Institutionen för kommunikation och information
Examensarbete i kognitionsvetenskap, 20p
C-nivå
Vårterminen 2007

En studie om användbarhetskrav:

hur valet av insamlingsteknik kan påverka identifieringen av olika aspekter av användbarhet.

Marcus Johansson

En studie om användbarhetskrav: hur valet av insamlingsteknik kan påverka identifieringen av olika aspekt av användbarhet.

Examensrapport inlämnad av Marcus Johansson till Högskolan i Skövde, för Kandidatexamen (B.Sc.) vid Institutionen för kommunikation och information. Arbetet har handletts av Anna-Sofia Alklind Taylor.

2007-06-02

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: _____

En studie om användbarhetskrav: hur valet av insamlingsteknik kan påverka identifieringen av olika aspekter av användbarhet.

Marcus Johansson

Sammanfattning

Sociotekniska system, är system där användare ingår i processen för att uppnå någon bredare typ av mål och ställer därför krav på användbarheten av ett system. Av mjukvaruprocessen är mjukvaruspecifikationen den process som är till för att specificera vilka krav systemet ska innehålla. Användbarhetskrav finns i olika former för att passa olika aspekter av användbarhet. Att arbeta med användbarhetskrav har visat sig vara en bra metod att uppnå användbarhet. Detta arbete ska undersöka om huruvida vissa insamlingstekniker kan fördelaktigt användas för att identifiera olika aspekter av användbarhet. Frågeställningen besvarades genom att utföra en genomgång av vetenskaplig litteratur och intervjuer av utvecklare om deras erfarenheter att arbetet med användbarhetskrav. Resultatet visade bland annat på att utvecklare väljer insamlingsteknik efter vilken erfarenhet denne har med att arbeta med den och vad som är möjligt att utför inom projektets ramar med avseende på tid, pengar och tillgång till användare. Vidare uppfattades även observation som en av de bättre insamlingsteknikerna att identifiera användbarhetskrav.

Nyckelord: Mjukvaruutveckling, användbarhetskrav, användbarhet, insamlingstekniker

Innehållsförteckning

1 Introduktion

Sociotekniska system är en kategori av system där användare ingår för att uppnå något bredare typ av mål (Sommerville, 2007). Typen av system kräver ett stort fokus på användbarheten, där det är viktigt att användaren har varit utgångspunkten för utvecklingen. Problemet med mjukvaruutveckling är att den ofta utgår från ett teknikorienterat synsätt. Användbarhet förutsätts vara någonting som är inbyggt, men det är få utvecklare som explicit arbetar emot den (Gulliksen & Göransson, 2002).

Processen att ta fram system kallas för mjukvaruprocessen, vilket är en process som finns i olika former beroende på vilken typ av system som utvecklas. Gemensamt för alla mjukvaruprocesser är däremot att de innehåller fyra kärnaktiviteter: mjukvaruspecifikation, mjukvarudesign och implementering, validering och mjukvaruevolution. Av dessa är mjukvaruspecifikationen den aktivitet som är ämnad åt att specificera vad systemet ska göra och vad det ska innehålla. Detta gör den till en kritisk aktivitet då det är den som kommer att sätta premisserna för hur utveckling av ett nytt system kommer att fortskrida i de andra delarna. Innehållet mjukvaruspecifikation brukar oftast sammanfattas i krav, mest dominerande är funktionella och ickefunktionella krav. Funktionella krav ska behandla de funktioner som en mjukvara ska leverera och ickefunktionella krav ska behandla de krav system ska uppvisa som helhet (Sommerville, 2007).

Användbarhetskrav klassificeras som ett ickefunktionellt krav och kan formuleras på olika sätt beroende på vilka aspekter av användbarhet ett system ska uppnå. Enligt Preece, Rogers och Sharp (2002) kan användbarhetskrav användas för att både driva och spåra utveckling av användbarheten av en produkt. Men för att uppnå ett användbart system kräver det av utvecklare att välja rätt användbarhetskrav för att arbetat mot rätt aspekter av användbarhet. För att guida utvecklaren i valet av användbarhetskrav har därför Lauesen (2005) skapat en sammanställning över vilka typer av användbarhetskrav som passar till vilka aspekter av användbarhet.

Problemet med att identifiera krav är däremot att en utvecklare inte bara kan fråga en användare om vad denne vill se i ett nytt system och få all den information den behöver. En användare är kanske inte till exempel medveten om ett krav eller så vet denne inte om att det är möjligt att utföra. För att överkomma detta måste utvecklare utföra olika insamlingstekniker för att fånga hela kravbilden. Att utföra olika insamlingstekniker är däremot fortfarande väldigt teknik- och funktionsinriktat. Lauesen och Younessi (1998) har emellertid identifierat ett antal tekniker som lämpar sig till att finna användbarhetskrav.

Detta arbete kommer att undersöka utvecklares erfarenheter om att identifiera användbarhetskrav till sociotekniska system. Syftet är att ta reda på om det finns någon koppling mellan aspekt av användbarhet och insamlingstekniker. Arbetet kommer att utföras baserat på genomgång av vetenskaplig litteratur och intervjuer av utvecklare.

I kapitel 2 kommer en genomgång av de grundläggande begrepp och vetenskapliga fakta som kommer att ligga till grund för problemområdet till rapporten. Problemområde och problemprecisering kommer att presenteras i kapitel 3. Kapitel 4 kommer att ta upp den metod som utfördes till arbetet, kapitlet kommer även att beröra tillvägagångssättet av arbetet och ta upp en presentation av de respondenter som deltog i studien. Kapitel 5 kommer att behandla resultatet av de intervjuer som

utfördes presenterat i fyra ämnesområden och slutsatsen till arbetet. Slutligen kommer kapitel 6 ta upp diskussioner kring tillvägagångssätt, resultat och fortsatta studier.

2 Bakgrund

Denna del av arbetet är avsedd att behandla bakgrunden till studien. Bakgrunden kommer att vara uppdelad i två delar. Den första ger en snabb överblick över sociotekniska system och mjukvaruutvecklingens grunder med fokus på kravhantering och användbarhetskrav. Del två av kapitlet kommer att ta upp hur processen går till att utvinna krav och vilka problem en utvecklare kan tänkas ställas inför.

2.1 Sociotekniska system

En enkel uppdelning av system kan göras i två delar, tekniska datorbaserade system och sociotekniska system och detta arbete riktar in sig på den senare av dessa två. Vad som kännetecknar ett sociotekniskt system är att de byggs för att uppnå en något bredare typ av mål och att det ingår användare i processen (Sommerville, 2007). Sociotekniska system kräver därför ett större fokus på användbarhet i mjukvaruutveckling. Problemet med att ta fram system idag är däremot att det är ett alldeles för stort teknikfokus och system utvecklas ofta för att testa någon ny teknik (Gulliksen & Göransson, 2002). Det dominerande teknikfokuset i mjukvaruutveckling har säkert flera anledningar och en av dem är att kompetensen om användbarhet brister bland utvecklare. I en undersökning av Mao, Vredenburg, Smith och Carey (2001, i Gulliksen & Göransson, 2002) visade det sig att användbarhetskompetensen är ofta väldigt centraliserad till en liten del av utvecklarna. Undersökningen visade däremot att användbarhet är något som börjar få ett allt större fotfäste i industrier och att det är en utveckling som sprider sig. Nästa del av bakgrunden kommer att fördjupa sig i mjukvaruutvecklingens grunder med fokus på kravetablering.

2.2 Mjukvaruutveckling

Enligt Sommerville (2007) är mjukvaruutveckling den disciplin som avser produktion av mjukvara. Kunskapsområdet är en del av systemutveckling som sätter in det i ett större perspektiv i produktion av hela system. Som vetenskapligt området skulle det kunna definieras i två punkter (Sommerville, 2007):

- *Ingenjörsdisciplin* Ingenjörer skapar saker genom att applicera teorier, metoder och verktyg för att lösa problem. Området kräver även uppfinningsrikedom för de problem som måste lösas där teorier, metoder eller verktyg inte kan brukas.
- *Alla aspekter av mjukvaruproduktion* Mjukvaruutveckling begränsas inte endast till den tekniska processen av att utveckla en mjukvara utan sträcker sig även till aktiviteter som projektstyrning, utveckling av verktyg, metoder och teorier som stödjer mjukvaruproduktionen.

2.3 Mjukvaruprocess

En mjukvaruprocess är den process som leder till produktionen av en mjukvara. Dessa processer kan vara komplexa och invecklade då de baseras på en kreativ process där mänskligt beslutsfattande ligger som grund. Det finns ingen ideal process som kan täcka alla behov av mjukvaruutveckling och olika mjukvarusystem kommer att kräva olika typer av processer. Till exempel kommer ett säkerhetskritiskt system kräva strukturerade processer som täcker alla aspekter av mjukvaruutveckling, medan affärssystem kräver processer som är mer flexibla för att tåla snabbt förändrande krav.

Gemensamt för alla processer är att de innehåller vissa fundamentala aktiviteter (Sommerville, 2007).

1. *Mjukvaruspecifikation* Ska definiera funktionaliteten och begränsningarna av funktionerna av mjukvaran.
2. *Mjukvarudesign och implementation* Utvecklingen av specifikationen i form av producerad mjukvara.
3. *Mjukvaruvalidering* Mjukvaran måste valideras för att kontrolleras att det når upp till kundernas krav.
4. *Mjukvaruevolution* Mjukvaran måste utvecklas genom evolution för att möta kundernas förändrande krav.

2.3.1 Mjukvaruspecifikation

Som tidigare nämndes behandlar mjukvaruspecifikationen den del av processen som är ämnad att förstå och definiera vad som ska erbjudas av mjukvaran. Detta gör den till en kritisk del av mjukvaruutvecklingsprocessen, då fel som uppkommer i mjukvaruspecifikationen och sprider sig till andra delar av processen kommer även att stiga i kostnader att reparera (Sommerville, 2007). Precis som processer tar olika form hos olika företag, tar även mjukvaruspecifikationen olika form i olika företag. Organisationer som utvecklar mjukvara till rymdfarkoster kommer inte att använda samma processmodeller som företag som skapar mjukvaruprodukter till persondatorer. Gemensamt för alla mjukvaruspecifikationsprocesser är att de ändå på en abstrakt nivå innehåller samma aktiviteter (Kotonya & Sommerville, 1998):

1. *Kravframställning* Mjukvarans krav utforskas och upptäcks genom konsolidering av intressenter, mjukvarudokument, domänkunskap och marknadsundersökningar.
2. *Kravanalys och förhandling* Kraven analyseras i detalj och olika intressenter förhandlar om vilka krav som ska accepteras. Detta är nödvändigt för att det kommer att uppstå konflikter mellan krav på grund av ofullständig information eller att kraven inte överensstämmande med budgeten. Inom mjukvaruspecifikationsaktiviteten finns det alltid en flexibilitet som möjliggör förhandling om vilka krav som ska accepteras.
3. *Kravdokumentationen* De överenskomna kraven dokumenteras i en kravspecifikation på lämplig detaljnivå. Kravspecifikationen ska göra sig förstådd av samtliga intressenter för mjukvaran.
4. *Kravvalidering* Det måste genomföras en kontroll av de dokumenterade kraven för konsistens och fullständighet. Denna process är till för att upptäcka problem med kravspecifikationen innan den lämnas vidare för design och implementering.

2.3.2 Innehållet i en mjukvaruspecifikation

Tidigare har innehållet i en mjukvaruspecifikation generellt behandlat input till systemet, output systemet ska producera och möjligtvis operationstider för producerad output (Lauesen, 2002). Allteftersom behoven av alltmer komplexa system växt fram har även innehållet av mjukvaruspecifikationerna växt till nya proportioner. Lauesen (2002) har identifierat sju delar en mjukvaruspecifikation bör innehålla:

1. *Datakrav* Behandlar vad för format input och output ska vara för systemet. Datakraven bör även besvara frågor som vad för data som ska lagras internt. Utöver detta ska även datakraven dokumentera vad för tillstånd systemet kommer att finna sig i.
2. *Funktionella krav* Specificerar de funktioner som systemet ska erbjuda och även dokumentera hur data ska beräknas, transformeras, sparas och skickas vidare.
3. *Ickefunktionella krav* Dokumenterar hur en mjukvara ska utföra sina funktionella krav. Enligt Lauesen (2002) innebär det att dokumentera krav som berör utförande av funktioner, användbarhet och underhåll. En utförligare beskrivning om ickefunktionella krav av Sommerville (2007) finns i kapitel 2.3.3.
4. *Andra "leverabler"* Utöver mjukvaran som levereras kommer även annan dokumentation krävas. Exempel på sådan information är de tjänster som tillkommer med systemet, till exempel vem som gör vad i att installera mjukvaran, träna användare och sköta systemet.
5. *Styrningskrav* Befinner sig mellan kravdokumentering och kontraktsfrågor. Denna del av mjukvaruspecifikationen kommer att behandla frågor som: när mjukvaran ska levereras, pris och när betalning ska ske, vem som äger mjukvaran, hur validering av mjukvarn ska ske och så vidare. Förutom detta kan även styrningskrav behandla hur mjukvaruprocessen ska skötas.
6. *Stödinformation* För att undvika en lång lista med endast krav i mjukvauspecifikationen rekommenderas det att ta med bakgrundsinformation till kraven. Denna information kan till exempel behandla vilken typ av problem kravet löser eller för vilken arbetsprocess kravet är viktigt. Annan typ av stödinformation som är bra att tillägga är en ordlista för domänspecifika begrepp och affärsmål som definierar vad organisationen arbetar åt.
7. *Organiserande delar* Denna del av mjukvaruspecifikationen behandlar inte så mycket en specifik del utan snarare en struktur den bör upprätta. Det finns många sätt att strukturera upp en mjukvaruspecifikation och den mest kända är att följa IEEE 830 standard om mjukvaruspecifikationer.

2.3.3 Ickefunktionella krav

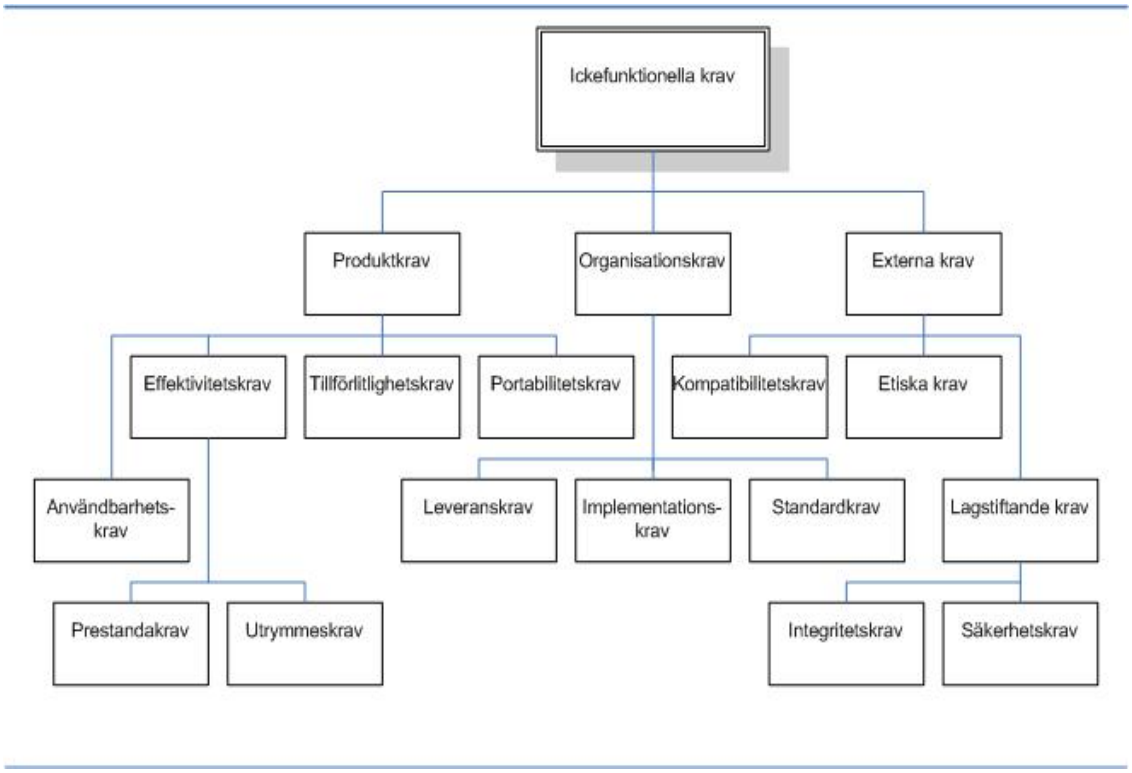
Ickefunktionella krav ska som tidigare nämndes specificera hur väl mjukvaran ska utföra sina funktioner (Lauesen, 2002). Sommerville (2007) definierar dem som krav som inte berör någon specifik funktion utav systemet. Dessa krav är alltså sällan associerade med någon individuell del av en mjukvara, utan specificerar snarare emergenta egenskaper och begränsningar systemet måste innehålla. Vad som menas med emergenta egenskaper i detta sammanhang är de egenskaper som systemet uppvisar som sin helhet, det vill säga egenskaperna som uppkommer när delarna av ett system sätts ihop för att utgöra en komplett mjukvara. Ickefunktionella krav kommer därför att beröra delar som säkerhet, användbarhet, tillförlitlighet och så vidare i ett system (se figur 1)(Sommerville, 2007). Detta innebär också att de kommer att spela en mycket mer kritisk roll i systemen än de funktionella kraven. En användare kan till exempel arbeta runt saknade systemfunktioner, men en mjukvara som inte når upp till ickefunktionella krav kan leda till att systemet blir helt obrukbart. För att förtydliga detta kan ett exempel vara tillhands: ett flygledarsystem som inte

når upp till krav på tillförlitlighet kommer inte klassificeras som säkert att användas (Sommerville, 2007).

Något som däremot saknas är en klar definition av vad som kan klassificeras som ett ickefunktionellt krav (Lauesen, 2002). Sommervilles (2007) definition av ickefunktionella krav skiljer sig från Macall och Matusumoto (1980, i Lauesen, 2002), som skiljer sig ifrån ISO 9126 (Lauesen, 2002). Lauesen (2002) förespråkar därför att använda de olika definitionerna som checklistor över vad som bör ses över i utvecklandet av en mjukvara. Sommervilles (2007) uppdelning av ickefunktionella krav sker i tre huvudkategorier som är uppdelade efter vilka källor de härstammar från, där följande uppdelning används (se figur 1 för utförligare uppdelning):

- *Produktkrav* Specificerar produktens beteenden. Exempel på produktkrav är: användbarhetskrav, tillförlitlighetskrav och portabilitetskrav.
- *Organisationskrav* Avser policys och procedurer från kund och utvecklare. Exempel på organisationskrav är: leveranskrav, implementationskrav och standardkrav
- *Externa krav* Denna kategori behandlar alla krav som härstammar från externa källor. Exempel på externa krav är: kompatibilitetskrav, lagstiftandekrav och etiska krav.

Ett vanligt problem med ickefunktionella krav är att de är svåra att verifiera. Detta beror på att användare och kunder ofta uttrycker dem i mål istället för krav, vilket lämnar utvecklarna i situationer där tolkningar och konflikter uppstår. En del av en systemutvecklares uppgifter är därför att omvandla mål till mätbara krav. Exempel på ett krav som är utformat som ett mål skulle kunna vara: ”mjukvaran ska vara designad så att användaren gör så få fel som möjligt” (Sommerville, 2007). Ett bättre sätt att uttrycka detta krav på och göra det mätbart skulle vara: ”en expertanvändare bör efter två timmars träning med mjukvarans funktioner använda systemets funktioner med ett genomsnitt högst två fel om dagen”. Denna omformulering gör att kravet blir mätbart och kan i sin tur valideras. Alla ickefunktionella krav kan däremot inte omvandlas till mätbara krav, till exempel är krav som berör underhåll av systemet svåra att utforma mätbara. Det kan därför vara bra att kombinera krav med mål i mjukvaruspecifikationen för att guida utvecklarna. I dessa situationer bör däremot kunderna informeras då det finns risker för feltolkningar med att inkludera mål i mjukvaruspecifikationen. Förutom svårigheterna med att specificera ickefunktionella krav dyker det även upp problem med att funktionella och ickefunktionella krav interagerar eller hamnar i konflikt med varandra. En viktig del i mjukvaruspecifikationsprocessen är därför att kompromissa och förhandla mellan olika typer av krav (Sommerville, 2007).



Figur 1. Uppdelning av ickefunktionella krav (inspirerad av Sommerville, 2007, s.122).

2.3.4 Användbarhetskrav

Enligt Lauesen (2002) består användbarhet av två faktorer: enkelhet i användandet och anpassad för användning. Anpassad för användning innebär att en mjukvara ska stödja de uppgifter användaren är tänkt att utföra i verkligheten och från ett kravperspektiv berör denna aspekt vilka funktionella krav systemet ska ha att göra med. Enkelhet i användandet är de aspekter som mer blivit associerat med användbarhet och är även de krav som kategoriseras som användbarhetskrav. Det finns många sätt att specificera användbarhet (enkelhet i användandet), men till detta arbete kommer det att specificeras utifrån Lauesen (2002) och bestå av följande faktorer:

- *Lätthet att lära sig* Mjukvaran måste vara enkelt att lära sig både för novisa användare som användare som har vana från andra liknande system.
- *Uppgiftseffektivitet* Mjukvaran måste vara effektivt att använda för frekventa användare.
- *Lätthet att komma ihåg* Mjukvaran måste vara enkel att komma ihåg för den vanliga användaren.
- *Förståelighet* Användaren måste förstå vad mjukvaran gör.
- *Subjektiv tillfredsställelse* Användaren måste vara nöjd med att använda systemet.

Det finns ingen enhetlig bild över vad användbarhetskravens funktion och bidrag är till mjukvaruutveckling; åsikterna varierar från onödiga till nödvändiga. Enligt Sommerville och Kotonya (1998) är processen att specificera användbarhetskrav inte lika effektiv som att arbeta med evolutionär prototyping. Evolutionär prototyping är den process där en funktionell prototyp skapas snabbt åt kunden och kommer sedan att modifieras under utvecklingens gång för att till slut bli ett fungerande system.

Preece, Rogers och Sharp (2002) tar ett annat perspektiv på synen av användbarhetskrav, där de ser dem som en viktig del av mjukvaruutvecklingsprocessen. Kraven ska behandla de användbarhetsmål och associerade mätvärden som är kopplade till en produkt. Preece med kollegor (2002) förespråkar ett användarcentrerat tillvägagångssätt, vilket innebär att tidigt dokumentera användbarhetskrav och använda dessa för att driva och spåra utvecklingen av en produkt. Detta möjliggör att användbarheten av en produkt kommer att få en högre prioritet och även förser processen med en spårbarhet av utvecklingen. Även Lauesen (2005) förespråkar en användarcentrerad process i kombination med användbarhetskrav, däremot med ett större fokus på verifiering, utvärdering, av ett gränssnitt. Enligt Lauesen (2005) fungerar användbarhetskrav som mätbara kriterier som kan verifiera att en design kommer att fungera mot slutanvändarna. Detta representerar tre synsätt på användbarhetskrav: den första tar ett mer systeminriktat perspektiv där användbarhetskrav har ett allt mindre fokus i mjukvaruprocessen; det andra och tredje representerar en MDI-inriktning (människa-datorinteraktion inriktning) på användbarhetskrav, där de främst är till för att öka uppmärksamheten på designen av gränssnittet. Denna uppdelning av användbarhetskrav är däremot inte helt främmande inom mjukvaruutveckling. Enligt Cysneiros, do Prado Leite och de Melano Sabat Neto (2001) beror detta på att de i vanliga systemutvecklingsprojekt först behandlas vid designfasen av mjukvaran. Konsekvenserna av detta är att de får en allt mindre central roll inom systemutvecklingsprojektet och att mjukvaran inte kommer att nå upp till de krav som finns kopplade till användbarhet.

2.3.5 Typer av användbarhetskrav

Att dokumentera användbarhetskrav för en produkt eller system är ingen enkel väg att gå. Beroende på vad för mål produkten ska möta kan kraven specificeras på olika sätt. Lauesen (2005) har identifierat sex olika stilar för att dokumentera användbarhetskrav:

Prestationskrav Denna typ av krav mäter användarens förmåga att utföra uppgifter inom ett tidsintervall. Exempel på ett prestationskrav skulle kunna vara: ”Användare med erfarenhet från andra bankomatsystem ska i deras första försök kunna göra ett uttag av ett bestämt belopp inom i genomsnitt två minuter”. Denna typ av krav är beroende av tre typer av data: lista över kritiska uppgifter för mjukvara, lista över användare till systemet och erfarenhet och prestationsdata för varje uppgifts- och användarkombination. Fördelen med prestationskrav är att de kan mäta nästan alla aspekter av användbarhetskrav med undantag av förståelseaspekten som endast kan mätas indirekt och subjektiv tillfredsställelse. Det största problemet med prestationskrav är däremot att de inte kan berätta för utvecklaren vad ursprunget till problemet är. En utvecklare kan observera att användare tar lång tid på sig att lära sig systemet, men inte vad som är det underliggande problemet till det.

Defektkrav Defektkrav är en variant av prestationskrav som sätter gränser för antal fel och hur allvarliga fel en användare får göra inom utförandet av en uppgift. Exempel på ett defektkrav är: ”Användare med erfarenhet från andra bankomatsystem får i genomsnitt högst utföra 0,6 fel i att göra ett uttag av ett förbestämt belopp”. Gemensamt för prestationskrav och defektkrav är att de kräver samma typ av data för att bestämma vad användarna ska utföra och vad för krav de ska uppnå, det är även valet av användare och uppgifter som är kritiska för resultatet av kravet. Fördelen med kravtypen är att de kan testas tidigt i ett mjukvaruprojekt. Det främsta problemet är att den inte täcker alla aspekter av användbarhet, utan kommer att endast indirekt kunna

mäta uppgiftseffektivitet, subjektiv tillfredställelse och förståelseaspekten. Detta beror på tvetydligheten i resultatet, en användare som till exempel inte stöter på några problem när denne utför uppgifter behöver inte vara nöjd med systemet.

Subjektiva krav Denna typ av krav definierar kriterier för tillfredsställelse med att använda systemet. Exempel på ett subjektivt krav är: ”80 % av användarna till bankomatsystemet som har prövat det åtminstone en gång måste finna det behagligt och hjälpfullt att använda, 60 % av användarna måste rekommendera det till sina vänner vid efterfrågning”. För att skapa subjektiva användbarhetskrav krävs det att hitta rätt subjektiva faktorer och rätt procenttal av tillfredsställda användare. Jämfört med prestationskrav och defektkrav beror subjektiva krav inte på att identifiera rätt uppgifter, utan täcker alla situationer en användare kan befinna sig i. Kravtypen täcker även in flest aspekter av användbarhet, även om de olika aspekterna inte kan mätas objektivt. Nackdelar med subjektiva krav är att de är svåra att använda under mjukvaruutvecklingsprocessen och att det är svårt att identifiera varför användaren inte är nöjd med systemet. Det mest allvarliga problemet är däremot att kravtypen inte säger så mycket om användargränssnittet är bra eller dåligt. En användare kan till exempel prestera dåligt, men fortfarande vara nöjd med systemet. En utvecklare kan därför inte ensamt förlita sig på subjektiva krav för att uppnå användbarhet.

Förståelsekrav För att mäta hur väl en användare förstår ett system kan en utvecklare enkelt fråga användaren om hur de tror systemet fungerar. Frågor om till exempel systemmeddelanden eller systemtillstånd ger utvecklare en uppfattning om användarens kunskaper om systemet. Förståelighetskrav tar vara på detta genom att ställa krav på användarens kunskaper om systemet. Exempel på ett förståelsekrav är: ”80% av svaren måste vara betygsatta med 4:a eller 5:a”. Kravets formulering medför att en utvecklare måste betygssätta (1-5) en användares svar för att kunna mäta kravtypen mot systemet. Fördelen med förståelsekrav är att de kan erbjuda information som är svår att finna med prestationskrav eller defektkrav och att de kan med enkelhet identifiera användbarhetsproblem med systemmeddelanden. Kravtypen kan även testas tidigt i mjukvaruprojekt vilket ger utvecklare information om problem som kan lösas tidigt. Nackdelen med kravtypen är att den endast mäter en aspekt av användbarhet, förståelseaspekten.

Tangenttryckningskrav Genom att dokumentera krav för antal knapptryckningar kan utvecklare få en uppfattning om uppgiftseffektiviteten för vana användare. Knapptryckningskrav dokumenteras genom att välja ut kritiska uppgifter och räkna antal tangenttryck och musklick en användare måste utföra för att avsluta uppgiften. Kravet dokumenteras sedan för att specificera maximalt antal knapptryckningar en användare får utföra. Exempel på ett tangentryckningskrav är: ”Användaren får maximalt genomföra sex tangenttryckningar för att utföra uppgiften”. Tangenttryckningskrav bidrar även med att utvecklare kan beräkna tiden för att utföra en uppgift. Genom att addera tiden det tar för varje tangenttryckning med tiden för systemresponsen får utvecklarna en uppfattning om detta. Denna typ information går däremot inte att använda utan komplikationer, problemet är var uppgiften börjar och slutar. För utvecklaren är det enkelt att specificera denna som när användaren börjar trycka på tangenterna, men för användaren kan det vara när denne tar del av den information som finns presenterad på användargränssnittet. Olika situationer kommer även att göra dessa gränser ännu otydligare, till exempel kommer inte ett bankomatsystem fungera på samma sätt som ett applikationsprogram i en kontorsmiljö. Fördelen med knapptryckningskrav är att den enkelt kan specificera uppgiftseffektiviteten för en mjukvara och att den kan utföras tidigt i utvecklingen och till och med utan användare. Nackdelar är att en utvecklare aldrig kan vara säker om

användaren kommer att utföra uppgiften på det mest effektiva sättet och att använda kravtypen riskerar även vilseleda utvecklare om vad egentligen uppgiften handlar om. Vanligtvis består en uppgift mycket mer än endast att trycka på några knappar, vilket förespråkas av knapptryckningskrav.

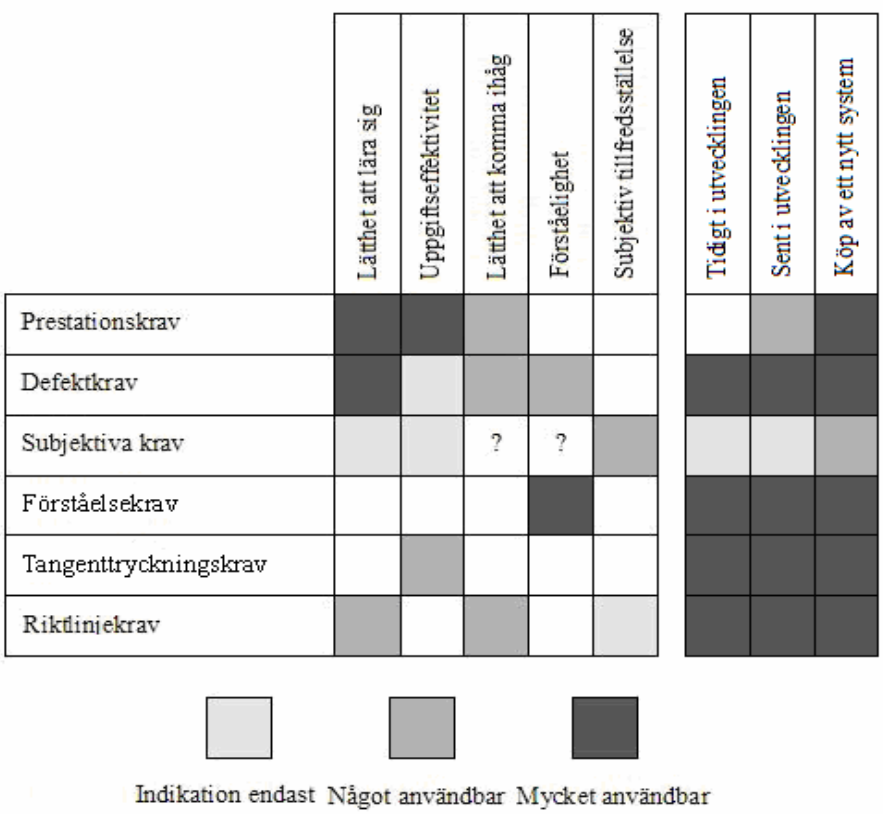
Riktlinjekrav Det är vanligt inom systemutveckling att använda sig utav riktlinjer för att uppnå användbarhet. Men även om användningen av riktlinjer är en bra metod är det långt ifrån en garanti att uppnå god användbarhet. Exempel på ett riktlinjekrav är:" Systemet ska följa MS-Windows riktlinjer för design av användargränssnittet". Fördelen med att använda sig av riktlinjer är att användarna kan ganska enkelt klara av övergången mellan olika programvaror, det vill säga aspekter som lättheten att komma ihåg och lärbarhet kommer att stödjas av denna kravtyp. Nackdelen är att riktlinjer inte säger så mycket om en mjukvara är enkel eller svår att använda. Riktlinjer är även väldigt generella, då de inte konkret beskriver hur de ska ta form i en design och kommer därför att tolkas olika av olika utvecklare.

2.3.6 Vilken typ av användbarhetskrav att välja

Att uppnå ett användbart system innebär ofta att arbeta mot flera aspekter av användbarhet och kombinera olika typer användbarhetskrav. Valet av aspekter och kravtyper kan däremot göra en utvecklare lätt förvirrad. Lauesen (2005) har därför tagit fram en sammanställning av vilka kravtyper och aspekter som hör ihop, presenterat i figur 2, och en beskrivning över vilka aspekter och kravtyper som passar till vilka typer av system.

I situationer där utvecklingen av ett system ska möta dagligt icke kritiskt användande är lättheten att lära sig och förståelseaspekten av användbarhet de viktigaste faktorerna. Genom att till exempel kombinera defektkrav tillsammans med förståelsekrav skulle dessa aspekter täckas. För system som måste utföra tidskritiska uppgifter är knapptryckningskrav ett bättre alternativ av användbarhetskrav. I webbsystem är det däremot viktigt att locka till sig kunder, den aspekt av användbarhet som därför mest står ut är subjektiv tillfredsställelse. Denna aspekt skulle kunna mätas genom att låta olika användarkategorier besvara enkäter om systemet.

Figur 2 tar även upp en beskrivning när i projektet och för vilket system användbarhetskrav är lämpliga att utföra. Enligt Lauesen (2005) är det i situationer av köp av ett nytt program viktigare att mäta prestationskrav snarare än defektkrav för att täcka uppgiftseffektivitet bättre. Vidare nämner han även att lättheten att komma ihåg är en aspekt som sällan testas då flertalet utvecklare antar att denna aspekt kan täcks av att ställa krav på lättheten att lära sig. Enligt Lauesen är detta däremot något som inte är bekräftat och bör undersökas vidare. Sammanfattningsvis är det viktigt att förstå att inga projekt är lika varandra och valet av användbarhetsaspekter bör göras med omsorg för varje projekt. I till exempel större projekt kommer olika kombinationer av användbarhetsaspekter behöva kombineras för att täcka olika kategorier av användare.



Figur 2. Typ av användbarhetskrav kopplat till aspekt av användbarhet (inspirerad av Lauesen 2005).

2.4 Kravframställning

För att bygga rätt system krävs information om arbetsdomänen och de problem som är associerade. Som källa till denna information finns användarna och det är utvecklarnas arbete att tillsammans med användarna att omvandla informationen till krav. Processen kan däremot inte ses som en passiv tolkningsprocess, utan utvecklarna måste även tillföra målet för vad mjukvaran ska erbjuda, det vill säga element av uppfinningsrikedom som mynnar ut i enklare, mer effektivare och intressantare arbetsprocesser för användarna (Robertson & Robertson, 1999). Enligt Robertson och Robertson (1999) innehåller denna process fyra etablerade steg:

1. Den första behandlar hur en utvecklare måste observera, lära sig och förstå det arbete användarna utför. Genom att arbeta tillsammans med användarna får utvecklarna tillfälle att fråga dem vad de utför för arbete och varför de utför arbetet.
2. Utvecklarens nästa uppgift är att tolka det arbete användaren utför. Detta innebär att objektivt analysera informationen från första steget och filtrera bort tekniken och utförandet och endast behålla kärnan av den, det vill säga varför användaren utför uppgiften.
3. Steg tre för utvecklaren är att hitta bättre metoder för att utföra det arbete användarna utför. Detta sker genom att studera kärnan i användarnas uppgifter (informationen från steg två) och identifiera vad mjukvaran måste göra för att stödja dessa arbetsuppgifter.
4. Som sista del måste utvecklaren dokumentera resultatet av processen i en kravspecfikation och analysmodeller. Utvecklaren och användaren måste nå

en gemensam grund om mjukvaran för att försäkra sig att rätt mjukvara har utvecklats.

2.4.1 Kravtyper

Grunden för ett lyckat mjukvaruprojekt är att fånga alla de krav som finns associerade med mjukvaran så tidigt som möjligt. Ett vanligt scenario är däremot att krav även kommer att upptäckas i de senare delarna av mjukvaruprocessen. Alltsom ofta beror detta på bristen att inspirera intressenterna att tänka bortom deras förutfattade meningar och kommunicera vad de vill ha (Robertson, 2001). Robertson (2001) har identifierat och kategoriserat tre typer av krav intressenter kommer att förmedla i ett mjukvaruprojekt.

Den första typen av krav kallar Robertson (2001) för *medvetn*a krav. Dessa krav är vad intressenter troligtvis kommer att förmedla till en utvecklare då de är medvetna om dessa behov. Tre orsaker detta kan bero på är: att det är syftet med att bygga en ny produkt, att en produkt inte nått upp till något specifikt mål eller att en intressent blivit medveten om någon ny teknologi. Gemensamt för alla dessa situationer är att intressenten är medveten om krav beroende på dennes kunskap om världen och kommer därför att troligtvis kommunicera dessa krav.

Den andra typen av krav är de som inte kommuniceras för att intressenten inte är medveten om att denne redan besitter dem. Robertson (2001) kallar dessa för *omedvetna krav*. Anledningen till dessa situationer är att intressenten blivit van vid kravets närvaro och därför inte längre behandlar det som ett krav, till exempel en kamera som alltid spolar tillbaka filmen efter att fotorullen är slut. Intressenten skulle först bli medveten om kravet när funktionen är bortplockad och skulle inte förstå varför kravet inte fanns med till produkten. Problemet kan även uppstå när intressenter har mycket kunskap om ett specifikt område och antar att andra har samma kunskap som denne. En annan orsak kan vara att intressenten känner att denne behandlar en utvecklare nedlåtande då denne tar upp alla detaljer om ett specifikt område.

Den sista typen av krav har Robertson (2001) kategoriserat som de *krav som intressenter inte visste var möjliga att uppfylla*. Dessa krav kopplas till situationer där intressenten har en klar idé om vad denne vet är möjligt att utföra och kommer inte att kommunicera krav som denne inte tror är möjliga att utföra. Detta kan bero på att intressenten inte tror det är nåbart inom projektet, till exempel att inte vattensäkra en kamera på grund av budgetbegränsningar, eller att intressenten inte är medveten om någon ny teknologi. Enligt Robertson (2001) är det viktigt att tänka på att det inte är intressenterna som kommer att hitta dessa krav utan det är utvecklarna och teknikspecialisterna. Det är därför viktigt för utvecklarna att inspirera intressenterna för att tidigt upptäcka dessa krav då de fortfarande kan förverkligas i ett system.

Robertsons (2001) uppdelning av krav har starka anknytningar till funktionella krav. Däremot är det svårt att se att uppdelningen kommer att skilja sig när det kommer till användbarhetskrav. Eftersom uppdelningen av krav baseras på intressentens kunskaper kan uppdelningen mycket väl även appliceras på användbarhetskrav.

2.4.2 Psykologiska aspekter av att identifiera krav

Robertsons (2001) uppdelning av krav kan även kopplas till psykologiska teorier. Inom kognitionsvetenskapen har det länge varit känt att igenkänning är en överlägsen minnesprocess över erinring för människor att återhämta information. Detta kan

förklaras med tvåstegs-teorin som antar att erinring kommer att producera ett sämre minnesresultat på grund av att den innehåller två steg och igenkänning endast ett steg (Watkins & Gardiner, 1979, i Eysenck & Keane, 2000). Intressentens tendens att förmedla medvetna krav, saknade funktionaliteter, kan därför ha sin förklaring i att intressenten blir påmind om dessa vid utfrågning. Teorin kan även erbjuda viss förståelse för varför intressenter kommer att ha svårare att förmedla omedvetna krav. Enligt Robertson (2001) tillhör de omedvetna kraven de krav som redan är en del av ett befintligt system och därmed redan uppfylls. Intressentens förmåga att inte förmedla dessa krav kan förklaras med att kraven inte uppstår som något nödvändigt behov från intressentens sida då det redan uppfylls och därför kräver att intressenten måste erinra dessa krav för att de ska bli kända för utvecklaren.

Intressentens tendens att inte ta upp tekniska krav som inte denne visste var möjliga, Robertsons (2001) tredje typ av krav, behöver inte förklaras med någon psykologisk teori. Kravtypen berör däremot inte endast omedvetna tekniska aspekter utan även krav som intressenten inte tror är möjliga på grund av begränsningar inom projektet. Denna benägenhet kan däremot ha sin grund i psykologiska aspekter. Inom psykologin har tester visat att individer överskattar hur kända deras åsikter, attityder och kunskaper är hos andra, vilket har kommit att kallas för falsk konsensuseffekt (Allport, 1924, i Augoustinos & Walker, 1995; Ross, Greene & House, 1977, i Augoustinos & Walker, 1995). Att en intressent inte förmedlar Robertsons tredje typ av krav kan förklaras med falsk konsensuseffekt, där intressenten tror att övriga involverade redan har kommit fram till samma slutsats som denne har. Detta fenomen kan även förklara varför intressenten inte tar upp omedvetna krav.

Som till hjälp för att hitta krav och överkomma dessa hinder finns ett flertal insamlingstekniker som utvecklare kan använda. Ingen av teknikerna är däremot fulländad att finna krav och det finns heller ingen insamlingsteknik som passar till alla situationer (Robertson, 2001).

2.4.3 Insamlingstekniker för att identifiera krav

Listan för insamlingstekniker att utvinna krav kan göras lång, endast Lauesen (2002) tar upp 19 tekniker. Att studera samtliga insamlingstekniker är utanför omfattningen av detta arbete och det måste därför ske ett urval av tekniker som ska studeras. I Easterbrook och Nuseibeh (2000) beskrivs ett antal tekniker för att identifiera krav. Deras beskrivning innefattar en kategorisering där de mer traditionella teknikerna har definierats som frågeformulär, intervjuer, analys av existerande dokumentation som processmodeller, standarder och manualer till existerande system. I Lauesen och Younessi (1998) nämns fem insamlingstekniker för att finna mätvärden för användbarhetskrav: intervju, gruppdiskussion, observation och analys av liknande produkter. Följande kapitel kommer att presentera en genomgång av traditionella insamlingstekniker och insamlingstekniker som lämpar sig för att identifiera användbarhetskrav.

2.4.4 Frågeformulär

Vad som skiljer denna teknik från övriga är dess förmåga att behandla ett större antal intressenter. Tekniken lämpar sig att användas för två situationer: att samla statistisk data för att bekräfta en misstanke och att samla åsikter eller förslag. I det första fallet måste intressenten svara på slutna frågor som: ”hur enkelt är det att få tag på kundstatistik med det aktuella systemet, väldigt enkelt, enkelt, svårt, väldigt svårt”. Resultatet kan sedan användas för att identifiera hur allvarligt ett problem är. I det

andra fallet kan mer öppna frågor ställas som till exempel: ”vad är de tre största problemen i ditt dagliga arbete eller hur skulle du vilja förbättra IT-supporten”. Som teknik är frågeformulär är inte helt befriad från problem, bland annat kan resultatet av frågeformulären vara svåra att tolka om intressenten missuppfattat frågan. Jämfört med intervjuer saknar även frågeformulär möjligheten att ställa mer utforskande frågor till intressenten. Viktigt innan frågeformulären skickas ut är att testa dem mot en målgrupp. Förekomsten av missförståelse är oftast större än vad en utvecklare tror. Resultatet bör sedan leda till en ny design av frågeformuläret som kan gå ut till intressenterna (Lauesen, 2002).

Frågeformulär som teknik skiljer sig mot övriga på så sätt att den kopplar ifrån intressenten ifrån utvecklaren. Situationen gör därför att den riktar sig främst emot att identifiera intressenternas medvetna krav. Detta beror på att intressenten är utlämnad att förmedla de krav denne tycker är viktiga och kommer inte få hjälp av utvecklaren för att undvika att uppvisa de psykologiska effekterna av falskkonsensuseffekt och tvåstegsteorin.

2.4.5 Intervju

Enligt Lauesen (2002) är intervjuer en bra metod för att fånga kunskap om arbetsdomänen och de aktuella problemen. Tekniken kan till exempel ge information om vad som är realistiskt att utföra inom ett mjukvaruprojekt eller identifiera konflikter som finns inom arbetsdomänen. Många utvecklare anser att intervju är den främsta tekniken för att identifiera krav och många användningsområden finns beroende på vilka som intervjuas och vilka frågor som ställs. Vilka som ska intervjuas är en kritisk punkt för att hitta rätt typ av information. Enligt Lauesen (2002) bör intervjuer utföras både på representativa intressenter från varje användargrupp, men även ickerepresentativa. Detta beror på att ledningen oftast väljer ut representativa intressenter att intervjuas från användargruppen, men det är enligt Lauesen inte alltid ledningen som vet vad som egentligen försiggår i de dagliga aktiviteterna. Vilka frågor som bör ställas beror på när de ställs. Som tumregel är det lämpligt att börja med att ställa breda frågor om dagliga aktiviteter och problem. Vidare kan frågor ställas om vilka uppgifter som är kritiska, när användaren arbetar under stress och när är det viktigt att uppgifter utförs helt korrekt. Denna typ av information kan inte identifieras med tekniker som observation och måste därför behandlas i intervjusituationer. Det är även viktigt att ta reda på varför användaren utför sina uppgifter. Problemet med denna fråga är däremot att intressenter kan ha svårt att förklara varför det sker. En god idé kan därför vara att istället ställa frågan *när utför du uppgiften* istället för *varför utför du uppgiften*. När dessa frågor sedan har besvarats kan mer detaljerade frågor ställas om datavolymer, uppgiftstider, detaljerade arbetsprocedurer och så vidare (Lauesen, 2002). Intervjuer kan utföras på många olika sätt, vanligast är däremot att utföra öppna intervjuer. Öppna intervjuer är de situationer där utvecklaren har förbestämda frågor, men får även ställa mer utforskande frågor som inte dokumenterats i förväg (Goguen & Linde, 1993).

Intervjuer är den kanske mest välanvända tekniken för att identifiera krav. Tekniken sätter utvecklare och intressent tillsammans för att diskutera krav. Detta gör att tekniken främst riktar sig åt att identifiera medvetna krav. Övriga kravtyper är däremot inte helt uteslutna, beroende på vilka frågor som ställs och hur utvecklaren väljer att utföra intervjun behöver inte en intressent uppvisa de psykologiska aspekter som kommer att hindra denne att förmedla omedvetna krav och krav som denne inte visste vara möjliga att uppnå.

2.4.6 Dokumentanalys

Som teknik skulle dokumentanalys kunna beskrivas som omvänd ingenjörskonst. Detta beror på att den analyserar dokument som skapats av tidigare system för att hitta krav som kan återanvändas till ett nytt. Analysarbetet innebär att genomgå alla typer av dokumenterad information som rapporter, filer, formulär, manualer och så vidare. Som utvecklare som utför dokumentationsanalys handlar det om att vara granskande. Kunden kommer att i de flesta fall vilja se nya lösningar på problem och inte se kopierade krav från tidigare system, onekligen kommer däremot vissa krav behöva överföras till nya system. Enligt Robertson och Robertson (1999) finns det många brister med dokumentanalys och den bör därför kombineras med andra tekniker. Att till exempel kombinera dokumentanalys tillsammans med datamodellering kan vara bra för att finna kategorier av data och objekt som kan vara till användning i implementationsdelen av projektet (Robertson & Robertson, 1999).

Dokumentanalys är den bästa metoden för att finna de omedvetna krav intressenter kan ha svårt för att förmedla. Eftersom tekniken innebär att analysera dokument av tidigare system kommer de krav som redan uppfylls upp till utvecklarnas kännedom. Utvecklarens arbete är sedan att avgöra om kravet ska behållas eller omformuleras till det nya systemet.

2.4.7 Observation

Enligt Lauesen (2002) är inte användarna alltid medvetna om vad de egentligen gör och hur de utför det. Observationer har till exempelvis visat att människors beskrivningar av hur de hittar ett recept i en kokbok inte stämmer överens med hur de faktiskt gör det. Ett sätt att komma runt denna problematik är att observera användaren i hur denne egentligen utför sina aktiviteter. En utvecklare kan spendera tid med användaren och observera eller använda sig av videokameror för att förlänga observationen. Fördelen med att utnyttja observationer är att det kraftigt ökar kunskapen om arbetsdomänen och de associerade arbetsproblemen. Tekniken fungerar även bra i kombination med andra för att verifiera annan information. Nackdelen är däremot att tekniken oftast missar de riktigt kritiska situationerna. Till exempel i ett kraftnätsdistributionssystem inträffar den kritiska situationen kanske en gång per år och oftast inte när utvecklaren är närvarnade för observation (Lauesen, 2002).

Som teknik lämpar sig observation för att fånga de krav som inte alltid användaren är medveten om att denne ställer på ett nytt system, det vill säga de omedvetna kraven. Eftersom tekniken endast innebär att observera användaren är den befriad från de psykologiska aspekter som kan hindra utvecklare att hitta krav. Tekniken är däremot inte fulländad, den ställer till exempel stora krav på utvecklarna att förstå vad användarna utför och varför de utför det.

2.4.8 Gruppdiskussion

Lauesen (2002) tar upp fyra olika former av gruppdiskussioner utvecklare kan använda sig utav för att hitta information till ett nytt system. Dessa tekniker omfattar idékläckning (brainstorming), fokusgrupper (focus group), domänarbetsgrupper (domain workshop) och designarbetsgrupper (design workshop).

Idékläckning Denna teknik går ut på att samla en grupp människor, skapa en kreativ och fokuserad stämning och låta deltagarna komma på idéer utan att bli kritiserade. Idéerna ska dokumenteras synligt utav utvecklaren för att deltagarna ska kunna bygga

vidare på dem. Under kravframtagningsfasen är det viktigt att tekniken inriktas på att ta fram mål till en ny programvara. För att uppnå detta kan utvecklaren ställa frågor som kan hjälpa deltagarna att hålla sig till ämnet. Idékläckning kan som teknik komma med många orealistiska krav på en mjukvara, detta gör att tekniken inte alltid lämpar sig för att identifiera krav.

Fokusgrupper Fokusgrupper liknar på många sätt idékläckningstekniken, men är mer strukturerad. Tekniken går ut på att först identifiera problem med deltagarnas arbete och sedan hitta bättre lösningar för att utföra arbetet. Deltagarna måste även komma på anledningar till varför en lösning är bättre då det hjälper utvecklaren att omformulera det till ett krav. En viktig del av att utföra fokusgrupper är att använda sig utav olika användargrupper och låta dessa prioritera vilka problem som är viktigast att lösa för dem. Detta göra att utvecklarna kan fokusera på de mest kritiska delarna av att bygga ett system och se även till att varje användargrupp får sina mest allvarliga problem lösta. Prioriteringen gör även de olika användargrupperna mer medvetna om andra användares krav på ett nytt system.

Domänarbetsgrupper Vad som skiljer denna teknik och nästföljande (designarbetsgrupper) mot övriga är att de sätter ihop intressenterna och utvecklarna för att analysera och designa något. Domänarbetsgrupper gör detta med fokus på affärsprocesser. Tekniken riktar sig mot att identifiera affärsprocesserna och beskriva dem i dataflödesdiagram eller aktivitetsdiagram, vilket hjälper utvecklaren att identifiera krav. Viktigt i att upprätta en domänarbetsgrupp är att samla expertanvändare som kan beskriva sina delar av organisationen. Ofta behövs flera expertanvändare då de är begränsade i sina kunskaper till en liten del av organisationen och utvecklare behöver ta del av hela affärsprocessen. En effekt av att arbeta med domänarbetsgrupper är att det kräver att utvecklarna måste specificera målen och de kritiska frågorna till ett nytt system, detta kan lättast göras med hjälp av ledningen av en organisation.

Designarbetsgrupper En designarbetsgrupp sätter ihop utvecklare och användare för att designa något, vanligtvis ett användargränssnitt. Denna teknik är väl omtalad, men resulterar alltför oftast i katastrof (Lauesen, 2002). Anledningen är att användarna blir så delaktiga i utvecklingsarbetet att de glömmer sin roll som användare och inriktar mer på sig att systemet ska uppnå tekniska lösningar, vilket överskuggar målet och uppgifterna mjukvaran ska stödja. För att denna teknik ska bli framgångsrik är det viktigt att kontinuerligt testa om användargränssnittet möter de uppgiftsbeskrivningar och mål som satts upp för mjukvaran. Detta bör även kompletteras med användartest med användare utomstående utvecklingsarbetet för att verifiera en design.

På grund av den öppna formen av interaktion mellan utvecklare och användare är gruppdiskussioner bra tekniker för att identifiera krav. Interaktionen och de olika formerna gör även att tekniken kan identifiera samtliga kravtyper. Idékläckning och fokusgrupper kommer att hitta intressenternas medvetna krav, då de tillåter intressenterna förmedla vad de skulle vilja se i ett nytt system som inte åstadkoms med det aktuella. Domänarbetsgrupper kommer att identifiera omedvetna krav, på grund av att det ges tillfälle till intressenterna att modellera hur deras verksamhet ser ut och fungerar idag. Till sist kommer designarbetsgrupper att identifiera de krav som intressenterna inte trodde var möjliga, då situationen ger utvecklarna tillfälle att kommunicera med intressenterna på ett sådant vis att de kan inspirera dem att tänka på saker de inte tidigare tänkt på eller visste var möjliga.

2.4.9 Studera andra produkter

En viktig källa av information är att studera liknande produkter producerade av andra företag. Tekniken kan enkelt skapa en lång lista av funktionella krav utvecklare kan vara intresserade av att använda till en ny produkt. Tillägget av funktionella krav på ett system blir däremot till en viss gräns överflödigt och det är därför viktigt att studera de ickefunktionella kraven i andra produkter. Detta beror på att alla tillverkare kan utan större ansträngning uppnå samma nivåer av funktionella krav. Genom att fokusera mer på de ickefunktionella kraven hos andra produkter, kan det sätta mål och krav om vad som ska uppnås för en ny mjukvara och även inspirera utvecklarna att hitta nya ickefunktionella krav som andra företag inte tidigare tänkt på (Lauesen, 2002).

Att studera andra produkter som teknik att hitta krav berör inte så mycket Robertsons (2001) kravtyper, då intressenten är helt frånkopplad tekniken. Den typ av krav den kan identifiera är intressentens omedvetna krav. Insamlingstekniken kan liknas vid att utföra en dokumentanalys med till skillnad att kraven är realiserade i en produkt.

3 Problemområde

I detta kapitel presenteras arbetets problemområde och problemprecisering som ligger till grund för rapporten. Som avslutande del kommer kapitlet även ta upp de avgränsningar som gjort i arbetet.

3.1 Motivering

Precis som funktionella krav måste dokumenteras för att specificera vad för funktioner en mjukvara måste innehålla, måste användbarhetskrav dokumenteras för att uppnå de krav användarna ställer på att använda systemet. Inom traditionell systemutveckling har tekniken oftast varit prioriterad. Det dominerade fokus på teknik har lett till att andra aspekter av mjukvaruutveckling har blivit försummade (se kapitel 2.1). Att erbjuda ett system med oändligt många funktioner och låta användaren arbeta mot ett textbaserat användargränssnitt, kommer däremot att inte hjälpa användaren att lättare utföra sina uppgifter. Även om exemplet med ett textbaserat användargränssnitt var extremt är det ändå en poäng som måste lyftas fram, det vill säga att om det inte finns några krav på hur användargränssnitt ska fungera spelar det inte någon roll hur många bra funktioner som erbjuds i en mjukvara. Problematiken ligger i att funktionella krav och ickefunktionella krav är så tätt sammankopplade att en utvecklare inte kan specificera det ena utan att specificera det andra (se kapitel 2.3.3). Att endast erbjuda en massa funktioner utan att ställa krav på hur de ska fungera mot användaren kommer leda till problem vid användning. Utvecklare kan vara duktiga i att ta fram gränssnitt som stödjer användaren i dess aktiviteter, men utan krav finns det inga mål att arbeta emot eller att sträva efter när det kommer till användargränssnittet (se kapitel 2.3.4). Området användbarhet är däremot något som mer och mer börjar få större uppmärksamhet inom systemutveckling. Allteftersom våra arbeten kompletteras med olika programvaror måste även människor som inte har en någon större erfarenhet med datorer arbeta med dessa system (se kapitel 2.1). Personer som Lauesen och Preece med kollegor har uppmärksammat detta och förespråkar nu en mer användarcentrerad systemutveckling där användarens aktiviteter och behov är i centrum för utvecklingsprocessen (se kapitel 2.3.4). Även företag börjar få upp ögonen för detta område då kundernas medvetenhet om bristen av användbarhet i produkter gör att de väljer bort de produkter som inte uppfyller deras krav. Fortfarande saknas det kompetens och metoder för att helt integrera användaren i utvecklingsarbetet och få fram mer användbarhetsanpassade produkter. Än är det dessutom alltför stor fokus på teknik när det kommer till att ta fram nya produkter (se kapitel 2.1). Anledningarna till detta kan vara utvecklarnas trygghet att endast fokusera på de tekniska och funktionella delarna. Än finns det nämligen inte mycket dokumenterat om hur användbarhetskrav stödjer utvecklingen av ett system och hur utvecklarna ska kunna ta nytta utav dem. Mycket av vad som finns dokumenterat är ofta ”luddigt” beskrivet utan att ge utvecklare någon explicit guidning om hur de ska gå tillväga. Detta gäller specifikt för insamlingstekniker att ta fram användbarhetskrav ur en domän, vilket ställer till med ännu mer problem då en av framgångsfaktorerna för ett lyckat mjukvaruprojekt är en väl utvecklad mjukvaruspecifikation (Standishgroup, 1994). Mycket av traditionell mjukvaruutveckling står nog däremot inför en förändring. Teknikfokus på mjukvaruutveckling är inte hållbart när datorer och teknologiska produkter blir en del av en människas vardag. Metoder för att ta fram mjukvara kommer troligen därför att se en förändring med ett större fokus på människan i utvecklingen (se kapitel 2.1).

Detta arbete avser därför att studera användandet av insamlingstekniker för att ta fram användbarhetskrav.

3.2 Problemprecisering

Arbetet gör ett antagande om att en koppling kan göras mellan insamlingsteknik och aspekt av användbarhet. I detta sammanhang avser aspekt av användbarhet: lärbarhet, uppgiftseffektivitet, möjlighet att komma ihåg, förståelighet och subjektiv tillfredställelse (se kapitel 2.3.4). Studien kommer att undersöka om en specifik insamlingsteknik kan fördelaktigt användas för att identifiera en aspekt av användbarhet, till exempel använda sig av intervju för att identifiera uppgiftseffektivitetskrav. Frågan till arbetet är:

Kan specifika insamlingstekniker användas för att fördelaktigt identifiera olika aspekter av användbarhet?

Arbetet kommer att använda Lauesens (2002) sammanställning av aspekter av användbarhet och användbarhetskrav som utgångspunkt för att analysera utvecklares erfarenhet att identifiera användbarhetskrav (se kapitel 2.3.6). Resultatet kommer sedan att analyseras för att undersöka om en koppling kan göras mellan aspekt av användbarhet och användbarhetskravtyp.

3.3 Avgränsning

Även om det finns en ganska enad samsyn över vad användbarhet berör skiljer sig definitionerna något emellan. Till detta arbete kommer Lauesen (2002) definition användas (se kapitel 2.3.4). Aspekterna berör är lätthet att lära sig, uppgiftseffektivitet, lätthet att komma ihåg, förståelighet och subjektiv tillfredsställelse.

Vidare kan även vad ett användbarhetskrav tolkas olika av utvecklare. Arbete gör därför en avgränsning med att endast behandla de användbarhetskrav som nämndes i kapitel 2.3.5. Andra användbarhetskrav kommer inte att beröras i denna undersökning.

Som sista del kommer detta arbete inte göra någon bedömning av hur bra användbarhetskrav fungerar för att uppnå användbara system. Det finns utvecklare som är kritiska till att arbeta med användbarhetskrav och förespråkar andra metoder för att uppnå användbara system. Detta arbete kommer endast att rikta in sig på hur arbetet med att identifiera användbarhetskrav går till.

4 Metod och genomförande

Detta kapitel kommer att behandla motivering av metodvalet och hur genomförandet av studien utfördes. Valet av metod motiveras utifrån frågeställningen som fokuserar på att dokumentera utvecklares erfarenheter av att arbeta med användbarhetskrav. Som avslutade del presenteras de respondenter som deltagit i arbetet.

4.1 Metod

Att utföra något metodiskt har beskrivits av Berndtsson, Hansson, Olsson och Lundell (2002) som ett systematiskt utförande i att behandla ett problem. Valet av metod kommer därför att både påverka kvalitén i resulterande data och slutsatsen som kan dras utifrån arbetets utgångspunkt (Berndtsson et al., 2002). En grov indelning av metoder kan göras mellan kvantitativa studier och kvalitativa studier. Eftersom arbetet riktade in sig på att besvara frågeställningen genom att dokumentera utvecklares erfarenheter med att arbeta med användbarhetskrav föll metodvalet på att utföra en kvalitativ studie. Enligt Patton (2002) är styrkan bakom att utföra en kvalitativ studie att kunna studera någonting i djup och detalj utan att begränsas av förutbestämda kategorier av analys av djup, öppenhet eller detalj. Vidare kan en kvalitativ undersökning producera en rik detaljerad mängd av data på ett mindre antal individer eller fall, vilket passar till denna studie då både resurser och tid var begränsade inom studien. Den däremot främsta anledning till att utföra en kvalitativ studie är den öppna formen av svar som möjliggör att förstå världen utifrån respondenternas perspektiv. Något som inte en kvantitativ undersökning kan utföra på grund av dess mer standardiserade och systematiska sätt att samla in och analysera data.

Kvalitativ data kan växa ur tre typer av tekniker: intervju, observation eller skrivna dokument. Data från intervju innebär direktciteringar av människors erfarenheter, åsikter, känslor och kunskap. Observation innebär detaljerade beskrivningar av individers aktiviteter, beteende, handlingar eller hela spektra av mellanmänskliga interaktioner och organisationella processer som är en del av det observerbara mänskliga beteendet. Skrivna dokument innebär att studera citeringar, officiella rapporter, enkäter och så vidare för analys av data. Till detta arbete föll det på att utföra intervjuer eftersom studien var till för att dokumentera utvecklares erfarenheter av att använda olika insamlingsmetoder och intervjuer är en bra teknik för att sätta sig in i andra människors perspektiv. Alternativt kunde en enkätstudie ha utförts för att beröra ett större antal respondent och på så vis få fram ett bättre generaliserbart material. Problemet hade däremot varit att skapa en enkät som kunde undersöka utvecklares erfarenheter, eftersom svaren kan vara varierade och behöva utvecklas om respondentens ståndpunkt inte framgår. Med anledning av detta föredrogs det att utföra intervjuer framför enkäter för att kunna ställa mer fördjupande frågor vid de tillfällen de behövs.

Enligt Patton (2002) finns det tre olika tillvägagångssätt att samla information från en intervju. Skillnaden mellan tillvägagångssätten ligger i vilken utsträckning intervjufrågorna är förbestämda och standardiserade innan intervjun. Den första intervjuteknik Patton (2002) beskriver är informell konversationsintervju. Denna teknik förlitar sig på spontan generering av frågor vid intervjutillfället och utförs oftast i kombination med andra tekniker som observation. Den andra intervjutekniken, intervjuguide, går ut på att skapa en generell struktur över hur intervjun ska gå till med fördjupande frågor. Guiden kommer då att fungera som en checklista för att säkerställa att relevanta ämnen berörs. Den tredje typen av intervju är standardiserad

öppen intervju där frågor specificeras med noggrannhet innan intervjun i syfte att samma frågor kan ställas vid flera tillfällen. Valet mellan dessa tre gjordes på att utföra en intervju med intervjuguide, detta på grund av att kunna ställa likartade frågor till respondenterna för att jämföra deras svar men även fördjupande frågor för att ta reda på mer information om vissa ämnen.

4.2 Genomförande

Efter att avslutat litteraturstudien till arbetet, där en genomgång av berörd litteratur utfördes, påbörjades skapandet av en intervjuguide. Intervjuguiden kom att förutom beröra de ämnesområden som behandlades i litteraturstudien, även att inrikta sig emot att besvara frågeställningen:

Kan specifika insamlingstekniker användas för att fördelaktigt identifiera en aspekt av användbarhet?

Upplägget av intervjuguiden var att dela upp den i fyra delar. Denna uppdelning gjordes för att inte endast låta deltagaren svara på direkta frågor berörande frågeställningen, utan för att skapa en komplett bild om deltagarens åsikter och förstå dem utifrån deras bakgrund. För att mjukstarta intervjun och göra deltagaren mer bekväm med intervjusituationen kom första delen att återspegla mer neutrala frågor. Första delen kom därför att behandla intervjudeltagarens bakgrund rörande ämnesområdet. Frågorna berörde bland annat deras erfarenhet som utvecklare av systemutvecklingsprojekt och metodanvändning. Del två fördjupade sig vidare i insamlingstekniker där deltagaren fick berätta om dennes erfarenheter kring användandet av olika insamlingstekniker. Del tre av intervjuguiden behandlade användbarhet generellt inom mjukvaruutveckling. Där fick deltagaren berätta om dennes åsikter om användbarhet och vad för betydelse användbarhet har för dem. Sista delen av intervjuguiden behandlade användbarhetskrav. Syftet med denna del var att koppla ihop delen som behandlade insamlingstekniker med den del som berörde användbarhet. Delen kom därför att behandla hur användbarhetskrav kan användas för att uppnå en god användbarhet på en produkt och hur de går tillväga för att identifiera användbarhetskrav, om det finns insamlingstekniker att föredra. Se bilaga I för intervjuguide.

För att utföra intervjuerna till studien kontaktades fem företag, varav tre visade intresse. På dessa tre organisationer utfördes totalt fem intervjuer, det vill säga på två av företagen kom det att utföras två intervjuer och på ett av företagen en intervju. Eftersom arbetet kom att behandla till stor del hur användbarhetskrav tas fram kom samtliga intervjuer att utföras med utvecklare med olika inriktningar. Med utvecklare i detta avseende menas att de på något sätt bidrar till utvecklingen av ett nytt system, oavsett roll inom mjukvaruutveckling. För att förbereda deltagarna skickades intervjuguiden ut i förväg för att de skulle kunna få tid över att reflektera över de frågor som sammanställts i intervjuguiden. Intervjuerna utfördes senare på plats hos de olika företagen som hade kontaktats. Längden på intervjuerna varierade mellan 45 till 60 minuter varav alla intervjuer spelades in på band med deltagarens medgivande. Inspelningarna kunde sedan användas för att sammanställas på papper och senare analyseras tillsammans med litteraturstudien. Analysen gick i stora delar ut på att se likheter och mönster i respondenternas svar för att kunna dra slutsatser om hur arbetet med användbarhet, insamlingstekniker och användbarhetskrav går till.

Utförandet av intervjuerna gick bra utan några större incidenter. Tack vare valet av intervjuguide kunde missförstånd bland frågor snabbt rättas till så att deltagaren kunde svara. Formen tillät även att ställa mer fördjupande frågor när det fanns ett

behov att ta reda på mer information och frågorna kan därför variera något bland deltagarna beroende på den information som gavs.

Respondenterna som deltog i undersökning presenteras i kapitel 4.3 och resultatet av intervjuerna kommer att presenteras och analyseras i fyra delar i kapitel 5.1 till 5.4. Den första kommer att behandla användbarhet, respondenternas syn och åsikter kring begreppet (kapitel 5.1). Del två berör användandet av olika insamlingstekniker (kapitel 5.2), del tre användbarhetskrav (kapitel 5.3) och del fyra användandet av olika insamlingstekniker för att identifiera användbarhetskrav (kapitel 5.4).

4.3 Beskrivning av respektive respondent

Nedan kommer en beskrivning av respektive respondent som deltog i intervjuerna. Beskrivningarna kommer att ta upp deras bakgrund i att utveckla system och behandla deras arbetsuppgifter, antal år inom branschen och en kort presentation av företaget de arbetar på.

Deltagare 1 arbetar på ett konsultföretag inriktat på att ta fram Internet-relaterade system åt bank- och finansorganisationer. Organisationen arbetar främst emot banker där de varit involverade i att ta fram kassasystem, Internetbanker, kortsystem och så vidare. Organisationen arbetar främst med Rational Unified Process (RUP) som utvecklingsmetod då den passar mot deras kunder som ställer höga krav på dokumentation. Deras roll som konsulter gör däremot att de måste vara flexibla med att anpassa utvecklingsmetod efter kund. Systemen som byggs är komplexa då de ofta ska interagera med ett flertal andra system och berör många personer.

Respondenten arbetar med systemarkitektur och som teknisk projektledare. Arbetet innebär varierande uppgifter från att bidra med teknisk kompetens till att arbeta med administrativa uppgifter om kostnadsförslag. Deltagare ett har arbetat 11 år inom organisationen och har en utbildning inom objektorientering.

Deltagare 2 arbetar på samma konsultföretag som deltagare ett, men har tidigare varit anställd på företag som utvecklat administrativa system åt polisen och posten. Respondenten har erfarenhet av flera utvecklingsmetoder som RUP, agila metoder och extrem programmering.

Respondenten arbetar med systemarkitekturen i utvecklingsprojekt. Arbetsuppgifterna går ut på att gå igenom de funktionella kraven för att plocka ut och prioritera och skapa ett utdrag till en arkitektur. Deltagare två har arbetat med utveckling av system i 10 år.

Deltagare 3 arbetar på ett större konsultföretag som utvecklar system åt andra organisationer. Systemen som framarbetats har varit varierande, från ärendesystem åt myndigheter till koncept för system för flygledning. På grund av konsultrollen får respondenten alltid anpassa utvecklingsmetod efter kund. Deltagare tre anser sig däremot kritisk till olika utvecklingsmetoder, då de i grunden fortfarande innebär att utföra samma steg. Vidare anser denne att organisationer ofta upprättar en ”best-practice”, ett tillvägagångssätt som inte följer en specifik metod, men ser till att ett utvecklingsprojekt blir klart.

Respondenten arbetar idag som användbarhetsdesignerkonsult. Rollen innebär att utföra analys, design och utvärdering efter ett användarcentrerat angreppssätt. Som utvecklare har deltagare tre arbetat med tio år utspritt över tre företag. Rollerna har skiftat under åren, men har alltid knutit an till ett användarcentrerat arbetssätt. De

senaste sju åren har deltagare tre varit anställd på det företag denne arbetar för idag, där denne endast arbetat med analys och design i utvecklingsprocessen.

Deltagare 4 arbetar på samma företag som respondent tre och har därför samma varierande bakgrund av att ta fram system. Deltagaren beskriver sitt arbetssätt som användarcentrerad systemdesign där användaren är i fokus för utvecklingen. Detta arbetssätt motiverar respondenten med att en användares beteende inte kan formaliseras till någon modell då användarens beteende till största del kommer att styras av miljön runt omkring.

Respondent fyra arbetar som användbarhetsdesignerkonsult. Deltagaren har flera års erfarenhet med att arbeta med denna roll att ta fram användbara system och användbarhetsprocesser åt olika företag.

Deltagare 5 arbetar på ett konsultföretag som erbjuder utveckling, analys och utbildning av olika system. Där respondenten bland annat vart med att ta fram system till handdatasystem för att lasta dricksbackar för lastbilschaufförer och bokningssystem för golftider. Som konsult måste respondenten anpassa utvecklingsmetoden efter kund och har erfarenhet av att bland annat arbeta efter RUP. Deltagare fem anser däremot att det är få företag som följer RUP utan endast använder sig av dess artefakter. Deltagaren har även erfarenhet av DELTA / GAM-metoden, som är en utvecklingsmetod skapad för att ta fram användbara system.

Respondenten arbetar som användbarhetsexpert, men beskriver sin roll i ett större perspektiv än att endast arbeta med design. Deltagare fem arbetar även med en mer rådgivande roll för projektledare som vill arbeta med användbarhet. Totalt har deltagare fem arbetat med användbarhet i sju år på två företag.

5 Analys och resultat

I denna del av rapporten presenteras resultatet och analysen av de intervjuer som utfördes. Analysen och resultatet är uppdelad i fyra delar som berör ämnesområdet till rapporten. Varje del börjar med att presentera analysen av respondenternas svar och avslutas med en sammanfattning av analysen. Som avslutande del till kapitlet presenteras slutsatsen till undersökningen som utfördes (se kapitel 5.5).

5.1 Deltagarnas syn på användbarhet

Samtliga deltagare anser att användbarhet är en av de viktigare faktorerna till att utveckla ett korrekt system. Definitionen av användbarhet skiljer sig däremot något mellan deltagarna. Respondent tre, fyra och fem uppgav ISO-förteckningen av användbarhet som den definition som står dem närmast, eftersom ISO-förteckningen inte endast berör användbarhet som ickefunktionellt krav utan även som funktionellt krav (se bilaga II för ISO-förteckning). Deltagare ett och två beskriver användbarhet på ett annat sätt, deltagare ett definierade användbarhet som:

”utifrån vad systemet är tänkt att göra, hur bra användarna kan genomföra det som är tänkt på ett enkelt och bra sätt.”

Deltagare två ansåg att användbarhet inte bara berör användargränssnittet utan även miljön runt omkring så som ergonomiska aspekter. Dessa skillnader i definitioner av användbarhet beror troligtvis på respondenternas olika bakgrunder som utvecklare, då deltagare tre, fyra och fem som arbetar som användbarhetsexperter och deltagare ett och två som utvecklare av arkitektur.

Vidare lyfter deltagare fem fram att det är nyttan som är det viktiga att fokusera på i användbara system. Om användarna inte kommer att uppleva någon nytta med att använda ett system, så kommer inte systemet användas. Deltagare fem anser att problemet inte är att utvecklare idag ignorerar användbarhet, utan tvärtom att alla säger att de utvecklar användbara system.

”säger man till folk att de ska bygga användbart så säger de bara, ja.” – Deltagare fem.

Enligt deltagare fem ligger snarare problematiken i att få in kvalificerat folk som kan arbeta med användbarhet och bidra till ett användbart system. Detta är även en av orsakerna till att respondent fem använder begreppet nytta allt oftare. Detta resonemang får stöd av respondent tre som anser att det är många som har en uppfattning om att användbarhet är något som bara uppstår vid skapandet av ett användargränssnitt.

”Det uppstår inte bara av sig själv i efterhand. Det är många som bara tror att det blir bra och användbart.” – Deltagare tre.

Dessa åsikter stämmer väl överens med deltagare två som menar att det är många som kommer att ha åsikter om hur saker ska vara placerade och på så sätt påverka användbarheten i ett system. Av respondent två, tre och fem åsikter uppfattas det som många gör en koppling mellan användbarhet och design, medan begreppet snarare innebär en kombination av analyser av användarna och design av ett gränssnitt. Att sedan kommentera hur ett användargränssnitt ska se ut utifrån sina egna erfarenheter gör inte ett system användbart, vilket är något som även gäller slutanvändare till ett nytt system. Något som urskiljdes av deltagare fem åsikter, som ansåg att låta tio slutanvändare kommentera en design till ett system behöver inte leda till ett system

mer användbart utan ett system anpassat efter tio användare. Det viktiga är snarare att gå ut och studera användarna i deras kontext, hur de verkligen arbetar för att identifiera de verkliga behoven till ett nytt system.

Däremot behöver inte användbarhet påverkas negativt av att andra utvecklare, som inte har en bakgrund som användbarhetsexperter, arbete med den. Både deltagare två och fem menar att ofta finns det utvecklare och programmerare att tillgå som bryr sig om användbarheten i ett gränssnitt vilket användbarhetsexperter kan ta del av för att laborera fram ett väl fungerande användargränssnitt. Även deltagare ett kommenterade detta med att uttala hur viktigt det är för olika grupper av utvecklare att samarbeta för att uppnå ett gemensamt mål, det vill säga en väl fungerande mjukvara.

Deltagare ett menar dock att storleken på system styr fokus på användbarheten. För till exempel projekt som kommer att beröra ett mindre antal användare behöver inte samma resurser läggas på användbarhet som på ett projekt som kommer att beröra flera hundra användare. Detta var något som uppfattades av mer eller mindre av alla respondenter. Beroende på typ av system som utvecklas kommer olika mycket tid att läggas på användbarhet. Detta ter sig ganska naturligt om en jämförelse skulle ske mellan ett flygledarsystem som ställer höga krav på att inget får gå fel mot en webbplats som förmedlar reklam om en ny produkt.

Ur intervjuerna uppfattades två synsätt på användbarhet: dels som en analysmetod och dels som ett område som behandlar designen av ett användargränssnitt. Deltagare tre, fyra och fem framförde användbarhet mer som ett sätt att analysera användarnas behov för att bygga ett korrekt system. Detta handlade till stor del om att utföra olika analyser av användarna till systemet. Deltagare ett och två framförde användbarhet mer som ett område som handlar om att skapa ett användargränssnitt, att se till att systemet fungerar på ett sådant sätt som det vara tänkt fungera. Det vill säga att specificera dialogen mellan systemet och användaren. Uppdelningen av synsätten på användbarhet var mest uppenbart hos deltagare fem som uttryckte att dennes arbete handlade mest om att utföra olika analyser för att hitta rätt lösning till deras kunder. Syftet med analyserna var inte att bygga rätt system åt kunderna utan att se till att de får rätt lösning, till exempel en reklamkampanj för att öka konsumenters medvetenhet om något.

Respondenternas olika åsikter kring begreppet användbarhet har säkert flera anledningar, men kan kopplas till de många olika sätten att specificera användbarhet (se kapitel 2.3.4). Där de olika definitionerna kan ha skapat en viss förvirring om vad det betyder och medför. Det vill säga om det handlar om ett synsätt på att utveckla system med fokus på att utföra olika analyser av användarna eller ett område som berör designen av ett användargränssnitt.

5.2 Deltagarnas syn på insamlingstekniker

Användandet av olika insamlingstekniker skiljer sig mellan de olika deltagarna, allt från användningsfall till observationer. För deltagare tre och fyra fungerade fältstudier som den primära insamlingstekniken för att samla information. Att utföra fältstudier beskrev de som att gå bredvid användarna och studera hur de utför sitt arbete samt fråga dem varför de utför det de gör. Enligt respondent fyra är fältarbete den enda insamlingstekniken som kan utföras för att fånga användarnas verkliga behov, detta motiverade deltagaren med att användarnas arbete är situerat i miljön och kan inte formaliseras till något diagram eller modell.

”Man kan inte ta uppgiften ur kontexten.” – Deltagare fyra

Vidare menar deltagare fyra att det är viktigt att se användarnas arbete som helhet för att identifiera vad det har för större syfte att tjäna, något som bara kan göras med fältstudier. Dessa åsikter är även något som stöds av deltagare tre som menar att det inte går att sammanfatta en användares arbete. Deltagare tre fortsätter med att understryka hur viktigt det är att observera användarnas beteende för att inte förlita sig på en beställares krav, då ofta användarna arbetar på ett annat sätt än vad kunden säger. Detta betyder inte att kunden ska kunna beställa nya saker för att få se en förändring av ett arbete, men i dessa fall är det fortfarande viktigt att studera och involvera användaren för att identifiera vilka behov som måste lyftas fram i ett nytt system, enligt deltagare tre. Vidare nämner även deltagaren arbetsgrupper (workshops) som en teknik för att samla in information, men menar att resultatet av dem ofta endast är till för att komplettera den information som gavs från fältstudierna.

Enligt respondent ett och två är det viktigt att de finns en väl utvecklad kravspecifikation att tillgå eller en expertgrupp att samarbeta med för att utveckla ett system. Vidare anser de att användningsfall fungerar som en bra insamlingsteknik för att fånga kraven och skapa en överblick över det nya systemet. Användningsfall beskriver en interaktion mellan en aktör och ett system, där aktören även kan agera som ett system. Deltagare ett motiverar användningsfall med att det ger utvecklare en frihet att vara innovativ och ta fram funktioner. Deltagare ett anser även att de skapar en bra överblick över vad som ska göras inför nästa iteration, vilket är främsta anledningen till varför deltagare ett anser att det är viktigt att utveckla mer användningsfall än vad som ska implementeras i systemet.

”finns det bara användningsfall över vad som ska implementeras, skär man i det innovativa med att skapa system.” – Deltagare ett.

Enligt deltagare ett räcker däremot inte användningsfall alltid till och kan behöva kompletteras med till exempel intervjuer för att fånga hela kravbilden, något som även deltagare två håller med om. Respondent två anser också att det är viktigt att producera användningsfall, men håller inte med deltagare ett om mängden användningsfall som ska produceras. Enligt respondent två bör antalet användningsfall begränsas för att minska bearbetningen kring systemen och för att koncentrera sig mer på att leverera någonting till kund.

Respondent fem uppgav intervjuer som den primära insamlingstekniken att identifiera krav. Deltagaren var däremot mer inriktad på att intervjuer är en bra insamlingsteknik att börja med för att hitta information, då denne ansåg att denna teknik kommer att behöva kompletteras av andra tekniker så som fokusgrupper och observationer. Precis som respondet tre och fyra uttryckte även deltagare fem att en utvecklare inte får förlita sig blint på vad kunden säger om hur användarnas arbete går till, då det kan skilja sig till ganska stor del från hur arbetet egentligen utförs. Det var även anledningen till varför respondenten ansåg att det är viktigt att variera användandet av olika insamlingstekniker.

”sen så behöver detta (intervjuer) ofta kompletteras när man ska kravställa mer detaljerat. Då så får man kolla med fokusgrupper eller att man skickar kraven på remiss till olika instanser i organisationen… Man måste på något sätt vidga gruppen man pratar med, så att man inte får en bild, utan hela spektrat”. – Deltagare fem

Enligt deltagare fem är det viktigt att leta efter grundproblemet, det vill säga vad det egentliga motståndet är till att utföra något. Med det menar respondenten att det inte

alltid är rätt att bygga ett nytt system och delar därmed linje med deltagare tre och fyra om att hitta vad det egentliga behovet är och lösa det på bästa möjliga sätt.

Förutom dessa insamlingstekniker nämnde alla deltagare prototyper som en bra teknik för att finna krav och behov till system. Prototyper är däremot svårt att enbart kategorisera som en insamlingsteknik då det även kan fungera som ett dokument för att samla upp den information som erhållits genom intervjuer, observationer och så vidare. Prototyper kan däremot fungera som ett konkret underlag för att diskutera krav och behov mellan kunder, användare och utvecklare. Även om det är svårt att enbart klassificera prototyper som en insamlingsteknik nämns det i denna del av arbetet på grund av att respondenterna uttryckte den som en insamlingsteknik.

Respondenternas olika val av insamlingstekniker har säkerligen många anledningar, men troligtvis beror det till väldigt stor del på hur de arbetar och vilka utvecklingsmetoder de använder för att ta fram system. Det är svårt att göra en bedömning av vilka insamlingstekniker som fungerar bättre än andra på grund av att de passar till olika situationer. För till exempel RUP-projekt kommer användningsfall att fungera bättre för att metoden är baserad på att funktionellt ta fram bra system, medan en användarcentrerad metod kommer att lämpa sig för att observera användarna i deras egna kontext. Val av insamlingsteknik beror därför till stor del på metodval och utvecklares erfarenhet av att använda metoden. Att det är svårt att avgöra om någon insamlingsteknik är bättre än någon annan för att de passar till olika situationer är även något som stöds av Robertson (2001). Enligt Robertsson finns det idag ingen insamlingsteknik som är fulländad och olika insamlingstekniker kommer att passa till olika situationer (se kapitel 2.4.2).

5.3 Deltagarnas syn på användbarhetskrav

Åsikterna går isär bland deltagarna när det kommer till användbarhetskrav. Respondent ett förklarar användbarhetskrav som krav som behandlar hur systemet ska användas. Det skulle till exempel kunna vara att ett system ska vara komplext för att användaren ska arbeta med stora pengar där fel är oacceptabelt.

”Det beror på användningen av systemet.” – Deltagare ett.

Deltagaren går vidare med att berätta hur användbarhetskrav ofta kan vara drivande för ett utvecklingsprojekt, till exempel att användaren ska göra så få fel som möjligt. Respondenten har arbetat med flera olika typer av användbarhetskrav som prestationskrav, defektkrav, knapptryckningskrav, designkrav och så vidare. Respondent två har som deltagare ett även arbetat med liknande typer av användbarhetskrav, men även med prototyper eller dialogspecifikationer som ersättning för användbarhetskrav.

Respondent tre har en annan syn på krav och anser att de generellt är något ”onaturligt” för systemutvecklingsprojekt. Enligt deltagaren är krav inget som kan tas fram till ett system, utan vad som finns är behov för användarna och för verksamheten. Detta får även stöd av deltagare fyra som också anser att arbeta med krav är något ”onaturligt”. Deltagare fyra anser vidare att användbarhetskrav som tider eller antal fel begränsar systemutvecklingsprojekt till mätningar som inte behandlar användbarhet.

”Jag är ingen kravmänniska, jag gillar inte krav då de begränsar en mätning till faktorer som inte behandlar användbarhet.” – Deltagare fyra.

Deltagare tre delar samma åsikt som deltagare fyra och ser heller ingen nytta med att specificera användbarhetskrav för att skapa en design. Enligt respondenten kommer analysfasen av utvecklingsarbetet leda till en design av användargränssnittet och en dokumentering av olika användbarhetskrav kommer inte att bistå denna process. Till de utvecklingsprojekt där användbarhetskrav varit involverade gör respondent tre en uppdelning mellan kvantitativa användbarhetskrav och kvalitativa användbarhetskrav. Kvantitativa användbarhetskrav är de krav där mätkriterier dokumenteras för att arbeta med, som tid eller felprocent och kvalitativa användbarhetskrav är de krav som formuleras mer som mål att till exempel ett system ska vara flexibelt. Utav dessa två kravtyper väljer deltagaren tre att helst arbeta med kvalitativa användbarhetskrav och detta motiverar denne med att kvalitativa krav bidrar mer till designprocessen. Enligt deltagare tre är problemet med kvantitativa användbarhetskrav att de är svåra att finna och verksamheten har sällan själva koll på dem. Att däremot ha tillgång till dem och testa mot dem är enklare än att verifiera en design med kvalitativa användbarhetskrav anser deltagare tre. Vidare ser deltagare tre ingen nytta med att dokumentera kvantitativa användbarhetskrav då respondenten ser dem som ett resultat av att uppnå ett kvalitativt användbarhetsmål.

”effektivitet kommer ju av att man snarare slipper stansa om eller slipper skriva ner personnummer på lappar och får utnyttja en massa fönster … Det är ju bara att designa bort så har man tjänat in tiden, sen om det tog 30 sekunder eller två minuter vem vet” – Deltagare tre.

Respondent fem definierar användbarhetskrav som ett krav att kunna mäta någonting.

”då måste det nästan vara att du ska klara en viss sak på viss tid med ett visst antal givna parametrar.” – Deltagare fem

Deltagare fem finner däremot att arbeta med denna typ av användbarhetskrav är väldigt svårt och brukar snarare formulera dem mer som mål, till exempel att en användare ska kunna genomföra någonting utifrån en given situation. Denna ståndpunkt stämmer ganska väl överens med deltagares tre åsikter om att arbeta med kvalitativa användbarhetskrav. Vidare håller även deltagare fem med respondent tre om att det inte alltid finns en nytta med att dokumentera användbarhetskrav för att arbeta med en design.

”Men jag tycker det ger lite för lite att lägga en massa tid på att skapa specifikationer för användbarhetskrav eller så är det jag som aldrig lyckats med det.” – Deltagare fem.

Detta motiverar respondent fem med att pengar alltid kommer att begränsa tiden en utvecklare kan lägga på att arbeta med användbarheten i ett system. Deltagaren berättar att när denne började arbeta med användbarhet var det väldigt viktigt att dokumentera användbarhetskrav och mäta mot dem. Idag är det däremot mycket mer resultatinriktat där deltagaren arbetar efter en best practice, det vill säga på ett arbetssätt som levererar resultat och sedan testar om möjligheten finns.

Av respondenternas åsikter uppfattas användandet av användbarhetskrav något komplext. Även om alla deltagare använt sig av kvantitativa användbarhetskrav så verkar det föredras att arbeta med andra artefakter som prototyper för att uppnå ett användbart system. Frågan bör därför väckas i vilken utsträckning användbarhetskrav behövs dokumenteras för ett system. Till stor del beror det nog på typ och storleken på systemet som byggs. För mindre system där en eller två användbarhetsexperter är involverade för att arbeta med användbarheten kan nog dokumenteringen av användbarhetskrav ifrågasättas. I detta fall vet användbarhetsexperterna vad som

behöver uppnås och behöver inte dokumentera det för att arbeta bättre. För större system där fler utvecklare är involverade för att ta fram användbara system är nog en dokumentation av användbarhetskrav oundviklig för att samtliga utvecklare ska arbeta mot samma mål och vara involverade i processen. Flertalet deltagare verkar föredra att arbeta med kvalitativa användbarhetskrav, detta är förståeligt då de bidrar mer till designprocessen av ett system än kvantitativa användbarhetskrav. Detta beror på att kvalitativa användbarhetskrav berättar mer om systemets karaktär, hur det ska vara uppbyggt, än kvantitativa användbarhetskrav som specificerar vad systemet ska prestera gällande användbarhet.

Deltagarnas delade uppfattning om användbarhetskrav går i linje med olika förespråkares uppfattning om användbarhetskrav (se kapitel 2.3.4). Där förespråkare som Lauesen (2005) menar att de är viktiga för att driva fram arbetet med användbarhen, medan Sommerville och Kotonya (1998) säger att de är överflödiga och evolutionär prototyping är ett bättre angreppssätt. Vidare motsätter sig de respondenter som menar att kvalitativa användbarhetskrav är ett bättre sätt att specificera användbarhetskrav Sommerville (2005) som menar att det är bättre att uttrycka krav mätbara (se kapitel 2.3.3). Sommerville nämner dock att ickefunktionella krav kan vara svåra att omformulera till mätbara och menar att det kan vara bra att kombinera mätbara krav med mål för att guida utvecklarna.

5.4 Deltagarnas syn på användandet av insamlingstekniker för att finna användbarhetskrav

Det är svårt att urskilja att någon av deltagarna väljer att använda speciella insamlingstekniker för att identifiera specifika typer av användbarhetskrav. Av respondent ett uppfattades det som att det är iterationerna i ett utvecklingsprojekt som styr användandet av olika insamlingstekniker, det vill säga ett systemutvecklingsprojekt kommer efter ett visst antal iterationer veta om informationen är komplett nog för att börja med nästa fas eller om informationsinsamlandet måste fortsätta. Deltagaren uppgav inte användning av någon specifik insamlingsteknik i att utföra detta arbete utan tog exempel som att fortsätta med intervjuer eller skapa användningsfall.

Respondent två uttryckte att observationer av användarna är en bra teknik för att skaffa information om användbarheten till ett system. Deltagaren beskrev att observationer är bra att utföra för att observera användarnas snabbkommandon eller identifiera vilket stöd de behöver i sina arbetsprocesser. Vidare nämner respondenten att flera insamlingstekniker kan fungera för att identifiera användbarhetskrav

”Det kan vara att analysera dokument från tidigare system till att se användare interagerar med tidigare system.” – Deltagare två.

Även deltagare tre höll med om att observationer fungerar som en bra insamlingsteknik för att identifiera behoven kring användbarhet. Respondenten förklarar däremot att de mer utför observationsintervjuer där de kan observera men även fråga användarna om deras beteenden. Deltagare tre nämner även arbetsgrupper som en bra insamlingsteknik för att finna användarnas behov. Respondenten förklarar detta med att arbetsgrupper är bra för att det ger utvecklarna tillfälle att lyssna på användarna om vad som är bra respektive dåligt med deras arbete. Detta är även något som även deltagare fyra anser sig hålla med om och menar att fältstudier är den insamlingsteknik som generellt bäst fungerar för att finna användarnas behov. Både deltagare tre och fyra pratar om fältstudier som insamlingstekniker för att hitta krav,

men med fältstudier så menar de inte endast en typ av teknik utan ett flertal som innebär att utvecklare är på plats hos användarna för att utföra informationsinsamlandet. Enligt respondent fyra är fältstudier det enda sättet att få någon förståelse för vad användarna utför och varför.

”För att få förståelse för detta så måste du gå ut på fältet för att observera detta beteende.” – Deltagare fyra.

Respondent fem nämner tre insamlingstekniker för att identifiera användbarhetskrav: intervju, deltagande studie och fokusgrupper. Primärt för deltagaren fungerar intervju, som enligt deltagaren är bra för att få en snabb överblick vad problemet är för något och vad som måste lösas. Intervju kan däremot inte endast fungera som ensam insamlingsteknik, utan måste kompletteras med andra, till exempel att använda sig av fokusgrupper, enligt deltagare fem. Deltagande studie motiverar deltagaren med att det ofta kan vara svårt för användaren att beskriva vad problemet är för utvecklaren. Genom att observera användaren i dess kontext kan utvecklarna överkomma detta genom att observera problemet. Samtliga insamlingstekniker respondent fem nämner samstämmer med åsikterna av respondent tre och fyra om vilka insamlingstekniker som ska användas för att identifiera användarnas behov.

Av respondenternas åsikter är det svårt att se någon direkt koppling mellan någon speciell typ av användbarhetskrav och insamlingsteknik. Alla respondenter kunde däremot avgöra vilka insamlingstekniker som lämpar sig bäst för att identifiera information som berör användbarhet. Generellt uppfattas observation som den insamlingsteknik som lämpar sig bäst för att finna information om användbarhet. Fördelen med observation är att användare inte behöver beskriva vad problemen är, utan en utvecklare kan ganska enkelt observera dem, som deltagare fem beskrev det. Av deltagarnas åsikter uppfattades även att insamlingstekniker måste kombineras för att få en komplett syn på problembilden. Att till exempel kombinera intervjuer med observationer gör att utvecklare får både höra användarens syn på problemet, men även skapa en egen uppfattning om det. Av respondent tre, fyra och fem uppfattades ingen skillnad mellan de insamlingstekniker som används generellt och de som används för användbarhetskrav. Detta kan till stor del bero på att deras roll som användbarhetsexperter inom mjukvaruutveckling har gjort att de är mest bekväma att utföra och att det är användbarhetsfrågor som de främst arbetar med vid framtagning av nya system.

Respondenternas uppfattning om observation som en av de bättre insamlingsteknikerna att finna information kring användbarhet har paralleller till att det är en bra insamlingsteknik att identifiera omedvetna krav (se kapitel 2.4.7). Det är mycket troligt att användarnas kunskaper om användbarhet kring deras system har blivit en sådan del av deras vardag att de inte längre behandlar dem som krav (se kapitel 2.4.1), vilket gör observation till den insamlingsteknik som lämpar sig bäst för att arbeta med användbarhet. Vidare får respondenterna även stöd i deras åsikter om att insamlingstekniker måste kombineras för att fånga hela kravbilden. Faktum är att det inte finns någon insamlingsteknik som idag är fulländad att hitta krav och de kan därför behöva kombineras för att fånga hela kravbilden (se kapitel 2.4.2).

5.5 Slutsatser

Detta arbete handlar om att svara på frågeställningen om specifika insamlingstekniker fördelaktigt kan användas för att identifiera olika aspekter av användbarhet. Arbetet handlar om att göra en vidare utveckling av Lauesen koppling mellan användbarhetskravtyper och aspekt av användbarhet (se figur 2) med

insamlingstekniker, till exempel att intervjuer passar till system som främst måste rikta in sig på lärbarhet.

Resultatet av den litteratur som utforskades till arbetet och intervjuerna visade sig något motsägande. Av litteraturen visade det sig hur viktigt det är för systemutvecklingsprojekt att specificera användbarhetskrav för att ett system ska uppnå en viss nivå av användbarhet. Intervjuerna till arbetet visade däremot komplikationerna med att specificera användbarhetskrav till system och att flertalet respondenter väljer att helst inte arbeta med dem. Flertalet respondenter uppger däremot att användbarhetskrav har varit involverade i olika systemutvecklingsprojekt, men då oftast med krav från kunden. Tillvägagångssätten att finna användbarhetskrav verkar däremot inte baseras på typen av användbarhetskravtyp, utan av vilka insamlingstekniker som generellt lämpar sig till att identifiera information om användbarhet. Av respondenternas svar uppfattades observation som den mest lämpade insamlingsteknik för att behandla användbarhet, vilket är förståeligt om användarnas kunskaper om användbarhet är ofta omedveten och observation är en av de bättre insamlingsteknikerna att identifiera omedvetna krav (se kapitel 2.4.1 och 2.4.7).

Slutsatsen av detta arbete är att det inte finns någon direkt koppling mellan val av insamlingsteknik och aspekt av användbarhet. Detta motiveras med att respondenterna inte baserar valet av insamlingsteknik på typ av användbarhetskrav som måste identifieras, utan baserar valet på tre faktorer:

- Vilka insamlingstekniker som generellt passar för att samla information om användbarhet.
- Utvecklarens erfarenhet att använda insamlingstekniken.
- Vad som är möjligt att genomföra inom utvecklingsprojektets ramar angående tid, budget och tillgång till användare.

Vad som kan fastställas efter detta arbete är att det finns insamlingstekniker som passar för att identifiera information om användbarhet, så som observationer, men inte insamlingstekniker som kan fördelaktigt användas för att identifiera olika aspekter av användbarhet.

6 Diskussion

Denna del av rapporten är avsedd för att föra en diskussion kring arbetet som utförts. Diskussionen kommer att föras kring tre ämnen tillvägagångssättet som genomfördes, resultatet av intervjuerna och fortsatt arbete som skulle kunna utföras med arbetet som utgångspunkt.

6.1 Diskussion kring tillvägagångssätt

Valet av metod föll ganska naturligt på en kvalitativ studie där djup och detalj var prioriterat framför kvantitet. Metodvalet möjliggjorde en jämförelse mellan genomgången litteratur och resultatet av intervjuerna som senare kunde presenteras i analys och slutsatsdel. Metoden hade kunnat genomföras med till exempel enkäter, men hade försvårat arbetet med analysen då resultat skulle ha blivit för ytligt för att kunna fördjupa sig i en analys mellan enkätsvaren och genomgången litteratur (se kapitel 4.1).

Litteraturen som granskades parallellt med att bakgrunden skrevs var något begränsad inom området. Mycket litteratur om systemutveckling och arbete med användbarhet generellt finns att tillgå, men väldigt lite om hur utvecklare ska gå tillväga för att identifiera användbarhetskrav. Bakgrunden som skrevs till arbetet är därför en blandning av traditionell systemutveckling och olika böcker skrivna om användbarhet.

Det som har varit mest intressant att utföra i detta arbete var att utföra de olika intervjuerna av utvecklare, om hur arbetet kring användbarhet går till. Samtliga respondenterna var mycket engagerade och välvilliga till att svara på de frågor som ställdes. Antalet intervjuer kunde däremot ha varit fler för att täcka en större grupp av utvecklare som arbetat med användbarhetskrav. Att till exempel inkludera utvecklare som inte arbetar som konsulter hade varit givande för analysen för att täcka ett större analysområde. På grund av tidsbegränsningar fick däremot antalet intervjuer hållas nere till fem för att inte insamlad data skulle vara för överväldigande att utföra en analys på.

Flertalet intervjuer gick bra att utföra. På en av intervjuerna uppstod däremot vissa tekniska problem som ledde till en ganska otydlig inspelning. För att samla ihop så mycket information som möjligt av den dåliga inspelningen började transkriberingen med denna intervju, medan den var färskt i minnet. Vidare transkriberades övriga intervjuer utan några större problem.

Kritik kan riktas mot utförandet av analysen som kan ha blivit något präglat av mina egna åsikter kring användbarhetskrav. Även om det är svårt att bortse från detta är det viktigt att noteras. Vidare är det inte heller alltid lätt att uppfatta vad respondenternas olika ståndpunkter är kring ämnena som analyserades i resultatsdelen, på grund av att svaren måste tolkas.

6.2 Diskussion kring resultat

Det förväntade resultatet av att utföra studien var att se en koppling mellan val av insamlingsteknik och aspekt av användbarhet. Resultatet visade däremot ingen koppling, men ändå på ett intressant resultat. Motsats till vad litteraturen förespråkar om hur viktigt det är att dokumentera användbarhetskrav för ett system så uppfattade respondent tre, fyra och fem svårigheter med att arbeta med användbarhetskrav. Dessa respondenter, som arbetar som användbarhetsexperter, menade att tiden det tar att identifiera och formulera användbarhetskrav motsvaras inte i form av en bättre design.

Diskussionen grundar sig på om användbarhetskrav ska formuleras kvantitativt eller kvalitativt, där de som riktade kritiken var för kvalitativa användbarhetskrav. Kvantitativa användbarhetskrav kan beskrivas som krav som dokumenteras med mätkriterier som till exempel tid eller felprocent och kvalitativa användbarhetskrav är de som dokumenteras mer som mål, till exempel att ett system ska vara flexibelt. Men att avgöra vilken typ av användbarhetskrav som är bäst är svårt. Kritiken respondenterna riktade mot kvantitativa användbarhetskrav är troligtvis befogad, men lika mycket kritik kan troligtvis riktas mot kvalitativa användbarhetskrav. De viktiga i debatten om hur de ska formuleras är att inse att de tjänar olika syften i att ta fram användbara system. Kvalitativa användbarhetskrav är bra för att de kan beskriva karaktären av ett system, till exempel att det ska vara flexibelt. Kvantitativa användbarhetskrav konkretiserar däremot användbarhetsmål med att ta fram mätkriterier som kan validera system. Fördelen med att arbeta med denna typ är att det skapar en medvetenhet hos alla involverade i att ta fram en mjukvara över vad som är viktigt att uppnå när det kommer till användbarhet. Kritik kan därför riktas mot förespråkare som Lauesen att kvantitativa användbarhetskrav inte bidrar speciellt mycket till skapandet av en design (se kapitel 2.3.4). Kritik kan även riktas mot respondenterna att kvantitativa användbarhetskrav behövs för att verifiera och skapa ett större fokus på användbarheten av övriga utvecklare involverade att ta fram ett system (se kapitel 2.3.3). Det viktigaste i valet mellan att formulera användbarhetskrav kvantitativt eller kvalitativt är att se att de kompletterar varandra och ska helst kombineras för att uppnå maximal användbarhet.

Av intervjuerna att döma så uppstår det ofta konflikter mellan utvecklare som arbetar med användbarhet och övriga utvecklare. Av respondenternas svar identifierades två orsaker till att konflikter uppstår**:** att användbarhetsexperter ignoreras av andra utvecklare och att användbarhetsexperter inte tillåter andra utvecklare bidra till designen av ett gränssnitt. Lösningen är däremot gemensam för de båda konflikterna, att öppna arbetet med användbarheten genom att involvera andra utvecklare än bara användbarhetsexperter. Detta innebär till stor del att anpassa arbetet med användbarheten efter mer traditionell systemutveckling som är kravbaserad. Genom att till exempel använda sig av kvantitativa användbarhetskrav kan övriga utvecklare mer påtagligt uppfatta vilka krav systemet måste uppnå och på så sätt känna sig mer delaktiga i att ta fram en design. Arbetet skulle föra samman användbarhetsexperter med andra systemutvecklare och minska risken för att konflikter ska uppstå i systemutvecklingsprojekten.

Vidare uttryckte en av respondenterna problemet med att utvecklare gärna ser till sina egna behov att utveckla system, att systemet uppnår just den aspekt utvecklaren arbetar med. Detta relaterar mycket till det som diskuterades innan om att det uppstår konflikter mellan användbarhetsexperter och systemutvecklare, men tar ett mer generellt perspektiv. Lösningen på detta är att utvecklare måste inse att leverera en mjukvara innebär att uppnå ett gemensamt mål av kvalité. Detta är speciellt viktigt för att system ska uppnå de ickefunktionella krav (se figur 1), då det är kvalitéer som systemet uppvisar som helhet, det vill säga uppstår, när delar av mjukvaran sätts ihop för att utgöra en helhet. För att uppnå dessa krav, där användbarhet berörs, är det viktigt att utvecklare inser att ta fram en mjukvara innebär att gemensamt uppnå mål och inte se till sina egna behov.

6.3 Förslag på fortsatt arbete

Med detta arbete som utgångspunkt har idéer väckts om hur ett fortsatt arbete skulle kunna utföras. Ett förslag till fortsatta studier skulle vara att utöka antalet intervjuer

för att täcka en större grupp av utvecklare. Till detta arbete intervjuades det två typer av utvecklare, de som arbetar med systemarkitekturen av ett system och användbarhetsexperter. Gemensamt för respondenterna var att de arbetade som konsulter och var därför tvungna att anpassa sig efter kund om utvecklingsmetoder och så vidare. Att till exempel utöka intervjuerna för att täcka de utvecklare som inte arbetar som konsulter skulle göra att en mer komplett bild skulle kunna skapas över hur arbetet med användbarhetskrav går till.

Vidare skulle det även vara intressant att utföra studien som etnografisk inspirerad studie där användandet av olika insamlingstekniker kunde observeras för att skapa en egen uppfattning hur de relaterar för att hitta användbarhetskrav. Av intervjuer är det svårt att bilda en egen uppfattning och det skulle vara intressant om hur resultatet skulle skilja sig.

Intressant skulle även vara att utföra en studie om skillnaderna mellan kvalitativa och kvantitativa användbarhetskrav. Hur de skiljer sig åt att skapa en design eller hur de skiljer sig åt att motivera arbetet med användbarhet.

För att utreda vilka situationer som passar vilka tillvägagångssätt skulle det även vara intressant att undersöka skillnaderna med att arbeta med prototyper respektive användbarhetskrav för att uppnå användbarhet.

Referenser

Augoustinos, M & Walker, I. (1995) *Social cognition - an integrated introduction.* London: Sage Publications.

Berndtsson, M., Hansson, J., Olsson, B. & Lundell, B. (2002) *Planning and Implementing your Final Year Project - with Success!: A Guide for Students in Computer Science and Information Systems.* London: Springer

Easterbrook, S. & Nuseibeh, B. (2000) Requirements Engineering:A Roadmap, I: A. Finkelstein (red), *The Future of Software Engineering* (s. 35-46). Twenty Second International Conference on Software Engineering (ICSE 2000), Juni, 2000, Limerick, Ireland.

Eysenck, M.W. & Keane, M.T. (2000) *Cognitive Psychology: A student's handbook.* Hove, East Sussex: Psychology Press Ltd.

Cysneiros, L. M., Leite, J. C. S. P. & Neto, J. S. M. (2001) A Framework for Integrating Non-Functional Requirements into Conceptual Models. *Requirements Engineering Journal*, Vol 6, Issue 2, 97-115.

Gulliksen, J. & Göransson, B. (2002) *Användarcentrerad systemdesign*. Lund: Studentlitteratur.

Goguen, J. & Linde, C. (1993). Techniques for Requirements Elicitation. I: S. Fickas & A. Finkelstein (red:er), *Requirements Engineering* (s.152-164). 1st IEEE International Symposium on Requirements Engineering, 4-6 January, 1993, San Diego, USA.

Kotonya, G. & Sommerville, I. (1998) *Requirements Engineering: Processes and Techniques.* Chichester: John Wiley & Sons.

Lauesen, S. (2002) *Software Requirements: Styles and Techniques*. Harlow: Addison-Wessly.

Lauesen, S. (2005) *User Interface Design, A Software Engineering Perspective.* Harlow: Pearson.

Lauesen, S. & Younessi, H. (1998) Six styles for usability requirements. I: E. Dubois, A. L. Opdahl & K. Pohl (red:er): *Proceedings of REFSQ'98*, Presses Universitaires de Namur, 1998.

Patton, M. (2002) *Qualitative Research & Evaluation Methods* (3:e upplagan). London: Sage Publications.

Preece, J., Rogers, Y. & Sharp, H. (2002) *Interaction design – beyond human-computer interaction.* New York: John Wiley & Sons, Inc.

Robertson, J. & Robertson, S. (1999) *Mastering the Requirements Process.* Harlow: Addison-Wesley.

Robertson, S. (2001) Requirements trawling: techniques for discovering requirements. *International Journal of Human-Computer Studies, 55*, s. 405-421.

Standishgroup (1994) *The Chaos Report.* The Standish Group. Tillgänglig på Internet: http://www.standishgroup.com/sample_research/chaos_1994_1. [Hämtad 07.03.13]

Sommerville, I. (2007) *Software engineering (8:e upplagan)*. Harlow: Addison-Wesley.

Bilaga I – Intervjuguide

Bakgrund

1. Hur skulle du vilja beskriva ditt yrke?
    - Vad arbetar du med
    - Hur länge du arbetat med det, detta företag, andra företag
    - Har du haft andra roller inom samma kategori av yrke

2. Vilka typer av system har du utvecklat? Inom vilka användningsområden?
    - Exempel

3. Projektets respektive karaktär (beskrivning av projektet)
    - Vad skulle du säga var storlek på projektet (systemet)?
    - Hur många var involverade?
    - Hur långt var projektet?

4. Vad för systemutvecklingsmodell användes på projekten
    - Vilka metoder har använts i respektive projekt?
    - I vilken utsträckning användes metoden?
    - Hur många år har metoden använts i organisationen?

    Om metod saknas:
    - Vilken arbetsprocess tillämpas?
    - Arbetsprocess, egen utvecklad metod?
    - Beskrivning av arbetssätt (processmetoden)?

Insamlingstekniker

5. En viktig del inom systemutveckling är att hitta krav, vilka insamlingstekniker används inom organisationen för att identifiera krav till system? (use-cases, intervju, observation)
    - Beskriv dem?
    - Motivera varför?

6. Anser du att någon av dessa tekniker kan fördelaktigt användas för att hitta någon speciell typ av krav?
    - Funktionella krav?
    - Någon typ av ickefunktionella krav?
    - Exempel

Syn på användbarhet

7. Vad är din definition av användbarhet?

8. Hur ser er nuvarande organisation på användbarhet?
    - Finns det någon dokumenterad förhållningspunkt?

9. Hur viktigt är det att dokumentera en produkts användbarhet för dig?

10. Vilka tillvägagångssätt används inom organisationen för att uppnå god användbarhet för ett system?
    - Prototyper
    - Användbarhetskrav
    - Evolutionär prototyping
    - Användarmedverkan
    - Annat

11. Hur kommer det sig att ni väljer använda dessa (svar 10)?
    - Fördelar

Användbarhetskrav/insamlinstekniker

12. Hur ser din definition ut på användbarhetskrav?

13. Hur ser organisationen på användbarhetskrav?
    - Stort/litet fokus

14. Hur dokumenteras krav som berör användbarheten för en produkt?
    - Tid för utförande
    - Antal fel vid utförande
    - Riktlinjer för design
    - Antal tangenttryckningar vid utförande
    - Förståelsebetyg vid utfrågning
    - Opinionsundersökningar,
    - Processkrav, hur designen ska gå till.
    - Designkrav, prototyp hur systemet ska se ut (styleguides)

15. Finns det insamlingstekniker att föredra för att hitta information som berör dessa krav?
    - Insamlingstekniker att föredra att hitta designriktlinjer, tider för utförande av uppgifter, mätkriterier för opinionsundersökningar och så vidare.

16. Hur tillämpas dessa krav i utvecklingsprojekten?
    - När kommer de in i processen
    - Vad har de för syfte i utvecklingsprojekten
    - Vad är det förväntade resultat

17. Vad anser du är de positiva egenskaperna med att arbeta med användbarhetskrav?

18. Vad anser du är de negativa egenskaperna med att arbeta med användbarhetskrav?

19. Vilken koppling anser ni finns mellan användbarhetskrav och uppnå god användbarhet för ett system?
    - Hur tror ni användbarhetskrav påverkar användbarheten för system
    - Motivera

    Om användbarhetskrav inte används:
    - Vad var anledningen till att användbarhetskrav inte medverkar i utvecklingsprojekten?

    Något övrigt som ni vill tillägga innan intervjun avslutas?

TACK!

Bilaga II –Användbarhet enligt ISO 9241-11

“den utsträckning till vilken en specificerad användare kan använda en produkt för att uppnå specifika mål, med ändamålsenlighet, effektivitet och tillfredsställelse, i ett givet användarsammanhang.” (ISO 9241-11, 1998)

Ändamålsenlighet definieras som:

”noggrannhet och fullständighet med vilken användarna uppnår givna mål.” (ISO 9241-11, 1998)

Effektivitet definieras som:

”resursåtgång i förhållande till den noggrannhet och fullständighet med vilken användarna uppnår givna mål.” (ISO 9241-11, 1998)

Användningssammanhanget definieras som:

”användare, uppgifter, utrustning (maskinvara, programvara och annan materiel) samt fysisk och social omgivning i vilken produkten används.” (ISO 9241-11, 1998)

(Gulliksen & Göransson, 2002)