DAT255 SOFTWARE ENGINEERING PROJECT

VT2018

# REFLEKTIONSRAPPORT

## - VESSEL 1 -


JONATHAN BOMAN
JOHN FRANSSON
OLLE HANSSON
REBECCA HJERTONSSON
JOHAN JOHANSSON
FREDRIK JOSEFSON
EMIL KINDGREN
NIKLAS WESTERBERG

# INTRODUKTION

Denna reflektion tar sin utgångspunkt i det Agila utvecklingsprojekt som genomfördes i kursen DAT255 Software Engineering våren 2018 på Chalmers Tekniska Högskola. I denna rapport har reflektionerna kring projektets framskridande delats upp på 16 delfrågor, som finns angivna på kurshemsidan. Under dessa delfrågor finns en sammanfattning av hur arbetet gått under projektets sprintar samt en beskrivning av nuläget, saker vi tar med oss till framtida projekt samt hur dessa skall implementeras. Viktigt att ha i åtanke är att svaren i rapporten är starkt hopknutna med varandra och att en del av reflektionerna går in i varandra. Därför bör ej rapporten läsas enbart utifrån en enstaka del utan snarare bör läsaren se den utifrån ett större och övergripande perspektiv för att reflektionen skall skapa en rättvist bild som läsaren kan förstå.

Som en introduktion till vår reflektion vill vi belysa hur lärorikt det har varit att genomföra ett Agilt projekt för första gången då ingen av projektgruppens deltagare sedan innan hade erfarenhet av detta. Att få testa på att jobba in en situation karaktäriserad av stora osäkerheter där gruppen successivt arbetat för att minska dessa i syfte att i slutändan kunna leverera värde, har stundtals varit både väldigt jobbigt och frustrerande. Det har också hjälp oss att växa som problemlösande individer då tvingats jobba trots att alla dessa osäkerheter har existerat i den omgivande miljön. Det finns givetvis många olika erfarenheter som kan sägas stamma från detta projekt men framförallt vill vi belysa hur väl det hjälpt oss inse vikten av att ständigt utvärdera och reflektera kring arbetets utfall i syfte att förbättra framtida projekt och därför vill vi inleda denna rapport med nedanstående citat.

*“Self-Reflection is the School of Wisdom”*

(Baltasar Gracian, 1640)

# INNEHÅLLSFÖRTECKNING

# 1 The Chosen Scope of the Application

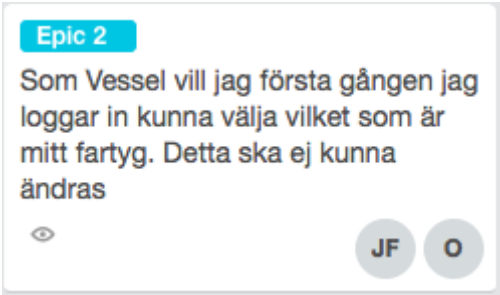
## 1.1 Sammanfattning av tidigare sprintar

Vid projektets uppstart hade gruppen höga ambitioner kring vad som skulle åstadkommas i form av nytt värde och nya funktioner i appen. Efter de första introduktionsveckorna blev det snabbt klart att det fanns tekniska svårigheter som påverkade möjligheterna för gruppen att nå de mål som sattes initialt. När vi blev tilldelade en slutanvändare och en produktägare startade arbetet med att i samråd med dessa ta fram user stories för funktionalitet i appen. Prioritering av user stories började även sättas upp. Fram till sprint 3 hade gruppen fortfarande en del tekniska problem med att få igång appen och förstå hur utvecklingsmiljön fungerade. Därför arbetades det under mitten på kursen hela tiden med att förstå strukturen på appen och JavaScript för att kunna implementera de user stories som satts upp.

Ett annat misstag som gjordes gällande scopet var att då gruppen hade stora problem med att få igång utvecklingsmiljön lyckades medlemmarna inte klara av de tänkta uppgifterna för de olika sprintarna. Detta resulterade i att arbetsmiljön försämrades och den generella stressnivån bland gruppmedlemmarna ökade. Då gruppen låg efter i sin planering ökade stressen kontinuerligt och trots att vi sedan tidigare veckor inte hunnit med scopet på grund av tidsbrist trappades detta upp ytterligare då vi kände att vi både behövde ta igen det vi missat i tidigare sprintar men även börja leverera värde till kund. Detta ökade scope ledde i sin tur att vi återigen misslyckades med att klara av att färdigställa alla stories i sprint loggen. Alltså uppkom i början av projektet, en ond spiral med ständigt accelererande storlek på scope och stressnivå. Genom att istället, i samråd med produktägare, välja ett smalare scope kunde denna spiral brytas.

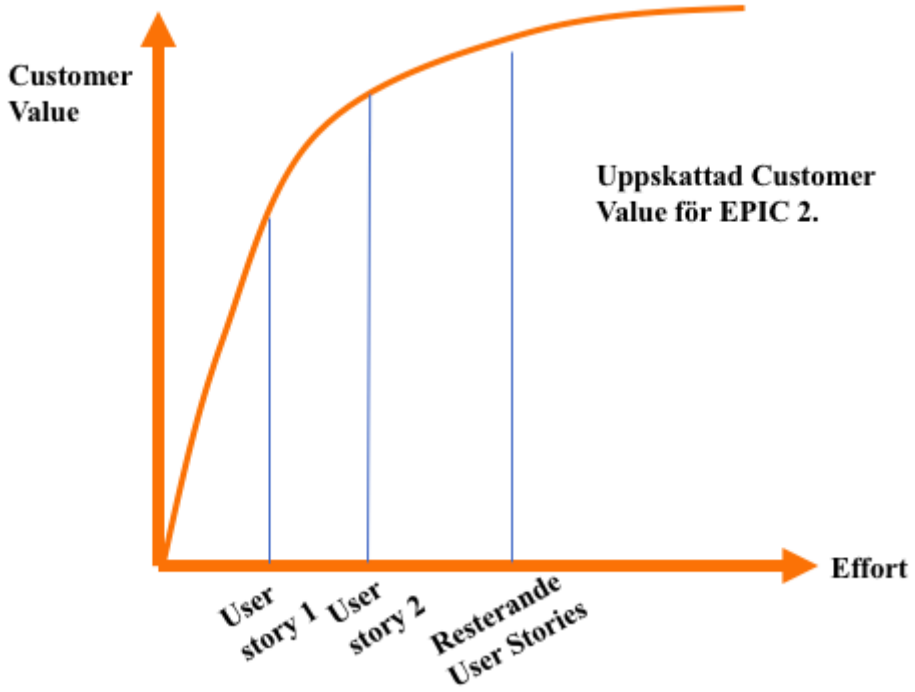
Gruppen började även sätta upp user stories gällande vad en medlem skulle vilja lära sig. Exempelvis “As a team member I would like to be able to push to git”. Syftet med dessa user stories var för medlemmarna att erhålla tillräcklig kunskap för att i slutändan fortsatt kunna skapa värde för produktägare och slutanvändare.

Under projektets gång skalade gruppen ner scopet för appen från att ha fokus på mycket ny funktionalitet till att fokusera på att skapa användarvänlighet samt ett par nya funktioner. I enlighet med föreläsning L7-MVP och konsultation med Håkan drogs även slutsatser om att det är mycket bättre att leverera något konkret, oavsett ifall det initialt kan ses som obetydligt. Detta dels för att öka självförtroendet i gruppen, då gruppen känner att de faktiskt har levererat något konkret, men även för att relativt enkla och små user stories kan leverera stort värde för slutkunden. Exempel på detta är Epic 2 vilket ses i figur 1.




*Figur 1: Beskrivning av Epic 2*

I Epic 2 fanns en user story som handlade om att låsa användaren från att byta skepp efter de har valt. Istället för att lova produktägaren att hela EPIC:en skulle vara klar under en sprint sattes fokus på att endast klara av tidigare nämnd user story. Detta var något som var relativt enkelt och smalt, men genererade stort värde för slutkunden och sattes då som fokus under en sprint, vilket ses i figur 2.




*Figur 2: Uppskattat värde minskar efter att den mest grundläggande funktionaliteten har lagts till. User Story 1 och 2 representerar de mest essentiella, medan resterande user stories representerar mer användarvänlighet och mindre justeringar.*

Mot slutet förbättrades gruppens kunskaper om utvecklingsmiljön och gruppen kunde mer effektivt tillföra ny funktionalitet som efterfrågas av produktägaren och slutanvändaren. Gruppen hade därmed kommit över en liten tröskel vilket ledde till större förståelse för de egna begränsningarna. Detta ledde till att scopet fortsatte skalas ner och prioriteringsordningen som satts upp av produktägaren användes som mall för att stegvis leverera lite i taget av det som efterfrågades av slutanvändare och produktägaren.

Ett misstag som gjordes under vissa sprinter var även att inte vara helt realistiska gällande hur mycket tid som skulle läggas, inte bara hur mycket vi kunde göra. Vissa veckor uppskattade vi att vår tillgängliga tid per person var 20 timmar trots att sprinten innehöll röda dagar och att merparten av gruppen visst att de skulle ägna stor del av sin tid på att skriva kandidatarbete.

## 1.2 Var är vi?

I dagsläget har gruppen bra förståelse för sin egna kapacitet. Det är väldigt lätt att falla för ett för stort scope då nya uppgifter oftast ser lättare ut på ytan än vad de faktiskt är. Genom ett för stort scope riskerar gruppmiljön att försämras, främst på grund av stressen då gruppen ej klarar att lösa alla de uppgifter som gruppen åtagit sig att göra. Genom att istället omvänt använda ett lite för smalt scope kommer gruppmedlemmarna erhålla en positiv känsla av att ha klarat av det man ska, och i vissa fall till och med mer. Genom att gå in med inställningen att ha ett smalare scope är man även mer benägen att förhandla med produktägaren gällande vad de vill ha gjort.

Lovar man ett för stort scope kommer produktägaren förvänta sig att man levererar detta och bli besviken när det ej går, medan med ett för litet scope kommer utvecklingsgruppen enklare kunna överträffa produktägarens förväntningar. Trots att slutresultaten kanske är desamma, kommer man genom att initialt ha ett litet scope som man sedan trappar upp ha en nöjdare kund i jämförelse om man startar med ett brett scope och sedan smalnar av.

Genom ett smalt scope kommer även de mest essentiella user stories kunna prioriteras och maximalt värde i förhållande till effort kommer enklare kunna levereras. Som beskrivet i figur 2 tenderar customer value att avta ju mer detaljerade user stories blir. Det är även dessa detaljrika user stories som fokuserar på “finputs” som oftast tar mer effort.

Gällande det nuvarande projektet och den nuvarande utvecklingsmiljön har gruppen bra insikt i vad de är kapabla till och kan på så sätt mer pricksäkert uppskatta ett korrekt scope. Detta diskuteras djupare under frågan om user stories där vi går djupare in på ämnet om velocity och effort.

## 1.3 Hur kommer det se ut nästa gång?

Nästa projekt kommer gruppen att undvika ett för stort scope, och undvika att hamna i situationer där orimligt mycket funktionalitet lovas till produktägaren som sedan inte implementeras. Trots att vi haft väldigt förstående produktägare som insett att vi gjorde misstaget att i början lova för mycket och sedan accepterade ett avsmalnande av scopet, är det inte alls säkert att detta är fallet nästa gång.

Nästa gång kommer gruppen inte bara vara mer realistisk gällande hur mycket tidigare kunskap som finns inom området och vad medlemmarna är kapabla till att genomföra, men även hur mycket tid som kommer kunna läggas. Som nämnt ovan uppskattades det att kunna lägga 20h på vissa sprinter, när detta i verkligheten inte var realistiskt.

## 1.4 Hur åstadkommer vi detta?

Externa effekter beaktas i större bemärkelser när man planerar scopet och hur mycket tid som skall läggas på varje sprint.

Ett smalt scope kommer väljas innan gruppen har fått tillräckligt stor kunskap kring hur de tillgängliga verktygen fungerar. Först när gruppen har erhållit bred kunskap kommer scopet kunna breddas och mer funktionalitet kommer kunna lovas till produktägaren. En mer genomtänkt prioritetslista kommer även göras tidigt, där de initiala och smala user stories som genererar stort värde för slutkund och produktägaren kommer prioriteras.

Genom att starta med ett initialt smalt scope kommer gruppen undvika den stressituation som beskrevs i sammanfattningen.

# 2 Social Contract

## 2.1 Sammanfattning av tidigare sprintar

Under den första veckan sattes vårt sociala kontrakt upp med fokus på upplägg kring mötestider och arbetsprocess. Dessutom definierades roller för sprintarna tidigt i projektet vilka dock senare fick revideras när gruppen märkte vad som egentligen behövdes för roller för att styra projektet i rätt riktning under varje sprint. Det förekom en del avvikelser från kontraktet under projektets gång. Som exempel på detta så hittade vi roller i litteratur som vi definierade i kontraktet,dock förstod inte gruppen värdet av att använda dessa roller till en början vilket gjorde att kontraktet inte följdes ur detta avseende. I slutskedet av projektet ändrades dock rollerna än en gång efter vår egen efterfrågan vilket gjorde att rollerna faktiskt användes på ett riktigt sätt. Dessutom förekom avvikelser från tiderna som sattes upp till följd av ledighet och på slutet ett naturligt fokus på kandidatarbetet innan dess slutgiltiga deadline.

Det sociala kontraktet revideras två gånger under projektets gång. Den första förändringen skedde i samråd med Jan-Philipp och ändrade start och slut för sprintarna från måndag-måndag till onsdag-onsdag. Detta gjordes för att anpassa sprintarna efter konsultationerna som låg under onsdag eftermiddag så att mötena med produktägare, slutanvändare och kursansvariga kunde användas på ett bättre sätt för att planera kommande sprint. Den andra revideringen skedde inför den näst sista sprinten och innebar en tydligare struktur med fler definierade roller. Behovet av detta uppstod när programmeringen kom igång ordentligt och arbetet med att implementera user stories som direkt skapar värde för slutkund.

## 2.2 Var är vi?

I dagsläget används ett bra och stabilt socialt kontrakt med tydligt definierade roller. Detta gör det enklare för gruppmedlemmar att veta vad deras största ansvarsområde är, och vad de skall fokusera sin tid på.

Gruppen har även insett vikten av att ha ett bra socialt kontrakt som man alltid kan luta sig tillbaka på. Detta är ett av de första projekten ett socialt kontrakt har använts. Dels för att gruppen består av så många medlemmar utan en tydlig ledare, men även för att vid arbete med SCRUM behöver gruppmedlemmarna träffas ofta. Det sociala kontraktet har agerat lite som en “objektiv lagbok” och det har hjälpt att minska onödiga diskussioner som kan tänkas uppstå. Detta har gjort gruppen mer effektiv under möten och minskat “startsträckan” som ibland kan uppstå utan ett socialt kontrakt.

I dagsläget har en stor del av det sociala kontraktet varit byggt på tillit och respekt, vilket har fungerat mycket väl inom gruppen. Detta beror på att gruppmedlemmarna tidigt gick ihop och konstaterade att inga “straff” skulle vara aktuella, utan alla litade på att de ingående medlemmarna skulle bidra.

## 2.3 Hur kommer det se ut nästa gång?

Vi vill på grund av ovanstående undvika roller utan tydligt syfte i framtiden. I senare projekt vill vi även att det sociala kontraktet ska vara hållbart under projektets gång och att grundprinciper, som exempelvis regler kring arbetsprocessen, inte behöver ändras under projektets gång. Självklart skall det finnas möjlighet att ändra vissa aspekter av det om behov uppstår för att kunna anpassa rollerna utifrån projektets framskridande.

Om nästa projekt visar sig vara större kommer det finnas möjligheter för att utöka det sociala kontraktet, genom att exempelvis bygga på med fler specifika roller. Med ett större projekt alternativt större grupp kommer ett striktare kontrakt vara nödvändigt.

I ett framtida projekt kommer det sociala kontraktet att skrivas med mer långsiktighet i åtanke kring mötesstrukturen för att anpassa sig till externa faktorer samt alla gruppmedlemmars olika scheman. Detta för att minska risken att vissa medlemmar inte kan närvara under möte.

## 2.4 Hur åstadkommer vi detta?

För att skapa ett hållbart socialt kontrakt bör gruppen i framtida projekt redan under uppstartsfasen genomföra en omfattande diskussion kring vad det sociala kontraktet bör innehålla för regler och roller vilket inte gjordes i detta projekt.

I framtiden kommer vi inte tillsätta roller där gruppen inte har formulerat ett tydligt syfte bakom rollen. Detta för att säkerställa att rollerna kommer efterlevas och genom de uppfylla ett effektivt arbetssätt.

Genom att använda tidigare kunskaper, exempelvis kring vilka roller som bör användas kommer även gruppmedlemmarna kunna ta med sig och implementera direkt. Ett exempel på detta är Trello-ansvarig - då gruppen inte ens var medvetna om att Trello var ett bra verktyg infördes inte rollen förrän ganska sent in i projektet. Med kunskapen medlemmarna har nu kommer denna rollen kunna tilldelas direkt under uppstartsveckan och ett mer strukturerat arbete kommer genomföras genomgående under hela projektet.

Redan i uppstartsfasen kommer en plan att läggas för hela projekttiden och mötestider kommer att planeras in utefter de förutsättningar som finns på förhand. Vid särskilda omständigheter kommer detta kunna ändras men eftersom det planeras så tidigt är förhoppningen att gruppmedlemmarnas eventuella andra åtaganden får planeras runt de tider som satts i projektet.

# 3 The Success Criteria for the Team

## 3.1 Sammanfattning av tidigare sprintar

Tidigt i projektet definierades framgång som att leverera enligt förväntningar från produktägaren. Detta sattes upp med förbehållning att det var mycket viktigt att balansera förväntningarna under projektet så att dessa var rimliga med avseende på vilka förkunskaper projektgruppen hade samt den tidsram som fanns för kursen.

Efter några veckor förändrades hur gruppen såg på framgång i takt med att de tekniska problemen med appen uppenbarade sig. Under de första sprintarna sattes därför kriterierna för framgång till att endast få igång existerande applikation genom att lösa de tekniska problemen. Veckorna som följde insåg gruppen att de hade missuppfattat hur success criteria skulle definieras. Efter att ha fått lite vägledning skrevs success criteria om mot slutet av projektet. Då definierades kriterierna om för att innefatta bland annat att produktägaren var nöjd med det värde som skapats och att all kod som pushats skall hålla god kvalitet.

## 3.2 Var är vi?

Till slut blev våra success criteria:

- Vi vill ha förståelse för hur Agil utveckling fungerar samt få kunskap om hur vi i framtida projekt ska kunna tillämpa “SCRUM-metodiken”.
- Vi vill skapa användarvänlighet för slutanvändaren till appen, i vårt fall sjökaptenen
- Vi vill skapa värde för produktägaren.
- Vi vill att koden vi har pushat har en god kvalitet.

Vad “värde” är har även det förändrats genom arbetets gång. När vi insåg att våra programmeringskunskaper kanske inte skulle räcka till för att implementera allt värdeadderande vi hade diskuterat med produktägare och slutanvändare bestämde vi oss i samråd med dessa att konceptuellt visa upp dessa funktioner i form av klickbara s.k. wireframes istället. Därför anser vi att vi lyckats att skapa värde för produktägare och slutanvändare. De funktioner vi faktiskt implementerade, som inloggningsfunktionen, tillförde också användarvänlighet till slutanvändaren. Att vi hade skapat värde och användarvänlighet stod klart när vi på demon onsdag 23/5 fick positiv respons från inblandade aktörer både på implementationer och modeller.

Att förstå hur vi på olika sätt kan addera värde för olika aktörer och att detta värde kan vara av olika sorter var för oss en stor del i att förstå hur Agil utveckling fungerar. Därför anser vi också att vi har förstått hur Agil utveckling fungerar och tror att vi kommer att kunna tillämpa metodiken i framtida projekt. Mer om hur vi har lärt oss om Agil utveckling går att läsa under frågan gällande just agile practices. Vi anser att koden vi har pushat håller en god kvalitet och detta kommer vi diskutera mer under avsnittet om kodkvalitet.

## 3.3 Hur kommer det se ut nästa gång?

En tydligare bild av hur värde skapas kommer tas fram och ligga till grund för att ha ett fortsatt fokus på att skapa värde för produktägare och slutanvändare, men även oss själva.

Viktigt att komma ihåg är också att det interna värdet kan vara av stor vikt, beroende på projekt. Exempelvis hade inte lika mycket vikt behövt läggas vid att lära sig utvecklingsmiljö och språk om projektet omfattat områden där gruppen har större förkunskaper.

Kriteriet om Agil utveckling kommer i princip falla bort i framtida projekt då vi anser att vi kommer bära vidare de kunskaper vi har lärt oss under detta projekt och således inte behöver lära oss dem på nytt. Att fräscha upp dem kan såklart vara av värde men att ha det som ett success criteria anses vi inte behövs.

Kodkvaliteten kommer vara av stor vikt för oss i framtida projekt. Får vi möjlighet att börja mer från grunden med en app eller liknande kan vi ha större kontroll och ansvar över koden i sin helhet istället för att bygga vidare på en stor kodbas innehållandes flertalet allvarliga varningar, enligt våra tester. Success criteria gällande kodkvalitet kan i framtida projekt finnas kvar men innebörden blir större om vi får ett mer övergripande ansvar för att bygga upp hela kodbasen.

## 3.4 Hur åstadkommer vi detta?

För att skapa oss en tydlig bild av hur värde skapas i ett specifikt projekt kommer en dialog tidigt att föras med slutanvändare och produktägare. Dialogen kommer förhoppningsvis att kunna föras på ett närmare plan i kommande projekt om vi inte är en av många grupper som tävlar om deras tid.

Internt värde kan skapas genom att exempelvis avsätta tid för personlig utveckling genom att gruppmedlemmarna får chans att utöka sina kunskaper i de bakomliggande principerna eller relevanta tekniska områden. Detta skapar då indirekt värde för produktägaren.

Genom att fortsätta som vi slutade detta projekt, med kodtester som inte bara testar för buggar utan även eventuella brister i koden, kan vi säkerställa att vi uppfyller kriteriet även i framtiden. Vi vill även hitta strukturerade sätt och metoder för att testa de övriga kriterierna för kodkvalitet. Precis som i detta projekt kan ett bra tillvägagångssätt för att hitta dessa metoder vara att fråga mer erfarna personer både med hjälp av internet och eventuella bekantskaper. Att ha en extern part som kontrollerar exempelvis läsbarheten kan underlätta framtida påbyggnad av applikationen då nya utvecklare ska sätta sig in i koden.

Vid ett större ansvar kan vi även välja att inkludera andra faktorer i begreppet kodkvalitet för att täcka in t.ex. tidseffektivitet och struktur och läsbarhet. Även detta går att läsa mer om i avsnittet om kodkvalitet.

# 4 Acceptance Tests

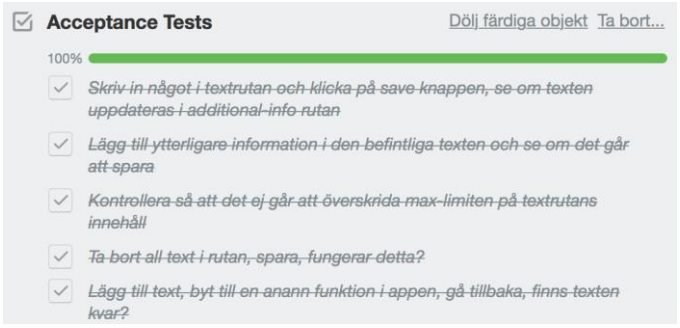
## 4.1 Sammanfattning av tidigare sprintar

Under de två inledande veckorna sattes inga acceptance tests då arbetet med applikationen ännu inte kommit igång. Efter det valde gruppen att ha en fungerande app som kunde ändras i genom att ändra kod och se så att allt ändrades genom expo. Dessa test var i form av att exempelvis ändra namn på en flik och se så att det ändrades i appen.

Inför den näst sista sprinten fick gruppen hjälp med hur acceptance tests skall utformas. Det stod klart för oss att det funnits missuppfattningar kring hur detta skulle göras precis som med success criteria. I samråd med Pontus (IT-support) sattes kriterier upp för varje task tillhörande en user story samt test definierades för att se huruvida kriterierna var uppfyllda.

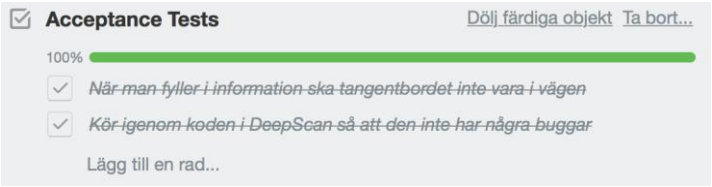
## 4.2 Var är vi?

Vi har nu i slutet givit varje user story mer genomgående acceptanstest. Nedan följer ett exempel på en mer utförlig lista som vi har testat att funktionen lever upp till de satta acceptanskriterierna. Vi utformade dessa tester för att simulera en användares interaktion med funktionen och därigenom hitta brister som annars inte hade upptäckts förrän efter release. Vi har i slutet på projektet blivit mycket bättre på att utforma acceptanskriterier som täcker de efterfrågade egenskaperna men vi har fortfarande problem med att kunna utforma dessa så att de är heltäckande.




*Figur 3: Exempel på Acceptans test*

Efter att vi hade förstått hur vi skulle arbeta med DeepScan lade vi även till det testet som ett kriterium för att mer kontinuerligt jobba med kodkvalitet ur ett med tekniskt perspektiv. Nedan visas ett exempel för en liten tilläggsfunktion som inte krävde fler tester än de två uppskrivna.




*Figur 4: Exempel på Acceptans test*

Användartesterna skiljer sig från tester på teknisk kodkvalitet eftersom en features funktion kan skilja sig mycket från en annan och därför är det svårt att utforma enhetliga tester som kan appliceras generellt. En sak som framkom under näst sista sprinten och även under demonstrationen 23/5 var att de konceptuella modellerna som visades genererade mycket input och frågor från intressenterna.

## 4.3 Hur kommer det se ut nästa gång?

I nästa Agila projekt kommer vi att fortsätta som vi slutade. I just denna aspekt känner vi nämligen att vi hittade en modell som passar oss bra. Får vi in rutinen att testa koden både ur ett användar- och tekniskt perspektiv redan från början av projektet underlättar detta för alla inblandade.

Teknisk kodkvalitet är relativt sett enklare att testa och få ett konkret resultat på. Detta eftersom t.ex. DeepScan är specialiserat för ett visst syfte och bearbetar koden på samma sätt oavsett vilken typ av funktion det gäller eller vad koden har för funktion. Testet måste alltså inte utformas annorlunda beroende på vilken task man testar.

Nästa gång kommer fokus även ligga på att utforma acceptanstesterna på sådant sätt att de är heltäckande och verkligen kan säkerställa att all önskad funktionalitet finns och att de lever upp till den satta kvalitetsgränsen.

I framtiden vill vi också få en bättre förståelse av användarens åsikter och krav.

## 4.4 Hur åstadkommer vi detta?

Vid nästa eventuella utvecklingsprojekt hade vi för fler funktioner först skapat modeller för att snabbare utvärdera dem med produktägare och slutanvändare för att ta fram lämpliga acceptance tests. Frågor som “varför ligger inte knappen där?” kan hjälpa oss att utforma kravspecifikationer och undvika missförstånd. En nackdel kan var att man i en modell kan visa på funktionalitet som gruppen inte besitter programmeringskunskapen för att senare implementera, vilket kan leda till att man missar sitt tänkta scope. En bra förståelse för utvecklingsmiljön och problemet borde hjälpa till med att hantera detta problem.

För att uppnå bättre bild av användarens åsikter och krav samt även utföra bättre acceptance tests inför implementationen av funktionerna kan man i framtiden använda sig utav konceptuella modeller i framtiden i ännu större utsträckning för att kunna samla in och ge en bättre bild över kravspecifikationen.

För att säkerställa att acceptanstesterna är heltäckande kommer de att utformas dels i samråd med kund men även genom att låta BETA-användare få testa appen och därmed få input på möjliga buggar och fel som måste hanteras med hjälp av acceptanskriterier.

# 5 The Design of Your Application

## 5.1 Sammanfattning av tidigare sprintar

Under större delen av projektet lades lite vikt vid design och val av API: er eller designmönster. Detta för att ingen större konkret funktionalitet lades till. Under näst sista sprinten reflekteras det över hur detta påverkar arbetssättet vid implementering av ny funktionalitet för första gången.

React Native har använts av det naturliga skälet att det använts för att utveckla den existerande appen. Nya tillägg laddades ner under senare delen av projektet för att kunna utveckla de funktioner som lagts till i appen under projektet. Exempel på detta var en Keyboard Aware Scroll View som automatiskt kunde justera sig till de implementerade funktionerna.

Det API som använts är PortCDM och vid implementation av nya funktioner har inget aktivt val av designmönster gjorts. Anledningen till detta är att funktionerna som lagts till kan ses som tillägg eller utökning av tidigare funktionalitet snarare än något nytt från grunden.

Fokus under projektets gång har varit att skapa värde för produktägare och slutkund. En stor del av det som efterfrågats av bägge dessa parter är konkreta idéer på ny funktionalitet samt hur denna kan tänkas införas, snarare än produktionskod. På grund av detta låg fokuset i stor del på att programmera väldigt enkel kod som kanske inte är den mest effektiva, i främsta syfte att demonstrera funktionalitet för kunderna. Denna avvägning gjordes för att kunna demonstrera mer funktionalitet och i samråd med produktägaren ansågs detta vara mer väsentligt än att producera mer tidseffektiv och stabil produktionskod gällande färre funktioner.

Ett försök till att lagra data i databasen gjordes, men då konkret API till detta saknades gjordes avvägning att istället lagra data lokalt. Detta ansågs vara okej då fokus skulle ligga på att demonstrera funktionalitet.

Gruppen hade många idéer på förbättringsområden som var utanför det möjliga scopet samt gruppens kompetens. Under diskussion med produktägaren framkom det att det fanns stora önskemål att få konceptuella modeller på hur dessa funktioner skulle kunna tänkas se ut, varvid detta skapades som komplement till den kod gruppen hade skapat.

Största fokus på den funktionalitet som försökt införas har legat på att förbättra användarvänligheten. Detta ansågs i samråd med produktägaren och slutanvändaren vara ett av applikationens största problem. Det fanns redan väldigt mycket funktionalitet i appen, den presenterades bara inte på rätt sätt. Ett exempel på detta var vid inloggningen - istället för att välja vilken vessel man skulle vara var sjökaptenen tvungen att manuellt välja ett portcall för att sedan kunna agera som just denna vessel. För att åtgärda detta infördes ny funktionalitet där man direkt efter inloggning kunde söka på sitt specifika skepp och sedan välja det, för att sedan “låsa fast” detta val och förhindra sjökaptenen från att kunna byta skepp.

## 5.2 Var är vi?

I dagsläget fokuserar gruppen mestadels på att omstrukturera existerande funktionalitet i syfte att skapa användarvänlighet. Koden som skrivs är i största del enkel, då syftet endast är att kunna demonstrera gruppens tankar till slutanvändaren och produktägaren. Detta på grund av att det är vad gruppen kom överens om med de båda parterna. Vi använder oss idag enbart utav PortableCDM som API och Redux som architecture pattern då vi inte haft behov att implementera  ytterligare.

## 5.3 Hur kommer det se ut nästa gång?

Beroende på vad som kommer efterfrågas i ett potentiellt nytt projekt kommer designen av applikationen variera. Kommer nästa projekt vara väldigt fokuserat på att skapa ny konkret funktionalitet med högkvalitativ produktionskod, som kommer användas i praktiken, kommer ett nytt angreppssätt behöva tillämpas där större fokus ligger på stabil och tidseffektiv kod.

Det största gruppen kan ta med sig gällande detta är att försöka vara flexibel vid designen och anpassa sig efter projektets utformning och vad produktägare samt slutkund efterfrågar. Genom att inte låsa in sig på ett visst designsätt kommer gruppen kunna tackla fler problemtyper och ta sig an fler projekt. Det är även viktigt att väga produktägarens krav mot slutkundens för att finna den funktionalitet som skapar störst absolut värde.

Om nästa projekt är av större karaktär tror vi att det är viktigt att projektets komponenter hänger samman på ett strukturerat sätt för att undvika att produktens tekniska plattform blir allt för spretig.

## 5.4 Hur åstadkommer vi detta?

Genom en öppen dialog tidigt med slutanvändare och produktägare där de olika parternas förväntningar konkretiseras kommer gruppen tydligt kunna välja vilken typ av approach de vill ha vid val av design. Genom att försöka vara flexibla och kreativa när medlemmar stöter på problem kommer dessa kunna kringgås. Exempelvis som när gruppen valde att spara data lokalt istället för till databasen då API: er saknades. Genom kreativ utnyttjande av Google samt IT-support kommer tillräcklig kunskap inom problemområden kunna erhållas och eventuella problem lösas.

För att se till att komponenterna i ett större projekt hänger samma anser vi att man bör använda sig utav lämpliga design principer eller mönster för att på så sätt styra gruppens medlemmar att ha enhetlig kod och karaktär på koden som skapas. Vi anser ytterligare att ett sätt att få en enhetlig struktur är att använda sig utav att givet bibliotek och noga välja vilka komponenter som bör adderas till projektet.

# 6 The Behavioural Overview of Our Application

## 6.1 Sammanfattning av tidigare sprintar

Denna punkt var svår att utvärdera under större delen av projektet på grund av bristfällig kunskap och erfarenhet i gruppen. Inför näst sista sprinten stod det klart att vi kunde ha dokumenterat mer av det vi faktiskt gjort vid implementering av user stories för att på ett enkelt sätt kunna dela kunskap inom vårt team men också undvika dubbelarbete. Vi använde oss inte av diagram eller liknande utan mycket kommunikation skedde istället inom teamet muntligt eller genom slack för att dela insikter och kunskap. Detta kunde ha gjorts på ett mer strukturerat sätt för att säkerställa kvalitet och kontinuitet.

Den dynamiska funktionaliteten har vi satt oss in i inför varje user-story genom att använda oss av PortCDM appen för att lokalisera vart våra nya funktioner bäst passar in. Vi har letat upp denna del av koden och härifrån har vi studerat relationerna genom att backtrack:a kring vart information hämtas/lämnas. Genom detta arbetssätt har vi kontinuerligt byggt upp mer förståelse för hur appens funktioner hänger ihop. Detta har dock som sagt inte dokumenterats eller ritats upp, utan varje enskild gruppmedlem har varje gång fått sätta sig in i den befintliga funktionaliteten. Efter diskussion med Håkan förstod vi att det kan finnas många fördelar med att dokumentera detta och har därför som mål att göra detta framöver.

## 6.2 Var är vi?

I dagsläget har gruppen inte en fullständig överblick över hur interaktion sker mellan olika delar av systemet. Detta tror vi beror till stor del på att vi under första halvan av projektet hade mycket tekniska problem. Därför upplevde gruppen en känsla av stress kring att vi låg efter när det kom till att leverera värde till produktägaren och slutanvändaren. När vi väl kom igång såg därför gruppen det som högre prioritet att direkt försöka skapa nytt värde. Först i efterhand har vi insett att vi förmodligen hade kunnat generera mer värde i slutändan genom att genomföra dokumentation och bygga upp en strukturerad kunskapsbas som kan användas av alla medlemmar för att implementera ny funktionalitet.

Specifika interaktioner har gruppen fått viss insikt i genom att undersöka olika områden inför införandet av ny funktionalitet. Denna information och kunskap dokumenteras inte på ett strukturerat sätt utan delas i dagsläget muntligt eller via informella kanaler. Detta leder potentiellt till dubbelarbete och problem med att uppskatta effort på olika tasks på grund av bristande insikt i hur olika komponenter hänger samman vilket påverkar effektiviteten negativt.

Ett exempel på hur vi jobbade i projektet gällande den dynamiska aspekten, var när vi skulle lägga till ny funktionalitet för att ”välja vessel”. Vi behövde då hämta ned möjliga Vessels från servern/backend och därför tittade vi på tidigare funktioner i appen för att få inspiration. Så om en annan medlem i gruppen skulle arbeta med en liknande funktion frågade denna de som arbetat med funktionen för att välja Vessel. Alltså så har vi delat information om den dynamiska funktionaliteten utan att reflektera över det formellt.

Vi kan konstatera att det inte riktigt håller att endast använda muntlig kommunikation för att dela dynamisk information mellan varandra utan det blir mycket tydligare om man kan dela informationen på annat vis, exempelvis genom att använda relevanta diagram eller modeller.

## 6.3 Hur kommer det se ut nästa gång?

Vi vill i kommande projekt ha en bättre bild av den dynamiska funktionaliteten och hur informationsflöden ter sig mellan olika delar av en applikation. Detta kommer att leda till en bättre förståelse för hur ny funktionalitet skall implementeras och därmed öka produktiviteten. Beroende på kommande projekts utformning kommer olika sätt att dokumentera eventuellt behöva användas för att situationsanpassa.

En annan fördel med att tidigare få en bättre överblick är att det kommer minska startsträckan innan kod kan börja levereras. Om vi tidigt dokumenterar interaktionen mellan olika delar av systemet blir det betydligt enklare att utifrån detta utforma konkreta user stories för ny funktionalitet. För oss tog det ca. 4 sprinter innan vi kunde införa konkret funktionalitet. Genom att använda oss utav exempelvis interaktionsdiagram hade detta kunnat gå snabbare, då vi minskar dubbelarbete samt att alla team medlemmar hade haft samma förståelse för den befintliga koden och då på ett enklare sätt kunnat jobba i samma riktning.

## 6.4 Hur åstadkommer vi detta?

I kommande projekt vill vi från början dokumentera befintlig interaktion i programvaran och se till att alla gruppmedlemmar får samma information. Efter det kommer vi att för varje ny funktion som skapas att genom diagram visa på vilken typ av information som skickas vart vid en händelse i appen och i vilken ordning denna information skickas mellan olika händelser.

Vi kommer också att använda oss utav use case diagram då vi tror att detta skulle kunna hjälpa oss att få en bättre blick och uppfattning av hur användarna interagerar med appen. Detta gör att vi skulle kunna få en bättre överblick av vad systemet gör eller skall göra. Detta kan dessutom göra det lättare för gruppen att matcha produktägarnas och eller end-users förväntningar och krav på produkten då use case diagrams möjliggör en högre grad av översikt av produkten.

Ett sätt att åstadkomma detta är att i början av ett projekt se över hur information hämtas/lämnas genom att backtrack:a all funktionalitet och skapa ett interaktionsdiagram över hur funktionaliteten hänger samman. Har du mycket kod att jobba med kan detta vara väldigt tidskrävande och därför kan det vara lämpligare att dokumentera löpande och samla informationen genom olika interaktionsdiagram eller dokumentera skriftligt.

# 7 The Structural Overview of Our Application

## 7.1 Sammanfattning av tidigare sprintar

Likt arbetet med behavioural overview har vi inte varit duktiga på att utvärdera den statiska strukturen i appen under arbetets gång. Den kommunikation som skett inom ämnet har varit muntligt eller sporadiskt i skrift. Arbetssättet kring den statiska strukturen har inletts med att först lokalisera var i appen en funktion bör placeras. Valet av denna plats har gjorts både med hänsyn till hur existerande kod har strukturerats men även till vad den aktuella user-storyn har för krav och påverkan på appen. Sedan har vi letat upp denna del av koden och härifrån har vi studerat vilka klasser/funktioner som kan vara relevanta att använda sig av eller modifiera. Gruppen anser att detta har varit ett bra tillvägagångssätt då det möjliggjort ett effektivt och snabbt sätt att lära känna appen och hur den statiska funktionaliteten är uppbyggd. Det har även varit passande för oss då vi inte har så pass goda bakgrundskunskaper i JavaScript eftersom vi samtidigt fått bättre förståelse för JavaScript när vi gått igenom kodens struktur. Även kring den statiska overviewn har vi först i slutet insett att vi borde ha dokumenterat detta under projektets gång och därför saknas en strukturerad beskrivning och dokumentation av appen och hur dess olika komponenter, klasser och objekt hänger samman.

## 7.2 Var är vi?

Det finns inom gruppen viss förståelse för strukturen i programmet. Anledningen till att gruppen inte fått en fullständig översikt är precis som för behavioural overview:n att vi hamnade efter i början och insåg för sent värdet av att skapa sig en god överblick genom tydlig dokumentation. Därför finns alltså i dagsläget en spretig förståelse inom gruppen för till exempel vilka klasser och komponenter som finns och hur dessa ser ut och hänger samman med andra klasser och komponenter. Det finns idag tankar om hur dokumentation skall ske för att förbättra förståelsen och överblicken men själva dokumentationen har inte gjorts i dagsläget. Slutsatserna av de diskussioner som förts i gruppen kring dokumentation presenteras under nästa avsnitt för hur vi vill att det ska se ut i framtiden.

Även om vi i dagsläget inte har använt oss utav några structure diagram så har vi i gruppen ändå skaffat oss en relativt god bild över de komponenter som vi har använt oss utav och modifierat. I dessa har vi förståelse för de ingående elementen och klasserna och hur en förändring av dessa ändrar utseendet eller en funktion på en viss del av appen. Exempelvis har vi lärt oss hur man använder sig utav Views och andra olika sorters komponenter från React Native.

## 7.3 Hur kommer det se ut nästa gång?

Vid kommande projekt kommer vi att ha en tydlig strukturerad bild av vilka klasser och komponenter som hänger samman och kommunicerar med varandra. Utifrån detta kommer vi kunna vara betydligt mer effektiva i utvecklingen av ny funktionalitet samt undvika dubbelarbete.

En kartläggning av klasserna hade också underlättat vid utformandet av user stories då det blir enklare att bryta ner den i konkreta uppgifter. Vi hade även minskat startsträckan innan konkret kodning kunde påbörjas.

Att få en bättre överblick på den statiska funktionaliteten kan hjälpa gruppen att tidigare i projektet få en bättre bild över hur de existerande klasserna och komponenterna fungerar och därmed underlätta en expansion. Vi anser att detta är viktigt då vi i framtiden tror att i majoriteten av projekten så kommer man att jobba med applikationer där det redan finns existerande kod. Därför kan förståelsen för den befintliga statiska strukturen vara avgörande för om projektet lyckas eller ej.

## 7.4 Hur åstadkommer vi detta?

För att lyckas med ovanstående mål i ett nytt projekt kommer vi att i uppstartsfasen av det nya projektet att tidigt identifiera vilka de berörda klasserna, filerna och eller komponenterna är och hur dessa berörs av en implementering av de aktuella nya funktionerna. När den identifieringen är färdig dokumenterar vi vilka delar av programmet som kommunicerar med varandra och på vilket sätt.

Dokumentationen kan ske genom att exempelvis att rita diagram över klasser och komponenter samt hur dessa förhåller sig till andra klasser. Beroende på vilken typ av uppgift vi ställs inför kan kartläggningens dokumentationssätt komma att variera

För att identifiera och dokumentera ett projekt tror vi att det kan vara viktigt att använda oss utav de redan existerande UML-verktygen för att dokumentera den statiska funktionaliteten. Vi tror att ett användande av exempelvis komponentdiagram kan göra det enklare att förstå en komplex mjukvarustruktur med många olika ingående komponenter och att detta kan snabba upp inlärningen och förståelsen för det givna programmet. Detta är något som vi upplevt som ett hinder i detta projekt då vår metodik för att lära oss appens uppbyggnad var väldigt tidskrävande och ibland ledde till dubbelt arbete då vi dels inte delade med oss av hur vissa komponenter egentligen hängde samman i gruppen samt att vi ibland glömde av det då vi inte dokumenterat det.

Ett annat typ av verktyg är klass diagram och detta kan underlätta arbetet med att förstå hur de ingående klasserna i exempelvis en komponent hänger samman. Dessa gör det enklare att förstå och analysera den statiska funktionaliteten i en applikation. Det kan även hjälpa till att beskriva hur klassernas ansvarsfördelning ser ut i programvaran. Detta är något vi tyckt varit problematiskt att förstå då det varit så pass många olika klasser i den existerande appen. På grund av detta har det varit svårt att förstå vad varje klass har ansvar för.

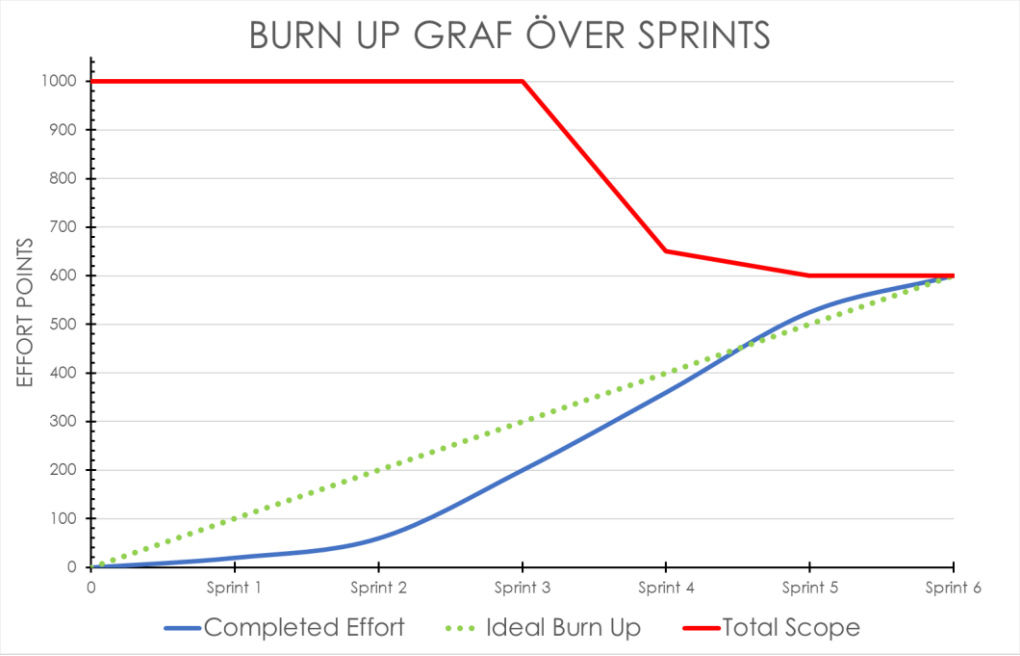
# 8 User Stories

## 8.1 Sammanfattning av tidigare sprinter

Under de första veckorna skrevs inga user stories. När detta arbete sedan kom igång hade vi problem med att bryta ner och definiera våra user stories på rätt sätt. När vi studerat litteratur och diskuterat med kursansvariga kom vi på rätt köl. Kniberg (2015) beskriver exempelvis hur man inte skall definiera user stories efter hur de ska implementeras utan istället efter vilken funktionalitet man efterfrågar. Efter några sprintar använde vi oss av en modifierad modell av uppbyggnaden av user stories som Kniberg (2015) presenterar. Den innehåller information om vilken Epic en story tillhör, en kort beskrivning av funktionaliteten som efterfrågas, vilken effort som uppskattas och senare även vilka kriterier som skall uppfyllas under ett acceptance test. Prioriteringsordning definierades av ordningen korten lades i Trello vilket också är något Kniberg (2015) påpekar som smidigare än att sätta en siffra på kortens prioritet.

Det har varit ett genomgående problem för gruppen att uppskatta vilken effort varje task har krävt. Dock har en tydlig förbättring kunnat ses som antas komma av att kunskapen kring appen och den bakomliggande strukturen har gjort det enklare att på förhand uppskatta hur tidskrävande olika implementationer är. Detta gjorde att vi i början hade en effort estimation som ej var korrekt och gjorde att vi fick en diskrepans mellan vår velocity och hur många user stories vi valde att placera i sprintens logg.

Gällande gruppens velocity valde vi att sätta denna till en fast nivå över veckorna på ett värde av totalt 100 för gruppens alla medlemmar. Över tid när våra kunskaper och förmågor utvecklades och blev bättre blev vi mer effektiva och därmed blev efforten för att göra vissa uppgifter lägre. Alltså valde vi att hålla velocityn fast och sänka vår uppskattade arbetsbelastning för en viss uppgift istället för att hålla efforten för en viss typ av uppgift fast och höja vår velocity under projektets gång. Detta är något vi reflekterat kring under tiden och kanske hade det varit mer lämpligt att använda oss utav en variabel velocity för att tydligare visualisera vår ändring i förmåga över tid.




*Figur 5: Burn Up graf över våra sprintar*

I figur 5 ovan visas hur projektets jobbade med sina user-stories. Den röda linjen representerar hur stort vi från början tänkte att projektets scope skulle vara men att vi senare, som beskrivits tidigare, i samråd med produktägare krympte scopet. Vi ser också att vår “burned” effort varierar kraftigt över tid och att vi i slutet presterade mycket mer.

Under tid blev gruppen bättre på att avgöra om en user story levde upp till grundkraven på en user story, alltså att den karaktäriseras av att vara Independent, Negotiable, Valuable, Estimable, Small, and Testable. Framförallt fokuserade vi på att försöka finslipa våra user-stories med avseende på att vara lagom små, värdefulla, oberoende, samt att vara möjliga att testa vilket knyter an till våra acceptance criterias.

För varje Epic bröt vi ned denna i mindre så kallade tasks eller task elements. Ett exempel på dessa presenteras i figur 6 nedan. Här har vi även kopplat våra acceptance criterias till varje task element. I slutet av varje sprint samlades gruppen för att utvärdera arbetet under sprinten och för att avgöra om vi hade färdigställt de items som hade lagts i sprint backloggen. För att ett task element skulle anses vara färdigt kontrollerades att de satta acceptanskriterierna hade mötts.

I figur 6 ges ett exempel på utvärderingen som gjorde efter sprint 5 där vi kan se att de flesta task elementen ansågs vara färdiga men att det fanns ett element som ej hade nått sitt acceptanskriterium fullständigt. I kolumnen placerad längst till höger ges för de flesta av task elementen en så kallad retrospective review. Detta är den reflektion och utvärdering som vi gjorde för varje task i slutet på sprinten och som användes för att ge input hur arbetet i nästa sprint skulle läggas upp.

Gällande uppdelningen av våra user stories hade gruppen som tidigare nämnt svårt att dela upp våra stories i tillräckligt små “slices”. Under de första sprintarna lades därför stor möda på att försöka förbättra denna förmåga hos gruppen och detta gav utdelning under de sista sprintarna då vi lyckades att bättre uppskatta hur stora user stories som var lämpliga att ha individuella för att det skulle var enkelt och lätt att förstå den och klara av att lösa den. Då vi lade stort fokus på storleken använde gruppen inte något annat standard pattern för att dela upp user stories.

| SPRINT 5 \| UTVÄRDERING | | | | | |
|---|---|---|---|---|---|
| Epic User Story | Task Element | Effort Estimation | Acceptance Criteria | Avklarad? | Retrospective Review |
| **EPIC 5** | Berth-scroolview ska vara sorterad i alfabetisk ordning. | **10** | När man går in på fliken Berths skall de vara sorterade i alfabetisk ordning | **JA** | |
| **EPIC 1** | Påbörja skapandet av en konceptuell modell för MyTrip för att visualisera hur en framtida sådan funktion kan se ut för produktägarna | **30** | Wireframe app används för att visualisera en kommande app | **Delvis** | Vi påbörjade denna task men han ej klart med alla funktionaliteter och därför kommer vi att låta vara den kvar till nästa vecka och försöka att färdigställa den innan redovisningen nästa vecka då vi vill visa upp den för de olika intressenterna |
| **EPIC 8** | En vessel ska ha en textruta under vesselinfo där övrig information kan läggas till | **10** | När man går in på vessel info finns där en ruta som har rubriken "Additional info" | **JA** | |
| **EPIC 8** | Rutan med övrig information skall kunna redigeras | **50** | Skriv in något i textrutan och klicka på save knappen, se om texten uppdateras i additional-info rutan<br><br>Lägg till ytterligare information i den befintliga texten och se om det går att spara<br><br>Kontrollera så att det ej går att överskrida max-limiten på textrutans innehåll<br><br>Ta bort all text i rutan, spara, fungerar detta?<br><br>Lägg till text, byt till en anann funktion i appen, gå tillbaka, finns texten kvar? | **JA** | Detta var en STOR task. Även om vi i slutändan lyckades med den under veckan hade vi kunna använda oss utav ytterligare task-breakdown för att göra det ännu mer troligt att lyckas med färdigställandet av det.<br>Till framtida sprintar tar vi med oss detta då vi jobbar med mer avancerade och stora task och stories att det är viktgt att bryta ned dem ordentligt.<br><br>Gruppen anser också att vi är väldigt nöjda att vi lyckades sätta upp ett större antal relevanta acceptance criteria's till denna uppgift som vi i slutändan kunden testa och uppnå, roligt! |

*Figur 6: User Stories Sprint 5*

## 8.2 Var är vi?

Vi har i dagsläget förbättrat oss avsevärt och uppskattar med relativt god precision vilken effort som krävs för en viss typ av task. Vi lyckades i sista sprinten även göra en korrekt effort estimation som väl svarade mot vår satta velocity samt att använda oss utav rimliga acceptanskriterier. Under arbetets gång hade vi, som nämns mer ingående i sammanfattningen, problem med att på ett bra sätt uppskatta efforten för de olika elementen. I slutet av sprinten upplever vi att vi har en större kompetens i att veta hur lång tid det tar att lösa exempelvis en viss funktion.

I dagsläget har vi fokuserat på att skära våra task element i lagom stora vertikala skivor då vi insett att detta har varit det viktigaste för att de ska vara genomförbara vilket är något vi lärt efterhand. Vår task breakdown har därför fokuserat på storlek som största faktor.

Att för varje vecka skriva ned en retrospective review för de task element som det varit relevant att göra detta för, oftast de som ej blivit färdigställda eller på annat sätt avvikit från den ursprungliga planen, under vår sprint review har hjälpt oss att få input i hur vi varje vecka bör förändra vår sprint planning och därmed hjälpa oss att på ett bättre sätt designa och välja våra user stories. Detta tror vi är en nyckel till att vi blivit bättre på effort estimation då vi tagit till oss reflektionerna inför nästa sprint.

Nedan presenteras i figur 7 hur våra user stories utvärdering för den sista sprinten såg ut och vad vi kom fram till. Att vi äntligen lyckats uppskatta vår effort och gör rimliga acceptanskriterier är något som vi är mycket nöjda över.

| SPRINT 6 \| UTVÄRDERING | | | | | |
|---|---|---|---|---|---|
| Epic User Story | Task Element | Effort Estimation | Acceptance Criteria | Avklarad? | Retrospective Review |
| **EPIC 7** | Som en båt vill jag ha ett standardfilter med endast relevanta time stamps att skicka för mig som båt. | **20** | Kan välja bort och/eller anpassa de valda favoriter<br><br>När jag går in på time-stamps visas bara favoriter | **JA** | |
| **EPIC 8** | Finslipa additional information så att tangentbordet ej döljer textinputen | **30** | När man klickar i textinputen ska ej tangentbordet dölja inputen | **JA** | Löstes genom input från Pontus där vi ta till extra space, går förmodligen att lösa snyggare i framtiden men för |
| **EPIC 1** | Utveckla MyTrip för att matcha projektägarens nya krav gällande konceptuell modell | **50** | Den konceptuella modellen ska vara komplett och göras i wireframe så att den går att köra och visa upp. Connection mellan alla "view" ska flyta smooth | **JA** | Denna är vi väldigt nöjda över och fungerade bra, nu laddar vi inför presentationen" |

*Figur 7: User Stories Sprint 6*

## 8.3 Hur kommer det se ut nästa gång?

I framtida projekt vill vi ta med oss att det är viktigt att våra user stories delas upp på ett korrekt sätt redan från början och att de då inte blir för stora, utan vi bibehåller tekniken med att ha smala “slices” eftersom det i vårt nuvarande projekt i början var alldeles för stora user stories vilket gjorde att vi dels hade svårt att veta hur vi skulle angripa dem för att kunna lösa dem samt att det var svårt att göra en rimlig effort estimation.

En annan insikt är hur viktigt det är att våra user stories kopplas väl till våra produktägares och slutanvändares krav då det annars kan bli så att vi utvecklar felaktiga och oönskade produkter och funktioner.

Vi tror också att det är viktigt att bli bättre att bättre uppskatta hur lång tid en user story tar att göra då detta är ett av de största problemen vi haft. Om man lyckas att uppskatta detta bättre är det lättare för teamet att dels våga lova hur mycket man kan göra för produktägaren men även för att kunna förhandla med densamme samt att det blir enklare att planera för projektteamet.

## 8.4 Hur åstadkommer vi detta?

För att bli bättre på att uppskatta hur lång tid en user story kommer ta att genomföra tror vi att det är viktigt att lägga större vikt vid att bli bättre på effort estimation. Detta tror vi hänger starkt ihop med att gruppen är ärlig när de sätter sig ned och planerar sprinten och att de vågar säga hur stora dels deras förmågor är men även hur mycket tid de har att lägga på projektet. Om detta görs på ett korrekt sätt kan

en första initial velocity tas fram. Utifrån denna velocity väljs sedan user stories till den mängd som anses leva upp till en korrekt effort.

Efforten för den första sprinten sätts utifrån magkänsla och erfarenheter då vi tror att det viktigaste är att gruppen testar hur allt fungerar första veckan. Efter första sprinten och utvärderingen anser vi att det då är viktigt att göra en ordentlig utvärdering och se hur väl gruppens uppskattning av velocityn var och justera den. Samtidigt bör gruppen uppdatera effort valet iterativt. Genom att göra detta kan vi få en task burn down som är mer jämn och stabil.

För att förenkla nedbrytningen av user stories i tasks kommer en lämplig strukturerad metod att appliceras. Exempel på detta kan vara att använda sig av work flow steps pattern om det lämpar sig väl för arbetsprocessen i det specifika projektet.

# 9 KPIs

## 9.1 Sammanfattning av tidigare sprinter

Under större delen av projektets gång har tre KPI: er använts. Dessa var Customer satisfaction, Velocity, Defects. Under de fem första veckorna hade vi problem att leverera något konkret varför den första och tredje av dessa var svåra att göra några mätningar av och därför sattes heller inte upp någon bra rutin för hur detta skulle göras.

Velocity:n har kunnat mätas löpande under varje sprint genom att jämföra hur stor effort som klarats av under sprinten med den velocity som sattes upp på förhand. Senare i projektet ändrades KPI:erna något för att göra dessa mer mätbara. Velocity:n var oförändrad eftersom det fanns ett tydligt sätt att mäta den. Defects kunde under senare delen av projektet börja mätas genom att producerad kod kördes genom debugprogrammet DeepScan. Customer Satisfaction bröts ner i fyra underkategorier som betygsätts på en skala från 0–10.

## 9.2 Var är vi?

Som beskrivs i stycket ovan har vi under kursens gång kommit fram till tre olika KPI: er som vi använt:

- **Customer Satisfaction:** Detta mäts genom att dela in kundnöjdheten i fyra olika kategorier och sedan sätta poäng på en skala mellan 0 och 10. Ju längre arbetet fortlöpt har dessa poäng stigit och vi har kunnat mäta framsteg vi gjort mellan de olika sprintarna. Dessa är kategorierna med dess slutpoäng:
    - *Antal implementerade funktioner:* 5
    - *Hur välarbetade och integrerade de implementerade funktionerna är:* 8 (I samråd med Produktägare)
    - *Funktionernas bidragande nytta för slutanvändaren:* 9 (I samråd med slutanvändare)
    - *Funktionernas bidragande nytta för produktägaren:* 9 (I samråd med Produktägare)

- **Velocity**: Genom att mäta veckans genomförda effortpoäng kan vi se hur duktiga och insatta vi blir i grundkoden samt språket JavaScript. Här krävs dock att vi är duktiga på att sätta rimliga och jämförbara effortpoäng genom hela arbetet. Den här titeln på KPI:n blev lite konstig eftersom vi använde oss av statisk velocity och förändrade efforten istället, se fråga 8 för mer detalj kring effort estimation. Den borde ha hetat något i stil med effort då det är detta som har varierat. Se figur 5 i fråga 8 för hur denna KPI har utvecklats under projektets gång.

- **Defects**: För att se hur bra programmet fungerade mättes de totala antalet defekter. Med defekter menar vi hur många fel som hittas i koden med hjälp av ett program som kontrollerar kodkvalitet. Genom att göra detta kontinuerligt under arbetets gång erhölls en bra bild av hur kodningen går och hur väl de implementerade funktionerna fungerade. Precis som de övriga KPI: er uppdaterades denna på slutmötet för sprinten på onsdagar och jämfördes med föregående sprint. Mer ingående beskrivs detta under fråga 10.

Från dessa KPI: er har vi kunnat se att gruppens uppskattningar av effort för de olika user stories blivit bättre och bättre under kursens gång. Anledningen till detta är att är att vi har lärt oss mer och mer ju

längre kursen gått. Vi har kontinuerligt lärt oss mer och mer om hur appen fungerar, bättre förstår oss på koden samt att våra kunskaper i JavaScript har ökat.

När vi väl kom igång ordentligt med dessa KPI: er så har det varit ett bra sätt att mäta hur bra vi har presterat på olika områden i projektet. Som vi beskrivit över hade vi dock lite problem med några av dessa i början. Hade vi inte haft det hade vi kunnat mäta vad vi presterat i ett ännu tidigare stadie, något som gynnat projektet.

Customer satisfaction är nedbrutet i fyra olika delar. Två av dessa; funktionernas nytta för slutanvändare samt funktionens nytta för produktägare har varit svåra att konkretisera och få ett tydligt svar på. Detta eftersom det är svårt för både slutanvändaren och produktägaren att värdera hur nöjda de är med olika funktioner.

Vi har även haft problem med att vi inte riktigt presterade och skapade värde för någon (förutom möjligtvis oss själva) i början av projektet. Osäkerheten var stor och vi visste inte riktigt vart vi skulle. Detta diskuteras mer under fråga 13.

## 9.3 Hur kommer det se ut nästa gång?

För att kunna prestera ännu bättre i framtida projekt bör KPI:er tas fram och användas så fort som möjligt. Anledningen är att ju tidigare vi ser om vi gör rätt och hur vi presterar ju fortare kan vi börja styra projektet åt rätt håll. Vi synliggör alltså de områden som problem kan tänkas finnas inom. Något vi verkligen hade behövt i början av detta projekt.

En annan fördel men att få igång tankesättet om KPI:er tidigt är för att det under hela projektet syns om gruppen skapar värde eller inte.

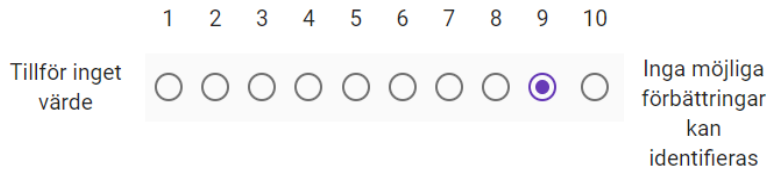
För framtida projekt av liknande form vill vi också att KPI:erna ska kunna hjälpa genom hela projektet samt att KPI:erna är mer heltäckande vad gäller projektets arbetsuppgifter. De KPI:er vi hade i detta projekt hjälpte mest till att mäta kodningen och inte övriga arbetsuppgifter.

Något annat som kommer förändras till nästkommande projekt är själva KPI:erna. Som beskrivs under “Var är vi?” hade vi problem med att få två delar av customer satisfaction värderat. I framtiden kommer vi tänka på att ha mer kvantitativa KPI: er som är enklare att mäta eller tydligare definiera, för att på så sätt få en mer rättvis bild om värdet på prestationerna.

## 9.4 Hur åstadkommer vi detta?

För att starta med KPI:er tidigt krävs det att vi snabbt tar reda på vad som kommer vara viktigt för att projektet ska lyckas. För att göra det bör ett möte med produktägare och slutanvändare hållas tidigt i projektet livscykel, där just detta diskuteras. Konkreta KPI:er bör därefter snabbt skapas och därefter följas. Det är också mycket viktigt att KPI:er väljs som produktägaren verkligen förstår och kan uppskatta. Annars snarare kan det leda till att gruppen får falska förhoppningar under projektets gång som sedan slutar med att produktägaren inte alls är så nöjd med slutprodukten som KPI:erna tytt på.

Vi kommer även använda oss av mycket tydligare riktlinjer om vilken poäng olika framsteg ska få. Detta var något vi inte använde under vårt projekt. Exempelvis skulle 0 poäng ges ifall inget värde alls hade tillförts, 5 poäng om implementationen var “helt okej” och 10 poäng om inget med implementationen kunde vara bättre. Som sagt använde vi inte oss av detta. Detta skulle kunna se ut som i figur 8.




*Figur 8: Exempel på hur en förbättrad  utvärdering av KPI:er skulle kunna se ut.r*

För att kunna ta hjälp av KPI:erna mer under hela projektets gång kommer vi att ha mer flytande KPI:er i framtiden. Med det menas att vi har KPI:er för det stadie projektet är i för tillfället. De bör alltså vara vissa KPI:er i början av projektet när fokus ligger på uppstart. När projektet kommer till en kodningsfas kan dessa KPI:er sedan förändras. KPI:n som refererar till hur mycket vi åstadkommer, “Velocity”, kommer i framtiden byta namn till “Effort” för att visa på det som faktiskt förändras under projektets gång.

# 10 Code Quality

## 10.1 Sammanfattning av tidigare sprinter

Kodkvaliteten var en av de KPI:er som användes under projektet för att mäta hur vi förbättrade oss. Som tidigare beskrivits var det naturligt så att inga mätningar gjordes under större delen av projektet. Efter att detta gjorts uppstod oklarheter kring hur vi skulle testa vår kod. Konsultation med den tekniska supporten för kursen gav inte så mycket hjälp då det uppenbarades att de som byggt appen inte använde sig av någon programvara för att testa kod. Tester som gjorts tidigare var endast användartester som säkerställde att appen fungerade som tänkt.

Till slut hittade vi ett program som vi kunde använda som heter DeepScan. Detta program användes för att testa kodens kvalitet genom att jämföra koden i sin helhet efter förändringar mot den på förhand existerande koden för appen. Detta då vi inte lyckades lösa tester endast på de delar som tillkommit under projektets gång. Bilder från DeepScan-körningar ligger under bilagor. Resultatet är dock även noterat under texten nedan.

Eftersom vi började med kodningen relativt sent började vi med DeepScan ännu senare. Detta har lett fram till att vi fått fokusera på att lösa de high och medium impacts och vi har fått acceptera att koden kommer ha ett antal low impacts då tiden att lösa dessa inte fanns. Detta eftersom vi fortfarande ville och var tvungna att skapa värde för både produktägaren och slutanvändaren de sista veckorna, vilket gruppen ansåg att vi gjorde bättre genom att utveckla nya funktioner i appen istället för att lösa low impacts på funktioner som redan fanns. Kodkvaliteten har kontrollerats i slutet på varje sprint och därmed antecknats.

## 10.2 Var är vi?

I Deepscan graderas koden efter hur många high, medium och low impacts programmet hittar. Den slutgiltiga koden innehåller 4 (5) high impacts, 21 (20) medium impacts samt 215 (176) impacts. Vi jämförde grundkoden mot koden med våra tillägg för att se vilken impact vi skapat. Alltså ser vi att antalet high impacts minskat med 1, antalet medium impact ökat med 1 samt att antalet low impact ökat något. Att reducera en high impact är enligt DeepScan mycket viktigt då det är dessa fel som har störst negativ påverkan på koden. Att vi fått 1 medium och några low impact är givetvis ej önskvärt men dessa fel, speciellt low, har ej någon markant påverkan på kodens kvalitet. Därför anser gruppen att vi lyckats med att hålla en god kodkvalitet utifrån gruppens förutsättningar.

## 10.3 Hur kommer det se ut nästa gång?

Vilken typ och vem projektet görs åt kommer att påverka hur hög kodkvalitet vi vill ha på framtida projekt. Är projektet åt en kund där säkerhet är superviktigt och buggar eller fel inte får förekomma kommer givetvis stor vikt läggas vid detta och det kommer strävas mot att ha noll impacts i koden. Är kraven från kund inte så stora kan istället en kod med vissa impacts tolereras, som exempelvis det

nuvarande projektet. Detta eftersom det isåfall kan vara viktigare att lägga fokus på att utveckla funktioner alternativt skapa ett annat värde för kunden.

Vi vill även börja kolla koden i ett program vid ett tidigare skede i framtida projekt. Anledningen att vi har low impacts är att tiden inte fanns för att lösa dem. Börjar vi kolla tidigare är chansen större att det kommer finnas tid till att lösa dessa ändringar då det går att göra direkt efter den nya koden har implementerats.

## 10.4 Hur åstadkommer vi detta?

När vi ser tillbaka på projektet och hur vi gjort ser vi direkt att något vi skulle ändra till framtida projekt är när kodkvaliteten kontrolleras. Som tidigare beskrivet har kodkvaliteten kontrollerats efter varje sprint. Efter att ha reflekterat över detta anser gruppen att en kontroll samt lösning av dessa impacts bör ske mer kontinuerligt, förslagsvis efter varje implementering av kod före den pushas till giten. Detta sätt är mycket bättre eftersom utvecklaren då har koden som ändrats färskt i huvudet och det bör därmed vara enklare och mer tidseffektivt att lösa dessa impacts direkt.

Vi åstadkommer rätt typ av kvalitetsfokus genom att komma igång med kodande tidigare i kommande projekt och då från start applicera kontinuerlig testning av kod både med hjälp av användartest och tekniska tester av kodens kvalitet.

Vi skulle även kunna vara bättre förberedda och kunna mer om Git och hur branches fungerar. I sådana fall kan vi använda branches för att skapa och koppla koden mot ett gemensamt testprogram. Detta gör att koden kan testas där innan den pushas upp till mastern.

# 11 Our Team Roles

## 11.1 Sammanfattning av tidigare sprinter

Under de två första förberedande veckorna användes inga definierade roller. Senare definierades rollerna: SCRUM-master, User story-responsible, Developer och Acceptance criteria testers. Dessa roller som definierades användes knapphändigt. Fram till de sista veckorna användes endast en SCRUM-master och de andra rollerna förblev odefinierade. Detta var till följd av att gruppen hade svårt att komma igång ordentligt och därför föll fokus bort från rollerna. I början av projektet valde vi att rotera rollerna inom gruppen då ville att alla skulle få testa på dem. Detta trodde vi skulle kunna förbättra arbetsprocessen då alla skulle få en bättre helhetsbild över projektet.

Inför de sista två sprintarna omdefinierades rollerna i vårt sociala kontrakt och konkretiseras mer för att fylla ett specifikt syfte som gruppen upplevde ett behov av. Behovet uppstod när alla 8 medlemmar kommit igång med arbetet och bristen på tydliga roller gjorde det svårt att veta vad varje medlem skulle arbeta med. Vi märkte också att en hel del dubbelarbete genomfördes. Vi valde då också att sluta rotera rollerna och hålla sig till fasta roller då vi ansåg att rotationen skapade mer förvirring än nytta då man ej hann komma in i rollen på en vecka.

Det var bristfällig ordning i Trello vilket ledde till ett behov av någon som ansvarade för strukturen där då alla i gruppen lade till kort utan någon given struktur.

Gruppen använde sig inte av branches i Git, efter tips från Pontus, vilket ledde till ett behov av att ha någon som såg över varje push till mastern för att undvika konflikter mellan versioner.

När kodens kvalitet väl skulle mätas blev det tydligt att vi behövde tydliga kriterier koden skulle uppfylla innan den kunde testas i programvara. Detta ledde till behovet av någon ansvarig för att varje user story hade relevanta acceptance criteria och att koden sedan testades.

Vi upplevde även problem med vår kommunikation med produktägaren. Det fanns osäkerheter i vad vi trodde att de tyckte var viktigt samt vad de hade för acceptanskriterier. Detta problem tillsammans med rekommendationer från gästföreläsare gjorde att vi införde rollen intern produktägare som har haft ansvar för kontakten med produktägaren.

## 11.2 Var är vi?

I vår näst sista sprint utvärderade vi rollerna en sista gång för att komma fram till följande roller inom teamet. Rollen som SCRUM master har det övergripande ansvaret för att se till så att vi jobbar enligt SCRUM metodiken, att teamet fungerar och jobbar effektivt. SCRUM mastern ska se till så att alla teammedlemmar arbetar i rätt riktning och har de förutsättningar som krävs för att göra ett bra jobb. SCRUM mastern har även koordinerat mötena och programmeringstillfällen samt deltagit på SCRUM-of-SCRUM: s varje onsdag för att stämma av med övriga grupper kring hur det går, få tips och utvärdera hur sprinten varit.

Vi har även infört en intern produktägare som har haft ansvar för kontakten med alla externa aktörer, så som produktägare och slutanvändare, samt som har kunnat representera dessa under våra interna möten för att då vara säkra på att vi jobbar emot vad de efterfrågar och att vi prioriterar våra user stories på rätt

sätt. Den interna produktägaren ansvarar även för att mäta customer satisfaction utefter det utformade poängsystem för att se till så att de utvecklade funktionerna faktiskt skapar värde. Efter införandet av denna roll upplevde gruppen ett skiftande fokus med en tydligare riktning mot produktägaren. Gruppen upplevde att vi kunde leverera mer värde till produktägaren efter införandet av denna roll, vilket även visas i KPI:et customer satisfaction.

Den tredje rollen är Trello-ansvarig som ska ha det övergripande ansvaret för Trello och med detta ansvara för att rätt user-stories är på rätt plats. Detta har hjälpt oss strukturera upp våra user stories mer konkret och gett oss en bättre överblick av vad som behövs göras.

En Git-ansvarig infördes som ska ha det övergripande ansvaret för all aktivitet i Git så att det inte blir några problem och även ansvara för att alla individuella och team-reflektioner laddas upp varje vecka. Även detta gav gruppen en bättre struktur och såg till att versionshanteringen sköttes på rätt sätt. Detta ansågs extra viktigt då vi inte jobbat med branches utan endast en master.

Den kvalitetsansvariges främsta funktion är att kontrollera kvaliteten i de user-stories som utformas. Detta innebär dels att testa koden för buggar via lämpligt program samt att sätta acceptance tests.

Att några utav gruppmedlemmarna fick olika roller gjorde inte att de behövde lägga ner mer tid än de utan någon specifik roll utan endast att det fanns ett ansvar kring att vissa saker skulle skötas på rätt sätt.

Övriga medlemmar i teamet har ingen specifik tilldelad roll utan arbetar aktivt med utvecklingen av appen och verkställandet och skapandet av user-stories. Vi ansåg att det inte ska tilldelas roller bara för sakens skull, utan något specifikt syfte, vilket gjorde att det inte fanns behov av fler ansvarstagande roller.

## 11.3 Hur kommer det se ut nästa gång?

Nästa gång så kommer vi analysera behovet av samt tilldela roller i projektet direkt vid start. Vi anser att detta behövs vid stora team för att alla ska jobba i rätt riktning och jobba produktivt även under de tidiga sprinterna vilket annars kan vara svårt vid nya grupper och vid ett nytt projekt. Detta underlättar sprintarna genom bland annat bättre samordning och kan även underlätta värdeskapande för de ingående aktörerna, genom exempelvis i vårt fall den interna produktägaren. Nästa gång kommer vi se till att rollerna är relevanta genom hela projektet och tilldelade på bästa sätt.

## 11.4 Hur åstadkommer vi detta?

Vi åstadkommer tilldelningen av roller tidigt i projektet genom att redan i initieringsfasen tydligt definiera behovet av roller, vilka kunskaper de olika team-medlemmarna sitter inne på och vilka andra variabler och omständigheter(om produktägaren är aktiv etc) som kan påverka projektets framgång.

Vi kommer att öka rollernas relevans genom att utvärdera rollerna i varje sprint för att se så att det inte uppstått behov av någon ny sorts roll eller att behovet av en redan införd roll inte längre finns och därför inte är lämplig att ha kvar. Ett exempel på detta kan vara genom att sätta tidpunkter för utvärderingen av rollerna. Det är också lämpligt att ej vara rädda för att byta roller beroende på vilka olika specialkunskaper medlemmarna sitter inne på för att då ta mest nytta utav allas specialkunskaper så att gruppen når upp till sin fulla potential.

# 12 The Agile Practices

## 12.1 Sammanfattning av tidigare sprintar

När förberedelsefasen var över började tillämpningen av de Agila begrepp vi lärt oss. User stories, Product backlog, sprint log och acceptance criteria var de första som gruppen började försöka applicera i arbetet. De dagliga SCRUM-mötena applicerades sedan även de fast i en modifierad variant som innebär lägre frekvens då kursen inte är på heltid och detta ansågs rimligt. Därför hade vi tre veckovisa möte som en utgångspunkt.

Under projektets gång blev det uppenbart att gruppen hade svårigheter med att tolka vissa av begreppen och hur de skulle definieras och användas. Exempelvis hade gruppen länge svårt att uppskatta effort på våra user stories och svårt att greppa vad success criteria innebar. Mot slutet gjorde gruppen en mer ordentlig genomgång av vilka Agila practices som finns och insåg att många av dessa användes något omedvetet och att vissa används med viss modifikation för att passa vårt projekt.

## 12.2 Var är vi?

Vi använder oss av flera av de Agila arbetssätt som beskrivits i kursens föreläsningar. User stories appliceras för att definiera vilka uppgifter vi skall jobba med som ger värde åt de olika aktörerna som beskrivet under fråga 8. När det kommer till användning av user stories har vi utnyttjat en sprint och Product backlog.

Korta iterationer används då sprintarna är satta till en vecka. Kontinuerlig testning genomförs men hade kunnat förbättras ytterligare vilket beskrivs i avsnittet om kodkvalitet. Par programmering har använts från start till slut, dels på grund av att utvecklingsmiljön inte fungerade på alla datorer men även för att gruppmedlemmarna ansåg att de presterade bättre i par. Medlemmarna upplevde att det var enklare att lösa svåra problem i par. Vi har även använt oss flitigt av sprint reviews för att snabbt kunna byta riktning vid behov och synliggöra eventuella problemområden.

## 12.3 Hur kommer det se ut nästa gång?

Nästa gång kommer vi att fortsätta arbeta med de Agila arbetssätt som fungerat bra under detta projekt. Dessutom kommer vi att försöka utnyttja ett mer tvärfunktionellt arbetssätt i våra team, exempelvis för att vi insett att det finns områden där vi har bristande kompetens.

Vid projekt mer inriktade på att leverera resultat än att lära sig metodik kommer även mer fokus läggas på att ha en entydig standard på koden som skrivs för att underlätta möjligheter att sätta sig in i varandras kod. Möten skulle vid projekt som utförs på heltid även kunna ske på daglig basis mer direkt kopplats till SCRUM-metodiken där problem mer frekvent kan lyftas. Om någon behöver hjälp kan denna person lättare få det. En mer jämn och hållbar arbetsbelastning samt takt kommer att eftersträvas i kommande projekt för att undvika potentiella stressperioder och öka välmåendet inom gruppen som beskrivet i fråga 1.

## 12.4 Hur åstadkommer vi detta?

Genom att ta med oss de lärdomar som detta projekt har gett kommer vi kunna förbättra vårt arbetssätt. Vid uppstart av projekt kan vi ta hänsyn till fördelen med att utforma teamet efter olika kunskaper och bakgrund. Exempelvis kan vi vid ett liknande projekt försöka ta in kompetens i vårt team från en annan utbildning. Entydig kodstandard kan lättare upprättas om förkunskapen för språk och utvecklingsmiljö är bättre i teamet.

Beroende på projektets omfattning och utformning kommer en mer strikt mötesstruktur med dagliga möten kunna sättas upp vid uppstart av projektet. Om projektet är längre kommer det naturligt bli mer jämn arbetsbelastning om då uppstartsfasen är en variationsfaktor enligt erfarenhet från detta projekt. Vår arbetstakt kommer även att påverkas genom att minska vårt scope och resultera i ett mer jämnt och hållbart tempo genom hela projektet. Som diskuterat i fråga 1 startades detta projektet med ett alldeles för brett scope och en ojämn arbetstakt, dels på grund av externa faktorer men även dålig insikt i hur SCRUM och ett Agilt arbetssätt skall appliceras. Med erfarenheten gruppen erhållit kommer detta förändras till nästa projekt.

# 13 The Time Spent on the Course

## 13.1 Sammanfattning av tidigare sprintar

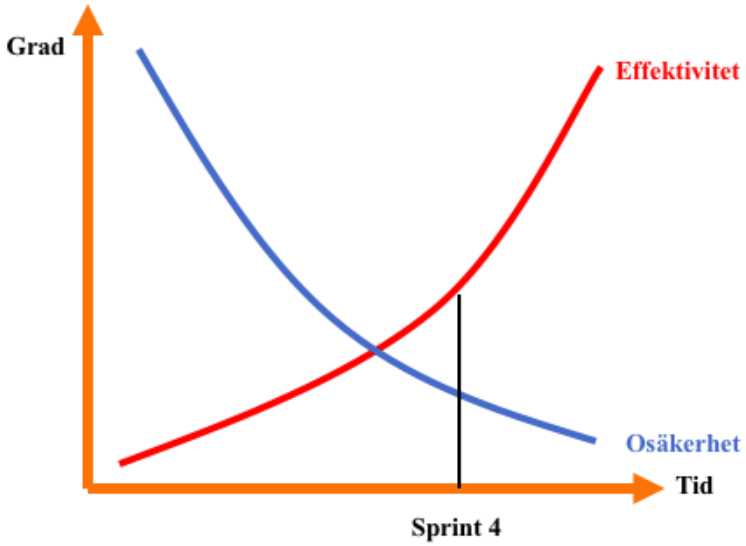
Målet var från början att lägga 20 timmar varje vecka. Detta blev dock inte utfallet utan det förekom en del variation från denna siffra av olika anledningar. Under uppstartsfasen hade gruppen tekniska problem som gjorde att något mindre tid lades för att kunna lösa problemen på ett mer tidseffektivt sätt under konsultationstillfällena. Senare lades mindre tid kring valborg och kandidatarbetets slutskede. Detta försökte gruppen kompensera genom att lägga mer tid då omständigheterna gav större utrymme för fokus på kursen.

Något som även framgår av individuella reflektioner är att tiden varierat något mellan gruppmedlemmarna, detta har gjorts med förståelse från resterande i gruppen på grund av olika omständigheter. Därför har inte exakt lika mycket tid lagts alla medlemmar under varje vecka men detta anses också fullt rimligt. Den sista veckan har en extra växel lagts in för att kompensera för de timmar som inte lagts under de tidigare veckorna och kandidatarbetet är helt färdigt. Den periodvisa variationen har inte varit optimal sett till arbetsbörda och stressnivåer men har ändå fungerat i gruppen då kommunikationen varit god.

Förutom mängden tid gruppen har spenderat varje sprint (ca. 20h) har valet av fokus under varje sprint förändras under projektets gång. Under projektets inledningsfas, de första prepveckorna, lades väldigt mycket tid på att förstå vad SCRUM innebär, vad projektet går ut på, hur scenariot tidigare år är upplagt osv. När sprinterna började komma igång låg fokus främst på att få igång utvecklingsmiljön hos samtliga medlemmar. Detta resulterade i att en stor del av tiden inte gick direkt till värdeskapande till slutkund, utan snarare indirekt. Genom mer kunskap kring hur utvecklingsmiljön är uppbyggd, vad projektet ämnar uppnå osv. kunde gruppen arbeta mer effektivt med värdeskapande fram mot de sista sprinterna.

## 13.2 Var är vi?

I dagsläget, och de senaste 2–3 sprintarna, har det största fokuset har legat på faktiskt värdeskapande för produktägaren och slutanvändare. När vi började få bättre koll på projektets och kursens mening samt hur utvecklingsmiljön var uppbyggd minskade vår osäkerhet och vår effektivitet kunde helt plötsligt börja öka. De 20 timmarna vi spenderade i snitt varje vecka blev mycket mer värda, och de senaste 2–3 sprintarna har vi lyckats tillföra mycket mer i jämförelse med de första.




*Figur 9: Visar hur graden av effektivitet ökar när graden av osäkerhet minskar. Ungefär runt sprint 4 började effektiviteten accelerera.*

Som nämnt under svar på första reflektionspunkten så hade gruppen lite svårigheter med att realistiskt uppskatta tiden varje medlem kunde lägga per sprint. Ofta överskattades hur mycket tid som fanns tillgängligt. Exempelvis va 20h inplanerade under en sprint som innehöll röda dagar.

## 13.3 Hur kommer det se ut nästa gång?

Osäkerhet inom ett projekt kommer alltid vara närvarande. Det är något som är oundvikligt till viss grad. Det gruppen dock främst kan ta med sig är lärdomarna kring SCRUM. En stor del av den initiala tiden på projektet gick åt att läsa på om hur man arbetar Agilt samt att förstå syftet med projektet. Genom att gruppen redan har erhållit kunskap inom detta område kommer tiden på ett bättre sätt att kunna disponeras på områden mer direkt relaterade till projektets syfte.

Kunskaper om hur ett socialt kontrakt kan få gruppen på rätt bana och minska antalet diskussioner kommer även det påverka hur väl gruppen disponerar sin tillgängliga tid. Genom att effektivisera möten genom ett stabilt socialt kontrakt kommer mer tid kunna spenderas på faktiskt värdeskapande.

Nästa gång kommer gruppen även vara mer realistiska med uppskattning av hur mycket tid som finns att disponera per sprint genom exempelvis inte uppskatta 20h på en sprint då det finns många röda dagar, och anpassa scopet efter detta. Gruppen kommer även försöka ha en mycket bättre framförhållning när det kommer till att planera den totalt disponibla tiden som finns tillgänglig under hela projektets gång, för att erhålla så jämna sprintar som möjligt.

## 13.4 Hur åstadkommer vi detta?

Som nämnt ovan kommer våra kunskaper inom SCRUM och Agilt samarbete kunna bidra till att gruppens tillgängliga tid disponeras mer effektivt. Genom att sätta sig ner i och se vilka förutsättningar som finns under varje sprint kommer en mer realistisk tidsplan projektets uppstartsfas att kunna sättas från början. Exempelvis kan man då tidigt se att en sprint längre fram har röda dagar och därmed planera in för förlorad tid. Vi bör alltså vara mer öppna för att beakta externa effekter som kan tänkas störa tidsplaneringen.

# 14 The Sprint Review

## 14.1 Sammanfattning av tidigare sprintar

I slutet av en sprint har vi alltid utvärderat hur den gått genom en sprint review och sedan satt scopet för nästkommande vecka. Till en början körde vi en sprint från måndag till fredag.

Vi har under varje utvärdering försökt att ha en bra interaktion med produktägaren och slutanvändaren för att sätta mer konkreta mål som säkerställer att värde skapas. Här har utvärderingarna varit ett bra sätt att se om det arbete vi utfört faktiskt ger de värde vi veckan innan uppskattade.

Till en början upplevdes det svårt för gruppen att uppskatta den tekniska lösningen och svårigheten i en user story. Detta för att vi inte var tillräckligt insatta i appens struktur samt de möjliga tekniska lösningarna i JavaScript. Reflektionerna varje vecka har dock hjälpt oss utveckla denna egenskap över tid då vi varje vecka kunnat jämföra den faktiska lösningen mot den tidigare uppskattade och genom de dra slutsatser för framtida uppskattningar. På det sättet tror vi att vi hela tiden utvecklats när det gäller att smalna ner user stories till ensamma uppgifter och skapa mindre beroende sinsemellan.

Exempel på ovan var att vi genom reflektioner över vad vi åstadkommit jämfört med vad som var kvar av vårt framförhandlade scope insåg i relativt god tid att vi behövde en omförhandling med produktägarna. Vi kom då fram till att en del av scopet skulle programmeras helt färdigt medans några delar skulle utvecklas i form av konceptuella modeller.

Även om vi försökt undvika att ändra i en pågående sprint när det kommer till dess scope och user stories har detta ibland behövt ske. Detta har då tillåtits när vi som team känt att den framförhandlade user storyn kunnat lösas med en bättre lösning. Här har vi upplevt ett problem med att vi bara haft tillgång till produktägare en gång i veckan. Det kan vara så att vi faktiskt har hittat en bättre lösning på problemet och då är det bara att köra, men det kan samtidigt vara så att vi missuppfattade vad de faktiskt velat ha och utifrån den trott oss hittat en bättre lösning som inte alls var fallet. Det här är sådana saker som kommit fram genom reflektionerna och möte med produktägarna i slutet av sprintarna.

## 14.2 Var är vi?

Vi har tillslut hamnat i en punkt där nästa veckas sprint förhandlas fram väldigt tätt med produktägare för att säkerställa värde samtidigt som varje genomförd user story utvärderas ihop för att säkerställa att den stämde överens med den bild som produktägare hade av vad som behövdes till appen.

Vi upplever att problemet kring avsaknaden av kontakt med produktägaren har lösts genom att vi utvecklade rollen “ intern produktägare”. Detta gjorde att vi hade en person som var ansvarig att säkerställa att det vi ändrar och det vi jobbar med under sprinten faktiskt skapar värde. Det här menar vi har minskat osäkerheterna kring värdeskapandet och missuppfattningarna under utvärderingarna mellan gruppen och produktägare.

Vi har genom utvärderingarna också blivit mycket bättre på att skriva user stories som har en uppgift som står på egna ben och inte är beroende av andra. De har också hjälpt till så att vi tillslut blivit bra på att uppskatta effort som ligger inom ramen av detta arbete.

## 14.3 Hur kommer det se ut nästa gång?

En första reflektion vi tar med oss från de erfarenheter vi fått i detta projekt är att från början se till att inte gå ut med ett för stort scope till intressenter i projektet. Att lovorda vad som kan åstadkommas under projektets tid bygger förväntningar och kan dessa senare inte uppfyllas kan en känsla av besvikelse infinna sig. Detta trots att det som åstadkommits faktiskt är bra och tillfört värde.

Nästa gång kommer det finnas en mer kontinuerlig kontakt med produktägare och slutanvändare då vi insett att det är svårt att ha för mycket kontakt. Den är verkligen värdefull när det gäller att se till att värde skapas i projektet hela tiden och att de Agila arbetssättet där man byter riktning på sprinten osv faktiskt är rätt val.

Den kunskap vi nu fått med oss genom att arbeta med kontinuerliga utvärderingar kommer vara till stor hjälp i nästa projekt när det kommer till att lyckas skapa värde till kund snabbare än vad som åstadkoms i detta projekt. Detta då vi kan säkerställa att våra user stories är i linje med produktägaren och slutanvändarens krav.

## 14.4 Hur åstadkommer vi detta?

I ett nästa projekt skulle vi fortfarande vilja utforma en del Epics till en början för att sätta interna mål med vad som ska/kan presenteras under projektet då vi tror detta hjälper till att få en ambitiös gruppdynamik. Utåt sett skulle vi dock försökt att smalna ner scopet något för att inte bygga förväntningar och genom det hellre överträffa.

Vi skulle vilja öppna upp en kommunikationsväg där enklare avstämningar kan göras för att dessa kan genomföras utan att produktägare ska känna att det är en allt för krävande process. Exempelvis telefon, databank där skärm dumpar och liknande kan läggas eller mailkontakt.

# 15 Best Practices for Using New Tools and Technologies

## 15.1 Sammanfattning av tidigare sprintar

Under den tidiga fasen sattes strukturen upp för möten och hur arbetet skulle organiseras. Git var föreslaget som verktyg för versionshantering och Github valdes som programvara för att utnyttja detta. Även enkla verktyg som Google drive och Calender samt Slack användes av gruppen. Senare implementerades en SCRUM board i Trellos gratistjänst och textredigeraren Visual Studio Code användes för att ändra och bygga ut koden för appen och Android Studios för att bygga appen.

Tanken var att först sätta en ranking i siffror på våra user stories för att avgöra deras respektive ranking. Detta inträffade aldrig då vi istället satte de med högst prioritet högre upp på tavlan, något som Kniberg (2015) säger är lämpligt att göra. Vad gäller versionshanteringen i Git fanns det tidigt tankar om att använda sig av branches för att undvika konflikter. Detta användes dock aldrig efter rådet från Pontus att som ovana användare endast pusha direkt mot mastern. Mot slutet användes programvaror som Marvel och Photoshop för att skapa de konceptuella modellerna som efterfrågats av produktägaren. Koden skulle även börja kvalitets testas mot slutet och för detta användes programmet DeepScan.

## 15.2 Var är vi?

Gruppen har använt ett flertal olika tekniker, metoder och verktyg under kursens gång, dessa finns beskrivna i figur 10. Dessa har på olika sätt bidragit till den slutgiltiga funktionaliteten och/eller till implementationen av SCRUM.

| VERKTYG, TEKNIKER & METODER | SYFTE |
|---|---|
| **Android studio** | Detta program har använts för att skapa en emulator för att köra applikationen PortableCDM. |
| **Expo** | Expo användes för att emulera appen. Vi använde oss utav Expo som installerades via terminalen och som byggde på NPM. Expo gjorde det även möjligt att komma åt de olika körningarna globalt och kunna använd Expo:s app för att testa appen direkt på egen smartphone. En negativ aspekt med Expo är att det är en relativt buggig teknik vilket gjort att den krånglat. |
| **React Native** | React Native är ett bibliotek som användes för att kunna utveckla versioner till Android och IOS simultant.<br>Detta gör att man sparar tid då man ej behöver ha två utvecklingsprojekt igång samtidigt. |
| **GitHub** | GitHub har dels använts som kurshemsida för kursen och därmed har en hel del kurslitteratur och föreläsningsunderlag återfunnits där som har varit grund för programmerandet och implementationen av scrum. Dessutom har Github använts för att versionshantera samt samordna så att alla arbetar med samma kod. Vi skapade ett repo som alla clonade och laddade ned med hjälp av desktop-versionen av GIT. På Github användes ett repo och vi valde att pusha direkt till mastern då vi ej hade nämnvärd erfarenhet sen innan och därför rekommenderades vi att ej använda oss utav branches. På Github lade vi även våra veckovisa team och individual reflections |
| **GIT** | Git har använts för att sköta hanterigen av att kunna "Pusha" kod upp till github och "pulla" kod ner från github. Detta program har varit väldigt användbart då vi har kunnatm motverka dubbelarbete och därmed maximerat utnyttjandet av exempelvis parprogrammering. Git har därför fungerat som en bro mellan våra lokala versioner och den globala mastern som ligger på vår Repo på Github. |
| **Trello** | Detta program har framförallt använts för att på ett effektivt sätt samla alla user-stories och tydligt kunna visualisera hur en user-story tar sig från product backlog till completed. Trello uppdaterades löpande under sprintens gång i takt med att våra user-stories rörde sig mellan de olika stegen. De "kort" som används under sprintarna är sprint backlog (De user-stories som ingår i veckans sprint, tagna från product backlog), testing (De user-stories som ska testas för buggar och eventuella fel), review (De user-stories som är redo att demonstreras för produktägare och slutanvändare) och completed (Alla user-stories som är klara och godkända). Programmet har varit väldigt användbart under sprintarna eftersom vi lätt kan följa och dela med oss av processen bakom varje user-story. |
| **Slack** | Slack har vi använt som kommunikationsmedel. Dels har det använts för att kommunicera privat med gruppen och dessutom har det använts för att kommunicera med andra grupper, kursansvariga, samt för att få IT-support från bland annat Pontus. |
| **Visual studio code** | Detta program har används som textredigerare och därmed är det här vi har gjort tillägg/ändringar i den redan befintliga koden. |
| **Marvel och Photoshop** | Dessa programvaror användes för att ta fram konceptuella modeller, vilket är något som var efterfrågat av produktägare, och kunna<br>visa upp potentiella framtida funktioner för produktägaren. Dessa program har varit väldigt användbara för att visualisera hur den tänkta implementation kan komma att se ut och fungera i praktiken. Det har underlättat kommunikationen mellan oss och de andra inblandade aktörerna. Vidare så har några i gruppen sedan tidigare en vana att använda dessa program och därför fanns en viss trygghet i användandet av denna programvara och vi kunde på ett mycket effektivt sätt leverera nya idéer till produktägare och slutanvändare. |
| **Google Docs** | Gruppen har använt google docs för att föra mötesanteckningar under bland annat sprint-start up meeting och sprintreview. Dessutom så användes docs som ett verktyg för att kunna skriva på exempelvis team-reflection tillsammans. Vi har för varje ny vecka skapat en ny mapp med relevanta dokument som varit väldigt lättillgängligt för hela gruppen. |
| **DeepScan** | DeepScan användes för att testa vår kodkvaltiet och avgöra hur vår kod stod sig gällande antal buggar och fel. Från DeepScan togs en rapport fram vilken går att se dels i APPENDIX men även under fråga 10. |

*Figur 10: Verktyg, Tekniker och Metoder använda under projektets gång*

## 15.3 Hur kommer det se ut nästa gång?

I framtida liknande projekt kommer många av de verktyg/tekniker/metoder som använts att kunna utnyttjas igen. Vi insåg till exempel att Photoshop och Marvel kan vara användbara för alla user-stories i och med att det är ett bra sätt att stämma av med produktägaren och slutanvändaren så att man är överens om vilken funktionalitet som faktiskt ska implementeras. Detta kan underlätta för gruppens bild av det som ska göras men ger också ett kvitto på vad produktägaren och slutanvändaren vill ha. Svårigheter kopplat till detta är att man kanske inte klarar av att leverera exakt de funktioner som man haft med i konceptuella modellen på grund av att man underskattat svårigheter kopplat till implementeringen, beskrivs mer ingående i fråga 4.

Mer generellt har vi kommit fram till att vilka verktyg/tekniker/metoder som kommer kunna användas i framtida projekt kommer variera mycket och bero på vilka förkunskaper/erfarenheter som gruppmedlemmarna har sedan tidigare. Har man stor vana att använda ett visst program kan det vara en väldigt stor fördel eftersom man då kan minska antalet veckor som ägnas åt inlärning. Med det sagt så har en stor utmaning för oss varit att nästan alla verktyg/tekniker/metoder har varit helt nya för oss och det har därför varit långa inlärningstider.

Ibland så upplevdes det att vissa av de givna verktygen kunde vara ett hinder som orsakade försening och osäkerheter i projekt. Exempelvis upplevdes Expo vara buggigt vilket gjorde att många timmar spenderades på felsökning snarare än värdeskapande till kund.

Inför framtiden så tänker vi att vi dels i och med denna kurs kommer har bättre förståelse för en del verktyg/tekniker/metoder som kan användas inom Agil-programvaruutveckling men också att vi ska försöka utnyttja tekniker som vi redan kan i största möjliga mån.

## 15.4 Hur åstadkommer vi detta?

För att komma dit vi vill i framtiden så kan det vara bra att bygga upp en bra bas av verktyg/tekniker/metoder som kan användas i en Agil-utvecklingsmiljö. Med det sagt är det alltid bra att reflektera över varför man valt sina verktyg/tekniker/metoder för att kunna förstå vilka andra eventuella användningsområden de kan ha.

Något vi har insett i kursen är att man kan applicera det Agila arbetssättet utanför kursens gränser. Vi har exempelvis fått användning av detta i kandidatarbetet. Detta har likt situationen vi har befunnit oss i denna kursen varit väldigt komplex. Under gästföreläsning har det framkommit att Agilt är lämpat för just sådana situationer där det inte finns tydliga kopplingar och mycket otydligheter. Principen har gått ut på att börja med något och sedan utvärdera efteråt. Att dela upp arbetet i sprintar gör att man får fördelen att man börjar med något och bestämmer sig för en riktning men ändå erhåller en viss flexibilitet i och med att man efter sprinten kan utvärdera och tänka om.

# 16 Relation to Literature and Guest Lectures

## 16.1 Sammanfattning av tidigare sprintar

Till en början användes nästan uteslutande det som togs upp på föreläsningarna för att skapa förståelse för de olika Agila koncepten. Även Lego-övningen bidrog till gruppens förståelse för det Agila arbetssättet. Senare kompletterades detta med att göra litteratursökningar i kurslitteraturen för att få ännu bättre förståelse för de olika komponenterna av SCRUM.

Under mitten av kursen fokuserade gruppen på att lösa de tekniska problem som uppstått. Här användes mycket Google och slack-meddelanden från kurskamrater tillsammans med kurshemsidan. När utvecklandet väl kom igång fortsatte läsningen men avtog successivt ju längre vi kom i arbetet och allt mer specifika sökningar gjordes mot slutet.

Många tips och koncept har använts från litteratur och föreläsningar. Exempelvis har konceptuella modeller använts istället för att koda direkt vilket Mikael tog upp under sin föreläsning. Dock viktigt att beakta nackdelar med konceptuella modeller som nämnt i fråga 4. Dessutom har givetvis de olika koncepten kring SCRUM använts som togs upp både under föreläsningar och i litteraturen.

Till en början uppfattades kursen upplägg som väldigt komplicerat med mycket osäkerheter som inte gruppen hade ett tydligt angreppssätt till. Detta upplevdes väldigt frustrerande där mycket av informationen behövdes tas reda på själv och blev inte serverad på ett silverfat. Nu såhär i slutskedet känner vi dock att vi lärt oss mycket på detta och förstår att detta arbetssätt efterliknar mer det “verkliga” sättet att jobba ute i arbetslivet och är därför glada över de erfarenheter vi fått med oss. Många medlemmar anser att de har blivit mycket bättre på att hålla sig lugna vid stor osäkerhet och försöka bena ut problemen i mindre komponenter, för att på så sätt iterativt minska osäkerheten.

## 16.2 Var är vi?

I sista sprinten använde vi inte aktivt kurslitteratur då vi känner oss väl införstådda med den. Fokus låg istället på att fixa till de sista funktionerna för presentationen. I sammanfattningen beskrevs hur de övriga sprinterna relaterade till litteratur.

## 16.3 Hur kommer det se ut nästa gång?

Nästa gång vi ställs inför ett projekt där vi har lite förkunskaper kommer vi att på samma sätt inledningsvis fokusera mycket på litteraturen i syfte att minska osäkerheten, se figur 9 under fråga 13. Vi hade gärna sett till att mer kontinuerligt under projektets gång upprätthålla litteraturanknytningen för att säkerställa att vi agerar enligt de Agila arbetssätten.

Vi hade även velat skapa en slags gemensam informationsbank där vi kan samla in de olika förkunskaper som finns och insikter gruppmedlemmarna samlar på sig under arbetets gång. Detta för att minska informationsgapet som finns mellan gruppmedlemmar då vi i detta projekt upplevt det som ett problem som negativt påverkat arbetets fortskridande.

## 16.4 Hur åstadkommer vi detta?

Genom att inför varje sprint börja mötet med att gå igenom de delar av kurslitteraturen som är relevanta, kortfattat då såklart, kan vi säkerställa att vi jobbar på rätt sätt och samlar information i en informationsbank. Då kan vi samtidigt och prata igenom vilka som har de kunskaper som krävs av tasksen och om någon har lärt sig något nytt. Detta kan man sen samla i t.ex. Google drive för lätt åtkomst.

Vi kommer fortsätta gå in i nya projekt med ett öppet sinne och leta aktivt efter nya informationskanaler som kan tänkas innehålla relevant litteratur, istället för att bli frustrerad när tydlig information inte blir serverad på ett silverfat.

# 17 APPENDIX

## 17.1 EPIC User-Stories

Nedan finns en sammanfattning av de EPIC:s som har avklarats under projektet. Notera att EPIC 4 och 6 ej är med då dessa togs bort i samråd med produktägare.

| EPIC | BESKRIVNING |
|---|---|
| **EPIC 1** | Som produktägare vill jag se olika konceptuella modeller för potentiell funktionalitet som kan adderas i framtiden. |
| **EPIC 2** | Som Vessel vill jag första gången jag loggar in kunna välja vilket som är mitt fartyg. Detta ska ej kunna ändras |
| **EPIC 3** | Alla medlemmar ska lära sig de relevanta verktygen, få utvecklingsmiljön att fungera, kunna starta och köra appen, få kunskap om hur man programmerar etc. |
| **EPIC 5** | Som en Vessel vill jag endast se information om de Berths som finns i ankomsthamnen |
| **EPIC 7** | Som en Vessel vill jag att det ska vara enklare att skicka statusuppdateringar |
| **EPIC 8** | Som en kaptein vill jag dela med mig av övrig information om min last, eller andra upplysningar, till de andra aktörerna |

'

*Figur 11: Beskrivning av EPICs*

## 17.2 User-Stories

I nedanstående figurer presenteras en sammanställning över User-Stories, Task Elements, Acceptance Criterias samt retrospective för varje sprint. Dessa har sammanställts från Trello. Retrospective review gjordes i slutet på varje sprint.

| SPRINT 1 \| UTVÄRDERING | | | | | |
|---|---|---|---|---|---|
| Epic User Story | Task Element | Effort Estimation | Acceptance Criteria | Avklarad? | Retrospective Review |
| **EPIC 3** | Installera Node.JS | **5** | Kunna starta programmet | **Ja** | |
| **EPIC 3** | Installera och lära sig GIT | **10** | Kunna starta GIT och använda dess funktioner | **Delvis** | Vi hade underskattat hur pass tekniskt krävande det var att lära sig hur GIT användes och hur dess funktioner kunde användas från terminalen. Vi lyckades installera programmet men vi förstod inte hur kommandona fungerade |
| **EPIC 3** | Installera och förstå EXPO | **10** | Kunna starta EXP från terminalen | **Nej** | Samma här som för GIT, fast med tillägget att vi inte alls lyckades installera EXP denna veckan. På nästa Open Arena samlar vi på oss den information vi behöver. Vår effort estimation var här för optimistisk. |
| **EPIC 3** | Installera Android Studio | **10** | Kunna starta Android Studio | **Ja** | |
| **EPIC 3** | Kunna starta den existerande applikationen | **5** | Kunna starta appen genom Android Studios emulator | **Nej** | Då vi ej lyckats installera och använda EXP har vi ej kunnat starta appen. Vår effort estimation var här för optimistisk. |

*Figur 12: Sammanställning Sprint: 1*

| SPRINT 2 \| UTVÄRDERING | | | | | |
|---|---|---|---|---|---|
| Epic User Story | Task Element | Effort Estimation | Acceptance Criteria | Avklarad? | Retrospective Review |
| **EPIC 3** | Installera och förstå EXPO | **15** | Kunna starta EXP från terminalen | **JA** | Vi lyckades starta EXP på hos alla användare. Vissa delar av gruppen upplever fortfarande problem vid starten av tunneln men majoriteten av gruppen lyckas och vi bestämde under retrospective att de som ej lyckat kan köra par-programmering tills de får igång EXPO, därför anses tasken vara avklarad |
| **EPIC 3** | Bygga appen i Android Studio | **5** | Bygga appen utan att fel uppkommer | **JA** | |
| **EPIC 3** | Kunna starta den existerande applikationen | **15** | Kunna starta appen genom Android Studios emulator | **DELVIS** | En del av gruppen lyckades under denna sprinten starta och köra den existerande applikationen. Dock lyckas fortfarande majoriteten av gruppen ej med detta varför task:en ej ses som avklarad och den får följa med till nästa vecka. |
| **EPIC 3** | Lära oss använda Trello | **15** | Alla ska veta hur man lägger till kort, flyttar dem samt hur tasks och tilldelningar fungerar | **JA** | Gruppen anser att vi nått en grundläggande kunskap men det finns fortfarande utvecklingspotentiell men vi tror det kommer lösa sig under nästkommande sprintar |
| **EPIC 1** | Som en kapten vill jag kunna logga in i appen och se en vy över min framtida rutt till olika hamnar | **50** | Kunna testa i appen | **NEJ** | När vi skrev denna task i början av sprinten var vi alldeles för ambitiösa och vi trodde att ALLT skulle vara lättare det det var. Då vi inte ens lyckats få grundappen att fungera för alla har vi inte lyckats klara att lägga till någon mer funktionalitet och definitivt inte så här avancerad. Vi tar med oss att vi behöver bli bättre på att sätta rimliga stories. |

*Figur 13: Sammanställning Sprint: 2*

| SPRINT 3 \| UTVÄRDERING | | | | | |
|---|---|---|---|---|---|
| Epic User Story | Task Element | Effort Estimation | Acceptance Criteria | Avklarad? | Retrospective Review |
| **EPIC 3** | Kunna starta den existerande applikationen | **15** | Kunna starta appen genom Android Studios emulator | **JA** | Denna veckan nådde vi ett genombrott på denna fronten och för första gången lyckades gruppen få igång den existerande applikationen ordentligt. Dock var det en medlem i gruppen vars dator inte verkar klara av att köra programmet men då vi ändå bestämt oss för att använda oss utav par-programmering och då vi vill börja skapa värde för produktägaren och end-user väljer vi att släppa denna task och gå vidare. |
| **EPIC 3** | Varje medlem ska lära sig appens uppbyggnad hur koden är strukturerad i olika filer | **40** | Kunna beskriva appen på mötet | **JA** | Trots ett mycket oklart acceptance criteria anser vi trots allt att vi lyckats med denna task då vi känner att alla i gruppen har läst in sig på appen och har förstått vilka olika komponenter det finns. Vi tar med oss att det är viktigt att tydligare sätta bättre acceptance criterias som är tydliga då det annars kan uppkomma missförstånd |
| **EPIC 3** | Varje medlem ska lyckas ändra något i appen för att testa på hur detta kan förändra appen | **40** | T.ex. ändra färg på ikon | **JA** | Ja, alla lyckades på något sätt ändra appens utseende .ex. ändrade vissa i gruppen innehåller i rubriker och andra bytte färg på ikoner etc. |

*Figur 14: Sammanställning Sprint:3*

| SPRINT 4 \| UTVÄRDERING | | | | | |
|---|---|---|---|---|---|
| Epic User Story | Task Element | Effort Estimation | Acceptance Criteria | Avklarad? | Retrospective Review |
| **EPIC 2** | När man som Vessel loggar in ska en sida visas | **5** | En tom sida visas när man loggar in | **JA** | |
| **EPIC 2** | På de första sidan visas en lsita över tillgängliga fartyg | **30** | Alla fartyg visas när man loggar in | **JA** | |
| **EPIC 2** | Från denna listan ska man kunna välja ett fartyg | **30** | Man skall kunna klicka på ett skepp och detta skepp lagras i appen som valt skepp | **JA** | |
| **EPIC 2** | På startsidan visas en sökfunktion där man söker fram vilket skepp man vill vara | **30** | En sökruta finns och när man söker på ett existerande skepp presenteras detta nedan | **JA** | Denna task fanns ej från början utan lades till under sprinten då denna ansågs vara bättre och skapa mer värde för kund. Vi tar med oss att försöka tänka på detta innan sprinten tar till senare sprintar men vi anser att detta är ett undantag |
| **EPIC 2** | När ett skepp har valts ska sidan med portcalls endast visa portcalls som hör till det valda skeppet | **35** | När man går in portcall list så finns där endat portcalls som hör till det valda skeppet | **JA** | |
| **EPIC 7** | Locations för portcalls under create portcall är sorterade efter bokstavsordning. | **20** | När man gå in på locations är de sorterade enligt bokstavsordning | **NEJ** | Vi prioriterade EPIC 2 och hann därför ej färdigställa denna. Dock la vi till en extra task på EPIC 2 vilket beskrivs i tabellen två steg upp då denna ansågs vara av högre prioritet än att färdigställa denna. Att ta med till senare tillfällen är att kanske fundera djupare kring varje EPIC för att se om en ytterligare task behöver läggas till |

*Figur 15: Sammanställning Sprint:4*

| SPRINT 5 \| UTVÄRDERING | | | | | |
|---|---|---|---|---|---|
| Epic User Story | Task Element | Effort Estimation | Acceptance Criteria | Avklarad? | Retrospective Review |
| **EPIC 5** | Berth-scroolview ska vara sorterad i alfabetisk ordning. | **10** | När man går in på fliken Berths skall de vara sorterade i alfabetisk ordning | **JA** | |
| **EPIC 1** | Påbörja skapandet av en konceptuell modell för MyTrip för att visualisera hur en framtida sådan funktion kan se ut för produktägarna | **30** | Wireframe app används för att visualisera en kommande app | **Delvis** | Vi påbörjade denna task men han ej klart med alla funktionaliteter och därför kommer vi att låta vara den kvar till nästa vecka och försöka att färdigställa den innan redovisningen nästa vecka då vi vill visa upp den för de olika intressenterna |
| **EPIC 8** | En vessel ska ha en textruta under vesselinfo där övrig information kan läggas till | **10** | När man går in på vessel info finns där en ruta som har rubriken "Additional info" | **JA** | |
| **EPIC 8** | Rutan med övrig information skall kunna redigeras | **50** | Skriv in något i textrutan och klicka på save knappen, se om texten uppdateras i additional-info rutan<br><br>Lägg till ytterligare information i den befintliga texten och se om det går att spara<br><br>Kontrollera så att det ej går att överskrida max-limiten på textrutans innehåll<br><br>Ta bort all text i rutan, spara, fungerar detta?<br><br>Lägg till text, byt till en anann funktion i appen, gå tillbaka, finns texten kvar? | **JA** | Detta var en STOR task. Även om vi i slutändan lyckades med den under veckan hade vi kunna använda oss utav ytterligare task-breakdown för att göra det ännu mer troligt att lyckas med färdigställandet av det.<br>Till framtida sprintar tar vi med oss detta då vi jobbar med mer avancerade och stora task och stories att det är viktgt att bryta ned dem ordentligt.<br><br>Gruppen anser också att vi är väldigt nöjda att vi lyckades sätta upp ett större antal relevanta acceptance criteria's till denna uppgift som vi i slutändan kunden testa och uppnå, roligt! |

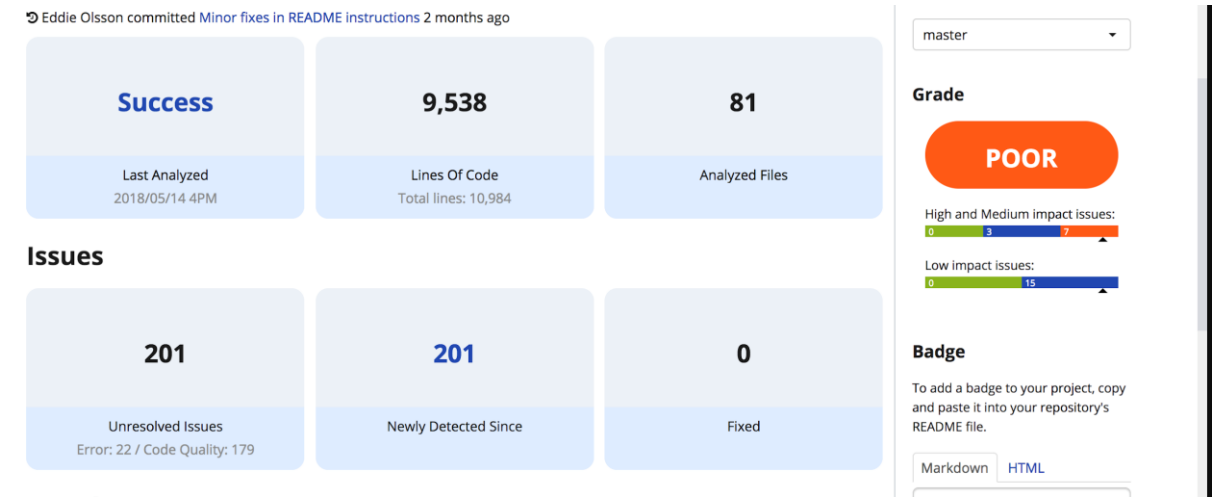
*Figur 16: Sammanställning Sprint:5*

| SPRINT 6 \| UTVÄRDERING | | | | | |
|---|---|---|---|---|---|
| Epic User Story | Task Element | Effort Estimation | Acceptance Criteria | Avklarad? | Retrospective Review |
| **EPIC 7** | Som en båt vill jag ha ett standardfilter med endast relevanta time stamps att skicka för mig som båt. | **20** | Kan välja bort och/eller anpassa de valda favoriter<br><br>När jag går in på time-stamps visas bara favoriter | **JA** | |
| **EPIC 8** | Finslipa additional information så att tangentbordet ej döljer textinputen | **30** | När man klickar i textinputen ska ej tangentbordet dölja inputen | **JA** | Löstes genom input från Pontus där vi ta till extra space, går förmodligen att lösa snyggare i framtiden men för |
| **EPIC 1** | Utveckla MyTrip för att matcha projektägarens nya krav gällande konceptuell modell | **50** | Den konceptuella modellen ska vara komplett och göras i wireframe så att den går att köra och visa upp.<br>Connection mellan alla "view" ska flyta smooth | **JA** | Denna är vi väldigt nöjda över och fungerade bra, nu laddar vi inför presentationen" |

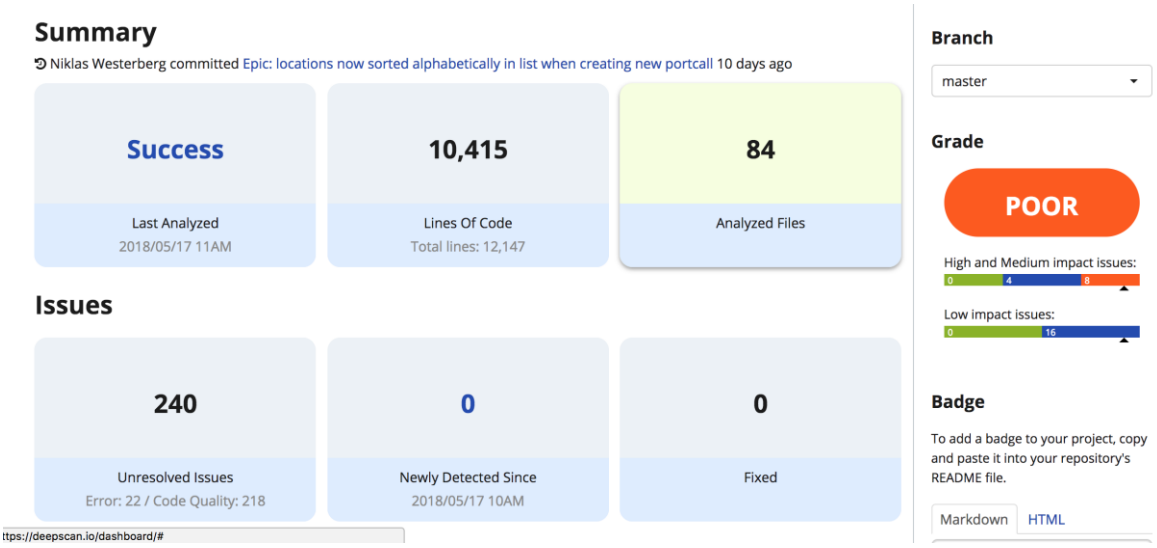
*Figur 17: Sammanställning Sprint:6*

## 17.3 DeepScan Resultat Buggtest Kodkvalitet

Att koden testats genom jämförelse mellan redan existerande kod och koden efter de förändringar som vi genomfört resulterar i bedömningen att den är “poor”. Eftersom vi inte gjort något försök att rätta till de befintliga problemen är därför programmets bedömning av koden ganska ointressant. Vad som blir intressant är att jämföra antalet fel som förekommer.




*Figur 18: Buggtest originalkod*




*Figur 19: Buggtest original kod + vår kod*