Kravvalidering
- ur ett kommunikationsperspektiv

(HS-IDA-EA-98-308)

Anders Isacson(a95andis@ida.his.se)

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

Examensarbete på det Systemvetenskapliga programmet under vårterminen 1998.

Handledare: Beatrice Alenjung

KRAVVALIDERING

- ur ett kommunikationsperspektiv

Examensrapport inlämnad av Anders Isacson till Högskolan i Skövde, för Kandidatexamen (BSc) vid Instutitionen för Datavetenskap.

980518

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:_____

Kravvalidering
- ur ett kommunikationsperspektiv

Anders Isacson (a95andis@ida.his.se)

Key words: Validation, communication, user-participation

Abstract

The communication between user and developer in the validationprocess depends on many things. The language used, the attitude of the developer and user and the method used to name a few. In this paper the cooperation between user and developer is reviewed. The problems in the communication between user and developer consists of many factors. The social part is one big reason why communication fails in many projects. Another is the method used, and how the developer choose to use it. This paper investigates a few methods and compare them against each other to see if one method is better with handling user-participation then the rest and if so, why?

Sammanfattning

Det här arbetet börjar med en introduktion till systemutveckling och ger exempel på ett par olika tillvägagångssätt som används i Skandinavien och England. I introduktionen presenteras också problem som har funnits och kanske finns fortfarande i systemutvecklingsbranschen. Vidare belyses också hur viktig kommunikationen är mellan användare och utvecklare genom hela utvecklingsarbetet. I de tidiga faserna av systemutveckling finner man kravhantering och den introduceras kort i slutet på introduktionen för att sedan presenteras i fullo i bakgrundskapitlet. I början på bakgrundskapitlet ges ett exempel på hur en fullständig utvecklingsmodell kan vara uppbyggd och efter det tas kravhanteringens alla delar upp för att sedan avslutas i hur kravvalideringen fungerar och används. Problembeskrivningen innehåller tre delar med början på det problemområde som jag valt, nämligen kravhanteringsprocessen. Denna följs av avgränsningen i form av valideringen, och då speciellt kommunikationen i valideringsprocessen som fungerar som problemprecisering.
Metodvalet visar på valet av metod samt de källor som använts med tillhörande bakgrundsinformation. Genomförandet presenterar en analys olika metodkategorier inom valideringsprocessen och undersöker dem med hjälp av tre kriterier för att få en rättvis jämförelse. I genomförandet finns också en översikt om användarmedverkan eftersom detta är en förutsättning för kommunikation mellan två parter. Sedan följer ett analyskapitel där metodkategorierna analyseras ur två perspektiv. I analyskapitlet presenteras också de erfarenheter som gjorts under detta arbete. En analys av materialet som använts sker också i samma kapitel. Till sist presenteras resultat av undersökningen och arbetet diskuteras i ett vidare perspektiv.

1. Introduktion

Utvecklingen på alla områden i Sverige går idag framåt med stor hastighet. Utvecklingen av avlastande och effektiviserande datasystem är inget undantag. För att nå mål som effektivitet och kvalitet investerar många organisationer i någon form av datasystem. Datasystem omfattar allt från administrativa hjälpsystem till avancerade produktionssystem inom industrin. I denna dynamiska situation är systemutveckling ett område som växer och utvecklas konstant. I det här arbetet kommer jag att benämna centrala personer, systemutvecklaren eller kort utvecklaren samt användaren, som han. Systemutveckling bedrivs givetvis av kvinnor också men att skriva han/hon genom hela arbetet blir för omständigt och jag har därför valt att skriva han.

1.1 Vad är systemutveckling?

Systemutvecklingen har funnits ganska länge men användningen av systemutvecklingen har accelererat mer på senare tid. Systemutveckling är den process som utförs hos en organisation som vill förbättra (rationalisera, effektivisera m.m.) sina manuella och automatiska rutiner. Dessa rutiner omfattar allt från ren administrativ behandling hos t.ex. en kommun till övervakningsrutiner på en fabrik. Systemutveckling sträcker sig från analysen av ett behov till implementeringen av ett nytt system. Systemutvecklaren har till uppgift att ta reda på vad kunden behöver. Sedan skall han med hjälp av diverse modeller och metoder ta fram ett lösningsförslag och förhoppningsvis få detta implementerat med hjälp av mjuk och hårdvarutekniker. Nya metoder, modeller och verktyg för detta ändamål dyker ständigt upp på marknaden.

1.2 Tillvägagångssätt

Systemutveckling kan bedrivas på många olika sätt. Hur systemutvecklingen bedrivs styrs av vilka modeller och metoder som används. Skillnaden mellan modeller och metoder är att modeller vanligtvis täcker hela utvecklingsarbetet medan metoderna enbart täcker delar av utvecklingsarbetet. Vissa modeller som t.ex. livscykelmodellen, Andersen (1994), förespråkar mycket användarmedverkan och andra som t.ex. SSADM (används mycket i England) använder nästan uteslutande av experter. Eftersom det skandinaviska perspektivet på systemutveckling förespråkar mycket användarmedverkan så har jag använt livscykelmodellen i min förklaring som kommer i bakgrundskapitlet. Systemutveckling har en mängd olika arbetsområden som oftast börjar med någon form av analys över nuläget och slutar med implementering av det nya systemet. Det finns även två andra områden, underhåll och drift samt avveckling, men dessa två kommer oftast till när själva utvecklingsarbetet är klart.

1.3 Problem

Enligt Persson (1995) har trenden varit att av tidigare system som tillverkats har en stor del visat brister. Brister innebär i det här avseendet för komplicerade för användarna, ej fullt funktionella, vissa egenskaper saknas etc. Av dessa tidigare system (informationssystem, datasystem etc) har endast en bråkdel varit direkt funktionsdugliga, en del har man kunnat rätta till tillfredsställande men många har blivit totalt oanvända. Detta kan naturligtvis bero på en mängd olika saker men ett av problemen har varit dålig kommunikation mellan beställare och leverantör. Med dålig kommunikation menar jag inte att parterna inte pratat med varandra utan att de pratat utan att förstå vad den andra parten egentligen menar. Detta har fått många att

inse vikten med systemutveckling. Bristen tidigare har varit att kunden inte har pratat samma språk som teknikerna och därför inte kunnat förmedla sina önskemål. Systemutvecklaren skall här fungera som en översättare med en fot i varje läger.

1.4 Kommunikation i utvecklingsarbetet

Kommunikationen är som sagt väldigt viktig i utvecklingsarbetet, i synnerhet de första delarna av utvecklingsarbetet. Det gäller att från början etablera en gemensam begreppsram som både kund och leverantör är nöjda med. Detta görs för att inga tvetydigheter eller missförstånd skall uppkomma vid kommunikationen mellan de bägge parterna. Vid kravhanteringen, som ligger tidigt i utvecklingsprocessen, är kanske kommunikationen viktigare än någonsin därför att det är här som grunden till det nya systemet läggs. Här skall kunden eller användarna av systemet förklara för systemutvecklaren vad de vill ha och hur det ska fungera. En miss i kommunikationen här kan ge mycket allvarliga konsekvenser. Systemutvecklaren kan med helt felaktig bakgrundsinformation designa ett system och detta kan bli ödesdigert för hela utvecklingsprojektet. Det finns dock en mängd olika sätt att kringgå detta oönskade problem. Man kan använda en rad olika metoder, allt från intervjuer till grafiska beskrivningar.

1.5 Kravhantering

Kravhantering är det begrepp som sträcker sig från kravframtagning, genom kravspecificeringen och slutar i kravvalideringsprocessen. Kravhanteringen resulterar förhoppningsvis i en kravspecifikation som både beställare och leverantör är nöjda med. Kravspecifikationen är det dokument som står som garanti för att beställaren får det han vill ha, men också en garanti för beställaren så att han kan bevisa att han utfört det som krävts av honom.

2. Bakgrund

2.1 Systemutveckling

Systemutveckling är process som kan se ut på många olika sätt. Det finns en mängd olika modeller och metoder för att genomföra en systemutveckling men de flesta har en hel del gemensamt. Jag har i detta arbete valt att inrikta mig på en fas som ligger ganska tidigt i utvecklingsprocessen, nämligen kravspecifikationen. För att ge en bild utav hur systemutveckling bedrivs och vart mitt ämnesval (kravspecifikationen) kommer in i bilden, har jag valt att göra en genomgång av hur ett systemutvecklingsprojekt kan se ut, enligt Andersen (1994), och vilken funktion kravspecifikationen fyller. Flera modeller borde presenteras för att få mer generalitet, men eftersom denna presentation enbart har till uppgift att få läsaren insatt i problemdomänen använder jag bara en modell.

2.2 Livscykelmodellen

Livscykelmodellen är ett sätt att se på systemutveckling. Detta perspektiv förespråkar mycket användarmedverkan vilket är typiskt för det skandinaviska perspektivet på systemutveckling. Modellen innehåller sju områden från början till slut. Jag kommer kortfattat att beskriva dessa områden i den följd som de tillämpas i utvecklingsprocessen.

Förändrings analys	Analys	Utformning	Realisering	Implementering	Förvaltning och drift	Avveckling

Figur 1

I figur 1 beskrivs de olika områdena grafiskt i den ordning de tillämpas i utvecklingsarbetet. Det skall dock tilläggas att de två sista områdena inte hör till själva utvecklingsarbetet utan äger rum efter detta. Här nedan följer en kort beskrivning av varje område:

- Förändringsanalys

Område ett innefattar att med verksamhetens problem och möjligheter som grund beskriva nuläget och den situation man önskar hamna i efter avslutat utvecklingsarbete. Man analyserar fram vilka förändringsbehov som finns samt tar fram alternativa åtgärder för att täcka dessa behov.

- Analys

Det här området är uppdelat i två delar, verksamhetsanalys och informationssystemanalys. I verksamhetsanalysen undersöker man hur det nya systemet skall stödja verksamheten. Man gör en analys av verksamheten för att på så sätt ta fram hur det nya systemet kan underlätta verksamheten. Informationsanalysen går ut på att bestämma vad informationssystemet skall innehålla.

- Utformning

Detta område består också av två delar, dels principiell utformning av teknisk lösning men också ett lösningsförslag på utformning av utrustningsanpassad teknisk lösning. Detta kan

tyckas rörigt men först tar man fram en abstrakt lösning för att sedan konkretisera och komma med en detaljlösning.

- Realisering

Här utarbetas ADB-program samt nya manuella rutiner. Det är inte enbart nya tekniska lösningar som tas fram under utvecklingsarbetet.

- Implementering

Här tas det nya systemet i bruk och man börjar även med de nya manuella rutinerna som man tagit fram.

- Förvaltning och drift

Här görs korrigeringar och man ser över systemet ut i fall det finns ett behov av förbättringar.

- Avveckling

Detta område kan vara kopplat till starten av ett nytt systemutvecklingprojekt. Man säkrar här information och gör beskrivningar av data och databaser.

2.5 Kravspecifikation

Kravspecifikationen är länken mellan analysfasen och utformningsfasen i livscykelmodellen. Efter att område ett och två har passerats får man en kravspecifikation som resultat. Denna kravspecifikation är ett dokument som innehåller alla de krav som kunden och leverantören gemensamt tagit fram. Här nedan presenteras ett prov på hur en kravspecifikation kan vara utformad och vad den kan innehålla. Variationer kan givetvis förekomma för olika system. Kravspecifikationer finns för mer system än just informationssystem men jag har valt denna sort då den ligger närmast min utbildning och ger mest för min framtida verksamhet.
Det finns en mängd olika kategorier av krav som alla hjälper till att göra kravspecifikationen komplett. Jag kommer att presentera vad en kravspecifikation kan innehålla och hur den kan se ut enligt Andersen (1994).

1. Avsikten med informationssystemet

Här beskrivs vilka mål man vill uppnå samt vilka vinster man räknar med att det nya systemet skall ge. Detta kan t ex vara ökad effektivitet, förenklad administrativ hantering eller att få ett styrande produktionssystem (MPS).

2. En övergripande beskrivning av informationssystemet

En kort övergripande beskrivning av informationssystemet där det viktigaste är att få fram vilka informationssystemets intressenter är, och vilken utväxling av information mellan informationssystemet och intressenterna som sker.

3. Organisatoriska och personliga förutsättningar

En beskrivning av de åtgärder av organisatoriskt och personalmässigt slag, som för att målen ska kunna uppnås, måste vidtas parallellt med utvecklingen av informationssystemet.

4. Informationssystemets generella funktioner

En beskrivning av egenskapskrav som gäller generellt för informationssystemet:

- Tillgänglighet - åtkomsten till systemet måste vara snabb och enkel.

- Användbarhet - användarna skall kunna utnyttja systemets alla funktioner och finesser fullt ut för maximal effektivitet.
- Säkerhet - systemet skall vara skyddat utifrån, obehöriga har ej tillträde till material de ej behöver.
- Kvalitet - systemet skall fungera och utföra det som det är avsett för.
- Utvecklingsmöjligheter - om utvecklingen kräver skall tillägg kunna göras på systemet för att täcka nya (realistiska) behov.

5. Funktionernas egenskaper
- Frekvens och spridning av rapporter - vilka rapporter skall systemet producera och var någonstans de blir tillgängliga
- Svarstider vid interaktiv databehandling - hur snabbt skall systemet utföra vissa funktioner.
- Kapacitet - vilka volymmässiga belastningar som ska klaras

6. Manuella funktioner
De manuella funktionerna kan beskrivas i linje med de som ska automatiseras. Manuella funktioner är de rutiner som utförs utan datastöd. För att försäkra sig om att de manuella rutinerna får tillräckligt med uppmärksamhet kan man beskriva dem funktionellt och egenskapsmässigt för sig.

7. Dokumentation
En beskrivning på kraven av dokumentation:
- (Information)systemdokumentation - det vill säga den allmänna dokumentationen av informationssystemet som också beskriver den tekniska lösningen.
- Användardokumentation - vad som skall dokumenteras för att hjälpa användarna i deras arbete.
- Driftsdokumentation - dokumentation under drift t ex störningar, vad som fungerar tillfredsställande och vad som fungerar som det ska.

8. Utbildning
En beskrivning av kraven på utbildning.

2.6 Definition av krav

Definitionen av de olika sorters krav som Loucopoulos och Karakostas (1995) presenterar är hämtade ur IEEE ‘610’(1990) och definierar krav på följande sätt:

1. ” *Ett tillstånd eller kapacitet som krävs för att en användare skall kunna lösa ett problem eller uppnå ett mål.* ”
2. ” Ett tillstånd eller kompetens som ett system eller systemkomponent måste inneha för att uppfylla ett kontrakt, standard, specifikation eller annat formellt dokument. ”
3. ” En dokumenterad representation av ett tillstånd eller kapacitet som i 1 och 2. ”

Krav kan i allmänhet delas upp i två kategorier: marknadsdrivna och kundspecifika krav. Marknadsdrivna krav härstammar från projekt som utförs utan att ha en speciell kund. Med detta menas att produkten kan vara avsedd för ett marknadsbehov och inte enbart för ett företag eller organisation. De kundspecifika är enbart avsedda för en kund. Detta arbete kommer att koncentrera sig på kundspecifika krav. Det finns många olika sätt att se på kravhantering. För att få en komplett kravspecifikation måste funktionella krav integreras med

icke-funktionella och företagsmässiga krav. Dessa tre kravkategorier ger tillsammans en bred förståelse för systemet i helhet. Ett funktionellt krav beskriver vad systemet skall uträtta. Ett icke-funktionellt krav beskriver t ex säkerhet, användbarhet, funktionalitet eller prestanda. Företagsmässiga krav ställer krav på att systemet jobbar i linje med företagets affärsplaner, att det kan utvecklas med företaget i framtiden etc.

2.7 Olika perspektiv

Det finns flera olika perspektiv när det gäller kravhanteringsprocessen. Två relevanta perspektiv tar Andersen (1994) upp och de är organisationsperspektivet och mjukvaruperspektivet. Dessa två perspektiv visar på ett flerdimensionellt tänkande vid kravhanteringsprocessen.

- **Organisationsperspektivet**

Organisationsperspektivet tar upp de mål och krav som organisationen ställer på det nya systemet. Det nya systemet skall jobba integrerat med organisationen inte som en separat del. Funktionerna på det nya systemet skall inte enbart vara anpassade för endast en funktion utan om möjligt gå i linje med organisationens mål och framtidsplaner. Tidigare har systemen endast automatiserat manuella processer. Nu vill man att organisationen skall utvecklas tillsammans med det nya systemet. Mål som tidigare har varit omöjliga att nå kan nås med hjälp av det nya systemet. Detta innebär att utvecklingen leder till mer än en prestandaökning inom den administrativa behandlingen.

- **Mjukvaruperspektivet**

När mjukvaruutveckling växte fram som en ingenjörsgren förstod man värdet med en pålitlig kravspecifikation. Tidigare hade alltför många utvecklingsprojekt gått under och de som klarat sig fick dras med stora underhållskostnader. Med en korrekt kravspecifikation kunde man undvika kostsamma ändringar sent i projektet. En ändring tidigt i projektet behövde inte få stora finansiella påföljder. Därför växte även kravhanteringen fram som en följd av mjukvaruutvecklingen.

Kravhanteringen utvecklades för att få en heltäckande, konsistent och formell kravspecifikation. Med konsistent menas att den inte innehåller krav som är motsägelsefulla. Motsägelsefulla krav kan t ex vara att systemet skall vara lättillgängligt och samtidigt vara utrustat med en kraftig säkerhetskontroll. När en säkerhetsåtgärd införs brukar systemet mista en del av sin lättillgänglighet. Detta är oundvikligt. Att kravspecifikationen är formell innebär att den är entydig. Om kravspecifikationen tas fram med en formell modell undviks risken för tvetydigheter och misstolkningar. Detta kan dock innebära en avsevärd skillnad i samarbetet med slutanvändare som kan ha svårt att förstå formella modeller.

2.8 Kravhantering

Denna process har jag beskrivit enligt Loucopoulos och Karakostas (1995). I denna bok består kravhanteringen av ett arbete i tre faser, detta för att uppnå en heltäckande, konsistent och formell kravspecifikation. Dessa tre är kravframtagning, kravspecificering och kravvalidering. Under dessa tre processer är kommunikationen extremt viktig eftersom det är här som grunden läggs för det nya system som skall utvecklas.

2.8.1 Kravframtagning

I den här processen tar man fram de olika krav som ligger till grund för det nya systemet. Målet med den här processen är att skaffa sig tillräcklig förståelse för den omgivning som det nya systemet skall verka i. Utvecklaren måste skaffa sig en klar bild av vad systemet skall göra, hur det skall gå till och hur systemet integreras med omgivningen. Problemet här är att informationen måste tas fram och sammanställas för att den skall vara användbar. När man analyserar fram de krav som skall ställas på det nya systemet gäller det att utvecklare och användare pratar samma språk. Med samma språk menar jag inte svenska eller engelska utan att man har en gemensam begreppsram. En gemensam begreppsram förhindrar att man pratar om olika saker med samma benämning, ett ord har inte nödvändigtvis samma innebörd för två personer. När man jobbar med kravframtagning finns det en mängd olika sätt att gå tillväga på och de skiljer sig markant från varandra. När man jobbar med användare i denna process bör utvecklaren betänka följande problem:

- Användare har kanske inte en klar uppfattning om vad de kräver av det nya systemet.
- Användarna kan ha svårt att förmedla sin kunskap om problemdomänen till utvecklaren.
- Kunskapen hos användarna och utvecklaren skiljer sig, de använder inte samma begreppsram.
- Användarna vill kanske inte använda ett nytt system och kan därför vara motvilliga till samarbete.

För att överbrygga dessa problem har ett antal tekniker skapats för att möjliggöra kommunikation mellan användare och utvecklare, och därmed förenkla kunskapsöverföringen mellan dessa parter.
De tekniker som jag valt att beskriva med hjälp av Loucopoulos och Karakostas (1995) är följande :

Intervjuer finns i flera former och den enklaste kallas öppen intervju och går ut på att användaren själv får prata om sin uppgift i systemet. Detta ger en avslappnad miljö för användaren men har sina brister i att informationen inte är formell. Denna formen är passande för att få en översikt över området man behandlar, den lämpar sig därför inte för detaljerad information. Anledningen till detta är att det inte är lätt för användaren att hålla alla detaljer i huvudet. Vidare finns det en intervjuform som kallas strukturerad intervju som går ut på att utvecklaren styr intervjun mot de områden som han vill analysera.

Mål och målsättningsanalys bekymrar sig mest för frågor som uppkommer vid starten av utvecklingsprojekt. Denna metod ställer frågan ”Varför behöver den här organisationen det som användarna har uttryckt sig om?”. Man skapar en hierarki bestående utav mål och underliggande mål. Hierarkin består också utav de hinder och mål man önskar uppnå längre fram i tiden. Denna valideras för att nå enhällighet bland alla inblandade parter. Man förhandlar och köpslår för att nå en total samstämmighet om den ovanstående hierarkin.

Scenario-baserad kravframtagning kan närmast liknas vid prototyping. Skillnaden ligger i att prototyping skapar ett provsystem medan den här formen av kravframtagning koncentrerar sig på en enda användar-datorinteraktion åt gången. Man tar fram hur ett gränssnitt skall se ut vid en viss process och låter användaren prova. Användaren får sedan kommentera vad som är bra, vad som fattas och hur man kan ändra på detaljer för att uppnå en så bra miljö som möjligt.

Formulär-analys ser inte användaren som den främsta källan i kravframtagningsprocessen. I stället förlitar sig den här metoden på formulär av olika slag. Genom att granska de formulär som cirkulerar i organisationen kan man modulera s.k. entitet-relationsmodeller. Dessa beskriver hur informationen som organisationen är beroende av är sammankopplade.

Naturligt språk har två tillvägagångssätt: direkt kontakt med användarna eller genom att granska skrivna dokument. Tre saker som gör naturligt språk attraktivt, nämligen vokabulären, informaliteten och syntaxen. Vokabulären med sina tusentals fördefinierade ord gör det till ett effektivt kommunikationsmedium. Informalitet är oftast en oönskad egenskap men den är utmärkt när det gäller att handskas med komplexitet. Den förhindrar att man gräver ner sig i detaljer som kan lösas senare i arbetet. Syntaxen är en önskad egenskap då den redan existerar hos alla.

Tekniker för återanvändning av krav förespråkas då kravframtagningen är den mest arbets- och tidskonsumerande uppgiften. Den bygger på att man kan återanvända krav från tidigare utvecklingsprojekt. Man kan också använda krav från liknande projekt efter att dessa har validerats och funnits korrekta. Uppskattningsvis är 15 procent av kraven unika för ett visst projekt och resterande 85 kan man hitta i liknande projekts kravspecifikationer.

Uppgiftsanalys för kravframtagning är en effektiv metod för att ta fram krav, speciellt för krav som rör människa-dator-interaktion. Uttrycket ”uppgiftsanalys” refererar till en mängd metoder och tekniker som analyserar och beskriver hur användarna gör sina jobb i form av de aktiviteter som användarna utför samt hur aktiviteterna är strukturerade. Det refererar också till vilken kunskap som krävs för utförandet av jobbet.

Kravframtagning som en social process finns i tre olika versioner. Teknik-centrerad design där experter informerar och rådfrågar kunden genom utvecklingens alla steg. Joint customer-specialist design där användare är involverade i alla utvecklingsstegen. Användar-centrerad design där alla användare bidrar till utvecklingen. I den här sortens metoder anser man att kraven inte ligger färdiga och väntar där ute utan att kraven tas fram genom kommunikationen mellan expert och användare. Man använder sig utav s.k. workshops där kraven upptäcks under samarbete. Etnografi är ett alternativt arbetssätt där experter observerar användarna under en längre tid för att sedan analysera sina observationer.

2.8.2 Kravspecificering

En kravspecifikation kan ses som ett kontrakt mellan användare och systemutvecklare som definierar önskat funktionellt uppträdande av det system som utvecklas. Detta definieras utan att visa hur det realiseras. Den här processen ger som resultat en formell modell som kommer att användas i efterkommande utvecklingsarbete. Syftet med att producera en formell modell är följande:

- Specifikationen används som en överenskommelse mellan systemutvecklare och användare för att påvisa vad problemet är som skall lösas med hjälp av det nya systemet.

- Specifikationen är också en ritning för utvecklingen av systemet.

För att genomföra denna process krävs kunskap om problemdomänen. Denna kunskap tillhandahålls genom kravframtagningsprocessen. Specifikationsprocessen använder och producerar en mängd modeller inklusive den slutliga formella kravspecifikationen. Processen är analytisk på grund av all kunskap som systemutvecklaren tar fram från problemdomänen måste undersökas. De flesta metoder för kravspecificering antar att resultatet blir en kravspecifikationsmodell, men det är mer passande att säga att resultatet blir fler modeller som var och en svarar mot olika vinklar av problemet. Med detta i åtanke kan sägas att kravspecificeringsprocessen producerar följande:

- **Användar-orienterade** modeller som specificerar beteendet, icke-funktionella egenskaper m.m. hos systemet som tjänar som referenspunkt mellan systemutvecklaren, kunden och användarna.

- **Utvecklar-orienterade** modeller som specificerar funktionella och icke-funktionella egenskaper hos systemet såväl som resurshinder och designhinder som tjänar som ritningar för kommande utvecklingssteg.

Specificeringsprocessen är en central process i kravhanteringen. Den kontrollerar både kravframtagningsprocessen och valideringsprocessen. Under specificeringen kan det krävas mer information om problemet och då får man gå tillbaka till framtagningsprocessen igen. Färdigställandet av en specificeringsdel utlöser en valideringsprocess och därmed styr specificeringsprocessen också valideringsprocessen.

2.8.3 Kravvalidering

Validering är processen att kontrollera kravmodellen som skapades i specificeringsfasen mot användarens uttalade mening, dvs stämmer modellen med användarens åsikter. Utvecklaren måste i detta skede försäkra sig om att användaren uttrycker sig med hjälp av samma ” språk ” som han själv. Gör han inte det kan användaren bekräfta att modellen är riktig utan att veta om att han inte menar samma sak som utvecklaren. Vissa valideringsmetoder underlättar kommunikationen medan andra försvårar den.

En annan sak som utvecklaren måste kontrollera är om modellen är korrekt. Detta kan eventuellt göras med hjälp av ett automatiskt CASE-verktyg. Korrekt innebär här att modellen inte innehåller ologiska eller motsägelsefulla definitioner. Vad utvecklaren inte vet är om modellen är den rätta för just det här utvecklingsarbetet. Vad som verkat vara användarens problem kanske i verkligheten är totalt irrelevant, dvs en situation kan uppstå där tid och kraft läggs på att analysera fel problem. Det är här som valideringen kommer in och hjälper till att göra rätt sak istället för att göra en sak rätt.

Enligt IEEE (1983) definieras valideringsprocessen som processen att utvärdera systemet vid slutet av utvecklingsprocessen för att försäkra sig om att systemet tillmötesgår kraven från kravspecifikationen. Frågan som ställs direkt efter denna definition är - Är det tillräckligt att validera i slutet av utvecklingsprocessen? Statistiskt sett har svaret visat sig vara nej. Det finns ett pris man får betala om man väntar för länge med valideringen. Ju längre man väntar med valideringen desto dyrare blir det att rätta till misstag i form av testning, felkorrigering och ny utveckling av samma modell. I en undersökning som genomfördes på 70-talet (Boehm, 1980) visade det sig att upp till 50 procent av buggarna (felen) var fel i kraven. Kostnaden för att rätta till buggarna var upp till fem gånger dyrare än kostnaden för fel i design och kod. När fel

görs i kravspecifikationen kommer en del av det resterande arbetet utföras med detta fel som grund. Följderna kan därför bli mycket allvarliga. Denna undersökning kan tyckas föråldrad i sammanhanget men faktum är att konsekvenserna av ett fel i kravhanteringsprocessen blir lika stora i idag som då, kanske ännu större då systemen tenderar att bli mer och mer komplicerade.

2.8.4 Riktlinjer för vad man bör validera

Enligt Loucopoulos och Karakostas (1995) skall man se på valideringsprocessen mer som avbuggingsaktivitet, än som en aktivitet där man försöker bevisa att man gjort rätt. I stället för att lägga ner kraft på att bevisa att man gjort rätt bör man koncentrera sig på att identifiera och avlägsna alla förekommande fel. Valideringen är en process som går parallellt med kravframtagningen och specificeringen och utförs varje gång ett nytt krav mottas, analyseras och integreras. Valideringen är i huvudsak en ostrukturerad process. Med detta menas att man inte kan applicera en algoritmisk procedur som resulterar i en validerad kravmodell. Det finns dock en mängd uppgifter som utförs under valideringen, dessa följer de metoder och formalier som används under konstruktionen av kravmodellen. Målet med dessa uppgifter är att identifiera en mängd önskade egenskaper i kravmodellen. Dessa är som följer:

Intern konsistens uppnås om inga motsägelsefulla slutsatser kan härledas från kravmodellen. Om modellen säger att ingen på företaget skall tjäna mer än 15000 i månaden och sedan säger att lönen för en viss anställd är 16000 så är kravmodellen inkonsistent. Det finns alltså motsägelser i kravmodellen.

Icke tvetydig är kravmodellen om den endast kan tolkas på ett sätt. Det är väldigt lätt att få in tvetydighet i en modell om man använder sig utav naturligt språk. Tvetydighet kan få katastrofala följder. När ett system hanterar t ex säkerhetsfunktioner vid ett kärnkraftverk får det inte finnas utrymme för mer än en tolkning av ett krav.

Extern konsistens får man om det som står i kravmodellen överensstämmer med det som finns i problemdomänen. Varje tänkbar kraftansträngning måste tas till för att se till att kravmodellen stämmer med problemdomänen. Loucopoulos och Karakostas (1995) tar upp en undersökning (Basili & Weiss, 1981) av kravfel hos ett flygplan vid flottans forskningslaboratorium i USA avslöjades att 77 procent av alla fel inte var avskrivningsfel. 49 procent av dessa fel bestod av inkorrekta fakta om problemdomänen. Om kraven är föränderliga, dvs de ändras ofta, måste varje försök göras för att hålla faktan i kravmodellen aktuell.

Minimalitet i ett kravdokument erhålls om man enbart beskriver de krav som ställs. Motsatsen kallas över-specifikation och inträffar när man både beskriver kraven och hur designlösningen skall se ut. Detta är fel då det inskränker på hur lösningsalternativen får se ut. Generellt sett skall allt som begränsar designerns val i onödan inte inkluderas i kravdokumentet.

Fullständighet uppnås i en kravmodell när ingen viktig information utelämnas som behövs för att möta användarens behov. En kravmodell innehåller mål som skall nås i en problemdomän, liksom regler, fakta, och begränsningar som hör till domänen och det system som skall operera i domänen. Misslyckande att uppnå någon av dessa ingredienser hos en kravmodell kommer att ha påverkan på modellens korrekthet.

- Utelämnandet av t.ex. ett mål kommer att resultera i en modell som inte representerar användarnas alla krav.

- Misslyckandet att fånga en regel eller fakta om domänen kommer att förhindra systemet från att vara en korrekt modell över domänen.

- Frånvaron av en begränsning i kravmodellen kan leda till inkorrekt beteende hos systemet.

Redundans är en egenskap hos ett krav när kravet kan tillhandahållas i någon annan del av systemet eller om det helt enkelt är någon egendom eller uppförande som inte är önskat i systemet. Att bestämma om ett krav är redundant eller inte är en aktivitet som kräver förståelse av användarnas förväntningar om systemet.

2.9 Mål och resurser för kravvalidering

Kvalité är en eftersökt egenskap hos varje system. När man eftersträvar kvalité bör man tänka på två saker, hur mycket tid och pengar får användas samt vilken kvalitetskriteria man vill bedöma kravmodellen efter. Kravmodellen bör kontrolleras utav utvecklaren, ibland med hjälp av användare eller mjukvaruverktyg, för att uppnå ovannämnda egenskaper. Denna process bör fortgå tills kravmodellen stämmer helt överens med användarnas förväntningar. Det idealiska är om valideringsprocessen tilldelas den tid som behövs, men detta är sällan ett faktum. Enligt Loucopoulos och Karakostas (1995) så ägnas endast ett fåtal procent utav livscykelmodellen till kravhantering i verkligheten. Utvecklaren vill få kravhantering genomförd för att kunna ta sig an designprocessen som anses mer intressant. Man skall dock betänka att utvecklaren har många problem att överbrygga, brist på pengar och tid, brist på användare m.m. Det skall också tilläggas att användare kan tröttna på att gå igenom kravmodellen gång på gång och därför blir mindre samarbetsvilliga ju mer de tvingas delta i kravvalideringen. Det finns ett antal olika tillvägagångssätt när man utför en kravvalidering.

2.10 Metodkategorier för kravvalidering

Olika metodkategorier lämpar sig för olika sorters utvecklingsarbeten men målet med varje metodkategori är det samma, att bekräfta kundens ursprungliga krav mot den kravmodell som analyserats fram. Anledningen till att jag valt benämningen metodkategori är att jag inte vill tala om en specifik metod utan mer om en metodkategori, det innebär alltså att det kan finnas flera olika sorters prototyping men jag benämner dem mer generellt med metodkategori.
Det finns en mängd olika metodkategorier för kravvalidering. Några av dem presenteras här nedan:

Prototyping
Loucopoulos och Karakostas (1995) beskriver prototyping på nedanstående sätt.
När man använder sig av prototyping gör man det genom hela utvecklingsarbetet. Detta innebär att prototypen man arbetar med skapas redan efter kravframtagningen. I specificeringsprocessen skapas en prototypmodell och denna valideras sedan av utvecklaren som till sin hjälp har användare. Prototypmodellen måste därför innehålla meningsfull information för användarna för att valideringsresultatet skall bli värdefullt. Det finns två olika sorters information som är viktig för användarna, nämligen strukturell information och beteendeinformation. Av dessa två informationsslag skapas två prototypmodeller, en beteendemodell och en strukturell modell. Beteendemodellen visar vad systemet skall göra och den strukturella modellen visar hur systemet skall göra det. Om prototypmodellerna skapas med hjälp av ett logiskt språk kan de kontrolleras så att de inte innehåller missade definitioner m.m. Ett passande sätt att utföra detta på är att använda ett programspråk som är logikbaserat

t.ex. Prolog. Parallellt med detta stämmer utvecklaren av kravmodellerna med användarna som hjälp.

Animation
Animation är enligt Loucopoulos och Karakostas (1995) en teknik som med framgång kan användas på real-tidsystem. En viktig del i valideringen av real-tidsystem är att kraven för konkurrerande och tidsbegränsade aktiviteter fångas på rätt sätt. Med dagens nya teknik typ högupplösningsskärmar m.m. har grafisk representation blivit tillgänglig. Med hjälp av en grafisk representation kan systemet visualiseras på en dataskärm. Systemet är också operativt på skärmen. Detta gör att utvecklaren kan testa och validera systemet genom att köra det i datorn. På detta sätt testas alla funktioner och processer, vilket gör att utvecklaren får omedelbar feedback på hur systemet uppför sig vid alla tänkbara situationer.

Simulering
Wieringa (1996) beskriver simulering som ett verktyg som till exempel kan användas för att validera ett dataflödesdiagram. Ett dataflödesdiagram kan vara den formella specifikation som kravspecificeringen kan resultera i. En utvecklare kan använda sig utav simulering när han vill undersöka hur ett system påverkas av olika indata. Han kan då lägga in en funktion hos det nya systemet i datorn och mata funktionen med indata och sedan följa hur datan flödar genom funktionen. På detta sätt kan han se på ett tidigt stadium hur systemet reagerar och validera om reaktionen är önskvärd eller inte.

Genomgångar
Genomgångar är också ett sätt att validera dataflödesdiagram. En genomgång kan enligt Wieringa (1996) bestå utav ett möte mellan utvecklare och användare där utvecklaren går igenom diagrammet från början till slut och får kommentarer från användarna. Kommentarerna kan gälla allt från fel till alternativa lösningar. Kommentarerna skrivs ned men åtgärdas inte direkt utan detta sker i ett senare läge, krävs det så sker en andra genomgång för att göra en slutgiltig kontroll. Vid användning av genomgångar är det viktigt att samarbetet mellan utvecklare och användare fungerar bra annars hämmas kommunikationen.

Checklistor för validering
Checklistor används enligt Sommerville och Saywer (1998) som ett slags fokuseringsdokument vid valideringen. Meningen är att listorna skall fokusera användarna och utvecklaren på det som är viktigt. Detta ger användarna en chans att komma in i valideringsarbetet på ett snabbt och smidigt sätt. Checklistor har fördelen att vara ett lågkostnadsalternativ samt lätta att använda. Det finns ett antal tumregler vid användandet av checklistor men dessa kommer jag att gå in på under genomförandedelen.

3. Problembeskrivning

3.1 Problemområde

I introduktionen och bakgrundsbeskrivningen har jag steg för steg närmat mig det område inom systemutveckling som jag valt att ha som problemområde. Detta område är kravhanteringen som har en ytterst viktig roll i systemutveckling. Jag har belyst att fel gjorda i kravhanteringen får allvarliga följder för kommande utvecklingsarbete. Att ta fram en korrekt och överensstämmande kravspecifikation är därför ett mål för alla parter i utvecklingsprocessen. Kravspecifikationen som är resultatet av kravhanteringsprocessen tjänstgör som garanti för beställaren, men också som ett grundande dokument när det gäller den fortsatta utvecklingsprocessen för utvecklaren. I tidigare kapitel har jag försökt belysa problemen med kommunikation genom hela utvecklingsprocessen och då i synnerhet kravhanteringen.

3.2 Avgränsning

Kravhanteringsprocessen är en process i flera delar, detta har jag gjort klart i introduktionen och bakgrundsbeskrivningen. Jag har valt att koncentrera mitt arbete runt valideringsprocessen, vars uppgift är att kontrollera kravmodellens korrekthet mot användarnas intentioner. Detta gör den till en extremt viktig process som trots allt ibland får stå i skymundan för de mer skapande processer som t.ex. design.

3.3 Problemprecisering

Problemen jag valt att undersöka är ett av de problem som finns i kravhanteringsprocessen, nämligen brister i kommunikationen. Jag har valt att jämföra diverse metodkategorier med inriktning på kommunikation och samarbete. Eftersom kravhanteringsprocessen är grundläggande för det nya systemet tycker jag att det är viktigt att belysa svårigheter i den här delen av utvecklingsarbetet. Jag har koncentrerat mig på den sista delen av kravhanteringsprocessen nämligen valideringen, som är den sista bekräftelsen på att kraven från användare har definierats rätt. När användarna och utvecklaren kommunicerar är det som sagt inte självklart att de ”pratar samma språk”. Vidare är den sociala biten lika viktig för att kravhanteringsprocessen skall lyckas, utvecklare och användare måste hysa en ömsesidig respekt för varandra om ett samarbete skall fungera. Jag har också valt att kontrollera hur valideringsmetoderna är uppbyggda för att på så sätt se om det finns några tecken på varför en metod gynnar eller missgynnar kommunikationen i valideringsprocessen. En annan viktig del som jag tar upp separat är användarmedverkan. Hur påverkar användaren valideringen? Kan problem med användarmedverkan undvikas?

3.4 Frågeställningar

- Hur är metoden uppbyggd?
- Hur fungerar kommunikationen mellan utvecklare och användare?
- Vilka faktorer påverkar samarbetet mellan användare och utvecklare?
- Vad bör man tänka på vid användarmedverkan?

3.5 Förväntat resultat

Jag hoppas finna att detta problemet med kommunikationen kan överbryggas genom användning av rätt metoder och ett intimt samarbete mellan utvecklare och användare. Vidare förväntar jag mig finna att vissa valideringsmetoder hanterar detta problem bättre än andra. Jag hoppas också kunna ta fram en del information om användarmedverkan och vad man bör tänka på när man tillämpar denna.

4. Metodval

Vid valet av metod undersökte jag tre olika tillvägagångssätt, intervjuer, enkätundersökning samt litteraturstudie. De här tre metoderna har alla sina fördelar och nackdelar. Men efter en översiktlig kontroll fastnade jag för en litteraturstudie på grund av att jag ville sammanställa information på ett översiktligt plan. Enligt egen erfarenhet från projektet i årskurs två har jag lärt mig att en enkätstudie är svår att genomföra, på grund av att svaren tenderar att komma tillbaks försent eller inte alls. En intervjuserie får en stark koppling till verkligheten men intervjupersoner med översiktlig kunskap om flera valideringsmetoder är svåra att finna.

Jag har valt att göra en metodanalys av olika valideringsmetoder genom en litteraturstudie med avseende på hur metoden används och hur kommunikationen mellan användare och utvecklare fungerar. För att få den övergripande informationen om varje metod har jag inte valt någon speciell metod utan analyserat den kategori som metoden tillhör, t.ex. prototyping eller animering. För att kunna få fram denna information har jag letat efter källor som har utbildningsbakgrund. Om en författare har undervisat i någon form av systemutveckling innehar han eller hon ofta ett översiktligt perspektiv såväl som en detaljkunskap. Detta gör att författaren kan uttrycka sig om modeller och metoder på ett metaplan. Jag har också valt att kontrollera hur användarmedverkan påverkar valideringen, detta för att se hur stor effekt användare har på valideringen.

4.1 Källor

- Loucopoulos & Karakostas (1995) *System Requirements engineering* är en bok som behandlar hela kravhanteringsprocessen från framtagning till validering. Olika metodkategorier beskrivs och kommenteras. Man ger också diverse exempel på hur scenarion inom kravhanteringen kan gå till.

- Andersen (1994) *Systemutveckling - principer, metoder och tekniker* är en bok som ger en grundläggande introduktion i modeller, metoder, tekniker och verktyg för systemutveckling.

- Flynn & Warhurst (1994) *An empirical study of the validation process within requirements engineering* är en artikel i kurskompendiet för kursen Systemutveckling III vid högskolan i Skövde. I artikeln får ett antal systemutvecklare svara på frågor om valideringsprocessen.

- Lubars, Potts & Richter (1993) *A review of the state of the practice in requirements modelling* är en studie som omfattar tio företag. I studien undersöks hur dessa tio företag definierar, tolkar och analyserar krav för deras mjukvarusystem och produkter.

- Sommerville & Sawyer (1998) *Requirements engineering - A good practice guide* är en del av en artikel på Internet som beskriver hur man definierar checklistor för validering och vad man bör tänka på när man använder dessa.

- Macaulay (1996) *Requirements Engineering* är en bok som förklarar hur viktigt det är att förstå behoven eller kraven hos de organisationer som man utför utvecklingsprojektet hos. Den sträcker sig över många viktiga områden som till exempel mänskliga och organisatoriska faktorer som kan påverka ett utvecklingsarbete.

- Wieringa (1996) *Requirements Engineering - Frameworks for understanding* är en bok som tar upp arbetet med att fånga de krav som behövs för att kunna designa det system som kunden önskar sig. Olika metoder beskrivs för att förklara hur den ovanstående processen går till. Man belyser också de andra steg som måste tas för att klara av ett utvecklingsarbete.

- Emam & Madhavji (1995) *A Field Study of Requirements Engineering Practices in Information Systems Development* är en artikel som innehåller rekommendationer för att förbättra kravhanteringsprocessen. Den poängterar vikten med att förstå problemen inom detta område. Den anger också en riktning för fortsatt forskning om metoder och verktyg.

4.2 Författare

De författare jag använt mig utav är följande:

- Professor Pericles Loucopoulos som är medlem av datainstutitionen vid universitetet i Manchester (UMIST). Mellan April 1992 och Mars 1994 var han chef på samma avdelning. Hans forskningsintressen ligger inom kravhantering, konceptuell modellering och systemutvecklingsmetoder. Han har varit medförfattare på fyra böcker, utgivare av en bok och författare av hundratals artiklar inom ovanstående ämnen. Det material jag främst har använt av hans material är Loucopoulos & Karakostas (1995). Denna bok har varit extremt värdefull då den belyser hela kravhanteringsprocessen. Det är viktigt att förstå hela kravhanteringsprocessen då den består av tre delar som är intimt förknippade med varann.

- Professor Vassilios Karakostas är också medlem av datainstutitionen vid UMIST. Han har forskat inom mjukvarutillverkning som deltagare av flera europeiska program. Även han har författat flera hundra artiklar och varit medförfattare av flera böcker inom dataområdet.

- Professor Erling S Andersen som innehar sin professur inom informationsvetenskap vid universitetet i Bergen. Han har haft en rad förtroendeuppdrag inom den nordiska databranschen. Han har mer än tjugo års erfarenhet av systemutveckling, både som lärare vid universitet och högskolor och från praktisk systemutveckling i organisationer.

- Professor Ian Sommerville är chef för datainstutitionen vid universitetet i Lancaster och har författat ett antal böcker samt skrivit artiklar för diverse tidskrifter inom mjukvarudesign.

- Dr. Peter Sawyer är anställd vid universitetet i Lancaster. Även han har varit författare och medförfattare av ett antal böcker. Vidare skriver han artiklar för olika tidskrifter inom dataområdet.

- Dr. Donald J Flynn är senior lecturer vid datainstutitionen vid universitetet i Manchester och medlem i Social/Humans Factors group. Han har författat och medförfattat böcker och artiklar inom kravhanteringsprocessen.

- Mitch Lubars arbetar vid Objective Reuse Technologies i Austin Texas och har varit medförfattare till artikeln *A review of the State of Practice in Requirements Modeling*.

- Professor Colin Potts undervisar vid College of computing vid Georgia Institute of Technology. Han undervisar i software engineering och människa-maskininteraktion.

- Charles Richter arbetar vid Objective System Solutions i Austin Texas och har varit medförfattare till artikeln *A review of the State of Practice in Requirements Modeling*.

- Dr. R.J Wieringa undervisar vid Faculty of Mathematics and Computer Science Vrije Universiteit, Amsterdam. Han har författat fem böcker och flertalet artiklar inom områden som konceptuell modellering, maskinintelligens och kravhantering.

- Linda Macaulay är senior lecturer vid universitetet i Manchester. Hon har författat ett antal kapitel i olika böcker samt producerat artiklar inom områden som kravhantering och mjukvaruunderhåll.

- Khaled El Emam innehar en Ph.D i software engineering. Han är anställd vid Fraunhofers institut för experimentiell software engineering.

- Dr. N.H Madhavji är anställd vid McGill University i Montreal, Kanada. Han är där professor vid skolan för dataforskning.

5. Genomförande

Jag har valt att studera hur kommunikationen sker i olika metoder av validering. Detta har skett genom en litteraturstudie där ingen speciell metod har analyserats utan snarare hur en viss metodkategori används. Med metodkategori menar jag t.ex. animering eller genomgångar. Varje metodkategori har undersökts med kommunikation och samarbete som högsta prioritet och då i synnerhet hur användare berörs. Vad som också är intressant i detta läge är hur samarbetet mellan användare och utvecklare påverkas av den sociala biten. Jag har valt att analysera varje metod med hjälp av följande kriterier:

1. Hur är metoden uppbyggd?
2. Hur fungerar kommunikationen mellan utvecklare och användare?
3. Vilka faktorer påverkar samarbetet mellan användare och utvecklare?

Dessa kriterier har jag valt för att kunna bedöma hur varje metod fungerar och hur kommunikationen sker mellan användare och utvecklare.

Sedan tar jag också upp användarmedverkan och försöker ta fram information och tips om hur man bäst använder sig av den för att på så sätt se mer generellt på användare inom valideringen.

5.1 Prototyping

1. Hur är metoden uppbyggd?

Prototyping är en metod som har en egen livscykel. Detta innebär att den appliceras genom alla tre stegen av kravhantering. Genom hela livscykeln arbetar användare och utvecklare nära varandra och mycket av kommunikationen sker med hjälp av naturligt språk. Eftersom användaren är med och ser hur prototypen växer fram med hjälp av hans kommentarer känner han sig delaktig i projektet och därför underlättas samarbetet mellan användare och utvecklare. Prototyping förekommer i olika former och den stora skillnaden mellan de olika formerna är om man skall behålla prototypen eller kasta bort den vid arbetets slut, detta till förmån för ett helt nytt system som bygger på den kunskap som framkommit vid arbetet med prototypen.

Kasta-bort-prototyping innebär enligt Wieringa (1996) att man producerar en prototyp som skall visa upp systemets egenskaper innan det verkliga systemet implementeras. Detta är ingen strategi för att implementera en produkt utan ett sätt att analysera fram kunskap för att senare skapa ett system med hjälp av den kunskapen. Prototypen behöver inte vara robust, välstrukturerad, pålitlig eller effektiv för den skapas enbart för att lära oss hur systemet kommer att uppföra sig innan det implementeras.

Det börjar med att användaren uttrycker några grundläggande funktioner och krav hos det system han vill ha. Med hjälp av dessa skapar utvecklaren en grundläggande prototyp som användaren får testa och ha synpunkter på. Användaren ser hur prototypen växer fram och kan under hela arbetet komma med kommentarer och invändningar på hur han ser på vissa saker. Detta ger också utvecklaren chansen att komma med förslag till användaren där han presenterar de olika lösningar och möjligheter som kan användas med dagens teknik. När man använder sig utav den här valideringsmetoden kan man utnyttja arbetssätt som inte är helt acceptabla vid konstrueringen av det riktiga systemet, s.k. snabba och smutsiga metoder. Dessa

snabba och smutsiga metoder, har inget med smuts att göra, används med insikten om att prototypen inte får implementeras. Med snabb och smutsig metod menas till exempel att rita användargränssnitt med hjälp av penna och papper istället för att konstruera ett gränssnitt på en dator. Detta sparar mycket tid mot att konstruera ett riktigt gränssnitt. Användaren kan då snabbt tycka till om hur han vill att gränssnittet skall se ut och om det fuskkonstruerade gränssnittet möter de krav han ställt genom utvecklingsarbetet. Om användaren godkänner konstruktionen kan ett riktigt gränssnitt skapas.

Ett annat sätt att validera prototyper är att först framställa en prototyp efter användarens önskemål och sedan låta honom prova prototypen i ett speciellt labb som är utrustat för just denna aktivitet. I ett sådant labb finns utrustning för att t.ex. spela in användarens manövrar på video. Användaren kan samtidigt som han arbetar med prototypen kommentera det han tycker är bra och dåligt. Utvecklaren tittar sedan på videon för att se om det uppstår några problem vid hanteringen eller om användaren har några kommentarer angående design eller funktion.

2. Hur fungerar kommunikationen mellan utvecklare och användare?
När man använder sig av den här sortens validering brukar kommunikationen inte vara ett problem därför att kunskapsöverföringen mellan användare och utvecklare oftast sker på ett ganska oformellt sätt. Detta innebär att användaren oftast inte behöver genomgå någon form av utbildning utan bidrar direkt med sin erfarenhet och kunskap. Det som är positivt ur kommunikationssynpunkt med prototyping är att användaren får en känsla av att vara en i utvecklingslaget. Detta är viktigt för att samarbetet mellan utvecklare och användare skall fungera tillfredsställande.

3. Vilka faktorer påverkar samarbetet mellan användare och utvecklare?
Vid prototyping är det viktigt att användaren vet vilka krav som skall ställas på det nya systemet för att kunna ge utvecklaren den information som krävs för att jobba med prototypen. Vid den här sortens validering är det viktigt att utvecklare och användare hjälper varandra. Utvecklaren kan upplysa användaren om tekniska möjligheter och användaren kan tala om vad som varit problemet tidigare så att samma problem inte uppstår en gång till.

För och nackdelar
Det finns både fördelar och nackdelar med prototypvalidering. Enligt Wieringa (1996) finns följande fördelar:

- Man vet med större säkerhet hur ett system kommer att påverkas innan det implementeras. Den kan med fördel användas i de tidigare delarna av utvecklingen för att minska osäkerheten på kraven, med detta menas att säkerheten på att kraven inte drastiskt skall ändras vid ett acceptanstest ökar.

- Vid klient-orienterad utveckling hjälper kasta-bort-prototyping till att minska osäkerheten hos kund eller användare när det gäller vad som kan åstadkommas eller vad de kan förvänta sig. Detta har två fördelar:
  1. Det gör det möjligt att minska antalet ändringar i kraven efter det att kraven är nedskrivna.
  2. Chansen att förbättra supporten för användare när det gäller den färdiga produkten ökar också.

Vidare finns det också en del nackdelar med prototypvalidering. Wieringa (1996) har listat följande:

- Den kan distrahera utvecklaren, användaren eller kunden från de viktiga egenskaperna. En användare kan koncentrera sig mer på hur en ikon skall se ut än hur en del av programmet skall fungera.

- Det tar tid att utveckla en prototyp så kasta-bort-prototyping kan öka utvecklingstiden. Många beställare kan tycka att spendera tid på att koda en prototyp är bortkastad tid, men om tiden används klokt kan en kasta-bort-prototyp minska behovet av korrigeringar och produktunderhåll.

- Utvecklaren kan bli fäst vid prototypen och behandla den som om den vore en del av slutprodukten. Vid klient-orienterade projekt kan beställaren eller användaren bli fäst vid prototypen. Båda dessa företeelser är farliga då prototypen är producerad utan hänsyn till säkerhet, pålitlighet, prestanda etc.

Ovanstående handlade alltså om kasta bort prototyping men det finns också en annan form som kallas **inkrementiell prototyping** där prototypen övergår till att bli det färdiga systemet bit för bit. För övrigt fungerar metoden som ovanstående.

5.2 Genomgångar

Valideringsprocessen går ofta ut på att analysera en formell modell som skapats i kravspecificeringsprocessen. En formell modell kan se ut på många sätt och ett av dessa sätt är dataflödesdiagram. Dataflödesdiagram används för att visa hur data och information flödar genom ett företag eller en organisation. Jag kommer hädanefter referera till dataflödesdiagrammet som modellen. Modellen används för att få en inblick i vad det nya systemet skall hantera, utan att för den skull tala om hur designen på systemet skall se ut. Detta är bra därför att om ett lösningsförslag kan ses i den formella modellen kan begränsningar på designen av systemet sättas. Utvecklaren kan omedvetet känna sig låst när det gäller att skapa det nya systemet. Ett sätt att validera modellen är genomgångar. Modellen kan representera det nuvarande systemet och då behövs egentligen bara en jämförelse mellan modellen och det verkliga systemet. Om modellen däremot representerar det nya systemet finns inget att jämföra med och då kan man använda sig utav en genomgång.

1. Hur är metoden uppbyggd?
En genomgång beskriver Wieringa (1996) som en kontroll av en produktspecifikation. I sin enklaste form är en genomgång ett möte mellan utvecklare och användarrepresentanter där någon som kan diagrammet går igenom det. Detta kan vara modellbyggaren men det kan också vara någon som läst igenom modellen noggrant före mötet. Alla fel eller misstag i diagrammet noteras men diskuteras inte. Modellbyggaren rättar sedan till alla fel och misstag och om nödvändigt genomgår modellen en ny genomgång. För att detta tillvägagångssätt skall lyckas måste genomgången fokuseras på modellen och inte på den som byggt det.

2. Hur fungerar kommunikationen mellan utvecklare och användare?
Kommunikationen sker med hjälp av naturligt språk vid möten mellan användare och utvecklare så ingen utbildning behövs, däremot kan en liten genomgång av dataflödesdiagram inte skada.

3. Vilka faktorer påverkar samarbetet mellan användare och utvecklare?
Deltagarna i genomgången får inte känna sig underlägsna modellbyggaren, de måste befinna sig på samma hierarkiska nivå för att våga kommentera eventuella fel och brister som modellen har. Känslan av att "jag inte kan lika mycket som modellbyggaren så därför är jag tyst" får inte finnas. Det är oftast modellbyggarens eller utvecklarens ansvar att förmedla vi-känslan på dessa möten då det är han som står högst i den abstrakta hierarkin. Det som krävs av användaren i den här sortens valideringar är domänkunskap och tillräckligt med självförtroende för att yttra sig om de fel och brister han kan upptäcka.

5.3 Simulering och animation

Simulering och animation är två andra sätt att validera ett dataflödesdiagram. Wieringa (1996) beskriver ovanstående metodkategorier på följande sätt:

Simulering
1. Hur är metoden uppbyggd?
En komplett dokumenterad dataflödesmodell kan exekveras utav en passande processor. En simulering av en enkel systemtransaktion innehåller sekvens av alternativa systemstatusar och systemhandlingar. Modellbyggaren kan simulera hur systemet uppför sig genom att skapa en händelse och följa hur datan flödar genom systemet under just den händelsen. Datan flödar genom systemet tills en reaktion produceras och förhoppningsvis produceras den rätta reaktionen. Detta förlopp kan också kontrolleras utav en mjukvarutolkare. Som ni förstår finns det inte mycket plats för användaren i detta skede då kontrollen utförs av utvecklaren eller någon mjukvarukunnig person. Användaren kan däremot bidra till att fylla förrådet av data som skall in i systemet. Här kommer användarens domänkunskap till användning, han kan också förmedla sin kunskap till utvecklaren som sedan använder sig av den när han levererar indata till systemet.

2. Hur fungerar kommunikationen mellan utvecklare och användare?
Utvecklaren och användaren diskuterar om systemet svarar på indatan som det är tänkt. Användaren får här bedöma om det är det förväntade händelseförloppet som sker eller inte. Det här är en metodkategori som ger användaren en möjlighet att direkt se resultaten av sina kommentarer och känna sig som en i utvecklingsgruppen.

3. Vilka faktorer påverkar samarbetet mellan användare och utvecklare?
Den sociala faktorn påverkar givetvis den här formen av validering. En annan faktor som påverkar är om användaren har fått tillräcklig utbildning på just dataflödesdiagram som den här metodkategorin appliceras på i det här fallet.

Animation
1. Hur är metoden uppbyggd?
En animation är en animerad presentation av en modellsimulering. En animerad simulering kan till exempelvis presentera ett dataflödesdiagram för användaren, och presentera datan i diagrammet som en kula. Avfyrningen av data kan sedan representeras av kulan som löper genom diagrammet. Om detta görs i passande hastighet kan detta resultera i en rörelse genom hela diagrammet. Den manipulerade datan kan sedan samlas i en rapport. Detta förfarande kan man se i t.ex. programmering, där man kan följa en variabels värde genom kodraderna för att se hur värdet påverkas av olika operationer. Användaren kan här vara med och följa systemets behandling av data och därmed få insikt i hur systemet fungerar. Animering förekommer ofta

som valideringsmetod i real-tids system där man måste vara säker på att systemet reagerar på rätt sätt inom en föreskriven tid.

2. Hur fungerar kommunikationen mellan utvecklare och användare?
Vid animeringen kan en användare sitta med och kommentera hur systemet bör uppträda vid olika statusar. Om systemet är komplicerat bidrar också användaren med domänkunskap som överförs med normal konversation. Ett bra samarbete mellan utvecklare och användare gör att båda kan tillgodose sig utav den andres kunskap på ett ganska okomplicerat sätt. Användaren behöver oftast inte genomgå någon komplicerad utbildning och detta underlättar hans insats i valideringen.

3. Vilka faktorer påverkar samarbetet mellan användare och utvecklare?
Här kan användaren få insyn i hur systemet fungerar inuti, om det är relevant eller inte låter jag vara osagt. Systemets nivå av komplexitet spelar här en roll om man ser till det totala förståendet av systemet för användaren. Kan användaren sätta sig in i hur systemet fungerar kommer han med största säkerhet att uppnå en högre grad av förståelse i sitt arbete. Om han däremot inte kan sätta sig in i hur systemet fungerar inuti bör han känna till systemets funktioner utåt för att kunna bistå utvecklaren i hans arbete.

5.4 Kontrollistor för validering

1. Hur är metoden uppbyggd?
Till att börja med vill jag säga att kontrollistor inte är en komplett metod utan ett stöd i valideringsprocessen. Dessa kontrollistor används för att fokusera utvecklare och användare på viktiga funktioner och egenskaper hos det blivande systemet. Man kan också säga att kontrollistorna fungerar som en minneslapp vid valideringen. Kontrollistorna används när kontrollen av kravdokumentet skall göras. Kravdokumentet är den handling som sammanfattar alla krav som framtagits under kravhanteringsprocessen. Enligt Sommerville och Sawyer (1998) finns följande fördelar med dessa kontrollistor:

- De hjälper till att fokusera valideringsprocessen.
- Kostnaden för introducering är låg.
- Implementeringskostnaden är låg.
- Det är okomplicerat att använda kontrollistorna.

Vidare säger Sommerville och Sawyer (1998) att kontrollistorna ger valideringsprocessen struktur. Strukturen minskar chansen att de som läser kontrollistorna glömmer någon aspekt av kravdokumentet. Kontrollistorna hjälper också personal som inte är van vid kravvalidering. Kontrollistorna ger dem tips och vinkar om vad de skall titta efter så de känner sig mer delaktiga i processen. Delaktighet är väldigt viktigt att känna för de medverkande eftersom detta ger dem teamkänsla och ett bevis på att deras insats är uppskattad. Detta är speciellt viktigt för användare och beställare som inte har någon erfarenhet av validering.

Kontrollistorna bör enligt Sommerville och Sawyer (1998) baseras på följande saker:

1. Är kraven kompletta, det vill säga vet kontrollanten av listan om några krav fattas eller om någon information fattas från individuell kravbeskrivning?
2. Är kraven konsistenta, det vill säga innehåller beskrivningarna av olika krav några motsägelser?
3. Kan man förstå kraven, det vill säga förstår läsarna av dokumentet vad kraven betyder?
4. Är kraven entydiga eller kan man tolka kraven på mer än ett sätt?

5. Är kravdokumentet strukturerat, det vill säga är relaterade krav samlade i grupper och i så fall finns det ett alternativt sätt att gruppera dem på?
6. Kan man spåra kraven, det vill säga är de entydigt identifierade och inkluderar de länkar som förbinder kraven på dokumentet med de krav som kommer från användare och beställare etc?
7. Överensstämmer kravdokumentet i helhet och de individuella kraven med de definierade standards som existerar?

2. Hur fungerar kommunikationen mellan utvecklare och användare?
Vid den här sortens validering ligger kommunikationen på en relativt normal konversationsnivå. Den formella modellen som specificerades i kravspecificeringen utvärderas med hjälp av kontrollistor som skall fokusera användare och utvecklare på de viktiga punkterna. Dessa kontrollistor måste utformas på ett sätt som användaren lätt kan ta till sig. Den här metoden är väldigt väl anpassad för användarmedverkan om man följer de tips och råd som Sommerville och Sawyer (1998) ger i artikeln.

3. Vilka faktorer påverkar samarbetet mellan användare och utvecklare?
Av ovan-stående kan man utläsa att metoden bryr sig om de användare som deltar i valideringsarbetet, jag tänker då i huvudsak på punkt tre. Med användare och beställare som deltar i valideringsarbetet i åtanke talar också Sommerville och Sawyer (1998) om följande tumregler eller faktorer som bör betänkas:

- Kontrollistor skall uttryckas i ganska allmänna termer så de kan förstås av slutanvändare som inte är systemexperter. Som en allmän regel skall kontrollistorna inte göras för långa och bör inte innefatta mer än tio saker. Om man tar upp mer än tio saker på kontrollistan kommer kontrollanten att ha svårt att komma ihåg dem alla och måste därför konsultera kontrollistan kontinuerligt.

- Faran med att uttrycka sig allmänt är att kontrollistorna blir vaga och detta medför att det blir svårt att svara på de frågor som ställs i listorna och svara på ett användbart sätt. Man måste alltså försöka hitta den rätta balansen mellan detalj och generalitet. Kontrollistor som pekar ut speciella fel detaljerat är följaktligen inte så användbara på grund utav den skillnad som kan finnas mellan olika system.

- Kontrollistorna kan användas på olika sätt. De kan användas som enkla påminnelser när man skall läsa kravdokumentet men också mer systematiskt och då innehålla en indikation på att ett speciellt krav har blivit behandlat.

5.5 Användarmedverkan

Eftersom kommunikation kräver två parter vill jag här belysa vikten av att parterna kommer överens och kan föra en vettig dialog i valideringsprocessen. Det finns många svårigheter vid arbetet med validering. Under det här examensarbetet har jag fått erfara själv och genom andra att vägen till en bra validering är lång. Det finns så många saker att ta hänsyn till i början, under och i slutet av ett valideringsarbete. Eftersom jag koncentrerat mig på kommunikation och samarbete inom min problemprecisering vill jag ta upp de saker jag funnit värda att tänka på när det gäller kommunikation och samarbete mellan användare och utvecklare.

Till att börja med kommer valet av valideringsmetod att ha stor inverkan på samarbetet och kommunikationen mellan ovannämnda parter. Om vi till exempel jämför genomgångar med simuleringar finner vi att en genomgång är mycket mer beroende av användare än vad en simulering är. Detta visar att samarbete och kommunikation kan begränsas direkt vid metodvalet. Nu är det därmed inte sagt att de metoder som inte förespråkar användarmedverkan gör det på grund av att de inte vill ha användare med i valideringsprocessen. Oftast finns det goda skäl för denna inställning, till exempel att arbetsuppgifterna är för komplicerade för användaren eller att åtkomsten av användare är starkt begränsad vid vissa sorters projekt. Nedan kommer jag att ta upp en del hinder som kan finnas för att uppnå ett bra samarbete och en vettig dialog.

I en undersökning gjord av Emam och Madhavji (1995) ställer man en rad frågor om användarmedverkan till systemutvecklare med skiftande erfarenhet inom valideringsområdet. Frågorna rör inte explicit valideringsprocessen utan hela kravhanteringen. Det står heller inget skrivet om att det inte skulle gälla valideringsprocessen. Frågorna lyder på följande sätt:

- Skall användare medverka i kravhanteringsprocessen?
- Om de medverkar, vad kan de bidra med?
- Vilka användare skall medverka?

Svaren på de här frågorna är sammanställda från de individuella frågestunder som artikelförfattarna har haft med systemutvecklarna. På frågan om användarna skall medverka i kravhanteringsprocesen blev svaret att de alltid skulle vara med. De sätt som ansågs öka användarnas bidrag till kravhanteringsarbetet var följande:

- Vanliga möten mellan användare och utvecklare.
- Workshops
- Kommunikation med användare via email.
- Samarbetsgrupper som antingen är baserade i användarorganisationen eller i utvecklingsgruppen.
- Besök på användaravdelningar av utvecklare.
- Användare som sitter med i projektstyrande kommittéer.
- Användare som kontrollerar och godkänner dokumentationssaker.
- Användare som själva utför vissa delar av kravarbetet.
- Användare som får betala utvecklingsarbetet ur sin egen budget, t.ex. avdelningsbudget.
- Utvärdering av användarnas medverkan vid utvecklingsarbetets slut med arbetets framgång i åtanke.

Artikelförfattarna samlade också in åsikter om vad för egenskaper som vore passande för användarna att ha vid utvecklingsarbetet. Resultatet av den insamlingen visas här nedan:

- Erfaren och van vid datoriserade system.
- God kunskap om den domän de arbetar i, företagsprocesserna och behoven hos användarorganisationen.
- God kunskap om systemutvecklingsprocesser, speciellt kravhanteringsprocesserna.
- God kunskap om planering och skötsel av informationssystemsimplementering.
- Bra sociala kunskaper samt bra kommunikationsförmåga.

Av detta kan utläsas att det inte är några lätta krav att uppfylla för en användare. Det skall dock tilläggas att detta är bara önskemål från utvecklarnas sida men det ger en insikt i vad de egentligen önskar. Det är lätt att förstå att om utvecklaren går och hoppas på sådana egenskaper hos de användare som bistår honom i projektet kan detta leda till förbistringar mellan dessa bägge parter. Tar man en noggrann titt på önskemålen här ovan finner man att det

inte är mycket som skiljer denna önskelista från den som en systemutvecklingsfirma skriver i sina jobbannonser när de anställa en ny systemutvecklare.

Enligt Emam och Madhavji (1995) fann man också en del intressant information angående användarmedverkan. Det visade sig nämligen att även fast användaremedverkan var en av faktorerna till framgång i kravarbetet fanns det saker som förhindrade ett bra samarbete mellan användare och utvecklare. Det mest förekommade var avsaknaden av samarbete mellan användare och utvecklare historiskt sett. Men lika tydligt var den öppna fientlighet som råder mellan projektmedlemmar och användare samt otillgängligheten hos användare. Dessa företeelser fick en projektledare, enligt Emam och Madhavji (1995, sid 110), att yttra sig om saken på följande sätt:

> ” *Utvecklingsfolket har policyn att bygga de produkter som de känner att användarna behöver och sedan försöka sälja den produkten till användarna. Företagsfolket är inte glada över detta. Utvecklingsfolket tror att de förstår organisationen så bra, ja till och med bättre än användarna så de försöker tvinga på användarna deras idéer. Utvecklingsfolket kanske förstår organisationen, men deras iakttagelse av organisationsmålen kan skilja sig från användarnas*. ”

Ett exempel på öppen fientlighet presenteras i artikeln då en äldre utvecklare bad om hjälp vid validering av dokumentationen. Den användare, även han äldre, som blev ombedd att hjälpa till vägrade och sa ”*vet du inte vad du håller på med, varför vill du att jag skall validera den?*”. Detta visar på avsaknaden av förtroende mellan parterna. Om man har sådana här händelser i huvudet när man startar en kravhanteringsprocess kan man lättare undvika dess fallgropar. Ovanstående text visar att ett bra samarbete måste inledas direkt om projektet skall fungera.

Det kan därför vara värt att skaffa sig en uppfattning om den användarkår som finns tillgänglig. Enligt Emam och Madhavji (1995) är det mycket värt att hitta en eller flera användare som är ledande inom användargruppen. Ledande användare innebär inte chefer i det här fallet. Ledande innebär att användaren har de andra i gruppens förtroende och respekt och kan föra deras talan. Fördelen med en sådan användare är att han kan hjälpa till i utvecklingsarbetet med att få de andra att acceptera samarbetet och göra sitt bästa ur resultatsynpunkt. Eftersom valideringen är det slutgiltiga testet för kravhanteringsprocessen är det extremt viktigt att samarbetet mellan användare och utvecklare fungerar så bra som möjligt.

En annan undersökning som utfördes av Flynn och Warhurst (1994) visar på andra problem med kommunikationen inom kravhanteringsprocessen. Eftersom kravhanteringsprocessens delar är intimt sammankopplade med varandra och utförs iterativt vid behov får problem i någon av delarna ofta följder i resterande delar. Artikeln av Flynn och Warhurst (1994) tar upp följande problem i kommunikationen mellan användare och utvecklare:

- Ett vanligt förekommande klagomål är att användare inte tillfredsställande kan förklara sina krav, vilket kan bero på att användare har svårt att välja ett medium att uttrycka sig med, eller att de inte ger sig själva tillräckligt med tid för att delge utvecklaren sina krav. Ett träffande uttalande av en utvecklare under undersökningen löd så här ”Ibland händer det att du får tag i en användare som vet men inte kan uttrycka sig och en chef som kan uttrycka sig men inte vet”. Ett problem av den här sorten kan väldigt lätt följa med genom hela kravhanteringsprocessen. Om användaren vet hur han vill ha det men inte kan uttrycka sig kan han då utföra valideringssysslan på ett tillfredställande sätt? När han ger kommentarer

under ovanstående process är det då säkert att han uttrycker sig klart och entydigt för utvecklaren?

- Det är inte alltid som användarna är tillgängliga när de behövs. Om användarna inte tar sig tid för att validera kommer kanske en del av kravspecifikationen inte att bli validerad. Resultaten av den här undersökningen kan endast vinkar om hur man skall förklara det här problemet, som till exempel frånvaron av stöd från chefer, ovilja att deltaga från användarnas sida eller det fakta att användarna är för geografiskt spridda för att kunna delta.

- Utvecklare har problem att få användarna att förstå kravspecifikationen som skall valideras. Undersökningen implikerar att detta beror på att användarna inte verkar förstå eller inte är förstående mot utvecklarens anteckningar även om dessa är i diagramform. Användarna vill helst hålla samtal och anteckningar inom deras egen begreppsram. Följden av detta blev enligt undersökningen att utvecklarna inte utnyttjade sina grafiska anteckningar som grund för valideringen.

6. Analys

Efter att ha presenterat ovan-stående metodkategorier har jag funnit att de skiljer sig avsevärt från varandra när det gäller tillämpning, samarbete mellan användare och utvecklare samt i vilka former av projekt de anses lämpliga etc. Om man jämför de olika metodkategorierna med varandra finner man att graden av användarmedverkan skiljer sig markant från metod till metod. Jag har därför valt att göra en sammanställning över de metoder jag presenterat här ovan och analyserat dem utifrån ett samarbets- och kommunikationsperspektiv. Sammanställningen tar upp var metod för sig och koncentrerar sig på hur användare och utvecklare samarbetar i respektive metod. Jag har vidare försökt belysa en del positiva och negativa sidor hos varje metod. Under egna kommentarer presenterar jag mina egna åsikter. Om det inte är skrivet något under själva metodkategorinamnet beror detta på att författaren som jag använt som källa inte har uttryckt sig om just kommunikationen mellan användare och utvecklare.

Prototyping

Prototyping använder enligt Wieringa (1996) sig utav användare kontinuerligt när det gäller validering. Med kontinuerligt menar jag inte under hela prototyparbetet utan så fort någon utvärdering skall utföras, oavsett om valideringen sker genom naturligt språk eller genom noggrant kontrollerade försök i ett laboratorium. Skillnaden mellan ett vanligt samtal och videofilmning med tillhörande kommentarer är nog störst för användaren. Båda sätten har sina fördelar och nackdelar.

Egna kommentarer

Om man ser till fördelarna med att validera i ett laboratorium kan man nämna att utvecklaren ser direkt hur användaren jobbar med prototypen och kan lättare se vad som är komplicerat för användaren att utföra. Kan sedan användaren på ett bra sätt kommentera varför han finner att något är svårt så underlättar detta arbetet för utvecklaren väsentligt. Det som kan vara negativt är att användaren kan lätt känna sig som ett försöksdjur istället för en i utvecklingslaget och detta kan bidra till ett sämre samarbete mellan honom och utvecklaren. Man kan lätt förstå om användaren känner sig utanför då han kan tvingas att sitta och utföra diverse manövrar för sig själv i ett laboratorium. Det är då lätt att säga att användaren inte alls behöver vara ensam utan kan ha utvecklaren vid sin sida. Men man bör nog betänka att om användaren känner sig lite utanför laget blir det inte bättre om han har ett ”proffs” vid sin sida som ser på när han gör fel och antecknar allt som försiggår. Det kan lätt bli att användaren tycker att det är hans kompetens som testas istället för prototypen. Om man istället samtalar på ett mer partnermässigt sätt kan nog teamkänslan hållas uppe och samarbetet flyta bättre.

Tittar vi sedan på prototyping utanför laboratoriet kan man säga att användaren känner sig lite mer på hemmaplan och kan därför slappna av och samarbeta bättre. Då kan det dock hända att utvecklaren får det svårare med sin del av arbetet. Han måste vid varje valideringsomgång dokumentera vad användaren har för sig och på detta sätt skapa sig en bild av hur användaren fungerar. Utvecklaren har inte heller chansen att visuellt återskapa valideringen av prototypen som han kan vid en videoinspelning. En balansgång mellan dessa båda sätt skulle kunna vara det bästa men det är nog svårt att uppnå denna balans. Om man från början av ett utvecklingsarbete kan främja en kamratlig atmosfär kan nog detta överbrygga alla känslor om underlägsenhet och att vara utanför.

Genomgångar
Genomgångar främjar enligt Wieringa (1996) kommunikationen mellan användare och utvecklare genom det sätt de utförs på.

Egna kommentarer
Att direkt välja ut användarrepresentanter för medverkan i valideringsprocessen vittnar om tilltro till användarna vilket bör göra samarbetet lättare. En sak som kan vara viktig och som togs upp tidigare i detta dokument är att det inte får finnas några hierarkiska klyftor mellan användare och utvecklare. Finns det sådana klyftor är chansen stor att användarna känner sig underlägsna och inte vågar kommentera eventuella fel och brister som de tycker sig se under genomgången. Detta kan dock undvikas genom att åter igen skapa temakänsla. Vad som också är viktigt är att användaren är fullt insatt i hur ett dataflödesdiagram är uppbyggt och hur det fungerar. Tar man sig tid att gå igenom hur diagrammet fungerar kan man spara åtskilligt med tid senare i valideringsprocessen och vinna både kvalitet och effektivitet dessutom. Med detta menas att om användaren är fullt insatt i hur ett diagram fungerar kan han förhoppningsvis bidra med mer utav sin erfarenhet och kunskap än om han inte har någon utbildning på diagram alls.

Simulering

Egna kommentarer
En simulering kan vara en ganska komplex sak för en användare eftersom man tittar på hur det nya systemet skall arbeta. Det är inte alls säkert att användaren vet hur systemet fungerar inuti. Han är oftast van vid att se gränssnittet på systemet och följer systemets handling genom de funktioner han använder. Detta innebär att han endast kan bidra med vilken sorts indata systemet skall kunna ta emot och vilken utdata det skall leverera. Men med hjälp från utvecklare kan han sättas in i hur systemet fungerar och därefter eventuellt bidra med information om hur de olika operationerna skall fungera. Enkelt förklarat kan han upplysa utvecklaren om vad som förväntas komma ut som utdata efter användning av en viss funktion. Detta gör att användaren kan vara värdefull vid en validering även fast han inte är helt insatt i hur systemet eller funktionen som valideras fungerar.

Animation
Loucopoulos och Karakostas (1995) påpekar att animeringar ofta används till realtidssystem därför att man tillåts följa händelseförloppet i systemet från det att indata har laddats in till det att en reaktion har ägt rum. Realtidssystem används som styrande enhet vid övervakning i t.ex. fabriker, kärnkraftverk etc. Det är därför viktigt att systemet reagerar i tid och reagerar rätt.

Egna kommentarer
Dessa miljöer som jag just presenterade är som ni förstår ganska komplicerade, detta gör det svårt för utvecklaren att skaffa sig all den domänkunskap som krävs för att sätta sig in i problemet. Men genom ett nära samarbete med den personal som jobbar med systemet dagligen kan han nå en djupare förståelse. Jag anser att i ett sådant här läge kan det faktiskt vara så att utvecklaren känner sig underlägsen mot användaren då utvecklaren kan ha svårt med en del domänkunskap. Här gäller det för båda parter att de går in för att upplysa varandra om sina respektive svårigheter. När det gäller till exempel ett kontrollsystem för en process är det viktigt att vid valideringen testa alla tänkbara förlopp som kan inträffa. Användaren och utvecklaren måste här hjälpa varandra att hålla alla viktiga företeelser i minnet. Visst finns det dokumentation att falla tillbaka på, men ett nära samarbete mellan båda parter kan frambringa

en mer komplett validering än om utvecklaren jobbar helt själv med denna uppgift. Om användaren är med vid valideringen kan han också medverka till att inget förbigås eftersom han kan rutinerna och processen bättre än utvecklaren.

Kontrollistor för validering
Den här sortens validering tar verkligen hänsyn till att användaren skall förstå vad han läser och under inga omständigheter känna sig underlägsen mot utvecklaren. Under de sju saker som Sommerville och Saywer (1998) påpekar att man bör betänka står det att man måste försäkra sig om att läsaren av listan förstår vad som menas med ett krav.

Egna kommentarer
Ovanstående visar på hänsyn till de som inte har samma begreppsram som utvecklaren då det oftast är han som sammanställer listorna. Listans språk måste därför vara väl balanserad mellan generalitet och detaljrikhet. Den får inte heller innehålla för många punkter eftersom detta ger upphov till att läsaren lätt kan glömma av vilka punkter listan innehöll. Vidare står det också listat under fördelar att kontrollistor hjälper till att introducera kravvalideringsprocessen för ovana. Kontrollistan ger dem tips om vad de skall titta efter och detta ger dem en känsla av att vara en i laget även fast deras rutin inom området är marginell.

Användarmedverkan
Det finns många aspekter att betänka vid användarmedverkan. Kraven som utvecklare ställer på användare är många och svåra. I den här undersökningen har jag tyvärr inte presenterat användarnas åsikter om utvecklare. Detta gör att studien inte är objektiv inom det här området. Men om vi sammafattar vad utvecklare tycker om användare och vad man bör tänka på vid användarmedverkan säger Emam och Madhavji (1995) att användare skall vara med vid valideringen och att det finns tillvägagångssätt som främjar användarmedverkan. Ovanstående artikel belyser också vikten av att välja rätt användare. Rätt användare är insatt i problemdomänen och har sina medarbetares respekt. Det presenterades också en önskelista som var ganska krävande för den användare som skulle uppfylla den. Jag själv anser att den var något överdriven då den i princip beskrev anställningskrav för en systemutvecklare.

7. Erfarenheter

Till att börja med vill jag säga att det här arbetet har varit svårt att genomföra rent materialmässigt. Valideringsprocessen som är en del av kravhanteringsprocessen får endast en bråkdel av uppmärksamheten i de källor jag har använt. Det kanske är på sin plats att säga att man inte kan generalisera alla författare som skriver om kravhanteringsprocessen men jag kunde dock skönja ett mönster hos de källor jag tittat på.

När jag började det här arbetet hade jag en naiv föreställning om att de tre delarna i kravhanteringen var relativt jämnt fördelade i litteratur om kravhantering. Sanningen är dock, med de källor jag har använt, att valideringen får endast en liten del av den totala uppmärksamheten omkring de tre delarna. Jag hade förhoppningar på att en del andra källor skulle visa sig vara värdefulla att konsultera men så var icke fallet. Ett exempel på en sådan källa var tidskriften Journal of Requirements Engineering som endast innehåller artiklar om just kravhanteringsprocessen. Jag hade stora förhoppningar om att det skulle finnas mycket information om just validering men även här stod kravframtagning och specificering i majoritet. Det skall dock tilläggas att de källor som jag har använt, till exempel Flynn och Warhurst (1994) pekar ut valideringsprocessen som ett svårt arbete med många påverkande faktorer. Om man sedan vill söka information om validering på Internet upptäcker man snabbt att de delar som får mest uppmärksamhet i kravhanteringsprocessen är just framtagningen av krav samt specificeringen. Det fanns dock några artiklar som innehöll information om valideringen men de flesta var avsedda för medlemmar av diverse universitet eller gick de att beställa för en påtaglig summa pengar.

Om jag sedan går över på själva valideringsprocessen i sig kan jag med facit i hand säga att när jag började undersöka valideringsprocessen hade jag en ganska enkelriktad uppfattning om vad som kunde ställa till problem. Men allt eftersom arbetet har fortskridit har min uppfattning ändrats och jag har kommit till nya insikter. När det gäller kommunikationen mellan användare och utvecklare jag fått erfara att den är beroende av många av en mängd olika faktorer.

Till att börja med måste samarbetet och dialogen mellan de bägge parterna grundläggas tidigt i projektet. Den första kontakten mellan användare och utvecklare bör tas så fort projektet har fått klartecken. Vid den här tidpunkten är det bra om man kan undersöka de användare som man kommer att vara beroende utav. Det påpekades i Emam och Madhavji (1995) att det är viktigt att få tag i rätt användare. Vad har då rätt användare för egenskaper? Det listades en hel del önskemål om just dessa egenskaper i ovan nämnda artikel och jag har skrivit en del kommentarer i samband med uppräkningen av önskemålen.

Det jag själv blev överraskad över var den fientlighet som existerade mellan användare och utvecklare enligt Emam och Madhavji (1996). Vikten av att få tag i användare med en positiv inriktad personlighet blev med ens väldigt klar för mig. Något som också på ett sätt överraskade mig var svårigheten att få tillgång till de användare som projektet behövde. Man förstår ju att företaget inte kan släppa till sina anställda hur mycket som helst, men samtidigt förväntar sig företaget att utvecklarna skall skapa ett system eller applikation som användarna trivs med och kan använda effektivt. För att kunna skapa ett system efter användarnas smak måste man ju få chansen att ta reda på vad som verkligen är användarnas smak. Det problem som jag förväntade mig vara det största men som kom lite i skymundan var på vilket sätt användare och utvecklare skulle föra sin konversation. Formellt eller oformellt språk var visserligen ett problem men inte så stort som jag trodde. Det poängterades visserligen i Flynn

och Warhurst (1994) men inte med den höga prioritering jag hade förväntat mig. Vidare visade det sig att användarna kunde hysa en viss ovilja mot att medverka i utvecklingsarbetet och det finner jag mycket underligt. Om man inte är helt och hållet bakåtsträvande bör man ju inse att utvecklingsarbetet är till för att förbättra sin egen arbetsmiljö. Detta är förstås min subjektiva åsikt och inget riktmärke för hur man bör tycka som användare.

7.1 Analys av insamlat material

Även om det var svårt att samla in material om validering är jag nöjd med det jag lyckades få tag i. Det mesta materialet är skrivet av erkända författare inom kravhanteringskretsar. Som jag påpekade tidigare i mitt arbete tycker jag att det är mycket värt att få tag i material som skrivits av författare som har undervisat i det aktuella ämnet. Författare med undervisningsvana håller sig ofta ajour med vad som händer inom sina områden när det gäller utvecklingar och framsteg. Att sedan delar av materialet är sammanställt utav intervjuer med utvecklare som har erfarenhet av arbete med just validering gör materialet starkt förankrat till verkligheten (Flynn och Warhurst 1994). Jag kan också säga att efter att ha analyserat ett antal metoder för validering och artiklar skrivna om den här processen att man kan välja metod efter projekt. Det finns metoder som hanterar kommunikationsproblemet bättre än andra. Men jag fick ett antal helt nya vinklar på problemet som jag inte hade tidigare. Detta var givetvis positivt och det ger mig ytterligare förståelse för varför valideringen anses vara svår.

8. Slutsatser

Det som jag kommit fram till under det här arbetet pekar klart och tydligt på hur viktig kommunikationen är för valideringsarbetet. Men det pekar också på att kommunikationen är viktig genom hela utvecklingsarbetet. Under mina undersökningar har det kommit fram att metodkategorierna som jag beskrivit i kapitel två skiljer sig från varandra på diverse sätt. En del främjar kommunikation och samarbete från början till slut, andra verkar inte vara direkt skapade för användarmedverkan.

8.1 Metodkategorier

Om man börjar med att kontrollera vilka metodkategorier som är direkt lämpade för användarmedverkan så finner man följande:

- Checklistor är direkt skapade med användaren i åtanke. Checklistornas tumregler eller faktorer för ett lyckat arbete belyser de problem som måste undvikas för att få ett bra samarbete mellan användare och utvecklare (Sommerville och Sawyer 1998). Checklistor hjälper inte bara till att fokusera användarnas blickar på viktiga saker utan de får användarna att känna sig delaktiga i processen utan några krav på utbildning. Detta anser jag vara en metodkategori som värnar om kommunikationen och samarbetet mellan användare och utvecklare.

- Om vi fortsätter med genomgångar så är detta en metodkategori som också är lämpad för användarmedverkan om den ges rätt förutsättningar. Vid de möten som hålls under valideringen när man använder genomgångar, krävs det att användarna förstår hur man tolkar till exempel ett dataflödesdiagram. Ger man användarna chansen att lära sig detta har man stora möjligheter att få positiva gensvar under valideringsprocessen. Desto mer en användare förstår om dataflödesdiagram desto mer kan han bidra i valideringsprocessen.

- Animering och simulering är liknande på många sätt och de liknar varandra även när det gäller användarmedverkan och kommunikation. Om ovanstående metodkategorier inte görs för komplicerade är de mycket användbara vid validering. Men det skall tilläggas, att om den grafiska beskrivningen hos animeringen eller testfunktionen hos simuleringen görs för invecklade kan användaren känna att han inte förstår och inte kan ta till sig den information som krävs för att validera. Med komplicerad menar jag till exempel att graden av detaljrikhet har här en stor inverkan på hur användaren kommer att uppfatta systemet och utföra valideringen. Detaljrikheten bör avvägas till kunskapen hos den användare som medverkar i valideringen. Det finns ingen anledning till att visa hur systemet eller funktionen skall fungera i minsta detalj om inte detta krävs. Användaren kommer antagligen endast att komma i kontakt med de yttre funktionerna, det vill säga vad gör systemet och inte hur gör systemet.

8.2 Summering av metodkategorier

Med allt ovanstående i tankarna kan man säga att det finns en klar skillnad mellan metodkategorierna. Genomgångar och checklistor gynnar användarmedverkan mer än simuleringar och animeringar. I uttrycket gynnar ligger tyngdpunkten på gynnar i grundutförandet. Animeringar och simuleringar kan vara helt acceptabla för användarmedverkan om man vidtar vissa mått för att få användaren effektiv från början. Ingen

av de metodkategorier jag presenterat använder sig av komplicerat språk vid kommunikationen med användare, detta gör att själva kommunikationen inte påverkas av hur insatt en användare är i formella specifikationer. Dessa kommentarer är mina egna och uttryckta i generella ordalag, men man kan dock säga att om animeringar och simuleringar utförs utan att tänka på användarens utbildning och erfarenhet, finns risken att användaren förblir en passiv partner i valideringsprocessen. Känner användaren att han inte kan bidra med något på grund av valideringens komplexitet blir både kommunikationen och samarbetet lidande. Detta kan få till följd att fel slipper igenom den kontroll som valideringen innebär. Följden av detta har jag diskuterat i tidigare kapitel.

9. Diskussion

Om man väljer att titta på kommunikations- och samarbetsbiten i ett större perspektiv finner man att kommunikation och samarbete är något som måste etableras vid ett utvecklingsprojekts start. Att enbart anordna en träff för alla inblandade räcker inte tror jag, man måste få en översyn på vilka användare som verkligen kan bidra med något till projektet. Givetvis måste användaren vara insatt i de funktioner som man skall bygga det nya systemet omkring, men han bör också ha en viss social kompetens. En användare som hyser motvilja mot den utveckling som skall ske är inte direkt lämplig för projektet. Det här examensarbetet har mycket kretsat på vad utvecklare kan göra och bör göra för att få användaren effektiv och nöjd, men användaren har ju också ett visst ansvar för att göra det bästa han kan för att projektet skall lyckas. Vid projektets start bör det poängteras att ett lyckat projekt ofta kräver samarbete från båda håll, för det kvittar hur mycket utvecklaren anstränger sig om användaren inte bryr sig. Utvecklare bör också förklara att de är beroende av användarnas kunskap och erfarenhet för att kunna ge användarna ett så bra system som möjligt. Givetvis finns det två sidor i en kommunikation och båda måste göra sitt för att dialogen skall flyta. En utvecklare måste också ha i tankarna att det kan finnas en viss motvilja mot “kostymprydda världsmästare” som kommer ut på arbetsplatsen och skall bestämma hur allt skall gå till. Det jag skrev i föregående mening är inget citat men väl ett exempel på hur användare kan reagera på att utvecklare kommer ut med en otrevlig översittsattityd. När man tar första kontakten är det väldigt viktigt att man inte utstrålar arrogans och nedvärderande tankar om omgivningen då detta garanterat kan sätta käppar i hjulet för allt kommande samarbete. Detta är på inget sätt unikt för dataområdet utan detta existerar på alla yrkesområden. Tar man sedan en titt på vad arbetsgivare rankar som bra egenskaper så ser man att den sociala kompetensen är oerhört viktig. Kunskap är givetvis väldigt viktigt men kunskapsbrist inom vissa områden kan åtgärdas, det är inte fullt så lätt att göra med den sociala kompetensen.

9.1 Uppslag till fortsatt arbete

Vill man utföra fortsatta studier inom det här området vill jag rekommendera en undersökning på vilka mått och steg man bör ta från början för att få kommunikation och samarbetet att fungera effektivt från start. Att det finns flera faktorer som påverkar kommunikationen mellan användare och utvecklare gör att området att studera blir vidare. Man kan undersöka både själva kommunikationen och det sociala förhållandet mellan användare och utvecklare. För är det något som har poängterats i de källor jag har använt så är det att förhållandet mellan parterna i ett projekt måste vara av det vänliga och respekterande slaget för att uppnå önskat resultat. En undersökning med användare och utvecklare som huvudaktörer och samarbete som huvudsakligt kritera för undersökningen borde kunna ge intressanta resultat och åsikter.

9.2 Kritik mot eget arbete

Man skall granska sitt eget arbete med kritiska ögon och jag har försökt att gjort just detta. Nedan följer två punkter som jag tyckte att jag ville dela med mig utav. Det finns givetvis mer kritik mot detta arbete men jag valde att presentera de här två eftersom dessa var de mest uppenbara enligt mig själv.

Fler infallsvinklar
Om jag skall ge mitt eget arbete kritik vill jag först och främst säga att arbetet presenteras ur en utvecklares synvinkel. Det vill säga att arbetet inte har full objektivitet. Men till mitt försvar kan jag säga att det inte är lätt att hitta litteratur som beskriver systemutvecklingsarbete med användarens ögon. Varför intervjuar du då inte användare som har varit med om ett utvecklings arbete?, säger kanske någon. Anledningen till detta är att användare sällan varit med om flera utvecklingsprojekt, vilket gör att deras åsikt om utvecklare endast grundas på just den utvecklare som var på deras arbetsplats. Jag vill ha mer generella åsikter om utvecklare som kan påpeka fel och brister hos utvecklare allmänt sett.

Mer källor
Just bristen på källor med översiktligt kunskap har jag tagit upp i min analys över material i kapitel sju. Men det kan inte nog poängteras att valideringen endast får en bråkdel av den totala uppmärksamheten kravhanteringsprocessen.

Referenser

Andersen, E.S. (1994), *Systemutveckling - principer, metoder och tekniker,*

Boehm B. W. (1980) *Software Engineering Econimics* Prentice Hall, Englewood Cliffs NJ. Book Company (UK) Limited. ISBN 0-07-707843-8.

El Emam, K. & Madhavji, N. K. (1995) *A field study of Requirements Engineering Practices in information System Development*, Proc. Second International Symposium on Requirements Engineering, York, England.

Flynn, D.J. & Warhurst, R. (1994) *An emperical study of the validation process within requirements engineering.* Informations Systems Journal 4 , pp 185-212. http://www.comp.lancs.ac.uk/computing/resources/re-gpg/re-gpg8_4.html.

IEEE- Std (1990) *IEEE Standard Glossary of Software Engineering Terminology*.

Loucopoulos P. Karakostas V. (1995) *System Requirements Engineering*. McGraw-Hill

Lubars, M., Potts, C., & Richter, C. (1993) *A review of the state of the practice in requirements engineering.* Proc IEEE Symposium on Requirements Engineering, San Diego , California.

Macaulay, L. (1996) *Requirements Engineering.* Springer-verlag Berlin Heidelberg New York.

Persson, A. (1995) Systemutveckling 1 (kurs). Högskolan i Skövde, institutionen för datavetenskap.

Sommerville, I. & Sawyer, P. (1998) *Requirements Engineering, A good practice guide* Studentlitteratur, Lund. ISBN 91-44-31042

Wieringa, R.J. (1996) *Requirements Engineering, Frameworks of understanding*. John Wiley & Sons Ltd. Baffins Lane, Chichester West Sussex PO19 1UD, England.