GÖTEBORGS UNIVERSITET

Anpassad modellering

Ett ramverk för att skapa användbara modeller inom systemutveckling

Adapted Modeling

A Framework for Creating Useful Models in System Development

JIMMY ANDERSSON
SVEN AXELSSON

Kandidatuppsats i informatik

Rapport nr. 2009:030
ISSN: 1651-4769

Göteborgs universitet
Institutionen för tillämpad informationsteknologi
Göteborg, Sverige, Maj 2009

Adapted Modeling
A Framework for Creating Useful Models in System Development

JIMMY ANDERSSON
SVEN AXELSSON

Department of Applied Information Technology

IT University of Göteborg
Göteborg University and Chalmers University of Technology

SUMMARY
Models are often used as a tool for managing complex situations in system development. However, opinions regarding how models are best used differ. In this thesis, we studied perspectives in which models either are central to system development or used as a supporting resource. Our intention was never to compare these perspectives, but rather to describe how they can be combined and adapted to local conditions. To examine this, we conducted a case study at a small software development company that intends to use models in a more active way in their system development process. The study was evaluated using a focus group where we, together with employees at the company, discussed the importance of models. This resulted in a framework describing important principles for how models can be useful for other organizations in the system development industry. Several conclusions could be drawn after the study: the models should be easy to administrate, easy to keep updated and created for a specific purpose. Furthermore, the models should abstract the problem area, be created in a cost efficient manner and contribute to a sufficient description of the system. The main focus in this study was to examine how various principles for modeling can be used in a practical manner and also how the principles could be combined. This was proven applicable since our framework includes principles from several different areas.

The report is written in Swedish.

Keywords: Model, modeling, framework, principles, system development, model driven development, agile modeling.

Anpassad modellering
Ett ramverk för att skapa användbara modeller inom systemutveckling

JIMMY ANDERSSON
SVEN AXELSSON

Institutionen för tillämpad Informationsteknologi

IT-universitetet i Göteborg
Göteborgs Universitet och Chalmers tekniska högskola

SAMMANFATTNING
Modeller används ofta som ett hjälpmedel för att hantera komplexa systemutvecklingssituationer. Det råder dock delade meningar om hur man på bästa sätt bör använda modeller. I denna studie har vi tittat närmare på perspektiv där modellerna står i centrum för systemutvecklingen respektive används som ett hjälpmedel för densamma. Vår avsikt har inte varit att jämföra dessa, utan snarare beskriva hur de kan kombineras och anpassas till lokala förhållanden. För att undersöka detta har vi genomfört en fallstudie på ett mindre mjukvaruutvecklande företag som har för avsikt att börja använda modeller på ett mer aktivt sätt i systemutvecklingsprocessen. Studien utvärderades med hjälp av en fokusgrupp, där vi tillsammans med anställda på företaget diskuterade modellernas betydelse. Detta resulterade i ett ramverk innehållande viktiga principer för hur modeller kan vara användbara för organisationer i systemutvecklingsbranschen. De slutsatser som kunde dras efter studien var att modeller bör vara lättadministrerade, enkla att hålla uppdaterade och skapade för ett specifikt syfte. Vidare bör modellerna abstrahera problemområdet, skapas på ett kostnadseffektivt sätt samt ge en tillräcklig beskrivning av systemet. Tanken med studien var både att undersöka hur olika principer för modellering fungerar rent praktiskt samt hur de kan kombineras, något som visade sig tillämpbart eftersom vårt ramverk innehåller principer från flera olika områden.

Rapporten är skriven på svenska.

Nyckelord: Modell, modellering, ramverk, principer, systemutveckling, modelldriven utveckling, agil modellering

Förord

Vi vill passa på att tacka alla respondenter som tog sig tid att medverka i vår studie. Ert stora engagemang ledde till en mycket bra diskussion om modeller och systemutveckling. Vi vill även tacka hela organisationen som trots ett späckat schema lät oss genomföra fallstudien hos dem.

Avslutningsvis vill vi tacka vår handledare Kjell Engberg för all feedback vi har fått under arbetets gång.

Innehåll

1 Inledning

1.1 Bakgrund

Modeller och modellering anses ofta vara ett bra hjälpmedel för att hantera den komplexa situation som systemutveckling kan innebära (OMG, 2005). Det finns dock många olika åsikter om hur modellering bör utföras, något som visas av bland andra Ambler (2003) och Uhl (2003). Vi vill belysa detta faktum, vilket exemplifieras med ett praktiskt fall där vi kombinerar olika teoretiska och praktiska modelleringsprinciper. Organisationen som utgör fallet är ett mindre företag som arbetar med utveckling, försäljning och support av en egenproducerad mjukvara. Organisationen har under en längre tid prioriterat implementeringen av sitt system, vilket har medfört att dokumentation och modellering har hamnat i skymundan. Det har funnits flera välgrundade orsaker bakom denna prioritering och det bör noteras att den agila utvecklingsmetoden Scrum har använts som förebild vid utvecklingen. Detta är ett synsätt där man fokuserar just på fungerande mjukvara framför dokumentation (Agile Alliance, 2001). Den nuvarande situationen karaktäriseras av ett fungerande system med bristande dokumentation, samt besvär att implementera en tydlig och övergripande arkitektur för systemet. Det intressanta i sammanhanget är dock den situation de står inför nu, nämligen utmaningen att utveckla en ny version av systemet. För att underlätta detta arbete behövs ett underlag på hur systemet ska fungera och det finns därför en idé om att använda modeller på ett mer aktivt sätt i systemutvecklingen. Dessa modeller bör även vara användbara i det dagliga underhållsarbetet och förvaltningen av systemet, vilket är en annan viktig utmaning.

Modeller kan användas på många olika sätt inom systemutveckling och skapas ofta enligt notationen UML, som är en standard framtagen av the Object Management Group, OMG (OMG, 2005). Modeller kan användas i systemutvecklingens tidiga faser för att öka förståelsen inom området och Mathiassen et al. (2001) menar att *”framgångsrik systemutveckling är mycket beroende av utvecklarnas förståelse av systemets praktiska användning”* (s. 20). Enligt Larman (2005) kan modeller användas i tre detaljnivåer: Som en skiss (ett informellt diagram), en ritning (ett mer detaljerat diagram) och som implementering, med hjälp av kodgeneratorer. Med detta sagt behövs en vägledning i vilken detaljnivå som ska väljas. Det behövs en grundläggande syn på vilken roll modellerna ska spela: är de endast ett hjälpmedel eller bör modelleringen ha en mer central roll i utvecklingsprocessen?

Agila eller lättrörliga utvecklingsperspektiv har ökat stort i popularitet under framförallt det senaste decenniet (Parsons et al., 2007). Med detta i åtanke är det relevant att undersöka hur dessa kan kombineras med mer modelldrivna perspektiv. Två av de mer välciterade perspektiven och där det har funnits en livlig debatt, är mellan skolorna Model Driven Development och Agil modellering (Ambler, 2003; Uhl, 2003). Model Driven Development (MDD), eller modelldriven utveckling, utgör ett perspektiv där modellerna står i centrum för systemutvecklingen. Enligt skolan leder användandet av modeller till att man undviker

specifika implementeringsproblem och istället kan fokusera på det verkliga problemområdet (Selic, 2003a). Detta perspektiv kräver att modellerna är mycket sofistikerade (Ambler, 2003) och som en reaktion till detta och andra dokumentationstunga synsätt, framträder Agil modellering (AM). Denna skola menar snarare att modeller bör användas som ett lättare stöd för tankearbetet där programmering fortfarande är den viktigaste uppgiften (Larman, 2005). Det råder med andra ord en diskrepans mellan flera olika forskningsskolor.

Organisationen i studien vill använda modellerna för att visa hur objekt och flöden förhåller sig till varandra och därför blir MDD ett för stort och detaljinriktat ramverk i denna studie. Även om AM:s förhållningssätt med att modeller främst bör användas som ett tankestöd är sant i det aktuella fallet, behövs modellerna även till ren dokumentation i ett förvaltningsskede. Med detta i åtanke måste modellerna gå att uppdatera samt vara tillräckligt kompletta för att visa hur en förändring påverkar befintliga objekt och flöden. Att skapa denna detaljnivå går delvis emot flera av de principer som AM bygger på.

Eftersom varken MDD eller AM passar in för den specifika situationen tar vi stöd av Boehm (2002) som menar att organisationer bör balansera mellan agila och plandrivna metoder, där en anpassning till rådande förhållanden bör ske. Vi bygger vidare på dessa tankar och anpassar dem för modellering, där vi tittar på balansen mellan MDD och AM. På så sätt vill vi hitta principer för modellering från båda forskningsskolorna och sedan kombinera dessa med praktiska principer som är relevanta i det aktuella fallet.

1.2 Syfte

Syftet med denna studie är att beskriva hur man kan kombinera principer för modellering från både modelldriven utveckling och agil modellering. Genom detta vill vi visa att en organisation kan uppnå mer nytta med modellering genom att anpassa den till den specifika situation som råder. Utöver detta vill vi även belysa vikten av att anpassa urvalet av modeller och diagram till situationen.

1.3 Frågeställning

- Vilka principer för modellering är tillämpbara i fallstudiens miljö?
- Hur väl fungerar dessa principer som en vägledning för modellering i praktiken?

1.4 Avgränsning

Med hänsyn till de tidsramar som finns och studiens omfattning har vi valt att göra ett antal avgränsningar. För det första kommer vi enbart att titta på modeller skapade enligt notationen UML 2. Detta väljer vi eftersom notationen både är generell och vanligt förekommande inom systemutveckling. För det andra har vi begränsat studien än mer då vi har gjort ett antagande om vilka av de 13 diagrammen enligt UML 2-standarden som vi tror är kompatibla med inriktningen av fallstudien. Detta antagande beskrivs i Bilaga 1.

1.5 Disposition

I kapitel 2 definierar vi för studien viktiga begrepp samt presenterar den teori som ligger till grund för arbetet. Kapitel 3 redogör för den metod som vi använt oss av i studien. I fjärde kapitlet presenteras själva resultatet, där vi först redovisar resultatet av litteraturstudien och därefter det empiriskt insamlade materialet. I kapitel 5 diskuteras de båda resultaten vilket slutligen ger ett ramverk bestående av viktiga principer vid modellskapande. Kapitel 6 innehåller vår slutsats samt en utvärdering av studien och förslag till vidare studier. I det avslutande kapitlet finns våra referenser och Bilaga 1 innehåller ett antagande om UML-diagrammens användbarhet.

2 Teori

2.1 Definitioner

Modell: Ordet modell är mycket centralt i denna uppsats och det är därför viktigt att definiera ordets betydelse. Då ordet kan betyda olika saker väljer vi att följa Larmans (2005) definiering av ordet:

”A description of static and/or dynamic characteristics of a subject area, portrayed through a number of views (usually diagrammatic or textual).” (s. 691)

Modellera: Med modellera eller modellering menar vi skapandeprocessen av en modell. Modelleringsarbetet resulterar således i en modell och beskriver ett specifikt område som exempelvis ett informationssystem eller en verksamhet.

Diagram: I definitionen av modell framgick det att diagram är ett av två vanliga sätt att beskriva en modell. Vi väljer således att endast göra en liten skillnad mellan orden diagram och modell där ett diagram är en visuell beskrivning av en modell.

Ramverk: För att undvika språkförbistring i uppsatsen väljer vi att kalla den slutgiltiga sammanslagningen av principer för ett ramverk. Detta kan vanligtvis kallas för en modell i uppsatssammanhang men vi väljer alltså att använda ordet ramverk då modell har en annan betydelse (se definitionen för modell).

Systemdokumentation: Även systemdokumentation är ett viktigt begrepp i denna uppsats. Systemdokumentation definieras av Forward (2002) och lyder:

”[Software Documentation is] an artifact whose purpose is to communicate information about the software system to which it belongs.” (s. 1)

Vidare kan systemdokumentation delas upp i två typer, inline-dokumentation och extern systemdokumentation där den senare består av all dokumentation som existerar utanför systemet i form av text och diagram. (Magnusson, 2004). I uppsatsen menar vi den externa systemdokumentationen när vi nämner begreppet som alltså kan bestå av diagram.

Modelldriven utveckling: Med modelldriven utveckling menar vi Model Driven Development eller MDD som beskrivs under punkt 2.3.

Agil modellering: Med agil modellering menar vi Agile Modeling eller AM, vilket är ett lättrörligt förhållningssätt till modellering. Agil modellering beskrivs mer utförligt under punkt 2.4.

2.2 Modeller och UML

Användandet av modeller inom traditionella ingenjörsdiscipliner för att bygga komplexa system är viktigt och hjälper ingenjörer att både verifiera sin design samt kommunicera den

till andra (Selic, 2003a). Ett syfte med modellering är att förstå ett problem men även lösningars potential, innan man går igenom den kostsamma processen att faktiskt bygga systemet eller artefakten (Selic, 2003b). Selic (2003a) menar att ingen skulle få för sig att bygga en bro innan man har gjort en modell eller ritning av den. Historiskt sätt har modeller inom systemutvecklingsbranschen dock spelat en ganska liten roll till skillnad från i de traditionella ingenjörsdisciplinerna. Förespråkare för modelldriven utveckling ställer sig undrande inför detta faktum då systemutveckling ofta innebär skapande av mycket komplexa artefakter och det anses därför som en självklarhet att branschen kan dra stor nytta av modeller (Selic, 2003a).

Det finns flera fördelar med att använda modeller inom systemutvecklingsprocessen. The Object Management Group, OMG (2005) hävdar att modeller är ett bra sätt att hantera komplexitet genom att möjliggöra arbete på en högre abstraktionsnivå. Gruppen ligger bakom modelleringsnotation Unified Modeling Language eller UML, som hjälper utvecklare att specificera, visualisera och dokumentera modeller av ett system. UML har blivit branschstandard för modellering (Larman, 2005) och är flexibelt nog för all typ av modellering, oavsett detaljnivå, typ av applikation eller vilken mjuk- och hårdvara som används (OMG, 2005).

En vidareutveckling av UML kom år 2005 då UML 2 introducerades. I denna utveckling tillkom ett antal nya diagram, en förbättrad möjlighet till funktionsmodellering, samt en tätare koppling mellan struktur och funktionsdiagram (OMG, 2005). UML 2-standarden består av 13 stycken diagram indelade i tre kategorier: statiska diagram, funktionsdiagram och interaktionsdiagram. En närmare beskrivning av diagrammen finns i Bilaga 1.

2.3 Modelldriven utveckling (MDD)

Som tidigare nämnts har förespråkare för modelldriven utveckling ställt sig frågande till modellers begränsade betydelse inom systemutveckling. Synsättet modelldriven utveckling, eller Model Driven Development (MDD), kännetecknas av att fokus flyttas från programmet till modellerna, vilket ger flera fördelar. Bland annat kan utvecklare fokusera på själva problemområdet istället för begränsningar som kan komma med ett specifikt programmeringsspråk. Detta leder i sin tur till att modellerna både blir enklare att förstå och underhålla (Selic, 2003a). Selic (2003a) nämner även att modeller har ett begränsat värde om de enbart används som systemdokumentation. En avgörande faktor för MDD är således att modeller kan omvandlas till exekverbar kod. Detta är ett tänk som länge funnits inom branschen men som aldrig slagit igenom, dock finns det två områden där stor utveckling har skett, nämligen effektiva kodgeneratorer och standarder (Selic, 2003a).

En annan viktig aspekt av modelldriven utveckling är kvalitén på modellerna. Oavsett hur bra kodgeneratorer och standarder blir kommer det alltid krävas att kvalitén på modellerna är hög för att de ska vara användbara och effektiva i systemutvecklingsarbetet. Selic (2003b) nämner att en modell måste abstrahera problemområdet, vara förståelig, korrekt, förutseende samt billig att producera. En anledning till att modeller har fått en begränsad genomslagskraft i

systemutvecklingsbranschen tror författaren beror på att de saknar någon av de nämnda egenskaperna.

2.4 Agil modellering (AM)

I kontrast till det modelldrivna synsättet där modeller bör vara så detaljerade att de kan omvandlas till kod, står agil eller lättrörlig modellering. Enligt detta synsätt har modellering ett helt annat syfte vilket visas av följande citat:

”The purpose of modeling (sketching UML, …) is primarily to understand, not to document.”
(Larman, 2005, s. 30)

Med detta som utgångspunkt menar författaren att modeller bör användas som ett snabbt och effektivt sätt för att utforska ett problemområde. Skaparen av agil modellering, Scott W Ambler, beskriver att en agil modell endast är tillräckligt bra och inget därutöver. Då tillräckligt bra är ett relativt begrepp kan det betyda allt från en skiss till en mer detaljerad modell, allt beroende på den aktuella situationen (Ambler, 2003). Författaren menar även att modelldriven utveckling inte fungerar i praktiken på grund av tre anledningar. För det första är de standarder som MDD bygger på inte tillräckligt utvecklade där exempelvis modelleringsnotationen UML inte möter de krav som finns i verkligheten. För det andra saknas de avancerade modelleringskunskaper som krävs för att göra detaljerade modeller bland många systemutvecklare och för det tredje finns inte de verktyg som krävs för att utveckla modeller till körbara program (Ambler, 2003).

Agil modellering är baserat på ett antal viktiga principer för hur man bör arbeta med modeller. Dessa principer handlar både om modellernas utformning och om modelleringsprocessen. Enligt Larman (2005) bör man inte modellera hela systemet i detalj utan endast de delar som är ovanliga, svåra, komplexa eller liknande. En annan viktig princip är att man ska använda enkla verktyg när man modellerar för att stödja snabba förändringar som uppkommer. Författaren föreslår t.ex. att man modellerar med hjälp av whiteboards och sedan fotograferar detta med en digitalkamera, men poängterar samtidigt att nyckelordet är lättrörlighet oavsett val av teknik. Vidare hävdar Larman (2005) att detaljerade UML-diagram inte är viktiga så länge systemutvecklarna förstår modellen och varandra samt att alla modeller är felaktiga i jämförelse med den slutgiltiga koden eller programmet.

2.5 Kombination av plandrivna och agila metoder

Många av förespråkarna för plandriva respektive agila metoder ser sig själva som motparter av två nästintill oförenliga perspektiv. Boehm (2002) menar dock att om man kombinerar de båda synsätten kan man förse utvecklare med ett brett spektrum av verktyg och alternativ. Vidare menar författaren att det finns utmärkande drag i projekt där plandrivna respektive agila metoder fungerar bäst. Bland annat fungerar agila metoder bäst i små projekt som omges av snabba förändringar i kravspecifikationen medan plandrivna metoder fungerar bättre i stora projekt där kraven är mer stabila (Boehm, 2002). Om man däremot har ett projekt som består av en mix mellan karaktäristiska drag från de båda metoderna bör man välja ett

hybridalternativ, eller en kombination av metoderna. Även om utvecklingen går mot agila metoder för tillfället måste organisationer försöka hitta den bästa balansen mellan plandrivna och agila metoder som bäst lämpar sig för deras specifika situation (Boehm, 2002).

3 Metod

3.1 Vetenskapliga ansatser

Vi har valt en kvalitativ inriktning vilket tillåter oss att undersöka och tolka aspekter kring vårt problemområde. Genom att arbeta kvalitativt vill vi kunna förklara och förstå problematiken som finns kring vår frågeställning. Patel & Davidsson (2003) menar att kvalitativa studier är bättre lämpade än kvantitativa när man vill förstå människors upplevelser, samt nå en djupare kunskap än den fragmenterade kunskap som ofta erhålls genom ett kvantitativt arbetssätt. Vårt problemområde och vår specifika situation lämpar sig väl för kvalitativa studier då många viktiga aspekter inte enkelt låter sig mätas i siffror eller tal: dessa aspekter består snarare av mjukare värden, där en mer tolkande ansats passar bättre. Detta motiverar även valet av en kvalitativ studie framför en kvantitativ.

Studien har även en deduktiv inriktning och kan därför beskrivas som teoridriven, snarare än empiridriven. Deduktion kännetecknas av att man *”utifrån allmänna principer och befintliga teorier drar slutsatser om enskilda företeelser”* (Patel & Davidsson, 2003, s.23). En huvudpoäng med studien är att undersöka hur väl olika teoretiska ståndpunkter är tillämpbara i praktiken. Studien har därmed en viktig koppling till det praktiska, men även en tydlig utgångspunkt i teorin.

3.2 Metodval

Initialt använde vi oss av en litteraturstudie för att söka efter principer som kan vara viktiga vid modellering. Utöver principerna från litteraturstudien har vi även samlat in krav och förväntningar från den aktuella organisationen som sedan omvandlats till praktiska principer. Detta kombinerade vi sedan med en fallstudie och en fokusgrupp för att samla in empirisk data. Att använda sig av fallstudier är särskilt lämpligt i utvärderingar där studieobjekten ofta är mycket komplexa (Backman, 1998). Då en del av studien bestod av att utvärdera de framtagna principerna anser vi att en fallstudie är en lämplig metod. Valet av fokusgrupp grundade sig i att dessa är bra på att fånga deltagarnas attityder och tankar kring ett ämne (Preece et al., 2007).

3.2.1 Litteraturstudie

Enligt Backman (1998) kan litteraturgranskning bland annat användas för att ge en översikt över tidigare samlad kunskap inom ett område och ge metodiska uppslag. Litteraturstudien har två huvudsakliga syften i denna studie, nämligen att bredda våra kunskapar i problemområdet samt finna viktiga principer vid modellering. Det är dessa principer som vi har för avsikt att kombinera med mer praktiska principer och sedan utvärdera genom en fallstudie.

3.2.2 Fallstudie

Fallstudier är ett vanligt inslag i kvalitativa studier. Backman (1998) menar att en fallstudie *”… undersöker ett fenomen i sin realistiska miljö eller sin kontext, där gränserna mellan*

fenomen och kontext inte är givna" (s. 49). Enligt Backman bör man initialt beskriva studiens forskningsfråga, för att sedan fortsätta med val av analysenhet och specifikt fall. Det fenomen eller den fråga som vi vill undersöka beskrivs närmare i uppsatsens inledning och frågeställning. Studiens analysenhet är ett mindre mjukvaruutvecklande företag och det specifika fall som ska studeras rör organisationens arbete med att använda modeller mer aktivt i sin systemutveckling. Fallstudier har olika avsikter (Backman, 1998) och vår kan beskrivas som explorativ och utforskande snarare än deskriptiv.

I fallstudien skapade vi modeller utifrån principerna från litteraturstudien. Principerna användes som en vägledning för att skapa användbara modeller för organisationen och för att prioritera tillgängliga resurser på rätt sätt. Fallstudiens syfte var således att testa principernas giltighet i den praktiska miljön. Vidare ville vi undersöka hur mycket principerna överlappade varandra vilket kunde tyda på att de borde sammanfogas till en mer övergripande princip.

Vid användandet av fallstudier är det viktigt att reflektera över resultatets generaliserbarhet. Generaliserbarheten vid fallstudier beror på urvalet av fall eller analysenheter som inkluderas i studien (Patel & Davidsson, 2003). Detta bör ses i förhållande till den totala populationen som man önskar generalisera om. I vårt fall studerar vi endast en organisation, vilket innebär att resultatet inte är generaliserbart i någon större uträckning, utan istället lokalt förankrat.

3.2.3 Fokusgrupp

Vi genomförde även en fokusgrupp, där vi tillsammans med anställda på företaget ville utvärdera fallstudien. Fokusgrupper har bl. a. använts flitigt inom marknadsföring och kundorienterade undersökningar, men även mer IT- och designinriktade tillämpningar förekommer (Preece et al., 2007). En fokusgrupp tar tillvara på det faktum att personer interagerar och påverkas av andra personer och det möjliggör för diskussionsledaren att direkt interagera med respondenterna. Detta kan man göra för att klargöra eller utveckla frågor under arbetets gång (Marczak & Sewell, 2002). Vidare nämner Preece et al. (2007) fler användningsområden för fokusgrupper: de kan fånga deltagares attityder och tankar kring ett specifikt ämne samt belysa områden där konflikt eller konsensus råder. Vid användandet av fokusgrupper bör man dock beakta risken att någon deltagare blir alltför dominant. Om detta skulle ske kan det hanteras genom att låta alla deltagare initialt få en viss tid att besvara en fråga (McNamara, 2006).

För att deltagarna skulle få en möjlighet att sätta sig in i modellering innan fokusgruppen försåg vi dem med ett underlag i form av modeller från fallstudien. Deltagarna informerades om att deras svar kommer att behandlas konfidentiellt. Vid genomförandet av fokusgruppen hade vi planerat för totalt två timmars diskussion och detta stämde väl överrens med den slutgiltiga tiden. Mötet tog plats i företagets konferenslokal där vi hade tillgång till projektor som användes för att komplettera det utskrivna underlaget.

Vidare antog vi en semistrukturerad inriktning där vi strukturerade upp fokusgruppen efter ett antal teman. En semistrukturerad inriktning innebär att man använder sig av både öppna och stängda frågor, där man utgår från ett antal fördefinierade ämnen (Preece et al., 2007).

Deltagarna var dock inte hindrade att lämna det aktuella temat, dessa användes snarare som ett diskussionsunderlag. Med utgångspunkt i litteraturstudien identifierades fem ämnen eller teman:

- Överblick och detaljnivå
- Modellernas bidrag och nytta
- Förståelse och beskrivningar
- Kostnad och tidsåtgång
- Förändringar

Det finns många faktorer som kan påverka utgången av en utvärdering och det är därför viktigt att känna till risken för felkällor, samt hur detta bör hanteras. Ett exempel är om de framtagna modellerna är felaktiga eller ofullständiga och att detta i sin tur påverkar deltagarnas bild av studien. För att hantera detta kommer vi i datainsamlingen även att utvärdera om modelleringsarbetet i helhet uppfyllde de krav och förväntningar som deltagarna hade på arbetet.

3.3 Urval

I fokusgruppen deltog samtliga fyra personer från organisationens utvecklingsgrupp. Vårt mål var att så många som möjligt från organisationen som arbetade med utveckling skulle delta, vilket därmed föll väl ut. Här följer en kort bakgrundsbeskrivning med deltagarnas titel och antal år de arbetat inom organisationen:

- Respondent A: Systemarkitekt, knappt 3 år.
- Respondent B: Systemutvecklare, 1,5 år.
- Respondent C: Utvecklingschef, 8 år.
- Respondent D: Systemutvecklare, 1,5 år.

Man kan givetvis diskutera om en fokusgrupp med endast fyra personer är tillräcklig, då detta har en inverkan på studiens generaliserbarhet. Eftersom syftet med fokusgruppen var att utvärdera fallstudien och de framtagna principerna fanns det ett antal krav på deltagarna. Dessa personer måste både inneha en kunskap om systemutveckling i allmänhet och en kunskap om det system som modellerna var skapade för. Eftersom fallstudien genomfördes på ett mindre utvecklingsföretag där det endast arbetar fyra personer med denna kunskap, är därmed urvalet motiverat.

3.4 Analys av data

Under fokusgruppen samlade vi in och dokumenterade arbetet med hjälp av ljudinspelning och egna anteckningar. Materialet började bearbetas direkt efter arbetets avslutande, när vi

fortfarande hade alla detaljer färskt i minnet. Detta resulterade i ett insamlat material av kvalitativ art som sammanställdes utifrån de teman som fokusgruppen var uppbyggd kring. Vid den fortsatta analysen arbetade vi iterativt med att komplettera och strukturera anteckningarna, där vi utgick ifrån anteckningarna men tog stöd i ljudinspelningen när detta krävdes.

4 Resultat

I avsnittet presenteras och beskrivs först viktiga principer för att skapa modeller vilket är ett resultat av litteraturstudien. I den andra delen presenterar vi det insamlade materialet från fokusgruppen, som strukturerats efter de teman som vi utgick från vid genomförandet. Den andra delen är ett resultat av fallstudien och fokusgruppen.

4.1 Principer för att skapa modeller

Nedan presenteras de principer vi har identifierat och som har hämtats från både teoretiska och praktiska förhållningssätt. De teoretiska principerna är hämtade från två skilda synsätt: modelldriven utveckling (MDD) och agil modellering (AM). Många av principerna överlappar varandra vilket visar på att det finns gemensamma drag mellan MDD, AM och den praktiska miljön vi undersökte.

4.1.1 Principer från praktiken

Principerna har framkommit genom de krav och förväntningar som organisationen hade på modelleringen där huvudpoängen var att modeller bör skapas för ett specifikt syfte.

En modell ska vara lättadministrerad (1): Det är viktigt att en modell är lätt att skapa och förändra. För att uppnå detta behövs ett bra IT-stöd där det är enkelt att skapa och förändra modeller. Vidare bör man kunna koordinera sina modeller med den verkliga utvecklingsmiljön på ett lättadministrerat sätt för att undvika dubbelarbete. Dessutom behöver man koordinera sina modeller med varandra för att underlätta uppdateringsarbetet när en förändring sker.

En modell ska vara enkel att hålla uppdaterad (2): När man ska göra en förändring i sitt system ska man kunna hålla sina diagram uppdaterade på ett enkelt och ekonomiskt fördelaktigt sätt. Detta görs möjligt genom att endast modellera de komplicerade delarna samt koordinera sina modeller med varandra för att tydliggöra vilka som behöver förändras och hur de hänger ihop. För att uppnå detta kan man skapa exekverbar kod direkt från sina modeller med hjälp av en kodgenerator. Tekniken kan även användas på motsatt håll genom Reverse Engineering, där man skapar och håller UML-diagram uppdaterade direkt från den exekverbara koden (Larman, 2005). Detta är såklart det smidigaste sättet att hålla modellerna uppdaterade då faktiska förändringar i koden syns direkt i modellerna. Dock finns det problem med denna teknik såsom skalbarhet och effektivitet, vilket gör tekniken svår att tillämpa i större skala. (Selic, 2003a). Oavsett om man använder sig av tekniken eller inte är det viktigt att modeller går att förändra när ett sådant behov framkommer.

En modell ska vara nyttig i nyutveckling och/eller i förvaltning (3): Det är viktigt att en modell inte bara skapas för skapandets skull utan det måste finnas en anledning till skapandet. Det är därför viktigt att tänka på om modellen behövs som stöd i ett nyutvecklingsskede eller i ett förvaltningsskede. I ett nyutvecklingsskede kan en modell exempelvis lösa ett komplext

problem, i ett förvaltningsskede kan den istället användas som systemdokumentation och förklara systemets funktionalitet.

En modell ska beskriva systemet (deskriptiv) eller de krav som finns på det (normativ) (4): Precis som vid den tidigare diskussionen om en modells nyttighet i ett nyutvecklings- eller förvaltningsläge, bör man vara införstådd med om en modell är deskriptiv eller normativ. En deskriptiv modell beskriver ett system eller en situation efter hur den faktiskt ser ut till skillnad från en normativ modell som beskriver hur något är tänkt att se ut (Papazoglou & Ribbers, 2006). Normativa modeller skulle kunna ses som ännu inte implementerade i praktiken, eller en beskrivning av hur praktiken bör fungera.

4.1.2 Principer från modelldriven utveckling (MDD)

Följande principer är hämtade från den modelldrivna utvecklingsskolan och har beskrivits av Bran Selic (2003a; 2003b). Principerna användes från början till att beskriva kriterium för om en modell ska räknas som användbar och effektiv. Vidare är principerna fritt översatta och beskrivna för att passa vårt fokus.

En modell ska abstrahera problemområdet (5): Att kunna visa en situation på ett överblickbart sätt är ett vanligt syfte med modellering. Att abstrahera en situation är även ett av de vanligaste sätten att hantera den komplexa situation som systemutveckling kan innebära (Selic, 2003a). Rent praktiskt kan man följa principen genom att dölja onödiga detaljer i sina modeller. Dessutom är många av UML-diagrammen bra på just abstraktion, exempelvis ger ett klassdiagram en övergripande bild av ett systems viktiga objekt och dess relationer (Larman, 2005).

En modell ska vara förståelig (6): I föregående princip beskrevs vikten av abstraktion. Detta betyder inte att man abstraherar bort alla detaljer, utan man måste göra en avvägning av vilka detaljer som är viktiga för förståelsen och som således bör finnas kvar. Förutom detta kan modellerna kompletteras med notationer och på så sätt förklara komplexa delar för att och öka förståelsen. Just att öka förståelsen för en situation är en styrka med modellering då förståelse av kod kräver hantering av en oerhörd mängd information (Selic, 2003a).

En modell ska visa problemområdet på ett korrekt sätt (7): Det är viktigt att representera problemområdet på ett så korrekt sätt som möjligt (Selic, 2003a). Detta kan utföras genom att modellera en representation som stämmer överrens med den verkliga bilden och som visar de mest intressanta aspekterna av systemet. Hur korrekt en modell är kan härledas till detaljnivån, då abstrahering av alltför många detaljer, inte ger en fullständig bild av systemet.

En modell ska vara förutsägbar (8): Med hjälp av modellen ska man kunna förutsäga vad systemet gör eller är tänkt att göra. Det är därför viktigt att välja en modell som på ett bra sätt beskriver den specifika situationen. Detta kan testas med ett praktiskt experiment där modeller översätts till exekverbar kod med hjälp av en kodgenerator. Detta är dock inte helt oproblematiskt vilket beskrevs tidigare i princip 2.

En modell ska vara kostnadseffektiv att skapa (9): En modell ska inte skapas bara för skapandets skull, utan med hjälp av modellen ska ett problem kunna lösas på ett mer kostnadseffektivt sätt än om man inte hade modellen (Selic, 2003a). Det är därför viktigt att endast skapa modeller som behövs för en viss situation. Skapas modeller som man inte har nytta av leder det till mycket merarbete då det tar både tid och andra resurser att skapa själva modellen samt att underhålla den i ett förvaltningsskede. Mängden diagram som ska hållas aktuella leder dessutom till en komplex situation, vilket i sin tur innebär att man har försummat en av huvudpoängerna med hela modelleringen.

4.1.3 Principer från agil modellering (AM)

Agil modellering bygger på ett antal kärnprinciper framtagna av Ambler (2003) som är fritt översatta och beskrivna. Principerna täcker ett stort område av systemutvecklingen och de som inte passar in har således exkluderats. Ambler (2007a) nämner dessutom att man inte behöver använda alla principer för att uppnå nytta med modelleringen.

En modell ska skapas för ett specifikt syfte (10): Anledningen till skapandet av en modell styr många aspekter av den såsom abstraktions- och detaljnivå. Skapas modellen för att ge en överblick, för att kommunicera någonting, eller för att dokumentera systemet för framtiden? Genom att vara medveten om syftet vid modelleringsarbetet, kan man enklare relatera till hur modellen bör utformas. Syftet tillsammans med tilltänkt målgrupp bör bland annat bestämma detaljnivån på modellen (Ambler, 2006).

En modell ska vara så enkel att den är värd att underhålla (11): Enligt denna princip vill man troligtvis inte behålla allt som modelleras, men bestämmer man sig för att behålla en modell, ska den också vara värd att underhålla när en förändring sker. För att kunna vara agil måste modellerna vara enkla och få. Man kommer troligtvis inte kunna upprätthålla detaljerade modeller på alla områden vilket innebär att man måste göra avkall på antalet modeller, antalet områden man vill täcka in, eller på detaljnivån i de modeller man väljer att behålla. Troligtvis måste man göra avkall inom alla tre områden för att få en hanterbar situation när en förändring sker.

En modell ska bidra till en fullständig beskrivning av systemet (12): För att kunna beskriva ett system på ett korrekt sätt kommer man att behöva flera olika diagram. Vilka diagram som behövs beror på situationen som man vill beskriva, dock räcker sällan ett diagram. Det är därför viktigt att tänka igenom diagrammens olika syften samt vilka delar som är viktiga, svåra eller komplexa för det specifika projektet och sedan fokusera på dessa.

En modell ska vara så enkel som möjligt (13): Den här principen handlar om att inte ta med onödig funktionalitet eller krav som inte behövs i dagsläget. Man bör även undvika att överarbeta modellerna eller att modellera på en alltför detaljerad nivå. Istället kan modellerna utvecklas vid ett senare skede i utvecklingsprocessen. Detta innebär att man kan börja väldigt enkelt och sedan fördjupa sig om detta skulle krävas. Man bör dock förutsätta att den enklaste lösningen även är den bästa.

En modell ska i slutändan leda till fungerande mjukvara (14): Kopplingen mellan modelleringsarbetet och framtagandet av fungerande och kvalitativ mjukvara som möter användarnas krav måste poängteras. Man bör därför fråga sig om en modell bidrar till målet att producera denna mjukvara. För diskussionens skull kan man ifrågasätta om en modell som användarfallsdiagram (eller användarfall i allmänhet) leder till fungerande mjukvara. Diagrammet går inte att översätta direkt till exekverbar kod, men har istället andra fördelar då det visar viktiga krav på systemet och de vanligaste scenarier som de tilltänkta användarna behöver systemet för (Larman, 2005). Därmed behöver modellen inte nödvändigtvis direkt resultera i fungerande mjukvara, så länge den utgör ett viktigt stöd i framtagandet av densamma. Principen tillåter att modeller som utforskar både användningsområdet och problemområdet skapas för att stödja processen mot en fungerande mjukvara.

En modell ska vara användbar i framtiden (15): Principen behandlar kopplingen till hur modellerna ska användas i ett framtida läge eller ett förvaltningsläge. Det är viktigt att ha fokus på nuläget och det som är aktuellt just nu, men man bör samtidigt ha framtiden i åtanke. Vilka personer ska arbeta med nästa steg av produkten? Hur ska kunskap överföras från nuvarande utveckling till framtida? Det finns alltid en risk att viktiga personer inte finns till hands och det kan därmed vara viktigt att överväga hur modellerna kan bidra till att överföra kunskap. Även det omvända gäller: om många med säkerhet kommer att stanna kvar, är troligen behovet av noggrann dokumentation mindre viktigt.

4.2 Empiriskt material

I detta avsnitt presenteras det resultat som framkom i fokusgruppen och fallstudien. Vi har strukturerat materialet efter ett antal teman som i sin tur byggde på de principer som presenterades ovan.

4.2.1 Överblick och detaljnivå

Detta tema behandlade följande principer:

- En modell ska abstrahera problemområdet (5)
- En modell ska vara så enkel som möjligt (13)

För att skapa en fullständig överblick var det absolut viktigaste kravet att modeller som syftar till detta inte var för detaljerade. Genom att abstrahera onödiga detaljer kan en bättre överblick skapas. Ett översiktligt komponentdiagram upplevdes kunna bidra med detta på en ännu högre nivå än till exempel ett klassdiagram.

Klassdiagram visade sig också användbara. Flera av deltagarna i diskussionen ansåg att ett mer abstrakt klassdiagram (s.k. analysklassdiagram) var användbart för att skapa en systemöversikt, framför allt i jämförelse med det mer detaljerade designklassdiagrammet. Det ansågs viktigt att ha detaljnivån i åtanke där alltför många detaljer skulle ha en negativ inverkan. Arbetsbördan med att uppdatera och underhålla diagrammet skulle öka och fler detaljer skulle även bidra till en försämrad överblickbarhet.

” ...[det krävs] ju mycket mer jobb att hålla det [designklassdiagrammet] ajour, än vad det krävs att hålla ett analysklassdiagram ajour” – Respondent D

Idéerna om att modellera på en hög detaljnivå återkom även i ett senare exempel angående ett aktivitetsdiagram, där liknande argumentation fördes fram.

” ... sådana stora förändringar borde inte vara så vanliga, som påverkar att flödet förändras helt och hållet. ... jag tycker att bilden ska va så pass övergripande ändå, [att förändringar sällan sker]” – Respondent C

Enligt ovan skulle modellering på en övergripande nivå även innebära att man kan hantera förändringar och uppdateringar på ett bättre sätt. Många deltagare ansåg att uteslutande av detaljer kunde ge modeller som inte kräver lika mycket underhåll. Dock uppstod en diskussion angående när man kan ha användning för mer detaljerade diagram. På frågan om diagrammen därmed borde vara mer detaljerade, för att visa fler detaljer kring till exempel en förändring, framfördes återigen att alltför detaljerade modeller lätt blir oanvändbara.

”Har man med allt blir det som att kolla på koden, om man skulle ha med allt, det går ju inte” – Respondent B

Jämförelsen mellan modeller och ren programkod fanns även med angående designklassdiagrammet, där en diskussion kring skillnaden mellan en konceptuell modell, ett klassdiagram och ett databasschema uppstod. Flera deltagare uttryckte här att alla modeller kanske inte är motiverade och att dess detaljgrad är kopplad till vad de ska användas för.

Ett systemsekvensdiagram kan användas för att ge en överblick, men nyttan med detta ifrågasattes, då flera av deltagarna ansåg att överlappningen mellan dessa och mer traditionella användarfall skulle vara för stor. Diagrammet kan ge en mer abstrakt bild än jämförelsevis ett mer detaljerat sekvensdiagram, men denna abstrahering upplevdes inte som meningsfull.

”… vinsten är ganska liten i det faktiskt [om systemsekvensdiagram]” – Respondent D

4.2.2 Modellernas bidrag och nytta

Detta tema behandlade följande principer:

- En modell ska skapas för ett specifikt syfte (10)
- En modell ska bidra till en fullständig beskrivning av systemet (12)
- En modell ska vara nyttig i nyutveckling och/eller i förvaltning (3)
- En modell ska vara användbar i framtiden (15)
- En modell ska vara förutsägbar (8)

- En modell ska i slutändan leda till fungerande mjukvara (14)

Deltagarna i fokusgruppen poängterade att det finns en skillnad i vilket skede ett diagram är användbart där vissa diagram är mer användbara i ett nyutvecklingsskede vid analys och kravspecifikation medan vissa är mer användbara vid underhåll i ett förvaltningsskede.

”… det här diagrammet [sekvensdiagram] har vi ingen nytta av i förvaltning ... medans ett analysklassdiagram har vi fortfarande nytta av.” – Respondent D.

En annan huvudpoäng som framkom var vikten av att tänka igenom vilka av diagrammen som man har nytta av. Valet av diagram som ska användas måste vara grundat i den situation som man befinner sig i och där man ser en nytta av diagrammet. Här poängterades även vikten av att organisationen fattar ett beslut och bestämmer vilka diagram som ska användas samt i vilka situationer som de behövs.

”… det gäller ju för oss att bestämma ... när och vilka diagram ska vara aktuella...” – Respondent C.

En modell kan även vara nyttig i andra situationer än vid nyutveckling eller förvaltning. En situation som nämndes var vid diskussion med andra intressenter som exempelvis företagets supportavdelning. Om supporten kunde följa ett systemflöde och således kunde hjälpa en kund, skulle detta vara en stor nytta med modellerna. Även om ett användarfall i text kan vara till nytta här uttrycktes det att diagram kan i vissa fall ge en tydligare beskrivning av situationen. Ett annat läge där modeller skulle kunna vara behjälpliga är vid skapandet av manualer för systemet samt att ge nyanställda en introduktion i organisationens system.

En stor del av diskussionen handlade om de olika modellernas bidrag och nytta, där man såg vissa skillnader. När det gäller klassdiagram kan man ha olika nyttor beroende på vilken detaljnivå som finns i diagrammet. Ett abstraktare klassdiagram ansågs exempelvis vara väldigt användbart för att ge en systemöversikt, samt i allmänhet mer användbart än ett detaljerat klassdiagram.

”Det är ju framförallt hur de olika objekten förhåller sig till varandra…[som är intressant att se i ett klassdiagram]” – Respondent A.

På frågan om aktivitetsdiagrammets bidrag tyckte en deltagare att det ger en väldigt bra överblick över flöden, viktiga beslut samt i vilken ordning saker sker. Detta är något som det uttrycktes en saknad för i dagens situation. På samma fråga om tillståndsdiagrammets bidrag betraktades det inte helt oväntat som användbart för att visa möjliga tillstånd för centrala objekt. Tillståndsdiagrammen kan vidare vara användbara vid både nyutveckling och vid förvaltning, enligt respondenterna. En deltagare berättade om nyttan med att skapa både ett aktivitetsdiagram och ett tillståndsdiagram då de beskriver olika saker.

”… man ser ju samma sak på olika sätt kan man säga... här [aktivitetsdiagrammet] ser man flödet och här [tillståndsdiagrammet] ser man mera ett tillstånd i tiden.” – Respondent C.

Som tidigare nämnts ansågs det abstrakta systemsekvensdiagrammet inte bidra i någon större utsträckning eftersom flera andra diagram visade samma sak. Ett aktivitetsdiagram nämndes som ett bättre alternativ, då deltagarna i många fall inte såg vinsten i förhållande till det arbete som systemsekvensdiagrammet innebär. Vid resonemang om sekvensdiagram rent generellt sågs en nytta av det vid diskussioner kring ett specifikt fall, det var med andra ord mer användbart för att kommunicera kring ett problem än att förstå själva problemet.

" … mest för att snacka om det. Sätter man sig själv och ska sätta sig in i [situationen] kanske man hellre kollar på koden." – Respondent B.

Ett annat diagram som diskuterades var objektdiagram. Även här rådde det delade meningar om nyttan med modellen, men i de skeden där ett klassdiagram blir för abstrakt kunde det vara användbart. Ett exempel på detta är om man tilldelar instanserna trovärdiga namn och attribut kan diagrammet användas för att öka förståelsen för icke-utvecklare, exempelvis personer på supportavdelningen.

Även komponentdiagrammet diskuterades och det framkom att diagrammet ger en väldigt bra översikt vilket alla respondenter höll med om. I deras situation finns det många kopplingar till andra system och gränssnitt som är viktiga att tänka på. Dessutom betonades vikten av att det var övergripande då det vore väldigt svårt att skapa ett gemensamt och övergripande klassdiagram för alla komponenter. En annan uttalad fördel med diagrammet var att det bedömdes som relativt lätt att hålla uppdaterat. Detta eftersom det beskriver stora delar av systemet där förändringar är relativt ovanliga.

4.2.3 Förståelse och beskrivning

Detta tema behandlade följande principer:

- En modell ska vara förståelig (6)
- En modell ska visa problemområdet på ett korrekt sätt (7)
- En modell ska beskriva systemet (deskriptiv) eller de krav som finns på det (normativ) (4)

Deltagarna i fokusgruppen tyckte att modellerna var enkla att förstå i stora drag. En bidragande anledning till detta är att de kan sitt egna system samt systemutveckling i allmänhet, vilket gör förståelsen av diagrammen enklare.

"Givet att man förstår UML-syntax så…[går modellerna att förstå utan ytterligare förklaringar eller beskrivningar]" – Respondent D.

Modellerna betraktades även som enkla att förstå för andra intressenter, även om det finns detaljer som kräver djupare kunskaper. Dock kan många förstå huvuddelarna och huvudsyftet med modellerna. Som tidigare nämnts ansågs modellerna även användbara för supportbefattningar samt vid skapandet av manualer.

En respondent betonade vikten av att förståelsen kan vara olika från roll till roll. Enligt respondenten bygger förståelsen mycket på vad personen i fråga har för nytta av modellen.

”Man förstår väl dem man behöver förstå, eller om man tittar på ett klassdiagram, det är ingen som behöver förstå det om man inte ska koda.” – Respondent B.

En annan diskussionspunkt gällde om modellerna beskriver de krav som finns på systemet eller systemet i sig. Här var deltagarna inte helt överrens utan det fanns olika uppfattningar om exempelvis ett klassdiagram eller ett aktivitetsdiagram beskrev kraven på systemet eller systemet i sig. En diskussion som framkom här var även skillnaden mellan krav på systemet och systemfunktioner. Vid en detaljerad situation har det funnits svårigheter att utifrån modellerna avgöra vad som är krav och vad som är en funktion som uppstår till följd av kraven. Den allmänna uppfattningen var att det kan finnas oklarheter kring hur modeller hanterar krav.

Vid diskussioner om de olika modellerna nämndes en skillnad mellan ett analys- och designklassdiagram där det förstnämnda ansågs mer spegla kraven på systemet medan designklassdiagrammet speglade hur systemet faktiskt såg ut. För att öka förståelsen för klassdiagram såg man även en poäng i att skapa objektdiagram i komplicerade situationer. Trots att överlappningen kan bli stor ansågs detta ändå gynna förståelsen i vissa skeden. Vidare i en diskussion om aktivitetsdiagrammet, betonades betydelsen av att modellen är korrekt och uppdaterad.

”Då är det är ju till att det fungerar som det ser ut i diagrammet ... om man antar att diagrammet beskriver verkligheten så måste det vara helt tight däremellan, annars blir det misskommunikation.” – Respondent D.

Vid en tidpunkt i fokusgruppen framkom det att modeller kan användas för att öka förståelsen för systemet. Det nämndes att en modell kunde öka förståelsen för delar av systemet, t.ex. vid situationer där mycket av kunnandet kring systemet är knutet till en viss person, vilket först märks när personen är frånvarande. Även om man i ett sådant fall förhoppningsvis kan titta i koden, trodde en respondent att man hade haft en vinst i att titta på modellen.

4.2.4 Kostnad och tidsåtgång

Detta tema behandlade följande principer:

- En modell ska vara kostnadseffektiv att skapa (9)
- En modell ska vara lättadministrerad (1)

I temat kostnad och tidsåtgång handlade diskussionen om att lägga ner tid på rätt saker. Deltagarna i fokusgruppen upplevde i allmänhet det som nyttigt att lägga ner tid på modellering. En fråga berörde specifikt om deltagarna upplevde att modellering leder till att projekt tar längre tid, där motsatsen hävdades:

”Det skulle va förödande om vi gör ett antal missar, som gör att vi måste i stort sett börja om. Så att lägga en månad extra på att göra en massa diagram kan vara värt det” – Respondent A

Respondent A får också medhåll från övriga deltagare som pekar på möjligheterna att spara tid genom användandet av modeller för att utforska problem. En deltagare lyfter fram produktens livslängd som en annan faktor som kan avgöra hur mycket tid som satsas på modeller. Modeller som kan användas för att dokumentera systemet blir viktiga i ett sådant fall, där det framtida underhållet kan förenklas.

”Jag ser framförallt att programmet kommer att leva i många, många år sen, och att ha den här dokumentationen är ju viktigt då” – Respondent C

En annan huvudpoäng som framkom vid diskussioner kring den tid som läggs ner på modeller, var att just själva arbetet med att rita diagram tvingar utvecklare att tänka igenom ett problem innan själva implementeringen. Detta ansågs kunna öka kvalitén på själva systemet.

”Bara det att man gör dom här [diagrammen] betyder ju att man tvingas att tänka till, oavsett hur lite av det här, så har man ju börjat tänka innan man börjar koda, och det höjer ju garanterat kvalitén” – Respondent A

Det upplevdes därmed som nyttigt att investera en viss tid i modeller, framför allt på områden som snabbt blir svåröverskådliga. Uppfattningen skiljde sig dock när det kom till sekvensdiagram, där deltagare upplevde att den tid dessa tar i anspråk för att framställa inte alltid var motiverad. Framför allt gällde detta i ett förvaltningsläge, där jobbet för att hålla dem uppdaterade över tiden skulle kräva mycket tid och troligen inte ge lika mycket tillbaka. Några deltagare påpekade dock att det kan finnas en nytta med vissa diagram i ett förvaltningsläge, men att detta skulle kräva att de hölls uppdaterade. Tidsaspekten visar sig här vara viktig för valet av vilka modeller som bör användas till dokumentation och förvaltning.

4.2.5 Förändringar

Detta tema behandlade följande principer:

- En modell ska vara enkel att hålla uppdaterad (2)
- En modell ska vara så enkel att den är värd att underhålla (11)

Alla deltagare i fokusgruppen var rörande överrens om att modellerna måste gå att uppdatera. En respondent såg svårigheten att hålla en modell som aktivitetsdiagram uppdaterad. Som replik på detta nämndes att förändringar i den storleksordningen att de påverkar systemets logiska flöden (vilket aktivitetsdiagrammet beskriver) bör vara relativt ovanliga. Affärsregler och logik är saker som ändras väldigt sällan. Detta ansågs då som ett kriterium för modelleringen, att den bör vara övergripande och inte visa detaljer som kräver många uppdateringar.

”Det är till att man lägger det på en viss nivå då, det är väl en ganska bra måttstock när man gör dem … det ska ändras så lite att vi kan underhålla det” – Respondent D.

Detaljnivån återkom vid flera tillfällen som ett hinder för att kunna hålla diagrammen uppdaterade. Som tidigare nämnts ansågs ett designklassdiagram alltför detaljerat för att kunna hållas uppdaterat. Vissa detaljer som diagrammet visar, såsom detaljerade attribut och operationer, ansågs kunna representeras bättre på andra sätt. Attribut kunde visas i databasschemat medan operationerna mer hörde hemma i verktyg som Javadoc (ett verktyg för att dokumentera källkod i Java (Sun, 2009)).

En respondent menade att ett diagram som inte är uppdaterat ändå är långt ifrån oanvändbart men på det hela taget var respondenterna överens om att modellerna måste vara uppdaterade.

En stor del av diskussionen kom att handla kring hur modeller ska hållas uppdaterade. Deltagarna var överrens om att det behövs en rutin eller regel för uppdateringsarbetet. Vid en uppstart av ett projekt nämnde en deltagare att man borde gå igenom hur man ska hålla diagrammen uppdaterade under projektets gång. En annan respondent berättade att man måste ändra och kontrollera diagrammen innan ändringar görs i själva koden. Här nämndes även problemet att veta vilka diagram som måste uppdateras vid en förändring.

”De svåra fallen är just de där … man känner att det här är självklart, den här ändringen ska ske, det kommer inte påverka någonting. Och då går man kanske inte ens in och tittar i diagrammet” – Respondent C.

Denna svårighet återkom i diskussionen där en deltagare menade att man måste arbeta väldigt hängivet med modellerna för att inte missa uppdateringar. Detta förstärktes av en annan deltagare som menade att om man missar uppdateringar faller hela modelleringen. På något sätt måste utvecklingen organiseras så att modeller som ska vara uppdaterade också är det, då detta är en viktig del i att skapa trovärdighet för att dokumentationen går att lita på. Detta arbete kommer att ta tid från andra aktiviteter och man måste därmed avväga hur mycket tid arbetet får ta.

En annan del av diskussionen om temat handlade om ansvar. En respondent menade att en person måste vara ansvarig för förändringar kring modellerna.

”När man börjar anamma flera diagram… det är då man börjar behöva nästan en person som har kontroll över alla diagram, som ansvarar…” – Respondent D.

Detta ansågs av en annan respondent som svårt att genomföra medan en annan menade att det är nödvändigt för att det ska fungera och efterföljas i praktiken. Dessutom diskuterades svårigheterna när två personer förändrar olika delar men som båda ska uppdatera modellerna.

Förutom att en person bör vara ansvarig för förändringar i modeller behövs även ett bra IT-stöd.

”Man måste ha en väldigt bra programvara för att göra det här … verktyget man använder måste vara lättjobbat” – Respondent A.

En deltagare hade erfarenhet av kodgenerering och berättade om detta. Den allmänna uppfattningen i fokusgruppen var att kodgenerering inte fungerar fullt ut i praktiken, endast en begränsad användning i att få fram ett skelett av kod identifierades. Flera respondenter hade även tidigare haft svårigheter i att hitta ett bra program att skapa modeller i. Den allmänna åsikten i ämnet var att många program för att skapa modeller led av stora brister.

5 Diskussion

I detta avsnitt diskuterar vi resultatet där vi knyter samman principerna med det insamlade materialet från fokusgruppen. Texten är strukturerad efter principernas ursprung där vi diskuterar hur väl varje princip överrensstämde med respondenternas svar i fokusgruppen. Avslutningsvis presenteras ett ramverk där vi har gjort ett urval av principerna baserat på vårt resonemang, vilket presenteras nedan. Således innehåller ramverket de principer som visade sig vara av störst betydelse.

5.1 Principer från praktiken

Den första praktiska principen, en modell ska vara lättadministrerad (1), visade sig vara en mycket central punkt i den genomförda studien. Principen förespråkar att man behöver ett bra IT-stöd som underlättar administrationen och det uttrycktes under fokusgruppen att det verktyg som används måste vara lättarbetat. Dock har organisationen tidigare haft svårigheter att hitta ett sådant verktyg. Förutom IT-stödet behöver man även koordinera sina modeller enligt principen, även detta återkom i studien där problematiken uttrycktes av respondenterna. Bland annat nämndes svårigheten att veta vilka diagram som ska uppdateras när man gör en förändring. Deltagarna var överrens om att en rutin behövs när man ska uppdatera sina diagram och det nämndes att en person bör ha ansvaret för denna uppgift. Även om vikten av att modellerna måste vara lättadministrerade fastställdes, presenterades ingen konkret lösning på problematiken.

En annan mycket central princip i studien var att en modell måste vara enkel att hålla uppdaterad (2). Ett hinder mot denna princip visade sig vara alltför detaljerade modeller, vilket kräver åtskilliga uppdateringar. Det föreslogs som ett krav på modellerna att de ska vara såpass övergripande att uppdateringar sällan behöver ske. Ett exempel på detta är aktivitetsdiagrammet som upplevdes som svåruppdaterat. Däremot bedömdes sådana stora förändringar som kräver att diagrammet uppdateras ske väldigt sällan. Vidare bör antalet modeller begränsas för att minska arbetsbördan vid uppdateringar. Principen nämner ett konkret sätt att hålla modeller uppdaterade och det är via kodgeneratorer där modeller översätts till exekverbar kod. Detta alternativ upplevdes däremot som orealistiskt eftersom flera deltagare tidigare hade negativa erfarenheter av kodgenerering.

En modells nyttighet i ett nyutvecklingsläge eller förvaltningsläge (3) diskuterades i fokusgruppen. Här såg deltagarna en tydlig skillnad mellan vissa modeller och i vilket läge de var användbara. Exempelvis nämndes klassdiagram som användbara i förvaltningsskedet medan man inte såg nyttan av sekvensdiagrammet i samma skede. Deltagarna poängterade även vikten av att tänka igenom vilka diagram som behövs och i vilket skede. Här kan man se en poäng i att vissa diagram inte behövs hållas uppdaterade då de inte fyller någon funktion i ett förvaltningsskede. De kan dock vara användbara för att diskutera ett komplext problem i en analysfas. Förutom att en modell kan vara nyttig i nyutveckling eller i förvaltning kan den även vara nyttig i andra situationer som vid kommunikation med intressenter såsom exempelvis en supportavdelning. Det sistnämnda tyder på att principen kanske är för specifikt

inriktad på de två nämnda dimensionerna då det kan finnas fler syften med att skapa en modell.

Vikten av att poängtera om en modell är deskriptiv eller normativ (4) rådde det delade meningar om i studien. Även om skillnaden mellan krav på systemet och det faktiska systemet nämndes och diskuterades upplevdes det inte som en central fråga för organisationen. Det fanns även olika uppfattningar om vilken sida modellerna egentligen beskrev vilket innebär svårigheter att dra gränsen mellan om en modell är deskriptiv eller normativ. Då denna gränsdragning visade sig vara svår och frågan på det hela mindre central finns det rimligen viktigare principer att fokusera på.

5.2 Principer från MDD

Principen om att en modell ska abstrahera problemområdet (5) ansågs central i studien. I många diskussioner i fokusgruppen uttrycktes vikten av att skapa överblick genom modellering på en övergripande nivå. Detta är även ett vanligt syfte med att skapa modeller från första början, samt att abstrahering ger modeller som är enkla att underhålla och hålla uppdaterade. Överblickbarheten som modellerna kan ge i jämförelse med programkod är således central i studien. Vidare är vissa av modellerna bättre än andra på att skapa denna överblick och som ett sådant positivt exempel nämndes komponentdiagrammet.

Enligt en princip bör en modell vara förståelig (6), vilket innebär att problemområdet inte bör abstraheras till den grad att det skadar förståelsen. Majoriteten av deltagarna såg dock inte detta som något stort problem, då modeller i allmänhet ansågs vara begripliga, givet att det finns en förståelse för den syntax som används. Därmed ansågs det mindre viktigt att diagram kompletterades med förklarande texter. Vissa modeller ansågs basala medan somliga krävde mer tid. En modell som inte gav en snabbare förståelse av situationen än ren kod ansågs mindre användbar. Möjligheten att använda flera modeller för att beskriva en viss situation lyftes fram, då detta skulle kunna öka förståelsen. Det uttrycktes också att alla modeller inte nödvändigtvis behöver vara begripliga för alla, utan snarare att modeller har olika användningsområden. Kontentan var att den som behöver använda modellen också ofta förstår den. Principen är därmed inte av central vikt då den omfattas av många andra principer, samt att modeller i allmänhet ansågs vara enkla att förstå.

Principen gällande att en modell ska visa problemområdet på ett korrekt (7) sätt fick ett visst stöd i studien. En deltagare i fokusgruppen uttryckte vikten av korrelation mellan verkligheten och diagrammet, ett ämne som även var uppe i samband med diskussioner om hur modeller kan hållas uppdaterade. Här bör dock syftet med att skapa modeller tas i åtanke, där det kan konstateras att endast modeller som skapas i ett dokumenterande syfte behöver visa problemområdet på ett korrekt sätt över tiden. Det är därmed en fråga om vilka modeller som är värda att hålla uppdaterade, samt om det behövs flera modeller. Om problemområdet avbildas korrekt hör också i viss mån ihop med modellens detaljnivå och här visar det presenterade resultatet inga större problem med att abstrahera bort vissa detaljer. Detta upplevdes snarare som någonting positivt.

Principen om att en modell ska vara förutsägbar (8) kan inte anses som central efter den här studien. Resultatet från fokusgruppen visar att en modell visserligen bör visa vad ett system är tänkt att göra eller kommer att göra, vilket dock inte tillför någonting till redan existerande arbetssätt. En bakgrund till principen var möjligheten att kunna omvandla modellerna till exekverbar kod, något som i praktiken visade sig vara förenat med svårigheter. Eftersom detta visade sig vara problematiskt tillför denna princip inget väsentligt.

En princip handlar om att det ska vara kostnadseffektivt att skapa att skapa modeller (9). Denna princip har verifierats som central och viktig i studien. Det ansågs viktigt att en modell bör skapas för att tillföra någonting, antingen tidigt i utvecklingen eller i ett senare skede där en modell kan underlätta underhållet. Vidare har det inte visat sig vara kostnadseffektivt att hålla alltför många modeller aktuella, vilket understryker principens betydelse.

5.3 Principer från AM

Att en modell ska skapas för ett specifikt syfte (10) var en poäng som återkom flera gånger i fokusgruppen. Organisationer måste besluta om vilka modeller som ska användas och i vilka situationer de behövs. Ett antal olika syften identifierades varav ett av dessa var att kommunicera en situation eller ett problem med hjälp av modellerna. Denna kommunikation kan ske inom utvecklingsgruppen för ett specifikt fall men även för kommunikation utåt mot exempelvis supportavdelningen. Ett annat syfte med modeller är att dokumentera och använda i ett förvaltningsskede. Här är det av stor vikt att modellerna är uppdaterade och således går att lita på. En modell kan även användas för att öka förståelsen för systemet. Studien har visat att syftet med att skapa en modell är centralt och påverkar många andra faktorer.

Enligt en princip ska en modell vara så enkel att den är värd att underhålla (11), vilket i stort stämmer bra överrens med respondenternas bild. Som tidigare nämnts ansågs en hög abstraktionsnivå vara ett kriterium för att kunna hålla modellerna uppdaterade på ett lättadministrerat sätt. Att begränsa antalet modeller nämndes även som viktigt vid andra tillfällen. Denna bild förstärks av det faktum att inga konkreta lösningar för lättadministrerade modeller presenterades under fokusgruppen, vilket troligtvis innebär att man måste göra avkall på antalet modeller och dess detaljnivå. Vidare kan man se stora likheter mellan denna princip och flera andra principer i vårt ramverk. Exempelvis finns dessa likheter med principen som säger att en modell ska vara enkel att hålla uppdaterad.

Enligt en annan princip ska en modell bidra till en fullständig beskrivning av systemet (12) vilket innebär att flera modeller troligtvis behövs. Principen visade sig vara av viss vikt, dock framfördes problematiken med att skapa alltför många modeller. I fokusgruppen fanns exempel där en överlappning av modeller framstod som positivt, men även fall där denna överlappning ansågs onödig och endast resulterade i merarbete. Detta handlade i stort om risken med att skaffa sig allför många modeller, då detta kan tynga ner resterande arbete. I komplexa situationer ansågs det dock motiverat att skapa överlappande modeller då det ger en mer fullständig beskrivning. Principen kan därmed sägas innehålla viktiga inslag, men med en kritisk underton, då det inte alltid är nödvändigt med en fullständig beskrivning ett system.

Enligt principen en modell ska vara så enkel som möjligt (13) bör man förutsätta att den enklaste lösningen är den bästa. Detta blev även en central del i själva studien då överblick och detaljnivå diskuterades. Deltagarna var överrens med principen om att modelleringen bör ske på en övergripande nivå för att kunna hantera förändringar och uppdateringar. Alltför detaljerade modeller ansågs lätt bli oanvändbara och syftet med modelleringen är att få en bättre överblick än programkod. Då en modell som är så enkel som möjligt även abstraherar problemområdet finns det stora likheter mellan dessa två principer.

Principen om att en modell ska leda till fungerande mjukvara (14) fokuserar på kopplingen mellan modeller och producerandet av kvalitativ mjukvara. Principen tillåter skapandet av modeller som inte direkt bidrar till fungerande mjukvara, då dessa istället anses höja kvalitén på densamma. Detta var även något som framkom under fokusgruppen, där det visade sig att deltagare ansåg det som meningsfullt att modellera för att kunna undvika framtida misstag. Vidare handlar principen om att de modeller som skapas faktiskt ska användas, eller kunna användas som ett stöd för fungerande mjukvara och hänger således ihop med många av de övriga principerna. Därmed blir den här principens egenvärde begränsat.

Att en modell ska vara användbar i framtiden (15) ansågs viktigt av deltagarna i fokusgruppen, dock måste man vara införstådd med att alla modeller inte behöver vara användbara i detta förvaltningsläge. Vilka modeller som är användbara i ett sådant läge beror på syftet med modelleringen samt den specifika situationen. Enligt principen kan modeller även vara användbara i framtiden om viktiga nyckelpersoner inte finns tillgängliga, vilket stämmer bra överrens med deltagarnas bild. Vissa modeller är således viktiga i förvaltningsläget, andra i ett nyutvecklingsläge, vilket även fastslås av andra principer.

5.4 Ramverk

Utifrån det empiriska materialet samt den tidigare diskussionen om principerna presenterar vi härmed vårt ramverk. I detta ramverk har de ursprungliga 15 funna principerna krympt till bara sex stycken. Dessa sex principer är de som visat sig vara absolut viktigast i vår fallstudie och därmed utgör essensen i ramverket. Borta är även indelningen av principerna i olika fack, då vi har kombinerat de principer som visat sig mest användbara, oavsett deras ursprung. Vidare har ett antal principer slagits samman och bildat en mer övergripande princip för att minska överlappningen och vissa av de ursprungliga principerna har utelämnats då deras bidrag visade sig vara av mindre betydelse i fallstudien. Vi har även bytt fokus från ”en modell ska” till ”modellerna ska” eftersom det fokuset bättre speglar vad de framtagna principerna kan användas för. Nyckeln i ramverket är att alla modeller gemensamt ska uppfylla principerna, inte var för sig.

Vi ser ramverket som en vägledning för hur man på bästa sätt bör prioritera tillgängliga resurser och därigenom uppnå nytta med modellering. Genom att anpassa modelleringen till rådande förhållanden tror vi att fler organisationer kan skapa modeller som är användbara i olika skeden inom systemutvecklingen. Nedan visas de principer som vårt ramverk består av.

Modellerna ska vara lättadministrerade (A): Denna princip bygger på *en modell ska vara lättadministrerad (1)* och har likartad innebörd. För att modellerna ska vara lättadministrerade behövs en omgivande miljö som stödjer arbetet med att skapa och uppdatera modeller. Detta ställer krav på det verktyg som används, vare sig det är ett IT-baserat stöd eller någon enklare form, samt på möjligheten att koordinera sina modeller.

Modellerna ska vara enkla att uppdatera (B): Principen är en sammanslagning av e*n modell ska vara så enkel att den är värd att underhålla (11)* och *en modell ska vara enkel att hålla uppdaterad (2).* För att underlätta uppdateringsarbetet är det viktigt att modellera på en övergripande nivå, då detaljerade modeller kräver många förändringar vilket tar mycket tid. Vidare bör man göra avkall på antalet modeller samt antalet områden som modellerna täcker in.

Modellerna ska skapas för ett specifikt syfte (C): Detta är en sammanslagning av tre principer: *en modell ska vara nyttig i nyutveckling och/eller i förvaltning (3)*, *en modell ska i slutändan leda till fungerande mjukvara (14)* och *en modell ska skapas för ett specifikt syfte (10).* Att inte skapa modeller bara för skapandets skull kan inte nog poängteras. Det måste således finnas en anledning till modellskapandet, behöver man den vid nyutveckling, förvaltning, eller för något annat syfte?

Modellerna ska abstrahera problemområdet (D): Principen består av *en modell ska vara så enkel som möjligt (13)* och *en modell ska abstrahera problemområdet (5).* Principen innebär att man bör dölja onödiga detaljer i sina modeller, då detaljnivån är en viktig faktor som även påverkar andra principer. Att nå en övergripande bild över problemområdet är önskvärt i många situationer. Vidare bör det noteras att modeller i allmänhet är bra på att abstrahera ett problemområde i jämförelse med ren programkod.

Modellerna ska skapas på ett kostnadseffektivt sätt (E): Principen bygger på e*n modell ska vara kostnadseffektiv att skapa (9)* och innebär att man ska ha nytta med de modeller man skapar. Med hjälp av modellen ska ett problem kunna lösas på ett mer kostnadseffektivt sätt än om man inte hade modellen. Man måste således uppskatta vilka modeller man behöver och har nytta av i en viss situation samt tänka på att alltför många modeller leder till problem när dessa ska uppdateras. Principen hänger därför ihop med *modellerna ska skapas för ett specifikt syfte (C).*

Modellerna ska ge en tillräcklig beskrivning av systemet (F): Principen är en sammanslagning av *en modell ska visa problemområdet på ett korrekt sätt (7)* och *en modell ska bidra till en fullständig beskrivning av systemet (12).* Enligt principen är det viktigt att beskriva systemet eller situationen korrekt och för detta arbete behövs troligtvis flera modeller. Man bör således förstå att modeller kan skapas för olika syften samt att en viss överlappning kan leda till en större förståelse av systemet. Vidare är ordet tillräckligt relativt vilket innebär att mängden modeller beror på den aktuella situationen. Om situationen är komplex behövs troligtvis fler modeller än vid en enklare situation.

Tabell 5.1 beskriver vilka av de ursprungliga principerna som ligger till grund för vårt ramverk. De kursiverade principerna är de som är utelämnade p.g.a. en begränsad betydelse i studien och finns därför inte med i det slutgiltiga ramverket.

Ursprunglig princip	Ny princip
En modell ska vara lättadministrerad (1)	Modellerna ska vara lättadministrerade (A)
En modell ska vara enkel att hålla uppdaterad (2)	Modellerna ska vara enkla att uppdatera (B)
En modell ska vara nyttig i nyutveckling och/eller i förvaltning (3)	Modellerna ska skapas för ett specifikt syfte (C)
En modell ska beskriva systemet (deskriptiv) eller de krav som finns på det (normativ) (4)	*Utelämnad*
En modell ska abstrahera problemområdet (5)	Modellerna ska abstrahera problemområdet (D)
En modell ska vara förståelig (6)	*Utelämnad*
En modell ska visa problemområdet på ett korrekt sätt (7)	Modellerna ska ge en tillräcklig beskrivning av systemet (F)
En modell ska vara förutsägbar (8)	*Utelämnad*
En modell ska vara kostnadseffektiv att skapa (9)	Modellerna ska skapas på ett kostnadseffektivt sätt (E)
En modell ska skapas för ett specifikt syfte (10)	Modellerna ska skapas för ett specifikt syfte (C)
En modell ska vara så enkel att den är värd att underhålla (11)	Modellerna ska vara enkla att uppdatera (B)
En modell ska bidra till en fullständig beskrivning av systemet (12)	Modellerna ska ge en tillräcklig beskrivning av systemet (F)
En modell ska vara så enkel som möjligt (13)	Modellerna ska abstrahera problemområdet (D)
En modell ska i slutändan leda till fungerande mjukvara (14)	Modellerna ska skapas för ett specifikt syfte (C)
En modell ska vara användbar i framtiden (15)	*Utelämnad*

Tabell 5.1: Sammansättning av principer till ett ramverk

6 Slutsats

Vi vill börja denna del med att återkoppla till de tidigare redovisade frågeställningarna och syftet.

Vilka principer för modellering är tillämpbara i fallstudiens miljö?

Denna fråga besvaras av de 15 principer som finns redovisade i stycke 4.1 i uppsatsens resultatdel. Vi finner att alla dessa principer är tillämpbara i fallstudiens miljö. Denna slutsats är dock mindre intressant så länge ingen utreder hur väl principerna faktiskt fungerar i praktiken, vilket är fokus för vår andra fråga.

Hur väl fungerar dessa principer som en vägledning för modellering i praktiken?

Det ovan redovisade ramverket utgör svaret på denna frågeställning. Ramverket är, i förhållande till svaret på föregående fråga, förfinat, sammanslaget och anpassat efter hur väl principerna kan fungera som en vägledning i praktiken.

Enligt ramverket och vad vi har kommit fram till som viktigt att fokusera på vid skapandet av modeller, bör modellerna vara lättadministrerade, enkla att hålla uppdaterade och skapade för ett specifikt syfte. Vidare bör modellerna abstrahera problemområdet, skapas på ett kostnadseffektivt sätt samt ge en tillräcklig beskrivning av systemet.

Vi vill även knyta an till det syfte som framfördes i början av uppsatsen, där vi i stort ville beskriva hur en kombination av principer kan användas av en organisation för att uppnå mer nytta med modellering. Genom att besvara frågeställningarna har vi kommit fram till en beskrivning av denna kombination. Vi ville även visa att en organisation kan uppnå mer nytta med modellering om den anpassar sig till den specifika situationen. Då ramverket består av principer som utan tvekan kan anpassas till olika situationer tycker vi därmed att det framförda syftet har uppnåtts.

6.1 Utvärdering av studien

Såhär i efterhand kan man diskutera det resultat som vi har kommit fram till. Då den genomförda studien är lokalt förankrad kan man fundera över dess generaliserbarhet. Det finns en risk att organisationen i studien hade speciella förhållanden som inte gäller för andra. Vi har dock inte hittat något som tyder på detta utan vi anser att fler organisationer bör kunna ta del av och ha nytta av vårt resultat. Utöver detta kan fokusgruppen som huvudsaklig datainsamlingsmetod diskuteras. Genom de frågor som vi ställde kan vi som diskussionsledare ha påverkat respondenternas syn på modeller. Vidare kan de modeller som vi skapade och visade upp under fokusgruppen ha påverkat resultatet, om det skulle visa sig att dessa var felaktiga, otydliga eller ofullständiga. Vi fick dock inga sådana indikationer vare sig under eller efter arbetets gång. Slutligen kan vårt antagande om UML-diagrammens användbarhet i studien ha påverkat resultatet.

6.2 Förslag på fortsatt forskning

Under arbetets gång har det blivit tydligare för oss att organisationer kan ha problem med att praktiskt arbeta med modeller, vilket ger ett antal förslag på vidare studier inom området. Det första och mest givna förslaget är att testa hur vårt föreslagna ramverk fungerar i praktiken. I vår studie har vi kommit fram till ramverket som ett resultat av studien, vi har däremot inte testat huruvida ramverket i sin helhet är applicerbart i praktiken. Vi föreslår även att man genomför utvärderingar i andra situationer. Vi har inriktat oss på ett mindre mjukvaruutvecklande företag. Det vore intressant med en studie som är gjord på ett eller flera större företag med olika karaktärsdrag. Utöver detta kan man undersöka om vårt antagande om UML-diagrammens användbarhet är rimligt. En utökad utvärdering som inkluderar alla UML-diagram enligt UML 2-standarden vore således intressant.

7 Referenser

Agile Alliance. (2001) *Manifesto for Agile Software Development*. http://agilemanifesto.org/ [2009-05-22].

Ambler, S.W. (2003) Agile Model Driven Development Is Good Enough. *IEEE*, 20(5), s. 71-73.

Ambler, S.W. (2006) *Agile Modeling (AM) Principles v2.* http://www.agilemodeling.com/principles.htm [2009-04-20].

Ambler, S.W. (2007a) *An introduction to Agile Modeling.* http://www.agilemodeling.com/essays/introductionToAM.htm [2009-04-20].

Ambler, S.W. (2007b) *Agile Models Distilled: Potential Artifacts for Agile Modeling.* http://www.agilemodeling.com/artifacts/ [2009-04-20].

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

Boehm, B. (2002) Get Ready for Agile Methods, with Care. *IEEE Computer*, 35(1), s. 64-69.

Forward, A. (2002) *Software Documentation – Building and Maintaining Artefacts of Communication*. University of Ottawa, Kanada. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.3813&rep=rep1&type=pdf [2009-04-20].

Larman, C. (2005) *Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design and Iterative Development.* Prentice Hall, NJ, USA.

Magnusson, Å. (2004) *Användningen och nyttan av systemdokumentation*. Göteborgs Universitet, Institutionen för informatik.

Marczak, M., Sewell, M. (2002) *Using Focus Groups for Evaluation*. University of Arizona, AZ, USA. http://ag.arizona.edu/fcs/cyfernet/cyfar/focus.htm [2009-05-13].

Mathiassen, L., Munk-Madsen, A., Nielsen, P.A., Stage, J. (2001) *Objektorienterad analys och design*. Studentlitteratur, Lund.

McNamara, C. (2006) *Basics of Conducting Focus Groups.* http://managementhelp.org/evaluatn/focusgrp.htm [2009-04-27].

OMG. (2005) *Introduction to OMG's Unified Modeling Language.* http://www.omg.org/gettingstarted/what_is_uml.htm [2009-04-20].

Parson, D., Ryu, H., Lal, R. (2007) The Impact of Methods and Techniques on Outcomes from Agile Software Development Projects*, IFIP*, 235, s. 235-249.

Patel, R., Davidsson, B. (2003) *Forskningsmetodikens grunder*. Studentlitteratur, Lund.

Preece, J., Rogers, Y., Sharp, H. (2007) *Interaction Design*. Wiley, West Sussex, England.

Selic, B. (2003a) The Pragmatics of Model-Driven Development. *IEEE*, 20(5), s. 19-25.

Selic, B. (2003b) Model-Driven Development of Real-Time Software Using OMG Standards. *Proceedings of the Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*.

Sun. (2009) *Javadoc Tool.* http://java.sun.com/j2se/javadoc/ [2009-05-20].

Uhl, A. (2003) Model Driven Architecture Is Ready for Prime Time. *IEEE*, 20(5), s. 70-72.

8 Bilagor

8.1 Bilaga 1 – UML-diagrammens användbarhet i fallstudien

Vi har gjort ett antagande om vilka diagram enligt UML 2-standarden som vi tror passar i fallstudien. Vi gör detta antagande för att begränsa studiens omfattning och nedan beskrivs de inkluderade och exkluderade diagrammen.

8.1.1 Inkluderade diagram

Aktivitetsdiagram (Activity Diagram): Aktivitetsdiagram visar både sekventiella och parallella aktiviteter och är användbart för att modellera affärsprocesser, dataflöden och algoritmer (Larman, 2005). I vårt fall kommer det att vara viktigt att förstå affärsprocesser, regler och logik som systemet ska hantera.

Klassdiagram (Class Diagram): Ett klassdiagram visar en överblick av problemområdet och beskriver relationer mellan klasser (Mathiassen et al., 2001). En viktig del i vår fallstudie är att hantera befintliga objekt och hur de hänger ihop med andra. Då klassdiagram är ett bra sätt att abstrahera onödiga detaljer, tror vi att detta diagram är lämpligt att använda. Det går även att använda i olika detaljnivåer vilket är praktiskt. Dessutom är det ett vanligt förekommande diagram som många känner till vilket gör det lättförståeligt.

Komponentdiagram (Component Diagram): Enligt Larman (2005) är ett komponentdiagram likt ett klassdiagram och det är sällan man behöver göra en skillnad. Det finns dock undantag, bland annat om gränssnitt är viktigt, då diagrammet visar hur systemkomponenter kommunicerar med andra komponenter utanför systemets gränser, vilket är en intressant aspekt i vår fallstudie.

Objektdiagram (Object Diagram): Ett objektdiagram kan vara användbart för att beskriva komplexa relationer mellan klasser i de lägen där klassdiagrammen blir för abstrakta (Ambler, 2007b). Ett objektdiagram kan visa en ögonblicksbild över en komplex situation vilket är klart intressant i vår fallstudie.

Sekvensdiagram (Sequence Diagram): Ett sekvensdiagram visar en dynamisk bild över systemet vilket Larman (2005) slår fast är en viktig del av modelleringsarbetet. I vårt fall kommer det att finnas situationer där långa flöden kan vara viktiga att klargöra vilket gör diagrammet intressant. Det bör dock poängteras att det endast bör användas för komplexa sekvenser och således inte alla sekvenser för ett system.

Tillståndsdiagram (State Machine Diagram): Ett tillståndsdiagram visar intressanta tillstånd för ett objekt samt dess reaktion på händelser som sker (Larman, 2005). I vår fallstudie kommer det att finnas många objekt som reagerar på händelser och byter tillstånd vilket ger en komplex situation. Genom att använda tillståndsdiagram tror vi oss kunna hantera denna komplexa situation.

Användarfallsdiagram (Use Case Diagram): Användarfall rent generellt är ett vanligt sätt att beskriva hur ett system används. Ett användarfall innebär att man abstraherar interaktionerna mellan systemet och de aktörer som interagerar med det samma (Mathiassen et al., 2001). Genom att specificera användarfall fångas flera typer av krav på systemet, dock främst funktionella (Larman, 2005). Tanken med ett användarfallsdiagram är att ge en översiktlig bild över användarfall och aktörer vilket gör det lämpligt i vår fallstudie.

8.1.2 Exkluderade diagram

Kommunikationsdiagram (Communication Diagram): Kommunikationsdiagram hette tidigare collaboration diagram men bytte namn i och med UML 2. (Ambler, 2007b). Tillsammans med sekvensdiagram bildar kommunikationsdiagram en grupp som brukar kallas interaktionsdiagram. Skillnaden mellan de två är att sekvensdiagram har en mer utbredd notation och semantik medan kommunikationsdiagrammets största fördel är att det tar mindre plats och är enklare att förändra i ett tidigt stadium (Larman, 2005). Då sekvensdiagram är mer utbrett och vanligare tror vi att de passar bättre in i vår studie och väljer därmed att exkludera kommunikationsdiagrammet.

Kompositstrukturdiagram (Composite Structure Diagram): Detta diagram kan användas för att beskriva vilka instanser som ingår samt dess roll i ett visst samröre (Ambler, 2007b). Då diagrammets nytta har ifrågasatts av bland annat Ambler (2007b) samt att många delar kan beskrivas med hjälp av andra diagram väljer vi att exkludera det i studien.

Deploymentdiagram (Deployment diagram): Diagrammet visar hur olika element placeras in i systemets fysiska arkitektur (Larman, 2005). Då diagrammet fokuserar på indelning i lager m.m. är det för tekniskt för att passa in i vårt fall.

Interaktionsöversiktsdiagram (Interaction Overview Diagram): Detta diagram sammanfattar andra interaktionsdiagram (sekvensdiagram och kommunikationsdiagram) och visar hur dessa är relaterade när det kommer till logik och processflöden (Larman, 2005) . Det kan med andra ord vara intressant för att ge en överblick men bör endast användas om situationen är rörig eller komplex, då risken för överlappning är stor då det kombinerar andra diagram. På grund av detta antar vi att diagrammet kommer bli mindre användbart i vår studie.

Paketdiagram (Package Diagram): Ett paketdiagram används ofta för att illustrera den logiska arkitekturen och placera in paket och delsystem i olika lager (Larman, 2005). Diagrammen kan vara intressanta för att ge en överblick om det finns ett behov att gruppering. Diagrammet blir dock mer intressant ju närmare implementering man kommer och vi tror därför inte att det kommer till användning i vår fallstudie

Timingdiagram (Timing Diagram): Timingdiagram används för att utforska ett eller flera objekts beteende över en given tidsperiod (Ambler, 2007b). Diagramtypen kan vara intressant om man har objekt som påverkas och förändras över tiden. I vårt fall tror vi inte att detta är centralt vilket leder till en exkludering av diagrammet.