Final Reflection Report Charizard

1. Customer Value and Scope

1.1 The chosen scope of the application under development including the priority of features and for whom you are creating value

Vid första sprinten började vi med ett ganska stort scope där vi främst utgått från vår mockup med visionen för projektet, som redovisade både den design och funktion vi planerade att implementera. Utifrån mockupen och vår business model canvas började vi skapa user stories som vi sedan prioriterade i en backlog. Under vår första sprint arbetade vi brett med applikationen och vi påbörjade utvecklandet av många olika funktioner. Vid slutet av första sprinten hade vi inte lyckats färdigställa så många av de user stories som vi hade planerat, trots att vi hade en hög velocity inom teamet. Detta insåg vi efter vår review med handledaren berodde på att vi hade ett alldeles för brett scope med alldeles för horisontella user stories som i princip var omöjliga att färdigställa under en sprint. På grund av vårt breda scope hade vi inte heller prioriterat de user stories som innehöll vår främsta value proposition som vi specificerat i vår business model canvas.Ytterligare ett problem vi insåg var att våra user stories inte var formulerade på ett sätt som fokuserade på kundvärdet utan snarare bara fokuserade på team-värdet. Ett exempel på en alldeles för horisontell user story som inte fokuserar något på kundvärdet visas i Figur 1 nedan:

Figur 1: Ett exempel på en horisontell user story

Denna typen av user stories som är alldeles för horisontella leder ofta till att man skapar något som man inte ens har användning för i framtiden. Detta är något som Henrik Kniberg skriver om i sina “Lean & Agile Slides” där han menar på att ungefär hälften av det man bygger aldrig används.[1]

Vid sprint två bestämde vi oss för att ordentligt se över vad vår applikation skulle innehålla för funktioner och vilka som var viktigast för att generera kundvärde. Vi utgick från vår business model canvas, som vi skapade i början av projektet, och försökte fokusera på att

[1] Henrik Kniberg, *Henrik’s Lean & Agile slides,* 2018 https://www.dropbox.com/s/oqwx0tvpnv0udm1/Henrik%20Kniberg%20Agile%20Lean%20Slides.pdf?dl=0, hämtad 29 maj 2019

realisera det vi bestämde skulle vara vårt främsta value proposition. Vi drog ner ambitionsnivån för att inte fokusera på funktioner som inte är relevanta för kundvärde den närmsta sprinten. Utifrån vårt nya scope förändrade vi våra user stories så att de hade fokus på kundvärdet samt prioriterade de user stories som vi ansåg viktigast för potentiella användare av applikationen utifrån vår ambition. Vi behöll samma fokus i de följande sprintarna, vilket verkligen underlättade vår utveckling och vi lyckades skapa vad vi ansåg gav mest kundvärde i applikationen medan vi bortprioriterade det som ansågs vara extra tillägg som inte ingick i vårt huvudsakliga value proposition.

Till nästa projekt vill vi tänka på att ha ett mindre scope under de inledande sprintarna och istället bredda det under utvecklingens gång. Med det i åtanke ser vi inget negativt med att skapa en mockup utifrån visionen av hur applikationen i slutändan skulle kunna se ut och dess funktioner. Det vi däremot måste tänka på är att vid varje sprint planning fokusera på ett mindre scope så att vi i slutet av sprinten lyckats skapa något som genererat kundvärde istället för att man påbörjar utvecklingen av alla funktioner som applikationen i slutändan ska ha och inte lyckas skapa något direkt kundvärde överhuvudtaget den sprinten. Sammanfattningsvis kan detta beskrivas som att vi i ett framtida projekt ska arbeta mer agilt från första början och undvika praxis som leder till vattenfallsmetoden eller andra sekventiella utvecklingsprocesser.

1.2 The success criteria for the team in terms of what you want to achieve within the project (this can include the application, but also your learning outcomes, your teamwork, or your effort)

Vid skapandet av det sociala kontraktet fördes en diskussion gällande allas individuella mål med projektet samt hur vi skulle kombinera dessa för att uppnå ett gemensamt mål för hela gruppen. Här sattes följande mål upp:

1. Skapa något som vi är nöjda med och som vi möjligtvis kan visa upp för en potentiell användare utanför vårt team utifrån gruppens förutsättningar.
2. Lära oss om processen och att arbeta med Scrum.
3. Arbeta effektivt för att nå deadlines.

Under de inledande sprintarna lade vi mycket fokus på vårt andra mål då de flesta gruppmedlemmarna inte hade arbetat med Scrum tidigare. Vi försökte applicera de principer som Scrum lär ut i vår grupp vilket gick förhållandevis bra. Under första sprinten kunde vi även konstatera att vi inte arbetade tillräckligt bra med det första och tredje målet, då vi efter handledningen med Jan-Philipp insåg att vi fokuserade alldeles för lite på kundvärdet och våra user stories var inte uppbyggda på ett sätt som möjliggjorde uppfyllandet av det första målet.

Mål tre var väldigt kopplat till att vi skulle klara våra user stories vi planerat för, vilket vi misslyckades med under första sprinten och som till stor del berodde på våra alldeles för horisontella user stories och för bred scope. Efter den första sprinten såg vi en klar förbättring i hur stor andel av våra deliverables som levererades vilket även kan utläsas från vårt första KPI i Figur 2.

Sprint för sprint har vi blivit bättre på att uppnå våra mål vilket vi främst tror berott på våra växande kunskaper om Scrum-processer samt hur vi ska planera backloggen och varje sprint.

Problematiken med de mål vi satte upp för projektet låg i hur lite vi arbetade mot att nå målen. Det berodde till stor del på att målen var väldigt svåra att mäta. Det var sällan, om någonsin, som gruppen reflekterade kring hur vi skulle arbeta för att uppnå målen och till framtida projekt tar vi med oss vikten av att sätta upp mål som gruppen kan stå bakom och enkelt kunna återkoppla till efter varje sprint.

Vi tycker även att mål 1 borde formulerats bättre då dess innebörd kan upplevas vara relativt oklar. För trots att vi vet vad målet innebär för oss bör det ändå finnas en tydlig förklaring av det. Det är även ett mål som är väldigt svårt att mäta framgång i eftersom vi inte har pratat med en faktisk potentiell kund och fått deras feedback under projektets gång. Utifrån den formulering som vi har nu mäter vi i princip hur nöjda vi som utvecklare är snarare än hur nöjd kunden är. Med tanke på att vi strävar efter att bidra till kundvärdet borde vi vara nöjda först när den potentiella kunden är nöjd, vilket vi inte har lyckats mäta i det här projektet. I ett framtida projekt hade vi troligtvis valt att formulera målet som “att skapa något som vi som team är tillräckligt nöjda med för att våga visa upp för en identifierad potentiell användare”.

1.3 Your user stories in terms of using a standard pattern, acceptance criteria, task breakdown and effort estimation and how this influenced the way you worked and created value

Som tidigare nämnt började vi vår första sprint med väldigt lite erfarenhet vad gäller att arbeta med Scrum och skapa user stories. Detta resulterade i otydligt formulerade och estimerade user stories, vilket gjorde det väldigt svårt att efter sprinten ha något konkret att presentera. Under vår första review blev en väldigt låg andel av sprintens arbete färdigställt, samtidigt som de inte skapade något/lite kundvärde. Själva skapandet av user stories var en iterativ process vilket till en början gjorde att de skapades av olika personer vid olika tillfällen och blev skrivna med olika formuleringar. Detta påverkade det initiala arbetet i projektet ganska mycket eftersom det var svårare att arbeta med en user story där man själv inte varit delaktig i skapandet. Eftersom de user stories vi hade inte var tillräckligt tydliga uppstod det ett antal oklarheter gällande när en user story skulle räknas som klar. Vi hade arbetat en del med acceptanskriterier redan vid första sprinten men dessa var inte tillräckligt konkreta för att arbeta efter.

Efter första sprinten och tillhörande review bestämde vi oss för att i grupp sitta ned under vår sprint retrospective och definiera hur en user story skulle formuleras, hur vi skulle bryta ned dem i tasks samt vilka acceptanskriterier de skulle ha. Vi bestämde då att varje user story skulle definieras som:

> *“**As a** <particular class of user>, **I want to** <be able to perform/do something> **so that** <I get some form of value or benefit>”*

Vi såg till att varje user story dessutom bröts ner i mindre tasks för att simplifiera processen att arbeta med och färdigställa dem. Vid små user stories bestämde vi att det bara krävdes acceptanskriterier då deras titel i princip redan beskrev vad som skulle göras. För alla user

stories, oavsett storlek fokuserade vi på att skapa konkreta acceptanskriterier för att göra det enklare för gruppen att veta vad som skulle levereras.

Tack vare omformuleringen och nedbrytningen av user stories kunde vi i gruppen lättare förstå vad som skulle göras och då lättare skapa kundvärde i applikationen. Eftersom vi många gånger var fler än en person som arbetade på samma user story var det även väldigt bra att dela upp dem i mindre tasks då det möjliggjorde parallellt arbete.

När det kommer till effort-estimeringen har det under hela kursens gång varit väldigt svårt att uppskatta varje enskild user story korrekt, då vi trots en del erfarenhet inom kodning har haft svårt att förutspå hur stor insats som har krävts för att klara en viss user story. Dock har det varit bra att använda effort-estimeringar då det har gjort att vi fått tänka över varje user story en extra gång och kunnat planera bättre utifrån den velocityn som vi bestämde i teamet för varje enskild sprint. I de tidigare sprintarna så fick varje gruppmedlem estimera vilken effort en user story skulle ha och ett genomsnitt av allas åsikt blev det slutgiltgliga värdet. Längre in i projektet lät vi de som tidigare arbetat med det berörda området estimera vad den specifika user storyn hade för effort, medan resterande medlemmar antingen ifrågasatte resonemanget eller hade förtroende för uppskattningen. Det senare alternativet effektiviserade sprint planning och varje effort blev mer anpassad efter sin user story.

När det kommer till estimeringen valde vi från början att sätta en effort-enhet till en timmes arbete för en person för att det skulle vara lättare att relatera uppskattningen till verkligheten. Detta övergick under projektets gång till att i slutändan endast vara en estimation baserad på en fiktiv effort istället för verkliga timmar. Övergången skedde i takt med att det blev enklare för oss att estimera hur stor möda en user story har då vi kunde relatera till liknande uppgifter vi löst under tidigare sprintar. Den erfarenhet vi samlat på oss under kursens gång ledde till att vi lärt oss att bättre uppskatta varje user story i effort, samtidigt som vi var medvetna om vad vi som grupp klarade av att leverera under en sprint på en veckas längd. Detta går tydligt att se i KPI 1 där vår effektivitet stabiliserades kring 100% ju längre in på sprinten vi kom, med undantag från den sista sprinten där vi hamnade över 100% till följd av att det dök upp en del “sista minuten buggar” som behövde lösas samma sprint utöver det som planerats som sprintens arbete.

Något som uppmärksammades under projektets gång var att vi inte om-estimerade user stories som ej var avslutade mellan sprintar. En konsekvens av detta blev, bland annat, att vi i början av den nästföljande sprinten planerade in en högre velocity för att uppväga den förväntade färdigställningen av gamla user stories, utöver de ordinarie. Detta gjordes trots att den effort som krävdes för att göra färdigt arbetsuppgiften behövde mindre arbete än det stod i dess estimation, då den i många fall nästan var färdig. Genom diskussion med handledare insåg vi problematiken med detta och ändrade till de efterföljande sprintarna så att halvfärdiga user stories om-estimerades och visade en ackurat effort.

I ett framtida projekt skulle vi vilja fortsätta använda ett standardiserat mönster för att skapa våra user stories då det enligt oss var ett bra sätt för att hela tiden fokusera på kundvärde samtidigt som det effektiviserar processen i sig. Det standardiserade mönstret för skapandet av user stories har även gjort det lättare för hela teamet att förstå vad som ska göras, alltså inte endast för personen som utformade den, då alla förstår sig på det gemensamma mönstret. I kommande projekt kommer det hjälpa hela laget tidigt att vara på samma bana

från start och lättare komma in i det arbete som skall göras istället för att fastna på mindre missförstånd. Ett standardiserat mönster gör även att projektet blir mer effektivt och agilt eftersom det inte låser en uppgift till den enda personen/-erna som skapade user storyn, utan öppnar upp för att hela teamet ska kunna lösa problemet.

User stories med tillhörande, tydliga och konkreta, acceptanskriterier och tasks är något som definitivt kommer följa med till kommande projekt då de underlättar väldigt mycket arbete med väldigt lite möda. Acceptanskriterier i user stories gör det lättare för alla teammedlemmar att vara på samma spår gällande vad som anses vara godkänt för uppgiften. Tasks är på samma sätt som acceptanskriterier någonting som bör finnas med i varje user story, vilket vi märkte relativt tidigt i projektet när det visade sig vara mycket värdefullt. Inkluderandet av tasks skulle förenkla arbetet med user stories eftersom det ger mindre delmål än hela uppgiften som är lättare att arbeta mot, samtidigt som det ger en känsla av tillfredsställelse/framsteg varje gång en task blir avklarad.

För att få tasks att fungera ännu bättre i kommande projekt så skulle man kunna använda sig av “the hamburger method” som bygger på att bryta ner större user stories till många mindre stories och därefter i en iterativ process välja ut de viktigaste uppgifterna inom varje delområde.[2] Denna metod hade antagligen fungerat bra för oss i början av projektet då vi inte hade mycket erfarenhet av att bryta ned från större till mindre user stories. Dock avstod vi från “the hamburger method” då vi ansåg att den skulle ta lång tid att genomföra på varje user story, vilket hade fungerat bättre i ett större projekt där sprintarna var längre. Dessutom så har det under vårt projekt fungerat väl att bryta ned större user stories till mindre eftersom vi tidigt kände att det skulle vara lättare att lösa dem om varje user story endast tog upp ett delproblem istället för stora problem.

1.4 Acceptance tests, such as how they were performed, with whom, and which value they provided for you and the other stakeholders

Acceptanstester genomfördes i samband med testning enligt “definition of done”, där två team-medlemmar, utöver den som var ansvarig för user storyn, testade applikationen och såg till att de acceptanskriterier som definierats för user storyn är uppfyllda. Eftersom vi inte har skapat dessa acceptanskriterier tillsammans med någon potentiell kund har kriterierna baserats på vad vi tror att kunden velat ha.

I ett senare projekt är det önskvärt att skapa dessa acceptanskriterier tillsammans med en potentiell kund för att försäkra sig om att man genom uppfyllandet av acceptanskriterierna även uppfyller kundernas krav och önskemål.

1.5 The three KPIs you use for monitoring your progress and how you use them to improve your process

Vid val av KPI-värden ville vi ha något som kunde mäta både subjektiva och objektiva värden, exempelvis subjektivt välmående och kodkvalité samt objektiv beräkning av andelen avklarad effort. Därav valde vi att använda oss av de tre KPI:erna:

---

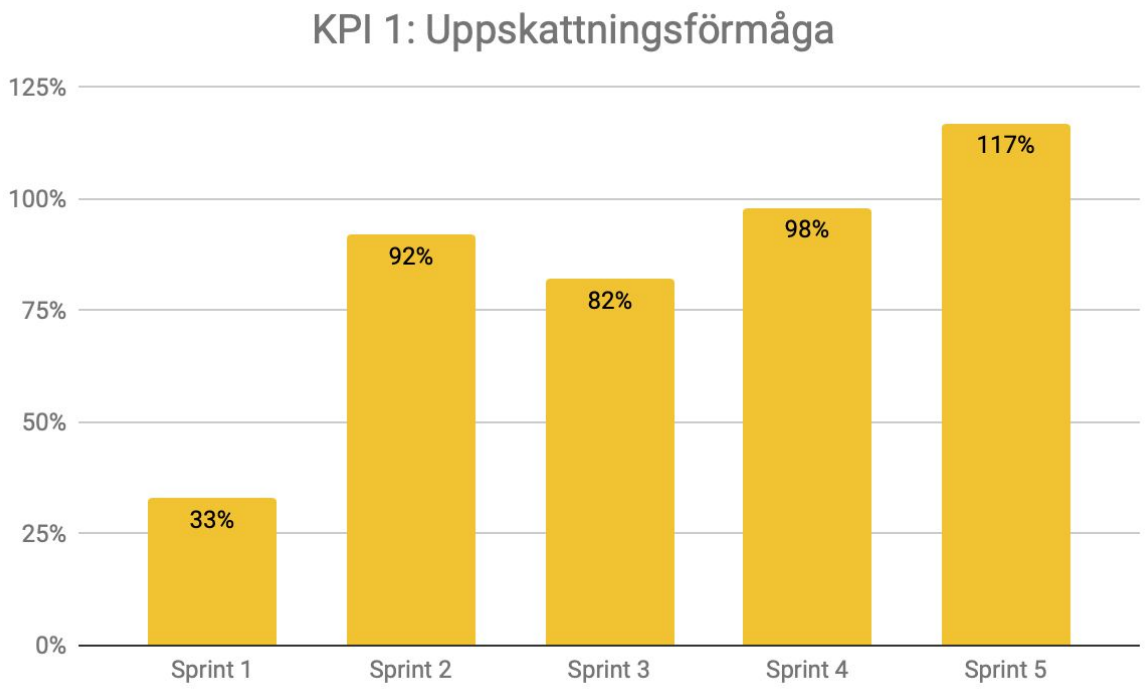
[2]Gojko Adzic, *Splitting user stories -- the hamburger method*, 2012, https://gojko.net/2012/01/23/splitting-user-stories-the-hamburger-method/, hämtad 29 maj 2019

1. Uppskattningsförmåga → Hur många procent av den effort som planerades för sprinten blev helt klart. Här räknas bara user stories kopplade till utvecklingen och inte effort för individual reflection, team reflection och så vidare.
2. Stress och motivation → Här får varje medlem ranka sin stress och motivation mellan 1-5. Här är 1 en låg stress/motivation och 5 hög stress/motivation.
3. Kodkvalité och grafisk design→ Här kollade alla medlemmar igenom koden samt såg över designen och fick sedan ranka dessa mellan 1-5.

Vid varje retrospective gick vi igenom och tog fram värden på våra KPIer. Efter detta jämförde vi dem mot tidigare sprintar för att se hur de hade utvecklats samtidigt som vi försökte använda oss av dem vid nästa sprint planning för att förbättra värdena.

KPI 1 Uppskattningsförmåga

Under de fem sprintar som genomförts har detta mått fått en tydlig förbättring från första sprinten vilket kan ses i Figur 2 nedan:



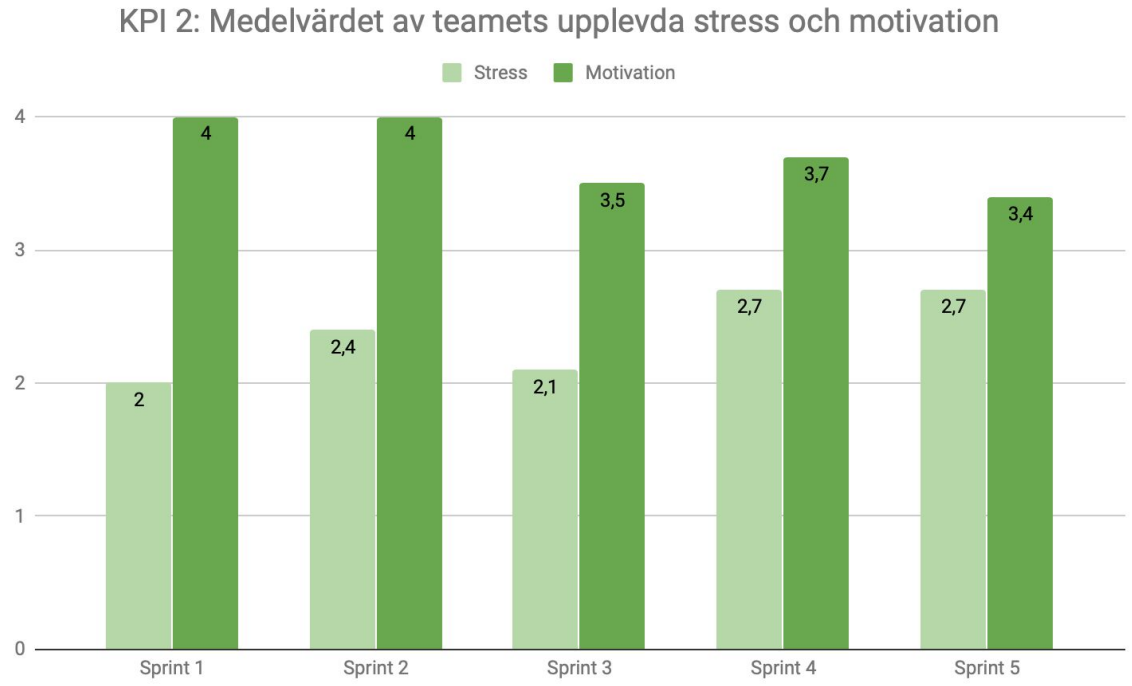
Figur 2: Diagram över KPI 1

KPI 1 var förhållandevis enkel att mäta då det var tydligt i scrum boarden vilka user stories som var klara, samt vilken effort de hade estimerats till. När vi mätte och analyserade KPI 1 försökte vi diskutera i gruppen vad en låg respektive hög procentsats berodde på och detta värde kunde även kopplas till vårt tredje mål om att arbeta effektivt mot deadlines. Vid första sprinten när värdet var väldigt lågt kom vi fram till att det berodde på otydligt definierade user stories som inte var optimalt uppbyggda för en agil arbetsprocess. Utifrån detta värde kunde vi komma fram till förbättringsåtgärder som vi ville arbeta utifrån vid nästa sprint planning då vi började använda oss av den tydliga mallen för user stories som beskrevs ovan.

Enligt Henrik Kniberg och hans Lean & Agile Slides bör man fokusera på värde och inte output och detta är något vi borde reflektera över kring KPI 1, då det endast mäter output i form av effort.[3] Han nämner även att “What you measure is what you get” vilket visar ytterligare på betydelsen av att mäta värde och inte effort. Ett bättre mått för detta hade varit att mäta vilket värde varje user story som avklarats har. Detta hade kunnat göras genom att man för varje user story kollar på dess kundvärde och ger det en siffra mellan exempelvis 1-10 för att vid slutet av sprinten räkna ihop hur mycket värde som teamet har lyckats skapa.

KPI 2 Stress och Motivation

Stress och motivationsnivån har varierat för varje individ mellan de olika sprintarna men som figuren visar har teamets medelvärde av de båda måtten ändå hållit sig någorlunda stabilt över hela projektet. Motivationen för projektet har minskat en aning vilket främst har berott på att buggar uppkommit samt att vissa medlemmar inte känns sig så stimulerade av sina user stories.



Figur 3: Diagram över KPI 2

KPI 2 var svår att endast koppla till projektet då många kände sig stressade av andra anledningar än projektets arbetsbelastning. Dock ansåg vi att det måttet var viktigt då skattningen bjöd in till diskussion gällande hur gruppens medlemmar mår vilket sedan kunde tas i beaktning under kommande sprint. Eftersom stressnivån skiljde sig väldigt mycket från sprint till sprint bland gruppens medlemmar var det svårt att använda måtten för framtida planering då en stressad sprint oftast följdes av en mindre stressad sprint. Vid tillfällen då

[3] Henrik Kniberg, *Henrik’s Lean & Agile slides*

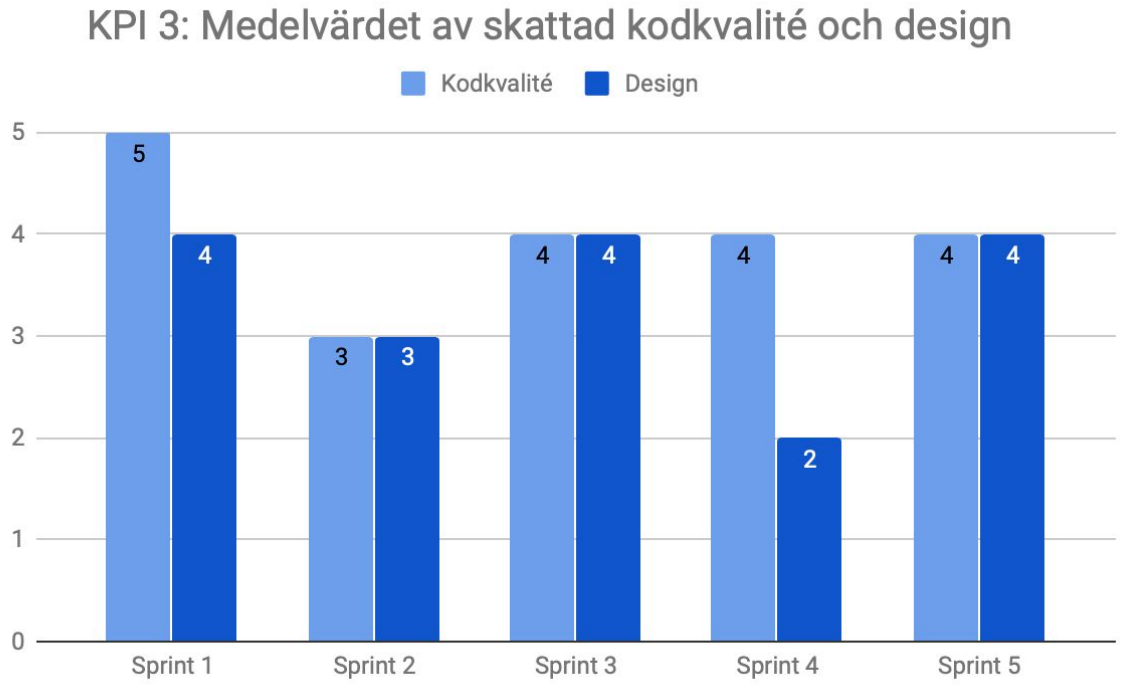
stressen visat sig komma från projektets arbetsbelastning tog vi dock med detta i åtanke när vi genomförde vår sprint planning då vi försökte anpassa teamets velocity utefter detta.

Även motivationen var viktig att mäta då en hög motivation skapar bättre förutsättningar för ett bra genomfört arbete. Motivationens nedgång berodde främst på buggar som uppkom under utvecklingens gång vilket även är svårt att undvika. För att detta måttet ska vara användbart kan det även i början av ett projekt vara bra att alla får berätta vad som motiverar dem i dessa typer av projekt. Eftersom alla inte motiveras av samma sak är detta viktigt att veta och det är något som vi inte gjorde detta projektet vilket vi tar med oss till ett framtida projekt. När motivationen var låg bland vissa försökte vi höja denna genom att vid sprint planning tilldela alla sådana uppgifter som de tyckte var roliga att arbeta med.

Ett annat KPI som kan vara av användning till framtida projekt är att se hur nöjd teamet är när det kommer till projektet i sin helhet. Beroende på hur nöjda teamet är så kan det påverka deras motivation vilket i sin tur kan komma att påverka följande sprintar. En regelbunden utvärdering av hur nöjd teamet är kan ge tidiga varningstecken för problem inom gruppen som kan åtgärdas innan det påverkar arbetet.

KPI 3 Kodkvalité och grafisk design

Kodkvalitén och den grafiska designen har skattats enligt figuren nedan. Vad som bör nämnas är att det första sprinten endast fanns en enda user story som var klar som kunde bli bedömd vilket ledde till de höga värdena där.



Figur 4: Diagram över KPI 3

Detta KPI var svårmätt och baserades främst på medlemmarnas subjektiva åsikt. Åsikten baseras dessutom på utvecklarnas egna kunskaper inom kodkvalité och om en annan mer erfaren utvecklare hade genomfört samma bedömning hade den troligtvis sett helt

annorlunda ut. KPI:et har tyvärr inte hjälpt oss lika mycket som vi önskat då det var svårt att bestämma vad som anses vara bra i ett objektivt perspektiv och ge oss en klar bild på den nuvarande kvaliteten på projektet.

En tanke som har slagit oss så här i slutet av projektet är att vi från start kanske inte borde ha använt oss av ett KPI som mätte den grafiska designen då vi framförallt fokuserade på funktionalitet och därför medvetet nedprioriterade designen under de första sprintarna. Det kan dock ändå fortfarande vara intressant att mäta vilken nivå som den grafiska designen anses uppnå då det ger en indikation på hur väl designen motsvarar eventuella förväntningar. Samtidigt känns det lite motsägelsefullt att mäta något som det går och som man vill förbättra men ändå inte gör det under en längre tid.

I ett framtida projekt är det önskvärt att komplettera ett liknande KPI med någon form av objektiv bedömning. Detta skulle kunna var genom att använda verktyg såsom FindBugs eller andra testnings-bibliotek. Det ger möjligheten att automatisera arbete och hitta problem som kan ha missats under manuell genomgång av kod-kvaliteten. Vi fick tips av handledarna att använda specifikt FindBugs vid sprint fyras review men vi bestämde oss att inte använda detta verktyg då ingen i gruppen var bekant med detta samtidigt som vi endast hade en sprint kvar och ville vara konsekventa över alla sprintar. Till andra projekt bör man tänka på ifall man vill inkludera sådana verktyg, då man potentiellt inte behöver lägga lika mycket tid på manuell genomgång av koden som vi har gjort.

Till nästa projekt kan vi även tänka på hur man kan bestämma KPI 3 bättre så att den reflekterar kvaliteten på specifika deliverables samt kvaliteten på projektet i helhet. På det sättet kan man bibehålla en hög standard på kod och design genom projektet och i slutändan kunna leverera en produkt med hög standard. Vi hade exempelvis kunnat implementera detta i det nuvarande projektet genom att utöver testning göra en uppskattning på kvaliteten av den nya koden för varje user story samt koden i helhet. Då kan man se om den nya koden är funktionell samt bedöma ifall koden håller samma standard som resten av projektet. För att göra detta kan det vara bra att använda en sprint gren för varje sprint för att kunna jämföra en specifik sprints arbete med en annan sprint och på så sätt kunna jämföra kodkvalitet. En sprint gren skulle kunna ses som en slags release gren där en sammanslagning först görs med en develop gren som i sin tur sammanfogas med master-grenen. Dessutom skulle vi även ha kunnat använda oss av en gren för hotfixes med syftet att snabbt kunna åtgärda eventuella buggar som följt med till mastern, samtidigt som utvecklingen inför den nya releasen kan pågå parallellt. Att använda en sådan struktur hade troligtvis gjort det lättare att jämföra kodkvalitet och grafisk design mellan sprintarna, jämfört med att sammanfoga feature grenar direkt in i mastern som vi har gjort i det här projektet.

2 Social Contract and Effort

2.1 Your social contract, i.e., the rules that define how you work together as a team, how it influenced your work, and how it evolved during the project (this means, of course, you should create one in the first week and continuously update it when the need arrives)

Det sociala kontraktet skapades tidigt i kursen. Detta var bra då några gruppmedlemmar inte kände varandra sedan tidigare och inte hade arbetat med varandra i grupp. Här specificerade vi exempelvis hur vi skulle hantera sena ankomster och hur arbetsfördelningen skulle se ut samt hur vi skulle förhålla oss till de olika Scrumrelaterade aktiviteterna som sprint planning, review, retrospective och daily Scrum. Vi specificerade även vad vi hade för betygsmål samt hur mycket tid man ville lägga på kursen.

Anledningen till att vi specificerade vad vi hade för betygsmål samt hur mycket man skulle kunna tänka sig jobba med projektet berodde främst på att sätta realistiska förväntningar på varandra och undvika konflikter.

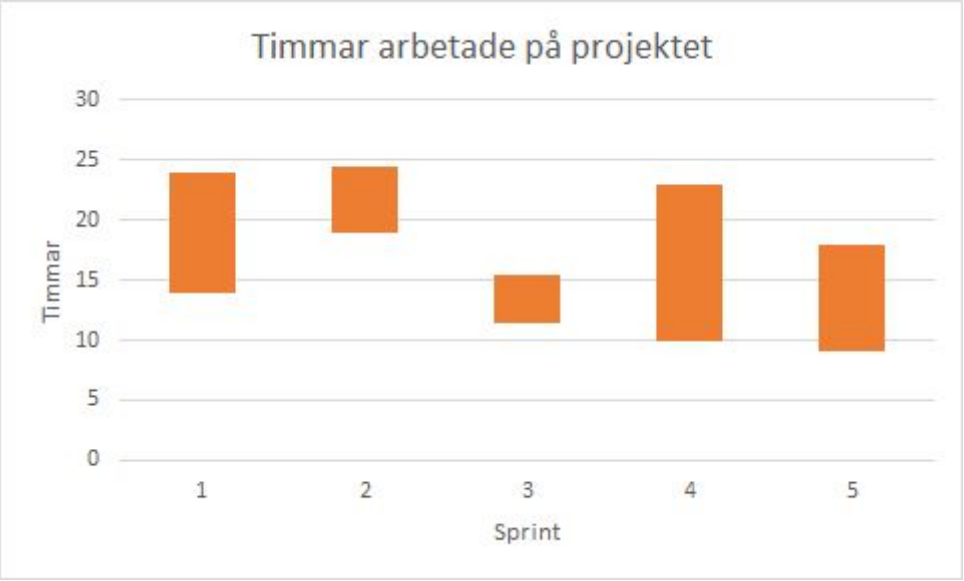
Det sociala kontraktet har fungerat med varierande resultat under de fem sprintar som varit och har följts till viss del. Det som fungerat sämst är de sena ankomsterna som varit väldigt frekventa. Vi insåg dock tidigt att det inte fungerade med något slags “pricksystem” för att få folk att komma i tid och valde därför att ta ur detta ur det sociala kontraktet. Vi försökte istället trycka på vikten av hur en sen ankomst påverkar arbetet i gruppen för samtliga inblandade, vilket förbättrade situationen något. Utöver detta så har det sociala kontraktet inte behövts användas för att lösa några dispyter eller andra problem i gruppen, eftersom sammanhållningen inom gruppen varit god. Bortsett från de sena ankomsterna så har kontraktets riktlinjer följts.

Även om det sociala kontraktet har fungerat väl finns såklart alltid saker att förbättra. Trots att vi kontinuerligt sett över vårt sociala kontrakt vid gruppens retrospective borde vi haft en diskussion kring hur man tycker de andra sköter det sociala kontraktet och hur man skulle hantera ett fall då någon av gruppmedlemmarna bryter mot detta.

Något som vi märkte den näst sista sprinten var de problem som kan uppstå när mycket kod från flera olika grenar ska sammanfogas med mastern i slutet av sprinten. En väsentlig mängd tid fick läggas på att lösa “merge conflicts” på måndagen, tid som vi hellre hade velat lägga på att få handledning eller påbörja veckans retrospective. När detta diskuterades med Håkan tryckte han på vikten av små grenar med små förändringar som sammanslås ofta då dessa oftast är mycket enklare att lösa snabbt och smärtfritt, vilket vi fokuserade mer på den nästkommande sprinten med goda resultat. Detta är emellertid något som borde specificerats i det sociala kontraktet, samt att man först bör ta in mastern till sin gren för att testa grenen innan man för in den i mastern.

2.2 The time you have spent on the course and how it relates to what you delivered (so keep track of your hours so you can describe the current situation)

Som nämnt ovan fick alla initialt i kursen specificera i det sociala kontraktet hur mycket tid man kunde tänka sig att lägga på kursen. Här ville fem av sju gruppmedlemmar lägga “mycket tid” på projektet medan resterande två gruppmedlemmar ville lägga “väldigt mycket tid”. Utifrån detta hade vi alltså ambitioner att lägga förhållandevis mycket tid på projektet vilket även märktes då vi under varje retrospective gick igenom antalet timmar som oftast låg runt 20h/vecka för de flesta.



Figur 5: Diagram över spannet antalet timmar arbetade av gruppens medlemmar

Som väntat lades det initialt i kursen mycket tid åt sprint planning, review och retrospective, vilket främst berodde på att det var relativt nya begrepp för många av oss. Samtidigt lades också mycket tid under de första två sprintarna på att lära sig några av de nya program som skulle användas, detta gällde särskilt för de som inte hade arbetat med GitHub och Android Studio. Utöver det hade vi första sprinten fokuserat alldeles för lite på kundvärdet och skapat för horisontella user stories vilket ledde till att den tid som lades på en del av utvecklingen av applikationen var bortslösad, då vissa funktioner inte ens kunde användas senare i utvecklingen.

Något som är värt att notera är att den nedlagda tiden inte nödvändigtvis är direkt kopplad till kvaliteten eller kvantiteten av det gruppen levererat varje vecka. De första sprintarna lades mycket tid ner på att lära sig bli effektiva i de program som vi arbetat i och lära sig hantera de vanligt förekommande problemen som uppstod. Det var även enligt handledarna tänkt att fyra timmar skulle vara tillräckligt med tid för gruppen att sköta allt det administrativa (till exempel sprint planning, retrospective och review), medan vi i själva verket de första sprintarna la mer än dubbelt så mycket tid på det.

Trots att vi såg värdet i och tyckte att sprint planning och retrospective var givande ansåg vi även att det tog upp en för stor andel av de timmar man förväntades lägga per vecka på

projektet. Vi tror däremot att samma upplägg hade fungerat bättre till en normal arbetsvecka på 40 timmar, eller ifall man kunnat arbeta effektivare med det administrativa, och således ha mer tid över till det faktiska utvecklandet av projektet. Detta är emellertid något vi kände att vi åstadkom för varje sprint som gick, och något vi hade velat fokusera mer på i nästa projekt.

3 Design decisions and product structure

3.1 How your design decisions (e.g., choice of APIs, architecture patterns, behaviour) support customer value

För att få fram en karta har vi använt Google Maps då det är ett för användarna väletablerat gränssnitt samtidigt som dess API är enkelt att implementera. Det har tyvärr varit omöjligt att testa i Android Studios inbyggda emulator (emulatorn hade inte tillräckligt mycket RAM och samtidigt tillgång till implementationen av google-play-services) och har istället krävt en fysisk android-telefon för att testa. Det har inneburit en del komplikationer, men i och med API:ets smidighet skulle vi förmodligen göra samma val i nästa projekt, även om man borde ha undersökt mer om hur man löser en fungerande emulator.

För att strukturera upp vår android applikation har vi använt Model-View-ViewModel (MVVM), som har använts tidigare av gruppmedlemmar, vilket gjorde det lättare att implementera. För android applikationer utmärker sig MVVM som en av de starkare strukturerna och genom att strukturera upp kod blir det lättare för oss att fortsätta utveckling av applikationen och det blir lättare för utomstående utvecklare att sätta sig in i kodbasen. I och med att det inte finns särskilt många bra alternativ skulle vi förmodligen köra på samma struktur ifall vi hade haft chansen att välja igen.

För att implementera en databas ansåg vi Firebase var det smidigaste verktyget då det är gratis (beroende på antalet användare), användarvänligt och väldokumenterat. Dessutom har det många tillägg och användbara kopplingar till bland annat Android Studio och reklam, som kan dra in pengar för fortsatt utveckling av applikationen. Ifall man skulle göra ett större projekt skulle vi behöva göra en kostnadskalkyl på vad liknande tjänster skulle kosta, relativt Firebase och eventuellt egen server, och sedan göra en avvägning mellan enkelheten med Firebase och eventuell kostnad.

Vi valde att utveckla en mobilapplikation istället för en webbapplikation (eller datorprogram) för att vi inte såg något större kundvärde för våra användare att kunna starta applikationen på andra medier än mobilen. Vidare valde vi att utveckla för operativsystemet Android i och med att majoriteten av gruppmedlemmar har androidenheter, vilket gör det lättare att testa applikationen. Dessutom har Android fler användare världen över, vilket ger oss en större kundbas och dess omfattande stöd för utveckling i Java gör det lättare för oss att utveckla, då det är det programspråk de flesta av oss är mest bekanta med.

3.2 Which technical documentation you use and why (e.g. use cases, interaction diagrams, class diagrams, domain models or component diagrams, text documents)

Vi valde att använda Trello som Scrum board för att det är ett enkelt gratisverktyg som flera gruppmedlemmar använt tidigare. Med tiden kom vi att uppskatta Trello mer och mer, både som verktyg att sätta upp sprint backloggen och som ett kommunikationsmedium kring individuella user stories. Vidare hjälpte det oss att få struktur på den agila processen och har varit ett väldigt bra hjälpmedel. En förbättringsmöjlighet hade dock varit att trycka mer på att utveckla en standardiserad dokumentation i Trello vilket är något vi tar med oss till ett framtida projekt.

För att dokumentera koden har vi använt oss av JavaDoc, då det är standard för all dokumentation av Java-kod, och det har gjort det lättare för oss att använda och förstå varandras kod. Vi hade dock kunnat vara bättre på att implementera JavaDoc korrekt och standardisera hur man skall skriva kommentarer, samt hur utförligt man skall dokumentera.

Vi använde oss inte av ytterligare dokumentation för strukturen, såsom UML-diagram eller domänmodeller, då vi inte ansåg att det skulle vara värt det. I ett större projekt hade det dock kunnat vara ett användbart verktyg för att få en bättre överblick av projektets struktur.

3.3 How you use and update your documentation throughout the sprints

Trello användes för att man skulle se hur långt vi kommit på olika user stories genom att se vilken kolumn korten låg i, samt kolla checklistor för tasks och acceptanskriterier. Genom möjligheten att länka korten till ansvarig person/personer var det lättare att hålla koll på vem som ansvarade för vad under respektive sprint. Användningen av checklistor underlättade för att följa upp hur långt man hade kommit med varje user story. Vid testning användes Trello genom att man lätt kunde gå in på korten för olika user stories och se dess acceptanskriterier, som visade vad som skulle fungera. I slutet lade vi även till kommentarer kring vilka klasser man hade ändrat i, vilket gjorde att man som testare kunde se vart man skulle kolla.

Under projektets gång utvecklade vi även vår användning av Trello genom att vi började skriva user stories för buggar till vår product backlog och sprint backlog under sprintar. Vidare började vi även omprioritera buggar under sprintens gång för att få ut så mycket kundvärde som möjligt i varje iteration. I ett framtida projekt skulle vi vilja vara bättre på att snabbare flytta korten till rätt lista och kryssa i checklistor, så att det syns tydligare vilket stadie som varje user story befinner sig i och därmed ge oss en bättre överblick av var fokuset för sprinten bör läggas. I slutet av projektet började vi som tidigare nämnt att lägga till kommentarer om vilka klasser som man hade ändrat i, vilket vi ser ett värde i att göra i ett framtida projekt, då det inte bara underlättade vid testning utan också då det kan ses som ett enkelt sätt att dokumentera var och när ändringar har skett under projektets gång.

Vi har försökt se till att man skriver JavaDoc för alla nya metoder man skapar innan grenen slås ihop med master, men vi hade däremot kunnat vara bättre på att uppdatera kommentarerna i koden när man ändrar på metoder. Ett enkelt sätt att bli bättre på det i ett framtida projekt skulle vara att alltid ta för vana att gå igenom samtliga metoder som man

har ändrat i, en sista gång för att se över att alla kommenterar fortfarande är aktuella och att det inte saknas kommentarer för de förändringar som har genomförts.

3.4 How you ensure code quality and enforce coding standards

Initialt i kursen, innan vår första sprint, bestämde vi en “definition of done”. Vi bestämde då att på grund av den grafiska naturen av applikationen köra på peer-to-peer testing, och för att få flytta en user story till “done” i vår Scrum board skulle minst två personer i gruppen testat och godkänt funktionaliteten för att se om den fungerade. Vi resonerade att detta skapade värde för våra kunder i form av att vi såg till att funktionaliteten fungerade som tänkt på flera olika enheter.

Detta fungerade mindre bra under första sprinten på grund av vårt breda scope och våra otydliga user stories, vilket gjorde att alla fokuserade på att arbeta med sitt egna och hade inte tid att testa andras user stories. Granskningen av koden och dokumentationen fungerade inte heller särskilt bra i början av projektet eftersom den “definition of done” som vi satt upp inte tydligt beskrev det som testarna faktiskt skulle göra. Vid senare sprintar insåg vi att vi borde skärpa våra test punkter och även fokusera mer på kodkvaliteten och dokumentationen av koden. Detta var delvis underförstått vid första definitionen men för att vi skulle ha tydliga restriktioner för dem som testar och för att alla test skulle vara lika strikta förtydligade vi detta i vår “definition of done”. I samband med testen gick man som testare även igenom koden för att kontrollera att den upprätthöll riktlinjer samt såg till att den var tillräckligt dokumenterad så att man som oinsatt förstod all funktionalitet. Innan testningen var klar skulle man även försäkra sig om att alla acceptanskriterier i user storyn var uppfyllda. Detta gjorde att det blev lättare att bedöma kodkvaliteten för varje user story, och därmed blev det lättare att bedöma kodkvaliteten för hela projektet (KPI 3).

Vid senare sprintar blev vi även bättre på att prioritera testing av färdiga user stories och testningen underlättades även av att vi satt och arbetade mycket tillsammans, vilket gjorde det enklare att utföra snabba tester på varandras user stories.

Utöver dessa tester bedömdes även koden då vi mätte KPI 3 (kodkvalitet och grafisk design) vid varje retrospective, där alla i teamet fick betygsätta vad som levererats på en skala 1-5. Detta tycker vi nu i efterhand är alldeles för subjektivt, och att bedömningen hellre skulle göras utifrån mätbara mått med hjälp av till exempel FindBugs eller Travis och att den gemensamma granskningen snarare skulle ses som ett komplement till dessa. Att dessutom implementera standardiserade tester där vi satt upp tydliga riktlinjer kring hur testningen skall gå tillväga, samt implementera grafiska testverktyg skulle också hjälpa oss vara mer objektiva vid testning.

En förbättrad variant av den peer-to-peer testing vi haft under projektets gång hade varit att ta in en oberoende part som potentiell användare och låta denne testa funktionaliteten. På så vis hade man fått en utomståendes perspektiv vilket kan vara av stort värde.

För att tydligare demonstrera vad man skrivit och lättare se över koden i samband med testning, hade det varit värt att testa att begära pull requests, något som vi valde att skippa i och med gruppens låga erfarenhet inom Git.

I framtida projekt skulle man antagligen kunna vara mer strikt vid testning av koden än vi varit under detta projekt. Vi var inte hårda med bedömningen av vår kod för att de som gjorde testerna ville göra det så fort som möjligt, för att kunna återgå till sina egna arbeten. En lösning för detta skulle kunna vara att avsätta lite effort i velocityn enbart för testning för att försäkra sig om att det finns tid att göra detta arbete. En annan lösning skulle kunna vara att man planerade in möten endast till för testning så att man gör det tillsammans. Detta skulle leda till att alla user stories blir bedömda på samma sätt och att gruppmedlemmarna lägger lika mycket tid på att testa koden och läsa igenom dokumentationen, samt skapar en djupare förståelse av koden för samtliga gruppmedlemmar.

4 Application of Scrum

4.1 The roles you have used within the team and their impact on your work

Eftersom flera i gruppen tidigare aldrig arbetat med Scrum valde vi till en början att applicera så mycket som möjligt av det ramverk som Scrum tillhandahåller och som kunde appliceras på vårt projekt. Därför valde vi även att första sprinten tillsätta en gruppmedlem som Scrum master och en som product owner. Under projektets gång innebar rollen som Scrum master främst att påminna Scrum-teamet om att arbeta agilt, genom exempelvis daily Scrum meetings. Detta ansåg gruppen fungera bra första sprintarna vilket gjorde att vi behöll en Scrum master i gruppen. De första sprintarna var det dock oklart vad rollen som product owner innebar och vilken funktion den skulle ha i teamet. Därför valde vi efter sprint två under dess retrospective att definiera exakt vilka uppgifter de båda rollerna skulle ha under kommande sprintar, vilka presenteras i Tabell 1 nedan.

Scrum master	Product owner
• Se till att teamet arbetar agilt genom att styra upp daily Scrum meetings • Se till att grupprum bokas • Styra upp reviewen och retrospective	• Ha en helhetsbild av teamets vision och se till att Scrum-teamet håller sig inom den och inte tappar fokus på detta vid sprint planning

Tabell 1: arbetsrollerna för Scrum master och product owner

Dessa riktlinjer gjorde att rollernas värdeskapande för gruppen blev tydligare och därav fortsatte vi använda oss av dessa roller. Vid den sista sprinten var det sällan hela gruppen kunde ses tillsammans och då nedprioriterades dessa roller, eftersom samtliga medlemmar var tvungna att fokusera mycket på att färdigställa de user stories vi planerat innan presentationen.

I ett framtida projekt hade det varit bra om den som har rollen som product owner inte är en del av själva projektgruppen, då det är lättare för en sådan part att ge oberoende kritik på det som levererats. En extern product owner skulle troligtvis ha kunnat kommunicera en mer utvecklad vision om vad som önskas av applikationen och haft möjlighet till att fokusera mer på att arbeta med stakeholders medan scrum-teamet fokuserar på utvecklingen. Samtidigt finns det en fördel i att en product owner inte är för extern då det är viktigt att den är engagerad och tillgänglig för Scrum-teamet. Nyckelfaktorn är att den som har rollen som product owner har en vision för det som efterfrågas och även om vi har haft en som haft rollen som product owner känner vi att det skulle ha kunnat utvecklats en ännu tydligare vision.

Vi ser även ett värde av att definiera rollerna i ett inledande skede i projektet och sen vid varje retrospective eventuellt ändra dessa rollbeskrivningar vid behov. Detta för att rollerna bör anpassas utifrån det behov gruppen har, vilket kan vara svårt att veta i ett för tidigt stadie.

4.2 The agile practices you have used and their impact on your work

De agila practices som vi utgått ifrån vid denna fråga är den lista som finns under List of Best Agile Practices[4] samt utifrån föreläsning fem. De agila practices som vi använt oss utav presenteras nedan.

1. Iterationer / Sprintar

Under projektets gång har vi arbetat i sprintar som varit en vecka långa. Innan dessa sprintarna så har vi genomfört en sprint planning där vi bestämt teamets Velocity som främst baserats på en känsla av hur mycket tid teamet skulle kunna tänkas lägga. Vid ett framtida projekt hade det varit önskvärt att ha bättre koll på hur mycket tid teamet kan lägga. Under projektets gång har det varit svårt att veta hur mycket tid varje gruppmedlem kan lägga varje vecka och därför har det varit svårt att planera för hela teamet. Vid ett framtida projekt vill vi även basera velocityn utifrån KPI 2 som handlar om teamets stress i en högre grad än vad som gjorts i detta projekt. Utifrån denna velocity har vi sedan planerat in user stories som ska genomföras under sprinten. Hur många user stories som planerats in har berott på hur stor effort de har estimerats till. Genom att arbeta i sprintar har vi även levererat värde mer frekvent, vilket Henrik Kniberg nämner är viktigt då han menar på att “Delivery frequency = Speed of learning” och att planeringen underlättas av frekventa releases.[5]

Inför varje sprint genomfördes en sprint planning där de user stories som var högst prioriterade i backloggen lades in i en sprint backlog där de sedan estimerades. Sprint planning var en väsentlig del för att vi skulle kunna planera och prioritera vår sprint backlog, vilket sedan lade grunden för hela sprintens arbete. Denna planering tog väldigt lång tid under de första sprintarna, speciellt då vi efter första sprinten fick se över alla de user stories som vi skapat för att de skulle ha ett större och tydligare fokus på kundvärde. Därefter blev vi hela tiden bättre på att genomföra vår sprint planning vilket ledde till att mer tid kunde läggas

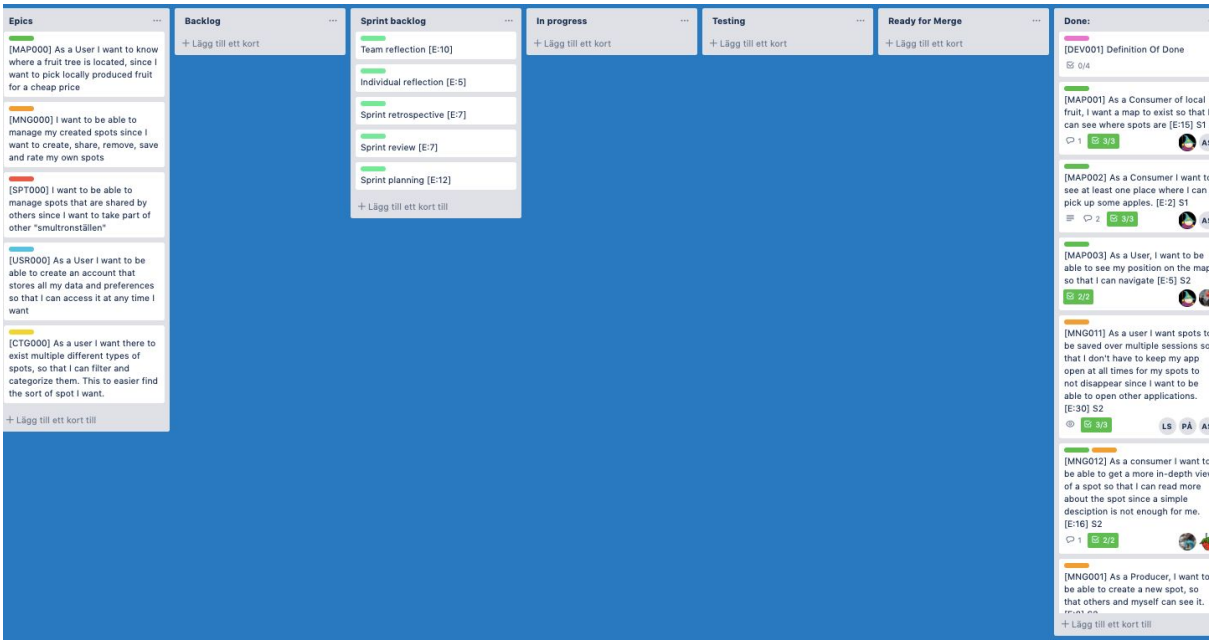
[4] Pavel Kukhnavets, *List of the Best Agile Practices*, 2018, https://hygger.io/blog/list-best-agile-practices/, hämtad 29 maj 2019.
[5] Henrik Kniberg, *Henrik’s Lean & Agile slides*

på utvecklingen. Vid ett framtida projekt vill vi ha tydligare mål för varje sprint som speglar något konkret som vi vill kunna göra med applikationen. Utöver det vill vi även se till att lägga in buggar i sprint backloggen och prioritera den så att buggarna åtgärdas så snabbt som möjligt för att vidare utveckling ska vara möjlig.

Genom att arbeta i sprintar har vi kunnat arbeta mer agilt och vara mer flexibla vid förändringar. Vid ett framtida projekt hade det varit önskvärt att arbeta i lite längre sprintar, då vi under dessa sprintar har lagt en stor andel av vår tid på review, retrospective och sprint planning och därmed mindre tid på utveckling av applikationen. En förutsättning för att man ska kunna jobba i längre sprintar är dock att det är ett större projekt.

2. Product backlog in a Scrum Board

Under projektets gång har vi även arbetat med en typ av product backlog. Anledningen till att det inte ses som en riktigt product backlog är för att vi inte haft en riktig product owner som har “ägt” denna backlog. Backloggen har legat i vår Scrum board som visas nedan:



Figur 6: Projektets scrum board som skapats i Trello

- Epics: Större övergripande beskrivning av de funktioner vi ville att applikationen skulle innehålla. Här försökte vi även prioritera de epics som vi ansåg var viktigast och som innehöll vår främsta value proposition.
- Backlog: Här lades de user stories som vi skapade utifrån våra epics. För att se vilken epic varje user story var kopplad till märktes dessa med etiketter som representerades av respektive epic. Backloggen prioriterades utefter de user stories som vi ansåg vara viktigast.
- Sprint backlog: Här placerades de user stories som planerats för den specifika sprinten och i samband med detta estimerades dem.
- In progress: När en user story påbörjats flyttades den hit.

- Testing: När user storyn ansågs vara klar enligt den/de som genomfört den flyttas den hit då den är redo för att testas av två andra teammedlemmar.
- Ready for merge: När testningen var klar och godkänd flyttades user storyn hit då den ansågs redo att slås ihop med mastern.
- Done: När user storyn placerades här ansågs den uppfylla “definition of done” och denna user story kan nu även demonstreras vid en review.

Backloggen har underlättat vår utveckling då den har hjälpt oss ha en bra översikt över vad som ska göras och hjälpt oss planera och prioritera de user stories som är viktigast utifrån kundvärdet och vår huvudsakliga value proposition.

Scrum Boarden har även hjälpt oss ha en bra översikt över hur det går för alla user stories som har planerats för en sprint genom att de flyttas mellan de olika kolumnerna beroende på deras utvecklingsstatus.

3. Daily Scrum meetings

Redan från start började vi med daily Scrum meetings där vi besvarade följande tre frågor:

- Vad gjorde du igår?
- Vad ska du göra idag?
- Har du stött på något hinder som du behöver hjälp med?

Dessa daily Scrum meetings bestämde vi skulle hållas varje vardag och om vi inte sågs fysiskt i grupp skulle mötena ske via en egen slack kanal som endast innehöll daily Scrums. Våra daily Scrum meetings följdes i varierad grad och missades ibland på grund av att vi skulle ses, men på grund av sena ankomster och schemakrockar visste vi inte hur vi skulle gå till väga. Vi hade inte heller definierat tydliga riktlinjer för dessa möten som exempelvis hur de skulle genomföras om tre av sju gruppmedlemmar kunde komma på mötet medan resten var hemma. I dessa fall blev det ofta så att de som var på mötet körde ett daily Scrum meeting vilket i princip tog bort hela poängen med daily Scrum då det syftar till att hela gruppen ska delta.

Även under den långa sprinten som sträckte sig över ett helt lov slarvades det med daily Scrum meetings på grund av att vi egentligen inte skulle arbeta under lovet. Det som gruppen dock konstaterade efter detta var att det vore bra att ha daily Scrum meetings i slack kanalen oftare. Detta för att hela gruppen ska ha koll på vad som görs samt kunna få hjälp om problem uppstår. Under andra sprinten (med lovet) var det även många som kände sig stressade över projektet just på grund av att de hade dålig koll på vad folk arbetade med, om de arbetade och hur mycket de arbetade.

Som ett komplement till den sista frågan i vår daily Scrum, som handlar om huruvida någon behöver hjälp eller inte, upprättades en egen slackkanal som vi kallade “kodhjälpen”. Här kunde man skriva när som helst om man stötte på problem gällande utvecklingen av applikationen. Denna kanal ansågs viktig då det gjorde det tydligare att personen behövde hjälp av en gruppmedlem, vilket innebar snabbare respons.

Vid ett framtida projekt vill vi fortsätta utveckla dessa daily Scrum meetings då vi trots till viss del bristfälligt genomförande av dem såg ett stort värde i denna typ av kontinuerlig kontakt. Vid ett senare projekt vill vi ha tydligare riktlinjer för hur och när dessa daily Scrum meeting ska genomföras. Vi skulle i största möjliga mån vilja genomföra dem face-to-face varje vardagsmorgon under 5-15 minuter då det blir mycket effektivare. Att genomföra dessa möten via en slack kanal skulle vi helst vilja undvika då det är betydligt jobbigare att läsa samt svårare att formulera sig via text. Ett alternativ är att man kopplar upp sig via en videolänk och kör mötena där vid dagar man inte ses för att ändå få en typ av face-to-face kommunikation. Betydelsen av face-to-face kommunikation är även något som Henrik Kniberg trycker på i sina Lean & Agile Slides. [6]

4. Review / Sprint demo meetings

Vid slutet av varje sprint hade vi en liten review i form av handledning med någon av handledarna. Under denna review gick vi igenom vad vi hade skapat för värde i form av de user stories som blivit färdigställda och handledaren fick kolla igenom appen själv och ge feedback på dess funktion och grafiska design. Denna feedback använde vi väldigt mycket när vi sen planerade för nästa sprint. Utöver review för handledare visade vi även upp de user stories som var klara internt för gruppen så att alla i gruppen kunde ge feedback.

Om vi hade gjort ett nytt liknande projekt hade vi gärna haft reviewen med en riktig product owner för projektet, samt andra typer av stakeholders som kan tänkas vara intresserade av applikationen. Detta hade gjort det mer verklighetsförankrat och det hade underlättat för att säkerställa sig om att det man gör verkligen levererar kundvärde. När man inte har en riktig Stakeholder/potentiell användare som deltar i reviewen kan det vara svårare att veta vad som bör prioriteras då man bara kan spekulera i vad som verkligen skapar kundvärde. Med flera riktiga Stakeholders är det mer troligt att aspekter som man annars hade missat verkligen tas upp under en review.

5. Retrospective

Efter varje review höll vi även en retrospective som innebar att skriva en team reflection där vi reflekterade över vår arbetsprocess. Detta fungerade bra under främst de två första sprintarna då vi tog med oss viktiga insikter som vi sedan applicerade på vår arbetsprocess nästa sprint. Exempel på detta är att vi tydligare definierade rollbeskrivningar för de agila rollerna vi använde, samt vår “definition of done”.

Vid senare sprintar tyckte vi att denna typ av retrospective inte gav särskilt mycket värde för teamet då många frågor var oförändrade och det tog väldigt lång tid att svara på alla frågorna. Detta var dock något som vi insåg att vi inte behövt göra då vi istället borde anpassat vilka frågor vi skulle svarat på efter varje sprint. I ett annat projekt känner vi att vi hellre velat ha en retrospective där exempelvis följande frågor besvaras:

- Vad gick bra?
- Vad kan vi förbättra?

[6] Henrik Kniberg, *Henrik’s Lean & Agile slides*

- Vilka åtgärdspunkter ska vi ta med oss till nästa sprint?

6. User stories

Vi har använt user stories som är en agile practice och hur vi använt oss av dessa beskrivs i avsnitt 1.3.

7. Agile Roles

De agila roller som vi använt beskrivs i avsnitt 4.1.

8. Code integration

Detta är något som vi arbetat med genom att för varje user story skapa en egen gren utifrån mastern. När denna sedan blivit klar har funktionen testats innan den slagits ihop med den gamla koden i master-grenen. Detta har underlättat testandet av nya user stories och minskat alla "merge conflicts", samt sett till att vi alltid har en fungerande version av appen. Till en början slarvades det lite med detta, vilket främst berodde på att vi inte hade tydliga restriktioner för när en ny gren skulle skapas och vissa hade även dålig koll på hur det skulle göras. I slutet av projektet blev vi dock bättre på detta. För att utöka kodintegrationen hade man kunnat implementera exempelvis GitFlow men eftersom alla inte var bekväma i Git valde vi att göra en liknande fast simplare lösning.

Utöver dessa agile practices skulle vi vid ett senare projekt även vilja använda oss av flera agile practices som till exempel:

- Customer-oriented Approach, vilket innebär att man har kontinuerlig kontakt med sin kund för att få feedback och möta deras kundkrav. Detta har inte varit möjligt i detta projekt då vi inte haft en tydlig kundkontakt och inte heller haft kontakt med några andra stakeholders. Att ha en kontinuerlig kontakt med en kund tror vi hade ökat motivationen för att leverera en bra produkt. Även detta är något som Henrik Kniberg nämner i sina slides då han menar på att det är viktigt att minimera avståndet mellan "maker" och "user" av det som skapas.[7]
- Automated testing är något som vi gärna hade velat arbeta med under projektets gång men vi hade svårt att hitta någon sorts testning som passade vårt projekt, då många av testerna handlade om att "testa" design och xml filer.
- Pair programming är något som vi inte arbetat så mycket med under projektets gång. Vi valde att inte göra det för att vi kände att det skulle lett till att mindre blivit gjort varje sprint, eftersom två personer då arbetat vid samma dator där den ena skriver koden medan den andra är mer passiv. En ytterligare anledning till att detta inte gjorts är att vi inte alltid kunnat arbeta tillsammans och då man genom parprogrammering blir begränsad av den andra personen och dess tider. Vid ett större projekt hade vi gärna använt oss av denna metoden i större utsträckning, då det är bra för inlärningen och koden blir direkt reviewad av en annan inom teamet vilket troligtvis leder till bättre kodkvalité.

[7] Henrik Kniberg, *Henrik's Lean & Agile slides*

4.3 The sprint review and how it relates to your scope and customer value (in the first weeks in terms of the outcome of the current week's exercise; in later weeks in terms of your meetings with the product owner)

Besvarat under listobjekt 4 i kapitel 4.2.

4.4 Best practices for learning and using new tools and technologies (IDEs, version control, Scrum boards etc.; do not only describe which tools you used but focus on how you developed the expertise to use them)

De verktyg/program/API:er vi använt oss av under projektet är följande:

- Firebase
- Git
- Trello
- Android Studio (inkl. Gradle)
- Google Maps
- Slack

Firebase och Google Maps var nytt för alla i gruppen och för några var även de andra programmen förutom Trello och Slack helt nya. För att lära oss de nya programmen kollade vi främst på tutorials på Youtube, diskussionsforum så som StackOverflow och Androids egna API dokumentation, Android Developers. Vid inlärning av Git tog vi främst hjälp av varandra då vissa har arbetat med Git tidigare. Denna inlärning underlättades av att vi satt mycket tillsammans då de som hade bättre koll kunde hjälpa resten i samband med att oklarheter uppstod. När vi stötte på något slags error i t.ex. Android Studio upplevde vi att den bästa strategin för att lösa det var att när vi satt i grupp först kolla om någon annan i gruppen hade upplevt samma problem och om hen i sådana fall kände till en lösning på problemet. Om så inte var fallet fann vi att det allra smidigaste sättet att lösa det på var att söka på det exakta felmeddelandet på internet istället för att sitta och försöka lösa det på egen hand. Många av de felmeddelandena som dök upp under projektets gång fanns det ofta relativt tydliga lösningar på, vilket gjorde att det var mer effektivt att göra på det sättet.

Innan vi bestämde oss för att skapa vår databas i Firebase började vi sprint 1 med att skapa en databas i PostgreSQL. Anledningen till det var att vissa redan hade arbetat med databaser och hade då gjort det i detta program. Efter den sprinten insåg vi dock att det vore mycket enklare att använda sig av Firebase då det är enklare att koppla ihop med Android Studio och för att det var lättare att anpassa till en föränderlig process. Något som hade kunnat göras betydligt bättre var alltså det faktum att vi lade en hel sprint på att skapa något som inte kunde användas senare.

Till nästa arbete vill vi tidigt kolla upp vilka program som är bra att arbeta med för att undvika det misstag vi gjorde med att börja utveckla något i ett program som inte ens kan användas senare i projektet. Eftersom att vi hade väldigt stor frihet i att välja vad vi ville göra för typ av projekt hade det dessutom varit bra om man redan innan hade kollat upp om de program som man hade tänkt att använda verkligen skulle fungera på det sätt som man önskar. Emulatorn i Android studio klarade till exempel inte av att simulera den applikationen som vi

valde att utveckla utan man var tvungen att testa på en riktig Android-telefon, vilket försvårade tester då man blev tvungen att använda varandras telefoner.

För att lyckas bättre i nästa projekt tror vi att det är viktigt att inte vara för ivrig när det kommer till att börja utveckla då det lätt blir att man inte tänker över sina beslut. Genom att lägga ned ännu mer tid på planering av vad som förväntas av slutprodukten är det lättare att identifiera vilka steg som behöver göras för att nå dit. Dessutom tänker vi att det är viktigt att inte göra på ett visst sätt eller använda en viss programvara bara för att man har erfarenhet av det. Vi bestämde oss till exempel för att utveckla en databas i PostgreSQL innan vi ens hade reflekterat över att det fanns ett annat alternativ (Firebase i detta fall) som passade vårt syfte bättre. Eftersom att några redan hade erfarenhet PostgreSQL så kändes det rimligt att använda det, men det blev mycket bättre när vi valde Firebase som ingen ens hade någon erfarenhet av.

I ett framtida projekt skulle vi till exempel även kunna dokumentera de felmeddelanden som man stöter på och en kort beskrivning av hur man löst dem i de fall som de beskrivningar som fanns online inte var tillräckligt tydliga.

4.5 Relation to literature and guest lectures (how do your reflections relate to what others have to say?)

Under projektets gång har vi inte arbetat så mycket med litteratur kopplat till Scrum och vi har inte haft några gästföreläsningar. Vi har lyssnat till vad våra handledare har sagt under sina föreläsningar och handledningstillfällen samt läst på om rollen som product owner och Scrum master. Inför nästa projekt hade vi kunnat använda oss av mer litteratur relaterat till Scrum och andra agila arbetssätt för att lättare ta till oss av olika förslag på strategier för bättre arbetssätt.

Henrik Kniberg lyfter till exempel hur ett agilt arbetssätt är tre gånger mer lyckat än vattenfallsmetoden.[8] Det är något som vi verkligen kan relatera till. Vi förstår att agila arbetssätt inte överskrider planerad tid och kostnad lika ofta som vattenfallsmetoden då det agila arbetssättet innebär att man snabbare kan upptäcka problem eftersom att processen är mer iterativ än sekventiell. Kniberg säger även att man inte ska motivera gruppen utan att man ska istället ta bort det som minskar motivationen. Vi har försökt att arbeta lite mer mot det sättet genom att låta alla i teamet arbeta med det som de är mest intresserade av.

Enligt Henrik Kniberg ska man fokusera på värde, inte effort eller output.[9] Vi har till exempel inte sett ett stort värde i att anteckna hur mycket tid vi har lagt på projektet (även om vi dock ändå har antecknat tiden) och hur det har avspeglat sig i vår output. Vi ser inte en direkt korrelation mellan tiden som vi har lagt på projektet och det som vi har levererat, utan vi har snarare fokuserat på att det som vi utvecklar ska skapa värde.

Henrik Kniberg säger även att “det som man mäter är det som man får” och att några av framgångsfaktorerna är att man sitter tillsammans, vilket vi verkligen har upplevt som bidragande till projektet.[10] Kommunikation och hjälp har fungerat mycket bättre när vi har

[8] Henrik Kniberg, *Henrik’s Lean & Agile slides*
[9] Henrik Kniberg, *Henrik’s Lean & Agile slides*
[10] Henrik Kniberg, *Henrik’s Lean & Agile slides*

suttit tillsammans. Det är en agil princip som vi verkligen har funnit användbar, att face-to-face kommunikation har varit den bästa kommunikationsformen. En annan framgångsfaktor som han nämner är att släppa versioner tidigt och ofta, vilket vi blev bättre på i slutet men som vi hade kunnat bli mycket bättre på. Till nästa projekt hade vi till exempel velat fokusera på att använda mer kortlivade grenar som sammanfogas med mastern allt oftare än vad vi har gjort i det här projektet.