HÖGSKOLAN
I GÄVLE

AKADEMIN FÖR TEKNIK OCH MILJÖ
Avdelningen för datavetenskap och samhällsbyggnad

Agil Kravprioritering

En kvalitativ studie om
prioriteringsprocesser inom agil mjukvaruutveckling
hos Monitor ERP System AB

Anouschka Aalbers
Linn Öberg

2021

Examensarbete 15hp,
Datavetenskap
för högskoleingenjörsexamen på Dataingenjörsprogrammet

Examinator: Carina Pettersson
Handledare: Ann-Sofie Östberg

Förord

Stort tack till vår handledare Niklas Oldéen och hela Team Stock på Monitor ERP, som varit så välkomnande och hjälpsamma. Tack Ann-Sofie Östberg som visat stort förtroende för oss under sin handledning från Högskolan i Gävle.

Tack Andreas för stöd, pepp och inblick i branschen.

Slutligen vill vi tacka varandra för ett fint samarbete och aldrig sinande stöd, samt för bra diskussioner och tillåtande atmosfär. Två starka viljor men aldrig starkare än att det bästa argumentet får företräde varje gång.

Detta arbete skrevs med hjälp av kaffe, Discord och kontinuerliga pauser.

Sammanfattning

Kravprioritering är ett av de viktigaste och mest inflytelserika stegen vid tillverkning av en mjukvaruprodukt. Processen är iterativ; den sker under hela produktens agila mjukvaruutvecklingsprocess. Genom kravprioritering beslutas det om vilka krav som ska utvecklas, i vilken ordning och varför.

Målet med denna studie är att undersöka hur mjukvaruutvecklande företag gör för att kravprioritera, samt identifiera vilka prioriteringsmetoder de eventuellt använder sig av. Studiens syfte är att få en förståelse för varför en väl avvägd prioritering är viktig, vilka särskilda prioriteringsfaktorer som ger värde till en produkt och att se hur dessa faktorer är relaterade till resultatet. Syftet är även att undersöka vilka svårigheter som finns i en prioriteringsprocess, samt att skapa en översikt över några av de mest vedertagna prioriteringsmetoderna inom agil mjukvaruutveckling.

Studien utförs i samarbete med mjukvaruföretaget Monitor ERP för att analysera företagets prioriteringsprocesser som används för att utveckla deras affärssystem Monitor. Metoden som används är en kvalitativ undersökning som består av observationer av möten kring prioriteringsarbete och semi-strukturerade intervjuer. Bearbetning av insamlat material skedde genom att organisera, analysera och sammanställa resultat enligt begrepp och kategorier som framkom utifrån litteraturstudien. Resultatet redovisar arbetsprocesser, gemensamma mål, prioriteringsaspekter och utmaningar i prioriteringsarbetet hos Monitor ERP.

En väl avvägd prioritering visade sig vara viktigt för att kunna leverera rätt funktionalitet i tid, för att kunna ge trovärdiga estimeringar om utvecklingen och det i sin tur leder till att kunder får förtroende för både produkten och företaget. En rad olika prioriteringsfaktorer som ger värde till programvaran Monitor identifierades, varav många bidrar till att öka kundnöjdheten och kvaliteten på produkten. Monitor ERP använder inte några särskilda prioriteringsmetoder, utan utvecklingsfilosofin *Minimum Viable Product* används som grund till deras prioriteringsval. Under prioriteringsarbetet upplevdes utmaningar såsom begränsade resurser, oförutsägbara uppgifter, svårigheter med tidsestimering och en utmaning i balansen mellan kundnytta och kundfokus.

Nyckelord: Kravprioritering, kravhantering, prioriteringsfaktorer, prioriteringsmetoder, agil mjukvaruutveckling

Abstract

Prioritizing requirements is one of the most important and influential steps in the creation of a software product. The process is iterative; it takes place during the entire agile software development. Through prioritizing requirements, it is decided which requirements are to be developed, in which order, and why.

The aim of this study is to investigate how companies that design software prioritize requirements and to identify which prioritization methods they might use during this process. The purpose of this study is to gain an understanding for why a well-balanced prioritization is important, which specific prioritization factors give value to a product, as well as identifying how these factors are related to the result. The purpose is also to investigate the difficulties that exist in a prioritization process, and to create an overview of some of the most used prioritization methods in agile software development.

This study is conducted in collaboration with the software company Monitor ERP in order to analyze the company's prioritization processes used to develop their business management system Monitor. The method used is a qualitative study that consists of observations of meetings about prioritization processes, and semi-structured interviews. Processing of collected material was done by organizing, analyzing, and compiling results according to concepts and categories that emerged from the literature study. The results documents work processes, common goals, prioritization aspects and challenges in the requirements prioritization at Monitor ERP.

A well-balanced prioritization proved to be important to be able to deliver the right functionality on time and to be able to provide dependable estimates of development, which in turn leads to customers gaining confidence in both the product and the company. A number of prioritization factors that give value to the Monitor software were identified, many of which contribute to increasing customer satisfaction and product quality. Monitor ERP does not use any specific prioritization methods, but the development philosophy Minimum Viable Product is used as a basis for their prioritization choices. During the prioritization process, challenges such as limited resources, unpredictable tasks, difficulties with time estimation, and a challenge in balancing customer value and customer focus were experienced.

Keywords: Requirement prioritization, requirements engineering, prioritization factors, prioritization methods, agile software development

Innehållsförteckning

1 Introduktion

Idag arbetar många mjukvaruutvecklande företag utifrån agila principer, med syftet att hålla nere kostnader och öka chanserna att skapa en bra produkt med många nöjda användare. För att uppnå detta används iterativa och löpande processer, där en kontinuerlig dialog förs mellan utvecklare och intressenter [1], [2].

I den inledande fasen av ett mjukvaruutvecklingsprojekt behöver utvecklingsteamet få en förståelse för vilken produkt som efterfrågas, vilka behov intressenterna har och hur uppgifterna rent tekniskt kan lösas. Detta förberedande skede kallas kravhantering (på engelska *Requirements engineering*) och är en oerhört viktig del i utvecklingsprocessen. Det skapar en stark grund för produktens fortsatta design och konstruktion.

Kravhantering består av flera steg som i viss mån går in i varandra och behandlas iterativt. Dessa steg omfattas av en inledande fas följt av insamlande, bearbetning, förhandling, specificering, validering samt underhåll av krav. Förhandlingssteget innehåller bland annat prioritering av krav, där det beslutas om vilka krav som ska utvecklas, i vilken ordning och varför. Kravprioriteringen är ett av de viktigaste och mest inflytelserika stegen vid tillverkning av en mjukvaruprodukt [3].

1.1 Syfte, mål, frågeställningar

Syftet med undersökningen är att få en förståelse för varför en väl avvägd prioritering är viktig, vilka särskilda prioriteringsfaktorer som ger värde till en produkt samt att se hur dessa faktorer är relaterade till resultatet. Syftet är även att undersöka vilka svårigheter som finns i en prioriteringsprocess, samt att skapa en översikt över några av de mest vedertagna prioriteringsmetoderna inom agil mjukvaruutveckling.

Målet är att undersöka hur företag som utvecklar mjukvara gör för att kravprioritera och identifiera vilka prioriteringsmetoder de eventuellt använder sig av.

Utifrån undersökningens mål och syfte definieras följande frågeställningar:

- Varför är prioritering inom agil mjukvaruutveckling viktigt?
- Vilka faktorer är värdefulla för prioriteringsprocessen?
- Vilka vedertagna prioriteringsmetoder används av mjukvaruutvecklande företag idag?
- Vilka utmaningar följer med kravprioritering?

Att prioritera är ett så pass omfattande och djupgående ämne och mjukvaruutvecklingsprocesser så föränderliga, att forskning och undersökning har stora hål där kunskap kan upptäckas. Vår vetenskapliga undersökning bidrar med kunskap inom kravprioritering och kartläggning av information om prioriteringsprocesser under nutidens mjukvaruutveckling inom ett svenskt företag.

1.2 Monitor ERP System AB

För att undersöka hur kravhantering samt prioritering går till hos mjukvaruutvecklande företag görs en kvalitativ studie på företaget Monitor ERP System AB. Deras produkt, Monitor, är ett affärssystem som stödjer tillverkande företag i planerings- och arbetsprocesser genom att ge företagen möjlighet att ha uppsikt över arbetsflödet. Den allra första generationen av Monitor skapades 1982 av Åke Persson och var ett av de första digitala affärssystemen. Idag har företaget totalt cirka 300 anställda över hela världen, varav ungefär 200 jobbar på huvudkontoret i Hudiksvall, där även majoriteten av utvecklingsavdelningen finns.

Monitor ERPs utvecklande avdelning består av sju team, som alla har olika ansvarsområden inom mjukvaran. Företagets struktur och arbetsprocesser redovisas under Resultat.

1.3 Prioriteringsprocesser under en pandemi

Under 2020 utbröt Corona-pandemin vilket förde med sig en mängd konsekvenser världen över. Många arbetsprocesser anpassades för att fungera digitalt. Monitor ERP är inget undantag och har under denna period anpassat sina arbetssätt för distansbaserad kommunikation och utveckling. Det arbete som vanligtvis brukar ske på kontoret sker under denna period till stor del via röst- och videokanalen Teams.

2 Teori

Under Teori presenteras forskning och litteratur om prioriteringsprocesser. Dessa omfattar en mängd olika element inom kravprioritering. Kapitlet behandlar syftet med kravprioritering, två olika prioriteringsperspektiv, en mängd prioriteringsfaktorer, områdesspecifika prioriteringar, utmaningar med kravprioritering samt avslutningsvis en överblick över vedertagna prioriteringsmetoder inom agil mjukvaruutveckling.

2.1 Varför prioritera?

Enligt en uppskattning av The Standish Group, en forskningsorganisation som fokuserar på att förbättra effektiviteten i mjukvaruutveckling, används 50% av de krav och funktioner som prioriteras nästan aldrig, 30% används sällan och endast 20% används ofta. De flesta funktioner som skapas bidrar enligt uppskattningen inte ens med ett värde [4]. De beslut som tas i en prioriteringsprocess kan med andra ord påverka det slutliga resultatet avsevärt och effekterna av kravprioriteringen är därför stora.

Att organisera krav i en viss prioritet hjälper utvecklingsteamet att förstå vilka krav som är de viktigaste och mest brådskande. På grund av begränsat tidsschema och en viss resursram behöver det beslutas om vilka krav och funktioner som ska prioriteras först. Ofta behöver vissa krav väljas bort eller senareläggas, för att kunna leverera i tid och till tänkt kostnad [5].

En väl avvägd prioritering synliggör vilka krav som faktiskt är viktiga och jämför hur stort behovet av ett specifikt krav är i förhållande till övriga krav. Särskilt i iterativa planeringar, såsom i agil utveckling, är målet med planering och prioritering att hitta värdet i produkten. Det gör det möjligt att motivera de val som görs i utvecklingen och att därav ta mer realistiska beslut.

För att kunna kravproritera realistiskt behövs således information om storleken på projektet, projektets deadlines samt vad som är ett rimligt resultat i slutet av varje deadline. Storlek och resultat kan inte besvaras konkret i agila projekt utan ska bestämmas via estimeringar. Estimeringen ska ske kontinuerligt under projektets gång och kan inte göras utan att alla involverade i utvecklingen vet vad som är viktigt och betydelsefullt i projektet [6].

Eftersom det under iterativ planering kontinuerligt inhämtas ny information om kraven och funktionaliteterna minskar osäkerheter runt utvecklingen under projektets gång. Om planeringen och prioriteringen justeras på grund av ny kunskap inom projektet kan alla inblandade i prioriteringsprocessen bättre estimera kostnad, tid och insats.

Prioriteringsarbetet kan minska de risker som finns inom mjukvaruutvecklingen. Det finns främst tre sorters risker: schematisk risk, kostnadsrisk samt funktionalitetsrisk. Schematisk risk innebär risken att utvecklarteamet inte kommer kunna hålla sig till satt deadline. Kostnadsrisken betecknas av risken att utvecklingen av ett projekt kommer kosta för mycket. Till sist bär utvecklingen alltid med sig en risk att en viss funktionalitet inte kan implementeras ordentligt; kan utvecklingsteamet skapa en fungerande produkt inom tidsramen? Ju större kännedom som finns om domänen, prioriteringen och kraven i ett projekt, desto bättre uppskattning kan företaget göra om vilka risker som finns [6].

Uppskattningar gjorda utifrån ett bra prioriteringsarbete medför att ett företags pålitlighet ökar gentemot sina kunder. Prioriteringen tillåter utvecklare att göra trovärdiga estimeringar om vilka funktionaliteter som ska finnas med i en versions-release [6]. Detta ger kunderna förtroende för företaget och höjer mest sannolikt även kundnöjdheten eftersom utvecklare kan lansera programuppdateringar i en stadig takt. Av den anledningen ökar sannolikheten för att de krav som kunderna värderar högst implementeras. Det minskar i sin tur sannolikheten att produkten kommer att ratas när den väl lanseras [5] .

2.2 Prioriteringsperspektiv

För att kunna prioritera effektivt behövs ett helhetsperspektiv på prioriteringen. Detta perspektiv bestämmer vilket resultat som utvecklingsprocessen förväntas mynna ut i. Resultatet i denna kontext handlar om vad som ger en produkt ett affärsvärde, på engelska *business value* [7]. Exempel på affärsvärde är företagets utvecklings- och omsättningsmöjligheter [8].

Mycket av prioriteringen inom mjukvaruutveckling bedöms utifrån ett perspektiv där projektets alla beståndsdelar anses vara likvärdiga. Detta innebär att varje beståndsdel i prioriteringen, såsom krav, användningsfall, programtest och buggar anses vara lika viktiga [8], [9]. När det genomsnittliga utvecklingsprojektet var mindre var detta sätt att prioritera lämpligt; lönsamheten i ett projekt bestämdes främst baserat på projektkostnad och att hålla schema [9]. Med andra ord bestämdes värdet på ett utvecklingsprojekt enbart av hur fort en produkt kunde publiceras på marknaden för att dra in vinst. Möjliga vinster som kunde genereras ur projektet beräknades genom att hålla utvecklingskostnader nere och leverera nya funktionaliteter kontinuerligt. Det var möjligt då utvecklingstiden var betydligt kortare än idag. Nu anses det vara ett kortsiktigt sätt att prioritera därför att det inte tar hänsyn till framtida utveckling av produkten [8].

I takt med att utvecklingssystem har blivit större och mer komplexa projekt behövs nya perspektiv på kravprioritering [7], [9]. Det finns mer konkurrens vilket som följd kräver högre kvalitet på utvecklingens slutresultat. Att ta fram värde innebär att värdeanalysen bestäms utifrån perspektiven av fler involverade parter:

"*When thinking in terms of value, the spectrum and variety of stakeholders involved in the decision making is wider than when thinking in terms of costs, and a broader range of factors is also considered (e.g. business, marketing, technical, etc.)*" [8].

Det innebär att kravprioritering borde ta ett större, mer långsiktigt perspektiv än enbart kostnad-vinst-perspektivet.

Ett förslag på hur ett företag kan ta ett långsiktigt prioriteringsperspektiv är värdebaserad mjukvaruutveckling, på engelska *Value Based Software Engineering* (VBSE) [8], [9]. Utvecklingsperspektivet analyserar vilken sorts produkt som ska utvecklas och vilka prioriteringar som stödjer projektets mål. Här tas det i stället hänsyn till vad som gör en produkt värdefull. Det är möjligt att ta hänsyn till fler perspektiv än enbart företagets förtjänst; bland annat kundens behov, programvarans kvalitet, företagets lönsamhet och många fler [7], [8].

Synsättet VBSE började träda fram i början av 2000-talet och har gett upphov till kopplingen mellan produktens värde och de prioriteringar som leder till det [9]. Utvecklingens fokus ändras härmed från affärsvärde till produktvärde och tack vare detta mer långsiktiga prioriteringsperspektiv är det bland annat möjligt att förbättra kvaliteten på produkten. En upptäckt under de senaste decennierna är att det finns en tydlig koppling mellan robusthet av mjukvarans arkitektur och det fortsatta utvecklingsarbetet. Robusthet av mjukvarans arkitektur påverkar i stor mån utvecklarnas insats för systemets utveckling samt underhållet av befintliga system [7].

2.3 Prioriteringsfaktorer

Inom mjukvaruutveckling finns det många krav som måste upprätthållas och för att kunna prioritera i utvecklingsarbetet finns det en del faktorer som ingår. Dessa faktorer har dock inte officiellt standardiserats, utan varje företag och projekt benämner samt prioriterar faktorerna olika. Trots dessa skillnader går det att identifiera några gemensamma nämnare.

Alahyari et al. [10] identifierar totalt 16 olika faktorer att ta hänsyn till under kravhanteringsfasen. Dessa är:

- Leveransprocess med avseende på tid
- Upplevd kvalitet
- Kostnad
- Faktisk kvalitet vad gäller produkt, kod, arkitektur eller robusthet
- Processer och arbetssätt
- Prestanda och användbarhet för slutanvändaren
- Innovation och kunskap om organisation
- Kundrelation
- Kunskap om funktionalitet för kund eller produktanvändning
- Intäkter och affärsvärde
- Funktionalitet
- Icke-funktionella krav och hedoniskt värde
- Konkurrenskraft
- Professionalitet och arbetsmoral
- Hållbarhet
- Pålitlighet

Alahyari et al. [10] beskriver inte alla ovannämnda faktorer djupgående men det finns förtydliganden omkring särskilda begrepp. *Leveransprocess med avseende på tid* innebär att utveckling sker inom den bestämda utvecklingstiden. En av det Agila Manifestets grundtankar är "*deliver often and in time*" och därför anses faktorn viktig. I samband med *upplevd kvalitet* beskrevs det att företaget vill vara den bästa inom utvecklingsområdet, men om det enbart handlar om företagets interna syn eller om det gäller kundernas åsikt framgick inte. *Kostnad* är en faktor som beskrivs bestå av 3 aspekter: kostnad vad gäller arbetskraft och tillgängliga resurser för mjukvaruutveckling, det upplevda värdet som kunden ser i produkten och vinsten företaget kan driva med slutprodukten. *Kostnad* tar hänsyn till flera perspektiv: Kan valda funktioner resursmässigt utvecklas, får kunden valuta för investeringen, och kommer företagets vinst gynnas av utvecklingen?

Processer och arbetssätt innebar bland annat optimering av agila arbetsmetoder. Detta beskrevs som särskilt betydelsefullt för större företag, eftersom deras arbetsprocesser är omfattande och måste simplifieras för att vara effektiva. Kvaliteten av mjukvaruprodukten anses vara relaterad till arbetsmetoder. *Prestanda och användbarhet för slutanvändaren* är ett bredare begrepp som handlar om hur användbar en funktion är för användaren. Det påpekas att den *faktiska kvaliteten* av hela produkten lider om det inte tas hänsyn till *Prestanda och användbarhet*. Som konsekvens kan det slutliga värdet i produkten minska.

Innovation och kunskap om organisation är i synnerhet två olika begrepp som kategoriserades tillsammans av Alahyari et al. [10]. *Innovation* gäller utveckling av produkter som inte funnits på marknaden tidigare. Det beskrivs att mycket av mjukvaruutveckling inte är proaktiv utan reaktiv och att utveckling ofta styrs av marknadsbehov. *Kunskap om organisation* kopplas till denna aspekt därför att organisationen och dess anställda måste veta vad som är viktig för företaget för att kunna lyssna till marknadens behov. *Konkurrenskraft* är således även relaterad till dessa aspekter.

Icke-funktionella krav och hedoniskt värde omfattar utveckling som bidrar i kvaliteten av en produkt. *Hedoniskt värde* refererar till den positiva upplevelsen en kund får av en produkt och det kan bland annat komma ifrån icke-funktionella krav såsom förbättring av prestanda och säkerhetsoptimering.

Torrecilla-Salinas [11] tar upp några kompletterande faktorer:

- Uppskattad ansträngning
- Minimi-kraven till leverans av första modell
- Första release-datumet för en användbar produkt
- Första release-datumet som kan skapa vinst
- Fördröjningsrisker under utvecklingen

Här inkluderar *uppskattad ansträngning* beroenden på externa faktorer såsom kontakt med tredje parter och andra avdelningar. *Minimi-kraven till leverans av första modell* tas fram i Torrecilla-Salinas undersökning med hjälp av prioriteringsmetoden ”Walking Skeleton” (mer om denna metod i 2.6.6). *Första release-datumet för en användbar produkt* samt *första release-datumet som kan skapa vinst* är nära besläktade, med skillnaden att den förstnämnda inte behöver generera vinst. Till sist tas även möjliga hinder upp i prioriteringsprocessen som kan fördröja arbetet på mjukvaruutvecklingen [11].

I Torrecilla-Salinas undersökning använde företagen som deltog i fallstudien flera olika metoder för att ta fram prioriteringarna. Däremot påpekas att många prioriteringsverktyg som används i agila arbetssätt, såsom Planning Game och Planning Blitz (en version av Planning Poker), inte tar hänsyn till hur man kan komma fram till prioriteringsfaktorerna. Planning Poker tar därefter också bara hänsyn till en faktor, nämligen hur mycket ansats ett delmål eller en funktion kräver för att utvecklas, utan hänsyn till värdet den ger till produkten eller andra faktorer som nämnts ovan [11].

2.4 Områdesspecifika prioriteringar

Vad som prioriteras beror mycket på utvecklingsområdet. Utvecklingsområdet är en annan beteckning på domän, vilket innebär vilken typ av produkt som utvecklas [10]. Fyra domäner undersöktes av Alahyari et al. [10]; Automation, Försvar, Telecom samt Konsultation. Inom automation kan detta innebära att skapa mjukvara som styr konstruktion av maskiner och bilar, medan inom försvardomänen kan det exempelvis handla om navigationssystem. Olika företagsdomän har därmed olika krav på sig.

I alla utvecklingsområden som undersöktes var det viktigt att hålla de deadlines som satts av projektledarna. Anledningarna till detta var bland annat att flera avdelningar var beroende av varandras framsteg för att kunna fortsätta utveckla. Den faktor som framställdes frekvent som huvudfaktorn i prioriteringen för alla de undersökta utvecklingsområdena var *leveransprocess med avseende på tid,* vilket innebär hur fort det går att leverera produkten eller funktionen [10][12].

Det påpekades att projekt inom försvar har detta krav särskilt högt upp på grund av att försvaret utvecklar stora system som består av många nästlade subsystem som alla är beroende av varandra. För utveckling inom försvardomänen var även två andra prioritetsfaktorer särskilt viktiga; *prestanda* och *kodkvalitet*. Detta för att produkterna inom detta område måste uppnå krav och hålla sig till specifikationer som bestäms inom utvecklingsområdet [10].

2.5 Utmaningar inom Prioriteringsprocessen

Det finns många svårigheter och utmaningar med att avgöra vad som ska prioriteras inom mjukvaruutveckling. En anledningen till detta kan vara komplexiteten av program som är uppbyggda av stora system som i sin tur består av mindre delsystem [13].

Att evaluera vilka prioriteringsfaktorer som är viktiga och således ger produktvärde, anses vara en stor utmaning. Många faktorer påverkar vikten av varje krav. Hur viktigt ett krav anses vara kan i sin tur skilja sig mellan olika intresseområden. Utifrån en ekonomisk aspekt bör de minst kostsamma funktionaliteterna, som samtidigt drar in störst vinst, utvecklas. Ur ett annat perspektiv kan det vara andra krav som premieras. Att sammanställa dessa intressen och leverera funktioner som både uppnår kundnöjdhet och passar in i projektets begränsningar, är av utmanande karaktär [14].

För att kunna sammanställa vilka prioriteringsfaktorer som är viktigast i utvecklingsprocessen är ett agilt arbetssätt, med iterationer och möten utifrån en tydlig backlog, av stor vikt. Det finns dock ibland en missuppfattning om att kvaliteten av utvecklingsprocessen utgör den faktiska kvaliteten på produkten. Utvecklingsprocessen påverkar den *uppfattade* kvaliteten av en produkt, men skiljer sig från den *faktiska* kvaliteten på den slutliga mjukvaran. Med missuppfattningen följer även antagandet att *uppfattade* kvaliteten sedan ger en kortare marknadsledstid för produkten, men även här är det i synnerhet den *faktiska* kvaliteten som avgör. Missförstånd av den här typen kan orsaka försenade deadlines och problem i utvecklingsprocessen, vilket i sin tur påverkar prioriteringen av hela projektet [10].

Ytterligare en svårighet omkring prioriteringsarbetet är att det i många fall inte finns en gemensam metod på företagsnivå för att ta fram ett gemensamt mål; det vill säga att de avdelningar som arbetar tillsammans för att framställa en produkt inte använder en och samma metod för att ta fram vilket *värde* utvecklingen har som mål.

Detta försvåras ytterligare av att det finns flera avdelningar som är avgränsade från varandra i större företag. Denna avgränsning sker i två olika dimensioner, nämligen tid och rum. Här innebär tid att olika avdelningar kan ha olika tidsmål och deadlines som sedan påverkar nästa länk i utvecklingskedjan. Rum talar om det geografiska avståndet mellan ett företags olika avdelningar [10], [13].

Det geografiska avståndet som kan finnas där olika företagsavdelningar har olika kontor kan leda till kommunikationssvårigheter, till exempel att kommunikationen mellan olika avdelningar tar längre tid att uppnå vilket ökar den strukturella komplexiteten av utvecklingsprocessen. Fördröjningar kan på så sätt påverka hela processen om det finns problem inom prioriteringen och planeringen [10], [13].

2.6 Prioriteringsmetoder

Det finns en mängd olika metoder för kravprioritering. Vilka metoder företag väljer att använda beror på flera faktorer. Bland annat behöver företagen ta hänsyn till hur mycket tid som finns till förfogande, hur mycket information om kraven som finns tillgänglig, hur mycket information som *behövs* för varje krav, vilken typ av projekt det är och i vilken fas i projektet företaget befinner sig. Vissa utvecklingstekniker innehåller en rekommenderad prioriteringsmetod men den är inte alltid den bäst lämpade metoden för aktuell uppgift [15]. Vanligt förekommande prioriteringsmetoder inom agil mjukvaruutveckling är Analytic Hierarchy Process (AHP), Wieger's method, MoSCoW, Planning Game, Hundred dollar method, Walking Skeleton samt User Stories och Story Points [5], [14]. Dessa beskrivs nedan.

2.6.1 Analytic hierarchy process

Analytic Hierarchy Process (AHP) är den metod som är mest använd inom prioriteringsarbete [5], [14]. Det är en metod som klarar att prioritera utifrån flera kvantitativa och kvalitativa aspekter. Metoden är grundad i linjär algebra och skapar matriser, som innehåller ”vikter” på de olika faktorerna. Ur dessa matriser kommer ett värde fram som bedömer hur högt upp ett visst krav ska vara [16].

Metoden använder sig av följande steg:

1. Nedbrytning av beslutsproblemet för prioriteringen i en hierarkisk struktur bestående av ett mål, alternativ för att nå målet och ett antal kriterier som alternativen utvärderas mot.
2. Prioritetsviktning beräknas för kriterierna och alternativen för att bestämma hur viktiga de är i förhållande till varandra genom parvis jämförelse där en ordinalskala med siffrorna 1–9 används. Detta görs en gång i processen. När två kriterier har jämförts med varandra så jämförs de inte igen.
3. Konsistensindex beräknas för att visa riktigheten på resultatet av den parvisa jämförelsen [15], [17].

2.6.2 Wieger’s method

I Wieger’s method beräknas kravets relativa prioritet genom att dividera summan av kravets värde för användaren och värdet på en ”bot”, på engelska *penalty*, vid avsaknad eller försening av implementation med summan av kostnaden för implementation och risken för misslyckad implementation (se ekvation 1). Skalan 1–9 används för att värdera de ingående faktorerna. Det finns även olika varianter av metoden där faktorerna viktas på olika sätt [5].

$$Prioritet = \frac{\text{Värde} + \text{Vite}}{\text{Kostnad} + \text{Risk}}$$

Ekvation 1. Wieger's method.

2.6.3 MoSCoW

MoSCoW är en akronym där versalerna representerar de fyra hierarkiskt ordnade prioriteringskategorierna: Must have, Should have, Could have och Won’t have.

Must have är krav som är absolut nödvändiga för att ha en fungerande produkt.

Should have är också viktiga krav som borde ingå i produkten om det är möjligt.

Could have är önskvärda om tidsramen tillåter men inte nödvändiga.

Won't have är krav som inte kommer att implementeras inom tidsramen men eventuellt i senare skede [15].

I MoSCoW metoden är det viktigt att alla involverade i prioriteringen är väl insatta i vilka funktionaliteter som är viktiga för produkten samt företagets mål och värdegrund. Metoden hjälper de priorititerande parterna att rangordna baserat på det de anser vara viktigt i utvecklingen genom diskussion mellan alla intressenter. Intressenternas erfarenhet av vad som ger en produkt värde är därför en viktig del av prioriteringsmetoden [18].

2.6.4 Planning Game

Den här metoden skapades inom eXtreme Programming och användes ursprungligen för att utvärdera arbetet i en iteration, men har nu fler användningsområden. I metoden utgår man från att det är användarna som har störst insikt i *vad* som behöver utvecklas och att utvecklarna sitter på kunskapen om *hur* det ska implementeras. Användarna kategoriserar kraven i tre områden utifrån affärsvärde: nödvändigt, eventuellt och valfritt (essential, conditional and optional). Utvecklarna uppskattar den tekniska och ekonomiska kostnaden för kraven. Dessa steg upprepas sedan till det att samtliga krav har uppskattats och organiserats. Under prioriteringsprocessens gång interagerar utvecklarna och användarna med varandra för att reda ut eventuella osäkerheter [5][11].

2.6.5 Hundred dollar method

Här görs en rangordning av kraven utifrån fiktivt monetära värden. Intressenterna i ett projekt ska föreställa sig att de har hundra dollar var, som får spenderas på specifika krav i mjukvaruprojektet. Pengarna ska sedan fördelas mellan de funktioner som finns att välja mellan utifrån hur eftertraktade dessa är. Slutligen räknas de spenderade pengarna ihop och kraven rangordnas utifrån hur mycket intressenterna är villiga att betala för dem [5] [15].

2.6.6 Walking Skeleton

När metoden Walking Skeleton används ligger fokus på att skapa en funktionell prototyp med minsta möjliga funktioner; både vad gäller antal och storlek. Metoden trädde fram i samband med den agila utvecklingstekniken *Minimum Viable Product* (MVP), vilken går ut på att skapa program utifrån ett evolutionärt tänk. Genom MVP skapas ett konceptbevis, på engelska *proof of concept*, som sedan utvecklas vidare i samråd med produktägaren. Det tillåter en kontinuerlig dialog som underlättar justering av vilken slutprodukten kommer bli [19].

Walking Skeleton skapar däremot inte ett utkast av en produkt, utan en fullständig produkt som kan köras, testas och till och med publiceras för kunderna. Det är en prioriteringsmetod som följer MVP-tankesättet och delar upp utvecklingskraven baserat på vilka som är de mest kritiska komponenterna för att skapa ett program. Den tar även hänsyn till utveckling av program som har flera olika funktionaliteter som samarbetar med varandra. Varje del i utvecklingen med Walking Skeleton testas vilket gör att misstag och utvecklingsproblem såsom buggar och fel i mjukvarans arkitektur upptäcks tidigt i processen [20].

2.6.7 User Stories och Story Points

User Stories är en prioriteringsmetod som använder betygsättning för att uppskatta storleken på utvecklingsuppgifter. En User Story är en kort beskrivning på funktionaliteten av en uppgift som ska utvecklas. Det finns inga bestämda regler för hur dessa ska skrivas, men vanligast är en formulering som liknar följande: ”Som en [typ av användare] vill jag kunna göra [händelse] så att jag kan [värde]”. Ett exempel på detta kan vara ”Som *kund* vill jag kunna *skriva in antalet varor jag vill ha* så att jag *snabbt kan ange antalet utan att behöva ladda om webbsidan*”.

Uppbyggnaden av User Stories består övergripande av tre delar; en kort beskrivning som används för planering och utveckling, diskussioner omkring detaljer av en User Story och beskrivning av tester och detaljer som bestämmer när en User Story anses vara färdigutvecklad [21].

User Stories kan betygsättas med hjälp av Story Points. Dessa poäng är en uppskattning av den insats, i form av exempelvis ansträngning, komplexitet och risk, som krävs för att utveckla en viss User Story. Betygen kan vara godtyckliga siffror. Det är inte betydelsefullt att utvecklare vet exakt hur mycket arbete som krävs för en viss funktionalitet, det är viktigare att jämföra storleken på User Stories relativt varandra [6].

Det finns två vanliga sätt att börja betygsättningen. Den första är att ta fram den User Story som är minst och sätta alla andra betyg i relation till den. Den andra metoden är att välja en medelstor story som betygsätts med ett medelstort betyg, följt av att betygsätta de andra utifrån det. Det går att estimera tid med hjälp av Story Points men det är inte främsta målet med denna prioriteringsmetod. Den ska främst ge en estimering av storleken på arbetet. Däremot kan utvecklingsteamets effektivitet beräknas genom att ta reda på avklarade Story Points under en eller flera sprints [6].

3 Metod

Metoden för examensarbetets utförande bestod av flera delar som tillsammans ligger till grund för de slutsatser och diskussioner som framskridit ur undersökningen. Efter en grundlig litteraturstudie undersöktes det aktuella prioriteringsarbetet på Monitor ERP genom observation av möten samt flertalet intervjuer. Samtliga steg beskrivs nedan.

3.1 Litteraturstudie

En kvalitativ litteraturstudie genomfördes för att dyka in i nuvarande kunskapsläge gällande prioritering inom agil mjukvaruutveckling. Det primära målet med denna var att undersöka varför prioritering inom agil mjukvaruutveckling är viktigt, vilka perspektiv och faktorer som ingår i ett prioriteringsarbete samt skapa en överblick över existerande prioriteringsmetoder.

3.1.1 Urval

Litteraturstudien består framför allt av vetenskapliga artiklar men även relevant studielitteratur har använts för att belysa centrala koncept inom områdena agil mjukvaruutveckling och kravprioritering. Särskild vikt har lagts vid att granska både publikationsdatum samt plattform av publikationer för att garantera att den nyaste kunskapen tagits fram. Granskning av plattform gjordes för att kunna ge ett så oberoende perspektiv som möjligt och förhindra särskilt bias i arbetet.

3.1.2 Genomförande

De vetenskapliga källorna har tagits fram genom användning av sökportalen Discovery via bibliotekets webbsida på Högskolan i Gävle. Via Discovery användes flera databaser såsom ResearchGate, IEEE- Xplore samt Elsevier. Vid sidan av Discovery har även Google Scholar använts som sekundär sökmotor. Anledningen till detta är att Google Scholar ibland har bättre resultat med engelska söktermer, vilket är särskilt viktigt när det kommer till datorkunskap som har många facktermer på engelska.

En samling av sökord har använts för att börja sökningsarbetet. Söktekniken inkluderade boolesk logik som inkluderar, utesluter eller specificerar söktermerna. Följande tabell redovisar kortfattat de viktigaste söktermerna.

Huvudterm	AND	OR
Prioritization Prioritisation	Software Engineering SCRUM Techniques Factors Challenges	Software development Agile
Analytic Hierarchy Process Wieger's Method MoSCoW Planning Game Hundred Dollar Method Walking Skeleton User story Story points	Software Engineering	Software Development

Tabell 1 redovisar litteraturstudiens sökord och hur dessa användes i kombination med varandra.

3.1.3 Bearbetning av data

Sökresultatet av litteraturstudien har organiserats och kategoriserats enligt frågeställningarnas upplägg. Det ger en strukturerad sammanställning av den valda litteraturen. Särskild insats har gjorts för att skriva så kortfattade och sakliga sammanfattningar som möjligt av den valda litteraturen, sådan att enbart den informationen som hade relevans till den fortsatta studien togs med i arbetet.

Litteraturstudiens *produkt* har syntetiserats till en sammanhängande teori som ger en relevant bakgrund till övriga delar av undersökningen och redovisar innehållet i förhållande till frågeställningarna [22].

3.2 Observation

För att undersöka det aktuella prioriteringsarbetet hos Monitor ERP har planerings- och uppföljningsmöten som rör utvecklingsarbetet observerats.

3.2.1 Urval

Urvalet av de tillfällen som skulle observeras gjordes utifrån förutsättningen att kravprioritering var ett naturligt inslag. Varken tidpunkt, fas i utvecklingsprocessen eller antalet deltagare var avgörande, så länge det fanns observationer att göra angående prioriteringsval och resonemang omkring prioriteringsprocessen.

Majoriteten av de möten som valdes rörde prioritering på team-nivå. För att kunna delta och få en översikt på hela prioriteringscykeln inom ett team gjordes valet att under observationerna fokusera på ett enskilt team, det som finns beläget i Gävle. Det bestämdes att de tillfällen som var relevanta att delta i skulle täckas av åtminstone en hel sprint, det vill säga en avgränsad fokusperiod av utveckling som är tänkt att resultera i en leverans av givna periodens utvecklingsmål. I fallet av denna studie innebar det en period på fem veckor.

Det skiljer sig mellan teamen hur de olika prioriteringsmötena organiseras och planeras. Utifrån studiens omfång och begränsningar i arbetsperioden har det inte ansetts möjligt att inkludera observationer av flera team.

3.2.2 Genomförande

Observationerna har varit av direkt och icke-deltagande karaktär. Det innebär att information har samlats in genom att aktivt lyssna och ta in det som sagts och gjorts under möten och planeringstillfällen. Detta har skett under så kallade ostrukturerade former, där observatörerna har antecknat det som ansetts intressant och relevant för studien [23].

Observationerna har ägt rum via röst- och videokanalen Teams. Samtliga deltagare har infunnit sig i mötena på det sättet. Ibland har några av deltagarna varit på plats på företaget och då kört gemensamt i ett mötesrum.

Under arbetet erbjöd Monitor ERP möjligheten att ta del av befintlig dokumentation angående deras arbetsprocess. Monitor ERP har skapat en databas med information om standardiserade arbetsflöden, beskrivningar av protokoll och andra stöddokument som företaget kan ha nytta av. Det innehåll i dokumentationen som ansågs vara relevant, har tagits med i resultatet av observationerna.

3.2.3 Bearbetning av data

Mötesanteckningarna analyserades genom att identifiera samt kategorisera viktiga tema, nyckeltermer och begrepp i samband med intervjumaterialet. De termer som eftersöktes i observationsmaterialet var baserade på litteraturstudien och analyserades enligt teman som upptäcktes under studiens gång.

3.3 Intervjuer

För en mer djupgående inblick i Monitor ERPs prioriteringsarbete har flertalet intervjuer utförts med personer på olika poster och inom olika ansvarsområden, inblandade i företagets arbete med prioriteringsfrågor. Genom intervjuerna har de utvalda personernas uppfattningar, erfarenheter och upplevelser angående företagets prioriteringsprocess kunnat beaktas.

3.3.1 Urval

Urvalet till intervjuerna gjordes genom att undersöka vilka roller som finns i Monitor ERPs företagsstruktur, med hjälp av tillgång till företagets intranät och i samråd med handledare på Monitor ERP. Utifrån det togs beslutet att avgränsa intervjuerna till de roller som sitter i främsta ledet vad gäller kravprioritering. Personerna på dessa positioner prioriterar inom följande områden i Monitors utvecklingsprocess: Road Map (planering för de kommande två åren), utvecklingsplanen (närmast prioriterade versionsrelease), sprint-planering (fyra veckors intervall) samt veckovis och daglig prioritering.

De två roller som var självklara för intervju var Business Analysts och Team Managers. Business Analysts prioriterar på sprintnivå utifrån den mer övergripande nivån (Road Map) samt är en viktig källa till information i sprint-prioriteringen för en Team Manager. Team Managers leder sprintplaneringen och ser till att teamet bibehåller prioriteringsfokus under själva sprintperioden. Genom ett nära samarbete mellan dessa två roller går det att kombinera olika perspektiv på Monitors utvecklingskrav.

Business Analysts och Team Managers från samtliga team i utvecklingsavdelningen som ansågs relevanta för undersökningen tillfrågades. Totalt tillfrågades 12 personer att delta i intervju och samtliga tackade ja.

En begränsning har gjorts av rollerna utvecklare och testare, därför att dessa tar en viktig men mer stödjande roll i prioriteringsprocessen. Dessa kommer således inte att behandlas vidare i den här undersökningen.

3.3.2 Genomförande

Intervjuerna har genomförts via röst- och videokanal i programmet Teams. En person åt gången har intervjuats, med ett undantag då två Business Analysts från samma team deltog i ett och samma samtal. Arbetsdelningen har bestått i att en intervjuare har lett samtalet och den andra har fört anteckningar samt agerat som stöd vid behov. Samtliga intervjuer har spelats in som reservmaterial i fall att anteckningarna inte var tillräckliga.

Intervjuerna har varit semistrukturerade, i den mening att de inrymt frågeområden som kan kopplas till undersökningens syfte och frågeställningar [23]. Varje enskild fråga har försökt formuleras så neutralt och öppet som möjligt för att undvika en ifrågasättande eller laddad ton, med utrymme för respondenten att svara utifrån sina egna uppfattningar och inte bli påverkad av ett visst uttryck [24]**.** De frågor som användes för att leda de semistrukturerade intervjuerna finns i Bilaga A.

De taktiker som har använts under intervjuerna är uppmaning, uppföljning och kontroll. Det har avsiktligt avvaktats med att ställa nästa fråga för att vänta in respondentens svar eller för att uppmana denne att säga mer. För att få mer utvecklade och detaljerade svar från respondenten har svaren i vissa fall behövt följas upp med ytterligare frågor. Kontrollfrågor har i vissa sammanhang ställts för en försäkran om att svaret tolkats på rätt sätt [23].

3.3.3 Bearbetning av data

Bearbetningen av intervjuerna var en stor och omfattande del av studien. För att effektivt kunna identifiera upprepande teman och begrepp i det inhämtade datat kodades informationen. Kodning är en teknik som kategoriserar data baserat på tema, begrepp och element. Inom kvalitativa studier är det ofta lämpligt att använda flera olika kodningsmetoder för att analysera datat noggrant [25].

Det har använts en kombination av flera kodningsmetoder. Det finns inga bestämda regler när det gäller kodning och därför har skribenterna tagit vissa friheter med att adaptera kodningsmetoder till eget behov. Som stöd användes färgkodning för att underlätta kodningsarbetet.

Det ingår två faser i kodningsarbete. Dessa faser är inkrementella steg, där den första försöker identifiera övergripande tema och kategorisera utifrån ett bredare perspektiv. Det finns sju subkategorier som kodningsmetoderna kan delas in i: Grammatisk, Elementär, Påverkande, Litterär och Språk, Explorativ, Procedural samt Tematisk [26].

Under denna studie har den första fasen bestått av att koda datat utifrån Grammatiska, Elementära och Explorativa sätt. De kodningsmetoder som användes under kategorin Grammatiska var *Subkodning*, *Strukturell kodning* under Elementära och *Holistisk kodning* som tillhör Explorativa kodningssätt.

Subkodning fokuserar på att samla mindre likartade kategorier som upptäcks under bearbetning under en större gemensam kategori. Här togs det hänsyn till upprepande företeelser och fenomen som passade under liknande kategorier.

Holistisk kodning användes för att gruppera uttalanden, i stället för att transkribera intervjuerna linje för linje. Metoden anses vara särskilt lämplig för nybörjare i kvalitativ undersökning och kan appliceras på många olika typer av data, exempelvis en kombination av intervjuer och observationer. Holistisk kodning användes i syftet att förbereda en mer djupgående analys som gjordes genom Strukturell kodning. Strukturell kodning kategoriserar enligt termer och begrepp som används tidigare under studien. Dessa härstammar ur teorin och undersökningsfrågorna till studien. [26].

De frågor som analyserades djupare med strukturell kodning var "*Vad tar du hänsyn till under prioriteringsarbetet?*" samt "*Vilka utmaningar upplever du under prioriteringsarbetet?"* därför att dessa innehöll mycket begrepp som framkom ur litteraturstudien. Frågorna var unika på grund av att de kunde kvantifieras och frekvensen av begrepp kunde analyseras.

Den andra kodningsfasen är en granskning av det bearbetade materialet. Även här finns det en rad metoder som kan användas, men Fokuserad kodning var mest relevant för studien på grund av den starka kopplingen till teorin. Metoden användes även för att hitta samband mellan till synes orelaterat svarsmaterial. Med hjälp av Fokuserad kodning identifierades anledningar och förklaringar inom det förarbetade datat.

4 Resultat

Det resultat som har framkommit under studien presenteras nedan. Under rubriken 4.1 *Introduktion till resultat* ingår företagsinformation av mer allmän karaktär, såsom upptäckter om företagets struktur, breddad förståelse av rollerna Team Manager och Business Analysts, samt arbetssätt och begrepp inom prioriteringsprocessen. Därefter följer sammanfattningar av observationstillfällen och intervjuer.

4.1 Introduktion till resultat

Monitor ERP består av sju olika avdelningar, varav Development G5 fokuserar specifikt på mjukvaruutveckling och framtagning av programvaran Monitor. Inom Development G5 finns sju team, varav sex är belägna i Hudiksvall och ett i Gävle. Varje team är antingen ansvarigt för ett specifikt område inom programvaran Monitor eller någon övergripande funktion. Det som observerades mellan de olika teamen är att alla har olika specialiserade områden och att de kan ha olika krav och förväntningar på sig på grund av det.

De roller som intervjuades under studien var Team Managers och Business Analysts.

Team Manager (TM): Leder ett utvecklingsteam och ser till att teamet arbetar mot de uppsatta målen i företaget. En TM säkerställer att alla i teamet har något att göra och att de jobbar med de mest prioriterade uppgifterna. De planerar in arbete och ser till att det blir utfört. Under arbetet har TM ett nära samarbete med teamets Business Analysts. En TM ansvarar även för en mer daglig prioritering.

Business Analyst (BA): Även nämnd ”specare” men gör mer än att skriva specifikationer och user stories och att göra dem förståeliga för utvecklarna. De tar hand om insamling av krav och är en länk mellan kunder, konsulter och utvecklingsavdelningen. En BA identifierar kundernas behov och hittar den mest effektiva utvecklingstidslinjen.

Många observerade möten gäller ett enda team. Detta team består av totalt åtta personer varav en Team Manager, en Business Analyst, fyra utvecklare och två testare. När det beskrivs att ”hela teamet var deltagande”, menas att samtliga av dessa medlemmar var med i mötet.

Med hjälp av programmet Azure Devops delas utvecklingsuppgifter upp i *Epics*, *Features* och *User Stories*. *Epics* är de övergripande funktionaliteter som ska finnas i programvaran Monitor. *Epics* bryts ner till *Features*, som specificerar en viss funktion i programvaran. Dessa funktioner fragmenteras ytterligare och beskrivs var och en i så kallade *user stories*, där det ska framgå hur användaren ska interagera med funktionen.

I många mötet används begreppet *sprintmål*. Detta innebär mål som teamet sätter ut i början av varje sprint, med konkreta mål de vill se i slutet av den sprinten. Dessa är i många fall inte *features* eller *user stories*, utan en sammanfattning på funktionaliteter varunder en rad uppgifter tillhör.

Ett viktigt verktyg som har nämnts i observationer och intervjusammanhang är *Önskemålsdatabasen*. Det är en databas som Monitor ERP har till sitt förfogande, där kundernas utvecklingsönskemål ligger samlade. Business Analysts använder denna databas som underlag för att planera och prioritera uppgifter som ska utvecklas.

4.2 Observationer

Utvecklingsavdelningen har en rad olika typer av sammankomster för att stödja prioriteringsarbetet. I tabell 2 finns en översikt över dessa sammankomster samt en kort beskrivning av deras innehåll.

Namn	Frekvens	Längd	Deltagare	Beskrivning
Stand up	Dagligen	15–20 minuter	Ett team	Kort återblick av gårdagens arbete samt genomgång av dagens arbete
Backlog Refinement	Varje vecka	30 minuter	Ett team	Finslipning och planering av backlog, 2–3 sprints framåt
Sprint Start	En gång i början av sprint	30 minuter	Alla team	Gemensamt möte för alla team vid start av ny sprint
Release Crunch	En gång i slutet av sprint	15 – 20 minuter	Ett team	Diskussion om vad som behövs för att nå sprintmålen
Sprintmål Avstämning	Varje 4-5 veckor	20-30 minuter	Ett team	Uppdatering om läget, prioritering inför nästa sprint

Sprint Retrospective	Varje 4–5 veckor	30 – 60 minuter	Ett team	Diskussion om den avklarade sprinten
Inför Produktrådsmöte	En gång per sprint	60 minuter	TM och BA från varje team, CTO	Diskussion om planering till Roadmap
Produktrådsmöte	En gång per sprint	90 – 180 minuter	CTO, Development Manager, BA från varje team	Diskussion och beslut för utvecklingsplanen

Tabell 2 ger ett kort översikt över de observerade möten inom en Sprint.

Totalt observerades åtta olika möten, alla med olika syften som ibland upprepades eller korsades i ett möte. Observationerna av dessa möten beskrivs i mer detalj under respektive rubrik.

4.2.1 Stand Up

Alla team inom utvecklingsavdelningen har vittnat om att de utövar någon form av dagligt morgonmöte där de stämmer av fram- och motgångar i utvecklingen. Upplägget för en *stand up* är dock olika från team till team. De *stand ups* som har observerats i denna studie gäller ett enda team och observationerna har skett under en period av en vecka (totalt fem möten). Alla medlemmar i teamet deltog i mötet. Varje möte tog mellan 15–20 minuter.

I *stand up* fick samtliga medlemmar i teamet en chans att väldigt kortfattat redovisa det som arbetades på dagen innan, samt vad de hade planerat att jobba på den dagen. Det uppmanade teamet att kommunicera med varandra och dela lösningar.

Det observerades relativt lite prioriteringsarbete under samtliga möten. Ett undantag är buggar; under observationerna togs det upp en del kommentarer om buggar som hade kommit in under gårdagen. I många fall påbörjades buggrättningen av dessa samma dag. Större buggar krävde mer planering och dessa tidsestimerades samt tillskrevs versionsutgåva. Sedan prioriterades dessa in i nästföljande sprint beroende på allvarlighetsgrad.

Under den perioden som observerades hade teamet relativt få problem med utveckling samt testning som påverkade prioriteringsprocessen. Trots inflöde av buggar som ofta prioriterades framför övriga uppgifter, tycktes effektiviteten av arbetet inte påverkas.

4.2.2 Backlog Refinement

Backlog refinement är ett kortare möte på 20–25 minuter där teamet går igenom den aktuella backloggen för att se om den behöver justeras. Inom det observerade teamet var alla i teamet deltagande. Totalt observerades två av dessa möten.

Båda mötena var väl förberedda och i möteskallelsen nämndes att Business Analyst har ansvaret att se till att teamets backlog är prioriterad innan *Backlog Refinement*.

Agendan för mötet innehöll följande punkter:

- Genomgång av prioriteringsändringar i pågående sprint.
- Följ upp uppgifter från föregående möte
- Gå igenom backloggens innehåll för kommande 2–3 sprintar, där målet är att ha en grovplanerad lista med *features* bestämd av Produktrådet.

Dessa *features* skulle sedan kort analyseras beroende på hur dessa kunde brytas ner till mindre *user stories*, om dessa var i behov av en riskanalys eller om beskrivningarna behövde förtydligas och kunde tidsestimeras. I riskanalysen ingår en diskussion om viss kod behöver refaktoreras för att ”*inte öka den tekniska skulden*”. I agendan nämndes prioriteringsmetoden *Scrum Poker*. Det stod att metoden kunde användas av teamet om det fanns osäkerheter omkring tidsestimeringen.

Under mötet observerades att hela teamet var deltagande, utbytte tankar och kommenterade om det fanns frågor. Relevanta *features* och *user stories* analyserades och diskuterades med hjälp av backloggöversikten med Azure Devops. Teamet kollade cirka två till tre sprintar i förväg och diskuterade om versionssättningarna fortfarande var realistiska. Teamet tog upp om det hade uppstått fördröjningar under utvecklingen samt om det fanns otydligheter inom uppgiftsbeskrivningar i backloggen. Det diskuterades även tillgänglighet av resurser och möjliga problem som skulle kunna uppstå i framtiden.

Några uppkommande *features* och *user stories* bröts ner och ordnades enligt frågeställningen; Har dessa prioriterats med Minimum Viable Product (MVP) i åtanke? I ett av mötena diskuterades även en *feature* från ett annat team som det observerade teamet skulle hjälpa till med. Deadlinen på den delegerade funktionen togs bort eftersom teamet ville göra sin egen tidsestimering samt avgöra storleken på funktionen. Team Manager påpekade att han hade varit tydlig med att inte lova utveckling av funktionen till andra teamet, utan att teamet skulle sätta sig in i uppgiften först.

Användning av Scrum Poker har inte observerats under något av tillfällena. Vid ett tillfälle fanns det osäkerhet om versionssättning av en viss *feature* och detta diskuterades inom gruppen, där de inblandade sade deras åsikter om saken.

Några faktorer som togs hänsyn till under diskussioner omkring prioriteringen var kompetens, beroenden inom och utanför teamet samt andra teams beroende av det egna teamet, efterfrågan, storlek på uppgiften och utvecklingstid. Beroenden togs upp flera gånger under mötet, i synnerhet om vissa team skulle kunna köra fast på grund av prioriteringsordningen. Testning var en omfattande del av diskussionen och i det ingick hur testningen kunde påverkas av tekniska krav samt hur mycket tid denna aktivitet skulle ta. Det observerades att testning hade egen prioriteringsordning, där versionstester prioriterades högre än andra tester inom ett tillfälle. Buggar diskuterades sist i mötet och där tog en kort diskussion plats om buggarnas allvarlighetsgrad. I båda möten prioriterades buggar framför allt annat arbete i den nästkommande sprinten.

Det observerades att kommunikation mellan testare, utvecklare och Business Analyst inom teamet skedde kontinuerligt under vanlig arbetstid. Detta för att kunna specificera, tidsestimera samt prioritera. Ett av mötena var klart efter 25 minuter. De sista 5 minuterna i detta tillfälle användes för att diskutera fler *features* som kom in från andra teams som skulle tas i framtida sprints.

4.2.3 Sprint Start

I början av en *sprint* inbjuds alla anställda i utvecklingsavdelningen till ett möte för att ha en *sprint* kick-off. Det innebar att det fanns fler än 70 deltagare i mötet och med anledning av att några av de anställda jobbar på Monitor utanför Sverige, hölls mötet på engelska. Mötet leddes av Chief Technical Officer och Development Manager och varade i totalt 30 minuter. Det bör påpekas att observerat Sprint Start möte var början av den sprint som följde efter den sprint som majoriteten av observerade möten tillhörde.

Syftet med *Sprint start* mötet var att snabbt redovisa föregående sprint och dess framgång samt introducera nästkommande sprint. Mötet tog upp syften med den nya *sprinten* och presenterade vad varje team kommer fokusera på, vilket i det här fallet var buggrättning samt optimering av prestanda.

Sprinten delades upp i flera delmål:

- Lösa fler rapporterade fel och komma i kapp med företagets buggrättningsmål
- Lösa rapporterade problem inom redan bestående programfunktioner
- Lösa kända prestandaproblem
- Fokusera på att förbättra kundnöjdhet hos nuvarande G5 kunder

Med företagets buggrättningsmål menas att majoriteten av rapporterade buggar ska rättas inom en version, det vill säga 4–6 veckor från upptäckt. Dessa delmål förde med sig att vissa faktorer och aktiviteter prioriterades högre än andra i den nästkommande sprinten:

- Mer fokus på prestanda, buggrättning och lösningen av problem under perioden april till augusti
- Kvarstående antalet buggar ska ligga under ett visst antal i slutet av augusti
- Business Analysts har uppdraget att i kundönskemålsdatabasen hitta förbättringsförslag som löser programbrister

Sprint start mötet som observerades innehöll mycket samarbete mellan samtliga team för att kunna nå de bestämda målen. Kapacitet räckte inte till inuti vissa team och det löstes med att resurser från andra team kallades in för stöd. En kommentar under presentationen påpekade att samarbetet mellan utvecklingsteamen bidrog med kundnöjdheten.

Som stöd presenterades statistik över effektiviteten. Här ingick en översikt av hur mycket tid som lagts på *user stories* jämfört med buggar, genom att jämföra *story points* med nedlagda utvecklingstiden i alla team.

Vidare gick mötet igenom varje teams specifika uppdrag och sprintuppgifter. Mötet avslutades med en genomgång av statistik över utvecklingsavdelningens produktivitet, så kallad ”Key Performance Indicators” (KPI). Ett exempel på dessa indikatorer var planerade *features* jämfört med levererade *features.* Vid sidan om statistiken observerades ett mål inom denna KPI; att optimera leveransprecisionen, det vill säga förbättra pålitligheten på planerade – levererade *features* varje månad. I statistiken ingick även hur många buggar som hade avslutats och hur många Story Points som användes under senaste 16 versionsreleaser. Det påpekades att antalet spenderade Story Points hade ökat och att detta var positivt. Statistiken visade även en trend nedåt för antalet rapporterade buggar, samt att det rättades fler buggar än att det loggades; ytterligare ett bra tecken enligt Development Managern.

4.2.4 Release Crunch

Release Crunch var ett kortfattat möte inom teamet, cirka en vecka innan avslutning av *sprinten.* Hela teamet var deltagande och i det observerade tillfället övergick en *stand up* direkt till *release crunch.*

Syftet med mötet var att skärpa arbetsfokus och diskutera vad inom den planerade sprintutvecklingen som kunde tas i mål. En agenda publicerades i samband med mötet och där framfördes fyra mål:

- Organisera backloggen så att de viktigaste uppgifterna ligger högst upp

- Bestämma vilka uppgifter som kräver mest fokus (beroende på kapacitet)
- Flytta fram de minst viktiga uppgifterna till nästa *sprint* om kapaciteten inte räcker till
- Göra teamet uppmärksam på risker vid nästa release som eventuellt kräver extra kvalitetssäkring.

Det skedde en justering av backloggen, där saker som var kopplade till sprintmål prioriterades högre och flyttades fram i listan. I denna lista ingick inte bara saker som var planerade inom sprinten, utan teamet hade tagit sig an fler uppgifter bredvid det planerade arbetet.

Justeringen av backloggen var bland annat på grund av att vissa resurser inte fanns tillgängliga. Vissa tester framsköts till följd av att kapacitet av testare inte räckte till; bland annat var en utvecklare otillgänglig under en sprint på grund av sjukdom. Trots det har teamet gjort försök att få uppgifterna i mål, men det kommenteras att uppgiftens komplexitet är större än förväntat. Det förklaras ytterligare genom att uppgiften i fråga påverkar många olika delar av programvaran.

Allmänt ansåg Team Manager att teamet låg ”ganska bra till”. Det påpekades att det fanns en risk att en planerad funktion inte skulle kunna släppas trots att mycket tid hade investerats i den under *sprinten*. Mötet avslutades med att Business Analyst skulle förbereda en plan för nästa *sprint*.

4.2.5 Sprintmålsavstämning

Sprintmålsavstämning var ett möte med hela teamet på 30 minuter som gick ut på att diskutera målen inför nästa *sprint*. Här skedde preliminär prioritering på de nästkommande *features* som skulle utvecklas. Det ingick även förberedande arbete, såsom tidsestimering av arbetsuppgifter och uppdatering av dokumentation av använd utvecklingstid.

En kortare uppdatering om den nuvarande *sprinten* samt en avstämning om fördröjningar av nuvarande *features* skulle kunna orsaka fördröjningar på nästkommande *sprint* gjordes. Teamet ansåg att de skulle nå de bestämda sprintmålen i nuvarande *sprint*. De tog ändå beslutet att flytta fram en stor del av de planerade uppgifterna. Dels på grund av att en del av dem inte var kundkopplade, dels för att det uppstått tekniska problem med hårdvara och därav fördröjningar med testning.

Det kommenterades att teamet skulle missa leveranssäkerhet men att det inte var så stor efterfrågan på dessa *user stories* just nu. En större *feature* var kundkopplad och ansågs viktig men den bestod av en så pass stor utveckling att den inte kunde påbörjas förrän flera *sprintar* framåt. En anledning till detta var bristande arbetskraft, då en medarbetare hade blivit sjuk under en längre period.

Prioritering gjordes utifrån kapacitet och det diskuterades om det förslagna arbetet var rimligt enligt alla teamets medlemmar. Det togs hänsyn till att medarbetaren som hade blivit sjuk möjligtvis inte skulle vara tillbaka för hela sprinten, samt att det skulle finnas flera röda dagar under *sprinten*. Teamet hade accepterat att utveckla vissa *features* för ett annat team som annars inte skulle bli klara inom ett till två kvartal. Det andra teamet hade gjort en förberedande prioritering baserat på vad ”de tyckte var viktigast”.

Det påpekades även att en stor del uppgifter saknade dokumentering om aktuell utvecklingstid. En ytterligare kommentar under mötet var att en *feature* visade sig vara för stor och i behov av uppdelning, vilket orsakade otydligheter i planeringen. Team Managern kommenterade att planerad kontra använd tid inte stämmer överens och att de skulle försöka synkronisera estimerat samt utfört arbete.

4.2.6 Sprint Retrospective

Retrospective är ett mindre möte för att avsluta en sprint. Mötet varade i ungefär 30 minuter och alla i teamet deltog. Det finns olika typer av Retrospective, beroende vad teamets behov är i stunden. Denna retrospective betecknades som ”light” och var inte så djupgående. I vanliga fall är det Team Manager som leder mötet men eftersom han inte var tillgänglig vid tillfället tog en erfaren utvecklare ledningen.

Mötet började med en mindre presentation om avslutade och kvarstående sprintmål. Det fanns totalt nio mål, varav åtta hade avslutats under *sprinten*.

Som hjälp under mötet användes Azure Devops digitala tavla, där alla i teamet kunde dokumentera saker som de upplevde bra, mindre bra eller önskemål under den senaste sprinten.

Några saker som framkom angående den senaste sprinten var:

- Kodgranskningar tog för lång tid
- Teamet ska våga fråga vid otydligheter
- Specning av uppgifter är inte alltid tydlig och ansvarig Business Analyst kan vara svår att hitta för tydliggörande
- Även testarna inkluderas i genomgång av *specningar* tillsammans med Business Analysts

- Andra team lägger en del tryck på teamet, som borde säga ifrån oftare
- Teamet behöver dokumentera utvecklingstiden bättre

Teamet diskuterade fritt om lösningsförslag till de olika problemen.

4.2.7 Inför Produktrådsmöte

Inför Produktrådsmötet är ett förberedande möte som tog plats några dagar innan det faktiska produktrådsmötet. I det 60 minuter långa mötet deltar en rad prioriteringsansvariga: minst en Business Analyst och Team Manager från varje avdelning, Chief Technical Officer. Syftet med mötet är att sammanställa hur varje team ligger till i förhållande till stora utvecklingsplanen, samt att diskutera prioritering av backloggen på feature-nivå.

Under öppningen av mötet kommenterades det att den nuvarande prioriteringssituationen är en speciell situation för företaget. Det förklarades att detta beror på att Monitors mål temporärt har skiftat fokus från nyutveckling till att förbättra kundnöjdheten. Det nämndes även att en konsekvens av denna omprioritering skulle bli en långsiktig framflyttning av utvecklingsplanen, totalt ett kvartal framåt.

Målet med omprioriteringen beskrevs tydligt som ”*förbättra kundnöjdheten*”. Strategin för detta presenterades i följande punkter:

- Optimera G5 för att komma närmare funktionaliteten av G4
- Balansera in- och utflödet av buggar över alla team
- Fixa kända brister i mjukvaran
- Förbättra programmets prestanda

Det fanns en tydlig kommunikation om skiftet av prioriteringsfokus igenom hela mötet. Det bestämdes att all nyutveckling skulle pausas för att prioritera optimering av programvaran i stället, dock kunde viss nyutveckling inte pausas på grund av lagstiftning och avtal. Utveckling som lovats kunder kunde heller inte pausas eftersom det skulle påverka kundernas åsikt om företaget negativt. En kommentar som uttalades om kundernas reaktion var *”det kan bli skrik”.* Allting utanför dessa kategorier skulle dock öppnas för granskning under detta förberedande möte.

En översikt om hur teamen låg till i förhållande till varandra visade att några team hade särskilt svårt att komma i mål med både den planerade utvecklingen samt teamets bugg-backlog. Responsen till det var att omfördela arbetskraften mellan några team. Under mötet observerades det att flera deltagare frågade varandra om råd och hjälp med både resurser och kunskap.

En preliminär omprioritering inom varje team hade förberetts av Team Managers. Ett mål som diskuterades var *”att inte släppa kunder”.* Varje utvecklingsteam hade särskilda områden de skulle fokusera på; för de avdelningar som låg värst till var bugg-backlogen främsta prioritering. Ett konkret mål för ett team var att få ner antalet buggar under en viss nivå.

Samtliga teams backloggar delades upp i utveckling som skulle fortsätta respektive pausas. Under genomgången blev det tydligt att flera team inte hade utrymme att pausa eller omprioritera nyutveckling. Anledningarna som nämndes var att vissa funktionaliteter redan var påbörjade och nästan klara, eller kopplade till avtal och krav. Prestandaförbättringar skulle dock prioriteras högst vid möjlighet. Därefter nämndes att kundöverföring var en viktig prioritet, vilket innebär att migrera kunder från Monitor G4 till Monitor G5.

Under genomgången upptäcktes en särskild svårighet med en funktionalitet som skulle optimeras under ”optimeringsperioden”. Funktionaliteten behövde en erfaren utvecklare med spetskompetens och utvecklaren som besatt dessa kunskaper hade en långvarig planerad semester som varade under en större del av optimeringsperioden. Detta skulle försöka lösas innan själva Produktrådsmötet.

En viktig observation var att några Team Managers tog upp tankar om kundernas reaktioner på framskjutningen av mjukvaruutvecklingen. Det diskuterades hur företaget kommer att förbereda och informera sina klienter om fördröjningarna. En åsikt som lyftes var att företaget har lovat mycket till kunder och att dessa löften inte kommuniceras tydligt mellan företaget, dess kunder och utvecklare.

Avslutningsvis uppkom en diskussion om nuvarande omprioriteringsläge. Det uttrycktes missnöje om omprioriteringen, där en Team Manager kommenterade att ”inga tycker att detta är bra aktiviteter”. Det påpekades ytterligare att denna omprioritering enbart är en temporär brandsläckning och att lösningar måste hittas för att undvika liknande situationer i framtiden. Ingen tydlig respons gavs på det.

En allmän observation var att alla planeringsförslag var preliminära. Dessa skulle tas tillbaka till varje enskilt team, där prioriteringen skulle diskuteras med hela teamet under de nästkommande dagarna inför *Produktrådsmötet*.

4.2.8 Produktrådsmöte

Produktrådsmötet följer efter *Inför Produktrådsmötet*. Här deltar minst en Business Analysts för varje team, utvecklingavdelningens Development Manager och Chief Technical Officer som leder mötet. Det är CTO:n som har som uppgift att styra alla utvecklingsteam åt samma håll, och den har även ansvaret att ta det slutgiltiga beslutet på vad som ska utvecklas. Produktrådet bestämmer vad som ska utvecklas på *Feature* nivå. Här sätts versionsrelease och prioriteras utvecklingsplanen på det stora övergripande perspektivet. Mötet tog cirka 90 minuter.

Eftersom *Inför Produktrådsmötet* lade grunden till detta möte, började mötet med en kort genomgång av de mål och syften som diskuterades i *Inför Produktrådsmötet*. Återigen betonades att detta är ett speciellt Produktrådsmöte, där en större omprioritering av utvecklingsarbetet sker under perioden av cirka ett kvartal.

Målet med omprioriteringen beskrevs således som:

- Rätta fel, buggar
- Fixa brister
- Fixa problem med prestanda

Fokus med omprioriteringen var att öka kundnöjdheten av Monitor G5. Återigen beskrevs att all nyutveckling kommer pausas och att förbättring av befintlig kod kommer prioriteras. Även målet att komma under 100 buggar var en brännpunkt. Efter att målen upprepats gick mötet igenom en kort sammanfattning av *föregående möte*.

Därefter påbörjades en öppen diskussion om eventuella problem som upptäcktes i *Inför Produktrådsmöte*. Det observerades att det fanns möjlighet att ta upp orosmoln under hela mötet. Ett problem var att den tid som hade avsatts för buggrättning inte skulle räcka till. Det kommenterades ”*Vi är tidsoptimister”,* varpå det svarades att detta skulle kikas över processen i ett lite senare skede.

Ytterligare ett problem var att det läggs särskilt tryck på utvecklingsavdelningen genom att andra avdelningar bland annat lovar för mycket utveckling till kunderna. Det förklarades att de extra resurserna som i så fall behövdes inte fanns tillgängliga. Härpå svarades att utvecklingsavdelningen måste börja ställa krav tillbaka på de avdelningar som kräver viss funktionalitet eller effektivitet.

Tillgänglig arbetskraft togs upp flera gånger till under mötet. De team som hade tillgängliga resurser erbjöd att bistå de andra teamen genom att ta över vissa *Features* eller bugg-rättning. I vissa fall var det dock inte möjligt av den orsaken att spetskompetens behövdes. Det kom även fram att anställda med spetskompetenser ofta hade en stor arbetsbörda. Alla anställda i utvecklingsteamen har extra uppgifter såsom ett halvt arbetspass på supporttjänsten per vecka, men det tar extra tid ifrån personer med särskild spetskompetens.

Det var mycket som påverkade ordningen av prioriteringarna. Backloggen beskrevs behöva städning; det fanns dubbletter, felplacerade *user stories*, buggar som inte var kundkopplade som orsakade svårigheter i att prioritera, planera och utföra arbetet. Tidsramen för denna förbättringsperiod ansågs också vara kort, där team uttryckte att de inte visste om de kunde hinna med bestämda deadlines. En Business Analyst upplyste alla om att en genomsnittlig bugg tar ungefär sex timmar att lösa. Ett team berättade att de har haft ett bugginflöde av 30+ buggar per månad.

En observation var att många avdelningar inte hade mycket utrymme att omprioritera och fokusera på prestandaförbättring. Förändringar över alla avdelningar observerades som minimala: många *Features* förändrade inte ordning och pausades inte heller. Det som observerades var att många deltagande parter ansåg att detta var en nödvändig omprioritering, men en stor del av alla teamen verkade vara låsta i de uppgifter de redan hade innan omprioriteringen. Vissa nyutvecklingar lyckades dock skjutas upp och en del *features* blev pausade tills vidare. Bara två teamsnämnde att de skulle fokusera på att överföra fler kunder från G4 till G5, det vill säga fokusera på utveckling av funktioner som fanns med i föregående utgåva av Monitor.

4.3 Intervjuresultat

I detta avsnitt behandlas och sammanställs det material som framkom ur intervjuer med Team Managers och Business Analysts från de sju olika team som utgör avdelningen Development G5 på Monitor. Svaren presenteras i löpande text och är i vissa fall grupperade utifrån frågeområde. Områdena är arbetssätt, gemensamma mål, prioriteringsaspekter och utmaningar i prioriteringsarbetet.

4.3.1 Arbetssätt

I detta stycke bearbetas frågor om när och hur ofta det behöver prioriteras samt hur mycket tid som uppskattningsvis läggs på prioriteringsarbetet.

Samtliga respondenter svarade att de använder sig av dagliga morgonmöten, så kallade *stand ups*. Vilka som deltar och vad mötena innehåller skiljer sig dock från team till team. Detta korta morgonmöte används för att medlemmar i teamet ska stämma av arbetsprocessen, se över vad som är viktigast och göra eventuella justeringar i prioriteringen (se 4.2.1).

Majoriteten av respondenterna vittnade om att justeringar i prioriteringen sker ganska ofta, ibland flera gånger dagligen. Detta kan till exempel bero på att förutsättningar eller teamets tillgänglighet förändras, eller på att kritiska problem som allvarliga buggar inträffar, vilket gör att en omprioritering av uppgifter behöver göras. En Team Manager uttryckte ”Att prioritera rätt saker är viktigt, men prioriteringen måste också kunna ändras”. Tre av de intervjuade hävdade att det sällan är stora omprioriteringar som behöver göras. Det skiljer sig även på olika nivåer av uppgifter. *Epics* verkar vara relativt låsta, *features* kan ändras ungefär en gång i månaden via Produktrådet och *user stories* samt buggar kan flyttas om dagligen.

Hur varje team arbetar utifrån agila principer skiljer sig mellan teamets storlek och placering, storleken på backloggen och ledningen från aktuell Team Manager, vilket i sin tur påverkar tiden och utformningen av en *stand up*. Respondenterna kommenterade att nya uppgifter och utvecklingskrav kommer in kontinuerligt, ibland mer och ibland mindre, vilket styr hur ofta prioriteringen görs och hur mycket tid som läggs på det. Uppgifter beskrevs komma från support-avdelningen, konsulterna och önskemålsdatabasen.

Produktrådet nämndes som ett prioriteringsmoment i respondenternas arbete, som sker ungefär en gång per månad (se avsnitt 4.2.7). Respondenterna sa utöver det att prioriteringen sker ”lite grann hela tiden” under deras arbete. En Team Manager nämnde att det var Business Analysts som gör ”stora jobbet” vad gäller prioritering.

När respondenterna fick frågan om hur mycket tid prioriteringsarbetet uppskattningsvis tog per vecka, svarade två av sju Team Managers med en tidsuppskattning. En uppskattning var ungefärligt två timmar. En annan var ungefär tio timmar per vecka, baserat på svaret att prioriteringen tog ungefär en halv timme per dag, plus en hel dag per vecka. Två Business Analysts gjorde uppskattningar på två-tre timmar per vecka samt fem timmar per vecka. Totalt svarade åtta av de tolv respondenterna att de inte visste hur mycket tid som går till prioriteringsarbete.

Samtliga av de intervjuade svarade spontant att de inte använder sig av någon prioriteringsmetod eller -teknik i prioriteringsarbetet. När det frågades specifikt om MoSCoW-metoden svarade alla att de hört talas om den. Vissa nämnde att det gjorts ett försök att använda metoden, eller att den används ”i stora drag” inom företaget. Uppfattningen var att den används mer som ett förhållningssätt, där prioritetsordningen av programvarans funktionaliteter utgår från vad som är absoluta måsten, än som en uttalad prioriteringsmetod. I det sammanhanget nämndes även Minimum Viable Product (MVP) som ett förhållningssätt.

En Team Manager svarade att han inte vet om det används någon specifik metod, mer än att de arbetar utifrån en backlog. En Business Analyst från ett annat team svarade att det inte använder någon prioriteringsmetod överhuvudtaget.

4.3.2 Gemensamma mål

Detta avsnitt redogör för vilka gemensamma mål som Team Managers och Business Analysts anser finns inom företaget och vad de tar hänsyn till när de kravprioriterar.

Frågan ”Finns det några gemensamma mål inom företaget” skapade en del osäkerhet hos de intervjuade. Många hade svårt att tolka frågan och några undrade om det fanns ett rätt svar. Det fanns många blandade uppfattningar om målen men två övergripande mål som nämndes av flest team var ”Dubbla omsättningen på fem år” och ”Kundnöjdhet”. Utöver det var det många spridda uppfattningar.

Några förmedlade på olika sätt att företaget har som mål att växa och bli större på marknaden. Exempel på det var att de skulle vara ”bäst i test” jämfört med liknande produkter, att de ska vara marknadsledande i deras nisch, samt att de ska växa och komma ut på utlandsmarknaden.

En Team Manager kunde ge ett självklart och utförligt svar på frågan och han svarade:

”Vi ska dubbla omsättningen på fem år, sedan 2018 tror jag. Det är ett stort övergripande mål. Kundnöjdheten ska ligga över en viss nivå. Personalomsättningen ska ligga på ca 3–4%. Sjukfrånvaro ska vara under 3%. Om det är över det sätter företaget in olika åtgärder.”

På frågan om vilka mål som finns inom teamet var buggrättning den överlägset mest nämnda. De flesta angav ett visst tidsspann inom vilket buggarna ska vara rättade. Buggar med kundkoppling, det vill säga de som rapporterats in från kunder till supporten, prioriteras högst. Ett undantag var de buggar som har hög allvarlighetsgrad.

Trots att det var en del oklarheter runt målen var det flera som antydde att det var viktigt att alla går åt samma håll mot en gemensam målbild. Någon uttryckte att de bör ”sitta i samma båt och ro åt samma håll”. Flertalet nämnde att skapandet av *Produktrådet* har bidragit till att få till en känsla av gemenskap.

4.3.3 Prioriteringsaspekter

På grund av det omfång som frågan ”vad tar du hänsyn till när du kravprioriterar” resulterade i, behövdes en djupare bearbetning av svaret göras. Ur bearbetningen kom en rad specifika begrepp fram som både Team Managers och Business Analysts sa sig ha i åtanke under prioriteringsprocessen. I tabell 3 visas en översikt över dessa, samt hur ofta de nämndes. Begreppen redovisas i fallande ordning från det mest nämnda till det minst nämnda bland alla respondenter. I detta avsnitt kommer Team Managers att benämnas TM och Business Analysts BA.

Hänsyn till…	Team Manager	Business Analyst	Total
Kundnytta	6/7	5/5	**11/12**
Dialog med andra	6/7	4/5	**10/12**
Kundfokus	4/7	4/5	**8/12**
Buggar	3/7	3/5	**6/12**
Förutbestämd ordning	2/7	4/5	**6/12**
Kapacitet	2/7	3/5	**5/12**
Magkänsla	2/7	3/5	**5/12**
Allvarlighetsgrad	3/7	2/5	**5/12**
Kompetenser	4/7	0/5	**4/12**
Arbeta i kapp G4	1/7	2/5	**3/12**
Marknadskonkurrens	0/7	2/5	**2/12**
Myndighetskrav	1/7	1/5	**2/12**
Avtal med kunder	1/7	1/5	**1/12**
Vet ej	1/7	0/5	**1/12**

Tabell 3 redovisar vad som togs hänsyn till vid kravprioritering.

De tre mest framträdande begreppen som nämndes i det här sammanhanget var *kundnytta*, *dialog med andra*, och *kundfokus*. Dessa begrepp kommer att sammanfattas individuellt nedan enligt deras ordning i tabell 3. Därefter kommer resterande begrepp ur tabellen redovisas utan inbördes ordning, styrt av hur nära relaterade dessa är till varandra.

Kundnytta är ett begrepp som berör funktioner som är till användarens fördel. Det kan vara funktioner som kunderna efterfrågar eller som användarna själva inte är medvetna om att de behöver. Det är funktioner som TMs och BAs anser vara nyttiga för programmet Monitor och som enligt dem gynnar kunderna i förlängningen.

Dialog med andra innebär att TMs och BAs tar kontakt med andra personer för att få stöd i prioriteringsprocessen. Det handlade bland annat om att få andra åsikter, bättre insikt i kundönskemål eller information om kundbeteende. Både TMs och BAs nämnde tillgängliga konsulter, utbildare och support-avdelningen som viktiga kontakter under prioriteringsarbetet. En specifik konsult som nämndes ett flertal gånger av framför allt BAs var Key Account Managers (KAM), som representerar större kunder. De flesta TMs nämnde även att de ofta involverar det egna teamets BA i prioriteringsprocessen.

Begreppet *kundfokus* innebär uppgifter som gör kunden nöjd. Det nämndes av totalt åtta av tolv respondenter. *Kundfokus* sker i form av att fullfölja uttalade önskemål från kunder och i största allmänhet fokusera på vad kunderna vill ha. Detta begrepp skiljer sig ifrån *kundnytta* utifrån att *kundfokus* enbart tar ett kundperspektiv. Det fanns en medvetenhet mellan alla respondenter att kundfokus är ytterst viktigt för företaget, vilket förstärktes med ett uttalande om att kunder kan hota med att lämna Monitor om de inte får de funktionaliteter de vill ha. Många BAs kommenterade att de räknar kundönskemål i önskemålsdatabasen och använder det som prioriteringsstöd för att välja vad som ska utvecklas först.

Hälften av de tillfrågade nämnde buggar som en viktig aspekt att ta hänsyn till under prioriteringen. Somliga respondenter kallade buggar för ”brus”, där buginflöde ansågs ha en betydlig påverkan på det ”vanliga” uppgiftsflödet. I alla avdelningar prioriterades buggrättning högt, beroende på hur kritiska dessa ansågs vara. En del avdelningar handskades med fler buggar än andra beroende på området som utvecklades, vilket gjorde att inte alla behövde prioritera buggar på samma sätt. En avdelning hade under en period prioriterat ner buggrättningen på grund av krav från kunder att utveckla nya saker. Där kommenterades att det fanns en kostnad som behövde betalas av i nästföljande *sprint*; fler buggar kom i ett senare skede i stället. Avdelningen bestämde efter det att prioritera buggrättning hårdare.

Under intervjuerna framkom att det under utvecklingsprocessen ofta finns ett kontinuerligt flöde av uppgifter som kan anses vara viktiga. Härunder faller buggar, men även funktionaliteter som behövde prioriteras högre än andra uppgifter. Ett specifikt exempel som nämndes var att en ny kund krävde en ny funktionalitet, vilket gjorde att nyutvecklingen prioriterades över redan prioriterade uppgifter i syftet att inte tappa en möjlig framtida kund.

Kapacitet är en faktor som innebär tillgänglighet av personal, hårdvara och tid. Med tillgänglighet av personal menas närvaro, kompetens och kunskap. Kapacitet i form av hårdvara kan innebära begränsningar i datorkraft, utrymme och andra tekniska behov som i stunden inte uppfylls. Både tid och arbetskraft nämndes vara en begränsning och så kallade ändliga resurser, som på något sätt behöver fördelas på de uppgifter som ska behandlas. Här var det flera respondenter som nämnde att de behöver väga funktionaliteter mot varandra utifrån storlek och antal. Ibland behöver det beslutas om det är mer fördelaktigt att utveckla en stor funktionalitet som tar längre tid än flera mindre på samma tid. I sådana fall tas det hänsyn till den största fördelen, vad som ger mest nytta mot minst insats.

För att kunna bedöma vilka uppgifter som kan göras, och således prioriteras i viss ordning, beskrevs begreppet *Kompetens* som en styrande faktor i samband med *Kapacitet*. Det begreppet handlar om specifika kunskaper som anställda på företaget har och om dessa var tillgängliga under perioden av utvecklingsarbetet. Respondenterna berättade att dessa roller i många fall har specifika spetskompetenser eller sakkunskaper för att kunna utveckla vissa funktionaliteter. Beroende på tillgängligheten av dessa ansvariga prioriterades sprint-backloggen olika. Detta var en särskilt viktig aspekt för TMs, varav fyra av sju svarade att detta var någonting som togs hänsyn till under prioriteringsarbetet. Det kommenterades att utvecklingsuppgifter ofta kräver spetskompetens som ett fåtal personer besitter. Om det finns flera uppgifter av den sorten behöver det prioriteras mellan dessa eftersom en och samma person inte kan arbeta med för många saker samtidigt.

Respondenterna nämnde att inte alla uppgifter prioriteras på sprint-nivå. *Bestämd utvecklingsordning* innebär ordningen på *Epics* och *features* som bestäms i Produktrådet. *Epics* är de övergripande funktionaliteterna i programvaran Monitor, och dessa kan principiellt inte omordnas på sprint-nivå, utan är ”låsta” i en viss ordning. Ordningen på dessa måste tas hänsyn till under prioriteringsarbetet, därför att deadlines och versionsnummer är kopplade till de bestämda uppgifterna.

Ett uttryck som nämndes i samband med *bestämd utvecklingsordning* var *Arbeta i kapp G4*. Respondenterna nämnde att utveckling av helt nya funktioner var viktigt men att inkludering av tidigare tillgängliga funktioner var särskilt prioriterat just nu därför att kunderna ofta saknade dessa när de ”uppgraderade” till den nya versionen av Monitor.

Marknadskonkurrens var någonting som två av fem BAs var mycket insatta i och tog hänsyn till vid prioritering. De beskrev att de hade särskild koll på funktionaliteter som konkurrenter till Monitor har eller saknar, för att kunna bli mer konkurrenskraftiga på marknaden.

Myndighetskrav samt *Avtal med kunder* är två nära relaterade begrepp som nämndes under intervjuerna. Myndighetskrav innebär lagar och riktlinjer som företaget måste förhålla sig till under bland annat expandering till nya länder. Dessa är inte förhandlingsbara. Avtal med kunder är även de låsta uppgifter som de behöver förhålla sig till, även om det inte går under lagstiftade krav. Enligt respondenterna hamnar de ibland i situationer där de skriver avtal med kunder för att behålla eller värva dem, vilket kan dra med sig böter och viten om avtalen av olika anledningar inte lyckas hållas.

Frågan om vad som tas hänsyn till under prioriteringen upplevdes som en särskilt svår fråga och majoriteten av respondenterna tog sig tid för att tänka igenom ett svar. En respondent nämnde att de inte vet vad som tas hänsyn till under prioriteringsarbetet, utan menade att det mer handlade om *magkänslan*. Lite under hälften av respondenterna har svarat att mycket av resonemanget kring deras prioritering grundar sig i erfarenhet och kunskap som ger en känsla för vad som borde prioriteras först. Det nämndes uttryck som att ”gå på känsla”. En BA uttryckte att det ofta är den personliga åsikten som avgör och att den i sin tur är styrd av känsla.

4.3.4 Utmaningar i prioriteringsarbetet

Detta avsnitt innehåller åsikter om hur prioriteringsarbetet fungerar, följt av vilka utmaningar som upplevdes av respondenterna.

Åsikterna gick isär angående hur prioriteringsarbetet fungerar och endast hälften av de intervjuade har svarat på denna fråga. Ett team har sina fasta, återkommande arbeten och tyckte därför att det fungerade bra. Teamet hinner i dagsläget med det som ska göras men tror att det kommer bli svårare i och med att företaget växer. Ett annat team nämnde hur slitsamt det är med prioriteringar och att det tar enormt med resurser. Ytterligare ett team sade att de lyckas hålla leveranserna i 50% av fallen, vilket de själva trodde kunde bero på att de lovar saker för tidigt. En Team Manager från ett lite mindre team berättade att de kör med kortare sprints på 2 veckor och att det fungerar väldigt bra. Teamet tyckte att det var överblickbart och att det inte blev så långa förseningar med saker om det drog ut på tiden.

Majoriteten av de tillfrågade upplever att prioriteringsarbetet är komplext. Den största anledningen till detta, som flest respondenter angav, är svårigheten att bedöma hur viktiga uppgifter är i förhållande till varandra. Någon uttryckte att det är ”svårt att veta var det brinner mest”. Denna uppfattning var återkommande hos flera av respondenterna och innefattar både svårigheten att bestämma buggarnas allvarlighetsgrad samt vad som flest kunder blir nöjda av. Flera nämnde också att resultatet av en prioritering ofta för med sig att något blir lidande eller att någon blir missnöjd.

När det kommer till att tidsuppskatta *features* och *user stories* skiljde sig åsikterna åt mellan teamen. Ungefär hälften ansåg att de är bra på det, medan andra hälften upplevde att det är en svår uppgift och att de inte riktigt lyckas göra en bra uppskattning. Några av de senare menade att tidsuppskattningen av själva uppgiften ofta stämmer bra men att det i slutändan ändå inte håller hela vägen på grund av alla oplanerade och oförutsedda händelser som dyker upp i en *sprint*. En respondent påpekade även den mänskliga faktorn som orsak till att prioriteringen ibland inte håller. Dessa aspekter menade respondenterna påverkar utvecklingsprocessen på så vis att en del av de prioriterade funktionaliteterna inte hinns med under tänkt sprint. Således behöver de flyttas fram till nästa sprint och därmed även till senare releaseversion.

Tid var generellt en aspekt som ur olika perspektiv lyftes fram som en utmaning. Både Team Manager och Business Analyst från ett av teamen uttryckte stora svårigheter med att hinna med testning av funktioner. De ställde sig frågande till varför tidsestimering av testningen inte ingår på samma sätt som för utvecklingen. Ytterligare en Business Analyst från ett annat team var av åsikten att tidsestimeringen ska gälla både utveckling och testning. Övriga uppfattningar var att saker av olika anledningar ofta drar ut på tiden och att funktionaliteter i programvaran ibland behöver skalas bort på grund av tidsbrist. En respondent beskrev tid som en resurs och att utmaningen främst ligger i att använda de resurser man har på bästa sätt.

Alla oplanerade och oförutsedda händelser var en återkommande aspekt som nämndes under intervjuerna. Två av teamen berättade att de har en relativt jämn ström av uppgifter, som är återkommande under varje sprint. De upplever därför inte samma oförutsägbarhet som övriga team. Ett av teamen angav plötsliga och oförväntade saker som huvudorsaken till de utmaningar som de ställs inför i prioriteringsarbetet. Exempel på plötsliga och oförväntade saker är akuta buggar, där något fel uppstår i programvarans mest grundläggande funktioner.

En utmaning som lyftes fram flera gånger var den snabba expanderingen av företaget och dess starka vilja framåt. Just den starka ambitionen hos Monitor ERPs anställda var något som också framhävdes som en styrka i företaget, men under intervjuerna lyftes särskilt två svårhanterliga konsekvenser av detta i kombination med företagets expansion fram. Det ena var att den snabba takten framåt ibland leder till att saker inte blir helt färdiga; att man lämnar grundläggande funktioner för tidigt och i stället börjar på funktionaliteter som skulle kunna vänta. Det andra var att målet att expandera stort kolliderar med målet att skapa nöjda befintliga kunder, vilket främst görs genom att hålla nere antalet buggar och inkludera befintlig funktionalitet från Monitor G4 i utvecklingen av Monitor G5. Det kommenterades att kapaciteten inte räcker till att göra båda med lika stort fokus.

Ungefär hälften av de intervjuade upplevde svårigheter med att balansera kundönskemål mot funktionaliteter som gynnar utvecklingen i stort. Några av respondenterna upplevde att det är svårt att avgöra hur mycket inflytande kunder ska ha vad gäller vilka funktionaliteter som ska utvecklas. De menade att många kunder ser till egen vinning och saknar ett nödvändigt helhetsperspektiv. I vissa fall blir det ”den som skriker högst” som får sin vilja igenom, vilket några av respondenterna uttryckte att de vill försöka komma ifrån. Det framkom även åsikter om att det ibland lovas för mycket gentemot kunderna, vilket ofta slår tillbaka i form av orimliga krav på utvecklingen och leveranstiden. Kunder, utbildare och säljare vill gärna ha fasta datum på när saker ska vara klara, vilket kan vara svårt att tillgodose.

Det var ingen större skillnad i vilka utmaningar de olika rollerna upplevde med prioriteringsarbetet. Dock var det enbart Team Managers som uttryckte att ”specningen” ibland var bristande, vilket kunde försvåra tolkning av uppgifter och därmed prioriteringen.

5 Diskussion

Detta avsnitt är uppdelat i två olika diskussionsdelar utifrån resultat och metod. I resultatdiskussionen kopplas teori till det material som framkommit under observationer och intervjuer och den är organiserad enligt studiens frågeställningar. Metoddiskussionen diskuterar studiens urval, genomförande och bearbetning.

5.1 Resultatdiskussion

Svaren som framkom i intervjuerna var generellt ganska lika mellan Team Managers och Business Analysts. De skiljde sig mer åt beroende på team än beroende på vilken roll eller titel den tillfrågade personen har. Rollerna har en del liknande uppgifter men hur Team Managers och Business Analysts kan utföra dessa uppgifter beror på uppbyggnaden av teamet, tillgängliga resurser samt aktuell arbetsbörda.

Diskussion runt resultatet av denna studie är komplex, eftersom alla aspekter av prioritering påverkar varandra, direkt eller indirekt. Ett ämne leder naturligt till nästa. Upplägget av resultatdiskussionen är dock av en ordning där främst studiens frågeställningar ämnas behandlas.

5.1.1 Varför är prioritering viktigt?

Att organisera krav utifrån en viss prioritet tydliggör vad som är viktigt. Utifrån det som har observerats på företaget sker kravprioritering kontinuerligt. En satt prioritering behöver ofta justeras, på grund av olika omständigheter. Det som blev tydligt när vi undersökte arbetet på Monitor ERP är att de just nu befinner sig i en speciell situation, där mycket fokus ligger på att stärka *kundnöjdheten*. Denna omprioritering har visat hur viktigt det är att prioritera rätt saker inom rätt tid, vilket diskuteras med hjälp av teori och resultatdelen.

Det nämns i litteraturen att prioriteringsarbetet bestämmer om programmet kommer innehålla funktioner som faktiskt används. Det observerades att Monitor ERP har bra översikt över kundernas behov, därför att de har flera verktyg till hands för att kartlägga kundönskemål. En av dessa var *önskemålsdatabasen*, som observerats användas av framförallt Business Analysts.

Utifrån undersökningen från The Standish Group, som nämns i teoriavsnittet, indikeras att kundernas åsikter är viktiga att ta hänsyn till eftersom det är de som ska använda produkten i slutändan. Monitor ERP verkar vara på rätt bana genom att involvera kundernas åsikter och önskemål som grund för deras prioriteringsordning. Det ser till att relevant utveckling sker och att produkten genererar *kundnytta*. Samtidigt har, precis som respondenterna själva uttrycker det, inte kunderna helhetsperspektivet. Om allas önskemål skulle tillgodoses skulle kvaliteten av produkten kunna påverkas. Det kan argumenteras att den nuvarande omprioriteringssituationen som Monitor ERP befinner sig i är ett tydligt tecken på det; förut har det funnits mycket fokus på kundnytta och nyutveckling. Icke-funktionella krav spelar däremot även en betydande roll i utvecklingen, och det är Team Managers, Business Analysts, utvecklare samt testare som har insikten till dessa krav. Det är en balansgång av dessa perspektiv som gör att prioriteringen kan fortsätta vara balanserad.

I teorin beskrivs att uppskattningar gjorda utifrån en bra prioritering medför att ett företags pålitlighet ökar gentemot sina kunder. Prioriteringen kan ge kunderna förtroende för företaget och höjer mest sannolikt även *kundnöjdheten* eftersom utvecklare kan lansera programuppdateringar i en stadig takt. Även prioritering av prestanda och buggar kan öka pålitligheten. Under intervjuerna nämndes det att Business Analysts räknar kundönskemål utifrån *önskemålsdatabasen.* Härunder faller även prestandaförbättringar och eventuella programfel. Beroende på hur många felanmälningar som har kommit in gör både Team Managers samt Business Analysts en bedömning på vilka saker som behöver mest uppmärksamhet. Kunderna märker av detta enligt kommentarer som gjordes i *Produktrådsmötet,* vilket verkar ge Monitor ERPs kunder förtroende för företaget och programvaran.

Prioriteringen tillåter utvecklare att göra trovärdiga estimeringar om vilka funktionaliteter som ska finnas med i en versions-release. Hos Monitor verkar detta vara uppenbart: genom att samtliga team prioriterar får de en uppfattning om vilket arbete som ska göras under en viss period, till exempel under en *sprint* eller ett kvartal. Särskilt inom kortare perioder är det möjligt att använda prioriteringen för att göra en realistisk planering, vilket kan ses i intervjusvaren under *Arbetssätt*.

Under *Varför Prioritera* i teorin beskrivs det att det kontinuerligt inhämtas ny information om kraven och funktionaliteterna. Att denna nya information minskar osäkerheterna runt utvecklingen under projektets gång. Eftersom mjukvaruutveckling är en iterativ process, där utveckling ofta sker i flera varv, är det logiskt att prioriteringen kommer behöver justeras på grund av det.

Det är dock viktigt att ny information förmedlas tydligt och noggrant, eftersom kravprioritering ibland enbart sker utifrån de beskrivningar som finns av uppgifterna. Några av respondenterna beskrev att *user stories* och *features* ofta innehöll otydlig information. Detta var även en förbättringspunkt observerat under *Retrospective*.

Teorin påpekade även att vissa krav behövs väljas bort eller senareläggas, med andra ord "bortprioriteras", för att kunna leverera i tid och till tänkt kostnad. I samband med den stora omprioriteringen som skedde på Monitor ERP under studien observerades tydliga presentationer som redovisade företagets prioriteringar. De flesta teamen har kommenterat att de har en bra översikt över arbetsflödet där alla prioriterade krav ligger, så att det är lättare att flytta runt och bearbeta prioriteringsordningen. Återigen är detta styrkande för hur viktigt prioritering är för att få en översikt på föränderligheten av utvecklingsprocessen.

5.1.2 Vilka kravprioriteringsmetoder används av Monitor ERP System?

Prioriteringsmetoder är enligt den teoretiska undersökningen ett stort stöd i kravprioritering. Det finns en stor variation av dessa; vissa är simpla och av övergripande karaktär medan andra är detaljerade och av ett komplicerat slag. Valet av vilken metod som bör användas kan sägas vara beroende på situation och uppgiftens omfång samt företagsmål.

Ingen av de intervjuade nämner att de använder sig av någon specifik prioriteringsmetod. I några av observationerna framkommer en försiktig antydan till att *Planning Poker* vid behov kan användas vid estimat. Vissa av teamen i Monitor kan anses använda någon form av *Planning Poker* vid estimeringen, utan att de nödvändigtvis benämner det så. Det är oklart om de är medvetna om att det är en uttalad prioriteringsmetod. Däremot observerades *Minimum Viable Product* användas som grundtanke för att prioritera minimikraven i en mer omfattande uppgift. En metod som kunde ha använts i samband med detta enligt teorin är *Walking Skeleton*, men denna metod nämndes inte. Samtliga observerade möten var semi-strukturerade, utan särskilda metoder eller tekniker för att prioritera.

MoSCoW nämns i resultat men utifrån intervjuerna och observationerna framgår det tydligt att de få som använder sig av den enbart gör det ytterst övergripande. Den används snarare som en filosofi än en metod. Enligt teorin är det utifrån *MoSCoW*-metodiken viktigt att alla involverade i prioriteringen är väl insatta i vilka funktionaliteter som är viktiga för produkten samt i företagets mål och värdegrund. *MoSCoW* kan inte användas effektivt om medarbetarna inte har koll på detta. När vi frågade medarbetarna är det dock inte alltid helt klart vilka mål som hägrar i företaget. Det framgår i studien att det finns en del osäkerheter om vilka gemensamma mål som finns.

Det som har observerats under studien på Monitor ERP är att prioritering sker kontinuerligt i flera olika typer av situationer, stora som små, och är ett naturligt inslag under hela arbetsprocessen. Att sätta sig in i en prioriteringsmetod är tidskrävande. Det kan diskuteras om det är mödan värt det arbete som krävs för att nyttan med att använda en metod ska finnas.

En av anledningarna till att ingen prioriteringsmetod helt har fått fäste i företaget kan vara just komplexiteten kring det, att det är för mastigt, överväldigande och tidskrävande. Det kan också vara anledningen till att majoriteten av de intervjuade säger sig mer gå på känn. Även en magkänsla baseras dock på något, i de flesta fall en viss erfarenhet eller kunskap.

Majoriteten av de tillfrågade kunde inte svara på hur mycket tid som lades på prioriteringen, vilket skulle kunna indikera att prioriteringen inte är en tydligt strukturerad del av arbetsprocessen. Dock visade det sig att prioriteringen är ett så pass naturligt inslag av arbetsprocessen på företaget, att respondenterna knappt verkar reflektera över att de gör det. Mycket verkar mer eller mindre ske per automatik. Det skulle kunna vara en anledning till varför så många av respondenterna uttrycker att det ofta är känslan som styr. I vissa fall märks dock att det skulle behövas en tydligare struktur, så att de exempelvis inte hamnar i situationer där den som skriker högst får sin vilja igenom, vilket några av de intervjuade vittnat om.

5.1.3 Vilka faktorer är värdefulla för prioriteringsprocessen?

Ett av målen med studien var att undersöka vilka prioriteringsfaktorer som är värdefulla för prioriteringsprocessen. Ur litteraturstudien framgick det dock väldigt tydligt att dessa prioriteringsfaktorer är beroende på utvecklingsområde samt vilken typ av produkt som utvecklas. Det som har visat sig vara viktigt att ta hänsyn till vid prioriteringen på Monitor ERP kan därför inte anses vara en generell sanning. Däremot finns det flera paralleller som kan dras mellan de prioriteringsfaktorer som har framkommit ur litteraturstudien och de prioriteringsaspekter som studien hos Monitor ERP mynnat ut i. Det indikerar att det finns elementära prioriteringsfaktorer som högst sannolikt går att applicera på agila mjukvaruutvecklingsprojekt i stort. I den här diskussionen presenteras de faktorer och aspekter som har tydliga, uppenbarliga kopplingar mellan varandra.

Enligt Alayahri et al. är *leverans med avseende på tid* en av de mest framstående prioriteringsfaktorerna. En del av Torrecilla-Salinas faktorer kan ses relaterade till dessa, bland annat *minimi-kraven till leverans av första modell*, *första release-datumet för en användbar produkt* och *första release-datumet som kan skapa vinst*. Alla dessa faktorer tar hänsyn till att leverera produkten så fort som möjligt, dock utifrån två olika perspektiv. Ett är *användbarhet* eller *kundnyttan* som framkom ur intervjuerna med Monitor ERP. Att komma fram till en första modell och efter det en användbar produkt görs genom att företaget involverar kunderna i utvecklingsprocessen via kundönskemål. Det andra perspektivet kan anses relaterade till ytterligare en faktor av Alayahri et al; *intäkter och affärsvärde*. Här finns en koppling till Torrecilla-Salinas *första release-datumet som kan skapa vinst*. *Leverans med avseende på tid* kan därför argumenteras vara huvudkategorin dit de andra faktorerna hör.

Att leverera en produkt i tid är en komplex uppgift men ur observationerna och intervjuerna är det tydligt att denna faktor även är en självklarhet i Monitor ERPs prioriteringsprocess. I *Inför Produktrådsmöte* beskrivs vikten av att leverera vissa funktionaliteter i tid för kunderna eftersom konsekvensen annars kan bli missnöje. I *Sprint Start* används ett särskilt begrepp som sammanfattar konsten att estimera samt leverera i tid, nämligen *leveransprecision*. Begreppet beskriver ett teams skicklighet i att estimera tiden det tar att utveckla samt leverera funktioner i tid. Det är även någonting som *Produktrådet* håller uppsikt över, som redovisade samtliga teams genomsnittliga *leveransprecision* med hjälp av statistiska *Key Performance Indicators* i *Sprint Start*. Det tyder på att de lägger stor vikt vid att hålla sina deadlines.

Att Monitor ERP värderar *pålitlighet,* som är en av de prioriteringsfaktorer som Alayahri et al.s nämner, kan motiveras genom att företaget håller bra koll på *Key Performance Indicators,* och i synnerhet *leveransprecision*. I *Sprint Start* beskrevs det att företaget även gick igenom statistik över bland annat hur framgången i buggrättning och utveckling för *user stories* låg till. Att kunna hålla ordet gentemot kunder och hålla en stadig utvecklingstakt är någonting Monitor ERP som företag lägger en tydlig mängd tid och energi på. Det är inte enbart via statistik som faktorn *pålitlighet* synliggörs. I *Inför produktrådet* uttryckte flera Team Managers tankar om hur det kommer uppstå fördröjningar i leveranser på grund av det större omprioriteringsarbetet som pågick under studiens gång och hur detta riskerar att göra flera kunder besvikna.

Utifrån intervjuerna med Team Managers och Business Analysts framkom att de olika teamen behöver prioritera olika saker. En del respondenter berättade under intervjuerna att de upplevde mer tryck i relation till *myndighetsavtal* samt *avtal med kunder*. Denna aspekt som Monitor tar hänsyn till kan ytterligare kopplas till Alayahri et al.s *intäkter och affärsvärde*. Det var i synnerhet de team som var beroende av avtal som också hade större krav när det gäller nyutveckling. I observationerna av *Inför Produktrådsmöte* och *Produktrådsmöte* framgick att dessa team inte hade mycket utrymme att omprioritera sina arbetsuppgifter eftersom de stod under avtal. Det framkom i intervjuerna att det är viktigt för företaget att följa avtal på grund av att det kan ha flera negativa konsekvenser om de inte gör det, bland annat viten och böter. Som nämnts i intervjuresultat är företaget beroende av vissa myndighetskrav för att kunna fortsätta expandera utomlands.

Monitor ERP använder *Road Map* som en långsiktig utvecklingsplan. Med hjälp av *Produktrådet* där bland annat Business Analysts ingår har det framlagts en ordning på vad som ska utvecklas. Planen har som avsikt att vara preliminär, men ur den kommer en viss bestämd utvecklingsordning som försöker följas. Det är här flera respondenter har nämnt att de tar hänsyn till "*Förutbestämd ordning*", och den ordningen bestämmer till viss del prioriteringen av *sprints*. Bland detta ingår hänsynsaspekten *att arbeta i kapp G4*, där funktionaliteter som fanns i den tidigare versionen av Monitor ska implementeras i den nya G5-versionen. Road Map används bland annat som en fingervisning på vilka funktionaliteter som ska utvecklas. Som följd observerades det att teamen upplevde en del press på grund av detta, eftersom fingervisningarna kunde tolkas som löften. Road Map, som består av *Epics*, har funktionaliteter som nuvarande samt möjliga framtida förväntar, och blir således låsta mer än bara vad *Produktrådet* bestämmer.

Framtagningen av en ny generation av produkten Monitor har medfört en stor del nyutveckling. Det har således uppstått mycket prestanda-relaterade uppgifter. Eftersom det naturligt blir en del reaktioner från kunderna har prioriteringsarbetet påverkats till att ta än mer hänsyn till kundernas önskemål än vanligt. Just nu är det förbättring av *kundnöjdheten* som väger tyngst, där förbättringar i *prestanda* och *kvaliteten* på mjukvaran Monitor prioriteras över andra faktorer.

Monitor ERP har valt att prioritera prestanda och buggrättning, vilket som nämnd ovan anses komma bidra en höjning av *kundnöjdheten*. Men utifrån omprioriteringen som skedde på Monitor ERP är även Alayahri et al.s F*aktiskt kvalitet vad gäller produkt, kod, arkitektur eller robusthet* lika värdefull. Under *Inför Produktrådmötet* kommenterades det att omprioriteringen ansågs vara en brandsläckning, och det kan möjligtvis bero på att *kundnöjdhet* enbart prioriterades utifrån vad kunderna efterfrågade; med andra ord fanns det ett stort *kundfokus*. Det är därför *kundnytta* anses vara en prioritet konkurrerar med *kundfokus*.

Kundnytta och *kundfokus* var två av de mest tydligt kommunicerade aspekterna att ta hänsyn till vid prioritering. Det kommuniceras via många olika kanaler, varför *Dialog med andra* synliggjordes ur resultatet. *Önskemålsdatabasen,* konsulter, Key Account Managers samt utbildare är källor till kravprioriteringen; alla är roller som står väldigt nära kunderna. *Dialog med andra* kan anses vara nära besläktat med Alayahri et al. prioriteringfaktor *Kundrelation*. Denna faktor är viktig därför att den bidrar till att skapa en betydelsefull produkt.

De involverade kanalerna verkar dock främst fokusera på *kundfokus*; vilka funktionaliteter kunderna önskar använda i programvaran Monitor. Som resonerats tidigare i diskussionen anses *kundfokus* resultera i *kundnöjdhet*, vilket även teorin beskrev som ett vanligt antagande. Enligt teorin är det mycket mer som förbättrar *kundnöjdheten*. Bland annat nämndes det att prioriteringsprocessen, inklusive dess mål, processer och faktorer sammanlagt förbättrar den genomsnittliga *kundnöjdheten*. Därför kan det argumenteras att det är mer naturligt att se *kundnytta* samt *kundfokus* som aspekter som passar in under *kundnöjdhet*. Med andra ord finns det en naturlig hierarki i dessa faktorer. Det som observerades under studien är att Business Analysts verkar lägga mer vikt vid *kundfokus* än vid *kundnytta*. Oavsett vilken av de två faktorerna som står i fokus verkar Monitor ERP dock alltid ta hänsyn till *kundnöjdhet*.

Inga av respondenterna nämner i intervjuerna specifikt begreppet kostnad som något de tar hänsyn till vid prioritering av funktioner. Dock belyser Alahyari et al. kostnad som en produkt av tre aspekter; tillgängliga resurser, värdet för kunden och företagets vinst. Resurser kan kopplas ihop med hänsynsaspekten *kapacitet* som innebär tillgänglighet av personal, hårdvara eller utvecklingstid. Ur observationerna på Monitor ERP framgick att vissa uppgifter fördröjdes på grund av bristande *kapacitet*. Under intervjuerna nämndes det flera gånger att team har behov av fler utvecklare och testare, och att den tillgängliga arbetskraften inte räcker till. I *Sprintmålsavstämning* beskrevs hur en medarbetare hade blivit sjuk, och att flera fördröjningar hade uppstått som följd av det. I prioriteringen observerades att det togs hänsyn till att denna kollega möjligtvis inte skulle vara tillbaka än, och även eventuella semesterdagar togs med i uppskattningen om vad utifrån denna *kapacitet* kunde utvecklas för nästkommande *sprint*. Således anses *kapacitet,* en subkategori av *kostnad,* ändå vara en faktor som Monitor tar hänsyn till under prioriteringsarbetet.

Även om några faktorer inte har nämnts av respondenterna betyder inte det att dessa inte är viktiga eller inte tas hänsyn till. Vissa team hade flera Business Analysts och dessa delar gemensamt ansvaret för prioriteringsarbetet i ett team. Det kan därför skilja sig per team och per Business Analyst vad som prioriteras och vilka krav som styr.

5.1.4 Vilka utmaningar följer med kravprioritering?

Utmaningar upplevs under alla utvecklingsprocesser och mjukvaruutveckling är definitiv inget undantag. Under litteraturens *Utmaningar inom prioriteringsprocessen* nämns det en rad utmaningar utifrån ett översiktligt perspektiv på prioritering. Därunder nämndes komplexiteten av system som innehåller många subsystem, vilket anses vara särskilt relevant i Monitor ERPs situation. Avdelningen Monitor G5 är uppdelad beroende på vilken del av programvaran som utvecklas, och trots att de är skilda är de inte oberoende av varandra.

Teamen behöver samarbeta för att tillsammans utveckla den totala produkten Monitor, även när de är skilda av både *tid* och *rum*. Varje team jobbar inom olika områden av programvaran och således utvecklas *features* och *user stories* som kan vara relaterade till varandra i olika tempo. Under observationerna framkom det att det observerade teamet tar särskild hänsyn till utvecklingstakten av en annan avdelning och prioriterar utifrån dess utvecklingshastighet. Fördröjningar av ett team kan således påverka ett annat team.

Enligt ett agilt förhållningssätt är den iterativa processen ett självklart inslag. I fallet med prioritering kan det tänkas gälla justering av backlog eller av andra prioriteringar. Monitor ERP stod under en stor omprioritering under tiden för observation. Någon uttryckte det som att det *"inte är kul för någon*". I intervjuerna kom det även fram ett förhållningssätt där man till största mån vill undvika omprioriteringar för att det upplevs som en utmaning. De flesta var dock medvetna om att justeringar var oundvikliga och förstod syftet med det i och med den agila arbetsprocessen. Omprioritering är i grund och botten essentiell och gynnsam för utvecklingen, vilket diskuterades tidigare i *Varför är prioritering viktigt?*

"Att sätta kundnöjdhet först" samt att "*företaget ska expandera i en viss takt"* är de två tydligast förmedlade företagsmålen enligt respondenterna, och dessa upplevs gå stick i stäv med varandra. Det påpekades av flera Business Analysts att det inte går att utveckla i den takten som företaget önskar med bibehållen kvalitet, medan de samtidigt även ska ta hand om prestandaförbättring och buggrättning. Både Team Managers och Business Analysts var ense om att det är svårt att balansera båda företagsmålen.

Tidsestimering är en svårighet som nämns flera gånger under resultat. Respondenter får kontinuerligt ny information, vilket drar med sig nya uppgifter som kommer in kontinuerligt, bland dessa *buggar*. Dessa gör att inte allt arbete som behöver ingå i en *sprint* kan överses, vilket försvårar arbete med estimeringar. Om tidsuppskattningar är dåliga leder dessa till en schematisk risk; att saker och ting inte hinns med vilket redogörs i teorin. Oförväntade uppgifter menade respondenterna påverka utvecklingsprocessen på så vis att en del av de prioriterade funktionaliteterna inte hinns med under tänkt sprint. Tidsestimering är en komplex men essentiell del av prioriteringsarbetet, därför att många beslut grundar sig i en uppfattning om hur mycket tid utveckling och testning kommer ta. Problem med tidsestimering kan således ha stora konsekvenser för prioriteringen.

Enligt avsnittet *Varför prioritera* i teorin kan prioriteringsarbete minska de risker som finns inom mjukvaruutvecklingen. Under observationerna och intervjuerna påpekades dock bara två av de intervjuade att de tog hänsyn till oförväntade uppgifter och buggs, eller rättare sagt risker. Under observationer användes begreppet riskanalys en gång under ett möte; här togs hänsyn till beskrivningarna av *user stories* behövde förtydligas och om dessa kunde tidsestimeras realistiskt och trovärdigt. Det beskrevs att riskanalys var nödvändigt för att ”*inte öka den tekniska skulde*n”. I detta sammanhang framgick det att *den tekniska skulden* innebär framtida uppgifter kring utvecklade funktionaliteter, men även brister i dokumentation av spenderad utvecklingstid.

Det antas att fokus på *Upplevd kvalitet* bidrar till bra programvara men enligt teorin är det den *faktiska kvaliteten* som avgör hur bra programvaran egentligen är. Respondenterna gav uttryck för att det är svårt att balansera hur mycket kunderna ska ha att säga till om. De anser att kunderna inte har helhetsperspektivet eller inblick i vad som ger en bra produkt i långa loppet. Respondenterna ser i det sammanhanget till produktens *faktiska* kvalitet.

5.2 Metoddiskussion

Studien är baserad på kvalitativa metoder. Det beror bland annat på att pålitligt kvantitativt data inte var tillgängligt under studiens gång. Kvantitativt data till denna studie kunde ha varit hur många *features* och *user stories* som prioriterades, hur många som togs i mål, hur ofta företaget behövde justera prioriteringen, etcetera. Det var dock inte möjligt för att det kräver en väldigt noggrann dokumentering av dessa händelser, och trots att Monitors dokumentation har varit relativt omfattande ansågs detta inte vara tillräckligt för att använda som underlag till en effektiv och brukbar studie.

Nedan kommer studiens arbetssätt att kortfattat diskuteras. Dessa kommer sedan ligga till grund för forskningsförslag i kapitel 6: Slutsatser.

5.2.1 Reflektion av litteraturstudie, observation och intervjuer

Litteraturstudien har medfört en djupare förståelse av ämnet kravhantering och i synnerhet kravprioritering; den fördjupning studien fokuserar på. Litteraturstudien har upplevts vara den viktigaste grundpelaren till att kunna formulera relevanta intervjufrågor. Den har även underlättat förståelsen för facktermer och områdesspecifika diskussioner vid undersökningens intervjuer och observationer.

Tack vare litteraturstudien upptäcktes flera viktiga nyckelbegrepp att ta med under arbetets gång. Litteraturstudiens upptäckter har gjort att problemformuleringen har omformulerats och finjusterats flera gånger. Det anses dock vara ett tecken på att litteraturstudien bidragit med betydelsefull kunskap.

Det bör dock påpekas att vetenskaplig forskning inom ämnet kravprioritering är ofullständig, på grund av att utvecklingsprocessen inom mjukvaruutveckling har expanderat mycket under det senaste decenniet. Det finns därför hål och brister i den kunskap som i dagsläget är publicerad angående prioriteringsprocesser. Det finns en stor efterfrågan på vidare forskning.

Omfattningen av insamlat observations- och intervjumaterial har varit stort. Det fanns fler möten att delta i än förväntat från början. Vi var inställda på att många av de tillfrågade skulle ge avslag på förfrågningen och garderade oss därför med många intervjufrågor, för att få in tillräckligt med material. Det blev inte fallet.

Observationerna analyserades utifrån ett övergripande perspektiv, men materialet anses varit mer än nog för att kunna analysera detaljerat. En möjlig nackdel med vald undersökningsmetod är så kallad intervjuareffekt, där intervjuaren eller intervjusituationen kan påverka hur respondenterna svarar. Det är även möjligt att vårt deltagande i möten påverkade dessa. Det kan bidra med låg tillförlitlighet av materialet som kommer fram ur observationer och intervjuerna. Det har dock funnits en medvetenhet om och försök att undvika detta under datainhämtningen, och skribenterna har tagit särskilt hänsyn till att vara diskreta undersökare.

En fråga som gällde vad respondenterna trodde skulle förbättra arbetet med kravprioritering har inte tagits med i bearbetningen av intervjumaterial eftersom den inte ansågs bidra till svar på undersökningens frågeställningar.

5.2.2 Reflektion om bearbetning av resultatet

Bearbetning av resultat är en process som är lika iterativ som agil mjukvaruutveckling. Särskilt när det kommer till en stor datamängd som i denna studie är det möjligt att granska och bearbeta materialet på många olika sätt, utifrån många olika synvinklar och i många olika etapper. Bearbetning av resultat är en fas i arbetet som kräver mycket tid och egentligen inte har något slut; det finns alltid mer att upptäcka.

Några saker skulle vi ha kunnat göra annorlunda. Det var utmanande att kategorisera och sortera datamängden utifrån alla olika perspektiv som kunde tas. Datamängden hade med fördel kunnat vara mindre. Eftersom studien har blivit så pass omfattande hade diskussionen utan tvekan kunnat bli dubbelt så stor som den redan är. Vi har behövt välja att fokusera på att besvara frågeställningarna utifrån ett mer översiktligt perspektiv, trots att det finns mer information omkring prioriteringsprocesser som kan diskuteras i detta arbete.

5.2.3 Aspekter på Miljö och Hållbar utveckling

Undersökningen har förhoppningen på att bidra till att effektivisera arbetsprocessen inom mjukvaruutveckling. Detta genomförs genom att ta fram tydliga mål för produkter som ska utvecklas, samt framställa prioriteringar som lägger fokus på produktvärde och tidseffektivisering. Detta förstärker en av FN:s globala mål för hållbar utveckling som beskriver följande delmål 12.6: ”Uppmuntra företag, särskilt stora och multinationella företag, att införa hållbara metoder och integrera hållbarhetsinformation i deras rapporteringscykel” [27]. Som resultat skulle det kunna ge konsumenter högre kvalitet på mjukvaruprodukter. Det innebär bättre kodkvalitet på slutliga produkten och effektiviseringen skulle även kunna bidra till minskat genomsnittlig arbetstid på ett arbetsprojekt, tack vare en pålitlig och fullständig prioriteringsprocess.

Som framkommer av studien är det mycket av det som utvecklas som inte används. Med en bra prioritering ökar chanserna för att utveckla produkter som faktiskt används och bidrar därför till ett bättre resursutnyttjande.

Om personuppgifter används för dokumentationsskäl under arbetsprocessen kommer Dataskyddslagen att följas [28]. Det kan inte garanteras anonymitet utifrån arbetets resultat på grund av att det kan vara möjligt att identifiera en person baserat på informationsbitar, men det som lovas att obehöriga inte kommer få tillgång till personinformationen.

6 Slutsatser

Slutsatser till denna studie har kommit fram ur diskussionsdelen. I denna del redovisas en kort sammanfattning av slutsatserna.

Varför är prioritering inom agil mjukvaruutveckling viktigt?

Slutsatserna kan gälla för agil mjukvaruutveckling då de anledningar som framkom

- För att kunna leverera rätt saker i tid
- För att kunna göra trovärdiga estimeringar
- För att få kundens förtroende i produkten och företaget

Vilka faktorer är värdefulla för prioriteringsprocessen?

Ur resultatet anser vi att går det att ta fram de följande prioriteringsfaktorer som är värdefulla för prioriteringen, med hänsyn till företaget Monitor:

- Kundnytta
- Kundfokus
- Bugghantering
- Förutbestämd ordning (av t.ex. Epics)
- Kapacitet
- Allvarlighetsgrad
- Tillgängliga kompetenser
- Uppgradering/Implementation av funktionaliteter från tidigare system
- Marknadskonkurrens
- Myndighetskrav
- Avtal med kunder
- Företagets mål

Dessa är organiserade enbart utifrån hur ofta de nämndes i intervjuerna av samtliga Team Managers och Business Analysts på företaget Monitor ERP. Det ska därför tas hänsyn till att prioriteringarna sannolikt skulle vara annorlunda beroende på vilket utvecklande företaget som undersöktes, samt vilken produkt som utvecklades. Inga generella slutsatser om andra företag kan dras.

Vilka vedertagna prioriteringsmetoder används av mjukvaruutvecklande företag idag?

Monitor använder i dagsläget inga vedertagna prioriteringsmetoder. De gör ett försök genom att tillämpa MoSCoW och Planning Poker, men dessa har inte observerats som ren användning av metodiken. Walking Skeleton är en möjlig prioriteringsmetod som företaget kan ha intresse i, eftersom Monitor ERP använder utvecklingsfilosofin *Minimum Viable Product*.

Att sätta sig in i en prioriteringsmetod är tidskrävande. Om nyttan med att använda en ska finnas, bör metoden vara så pass bekant och inarbetad att det inte enbart medför extra arbete och otydligheter. Om metoden inte appliceras utförligt förlorar den sitt syfte.

Vilka utmaningar följer med kravprioritering?

Även utmaningar anses skilja sig beroende på företaget och produkten som utvecklas. Återigen är det inte möjligt att dra generella slutsatser, utan gäller dessa utmaningar förföretaget Monitor ERP.

Det identifierades följande utmaningar:

- Begränsade resurser
- Oförväntade och oförutsägbara uppgifter
- Svårigheter inom tidsestimering
- Motsägelsefulla företagsmål som försvårar prioriteringsarbetet
- Balansering mellan kundnytta och kundfokus
- Osynkroniserad utveckling över flera avdelningar

6.1 Fortsatt forskning

Här nämns några forskningsförslag för framtida forskningsstudier inom området kravprioritering.

En av dessa är att undersöka flera team mer djupgående och jämföra hur dessa arbetade i jämförelse med varandra. Det skulle även gärna ha gjorts en undersökning på vilka prioriteringskrav var mer betydelsefulla än andra, men för att kunna dra lyckade slutsatser behövs en ny, djupgående undersökning som analyserar flera av Monitors sprints över en längre period.

Ytterligare ett forskningsförslag är en multikriterieanalys som undersöker vilka konsekvenser en vald prioritering har på utkomsten av en produkt. En multikriterieanalys skulle dock behöva en längre undersökningsperiod.

Referenser

[1] M. L. Drury-Grogan and O. O'Dwyer, "An investigation of the decision-making process in agile teams," *International Journal of Information Technology and Decision Making*, 2013. .

[2] R. S. Pressman and B. R. Maxim, *Software engineering: A practitioner's approach*. 2020.

[3] A. R. Asghar, S. N. Bhatti, A. Tabassum, Z. Sultan, and R. Abbas, "Role of Requirements Elicitation & Prioritization to Optimize Quality in Scrum Agile Development," 2016.

[4] M. Casternfors and M. Christensen, *Product discovery: A no-nonsense guide to building unbeatable products and services*. 2021.

[5] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. R. Mahrin, "A systematic literature review of software requirements prioritization research," *Information and Software Technology*, vol. 56, no. 6. Elsevier B.V., pp. 568–585, Jun. 2014, doi: 10.1016/j.infsof.2014.02.001.

[6] M. Cohn, *Agile Estimating and Planning*. 2014.

[7] H. Schön and F. J. Furrer, "Gute Softwarearchitektur ist Business Value: Ein Ansatz zur Bewertung von SW-Architektur," *Informatik-Spektrum*, vol. 41, no. 4, pp. 240–249, Aug. 2018, doi: 10.1007/s00287-018-1108-z.

[8] E. Mendes, P. Rodriguez, V. Freitas, S. Baker, and M. A. Atoui, "Towards improving decision making and estimating the value of decisions in value-based software engineering: the VALUE framework," *Softw. Qual. J.*, vol. 26, no. 2, pp. 607–656, Jun. 2018, doi: 10.1007/s11219-017-9360-z.

[9] B. W. Boehm, "Value-based software engineering: Overview and agenda," in *Value-Based Software Engineering*, Springer Berlin Heidelberg, 2006, pp. 3–14.

[10] H. Alahyari, R. Berntsson Svensson, and T. Gorschek, "A study of value in agile software development organizations," *J. Syst. Softw.*, vol. 125, pp. 271–288, Mar. 2017, doi: 10.1016/j.jss.2016.12.007.

[11] C. J. Torrecilla-Salinas, J. Sedeño, M. J. Escalona, and M. Mejías, "Estimating, planning and managing Agile Web development projects under a value-based perspective," *Inf. Softw. Technol.*, vol. 61, pp. 124–144, May 2015, doi: 10.1016/j.infsof.2015.01.006.

[12] A. Jarzebowicz and N. Sitko, "Agile requirements prioritization in practice: Results of an industrial survey," *Procedia Computer Science*, vol. 176. pp. 3446–3455, 2020, doi: 10.1016/j.procs.2020.09.052.

[13] H. Töhönen, M. Kauppinen, and T. Männistö, "Evaluating the business value of information technology: Case study on game management system," in *2014 IEEE 22nd International Requirements Engineering Conference, RE 2014 - Proceedings*, Sep. 2014, pp. 283–292, doi: 10.1109/RE.2014.6912270.

[14] N. H. Borhan, H. Zulzalil, S. Hassan, and N. M. Ali, "Requirements prioritization techniques focusing on agile software development: A systematic literature review," *Int. J. Sci. Technol. Res.*, vol. 8, no. 11, pp. 2118–2125, 2019, [Online]. Available: https://www.researchgate.net/publication/337562062_Requirements_Prioritization_Techniques_Focusing_on_Agile_Software_Development_A_Syste

matic_Literature_Review.
[15] S. Hatton, “Choosing the ‘right’ prioritisation method,” *Proc. Aust. Softw. Eng. Conf. ASWEC*, pp. 517–526, 2008, doi: 10.1109/ASWEC.2008.4483241.
[16] S. Kaur, Y. Singh, and N. Kaur, “Applications of Multi-criteria Decision Making in Software Engineering,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 7, 2016, doi: 10.14569/ijacsa.2016.070765.
[17] M. Shameem, R. R. Kumar, C. Kumar, B. Chandra, and A. A. Khan, “Prioritizing challenges of agile process in distributed software development environment using analytic hierarchy process,” *J. Softw. Evol. Process*, vol. 30, no. 11, pp. 1–19, 2018, doi: 10.1002/smr.1979.
[18] K. S. Ahmad, N. Ahmad, H. Tahir, and S. Khan, “Fuzzy-MoSCoW: A fuzzy based MoSCoW method for the prioritization of software requirements,” *2017 Int. Conf. Intell. Comput. Instrum. Control Technol. ICICICT 2017*, vol. 2018-Janua, pp. 433–437, 2018, doi: 10.1109/ICICICT1.2017.8342602.
[19] L. Ibanez, “Productivity: MoSCoW the simple prioritization technique for small products,” *Tech Agile Leaders*, 2020. https://lazaroibanez.com/management-walking-skeleton-the-simple-prioritization-technique-for-mvps-5d99490dad59 (accessed May 23, 2021).
[20] H. Kousar and K. Kumar, “Walking Skeleton Strategy in a Test Driven Development,” *Int. J. Sci. Res. Publ.*, vol. 4, no. 4, pp. 1–6, 2014, doi: 10.1.1.432.9280.
[21] M. Cohn, *User Stories Applied For Agile Software Development*. Addison Wesley Professional, 2004.
[22] J. J. Randolph, “A guide to writing the dissertation literature review,” *Pract. Assessment, Res. Eval.*, vol. 14, no. 13, 2009.
[23] K. Säfsten and M. Gustavsson, *Forskningsmetodik för ingenjörer och andra problemlösare*. 2019.
[24] SCB, *Frågor och svar om frågekonstruktion i enkät- och intervjuundersökningar*. 2014.
[25] A. Hedin and C. Martin, “En liten lathund om,” 1996, [Online]. Available: https://studentportalen.uu.se/uusp-filearea-tool/download.action?nodeId=459535&toolAttachmentId=108197.
[26] D. Wicks, *The Coding Manual for Qualitative Researchers (3rd edition)*, vol. 12, no. 2. 2017.
[27] FN, “Globala målen för hållbar utveckling - Svenska FN-förbundet,” 2019. https://fn.se/globala-malen-for-hallbar-utveckling/ (accessed Mar. 20, 2021).
[28] “Dataskyddsförordningen (GDPR) - Integritetsskyddsmyndigheten.” https://www.imy.se/lagar--regler/dataskyddsforordningen/dataskyddsforordningen---fulltext/#9 (accessed Mar. 19, 2021).

Bilaga A: Intervjufrågor

Följande intervjufrågor användes under intervjuerna med samtliga intressenter:

- I vilka situationer i ditt arbete behöver du prioritera?
- Hur ofta kravprioriterar du i ditt arbete?
- Hur ofta behövs prioriteringen justeras?
- Vad kan omprioriteringen bero på?
- Hur mycket tid lägger du uppskattningsvis på prioriteringsarbetet (per vecka)?
- Finns det några gemensamma mål inom företaget?
- Finns det några gemensamma mål inom avdelningen/teamet?
- Hur avgörande tror du att kravprioriteringen är för att nå företagets mål?
- Vad tar du hänsyn till med dina/era kravprioriteringar?
- Använder du någon/några metoder eller tekniker för att kravprioritera?
- Varför/Hur kom ni fram till att använda dessa metoder/tekniker?
- Hur fungerar prioriteringsarbetet?
- Går det att följa den föreslagna kravprioriteringen?
- Vilka utmaningar upplever du med kravprioriteringsarbetet?
- Hur hanterar du dessa utmaningar?
- Har du några vidare åsikter att dela om kravprioriteringsprocessen?