap om databaskonstruktion, och förståelse för hur databastransaktioner fungerar och påverkar det praktiska arbetet vid programmering och administration. Vidare skall studenten förvärva tillämpad kunskap vad avser teknik för datamodellering och programmering.

5 DISKUSSION

5.1 Cross Industry Solutions - Raindance

På Raindance träffade vi Mikael Skogström på en intervju som vi upplevde ärlig och öppen. När vi efteråt gick igenom materialet fann vi ganska snart att Mikaels Skogströms uttalanden var paradoxala på frågorna om användningen av UML, där han i början av intervjun sa att man inte ska vara för detaljerad i UML för att det kan skapa problem senare i systemutvecklingen. För att sedan avsluta intervjun med att säga att han på fritiden använt UML och då upplevt att koden faller snyggt på plats och detta bidrar till färre problem.

När vi pratade om vilka problem som kan uppstå fick vi ett intressant svar att de ofta vet vilka problem kunderna har och vilka följeffekter dessa problem kan skapa. Tanken borde väl ändå vara, anser vi att om man vet om problemen sedan tidigare borde de inte upprepas och på så vis inte skapa några konsekvenser. Självklart pratar inte Mikael Skogström om triviala problem men vi tror att vissa av problemen faktiskt upprepas gång på gång trots full vetskap om dem.

Mikael Skogström säger att många av problemen, speciellt vid implementering av det objektorienterade synsättet kan lösas med internutbildningar inom ämnet. Han är införstådd med att detta är ingenting man lär sig på en eftermiddag utan det krävs utbildning och det tar tid, tid som man inte har.

Raindance sätter uppenbarligen användarna i fokus då de har en användarsnittgränsgrupp inom företaget som enbart arbetar med att få fram den bästa användbarheten. Det visar sig också tydligt då Mikael Skogstöm poängterar att de vänder sig till slutanvändaren vid testning av prototyper istället för att vända sig till chefen. Vi tycker att detta sätt är mycket bra eftersom detta leder till att slutanvändarna verkligen får det system de vill ha.

5.2 WM-data Sverige AB

Det var intressant att få studera två bolag inom samma koncern. Vi träffade Carol Wittgren och Susanne Källefjärd under ett frukostmöte. Det framgick att Susanne har läst ADB-programmet på Informatik i Göteborg. Hon kände till författaren Brown men har inte tidigare varit i kontakt med Mathiasen.

Det framgick ganska tidigt att de var införstådda i tyngden med att arbeta objektorienterat, både analysmässigt och programmeringsmässigt. Ett bevis på detta fann vi när Carol Wittgren förklarade att, ofta vill inte kunden betala för an analysfas, men ibland är det ett måste. Då kan man paketera in analysdelen som en del i helheten eller helt enkelt kalla det något annat. Vi tycker att det är bedrövligt att analysdelen måste gömmas för kunden. Kunden borde ha fullt förtroende för sin leverantör, om då leverantören vill ha med ett analysmoment borde det vara skäl nog. Vad vi har förstått genom vår utbildning beror slutresultat mycket på hur pass mycket tid man lägger ner på analysfasen i ett projekt, minskar man den delen kommer underhåll att öka och det är något man ska försöka undvika eftersom detta kan ge merkostnader och i värsta fall missnöjda kunder. Detta i sig borde vara ett argument som kunden absolut borde nappa på.

Carol Wittgren säger nämligen att det har funnits en del projekt där timmar har försvunnit för att förarbetet har varit dåligt. Efterarbetet eller förvaltningen minskar möjligtvis om analysdelen blir mindre, men systemutvecklingsfasen är den största delen, programmeringen tar mest tid.

Vikten av god dokumentation är betydelsefull för kunden och eventuell vidareutveckling av systemet. Dokumentationen bör skrivas på ett naturligt språk och kan innehålla figurer och tabeller, för att underlätta förståelsen (Mathiasen et al., 2001). Vi tror att det i många fall kan var så att man bara lämnar ifrån sig en manual istället för en fullständig dokumentation. Manualer är självklart viktigt för att kunden ska kunna använda systemet till fullo. WM-data är fullt införstådda att dokumentationen är viktig men att den ibland kan brister kan mycket väl bero på att det är det sista man gör innan avslut av ett projekt. Av egen erfarenhet vet vi också att dokumentationen inte tillhör det roligaste i ett projekt, att människan av naturen är lat och därför skjuter det tråkiga framför sig. Resultatet av detta kan då bli en snabbt hopskriven dokumentation.

Precis som Raindance använder sig WM-data Sverige av workshops och intervjuer för att få användarkraven att fungera. De försöker i de flesta fall att arbeta direkt mot användarna istället för att prata med projektledaren som vanligtvis är en person. Helst arbetar man med både projektledaren samt användarna. Att arbeta med workshops är nästan något som krävs för att kunna få fram samtliga delar för att definiera problemområdet. Frågan kvarstår dock om dessa workshops har som mål att faktiskt kartlägga problemområdet eller om man endast låter användarna diskutera vad systemet skall göra. Det är i grund och botten leverantörens uppgift att vägleda kunden genom analysfasen. Det är inte kunden som skall rita diargrammen eller skriva dokumenten men att kunden ska vara en del av framtagningen. Många gånger har säkert kunden en uppfattning om hur systemet skall vara fast att den kan vara felaktig för att kunden inte vet bättre.

5.3 COGZ

Vår uppsats fick en oväntad vinkling då vi efter ett samtal med Republic kom i kontakt med Camilla Johansson och Andréas Bengtsson på COGZ. Republic använder sig av COGZ i större projekt för att på rätt sätt kunna bygga användbarheten och dokumentera projektets gång.

COGZ har en arbetsmetod som låter användaren hamna i fokus, inte teknikerna. Deras arbete är att sköta kommunikationen mellan teknikerna och användarna och på så vis fokusera på att ingen information går till spillo. Ibland känner de ändå att vissa saker brister där en systemvetare skulle kunna sköta viss kommunikation bättre, då en sådan person lättare kan kommunicera med teknikerna. Eftersom det också är COGZ uppgift att ta fram och finna strukturen kan det så klart vara lättare att arbeta fram strukturen tillsammans med en person som kan strukturera och tänka som en utvecklare.

Det var intressant att träffa ett företag som arbetar på ett annat sätt när det gäller analysarbetet än det vi studerat. Deras arbete grundar sig på människors beteenden och hjärnans funktion, istället för att nästan enbart fokusera på vilka funktioner och objekt som skall finnas med i systemet. Deras arbetssätt är att träffa samtliga roller involverade i projektet under t ex en workshop, där man med hjälp av primitiva verktyg så som penna, papper och post-it lappar definierar problemområdet och så kallade personas. Personas är en fiktiv roll i systemet med egenskaper och beteenden som dokumenteras. T ex ”Kalle är 40 år, singel och boende i villa.

Han har hela sitt liv arbetat med system och datorer.” Detta arbetssätt är som en förlängning på aktörer och aktörstabeller, då man under workshopen och utvecklingsfasen lättare kan anpassa funktioner efter de olika personer man lärt känna.

Efter att otaliga gånger gått igenom intervjun och analyserat deras arbetsmetoder, känns det som om de skulle kunna vara ingångsporten för oss att få företag/kunder att vilja arbeta mer med framtagning av system som till grund och botten löser ett problemområde. COGZ nämner själva att en systemvetares kunskaper allt som oftast saknas. Genom dem skulle vi kunna kombinera deras kunskap om människor beteenden och människans hjärna med de modeller och metoder som vi i denna uppsats förespråkar.

Andréas Bengtsson berättar att han har erfarenhet från projekt där det i förväg har tagits fram ett flertal designförslag och där kunden sedan får välja ett av dessa alternativ. COGZ menar, liksom denna uppsats, att det blir onödigt arbete att ha flera designförslag. Genom att först definiera problemområdet och arbeta fram vilka olika delar som krävs och behövs till systemet skulle man undvika denna typ av dubbelarbete. Det är intressant att kunden utan problem väljer att betala för förslag som de i redan vet att de inte kommer att använda sig av. Kan det verkligen vara så svårt att förklara för dessa kunder att de tjänar på att ta fram det rätta förslaget från början?

COGZ har märkt att det numera finns en medvetenhet hos företag, att systemutvecklings projekt inte alls ser likadant ut som för tio år sedan. Detta tycker vi spår om en ljusnande framtid inom de objektorienterade systemutvecklingsmetoderna.

COGZ berättar om leverantörer som arbetar efter sina egna modeller, vilket gör det svårt för dem att börja implementera deras arbetssätt. Givetvis har företag egna metoder som de arbetat efter i flera år. De företag som över huvud taget inte har någon strategi i sina projekt överlever nog inte länge. Men det är som ordspråket säger ”svårt att lära gamla hundar att sitta”, det vill säga att det kan vara svårt att bryta deras arbetsmönster. Det skulle vara intressant att studera en av dessa metoder för att se om det möjligtvis går att anpassa till de olika modeller vi beskriver i denna uppsats. Och om det i så fall skulle skapa några förbättringar.

Många kunder till olika systemutvecklingsföretag idag kan nog känna sig missförstådda då det inte vet exakt hur de skall kommunicera med företag. Eftersom olika företag kommunicerar på olika vis är det inte konstigt att det ofta blir brister i kommunikationen. Tänk så mycket det skulle underlätta om samtliga utvecklingsprojekt skulle använda sig av de standardmodelleringar som finns samt en kognitionsvetare och en systemvetare.

5.4 Republic Factory

Det var intressant att studera ett mindre företag som överraskande har samarbetat med ett annat företag i studien, WM-data Raindance.

Vi kan förstå att Republic själva inte kan lägga ner en massa tid på att dokumentera och analysera ett problemområde tillsammans med kunden. Många av deras projekt är av mindre skala, i tid räknat, där det många gånger finns funktioner de kan återanvända. Men vissa problem tror vi man skulle kunna undvika genom att faktiskt skriva en något mer detaljerad kravspecifikation. Republic poängterar själva att det ofta är så att kunden ändrar sig under systemutvecklingens gång. Detta är definitivt ett problem man skulle kunna lösa genom att tillsammans med kunden mer noggrant konstatera vilka delar som måste vara med och vilka berörda delar av organisationen som kommer att använda systemet.

Republic har en intressant kunskap att dela med sig av, att kunder i vissa fall kommer på nya funktioner när de väl sätter sig in i deras nya system. Detta tror inte Republic beror på en bristande analysstudie, utan snarare ett nytt tankesätt som träder fram när kunden inser möjligheterna med deras nya system.

Självklart tror inte vi att en ingående analysstudie alltid innehåller *alla* svar, att ibland spelar det ingen roll hur mycket man analyserar, kunden kommer ändå att ändra sig under utvecklingen gång.

5.5 Utvalda kurser

Vi valde att ta med andra universitets kurser i uppsatsen för att visa och få stöd för att många universitet har ett stort utbud av kurser inom ämnet objektorienterad analys och design. Val av universitet var beroende på kursutbud och geografisk placering för att samtidigt studera om litteraturen vi förespråkar i denna uppsats återfinns på andra universitet. Efter att ha läst åtskilliga kursplaner såg vi att kursinnehållt var relativt lika och att den stora skillnaden var kurslitteraturen. Av alla kurser valde vi att ta med de vars innehåll på något sätt skiljde sig åt.

Vi fann Mathiasen et al. (2001) på ett flertal universitet men valde att endast ta med två av dessa kurser, utöver Göteborg Universitet. Detta för att visa att det finns liknande kurser med annan litteratur samt underlätta en eventuellt fortsatt studie inom området för oss eller andra studerande. För vidare läsning, se Bilaga C.

5.6 Hur ser företagen på modellering?

På WM-datas hemsida möts vi bland annat av texten: ”Fungerande och värdeskapande lösningar kräver en effektiv samverkan mellan människor, applikationer och teknik.” Denna text innebär att de på något vis måste arbeta fram exakt vad kundens problem består av genom att kommunicera mellan människa, applikationer och teknik. Om detta är det första som kunden möter blir kunden också medveten om målsättningar som sätts i projekt. Vårt intryck är att det verkligen finns ett moment där analys ingår. Efter att ha intervjuat WM-data inser vi att självklart använder de sig av analytiska moment och att WM-data verkligen förstår tyngden av UML och RUP, men inte alltid kan realisera dem.

Generellt fick vi intryck av att företagen var öppen gentemot modellerna i UML och RUP, men att de tar för mycket tid i förhållande till om man inte skulle arbeta med dessa metoder. Då kan det bli svårt att argumenterar för kunden att detta är något de skall betala för. En mening som förekom i samtliga intervjuer är ”i den bästa av världar…”

WM-data har idag utbildat sin personal inom RUP, och vissa har sedan tidigare utbildning inom UML, men de verkar som om de inte riktigt vet hur de ska realisera det effektivt på befintliga och nya projekt. Kanske vågar man inte satsa eftersom det tar tid att lära sig att arbeta med dessa metoder och istället håller sig till sina egna rutiner. Man måste kanske som företag räkna med att de första projekten inte blir lönsamma för företaget i förhållande till betalda timmar, men i längden tjänar de på det.

Företagen idag har ibland en felaktig uppfattning om hur man arbetar med RUP och UML. Att kunden måste ha förståelse för metoderna, men så är ju inte fallet. Vi ska ju vägleda dem, för att sedan själva rita upp klasserna och objekten enligt standard. Många kunder kan nog

känna sig skrämda av leverantörer då de använder uttryck som modeller och objekt, något som de tycker låter dyrt och komplicerat.

Vi tror att många webbyråer ofta bagatelliserar det analytiska momentet inför ett systemutvecklingsprojekt för att webbaserade system många gånger är av mindre skala. Men idag finns det många stora och omfattande system som är webbaserade, vilket också leder till att även denna typ av system borde analyseras. Utvecklingen har dock gått rasande fort vilket också kan ha lett till att denna typ av utvecklare inte förstår tyngden av detta sätt att arbeta.
Med den stundande vändning inom IT-sektorn tror vi att det är viktigt att hänga på de trender som finns idag gällande UML och RUP. Man kan inte bara kasta sig in i projekt då kunder idag kräver mer i form av hög användarvänlighet.

Analysarbetet tar lång tid, med detta leder inte vill att programmeringsdelen minskar. Detta gör att den slutgiltiga projekttiden blir längre och därmed dyrare. I dagens samhälle kan det kännas stressande när saker och ting tar längre tid än vad det gjort tidigare. Kunden vill ha sitt system fort! Tidsbrist är ett faktum!

Ett vanligt missförstånd hos företag är att de tror att alla delarna av analys och design ska tas med för att det ska vara enligt skolans normer, men detta är inte sant. Universitet förespråkar att man ska ha kunskap inom hela området, men att man också ska lära sig att använda sig av de delarna som är viktiga för en viss del av ett projekt. Många företag vågar kanske inte satsa på denna typ av analyser på grund av detta.

Företag idag tycker att det är svårt sälja in något moment innan systemutvecklingen. Kunden ser ofta att själva produkten är när man börjar utveckla.

Företag är inte helt övertygande om att dessa metoder kan skapa rätt produkt från början. Som Republic påpekade finns det kunder som kommer på funktioner de vill ha till sitt system först när de arbetat ett tag med det. Vi tror dock på att utvecklingen av befintliga system borde underlättas av att arbeta metodiskt.

5.7 Styrkor/Svagheter med Objektorienterad Analys och Design

Efter att ha studerat ett flertal kurser inom ämnet objektorienterad analys och design är vi såklart förespråkare av dessa metoder. Efter all litteratur vi läst och genom den kunskap vi erhållit tycker vi att det är lättare att finna fler styrkor än svagheter. Efter att ha diskuterat vad vi upplever som styrkor och svagheter blev resultatet följande punkter.

Styrkor

- *Moduler:* Möjlighet att skapa delar av systemet som är oberoende av varandra.
- *Återanvändbarhet:* Möjlighet att återanvända bitar av systemet inom samma projekt eller på nya projekt i framtiden.
- *Mindre underhåll:* För att rätt system arbetas fram från början, med de funktioner som skall lösa problemområdet. Genom detta får man mindre fel och därmed lägre kostnader.
- *Färre missförstånd:* För att man tillsammans med användare diskuterar vilket problem systemet skall lösa.
- *Användarvänligt:* Man arbetar fram funktioner som är anpassade efter de arbetsområden systemet skall implementeras i. Målet med systemet är att strukturen skall vara lättöverskådlig och lättnavigerad och fylla den funktion som det är tänkt.
- *Fullständig dokumentation:* Genom att arbeta med alla de delar uppsatsen tar upp får man med alla bitar som en fullständig dokumentation kräver.

Svagheter

- *Tid:* Det tar såklart längre tid att utveckla analysdelen.
- *Merarbete:* Vissa delar kan kännas som merarbete då det ibland är svårt att överväga vilka delar man behöver dokumenteras.
- *Kan bli dyrare:* Framförallt kan det bli dyrare då antal timmar i ett projekt ökar. Varje gång kanske inte en analysdel effektiviserar arbetet då det kan jämföras med ett arbete som utför av människor med utvecklingsrutin.
- *Kräver kunskap:* Att arbeta med dessa metoder kräver någon slags utbildning i området. Man kan inte bara som lekman sätta sig och arbeta sig igenom denna typ av metoder.

6 SLUTSATS

Hur skiljer sig arbetssätt och metoder mellan företag och hur skiljer sig dessa företag med de teorier universitet i Sverige undervisar?

Vi fick inte exakt veta hur företagen in i minsta detalj arbetar i systemutvecklingsprojekt, det är därför svårt att visa exakta skillnader på objekt och klassnivå. Samtliga företag har någon gång arbetat med framtagning av diagram och liknande, men inte i slutändan visat detta för kunden eller haft med det i någon dokumentation. Det som var en stor förvåning är att några företag inte kände till eller arbetade med klassdiagram och händelser, punkter som vi använder oss av i stor utsträckning på universitet.

En punkt som vi som studerar inte behöver ta hänsyn till är att kunden ibland vill ha sitt nya system omgående. Ett vanligt argument idag är att ”system krånglar ändå, så det är lika bra att vi får igång systemet fort, så löser vi buggarna sen”. Inte tänker man så när man köper en bil. Jag köper bilen nu, så får jag i efterhand se om bromsarna fungerar och i så fall fixar felet då.

Det som skiljer kognitionsvetare från de övriga företagen är att de har fasta metoder och rutiner för framtagning av de olika delarna i ett systemutvecklingsprojekt. De personer som arbetar på COGZ är utbildade kognitionsvetare vilket också leder till att deras metoder har ursprung från litteratur inom ämnet. Dock arbetar de inte med RUP och UML som metodstandarder. De övriga tre företagen använder sig av metoder, men de skiljer sig i olika projekt. Dessa metoder är inte tagna ur en bok, utan de är framarbetade i arbetsgruppen eller av företaget.

Metoder och arbetssätt företag emellan upplevde vi inte skiljer sig speciellt mycket. Den största skillnaden är egentligen att storleken på projekt är varierande i omfång och därmed beroende av företagets storlek. Intressant var dock att företagen mer eller mindre har haft något slags samarbete. Det enda samarbete med COGZ har Republic haft, men kanske kommer nya samarbeten att inledas.

Konkret skiljer sig företagen arbetsmässig genom att:

- Raindance har en färdig mjukvara som funnits på marknaden i flera år och redan har en färdigutvecklas plattform. Utveckling av Raindance sker kontinuerligt.
- Republic arbetar uteslutande med webbaserade system, som oftast sträcker sig under kortare perioder
- WM-data Sverige arbetar mer med större systemutvecklingsuppdrag, främst webbaserade samt konsultarbete.
- Cogz arbetar endast med framtagning av användarvänlighet och arbetar inte alls med systemutveckling i programmeringstermer menat.

Om man sätter företagen i förhållande till vad universitet idag undervisar är en av de största skillnaderna att deras förstudier inte ens är hälften så stor som vi har fått lära oss. Att de inte heller tänker ut problemområdet i samma utsträckning och att de inte delar in arbetsmomenten i klasser, objekt, händelser och tillstånd. Några av företagen plockar ut de delar *de* tycker är

viktiga och känner till ur UML och RUP, men dessa metoder används inte alls i någon större uträckning.

En annan stor skillnad mellan universitet och företag, är som vi nämnt tidigare, att företag väljer att minska på analysdelen. Svaret på det är för att företag vill få ut produkten till kund vid en viss deadline, för att de är beroende av att pengar rullar in. Men varför väljer företag att minska analysdelen när metoder som undervisas på universitet visar att en väl genomarbetad analysfas är lösningen på en bra slutprodukt som kräver mindre underhåll. Vad *vi* inte vet är vad underhåll innebär. När vi i skolan är klara med ett projekt behöver vi inte tänka på vidareutveckling eller underhåll, något som företag idag har som en stor del av sin vardag. Därför är argumentet att underhåll minskar vid en ordentlig analys och designstudie är svår att argumentera för som student.

Vi skulle vilja avsluta med att säga att det är viktigt att tänka på att objektorienterad analys och design går ut på att utveckla system och inte förändra omgivningen, men man bör alltid ha i åtanke att allt som görs kan skapa förändringar någon annanstans. Rädslan av att bli ersatt med ett system finns hos många användare, men detta är inte något ett företag som sysslar med systemutveckling kan ha i åtanke.

6.1 Framtida forskningsförslag

Under vår studie har det vuxit fram idéer och tankar kring vad som skulle kunna vara intressant att fortsätta att fördjupa sig i. En intressant aspekt skulle vara att göra en fallstudie av endast ett företag för att studera deras arbetsmetoder mer djupgående. Ett alternativ skulle kunna vara att följa ett projekt där man studerar projektledare och produktionsledares sätt att arbeta sig igenom ett systemutvecklingsprojekt.

Eller spinna vidare på frågan: Varför väljer företag att minska på analysdelen när metoder som undervisas på universitet visar att en väl genomarbetad analysfas är lösningen på en bra slutprodukt som kräver mindre underhåll?

7 KÄLLFÖRTECKNING

7.1 Litteratur

Andersen, ES. (1994) *Systemutveckling/Principer, metoder och tekniker*. Lund: Studentlitteratur.

Apelkrans, M. och Åbom, C. *OOS/UML en objektorienterad systemutvecklingsmodell för processorienterad affärsutveckling.* Lund: Studentlitteratur.

Backman, J. (1998). *Rapporter och Uppsatser.* Lund: Studentlitteratur.

Brown, D. (2002) *An Introduction to Object-Oriented Alaysis: Objects and UML in plain English.* John Wiley & Sons ltd.

Kruchten, P. (2002) *The rational unified process: en introduktion, svenska utgåvan.* Addison-Wesley, London.

Lunell, H. (2003) *Fyra rundor med RUP*. Lund: Studentlitteratur.

Mathiassen, L., Munk-Madsen, A., Nielsen, P, Stage, J., (1995). *Objektorienterad analys och design.* Andra upplagan. Lund: Sudentlitteratur.

7.2 Internet

Handelshögskolan vid Göteborg universitet
Vasagatan 1, Box 600, SE 405 30 Göteborg.
Kursinnehåll [www dokument]. URL http://www.handels.gu.se

Umeå universitet
SE-901 87 Umeå
Kursinnehåll [www dokument]. URL http://www.umu.se

Växjö universitet
351 95 Växjö
Kursinnehåll [www dokument]. URL http://www.vxu.se

Högskolan Dalarna
791 88 Falun
Kursinnehåll [www dokument]. URL http://www.du.se

Getting started with UML (Januari 2005) [www dokument].URL http://www.uml.org

Object Management Group (Maj 2005) [www dokument]. URL http://www.omg.org

International Data Group [www dokument]. URL http://www.idg.se/

Fägnell, U. (Mars 2003) *RUP - En introduktion* [www dokument]. URL http://www.pellesoft.se

BILAGOR

7.3 Bilaga A

7.3.1 Mall för analysdokument

1. Uppgiften. Kort beskrivning av dokumentets bakgrund och sammanhang

1.1 Syfte. Den övergripande avsikten med systemutvecklingsprojektet.

1.2 Systemdefinition. Sammanfattning av systemets övergripande egenskaper. Se VATOFA-kriterier i avsnitt 2.7

1.3 Omgivning. Beskrivning av relevanta aspekter av omgivningen. Kan bland annat inkludera rika bilder. Se avsnitt 2.3

1.3.1 Problemområde. Informell presentation av centrala fenomen i systemets problemområde.

1.3.2 Användningsområde. Informell presentation av aktörer och arbetsuppgifter.

2. Problemområde. Beskrivningar av klasser, strukturer och dynamik. Se del II.

2.1 Kluster. Klusterstruktur. Se avnsitt 4.2

2.2 Struktur. Klassdiagram som täcker generaliserings-, aggregat- och associationsstrukturer. Se kapitel 4.

2.3 Klasser. Klasserna beskrivs individuellt. Ge för varje klass en beskrivning av:

2.3.X.1. Definition. Kort beskrivning av klassens objekt.

2.3.X.2. Beteendemönster. Detta kan till exempel beskrivas med användning av ett kommenterat tillståndsdiagram. Se avsnitt 5.2

2.4 Händelser. Händelsetabell och sekvensdiagram för relevanta gemensamma händelser. Se kapitel 3.

3. Användningsområde. Fullständig beskrivning av användning, funktioner, gränssnitt och andra krav på systemet. Se del III.

3.1 Användning. Beskrivning av systemets interaktion med omgivningen. Se kapitel 6.

3.1.1 Överblick. Aktörtabell som visar vilka aktörer och användarfall som är inblandade i interaktionen.

3.1.2 Aktörer. Aktörspecifikation för alla aktörer.

3.1.3 Användarfall. Användarfallspecifikationer eller tillståndsdiagram för alla användarfall.

3.2 Funktioner. Beskrivning av systemets funktionalitet. Se kapitel 7.

3.2.1 Fullständig funktionslista. Lista över alla funktioner, med funktionstyp och komplexitetsuppskattning för var och en.

3.2.2 Specifikation av funktioner. Komplicerade funktioner specificeras så detaljerat som krävs.

1.3 Användargränssnitt. Sammanhängande presentation av centrala krav på systemets användargränssnitt. Se kapitel 8

1.3.1 Dialogstil. Beskrivning av den grundläggande stilen för presentation och dialog och fullständig lista över komponenter i användningsgränssnittet.

1.3.2 Överblick. Ett navigeringsdiagram för användargränssnittet som helhet

1.3.3 Exempel. Kommenterade exempel på användargränssnittet.

1.4 Den tekniska plattformen. Skiss av den teknsika plattformen och gränssnitt till andra system och apparater.

2. **Rekommendationer.** Argumentation gällande det efterföljande utvecklingsarbetet.

1.1 Systemets användbarhet och genomförbarhet. En bedömning av kravens förhållande till omgivningen och de tekniska möjligheterna.

1.2 Strategi. Rekommenderad strategi för det efterföljande utvecklingsarbetet.

1.3 Utvecklingsekonomi. Uppskattning av resurs- och tidsåtgång i det efterföljande utvecklingsarbetet.

7.3.2 Mall för ett designdokument

1. Uppgiften. Kort beskrivning av uppgiften och de uppställda kvalitetsmålen.

1.1 Syfte. Den övergripande avsikten med systemutvecklingsprojektet.

1.2 Rättelser av analysen. Rättelser av fel samt nödvändiga modifikationer och kompletteringar av analysdokumentet.

1.3 Kvalitetsmål. Sammanfattning av prioriterade designkriterier och kompletterande mål för arkitekturen. Se kaptiel 9.

2. Tekniska plattform. Kort beskrivning av designspråket och av utrustning, basprogramvara och system på vilka systemet utvecklas och realiseras.

2.1 Utrusning. Beskrivning av relevant utrustning

2.2 Basprogramvara. Beskrivning av relevant programvara

2.3 Systemgränssnitt. Beskrivning av gränssnitt till system som systemet kommer att interagera med.

2.4 Designspråk. Beskrivning av det använda designspråket med hänvisning till kända språk och standarder.

3. Arkitektur. Beskrivning av systemets strukturering i komponenter och processer. Till detta hör en beskrivning av standarder för arkitekturdesign. Se del IV.

3.1 Komponentarkitektur. Kommenterat klassdiagram som visar systemets strukturering i relaterade komponenter. Se kapitel 10.

3.2 Processarkitektur. Fördelningsdiagram som visar de tillgänligaprocesserna, de aktiva objekten och deras förbindelser. Se kapitel 11.

3.3 Standarder. Använda designstandarder.

4. Komponenter. Beskrivning av modell-, funktioners-, systemgränssnitt, användargränssnitts- och andra komponenter. För varje komponent beskrivs:

4.X.1. Struktur. Klassdiagram som beskriver komponenternas klasser och deras strukturella relationer och även ger namnen på deras attribut och icke-triviala operationer.

4.X.2. Klasser. Klasserna beskrivs i den utsträckning som behövs om beskrivningen inte redan framgår av punkt 4.X.1. för klassen Y ges nödvändigt en beskrivning av:

4.X.2.Y.1 Klassens namn

4.X.2.Y.2 Kort beskrivning av klassens ansvar och syfte

4.X.2.Y.3 Attribut (i from av en lista)

4.X.2.Y.4 Komplicerade operationer, med användning av operationsspecifikation. Se kapitel 13.

4.X.2.Y.5 Tillståndsdiagram för att beskriva klassens beteendemönster.

5 Rekommendationer. En underbyggd plan dör det följande utvecklingsarbetet.

5.1 Systemets användbarhet. En övergripande utvärdering av hur designen förhåller sig till omgivningen, grundad på de uppställda kvalitetsmålen.

5.2 Plan för ibruktagande. Rekommenderad plan för hur systemet ska tas i bruk.

5.3 Implementeringsplan. Rekommenderad plan för realiseringen av systemet, med aktiviteter och uppskattningar av resurs- och tidsåtgång. Kanske bara en referens till den relevanta projektplanen.

7.4 Bilaga B - Intervjuer

7.4.1 WM-data, Cross Industry Solutions AB, Raindance

Intervju med Mikael Skogström, projektledare för Raindance användargränssnitts grupp på Cross Industry Solutions AB ett bolag inom WM-data koncernen.

Raindance är ett ekonomisystem som finns installerat hos cirka 600 kunder – allt från stora industriföretag till små kommuner. Omkring 120 personer i Sverige och Finland arbetar med att utveckla, anpassa och ge support på systemet. Som användare är man med och påverkar den fortsatta utvecklingen.

Kan du förklara hur ni går tillväga i ert arbete i analysfasen inför ett projekt, på vilket sätt arbetar ni?

Vi har en produkt och det som händer för oss är mycket förvaltning, vi får önskemål om nya produkter eller om vi han idéer till nu funktioner som vi tycker borde skrivas om. Vi kanske tycker att de behöver skrivas om för att de t.ex. är dåliga eller ålderdomliga. Då har vi en användarförening som vi bollar med sen har vi systemförvaltare och kontaktpersoner hos kunder som vi också pratar med och det kan vara kunder som skulle kunna vara intressanta för det här eller kunder som redan använder funktionen, eller som har sagt att de skulle vilja att den skrevs om, det finns alltså olika varianter av kontaktpersoner. Vi frågar dessa personer hur de skulle vilja att det fungerade och vilka funktioner de saknar, och vilka möjligheter som behöver tillföras, och sen skriver vi ihop en kravspecifikation som de tittar på vi föreslår också en lösning där i, den lösningen är då dels hur användare ska uppleva den, dels hur programmeraren skall utföra den. Vi har alltså dels hur användaren ska interagera dels för utvecklaren hur detta skall genomföras och hur problemen skall lösas. Den passerar sen ett antal instanser, iblandade kunder, användarföreningen, och vår kundservice. De vet ofta vilka problem kunderna har och vilka följdeffekter detta kan få.

Det är väl så att vi använder mest ord för detta vi uttrycker funktionaliteten i ord. Skärmbilden försöker vi visa med antingen ett gränssnitt eller så ritar vi fejkade skrämbilder, vi använder inte UML för detta, utan enbart text och jag skulle kunna tro att det beror på att de flesta som jobbar här har jobbat med detta väldigt länge innan UML existerade. Då har am, det finns nya som har jobbat med UML om har tyckt att det är dålig återkoppling mellan vad man har ritat och vad som faktiskt sen blir kod. Är det så att du har gjort kod av det kan det vara svårt att få koden att uppdatera modellen så att dokumentation och kod stämmer överens. På detta sätt arbetar vi.

Vi använder oss dock av användningsfall, vi använder kanske inte tillräckligt många noviser men att man skulle behöva finfördela det mer.

När ni skriver ihop kravspecifikationen ofta händer det att kunden är med och bestämmer, att kunden berättar hur det fungerar i dess företag, vilka scenarios man har osv?

Scenario är användningsfallen och det är olika för olika funktioner om det handlar om att skriva om en befintlig funktion då har kunden en hel del möjligheter att påverka sen är det så att en del kunder fungerar annorlunda än andra, när de märker att man lyssnar på dem å vill de plötslig ha mycket mer att säga till och det gäller det att lägga det på en normal ambitionsnivå för annars läggs för mycket tid på en viss funktion eller en viss kunds önskemål. Vi ska ju

försöka fördela ett väldigt begränsat antal timmar på så många som möjligt och se till att vi gör störst nytta.

Raindance, är det ett konsultföretag, arbetar ni härifrån eller sitter ni hos kund?

Raindance är en produkt ett ekonomisystem och det är ett system som används av en mängd olika organisationer och företag bland annat Claes Olson, Åhléns, Telia, Västra Götalandsregionen, Region Skåne, då pratar ja hela landstingen, Luleå kommun, Chalmers, Stockholmsläns landsting osv. det är både offentlig och privat sektorn men vi har både sistone vinklat in oss till den offentliga sektor eftersom på den privata sektor där finns det många andra aktörer så vi känner att är vi bättre på de rutiner som finns inom den offentliga sektorn. Om vi ska tävla med Davision, Acapta och SAP och alla möjliga då känner vi att vi har en bättre möjlighet på den offentliga sektorn.

Just detta med att arbete mer på att dokumentera att göra mer utförliga användarfall mer enligt UML eller andra klassbeskrivningar där man tar fram klasser och databaser och allt det, om vi hade tid och inga begränsningar skulle ni vilja arbeta så?

Det finns vissa tekniska förutsättningar som begränsar oss vi kan säga så här överhuvud taget så här, är mina kollegor väldigt medvetna om var det brister och vad man skulle kunna förbättra och ibland kan det kanske inte finnas gehör för vissa förändringar som skulle kunna behövas. Ledningen tycker att dessa idéer är bra med dessa skulle stjäla tid, på så sätt att det skulle vara förändringar som inte syns utåt för kund och då har man vi har nytta av det men kunderna skulle ifrågasätta vad vi håller på med det syns inte att det händer något, men en sådan förutsättning är att vi har e egen plattform, man kan säga att den här produkten startade fanns inte Java och man ville ha något om gick att köra på flera operativsystem, man utvecklade en egen pattform och applikationerna skriver i ett eget språk som liknar Pascal eller den typen av språk och det är då applikationer som innebär väldigt mycket kod och det skriver man inte om, men detta har lett till att man har ett system som t.ex inte kan exportera sin systemmodell som xml så att man skulle kunna suga upp det ur rational rode tex. Sådana kopplingar skulle man kunna bygga och det har nu börja eftersökts av allt fler, och det har funnits diskussioner om man verkligen ska ha en egen databas, vi har även stöd för externa databaser, sql-databaser, systemet levereras ned en intern databas eftersom det var det bästa när det en gång storartade och det betyder att det kan vara lurigt den databasen är väldigt snabb oh effektiv men den er lite andra förutsättningar än en sql-databas de ä väldigt bra på trasaktier men den är inte bra för datalager tex. Det är lite svårt att underhålla datamodellen som vi har i diagrammodellen med vi löser det då men hjälp av manuella krafter vi har en grafisk beskrivning av datamodellen och den ligger i ett verktyg.

Hur arbetar ni med ett system som är gammalt och vissa delar inte objektorienterat, ändar ni det för att t.ex. få ett snabbare och bättre system?

Vi har underhållsavtal och dessa fungerar att vi hjälper kunderna med deras problem och ser till att kunden får ett antal uppdateringar varje år och då förväntar de sig att vi betalar för detta här och även de nya funktionera. Detta är också en avvägning, vad är nya funktioner och vad ska vara uppdatering. Man kan säga det att det finns många medarbetare, och då menar jag både gamla och unga, då finns det ett konsensus att vi borde skriva om viktiga delar av systemet objektorienterat. Vi ser fördelar med att kapsla in problemen osv, men för att kunna göra det måste vi kanske börja i en annan ände med att dra ut modellen över hur systemet ser ut idag, detta har vi koll på men det är lite krångligt.

De som gör bas applikationen i själva operativsystemet håller på med ett jättearbete att skriva om det till en ny teknik som gör det möjligt att dels jobba med att köra de gamla applikationerna, men också att göra en ny typ av applikationer som då är mer modulera. Där vi dessutom skulle kunna kapsla in externa programvaror i vår egen programvara, vi skulle kunna använda oss av CVS ordhantering istället för vårt egna. På detta sätt har vi då infört objektorientering men utan arv, men när vi skriver ny kod skriver vi objektorienterat

Hur mycket efterarbete är det i förhållande till förarbete, själva analysdelen?

Analysen av en funktion som inte är jämförbar med en installation av ett helt system. Det som jag skulle vilja påstå vad som gäller analysen är om man gör den bra behöver man inte göra om lika mycket, det är våra erfarenheter. Jag jobbar med en grupp som heter användar-gränssnittsgruppen och vi gjorde så i våras att vi började jobba med et nytt gränssitt för våra portaler. Det är ett gränssnitt där vi skulle lägga menyn vägrätt i stället för lodrätt, den skulle vara uppbyggd på ett annat sätt menyvalen skulle ha hand om begrepp. Menyn skulle också fungera på ett sätt som gör att genom en klickning på den veta vad man har valt så att man får en återkoppling på vart man är någonstans. I och med detta skulle vi få en annan sidbredd vi behövde därför också titta på hur designen skulle göras om. Vi har gjort om väldigt mycket, och vi har då för vissa av den här fasen gjort tester med slutanvändare, folk som då sitter på Gislaveds kommun i Skövde sjukhus. Vi bry oss inte om vad kundens IT-chef tycker om det, vi ville se vad slutanvändaren tycker. Vi kan egentligen inte säga att testen överraskade oss, för vi hade nog tänkt på dessa saker tidigare, men däremot fick vi stöd för det vi hade upplevt kunde vara problem och det som vi upplevde vara bra. Vi kunde få stöd för att vi kunde jobba mer med de här problemen.

De personerna ni kontaktade, hade de jobbat med det gamla gränssnittet?

Ja, vi ville se om de kunde hitta i det nya gränssnittet och förstå vad de skulle göra. Eftersom vi hade bytt beteckningar var det också en liten utmaning för användaren och det här arbetet gjorde vi inte ensamma, vi använde oss av WM-datas User Focus grupp, som är en användarenhetsgrupp, för att få ett bollplank att jobba med.

Annars är det ovanligt att man pratar med slutanvändare, utan många gör det vanliga att man pratar med kundens representant och det tyckte vi var väldigt viktigt att man lät bli. Det vi kan se av detta var att vi fick mer trovärdiga svar, representanterna har säkert en väldigt klar blid, men det är bara en person, nu får vi flera personer på samma företag. Vi får också ett bättre statistisk underlag och det betyder när de då i efterhand dyker upp åsikter kan vi motivera varför vi har löst det på ett visst sätt.

Hur ställde sig användarna till det nya gränssnittet?

De tyckte att det nya såg bra ut och för vi kunde ju förklara till varför vi gjorde förändringen. Vi tog bort en meny till vänster för att ge större arbetsyta och vi ville samtidigt ha en meny som var mer arbetsinriktad.

Vårt system sälj i en grundfunktion man får en server och en klient och alla ekonomi-applikationer, man får egentligen allt tillsammans, men alla webbdelarna säljs individuellt t.ex. om man vill ha en fakturaportal eller kundfakturaportal, men nu har vi då Raindance-portalen som innehåller leveransfaktura, internfaktura, kundfaktura, anskaffning och den

innehåller den även en informationsmodul som består av olika delar, men det beror på vad de vill använda för slags produkter.

Har ni fasta roller inför ett projekt, eller kan detta variera beroende på olika projekt?

När det finns en funktion eller ett önskemål om en funktion, hamnar den i en lista man vet t.ex. att den hör till en speciell del av produkten och då är det så att den delen an av produkten har en projektledare och styrgrupp, och man har en diskussion mellan styrgruppen, projektledaren och användarföreningen. Detta synkas inom styrgruppen och det bestäms vilka funktioner man detta år ska satsa på, men är man ändå är inne i koden kan man göra ändringar, det blir mer effektivt så.

Jag själv har börjat titta på kravhantering för jag tycker att det är en väldigt viktig del som man inte alls pratar om inom utbildningen och den delen att hitta bra verktyg och hitta bra metoder känns någonting om missas. Man pratar om UML, man pratar om användarfall men dessa användarfallen brukar ha sin grund i krav, och hur ska man prioritera krav och hur ska man hantera dem.

När kravspecifikationen är klar, blir den en offert, eller en bilaga?

Vi jobbar inte med offerter, utan vi jobbar kontinuerligt, men vi kan ha en blivande kund som ställer vissa krav och de kraven går då in i vår kravprocess och sen får vi se hur mycket av det som går att genomföra, sen får vi återkomma med det till kunden.

Hur fungerar arbetsgången?

Vi har säljare som först skapar relation med blivande kunder, när sedan dessa intressenter har blivit kunder går kommunikationen över till kundtjänst när det gäller problem och önskemål, men då är det mera löpande önskemål. Är det en specifik funktion de skulle behöva är det något kunden tar upp med konsulten när denne är på plats vid t.ex. installation eller underhåll av olika slag. Detta kan leda till konsultuppdrag eller att det återkommer till utvecklingsgruppen och blir en funktionsförbättring.

Vilka problem kan uppstå under utvecklingsfasen?

Det finns ett par problem, ett problem är att man oftast ska bygga ihop det här med en befintlig kod och att man inte förstår vad den koden gör, eller olika variabel betydelse, det finns alltid olika traditioner mellan olika systemutvecklare. Det finns bra skrivregler som man kan följa, kodecomplte från Microsoft t.ex. Mycket av problemen skulle läsas om man hade mer disciplin och följde regler. Det har allmänt och traditionellt sätt funnits ett motstånd hos utvecklare och programmerare att var disciplinerade, för de vill vara fria konstnärssjälar. Detta gäller de flesta organisationer jag har stött på. Men det är inte så att folk är rabiata motståndare, men mycket av detta skulle kunna vara internutbildningar, men det finns sällan tid till det.

Detta är ett kvalitativt arbete, och det är generellt sätt svårt att jobba med det, men vi har försökt att förbättra oss på detta, kvalitetsarbete måste på lång sikt vara förebyggande man ser till att undvika problem och det är därför jag tycker att tester och analyser är viktigt, för det är ett sätt att förebygga eventuella problem. Att rita figurer istället för att beskriva i text är ett

tydligt och konkret sätt att visa ritningar, att jämföra vid t.ex. ett husbygge. Detta kan appliceras på programmering. Ingen skulle kunna acceptera en läkare med gamla kunskaper, man kräver att en läkare vet det senaste annars kan han inte hjälpa, och detta får man räkna med när man jobbar med kunskapsyrken, man måste hela tiden lära sig nytt och vidareutbilda sig.

När installationen är klar, får kunden utbildning?

Det ingår i offerten, utbildning finns med i paketet, och det är konsulterna som håller i det och konsulterna håller även i installationen. Ett ekonomisystem är inte bara till att installera, det ska konfigureras också, det finns ingen färdig lösning utan det är olika hos olika organisationer. Cirkulationsmallar ska läggas upp för olika fakturor och detta blir alltid olika för olika kunder. De som gör detta är inte tekniska konsulter utan ekonomiska konsulter. Ekonomikonsulterna är också inblandade i framtagningen av gränssnittet, men testet sker i slutändan på användaren.

Man borde jobba mer förebyggande, och hitta problemen före kunden gör det eller åtminstone samtidigt, men det är så att konsulterna åker till kund om det får betalt och underhållsarbete kan vara svårt att debitera mer mindre än att kunden säger att vi att ni gör detta eller vi vill att ni ska installera den här funktionen.

Vi håller på att installera ett ärendehanteringssystem som kommer hantera ärenden. Det är utvecklingspunkter, eventuella fel, allt som har med utvecklings och felaktighetsarbetet att göra. Det kommer att finnas tillgängligt för alla som har en webbläsare, där kan alla inblandade gå och registrera och titta, jag tror mycket på den lösningen. Jag tror att det är många utvecklingsföretag är dåliga på att dokumentera idéer, felaktigheter och ärenden.

Finns det något annat sätt du skulle vilja arbeta på?

Klassisk systemmodellering handlar om att ta reda på vilka objekt som finns och dessa binds ihop för at se vilka relationer de har och hur information skall flöda och hur detta skall interagera. Men användargränssitt är det svårare för då kan det vara så att du har ett gränssnitt som innehåller andra gränssitt som kommer från annat håll. Det finns försök att beskriva sånt i UML, det finns en variant som handlar om interaktion, men den känns inte riktigt fullvärdig än. Det slutar nog med att vi kommer att använda flash från Macro Media för det går ganska snabbt att bygga ihop något som går att klicka sig igenom, prototyper är väldigt bra för att hitta problematik. Men har man gjort prototypen vill man bakom den ha en systemmodell, att man kan få ett diagram som visar hur allt flödar, och det enda verktyg jag har stött på är ett från Israel, som var alldeles för dyrt. Det jag saknar är ett bra sätt beskriva sådana flöden och modeller. Angående vanlig systemutveckling tycker jag UML fungerar mycket bra, jag tror inte att man ska bli för detaljerad i UML, för då får man problem senare i uppdateringen.

Privat har jag använt UML och har genom det märkt att koden faller rätt snyggt på plats av sig självt, jag får färre problem i testningen och dessa problem har varit lätta att rätta till, t.ex. saker i koden jag inte har sett förut. Som snickare måste du ha en ritning, om du börjar snickra på ett hus utan ritning är du tvungen att ta problemen när de dyker upp och det kan bli mycket dyrt.

7.4.2 WM-data Sverige AB

Den övergripande affärsidén för WM-data är att genom ett brett utbud av design- och IT-relaterade tjänster skapa ökad effektivitet och konkret nytta för valda kundsegment. Affärsidén uttrycker deras fokus på värdet av det de producerar. Fungerande och värdeskapande lösningar kräver en effektiv samverkan mellan människor, applikationer och teknik. Det anser WM-data åstadkomma genom ett komplett utbud av tjänster, vilka de för närvarande levererar från verksamheter indelade i tre områden: bransch-, specialist- och infrastrukturverksamhet

WM-datas hemmamarknad är den nordiska marknaden. Den primära målgruppen är större företag och organisationer i respektive nordiskt land.

Intervju med Carol Wittgren, projektledare på WM-data Sverige AB inom affärsintegration dvs enterprise information portals, intranät, extranät och webbsidor med kopplingar mot diverse affärssystem. Utöver projektledning arbetar Carol Wittgren med analyser, förstudier, framtagning av kravspecifikationer för kunds räkning.

Susanne Källefjärd, systemutvecklare främst inom .Net, och har den senaste tiden arbetat med intranät- och internetlösningar i .Net. WM-data Sverige AB.

Hur arbetar ni?

Vi har väldigt få färdig produkter inom WM-Data utan vi har satsat på antingen förvalta färdiga system hos olika kunder eller systemutveckling, men väldigt lite produktutveckling.

Efter beställning av ett system eller projekt brukar vi ha en utredningsfas eller analysfas där man specificerar upp vad som skall göras och implementeras. Utredningsfasen beror lite på hur insatt kunden är oftast vill inte kunden betala för analysfasen. Antingen får man paketera in det så att det blir som en del i själva lösningen eller helt enkelt kalla det för något annat. För det är en mycket viktig del för att få någonting lyckat hos kunden men det är inte alltid kunden förstår det.

Det andra steget är systemutveckling eller anpassning om det är någon annan produkt man ska arbete med från t.ex. Microsoft. Därefter är det installation hos kund, test och verifiering och efter det lansering.

När ni får in ett projekt, tar ni fram dokument och ritar och låter kund vara med och diskutera?

Under utredningsfasen är kunden väldigt tätt inblandad det är de som har verksamhetskraven, tillsammans med dem tar man fram riktlinjer av vad som skall göras. I vissa fall använder vi oss av UML men i många fall används det inte. Men det är ett mycket bra sätt att beskriva hur ett usecase ska se ut. Sedan kan man återanvända det vid testfasen för att se om det stämmer. I en del projekt går vi tillbaka och tittar men inte i alla.

Vilka har ni kontakt med hos kund, är det projektledare, eller slutanvändaren?

I utredningsfasen är det båda delarna oftast, oftast har de en kontaktperson, men för att få användarkraven måste man prata med användarna och då använder vi oss av workshops, arbetsmöten eller intervjuer.

Om ni hade möjlighet skulle ni använda dokumentation i större utsträckning, och skulle vilja ni analysera mera inför ett projekt?

Det beror lite på vilket uppdrag, och hur stor projektet är, det är alltid ett stöd att ha en tydlig del i början med diagram. Har man ett mindre projekt på ca 500 tim har man inte den tiden att lägga på en tror analysdel. Men om projektet är på ca 5 000 timmar har man möjlighet att lägga mer tid på analysen.

Har en programmerar möjlighet att använda sig av dokumentationen eller ändrar sig det under tiden?

När vi har tagit fram ett lösningsförslag efter utredningsfasen, det förslaget tar projektledare och programmerare tillsammans fram, ex hur kan man lösa det på bästa sätt osv. Då har man redan ändrat det som inte är möjligt så att projektledaren är redan medveten om att vissa sätt går inte att lösa. Programmeraren ser till så att projektledaren inte lovar guld och gröna skogar.

Vilka är de vanligaste missförstånden?

Det är att man pratar olika språk, att kommunikationen brister.

Använder ni bilder för att underlätta för kunden att förstå, i stället för att kanske bara beskriva i text?

Ibland tar vi fram prototyper på något exempel, men i övrig tar vi inte fram bilder som man t.ex. gör i UML.

Skulle det underlätta mer om ni skulle använda er mer av bilder i er beskrivning?

Det krävs att man har insikt i UML, de olika objekten, vad det är osv. Det ställer en del krav på kunden att de ska har lite förståelse för det.

Det är svårt för användarna de har svårt att se hur ett system ska funka innan de har sett det. När de väl får se ett system tittar de bara på bilder, textstorlek färger osv. inte själva funktionaliteten för systemet.

Om ni visar ett gränssnitt vad säger kunden?

Användaren lägger mycket tid på att bedöma färger och typsnitt. Vi har ett exempel nu med en kund som har två ledningsgrupper, företaget består alltså av två bolag. I ledningsgrupperna sitter de alltså och bråkar om bilder och färger. Saken är den att visa kunden en överbild av hur det kan se ut, men användaren har svårt att de detta.

Det är inte alltid vi gör den grafiska designen, men diskussionen kommer alltid upp. Användaren har en subjektiv uppfattning om hur det skall se ut, att det fungerar är en helt annan femma. Allt är relaterat till vad kunden vill betala, de vill inte se ett stort analysprojekt, med diagram och bilder över det blivande systemet. I den bästa av världar skulle man vilja arbete efter metodens alla regler, men det finns ingen tid till det och kunden vill inte betala för det.

Har ni standarder för hur ni systemutvecklar?

Vi har ungefär samma sätt att skriva, det är lite underförstått att vi ska driva på ett visst sätt, men vi sätter oss inte ner ock bestämmer t.ex. namn på olika variabler. Vi har ofta samma roller, men är det helt nya grejer diskuterar vi det mellan oss, någon tycker kanske att just den delen på ett system verkar vara roligt då tar den personen den delen. Detta fungerar bra för vår grupp men vi vet att det absolut inte fungerar så i alla projekt..

Hur fungerar kommunikationen mellan varandra internt under ett projekt?

Vi har möten ofta, men samtidigt sitter vi placerade så att vi arbetar tätt tillsammans är det lätt att hålla e bra fungerande kommunikation. Men under större projekt när man är placerade på olika ställen håller vi kontakt via mail, telefon och arbetsmöten.

Blir underhållsfasen mindre om man ökar analysfasen?

Desto mer man specificerar och förbereder innan desto mindre fel blir det när man ska genomföra arbetet. Det har varit en del projekt som timmar har försvunnit för att förarbetet har varit dåligt.

Enligt våra böcker finns det staplar för olika faser under ett systemutvecklings projekt. Minskar man analysfasen ökar underhållet, tycker ni att detta stämmer?

Efterarbetet eller förvaltningen, blir kanske inte större. Men systemutvecklings delen är den största delen, programmeringen tar mest tid. Använder man sig av RUP och dess sätt blir analysfasen större. Men som ni säger med era böcker så stämmer det inte. Även om man lägger mycket tid på analysen tar ändå själva utvecklingen mycket tid.

Dokumentationen är det något kunden får?

Avslut det överlämnar man till kund i form av en rapport.

En anledning till att man gör dokumentation är till för att gå tillbaka till ett gammalt projekt gör ni det?

Absolut många gånger, det är lätt att gå tillbaka och se vad man gjort tidigare för att sedan fortsätta på det. Det lättaste är om man går tillbaka och tittar på det man själv tidigare har gjort. Det underlättar också för vidarearbete om t.ex. någon annan tar över, då är det viktigt att dokumentationen är bra. Det är ganska vanligt att dokumentationen brister dessvärre.

7.4.3 COGZ

Intervju med Camilla Johansson, Användbarhetskonsult och Patrik en till??

COGZ är specialiserade på interaktionen mellan människa och dator. De arbetar med att säkerställa användbarhet hos IT-system och webbapplikationer i samarbete med produkt-utvecklare, samt genom konsultation och utbildningar. COGZ:s mål är att informationsteknik ska vara enkelt, effektivt och tillfredsställande att använda. Först då uppnås syftet med en produkt. COGZ företagsidé är att i samverkan med projektutvecklingsteamet bidra till att

utveckla en produkt som i slutänden blir både billigare och mer omtyckt COGZ arbetar med användbarhet på bred front, alltid ur användarens perspektiv. De anser att deras metoder är kvalitativa och kostnadseffektiva.

Deras tjänster placeras in i följande områden, informationsinhämtning och analys, funktionalitet, struktur & design, testning och utvärdering, utbildning, workshop och beställarstöd

Hur arbetar ni, vilka projekt deltar ni i?

Vi jobbar mest med webbgränssnitt, en del PA gränssnitt. Det enkelt att peka på vad som är fel, det är en enkel värld, det är enkelt att göra om, men i affärssystem är det svårare att peka på vad som kan vara fel, det krävs mycket mer analysdelen. För oss är det då väldigt tacksamt att arbeta med webb det kostar inte lika mycket att göra en analys över ett webbgränssnitt, men med t.ex. webbapplikationer är det inte lika lätt, det är en mer beställningsapplikation.

Vi sysslar med användbarhet, det finns ett skäl till det, vi vill vara oberoende och vi vill stå lite på användaren sida, inte på teknikernas eller köparen sida.

Berätta hur ni arbetar från ett projekts start.

Det är väldigt olika, vi producerar inget, vi arbetar antingen med färdiga system eller med en projektgrupp tillsammans med en teknisk leverantör, vi skulle ju kunna jobba tillsammans med t.ex. WM-data eller Republic och ge dem den kunskap som de saknar. Det varierar också i vilket tidpunkt av projektet vi kommer in, som oftast är för sen. Av olika anledningar är det så, något kan ha gått snett. En kund som t ex. har problem med att inte få kontroll över sin informationsstruktur. Det kan också vara en komplex webbplats med t.ex. 1 000 sidor då ett företag behöver hjälp med att se om de är på rätt väg och då kan vi komma in och bedöma just informationen och strukturupplägg.

Nästan alltid har vi ett konkret projekt där kunden har ett uttalat behov av våra kunskaper. Det är aldrig så att man tar med användare lite för skoj skull, utan man har kunder med ganska höga krav.

Vi försöker att lära upp leverantörer att analysen är viktigt, som med Republic där vi har ett samarbete som fungerar. Det är helt klart på det sättet att desto mer vi arbetar med teknisk leverantör ju mer inser de hur viktig vår kunskap är. Vi har upplevt att det är väldigt svårt att komma in och samarbeta med en teknisk leverantör som redan har fastlagda rutiner, de jobbar efter vissa modeller osv. Då är det svårt för oss att komma in i ett projekt, speciellt när de redan är definierade, hur ska vi då kunna komma in med vår analys om de redan har sin analys som de följer. Vi vet att t.ex. en leverantör känner att de har nytta av vår kompetens, men det är svårt att hitta ett sätt att samarbeta för att de är svårt att redogöra för slutkunden att ett samarbete med en extern leverantör kommer att ske för att lägga mer tid på användbarheten och det finns inte alltid utrymme för det i budgeten. Slutkunden kan också tycka att leverantören skall ha den kunskapen redan.

Jobbar ni i både stora och små projekt?

Det är svårt att säga vad som är ett stort projekt och vad som är ett litet. Vi tittar på kundnyttan, låt oss säga ett resebolag, de kanske har en enkel funktion som de lägger 10 000 kr på den, och för det får dem en analys av deras lösning och som gör att de tjänar 10 miljoner per år. Där kan vi lägga mycket undersökning på ett sånt företag som direkt tjänar pengar på det. Men sen finns det företag med intranätlösning som inte vill lägga pengar alls för de anser att d ras anställda redan kan redan detta, det är svårare att se kundnyttan där kanske. Det är väldigt olika, vi jobba både med stort och smått. Det beror väldigt mycket på vår slutkund också, vilka krav som finns och vilka kunskaper som finns runt detta och vilken kompetens de har själva. Det beror på hur de själva tänker med att jobba användarcentrerat, sen finns det projekt när det verkligen har varit ett krav, att detta system måste vara jätte enkelt.

Är ni med i framtagning innan projektet egentligen startar?

Tyvärr inte, vi är med i analysfasen men tittar på användarbehov och där kan vi ibland brista, för vi har inte det holistiska tänkandet när det gäller affärsnyttan utan vi är egentlige bara medling mellan tekniken och användare, oftast har man beställt en databas t.ex. för en viss funktion och sen får vi jobbet vad gäller att strukturera gränssnittet inte tekniken bakom bryr vi egentligen inte om oss, men vi skulle vilja göra det. Oftast har analysen skett och man inser att detta kommer att bli en komplex databas och hur ska vi få ihop detta på ett snyggt sätt, resultatet kan då bli en djungel av information, och man säger att det kan bli lätt att förstå, men så är det inte, då är det vårt jobb att ta det man har och strukturerar om och få det mer greppbart och det är oftast fel väg att gå, man behöver börja mycket tidigare med det.

Om man går till större webbolag har de den kompetensen, men vi är inte säkra på att man lägger mycket arbete på att ta fram den perfekta lösningen utan tar nog ofta bara det som blir.

Vi har kommit i kontakt med ett företag som vi arbeta på vårt sätt med informationsdesign, användbarhetsanalys i alla projekt från början och detta känns verkligen roligt, medvetenhet har ökat hos företag.

Vi arbetar med något som heter *Personas* som går ut på att ta reda på vilka personer som ska använda systemet. Intervjuer eller enkäter görs för att ringa in olika användargrupper beroende på t.ex. yrkesgrupp, intressen fritid osv., sen får man ut grupper av människor som har gemensamma egenskaper och då skapar man en fejkad identitet, en så kallad *persona*, denna person får ett namn tex. Kalle, han är 40 år och bor i villa, han får representera en del av användaren. Detta kan tyckas abstrakt, men det roliga är att alla i en projektgrupp kan använda sig av detta. Man får in ett tänkande, t.ex. att just den funktionen man jobbar med nu kommer inte att fungera för Kalle. Från detta som kan tänkas väldigt abstrakt blir det konkret tillslut. Det är en del av analysarbetet att göra på detta vis, det blir mycket tydligt.

Hur tar man fram dessa personas?

Innan tänker man igenom vad som kana vara viktigt och vad som kan påverka systemet, t.ex. ålder och så frågar man användarna om t.ex. ålder intressen osv. Efter det kan man passa ihop olika användare och på det sättet skapar man då en fejkad person helt enkelt.

Vilka metoder använder ni utöver personas?

Får vi chansen att jobba som vi vill, jobbar vi utifrån vad vi kallar användarcentrerad systemutveckling, och det låter väldigt enkelt, men det innebär i princip att ni i er roll som

Systemutvecklare sitter inte ensamma på en kammare och tänker utan ni är ute och gör intervjuer. Då finns det en fördel att kontrollera tankar rakt av och har ni då era diagram och sitter med ett gäng användare, vi kan kalla det för referensgrupp, och förklarar för dem via ritningarna ur informationen går, att ett mail kommer in går det via chef osv., men visar ni den här strukturen med diagram för hur informationen flödar, kan det missuppfattas väldigt lätt av en person som aldrig har sett detta förut och då använder vi oftast mycket enklare metoder.

Vi använder oss av stor A3 papper, spritpennor, post-it-lappar och visar till och med bilder och planskisser över en organisation. När det gäller webbplatser bygger vi upp webbens utseende rent konceptuellt vi visar att det finns en sökfunktion och var den ligger någonstans spelar ingen roll. Då kan personen sitta bredvid och säga t.ex: ”den här menyn fungerar inte det måste fixas.” På detta sätt kan vi få in mycket feedback genom att folk diskuterar fram lösningar. Detta kallas då för användarcentrerad systemutveckling.

Vårt arbete innebär att medla mellan tekniker och användare, och i den bästa av världar skulle Systemvetare vara med och ta hand om strukturering och vi kan satsa på gränssnitt och vad som händer på skärmen.

Att jobba användarcentrerat om det ska ske efter boken, ska man göra det från koncept till vilka uppgifter som skall göras och därifrån skall man gå till interaktion, knappar och dialoger. Efter det kommer man till användargränssnitt.

Vi har erfarenhet från ett företag som arbetade på ett traditionellt reklambolags vis. Där lyssnar man på kundens önskemål och tar fram tre designförslag. Kunden väljer då kanske förslag B, men då har man lagt ner tid och pengar på förslag A och C, egentligen vet du i förväg att massor av timmar kommer att läggas på förslag i onödan. Vår tanke är att om du visste vad kunden ville ha från början hade man kunnat strunta i förslag A och C, och tjänat in den tiden för att lägga på kvalité istället. Då gjorde vi ett försök genom att erbjuda workshoppar istället där vi samlade alla som skulle använda sig av systemet tillsammans med teknikerna och cheferna som skulle betala. Vi samlade ibland upp till 25 personer under två dagar, vilket kan tyckas dyrt, men vi är helt övertygade efteråt att det här är sätet att arbeta på, och ett bra bevis på detta är att dessa kunder vägrar nu att gå in i projekt utan att göra workshops innan, och det är vad jag kallar användarcentrerat tillvägagångssätt.

Sen får inte vi alltid jobba med analys och koncept eller med hela projektet på det här sättet, men just att man tjänar i att alla pratar samma språk, det är otroligt värdefullt. Vi tror att andra företag skulle tjäna på att göra så, att t.ex. jobba med workshop i den här utsträckningen.

Vi har också märkt att om du har en person som sitter med i en workshop och tycker till osv, sen efter två månader när produkten är klar får vi aldrig höra att det här är inte vad jag beställt utan alla är alltid väldigt nöjda. Skulle det vara så att han satt och sov på workshopen inser han att han hade sin chans då. Vi får aldrig klagomål om vi arbetar på detta sätt, men vi vet att det finns företag som behöver bygga om slutprodukten för hundratusentals kronor, för kunden säger att det var inte det de ville ha.

Det viktigaste är att man måste fastställa rutiner som man arbetar efter användarcentrerat, och att man från början tar reda på vad som behövs och hur man egentligen gör.

När programmerarna tar över har ni fortsatt kontakt då?

Ja, vi kan ha gjort för dåligt kompetens, vi kan erkänna det då att vi saknar systemvetarens kompetens, att verkligen göra det i detaljnoga, men i vår värld har det aldrig hänt. Vi har aldrig varit med i ett projekt där allt har fungerat, och då är det så att man lämnar oftast mycket till programmerarn, i alla fall i webbranschen. Men vi har märk att programmerarna känner en viss säkerhet att kunna fråga.

Hur kommer ni fram till vem som skall göra vad?

Det är ganska enkelt för oss, vi har olika bakgrund och olika expertis. Jag Patrik har jobbat med detta länge och jag har ett mer holistisk synsätt, jag är också mer slarvig som person och vissa typer av uppgifter passar inte mig, t.ex. att utvärdera ett gränssnitt kräver ganska mycket man måste vara strukturerad. Det skiljer sig lite för att vi har olika erfarenheter och olika personligheter. Annars skulle vi kunna göra samma saker i och med vi har samma utbildning.

Patrik har hand om våra workshops, han har mer erfarenhet av det. Vår kunskap blir aldrig gammal det enda vi behöver egentligen är papper, penna och hjärnan, allt kommer tillbaka till hur hjärnan fungerar. Vi har knappt några digitala verktyg när vi jobbar, vi kan möjligtvis jobba med webbforumlär när vi gör undersökningar på distans. Annars är det t.ex. så att folk kan komma till oss och säga att de har en manual till ett dataprogram, detta är lite för långt och de skulle vilja att vi ser över den och kollar att allt fungerar, det skulle kunna vara ett alldeles utmärkt jobb för oss. Det skulle också kunna vara en kund som säger jag har gjort ett dataprogram, men jag kan inte sälja det för det är svårt att förklara vad det gör. Då har vi åter vår kunskap, vi är inte marknadsförare eller reklammänniskor, men frågan är vad försöker han försöker säga och vad vill han kommunicera. Det är här vi försöker hjälpa till. Kunden kanske inte förstår en Power Point presentation för det är för komplext, då kan vi se över den och hjälpa till och titta på hur man kan strukturera det annorlunda, lyfta fram vissa saker och börja på en övergripande nivå, och detta har inget med tekniken att göra egentligen .

Vilka är de vanligaste kommunikationsproblemen ni stöter på?

Framför allt okunskap om vad vi gör. Vi gör vårt bästa för att förstå vad problem och missförstånd är, vi har ju ett öga för det. Det kan bli lite konflikter när vi kommer in i ett projekt, i och med att vi är väldigt kritiska, det är vårt jobb att vara kritiska till en lösning och är den då redan klar är vi de personerna måste säga att det är fel.

Många säger att det vi sysslar med är vanligt sunt förnuft, men det räcker med fem minuter framför ett gränssnitt sen har vi pekat ut ett antal fel. Många kanske är triviala personer men det räcker att peka ut ett stor grundfel och kunden säger då: ”Ja just det, det tänkte vi inte på.”

Vin måste alltid komma med konstruktiv kritik och med förslag om att göra på ett annat sätt, i slutändan är det ändå alltid kunden som bestämmer om man vill göra på det sättet, och ibland vet man inte heller själv vad som är den bästa lösningen.

När vi som konsulter kommer in, och kunden vet om sina problem och säger att detta ska de ta tag med en gång men det händer inget, och det är frustrerande.

Vi skulle vilja jobba mer nära slutkunden, vi arbetar nu nära leverantören, men vi skulle vilja missionera och säga att vi ställer högre krav, företag levererar inte användbarhet. Det finns mycket dåliga produkter som säljs, många som man inte skulle behöva betala pengar för. Det

finns tidsredovisningssystem som är helt makalöst dåliga och på detta läggs det mycket pengar, vilket mycket beror på okunskap och att man trivialiserar problemen.

Skulle ni kunna gå in ett företag och säga vart problemen finns i informationshanteringen?

Vi skulle vilja säga att de ska använda en systemutvecklare i stället, vi kan gå in och se användarproblem. Vi gör analyser, vi kan t.ex. göra en effektanalys för att se om det verkligen ger den effekt man önskar. Man tittar på orsak och värde, man utgår ifrån effekten som man vill uppnå och vad som krävs för det. Det kan visa sig att vissa utav funktionerna som har framkommit ur en analys inte uppnår något mål, och då kan man strunta i den funktionen..

Teknikföretag och kunder som ni jobbar med har dom märkt skillnad?

Vi tror att många kan tycka att det kan bli dyrare, men vi har också hört kunder som har sagt. ”Ja, vi kör! Men vi måste ha med en kognitionsvetare först.”

7.4.4 Intervju med Republic

Hur gör ni arbetar i projekt?
Beror ofta på vilken del de kommer in. Ibland har en reklambyrå startat och gjort en grund och vi ska producera det, eller så kommer direkt till oss.

Om kund från början, hur vet ni vem som gör vad med strukturer sånt?
Skiftar från kund till kund. Vissa kunder har en klar, detaljerad kravspecifikation när de kommer och vet att det här och det här vill vi ha och det här behöver vi inte. Andra har en ide, t ex att de vill ha ett intranät för att skicka lite meddelanden till varandra. Olika sätt att ta sig an projektet

Vi ställer ju frågor också. Om de kommer med en luddigt formulerad kravspecifikation med vad de vill ha för något så får ju vi ställa frågor för att kunna reda ut exakt vad de är ute efter. Där handlar det rätt mycket om erfarenhet och rutin för att få ut rätt information.
Kommer kunden och säger att de vill ha ett intranät sätter ju inte vi bara igång och bygger någonting utan ställer frågor för att dra ut så mycket som möjligt från kunden.
Dokumentationen som görs är mötesprotokoll, anteckningar från produktionsmöten med kunden, det är inga formella dokument som skrivs. Det finns ju färdiga projektmallar med vilka delar som skall vara med, men något sånt fyller vi inte i. Vi brukar stämma av via mail och sånt också.

Är det så att kunden ofta har en färdig kravspecifikation när de kommer?
Det är olika. Vissa kunder har det andra inte. Det är väldigt få som har det. När de har en färdig kravspecifikation är det vanligt att de har något kopierat från någon annan och så får man ändå gå igenom dokumentet och ställa frågor om de verkligen vill ha det här och det här. Är det säkert att detta skall vara med?

(Ni kastar inte er i systemutvecklingen, utan ni har ändå ett analytiskt moment..)

I de riktigt stora projekten är det då ni samarbetar med t ex Cogz?
Ja, de har ingått som en person i projektgruppen om man säger. Så vi har fört diskussionen men och hon har på nått sätt satt det på papper. Camilla (från Cogz) har vi använt på det sättet att hon har arbetat fram användarvänligheten och gränssnittet.

I skolan arbetar vi mycket med objekt och klasser för att ta fram en fullständig databas. Det får inte finnas någon dubbellagring. Använder ni er av sådana metoder redan från början?
Nu går det bättre att hitta den slutgiltiga databasen/klassdiagrammet ganska tidigt. Det är väldigt mycket erfarenhet. Ibland blir den inte helt korrekt. Men det var många år sedan vi hade problem med att till exempel en person ligger på flera ställen i en databas. Man lärde sig så från början. Jag har läst lite om detta. Man lägger ju inte ner så lång tid på att ta fram klasser till en liten accessdatabas med några få tabeller.

Eftersom ni är ett ganska litet företag är det kanske inte så svårt att komma fram till vem som skall göra vad.
Vi har ju fortfarande folk som inte sitter här på detta kontor som hjälper oss i olika projekt. Det är vi tre som är delägare och sen har den andra Republic två anställda. Så det är inte så svårt att hitta vilka roller som finns till de olika projekten. Alla måste kunna nästan allt. Vi har ju fått rätt bred kunskap så man ändå kan svara på frågor om det är möjligt och så.

Hur detaljerade är ni när ni skriver en kravspecifikation */offert?*
Databasen har vi aldrig tagit med i dokumentationen. Oftast är det ju så att man kommit överrens om vissa saker på ett möte och mail och sådär så offerten mer bekräftar det. Jag har sett vissa offerter som väldigt exakta och uppspecificerad, det känns som om oftast så behöver man inte har det så detaljerat heller. Ibland kan man känna att vissa har väldigt många avstämningar/”sign offer” för varje delmoment och så. Det känns som om det blir så mycket byråkrati.

När vi arbetade mot ett engelskt företag var det ”sign offer” varannan dag liksom, väldigt byråkratiskt. Och då var det som gällde, man fick verkligen inte stiga ifrån den ramen. När vi arbetade med länsstyrelsen var det också så att offertarbetet måste tas in och alla upphandlingar med extrema mängder text som skulle tas fram.

Om ni får in ett projekt, t ex en hemsida som innehåller ganska många funktioner. Tar ni de funktionerna efter hand eller tänker ni igenom innan vilka funktioner som skall vara med? Stöter ni på problem så går ni en annan väg..
Vi specificerar alla funktioner för annars är det svårt att sätta pris och så där. För att kunna sätta priset så måste man sätta sig in i varje funktion. På så vis bryter vi ner systemet/problemet i bitar för att komma fram till priset. Ibland kan det ju vara så att någon del liknar något vi gjort tidigare och då har man en hum om hur lång tid det tar att lösa problemet. Vi säljer ju inte paket där vi inte är säkra på vad det innehåller. Vi vill ju inte sälja grisen i säcken.

Ni arbetar mest med webbutveckling eller arbetar ni även med mjukvaruutveckling?
Vi har arbetat med Raindance, där gjorde vi gränssnittet. Vi arbetade mot användargruppen och stämde av med dem. Så det var vi som var med och tog fram det nya gränssnittet med nya ord och ny meny och så.

Är kunden med vid framtagningen av funktioner?
Vi frågar. Ofta har kunden redan någonting som de inte är nöjda med och då kollar vi varför de inte är nöjda. Republic Consulting har gjort en del uppdrag där vi har byggt system och intranät och de har varit med och implementerat och samlat information. Har haft hand om ”kundträffar”.

I skolan får vi lära oss att om man lägger ner mer tid på analysdelen kommer implementeringen att ta kortare tid och underhållet minska, det vill säga antal fel kommer att minska. Tror ni att det skulle göra någon skillnad om ni arbetade mer med analysdelen än ni gör idag?
Det beror ju lite på vad det är. Det känns ju inte som att det är rocket-sicence vi håller på med. Vi har ju gjort ganska mycket innan. Vi specar ju ner vilka delar som kommer att ingå i systemet och sätter pris på detta. Skulle vi då sätta en stor del på analysdelen så kommer priset att hamna på enorma summor och så känns det som om kundens analys brister. Det är klart att ju mer man tänker efter innan på hur systemet skall se ut ju smidigare går det. Men oftast så är det kunderna som ändrar sig under tiden. De kanske kommer att ”vänta lite vi kanske skulle kunna ha en sådan här”.

Det här sättet att arbeta med analys så ska man ju, tillsammans med kunden komma detta innan när man diskuterar.
Folk kommer ofta till oss och då har de internt diskuterat det hela i ett halvår eller någonting. De kommer hit och säger att om två veckor måste vi ha detta för då är det mässor och grejer. Och då kan man inte säga ”ja, då ska vi bara analysera här lite i två veckor..”
Det är svårt också, för det känns som att hela den här analysmodellen är jättebra på pappret men kunderna börjar tänka på ett nytt sätt när de får ett nytt verktyg. De sätter sig och testar och kommer på att det är ju jättesmart nu när man kan skicka den här grejen, detta föder nya idéer och tankar. Det är väldigt sällan exakt så här vill vi ha det, så här ska det fungera. Oftast blir det ganska mycket process.

Har ni tänkt på det innan, det vill säga när ni började utveckla, att kunden kanske kommer på nya idéer och på så vis förberett systemet för utveckling?
Strategiskt. Merförsäljning. Den stora summan är inte i början utan det är ju att ge dem smarta funktioner och då ger man dem inte hela paketet direkt. Oftast är det bra att börja så för då märker de själva att det är bra och så.

Är ni med vid implementeringen också?
En vanlig hemsida är ju inte direkt så mycket. De är ju uppföljning hela tiden. Kunden kan ju under tiden som vi arbetar gå och kolla hur arbetet fortlöper.
Om vi tex. Har byggt ett internt nät har vi har haft små endagarskurser med folk som ska sitta med det. Men det är mest att man har gått igenom att så här och så här funkar de olika funktionerna. Oftast har de lärt sig det mesta under själva arbetsgången. Problemet med att låta kunden se arbetsgången är att de kan kommentera saker som inte ens är klart och så. Ibland är det bra att kunna ge dem en bild om att något faktiskt händer med systemet. Ofta är det otacksamt eftersom det under ytan inte syns, utan de ser ju mest att bakgrundsfärgen är bra.
Webbutveckling är ändå en relativt snabb utveckling i förhållande till mjukvaruutveckling. Om man jämför med stora IT-konsulter, de stora it projekt de ligger ju oftast på 3-4 år kanske från projektstart till projektslut. Några sådana jätteprojekt gör ju inte vi.

Är folk öppna för IT-tsystem, känner ni av någon vändning.
Det har de sagt länge nu, men det känns lovande. Det var ett tag som folk var lite skeptiska och inte riktigt vågade och inte riktigt visste vad de skulle ha. Vi tror ju att IT är här för att stanna. Vi vet ju inte riktigt hur det är för de riktigt stora IT-projekten, om de fortfarande väntar på större satsningar.

Misstag, vanliga misstag och stora misstag.

Generellt så är det man jobbar på olika sätt med olika kunder. Vissa är ju lite lösare och man har ingen kravspecifikation, ”vi förstår ungefär vad ni vill ha” och så bygger man sådär. Och med vissa kunder funkar det jättebra att jobba så. Problemet med det är att det är svårt att veta vad exakt som står i offerten vilket leder till att projekt har dragit ut på tiden för de har förväntat sig mer än vad vi har tolkat det som. Det tar på krafterna med att ha projekt som bara flyter ut. Det handlar mycket om personkemin, ibland blir det bara en korrektur och sen är allt färdigt. Medan en andra personer har femtio korrekturer på en vecka och i mailkontot ligger det 248 mail.

Är det vanligt att kunden kommenterar småsaker. T ex, vi vill inte ha en blå bakgrund?
Sånt har vi alltid stämt av innan. Men det blir ganska ofta så ändå. Man försöker att leda dem rätt och tillmötesgå dem. De får oftast gå längsta vägen. Ofta har vi en sådan punkt i offerten. Efter avstämning av design, är det den som gäller. Det funkar inte att de håller på och byter fram och tillbaka.

Vi har arbetat med ett företag som hade två huvudkontor i två olika länder som skulle vara med och bestämma. Det ena kontoret var allergiska mot orange medan det andra älskade färgen orange. Så när man hade tagit fram den rätta orange blev det ramaskri från det andra kontoret.

Hur länge har ni funnits? Märker ni några skillnader i framtagningen av projekt?
1994. I början var det väldigt simpla funktioner och statiska saker. När själva boomen kom gjorde vi ett litet sidospår och följde inte med det här wap eller gjorde en förstudie på en halv miljon. Folk ville köpa upp och bli delägare, då hade vi nog inte funnits kvar.

Kommunikationen er emellan och till företag?
Eftersom vi är så få på ett litet kontor är det bara att öppna munnen, eller springa runt det lilla bordet för att hjälpa varandra. När man har varit på möte eller att det är något på gång är det inte heller något jätteofficiellt utan vi har lite planeringsmöten. Ofta blir det att man bara drar igenom informationen för varandra. Det är ganska inofficiellt.

Missförstån?
Om man inte är tillräckligt tydlig. Oftast är de som beställer rätt okunniga. Vi har ju haft en del kunder som tycker att det är underförstått att man skall kunna uppdatera sin hemsida själv. Det ska man inte ens behöva säga. Det har man lärt sig att man måste vara tydlig med sådana grejer. Det sker missförstånd med både dokumentering och muntligt. En del kunder tycker att en mening betyder något underförstått. Vi tycker inte att direkt att det brukar bli så många missförstånd.

En del kunder vill ju ha allting enligt ”skolans regler”, med mallar och allt och då är det bra att anlita någon som verkligen kan det där. (cogz)
Gränssnittsdesign och sånt känns det som om vi kan ganska bra, men vissa användaranalyser är inget som vi har sysslat med så det kan man behöva ha hjälp med. Ofta är det väldigt bra argument gentemot kunder att man arbetar med kognitionsvetare. Ibland vill kunder stryka den biten, ibland inte.

Vad tycker ni skulle kunna förbättra ert arbetssätt?
Ibland känns det som om det kan köra ihop sig eftersom vi är så få. Alla är iblandade i tre projekt samtidigt plus att man har några själv. Det kan vara svårt att planera ibland. Ibland skulle vi behöva strukturera upp bättre, att det här skall vara klart då och detta då.

7.5 Bilaga C

7.5.1 Handelshögskolan vid Göteborgs universitet

Databaser och systemutveckling, 10p
Kursen ges på B-nivå på Systemvetarprogrammet och som fristående kurs. Kursens syfte är att ge teoretiska kunskaper och praktiska färdigheter i analys, design, och realisering av databasbaserade informationssystem. Denna kurs finns dock inte längre idag utan har ersatts med en 5-poängskurs med objektorienterad programmering där dokumentationen skall vara enligt Mathiasen.

Kursens huvudsakliga innehåll är objektorienterad systemutveckling i form av metoder och tekniker för analys och design av informationssystem. Grunderna i realisering av databasbaserade informationssystem både som objektdatabaser men huvudsakligen som relationsdatabaser. Kopplingen mellan objektorienterad modellering och realisering med relationsdatabaser. Databasspråket SQL och dess användning via programmeringsspråket Java. Laborationer i analys och design samt en realisering i ett relationsdatabassystem.

Kurslitteratur: Lars Mathiassen et al. (2001) Objektorienterad analys och design.
Greg Riccardi: (2001) Principles of Database Systems with Internet and Java Applications, Addison Wesley.

7.5.2 Umeå universitet

Systemutveckling och organisationsförändring 10p
Kursen ges på Programmet för systemvetenskap och diskuterar systemutveckling i relation till fenomen som organisationsförändring och digitala affärer, samt modellering i allmänhet och objektorientering i synnerhet.

Kurslitteratur: Holme. Idar Magna & Solvang, Krohn Bernt (1997). Forskningsmetodik - Om Kvalitativa Och Kvantitativa Metoder. Mathiassen, Lars, Munk-Madsen, Andreas, Nielsen, Peter Axel, Stage Jan (2002). Objektorienterad analys och design.

Systemdesign, 5p
Kursen ges som delmoment på B-nivå i kursen Informatik B, 20 poäng på Programmet för systemvetenskap.

Detta kursmoment avser att introducera till centrala metoder, tekniker, och verktyg som används vid systemdesign. Syftet är att skapa en djupare förståelse för möjligheter och begränsningar hos dessa med hänsyn till olika förhållanden av vikt inom de verksamheter i vilka olika former av informationstekniska system ska utvecklas och användas. Kursmomentet syftar också till att skapa en förståelse för allmänna teorier om systemutveckling och användandet av formella modeller liksom kunskaper om projektorganisering och dess relation till både datorsystem och organisationer.

Kurslitteratur: Avison D.E. & Fitzgerald, G. (2002). Information Systems Development: Methodologies, techniques and tools. (Third edition) London. McGraw & Hill. Stolterman, E. (1991). Designarbetets dolda rationalitet. Umeå universitet. Holme, I. M. & Solvang, B. K. (1997). Forskningsmetodik: Om kvalitativa och kvantitativa metoder. Lund.

7.5.3 Växjö universitet

Design av informationssystem, 10p
Kursen ges på systemvetenskapliga programmet och som fristående kurs. Kursen syftar till att ge grundläggande kunskaper inom området design av informationssystem. Genom introduktion av olika modeller, metoder och tillvägagångssätt skall man visa på sambandet mellan en organisations verksamhet och dess informationssystem. Kursen omfattar bla systembegreppet, hårda och mjuka system. Design och redesign av verksamheter/organisationer. Objektorienterad modellering och arkitekturbegreppet. Kognitiva aspekter och användargränssnitt. Samt grundläggande webbdesign och prototyping.

Kurslitteratur: Lövgren, J & Stolterman, E (1998) Design av informationsteknik.
Mathiassen, L m fl (1998) Objektorienterad analys och design. Fagerström, J m fl (1998) Objektorientering i hela företaget - en Integrerad Utvecklingsprocess.
Flensburg, P & Friis, S (1999) Mänskligare datasystem - utveckling, användning och principer.

7.5.4 Högskolan Dalarna

Introduktion till informatik och IT, 10p
Kursen ska ge grundläggande kunskaper om programvara, datorer och datornätverk. Efter kursen ska man kunna förstå och redogöra för olika begrepp inom informatik. Dessutom ska man kunna redogöra för informatikens utveckling som ämne. Kursen innefattar datorers användningsområden som uppbyggnad och funktion samt terminologin inom dataområdet. Datoriseringens sociala, ekonomiska och politiska förutsättningar och konsekvenser samt IT-säkerhet. Praktiskt handhavande av datorer och datornätverk. Kursen redogör också för olika begrepp inom informatik och vilka typer av informationssystem som finns och deras användning i verksamheter.

Kurslitteratur: Checkland, Peter & Holwell, Sue. (1998) Information, Systems and Information Systems: making sense of the field. J Wiley & Sons, Chichester. Lunell, Hans (1994) Datalogi - begrepp och tekniken. Andersen N E, Kensing F m fl. (1986) Professionell systemudvikling. Teknisk Forlag A/S, Köpenhamn. Petersson G. (1997) Att skriva rapporter. Om formen och dess betydelse för innehållet.

Databaser och informationssystem, 10p
Kursen skall ge studenten faktakunskap om databaskonstruktion, och förståelse för hur databastransaktioner fungerar och påverkar det praktiska arbetet vid programmering och administration. Vidare skall studenten förvärva tillämpad kunskap vad avser teknik för datamodellering, och programmering i SQL och procedurella databasspråk, t ex Oracle PL/SQL. Efter godkänd kurs skall studenten inneha grundläggande kunskaper i databasprogrammering och förståelse för hur en databasadministratör arbetar.

Kurslitteratur: Nilesh Shah. (2002) Database system using Oracle. Prentice Hall.Axelsson L, Hidefjäll M. (1993) Praktisk datamodellering - ta greppet om begreppen. Lund,. Ågerfalk P, Goldkuhl G. (2000) Actability: A way to understand information systems pragmatics.

7.6 Bilaga D

Generella utvärderingskriterium,

- Ligger klassen eller händelsen inom systemdefinitionen?
- Är klassen eller händelsen relevant för modellen av problemområdet?

(Mathiasen et al., 2001, s. 82)

Klasser ska de även uppfylla följande kriterium:

- Kan man identifiera objekt från klassen?
- Innehåller klassen unik information?
- Omfattar klassen flera objekt?
- Har klassen ett lämpligt och hanterligt antal händelser?

(Mathiasen et al., 2001, s. 83)

Händelser ska de även uppfylla följande kriterium:

- Är händelsen momentan?
- Är händelsen atomär och kan inte brytas ner i delhändelser?
- Kan händelsen konstateras ha inträffat när den inträffar?

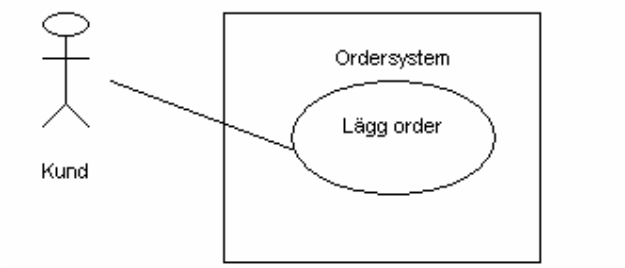
(Mathiasen et al., 2001, s. 84)

Händelsetabell:

	Person	Deltagare	Författare
deltagare	+	+	
artikel registrerad	+		
beslut meddelat			+

(Mathiasen et al., 2001, s. 408)

Användarfallsdiagram:



(http://www.idg.se)

Akörstabell:

	Komitésekreterare	Programkonstruktör
Registrera person	+	
Skapa aktivitet		+
Fråga	+	+

(Mathiasen et al., 2001, s. 410)

Aktörsspecifikation:

Komittésekreterare
Syfte: En person som utför administrativt arbete för en organisationskomitée. **Beskrivning:** De personer som utför arbetet har ett skiftande mått av administrativ erfarenhet **Exmepel:** En student som är tillfälligt anställd för att hjälpa till att organisera konferensen. Erfaren PC-användare.

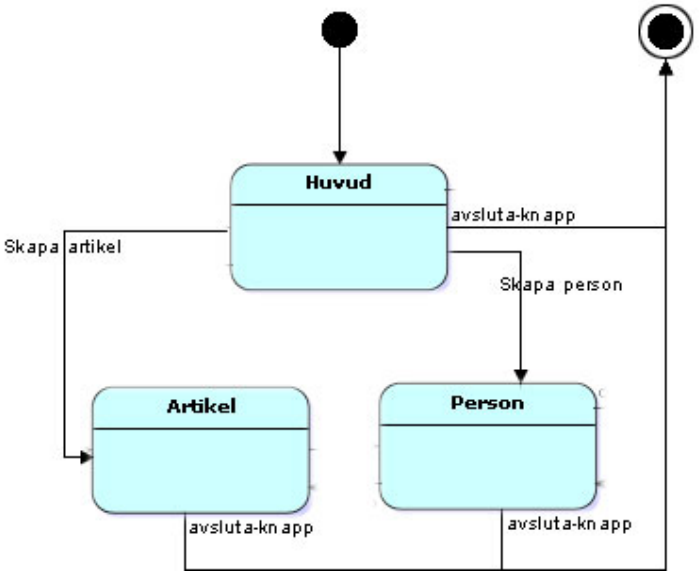
(Mathiasen et al., 2001, s. 411)

Funktionslista:

Funktion	Komplexitet	Typ
Skapa konferens	Normal	Uppdatering
Skapa uppdatera	Enkel	Uppdatering
Kontrollera program	Komplicerad	Avläsning

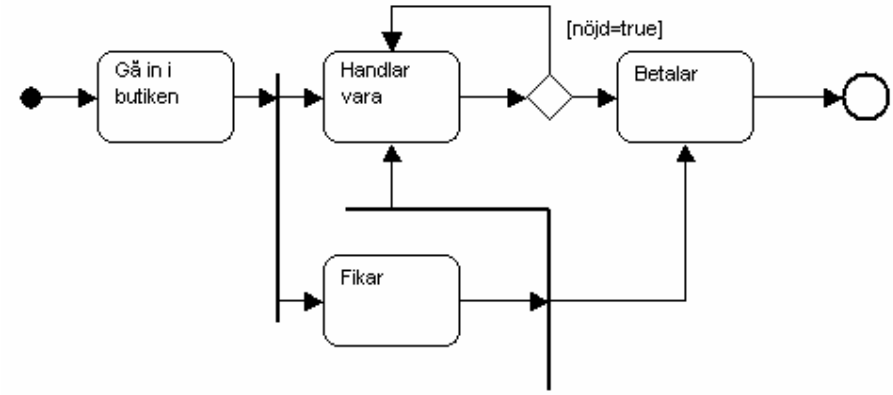
(Mathiasen et al., 2001, s. 412)

Navigeringsdiagram:



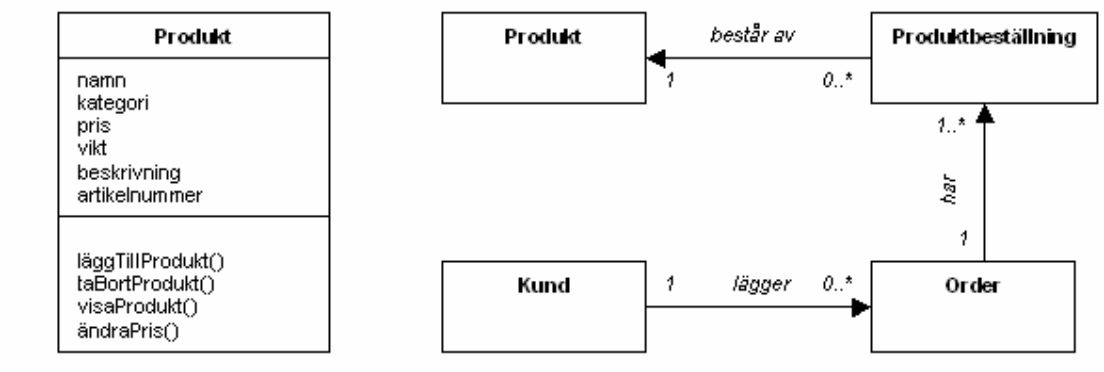
(Mathiasen et al., 2001, s. 415)

Aktivitetsdiagram:



Ett aktivitetsdiagram visar de olika aktiviteter, processer som skall finnas i systemet samt vilken ordningsföljd de har. I ett aktivitetsdigram är det även tillåtet att skriva ut en syntax, det vill säga en kort programmeringsrad. (www.idg.se)

Klassdiagram:



Ett klassdiagram består av samtliga framarbetade klasser och dess relationer till varandra. Det är viktigt att skriva ut vilken slags relation de olika klasserna har. (www.idg.se)

Klasspecifikation:

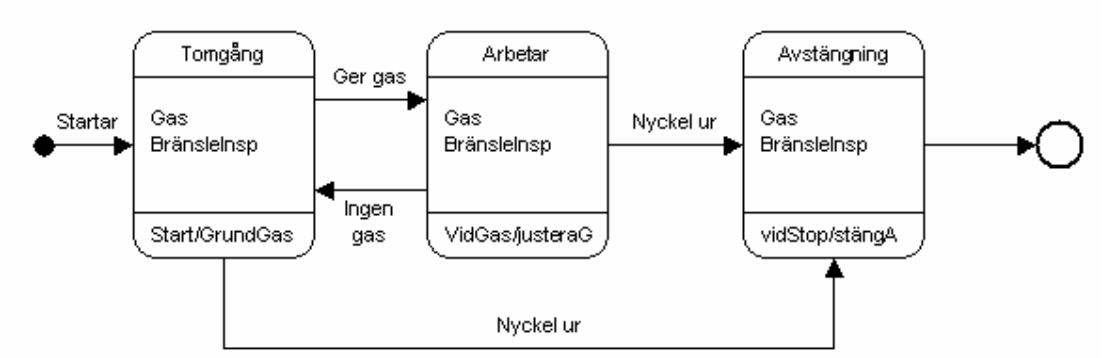
Person

Syfte: Registrera basupplysningar om alla inblandade personer och aggegera varje persons konferensroller.

Attribut: Efternamn, förnamn, institution, adress, e-mail

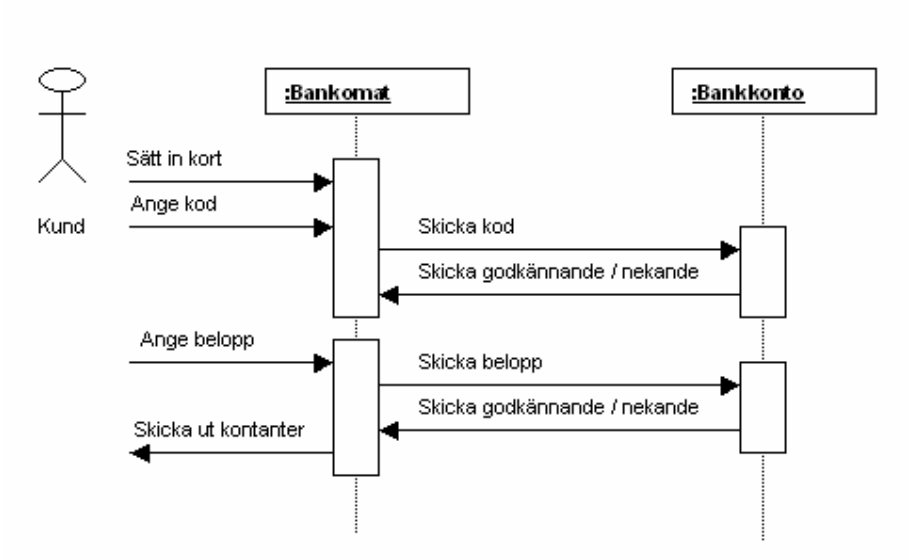
Operationer: Skapa person, skapa roll

Tillståndsdiagram:



Tillståndsdiagrammet skall beskriva de olika tillstånd olika klasser och objekt kan befinna sig i samt vilken faktor som gör att de byter tillstånd. Föds – Lever – Dör. (http://www.idg.se).

Sekvensdiagram:



(http://www.idg.se)