Use Case - en teknik för att representera krav i RE-processen

(HS-IDA-EA-01-406)

Johanna Enocksson (a97johen@student.his.se)

Institutionen för datavetenskap
Högskolan i Skövde, Box 408
S-54128 Skövde, SWEDEN

Examensarbete på det dataekonomiska programmet under vårterminen 2001.

Handledare: Alexander Backlund

Use Case - en teknik för att representera krav i RE-processen

Examensrapport inlämnad av Johanna Enocksson till Högskolan i Skövde, för Kandidatexamen (B.Sc.) vid Institutionen för Datavetenskap.

2001-06-05

Härmed intygas att allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och att inget material är inkluderat som tidigare använts för erhållande av annan examen.

Signerat: _____

Use Case - en teknik för att representera krav i RE-processen

Johanna Enocksson (a97johen@student.his.se)

Sammanfattning

Detta examensarbete ger en introduktion till Requirements Engineering (RE), kravprocessen i informationssystemutveckling. Det slutliga informationssystemet är i stor utsträckning beroende av en effektiv RE-process. Ett framgångsrikt informationssystem beror på hur väl det färdiga systemet stämmer överens med användarens och andra intressenters behov och förväntningar. Use Case är en teknik som kan användas i RE-processen för att representera krav. Arbetets syfte är att finna de för- och nackdelar som finns med Use Case för att representera krav. Dessutom undersöka hur kravrepresentationen anpassas utefter de för- och nackdelar som framkommit.

Det som presenteras i examensarbetet visar på hur systemutvecklare uppfattar Use Casetekniken utifrån dennas praktiska erfarenheter. Av resultatet framgår en rad olika för- och nackdelar och dessutom vid vilka tillfällen respondenterna valt att komplettera Use Case med annan teknik. Avslutningsvis diskuteras genomförandet av arbetet samt en diskussion kring resultatet av undersökningen.

Nyckelord: Requirements Engineering, Use Case, Informationssystemsutveckling, Kravrepresentation

Innehållsförteckning

Bilaga 8: Intervju – Respondent 7

Bilaga 9: Intervju – Respondent 8

Bilaga 10: Intervju – Respondent 9

1 Introduktion

Utveckling av datorbaserade system har sedan 1960-talet lidit av problem som har gjort att utvecklingen av mjukvara misslyckats (Kotonya och Sommerville, 1998). Enligt Kotonya och Sommerville (1998) är kravhantering en av huvudorsakerna till problemen och nedan ges exempel på problem som kan påverka mjukvarans resultat.

- Kraven avspeglar inte kundens behov.
- Kraven är inkonsistenta och/eller ofullständiga.
- Brister och fel upptäcks inte förrän systemet har implementeras.
- Missförstånd som uppstår mellan kunder, de som arbetar fram kraven för systemet och de som implementerar det nya systemet.

Problem kan generera misslyckanden i form av försenad leverans, budget som spricker, system som inte möter slutanvändarnas behov och inte heller möter de krav som organisationen ställt på systemet (Kotonya och Sommerville, 1998). Misslyckanden som dessa är vanligt förekommande i dagens mjukvaruutveckling (Kotonya och Sommerville, 1998; Regnell, 1999). Ibland beror misslyckade mjukvaruprojekt på anställda eller mjukvaruutvecklare, dock är huvudorsaken ofta relaterade till krav (Kotonya och Sommerville, 1998; Leffingwell och Widrig, 2000). Leffingwell och Widrig (2000) menar vidare att minst en tredjedel av alla projekt får problem som är direkt relaterade till kravhantering som kravinsamling, förståelse för problemet och dokumentering av krav.

För att systemutvecklare ska kunna utveckla ett system som är tillfredsställande för alla intressenter, krävs det att det framgår från de olika intressenterna vad de förväntar sig av systemet (Loucopoulos och Karakostas, 1995). Det är viktigt att kraven som intressenterna ställer på systemet blir korrekt uppfattade, att de dokumenteras rätt och slutligen valideras på ett korrekt sätt (Pohl, 1996). Processen som tar fram, specificerar och validerar krav kallas *Requirements Engineering* (RE) och är en process som används i systemutvecklingsarbetet för att uppnå de förväntningar som olika intressenter ställer på systemet (Regnell, 1999).

En teknik som används inom RE–processen är *Use Case*, vilken utvinner, dokumenterar och validerar krav (Regnell, 1999). Detta arbete kommer att utgå ifrån att undersöka hur väl Use Case representerar (beskriver och dokumenterar) de olika krav som ställs på systemet. Undersökningen behandlar de för- och nackdelar systemutvecklare upplever att det finns med Use Case vid kravrepresentation och om för- och nackdelarna anpassas till kravrepresentationen. Det finns olika sorters Use Case och undersökningen koncentrerat på Use Case utifrån *Unified Modeling Language* (UML) därför att UML är ett standardspråk och det är ett vanligt förekommande modelleringsspråk i dagens systemutveckling. Kvaliteten i de tidiga faserna inom systemutveckling är oerhört viktig för systemets slutgiltiga resultat. Detta är orsaken till att det är viktigt att göra denna undersökning för att undvika att tekniken inte används vid fel tillfällen.

1.1 Rapportens uppbyggnad

I kapitel 2, Bakgrund, beskrivs centrala begrepp för att ge en förståelse för det problemområde som kommer att behandlas i detta examensarbete. Kapitel 3, Problembeskrivning, ger ytterligare en beskrivning av problemområdet och dessutom presenteras den specifika fråga som arbetet kommer att behandla, avgränsningar för problemet samt förväntat resultat. I kapitel 4, Metod, presenteras olika tekniker som är möjliga att använda för att besvara problempreciseringen och sedan presenteras den

teknik som kommer att användas för att samla in material under undersökningen av detta arbete. Dessutom ges en beskrivning av hur undersökningen är tänkt att genomföras och vissa förberedelser inför undersökningen. Kapitel 5, Genomförandet, redovisar hur undersökning genomfördes och erfarenheter från undersökningen. I kapitel 6, Materialpresentation, presenteras en sammanfattning av det material som framkom under undersökningen. Därefter i kapitel 7, Analys, analyseras och värderas det insamlade materialet. Efter att materialet analyserats presenteras resultatet av undersökningen i kapitel 8, Resultat. Slutligen i kapitel 9 sker en diskussion kring resultatet samt erfarenheter från arbetet. Förslag på framtida arbeten presenteras även under kapitel 9.

2 Bakgrund

Bakgrunden kommer att ge en presentation av de områdena om ligger till grund för detta examensarbete. Först ges en kort introduktion till området informationssystem och systemutveckling. Därefter förklaras vad ett krav är, vilka olika sorters krav det finns och varför kravhanteringen är viktig under en systemutvecklingsprocess. För hantering av krav finns ett gemensamt begrepp, Requirements Engineering (RE)[1], vilket liksom de olika faserna i RE-processen kommer att beskrivas. Bakgrundens sista del kommer att redogöra för en teknik, Use Case[2], som kan användas inom Requirements Engineering för att representera och modellera krav.

2.1 Informationssystem och Systemutveckling

Som världen ser ut idag, med den hårda konkurrens som existerar mellan olika verksamheter, är det enligt Andersen (1994) viktigt med ett *informationssystem* som ger verksamheten fördel i konkurrensen. En väl vedertagen definition av informationssystem är: ”Ett informationssystem är ett system för insamling, bearbetning, lagring, överföring och presentation av information” (Andersen, 1994 s15). Informationssystemet existerar för att tjäna en verksamhet, likaså är det en del av verksamheten. Det innebär att det inte finns några generella informationssystem, det vill säga att informationssystemet skall formas efter verksamheten där det ska ses i förhållande till en viss uppgift i en bestämd verksamhet. (Andersen, 1994)

Systemutveckling kan ses som en process med olika faser, vilka återges i en modell, *livscykelmodellen* (Andersen, 1994). Modellen beskriver de olika faserna för ett systems livscykel, från att verksamheten analyseras, designas, implementeras, underhåll och slutar inte förrän systemet avvecklas. En liknande livscykelmodell beskrivs av Britton och Doake (1993) och de menar att livscykeln var en följd av 1960- och 1970-talets försök att utveckla stora komplexa informationssystem.

Ofta uppstår problem under systemutvecklingsarbetet. Mer än hälften av alla mjukvaruprojekt överskrider sin budget (Leffingwell och Widrig, 2000). Andra problem är till exempel försenade leveranser, system som inte stämmer överens med användarnas behov och att systemen inte utnyttjas till fullo av användarna (Britton och Doake, 1993). Detta är en följd av de problem som uppkommer under systemutveckling, det finns olika orsaker till att system tenderar att fallera, dock är huvudorsaken ofta relaterade till krav (Kotonya och Sommerville, 1998; Leffingwell och Widrig, 2000). Dessa problem lever kvar även i dagens systemutveckling, vilket är både tids- och kostnadskrävande (Kotonya och Sommerville, 1998; Regnell, 1999).

2.2 Krav

Krav är viktigt i systemutvecklingsprocessen därför att kraven avgör systemets acceptans och användbarhet och hur framgångsrikt systemet ska bli (Loucopoulos och Karakostas, 1995). Om ett system kan betraktas som framgångsrikt beror på systemets förmåga att möta de behov och förväntningar som användare och kunder ställer på systemet

[1] I svenska språket finns det inget motsvarande begrepp för Requirements Engineering och därför används det engelska begreppet genom denna rapport.

[2] Det svenska begreppet för Use Case är användningsfall, Use Case är ett väl vedertaget begrepp inom systemutveckling därför används det engelska uttrycket i denna rapport.

(Loucopoulos och Karakostas, 1995). Framgångsrik kravhantering identifierar och definierar, på ett tidigt stadium i utvecklingsprocessen, de egenskaper som systemet ska uppfylla (Karlsson, 1996). Generellt sett ska kraven tala om vad systemet förväntas att göra, en specifikation på den service som systemet kan åstadkomma (Kotonya och Sommerville, 1998). Enlig Karlsson (1996) finns det krav som beskriver både *vad* ett programvarusystem ska göra och *hur* det ska göras. Det sker ibland misstolkningar av de krav som beskriver hur något ska utföras. Dessa krav uppfattas ibland som design eller konstruktion istället för ett krav. Det finns tillfällen då det nästintill är omöjligt att utesluta en beskrivning av hur kravet ska genomföras, dock ska det inte begränsa lösningsrymden mer än nödvändigt.

Begreppet krav används i många olika betydelser och tolkas olika av olika personer (Karlsson, 1996). Ett krav ska betraktas som en önskvärd egenskap hos ett system och inte som ett villkor som ovillkorligen ska uppfyllas. Definitionen av krav enligt IEEE-standard 610 används av ett flertal författare, bland annat av Loucopoulos och Karakostas (1995), Pohl, (1996) och Regnell (1999). De definierar krav som:

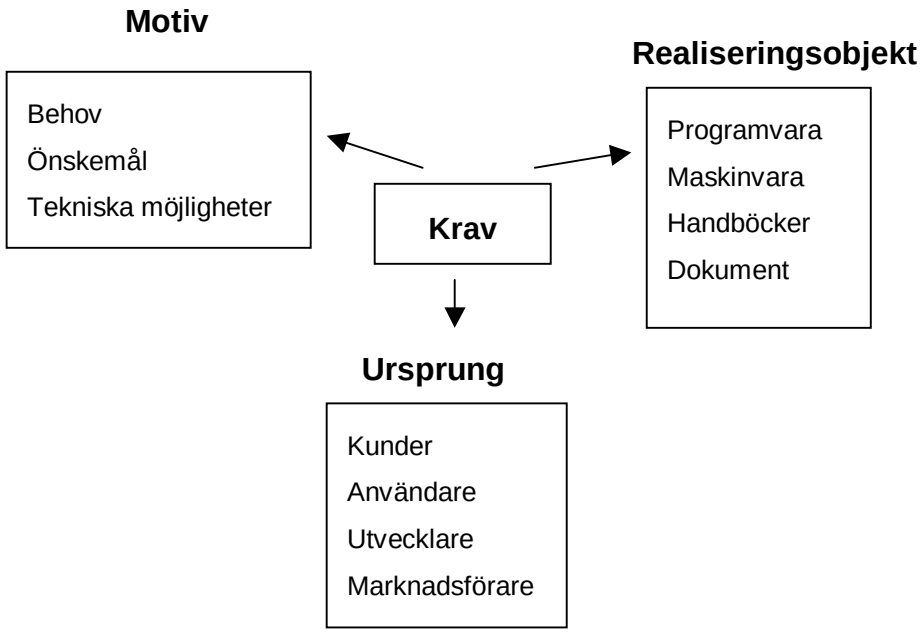
- En funktion eller egenskap som användaren behöver för att lösa ett problem eller uppnå ett mål.
- En funktion eller egenskap som systemet måste uppnå för att tillfredsställa kontrakt, standard, specifikation eller annat formellt dokument.
- Ett dokument som representerar en funktion eller egenskap såsom i någon av ovanstående punkter.

Vid en närmare granskning av definitionen ovan, beskriver den krav i tre punkter. Den första punkten avser att lägga fokus på användarens behov av systemet. Det är inte endast slutanvändarnas behov som ska tillgodoses, det finns även andra intressenter som kund, och utvecklare av systemet vars behov ska tillgodoses för att ett system skall bli framgångsrikt (Karlsson, 1996). Därför kan begreppet *användare* i definitionen betraktas som lite vagt beskrivet, då den första punkten borde inkludera alla intressenters behov. I andra punkten står det att det är en funktion eller egenskap som systemet *måste* uppnå för att tillfredsställa exempelvis ett kontrakt. Detta kan jämföras med vad som nämnt tidigare, enligt Karlsson (1996), att ett krav inte ska ses som ett villkor som ovillkorligen ska uppfyllas. Ibland kan användare ha krav som beroende på teknik och vilken kostnad det medför, inte är genomförbara. Dock kan det finns standards och lagar som måste tillgodoses av ett programvarusystem. Utifrån denna givna information kan ett krav i vissa avseenden vara tvunget att uppfyllas och i andra fall exempelvis ses som en önskvärd egenskap hos ett system.

Enligt (Karlsson, 1996) är krav en önskvärd egenskap som har ett ursprung, ett motiv och ett realiseringsobjekt. Förklaringen nedan av de olika begreppen till figur 1 är hämtat från Karlsson (1996). Ett krav har alltid ett *ursprung* i en eller flera intressenter. Exempel på intressenter är kunder, användare, utvecklare, marknadsförare och testare. Det är intressenterna som föreslår kraven. Dessutom har krav också alltid ett *motiv* som bidrar till att kravet existerar. Exempel på motiv är :

- Kravet speglar användarens uttryckta behov.
- Kravet motsvarar en teknisk möjlighet som gör att en särskild uppgift för användaren kommer att effektiviseras.

Slutligen har kravet alltid något eller några *realiseringsobjekt*, exempelvis för en programvara är programvaran ett realiseringsobjekt andra alternativ är maskinvara och handbok. Det måste alltid finnas minst ett realiseringsobjekt för varje krav.



Figur 1. Ett kravs tre beståndsdelar (från Karlsson, 1996 sidan 9).

2.2.1 Kategorisering av krav

Den mest traditionella kategoriseringen av krav vad det gäller programvarusystem är uppdelningen av krav i funktionella och ickefunktionella krav (Karlsson, 1996). *Funktionella krav* uttrycker hur systemet beter sig, systemets funktionalitet, då det förses med input eller vid olika tillstånd (Leffingwell och Widrig, 2000; Karlsson, 1996). Detta innebär att beskrivningen av de funktionella kraven ofta blir mycket omfattande (Karlsson, 1996). Enligt Karlsson (1996) beskriver de funktionella kraven de tjänster som programvarusystemet ska tillhandahålla användaren. Dessa krav är vanligtvist händelsebaserade, när en användare gör X gör systemet Y (Leffingwell och Widrig, 2000; Karlsson, 1996). *Ickefunktionella krav* är viktiga när det gäller kvalitet på ett systems funktioner (Leffingwell och Widrig, 2000). De används för att beskriva övriga egenskaper som ett programvarusystem ska uppfylla, men som inte kan karaktäriseras som tjänster (Karlsson, 1996). De ickefunktionella kraven beskriver användbarhet, tillförlitlighet, effektivitet, kapacitet och säljbarhet hos ett system (Leffingwell och Widrig, 2000; Karlsson, 1996). Det är svårt att på ett entydigt sätt skilja de funktionella kraven från de ickefunktionella kraven. Det är inte ovanligt att ett krav först klassas som ickefunktionellt krav för att sedan förändras till ett funktionellt krav (Loucopoulos och Karakostas, 1995; Karlsson, 1996).

Ett annat sätt att se på krav är dess förmåga att tillfredsställa de olika intressenterna (Karlsson, 1996). Ur detta perspektiv finns de tre olika sorters krav nämligen sensationella, normala och förväntade krav. *Sensationella krav* är krav som inte är uttalade av intressenterna, men som kommer att leda till stor tillfredsställelse om programvarusystemet uppfyller dem. Eftersom det är krav som inte är förväntade ökar tillfredsställelsen då kraven inte realiseras. *Normala krav* är ofta uttalade av kund eller användare. Dessa krav kännetecknas av att graden av kund- och användartillfredsställelse är direkt beroende av hur väl kraven blivit tillgodosedda. Ju bättre systemet tillgodoser användarens och kundens krav desto högre kund- och användartillfredsställelse. *Förväntade krav* är ej uttalade av kund eller användare men kommer att leda till missnöje

om systemet inte uppfyller de förväntade kraven. Detta är ofta grundläggande behov hos kunden och så triviala att de inte blir uttalade.

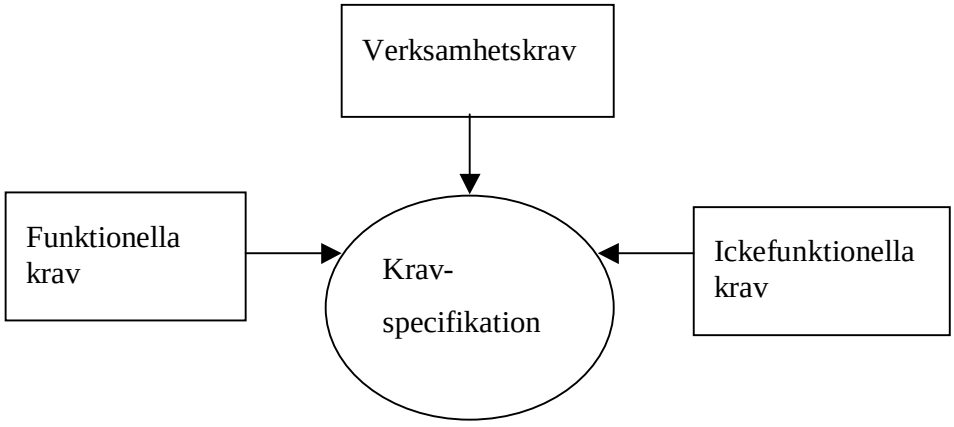
En nyckelkomponent i kravhanteringen är kravspecifikationen, vilken ska dokumentera de krav intressenterna ställer på systemet. Kravspecifikationen ska användas för att nå ett framgångsrikt programvarusystem som möter intressenternas behov och förväntningar. (Loucopoulos och Karakostas, 1995)

2.2.2 Kravspecifikation

De dokument med insamlade krav som framkommit under systemutredningen och systemanalysen kallas vanligtvis för en *kravspecifikation* (Loucopoulos och Karakostas, 1995). Kravspecifikation ska tala om vad systemet ska göra och inte hur det ska göras (Pohl, 1996). Det finns olika anledningar till att utveckla en kravspecifikation. Loucopoulos och Karakostas (1995) och Karlsson (1996) förklarar att specifikationen kan användas i syfte att:

- Kommunicera fram förståelse för problemdomänen, verksamheten och det tänkta programvarusystemet.
- Fungera som ett kontrakt mellan kunden och systemutvecklare. Det kan vara speciellt viktigt då någon utomstående systemutvecklare anlitas för att utveckla informationssystemet.
- Användas som ett underlag för att uppskatta slutprodukten i förhållande till de krav som ställts på systemet, likaså spelar kravspecifikationen en viktig roll då systemet testas.

En traditionell syn på kravspecifikation enligt Loucopoulos och Karakostas (1995) beskriver bara funktionella krav vilka beskriver systemets olika funktioner. En kravspecifikation enligt Pohl (1996) ska innehålla både de funktionella kraven och de ickefunktionella kraven. Det finns olika syn på vad en kravspecifikation ska innehålla men den som presenteras genom denna rapport är av bred karaktär, det vill säga innehåller inte bara de funktionella kraven (se figur 2) och är hämtad ur Loucopoulos och Karakostas (1995). Förutom de funktionella kraven ska domänen där det tänkta systemet skall operera förklaras, vilket är en förutsättning för att skapa en gemensam förståelse för problemet av systemets kunder, användare och utvecklare (verksamhetskrav). Dessutom skall de ickefunktionella kraven beskrivas.



Figur 2. En konceptuell bild av ett ramverk för en kravspecifikation (från Loucopolous och Karakostas 1995, sidan 12).

Det finns en mängd önskvärda egenskaper hos en kravspecifikation. Det är flera författare, exempelvis Pohl (1996), Loucopoulos och Karakostas (1995), Leffingwell och Widrig (2000) som hänvisar till IEEE Standard 830, vilken säger att en kravspecifikation bör vara:

Fullständig

En kravspecifikation betraktas som fullständig då alla krav som är av vikt för intressenterna, både funktionella och ickefunktionella krav ingår (Leffingwell och Widrig, 2000).

Konsistent

En konsistent kravspecifikation innehåller inte några enskilda krav som på något sätt kan motsäga varandra eller skapa en konflikt dem emellan (Leffingwell och Widrig, 2000). Det får heller inte finnas något krav som står i konflikt med annan information som kontrakt och projektplan (Karlsson, 1996). Detta innebär att alla begrepp och beteckningar beskrivs på ett konsistent sätt genom hela kravspecifikationen.

Modifierbar

Kravspecifikationen anser vara modifierbar då den är så bra strukturerad att om ändringar ska göras på kraven ska det enkelt kunna genomföras utan att kravspecifikationens struktur behöver ändras (Leffingwell och Widrig, 2000). Ändringarna ska kunna utföras på ett effektivt, komplett och konsistent sätt (Karlsson, 1996).

Entydig

Med en entydig kravspecifikation avses att det endast ska vara möjlig att tolka kraven på ett sätt (Karlsson, 1996). Kraven ska beskrivas med ett sådant formellt språk att kraven inte kan tolkas olika av olika personer. Om ett krav tolkas olika av till exempel användare och utvecklare är det stor risk att det slutliga systemet inte upplevs som ett framgångsrikt system för användaren (Leffingwell och Widrig, 2000).

Verifierbar

Krav är verifierbara om det finns ett ändligt och kostnadseffektivt sätt att visa att programvarusystemet möter de önskvärda egenskaperna (Pohl, 1996).

Spårbar

Krav är spårbara om de är beskrivna och specificerade på så sätt att en hänvisning till varje enskilt krav kan göras (Karlsson, 1996). I en kravspecifikation innebär spårbarhet att det ska var möjligt att spåra tillbaka till ett krav. Exempelvis kan det uppstå ett fel då en funktion eller egenskap av systemet testas. Det ska då vara möjligt att spåra tillbaka genom kravspecifikationen till det krav där det finns information om hur funktionen eller egenskapen var tänkt att fungera (Leffingwell och Widrig, 2000; Pohl, 1996). Det ska även gå att spåra åt andra hållet det vill säga att spåra från kravet fram till design och kodning.

Användbar under utveckling, drift och underhåll av systemet

Dokumentet ska vara användbart genom hela programvarusystemets livscykel. Det vill säga att det är designat så att det är möjligt att referera till och modifiera kravspecifikationen under utveckling, drift och underhåll av systemet Leffingwell och Widrig (2000).

2.3 Requirements Engineering

Requirements Engineering (RE) är en relativt ny term som tagits fram för att täcka alla aktiviteter som är inblandade i en iterativ process för att ta fram, analysera och dokumentera en mängd krav för ett mjukvarusystem (Kotonya och Sommerville, 1998). För att nå ett lyckat resultat av utveckling av en mjukvara är kvalitén i Requirements Engineeringsprocessen en viktig faktor (Regnell, 1999). För att ett system skall vara tillfredsställande för kunden och slutanvändaren, ska systemet motsvara de förväntningar som ställts på systemet (Pohl, 1996). De behov och förväntningar som ställs på systemet är de krav som definieras under de tidiga stegen i utvecklingsprocessen som en specifikation av vad som ska implementeras (Kotonya och Sommerville, 1998). Den primära produkten som arbetas fram under RE-processen är kravspecifikationen som ska tala om *vad* systemet ska göra (Pohl, 1996; Regnell, 1999).

Det råder olika uppfattningar om innehållet i Requirements Engineering och det finns inte någon gemensam definition som de olika forskarna har enats om (Loucopoulos och Karakostas, 1995; Pohl, 1996). Loucopoulos och Karakostas (1995) definierar RE så här: “en systematisk process bestående av att utveckla krav genom en iterativ och ömsesidig process bestående av analysera problemet, dokumentera resultat från observationer i olika representationsformer och kontrollera exaktheten i den förståelse som erhållits”

En annan definition av RE är: ”de aktiviteter som är involverade i att upptäcka, dokumentera och underhålla en mängd krav för ett datorbaserat system” (Sommerville och Sawyer, 1997).

De flesta forskare är överens om att ett krav måste utvinnas, specificeras och valideras (Pohl, 1996). Detta har de olika definitionerna gemensamt, en skillnad mellan de ovan nämnda definitionerna är hur detaljerat beskrivna de är. Målet med RE är dock detsamma det är vägen dit som skiljer de olika forskarna åt (Loucopoulos och Karakostas, 1995). Locuopoulos och Karakostas (1995) definition beskriver RE som en *iterativ* process för att ta fram, dokumentera och validera krav och Sommerville och Sawyer (1997) nöjer sig med att endast nämna de olika aktiviteterna. Loucopoulos och Karakostas (1995) definition tar även upp att RE är en *ömsesidig* process, där de olika intressenterna är viktiga för att identifiera de olika behov som ställs på systemet. Definitionen innehåller även information om att resultat ska *dokumenteras* i olika former vilket möjliggör att olika personer med olika kunskaper och erfarenheter kan förstå dokumentationen. Därefter säger definitionen att de framtagna och dokumenterade kraven ska *valideras*, vilket är väldigt viktigt för att slutresultatet ska stämma överens med intressenternas behov.

I Loucopoulos och Karakostas (1995) beskrivs RE-processen i tre aktiviteter till skillnad från Kontonya och Sommerville (1998), Sommerville och Sawyer (1997) och Pohl (1996) där RE-processen är uppdelad i fyra aktiviteter. Målet är dock det samma för de olika författarna, att finna de krav som ställs på systemet och genom RE-processen undvika misslyckanden i form av försenad leverans och system som inte stämmer överens med intressenternas behov och förväntningar (Loucopoulos och Karakostas, 1995). Flera forskare refererar till definitionen av Requirements Engineering från Loucopoulos och Karakostas (1995). Då den beskriver viktiga aspekter i en systemutvecklingsprocess, exempelvis intressenters medverkan och kontroll av korrekthet i de framarbetade kraven kommer denna definition ligga som grund i den fortsatta rapporten. Utifrån Loucopoulos och Karakostas (1995) definitionen kan tre aktiviteter identifieras nämligen kravutvinning, kravspecificering och validering av krav.

2.3.1 Kravutvinning

Det första som behöver göras för att lösa någon annans problem är att ta reda på mer information om problemet (Pohl, 1996; Loucopoulos och Karakostas, 1995). Aktiviteten *kravutvinning* har till syfte att skapa kunskap och förståelse för problemedomänen och hitta de krav som ställs på det nya systemet (Loucopoulos och Karakostas, 1995; Sommerville och Sawyer, 1997; Pohl, 1996). Att utvinna krav innebär enligt Loucopoulos och Karakostas (1995) att identifiera alla källor som kan tänkas ställa krav på systemet och förvärva kraven. De olika källorna är inte bara kunder och användare, det kan var lagar, standarder och existerande system som påverkar det nya systemet (Pohl, 1996). För att finna kraven sker en kommunikation mellan systemutvecklare, kund, slutanvändare och andra som har intresse av systemet (Sommerville och Sawyer, 1997). Kommunikationen går inte enbart ut på att ta reda på vad intressenterna vill att systemet ska kunna utföra, det krävs också en noggrann analys av organisationen, domänen systemet skall verka i och hur intressenterna har tänkt använda systemet. Vidare ska dessutom de framtagna kraven ses till vilken relevans de har till det aktuella systemet (Loucopoulos och Karakostas, 1995).

Framtagning av krav är den första aktiviteten i RE-processen och fortlöper genom hela processens livscykel parallellt med kravspecificering och validering av krav (Loucopoulos och Karakostas, 1995). Kravutvinningen är en tids- och resurskrävande aktivitet men är också en viktig del för att undvika ett misslyckat mjukvaruprojekt, det vill säga att systemet inte möter intressenternas behov och förväntningar (Loucopoulos och Karakostas, 1995; Sommerville och Sawyer, 1997). Vanliga tekniker för att utvinna krav är intervjuer, scenario, prototyper och simulering (Loucopoulos och Karakostas; Pohl, 1996).

2.3.2 Kravspecificering

Syftet med *kravspecificering* är att beskriva och dokumentera de krav som har tagits fram i kravutvinningsfasen (Loucopoulos och Karakostas, 1995; Regnell, 1999). Kravspecifikationen är RE-processens mål och kan ses som ett kontrakt mellan användare och mjukvaruutvecklare som definierar de önskade funktionella och ickefunktionella beteenden på mjukvaran, utan att tala om hur det ska genomföras (Loucopoulos och Karakastas, 1995; Pohl, 1996). Kvaliteten på dokumentationen är viktig ty dokumentationen ska fungera som ett underlag för kommande utvecklingsaktiviteter eftersom den beskriver de önskvärda egenskaper som ska realiseras (Karlsson, 1996). Dokumentationen har en central roll i RE-processen och kontrollerar både framtagning och validering av krav, dokumentet kan användas vid testning för att verifiera att systemet uppfyller de önskvärda egenskaperna (Loucopoulos och Karakostas, 1995; Pohl, 1996). En viktig aspekt av kravdokumentation är som ovan nämndes, verifieringen av krav. Dokumentationen, exempelvis kravspecifikationen, ska fungera som ett diskussionsmedel mellan systemutvecklaren och användaren för att arbeta mot en version som stämmer överens med de krav som ställs på systemet (Loucopoulos och Karakostas, 1995; Leffingwell och Widrig, 2000; Britton och Doake, 1993). Under utvecklingsprocessen pågår arbete med att identifiera krav och att kontrollera att dokumentationen av kraven är fullgod för att fortsätta med designen av systemet (Regnell, 1999). Under utvecklingsprocessen och även efter implementationen är det vanligt att krav läggs till och förändras, därför är det viktigt med någon form av kravhantering genom hela systemets livstid (Regnell, 1999). Kravhanteringen är i detta fall dokumentationen av kraven. Dokumentationen fungerar som ett kontrakt på vad systemutvecklare och intressenter kommit överens om, vilket kan vara viktigt om tveksamheter kring tidigare beslut uppstår (Karlsson, 1996).

Som tidigare nämnts är kravspecifikationen målet med RE-processen. Alla specifikationer, det vill säga alla dokument, har i uppgift att bevaras konsistenta då till exempel krav ändras. Det betyder att det ska vara möjligt att referera till dokumentationen under hela systemets livscykel (Pohl, 1996). Dessutom ska dokumenten vara spårbara, vilket innebär att det ska vara möjligt att spåra utifrån och in och inifrån och ut (Leffingwell och Widrig, 2000; Pohl, 1996). Det vill säga det ska vara möjligt att spåra ett insamlat och specificerat krav till implementation och kodning. Det ska finnas spårbarhet även åt det andra hållet, det vill säga då ett fel uppstår i ett implementerat system ska det vara möjligt att spåra tillbaka till ett specifikt krav.

Under specifikationsprocessen produceras ett antal modeller som åskådliggör problemet ur olika synvinklar (Loucopoulos och Karakostas, 1995; Pohl, 1996). En av de viktigaste modellerna i kravspecificeringsfasen är specifikationsmodellen. Det är en modell som används för att kontrollera att problemet (alla de olika kraven) tolkas på samma sätt av både systemutvecklare och användarna. Att beskriva och dokumentera krav kräver kunskap om problemdomänen vilken införskaffas i kravutvinningsfasen, det krävs dessutom kunskap om organisationen och kunskap om vilka krav som är relevanta (Loucopoulos och Karakostas, 1995). Modeller används även för att visualisera och kontrollera systemarkitekturen, öka förståelse för systemet, upptäcka risker i tidigt stadie av utvecklingsarbetet och för att dokumentera de beslut som fattats (Loucopoulos och Karakostas, 1995; Britton och Doake, 1993).

2.3.3 Validering av krav

Validering av krav syftar till att intyga att kravmodellen med de specificerade kraven överensstämmer med intressenternas syfte med systemet (Loucopoulos och Karakostas, 1995; Regnell, 1995; Karlsson, 1996). En viktig aspekt i valideringsprocessen är att intressenterna ska komma överens (Karlsson, 1996). Validering av krav pågår under hela RE-processen. Att validera ett krav innebär inte endast processen att jämföra ett krav med intressenternas behov, det innebär även att validera ett nyss insamlat krav och avgöra vilken relevans kravet har för det tänkta systemet (Loucopoulos och Karakostas, 1995).

Det finns vissa krav som är problematiska att validera, exempelvis användbarhet och effektivitet (Karlsson, 1996). Ett sätt för att bland annat validera dessa krav på är att utveckla en prototyp som illustrerar hela eller en del av det kommande systemet. Prototypen skapar en utgångspunkt för diskussioner mellan de olika intressenterna, vilket är viktigt för att kontrollera att det är rätt krav som dokumenterats och att de beskrivs på rätt sätt. Prototyper innebära både tids- som kostnadsbesparingar eftersom fel identifieras tidigt och därmed minskar kostnaderna för eventuella korrigeringar.

2.4 Use Case

Use Case har till syfte att beskriva interaktionen mellan en användare och systemet och fokuserar på vad systemet kan göra för användaren (Leffingwell och Widrig, 2000). Use Case är en teknik som kan användas i RE-processen för att ta fram krav, specificera krav och validera krav. Nedan följer några olika författares definitioner på Use Case.

Rosenberg och Scott (1999, sidan 38), definierar ett Use Case som: ”a sequence of actions that an actor (usually a person, but perhaps an external entity, such as another system) performs within a system to achieve a particular goal.”

Booch, Rumbaugh och Jacobson (1999 sidan 468) definierar Use Case så här ”A description of a set of sequences of actions, including variants, that a system performs that yields an observable result of value to an actor.”

Slutligen en fokusering på definitionen av Use Case ur UML:s synvinkel som är hämtad ur Leffingwell och Widrig (2000, sidan 252). ”A use case is a description of a set of actions, including variants, that a system performs that yields an observable result of value to a particular actor.”

Vid en närmare granskning av de olika definitionerna framgår det vissa skillnader. Rosenberg och Scott (1999) säger att Use Case *är en sekvens* av händelser, medan de andra två definitionerna uttrycker att Use Case *är en beskrivning* av händelser. I den förstnämnda definitionen är Use Case ett begrepp för en sekvens av händelser medans enligt de andra två definitionerna är Use Case beskrivningen av händelserna. I UML definition av Use Case står det *en mängd av händelser*, i Rosenberg och Scott (1999) *en sekvens av händelser* och slutligen i Booch et al (1999) *en mängd av sekventiella händelser*. Här skiljer sig alla tre definitionernas fokus åt, den ena definitionen har inte med *sekventiella* händelser och de två kvarvarande definitionerna skiljer också sig åt i fokus. Rosenberg och Scott (1999) har inte med att Use Case även inkluderar de alternativa flödena, *including variants*, vilket de andra definitionerna beskriver. Rosenberg och Scott (1999) menar att aktören utför en sekvens av händelser för att denna ska uppnå ett mål. De andra definitionerna menar att systemet utför händelser som resulterar i någon form av ett observerbart värde eller tillfredställelse för kunden.

Det finns olika sorters Use Case, bland annat olika versioner av UML:s Use Case. Bland annat skiljer sig utformningen på Use Case i de olika versionerna UML 1.1 och UML 1.3 (Simons, 1998). UML:s Use Case har sitt ursprung i Jacobsons Use Case, vilket Ivar Jacobson är upphovsman till (Simons, 1998). Undersökningen i rapporten kommer att baseras på användning av Use Case beskriven utifrån Unified Modeling Language (UML)[3] (det framgår inte vilken version) presenterad av Leffingwell och Widrig (2000). Detta val beror på att det idag är allt vanligare med UML som modelleringsspråk vid systemutveckling (Leffingwell och Widrig, 2000). Use Case är en teknik i UML som kan användas bland annat för att ta fram krav och specificera krav. UML:s definition på Use Case innehåller ett antal nyckelord som kommer att förklaras för att ge en tydligare bild av vad ett Use Case är.

A set of actions, är en mängd händelser som beskriver en funktion som kan utföras av systemet (Leffingwell och Widrig, 2000). Mängden händelser framkallas då en aktör initierar ett Use Case genom att förse systemet med input. *Variants*, beskriver de alternativa händelser (flöden) som kan inträffa i ett Use Case. *System performs*, innebär att den funktionalitet som beskrivs i ett Use Case utförs av systemet. *An observable result of value*, är en viktig del av definitionen därför att resultatet av ett Use Case måste resultera i ett värde för användaren, annars är det inte något giltigt Use Case. Exempelvis är ”en student trycker på ljusknappen” inget Use Case därför att systemet inte utför något åt användaren. Däremot ”student trycker på ljusknappen och systemet tänder lampan”, är ett Use Case därför att ett värde resulteras till användaren från systemet. *A particular actor*, är den aktör (individ eller system) som initierar händelsen. En *aktör* är någon eller någonting som interagerar med systemet. Aktören är inte en del av systemet som beskrivs, utan existerar utanför systemet. En aktör motsvarar en särskild användarkategori eller en särskild roll som använder systemet. (Karlsson, 1996; Fowler och Scott, 2000; Rosenberg och Scott, 1999). Aktören kan vara både en människa eller ett externt system

[3]Unified Modeling Language (UML) kommer inte att beskrivas i detta arbete. För vidare läsning rekommenderas ”*The Unified Modeling Language User Guide*” av Booch, G och Rumbaugh, J Jacobson, I.

(Fowler och Scott 2000). En användare kan spela flera roller (vara olika aktörer), exempelvis kan en chef även vara en säljare och en orderläggare. Det är viktigt att tänka sig användaren som den roll den har, vilken särskild användarkategori, och inte för vilken jobbtitel som användaren ha (Karlsson, 1996). Likaså kan en aktör utgöras av flera användare exempelvis kan en aktör ”orderläggare” utgöras av både chefen, och ekonomiansvarig på företaget (Fowler och Scott, 2000; Rosenberg och Scott, 1999).

2.4.1 Use Case i textform och Use Casediagram

Use Case är ett effektivt sätt att hantera funktionella krav hos ett programvarusystem (Karlsson, 1996). Olika applikationsdomäner behöver olika notationer för att dokumentera Use Case (Regnell, 1999). Det finns olika sätt att beskriva Use Case på. Ett sätt är att beskriva Use Case med vanlig text, där de olika händelserna (interaktionen mellan användare och system) skrivs ned sekventiellt. Det kan även finnas alternativa händelser i ett Use Case, även dessa beskrivs i listan. (Fowler och Scott, 2000)

Exemplet beskriver i text ett Use Case, *att köpa en produkt*, hämtat ur Fowler och Scott (2000, sidan 40). Köpet av produkter sker elektroniskt.

1. Kunden läser ur katalogen och väljer ut artiklar som denna ska köpa
2. Kunden väljer att checka ut
3. Kunden fyller i leveransinformation (adress; nästa dag – eller tredagars leverans)
4. Systemet presenterar information om det totala priset på varorna, inklusive leveransavgift.
5. Kunden fyller i information om kreditkort
6. Systemet bemyndigar inköpet
7. Systemet bekräftar försäljningen omedelbart
8. Systemet skickar en bekräftelse via elektronisk post till kunden

Alternativ: Att det felas när inköpet ska bemyndigas

Vid steg sex misslyckas systemet med att bemyndiga kreditinköpet

Låt kunden en gång till fylla i kreditkortsinformationen och försök igen

Alternativ: Stamkunder

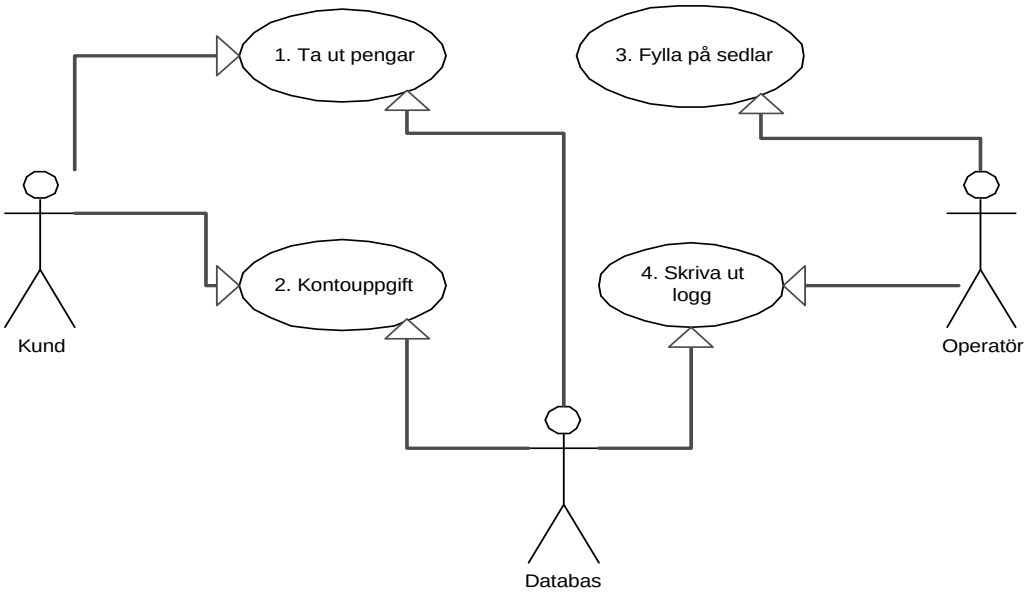
3a. Systemet visar innevarande leveransinformation, prisuppgifter och de fyra sista siffrorna från kreditkortet

3b. Kunden kan acceptera eller upphäva bristerna

Gå tillbaka till utgångsscenariot vid steg sex

Ett annat sätt att beskriva Use Case på är i ett diagram. I diagrammet beskrivs aktörer, Use Case och relationerna mellan aktör och Use Case (Fowler och Scott, 2000). Ett Use Casediagram för ett system innehåller alla aktörer och alla varianter av Use Case och avser att beskriva det totala funktionella beteendet av systemet (Leffingwell och Widrig, 2000). Use Casediagrammet åskådliggör även relationer mellan olika Use Case vilket kan bidra till ökad förståelse av systemet. I figur 3 visas ett Use Casediagram där streckgubbarna är aktörer och de ovala ringarna representerar Use Case. Diagrammet beskriver en bankomat. De olika aktörerna är kund, databas (ett externt system) och en operatör. När kunden kommer till automaten ska denna kunna ta ut pengar och göra ett kontoutdrag (Karlsson, 1996). Databasen ska kunna skicka iväg ett kontoutdrag, leverera

pengar och skriva logganteckningar vilket operatören vill ha information om. Operatören vill förutom att tillgodoses med loggarteckningarna kunna fylla på bankomaten med pengar. Dessa olika Use Case formar de krav som ställs på systemet. Use Casediagrammets syfte är att ta fram och dokumentera vad de olika intressenterna vill att systemet ska kunna göra. Mellan aktörer och Use Case finns olika slags relationer[4]. Enligt Fowler och Scott (2000) används aktörer i ett Use Casediagram för att identifiera systemets olika Use Case. Vid utveckling av ett stort system kan det ibland vara svårt att direkt lista upp systemets olika Use Case. Det är då lättare att först identifiera aktörerna och därifrån ta fram Use Case. Det viktiga är inte de exakta relationerna mellan aktör och Use Case, huvudsyftet är att identifiera alla relevanta Use Case för systemet (Fowler och Scott, 2000).



Figur 3 Use case diagram (Från Karlsson, 1996, sidan 32)

Ett Use Case behöver inte vara detaljerat beskrivet. Viktigt är dock att det är tillräckligt specifikt så att användarna förstår grundidén. Hur detaljerat ett Use Case ska vara beror också på hur stor riskfaktor det har (Fowler och Scott, 2000). Inom mjukvaruutveckling görs riskanalyser där de möjliga problem, som kan uppstå vid implementation av de framtagna kraven, identifieras (Sommerville och Sawyer, 1997). Riskanalysen klargör också hur stor risk det är att problemet uppstår och vilka effekter som det skulle föra med sig. I en iterativ process, beskrivs till en början ett Use Case enkelt med ett få antal sekventiella händelser. Där det behövs ytterligare beskrivningar av ett Use Case fylls det på och eventuellt tas bort händelser under följande iterationer (Fowler och Scott, 2000; Rosenberg och Scott, 1999; Leffingwell och Widrig, 2000). Ändringar och tillägg av händelser kan bero på att krav förändras och förståelsen för systemet växer allt eftersom arbetet fortgår (Leffingwell och Widrig, 2000).

2.4.2 Use Case Kritik

Det finns nästan alltid för- och nackdelar med olika metoder, tekniker och vektyg. Nedan presenteras både positiv och negativ kritik från olika författare om användning av Use Case.

[4] Relationerna kommer inte att redovisas i detta arbete. För ytterligare information om Use Casediagram och dess relationer hänvisas läsaren till *UML Distilled Second Edition* skriven av Fowler, M och Scott, K (2000).

När det är bra att använda Use Case

Enligt Regnell (1999) är det en fördel att använda Use Case för att hantera komplexitet. Det vill säga att det genom att använda Use Case är möjligt att fokusera på en del av systemet i taget. Med hjälp av Use Casemodellen kan utvecklarna handskas med komplexiteten genom att beskriva olika detaljerade nivåer av systemet (Regnell, 1999). Regnell (1999) anser att med hjälp av Use Case kan kunder och slutanvändare vara aktiva och medverka i processen med att beskriva och dokumentera krav. Detta bidrar även till att systemutvecklarna kan lära om användarna, deras behov och deras typiska beteenden. Use Case erbjuder en möjlighet att beskriva krav på så sätt att både kunden och utvecklaren har lätt för att lära och förstå. Use Case ger stöd för spårbarhet (se kapitel 2.2.2) och dessutom kan testfall baseras på Use Case.

När är Use Case inte det bästa valet?

Fowler och Scott (2000) påstår att det idag inte finns någon situation som Use Case inte kan användas. De menar att Use Case är en väsentlig teknik i kravinsamling och i planering och kontrollering av iterativa projekt. Leffingwell och Widrig (2000) menar däremot att det finns vissa typer av system och vissa sorters krav där Use Case inte är den mest lämpade tekniken att använda. System med endast ett fåtal eller inga användare och med ett minimalt gränssnitt, är exempel på system där Use Case inte ses som den effektivaste tekniken. Då Use Case har som huvuduppgift att beskriva krav ur användarnas synvinkel är det en enkel förklaring till varför Use Case inte bör användas där det inte är så många användare involverade. Det kan vara viruskontrollsystem som körs utan några operatoriska interaktioner eller system som utför vetenskapliga uträkningar eller simuleringar (Leffingwell och Widrig, 2000).

Use Case är heller inte en bra teknik att tillämpa då system domineras av ickefunktionella krav, till exempel användbarhet, tillförlitlighet och prestanda (Leffingwell och Widrig, 2000; Regnell, 1999). Det finns tillfällen då de ickefunktionella kraven kan beskrivas med hjälp av Use Case, exempelvis då det ickefunktionella kravet ska appliceras på ett eller ett fåtal Use Case och inte generellt för hela systemet (Leffingwell och Widrig, 2000). Use Case kan medföra redundans genom de olika sätten att uttrycka ett budskap, vilket kan öka kravdokumentationens storlek. Redundansen kan uppstå då många Use Case är snarlika, men ändå kräver en egen beskrivning. Problemet med att använda Use Case uppstår då ett beteende, som förekommer i många Use Case, ändras. Ett ändrat beteende påverkar flera Use Case och kräver ändringar i samtliga Use Case för att nå korrekthet (Leffingwell och Widrig, 2000).

Regnell (1999) har identifierat nackdelar med Use Case där han bland annat anser att det är svårt att tolka aktör- och Use Casekoncepten. Det vill säga att det inte finns någon klar definition av semantiken hos Use Case och att det inte finns några riktlinjer för hur Use Case skall beskrivas. Exempelvis spelar en aktör en roll, och en användare kan vara flera olika sorters aktörer. Detta kan enligt Regnell (1999) lämna utrymme för fri tolkning av aktör- och Use Casekoncepten och orsaka förvirring vid mjukvaruutvecklingen. Ett annat problem med Use Case är att det är svårt att veta hur omfattande ett Use Case ska vara och med vilken detaljnivå som ska användas vid kravrepresentationen.

3 Problembeskrivning

Det här kapitlet kommer först att ge en kort beskrivning av problemområdet och därefter presenteras den frågeställning som examensarbetet kommer att behandla.

3.1 Problemområde

Requirements Engineering är den process inom systemutvecklingen som utvinner, specificerar och validerar krav. Loucopoulos och Karakostas (1995) menar att det kostar mycket pengar då systemet inte tillfredsställer de behov och förväntningar som intressenterna har på systemet. Att komma till rätta med problemen på ett tidigt stadium i systemutvecklingsprocessen skulle medföra både tids- och kostnadsbesparingar (Leffingwell och Widrig, 2000).

En viktig del i RE-processen är kravrepresentationen. Med att representera krav avses i denna rapport att beskriva och dokumentera de framtagna kraven. I de flesta systemutvecklingsprojekt finns det flera systemutvecklare och ett antal olika intressenter som alla har olika behov och förväntningar på systemet (Leffingwell och Widrig, 2000). Det gäller att nå en överenskommelse mellan alla parter. De dokumenterade kraven fungerar som ett hjälpmedel vid kommunikation mellan användare och systemutvecklaren och mellan systemutvecklare som tagit fram krav och de utvecklare som implementerar systemet (Leffingwell och Widrig, 2000). Dokumentationen används som stöd för att förstå problemen och validera kraven (Pohl, 1996). Det är viktigt att välja rätt metod och teknik för att nå ett lyckat slutresultat och som tidigare nämnts sparas både tid och pengar genom att upptäcka brister på systemet i tidigt stadium i systemutvecklingsprocessen. Use Case är en teknik som används för att representera användarnas krav (Leffingwell och Widrig, 2000). Med hjälp av Use Case uttrycks de framtagna kraven på ett sådant sätt att det ökar intressenterna förståelse för de framtagna kraven (Cockburn, 2001). Beslut om vilken teknik som ska väljas vid representation av olika sorters krav underlättas om det framgår av tekniken vilka för- och nackdelar den har. Därför finns det intresse av att undersöka vilka för- och nackdelar det finns med att använda Use Case som teknik vid kravrepresentation.

3.2 Problemprecisering

Att representera krav är en viktig del av RE-processen. Det finns olika tekniker för att representera krav varav Use Case är en teknik som börjar etablera sig alltmer inom mjukvaruutveckling. För att slutprodukten ska nå ett lyckat resultat är det viktigt att de olika teknikerna används rätt och att systemutvecklarna vet när en teknik är lämplig att användas.

Frågeställningen för detta examensarbete är följande:

Vilka för- och nackdelar finns det med att använda Use Case för att representera krav och anpassas kravrepresentationen efter dessa?

Avsikten är att ta reda på vad de som arbetar/har arbetat med Use Case upplever som för- och nackdelar med att använda Use Case som teknik vid kravrepresentation. Representera krav avses i detta examensarbete att beskriva och dokumentera krav. Tidigare har nämnts hur viktig kravrepresentationen är för att slutprodukten ska tillfredsställa de olika intressenternas behov och förväntningar på systemet (se kapitel 2.3.2). Det är viktigt att för- och nackdelar identifieras för en teknik så att inte en otillräcklig teknik bidrar till ett mindre lyckat resultat. Med fördelar och nackdelar avses bland annat de olika tillfällen

och de olika sorters krav som fungerar bra respektive fungerar dåligt att representeras med hjälp av Use Case. Undersökningen ska också ge en bild av huruvida de som använder Use Case tar hänsyn till vad de själva har uppfattat som för- och nackdelar med att använda Use Case, genom att komplettera tekniken för att bättre passa deras behov.

3.3 Avgränsning

Den information som framkommer av undersökningen i detta examensarbete kommer att visa på för- och nackdelar med att använda Use Case. Examensarbetet kommer inte att svara på huruvida det går att använda Use Case eller ej, i alla situationer.

Det här examensarbetet kommer endast att behandla kravrepresentationen inom RE-processen, det vill säga hur krav beskrivs och dokumenteras. Att utvinna krav och att validera krav kommer inte att beröras i detta arbete.

Undersökningen kommer att avgränsas till att undersöka företag som använder UML som modelleringsspråk, inte någon speciell version av UML, där Use Case är en teknik för att bland annat representera krav.

3.4 Förväntat resultat

Förväntningarna på undersökningen till examensarbetet är att ta fram och beskriva de för- och nackdelar som systemutvecklare upplever med att använda Use Case för att representera krav. Utifrån de svar som problemställningen ger ska det även framgå om det behövs kompletterande tekniker för kravrepresentation.

4 Metod

Med *metod* avses enligt Ejvegård (1996) ett vetenskapligt tillvägagångssätt, ett sätt att lösa problem och komma fram till ny kunskap om ett speciellt ämne. Exempelvis går det att göra en enkel beskrivning, binda sig vid att göra vissa jämförelser, att formulera hypoteser eller att göra förutsägelser. En *teknik* är det sätt som materialet samlas in på för att kunna beskriva, jämföra, sätta upp hypoteser eller förutsäga något.

I detta kapitel presenteras intervju, enkät, observation och litteraturstudier som är några exempel på olika tekniker som skulle vara relevanta för att besvara problemställningen. Därefter motiveras de val som gjorts och hur det bidrar till att besvara problemställningen. Slutligen redovisas hur det är tänkt att genomföra undersökningen.

4.1 Litteraturstudie

Med litteratur avses i forskningssammanhang nästan allt som är tryckt material såsom böcker, artiklar, rapporter, uppsatser och essäer (Ejvegård, 1996). En *litteraturstudie* kan besvara frågor kring faktiska förhållanden och faktiska skeenden (Patel och Davidson, 1994). Litteraturstudien kan också användas för att besvara frågor som rör en individs upplevelser av något förhållande eller skeende. Om en litteraturstudie skulle användas för att få fram de för- och nackdelar som upplevs med Use Case för att representera krav skulle den baseras på olika författares arbeten inom ämnet. Detta för att skildra olika författares uppfattning om samma ämne eller område. Litteraturen ska ge en så fullständig bild som möjligt, det vill säga att informationen blir belyst ur flera synvinklar. De olika författarnas åsikter skulle jämföras och dessutom kritiskt granskas annars skulle rapporten endast spegla en begränsad del av verkligheten. Enligt Ejvegård (1996) är viktigt att ställa sig kritisk till olika författares information om samma fakta och dessutom inte endast välja fakta som bekräftar ens egen uppfattning. För att få fram för- och nackdelar med användning av Use Case vid kravrepresentation skulle det innebära att hitta litteratur där även författarna kritiskt granskar tekniken. Att bedriva en litteraturstudie är enligt Patel och Davidson (1994) en relativt tidskrävande process.

4.2 Intervju

Att samla in information kan göras med hjälp av *intervjuer*, det vill säga att informationen samlas in genom frågor. För att besvara tidigare definierad problemställning med hjälp av intervjuer finns det olika sätt intervjun skulle kunna utföras på. Alternativ som skulle vara intressanta för den här undersökningen är personlig intervju eller telefonintervju. Det går att styra intervjuerna genom att lämna stort eller litet utrymme för respondenten att svara, vilket kallas grad av *strukturering*, och hur frågorna utformas och deras inbördes ordning, vilket kallas grad av *standardisering* (Patel och Davidson, 1994). En helt strukturerad intervju lämnar litet utrymme för respondenten att besvara frågorna, det är även möjligt att förutsäga vilka svarsalternativ som finns (Patel och Davidson, 1994). I en ostrukturerad intervju lämnas helt fritt utrymme för respondenten att svara på frågorna. Intervjufrågorna för att besvara detta arbetes problemställning skulle vara utformade så att respondenten gavs möjligheten att svara fritt på frågorna, det vill säga vara ostrukturerade. Intervjufrågorna skulle dessutom ha en låg grad av standardisering, vilket innebär att ordningen på frågorna inte strikt behöver följas och alla frågorna är inte i förväg framarbetade. Avsikten med att låta intervjuerna ha låg grad av standardisering är att då det eventuellt skulle dyka upp oförutsedda frågor ska det vara möjligt att ställa dem till respondenten. Detta till skillnad mot en standardiserad intervju där respondenten

enligt Krag (1993) under en standardiserad intervju i stort sett endast får välja mellan förutbestämda svarsalternativ.

En fördel med att använda standardiserad intervju skulle vara att det skulle gå snabbt att intervjua ett stort antal personer och det kräver inte att intervjuaren har så mycket erfarenhet av att intervjua (Krag, 1993). Men den standardiserade intervjun har också begränsningar, bland annat är intervjun inte flexibel och det går inte att fånga upp det oförutsedda.

4.3 Enkät

Enkäter liksom intervjuer används för att samla in information med hjälp av frågor. Enkäter kan vara strukturerade eller ostrukturerade, standardiserade eller ostandardiserade (se kapitel 4.2) på samma sätt som intervjuer (Patel och Davidson, 1994). Det finns olika sätt att hantera enkätundersökningar på, till exempel kan enkäter skickas till respondenten per post och besvaras på samma sätt. En enkät kan även besvaras under ledning, det vill säga att enkäten fylls i av respondenten samtidigt som intervjuaren finns tillgänglig för att förtydliga och förklara frågor. Någon av ovan nämnda alternativ av enkätundersökning skulle kunna användas för att ta reda på vad systemutvecklare anser vara för- och nackdelar med att använda Use Case för att representera krav. Vidare menar Patel och Davidson (1994) att en viktig aspekt vid utformningen av enkäten skulle vara att arbeta systematiskt och konstruera frågor kring alla delområden så att problemet täcks upp ordentligt. Har inte problemet täckts upp av frågorna på enkäten finns det inte någon möjlighet att komplettera enkäten till skillnad mot ostandardiserade intervjuer där det är möjligt att ställa följdfrågor. Med en enkätundersökning skulle det vara möjligt att fånga en större mängd undersökningspersoner än vid intervjuer. Men för att få personerna att medverka i undersökningen skulle det då vara viktigt att motivera de individer som är intressanta för undersökningen till att vilja medverka i undersökningen. Detta skulle ske bland annat genom att klargöra syftet med undersökningen och försöka relatera det till individens egna mål. Detta skulle vara särskilt viktigt vid den första kontakten med individen. Sker exempelvis första kontakten via ett brev med bifogad enkät ska brevet säga så mycket att den som läser det fattar det intresse som krävs för att fylla i enkäten (Krag, 1993).

4.4 Observation

Observationer kan ske antingen slumpmässigt eller som en vetenskaplig teknik för att samla in information (Patel och Davidson, 1994). Slumpmässiga observationer sker dagligen för att införskaffa information om omgivningen. Vetenskaplig observation måste däremot vara systematisk och planerad, dessutom ska även registrering av information ske systematiskt.

Att undersöka examensarbetets problemställning med hjälp av observation skulle innebära att observatören skulle finnas med under ett projekt och under tiden studera vad systemutvecklarna uppfattar som för- och nackdelar med att använda Use Case för att representera krav. Dessutom skulle observatören kunna identifiera vid vilka olika tillfällen som systemutvecklarna dessutom väljer att komplettera Use Casetekniken med annan teknik. För att få en representativ undersökning skulle flera olika projekt behöva observeras. En nackdel med observationer är att de är dyra och tidskrävande (Patel och Davidson, 1994). En observation skulle också kunna användas för att se huruvida Use Case lämpar sig som teknik för att representera krav för en specifik typ av projekt.

Därigenom kan observatören få fram de för- och nackdelar som upplevs med tekniken och hur utvecklarna anpassar utvecklingsarbetet utefter dessa.

4.5 Val av teknik för materialinsamling

Litteraturstudie är en tänkbar teknik för att besvara examensarbetets problemställning då det torde finnas utvärderingar och rapporter från tidigare gjorda projekt. Men då tanken är att undersöka i olika organisationer hur de som arbetar eller har arbetat med Use Case förhåller sig till problemställningen, har denna teknik valts bort. Det skulle dock vara tänkbart, att kombinera olika tekniker som till exempel litteraturstudie och intervjuer. Men då de olika teknikerna kräver mycket tid var för sig, väljs inte detta alternativ.

En enkätundersökning är en teknik som kan användas till att ge svar på examensarbetets problemställning. I kapitel 4.3 nämns olika sorters enkätundersökningar, bland annat *enkät under ledning*. Detta är ett tänkbart alternativ, men det är ändå att föredra i denna undersökning att ha möjlighet att förtydliga och diskutera frågorna och framförallt att kunna ställa oförutsedda följdfrågor. Som tidigare nämnts är det viktigt att kunna utforma enkäten och bifogade brev så att de utvalda personerna fattar intresse för undersökningen (se kapitel 4.3). På grund av oerfarenhet av att använda och formulera enkäter är även detta en orsak till att enkät inte används i denna undersökning.

Observation är också en möjlig teknik för att besvara problemställningen, dock är tidsbristen den största orsaken till att det inte är genomförbart i detta examensarbete. En systemutvecklingsprocess kan vara en lång process som varar i flera månader och år. För att få ett representativt svar på frågeställningen genom observation, krävs observationer från många olika projekt vilket inte är möjligt med tanke på tidsbegränsningen för detta arbete.

Intervju är den teknik som valts för att besvara examensarbetes problemställning. Valet av teknik stod mellan intervju och enkät. Både vad det gäller enkät och intervju är det viktigt att frågorna är rätt utformade så att de inte kan misstolkas. Den främsta orsaken till att valet av teknik blev intervju, är möjligheten att kunna ställa följdfrågor till respondenten, vilket inte enkäter tillåter. En ytterligare fördel med intervju är att intervjuaren ges möjlighet att be respondenten utveckla de svar som intervjuaren inte förstått, trots allt är respondenten experten inom ämnet och kan röra sig med uttryck som inte förstås av intervjuaren.

4.6 Planering av genomförandet

Undersökningens syfte är att ta reda på de för- och nackdelar som upplevs med Use Case för att representera krav och se hur systemutvecklarna anpassar kravrepresentationen utefter dessa. Intervju är den teknik som har valts för undersökningen och det krav som ställs på respondenten är att denna någon gång har använt Use Case för att representera krav. Nedan presenteras de olika steg av planering som ligger till grund för intervjuerna.

4.6.1 Tillvägagångssätt för kontakt med respondenter

För att hitta lämpliga respondenter till undersökningen kommer Internet att användas som sökverktyg. Internet har valts därför att det förmodligen kommer att erbjuda många möjliga sökalternativ och många möjliga företag att ta kontakt med. Sökningen kommer att koncentreras på systemutvecklingsföretag. Då sökningen har resulterat i ett antal systemutvecklingsföretag, kommer påbörjandet av en första kontakt att tas med olika

företag. Urvalet av företag från de framtagna listorna kommer att ske genom en systematisk avbetning av företag uppifrån och ned på listorna. Om det uppstår en situation där en avgränsning behöver göras på grund av för många sökträffar, vilket inte är så ovanligt då sökning sker på Internet, kan avgränsningen ske geografiskt för att det eventuellt kan resultera i en personlig intervju (se kapitel 4.6.2). Den listan skulle därefter också behandlas på samma sätt som beskrivits ovan. För att hitta intervjuobjekt kommer kontakt med de olika systemutvecklingsföretagen att ske både via elektronisk post och via telefon. I första hand kommer den första kontakten med företag att ske via telefon. En fördel med telefonkontakt är möjligheten att undersöka huruvida företaget använder eller har använt sig av Use Case för att representera krav. Om företaget arbetar med Use Case finns även möjligheten att via telefon få information om vilken person som är lämplig för intervju. Att ta kontakt med företag är även möjligt via e-post men det är större risk att kontaktpersonen inte svarar på meddelandet. Elektronisk post kommer att fungera som kommunikationsmedel när det av olika anledningar är svårt att få tag på personer via telefon.

4.6.2 Intervjuer och intervjufrågor

På grund av den begränsade tid som ställs till förfogande för examensarbetet kommer antalet intervjuer att begränsas till mellan fem och tio intervjuer. En större mängd intervjuer skulle vara att föredra för att bättre kunna analysera och värdera undersökningens resultat. Undersökningen kommer att hantera både personliga intervjuer och telefonintervjuer beroende på att undersökningen inte har koncentrerats till ett speciellt geografiskt område. Avstånden till de olika företagen kan komma att variera vilket gör det omöjligt att enbart genomföra personliga intervjuer. Där det är möjligt att genomföra personliga intervjuer är detta att föredra då det ger möjlighet till större insikt i företagets arbete med Use Case, exempelvis genom en demonstration av projektets kravdokumentation. Vid personlig intervju kommer en bandspelare att användas för att dokumentera informationen från intervjun. Vid telefonintervju är det ännu osäkert om intervjun kommer att spelas in på band eller om det enbart blir anteckningar. På grund av tidigare erfarenhet av misslyckade bandinspelningar förs vid bandinspelningen anteckningar som extra stöd. En fördel med att använda bandspelare är att hela intervjun återfinns på bandet och ingen information går förlorad, givet en lyckad bandinspelning (Krag, 1993). Att enbart anteckna gör det svårt att fånga upp all information från intervjuerna. En nackdel med att spela in intervjuer på band kan vara att den som intervjuar slappnar av och inte är lika uppmärksam på bland annat ord och begrepp den inte förstår. Ytterligare en nackdel är den tid det tar att lyssna igenom bandet och skriva ned innehållet. Respondenten kan påverkas negativt både av bandinspelning och av att intervjuaren för anteckningar under intervjun. Vidare menar Krag (1993) bland annat att respondentens koncentration kan riktas mot bandspelaren istället för på intervjufrågorna. Respondentens koncentration kan också riktas mot intervjuarens nedtecknande av information. Valet att använda bandspelare baseras på att det är svårt att både vara koncentrerad på det respondenten säger och samtidigt hinna anteckna allt som sägs under intervjun. Det kan vara särskilt svårt att hinna anteckna svaren då ostrukturerade frågor används i intervjun.

Vissa frågor kommer att vara framarbetade i förväg, men det ska vara möjligt att kunna ställa ytterligare frågor som anses intressanta och relevanta under intervjun. Det vill säga att det ska vara möjligt att fånga upp det oförutsedda (se kapitel 4.2). Dessutom kommer ordningen på frågorna inte att spela någon roll för resultatet av intervjun, intervjuerna kommer därmed att genomföras med låg grad av standardisering. Intervjuerna kommer för övrigt att vara ostrukturerade vilket innebär att frågorna kommer att ge respondenterna stort svarsutrymme (se kapitel 4.2). Att låta respondenten få berätta fritt om de olika

frågorna är lämpligt i detta fall, då det inte finns några förutbestämda svarsalternativ och då de frågor som är framarbetade i förväg är öppna frågor. Fråga 11 och 12 (se bilaga 1) är ja- eller nejfrågor som därefter följs av en följdfråga som lämnar större svarsutrymme åt respondenten.

De första tre frågorna (se bilaga 1) är tänkta att skapa en bild av respondentens erfarenhet inom företaget och inom systemutveckling. Informationen om respondenten kan vara intressant då det är dags att utvärdera de svar som framkommit från olika respondenter. Svaren kan skilja sig beroende på om respondenten har arbetat i ett eller tio år, ju fler genomförda projekt desto mer erfarenhet. Dessutom är det viktigt att inte ställa huvudfrågorna direkt, utan först låta respondenten känna sig väl till mods i situationen (Krag, 1993). Det är även viktigt att låta respondenten känna att intervjuaren har ett intresse för respondentens kunskap och erfarenhet (Krag, 1993). Det ökar respondentens motivation och vilja att besvara frågorna (Patel och Davidson, 1994). Därefter följer intervjuns huvudfrågor, fråga 4-13 (se bilaga 1), varav fråga 4 har till syfte att klargöra hur företaget eller projektet valt att representera krav, exempelvis i textform eller i Use Caediagram (se kapitel 2.4.3). Svaret på fråga 4 avser att ge en bild utav hur de olika respondenterna har valt att representera kraven med hjälp av Use Case, i textform eller diagram. Fråga 5 – 10 (se bilaga 1) ska besvara första delen av problempreciseringen, nämligen vilka för- och nackdelar respondenten upplever att Use Casetekniken har vid kravrepresentation. Frågorna 5 - 10 liknar varandra, men fokuserar på olika aspekter av krav. Detta för att få ut så mycket information som möjligt av respondenten. Fråga 11 - 13 (se bilaga 1) ger svar på sista delen av problempreciseringen, huruvida systemutvecklarna anpassar kravrepresentationen till de för- och nackdelar som upplevs med Use Casetekniken. Fråga 13 är en viktig aspekt vad det gäller att nå en effektiv hantering av krav för att undvika den extra tid och kostnad som uppkommer vid eventuella misslyckanden i RE-processen (se kapitel 2.3).

5 Genomförande

I denna del av rapporten presenteras de intervjuer som ligger till grund för att besvara examensarbetets tidigare definierade problemställning. Dessutom beskrivs genomförandet av intervjuerna. Slutligen värderas materialet, och de erfarenheter som undersökningen givit redovisas.

5.1 Intervjuer

Under denna undersökning har nio intervjuer genomförts med systemutvecklare från olika projekt eller olika företag. Intervjuerna har genomförts både som personliga intervjuer och telefonintervjuer, beroende på de olika avstånden till de olika respondenterna.

5.1.1 Att kontakta respondenter

Beslutet att söka systemutvecklare via Internet (se kapitel 4.6) grundade sig i att där borde det finnas gott om företag som arbetar med systemutveckling. Först söktes kontaktpersoner från två välkända IT-företag som båda är verksamma inom systemutveckling. Ett av företagen förekommer på flera ställen runt om i Sverige. När sökning skedde på företagsnamnet resulterade det i telefonnummer och e-postadresser till kontaktpersoner för de olika företagen från norr till söder i Sverige. Sedan användes sökordet *systemutveckling* vilket resulterade i flera hundra träffar. För att begränsa antalet träffar valdes företag som var belägna i Västra Götalands Län, med tanke på att det skulle vara möjligt att utföra personliga intervjuer beroende på avståndet. Även detta resulterade i en lista med telefonnummer och e-postadresser.

Nästa steg var att välja ut företag som förekom på de framtagna listorna och kontakta dem för att kontrollera om de använder eller har använt UML:s Use Case som teknik vid kravrepresentation och dessutom undersöka deras intresse av att bli intervjuade. Ett av IT-företagen är beläget i Skövde och via en personlig kontakt erhölls en e-postadress till en person som arbetar med Use Case. Detta företag valdes ut på grund av att närheten till respondenten skulle göra det möjligt att utföra personliga intervjuer. Utifrån de andra två listorna med olika företag valdes att ringa företagen uppifrån och ned på de olika listorna. Då ett företag inte var tillgängligt via telefon valdes nästa systemutvecklingsföretag på listan och vid ett senare tillfälle kontaktades vissa av företagen om igen för ett ytterligare försök få kontakt med dem. Förutom företaget beläget i Skövde skedde den första kontakten med företagen via telefon. Den första kontakten med Skövdeföretaget var ett e-postmeddelande. Tanken var att om svar inte skulle erhållas från e-postmeddelandet skulle nästa kontaktförsök ske via telefon. Skövdeföretaget lämnade ett positivt svar och till en början var det fyra personer som kunde medverka i undersökningen. Den fortsatta kontakten men intervjupersonerna från Skövdeföretaget, för bland annat bokning av tid för intervjuer, skedde via e-post. I övrigt var det inte svårt att finna företag som använder Use Case (UML:s Use Case), däremot var det svårt att nå fram till de specifika personer som arbetat med Use Case för att representera krav. Flertalet av de personer som fungerar som kontaktpersoner för företagen, lovade att höra av sig via e-post eller via telefon för ett positivt eller negativt svar angående intressanta intervjupersoner. Till en början var det trögt att få några direkta svar och 10 – 15 personer som lovat höra av sig via telefon eller via e-post hörde inte av sig. Då ett nytt försök gjordes för kontakt med personerna meddelade de exempelvis att de inte hittat någon lämplig person att intervjua eller så var de på tjänsteärenden resten av veckan. Arbetet med att söka intervjuobjekt fortsatte, vilket ledde till många samtal under en veckas tid. Målet var att finna mellan fem till tio intervjuobjekt och när åtta personer givit ett positivt svar att medverka i undersökningen kändes det tillräckligt för att få svar på problemställningen. När de åtta intervjuerna var

genomförda e-postade ytterligare en person från skövdeföretaget som meddelade att denne kunde medverka i undersökningen. Undersökningen utökades då till nio intervjupersoner. Vid den första kontakten, både via telefon och via e-post, där bokning av intervju skedde klargjordes syftet med undersökningen och respondenten förbereddes på vad intervjun skulle beröra för ämne. Det vill säga vilka för- och nackdelar som respondenten upplever med Use Case för att representera krav.

5.1.2 Respondenterna

Nedan presenteras de respondenter som medverkat i undersökningen. Det ges en kort beskrivning av respondenternas erfarenhet inom systemutveckling och Use Case, och vilken typ av företag de arbetar på. Samtliga respondenter har i någon form varit i kontakt med Use Case för att representera krav. Respondenterna presenteras inte i den ordning de intervjuades.

Respondent 1 – 5 arbetar på ett IT-företag i Skövde där respondent 1 och 3 har arbetat i samma projekt. Respondent 1 har arbetat med systemutveckling i fem år varav ett år med Use Case. Respondent 2 har arbetat med systemutveckling i elva år och under de senaste två åren med Use Case för att representera krav. Respondent 3 har arbetat med Use Case i två projekt under det senaste 1,5 året och med systemutveckling i tre år. Respondent 4 har arbetat som systemutvecklare i 7 år, varav 2,5 år i projekt som använt Use Case för att representera krav. Respondent 5 arbetar som metodutvecklare och har varit med i ett pilotprojekt där denna skulle utvärdera huruvida Rational Unified Process (RUP)[5] och då även Use Casetekniken lämpade sig för företagets produktion. Mellan 1989 och 1995 arbetade respondent 5 med systemutveckling och sedan 1995 har respondenten arbetat med metodutveckling.

Respondent 7 och 9 arbetar på ett konsultföretag i IT-branchen i Stockholm i helt skilda projekt. Respondent 7 har arbetat inom systemutveckling i nio år och har under de senaste fem åren arbetat till och från i olika projekt med Use Case för att representera krav. Respondent 9 har arbetat med systemutveckling sedan 1973 och har under de senaste två åren arbetat i projekt som använt Use Case vid kravrepresentation.

Respondent 6 arbetar i ett IT-företag i Linköping som är verksamma inom programutveckling. Respondenten har arbetat inom systemutveckling i fem år och har under alla de åren till och från arbetat i projekt som använt Use Case vid kravrepresentation.

Respondent 8 arbetar som systemutvecklare på ett IT-företag i Umeå. Av de nio år som respondenten arbetat inom systemutveckling har det under de senaste fem åren förekommit projekt där Use Case har använts för att representera krav.

5.1.3 Genomförande av intervjuer

Intervjuerna genomfördes både som personliga intervjuer och telefonintervjuer. Som tidigare nämnts hade personliga intervjuer varit att föredra (se kapitel 4.6.2). Men på grund av den spridning som det blev geografiskt mellan de olika respondenterna och på grund av otillräckliga resurser kombinerades de olika intervjuteknikerna. Dessutom ansåg respondenterna från Stockholm och Linköping att telefonintervju passade dem bäst.

[5] Rational Unified Process är en systemutvecklingsprocess. För vidare information hänvisas läsaren till *The Rationa Unified Process, An Introduction*, Kruchten (2000).

Respondent 5 arbetar på Skövdeföretaget men på grund av att respondenten anmälde sitt intresse för att medverka i undersökningen väldigt sent, är tidsbristen en bidragande orsak till att denna intervju genomfördes via telefon trots närheten till respondenten. Fyra intervjuades personligen (responden 1 till 4) på respondenternas arbetsplats och fem intervjuades per telefon (respondent 5 till 9). Både de personliga intervjuerna och telefonintervjuerna tog mellan tjugo minuter och en halvtimma att genomföra. För att registrera informationen från de personliga intervjuerna användes en bandspelare. Även vid telefonintervjuerna användes en bandspelare för att registrera informationen. Vid första telefonintervjun (respondent 6) misslyckades dock inspelningen. Som komplement till alla intervjuerna fördes anteckningar vilket bidrar till att intervjun med respondent 6 kan användas som underlag till undersökningen. Till skillnad från de andra intervjuerna som kommer att sammanställas och tolkas utifrån bandinspelningen kommer intervjun med respondent 6 att sammanställas och tolkas utifrån anteckningarna från intervjun.

För varje respondent klargjordes syftet med undersökningen innan intervjun påbörjades. Respondenten fick godkänna att bandspelare användes under intervjun och dessutom klargjordes att materialet kommer att behandlas konfidentiellt. Samtliga respondenter godkände bandinspelningen.

Intervjuerna startades med de inledande frågorna där respondenten fick berätta om sina arbetsuppgifter och sin erfarenhet inom systemutveckling. Därefter följde huvudfrågorna och eventuella följdfrågor som ansågs relevanta. Under vissa intervjuer berättade respondenten så fritt att flera av de resterande huvudfrågorna besvarades utan att de behövdes ställas. De olika respondenterna använde ibland ord och begrepp som inte var bekanta för intervjuaren och då fick respondenten förklara dessa. De följdfrågor som har förekommit under undersökningen har inte använts under alla intervjuer och några följdfrågor kan ha baserats på något som intervjuaren funnit intressant under en tidigare intervju. Ordningen på frågorna har i de flesta intervjuer skett i den ordning som finns presenterat i bilaga 1, det har fallit sig naturligt att prata om kravrepresentationen i den ordningen.

Två av de respondenter som intervjuades personligen (respondent 2 och 4) demonstrerade även med bilder och medhavd dokumentation av Use Case. Dessa intervjuer tog något längre tid än de övriga intervjuerna.

5.1.4 Värdering av intervjumaterial

Undersökningen har bestått av nio intervjuer vilket är ett acceptabelt antal intervjuer för att få svar på problemställningen. Intervjuerna ger en representativ bild av vilka för- och nackdelar det finns med att använda Use Case för att representera krav och vid vilka tillfällen Use Case bör kompletteras med annan teknik. Materialet som framkommit under de olika intervjuerna är mestadels relevant för den tidigare definierade frågeställningen. Svaren har vid några tillfällen svävat ifrån frågan något vilket inte är konstigt då frågorna var ostrukturerade och därmed lämnade stort utrymme för respondenterna att svara.

5.1.5 Erfarenheter från intervjuprocessen

Vid den första kontakt som togs med respondenten då intervjun bokades blev respondenten underrättad om syftet med intervjun och att frågorna skulle beröra vilka för- och nackdelar respondenten upplever med att använda Use Case för att representera krav. Om respondenterna hade tagit åt sig den informationen är inte registrerat, men samtliga respondenterna har givit svar på huvudfrågorna. Att huvudfrågorna har besvarats har bidragit till ett material som kan användas för att besvara problemställningen. Däremot

finns det tillfällen då respondenten lämnat ett svar, som under intervjun uppfattades som svar på frågan, men i efterhand då bandet lyssnades av, upptäcktes att svaret var ofullständigt.

Genom den bandupptagning som gjordes under intervjuerna gavs en möjlighet att lyssna på sig själv som intervjuare. Det framgick tydligt från bandinspelningen att det krävs erfarenhet för att intervjuer ska bli riktigt lyckade. Det var svårt att formulera tydliga frågor, så att respondenterna inte missförstår eller misstolkar frågorna. Det skapade inte några problem just för tillfället under intervjun, däremot då materialet skulle analyseras saknades i vissa fall förklaringar till speciella uttryck som respondenten använt sig utav.

Upplägget för undersökningen med intervjuer som varit ostrukturerade och haft låg grad av standardisering har fungerat bra. Det har gett intervjuaren möjlighet att fånga upp det oförutsedda (se kapitel 4.2). Intervjuaren har kompletterat intervjuerna med frågor som gjort att respondenten fått förklara främmande begrepp, vilket är viktigt vid sammanställning och utvärdering av materialet.

Tidsmässigt har intervjuerna varit lagom långa, dels för intervjuaren att hålla koncentrationen uppe under intervjun och dels för det efterarbete som följt av bandinspelning av intervjuerna. Bandinspelningen har varit ett väldigt bra stöd vid sammanställning av materialet.

Intervjufrågorna 4 – 10 (se bilaga 1) skulle bidra till att besvara vilka för- och nackdelar som respondenterna upplever med Use Case för att representera krav. Fråga 9 och 10 (se bilaga 1) bidrog vid flera intervjutillfällen inte med någon ytterligare information. De var i de flesta fall överflödiga, men fanns med för att få en heltäckande bild av för- och nackdelarna med Use Case. Fråga 13 (se bilaga 1) behövde inte ställas till någon respondent eftersom de kompletterade kravrepresentationen där de ansåg att det fanns behov av det. En respondent nämnde att denna vid några tillfällen bedömt det lämpligt att använda kompletterande teknik för att representera krav, men inte gjort det. Detta berodde på att respondenten arbetat i stora projekt med spridning runt om i Sverige och utomlands och utvecklarna har varit tvungna att följa de beslut som finns angående metoder, tekniker och verktyg inom inom det systemutvecklingsprojektet.

Under intervjuerna med respondent 2 och 4 demonstrerades bland annat olika Use Casedokumentationer och beskrivning på olika nivåer av Use Case. Detta upplevdes positivt av intervjuaren. Det gav intervjuaren en möjlighet att bilda en uppfattning av hur respondenterna använder sig av Use Case när de representerade krav.

6 Materialpresentation

I det här kapitlet presenteras det material som framkom under den undersökning som tidigare presenterats i genomförandet (se kapitel 5). Då intervjuerna var ostrukturerade och hade låg grad av standardisering har det bidragit till att svaren i vissa fall är långa. För att återge materialet från intervjuerna på ett lättöverskådligt sätt, presenteras svaren på huvudfrågorna i detta kapitel. En utförligare presentation av intervjuerna återfinns i bilaga 2 – 10.

Frågorna 4 – 10 (se bilaga 1) var avsedda att ta fram de för- och nackdelar som respondenterna upplever med att använda Use Case för att representera krav. Svaren från dessa frågor sammanställs nedan under två rubriker, *fördelar med Use Case* och *nackdelar med Use Case*. Därefter presenteras fråga 11 – 13 (se bilaga 1) under *kompletterande tekniker*.

Frågorna 1 - 3 (se bilaga 1) har redovisats till viss del i genomförandet som en presentation av respondenternas erfarenhet inom systemutveckling. Fråga 4 presenteras i bilaga 2 – 10. Denna fråga ger en bild av hur respektive respondent valt att representera kraven med Use Case.

Vid intervjun med respondent 6 misslyckades bandinspelningen (se kapitel 5.1.3) och det material som presenteras för respondent 6 är till skillnad från de andra respondenterna baserad på anteckningar från intervjun.

6.1 Fördelar med Use Case

Respondent 1

Respondent 1 anser att det är lätt för användarna att förstå hur ett Use Casediagram fungerar och förstå vad ett Use Case och en aktör är. Det går också att använda Use Casediagrammet som ett diskussionsunderlag. En dialog skapas mellan systemutvecklare och användare om aktörer och Use Case. Exempelvis diskuteras vad användare och utvecklare vill ha ut av systemet, vad de vill att systemet ska göra och de kan prata ihop sig så att utvecklarna och användarna menar samma sak. Respondent 1 anser också att Use Case beskriver flödet på ett bra sätt. Projektet har dokumenterat olika Use Case i den ordning som de sedan ska utföras då systemet är implementerat. Med Use Case går det att få ut ett normalflöde, det vill säga det flöde då allt fungerar som det ska och där inga fel uppstår. Respondenten anser dessutom att Use Case följer som en röd tråd genom hela systemets livscykel (inte endast under projektet). Det är enligt respondenten tänkt att det ska gå att bygga ut systemet runt Use Casen och att följa upp testfall.

Respondent 2

Respondent 2 anser att den största fördelen är att Use Case är ett bättre sätt att möta användarna på. Användarna får möjlighet att delta i analysarbetet på ett bättre sätt i jämförelse med traditionella sätt att utveckla system. Det är användarnas krav som ska arbetas fram med hjälp av Use Case. Use Case är också ett itererande arbetssätt vilket är en fördel för att upptäcka kraven på ett tidigt stadium. Enligt respondenten är det även lättare för användare och andra intressenter att förstå kraven som representeras med hjälp av Use Case i jämförelse med traditionella systemutvecklingsmetoder. Vidare anser respondent 2 att det är bra att Use Case ger möjligheten att beskriva krav på olika nivåer för olika intressenter. Exempelvis är inte chefen intresserad av samma information som

programmeraren är. Slutligen tycker respondenten att Use Case lämpar sig bättre på administrativa system, där det finns en riktig användare (då aktören är en människa).

Respondent 3

Respondent 3 anser att Use Casediagrammet hjälper till att ge en helhetsbild direkt. Det går fort att få fram en helhetsbild av det tänkta systemet som längre fram under utvecklingsarbetet blir mer detaljerad. Use Case betyder på svenska användningsfall och är tänkt att beskriva användarnas krav på systemet och tillåter att användarna är mycket inblandade i utvecklingsprocessen. I och med att det är användarkraven som ska beskrivas med Use Case lämpar det sig bra att Use Case representerar krav där användare är inblandade. Det vill säga när det finns många olika aktörer som interagerar med systemet.

Respondent 4

Respondent 4 anser att den största fördelen med användning av Use Case är att det är ett bra sätt att möta kunden på. Framförallt då Use Casediagrammet används tillsammans med användare och kunden. Respondenten menar att kunden får en större inblick i vad utvecklarna kan få ut ur ett system. Use Casediagrammet fungerar som kommunikationsmedel mellan användare och utvecklare. Det är ett bättre sätt att kommunicera på i jämförelse med traditionella tekniker. Use Case fångar behoven/kraven ur en användares synvinkel. Det är ett väldigt bra sätt att strukturera upp informationen på. Systemutvecklarna kan tillsammans med användarna rita och prata kring hur saker (olika krav) uppfattas av både utvecklare och användare. Dessutom anser respondenten att Use Casediagrammet är lätt för användarna att förstå. Användarna får vara med i kravhanteringen och kan vara med och påverka. Dessutom ökar kvaliteten på resultatet menar respondenten, vilket är ett resultat av att användarna har fått vara med att dokumentera och göra modeller. Use Case är också en bra dokumentationsteknik enligt respondenten. Vid stora projekt delas arbetet upp, det produceras Use Casediagram på olika ställen och av olika personer i ett projekt. Tack vare att Use Casedokumentationen är så lätt att förstå kan olika personer i olika delar av projektet läsa varandras Use Case och förstå vad respektive del av projektet håller på med och vilka resultat som framkommer. Respondent 4 menar vidare att det är ett bra sätt att verifiera användarfunktioner, att kontrollera att alla funktioner finns med. Respondent 4 anser att Use Case också fungerar väldigt bra vid spårbarhet och testning. Det vill säga om användarna inte är nöjda med någon del av systemet är det möjligt att gå tillbaka till Use Casedokumentationen och se vad som verkligen beslutades i ett tidigare skede. Att kunna gå tillbaka till redan specificerade krav kallas också spårbarhet. Det finns bra verktygsstöd i UML för att gå tillbaka och verifiera vad ett system är tänkt att göra, det fungerar därför också bra vid testning. Vid testning är det möjligt att gå tillbaka till ett Use Case och kontrollera vad det var tänkt att systemet skulle göra, stämmer det med verkligheten eller inte. Ett Use Case fungerar som en verifikation på det som en gång sades. Respondenten tycker också att Use Case passar särskilt bra vid komplexa krav. Det vill säga där det är väldigt svårt att i bara ord tala om hur det fungerar. Med ett Use Casediagram går det att se samband mellan olika händelser på ett annat sätt. Dessutom går det att för varje händelse eller Use Case lägga till text som förklarar ytterligare.

Respondent 5

Respondent 5 anser att i jämförelse med den problemanalys som respondenten tidigare använde vid kravrepresentation är det ett enkelt koncept vilket innebär att det går snabbt att lära en slutanvändare vad en aktör är och vad ett Use Case är. (Det gäller även för alla intressenter och systemutvecklare) Enligt respondenten är det enkla konceptet framgången med Use Case, att det är så enkelt att komma igång med. Dessutom är tekniken väldigt generell och väldigt lätt att använda. Respondent 5 anser att om

utvecklarna gör en detaljerad kravspecifikation där Use Case ingår och sedan realiserar efter den och dessutom får kunden att skriva på att de här kraven som tagits fram och dokumenterats, är det kunden vill ha. Då har utvecklarna ett dokument som är godkänt av kunden och då kan det användas som testfall. Testpersoner kan titta på de olika flödena, om det faller ut så som det står i kravspecifikationen, de kan titta på avvikelser som kan uppkomma i vissa fall och så vidare.

Respondent 6

Respondent 6 tycker att Use Case ger ett bra sammanhang. Det ger en bild av när kravet är relevant, vem (vilken aktör) som ska utföra en specifik funktion och när funktionen ska utföras. Respondenten anser också att Use Case är speciellt användbart vid test och dokumentation. Det är lättare för alla i projektet att förstå Use Casedokumentationen, i jämförelse med traditionell dokumentation ger Use Case en bra förståelse för dokumentationen. Enligt respondent 6 är det väldigt bra att använda Use Case vid diskussion kring krav och behov med kund. Dessutom beskriver Use Case bättre då aktörerna i ett Use Case är människor och inte andra externa system, det vill säga då det är en människa som interagerar med systemet.

Respondent 7

Respondent 7 anser att Use Case är bra för att beskriva interaktionen mellan en användare och systemet eller interaktionen mellan två system. Det ger en bra beskrivning på hur omvärlden vill använda systemet. Use Case fungerar också bra vid testning. Det vill säga att ett Use Case beskriver ett slags beteende och fungerar som ett normalfall är tänkt att fungera. Sedan finns det andra orsaker runt omkring som kan ställa till det, vilket är alternativa händelser i ett Use Case (exempel på alternativa flöden i kapitel 2.4.1). En Use Casedokumentation ger en röd tråd att följa och en bas att utgå ifrån vid testning.

Respondent 8

Respondent 8 menar att den största fördelen i användning av Use Case är att det går att hitta en nivå av tydlighet där kravställaren och kravutföraren (beställaren och genomföraren) förstår varandra. En traditionell beskrivning av krav i text är så mångtydig att det är väldigt svårt att tro att både den som ställer kraven och den som genomför kraven gör samma tolkning, vilket kan leda till missförstånd kring ett krav. Dessutom anser respondenten att Use Case är planeringsbara. Det vill säga att vid utveckling av ett system ska en mängd Use Case definieras. Alla definierade Use Case behöver inte implementeras samtidigt. Utvecklarna utgår från en mängd Use Case och implementerar dessa först och sedan tar utvecklarna nästa paket med Use Case för att implementera dessa. Det blir väldigt hanterbart att implementera en mindre mängd Use Case i taget. Respondenten tycker också att Use Case fungerar väldigt bra vid testning. Med Use Case blir det väldigt tydligt hur utvecklarna ska gå tillväga vid testning. Det blir i princip ett testfall vid varje scenario i ett Use Case. Det finns tydliga relationer mellan Use Casemodellen och testmodellen.

Respondent 9

Respondent 9 ser en fördel med hanteringen av Use Case ur risksynpunkt och inkrementplanering (se bilaga 10). Det ges möjlighet att tala om vilka Use Case som projektet prioriterar och i vilken ordning som Use Case ska implementeras. Respondenten menar också att det är bra med Use Case när utvecklarna ska gå igenom kraven med användarna. Dessutom fungerar Use Case, i kombination med bland annat användargränssnittet, bra vid testning.

6.2 Nackdelar med Use Case

Respondent 1

Respondent 1 anser att det är lätt att drunkna i allt som ska representeras och dokumenteras i de olika faserna. Samtidigt saknas riktlinjer för hur Use Case ska var utformade. Exempelvis är det svårt att veta hur mycket som ska skrivas ned i text, tio rader, tjugo rader eller ännu mer. Olika utvecklare skapar sin egen version av exempelvis hur Use Case ska dokumenteras och sedan ska de olika versionerna sammanställas till en gemensam version, detta på grund av att klara riktlinjer saknas. Dessutom anser respondenten att Use Case inte är bra för att representera de ickefunktionella kraven. Enligt RUP ska dessa ligga i ett särskilt dokument.

Respondent 2

Respondent 2 tycker inte att de ickefunktionella kraven går att beskriva med hjälp av Use Case. Detta enligt RUP, den process respondenten arbetar utefter. Det är också komplicerat att använda Use Case i ett produktionssystem. Där kommunikationen sker mellan maskiner, det vill säga då aktören är en maskin.

Respondent 3

Respondent 3 anser att det är en nackdel att använda Use Case när det är få eller inga aktörer som är inblandade i systemets funktionalitet. Respondenten tycker dessutom att det inte fungerar bra vid väldigt tekniska system, där det ska specificeras på detaljerad nivå.

Respondent 4

Respondent 4 menar att det är en nackdel att använda Use Case vid små projekt och små utvecklingsuppdrag. Det känns onödigt att lägga ner så mycket arbete på strukturering och dokumentation för att komma fram till en enkel lösning. Dessutom tar det lite mer tid att använda Use Case beroende på all dokumentation, i jämförelse till traditionell dokumentation. Dels tar det mycket tid att strukturera upp arbetet, Use Casemodellering och det tar även mer tid då användarna är mer inblandade i utvecklingsarbetet.

Respondent 5

Respondent 5 anser inte att det finns några nackdelar, tycker bara att det är ett annorlunda sätt.

Respondent 6

Respondent 6 tycker att inte att Use Case fungerar bra då aktören i ett Use Case är ett annat system. Respondenten tycker också att det är besvärligt med krav på olika nivåer, när det finns subsystem och ytterligare subsystem. Det är svårt att bryta ner kraven, det blir mycket dokumentation och väldigt rörigt. Dessutom beskriver Use Case inte de ickefunktionella kraven tillfredsställande.

Respondent 7

Respondent 7 anser att Use Case kan var en nackdel när tekniska system ska byggas (produktionssystem, styrsystem och övervakningssystem). Use Caset kan fungerar som ett komplement, men inte enskilt.

Respondent 8

Respondent 8 tycker att det finns otydlighet i begreppsvärlden som kan orsaka förvirring i projekten. Det finns otydlighet bland annat i hur implementeringen ska gå till, det är svårt att hitta begrepp som beskriver vad det är som ska göras i de olika stegen. Det är även otydlighet vid återanvändbarhet. Många Use Case kan dela på vissa designkonstruktioner och en del designkonstruktioner kan vara specifika för ett Use Case. Även här spelar

begreppsvärlden in, det finns ingen begreppsvärld som idag talar om hur det är tänkt att gå tillväga för att identifiera vad som är delat och vad som är specifikt för ett Use Case.

Respondent 9

Respondent 9 tycker inte att det är lämpligt att bygga en kravspecifikation endast på Use Case. Use Case kan fungera som ett komplement i kravspecifikationen. Dessutom anser respondent 9 att en kund har svårt att förstå Use Case. Respondenten poängterar att Use Case inte är något revolutionerande. Det är flera bitar som behövs kompletteras (se kapitel 6.3).

6.3 Kompletterande tekniker

Respondent 1

Respondenten har ibland använt sig utav kompletterande bilder för att få fram flödet, bland annat statusdiagram. Use Case beskriver inte användargränssnittet på ett tillfredställande sätt därför brukar respondenten komplettera med det. Enligt RUP, som är den process som företaget arbetar efter så ska de ickefunktionella kraven specificeras i ett separat dokument. Respondenten kompletterar dessa i ett separat textdokument.

Respondent 2

Respondenten anser att Use Case ibland behöver kompletteras med GUI-bilder för att framhäva sådant som Use Casetekniken inte klarar av. De ickefunktionella kraven behandlas inte överhuvudtaget i Use Casen och därmed placeras dessa i ett separat dokument för att fånga upp alla krav som ställs på ett system.

Respondent 3

Respondent 3 anser inte att Use Case behöver kompletteras med något. Respondenten menar att i och med att det finns möjlighet att komplettera Use Casediagrammet med text, går det alltid att få fram det som önskas antingen genom bild eller text. De ickefunktionella krav som gäller för ett specifikt Use Case dokumenteras som kompletterande text i Use Casediagrammet. Däremot de ickefunktionella krav som gäller generellt för hela systemet läggs i ett särskilt dokument, detta enligt RUP, processen de arbetar utefter.

Respondent 4

Respondent 4 anser att Use Case ibland behöver kompletteras. Use Case har funnits några år men är en relativt ny teknik och ibland saknas det en del verktyg. Bland annat tycker respondenten att det saknas verktyg för att strukturera och samla in krav. Dessutom används alltid en processmodell för att beskriva det verkliga flödet. Respondenten anser att Use Case inte beskriver det verkliga flödet så bra som processmodellen gör.

Respondent 5

Respondent 5 anser att Use Case bör kompletteras, bland annat med vanliga textdokument. De krav som inte gäller för enskilda Use Case utan generellt för hela systemet hamnar i ett separat dokument, ”Supplementary”. Detta gäller krav som exempelvis ickefunktionella krav, användargränssnitt och vilken standard som ska användas.

Respondent 6

Respondent 6 menar att det finns krav som inte kan representeras med hjälp av Use Case. Det finns även tillfällen då Use Case känns onödig, det produceras alltför mycket dokument gällande för små enkla funktioner. Respondenten kompletterar med separata

dokument för ickefunktionella krav. Dessutom används Use Case inte för att beskriva protokollspecifikationer. Respondenten använder då en annan form av notation och dokumentation. Enligt respondenten är spårbarheten inte tillfredställande med enbart Use Case och behöver kompletteras med sekvensdiagram för att nå bra spårbarhet.

Respondent 7

Respondent 7 anser att Use Case behöver kompletteras. Det bör kompletteras (vilket också görs) med ett separat dokument för de ickefunktionella kraven. Dessutom bör det kompletteras med någon annan teknik då små system ska utvecklas, detta för att slippa så mycket onödig dokumentation. Use Case bör också kompletteras då det är tekniska system som ska utvecklas. Då det är tekniska systemen som ska utvecklas behöver det göras en ordentlig analys, det innebär bland annat enligt respondenten att verksamheten ska modelleras upp för att ge en konceptuell bild av systemet. Use Case kan användas som komplement till analysen som då exempelvis i ett styrsystem kan tala om hur systemet ska styras.

Respondent 8

Respondent 8 anser att med Use Case representeras enbart de funktionella kraven. De ickefunktionella kraven, eller som respondenten kallar dem karaktäristikkrav, kompletteras i ett vanligt textdokument. När hårdvaruorienterade krav som inte är händelseorienterade ska beskrivas väljer respondenten att komplettera med annan teknik på grund av att det är svårt att modellera hårdvarukrav med Use Case.

Respondent 9

Respondent 9 anser att Use Case inte är något revolutionerande och behöver absolut kompletteras. För att en kravspecifikation ska vara bra för utvecklarna bör Use Case kompletteras med ickefunktionella krav, som inte representeras med hjälp av Use Case. Även ett användargränssnitt bör kompletteras och det ska vara så bra beskrivet att utvecklarna kan förstå det. Respondenten menar att sekvensdiagram ska användas som komplement för att beskriva hur interaktionerna ska vara mellan användare och systemet och vilka regler som ska finnas däremellan. Dessutom bör kravspecifikationen kompletteras med en objektmodell och arkitekturmodell.

7 Analys

I analyskapitlet kommer det insamlade materialet (se kapitel 6) att värderas för att se hur det besvarar problemställningen. Under värderingen av materialet kommer respondenternas olika och liknande svar ställas mot varandra. Då det finns kopplingar till information som finns dokumenterad i bakgrunden kommer detta också att tas i beaktande i analysen.

Analysen är indelad i tre delar. I den första delen presenteras en analys kring likheter och olikheter av de fördelar som upplevts med att använda Use Case för att representera krav. På samma sätt analyseras i den andra delen de nackdelar som upplevts med Use Case vid kravrepresentation. I den tredje delen sker analysen utifrån huruvida Use Case bör kompletteras med annan teknik eller inte.

7.1 Fördelar med Use Case vid kravrepresentation

Sju av nio respondenter anser att Use Case fungerar bra i samarbetet med användarna. Möjligheten att med hjälp av Use Case, i många fall Use Casediagram, diskutera kring kraven och stämma av så att kraven uppfattas på samma sätt av både utvecklaren och användaren ses av flertalet respondenter vara den största fördelen med Use Case. Ett eller en samling Use Case utgör en bas för kommunikation mellan kunden och systemutvecklarna (Fowler och Scott, 2000). Att Use Case fungerar bra tillsammans med användarna och framför allt att Use Casedokumentationen fungerar som diskussionsunderlag, kan relateras till hur viktigt det är att de framtagna kraven stämmer överens med de krav och behov som användare och andra intressenter ställer på systemet (se kapitel 2.1). Då majoriteten av respondenterna och litteratur som stärker den faktorn tyder det på en fördel med att använda Use Case för att representera krav.

Fem av nio respondenter anser att Use Case är lätt för användaren att förstå. Respondent 9 däremot har tagit upp det som en nackdel med Use Case. Respondent 9 anser att det är svårt för kunden att förstå ett Use Case. Vid en närmare analys av vad respondent 9 ser för fördelar med Use Case, nämnde denna person att det var bra att använda Use Case när utvecklarna ska gå igenom kraven med användarna. Sammanfattningsvis anser respondent 9 att Use Case är svårt att förstå för *kunden*, men att det fungerar bra när utvecklarna ska gå igenom kraven med *användarna* (kunden och användare betraktas som en intressent, i och med att en kund och användare kan vara samma person). Respondent 9:s ovan nämnda åsikter om Use Case är motsägande i det avseende att det inte verkar rimligt att det skulle vara bra att använda Use Case för att gå igenom kraven med användaren då det är svårt för denne att förstå ett Use Case. Utifrån intervjun med respondent 9 framgår det inte exakt vad respondenten menar med att det är svårt för kunden att förstå ett Use Case. Hur ett Use Casediagram ska utformas (att modellera Use Case) kanske anses vara svårt för användarna att förstå, medan förståelsen för vad ett Use Case är inte är svårt att förstå, men vad respondenten avser framgår inte av intervjun. Då fem av nio respondenter anser att det är lätt för användare att förstå Use Case och en respondent, som talar emot sig själv, upplever motsatsen tolkas detta som en fördel med att använda Use Case vid kravrepresentation.

Sju respondenter anser att Use Case fungerar bra vid testning. Ingen annan av respondenterna motsäger det. Karlsson (1996) menar att testfall kan baseras på Use Case. Denna information tolkas som en fördel med att använda Use Case för att representera krav.

Respondent 4 tycker att Use Case lämpar sig bra för att beskriva komplexa krav. Det vill säga där det är väldigt svårt att i bara ord tala om hur något ska fungera. Respondenten

menar att det med ett Use Casediagram går att se samband mellan olika händelser, vilket är bra vid beskrivning av komplexa krav. I RE-processen fungerar Use Case bra för att hantera komplexiteten hos systemet genom att identifiera olika Use Case och analysera dem (Regnell, 1999). Detta ger möjlighet att fokusera på endast en aspekt av systemet i taget (Regnell, 1999). Enligt Leffingwell och Widrig (2000) är Use Case speciellt utvecklat för att passa objektorienterade metoder (bland annat RUP) som i sin tur är utvecklade för att hantera komplexa system. Respondent 8 och 9 anser att Use Case är planeringsbara och bra att använda ur risksynpunkt. De menar att olika Use Case identifieras och delas upp i paket med olika riskfaktorer. Dessa paket kan implementeras var för sig där de paketerna med hög riskfaktor implementeras först. Det kan vara de krav som kunden vill ha hjälp med först. Detta kan jämföra med litteraturen (se kapitel 2.4.2). Enligt Regnell (1999) är Use Case bra vid komplexa system, där det går att fokusera på en del av systemet i taget. Detta kan vara det som respondent 8 och 9 menar med att dela upp kraven i olika paket och fokus kan läggas på den del som är viktigast just nu. Då litteratur (se kapitel 2.4.2) stödjer respondent 4 åsikt om komplexa krav och då det finns koppling mellan komplexa krav och planeringsbarhet betraktas båda dessa faktorer som fördel med att använda Use Case vid kravrepresentation.

Respondent 2 ser möjligheten att representera Use Case i olika nivåer som en stor fördel. Respondenten menar att det i olika nivåer går att beskriva kraven på olika sätt för olika intressenter. Exempelvis är chef och programmerare inte intresserade utav samma information. Respondent 4 nämnde att denna också använder olika nivåer för att representera krav fast respondent 4 har inte framfört det som en fördel eller nackdel utan mer som en förklaring på hur respondenten arbetar med Use Case vid kravrepresentation. Däremot har respondent 6 en helt annan syn på att skapa nivåer för att dokumentera krav med Use Case. Respondent 6 har erfarenhet av att det är bevärligt med olika nivåer vilket leder till att dokumentationen blir rörig. Det finns inte något som pekar på att det är en fördel eller en nackdel med att använda olika nivåer för att representera krav. Istället tyder det mer på ett sätt att arbeta, det vill säga att systemutvecklarna väljer det sätt de själva tycker känns bra för att representera kraven på.

Respondent 2 tar upp en viktig aspekt, nämligen det itererande arbetssättet som finns i arbetet med Use Case. Det itererande arbetssättet bidrar till att kraven upptäcks på ett tidigt stadium i systemutvecklingsarbetet. Detta är en viktig faktor som kan bidra till ett framgångsrikt system (se kapitel 2.3).

7.2 Nackdelar med Use Case vid kravrepresentation

De nackdelar som framkom av respondenterna visar inte lika tydligt någon eller några gemensamma faktorer i jämförelse med vilka fördelar respondenterna upplevde.

Tre av nio respondenter menar att det inte går att beskriva ickefunktionella krav med Use Case. Leffingwell och Widrig (2000) menar att Use Case inte är att rekommendera när det gäller att representera de ickefunktionella kraven. Dessutom upptäcktes vid en närmare granskning att åtta av nio respondenter väljer att komplettera de ickefunktionella kraven med ett separat dokument (se kapitel 6.3). Om respondenterna väljer en kompletterande teknik bör det finnas något som bidrar till att de ickefunktionella kraven inte beskrivs tillfredsställande med hjälp av Use Case, någon nackdel som talar emot användning av Use Case. Informationen ovan pekar på att Use Case inte bör användas för att beskriva de ickefunktionella kraven.

Två respondenter anser att det finns otydligheter i begreppsvärlden. Det saknas riktlinjer för hur Use Case ska vara utformade. Bland annat saknar respondenterna riktlinjer för hur implementeringen ska gå till och hur detaljerad dokumentationen bör vara. Regnell

(1999) har identifierat nackdelar med Use Case där han bland annat anser att det är svårt att tolka aktör- och Use Casekoncepten. Vidare menar Regnell (1999) att det inte finns någon klar definition av semantiken hos Use Case och att det inte finns några riktlinjer för hur Use Case skall beskrivas. Detta tyder på ytterligare en nackdel med Use Case vid kravrepresentation.

Respondent 4 anser att det är en nackdel att det tar mer tid att använda Use Case vid kravrepresentation. Det beror bland annat på den omfattande dokumentation som ingår i UML och Use Case. Men här vill respondenten poängtera att trots den extra tiden i arbetet med Use Case så tjänar systemutvecklingen på det i det långa loppet. Respondent 4 menar att resultatet har blivit bättre i användning av Use Case än vid traditionell systemutveckling. Detta tyder på att tidsåtgången är en nackdel med att använda Use Case, dock ska här inte användas någon kompletterande teknik ty resultatet tenderar att bli bra. Respondent 4 menar också att vid små projekt och små utvecklingsuppdrag känns det onödigt att använda Use Case i och med all dokumentation som krävs. Respondent 1 anser också dokumentationen kan kännas arbetsam, då det blir mycket som ska representeras och dokumenteras. Respondenterna 1 och 4 åsikter om dokumentationen kan höra samman med att det ibland saknas riktlinjer för hur kravrepresentationen ska skötas. Det saknas begrepp som talar om vad som ska göras i de olika stegen och hur mycket dokumentation som krävs.

Då det är få eller inga aktörer inblandade i systemet är det enligt respondent 3 en nackdel att använda Use Case för att representera krav. Leffingwell och Widrig (2000) har samma åsikt som respondent 3. System med endast ett fåtal eller inga användare och med få externa gränssnitt, är exempel på system där Use Case inte ses som den effektivaste tekniken det finns i dessa fall enklare sätt att beskriva majoriteten av kraven på (Leffingwell och Widrig, 2000).

Som tidigare nämnts menar Fowler och Scott (2000) att aktören i ett Use Case kan vara både en människa eller ett extern system. Fowler och Scott (2000) säger däremot inget om att det skulle vara någon skillnad på att representera aktören som en människa eller maskin. Respondent 2 och 6 anser att det är svårare att använda Use Case för att representera krav i system där kommunikationen mestadels sker mellan maskiner, när aktören är en maskin, än vad det är när aktören är en människa. Respondent 2 menade att ett sådant system kan vara ett produktionssystem. Respondent 3 och 7 anser att Use Case kan vara en nackdel att använda då tekniska system ska byggas. Respondent 7 säger under intervjun att tekniska system som exempelvis produktionssystem, styrsystem och övervakningssystem borde representeras av någon annan teknik än Use Case. Det är svårt att avgöra utifrån informationen från intervjuerna huruvida respondenterna som nämnt dessa två olika faktorer (då aktören är en maskin och tekniska system) avser samma sak. Med all sannolikhet uppstår det tillfällen då det både handlar om tekniska system som ska utvecklas och där aktörerna mestadels är maskiner. Dessa två faktorer pekar på att Use Case fungerar mindre bra att använda vid kravrepresentation för tekniska system och där aktören är en maskin.

7.3 Kompletterande tekniker

Åtta av nio respondenter kompletterar kravrepresentationen med ett separat dokument, vanligtvis ett vanligt textdokument, för att beskriva de ickefunktionella kraven. Den nionde respondenten (respondent 4) torde med all sannolikhet också göra det, då det framgår av intervjun att respondenten arbetar utefter RUP. I RUP ska de ickefunktionella kraven, enligt respondent 1 och 3, dokumenteras i ett separat dokument som kallas

”supplementary”. Som nämndes i kapitel 7.2 borde ickefunktionella krav enlig Leffingwell och Widrig (2000) inte representeras med hjälp av Use Case, de menar att det borde överlämnas åt andra tekniker. Enligt respondent 3 kan Use Case användas för att beskriva de ickefunktionella krav som hör till ett visst Use Case, men då det ickefunktionella kravet gäller generellt för hela systemet placeras de kraven i ett särskilt dokument. Enligt respondent 5 är de krav som gäller generellt för hela systemet, ickefunktionella krav och de krav som gäller enbart för ett specifikt Use Case är funktionella krav. Leffingwell och Widrig (2000) säger också att det finns tillfällen då de ickefunktionella kraven kan beskrivas med Use Case, det vill säga då kraven endast ska appliceras på ett eller ett fåtal Use Case, inte generellt för hela systemet. Enligt Loucopoulos och Karakostas (1995) finns det inte helt klara *regler* som talar om skillnaderna mellan funktionella och ickefunktionella krav. Det är inte ovanligt att ett krav först klassas som ickefunktionellt krav för att sedan förändras till ett funktionellt krav. Denna värdering av insamlat material pekar på att det inte alltid är lätt att avgöra vad som är ett funktionellt eller ickefunktionellt krav. Men det är ändå tydligt att de ickefunktionella krav som gäller generellt för hela systemet bör kompletteras med ett annat dokument.

Enligt respondent 1 beskriver Use Case flödet bra. Respondent 4 däremot ansåg att flödesbeskrivningen borde kompletteras med en annan teknik. I detta fall använde respondent 4 en processmodell som komplement för att beskriva det verkliga flödet. Även respondent 1 använder kompletterande teknik för att beskriva flödet, ett statusdiagram. Respondent 1 säger mot sig själv då respondenten först tycker att flödet beskrivs bra och sedan använder kompletterande teknik för att beskriva flödet. Kanske denna motsägelse bero på vilken typ av flöde som ska beskrivas. Respondent 4 kompletterar det verkliga flödet och respondent 1 kompletterar det flöde som talar om i vilken ordning som funktioner utförs. Det framgår inte av intervjun vad de olika respondenterna exakt menar med de olika flödena, om de menar samma sak eller om det är olika slags flöden de syftar på. Hur som helst är det tydligt att någon form av kompletterade flödesbeskrivning används för att förtydliga flöden i systemet.

Som tidigare nämnts i kapitel 7.2 ansåg respondent 7 att Use Case fungerade mindre bra vid tekniska system. Respondent 7 anser att en ordentlig analys av systemet ska göras och det ska kompletteras med konceptuell bild av systemet. Use Case kan användas som komplement för att beskriva hur systemet exempelvis ska styras.

Respondent 9 ansåg att Use Case inte är något revolutionerande för systemutvecklingen. Use Casetekniken bör kompletteras med bland annat användargränssnitt, sekvensdiagram, objektmodell och arkitekturmodell. Respondent 9 tenderar att under intervjun fokusera på Use Case ur kravspecifikationens synpunkt. Respondent 9 arbetar med att ta fram kravspecifikationer och menar att den inte är tillfredsställande om den enbart skulle bygga på Use Case. Kravspecifikationen bör enligt respondenten kompletteras med ovan nämnda tekniker. Det har under intervjun framkommit information om att Use Case behöver kompletteras med olika tekniker, men inte så mycket varför. Men trots allt är kravspecifikationen som tidigare nämnts i kapitel 2.2.2 en viktig ingrediens i kravhanteringen och därför är det viktigt att även nämna dessa faktorer.

7.4 Värdering av material

Vid flera tillfällen angav respondenten en faktor som för- eller nackdel, men förklarar inte varför den upplevs som en för- eller nackdel. Detta har bidragit till att det ibland varit svårt att analysera materialet. Det är svårt att veta hur respondenten har tänkt, vilket i viss mån kan ha påverkat resultatet. Trots vissa oklarheter från intervjuerna finns det så mycket insamlat material att det finns underlag för att besvara problemställningen.

Respondent 5 ansåg att denna inte finner några nackdelar med att använda Use Case för att representera krav. Detta kan så vara, men då respondenten under intervjun säger: ”*men nackdelar vet jag inte om det finns det är bara ett annat sätt, jag kan inte säga att det är en fördel eller en nackdel mot det andra*” förmodas att respondenten jämför Use Case mot det tidigare sättet denne har använt för att representera krav, problemanalysen. Avsikten med intervjufrågan var att ta reda på om respondenter funnit några nackdelar med tekniken, inte jämföra med tidigare använd teknik vid kravrepresentation. Detta upptäcktes inte förrän materialet i efterhand bearbetades. Att respondenten missuppfattade avsikten med frågorna för att ta fram nackdelar med Use Case, kan bero på dennas nuvarande arbetsuppgift. Respondentens arbetar för tillfället som metodutvecklare. Missuppfattningen kan även bero på intervjuarens bristande erfarenhet av att intervjua och formulera frågor. Det är möjligt att om en pilotintervju genomförts först och att materialet hade bearbetats och analyserats på samma sätt som ovan, kunde sådana här misstag ha undvikts.

8 Resultat

I detta kapitel presenteras resultatet av intervjuerna med avseende på rapportens problemställning:

Vilka för- och nackdelar finns det med att använda Use Case för att representera krav och anpassas kravrepresentationen efter dessa?

8.1 Fördelar med Use Case för att representera krav

- Use Case är ett bra sätt att representera krav på för att det är lätt för användare, kunder och andra intressenter att förstå vad ett Use Case är och vad en aktör är. I och med att Use Case är så enkel att förstå är det ett bra sätt att möta användaren och kunden på.
- Att dokumentera krav med hjälp av Use Case fungerar bra. Use Casediagrammet, används som ett diskussionsunderlag mellan systemutvecklaren, användaren och kunden för att kontrollera att de framtagna kraven har tolkats på samma sätt av de olika parterna. Detta arbetssätt tillåter att användarna är mer inblandade i utvecklingsprocessen i jämförelse med traditionellt sätt att representera krav på.
- Use Case fungerar bra vid testning. Det vill säga att det är möjligt att gå tillbaka till dokumentationen av systemets olika Use Case och kontrollera att systemet fungerar som det var tänkt att det skulle göra. Ett Use Case fungerar som en verifikation på det som en gång sades.
- Use Case är bra för att beskriva komplexa krav, det beskriver samband mellan olika händelser. Dessutom beskriver Use Case kraven bättre då aktören är en människa än om aktören är en maskin.
- Use Case är planeringsbara och bra ur risksynpunkt.
- Det är ett itererande arbetssätt som bidrar till att kraven upptäcks på ett tidigt stadium.
- En bra teknik då det är många användare inblandade. Det vill säga när det finns många olika aktörer som interagerar med systemet.

8.2 Nackdelar med Use Case för att representera krav

- I ett Use Case går det inte att beskriva de ickefunktionella kraven.
- Det finns otydligheter i begreppsvärlden det vill säga att det saknas riktlinjer för hur Use Case ska vara utformade.
- Use Case kräver mycket dokumentation, vilket är tidskrävande. Det är en nackdel att använda Use Case vid små projekt och små utvecklingsuppdrag på grund av att dokumentationen blir så omfattande och det går åt mycket dokumentation för att komma fram till en enkel lösning.
- Det är svårt att använda Use Case för att representera krav för ett tekniskt system och system där kommunikationen sker mellan maskiner, då aktören är en maskin.
- Då det är få eller inga aktörer lämpar det sig inte att använda Use Case. Det är kraven ur användarnas synvinkel som ska fångas med hjälp av Use Case och då behöver det användare för att använda tekniken.

8.3 Anpassning av kravrepresentation

De ickefunktionella kraven som gäller generellt för hela systemet bör kompletteras med ett separat dokument. Ofta används ett vanligt textdokument som komplement till Use Casedokumentationen.

Use Case kompletteras också då olika flöden ska beskrivas. Bland annat används en processmodell för att beskriva det verkliga flödet och ett statusdiagram för att beskriva i vilken ordning funktionerna ska utföras. Som nackdel nämndes att Use Case fungerade mindre bra vid tekniska system och där sker istället en ordentlig analys av systemet med en kompletterande konceptuell bild av systemet. Det framgår att Use Case kan användas som komplement för att beskriva hur exempelvis ett styrsystem ska styras.

Use Casetekniken bör kompletteras med bland annat användargränssnitt, sekvensdiagram, objektmodell och arkitekturmodell för att en kravspecifikation ska vara bra. Sekvensdiagram ska användas som komplement för att beskriva hur interaktionerna ska vara mellan användare och systemet och vilka regler som ska finnas däremellan. Det bidrar till att spårbarheten hos ett Use Case blir bättre.

Då det finns otydligheter i begreppsvärlden behövs kompletterande verktyg för att strukturera upp arbetet. Det finns idag några verktyg som används men det saknas tydligare riktlinjer för hur Use Case ska användas.

8.4 Övriga resultat

Att beskriva Use Case på olika nivåer betraktas som individuellt, att respondenterna väljer det sätt de själva tycker passar dem bäst att arbeta utefter. Därför kan inte någon slutsats dras som pekar på att det entydigt är en för eller nackdel.

Att använda Use Case för att beskriva flödet har inte kunnat identifieras som varken en fördel eller en nackdel då två respondenter talade mot varandra. Motsägelsen kan bero på vilket flöde som beskrivs.

9 Diskussion

I detta kapitel redovisas en diskussion kring resultatet av undersökningen. Därefter presenteras de erfarenheter som gjorts under arbetets gång och slutligen lämnas förslag på fortsatta arbeten.

9.1 Diskussion kring resultatet

Erfarenhet av intervjuer är viktigt för att resultatet ska blir tillfredsställande. Vid några tillfällen under intervjuerna saknas det en förklaring på vad respondenten avser med exempelvis en specifik för- eller nackdel. Om intervjuaren varit observant på dessa oklarheter redan under intervjutillfället och bett respondenten utveckla svaret, hade det i efterhand varit lättare att analysera svaren. Detta kunde intervjuaren ha varit bättre på och det kunde ha resulterat i ett fylligare och mer beskrivande material.

Materialet från intervjun med respondent 6 baseras, till skillnad från resterade respondenter, på anteckningar från intervjun. Detta medförde att en del information från intervjun inte har registrerats. Detta anses inte ha påverkat resultatet, då informationen om huvudfrågorna blivit väl dokumenterade under intervjutillfället. Övrig information kunde ha blivit mer utförligt beskrivet, dock anses inte detta vara av så stor vikt att det påverkat resultatet.

Då fem av de nio respondenterna arbetar på samma företag skulle det finnas anledning att tro att det skulle påverka resultatet. Men då endast två av de fem respondenterna arbetat i samma projekt anses det inte ha påverkat resultatet i det stora perspektivet. Däremot kan det finnas vissa likheter i deras svar. Men likheter i svaren finns det mellan de flesta respondenterna mer eller mindre.

Undersökningen genomfördes både via personliga intervjuer och via telefonintervjuer. Till en början var tanken att, där det var möjligt, genomföra personliga intervjuer. Avstånd och geografisk spridning var orsaker till att telefonintervjuer genomfördes. Dessutom ansåg respondenterna i Stockholm och Linköping att telefonintervju passade dem bäst, vilket kan bero på att de förmodade att en telefonintervju skulle gå snabbare att genomföra. Det har inte upplevts någon skillnad på materialet som samlats in genom de två olika intervjusätten. Båda sätten att genomföra intervjuerna på gav tillfredställande material för att besvara problemställningen.

Som tidigare nämnts ställdes inledande frågor för att skapa en kort bakgrund av de olika respondenterna (se kapitel 4.6.2). Det skulle även vara intressant att analysera huruvida respondenternas olika långa erfarenheter med Use Case har haft någon inverkan på resultatet. Efter en närmare granskning av det insamlade materialet framgick det inte något som pekar på att erfarenheten bland respondenterna har haft någon inverkan på resultatet. Det vill säga det har inte funnits någon markant skillnad mellan exempelvis de respondenter som arbetat två år med Use Case och de respondenter som har arbetat över fem år med Use Case. Om en undersökning med ett större antal respondenter skulle ha genomförts skulle denna aspekt kunnat se annorlunda ut.

I kapitel 2.4 nämns det kort att det finns olika versioner av Use Case och olika versioner av UML. Då detta inte har behandlats i denna rapport är det svårt att avgöra om det hade påverkat resultatet ifall de olika respondenterna hade använts sig av olika versioner av Use Case.

9.2 Erfarenheter av arbetet

Både positiva och negativa erfarenheter kan i det här stadiet av examensarbetet identifieras. Planering är en viktig del av arbetet och särskilt viktig var det då undersökningen påbörjades. Det går inte att förvänta sig att alla personer som ska kontaktas är anträffbara, eller har tid när som helst för att i detta fall, intervjuas. Det var inte några problem med tidsplaneringen för detta examensarbete då kontakten med respondenterna gjordes i god tid. Dock fanns det andra problem i kontakten med olika företag. När företagen kontaktades var det svårt att få tag på de personer på företagen som arbetar med Use Case för att representera krav. Det var den första kontakten med företaget som var besvärlig därför att kontaktpersonen inte var insatt i vad de olika personerna på företagen specifikt arbetade med. Vid flertalet tillfällen lovade kontaktpersonerna höra av sig då de funnit någon person som var lämplig för intervju. Då kontaktpersonerna inte hörde av sig gjordes nya försök att kontakta dem. Svaren som kontaktpersonerna meddelade, vilka ofta var negativa, kändes ibland som undanflykter. Orsaker till att det kontaktpersonerna svarade, som jag uppfattades som undanflykter, kan vara att kontaktpersonerna själva var väldigt upptagna eller inte fann något eget intresse i undersökningen. Det var även svårt att veta hur de olika kontaktpersonerna upplevde den presentation som gavs då den första kontakten togs med företagen. Som tidigare nämnt är det viktigt att den första kontakten med företag och eventuella respondenter är intressant för dem och att de finner något eget intresse i att medverka i undersökningen.

Att intervjua personer har upplevts som både intressant och svårt. Det har varit intressant att tagit del av någon annan persons erfarenhet inom ett ämne som är intressant för både respondent och intervjuaren. Under intervjuerna har det varit svårt att bland annat formulera frågor så att de inte ska misstolkas och att inte ställa ledande frågor till respondenten. Att använda intervjuer för att samla in material har trots svårigheterna, varit en tillfredsställande teknik, då det har gett möjligheter att förtydliga frågor, ställa följdfrågor och utreda konstiga begrepp.

Antalet intervjuer har givit en bild av vad olika systemutvecklare finner som för- och nackdelar med att använda Use Case för att representera krav. Det insamlade materialet har även visat exempel på tekniker som används som komplement till Use Case. Det kan tänkas att resultatet kunde ha blivit annorlunda om intervjuerna hade kompletterats med en litteraturstudie. Dock återfinns i analysen, kopplingar mellan intervjuerna och olika författares kritik till Use Case som finns presenterat i bakgrunden.

Det har inte gjorts några uppföljande intervjuer för att kontrollera att det insamlade materialet har tolkats rätt av intervjuaren. Detta är något som borde ha gjorts för att vara säker på att det presenterade materialet stämmer överens med vad respondenterna avsåg med sina svar. Hade en kontroll av insamlat och bearbetat material gjorts, så hade oklarheter och missförstånd kunnat redas ut.

9.3 Förslag på framtida arbeten

Efter att genomfört detta arbete har det dykt upp några förslag på fortsatta arbeten. Vissa av de fördelar och nackdelar med Use Case vid kravrepresentation som framkommit under detta arbete tyder på att vara representativa för tekniken Use Case. Om den här undersökningen ses som en förundersökning skulle det vara intressant att undersöka huruvida resultatet skulle blir detsamma vid en större undersökning. Det skulle även vara intressant att undersöka hur utvecklingsarbetet med Use Case under de olika faserna i RE-processen uppfattas ur användarnas perspektiv. Att undersöka vad som upplevs som positivt/negativt från användarens sida. Spårbarhet är också en väldigt intressant aspekt och dessutom väldigt viktigt vid kravrepresentationen. Vad

krävs för att spårbarheten i en Use Casedokumentation ska vara tillfredsställande. Hur fungerar spårbarheten mellan ett implementerat system och Use Case.

Referenser

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

Booch, G. och Rumbaugh, J. och Jacobson, I. (1999), *The Unified Modeling Language User Guide*, Addison Wesley Longman.

Britton, C. och Doake, J. (1993), *Software System Development A gentle introduction*, McGraw-Hill International.

Cockburn, A. (2001), *Writing Effective Use Cases*, Addison Wesley.

Ejevegård, R. (1996), *Vetenskapliga metoder*, Studentlitteratur, Lund.

Fowler, M. och Scott, K. (2000), *UML Distilled Second Edition*, Addison Wesley Longman.

Karlsson, J. (1996), *Framgångsrik kravhantering – vid utveckling av programvarusystem*, Sveriges Verkstadsindustrier.

Kotonya, G. och Sommerville, I. (1998), *Requirements Engineering*, John Wiley & Sons: New York.

Krag, Jacobsen, J. (1993), *Intervju Konsten att lyssna och fråga*, Studentlitteratur, Lund.

Leffingwell, D. och Widrig, D. (2000), *Managing Software Requirements*, Addison Wesley Longman.

Loucopoulos, P. och Karakostas, P. (1995), *Systems requirements engineering*, McGraw-Hill: International.

Patel, R. och Davidson, B. (1994), *Forskningsmetodikens grunder*, Studentlitteratur, Lund.

Pohl, K. (1996), *Process-Centred Requirements Engineering*, Research Studies Press Ldt.

Regnell, B. (1999), *Requirements Engineering with Use Cases – Basis for Software Development*, Teknisk rapport nr 132, Lund Universitet.

Rosenberg, D. och Scott, K. (1999), *Use Case Driven Object Modeling with UML*, Addison Wesley Longman.

Simons, A.J.H. (1998), *Use Cases Considered Harmful*, http://www.computer.org/proceedings/tools/0275/02750194abs.htm [hämtat 2001-02-25].

Sommerville, I. och Sawyer, P. (1997), *Requirements Engineering A good practice guide*, John Wiley & Sons.

Bilaga 1 – Intervjufrågor

Inledande frågor

1. Hur länge har du varit anställd på företaget?
2. Hur lång erfarenhet har du inom systemutveckling?
3. Hur lång erfarenhet har du av att arbeta med Use Case?

Huvudfrågor

4. Har projekten du arbetat i valt att representera kraven i textform eller med hjälp av Use Casediagram? Varför?
5. Vilka fördelar ser du med att använda Use Case för att representera krav?
6. Vilka nackdelar ser du med att använda Use Case för att representera krav?
7. Vid vilka typer av krav, exempelvis funktionella krav och ickefunktionella krav, har Use Case fungerat tillfredsställande vid kravrepresentation?
8. Vid vilka typer av krav har Use Case inte fungerat tillfredsställande vid kravrepresentation?
9. Vid vilka tillfällen har Use Case fungerat tillfredsställande för att representera krav?
10. Vid vilka tillfällen har inte Use Case fungerat tillfredsställande för att representera krav?
11. Anser du att Use Case bör kompletteras med någon annan teknik för att representera krav? Varför / varför inte.
12. Har du använt någon kompletterande teknik för att representera krav? Varför / varför inte.

Om ett ja-svar på fråga 11 och ett nej-svar på fråga 12 ställs fråga 13

13. Om du anser att kravrepresentationen bör kompletteras och ändå inte gjort det, vad tror du det har haft för konsekvenser för kravrepresentationen?

Bilaga 2 – Intervju respondent 1

Respondent 1 har arbetat på företaget i fem år med systemutveckling och utav de fem åren har det varit ett års arbete med Use Case. Respondent 1 arbetar i samma projekt som respondent 3 och de arbetar utefter RUP med UML som modelleringsspråk. Då Use Case används vid kravrepresentation används både Use Casediagram och Use Case i textform.

Fördelar

Respondent 1 anser att det är lätt att få upp roller i systemet, användarna kan lätt förstå och begripa vad som menas med Use Casediagrammet. Respondenten menar att Use Case utgår från aktörerna, utifrån dem skapas en dialog mellan utvecklare och användare om vad aktörerna representerar, vad ska varje aktör göra. Det sker en kontroll att utvecklare och användare menar samma sak om kraven. Respondenten anser också att Use Case beskriver flödet på ett bra sätt, respondenter menar att projektet skrev Use Casen i den ordning som de ska utföras sedan. Respondenten tycker att det är bra att systemutvecklarna kan, genom att använda Use Case, få ut ett normalflöde, de steg där flödet fungerar där det ska. Respondenten anser också att Use Casen följer som en röd tråd genom hela systemets livscykel (inte bara under projektet), det är tänkt att utvecklarna ska kunna bygga hela systemet runt Use Casen, och sedan kunna följa upp med testfall.

Nackdelar

Respondent 1 tycker att utvecklarna kan ”drunkna” i hur mycket som ska representeras och dokumentera i olika faser. Det finns inga riktlinjer för hur ett Use Case ska vara utformat, exempelvis har vi haft mycket diskussioner kring hur mycket som ska skrivas ned i text för ett Use Case , tio rader, tjugo rader eller hundra rader? Det finns inte någon som ger något svar, bara olika bud. Det är svårt att veta om systemutvecklarna gör rätt eller fel, det märks först i efterhand. Exempelvis den tid som lagts ner på att skriva har ibland gjorts i onödan på grund av att det inte funnits någon eller något som hjälp till att avväga hur mycket som ska skrivas och dokumenteras. Alla utvecklare skapar sina egna versioner av hur denna vill ha det och sedan ska olika åsikter samlas till ett sätt som ska följas av alla. Respondenten tycker att det är lätt att det blir mycket text, på grund av all dokumentation som ska göras. Detta är inte bra i de avseende att kunderna upplever det som jobbigt att gå igenom all dokumentation. Dessutom använder inte respondenten inte sig utav Use Case för att beskriva de ickefunktionella kraven. Enlig RUP, som respondenten arbetar utefter, ska de ickefunktionella kraven ligga i ett särskilt dokument.

Kompletterande tekniker som respondenten använt sig utav är andra kompletterande bilder för att bland annat få fram flödet, exempelvis statusdiagram (saker som ska ske i viss ordning). Vi har ritat användargränssnitt som behövs som komplement till Use Case. Dessutom används ett separat dokument för de ickefunktionella kraven.

Bilaga 3 – Intervju respondent 2

Respondent 2 har arbetat som systemutvecklare på företaget i 5,5 år. I ungefär 2 år har respondenten arbetat med Use Case. Respondenten har 5 års erfarenhet inom systemutveckling innan dess att denna anställdes på nuvarande arbetsplats. De projekt som respondenten har använt sig av Use Case för att representera krav har denna kombinerat Use Casediagram med Use Case i textform. Större delen av alla Use Case är dokumenterade i text, där diagrammet kompletterar med att grafiskt beskriva övergripande hur arbetsflödet är tänkt att fungera. Respondenten beskriver Use Case på olika nivåer. Syftet är att dokumentera användarnas krav, och få en bekräftelse från verksamheten på att det stämmer överens med deras krav, dessutom ska IT människorna kunna förstå den.

Fördelar

Respondent 2 anser att det är ett bättre sätt att möta användaren på. Användarna kan delta i analysarbetet på ett bättre sätt. Det är ju deras krav som ska beskrivas i Use Casediagrammet. Use Case är ett itererande arbetssätt vilket är en fördel för att upptäcka kraven på ett tidigt stadium. Det är även lättare för användare och andra intressenter att förstå vad ett Use Case är. Ett Use Casediagram är en fördel att använda då det genererar i en grafisk bild. Dessutom beskriver respondenten Use Case i olika nivåer, vilket denna anser vara en fördel för att slippa dokumentera allt på detaljnivå. Det är endast det som ska visas för programmerare som behöver beskrivas i detalj. Respondenten förklarar att den informationen som en chef är intresserad utav är inte programmeraren intresserad av. Slutligen menar respondenten att Use Case passar bättre att representera krav i administrativa system där det finns en riktig användare

Nackdelar

Respondenten tror att det är svårare att använda Use Case i ett produktionssystem där det är kommunikationen sker mellan maskiner. Det vill säga då aktören är en maskin. Enligt den metod som respondenten arbetar utefter beskrivs inte de ickefunktionella kraven med hjälp av Use Case. Respondenten menar att det kanske inte är en nackdel, men att det har varit svårt och tagit mycket tid att skriva Use Case på rätt nivå. I och med att Use Case förespråkar användarnas delaktighet och att respondenten har arbetat väldigt nära användare har beskrivningen på rätt nivå upplevts som problematiskt. Användarna vill att Use Casen ska skrivas på ett ”språk”, programmeraren på ett ”språk” och respondenten själv på ett ”språk” vilket har medfört att dokumentationen upplevts som problematiskt. Det är viktigt att dokumentationen blir ordentligt gjord och att den är i användarens vokabulär. Vi använder Use Case i olika syften, dels för att få en bekräftelse från verksamheten (användarna) och för att IT-människorna ska kunna förstå dokumentationen.

Use Casetekniken kan ibland behöva kompletteras med GUI bilder för att framhäva vissa saker ännu tydligare än vad som kan göras med Use Case. Men detta har inte gjorts i de projekt som Mikael arbetar / arbetat i, ty de arbetar efter en strikt mall som just det IT-projektet arbetar utefter. Jag som enskild analytiker får inte gå utanför de ramar som finns och som det har tagits beslut om. Dessutom kompletteras de

ickefunktionella kraven i ett separat textdokument. Respondenten nämnde att Use Case inte skulle fungera så bra vid produktionssystem. Respondenten tror att det borde finnas bättre tekniker att använda, men då denna inte har arbetat med att utveckla produktionssystem vet respondenten inte exakt hur det verkar.

Bilaga 4 – Intervju respondent 3

Respondent 7 har arbetat med systemutveckling på IT-företaget i tre år de tre åren. Respondenten har arbetat med Use Case i två projekt.. Jag och två till var med i ett pilotprojekt med att arbeta utefter RUP, det var 1999, och det andra är det jag håller på med nu. Totalt blir det väl 1,5 år med Use Case. Respondenten har arbetat som analytiker, systemanalytiker. Use Case presenteras i en Use Casemodell, med Use Case och aktörer denna modell kompletteras också med beskrivande text. Detta rekommenderas i RUP som det projektet arbetar efter.

Fördelar

Use Case ger en bra helheltsbild direkt. Det går fort att få fram en helhetsbild som givetvis senare blir mer detaljerad. Användarna är även med när den görs.

En annan fördel är att användarna är inblandade i utvecklingsarbetet.

Nackdelar

När det är få eller inga aktörer inblandade i ett Use Case.

När det är väldigt tekniska system, där det är specning på låg nivå

Kompletterande tekniker

Nej, respondenten anser att i och med att Use Casediagrammet kompletteras med text, kan man alltid få fram det man vill antingen genom bild eller text.

Respondenten pratar om att Use Casen är användarkraven och skiljer sig från det traditionella sättet där det fokuseras mycket på de funktioner som systemet skall utföra.

Use Casediagrammet beskriver de funktionella kraven. Den beskriver även de ickefunktionella kraven som hör till ett visst speciellt use case, men gäller det ickefunktionella kravet generellt över hela systemet hamnar de kraven i ett särskilt dokument. Detta enligt processen RUP som de arbetar utefter.

Bilaga 5 – Intervju respondent 4

Respondent 4 har arbetat som systemutvecklare på sin nuvarande arbetsplats sedan 1995. Respondenten har arbetat med systemutveckling i det breda perspektivet, det vill säga med programmering, databashantering och allteftersom gått mot en mer analytisk del med förstudie, projektering och Use Casemodellering. Under 2,5 år har respondenten arbetat med Use Case. Just nu arbetar respondenten i ett projekt som är internationellt, ett väldigt stort projekt med 40 – 50 utvecklare runt om i världen.

Då respondent 4 representerar krav används två steg. Det ena steget är en processmodell som projektet gör utefter verksamhetens arbetssätt och önskemål för att se flödet, händelserna i dess verksamhet. Denna modell kommer sedan att översättas i det andra steget, Use Casemodellen. Use Case innehåller användarkrav, dels som Use Case med dess relationer till andra Use Case och aktörer, och även i ren text. Respondent 4 menar att de försöker så mycket som möjligt både använda modeller och text. Det är systemutvecklarnas roll vid krav, att transformera eller konvertera användarnas modeller och krav till förståligt språk som IT-människor förstår. Vid framtagandet av Use Casediagrammet är ofta användare med och hjälper, tillsammans med systemutvecklarna, till att utforma Use Casemodellen. Ofta kan användarna även läsa och förstå Use Case. Användarna är med och gör modellen tillsammans med systemutvecklarna, men gör det aldrig själva. Respondenten menar att det är vid Use Casemodellen som IT-människor och användare möts. Processmodellen ska användarna egentligen göra själva, men världen är inte perfekt vilket leder till att i de flest fall hjälper systemutvecklarna till med den. Modellerna är bra för att få användarna att blir aktiva i kravhanteringen. Användarna får vara med fram till och med Use Casemodellen, sedan tar systemutvecklarna över helt och hållet.

Respondenten har arbetat i projekt där Use Case beskrivs i olika nivåer, men helst inte mer än tre nivåer. Respondenten tycker detta är ett bra sätt. Olika intressenter som till exempel användaren, programmeraren och chefen, vill höra och se informationen på olika sätt, då är nivåerna bra.

Fördelar

Respondent 4 menar att Use Case är ett jättebra sätt att strukturera upp informationen på, att tillsammans med användaren rita och prata om hur saker har uppfattats. Respondenten menar att Use Case fungerar som kommunikation mellan systemutvecklare och användare. Åsikter från alla parter framkommer och det leder till bekräftad information eller förändringar. Use Casemodellen fungerar både som ett arbetssätt och dokumentation. Största fördelen med användning av Use Case tycker respondenten är att det är ett bra sätt att möta kunden på. Det är också en fördel att Use Casediagrammet är lätt för användarna att förstå. De får var med och påverka. Det är ett bra sätt att använda Use Casemodelleringen för att möta användarnas behov. Även kunden får större inblick i vad ett system kan prestera. Dessutom fångar Use Case behov och krav ur användarens synvinkel, ett bättre sätt att kommunicera i jämförelse med traditionella tekniker. Respondenten tycker också att det är lättare att verifiera användarfunktioner, att kontrollera att de funktioner utvecklarna vill ha med är med, så att inga funktioner missas. Ett sätt att i ett senare skede gå tillbaka och se

vad som är bra. Respondenten anser att Use Case är en bra dokumentationsteknik. Vid stora projekt delas arbetet upp, det görs Use Casediagram på olika ställen av olika personer i projektet. Use Casedokumentationen är lätt att förstå vilket medför att olika delar av projektet kan läsa varandras Use Case och förstå vad respektive utvecklare håller på med och vilka resultat som framkommer. Respondenten menar vidare att Use Case fungerar väldigt bra för vidare utveckling. Det vill säga att om användare inte är nöjda kan det härledas tillbaka till Use Caset ”det var faktiskt så här vi sa och beslutade en gång i tiden” – kallas spårbarhet. Att kunna gå tillbaka till specificerade krav. Där finns det faktiskt ganska bra verktygsstöd för att gå tillbaka verifiera vad system ska göra, underlag vid testning. Vid testning går systemutvecklarna tillbaka till Use Caset och kontrollerar vad var det tänkt att systemet skulle göra, stämmer det eller inte. Use Case fungerar som en verifikation på det som en gång sades.

Kvaliteten ökar på resultatet. Ett resultat av att användaren får vara med att dokumentera och göra modeller. Användarna är mer involverade än de var tidigare. Slutligen passar Use Case för att representera komplexa krav. Där det är väldigt svårt att i bara ord tala om hur det fungerar. Det är möjligt att se samband på ett annat sätt, på olika händelser.

Nackdelar

Respondenten tycker att Use Case inte bör användas vid väldigt små projekt och vid väldigt små utvecklingsuppdrag. Det känns onödigt att lägga ner allt det arbetet med strukturering och dokumentation för att komma fram till en ganska enkel lösning. Vid små arbeten har utvecklarna kontroll på vad som behöver göras, att det känns onödigt att använda Use Case som kräver mycket dokumentation och strukturering. Det tar lite mer tid att använda Use Case beroende på all dokumentation, bland annat ingår det att göra modeller, strukturera upp kraven och inblandning av användaren i utvecklingsarbetet tar mycket tid. Att det tar lite extra tid är en nackdel men det är något som tjänas in i längden, då resultatet förbättras i användning av Use Case.

Bör Use Case kompletteras?

Respondent 4 anser att det ibland kan behövas kompletterande tekniker. Use Case har funnits några år men är fortfarande en relativt ny teknik. Det saknas ibland lite verktyg. Det finns några tekniker som respondentens företag testar från företaget Rational, ett kravinsamlingsverktyg, dock saknar respondenten fortfarande mer verktyg för att strukturera och samla krav. Dessutom kompletteras systemutvecklingen med verktyg vid kravinsamling. Respondenten anser att processmodellen bättre beskriver det riktiga flödet. Use Case beskriver inte flödet på det sättet. Use Casemodellen kompletteras med ren text, vilket är ett annat sätt att beskriva Use Case på. Respondenten menar att Use Case inte måste innebära en IT-lösning.

Bilaga 6 – Intervju respondent 5

Respondent 6 har arbetat på företaget sedan 1989. Denna person började 1989 att arbeta med systemutveckling men sedan 1995 arbetar hon med metodutveckling. Det innebär ett arbete med de olika systemutvecklingsmetoder som används för att utveckla system. Arbete med systemutveckling kommer in i arbetet med metodutveckling. Bland annat har respondenten varit med i ett pilotprojekt 1999 där projektet skulle utvärdera huruvida RUP, där UML och Use Casemodellering ingick, kunde användas på företaget. Detta var första gången som respondenten kom i kontakt med Use Casemodellering. Innan dess gjorde företaget något som heter problemanalys och kravanalys och spårbarhetsanalys som också är en form av kravinsamling.

De projekt respondenten varit inblandad i har representerat krav med hjälp av både textUse Case och Use Casediagram. Utifrån aktörerna gör projektet en Use Casespecifikation. För varje Use Case beskriver utvecklarna i text flödet för varje Use Case i detalj. Detta är ett dokument som länkas i verktyget Rose till Use Casemodellen. De krav som inte kan relateras till ett specifikt Use Case utan gäller generellt för hela systemet kallas ickefunktionella krav. De ickefunktionella kraven läggs i ett dokument – supplementary.

Fördelar

Respondenten anser att i jämförelse med tidigare använda problemanalysen är det ganska lätt att lära en slutanvändare vad en aktör är och vad är ett Use Case. Tidigare skrev respondenten en problemanalys och övriga krav i långa listor. Materialet dokumenterades i ett excelark och blev inte lika överskådligt i jämförelse med Use Casediagrammet. Med Use Case blir dokumentationen mer visuellt, det är lättare att ta till sig och förstå. Det blir enklare för slutanvändaren. Det är diagrammet som systemutvecklarna visar mot slutanvändaren. Så när utvecklarna har gjort en lista på krav ställs motfrågor mot användarna, ”är det så här ni menar”, ”är detta rätt”. Projektet verifierar kraven mot kravlistan. Men det är lättare att ha Use Case att diskutera kring, för då går det att fokusera mot en speciell funktionell del av systemet i taget. En bild säger mer än text. Respondenten tror att det blir lätt för slutanvändaren att ta till sig Use Casedokumentationen. Vidare menar respondenten att Use Case är enkelt och går snabbt att lära ut. Både användare och utvecklare har lätt att ta till sig och förstår vad ett Use Case är.

Nackdelar

Respondenten kan inte finna några nackdelar med Use Case, denna tycker bara att det är ett annorlunda sätt. Förut tittade systemutvecklarna på vilka problem som skulle lösas och genom det samlades kraven in. Men nackdelar vet jag inte, bara ett annat sätt. Jag kan inte säga att det är en fördel eller nackdel mot det andra. Förut fokuserade utvecklarna på problem nu fokuseras det på funktioner hos systemet.

Bör Use Case kompletteras?

Respondenten anser att Use Case ibland bör kompletteras med vanliga textdokument.

Dessutom används användargränssnitt som en kompletterande teknik. De ickefunktionella kraven kompletteras med ett textdokument, ty det fungerar inte att beskriva dem i ett Use Case. Respondenten menar vidare att vilken standard som ska användas, dokumenteras i ett separat dokument. De krav som inte gäller för enskilda Use Case utan generellt för hela systemet de hamnar i ”Supplementary” Till exempel de ickefunktionella kraven, användargränssnitt och vilken standard som ska användas.

Hur fungerar Use Case vid testning?

Om utvecklarna använder användarspecifikationen och gör kravspecifikationen på detaljnivå och realiserar efter den och bygger efter den och dessutom får kunden att skriva på att det här är det som önskas. Då har man en överenskommelse på att det är så här det ska vara. Då kan kravspecifikationen användas till testfall.

Bilaga 7 – Intervju respondet 6

Respondent 6 har arbetat på företaget i 3.5 år och inom systemutveckling sedan –97. Respondenten har varit i kontakt med Use Case i 5 år. De use case företaget använder är UMLs definition på use case. Företaget arbetar efter RUP när de hjälper sina kunder och använde då även UML som modelleringsspråk.

När respondenten representerar krav sker det både i diagram och i dokument (textdokument). Respondenten anser att bara användning av diagram inte är tillräckligt som kravrepresentation. Det krävs dokument. Också behövs diagrammen, det skulle var fattigare utan diagrammen, men de klarar sig inte själva. Diagrammen ger en bra överblick men säger inte allt. Det går lika bra att göra modeller och figurer i powerpoint.

Fördelar:

Positivt med UML är att det blir sammanhängande, om man följer UML så blir det en enhetlig stil på det hela. Use case ger ett bra sammanhang. Det ger en bild av när kravet är relevant, vem som ska göra en funktion (med hjälp av aktörerna). När det ska utföras till exempel en funktion, use case talar om när funktionen skall fungera, vid detta tillfälle eller vid alla tillfällen. Till exempel skrivare vid denna utskrift eller alla utskrifter.

Use Case är användbart för andra än respondentens företag, speciellt för test och dokumentation. Men även för analyster och designers.

Det är lättare för alla i projektet att förstå. I jämförelse med traditionell dokumentation ger use case en bra förståelse för dokumentationen. Bra när man ska diskutera med kund. Beskriver bättre, är starkare, när aktörer är människor, use case fungerar bättre människa – maskin.

Ger en bra förståelse för systemet och hur det ska fungera

Nackdelar:

Beskriver sämre när aktören är ett system men fungerar dock, use case fungerar inte så bra maskin – maskin. Besvärligt med krav på olika nivåer. När det finns subsystem och subsubsystem. Det är svårt att bryta ner kraven.

Använder ni use case dokumentationen med användarna ?Nej inte då våra användare och kunder är långt ifrån oss. Då projekten är närmare och lite mindre kan vi göra det.

Har ni olika nivåer när ni representerar era krav?

Nej vi försöker hålla oss till en nivå, ibland två men det är absolut en gräns där, då får man lösa det på något annat sätt. Tre och fyra nivåer är alldeles för mycket, då blir det så mycket dokumentation så att det bara blir rörigt.

Man ska ej representera protokollspecifikationer med use case. Exempel ”så här ska man kommunicera i en databuss…” Det blir så oerhört mycket dokument, vem orkar läsa så mycket!

Bör man komplettera med något?

Ja. Det finns krav som inte går att representera med hjälp av use case. Och det finns tillfällen då use case känns onödig, den ger så mycket dokument att det blir ohållbart. Det är ingen som vill läsa igenom all den dokumentation som blir.

Istället för att skriva protokollspecifikationer med use case så använder vi oss av annan form av notation och dokumentation. Kompletterar även med separata dokument för de ickefunktionella kraven. Varför fungerar det inte att representera de ickefunktionella kraven med use case? Use case är av sin natur informell och sekventiell. Så fungerar inte de ickefunktionella kraven.

Tycke respondenten att use case fungerar bra vid testning?

Respondenten tycket det fungerar ganska bra.

Hur tycker respondenten use case fungerar vid spårbarhet?

Lite besvärlig när det gäller spårbarheten mellan use case och kod.

Då behöver man använda sig av sekvensdiagram där emellan.

Bilaga 8 – Intervju respondent 7

Respondenten har arbetat på IT-företaget i två år och har arbetat inom systemutveckling i nio år. Av dessa nio år har denna till och från under fem år arbetat med Use case vid systemutveckling. På företaget är finns det cirka 900 anställda. Respondenten arbetar som konsult på företaget, vilket gör att denna kommer i kontakt med olika kunder som ibland vill arbeta på ett speciellt sätt. Det bidrar till att ibland förekommer det vissa bestämmelser på hur kunderna har arbetat tidigare och hur de vill jobba nu. Den beskrivning som respondenten ger på hur denna vill beskskriva use case utgår ifrån hur denna själv anser att det ska beskrivas och dokumenteras. När företaget arbetar med use case är det med utgångspunkt från RUP och UML.

Som jag vill ha det så vill jag ha ett use casediagram som en översikt av vilka use case som har identifieras och vilka aktörer som är inblandade. Sen ett dokument som definierar aktörerna lite grann. I ett textdokument. Där de olika aktörerna definieras, för de kan ju ingå i flera use case. Ibland finns det verktyg som gör att man kan dokumentera text direkt i use casen, en kombination.

Sedan ska det finnas en textbeskrivning av varje use case enligt en liten mall:

*Rubrik ex ”Use case konfigurerar systemet”

*Inledande text som talar om vad usecaset är bra för

*Räkna upp de aktörer som är involverade

*Trevillkor – de villkor man vill ska vara uppfyllda innan man drar igång use caset. Ex Man måste logga på systemet, andra system måste starta först för att kunna kommunicera…

*Huvudflödet – use caset startar med att.. och så vad en aktör ska göra och avslutas med…

*postvillkor ”sånt som ska tala om i vilket skick use caset ska vara i när du är klar med use caset”

*alternativa flöden – om det är ett speciellt villkor som gör att man måste göra något annolunda ex om en kund ska göra en beställning och så finns inte kunden inlagd, då måste något annat ske innan beställningen kan göras. Detta ”annat” är det alternativa flödet.

*felflöden, de fel som kan inträffa hamnar som felflöden och då definieras det vad som ska hända när ett speciellt fel inträffar.

*Speciella krav hamnar längst ner i use caset. Om det finns några speciella krav eller villkor som gäller för use caset.

Use casen definierar och beskriver de funktionella kraven. De ickefunktionella kraven hamnar i ett dokument vid sidan av use casedokumentationen.

Fördelar

Respondenten menar att Use Case är bra för att beskriva interaktionen mellan en användare och systemet eller interaktionen mellan två system. Det här systemet skall kunna utföra det här och det här. Det ger en bra beskrivning på hur omvärlden vill använda systemet. När man har en omvärld som vill använda systemet på ett visst sätt.

Det spelar ingen roll om aktören är en människa eller en maskin, men det är då viktigt i det dokument som ovan nämnts att man definierar om aktören är en människa eller maskin. Vad som avses med vare aktör.

Dessutom fungerar Use Case bra vid testning. I och med att use case beskriver ett slags beteende har man ju ett slags normalfall att utgå ifrån ett normalfall som SKA fungera. Sedan finns det ju de orsaker runt omkring som kan ställa till det. Man får röd tråd att följa och en bas att utgå ifrån.

Nackdelar

När man bygga tekniska system (produktionssystem, styrsystem – styra processer, eller övervakningssystem) då ska man inte bara fokusera på use case för då behöver man även göra en ordentlig analys av systemet, där man modellerar upp systemet som en konceptuell bild av systemet. Olika moduler modelleras upp konceptuellt.

Use caset kan då fungera som ett komplement, hur ska man styra, de andra sakerna runt styrsystemet (de kan fungera som aktörer). Man kan utifrån analysen som man gjort sätta upp krav med hjälp av use case. Att man inte glömmer bort den logiska strukturen, use case som komplement, så att man får både den logiska strukturen och beteendet.

Det fungerar också sämre vid mindre system. Man ska inte överarbeta då det inte finns någon anledning till det. Är det ett mindre system som det går bra att specificera i ett vanligt dokument utan diagram så ska man inte överarbeta detta. Use Case behöver inte användas till varje pris. Det fungerar ju men det kostar dels administrativa timmar för att modellera och sedan blir det då ibland onödigt mycket dokumentation för lite arbete.

Behöver use case kompletteras?

Ja vid ickefunktionella krav, vid små system, vid tekniska system och om det finns ett interaktivt system (är det tekniskt system)

Ordlista behövs även för att ord och begrepp ska tolkas lika av olika personer. ”Vad menas med kund vad menas med omsättning”.

Kompletterar du?

Ja med just detta som nämndes ovan.

Det är också viktigt med krav att man referensmärker kraven att de har en referens eller ett nummer på varje krav, så att det ska gå att se om de är implementerade och vilka som är testade. Att varje use case får ett nummer och varje ickefunktionellt krav får också ett nummer.

Bilaga 9 - Intervju respondent 8

Respondent 8 har arbetat med systemutveckling på sin nuvarande arbetsplats i 8,5 år. Innan respondenten anställdes på företaget hade denna ytterligare 6,5 års erfarenhet inom systemutveckling. Av de 15 åren av erfarenhet inom systemutveckling har det sedan 1995 förekommit projekt med Use Case för att representera krav. Företaget beläget i Umeå har 155 anställda, men företaget förekommer på ytterligare två ställen i Sverige så totalt är det omkring 700 anställda.

På företaget där respondenten arbetar har Use Case används även innan Use Casemodellen enligt UML togs fram. Men under senare tid arbetar företaget med Use Case utifrån UML tolkning och definitioner. Respondenten anser att begreppet Use Case inte skiljer sig åt från det ursprungliga Jacobssons Use Case, men hur realiseringen av Use Case ska utföras har förändrats.

Respondenten använder Use Case i både text och diagram, men det är textformen som är den väsentliga delen i Use Casedokumentationen. Diagrammen används för att beskriva vilka Use Case som existerar och hur de förhåller sig till varandra. Sen kan även diagrammet användas för att beskriva hur ett Use Case är uppbyggt och då finns sekvensdiagrammet som är bra för att visa beteende, hur systemet beter sig i en viss situation. Men ofta går motsvarande information att göra i tabellform i ett vanligt textdokument. Respondenten själv föredrar att representera krav i textform.

Fördelar

Den största fördelen är att det går att hitta en nivå av tydlighet, där kravställaren och kravutföraren (beställaren och genomföraren) förstår varandra. En traditionell beskrivning av krav i text är så mångtydig att det väldigt svårt att tro att både den som ställer kraven och den som genomför kraven gör samma tolkning, vilket kan leda till missförstånd kring ett krav. Poängen med Use Case är att det är en form som är ganska tydlig. Just att utvecklare och användare kan blir ganska överens om vad kravet egentligen innebär. Respondenten anser att Use Case fungerar väldigt bra vid testning. Use case gör att det blir väldigt tydligt hur utvecklarna ska gå tillväga med testning. I princip ges ett testfall vid varje scenario i ett Use Case. Det finns tydliga relationer mellan Use Casemodellen och testmodellen. I ett Use Case ska utvecklarna uttrycka vad en extern aktör vill uppnå med systemet. Just den formen gör kravrepresentationen så tydlig, den beskriver exakt vilket beteende som en användare vill ha ut av systemet. Utgångspunkten är hela tiden användaren av systemet, vad denna vill uppnå och varför vill denna använda systemet. Genom Use Case ges en väldigt strikt uppdelning på funktionella krav och karaktäristikkrav (ickefunktionella krav). Use Case är väldigt funktionellt inriktat, hur systemet används. Use Case är bara lämplig då det finns ett tydlig händelseorienterat flöde där information tillförs vid ett visst tillfälle och så skickar systemet informationen vid ett speciellt tillfälle. Respondent 8 anser att Use Case är ganska planeringsbara. När ett system ska byggas definieras en mängd Use Case som ska implementeras, men alla Use Case behöver inte implementeras samtidigt. Respondenten samlar ihop Use Case i paket och kan därefter implementera kraven paketvis. I projekt finns alltid en massa krav som ännu inte är gjorda, dessa lagras i en buffert med Use Case som efterhand implementeras. Det här fungerar inte på ett karaktäristikkrav. I exemplen i böcker beskrivs nästan alltid människa-maskin interaktioner (aktören är en människa). Det är

ganska enkelt att förstå ett Use Case när aktören är en människa. Maskin-maskin interaktioner då aktören är en maskin är inte lika enkelt riktigt att förstå hur modelleringen ska gå till.

Bilaga 10 – Intervju respondent 9

Respondent 9 har arbetat på företaget i 6år. Denna har sedan 1973 arbetat med systemutveckling varav de senaste två åren till och från med Use Case. Totalt har respondenten arbetat väldigt aktivt med Use Case i ett år. Respondenten fungerar för närvarande som projektledare. Men respondenten går också in i och gör verksamhetsanalys inför förstudien, vilket innebär att arbeta fram kravspecifikationen och modelleringsarbete. Sammanfattningsvis är respondenten projektledare med mer administrativ roll och sedan modellering arbetssättet man styr för att ta fram en kravspecifikation och ta fram användningsfall. Respondenten använder sig till största delen av Use Case i textform för att representera krav. I något projekt där respondenten varit projektledare har projektet använt Use Casediagram, men huvudsakligen beskrivs Use Case i textform.

Fördelar

Respondenten anser att en fördel med Use Case är att det går att bena upp, det vill säga tala om vilka Use Case projektet prioriterar, i vilken ordning Use Casen ska behandlas. Vidare menar denna att en annan fördel är att utvecklarna ska kunna peka ut de kritiska Use Casen, så att de svåra bitarna behandlas först. Use Case är alltså viktigt ur risksynpunkt och inkrementplanering.

FRÅGA: Vad är inkrementplanering, inkrement?

Systemutvecklarna grupperar Use Case som därefter implementeras i olika omgångar. Först kanske det finns tre Use Case ”skapa order”, ”ta bort order” och ”ändra order” dessa grupperas och bildar ett inkrement. Så här använder utvecklarna inkrementen och visar dem för kunden, ”så här fungerar de här Use Casen” Vid nästa inkrement har man kanske sex nya användningsfall, det hela handlar om att utvecklarna jobbar i inkrement, först designa och sen så implementeras Use Case och därefter visar utvecklarna det hela för kunden och så går man vidare med nästa sväng. Respondenten menar att detta gör att som projektledare går det att planera resurserna inom projektet. Först väljs de inkrement ut som är mest kritiska, ofta hänger det ihop med vad kunden vill ha, det kan hända att kunden säger att ”vi behöver de här bitarna nu eller behöver allt på en gång”. Respondenten tycker att Use Case fungerar bra.vid testning. För systemutvecklarna delar upp vilka Use Case som har det första inkrementet och då går det att göra testfallen utifrån det. Systemutvecklarna tittar både på användningsfallen och användargränssnittet för att få fram bra testfall. Användningsfallen är ett bra komplement till testningen.

Nackdelar

Respondenten tycker att när systemutvecklarna bygger upp en kravspecifikation är inte enbart Use Case lämpligt att användas. Bland annat behöver de ickefunktionella kraven kompletteras och användargränssnittet bör kompletteras för att visa hur systemet interagerar med användaren och vilka regler som finns däremellan. Glömmer utvecklarna det och bara bygger kravspecifikationen utifrån användningsfall, då blir kravspecifikationen tunn, den blir inte hållbar. Bland annat skulle det var väldigt svårt att göra en tidsuppskattning och kunna bedöma hur lång tid

det skulle ta att implementera. Komplement till Use Case är också sekvensdiagram. Respondenten menar också att Use Case är en nackdel då en projektledare ska fördela ut arbetet till ett antal utvecklare. Om det existerar tvåskiktslager eller treskiktslager där några jobbar på serversidan och några jobbar på klientsidan är det svårt att fördela ut arbetet. Det finns ett gränssnitt, ta exempelvis ”uppdatera kund”, då går det mellan serversidan och tillbaka till klientdelen där det presenteras. Och gör man det att man ber att utveckaren ska köra det i ett stuprör att man utvecklar både på klient och serversidan och några andra utvecklare också utvecklar något annat användningsfall både på klient och serversidan, så blir det inte någon bra lösning på serversidan. Det är bättre att någon utvecklare sköter utvecklingen på serversidan. Nackdelen med att dela upp utvecklare som sköter endast klient och endast serversidan är att de som sköter utveckling av klient kanske blir snabbare klara än serversidan och behöver kanske något från serversidan, det tar då lite tid innan dessa två sidor jackar ihop med varandra och då ser man om det här inkrementet håller. Man måste tänka sig för, att bena upp vem som gör vad – detta kan bero på om det är stora eller små projekt. Är det stora projekt ska man ha ansvariga på klientsidan respektive serversidan. Är det små projekt kan man gör allt tillsammans, snabbdialoger mellan varandra. Det beror alltså på storlek på projektet och hur man ska implementera och antal delsystem. Nackdelen är att stirra sig blind på användningsfall stuprörs. Att en utvecklare arbetar på både klient och serversidan, det blir inte en bra lösning. Respondenten menar också att Use Case inte är något revolutionerande inom systemutvecklingen. Utan det är många andra bitar måste finnas på plats. Användargränssnittet ska vara beskrivet på ett bra sätt så att de som ska utveckla förstår det. Och objektmodellen och en arkitektur är med. Respondenten menar att det kan vara bra att ha med ett användningsfall och gå igenom det med användarna, men användarna förstår inte en kravspecifikation med bara användningsfallen. En användare har svårt att förstå användningsfallen. Visas däremot ett användargränssnitt och berättar vad som kommer upp och hur man går vidare förstår kunden bättre.

Behöver Use Case kompletteras?

Respondenten menar att Use Case absolut måste kompletteras. Ovan har respondenten redan nämnt de flesta tekniker som denna anser Use Case bör kompletteras med. Bland annat bör användargränssnitt, arkitekturmönster, sekvensdiagram, objektmodell och de ickefunktionella kraven kompletteras.