Uppgörandet av kravspecifikation för orderuppföljningssystem

Mathias Börg

Examensarbete för ingenjörsexamen (YH)

Utbildningsprogrammet för informationsteknik

Vasa 2015

EXAMENSARBETE

Författare:	Mathias Börg
Utbildningsprogram och ort:	Informationsteknik, Vasa
Handledare:	Kaj Wikman

Titel: *Uppgörandet av kravspecifikation för orderuppföljningssystem*

Datum 26.11.2015 Sidantal 27 Bilagor

Abstrakt

Detta examensarbete utfördes åt avdelningen Content Distribution, Technical Information, Services på företaget Wärtsilä Finland Oy. Examensarbetet behandlar planering, utförande samt färdigställning av kravspecifikationsdokument för ett program. Avdelningen har ett behov av ett program för orderhantering och en ny version skulle utvecklas av programmet. Inget kravspecifikationsdokument fanns från tidigare, därför fick jag uppdraget att göra ett sådant.

Resultatet blev ett kravspecifikationsdokument som kan användas både i vidareutvecklingen av det aktuella och i kommande versioner av programmet.

Språk: svenska Nyckelord: kravspecifikation, programvaruutveckling, kravinsamling

---

BACHELOR'S THESIS

Author:	Mathias Börg
Degree Programme:	Information Technology, Vasa
Supervisor:	Kaj Wikman

Title: *Creation of a software requirements specification for an order handling system.*

Date 26.11.2015 Number of pages 27 Appendices

Summary

This thesis was made on request from the department Content Distribution, Technical Information, Services at the company Wärtsilä Finland Oy. The thesis covers planning, execution and completion of a software requirements specification. The department has a need for order handling software and a new version of the software would be developed. Due to the lack of any previous documentation available I was given the task to write a software requirements specification.

The result was a software requirements specification that can be used in the development of this version of the software and possible future versions.

Language: swedish Key words: software requirements specification, development

---

OPINNÄYTETYÖ

Tekijä:	Mathias Börg
Koulutusohjelma ja paikkakunta:	Tietotekniikka, Vaasa
Ohjaaja:	Kaj Wikman

Nimike: *Ohjelmiston vaatimusmäärittelyn tekeminen tilaustenhallintajärjestelmään*.

Päivämäärä 26.11.2015 Sivumäärä 27 Liitteet

Tiivistelmä

Tämä opinnäytetyö oli tehty pyynnöstä osastolle Content Distribution, Technical Information, Services yrityksessä Wärtsilä Finland Oy. Opinnäytetyö käsittää ohjelmiston vaatimusmäärittelyn suunnittelun, toteutuksen ja valmistumisen. Osastolla pitää olla käytössä tilaustenhallintaohjelma ja uusi versio ohjelmasta oli kehitettävä. Koska mikään vanha dokumentaatio ei ollut saatavilla sain tehtäväkseni tehdä ohjelmiston vaatimusmäärittelyn.

Lopputulos oli ohjelmiston vaatimusmäärittely, jota voi käyttää ohjelmiston uuden version kehityksessä ja myös tulevaisuudessa mahdollisissa uusissa versioissa.

Kieli: ruotsi Avainsanat: ohjelmiston vaatimusmäärittely, kehitys, ohjelmisto

---

Innehållsförteckning

Figurförteckning

1 Introduktion

I det här kapitlet finns bakgrunden till arbetsuppgiften samt en introduktion till arbetsgivaren.

1.1 Arbetsgivaren

Wärtsilä är ett bolag som är i dagens läge en världsledande leverantör av kompletta kraftlösningar för marin- och energimarknaderna. Bolaget grundades 1834 och började som ett sågverk i Karelen.

Idag finns bolaget i närmare 70 länder och har nästan 18000 anställda. I Finland uppgår arbetsstyrkan till ungefär 3600 personer på orterna Vasa, Åbo, Helsingfors samt Esbo. Wärtsiläs huvudkontor finns i Helsingfors. [1]

Figur 1. Wärtsiläs huvudkontor i Helsingfors [2]

1.2 Bakgrund

Den avdelning på Wärtsilä som sköter orderhantering av dokumentation för motorer och andra produkter är Content Distribution som hör till Technical Information. Avdelningen har ett behov av ett orderhanteringssystem för att hantera de beställningar på dokumentation som de får samt möjlighet att följa upp aktuellt status för olika pågående dokumentationsbeställningar.

Avdelningen hade använt olika metoder och program för att utföra dessa arbetsåtaganden. Nu fanns ett behov för ett nytt och bättre program. Problemet var att ingen tidigare hade sammanställt ett kravdokument över vad som behövs av ett sådant program. Det har gjort att varje gång man bytt system har man behövt reda ut i princip allt på nytt. Därför fanns nu ett behov av att uppgöra ett sådant dokument.

Vid utvecklingen av det program som just nu används (TIMS) hade utvecklingen gått till så att funktioner lades till vartefter någon sade att de behövdes. Problemet med detta var ju att i det skedet nya funktioner som skulle behövas till programmet så fanns redan basen för programmet uppbyggd. Det gjorde att vissa funktioner inte kunde implementeras ordentligt och andra funktioner inte alls.

1.3 Uppgift

Det som efterfrågades av mig var att jag skulle uppgöra en kravspecifikation som skulle gå att använda vid utvecklingen av det nya programmet samt sparas som referensdokument. Jag skulle analysera det nuvarande programmet, ta reda på krav från avdelningarna i Vasa samt kontrollera om de avdelningar som finns i Italien, Holland och Norge hade andra krav för programmet.

Jag skulle på olika sätt samla in kraven samt göra kravspecifikationen så allmän som möjligt och så lite influerad av de föregående versionerna som möjligt.

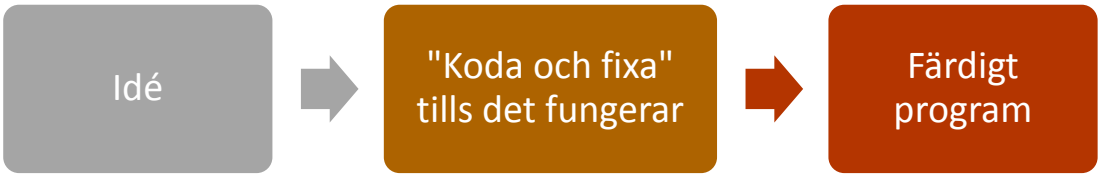
Personerna jag skulle fråga till vad de behövde programmet till hade olika bakgrund och har olika arbetsuppgifter så till stor del visste de inte hur utveckling av program går till. Så min uppgift var även att formulera frågor åt användarna så att de svarar på så sätt att jag sedan kunde analysera deras svar och formulera krav från dessa.

2 Teori

Detta kapitel innehåller teori om olika programutvecklingsmetoder samt om kravhantering.

2.1 Programutvecklingsmetoder

Före man börjar med ett IT-projekt måste man välja utvecklingsmetod. Det finns många olika metoder man kan använda sig av, de vanligaste nämns här. Den enklaste metoden är den som i princip alla utvecklare åtminstone någon gång använt sig av. ”Koda och fixa” betyder kort och gott att man från att ha fått en idé till ett program bara börjar koda och vartefter man stöter på diverse problem så åtgärdar man dessa. Denna metod är inte bra men är mycket enkel att följa. [3]



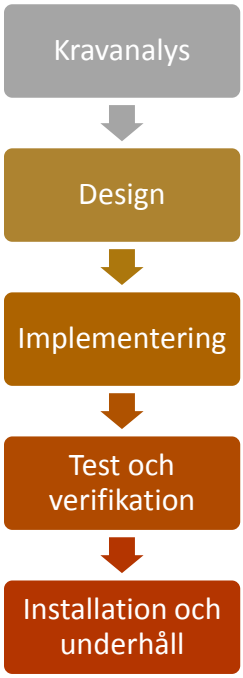
Figur 2. “Koda och fixa” metoden.

Om man utvecklar ett mycket litet program kan denna metod vara den bästa att följa då det är få variabler att planera efter. Om man däremot börjar utveckla ett större program med denna metod finns det stor risk att man stöter på sådana problem som gör att man i princip måste börja från början igen. Det blir också i princip omöjligt att beräkna tidsåtgång med denna metod.

2.1.1 Vattenfallsmetoden

Vattenfallsmetoden är en välbeprövad metod som använts sedan 1970-talet. I dagens läge har den fått kritik för att vara gammaldags och icke flexibel. Men det finns också många positiva aspekter till denna metod då den är välbeprövad och har klara steg som man skall ta sig igenom för att komma till den slutgiltiga produkten. Med denna metod får man bra dokumentation då man dokumenterar de olika skedena före man går vidare.

Metoden är en sekventiell metod som betyder att man går steg för steg genom den fram till slutprodukten. Det som kan vara ett problem med denna metod är att det är svårt att gå tillbaka och ändra i de redan slutförda stegen. Det gör denna metod olämplig för vissa typer av projekt där man t.ex. har svårt att reda ut vad kunden sist och slutligen önskar.



Figur 3. Vattenfallsmetoden.

I kravanalyssteget tar man reda på kraven och specificerar dessa i ett dokument.

I designsteget utgår man från kravspecifikationen och planerar hur programmet skall se ut och de olika funktionerna fungera.

I implementeringssteget programmerar man utgående från de tidigare gjorda stegen.

I test och verifikationssteget testar man att programmet funkar samt att det uppfyller de tidigare fastställda kraven.

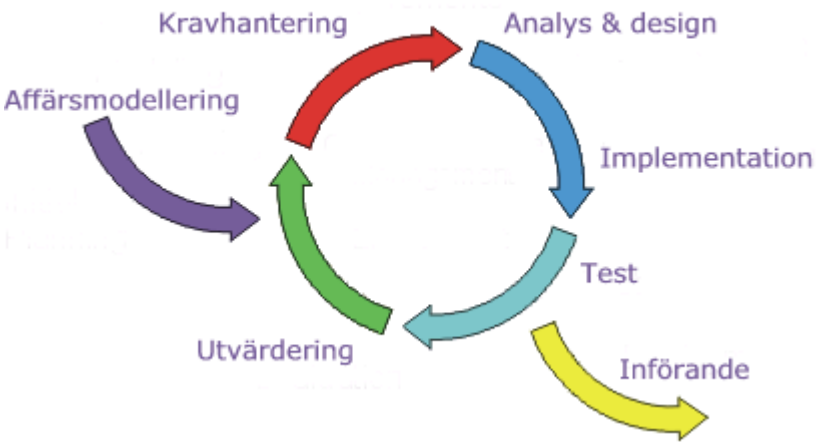
I det sista steget installation och underhåll tar man slutprodukten i bruk börjar vid behov en underhållsfas.

2.1.2 Iterativa metoden

Den iterativa metoden är en annan typ av metod som skiljer sig från vattenfallsmetoden huvudsakligen på det sätt att man går igenom de olika stegen flera

gånger istället för bara en gång. I denna metod delar man på sätt och vis upp projektet i flera delar som man slutför skilt för sig. På det sättet får man relativt snabbt en del färdig att testas så att man t.ex. kan få kommentarer av slutkunden på ett mera konkret sätt då det finns något att visa upp som fungerar.

Denna metod har dock också sina nackdelar varav en är att det t.ex. efter flera iterationer av utvecklingen uppkommer ett behov som kräver att även de tidigare gjorda iterationerna som redan är kodade skulle göras om. Det kan också vara svårt att uppskatta tidsåtgång för hela projektet med denna metod. [4] [5]



Figur 4. Iterativa metoden [4]

2.1.3 Rational Unified Process (RUP)

Rational Unified Process (RUP) är en iterativ process som har blivit vanlig att använda i dagens läge. RUP är en metod som är ganska lik vattenfallsmetoden men den största skillnaden är den att RUP är baserad på iterationer. Den kan delas upp i fyra stadier:

Förberedelse – Steget då man startar projektet och tar reda på vad man skall göra. Utifrån detta gör man upp en projektplan.

Etablering – I det nästa steget kontrollerar man om problemet är lösbart som man planerat i projektplanen. Man utvecklar en funktionell prototyp för att ta reda på om de mest viktiga problemen kan lösas. Man kanske kan använda sig av denna kodbas då

man går vidare i utvecklingen, men det skiljer sig från fall till fall. Om projektplanen inte håller måste man gå tillbaka till det första steget och göra det på nytt.

Konstruktion – I detta steg som är den första produktionsfasen börjar man producera den kod som behövs för de olika delarna i projektet. Man jobbar i iterationer där varje iteration tar fram en fungerande version. Detta steg är färdigt då man kommit fram till ett program som fungerar samt kan börja användas av slutanvändarna.

Överlämning – Detta steg är den andra produktionsfasen där man kontrollerar att allting verkligen fungerar i vanlig användning samt att det enligt slutanvändarna uppfyller deras förväntningar på programmet. Också denna fas görs i iterationer där man förändringar i kod och dokumentation. [3]

2.1.4 Scrum

Scrum är en så kallad lättviktig metod som inte bygger på traditionell projekthantering. I den här metoden använder man sig inte av dokumentering och dokumentation på samma sätt som i andra metoder. Man använder sig av en vision för hur programmet skall bli och utifrån denna gör man upp en lista över funktioner och krav som programmet måste kunna utföra för att kunna betecknas som färdigt.

I ett projekt som använder denna utvecklingsmetod finns tre roller. Utvecklare, produktägare samt scrummaster. Scrummastern är inte det samma som projektledare men är ändå en person som har ett ansvar att hjälpa till under processen. Alla som är del av utvecklingsteamet har ett kollektivt ansvar för utvecklingen av programmet. Produktägaren är den som håller koll på vad som skall göras och varför. Utvecklarna är de som utför huvuddelen av utvecklingen under projektet. [3]

2.2 Kravhanteringsteori

I denna sektion beskrivs varför det är bra att göra en kravspecifikation samt olika metoder man kan använda sig av för att skapa kravspecifikationen.

2.2.1 Mål med en kravspecifikation

Ett vanligt scenario som kan inträffa på vilket bolag som helst är följande. En vanlig användare har ett behov som deras program inte klarar av i nuläget. Användaren får en vision av hur problemet skulle kunna lösas och går till programmeraren och förklarar visionen och undrar hur länge det tar innan den nya programvaran är färdig.

När ett behov uppstår för ett nytt program finns ofta en idé till hur problemet skulle kunna lösas. Då kan det kännas som det är ”bara att börja programmera” och lösa möjliga problem som uppstår efterhand. Det man kanske inte direkt tänker på är att idén till hur man skall lösa problemet är bara en vision och det är inte sagt att det går eller är rätt sätt att lösa problemet på detta sätt.

Ofta är också den som har behov av programmet och programmeraren olika personer. Det kan göra att den som begär programmet av programmeraren tycker det räcker med visionen och vill ha t.ex. en tidsuppskattning för hur länge det kan ta att skapa programmet. Om programmeraren då skulle börja utveckla programmet utan att ha mera information än visionen är det en stor sannolikhet att slutresultat inte blir bra.

”En orsak till att IT-projekt ofta blir försenade och betraktas som misslyckade är svårigheten i att omvandla den vision som finns till en fullgod specifikation. Missförstånd uppstår lätt mellan kund och utvecklare.” [3]

Man måste göra en ordentlig utredning av det problem som finns och om visionen faktiskt är den verkliga lösningen för problemet. Det kan finnas andra problem som ligger bakom det problem som visionen skulle lösa. Då är det de problemen man skall sikta på att lösa för att den vägen uppfylla den ursprungliga visionen. Då får man krav som motsvarar de verkliga behoven.

Även om man aldrig igen läser kravspecifikationen efter man skrivit den har den gjort nytta. Man har analyserat vilka verkliga behov som finns, diskuterat olika sätt man skulle kunna lösa saker på och fått en klarare bild av vad som egentligen skall göras.

Vad som också framkommer under en specifikationsskapning är att man ser saker som inte är nödvändiga att ha med i programmet.

Kravspecifikationen man skapar är inte bara intressant för den som skall programmera utan det finns många andra intressenter som har nytta av en ordentlig specifikation.

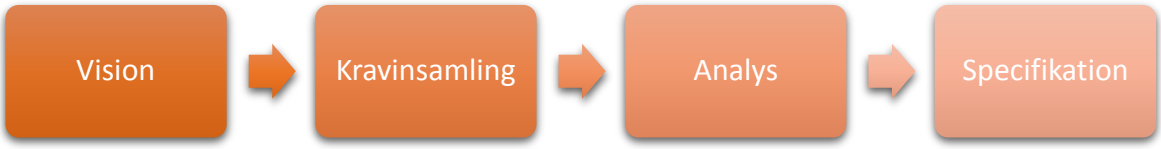
- Projektansvarig har nytta av den för att kunna beräkna kostnader och fördelning av arbete
- Utvecklare har nytta av den så att de vet vad de ska göra
- Testgrupp så att de kan förbereda testfall
- Den som beställt programmet så att de kan kolla att alla deras behov finns med i planeringen och utförande av projektet
- Nya personer kan enklare komma in i projektet efter start då de har dokumentation att tillgå
- Övriga inblandade som har intresse av programmet samt vilka uppgifter det kommer innehålla.

Så även om en kravspecifikation kan i början kännas onödig att göra kommer man ha mycket nytta av den under hela processen.

En kravspecifikation skall även innehålla annan information än bara vilka funktioner som programmet skall klara av. Möjlig kommunikation med andra system, prestanda samt tillförlitlighet är exempel på sådant som kan vara med i kravspecifikationen.

Hur invecklad kravspecifikation blir beror bland annat på projektstorlek.

[3] [6] [7] [8]



Figur 5. Från vision till specifikation.

2.2.2 Tillvägagångssätt för insamling av krav

Då man insett att man faktiskt skulle dra nytta av att göra en kravspecifikation kommer nästa steg. Hur skall man riktigt gå tillväga för att få insamlat alla de krav som finns. Det finns många olika metoder man kan använda sig av för att få kraven insamlade. Till att börja med måste man tänka på att alla tänker på olika sätt och kanske inte fullt förstår på vilket sätt krav skall vara formulerade för att de skall gå att använda sig av i en kravspecifikation.

”Det räcker inte att fråga vad en person vill ha (där svaret ofta blir ära, rikedom och evig berömmelse i olika mer nyanserade version).”

Man måste ta reda på vad de vill kunna göra med programmet men främst varför de vill kunna göra det. Dessa saker måste man ta reda på av alla som på ett eller annat sätt skall använda programmet eller kommer att påverkas av programmets funktion.

Man måste också försöka få de man frågar att frångå att för mycket tänka på det program de idag använder sig av. Om de enbart utgår från det då de berättar vad de har för krav på programmet är risken stor att det nya programmet blir väldigt likt det program som de använder i nuläget. De man frågar behöver nödvändigtvis inte heller förstå detta så det är ännu en orsak till att ha väl valda formuleringar på frågorna. Det är ju de grundläggande problemen man vill komma åt, inte skriva ner hur de gör i det nuvarande programmet. Som man frågar får man svar.

Man måste också ta hänsyn till sådana frågor som att olika intressenter kan ha väldigt olika krav. Man måste hitta en balans så att man inte t.ex. bara frågar ledningen vad programmet måste klara av. Då kan det hända att slutprodukten är ett program som t.ex. kan rapportera allting tydligt och bra till ledningen men att programmet inte klarar av de saker som behövs för att det egentliga arbetet skall gå att slutföra. Det bästa är ju att få de olika intressenterna att samarbeta så att de tillsammans kommer fram till de olika kraven.

I en kravspecifikation är funktionskraven viktigast och tar upp den största delen av en kravspecifikation. De flesta av kraven som kommer fram under intervjuer och förfrågningar är av denna typ av krav. ”Det skall vara möjligt att skapa en ny dokumentationsorder.” är ett exempel på ett funktionskrav.

Andra typer av krav kan även finnas med såsom följande:

- Kvalitetskrav: Denna typ av krav innefattar krav på användarvänlighet. T.ex. ”Det skall vara möjligt för användaren att förstå felmeddelanden”
- Flexibilitetskrav/skalningskrav: Denna kategori kan innehålla krav som kräver framtida möjligheter för förändringar av olika slag. Ett exempel kan vara att vissa delar av programmet skall vara möjligt att relativt enkelt gå att uppdatera.
- Plattformskrav: Dessa krav kan behövas om det finns färdiga system som programmet skall köra på. Till exempel kan det vara att det redan finns en Windows-miljö och därmed inte är ett alternativ för ett program som enbart skulle gå att köra på OS-X.
- Tillgänglighetskrav: Oftast måste program alltid vara tillgängliga att köra. Det är inte acceptabelt med ett program som funkar ibland. Ofta är kravet på upptid av systemet mera än 99 %.
- Prestandakrav: Dessa krav är viktiga för som användare kanske man inte direkt tänker på vad som händer med olika program vid kraftig belastning. Därför definierar dessa krav till exempel att en sökning på ett produktnummer i programmet inte får ta mera än 5 sek.
- Interoperabilitetskrav: Dessa krav kan behöva finnas om det finns andra program som det skall samverka med. Till exempel om det finns ett annat program som behöver ha informationen tillgänglig i XML-format för att det skall kunna använda sig av informationen.
- Säkerhetskrav: Dessa krav kan skilja sig kraftigt från program till program. Om det är ett program som skall vara tillgängligt via det publika internet kan det krävas mycket bättre säkerhet än om det är ett program som skall användas internt utan internetuppkoppling.

Här tas upp de mest relevanta kravtyperna för många projekt. Men det finns även flera andra kravtyper utöver dessa. Då man gör arbetet med att samla ihop alla dessa krav kan man börja med att dela upp arbetet i två delar. En del där man samlar in vad användarna vill kunna göra och en del med de mera tekniska kraven. Det man också skall tänka på är att vid insamlingssteget är alla typer av krav lika viktiga att samla ihop. Om man skulle missa något kritiskt krav såsom ett tillgänglighetskrav skulle det kunna bli mycket svårt om ens möjligt att åtgärda det i efterhand.

2.2.3 Kravinsamlingsmetoder

Det finns många olika metoder man kan använda sig av för att samla in kraven. Det beror lite på vilken typ av projekt det är och vilken typ av grupp man skall fråga kraven av vilken typ av kravinsamlingsmetod som passar bäst. Här nämns inte alla metoder men några av de mest användbara.

2.2.3.1 Möten i olika form

Möten i form av workshops där man samlar en grupp av de personer som har krav på vad programmet skall kunna utföra. Man kan ha ett möte där man tillsammans med gruppen börjar med att skriva ner korta huvudpunkter som programmet skall klara av. Då kan man få snabbt ner de övergripande kraven som man sedan kan gå in i mera detaljnivå med t.ex. diskussion. Man kan även med denna metod få fram prioriteringar på olika krav.

Skillnaden på ett sådant här möte och ett vanligt är att det i ett vanligt möte ofta är turordning och att det är mera officiellt. I ett workshop-liknande möte är det diskussionsfokus och mindre officiellt så mötesdeltagarna har enklare att diskutera fram olika saker som kanske inte ens är helt korrekt. Men man kanske får en bra idé från diskussionerna som man kan fundera vidare på. Då man aktivt jobbar i grupp på ett möte får deltagarna också känslan av ett mera gemensamt ansvar för vad de kommer fram till, tröskeln är lägre att säga vad man tycker om olika ämnen. I ett vanligt möte förväntar deltagarna sig oftare att mötesledaren skall vara ansvarig för att mötet skall gå framåt.

2.2.3.2 Intervjuer med enskilda personer

Det finns flera olika intervjumetoder man kan använda sig av. Oförberedd intervju, förberedd intervju eller en blandning av båda. I en oförberedd intervju har man inte förberedda frågor utan det är mera fri diskussion mellan två parter. På det här sättet kan man som intervjuare följa upp de svar man får och på det sättet gå in mer på djupet baserat på de svar den intervjuade ger. Nackdelar med det här är att man som intervjuare kan ha problem med att hålla sig till sak och att diskussionerna kan lätt bli på ett för litet område som kanske inte leder till några krav.

I en förberedd intervju har intervjuaren förberett frågor för hela intervjun. Frågorna skall vara planerade på det sätt att man får svar som kan användas och tolkas som ett krav. Det som är bra med det här sättet är att man kan t.ex. ställa samma frågor åt flera

personer och på det sättet få flera vinklingar på samma problem. Om det är fråga om ett riktigt stort projekt så att många personer är involverade i kravinsamlingen kan man på det här sättet ha flera intervjuare också. Ett problem kan dock vara att man blir för låst till de frågor man planerat. Det här sättet ställer också höga krav på intervjuarens förarbete för att man ska få med alla relevanta frågor.

Största skillnaden på dessa två metoder är att i den oförberedda intervjun kan det vara svårt att få allt viktigt med då man intervjuar flera personer då det inte är säkert man får med alla frågeområden varje gång. Man kan även utföra intervjuerna med en blandning av dessa metoder. Då kan man ha förberedda frågor men lämna dem åt sidan om man kommer in på relevanta ämnen man inte har frågor förberedda åt.

Före man alls utför intervjun behöver man åtminstone ta reda på lite om de personer man skall intervjua och vilken funktion de har i företaget så att man har lite bakgrundsinformation att börja intervjun med.

2.2.3.3 Observation

Denna metod innebär att kravinsamlaren sätter sig ner tillsammans med en person som använder den nuvarande programvaran. På detta sätt kan man få reda på krav som personerna annars kanske inte tänker på att berätta om på en intervju.

Det som kan vara problem med denna metod är att det kan vara svårt för användaren att kanske hitta ett bra exempel att utföra då kravinsamlaren iakttar personen. Den här metoden kan också vara väldigt tidskrävande beroende på vilken typ av arbetsuppgifter personen utför. Det kan även bli problem med att hålla relevant information separerad från övrig information då användaren antagligen använder sig av många olika program under en arbetsdag.

En positiv aspekt med denna metod kan vara att man som kravinsamlare får en helt annan bild av olika arbetsskeden då man själv ser dem istället för att personer berättar vad denne gör.

2.2.3.4 Enkät

Denna metod kan man använda sig av om t.ex. programmets användare är en så stor grupp människor att möten eller intervjuer inte är gångbart. Som i alla enkäter kan man använda sig av öppna frågor eller frågor med färdigsatta alternativ. Ett par fördelar med denna metod är att man kan snabbt nå en stor grupp personer och att man kanske får andra svar än man skulle fått under ett möte. Som nackdel kan man

räkna den tid det tar att planera en bra enkät samt det faktum att många inte har tid att svara ordentligt på en enkät. Man kan även få dåliga svar om någon till exempel förstått frågan fel.

2.2.3.5 Personas

Denna metod kan man använda sig av om det är svårt eller omöjligt att hitta specifika personer som kan ställa kraven på programmet. Man skapar en påhittad användare av systemet där man planerar in vilka funktioner denne person skulle använda. På det här sättet kan man utifrån dessa påhittade personer fundera ut vilka krav de skulle ha på programmet och då gå vidare utgående från detta. Man kan inte enbart använda sig av denna metod då den här metoden enbart hjälper till att få en enhetligare bild av krav då metoden i sig inte inhämtar någon information.

2.2.3.6 Kravdagbok

Den här metoden går att använda sig av om man till exempel skall uppgradera en gammal version av ett program med en ny version. Metoden går ut på att man begär att användarna under användning av det nuvarande programmet skriver ner problem med det befintliga programmet, den funktionalitet de använder idag samt sådana funktioner de saknar. Problem med den här metoden är att den kräver stort initiativ av användarna samt är tidskrävande. Metoden är dock enkel att tillämpa samt borde samla in de krav som behövs.

[3] [6] [7] [8] [13]

2.2.4 Bearbetning av kraven

Efter att man anser att kravinsamlingen är färdig går man vidare till nästa steg där man skall gå genom all insamlad information. Det här steget är minst lika viktigt som själva insamlingen av krav. Det är i det här steget man skall framställa de slutgiltiga kraven som skall vara med i dokumentet.

Det man behöver komma fram till är vilka krav som skall tas med, vilka krav som är redundanta med varandra samt prioritera kraven.

Ett av de stora problemen med det här steget kan vara att bedöma vilka krav och vilka personers krav som är viktigast. Om ett projekt till exempel har begränsad budget vilket ofta är fallet kan inte alla önskemål tas med. Denna bedömning måste göras objektivt och med tanke på användarna. I riktigt stora projekt kan det vara en skild

person som gör enbart detta då det är ett så pass viktigt skede i processen. Om man av något skäl såsom budget måste skala bort onödiga krav behöver man diskutera fram vilka krav som är de affärskritiska kraven. Detta gör man tillsammans med olika intressenter. Denna process kan vara problematisk att genomföra men nödvändig.

De krav som är redundanta med varandra kommer man fram till genom att gå igenom alla de inkomna kraven och analysera dem till den grad att man kommer till rotbehovet för kravet. Som tidigare sagt är ”Varför” ett väldigt viktigt nyckelord att fråga igenom hela processen, även i detta steg.

Om man beslutar att man enbart tar med de krav som skall implementeras och inte har med några sådana krav som kan prioriteras bort kan man ändå ha nytta av prioritering av krav. Till exempel kan det finnas en kärnfunktionalitet som sist och slutligen är den viktigaste delen av programmet, då får man fram det genom prioritering.

2.2.4.1 Formulering

Då man kommit så långt att man skall börja skriva de slutgiltiga kraven i kravspecifikationen finns det riktlinjer man bör följa för att texten skall gå att läsas. Det är viktigt att använda sig av ett konsekvent språk och göra dokumentet enhetligt på det sätt att alla krav är skrivna med samma struktur för att minimera risken för feltolkningar av texten. Till detta hör att man skall begränsa sin användning av synonymer då man skriver kraven då det inte är ett dokument att visa sin språkliga förmåga i.

I IEEE standarden 830-1998 står det följande rekommendationer för en kravspecifikation.

- Korrekt
- Entydig
- Komplett
- Konsistent
- Prioritetssorterad
- Verifierbar
- Modifierbar
- Spårbar

Då det gäller kravspecifikationer finns det dock inte en fast struktur som alltid måste följas utan man kan anpassa den mängd information man anser att är nödvändigt att

den skall innehålla. Därför är detta just en rekommendation och inte till exempel en mall.

Kraven skall hållas korta och vara rakt på sak utan onödiga utfyllnadstexter. Det finns flertalet ord som rekommenderas att man använder sig av då man skriver kraven. Dessa ord samt definitioner rekommenderas att inkludera en referens till dess innebörd i början av kravspecifikationen för att undvika möjliga feltolkningar.

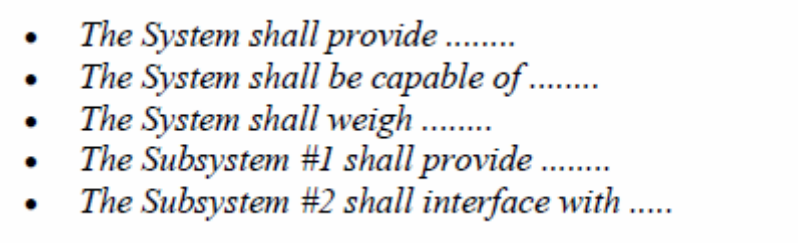
Skall – Betyder att det som beskrivs är ett absolut krav på att uppfyllas. Orden ”Måste” och ”Krävd” innehar samma betydelse enligt definitionen i RFC 2119.

Skall inte – Betyder att det som beskrivs är ett totalt förbud på att inträffa. Ordet ”Måste inte” innehar samma betydelse.

Bör/Borde – Betyder att det kanske finns orsaker till att det beskrivna inte skall uppfyllas men att man måste förstå innebörden och noga avväga före man går vidare. Ordet ”Rekommenderas” innehar samma betydelse.

Bör inte/Borde inte – Betyder att det kanske finns orsaker som gör att det kan finnas tillfällen då det oönskade beteendet är acceptabelt eller till och med användbart. Här gäller även att man bör förstå innebörden och noga avväga före man går vidare. Ordet ”Rekommenderas inte” innehar samma betydelse.

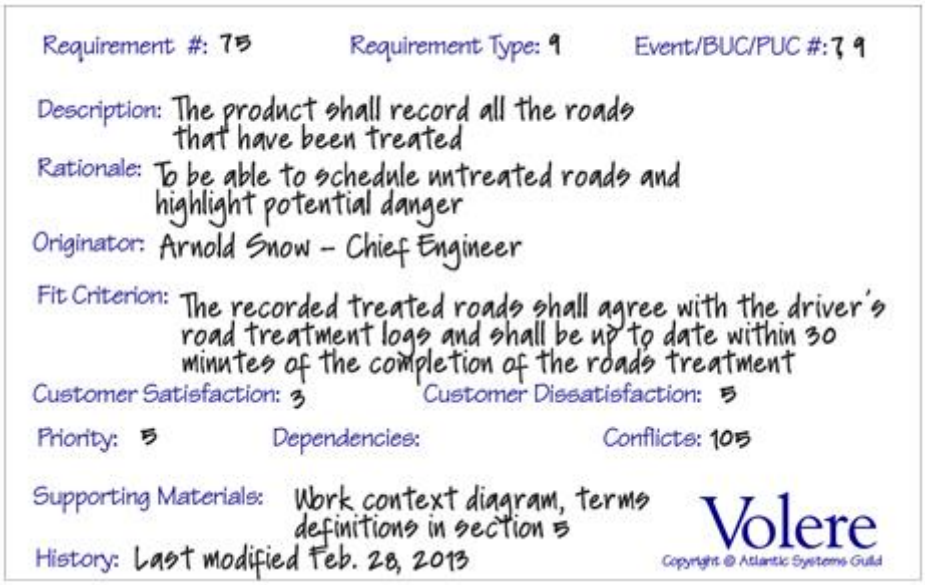
Även om dessa ord kan rekommenderas att använda sig av är det inget måste då en kravspecifikation går att skriva på många olika sätt.



Figur 6. Exempelmeningar med användning av ordet "Skall".

[3] [6] [7] [9] [10] [11] [14]

Ett exempel på ett funktionellt krav ur en kravspecifikation ur Voleres mall. [12]



Figur 7. Exempel på funktionellt krav.

2.2.5 Kravspecifikationens innehåll

I en kravspecifikation skall det inte enbart finnas kraven listade. Även andra punkter av information som är av intresse kan vara del av dokumentet. Det finns mängder av olika mallar samt förslag på vilka sorts punkter man kan använda sig av för att dokumentera olika typer av relevant information för projektet.

I boken Software Requirements av Karl E. Wiegers nämns följande rubriker i bokens mall för en kravspecifikation. [7]

- Introduktion. Innehåller syfte och vision för projektet. Om projektet är av större typ kan man uppgöra ett skilt visionsdokument, annars inkludera det i introduktionen av kravspecifikationen.
- Beskrivning av projektet. Beskriver vid behov mer detaljerat produkten kravspecifikationen gäller samt annan information såsom vilka som kommer att använda den färdiga produkten och annan nödvändig information om projektet.
- Systemets egenskaper. Detta är den största delen av kravspecifikation då det är här de funktionella kraven är listade.
- Externa kommunikationskrav. Är sektionen där krav för externa system är listade. Hit hör behov av kommunikation med befintliga system såsom andra program eller databaser som det planerade programmet vid behov skall kunna kommunicera med.

- Icke-funktionella krav. Här listas alla de icke-funktionella kraven som man sammanställt.
- Övriga krav. Denna rubrik kan man använda sig av om man har krav som inte passar under någon annan kategori. Man kan även lämna bort denna rubrik eller lägga till andra efter behov.

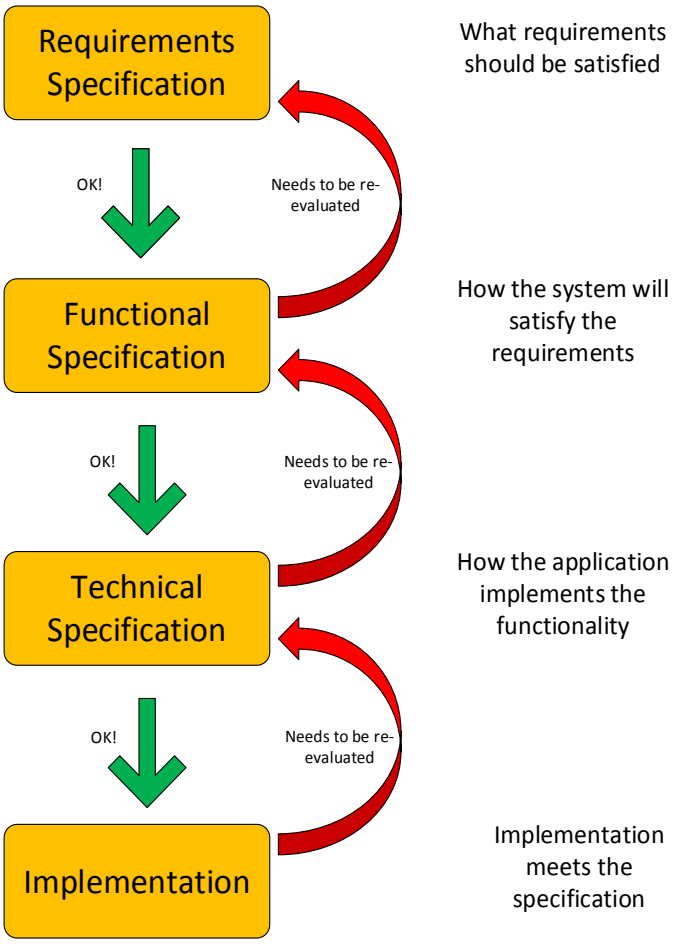
[7] [14]

3 Utförande

I det här kapitlet beskrivs arbetsprocessen som ledde till slutresultatet.

Före projektet påbörjades bestämdes att projektet skulle ha ett utvecklingsteam på tre personer. Det bestod av en som skulle vara projektansvarig, en programmerare samt jag som skulle vara ansvarig för att göra själva kravspecifikationen. Tidsplanen för utredningen av kraven samt uppgörandet av kravspecifikationsdokumentet var 5 mån. Orsaker till den relativt långa processen var att ingen av projektmedlemmarna skulle jobba med detta på heltid då jag ännu gick i skola 4 dagar i veckan samt de två andra projektmedlemmarna hade de vardagliga arbetsuppgifterna att sköta. Under hela processen jobbade projektteamet med det här projektet både den 5te vardagen varje vecka samt någon kväll per vecka. Mot slutet av projektet utökades arbetstiderna.

Som utvecklingsmetod valdes vattenfallsmetoden dels av den orsaken att ingen av projektdeltagarna skulle jobba med projektet på heltid samt att god dokumentation var önskvärd.



Figur 8. Grafisk presentation av den valda arbetsmetoden.

Då projektet påbörjades var många personer på ledigt och därför skulle den första frågerundan göras via e-post. För att kunna gå vidare med mera riktade frågor behövdes kunskap om vad olika personer överhuvudtaget gjorde i programmet. Så i det första mejlet ombads de tillfrågade personerna berätta vad de behöver kunna göra i programmet. Ett exempelsvar bifogades så att det skulle vara enklare att bearbeta de svar som inkom.

Example description of a task:

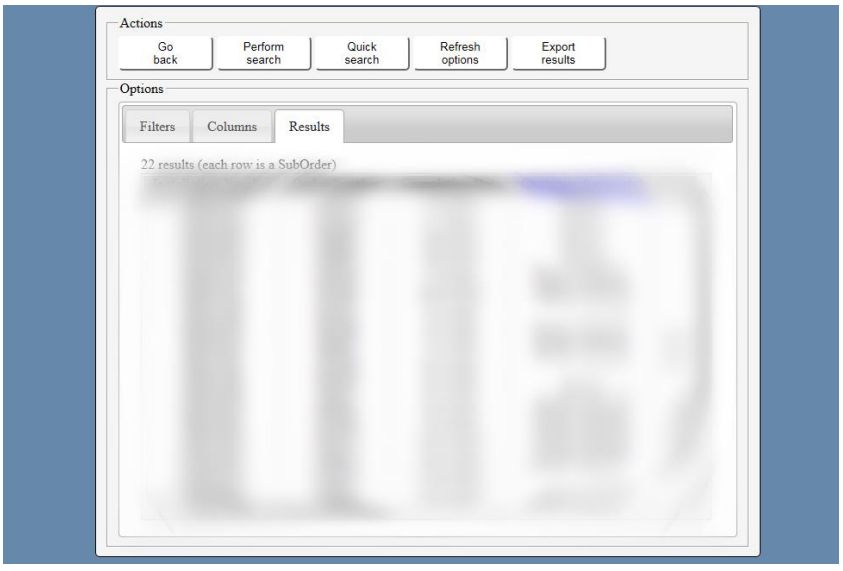
- I need to be able to list all spare parts catalogues that are assigned to me and estimated for delivery in the coming month.

Figur 9. Exempelsvar från E-post förfrågan

De svar som mottogs på förfrågan var av varierande grad. Somliga berättade steg för steg vad de gjorde i det nuvarande programmet medan andra skrev en mening om vad de behöver kunna uträtta i programmet. Då alla tillfrågade hade sina vardagliga arbetsuppgifter att utföra tog det ganska lång tid innan vi fått svar av alla.

I början kom det också in relativt mycket feedback som fokuserade på problem i det nuvarande programmet. Till viss del kunde de förslagen också ändras till krav.

Under hela den kommande processen höll vår grupp regelbundet möten, oftast en gång i veckan, där teamet under de första veckorna planerade hur projektet skulle gå vidare. Då alla projektmedlemmar hade använt det aktuella programmet var det viktigt att komma fram till de nyckelpunkter var programmet hade sina största brister i nuläget.



Figur 10. Skärmdump från det gamla programmet

Då detta också var den första riktiga kravspecifikation som de involverade skulle skriva var det viktigt att göra allting noggrant och kontrollera att ingenting blir för mycket influerad av det nuvarande programmet.

I ett tidigt skede föreslog också programmeraren att vi skulle fundera lite på datamodellen för att åtminstone teamet skulle få en annan syn på programmet. Den nuvarande datamodellen gicks igenom och det diskuterades om var de största bristerna hade visat sig vara i den nuvarande lösningen. Dessa möten hjälpte teamet att tänka mera ”out-of-the-box” då det gällde programmet och insåg att det var viktigt att man också gör det då man skriver kravspecifikationer. Annars finns det risk att den dokumentation man gör upp skulle endast kunna leda till en viss typ av slutresultat. Som följande planerades några olika datamodeller och det diskuterades vilka problem de skulle lösa alternativt skapa.

Vid mötena diskuterades allt möjligt som hade med programmet att göra. Ofta ledde diskussionerna till ämnen som inte direkt hörde till vad som skulle krävas av programmet så efter varje möte fick anteckningarna kontrolleras vad som kunde användas som krav. Men det var samtidigt viktigt att ämnena diskuterades då det ibland framkom saker som kunde påverka olika behov för programmet.

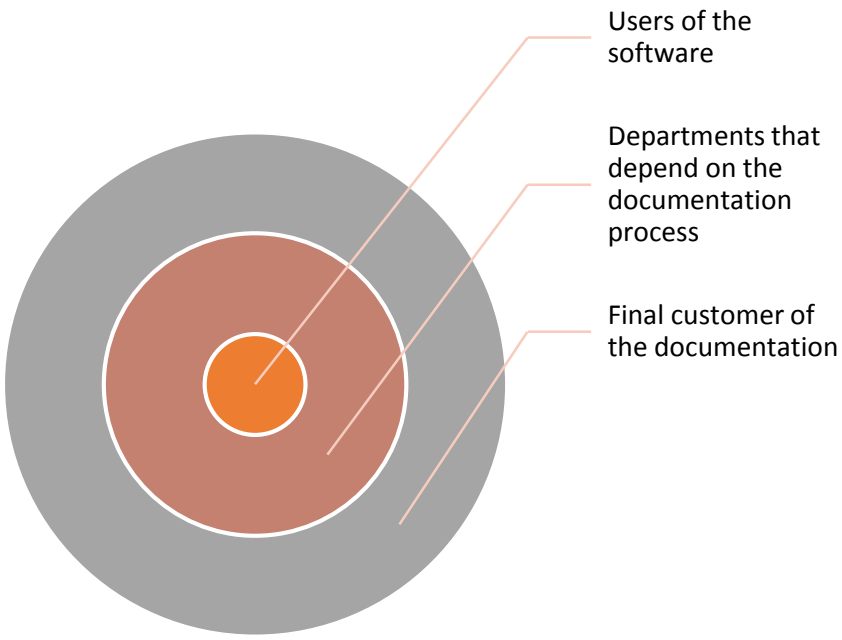
Under mötena var det också ofta man fick fråga ”Varför?” av de som berättade vad de behöver göra. Det var nämligen ofta som många olika förklaringar till vad de behöver göra och hur som man kunde slutleda till ett och samma krav efter man frågat varför de behöver göra vissa saker. Detta var mycket effektivt och fick på detta sätt många redundanta krav bortfiltrerade redan under mötenas gång.

Då programmet skulle komma att användas också av andra avdelningar än den som projektet utfördes åt planerades möten in med nyckelpersoner från andra avdelningar. På dessa möten fick man diskuterat bättre än då man endast skickade e-post med varandra. Men då programmet även skulle användas av avdelningar i Norge samt Italien så gjordes det via e-post. Vid behov fanns möjligheten till virtuella möten via internet men med hjälp av välformulerade e-post emottogs fullgoda svar.

Under mötena med andra avdelningar kom också kravförslag som efter närmare diskussion och överläggning beslöts att de kravförslagen inte hörde till det här programmets syfte. Detta var ibland ett utmanande arbete att förklara åt de personer man hade möte med att just detta krav inte hörde till detta program och inte skulle

tillgodoses. Till saken hör att de arbetsuppgifter som dessa krav berörde kunde de utföra i andra system men för att förenkla sina arbetsuppgifter önskade de så mycket funktionalitet som möjligt till samma program. Detta var ju förstås inte möjligt då projektet skulle växa ur sitt syfte och bli ohanterligt att hantera under den vidare planeringen.

Alla de involverade avdelningarna kan man kalla intressenter och för att visa detta på ett simpelt sätt gjordes en grafisk återgivning på de olika grupper intressenter av programmet som finns. Kärngruppen är de avdelningar som direkt kommer att använda programmet. Utanför kärngruppen finns de avdelningar som jobbar med de avdelningar som hör till kärngruppen. Den yttersta cirkeln består av de intressenter som på ett eller annat sätt endast är intresserade av slutresultatet som de två inre grupperna utför. Då programmet är ett orderhanteringssystem för dokumentation hör till exempel slutkunderna som beställt dokumentationen till den yttersta intressentgruppen.



Figur 11. Simplifierad grafisk representation av intressenterna

Användarna av programmet tillfrågades även vilka 5 saker de ansåg skulle vara viktigast att förbättra i programmet. På denna fråga emottogs åtminstone ett gemensamt svar från i princip alla tillfrågade. Denna fråga gjordes via e-post och det

kom bra svar då det var relativt enkelt och snabbt för personerna att svara på denna fråga.

Säkerheten för systemet diskuterades också men då programmet är internt och skyddas av de yttre brandväggarna behövs ingen separat säkerhet på samma sätt som om programmet skulle vara för externt bruk.

Under projektets gång blev det diskussioner om olika delar av systemet och hur det skulle komma att fungera i praktiken och vad viss data riktigt betydde och varifrån den kom till systemet. För att fullt förstå vissa funktioner och viss data behövdes det ta reda på ganska mycket information om saker som inte direkt hör till den avdelning projektet utfördes åt men det var sådan information som utan mera kunskap såg ut att komma från tomma intet. Dessa utredningar blev ibland komplicerade och tog mycket tid men resultatet av dem var ändå värdefulla då bättre förståelse uppstod hur informationen som behövs i programmet kommer till. Denna kunskap gjorde också att viss data inte skulle gå att förlita sig på för att bygga funktioner kring, denna information var viktig så att kraven kunde anpassas till detta. Vissa problem kunde tyckas vara trivialt att lösa så att informationen som skulle komma till systemet skulle gå att lita på. Men sådant som hör till andra arbetsprocesser kunde inte ändras enbart för att anpassa dem till vårt program. Man insåg relativt snabbt hur mycket tid det går åt att göra program där många informationskällor och arbetssätt redan finns som data som kommer från då dessa dessutom står utom programutvecklingsteamets kontroll. Detta låser ju naturligtvis också vissa typer av lösningar till problem.

Under processen tillfrågades även mellanchefer av olika grad om de hade krav på programmet även fastän de aldrig själva använde sig av det. Detta gjordes för att de var intressenter på ett eller annat sätt i programmets funktionalitet och att de avdelningar som använder sig av programmet utför sina arbetsuppgifter korrekt.

Processen med att klargöra och dokumentera de icke-funktionella kraven gick relativt smärtfritt då teamet i det skedet hade de funktionella kraven i stort sätt färdigt insamlade och en bra kunskap om vad programmet skulle komma att utföra och hur många användare det skulle röra sig om.

RNF-04 | Time zone	
Purpose	The system shall use the same time worldwide
Description	The internal time for the system shall be UTC +2.
Exceptions	
Comments	E.g. deadlines are compared against this time.

Figur 12. Exempel på icke-funktionellt krav

Under hela processen fanns ett utkast av kravspecifikationen där jag både uppdaterade kraven samt formuleringen av dessa. Detta var användbart både då man behövde visa åt någon hur det gick framåt samt att man hade den slutgiltiga kravspecifikationen hela tiden i tankarna. Formuleringen av kraven ändrades till det bättre relativt ofta under processen vilket ledde till flera omskrivningar av hela utkastsdokumentet före det slutgiltiga formuleringssättet var fastställt.

När kravspecifikationsprocessen började närma sig sitt mål hade projektteamet ett gemensamt möte med mellanchefer fyra nivåer uppåt. På det mötet gicks igenom vad som gjorts och vad som skulle göras efter denna process var slutförd.

Efter detta möte var i princip kravspecifikationssteget klart och endast finslipning samt mindre förändringar av kravspecifikationen gjordes efter det.

4 Resultat och diskussion

I det här kapitlet beskrivs resultatet samt diskussion om diverse problem och vidareutveckling.

4.1 Resultat

Slutprodukten av detta slutarbete var ett kravspecifikationsdokument (Software Requirements Specification) som skulle användas för den vidare utvecklingen av programmet. Kravspecifikationen lästes igenom av både projektmedlemmarna samt några andra intressenter inklusive min chef.

4.2 Problem

Under processen uppkom flertalet problem som löstes vartefter. I början av projektet var det vissa svårigheter med att börja insamlingen av kraven då många antingen var på julledigt eller just kommit tillbaka från och då hade mycket vardagligt arbete att göra.

Ett annat problem var att få de tillfrågade att svara på frågor på det sätt som jag behövde för att kunna formulera krav. Detta gick dock ofta att diskutera fram men var allmänt tidskrävande både vid möten samt e-post kommunikation.

Tidvis var det också problem med vissa krav som visade sig vara utanför projektets syfte och egentligen hörde till något annat system. Då måste det kravet filtreras bort och diskutera vad detta betydde med den som uppgett det kravet. Detta var nödvändigt att göra då programmet annars skulle växt sig ohanterligt stort i de följande stegen av utvecklingen och därmed göra hela projektet lidande.

Formuleringen av kraven var även en utmaning då jag i ett skede märkte att jag skrivit en del krav för detaljerade så att de skulle ha styrt utvecklingen av programmet i en viss riktning som inte nödvändigtvis skulle vara den bästa. Efter några omskrivningar blev kraven bra formulerade.

4.3 Vidareutveckling

Som nästa steg efter detta skulle projektet gå vidare med designsteget var programmets grafiska utseende samt olika funktioner skall få sin form. Vid

projektets början var det ännu oklart om jag skulle medverka i de andra delarna av projektet efter att kravspecifikationen blev klar. I ett senare skede blev det dock klart att jag skulle medverka under hela projektets gång men även i fortsättningen skulle utvecklingen ske på sidan om det vardagliga arbetet så tidsplanen beror på den vanliga arbetsbelastningen. Varför utvecklingen inte har en strikt tidsplan är att avdelningen klarar sig med det nuvarande programmet även om det har sina brister som det nya programmet kommer att åtgärda.

4.4 Diskussion

Projektet var hela tiden intressant och projektteamet bestod av personer som det gick smidigt att diskutera och arbeta med. Slutresultatet blev en kravspecifikation på 43 sidor och alla inblandade var nöjda med resultatet. Dokumentet arkiverades också i Wärtsiläs dokumentarkiv så att det vid framtida programuppgraderingar skulle finnas tillgängligt att utgå från. Detta var en ny möjlighet för framtida projekt då ett dokument av denna typ inte gjorts vid de tidigare programversionerna.

Jag lärde mig mycket under hela processen både om hela kravspecifikationsprocessen samt mycket om hur Wärtsiläs interna processer för olika funktioner och avdelningar fungerar.

5 Källförteckning

[1] “Wärtsilä” [Online].
http://www.wartsila.com [hämtat 30.08.2015].

[2] ”Wärtsilä” [Online].
https://fi.wikipedia.org/wiki/W%C3%A4rtsil%C3%A4 [hämtat 30.08.2015].

[3] Görling, Stefan. (2009). *Att arbeta med IT-projekt.* Studentlitteratur AB, Lund.

[4] ”RUP – En introduktion” [Online].
http://www.pellesoft.se/article/566/rup---en-introduktion [hämtat 15.01.2015].

[5] ”Using Both Incremental and Iterative Development” [Online].
http://static1.1.sqspcdn.com/static/f/702523/9242211/1288741989673/200805-Cockburn.pdf?token=iiTP%2B4Hzq6DRMGeoIULln4dv3CM%3D [hämtat 15.01.2015].

[6] Eriksson, Ulf. (2007). *Kravhantering för IT-system.* Ulf Eriksson och Studentlitteratur.

[7] Wiegers, Karl E., (2003). *Software Requirements 2nd edition.* Microsoft Press.

[8] Orozco, Judi. (2003). *The New York State Project Management Guidebook, Release 2.* New York State Office for Technology.

[9] IEEE-SA Standards Board. (1998). *IEEE Recommended Practice for Software Requirements Specifications.* New York, The Institute of Electrical and Electronics Engineers, Inc..

[10] “Key words for use in RFCs to Indicate Requirement Levels” [Online].
http://www.ietf.org/rfc/rfc2119.txt [hämtat 12.02.2015].

[11] ”Writing Good Requirements” [Online].
http://homepages.laas.fr/kader/Hooks.pdf [hämtat 12.02.2015].

[12] Robertson, James & Robertson, Suzanne. (2015). *Volere Requirements Specification Template, Edition 17 – 2015.* Atlantic Systems Guild Limited.

[13] “Non-Functional Requirements” [Online].
https://www.utdallas.edu/~chung/SYSM6309/NFR-18-4-on-1.pdf [hämtat 31.03.2015].

[14] “A Requirements Elicitation Approach Based in Templates and Patterns” [Online].
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.6559&rep=rep1&type=ps [hämtat 25.03.2015].