INTERNATIONELLA HANDELSHÖGSKOLAN
HÖGSKOLAN I JÖNKÖPING

Tillämpning av UML

Hur och varför

Filosofie kandidatuppsats inom Informatik

Författare: Johanna Isaksson
Johanna Jansson

Handledare: Britt-Marie Johansson

Jönköping Januari 2005

JÖNKÖPING INTERNATIONAL BUSINESS SCHOOL
Jönköping University

Application of UML

How and why

Bachelor's thesis within Informatics

Author: Johanna Isaksson
Johanna Jansson

Tutor: Britt-Marie Johansson

Jönköping January 2005

Kandidatuppsats inom Informatik

Titel:	**Tillämpning av UML- Hur och varför?**
Författare:	**Johanna Isaksson, Johanna Jansson**
Handledare:	**Britt-Marie Johansson**
Datum:	**2005-01-27**
Ämnesord	**Modellering, modelleringsspråk, objektorientering, systemutveckling, UML**

Sammanfattning

I slutet av 80-talet gick systemutvecklingen in i ett nytt skede. Detta fick som följd att många nya metoder och utvecklingsmodeller för systemutveckling skapades vilket i flera fall ledde till problem vid val av systemutvecklingsmetod och modell. Till följd av detta skapades det idag standardiserade modelleringsspråket UML (Unified Modeling Language). UML är anpassat för att stödja många olika typer av projekt eftersom det tillåter företagsspecifika anpassningar och förändringar.

Syftet med studien är att undersöka hur och varför företag använder sig av UML samt vilka erfarenheter och uppfattningar de som arbetar med UML har av att tillämpa det i praktiken.

För att uppfylla syftet har vi valt att genomföra en kvalitativ studie med semistandardiserade intervjuer. Intervjuerna utfördes på fyra företag i Jönköpingsregionen.

Resultatet av studien visar att den främsta anledningen till att företag modellerar är att det ger en bra dokumentation vilket underlättar utveckling, förvaltning och drift. Vidare har studien visat att UML har valts på grund av att det är en standard som lämpar sig för många olika projekt samt för att UML passar den utvecklingsmodell som tillämpas på företaget. Standardiseringen är även enligt samtliga företag den främsta styrkan med UML. Svagheter i UML anses vara avsaknaden av processdiagram samt bristen på standardiserad syntax i verktygen.

Ju längre UML har använts på företagen desto fler diagram används. De diagram som tillämpas av samtliga företag är användningsfallsdiagram, klassdiagram och sekvensdiagram. I övrigt beror användningen av diagram för ett specifikt projekt på projektets typ och storlek. Däremot utnyttjar inget av företagen UML:s flexibilitet att anpassa syntaxen.

Samtliga företag kombinerar i någon utsträckning UML med RUP eller egenutvecklade utvecklingsmodeller med liknande egenskaper som RUP. Det skiljer sig dock i hur företagen använder diagrammen i samband med utvecklingsmodellerna. Detta beror troligtvis på det iterativa sätt företagen arbetar efter där diagrammen följer med i hela systemutvecklingsprocessen.

Bachelor's Thesis in Informatics

Title:	Application of UML- How and why
Author:	Johanna Isaksson, Johanna Jansson
Tutor:	Britt-Marie Johansson
Date:	2005-01-27
Subject terms:	Modeling, modeling language, object orientation, system development, UML

Abstract

In the end of the 80´s the area of system development moved into a new era. As a consequence many new methods and development models emerged which in many cases resulted in problems when choosing system development models and methods. As a result of these problems the today standardized modeling language UML (Unified Modeling Language) was created. UML is tailored to support many different types of projects. This is possible because of UML's capacity to be adjusted and adapted to a specific company environment.

The purpose of this bachelor thesis is to investigate how and why companies use UML and what experiences and opinions those who use UML have of using UML in practice.

To fulfill our purpose we have chosen to carry out a qualitative study with semi-standardized interviews. The interviews were accomplished on four companies in Jönköping.

The result of the research shows that the primary reason for companies to carry out modeling is because it results in good documentation which makes development, administration and operation easier. Furthermore, the study has shown that the reason that companies have chosen UML is because it is a standard which is suited for various different projects and also for the development model used in the company. The standardization is also, according to all companies, the primary strength with UML. Weaknesses in UML are considered to be the lack of process diagrams and standardized syntax in modeling tools.

There was found to be an increase in the number of diagrams used the longer the companies have used UML. The diagrams applied by all companies are: use case diagram, class diagram and sequence diagram. Moreover, the use of diagrams for a specific project is dependent on the project type and size. However, none of the companies utilize the flexibility to adjust the syntax.

All companies combine UML with RUP or business customized development models with characteristics from RUP. There is, however, a difference in how companies use the diagrams in combination with the development models. This probably depends on the companies' iterative way of working where the diagrams are involved in the whole system development process.

Innehåll

1 Inledning 1
1.1 Bakgrund 1
1.2 Problemdiskussion 2
1.3 Syfte 3
1.4 Intressenter 3
1.5 Definitioner 3

2 Metod 5
2.1 Angreppssätt 5
2.2 Insamling av data 5
2.3 Val av intervjutyp 6
2.4 Val av respondenter 7
2.5 Förberedelse och tillvägagångssätt vid intervjuerna 8
2.6 Uppsatsens trovärdighet 8
2.6.1 Tillämplighet 9
2.6.2 Rimlighet 9
2.6.3 Pålitlighet 9
2.6.4 Noggrannhet 10

3 Referensram 11
3.1 Systemutveckling 11
3.1.1 Livscykelmodellen 12
3.1.2 RUP – Rational Unified Process 12
3.2 Modellering 13
3.3 Objektorienterad utvecklingsansats 14
3.4 UML:s framväxt 15
3.5 OMG och Standardisering 16
3.6 UML:s uppbyggnad 16
3.6.1 Användningsfallsdiagram 17
3.6.2 Klassdiagram 18
3.6.3 Objektdiagram 19
3.6.4 Tillståndsdiagram 19
3.6.5 Aktivitetsdiagram 20
3.6.6 Samarbetsdiagram 21
3.6.7 Sekvensdiagram 21
3.6.8 Komponentdiagram 22
3.6.9 Fördelningsdiagram/Grupperingsdiagram 23
3.7 Diagrammens plats i RUP och livscykelmodellen 23

4 Resultat av datainsamling 25
4.1 Jordbruksverket 25
4.1.1 Systemutvecklingsmodell 25
4.1.2 Modellering 26
4.1.3 Styrkor och svagheter med UML 27
4.1.4 Tillämpning av UML 27
4.1.5 Anpassning av UML samt dess tillämpning i olika projekt 29
4.1.6 UML:s plats i systemutvecklingsprocessen 29

4.2 Pdb 30
4.2.1 Systemutvecklingsmodell 30
4.2.2 Modellering 31
4.2.3 Styrkor och svagheter med UML 31
4.2.4 Tillämpning av UML 32
4.2.5 Anpassning av UML samt dess tillämpning i olika projekt 33
4.2.6 UML:s plats i systemutvecklingsprocessen 33
4.3 Saab Combitech Systems 34
4.3.1 Systemutvecklingsmodell 34
4.3.2 Modellering 35
4.3.3 Styrkor och svagheter med UML 36
4.3.4 Tillämpning av UML 36
4.3.5 Anpassning av UML samt dess tillämpning i olika projekt 37
4.3.6 Tillämpning av UML i systemutvecklingsprocessen 38
4.4 WM-data 39
4.4.1 Systemutvecklingsmodell 40
4.4.2 Modellering 40
4.4.3 Styrkor och svagheter med UML 41
4.4.4 Tillämpning av UML 41
4.4.5 Anpassning av UML samt dess tillämpning i olika projekt 43
4.4.6 UML:s plats i systemutvecklingsprocessen 43

5 Analys 45
5.1 Varför företag använder sig av UML 45
5.1.1 Företagens syn på modellering 45
5.1.2 Orsaker till valet att tillämpa UML 45
5.1.3 Styrkor respektive svagheter med UML 46
5.2 Företagens sätt att använda UML 48
5.2.1 Användning av UML på företagen 48
5.2.2 Diagrammens plats i systemutvecklingsprocessen 49
5.2.3 Anpassning av UML samt tillämpning i olika projekt 51

6 Slutsatser 52

7 Avslutande diskussion 53
7.1 Reflektioner 53
7.2 Förslag till fortsatta studier 54
7.3 Tack 55

Referenslista 56

Figurer

Figur 3.1 Livscykelmodellen (Andersen., 1994, s. 48)..... 12
Figur 3.2 Rational Unified Process (Kruchten, 2000, s. 23) 13
Figur 3.3 Användningsfallsdiagram ”Funktionella krav på skolsystemet”. 18
Figur 3.4 Klassdiagram ”Skolsystem”. 18
Figur 3.5 Objektdiagram ”Kalles kurser”. 19
Figur 3.6 Tillståndsdiagram ”Ansökan till kurs”..... 20
Figur 3.7 Aktivitetsdiagram ”Registrera bok”. 20
Figur 3.8 Samarbetsdiagram ”Skriv ut student- och litteraturlista”. 21
Figur 3.9 Sekvensdiagram ”Skriv ut student- och litteraturlista”. 22
Figur 3.10 Komponentdiagram ”Registrera student”. 22
Figur 3.11 Fördelningsdiagram “Delsystem ” 23
Figur 3.12 UML i Rational Unified Process..... 24
Figur 3.13 UML i livscykelmodellen 24
Figur 4.1 Systemutvecklingsmodell på Jordbruksverket 26
Figur 4.2 Befintlig systemutvecklingsmodell på Jordbruksverket..... 30
Figur 4.3 Pdb – Avdelningar. 30
Figur 4.4 Systemutvecklingsmodell för Saab Combitech Systems: ”Parts”. . 39
Figur 4.5 Systemutvecklingsmodell för WM-data 44

Tabeller

Tabell 1 Diagrammens användning av respektive företag 49

Bilagor

Bilaga 1 - Begreppsordlista..... 1
Bilaga 2 – Intervjuunderlag 2

1 Inledning

I följande kapitel kommer vi att beskriva bakgrunden till arbetet samt de problemformuleringar och syfte vi kommit fram till. Vidare kommer även arbetets intressenter nämnas och definitioner av begrepp beskrivas.

1.1 Bakgrund

Enligt Apelkrans och Åbom (2001) var det oftast de stora företagen som hade möjlighet att använda sig av datateknik när datorer introducerades på marknaden. I slutet av 70-talet blev det även möjligt för mindre företag att dra nytta av datorernas potential och införa informationssystem, då datorer blev mer tillgängliga för allmänheten. I och med detta har systemutvecklingsområdet växt och idag är det ett område som nästan alla organisationer kommer i kontakt med. Utveckling av ett system kan ske på flera olika sätt. Enligt Andersen (1994) är systemutveckling en omfattande uppgift som kräver att systemutvecklaren har goda kunskaper om den miljö som systemet ska utvecklas i. Till en början utgick systemutvecklingsarbetet från ett funktionsorienterat synsätt. Ett funktionsorienterat synsätt innebär att fokus läggs på de funktioner som verksamheten ska utföra. Dessa funktioner bestämmer verksamhetens uppgifter och därmed även innehållet i informationssystemet.

Eriksson, Penker, Lyons och Fado (2004) hävdar att i slutet på 80-talet gick systemutvecklingen in i ett nytt skede och fokus flyttades från funktioner till objekt. Den nya objektorienterade systemutvecklingen skapades. Detta område har sedan grundandet utvecklats snabbt och många olika objektorienterade metoder och utvecklingsmodeller har skapats. Enligt Apelkrans och Åbom (2001) har det objektorienterade synsättet underlättat arbetet och förståelsen för systemutveckling. Idag använder systemutvecklare och programmerare nästan uteslutande objektorienterade verktyg i sitt arbete.

I och med den snabba utvecklingen inom objektorientering och informationsteknologi har enligt Eriksson et al. (2004) behovet av modelleringsspråk hela tiden förändrats och nya har skapats. De menar vidare att det stora utbudet av modelleringsspråk har medfört problem vid kommunikation mellan företag där de olika parterna inte förstår varandras modelleringsspråk. Det kan även vara tidskrävande och svårt att välja modelleringsspråk då olika språk lämpar sig för olika typer av projekt. Till följd av de problem som uppstod skapades modelleringsspråket UML (Unified Modeling Language) som är anpassat för att stödja många olika typer av projekt.

UML bygger i huvudsak på tre tidigare metoder vilka är Booch, OMT-2 samt OOSE. Det var skaparna till dessa tre metoder som under mitten av 1990-talet gick samman och utvecklade modelleringsspråket UML. När UML skapades fick det stöd av många stora företag och grundarna kom då med förslaget att försöka få UML standardiserat av OMG (Object Management Group). OMG som är en icke vinstgivande organisation bestående av många välkända företag, (se avsnitt 3.5) utsåg UML till en industristandard och Moore (2001) menar att detta kan vara en orsak till att UML fått en stor genomslagskraft. Även Kobryn (2002) anser att UML är ett populärt och välanvänt

modelleringsspråk. Han tillägger att metoder och verktyg som stödjer UML har utvecklas och blivit fler i takt med UML:s utveckling och utbredning vilket i sin tur kan underlätta arbetet med UML och bidra till dess framgång. UML är även konstruerat för att tillåta företagsspecifika anpassningar och förändringar såsom ändringar i syntaxen eller komplettering av diagram som inte tillhör UML. Detta för att UML ska kunna tillämpas i många olika typer av projekt, vilket ger användaren en stor frihet. UML är även ett licensfritt modelleringsspråk vilket innebär att det är tillgängligt utan kostnad för alla (Eriksson et al. 2004). Möjligheten till anpassning samt tillgängligheten tror vi kan vara två bidragande orsaker till att UML, som vi uppfattat det, är så vanligt bland systemutvecklare idag.

1.2 Problemdiskussion

Vi har under vår tid på Handelshögskolan i Jönköping genomgått flera systemutvecklingskurser och har samlat på oss diverse kunskaper om modellering och UML samt dess tillämpning. Från skolans sida har det ofta betonats att UML är ett viktigt och ofta förekommande modelleringsspråk vid systemutveckling. Detta har genomsyrat kurserna och vi har bland annat genomfört ett systemutvecklingsprojekt där UML användes som modelleringsspråk. Med dessa intryck av UML har vi gjort vidare litteraturstudier där den uppfattning vi har fått, att UML är ett vanligt och omtyckt modelleringsspråk, har stärkts.

De erfarenheter samt den uppfattning vi erhållit från vidare litteraturstudier har fått oss att fundera på frågor såsom varför företag modellerar och varför UML används som modelleringsspråk i samband med systemutveckling. Är anledningen till detta att UML är ett standardiserat modelleringsspråk eller för att det är, som OMG själva nämner, ett flexibelt modelleringsspråk som passar många olika typer av projekt? I samband med funderingen kring varför UML används som modelleringsspråk har vi även blivit intresserade av vilka styrkor och svagheter som kan identifieras.

UML tillåter bland annat företagsspecifika anpassningar, ändring av syntaxen samt komplettering av andra diagramtyper än de som finns specificerade i UML. Då vi anser att flexibilitet ger upphov till variation i tillämpningen av UML har även frågor angående hur UML verkligen används i samband med systemutveckling uppstått. Hur utnyttjar företag den frihet som UML ger? Är det vanligt att företag ändrar i diagrammens syntax och i sådana fall vad beror det på och vilka konsekvenser får det? Vi anser det även vara av intresse, med anledning av UML:s flexibilitet, att undersöka hur modelleringsspråket kan kombineras med olika typer av systemutvecklingsmodeller samt hur företag arbetar med UML i den valda utvecklingsmodellen.

Även funderingar kring diagrammens användande har uppkommit då UML enligt OMG har som mål att kunna användas i många olika projekt. Kan samtliga diagram, som UML består av, verkligen användas i alla typer av projekt eller finns det projekt där något eller några diagram inte passar in? Med anledning av de funderingar som uppstått kring modelleringsspråket UML kommer vi att gå ut till företag som använder sig av UML och undersöka hur verkligheten ser ut.

Frågeställningarna vi kommer att använda oss av i uppsatsen har vi valt att dela upp i två grupper. De första behandlar varför och de andra hur UML används i praktiken.

För att ta reda på varför UML har valts som modelleringsspråk kommer vi att fokusera på följande två frågor:

- Vilka är anledningarna till att företag modellerar och varför har de valt UML som modelleringsspråk?

Inom ramen för varför UML används har vi även som mål att undersöka vilka erfarenheter och uppfattningar företagen har av UML och vi kommer därmed att ta hjälp av nedanstående fråga för att få svar på detta?

- Vilka styrkor och svagheter kan identifieras i UML?

Vidare kommer vi att fokusera på följande frågor för att undersöka hur UML används:

- Vilka UML- diagram används?
- Hur kombineras UML med olika systemutvecklingsmodeller?
- Görs några anpassningar av UML-diagrammen och kan de användas i alla projekt?

1.3 Syfte

Syftet är att undersöka hur och varför företag använder sig av UML samt vilka erfarenheter och uppfattningar de som arbetar med UML har av att tillämpa det i praktiken.

1.4 Intressenter

Uppsatsen vänder sig till personer med kunskap om systemutveckling. Det kan vara en student som vill lära sig om UML eller ett företag som planerar att införa UML som modelleringsspråk. Uppsatsen kan även vara intressant för en person som är insatt i UML och vill ta del av andra användares erfarenheter.

Vissa begrepp kommer inte att beskrivas löpande i texten utan kommer istället att förklaras i en begreppsordlista (se bilaga 1). Dessa begrepp kommer att märkas med en asterisk (*) vid första förekomsten i varje avsnitt. Vi anser att begreppsordlistan utgör en hjälp för dem som inte är införstådda med vissa av begreppen som förekommer.

1.5 Definitioner

Då modell och metod är centrala begrepp i arbetet samt kan ha många olika betydelser i olika sammanhang, anser vi det viktigt att klargöra vad vi menar med de olika begreppen.

- **Modell –** Enligt Andersen (1994) kan ordet modell användas för två olika betydelser. Den ena betydelsen är att en modell är en översikt över utvecklingsarbetet, även kallat ramverk. Den är ofta indelad i olika faser vilka visar olika arbetssteg. Den här typen av modell kan benämnas som utvecklingsmodell och livscykelmodellen (se avsnitt 3.1.1) är ett bra exempel på denna typ av modell. Den andra typen av modell benämner Andersen (1994) analysmodell. Den här typen av modell utgör en beskrivning som grund för en analys och är ofta en grafisk beskrivning. Ett klassdiagram (se avsnitt 3.6.2) kan till exempel vara en modell som används för att analysera systemets statiska struktur. I detta arbete har vi behov av att använda båda betydelserna av ordet modell. Vi kommer, för att undvika missförstånd, att använda oss av de mer detaljerade begreppen utvecklingsmodell och analysmodell. Ibland används även begreppet systemutvecklingsmodell vilket är synonymt med utvecklingsmodell samt diagram vilket är synonymt med analysmodell.

- **Metod –** En metod är enligt Andersen (1994) en beskrivning av hur ett problem ska lösas och kan användas i flera faser i en utvecklingsmodell. Den är alltså mer detaljerad än en utvecklingsmodell. En metod beskriver vilket arbete som skall utföras, hur arbetet bör organiseras samt hur och vilka beskrivningstekniker* som ska användas. En metod är anpassad till olika typer av problem och det är viktigt att veta vilken metod som är bäst lämpad för ett visst problem. The Object Modeling Technique (se avsnitt 3.4) är ett exempel på en metod. Vi kommer i vårt arbete att använda oss av Anderssons definition av metod.

2 Metod

Kapitlet inleder med valet av undersökningsmetod varefter insamlingsmetod för primär- och sekundärdata anges. Då vi valt att genomföra intervjuer följer en beskrivning av olika intervjutyper samt val och motivering av dessa. Vi beskriver och reflekterar sedan över hur själva intervjuerna genomförts samt ger en motivering av valet av respondenter. Avslutningsvis kommer vi att diskutera uppsatsens trovärdighet.

2.1 Angreppssätt

Enligt Lekvall och Wahlbin (1993) bör undersökningsmetod väljas utifrån problemet och syftet i en undersökning. Det finns i huvudsak två olika undersökningsmetoder: den kvalitativa och den kvantitativa. De hävdar vidare att det som är specifikt för det kvantitativa tillvägagångssättet är att det som ska undersökas handlar om antal, fördelningar eller exakta mätvärden. Den kvalitativa metoden handlar om kvalitet, vilket betyder öppenhet för alla resultat samt att försöka förstå istället för att bevisa en viss teori.

Lekvall och Wahlbin (1993) menar även att fördelarna med en kvantitativ undersökning är att den oftast inte tar lika lång tid eller är lika resurskrävande som en kvalitativ undersökning. Det är också med en kvantitativ metod möjligt att på ett effektivare sätt få in rakare svar samt statistiskt material. Med den kvalitativa metoden däremot är möjligheten större att få ett mer uttömmande svar då respondenten har en större frihet att uttrycka sig och inte är tvingad till att svara på/fylla i förutbestämda alternativ. Lekvall och Wahlbin (1993) menar också att det vid ett kvalitativt angreppssätt exempelvis kan vara svårt att få tag på rätt person att intervjua, en person som bör representera en större ”massa” och som är kunnig inom ämnet.

Vi har valt att använda oss av ett kvalitativt angreppssätt. Detta då syftet med uppsatsen är att få en djupare förståelse för tillämpningen av UML och inte att redogöra för nyttjandet av UML i kvantitativa värden. Valet baseras även på möjligheten till mer uttömmande svar och därmed tillgång till den information som är nödvändig för studien. Vi anser att den kvalitativa undersökningen kommer att ge oss en bättre uppfattning av företagen samt hur de tillämpar UML i sitt dagliga arbete vilket vi inte skulle ha kunnat få vid en kvantitativ undersökning.

2.2 Insamling av data

Det finns flera olika tillvägagångssätt vid insamling av data, speciellt då det handlar om primärdata. Primärdata är enligt Eriksson och Wiedersheim-Paul (1991) data som samlats in på egen hand för den aktuella studien. Vi har i vårt arbete valt att hämta en stor del av den information vi behöver från primärdata.

De metoder vi anser vara relevanta för insamling av primärdata, inom ramen för det kvalitativa angreppssättet, är frågor i form av intervju eller enkät. Enkätundersökningar används, enligt Trost (2001), vanligtvis i samband med kvantitativa studier. Dock kan det i en kvalitativ studie användas enkäter med öppna frågor. Vid använd-

ning av enkätformulär i allmänhet, och inom detta arbete i synnerhet, skulle konsekvensen bli att frågorna måste formuleras mycket noggrant så att inga missförstånd uppstår. Det skulle inte finnas samma möjlighet att förtydliga frågor, ställa följdfrågor eller att få ett mer utvecklat svar, vilket skulle vara möjligt vid en intervju. Ett annat problem Trost (2001) nämner är att det kan vara svårt för respondenten att veta hur omfattande svar som efterfrågas. En enkät ger dessutom respondenten tillfälle att utlämna komplicerade frågor som i ett intervjusammanhang skulle kunna förtydligas omgående. I vissa fall kan det även vara lättare att förklara något muntligt än i skrift. Det positiva med att använda sig av enkätformulär är enligt Trost (2001) att respondenten själv kan bestämma när frågorna ska besvaras. Alla frågor behöver inte heller besvaras vid samma tillfälle. En fördel med intervju är att den ger en större möjlighet till att illustrera vissa svar samt att det underlättar för den som intervjuar att ta till sig informationen. En svaghet som Trost (2001) nämner med intervju är risken att respondenten blir påverkad av den som intervjuar då han eller hon kan ställa ledande frågor.

Av nämnda alternativ har vi enats om att intervju är ett bättre tillvägagångssätt än en enkät i detta arbete. Orsaken är att de frågor vi ämnar ställa till företagen kan resultera i komplicerade svar där både diskussion och följdfrågor kan vara nödvändiga för ett tillfredsställande svar.

För att försäkra sig om att rätt frågor ställs under intervjun är det enligt Christenssen, Andersson, Carlsson och Haglund (1998) viktigt att gå igenom frågeställningarna och gärna göra en pilotundersökning innan intervjutillfället. En pilotundersökning utförs för att ta reda på om den valda metoden fungerar och ger relevanta svar. En pilotundersökning kommer dock inte vara möjligt för vårt arbete då undersökningen är relativt omfattande samt att vi anser det svårt att få ett företag med rätt kompetens att ställa upp på en undersökning som inte kommer att innefattas i arbetet. Vi tror inte att en utebliven pilotundersökning kommer att påverka resultatet i någon betydande utsträckning.

Vi har även genomfört litteraturstudier för att samla in den information som utgör referensramen. Referensramen till den här studien fungerar som ett stöd för att kunna sätta resultatet i ett perspektiv samt öka förståelsen för den information som redovisas i resultatet. Referensramen kommer även att användas för vidare analys. För insamling av information till referensramen har vi främst använt oss av litteratur i form av böcker och tidskrifter samt information från Internet. Vid insamling av information från Internet är det viktigt att försäkra sig om att informationen kommer från en pålitlig källa. De Internetkällor vi använt oss av utgör respektive företags hemsidor samt officiella hemsidor för olika organisationer och kan således anses vara pålitliga.

2.3 Val av intervjutyp

När det gäller intervjuer finns det enligt Lundahl och Skärvad (1999) tre olika typer. Den första är standardiserad intervju, där man redan innan intervjun bestämt frågor och ordningsföljd på frågorna. Denna typ av intervju anser Lundahl och Skärvad (1999) vara passande i kvantitativa studier då den är lämplig för insamling av exempelvis försäljningsvolymer och annan statistik. Svensson och Starrin (1996) instämmer

och tillägger att den standardiserade intervjun oftast förekommer i kvantitativa studier då frågor och svarsalternativ ska vara samma för alla respondenter.

Den andra typen av intervju är ostandardiserad intervju. Lundahl och Skärvad (1999) hävdar att vid denna typ av intervju är frågor och ordningsföljd obestämd innan intervjun startar. Svensson och Starrin (1996) menar att den ostandardiserade typen av intervju används när man i förväg inte vet vilka frågor som är betydelsefulla. Den här typen av intervju kräver att intervjuaren är uppmärksam och vet vilka frågor och följdfrågor som är lämpliga för att täcka informationsbehovet. Lundahl och Skärvad (1999) framför också att denna typ av intervju ofta används vid kvalitativa undersökningar då den lämpar sig för att samla in data om personers bedömning eller åsikt om en viss situation.

Lundahl och Skärvad (1999) tar upp en tredje typ som benämns semistandardiserad intervju. Här förbereds intervjun med ett antal fördefinierade frågor som ställs till samtliga respondenter. Dessa frågor följs sedan upp med lämpliga följdfrågor under intervjun.

För att intervjuerna ska fortlöpa på ett bra sätt och för att vi ska kunna styra respondenten att svara inom de områden vi är intresserade av anser vi att vissa förutbestämda frågor är nödvändiga. Beroende på respondentens svar kommer vi även att behöva ställa relevanta följdfrågor för att få den information vi behöver. En del svar kan kräva en mer utförligare förklaring och andra kan behöva följdfrågor såsom ”Vad är anledningen till detta?”. Med hänsyn till ovannämnda faktorer anser vi att den semistandardiserade typen av intervju är bäst lämpad för vår studie.

2.4 Val av respondenter

Vi har valt att utföra den empiriska studien på fyra företag i Jönköpingsregionen som använder sig av UML i någon utsträckning. De företag som använts i studien har valts slumpmässigt från en lista med åtta företag i Jönköpingsregionen. Denna information har vi fått tillgång till genom Internationella handelshögskolan i Jönköping. Att Jönköpingsregionen har valts beror på att vi lättare ska kunna komma i kontakt med företagen. Det kan även vara praktiskt om det blir aktuellt med eventuell uppföljning av första intervjutillfället. Företagen i fråga är tre konsultföretag och ett företag som enbart arbetar med systemutveckling inom den egna organisationen.

Vi anser att fyra företag kan vara ett lämpligt antal då en kvalitativ undersökning i intervjuform kräver att mycket tid läggs ned på vart och ett av företagen. Vi anser att möjligheten att få en mer trovärdig bild ökar med antalet företag som intervjuas. Om färre än fyra företag hade valts hade risken att gå miste om värdefull information samt att få en felaktig bild av UML varit större. Vi hade inte heller haft samma möjlighet att jämföra de svar som fåtts mellan företagen. Vi anser därför att antalet företag som valts för intervju kommer att ge oss möjligheten att uppfylla syftet med arbetet. Vi är dock medvetna om att även fyra företag kan ge en missvisande syn på UML men att risken ändå är mindre. Då syftet med denna kvalitativa studie inte är att generalisera svaret behöver vi inte heller ta hänsyn till det vid val av antal företag (Lundahl & Skärvad, 1999).

2.5 Förberedelse och tillvägagångssätt vid intervjuerna

För att komma till stånd med intervjuerna kontaktade vi mycket snart efter starten av uppsatsen företagen via telefon för att ta reda på vilka personer som skulle vara lämpliga att svara på frågorna. Dessa personer kontaktades sedan via e-post där vi berättade om studien och frågade om de var villiga att ställa upp på en intervju. Vi meddelade även önskvärd tidpunkt för intervjun samt att vi skulle återkomma per telefon efter någon vecka för att bekräfta tidpunkten. De problem som vi stötte på var att den kontaktperson som uppgetts vid första telefonsamtalet i vissa fall inte var den bäst lämpade. Det e-postmeddelande som skickats ut fick därför vidarebefordras till flera personer innan ett slutgiltigt svar kunde erhållas.

De uppföljande telefonsamtalen resulterade i fyra företag som var villiga att ställa upp på en intervju. Redan vid detta tillfälle passade vi på att boka ett datum då intervjun skulle äga rum för att få tillgång till den tid vi behövde på företagen. Alla företags kontaktpersoner var mycket tillmötesgående och intresserade av att diskutera ämnesområdet med oss. Ett problem var att det i vissa fall visade sig svårt att få kontakt med respektive kontaktperson per telefon vilket resulterade i ett antal telefonsamtal utan resultat.

För att vara förberedda till intervjuerna arbetade vi fram ett frågeunderlag (se Bilaga 2) där vi tog upp det vi ansåg kunde vara av intresse för studien. Detta frågeunderlag skickades sedan ut till respektive kontaktperson några dagar innan intervjutillfället för att även de skulle kunna förbereda sig på bästa sätt. Vi ansåg att detta skulle göra att intervjun fortlöpte på ett bra sätt så att vi kunde få ut mesta möjliga av varje intervjutillfälle.

Alla intervjutillfällen spelades in då vi ansåg att endast anteckningar inte skulle vara tillräckligt. Detta medförde att vi under intervjuerna kunde koncentrera oss på att ställa rätt frågor för att få fram väsentlig information. Inspelningen kompletterades med anteckningar och olika grafiska illustrationer som demonstrerades av respondenten på whiteboardtavla eller liknande. Efter intervjuerna transkriberade vi all den information vi fått för att därefter formulera resultatet (se kapitel 4).

2.6 Uppsatsens trovärdighet

Att utföra en kvalitativ studie innebär enligt Patel och Tebelius (1987) ett annat tankesätt för forskarna än vad en kvantitativ studie gör. I en kvalitativ studie kretsar arbetet runt forskarnas sätt att arbeta, det vill säga hur de samlar in och tolkar informationen. Då kvalitativa studier inte kan mätas eller bedömas utifrån siffror är det upp till forskarna att kritiskt granska den studie de gör. Detta kan göras genom att ställa frågor under arbetets gång, frågor som rör bland annat arbetets rimlighet och pålitlighet. Vi har valt att diskutera vår studie i samband med begreppen tillämplighet, rimlighet, pålitlighet samt noggrannhet. Detta eftersom dessa begrepp, enligt Patel och Tebelius (1987), är bättre anpassade till kvalitativa studier än vad begreppen reliabilitet, validitet samt generaliserbarhet är.

2.6.1 Tillämplighet

Då det som studeras i en kvalitativ studie kan ses ur olika synvinklar beroende på person och situation är det enligt Patel och Tebelius (1987) viktigt att tillvägagångssätt väljs omsorgsfullt så att tillämpligheten blir god. Även val av tid och plats för insamling av information samt respondenter är viktigt eftersom människor tolkar verkligheten annorlunda. Respondenterna bör därmed väljas utifrån exempelvis vilken position de har på företaget. Vi anser att den intervjuteknik vi använt oss av och de människor vi intervjuat har ökat tillämpligheten för studien. Detta eftersom intervjuerna ägde rum ute på företagen där respondenterna kände sig hemma samt att personerna som intervjuats har deltagit i eller lett utvecklingen av UML på företagen.

2.6.2 Rimlighet

Då vi har valt en kvalitativ inriktning på vår studie behöver vi utgå från andra kriterier för att kunna fastställa rimligheten i arbetet till skillnad mot tillvägagångssättet vid en kvantitativ studie. För att studien ska ha hög rimlighet bör forskarna enligt Patel och Tebelius (1987) under arbetets gång ställa sig frågor såsom om den information som samlats in är trovärdig. De bör även kunna visa i studien att den kvalitativa informationen har sitt ursprung ur ett rikhaltigt material.

Det är enligt Patel och Tebelius (1987) viktigt att valet av teknik vid insamlandet av information ger möjligheten till ett relevant resultat. Detta för att rimligheten i arbetet ska bli så hög som möjligt. Den teknik vi valt att använda oss av är intervjuer då vi anser detta vara den teknik som är lämpligast för denna studie och vilken ger oss möjlighet att samla in den information som behövs för att uppfylla syftet. För att undersökningens resultat inte ska hamna utanför ramen för syftet är det viktigt att rätt frågor ställs under intervjuerna (Lundahl och Skärvad, 1999). Vi har därför noga tänkt över vilka frågor som ska ställas vid intervjuerna för att de ska hålla sig inom ramen för studien. Även Patel och Tebelius (1987) påpekar hur viktigt det är att den information som används verkligen berör det som undersöks.

För att respondenterna skulle kunna generera bra och genomtänkta svar skickade vi ut frågeunderlagen några dagar innan intervjutillfällena. Vid intervjutillfällena visade det sig att alla respondenter var väl insatta i det område studien omfattar. Vi har även haft möjlighet att höra av oss om förklaringar behövts eller om ytterligare frågor dykt upp under arbetets gång. Detta i samband med det att respondenterna har läst igenom det färdiga resultatet höjer enligt Patel och Tebelius (1987) rimligheten på arbetet.

2.6.3 Pålitlighet

För att kunna visa på en pålitlighet i en studie måste forskarna enligt Patel och Tebelius (1987) kunna visa ett de tolkningar som gjorts inte grundats på förutfattade meningar eller stereotypa uppfattningar. Vi anser att det sätt informationen i studien samlats in (se kapitel 2) kan bidra till en pålitlighet i resultatet. Patel och Tebelius (1987) menar att det är genom att visa hur insamlingen av data samt tolkningarna utförts som pålitlighet går att påvisas.

En viktig aspekt enligt Patel och Tebelius (1987) är att den eller de som utför studien är öppna för den information som ges av respondenterna men de bör även kunna upptäcka och tolka handlingar som görs men som inte sägs rakt ut. Patel och Tebelius (1987) menar dock att det krävs vaksamhet från den eller de som utför studien så att inte egna reflektioner kryper in bland respondentens. För att undvika detta har vi, som Patel och Tebelius (1987) rekommenderar för att öka både rimlighet och pålitlighet, låtit respondenterna läsa igenom och godkänna det färdiga resultatet av intervjuerna.

En annan viktig aspekt enligt Patel och Tebelius (1987) för att få en så hög tillämplighet men även pålitlighet i arbetet som möjligt är att välja rätt person som uppgiftslämnare. Respondenten bör vara insatt i ämnet och motiverad till att diskutera ämnet i fråga. Personerna som vi intervjuat har enligt oss varit väl insatta i ämnet och då många av dem, som tidigare nämnts, bland annat varit med vid införandet av UML på arbetsplatsen även varit rätt personer att prata med. Vi anser att alla respondenter under intervjutillfället var mycket tillmötesgående och var villiga att svara på alla frågor. Vid en kvalitativ studie bör insamlingstillfället noga studeras för att se om respondenten blir störd av något eller någon och att därmed pålitligheten i svaren minskar (Patel & Tebelius 1987). Respondenterna hade avsatt tid för intervjun och blev således inte störda av andra omständigheter. Att alla intervjuer genomfördes i likartade miljöer utan större störningsmoment har enligt oss bidragit till en pålitlighet i arbetet.

2.6.4 Noggrannhet

Forskarnas noggrannhet vid en kvalitativ studie är enligt Patel och Tebelius (1987) viktig då, som tidigare nämnts, arbetet kretsar kring forskarnas tolkningar och kunskaper. Det är bland annat viktigt att forskarna är uppmärksamma vid intervjutillfället och inte påverkar eller pressar fram svar av respondenterna. Vi tror att vi till viss del reducerat den risken genom att gå igenom de frågor vi hade innan intervjutillfället så noggrant som möjligt samt att vi spelade in intervjuerna och kunde därmed efteråt lyssna igenom intervjuerna för att upptäcka eventuella påverkningar. Patel och Tebelius (1987) nämner även att information som senare upptäcks inte passa in i studiens analys inte skall tas bort då denna information ändå kan bidra till arbetets noggrannhet. Detta är något som tagits hänsyn till och diskuterats i aktuella fall.

3 Referensram

Kapitlet inleds med en beskrivning av vad systemutveckling innebär för att läsaren ska få en tydligare bild av i vilket sammanhang modelleringsspråket UML används. Vi fortsätter med att förklara objektorientering och dess betydelse i systemutvecklingens mognande samt skapandet av modelleringsspråket UML. Inom ramen för systemutveckling tar vi upp livscykelmodellen samt RUP (Rational Unified Process), vilka är två vanliga utvecklingsmodeller. Både livscykelmodellen och RUP nyttjar vi sedan för att, i slutet av kapitlet, klargöra i vilken del av systemutveckling modelleringsspråket UML förekommer. Vidare kommer en kort beskrivning av UML att göras och då målet är att bland annat ta reda på hur företag använder modelleringsspråket kommer vi att förmedla vilka de olika diagrammen i UML är. Detta för att göra det möjligt att förstå vår analys och vårt resultat.*

3.1 Systemutveckling

Många organisationer använder sig idag av något sorts informationssystem som stöd till deras verksamhet (Bahrami, 1999). Andersen (1994) definierar ett informationssystem som ett system för bearbetning, insamling, lagring, överföring samt presentation av information. Vidare diskuterar han människans del i ett informationssystem och menar att människor som behandlar information kan vara en del av informationssystemet.

Ett informationssystem behöver i de flesta fall skapas specifikt för varje organisation och det är enligt Andersen (1994) arbetet med att utveckla ett informationssystem som benämns systemutveckling. Bahrami (1999) är av liknande uppfattning men utvecklar begreppet och menar att alla aktiviteter* som utförs i samband med utvecklingen av ett informationssystem sammanfaller under begreppet systemutveckling.

Behovet av att skapa eller förbättra informationssystem är dynamiskt och är under ständig förändring (Bahrami, 1999). Idag är systemutveckling ett behov som både stora och små företag har. Det finns många olika tillvägagångssätt vid utveckling av informationssystem. Det finns till exempel ett flertal olika metoder, utvecklingsmodeller och verktyg att använda sig av beroende på företagens struktur, storlek samt informationssystemet som ska utvecklas.

De olika tillvägagångssätten inom systemutveckling är till för att skapa ett underlag för bedömningen av lösningsförslag, det vill säga hur det nya informationssystemet kommer att se ut (Apelkrans & Åbom, 2001). Apelkrans och Åbom (2001) anser att en systemutredning bör tillämpas för att få en bättre förståelse för hur informationssystemet ska användas inom organisationen samt vilka behov och krav användarna ställer på systemet.

Vid systemutveckling i en organisation är första steget oftast att kartlägga eller skapa en analysmodell av den del som ska utredas i verksamheten. Anledning till att analysmodeller används vid systemutveckling är att de ger en möjlighet att avbilda en komplex verklighet. Att kartlägga verksamheten underlättar, enligt Apelkrans och Åbom (2001), ofta kommunikationen mellan parterna användare och utvecklare samt

ökar förståelsen för verksamheten och de förändringar som bör göras. Att kartlägga ett verksamhetsområde brukar benämnas systemering eller modellering. Resultatet av en systemutveckling bör enligt Apelkrans och Åbom (2001) vara ett informationssystem som skall fungera som ett verktyg för den övriga verksamheten inom organisationen. De menar även att detta tankesätt måste anammas genom hela utvecklingen av det nya informationssystemet.

3.1.1 Livscykelmodellen

Vid systemutveckling används, som tidigare nämnts, ofta en utvecklingsmodell som hjälp för att styra utvecklingsarbetet. Vi kommer nedan att beskriva livscykelmodellen då den är en vanlig utvecklingsmodell och ett synsätt inom systemutvecklingen som har kommit att användas i stor utsträckning av systemutvecklare (Andersen, 1994). Livscykelmodellen har kommit att passa olika typer av projekt inom informationssystemutveckling men är enligt Andersen (1994) speciellt användbar inom företag som vill skapa ett informationssystem för en verksamhet som är stabil och bekant. Livscykelmodellen passar även bra då företag vill automatisera manuella rutiner. En av livscykelmodellens styrkor är att den inkluderar hela informationssystemets livscykel, från analys till förvaltning och därefter avveckling (se Figur 3.1). Livscykelmodellen består av sju olika steg eller faser där varje fas mynnar ut i ett resultat som sedan behandlas vidare under nästa steg i cykeln.

FA Förändring sanalys	A Analys	U Utformning	R Realisering	I Implementering	F/D Förvaltning /Drift	Av Avveckling

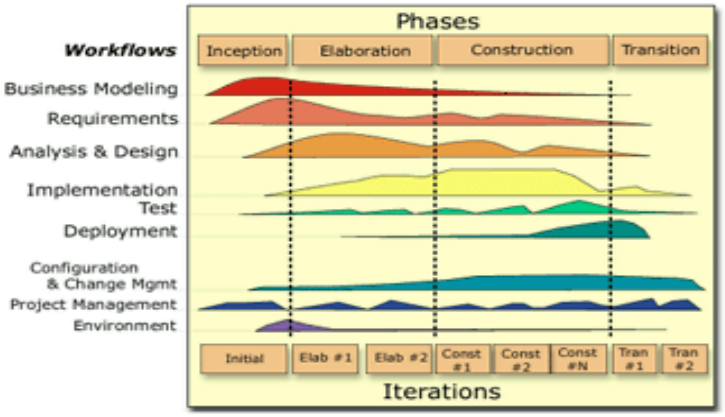
Figur 3.1 Livscykelmodellen (Andersen., 1994, s. 48)

Förändringsanalysen som är den första fasen i livscykelmodellen är mycket viktig för det fortsatta arbetet med utvecklingen av ett informationssystem. I denna fas skall företagets förändringsbehov belysas och därefter läggs det upp en plan med åtgärder (Andersen, 1994). Nästa steg, analys, är att se över vilka funktioner systemet skall ha och vad det ska innehålla. Modelleringsspråket* UML används främst under tre av livscykelmodellens faser: analys, design samt realisering. Detta kommer att beskrivas mer detaljerat i kapitel 3.4. Resultatet av analysfasen blir en kravspecifikation som sedan behandlas i utformningsfasen och omsätts till ett realiserbart informationssystem. Under realiseringsfasen ska systemet och de manuella rutinerna utarbetas och sedan realiseras så att resultatet blir ett färdigt informationssystem som under implementationsfasen skall införas på företaget. De två sista faserna är förvaltning och drift samt avveckling. Att sköta och underhålla ett system kan göra stor skillnad på informationssystemets livslängd.

3.1.2 RUP – Rational Unified Process

RUP är en annan vanlig systemutvecklingsmodell vi valt att beskriva då Kruchten (2000) poängterar att RUP är utvecklad för att passa tillsammans med UML. Han menar även att RUP är en mångsidig utvecklingsmodell som lämpar sig för många typer av projekt, vilket gör den till en populär systemutvecklingsmodell. Zanoni och

Audy (2004) menar att RUP är ett speciellt bra val vid objektorienterad systemutveckling medan Scott (2002) poängterar att den används mycket i kombination med UML då många av de anvisningar som anges i RUP involverar användningen av UML-baserade diagram. Enligt Scott (2002) är UML:s användningsfallsdiagram (se avsnitt 3.6.1) en av grundstenarna i RUP och hela utvecklingsprocessen utgår från dessa. En annan viktig grundsten är enligt Scott (2002) att RUP arbetar efter en iterativ och inkrementell metod. Han menar att systemet ska förbättras inkrementellt, det vill säga stegvis, för varje iteration.



.

Figur 3.2 Rational Unified Process (Kruchten, 2000, s. 23)

Det som skiljer RUP från livscykelmodellen är att RUP är en iterativ systemutvecklingsmodell. Det betyder att i varje fas förekommer ett antal iterationer och till exempel krav behöver inte vara helt definierade efter första iterationen. Det här medför enligt Scott (2002) att systemutvecklingsarbetet blir mer flexibelt och öppet för förändringar. Mellan varje fas finns dock en milstolpe definierad. Scott (2002) klargör vidare att varje fas i RUP innefattar ett antal iterationer vilka alla avslutas med att systemet testas för att bestämma om systemet är redo för att gå in i nästa fas. Den första fasen är ”inledning” i vilken innehåll, arkitektur och risker definieras. Även diskussioner om projektets genomförbarhet utförs här. Den andra är ”utveckling” i vilken funktioner väljs ut samt arkitekturen utvecklas. Även risker och projektplan diskuteras. Under tredje fasen, ”konstruktion”, konstrueras en prototyp som kan testas i kundorienterade miljöer. Den fjärde och sista fasen är ”övergång” i vilken det slutliga målet är att lämna över ett färdigt system till kunden.

3.2 Modellering

Booch (1998) nämner ett antal orsaker till varför det är så viktigt att modellera. Han menar att modeller byggs för att bättre förstå det system som skapas och att en modell är en förenkling av verkligheten. Vidare nämner han fyra saker som uppnås med hjälp av modellering varav den första är att modellering hjälper till att visualisera system. Övriga anledningar till att modellera är att modelleringen tillåter oss att specificera systemets struktur och beteende, vägleder oss i konstruktionen av system samt dokumenterar de beslut som tagits under arbetets gång. Han hävdar även att god mo-

dellering är en gemensam faktor i lyckade systemutvecklingsprojekt medan det i misslyckade projekt är svårt att finna någon generell orsak.

Zanoni och Audy (2004) poängterar modellering genom att framhäva att UML är ett bra modelleringsspråk* som fokuserar på att specificera och dokumentera systemkrav för projektet samt underlättar kommunikationen mellan klienter och projektmedlemmar. Även Beckworth (2001), som talar om utveckling av realtidssystem, menar att UML gör att kraven blir lättare att förstå.

Detta för med sig att om UML används korrekt genererar det dokumentation som underlättar kommunikationen med och mellan klienter och projektmedlemmar. Zanoni och Audy (2004) anser det viktigt att standardisera kommunikationen gällande kravspecifikationen och därför rekommenderar de UML för detta och även genom resterande delar av utvecklingsprocessen.

3.3 Objektorienterad utvecklingsansats

Inom systemutveckling finns det olika synsätt att arbeta efter och idag är det främst det objektorienterade synsättet som tillämpas. Innan den objektorienterade ansatsen utvecklades tillämpades något som kallas funktionsorienterad systemutveckling och denna ansats används än i dag i vissa sammanhang. Det funktionsorienterade synsättet utgår enligt Andersen (1994) och Bahrami (1999) från de funktioner som verksamheten ska utföra och fokuserar inte på verksamhetens objekt*. Funktioner i verksamheten kan till exempel vara produktion och marknadsföring. Det är dessa funktioner som bestämmer verksamhetens uppgifter och därmed vilket informationsbehov som informationssystemet skall täcka.

Objektorientering (OO) är ett område som utvecklats mycket snabbt då Eriksson et al. (2004) menar att intresset för OO inte fångades förrän i slutet av 80-talet. Grunderna för OO lades däremot redan i mitten av 70-talet men ett liknande synsätt har enligt Andersen (1994) även använts inom tekniska system ända sedan 1960-talet. Objektorientering var till en början inriktad på programmering och tog form i det än idag mycket välkända programmeringsspråken C++ och Smalltalk (Eriksson et al, 2004). Idag kan den objektorienterade ansatsen användas vid alla typer av system.

Anledningarna till att det objektorienterade synsättet har slagit igenom de senaste åren är enligt Bahrami (1999) och Apelkrans och Åbom (2001) många. Båda argumenterar för att detta synsätt kan hantera komplexare system tack vare det grafiska tillvägagångssättet som skapats. Detta är viktigt då dagens system blir större och ofta mer komplexa. Tack vare att grafiska lösningar används kan verkligheten avbildas på ett mer naturtroget sätt vilket i sin tur leder till att de som inte är insatta i systemutveckling har en större möjlighet att förstå vad det innebär. Det objektorienterade sättet att arbeta skapar även förutsättningar för att lättare kunna förändra, göra tillägg eller att återanvända hela system eller delar av system.

3.4 UML:s framväxt

När de objektorienterade programmeringsspråken slog igenom skapades genast ett behov av objektorienterade systemutvecklingsmodeller och metoder och en blomstring av metoder för objektorienterad systemutveckling tog fart. År 1994 fanns det enligt Eriksson et al. (2004) redan ett 50-tal olika metoder och modelleringsspråk* som stöd för objektorienterad systemutveckling. Allt eftersom marknaden mognade skapades nya modelleringsspråk som grundade sig på de bästa komponenterna från de tidigare modelleringsspråken. Eriksson et al. (2004) nämner de mest framgångsrika metoderna som OOSE (Object Oriented Software Engineering), OMT-2 (Object Modeling Technique) och Booch´93. Det är dessa tre metoder som utgör grunden för UML.

Eriksson et al. (2004) anser vidare att det stora utbudet av modelleringsverktyg skapade problem vid start av olika projekt. Det krävdes ofta noggranna och tidskrävande utredningar för att bestämma vilken av de olika metoderna som bäst lämpade sig för ett specifikt projekt. Zanoni och Audy (2004) instämmer med detta problem och menar att svårigheten med systemutveckling ligger just i förmågan att välja, anamma och integrera rätt systemutvecklingsmodell och metod så att de passar omgivningen. Att många olika utvecklingsmodeller och metoder användes resulterade enligt Eriksson et al. (2004) ofta i att systemen såg olika ut och var uppbyggda på olika sätt. Apelkrans och Åbom (2001) menar även att om informationssystemen ser olika ut kan det orsaka problem vid samarbete mellan företag. Det här kan vara en av anledningarna till att Zanoni och Audy (2004) poängterar behovet av mer standardiserade produkter för systemutveckling. Ovanstående problem är vad som gav en framstående grupp utvecklare idén att skapa det enhetliga modelleringsspråk som vi idag kallar UML (Eriksson et al. 2004).

Arbetet med UML startades egentligen 1994 av grundarna Booch och Rumbaugh vilka även är grundarna till metoderna Booch och OMT-2. Under 1995 kom en tredje person in i arbetet, nämligen skaparen till OOSE, Jacobson. De tre påbörjade ett gemensamt arbete med de tre tidigare metoderna som grund för ett nytt standardiserat modelleringsspråk. Enligt Eriksson och Penker (1998) kom Booch, Rumbaugh och Jacobson även att inspireras av andra metoder i sitt arbete med UML. Deras arbete resulterade i det första språket som hade en klar grammatik för grafisk beskrivning (Apelkrans & Åbom, 2001).

De mål som de tre grundarna satte upp för arbetet med UML var bland annat:

- *”To model systems (and not just software) using object-oriented concepts”*
- *“To establish an explicit coupling to conceptual as well as executable artefacts”*
- *“To address the issues of scale inherent in complex, mission-critical systems”*
- *“To create a modeling language usable by both humans and machines“*

Eriksson & Penker, 1998, s.5

Utvecklarna av UML har enligt Eriksson et al. (2004) sett till att det hela tiden funnits möjligheter att utveckla, förändra och kombinera UML med egenutvecklade delar. Det går till exempel att utesluta vissa delar i UML som inte anses relevanta för ett visst projekt. Detta är viktigt då vi lever i ett föränderligt samhälle där sättet att utveckla informationssystem hela tiden förändras och där det, enligt Bahrami (1999), aldrig kan sägas hur morgondagens utvecklingsmodeller och metoder kommer att se ut. Det har under årens lopp kommit ut flera versioner av UML där både små och stora förändringar gjorts och mycket snart släpper OMG den slutliga specifikationen av UML 2.0. I denna version har relativt stora förändringar gjorts (Object Management Group, 2004).

Zanoni och Audy (2004) nämner UML som ett bra val vid objektorienterad systemutveckling. Det bör dock nämnas att UML inte behöver begränsas till att bara användas vid objektorienterad systemutveckling utan kan även appliceras vid till exempel datamodellering (Rational Software Corporation, 2000).

3.5 OMG och Standardisering

Som nämnts tidigare pågick det ett metodkrig under 1990-talet som ledde fram till skapandet av UML. Skapandet av UML fick under 1996 ett stort stöd av företag såsom IBM, Microsoft, Hewlett-Packard, Texas Instruments och Rational. Anledningen till att de gav sitt stöd till UML var att alla dessa organisationer såg nyttan i att ha tillgång till ett modelleringsspråk* som UML, både för att underlätta systemutvecklingsarbetet samt att skapa enhetlighet (Eriksson et al., 2004). Enligt Eriksson et al. (2004) enades skaparna av UML om att modelleringsspråket inte skulle tillhöra ett specifikt företag vars mål kanske hade varit att tjäna pengar på UML. De ville istället att ägarna skulle se till användarnas bästa. Målet blev att få UML att bli upptaget av OMG, vilken är en icke vinstgivande organisation, som en OMG standard. Detta blev också fallet och UML tillhör idag OMG och är den officiella standarden för att beskriva arkitekturen vid utvecklandet av ett informationssystem (Hanscome, 2000).

OMG skapades 1989 och som nämnts ovan är de inte beroende av någon inkomst från deras produkter och kan därmed handla för ”kundernas bästa”. OMG bestod vid starten av 11 företag och har idag över 800 medlemmar. Några av de ursprungliga medlemmarna är; 3Com Corporation, American Airlines, Canon Inc, Sun Microsystems, Data General, Hewlett-Packard, Philips Telecommunication N.V and Unisys Corporation (Eriksson et al., 2004).

Modelleringsspråket UML är tillgängligt för alla utan några licenskostnader och kan laddas hem från bland annat OMG:s hemsida. Det är även tillåtet att skapa egna verktyg för arbetet med UML samt att modifiera delar eller använda UML tillsammans med andra metoder och analysmodeller (Eriksson et al., 2004)

3.6 UML:s uppbyggnad

UML består av ett antal diagramtyper vilka beskriver ett system eller dess delar ur olika synvinklar. Man brukar även säga att UML består av fem olika vyer utifrån vilka man kan beakta systemet. Vyerna kommer dock inte att förklaras mer ingående då

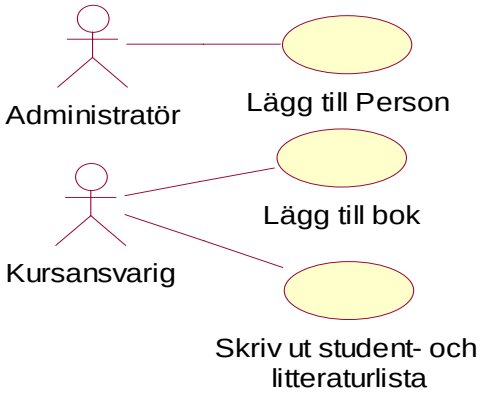
det inte har betydelse för studiens resultat. Det viktiga är att känna till att systemet kan betraktas utifrån olika synvinklar och abstraktionsnivåer. En typ av diagram kan användas för att illustrera systemet ur flera synvinklar, men det är upp till utvecklaren att bestämma vilka diagram som är användbara för projektet (Eriksson et al., 2004).

Variationen av vilka diagramtyper som används i projekt är stor. Det kan röra sig om ett par diagram till hela serien av UML-diagram plus ett antal egenutvecklade diagramtyper. Det förekommer även att diagram inom UML tas i anspråk och modifieras till önskad form (Eriksson et al., 2004). Zanoni och Audy (2004) menar att det inte bör ingå för många detaljer vid modelleringen då de oftast inte är nödvändiga. I följande avsnitt beskrivs de olika diagramtyperna i UML med dess syfte, användningsområde och syntax*. För att ytterligare förtydliga exemplifieras diagrammen med ett enkelt skolsystemexempel.

3.6.1 Användningsfallsdiagram

Oestereich (2002) beskriver ett användningsfall som en funktion i systemet som leder till ett tydligt resultat för en viss aktör. Ett resultat kan till exempel vara en registrerad bok i databasen vilket är ett resultat från det mellersta användningsfallet i Figur 3.3. Användningsfallsdiagrammet visar alla funktioner (användningsfall) som systemet förväntas hantera och kan således även fungera som en specifikation för kundens funktionella krav. I Figur 3.3 kan kundens funktionella krav utläsas som att systemet skall kunna lägga till nya personer, lägga till böcker samt att skriva ut student- och litteraturlista. Zanoni och Audy (2004) nämner en styrka med UML som just förmågan att förmedla krav på ett klart och tydligt sätt. I kombination med varje användningsfall skapas en textuell beskrivning som mer utförligt beskriver användningsfallets funktionalitet. Varje användningsfall har en relation till en aktör vilket i UML illustreras med en streckgubbe (se Figur 3.3). Enligt Fowler och Scott (2000) är aktören en person eller annan entitet som interagerar med systemet och varje användningsfall är knutet till en eller flera aktörer.

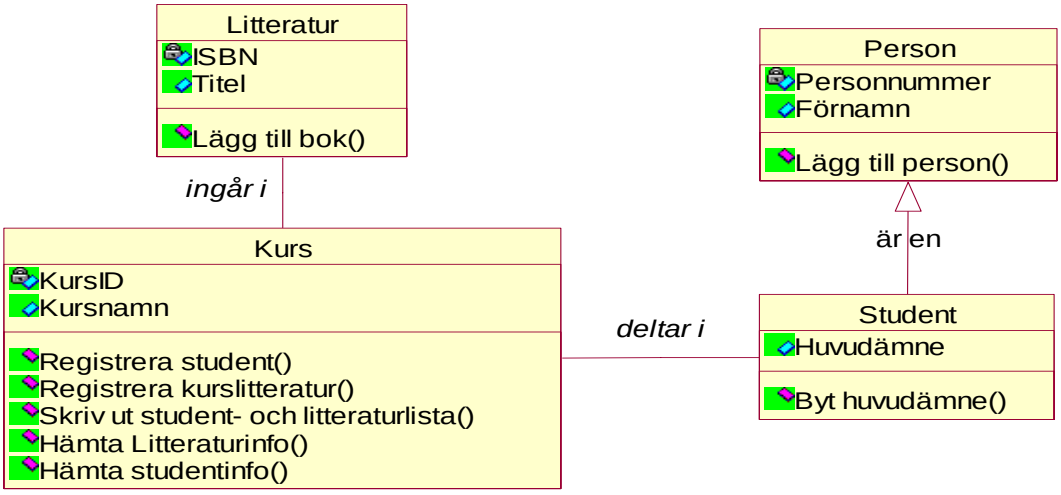
I Figur 3.3 finns två olika aktörer. Administratören är en aktör som ansvarar för funktionaliteten att lägga till nya personer i systemets databas. Den andra aktören i exemplet är kursansvarig som ansvarar för att registrera nya böcker samt för att skriva ut student- och litteraturlista. Resultatet av dessa användningsfall är en ny person i databasen, en registrerad bok på en kurs samt en utskriven lista på studenter och litteratur.



Figur 3.3 Användningsfallsdiagram ”Funktionella krav på skolsystemet”.

3.6.2 Klassdiagram

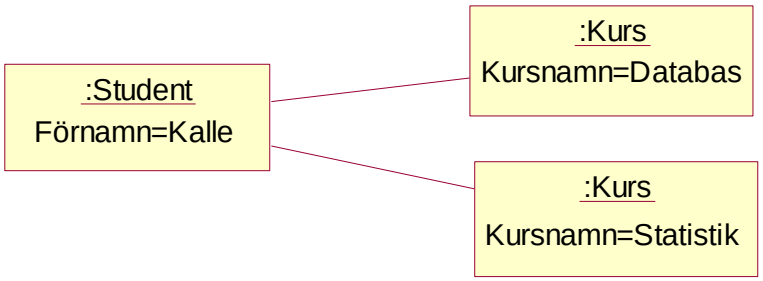
Enligt Eriksson et al. (2004) är ett klassdiagram en analysmodell som beskriver systemets statiska struktur. Med det menas klasser* och relationer mellan klasser samt klassens interna struktur i form av attribut och operationer. Fowler och Scott (2000) nämner två typer av relationer mellan klasser. Den första är associationer, vilket är ett vanligt beroende mellan klasser, den andra är subtyper, vilket är aggregat eller arv. I Figur 3.4 nedan illustreras fyra klasser tillsammans med dess attribut och operationer. Klassen Person har attributen ”Personnummer” och ”Förnamn” och operationen ”Lägg till Person”. Mellan klasserna Person och Student visas exempel på en arvsrelation som säger att en Student ”är en” Person och mellan de övriga klasserna förekommer vanliga associationer. Dessa associationer har i exemplet namngivits för att ytterligare klargöra relationen. Litteratur ”ingår i” en kurs och en student ”deltar i” en kurs. Klassdiagrammet är enligt Fowler och Scott (2000) samt Eriksson et al. (2004) det mest grundläggande diagrammet i UML.



Figur 3.4 Klassdiagram ”Skolsystem”.

3.6.3 Objektdiagram

Ett objektdiagram har enligt Fowler och Scott (2000) samma struktur som ett klassdiagram men beskriver relationer mellan objekt* (se Figur 3.5). Objektdiagram kan vara nödvändiga när relationerna mellan objekt upplevs komplicerade. I exemplet med skolsystemet kan det i objektdiagrammet klargöras att Kalle är registrerad på kurserna Statistik och Databas.

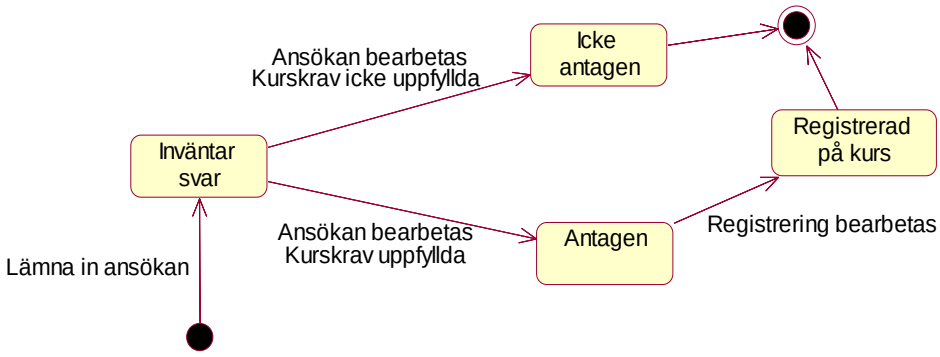


Figur 3.5 Objektdiagram ”Kalles kurser”.

3.6.4 Tillståndsdiagram

Tillståndsdiagram kan beskrivas som ett komplement till klassbeskrivningen. Här visas alla de tillstånd ett objekt* av en klass* kan inta under en viss händelse, hur ett objekt reagerar på en aktivitet* samt de aktioner som initierar en förändring av ett tillstånd (Eriksson et al., 2004).

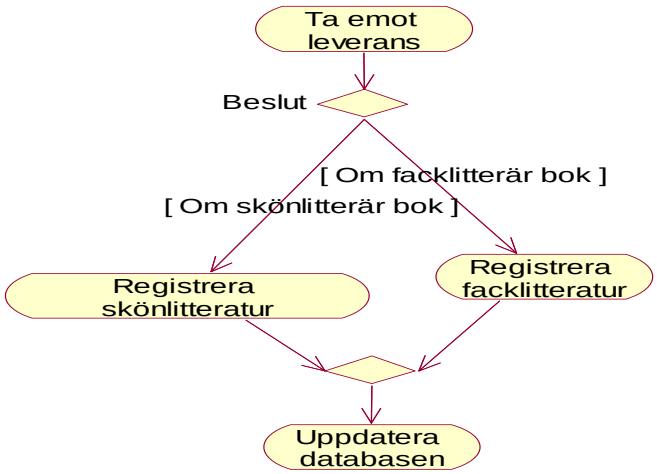
I Figur 3.6 kan vi se hur ett studentobjekt ändrar tillstånd vid ansökan till en kurs. Händelsen initieras av att studenten lämnar in en ansökan om antagning till kursen. Studentens tillstånd förändras till att invänta svar på ansökan. I nästa skede bearbetas ansökan och beroende på vilket villkor som faller in på studenten (uppfyllda/icke uppfyllda kurskrav) anhåller studenten ett svar och ändrar återigen tillstånd till att antingen vara antagen eller icke antagen till kursen. Om studenten inte är antagen avslutas tillståndsväxlingen och studenten förblir icke antagen. Om studenten däremot är antagen på kursen bearbetas registreringen och sedan skiftar studentens tillstånd från antagen till registrerad på kursen. Därefter avslutas studentens tillståndsväxling och studenten förblir registrerad på kursen. Fowler och Scott (2000) påstår vidare att tillståndsdiagram endast bör utföras för de klasser som är av intresse att studera närmare och behöver således inte utföras för alla klasser i systemet. De uppmuntrar även till att använda tillståndsdiagram i kombination med något annat diagram eftersom tillståndsdiagram inte inkluderar någon beskrivning av hur olika objekt interagerar.



Figur 3.6 Tillståndsdiagram ”Ansökan till kurs”.

3.6.5 Aktivitetsdiagram

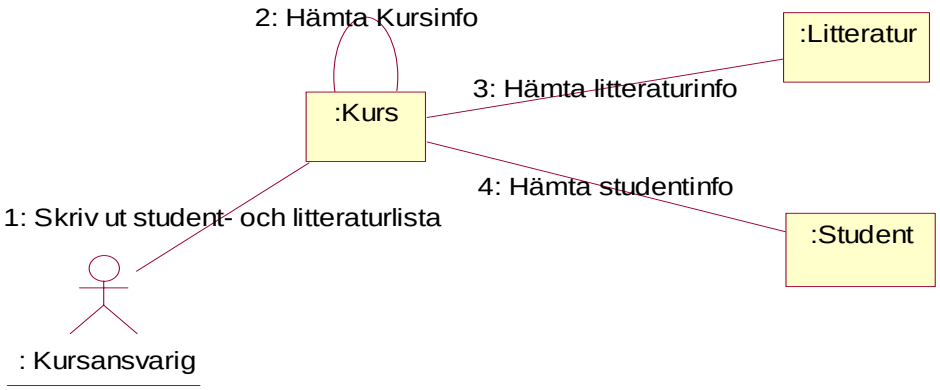
Enligt Oestereich (2002) är en aktivitet* ett av stegen i en process* och aktivitetsdiagrammet illustrerar en serie av aktiviteter (se Figur 3.7). Ett aktivitetsdiagram kan exempelvis beskriva alla aktiviteter som ingår i ett visst användningsfall. Eriksson et al. (2004) tillägger att aktivitetsdiagram kan åskådliggöra hur systemet reagerar på olika influenser från omgivningen samt hur meddelanden skickas i samband med olika aktiviteter. I aktivitetsdiagrammet finns även syntax* för villkor och parallella aktiviteter. Aktivitetsdiagram används när man anser att ett objekts* aktiviteter är viktigare än de tillstånd objektet hamnar i. Figur 3.7 beskriver händelseförloppet för att registrera en bok. Aktiviteten initieras av att en bokleverans anländer och tas emot. Därefter måste ett beslut komma om boken är av facklitterär eller skönlitterär art. Om beslutet resulterar i facklitterär bok registreras boken som facklitterär och sedan uppdateras databasen för att boken ska ingå i bokregistret. Samma förfarande sker naturligtvis om boken är skönlitterär, men då med skillnaden att boken registreras som skönlitterär istället för facklitterär.



Figur 3.7 Aktivitetsdiagram ”Registrera bok”.

3.6.6 Samarbetsdiagram

I ett samarbetsdiagram beskrivs enligt Oestereich (2002) samspelet mellan objekt* i ett visst sammanhang. Meddelanden illustreras i kronologisk ordning med hjälp av numrering. Fowler och Scott (2000) menar vidare att samarbetsdiagram inte lämpar sig för att illustrera en viss händelse men däremot för att studera hur objekt beter sig och samarbetar under händelsen. Figur 3.8 illustrerar hur objekt ur klasserna* Kurs, Litteratur och Student samarbetar vid utskrift av student- och litteraturlista. Händelsen inleds med att kursansvarig meddelar att denne vill skriva ut en student- och litteraturlista. För att skriva ut student- och litteraturlistan måste kursobjektet ha information om kurs, litteratur och student. Kursobjektet hämtar därför information om sig själv samt skickar meddelanden till litteratur- och studentklasserna om att erhålla information från dessa. De efterfrågade uppgifterna returneras till kursklassen vilken därefter har all nödvändig information för att skriva ut student- och litteraturlistan.

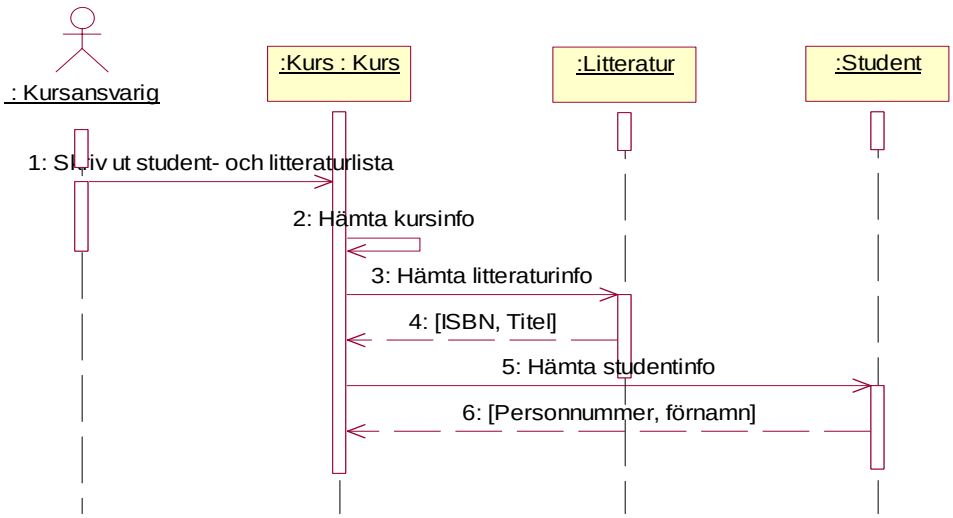


Figur 3.8 Samarbetsdiagram ”Skriv ut student- och litteraturlista”.

3.6.7 Sekvensdiagram

Ett sekvensdiagram liknar ett samarbetsdiagram men fokuserar mer på i vilken ordning aktiviteter* utförs än på meddelanden mellan objekten*, vilket är fallet i samarbetsdiagram (Fowler & Scott, 2000). I ett sekvensdiagram kan man således se i vilken ordning aktiviteter utförs för flera objekt som alla ingår i samma händelse (se Figur 3.9). Oestereich (2002) instämmer med att hävda att sekvensdiagram och samarbetsdiagram beskriver samma sak men ur olika synvinklar. I Figur 3.9 visas samma händelse som i samarbetsdiagrammet (Figur 3.8) men med tydligare skildring av i vilken ordning aktiviteterna utförs då de streckade axlarna nedanför varje klassnamn återger en tidsaxel. Exemplet i Figur 3.9 visar i vilken ordning aktiviteterna för objekten i Kurs, Litteratur och Student utförs för att uppfylla funktionaliteten att skriva ut student- och litteraturlista. Händelsen inleds med att kursansvarige meddelar kursklassen att den vill skriva ut en student- och litteraturlista. Kursklassen hämtar då först information om sig själv varefter ett meddelande med förfrågan om att erhålla vissa egenskaper skickas till litteraturklassen vilken returnerar sina uppgifter. Därefter skickas ett likadant meddelande till studentklassen som även den returnerar sina upp-

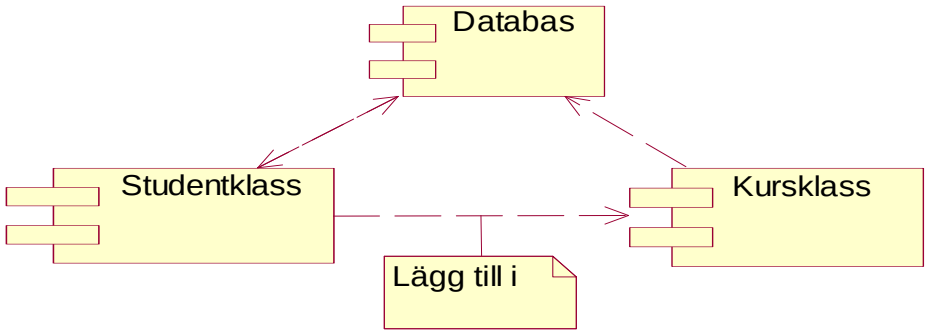
gifter till kursklassen. Efter detta är kursklassen redo att fullfölja sitt uppdrag att skriva ut student- och litteraturlistan.



Figur 3.9 Sekvensdiagram ”Skriv ut student- och litteraturlista”.

3.6.8 Komponentdiagram

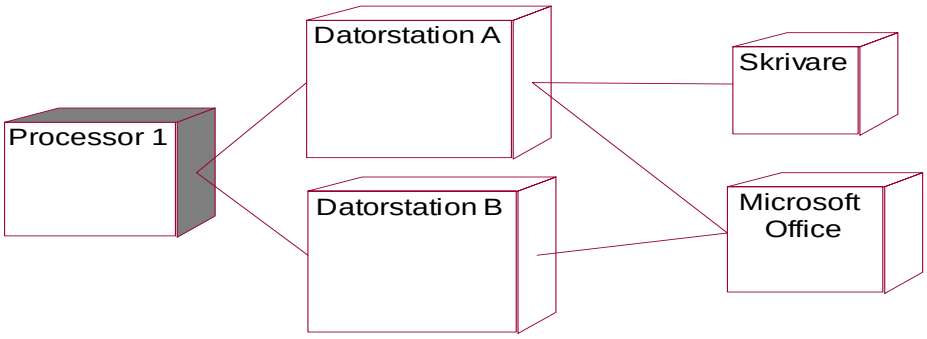
Komponentdiagram visar beroenden mellan komponenter i ett system (Fowler och Scott, 2000). En komponent innehåller kod för en viss del av systemet. Det kan vara en komponent innehållande kod för att registrera en student på en kurs. Beroenden mellan komponenter visar hur ändringar i en komponent påverkar andra komponenter och eventuellt medför ytterligare ändringar. Figur 3.10 visar ett exempel på vilka komponenter som är inblandade när en student registreras på en kurs. Studentklassen hämtar först uppgifter om den aktuella studenten från studenttabellen i databasen. När studenten sedan läggs till på kursen sparar kursklassen studentens uppgifter i kurstabellen i databasen (Fowler och Scott, 2000).



Figur 3.10 Komponentdiagram ”Registrera student”.

3.6.9 Fördelningsdiagram/Grupperingsdiagram

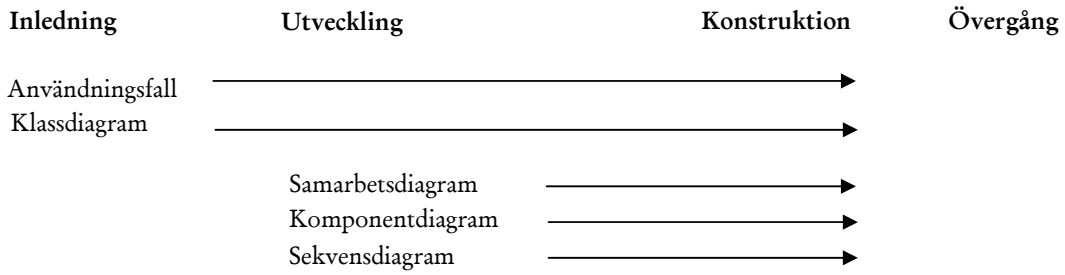
Ett fördelningsdiagram visar de fysiska relationerna mellan mjuk- och hårdvarukomponenter i ett system (Scott & Fowler, 2000). Oestereich (2002) menar att syntaxen* i fördelningsdiagram ofta byts ut mot symboler som föreställer den hård- eller mjukvara den representerar. Figur 3.11 visar att datorstation A och B är kopplade till samma processor. Vidare kan man se att både datorstation A och B innehåller programvaran Microsoft Office, men att bara datorstation A är kopplad till en skrivare.



Figur 3.11 Fördelningsdiagram "Delsystem "

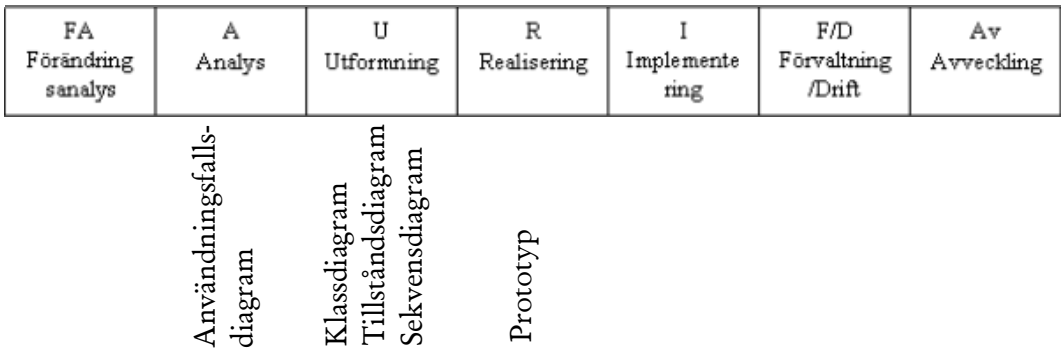
3.7 Diagrammens plats i RUP och livscykelmodellen

Som nämnts i avsnitt 3.1.2 används RUP mycket i kombination med UML och Zanoni och Audy (2004) anser att denna kombination är ett bra val vid objektorienterad systemutveckling. Faktum är att RUP enligt Kruchten (2000) är utvecklat för att kombineras med UML och talar därför om hur de olika komponenterna i UML kan användas på ett effektivt sätt vid modellering. Dessa direktiv gör det enklare för användaren att tillämpa UML genom hela utvecklingsprocessen, vilket är något som Zanoni och Audy (2004) poängterar som viktigt då de menar att flera osammanhängande delar lätt gör att kontrollen över projektet förloras. Scott (2002) förmedlar under vilken fas i RUP några av UML-diagrammen har mest fokus (Figur 3.12). Han menar att arbetet med användningsfall och klassdiagram startar redan i inledningsfasen. Under den första iterationen modelleras endast en grund för att komma igång med arbetet, men under följande iterationer utvecklas både användningsfall och klassdiagram kontinuerligt. Under utvecklingsfasen fortsätter arbetet med klassdiagram och användningsfall som även kompletteras med övriga diagram som anses viktiga för projektet. Scott (2002) nämner särskilt sekvensdiagram och komponentdiagram, men även övriga beteendediagram modelleras här. Kruchten (2000) nämner till exempel samarbetsdiagrammet i denna fas. Utvecklingen av dessa diagram fortsätter under ett flertal iterationer och även in i konstruktionsfasen där systemet konstrueras. Diagrammen används i denna fas men inget nytt diagram påbörjas. Inte heller övergångsfasen har specifikt fokus på något diagram eftersom systemet här är klart för att överlämnas till kunden.



Figur 3.12 UML i Rational Unified Process

Livscykelmodellen har inte samma starka koppling till UML som RUP, men eftersom även livscykelmodellen är en vanlig utvecklingsmodell som lämpar sig för många projekt kan UML kombineras bra även med denna utvecklingsmodell. Enligt Andersen (1994) ligger mest fokus på UML under tre av livscykelmodellens faser. Som Figur 3.13 visar är de tre aktuella faserna analys, utformning/design samt realisering. Några av de diagram som används i UML finns representerade i Figur 3.13 där det även påvisas under vilka faser diagrammen kan användas. Andersen (1994) menar att analysfasen resulterar i användningsfallsdiagram vilket illustrerar systemets krav. Användningsfallsdiagrammet står sedan till grund för utvecklingen av klassdiagram, sekvensdiagram och tillståndsdiagram vilka står i fokus under utformning-/designfasen. Även övriga UML-diagram skulle kunna modelleras här beroende på typ av projekt. Alla dessa diagram utgör sedan grunden för realiseringsfasen där projektteamet konstruerar en prototyp av systemet som sedan testas och utgör grunden för vidare beslut om systemutvecklingsarbetet.



Figur 3.13 UML i livscykelmodellen

4 Resultat av datainsamling

Insamlingen av data till detta kapitel har skett genom intervjuer på fyra olika företag i Jönköpingsregionen som använder sig av UML. Företagen är Jordbruksverket, Pdb, Saab Combitech Systems samt WM-data. I kapitlet kommer de svar vi fått redovisas företagsvis. Rubrikerna kommer dock att vara lika under samtliga företag för att underlätta läsningen.

4.1 Jordbruksverket

Jordbruksverket hanterar frågor angående jordbruk och livsmedel. De ansvarar för administrationen av EU:s jordbrukspolitik och arbetar för att förenkla EU:s regleringar inom jordbruket samt för att främja den regionala utvecklingen. Jordbruksverket har cirka 1000 anställda varav ungefär 590 arbetar på huvudkontoret i Jönköping (Jordbruksverket, 2004).

IT-avdelningen på Jordbruksverket är organiserad i tre enheter, en drift och supportenhet samt två systemutvecklingsenheter. Sammanlagt arbetar 55 personer inom IT-avdelningen varav 15 ingår i driftteamet som hanterar ett 100-tal servrar. Resterande är systemutvecklare men ofta arbetar ytterligare mellan 10 och 30 inhyrda konsulter på Jordbruksverket med att utveckla nya system samt att underhålla befintliga system. Jordbruksverkets IT-avdelning bedriver även utbildningar inom metodarbete samt systemutvecklingsområdet.

Jordbruksverket arbetar inte som konsulter åt utomstående kunder. Istället är det olika avdelningar inom Jordbruksverket som beställer ett projekt. Dessa avdelningar kan dock jämföras med utomstående kunder och ärenden behandlas på samma sätt som vid projektarbete mot en utomstående part.

Leif Bengtzohn arbetar som konsult inom systemutvecklingsenheten på Jordbruksverkets IT avdelning. Intervjun med Leif Bengtzohn ägde rum den 17 november 2004 hos Jordbruksverket. Han har arbetat på Jordbruksverket sedan 2001 och han är även en av dem som introducerat UML i företaget. Bengtzohn menar att Jordbruksverket tog till sig UML 2001 men att praktiskt använda UML som modelleringsspråk* är fortfarande under införande.

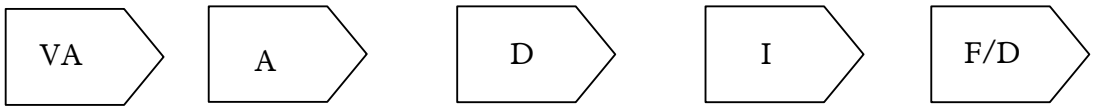
4.1.1 Systemutvecklingsmodell

På Jordbruksverket finns en egen systemutvecklingsmodell som ska användas vid all systemutveckling. Emellertid tillämpas modellen olika från projekt till projekt. Personalen har olika sätt att ta sig an system och de jobbar på olika sätt. Vissa projekt har därmed jobbat enligt RUP och andra enligt Jordbruksverkets egen systemutvecklingsmodell.

Den egna modellen innefattar nästan samma faser som livscykelmodellen (se avsnitt 3.1.1). Den första fasen i Jordbruksverkets systemutvecklingsmodell är verksamhetsanalys och sedan följer analys, design, implementation samt förvaltning och drift

(Figur 4.1). Skillnaden mot livscykelmodellen ligger i att Jordbruksverket har ersatt utformning och realisering med en designfas samt att de slopat avvecklingsfasen.

Bengtzohn menar att UML går att använda i alla systemutvecklingens faser, såväl i verksamhetsanalys som i design. UML är bara en notation och kan därmed användas för att modellera vad som helst. UML har dock vissa brister när det gäller de tidiga faserna, såsom verksamhetsanalys.



Figur 4.1 Systemutvecklingsmodell på Jordbruksverket

4.1.2 Modellering

Den främsta orsaken till att Jordbruksverket modellerar är för att säkerställa kvalitativa leveranser av system samt minimera fel som kan uppstå i systemen. Ju senare felen påträffas desto dyrare blir det att åtgärda. En annan orsak är den dokumentation som produceras, vilken kanske är onödig för utvecklingen, men viktig för senare förvaltningsarbete där någon utomstående kommer in och ska analysera systemet. Förvaltningsarbetet blir billigare om konsulten kan studera systemet i analysmodeller istället för kod då kodläsning tar lång tid och därmed kostar mycket pengar, enligt Bengtzohn. En bra dokumentation ger alltså bättre förvaltning, underhåll och felsökning.

Diagrammen ska enligt Bengtzohn underlätta kommunikationen mellan aktörerna i systemutvecklingskedjan. En analysmodell som inte är enklare att kommunicera än en text fyller ingen funktion då syftet med att producera analysmodeller är att underlätta kommunikationen. Diagrammen måste modelleras på en nivå som mottagaren förstår och det är även nödvändigt att utbilda denne så att han eller hon förstår syntaxen*. Utbildningsfrågan anses vara ett litet problem. Det svåra är för modelleraren att göra en analysmodell som är enkel och korrekt och på rätt nivå för användaren. Utvecklaren vill kanske till exempel ha med anrop och variabler medan en användare oftast vill ha en enklare beskrivning.

Bengtzohn tror på UML men hävdar att kunden oftast vill ha fungerande system så snabbt som möjligt. Kunden ser inte någon nytta med modellering och inte heller risken med förvaltningskostnaden som uppstår om systemet blir dåligt på grund av bristande modellering. Det är viktigt att sälja in vikten av förarbetet och poängtera att modellering inte är en merkostnad. Att få genomslag för denna åsikt är ett stort jobb.

Jordbruksverket har som alternativ att fortsätta som nu men tyvärr har flera brister i detta arbetssätt identifierats. Bengtzohn förklarar att han börjat arbeta med en ny modell då han anser att Jordbruksverkets befintliga utvecklingsmodell har brister, speciellt i designfasen. Idag börjar de med att modellera användningsfall och går sedan direkt över till datamodell* och kravspecifikation. De gör en design av systemet, men inga analysmodeller som beskriver hur systemet är uppbyggt, vilka komponenter det

består av och hur komponenterna kommunicerar med varandra. En av anledningarna till valet av UML är således att de behöver skapa mer analysmodeller för designarbete. Ett projekt startades för att analysera detta problem. Resultatet blev att ta tag i problemet redan i analysfasen då analysen är grunden för en bra design. Det är här Jordbruksverket är idag. De har, enligt Bengtzohn, bestämt sig för att införa UML ”sakta men säkert”. Bengtzohn poängterar även vikten av att Jordbruksverket lär sig sitt eget sätt att använda UML. Han menar att eftersom UML är relativt nytt så kan det vara svårt att ta till sig hur man får ut det bästa av modelleringen. Det är viktigt att ha en duktig mentor, men de är svåra att hitta i branschen, då många fortfarande är ovana vid att använda UML.

4.1.3 Styrkor och svagheter med UML

En styrka med UML är att det är ett regelverk och en första standard i branschen. Det behövs inga diskussioner om betydelsen av olika element eller analysmodeller eftersom tolkningen finns tydligt beskriven i UML. Det går att hålla en diskussion där alla förstår och kan delta. När någon väl lärt sig syntaxen* så blir det lättare att kvalitetsgranska dokument och analysmodeller och därmed fånga felen tidigt. En styrka enligt Jordbruksverket är att alla kommer arbeta efter samma modell. En annan styrka är att det finns verktyg som stödjer UML. Verktygen gör det även möjligt att generera kod vilket inte är aktuellt för Jordbruksverket idag men som det ser nyttan av och troligtvis kommer använda i framtiden då de känner sig mer säkra på UML. Ytterligare en styrka är möjligheten att skapa egna diagram i UML.

Den största svagheten med UML enligt Bengtzohn är att det är så pass nytt att det är förhållandevis få inom branschen som lärt sig att praktiskt tillämpa UML på ett bra sätt. Många tror att UML är lösningen på allt, men det är viktigt ge det tid och lära sig den rätta tillämpningen av UML, anser Bengtzohn.

4.1.4 Tillämpning av UML

Jordbruksverket har ett sätt att modellera som utvecklats i samarbete med Internationella handelshögskolan i Jönköping och infördes 1998. Den systemutvecklingsmodell som togs fram vid detta samarbete används än idag men Jordbruksverket har bestämt sig för att införa UML fullt ut men de vill inte stressa fram utvecklingen. Det är viktigt att skaffa kompetens och införa de nya UML-diagrammen (det vill säga diagram som de inte använder idag) med sunt förnuft. Den version av UML som tillämpas idag är UML 1.5 vilken är den senaste versionen innan den nya UML 2.0.

Mycket av systembeskrivningen innan UML kom in i bilden var textuella beskrivningar, men även andra grafiska analysmodeller har använts. Problemet med dessa var att de inte kunde användas genom hela systemutvecklingen. Det var svårt att hitta kopplingar mellan de olika beskrivningarna. Livscykeldiagrammen och de andra icke UML-baserade diagrammen härstammar från andra modelleringsspråk* men det är inte känt vilka.

De diagram som används idag är:

- Användningsfall i alla projekt
- Klassdiagram i nästan alla projekt
- Sekvensdiagram
- Aktivitetsdiagram
- Handlingsgrafer*
- Livscykeldiagram
- Entitetsmodell*
- Datamodell*
- Begreppslistor

Handlingsgrafer, Livscykeldiagram, datamodell samt begreppslistor görs idag utöver UML digram. Datamodellen kommer att fortsätta användas. Den är nödvändig då de jobbar i Oracles miljö och skapar relationsdatabaser. Jordbruksverket måste hitta ett sätt att gå från klassdiagram till datamodell men ingen lösning har kommit fram ännu.

En utvärdering av användningsfall har även gjorts i företaget och resultatet blev att många inte kunde se syftet med användningsfallen. Jordbruksverket har diskuterat orsakerna till detta och kommit fram till att det tidigare gjordes alldeles för många användningsfall. Nu modelleras färre och större användningsfall som verkligen ger någon nytta till aktören.

Då Jordbruksverket är mitt i införandet av UML finns många önskemål inför framtiden. Ett önskemål är att göra tillståndsdiagram istället för livscykeldiagram samt processdiagram och aktivitetsdiagram istället för handlingsgrafer. De har även planer på att införa komponentdiagram och fördelningsdiagram. Komponentdiagrammen anses vara speciellt bra för support och driftfunktionen då det är önskvärt att även de tar till sig UML. I framtiden kommer troligtvis alla diagram att användas men Jordbruksverket har inte kommit så långt i utvecklingen ännu för att kunna bestämma nyttan med alla diagram. Ett mål är att inga rena textuella beskrivningar eller andra analysmodeller ska användas. Ett diagram som troligen inte kommer att användas är samarbetsdiagrammet då de anser att sekvensdiagrammet är mer användbart.

För att kunna ersätta de gamla diagrammen med nya krävs ett verktyg där hela systemutvecklingsprocessen kan modelleras. Jordbruksverket utvärderar just nu vilket verktyg som ska köpas in men inget beslut har tagits ännu. Det är önskvärt att verktyget ska stödja hela systemutvecklingsprocessen samt kunna koppla användningsfall till processer* för att få en koppling mellan processgraferna och UML. Idag har Jordbruksverket inget verktyg som hjälp vid modellering, som stödjer UML. Det som används är Microsoft Word och PowerPoint.

4.1.5 Anpassning av UML samt dess tillämpning i olika projekt

UML:s syntax* används i samtliga diagram. Styrkan med UML är ju att det är ett regelverk. Det är möjligt att lägga till men inte förändra analysmodellerna. Jordbruksverket har inte kommit så långt att de lägger till något än, men det kan bli aktuellt i framtiden då de känner sig mer bekväma med UML. Att skapa analysmodeller generellt är Jordbruksverket inte så bra på, men när de känner sig säkra på att de uttrycker sig rätt i rätt analysmodell, då kanske de kan börja tänka på egna tillägg som visar annat, enligt Bengtzohn.

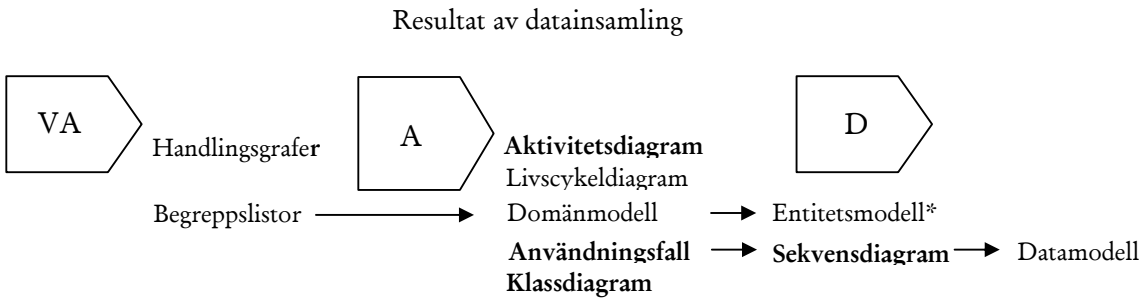
Olika diagram används beroende på typ av system som ska utvecklas. Bengtzohn anser att det inte är någon vits med att modellera bara för modellerandets skull. Han tillägger även att det är viktigt att ha känsla för vilken detaljeringsgrad som krävs för ett visst projekt samt hur mycket dokumentation som behövs för framtida förvaltning. Det är det som är det svåra. Om projektgruppen består av många duktiga utvecklare behövs eventuellt mindre dokumentation då det går att nyttja det som redan finns sedan tidigare. Modelleringen är till för att få programmerarna att skriva rätt kod och säkerställa att leverera den funktionalitet användaren vill ha.

4.1.6 UML:s plats i systemutvecklingsprocessen

Figur 4.2 visar hur Jordbruksverkets systemutvecklingsprocess ser ut idag. Den analys som Jordbruksverket gör av en verksamhet, det vill säga verksamhetsanalys, resulterar vanligtvis i handlingsgrafer* och begreppslistor, vilket är ett sätt att kartlägga verksamheten på. Begreppslistor används sedan för att skapa en domänmodell. Domänmodellen beskriver, i Jordbruksverkets fall, statiskt de begrepp som finns i verksamheten samt hur de hänger ihop med hjälp av den syntax* som används i klassdiagram (se 3.6.2). Jordbruksverket skulle gärna vilja använda sig av en processmodell istället men tyvärr finns det inget stöd för processmodeller i UML.

I nästa fas, analysfasen, modellerar Jordbruksverket användningsfall, entitetmodell* samt livscykeldiagram för det system de arbetar med. Även begreppslistorna fortsätter att utvecklas under denna fas. För tillfället pågår ett pilotprojekt på Jordbruksverket där de genomför en så kallad användningsfallsrealisering, vilket bland annat resulterar i att sekvensdiagram tas fram. Dessa diagram visar, på en logisk nivå, hur den funktionalitet, som beskrivs i ett användningsfall, ska tas fram. Under analysfasen modellerar de även scenarioträd (basflödesmodell) som är till för att visa aktiviteter* vid en viss händelse. För att göra detta tar de hjälp av syntaxen för aktivitetsdiagram. Även Jordbruksverkets livscykeldiagram använder syntaxen från ett UML-diagram nämligen tillståndsdiagrammet. Det sista som sker i analysfasen är att de skapade domänmodellerna utvecklas till entitetmodeller.

I designfasen förtydligas de tidigare skapade sekvensdiagrammen för att senare utvecklas till en datamodell*. Textbaserade kravspecifikationer samt en del andra RUP-baserade analysmodeller (se avsnitt 3.1.2) kan också tas fram under denna fas om Jordbruksverket anser det behövas för projektet.



Figur 4.2 Befintlig systemutvecklingsmodell på Jordbruksverket

4.2 Pdb

Pdb är ett personalägt IT-konsultföretag som startades 1983 och har idag cirka 50 anställda. Företaget både vidareutvecklar tidigare system samt erbjuder sina kunder nya systemutvecklingslösningar. Pdb är uppdelat i sex olika team (se Figur 4.3) som har olika arbetsuppgifter.

BI-team	Teknik-team	IBM-team	Microsoft-team	Java-team	Affärs-team

Figur 4.3 Pdb – Avdelningar.

Vidareutveckling av system arbetar IBM-teamet mycket med medan Microsoft- och Java-teamen i större utsträckning arbetar med både vidareutveckling och utveckling av nya system. Affärsteamet är specialister på kundens affärsprocesser och arbetar även mycket marknadsinriktat. Ofta är det affärsteamet som knyter nya kunder till Pdb. Teknikteamet hjälper de övriga teamen under systemutvecklingsprocessen med bland annat tekniska lösningar. BI-teamet är en enhet som arbetar med utvecklingsfrågor rörande det egna företaget.

Pdb är ett kunddrivet företag då de flesta uppdrag kommer från kunder, men de arbetar även med interna projekt. De har inte några egna produkter som de tillverkar och säljer. Pdb har bland annat stora företag såsom Husqvarna och Electrolux som kunder. De olika teamen inom företaget arbetar med olika typer av kunder. IBM och Java arbetar med mest stora företag och Microsoft arbetar mer mot företag med mindre och fler uppdrag. För alla team är tillverkande företag vanligast och det är dessa företag de fokuserar mest på.

Mathias Rehnström är teamledare och systemutvecklare på företaget Pdb och intervjun med Rehnström ägde rum hos Pdb den 16 november 2004.

4.2.1 Systemutvecklingsmodell

Idag använder sig Pdb av en systemutvecklingsmodell som de kallar PUM (Pdb:s uppdragsmodell) PUM är en egentillverkad utvecklingsmodell och bygger på formaliserade erfarenheter som de vet fungerar. PUM består av olika faser såsom kravspecifikation och detaljspecifikation. Mellan varje fas görs en avstämning för att kontrollera projektets förlopp vilket kan liknas vi en milstolpe i RUP (se avsnitt 3.1.2). Specifika-

tionerna är uppdelade i en funktionell och en teknisk beskrivning. I systemutvecklingsmodellen sker en iteration inom varje fas och PUM kan liknas vid RUP (se avsnitt 3.1.2) eller en iterativ livscykelmodell (se avsnitt 3.1.1). Utefter de olika specifikationerna skapas en prototyp. PUM används som ett riktmärke i alla uppdrag för att få upp en standard i uppdragen med dokumentation, checklistor samt vilka dokument som ska finnas. PUM har bara använts av Pdb under något år och den har vidareutvecklats sedan införandet. Det har dock funnits problem längs vägen. Rehnström menar att det ofta uppkommer problem vid användandet av nya metoder och utvecklingsmodeller. Detta anser Rehnström även vara fallet med UML.

Sammanfattningsvis kan sägas att Pdb:s tidigare arbete samt stor del av dagens arbete består av textuella beskrivningar och sunt förnuft enligt Rehnström.

4.2.2 Modellering

Pdb:s systemlösningar bygger på att kunna integrera de nya systemen mot vilket annat system som helst. Detta innebär att högre krav ställs på exempelvis dokumentation och design till skillnad från system som inte är tänkta att integreras med andra eller som inte ska vidareutvecklas. Detta har lett till att Pdb delvis tagit steget från textuella beskrivningar till att modellera system med hjälp av UML då det ger en bra dokumentation, bättre förståelse samt att det är en standard.

Rehnström säger att UML kan vara ett bra sätt att kommunicera men de har inte haft erfarenheter av det än på Pdb. De har använt sig av användningsfallsdiagram i kravspecifikationer och anser att det är väldigt få kunder som förstått dem. Han menar att kunder istället vill ha en skärmdump eller en prototyp. Det finns dock ett undantag och detta är inom IBM teamet, där kunderna oftast är kravställarna själva och har stora erfarenheter av systemutveckling.

Rehnström använde sig mycket av modellering på sin tidigare arbetsplats och det modelleringsspråk* som tillämpades var UML. När han kom till Pdb förde han med sig idén och kunskapen om UML. Användandet av diagram i systemutvecklingsprocessen fick bra respons av andra inom Pdb och intresset väcktes.

Anledningen till att Rehnström förespråkar användandet av UML, som stöd till PUM, inom Pdb är att UML ger möjligheter att beskriva komplexa flöden bättre än de textdokument som används idag. Rehnström menar att *”en bild säger mer än tusen ord”*.

Ett starkt skäl till att börja arbeta med UML på Pdb är att kunna vidareutveckla andras system och inte bli personberoende på företaget det vill säga att bara en person kan utföra ett visst arbete. Det är bättre att använda en standard som alla inom företaget känner till. UML bidrar även till en bra dokumentation av systemet vilket underlättar förvaltning och drift.

4.2.3 Styrkor och svagheter med UML

Enligt Pdb är en styrka med UML att det kan underlätta arbetet med andras system som är dokumenterade med UML. De anser att de lättare kan få en bild över hur sy-

stemet är uppbyggt, vilka flöden som påverkar varandra mm. Tidigare (eller för system som inte är dokumenterade med UML) behövdes en ”reverse engineering” göras på systemet för att få en överblick. Reverse engineering betyder att systemet bryts ned i mindre delar för att lättare förstå hur systemet är uppbyggt. Rehnström menar att UML lyfter systemet till en högre nivå och skapar bättre förståelse och är speciellt användbart då ett system består av komplexa flöden. Han tillägger även att ett diagram tar mindre plats och andra kan lättare vara med i diskussionen över ett diagram än då något är textuellt beskrivet. Det kan vara svårt att få andra att läsa en text och ännu svårare att få dem att förstå.

Vid användning av UML inom Pdb finns dock ett problem och det är att hitta minsta gemensamma nämnare mellan de olika teamen. Inom Java och Microsoft skulle de kunna använda hela UML men inte inom IBM, då de använder ett icke objektorienterat arbetssätt. De vill inte att företaget ska ”spreta” för mycket utan arbeta så lika som möjligt. Det blir svårt att tillämpa UML i sin helhet på ett företag såsom Pdb då nästan alla delar av UML stödjer ett objektorienterat arbetssätt vilket Rehnström anser är synd då det fortfarande finns många som arbetar icke objektorienterat. Han anser att UML är ett komplext språk som kan vara svårt att sätta sig in i samt tidskrävande att arbeta med, speciellt i mindre projekt. Det är viktigt att hitta en balansgång mellan ”tid och nytta”, diagrammen bör uppfylla sitt syfte och inte bara användas för användandets skull. Ett annat problem med UML är enligt Rehnström verktygen. Det finns många olika verktyg som stödjer UML, både komplexa och lite enklare varianter. Pdb vill ha ett relativt enkelt verktyg, då det annars kan var lätt att fastna i verktygets komplexitet, men problemet med ett sådant verktyg är ofta att det inte kan användas under hela systemutvecklingsprocessen. Verktygen som finns idag använder inte heller alltid samma syntax* vilket kan göra arbetet med UML komplicerat.

4.2.4 Tillämpning av UML

Pdb har, som tidigare nämnts, kommit i kontakt med UML via Mathias Rehnström som arbetat med UML på sin förra arbetsplats. Innan införandet av UML har det inte använts några andra modelleringsspråk* inom Pdb. Det förekom dock att de använde Flowchart för att beskriva flöden, vilket de i viss utsträckning gör än idag. Detta diagram har mestadels använts av affärsteamet för bland annat kartläggning av processer*. Det är, som nämnts, sunt förnuft och textuella beskrivningar som har styrt Pdb:s systemutvecklingsarbete.

UML har stegvis kommit in i företaget och har använts mer eller mindre aktivt under cirka tre år. Modelleringsspråket används inte aktivt bland alla team och dominerar inte systemutvecklingsarbetet. Målet för Pdb är att alla team som är engagerade i systemutvecklingsarbete ska använda sig av UML.

Alla anställda på Pdb har idag utbildning på UML. Det är dock ingen omfattande utbildning utan bara ett antal endagars kurser har genomförts. Rehnström anser att ingen behärskar UML fullt ut då UML är stort och komplext och inget man lär sig på några dagar.

Följande diagram använder Pdb idag:

- Användningsfallsdiagram
- Klassdiagram (endast inom Java-teamet)
- Sekvensdiagram
- Flowchart

Flowchart används speciellt av affärsteamet för att beskriva flöden i ett system. Flowchart liknar till viss del ett tillståndsdiagram.

Inom Pdb idag används många olika typer av verktyg varav ett är en relativt ny investering. Rehnström menar att detta verktyg inte är helt optimalt för Pdb i dagens läge då de är ganska nya användare av UML men verktyget kommer antagligen att bidra till att företaget arbetar mer och mer med UML kopplat mot kod. Att generera kod utifrån en analysmodell är Rehnström dock tveksam till. Han ser faran i att tappa kontrollen över koden och Pdb kommer antagligen att vänta med att ta detta steg. Pdb använder idag version 1.5 av UML men om företaget bestämmer sig för att satsa på UML kommer de antagligen att införa UML 2.0 som Pdb tycker verkar bra och som stöds av det nya verktyget.

UML ska i framtiden vara ett komplement till befintliga specifikationer och användas i så stor utsträckning som möjligt inom Pdb. Exempel på diagram som dock inte kan användas överallt är klassdiagram och tillståndsdiagram. Detta beror på, som nämnts tidigare, att de behandlar objekt* och dess tillstånd vilket inte ett funktionellt arbetssätt och stödjer således inte IBM-teamets arbetssätt. Två diagram som Pdb enligt Rehnström borde införa är fördelningsdiagram och samarbetsdiagram. Dessa kan fungera inom alla team och fördelningsdiagrammet är dessutom viktigt för att stödja kommunikationen med teknikteamet.

4.2.5 Anpassning av UML samt dess tillämpning i olika projekt

Förändringar i UML-diagrammen görs inte av Pdb, utan de arbetar med den syntax* som verktygen stödjer. Verktygens syntax kan däremot skilja sig från varandra.

Vilka diagram som Pdb använder sig av i sitt arbete är projektberoende och även beroende på vilket team som är inblandat. Inget av diagrammen är en standard inom företaget det vill säga något som måste/ska användas inom alla projekt. Beslutet att använda diagram beror på komplexiteten av projektet och vem som arbetar med projektet. Generellt så använder dock Pdb UML i de uppdrag där de har väldigt komplexa flöden och inte vill eller har svårighet att beskriva något i löpande text.

4.2.6 UML:s plats i systemutvecklingsprocessen

Tillämpbara diagram inom IBM-teamet är främst användningsfall och sekvensdiagram. Dessa diagram används för att kartlägga flöden. UML används i övrigt främst i detaljspecifikationen, som kan likställas med design/utformningsfasen, då det är där de kontrakterar vad som ska byggas. Inom Pdb använder de ofta UML som ett kon-

trollredskap eller kvalitetssäkring, de kontrollerar koden de gjort genom att skapa ett diagram från den.

4.3 Saab Combitech Systems

Saab Combitech Systems är ett kunskapsorienterat konsultföretag med spetskompetens på inbyggda realtidssystem. Företaget arbetar både med externa och interna kunder där de interna kunderna är de som innefattas i Saabkoncernen. Det externa konsultarbetet är det som dominerar och cirka 60-70 % av alla uppdrag är externa. Det interna konsultarbetet är således cirka 30-40 % av den totala arbetsmängden.

Den dominanta uppdragstypen på Saab Combitech Systems är produktutveckling i realtidssytem men de arbetar även med verksamhetsförbättring. Saab Combitech Systems erbjuder dessutom sina kunder utbildning i UML och introducerar dem i objektorienterad teknik och metodik. Anledningen till att kunder vill lära sig UML är för att kunna arbeta med det på egen hand och utveckla egna system och därmed effektivisera sin verksamhet och kunna utveckla sina produkter.

Saab Combitech Systems kunder utgörs av stora, medelstora och små företag som arbetar med produkter innehållande programvara. De utvecklar system som används i olika typer av produkter såsom bilar men inte informationssystem som exempelvis affärssystem till kunder. De arbetar inte heller mot företag som utvecklar informationssystem.

Kunder till Saab Combitech Systems kan vara bilindustrin, försvarsindustrin, telecomföretag eller företag som arbetar med medicinteknik. Företaget är inte inriktat mot någon speciell bransch utan emot en viss teknik eller typ av system dvs. realtidssystem.

Anders Mattsson arbetar som affärsenhetschef samt konsult på Saab Combitech Systems i Jönköping. Vår intervju med Mattsson ägde rum den 24 november 2004 på Combitechs kontor. När vi i detta avsnitt talar om Saab Combitech Systems refererar vi till Jönköpingskontoret.

4.3.1 Systemutvecklingsmodell

Saab Combitech Systems har en egenutvecklad systemutvecklingsmodell kallad Parts. Vilken systemutvecklingsmodell som företaget använder sig av är till viss del kundstyrt. De kunder de arbetar med har oftast krav på hur arbetet ska utföras. Om så inte är fallet eller om kundens tillvägagångssätt är bristfälligt introducerar Saab Combitech Systems sitt eget arbetssätt Parts för kunden.

Parts är liksom RUP iterativ och fasindelad med milstolpar mellan de olika faserna. Det som skiljer är att Parts inte är lika omfattande samt att den sista fasen benämns Införande istället för Övergång (se Figur 4.4). De fyra faserna i Parts är Inledning, Utveckling, Konstruktion samt Införande. Varje fas i systemutvecklingen avslutas med en milstolpe, där beslut tas huruvida projektet ska drivas vidare eller inte. Milstolparna är en viktig del av utvecklingsmodellen. Mattson förklarar att Parts i första hand är en metamodell som talar om vilken information som ska tas fram. Den inne-

håller de analysmodeller som ska arbetas fram och information om hur de ska modelleras. Parts består även av förslag på hur arbetet med analysmodellerna kan gå till.

De gånger då Parts inte används i ett projekt är det ofta kundens branschtillhörighet som avgör vilken utvecklingsmodell eller process* som ska användas. RUP är väldigt frekvent förekommande bland många kunder men framförallt inom telecomföretag. Inom försvarsindustrin däremot baseras processer nästan alltid på Milstandard 498 som är en speciell dokumentationsstandard inom försvarsmakten. Dokumentationen är för det mesta det som avgör valet av systemutvecklingsmodell från kunden sida. Saab Combitech Systems kunder har i sin tur kunder och det kan även vara dessa som styr hur dokumentationen bör se ut. Det kan till exempel handla om hur vissa specifikationer eller milstolpar ska illustreras. Då arbetet för Saab Combitech Systems är mycket kundstyrt är det vanligt att kunden har en egen utvecklingsmodell som används i systemutvecklingsarbetet.

4.3.2 Modellering

Den främsta anledningen till att Saab Combitech Systems modellerar är dokumentationen. De har ofta höga krav från beställaren och på sig själva att arbetet och systemet blir väldokumenterat, så att systemet inte blir personberoende och att andra kan fortsätta arbetet med systemet. Det är dock varierande vad beställaren anser om modellering och hur delaktiga de vill vara i systemutvecklingsarbetet. Beställarens kunskap varierar mycket och därmed även deras synpunkter på modellering. Mattson menar dock att en grafisk lösning kan vara lättare att acceptera än en komplicerad text. De enda gångerna UML används i ett funktionellt sammanhang är om kunden har ett gammalt system som är designat genom funktionsorientering (se avsnitt 3.3). Det blir då aktuellt att försöka göra en objektorienterad design av systemet för att underlätta fortsatt underhåll. Mattsson anser det vara svårt att utföra ett bra designarbete med UML utan att använda ett objektorienterat angreppssätt. Det är i stort sett bara användningsfallsmodellering som går att anpassa till det funktionsorienterade utvecklingsarbetet.

Från Saab Combitech Systems sida är det önskvärt att arbeta med en engagerad kund då detta leder till ett bättre system. Mattson berättar vidare att det inom vissa branscher kan vara svårare att få kunden inblandad i arbetet än i andra. I de fall kunden är engagerad är UML ganska tacksamt att arbeta med då det finns relativt stor förståelse för objektorientering.

Det fanns flera olika faktorer som bidrog att Saab Combitech Systems gick över till att arbeta med UML. Då företaget har arbetat objektorienterat sedan början av 1990 och använt sig av bland annat OMT som legat till grund för UML var det inte ett så stort steg att ta, att gå över till UML. UML är dock ett semantiskt rikare språk än många av sina föregångare och de grundläggande principerna i språket är inte svåra. Mattson anser att något som saknas i UML jämfört med OMT eller Coad/Yordon är att dessa innehåller metodik, det vill säga hur och i vilken ordning samt vilka diagram som ska tas fram. Vid modellering med UML tar företag hjälp av modeller såsom RUP, eller så skapar de ett eget tillvägagångssätt. Saab Combitech Systems hade inte heller några produkter som var beskrivna med andra språk (exempelvis ett affärssy-

stem) och som de behövde ta hänsyn till det vill säga inga strategiska problem. Företagets ständiga teknikbevakning samt intresse av att kunna det som kunden tros vara intresserad av bidrog till övergången. Den sista men inte minst viktiga anledningen är att UML är ett standardiserat modelleringsspråk* som skapar enhetlighet och som innehåller det bästa av tidigare språk. Mattson menar att nästan alla använder UML vilket är en stor fördel och en mycket god anledning till att arbeta med just detta modelleringsspråk.

4.3.3 Styrkor och svagheter med UML

Styrkan med att modellera på en högre nivå så som med UML är att det blir lättare att kommunicera design med andra som inte är med i projektet då högnivådesign ligger närmare hur vi människor tänker. Alternativet är att använda kod som ligger på en lägre abstraktionsnivå vilket gör det svårare att beskriva och ta till sig framförallt struktur och tillstånd. Det är även svårt att översätta fram och tillbaka från kod, svårt att bara genom att titta på kod se hur exempelvis olika klasser är relaterade till varandra. Genom att använda sig av UML kan även misstag upptäckas på en tidig nivå. Mattsson anser att om UML har använts tidigare i ett system, och det stämmer överens med koden, så är det oftast mycket bättre designat än med ett annat arbetssätt. UML leder därmed till bra design av system och underlättar fortsatt arbete med system.

Att det finns verktyg som stödjer UML och som är dubbelriktade, det vill säga att koden kan utformas utifrån analysmodellen och analysmodellen utifrån ändringar i koden, är mycket positivt enligt Mattsson. Det kan vara en stor fördel att inte tvinga folk att arbeta i kod först, utan att ha en analysmodell att arbeta med och utgå från. Enligt Mattson så kan ibland kod och analysmodell skilja sig åt och det beror ofta på att koden vidareutvecklas utan att utöka analysmodellen, då det inte finns verktyg som stöder kopplingen mellan kod och analysmodell Mattson menar att en svaghet med verktygen är att de kan ha olika syntaxer* för UML. Syntaxen är visserligen ganska lika, men det kan ändå skapa missförstånd i diskussioner. Han anser därför att syntaxen är något som behöver standardiseras i verktygen.

En annan svaghet med modelleringsspråket* UML är att det är ett generellt språk (har många olika användningsområden) som är relativt omfattande och komplext och det är oftast bara en del av språket/vissa mekanismer som behövs. Mattsson menar dock att själva språket ändå är relativt lätt att lära sig och att svårigheten ligger i hur systemet ska designas på ett bra sätt.

4.3.4 Tillämpning av UML

Saab Combitech Systems har, som nämnts tidigare, arbetat med objektorienterad metodik och modellering sedan detta arbetssätt introducerades i början av 1990. De arbetade huvudsakligen med modelleringsspråken* OMT och Coad/Yordon. Mattsson förklarar att sättet att modellera i OMT och Coad/Yordon är relativt likt modellering i UML, speciellt OMT som är en av föregångarna till UML. Innan UML introducerades fanns det många olika modelleringsspråk, vilket enligt Mattsson var ett problem. Det kunde vara svårt att välja modelleringsspråk och det fanns ingen enhet-

lighet mellan språken. Det var även problem med att välja verktyg då det fanns en mängd olika verktyg att välja mellan för de olika modelleringsspråken. Att välja modelleringsspråk och verktyg för varje projekt kunde vara komplicerat och tidskrävande och därmed även kostsamt.

Då UML är uppbyggt av tidigare modelleringsspråk är det inte så konstigt att Saab Combitech Systems använder ungefär samma diagram idag som de gjorde innan UML. Det som skiljer diagrammen då från nu åt är att de ritas lite annorlunda. Det grafiska har utvecklas i UML. De diagram som kan kännas igen från UML och som användes tidigare är sekvensdiagram, klassdiagram och tillståndsdiagram.

UML har inom Saab Combitech Systems använts sedan introduktionen 1997 och de har hängt med i utvecklingen sedan dess.

De diagram som används av företaget idag är:

- Klassdiagram
- Användningsfallsdiagram
- Sekvensdiagram
- Tillståndsdiagram
- Objektdiagram (Samarbetsdiagram)
- Komponentdiagram
- Fördelningsdiagram:

Saab Combitech Systems använder även till viss del olika hårdvaruspecifikationer som innehåller diagram.

Tillståndsdiagrammen anser Mattson vara av stor vikt i de flesta realtidssystem då det är händelseorienterat. Komponentdiagram används däremot inte i någon större utsträckning då de enligt Mattson är semantiskt fattiga och inte stöds av alla verktyg. Även Fördelningsdiagrammen används väldigt sällan då det enligt Mattson är en förutsättning att det finns ett verktyg som stödjer dessa och som kan generera kod från diagrammen samt allokera processer* och objekt*.

Något som företaget har behov av är processdiagram. Det finns idag inga processdiagram som kan visa på hur processer kommunicerar i UML. De vill ha möjligheten att visa processerna klart och tydligt samt i samband med UML det vill säga hur processerna är kopplade till objekt, vilka objekt som går i vilka processer.

4.3.5 Anpassning av UML samt dess tillämpning i olika projekt

Saab Combitech Systems modellerar diagrammen i UML enligt den syntax* som finns med reservation från de skillnader i syntax som finns i de olika verktygen. De använder således den syntax som verktygen stöder och i dem kan både utökningar och vissa avsteg från UML finnas.

Mattsson anser att de flesta diagram som företaget använder (se avsnitt 4.3.4) behövs i alla projekt oberoende av storleken på projekt eller företaget de arbetar med. Mattsson berättar vidare att alla diagram de använder är viktiga för att kunna beskriva ett system på ett så bra sätt som möjligt och anser att bara för att det är ett mindre system som ska utvecklas är det inte mindre viktigt att det blir bra. Det kan dock vara så att diagram utelämnas för att det inte finns någon funktion för dem, om det exempelvis inte finns något tillståndsberoende beteende i systemet så kan inte tillståndsdiagram användas. Samma sak kan gälla för användningsfallsdiagrammet som kan väljas bort om kraven redan har dokumenterats på ett tillfredställande sätt innan Saab Combitech Systems är delaktiga i arbetet. Mattsson tillägger även att en del diagram är mindre viktiga ur aspekten att kunna generera kod från dem, exempelvis så genereras inte kod från sekvensdiagrammet utan det visar på hur olika delar av systemet ska arbeta med varandra. Sekvensdiagrammet är viktigt för att kunna anknyta till kraven samt för att kunna utföra bra testfall.

Ibland har det dock hänt att diagram utlämnats på grund av tidsbrist men detta påpekar Mattson inte är bra och menar att samtliga diagram ska användas i största möjliga utsträckning men självklart inte i onödan, utan de ska vara relevanta för projektet.

Vilken version av UML som används inom Saab Combitech Systems idag är beroende på projekt och verktyg, då olika verktyg stödjer olika versioner. Utvecklingen inom företaget går mot användandet av UML 2.0 då den har ett flertal nya intressanta mekanismer enligt Mattsson. UML 2.0 innehåller nya mekanismer som är inriktade mot den sorts system de arbetar med det vill säga realtidssystem.

4.3.6 Tillämpning av UML i systemutvecklingsprocessen

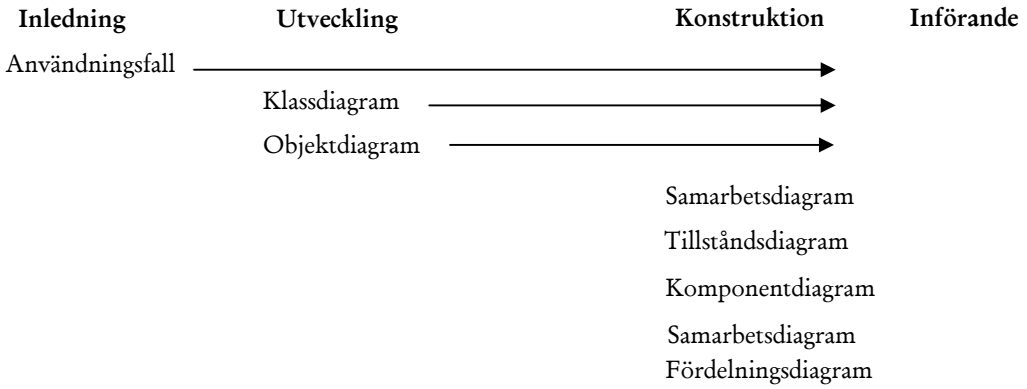
Faserna i en systemutvecklingsmodell är, som nämnts tidigare, ett hjälpmedel för att fokusera på rätt saker i rätt ordning. Det är dock enligt Mattsson så att nästan alla diagram finns med under varje fas men det fokuseras och arbetas *olika mycket* med dem beroende på fasen (se Figur 4.4).

I inledningsfasen handlar arbetet om att se om systemet är realiserbart samt att ställa krav på hur systemet ska se ut. I denna fas arbetas det även mycket med analys samt en del med design och prototyping. Användningsfallsdiagram används för att beskriva krav, därav används de mestadels i den inledande fasen. Om arbetet är iterativt kommer sedan användningsfallen att underhållas över tiden, då arbetet går in i de andra faserna. Även klassdiagrammen kan påbörjas i denna fas.

Utveckling är den fas där projektet egentligen börjar för företaget, då inledningsfasen är en slags förstudie. Här fortsätter arbetet med definitionen av krav som bör bli i stort sett klar. Arbetet med att beskriva hur objekt* i en viss klass* beter sig är viktigt under denna fas därav modellering av klassdiagrammen. Till viss del inleds även utformningen av processer*.

Under konstruktionsfasen ligger mycket fokus på fortsatt utveckling av klassdiagram och tillståndsdiagram. Modellering av sekvens och/eller samarbetsdiagram samt komponentdiagram påbörjas. Om både sekvens- och samarbetsdiagram tillämpas sker

arbetet parallellt då de är mycket lika varandra. Största delen av koden konstrueras under denna fas och komponentdiagrammet används till exempel för att tillämpa kodgenerering. Ett antal iterationer sker av varje fas och i varje iteration genereras/körs systemet så användningen av komponentdiagrammen är fördelad över de olika faserna. I den avslutande fasen införande sker implementation av systemet.



Figur 4.4 Systemutvecklingsmodell för Saab Combitech Systems: ”Parts”.

Mattsson tycker det är bra att kunna generera kod från analysmodeller och försöker övertyga sina kunder om att arbeta i diagram/analysmodeller för att sedan kunna gå hela vägen så att även de genererar kod. Idag finns inte denna möjlighet i så stor utsträckning. De flesta verktyg stödjer generering av kodskelett, men det är mycket få som stödjer generering av fullständig kod från analysmodeller. Han menar att det är en bra vision som OMG har med UML, att det ska kunnas användas hela vägen och det här är även något som Saab Combitech Systems utnyttjar. Att generera kod från en analysmodell resulterar i både tid- och kostnadsminskningar samt leder till bättre design. Mattsson anser dessutom att det finns problem med att göra tvärt om, att gå från kod till analysmodell. Det är då är det lätt hänt att tänka för mycket på koden och designen blir oftast sämre.

4.4 WM-data

WM-data grundades 1969 och har cirka 8000 anställda inom olika IT-områden. Deras affärsidé innefattar bland annat att leverera system som ger användbar och bestående kundnytta. Ett av WM-datas kontor är beläget i Jönköping och där finns cirka 40 anställda (WM-data, 2004).

De största kunderna för Jönköpingskontoret är Domstolsverket och olika försäkringsbolag då WM-data har ett standardsystem för försäkringsbolag. Trygghansa är en av dem som tagit till sig detta standardsystem. Volvo IT Skövde är en annan stor kund för Jönköpingskontoret och de arbetar även med en del mindre projekt hos privata företag.

För Jönköpingskontoret är ungefär hälften av projekten nyutveckling och hälften vidareutveckling. De senaste två tre åren har IT-budgetar gjort att företag satsar mer på den billigare varianten, alltså vidareutveckling. Det är också så att fler och fler företag redan har ett väl fungerande system.

Jonas Florvik är konsult på företaget WM-data i Jönköping. Vår intervju med Florvik ägde rum hos WM-data den 17 november 2004. Vi vill även nämna att när vi skriver WM-data menas generellt Jönköpingskontoret och inte hela WM-data.

WM-data i Jönköping har använt UML aktivt mot kunder sedan 2000 då Florvik själv förde med sig UML in i företaget. Inom WM-data har det däremot funnits sedan det lanserades 1997. Sättet Jönköpingskontoret arbetar på kan inte räknas som någon standard för hela WM-data. Florvik menar att det kan finnas helt andra standarder på andra kontor.

4.4.1 Systemutvecklingsmodell

RUP är den systemutvecklingsmodell som används generellt inom WM-data (se avsnitt 3.1.2). Den används i en mängd olika projekt beroende på sin enorma kapacitet och flexibilitet. Utifrån RUP och UML har WM-data arbetat fram en standardmodell som en grund för vad som ska eller bör göras i ett projekt men den anpassas självklart till varje enskilt projekt.

WM-data har valt RUP bland annat på grund av dess samklang med UML. UML finns i alla RUP:s faser även om vikten av diagrammen varierar mellan de olika faserna. En annan anledning till valet av RUP är att det är en vedertagen utvecklingsmodell.

4.4.2 Modellering

Den största anledningen till att WM-data modellerar är att de anser att resultatet efter en noggrann modellering blir ett system med hög kvalitet. Andra anledningar är att modelleringen ger en bra dokumentation som beskriver vad systemet gör, hur det fungerar samt vilka krav som finns. Det är viktigt att göra både en övergripande beskrivning av ett system och en detaljerad beskrivning som visar egenskaper för olika komponenter. Dokumentation underlättar även arbetet avsevärt när det är aktuellt att sätta sig in i hur ett befintligt system fungerar. En sådan situation uppstår till exempel när projektet består i att vidareutveckla ett befintligt system eller om en ny person kommer in i projektgruppen.

Det händer ibland att kunden inte förstår syftet med modellering. Florvik anser dock att detta inte bara gäller kunden utan också interna medarbetare som gärna går direkt på själva programmeringen. Det finns alltid ett behov av att jobba med förståelsen för modellering när ett projekt drar igång. Om kunden är villig att lägga pengar på modellering så är det önskvärt att hålla en liten utbildning i början av projektet. Florvik poängterar dock att förståelsen för modellering har ökat då det har visat dig att resultatet efter en noggrann modellering blir ett system med hög kvalitet.

WM-data använder sig av standardiserade modeller och metoder som är vedertagna. Det är därför de använder både UML och RUP. Beslutet att använda UML som utgångspunkt vid systemutveckling kom då de tittade på hur dom drev projekt och även mycket på hur RUP skulle passa in i bilden. I samband med detta hamnade WM-data i Jönköping i ett stort projekt där de skulle ha ansvar för systemutveck-

lingsmetodik, dvs. hur utvecklingsarbetet går till. Florvik tog då fram en anpassad version av RUP där hela systemutvecklingsprocessen skulle vara väldokumenterad från krav till testad kod. Valet av modelleringsspråk* blev då UML eftersom det låg närmast RUP. Ännu ett skäl var att de skulle jobba med ett antal inlånade konsulter från olika orter och ville arbeta efter något som är standard för att lättare komma igång med arbetet. Det är vanligt att andra konsulter använder UML och det var därför passande. Det skulle dessutom spara mycket tid som annars hade gått till utbildning. UML ingår även i många av IT-utbildningarna på Högskolor och Universitet vilket WM-data ser som en fördel vid rekrytering av personal.

4.4.3 Styrkor och svagheter med UML

En av styrkorna med UML är enligt Florvik att det är en standard eller ”i alla fall det närmsta vi kommer en standard i modelleringsvärlden”. Detta medför att många är insatta i hur UML fungerar. WM-data har uppfattningen att det till exempel är vanligt att konsulter och nyutexaminerade studenter redan är bekanta med UML när de kommer in i organisationen. Det kan till och med vara så att kunden jobbar med UML. En annan styrka som Florvik nämner är den höga kvaliteten på UML som modelleringsspråk*. UML är en sammansättning som består av det bästa från tidigare metoder och den har utvecklats till att vara anpassningsbar till många olika typer av projekt. Det finns även många verktyg för UML numer. Att det även finns verktyg för att generera kod är ett framsteg enligt Florvik då det går att få fram en bra arkitektur för konstruktionen genom kodgenerering. WM-data anser att kodgenerering kan vara särskilt användbart i små projekt då det sparar mycket tid. En annan väsentlig styrka med UML är att UML passar utmärkt som dokumentation som hjälp vid förvaltning av systemet.

En brist i UML anser Florvik är att klassdiagrammet beskriver vad som ska vara i databasen men att det är lätt att tappa bort det senare eftersom klassdiagram fokuserar på objekt*. Det är därför viktigt att tidigt ha någon form av logisk datamodell* där det anges vilka begrepp som finns i organisationen samt vad som menas med begreppen. Det är viktigt att försäkra sig om att klassdiagrammet verkligen visar vad som ska finnas i databasen.

En annan nackdel uppstår enligt Florvik om syntaxen* följs för slaviskt. Det går att diskutera hur en pil ska se ut i timmar och det är ju onödigt. Det är viktigt att inte bli för ambitiös och använda alla delar till punkt och pricka även då de inte behövs.

4.4.4 Tillämpning av UML

WM-data i Jönköping har, som tidigare nämnts, använt UML aktivt mot kunder sedan 2000. Även innan år 2000 har många använt delar av UML. Ett exempel är användningsfall som då har förekommit i andra skepnader såsom den ursprungliga formen av användningsfall Jacobsson utvecklade i OOSE-metoden (se avsnitt 3.4). Användningsfall är de som har funnits inom WM-data längst. Klassdiagram har också funnits länge, men inte heller de med exakt UML-syntax. Dessutom har de tidigare använt sig av tillståndsdiagram och sekvensdiagram. Även andra typer av modellering används inom WM-data. Ett exempel på detta är JSP* (Jackson Structured Program-

ming) men även en mängd andra sätt som inte tillhör en specifik metod. Det kan till exempel vara något som kunden använder eller en analysmodell som någon hittat på själv för att göra en enkel illustration.

Systemutvecklingsarbetet utgår idag från UML, men vissa undantag kan göras. Ett undantag kan vara om många i projektgruppen inte använt UML tidigare vilket gör att modelleringen skulle bli mycket tidskrävande. Det kan också vara aktuellt att använda kundens sätt att modellera i de fall det råder tidspress att leverera systemet i tid eller att kunden har som önskemål att deras sätt ska användas. De enda tillfällen då WM-data arbetar med funktionell systemutveckling är när kundens system är av den typen. Även då kan det hända att WM-data tillämpar en objektorienterad ansats. Florvik menar att användningsfall alltid kan användas oavsett vilken typ av systemutveckling som bedrivs. Klassdiagram är svårare eftersom de använder objekt* men de går dock att anpassa. Sekvensdiagram är flöden men om en liten anpassning görs är det inte tvunget att se det som objekt i flödet.

Användningsfall är en grund som används i många projekt liksom klassdiagram som också de används nästan uteslutande vid objektorienterad utveckling. Objektdiagram används inte då klassdiagram och objektdiagram illustrerar i stort sett samma sak. För att beskriva flödet i systemet används antingen sekvensdiagram eller samarbetsdiagram men vanligtvis samarbetsdiagram då samarbetet mellan objekten vanligtvis är viktigare att illustrera än tiden. Tillståndsdiagram används även de mycket sällan. Komponentdiagram används antingen i detalj eller övergripande för att beskriva hur systemet struktureras upp. Aktivitetsdiagram och fördelningsdiagram är något som vanligtvis inte används. De diagram som används beror ofta på projektets storlek. I små projekt används inte alla diagram och även de som används modelleras ofta på en högre abstraktionsnivå då systemet inte är så komplext.

Inga speciella diagram från andra modelleringsspråk* nyttjas. Om det är några förändringar är det ofta varianter av till exempel klassdiagram eller något som kunden använder. JSP används dock fortfarande i vissa fall.

De diagram som är aktuella idag är:

- Användningsfallsdiagram
- Klassdiagram
- Samarbetsdiagram
- Sekvensdiagram
- Komponentdiagram
- Aktivitetsdiagram
- Tillståndsdiagram
- Fördelningsdiagram

WM-data använder redan många av diagrammen i UML, men det finns inget som utesluter att fler diagram kommer att användas i framtiden. Florvik hoppas till exempel att WM-data i framtiden ska kunna använda sig mer av generering av kod. Det kräver dock ett bättre verktyg. WM-data håller just nu på att utvärdera verktyg för UML 2.0 och överväger att ta till sig denna nya version av UML. WM-data i Jönköping vill använda ett verktyg som ger bra resultat i slutändan oavsett korta eller långa projekt. Då WM-data är en stor organisation som arbetar med många olika typer av projekt använder de idag nästan alla utvecklingsverktyg som finns.

4.4.5 Anpassning av UML samt dess tillämpning i olika projekt

Vissa förändringar i diagrammen görs. Det kan till exempel vara att man skalar ner lite för att ta med det som måste tas med istället för att ta upp tid med mer detaljerade beskrivningar. Enligt Florvik är det viktigt att tänka på att inte dokumentera för dokumenterandets skull. Anpassningar får däremot inte bli för anpassade så de går ifrån UML. Ett exempel på en anpassning är att de vid vissa tillfällen använder heldragen pil för alla relationer då typ av relation är inte så viktig. Kanske förenklas även beskrivningar i klassdiagram etc. Det blir däremot viktigare att vara noggrann med syntaxen* när modelleringen ska användas för att generera kod.

Det är storleken på projektet som ofta styr antal diagram som används. En princip som följs är att göra så lite jobb som möjligt. Med det menas att för att bestämma vilka diagram som är nödvändiga är det viktigt att utvärdera varför en viss beskrivning behövs och vad den tillför. Det är även viktigt att hålla diagrammet uppdaterat genom hela utvecklingsarbetet, annars fyller det ingen funktion. Ett inaktuellt diagram finns det ingen användning för. Ibland görs diagram i onödan och först efter några veckor upptäcks att diagrammet inte tillför något. Då tas det bort och hänger inte kvar bara för att man gjort det.

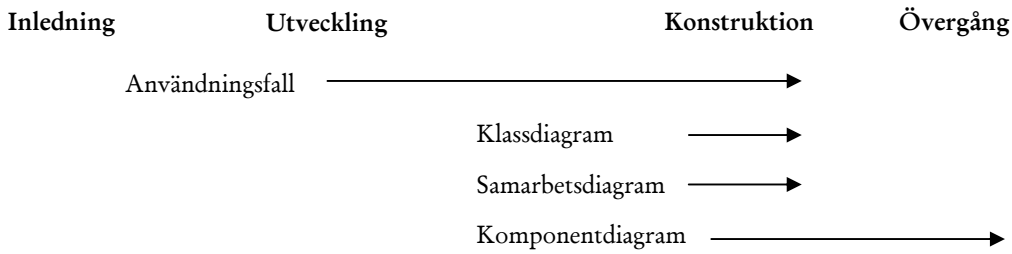
Användningsfall och klassdiagram används dock nästan uteslutande i samtliga projekt. Att vissa diagram inte används beror på att de inte behövs för just det projektet. Det har inte att göra med bristfällig modelleringsteknik eller att de är svåra att förstå. Det är nästan aldrig aktuellt att använda alla diagram.

4.4.6 UML:s plats i systemutvecklingsprocessen

Florvik anser att det är svårt att placera i vilka faser i systemutvecklingsarbetet olika diagram befinner sig. Alla diagram följer egentligen med i alla faser, vilket är ett resultat av den iterativa systemutvecklingsmodell de arbetar efter. Däremot kan Florvik placera ut i vilken eller vilka faser det ligger störst fokus på de mest använda diagrammen (Figur 4.5).

I Figur 4.5 visas att användningsfall fokuseras mest på i de två första faserna inledning och utveckling. Även i konstruktionsfasen kommer användningsfallen i fokus då de utgör grunden för testning. Under utvecklingsfasen modelleras klassdiagram, samarbetsdiagram och komponentdiagram samt eventuella andra diagram som är nödvändiga för just detta projekt. Alla dessa diagram kommer igen i konstruktionsfasen då de ligger till grund för själva konstruktionen.

Kraven kan ändras under arbetets gång men ska vara relativt stabila i konstruktionsfasen. Det är användningsfallen som utgör kraven och används för att diskutera med kunden. De andra diagrammen stödjer mer systemutvecklaren i konstruktionsarbetet. Förändringar och nya krav som uppkommer vid testning gör att iterationscykeln börjar om och genomgår alla faser ännu en gång.



Figur 4.5 Systemutvecklingsmodell för WM-data

5 Analys

I följande kapitel kommer den information vi samlat in under intervjuerna att jämföras och analyseras för att ge svar på de frågeställningar vi definierat. Vi kommer även att väva in intervjumaterialet med data från referensramen. Analysen är indelad i två delar då frågeställningarna utgår från två olika synvinklar. Under den första delen besvaras två av frågeställningarna och under den andra delen resterande tre.

5.1 Varför företag använder sig av UML

I problemdiskussionen ställer vi ett antal frågor om modellering, valet av UML samt vilka erfarenheter och uppfattningar som existerar om UML. I det här avsnittet kommer vi således att analysera svaren på frågeställningarna angående varför företag modellerar, anledningen till valet av UML som modelleringsspråk* samt vilka styrkor och svagheter som identifierats i UML.

5.1.1 Företagens syn på modellering

Samtliga studerade företag är eniga om att modelleringen skapar en bra dokumentation av systemet då den beskriver systemets krav samt hur systemet fungerar. Jordbruksverket och WM-data menar även att modellering görs då det resulterar i ett system med hög kvalitet. Jordbruksverket och WM-data är även överens om att modellering underlättar arbetet för en utomstående part som ska sätta sig in i hur systemet är uppbyggt för att till exempel hantera förvaltning och drift. Denna åsikt delas av Apelkrans och Åbom (2001) som menar att en kartläggning av verksamheten underlättar kommunikation mellan parter, användare och utvecklare (se avsnitt 3.1). Jordbruksverket anser dessutom att modellering förenklar kommunikationen med kunden då de menar att det är lättare att kommunicera genom en analysmodell än genom en text. Pdb poängterar att modellering är viktigt då de arbetar mycket med systemintegration vilket kräver hög kvalitet på design och dokumentation.

En slutsats som kan dras utifrån detta är att den främsta orsaken till att företagen modellerar är att det skapar en bra dokumentation vilken även kan ligga till grund för både analys av systemet samt fortsatt drift.

5.1.2 Orsaker till valet att tillämpa UML

Kontaktpersonerna på alla fyra företagen är förespråkare av UML och ser därför positivt på att respektive företag valt att arbeta med UML. Både hos Jordbruksverket och hos WM-data infördes UML efter utredningar av deras dåvarande systemutvecklingsarbete där de kom fram till att deras tidigare arbetsformer inte var kompletta. Zanoni och Audy (2004) poängterar svårigheterna med att hitta ett modelleringsspråk* som passar bra till alla projekt och anser därför att UML, som kan anpassas till många olika projekt, är ett bra val (se avsnitt 3.4). En annan anledning för WM-data att införa UML var att det är ett standardiserat modelleringsspråk och att det passar bra ihop med RUP. Zanoni och Audy (2004) menar, liksom WM-data, att UML medför standardiserad kommunikation mellan klienter och projektmedlemmar samt att

UML går bra ihop med RUP (se avsnitt 3.2). Även Kruchten (2000) och Scott (2002) hävdar att RUP och UML är en bra kombination då RUP är utvecklat för att passa ihop med UML samt involverar användningen av UML-baserade diagram (se avsnitt 3.1.2). Saab Combitech Systems såg också standardiseringen av UML som en bra anledning till att börja använda just detta modelleringsspråk. De såg även möjligheten i UML att få det bästa från tidigare modelleringsspråk samlat i ett nytt och standardiserat språk. Pdb:s beslut om att införa UML grundar sig på att de inte vill bli personberoende på företaget. UML innebar lösningen då det är en standard som många känner till. Dessutom anser de att UML ger en bra dokumentation för förvaltning, drift och vidareutveckling av system. Pdb valde UML som ett komplement för att beskriva komplexa flöden som tidigare beskrivits i text.

Anledningen till införandet av UML skiljer sig en del från företag till företag men de flesta är ändå eniga om att huvudanledningen till införandet av UML var att det är ett standardiserat modelleringsspråk. Att användarna tog till sig UML på grund av standardiseringen påvisar att skaparna till UML tog rätt beslut när de strävade efter att skapa ett standardiserat modelleringsspråk som är tillgängligt för alla (se avsnitt 3.5).

5.1.3 Styrkor respektive svagheter med UML

Många styrkor med UML har identifierats och ett flertal åsikter delas av företagen. Alla nämner till exempel styrkan i att UML är en standard som kan användas för att kommunicera och diskutera med många parter eftersom det är många som känner till den syntax* som används. WM-data hävdar även att kunden ofta har kunskaper om UML och kan delta effektivare i utvecklingsarbetet vilket leder till att felaktigheter och brister kan identifieras på ett tidigt stadium. Möjligheten att tidigare hitta brister är något som även Jordbruksverket och Saab Combitech Systems håller med om. Jordbruksverket anser att UML gör det lättare att kvalitetsgranska dokument och Saab Combitech Systems anser att UML ger en bättre design än andra arbetssätt samt att det är lättare att analysera analysmodeller än att gå igenom koden för att hitta brister. Zanoni och Audy (2004) menar att UML har ett klart och tydligt sätt att förmedla krav vilket kan bidra till att färre brister uppstår i systemet (se avsnitt 3.6.1). Att kraven blir lättare att förstå är något som även Beckworth (2001) tar upp som en styrka vid utveckling av realtidssystem med hjälp av UML (se avsnitt 3.2).

Jordbruksverket, WM-data och Saab Combitech Systems är även överens om vikten av att de verktyg som stödjer UML är bra och poängterar särskilt möjligheten att generera kod som en stor styrka. De tre företagen anser att det idag finns ett relativt bra urval av verktyg som stödjer UML men att det finns brister och att det är viktigt att verktygen följer med i utvecklingen av UML. Även Pdb påpekar vikten av att hitta rätt verktyg men de anser att många verktyg är för komplicerade och att de enkla verktygen i sin tur inte stödjer en hel systemutvecklingsprocess. Generering av kod är enligt Pdb ett riskabelt moment i UML och de är fortfarande tveksamma till detta. Dokumentationen är dock något som Pdb och WM-data anger som positivt med UML. De anser att UML:s dokumentation bidrar till ett bättre förvaltningsarbete. Även Jordbruksverket anser att dokumentationen i UML är bra. Anledningen till detta är att de inom företaget kan arbeta på ett enhetligt sätt i alla projekt. Jordbruks-

verket menar även att det är en styrka att kunna skapa egna diagram i UML. Pdb och Saab Combitech Systems anser avslutningsvis att UML lyfter systemet till en högre nivå och därmed skapar bättre förståelse då högnivådesign ligger nära hur vi människor tänker.

Svagheterna med UML är inte så många som styrkorna och åsikterna skiljer sig åt mellan företagen. Jordbruksverket menar att den största svagheten är att UML är så nytt att mycket få människor lärt sig nyttja UML praktiskt på ett bra sätt. Att UML inte innehåller några processdiagram är en ytterligare brist enligt både Jordbruksverket och Saab Combitech Systems. De menar att deras arbete skulle underlättas om de i UML kunde se kopplingen mellan processer* och objekt*. WM-data anser att en svaghet med UML är svårigheten att bibehålla klassdiagrammets koppling till databasen då klassdiagrammet fokuserar på objekt. De ser även ett hot i att använda syntaxen för slaviskt då det i många fall inte är nödvändigt utan endast tidskrävande. Pdb har också uppfattningen att UML är tidskrävande och därmed ofta olämpligt i mindre projekt vilket gör det viktigt att kunna ha en balansgång mellan ”tid och nytta”. Detta stämmer med vad Zanoni och Audy (2004) anser, att detaljer endast bör inkluderas då det är speciellt nödvändigt (se avsnitt 3.6). Pdb anser det också vara ett problem att det i olika verktyg kan skilja något i syntaxen och de önskar att verktygen blev mer standardiserade. Saab Combitech Systems håller med och menar att det fortfarande saknas stöd i verktygen för koppling mellan kod och analysmodell samt att UML:s generella natur och avsaknad av metodik gör det svårt att avgöra vad som ska användas till ett specifikt projekt. Detta visar att flexibiliteten i UML inte utnyttjas och att företagen hellre vill ha tydligare riktlinjer för modelleringsarbetet. Pdb menar att faktumet att UML är svårt att använda i funktionella miljöer gör det svårt för dem att synkronisera tillämpningen av UML inom företaget.

Åsikterna om UML:s styrkor skiftar mellan de olika företagen men en styrka delad av samtliga företag är att UML är en standard som många känner till. Andra åsikter delas mellan några företag och flera åsikter kan även härledas till företagets specifika omgivning som till exempel deras arbetssätt. Svagheterna är också skilda åt och de flesta svagheter går att härleda till företagets erfarenheter och arbetssätt såsom att Pdb nämner att UML brister då det handlar om funktionell utveckling. Det kan sägas vara ett resultat av att de ofta kommer i kontakt med denna typ av utveckling och saknar stöd för detta i UML.

Verktyg som stöd för systemutvecklingen har visat sig vara något som alla företag lägger stor vikt vid, men som de inte är riktigt nöjda med. Utbudet av verktyg är stort men de stödjer inte samma syntax och många stödjer inte heller generering av kod vilket är av intresse för samtliga företag. Dessutom anses vissa verktyg vara komplicerade eller ofullständiga. Vi har däremot förstått att om verktygen används på rätt sätt kan de vara mycket användbara och tidssparande. Generering av kod utifrån modeller är något som specifikt nämns som en tidssparande aktivitet. En uppfattning vi fått utifrån detta är att verktygen både behöver förbättras och förenklas. Ett resultat av förbättrade verktyg tror vi kan bli att fler företag börjar använda sig av UML samt att själva arbetet med att införa UML blir enklare då verktygen ger ett bättre och enhetligare stöd.

5.2 Företagens sätt att använda UML

I följande avsnitt behandlas hur företagen arbetar med UML och därmed analyseras svaren på frågeställningarna angående hur UML kombineras med olika systemutvecklingsmodeller, vilka diagram som används i projekten, om anpassningar görs av diagrammen samt om de används i alla projekt.

5.2.1 Användning av UML på företagen

Då det objektorienterade synsättet och objektorienterad modellering är ett relativt ungt område har det inte varit självklart för företagen att använda sig av modellering innan UML släpptes 1997. Detta kan vara en förklaring till skillnader mellan företagen och de olika sätt att modellera eller inte modellera, innan införandet av UML på företagen. Vi har utifrån dessa fyra företag fått erfara att tidigare arbetssätt varierat. Det går dock att dela upp företagen i två olika grupper varav den ena gruppen, som består av WM-data och Saab Combitech Systems, har lång erfarenhet av ett objektorienterat arbetssätt och har tagit del av utvecklingen inom detta område under en längre tid. Dessa har således arbetat med UML:s föregångare och var därmed bekanta med UML innan införandet av detsamma. Pdb och Jordbruksverket har däremot inte denna bakgrund och det objektorienterade området samt grafiska beskrivningar är relativt nytt för dem.

Vid tillämpningen av UML menar Eriksson et al. (2004) att det råder stor variation i antalet och vilka diagramtyper som används i projekt (se avsnitt 3.6). De menar även att vissa egenutvecklade diagramtyper kan förekomma vilket stöds av vår studie. Då företagen har olika erfarenheter av UML och är i olika stadier av användandet skiljer det sig till viss del hur företagen arbetar med UML idag vilket visas i Tabell 1. Jordbruksverket och Pdb menar att de fortfarande är i utvecklingsfasen av att arbeta med UML. De använder idag vissa UML-diagram som de är nöjda med och diskussioner om och hur de ska gå tillväga vid införandet av flera diagram råder just nu på företagen. Båda företagen samt Saab Combitech Systems använder idag något eller några andra diagram utöver UML-diagrammen i sitt modelleringsarbete. Jordbruksverket och Pdb anser att de anställda behöver mer utbildning av UML innan vidare steg kan tas. Samtliga företag menar att UML är ett omfattande och komplext modelleringsspråk* men med stora möjligheter, då det bland annat kan anpassas efter många olika typer av system.

WM-data och Saab Combitech Systems har kommit längre i sitt användande av UML och som vi kan se i Tabell 1 nedan använder de sig av nästan samtliga diagram i UML, dock mer eller mindre frekvent. De diagram som samtliga företag använder sig av idag och som visas i Tabell 1 är användningsfallsdiagram, klassdiagram och sekvensdiagram. De som använder både sekvensdiagram och samarbetsdiagram nämner att det ofta är fråga om att använda en utav dessa beroende på typ av projekt. Det här kan förklaras av Oestereich (2002) som nämner att sekvensdiagram och samarbetsdiagram beskriver samma sak men ur olika synvinklar (se avsnitt 3.6.7). En anledning till varför just ovan nämnda diagram används i samtliga företag och dessutom i nästan alla projekt kan, som WM-data och Saab Combitech Systems nämner, vara ett resultat av att dessa diagram är grundläggande och underlättar systemutvecklingsarbetet på

många olika sätt. Eriksson et al. (2004) samt Fowler och Scott (2000) nämner specifikt klassdiagrammet som det mest grundläggande i UML (se avsnitt 3.6.2). Att användningsfallsdiagram används ofta av samtliga kan även vara en följd av att användningsfall enligt Fowler och Scott (2000) beskriver kundens funktionella krav, vilket då inte behöver göras i en separat kravspecifikation (se avsnitt 3.6.1).

Förklaringen till att Saab Combitech Systems, till skillnad från de andra företagen, använder tillståndsdiagram i alla projekt är att Saab Combitech Systems jobbar med utveckling av realtidssystem där det är mycket viktigt att analysera hur ett objekt* reagerar på en viss aktivitet* samt vilka olika tillstånd objektet kan hamna i.

Tabell 1 Diagrammens användning av respektive företag

Diagramtyp	Jordbruksverket	Pdb	Saab Combitech Systems	WM-data
Användningsfallsdiagram	I alla projekt	I flertalet projekt	I flertalet projekt	I flertalet projekt
Klassdiagram	I flertalet projekt	Inom Java-teamet	I alla projekt	I flertalet projekt
Objektdiagram			I flertalet projekt	
Sekvensdiagram	I flertalet projekt	I flertalet projekt	I flertalet projekt	Ibland
Samarbetsdiagram			I flertalet projekt	I flertalet projekt
Tillståndsdiagram			I alla projekt	Ibland
Komponentdiagram			I flertalet projekt	I flertalet projekt
Fördelningsdiagram			I flertalet projekt	Ibland
Aktivitetsdiagram	Ibland			Ibland

Företagen har olika planer för framtiden lite beroende på var de befinner sig idag, hur mycket de använder sig av UML, samt vilka kunder de har. Sammanfattningsvis kan vi säga att alla företagen oberoende vart de står idag är intresserade av UML:s utveckling och vill gärna ta till sig UML 2.0. Företagen ser även fram emot hur utvecklingen med kodgenerering från analysmodell kommer att gå samt vill gärna utveckla användningen av detta i framtiden.

5.2.2 Diagrammens plats i systemutvecklingsprocessen

Vilken utvecklingsmodell företagen arbetar med bestäms ofta av kunden eftersom denne är delaktig i projekten. Zanoni och Audy (2004) påpekar vikten av att den valda systemutvecklingsmodellen underlättar kommunikationen mellan projektmedlemmar (se avsnitt 3.2) men även på svårigheten att välja, anamma och integrera rätt systemutvecklingsmodell och metod så att de passar omgivningen (se avsnitt 3.4). Det har även framkommit att egenutvecklade systemutvecklingsmodeller är vanliga, vilka i någon utsträckning är influerade av andra utvecklingsmodeller. Zanoni och Audy (2004) menar att RUP (se avsnitt 3.1.2) är ett mycket bra val av systemutvecklings-

modell inom det objektorienterade synsättet (se avsnitt 3.7). WM-data använder sig nästan uteslutande av RUP och både Saab Combitech Systems egenutvecklade modell Parts (Figur 4.4) och Pdb:s utvecklingsmodell PUM baseras på RUP. PUM innehar dock även egenskaper från livscykelmodellen (se avsnitt 3.1.1) och kan liknas vid en iterativ livscykelmodell. Jordbruksverkets utvecklingsmodell baseras helt på livscykelmodellen men vissa konsulter arbetar dock enligt RUP. Även på Saab Combitech Systems förekommer tillfällen då RUP används i sin helhet.

Saab Combitech Systems och WM-data använder sig båda av de faser som finns i RUP men vi kan genom en jämförelse av Saab Combitech Systems systemutvecklingsmodell (Figur 4.4) och WM-datas systemutvecklingsmodell (Figur 4.5) se att fokus på diagrammen skiljer sig något. Vi kan dessutom genom att titta på Figur 3.12, vilken anger UML:s koppling till RUP enligt Scott (2002) och Kruchten (2000), se att det även här finns skillnader gentemot WM-data och Saab Combitech Systems koppling mellan RUP och UML-diagrammen. De skillnader som kan identifieras är små och kan förklaras genom att påpeka det iterativa arbetssättet som används i RUP (se avsnitt 3.1.2) samt företagens uttalande om svårigheterna att placera diagrammen i specifika faser då samtliga diagram i själva verket följer med under hela systemutvecklingsprocessen. Att använda UML genom hela systemutvecklingsprocessen är enligt Zanoni och Audy (2004) mycket viktigt för att inte tappa kontrollen över projektet (se avsnitt 3.7). Saab Combitech Systems och WM-data förklarar vidare att nästan alla diagram finns med under varje fas men att det fokuseras och arbetas *olika mycket* med dem beroende på vilken fas de befinner sig i.

Jordbruksverkets egen systemutvecklingsmodell som de använder sig av idag är lik livscykelmodellen och dess faser (se avsnitt 4.1). Eftersom Jordbruksverket idag endast använder ett fåtal av UML:s diagram och ett stort antal andra diagram (se Figur 4.2) är det svårt att bedöma hur Jordbruksverket kopplar UML till sin utvecklingsmodell. Vid en jämförelse av Jordbruksverkets systemutvecklingsmodell och Andersens syn på livscykelmodellens koppling till UML (se avsnitt 3.7) kan vi dock se att UML:s plats i utvecklingsmodellen skiljer sig till viss del. Pdb:s systemutvecklingsmodell kan liknas med både RUP och en iterativ livscykelmodell vilket gör det svårt att göra en jämförelse. Pdb menar dock att de använder UML främst i utformningsfasen av livscykelmodellen.

Vi kan se att kunden har stort inflytande över vilken utvecklingsmodell som används samt att det är vanligt att använda egna företagsanpassade versioner av olika systemutvecklingsmodeller. RUP används i stor utsträckning då samtliga företag använder RUP eller utvecklingsmodeller som baseras på RUP. Det tyder på att RUP anses lämplig att kombineras med UML samt att den passar bra för många olika företag, projekt och kunder. Kruchten (2000) hävdar att RUP är en populär utvecklingsmodell som är anpassad till UML och dess mångsidighet gör att den passar många olika typer av projekt (se avsnitt 3.1.2). Även Scott (2002) poängterar RUP:s starka koppling till UML. Hur UML kombineras med olika systemutvecklingsmodeller skiljer sig dock mellan företagen och även från de rekommendationer som ges av hur UML ska kombineras med livscykelmodellen (Figur 3.13) samt RUP (Figur 3.12). Företagen har dock poängterat svårigheten med att placera diagrammen i specifika faser då de egentligen följer med under hela systemutvecklingsprocessen.

5.2.3 Anpassning av UML samt tillämpning i olika projekt

Jordbruksverket, Pdb och Saab Combitech Systems hävdar att de använder syntaxen* precis som den anges i UML bortsett från de skillnader som de olika verktygen för med sig. Då Jordbruksverket ännu inte har något tillfredställande verktyg att jobba med är detta inte ett problem för dem och de anser även att UML är ett regelverk där det inte finns rum för förändringar. Denna åsikt motsäger Eriksson et al. (2004) som menar att UML har flexibiliteten att utvecklas, förändras och kombineras med egenutvecklade delar (se avsnitt 3.4). I framtiden kan Jordbruksverket dock tänka sig att vissa tillägg kommer att göras för att anpassa UML till deras sätt att arbeta. På WM-data förekommer vissa förändringar i syntaxen. Ett exempel är att en heldragen pil i vissa fall används för alla relationer när typ av relation saknar betydelse. De inser dock vikten av att vara noggrannare med syntaxen då den ska användas för att generera kod.

Alla företagen är överens om att vilka diagram som används är beroende på projektets typ och storlek. De delar även uppfattningen att det är viktigt att modellera endast de diagram som tillför något till projektet. Saab Combitech Systems hävdar dock att många av diagrammen är nödvändiga i alla projekt oberoende av typ eller storlek. WM-data menar att de sällan använder alla diagram i samma projekt och Jordbruksverket menar att det ofta är samma diagram som används, men detaljeringsgraden skiftar beroende på projektets typ. De anser även att en viktig beslutsfaktor är att bestämma vilken dokumentation som behövs för framtida drift. Även Zanoni och Audy (2004) menar att dokumentationen är en viktig del i systemutvecklingsprocessen och de anser att UML ger en mycket bra dokumentation och är därför ett bra val av modelleringsspråk* (se avsnitt 3.2). Inom Pdb beror beslutet om aktuella diagram på vilket team som har hand om projektet då olika team är mer eller mindre inriktade på objektorienterad systemutveckling. Att diagram utelämnas beror enligt alla parter på att de inte ger något i det specifika fallet och således inte på någon upptäckt brist i UML.

Sammanfattningsvis kan vi säga att användningen av diagram beror på projektets typ och storlek. Varje använt diagram måste också tillföra något till projektet. I övrigt råder olika uppfattningar om vad som används. Vissa anser att nästan alla diagram är nödvändiga mer eller mindre detaljerade medan andra anser att endast ett fåtal diagram används i varje projekt. Även faktorer såsom företagsstruktur kan ha inverkan på vilka diagram som är aktuella att modellera. Trots den flexibla naturen hos UML är det inget av företagen som gör några större anpassningar i syntaxen. En slutsats av detta är att UML:s syntax anses enkel och bra att använda för modellering.

6 Slutsatser

I detta kapitel kommer vi att presentera våra slutsatser av studien. De slutsatser som dragits är ett resultat av den analys som gjorts.

- Den viktigaste anledningen, för samtliga företag i studien, till att modellera är dokumentationen. De anser att modellering skapar bra dokumentation vilket underlättar utvecklingen och förvaltningen av ett system.

- Att UML är ett standardiserat modelleringsspråk* är enligt studien en viktig eller avgörande faktor vid valet av UML som modelleringsspråk. Standardiseringen ses därmed även som den främsta styrkan med UML. I övrigt skiljer sig de avgörande faktorerna åt mellan företagen men några exempel är att UML lämpar sig för många olika projekt samt att UML passar den systemutvecklingsmodell som tillämpas inom företaget.

- Majoriteten av företagen anser att UML ger en bra dokumentation samt är ett bra hjälpmedel för att så tidigt som möjligt identifiera fel och brister i ett system. Ytterligare en styrka som nämns av huvudparten är möjligheten att generera kod utifrån analysmodeller med hjälp av verktyg. De poängterar dock att generering av kod kräver ett bra verktyg och en standardiserad syntax vilket de hoppas på i framtiden.

- Det har uppkommit önskemål om processdiagram i UML samt en standardisering av syntaxen* i de verktyg som stödjer UML. Andra identifierade brister med UML är dess generella natur och svårighet att avgöra vad som ska användas till ett specifikt projekt samt att UML anses tidskrävande vid mindre projekt. Flera brister kan relateras till företagens erfarenheter och arbetssätt.

- Användningsfallsdiagram, klassdiagram samt sekvensdiagram används frekvent av samtliga företag. Dessa diagram kan därmed anses vara grundläggande vid modellering av system. Användningen av övriga diagram är beroende på typ och storlek av projekt samt företagens kunskaper och erfarenheter om UML. I allmänhet använder de företag som arbetat med UML under en längre tid fler diagram.

- Flexibiliteten i UML utnyttjas genom att företagen använder diagrammen olika beroende på projekt. Däremot är det inget av företagen som gör några större anpassningar av syntaxen. Det har snarare visat sig att företagen önskar flera fasta riktlinjer för både syntax och metodik.

- Genom studien har RUP kunnat identifieras som den systemutvecklingsmodell som används mest frekvent tillsammans med UML. Hur diagrammen i UML kombineras med systemutvecklingsmodellerna skiljer sig mellan företagen och även från rekommendationer av hur UML ska kombineras med olika systemutvecklingsmodeller. De olikheter som framkommit kan relateras till det iterativa sätt företagen arbetar efter, där diagrammen följer med i hela systemutvecklingsprocessen.

7 Avslutande diskussion

I följande kapitel kommer reflektioner kring uppsatsen att diskuteras samt förslag till fortsatta studier att ges. Kapitlet avslutas därefter med ett tack till alla som hjälpt till att genomföra uppsatsen.

7.1 Reflektioner

Genom vår studie har vi förstått att UML är ett utbrett modelleringsspråk* men kanske inte i den grad som vi tidigare trott. Vi hade även förväntat oss ett större utnyttjande av den fria användning som UML ger möjlighet till, det vill säga att företag i större utsträckning skulle ha ändrat bland annat i diagrammens syntax*. Vi har istället kunnat se ett önskemål om en standardiserad syntax för UML och dess verktyg då företag tycker att det är ett problem med olikheter. Det vi inte förväntade oss skulle ha en sådan stor påverkan på användandet av UML är verktyg och kunder. Det har uppkommit att verktygen är viktiga och vi tror att bättre verktyg kan leda till fler UML-användare. Det har även framkommit att det ofta är kunden som avgör vilken utvecklingsmodell som ska användas. Sammanfattningsvis anser vi att studien har varit intressant och givande då vi lärt oss mycket som kan vara till nytta både för vidare studier samt i arbetslivet.

För insamling av data använde vi oss av semistrukturerade intervjuer vilket vi upplevt vara ett bra alternativ vid en studie som denna då ämnet krävt att vi ställt följdfrågor samt haft diskussioner kring vissa frågor och funderingar. Vi tror inte att standardiserade intervjuer eller en enkätundersökning hade varit lämpligt då vi blivit låsta kring de frågorna som formulerades från början. Vi anser därmed att metoden varit bra men i efterhand hade vi gärna intervjuat fler personer ute på företagen, då detta antagligen hade gett en mer rättvis bild av UML. Inställningen till UML bland respondenterna var positiv och ytterligare intervjuer hade kanske kunnat fånga in fler typer av åsikter. Då det i vissa fall kan vara svårt att förklara ett visst agerande eller tillvägagångssätt kunde det även varit av intresse för studien att se hur företagen använder UML i praktiken det vill säga att följa hur UML tillämpas i ett systemutvecklingsprojekt. Detta hade dock inte varit möjligt för vår studie då ett systemutvecklingsprojekt oftast sträcker sig över en längre period.

Då vi i början av studien hade svårt att begränsa oss och ville täcka in så många delar som möjligt resulterade detta i ett frågeunderlag som var för omfattande för studien. Vi granskade frågeunderlaget innan intervjuerna och ansåg då att frågorna höll sig inom ämnet. I efterhand har vi dock förstått att vissa frågor kunde ha utelämnats. Till följd av detta resulterade vissa frågor i svar vi senare inte använt oss av i resultatet eller analysen. Vi tror att en pilotundersökning kunde ha hjälpt oss att specificera frågorna lite bättre och därmed eventuellt lett till mer precisa svar. Det positiva med det breda spektrum av frågor som använts är dock att vi fått tillgång till intressant information som gett oss en bättre bild av företagen och dess relation till UML.

En del som vi saknat i arbetet är en mer utförlig beskrivning av Pdb:s systemutvecklingsmodell. Detta för att kunna relatera diagrammen bättre till systemutvecklings-

modellen och därmed även kunna jämföra deras arbetssätt med de övriga företagen på ett mer ingående sätt. Anledningen till den ofullständiga kopplingen mellan PUM och UML-diagrammen är bristen på information kring dem. Vi ansåg oss under intervjutillfället ha en god uppfattning om hur PUM fungerar och hur den kombineras med olika UML-diagram. Det visade sig dock att så inte var fallet och vi har dessutom haft svårigheter att få den kompletterad från Pdb. Vi anser att en tydligare bild av PUM hade genererat en bättre analys av problemfrågan: ”Hur kombineras UML med olika systemutvecklingsmodeller?”

7.2 Förslag till fortsatta studier

Något som diskuterats med alla respondenter är generering av kod utifrån analysmodeller i UML. Inställningen till denna funktion skiljer sig mellan företagen men majoriteten är positiva till att använda sig av generering av kod utifrån analysmodeller. Än så länge pågår utvecklingen av denna funktion och det kan vara orsaken till att respondenterna ännu inte litar fullständigt på denna typ av programmering. De menar att generering av kod sparar mycket tid men att nackdelen ligger i minskad kontroll över koden. De verktyg som finns idag klarar generering av kodskelett från UML-baserade analysmodeller men inte mer detaljerad kod. Vi anser att denna utveckling skulle vara intressant att följa då den inom en snar framtid kan ge systemutvecklare goda möjligheter att effektivisera sitt arbete. Studien skulle fokusera på utvecklingen av verktyg för kodgenerering och hur företagen, och kanske i synnerhet programmerare, ställer sig till att använda dessa verktyg. En annan intressant infallsvinkel är att undersöka vilken effekt generering av kod från analysmodell får på själva modelleringen. Går det att skapa modeller som är tillräckligt korrekta för att använda till mer detaljerad kodgenerering?

En annan intressant studie skulle vara att fortsätta följa utvecklingen på de två företag som i denna studie fortfarande håller på med införandet av UML. Hur kommer de att använda sig av UML i framtiden och har åsikterna om UML ändrats med tiden? Att återigen göra en jämförelse mellan de fyra företagen i denna studie när samtliga har skaffat sig erfarenhet av modelleringsspråket* kan ge intressanta resultat. Dessa resultat skulle sedan kunna analyseras och jämföras med de resultat och slutsatser som redovisats i denna studie för att se hur åsikter och arbetssätt har ändrats med tiden.

En annan infallsvinkel vid en studie av de företag som fortfarande är i fasen att införa UML är att följa arbetet med införandet. Det är snart dags för OMG att släppa den slutliga versionen av UML 2.0 och det skulle vara intressant att se hur företagen fortsätter sitt arbete med att införa UML. Kommer de att anamma denna nya version av UML eller fortsätta med den version de börjat implementera? Hur stora är skillnaderna mellan versionerna, och vad skulle det betyda för arbetet med att införa ett nytt modelleringsspråk om de bestämde sig för att under införandet byta till att införa en nyare version av UML?

7.3 Tack

Slutligen skulle vi vilja tacka följande respondenter, handledare och övriga personer som bidragit till att genomföra denna studie. All medverkan, intervjuer såväl som goda råd, uppskattas mycket av författarna.

- Leif Bengtzohn på Jordbruksverket.
- Mathias Rehnström på Pdb.
- Anders Mattsson på Saab Combitech Systems.
- Jonas Florvik på WM-data.
- Britt-Marie Johansson, vår handledare på Internationella Handelshögskolan i Jönköping.
- Ulf Larsson, adjunkt vid Internationella Handelshögskolan i Jönköping.
- Opponentgrupper och övriga handledare på Internationella Handelshögskolan i Jönköping som gett oss viktig feedback under arbetets gång.
- Övriga personer som hjälpt oss med bland annat korrekturläsning och goda råd.

Referenslista

Andersen, E. S. (1994). *Systemutveckling – principer, metoder och tekniker*. Lund: Studentlitteratur.

Apelkrans, M., & Åbom, C. (2001). *OOS/UML- En objektorienterad systemutvecklingsmodell för processorienterad affärsutveckling*. Lund: Studentlitteratur.

Bahrami, A. (1999). *Object Oriented Systems Development-using the Unified Modeling Language*. Singapore: McGraw-Hill Companies.

Beckworth, T. (2001, 21 Mars). Using UML in the Development of OO real-time systems - part 2. *Electronic Engineering*. Hämtad 2005-01-10, från http://proquest.umi.com/pqdweb?index=13&did=70381371&SrchMode=1&sid=4&Fmt=3&VInst=PROD&VType=PQD&RQT=309&VName=PQD&TS=1104667719&clientId=17918

Booch, G. (1998, Februari). The Visual Modeling of Software Architechture for the Enterprise. *MSDN*. Hämtad 2005-01-12, från http://msdn.microsoft.com/library/default.asp?url=/archive/en-us/dnaruml/html/msdn_visualmod.asp

Christenssen, L., Andersson, N., Carlsson, C., & Haglund, L. (1998). *Marknadsundersökning – en handbok*. Lund: Studentlitteratur.

Eriksson, H-E., & Penker, M. (1998). *UML Toolkit*. New York: Wiley Publishing Inc.

Eriksson, H-E., Penker, M., Lyons, B., & Fado, D. (2004). *UML 2 Toolkit*. Indianapolis: Wiley Publishing Inc.

Eriksson, L. T., & Wiedersheim-Paul, F. (1991). *Att utreda, forska och rapportera*. Malmö: Lieber-Hermods.

Fowler, M., & Scott, K. (2000). *UML Distilled- A Brief Guide to the Standard Object Modeling Language* (2:a uppl.). Ontario: Addison Wesley Longman Inc.

Goldkuhl, G., & Röstlinger, A. (1988). *Förändringsanalys: Arbetsmetodik och förhållningssätt för goda förändringsbeslut*. Lund: Studentlitteratur.

Hanscome, B. (2000, Mars). Focus on UML. *Software Development Magazine*. Hämtad 2004-11-01, från http://www.sdmagazine.com/documents/s=815/sdm0003z/

Jordbruksverket. (2004). *Om Jordbruksverket*. Hämtat 2004-11-26, från http://www.sjv.se/startsida/omjordbruksverket.4.7502f61001ea08a0c7fff122625.html

Kobryn, C. (2002, Januari). Will UML 2.0 be Agile or Awkward?. *Communications of the ACM*. Hämtad 2004-12-09, från http://www.uml-forum.com/docs/papers/CACM_Jan02_p107_Kobryn.pdf

Kruchten, P. (2000). *The Rational Unified Process – An Introduction* (2:a uppl.). New Jersey: Addison Wesley.

Lekvall, P., & Wahlbin, C. (1993). *Information för marknadsföringsbeslut*. Göteborg: IHM Förlag.

Lundahl, U., & Skärvad, P-H. (1999). *Utredningsmetodik för samhällsvetare och ekonomer* (3:e uppl.). Lund: Studentlitteratur.

Moore, A. (2001, Juni). Extending UML to Enable the Definition and Design of Real-Time Embedded Systems. *Crosstalk-The Journal of Defense Software Engineering*. Hämtad 2004-12-09, från http://www.stsc.hill.af.mil/crosstalk/2001/06/moore.html

Object Management Group. (2004, Mars). *UML Resource Page*. Hämtad 2004-12-09, från http://www.uml.org/

Oestereich, B. (2002). *Developing Software with UML- Object-Oriented Analysis and Design in Practice* (2:a uppl.). London: Pearson Education Limited.

Patel, R., & Tebelius, U. (1987). *Grundbok i forskningsmetodik*. Lund: Studentlitteratur.

Rational Software Corporation. (2000). *The UML and Data Modeling*. Hämtad 2004-11-01, från http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Tp180.PDF

Saab Combitech Systems. (2004, 24 Mars). *Branshinriktning*. Hämtad 2004-11-29, från http://www.combitechsystems.com/node1954.asp

Scott, K. (2002). *The Unified Process Explained*. Indianapolis: Pearson Education Inc.

Svensson, P-G., & Starrin, B. (1996). *Kvalitativa studier i teori och praktik*. Lund: Studentlitteratur.

Trost, J. (2001). *Enkätboken* (2:a uppl.). Lund: Studentlitteratur.

WM-data. (2004). *Om Oss*. Hämtat 2004-11-26, från http://www.wmdata.se/wmwebb/content.asp?

Zanoni, R., & Audy, J. L. N. (2004, Juni). Project Management Model: Proposal for Performance in a Physically Distributed Software Development Environment. *Engineering Management Journal*. Hämtad 2004-12-15, från http://proquest.umi.com/pqdweb?index=0&did=661301301&SrchMode=1&sid=1&Fmt=4&VInst=PROD&VType=PQD&RQT=309&VName=PQD&TS=1104847874&clientId=17918

Bilaga 1 - Begreppsordlista

Aktivitet – En aktivitet är det arbete som utförs i ett givet tillstånd och som får objektet* att ändra tillstånd. En aktivitet är en del i en process*. En aktivitet i processen ”skriv ut order” kan vara att trycka på knappen ”skriv ut” (Apelkrans och Åbom, 2001).

Beskrivningsteknik - En beskrivningsteknik visar enligt Andersen (1994) vilka symboler som är tillåtna och i vilka kombinationer, på vilket sätt text kan kombineras med symbolerna och när och var varje symbol bör användas.

Datamodell – Visar vilka begrepp som finns i verksamheten och hur verksamhetsinformationen är strukturerad. Den vanligaste typen av datamodell idag är relationsmodellen vilken visar data i form av tabeller. Varje entitetstyp överförs till en tabell (Andersen, 1994).

Entitetsmodell – En entitetsmodell liknar ett klassdiagram och skapas utifrån användningsfallen. Den illustrerar de olika entiteterna (objekten), dess attribut och relationer men inte operationer. Även syntaxen* skiljer sig från klassdiagrammet (Andersen, 1994)

Handlingsgraf – En metod för att detaljerat beskriva flödet i en affärsprocess. Flera olika handlingsgrafer kopplas ihop för att se relationer mellan affärsprocesser (Goldkuhl & Röstlinger, 1988)

JSP (Jackson Structured Programming) – En metod för att utveckla programvara vilken har en strikt arbetsgång. Innehåller ett antal diagram (Apelkrans & Åbom, 2001).

Klass – En klass är en samling objekt med liknande egenskaper, såsom en grupp bestående av flera personer. Alla personer har egenskapen namn, men med olika värden. En kan ha värdet Kalle och en annan har värdet Anna (Andersen, 1994).

Modelleringsspråk – Ett modelleringsspråk används för att specificera, konstruera och dokumentera ett mjukvarusystem för att ge en bild av systemet som är enklare för människan att förstå. Modelleringsspråket innehåller regler för hur dessa specificeringar ska se ut och kan liknas vid en beskrivningsteknik. UML är ett exempel på ett modelleringsspråk (Apelkrans och Åbom, 2001).

Objekt – Ett objekt är en företeelse i verksamheten som har en identitet, egenskaper samt tillstånd. Ett objekt kan till exempel vara en person, en order eller en avdelning (Apelkrans och Åbom, 2001).

Process – Med process menas i denna studie en rad av sammanhängande aktiviteter. Att utföra en av objektens operationer kan vara ett exempel på en process.

Syntax – Syntaxen anger vilka begrepp och symboler som kan användas (till exempel i UML eller något programmeringsspråk) samt hur de kan kombineras (Apelkrans & Åbom, 2001).

Bilaga 2 – Intervjuunderlag

Arbetar ni endast gentemot ert eget företag eller agerar ni även konsulter åt andra?

Vilken typ av företag arbetar ni gentemot?

Allmänt UML

Hur länge har ni använt er av modelleringsspråket UML?

Använder ni er av flera modelleringsspråk än UML?

Vad använde ni er av innan ni började använda UML?

Vad är anledningen till att ni bytte till UML?

Vad har blivit bättre och vad har blivit sämre sedan bytet?

Jämför med andra modelleringsspråk, styrkor och svagheter?

Använder ni er av vyerna i UML?

Underlättar vyerna arbetet med att illustrera systemet?

Genomgång av diagram

Vilka diagram inom UML använder ni er av?

Använder ni er alltid av samma UML-diagram eller är de projektberoende?

Vad är det som avgör vilka UML-diagram ni använder er av i ett specifikt projekt?

Är det några diagram inom UML som ni aldrig använder er av?

Använder ni diagrammen exakt som de modelleras i UML det vill säga med UML:s syntax? *Om nej:*

- Vad byter ni ut och mot vad?
- Byter ni ut någon del av UML mot andra modelleringsspråk, egenkomponerade diagram eller dylikt?
- Härstammar avvikelser från UML från tidigare modelleringsspråk ni använt?

Faser

Vid systemutveckling, utgår ni från någon specifik modell såsom livscykelmodellen?

Vilken, och vilka faser ingår?

Exempel: Livscykelmodellen består utav sju olika faser, förändringsanalys, analys, design/utformning, realisering, implementation, förvaltning/drift samt avveckling. Av dessa faser är det främst i analysfasen, design-/utformningsfasen samt i realiseringsfasen som det fokuseras på UML.

Hur använder ni UML-diagrammen i samband med er utvecklingsmodell, det vill säga kan diagrammen härledas till specifika faser?

Använder ni er av UML inom alla dessa faser?

Om ni inte använder er av UML, använder ni då något annat diagram eller liknande och under vilken fas?

Övrigt

Använder ni eller har ni planer på att införa UML 2.0?

Har detta haft inverkan på val av UML som modelleringsspråk?

Är det lätt att förstå UML-diagram som någon annan har utformat?

Har ni någonting emot att vi nämner ert och/eller företagets namn i uppsatsen?

OMG och Standard

Enligt OMG är UML ett standardiserat modelleringsspråk.

Upplever ni UML som en standard (oavsett om ni byter ut delar i UML eller inte)?

Vad är anledningen till svaret på ovanstående fråga?

Vilka styrkor och/eller svagheter kan ni se med att UML är en standard?