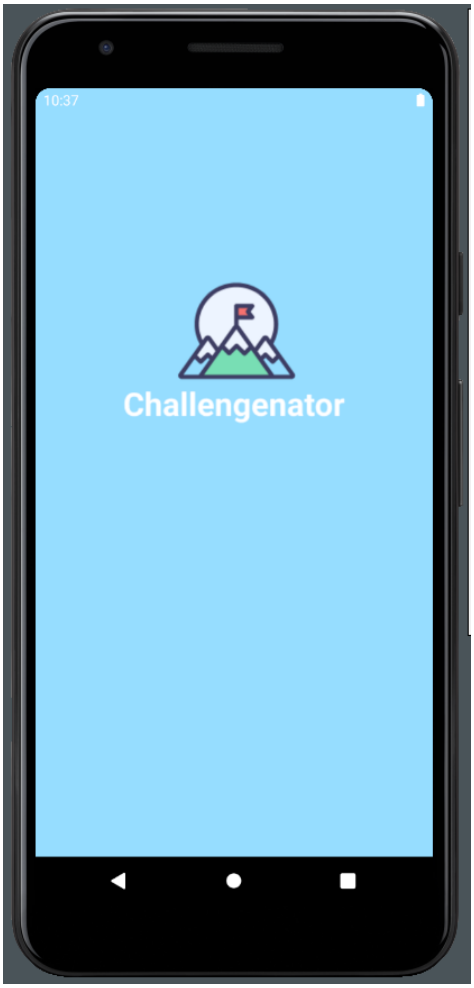
Final Team Reflection

DAT257 - Agile software project management



Grupp: Sony MZ1

Projekt: Challengenator

Medlemmar aka GitHub-namn: Felix Oliv aka Fajlix, Elina Wahlström aka elinawahlstrom, Wendy Pau aka Wendaaj, Viktor Johannesson aka TJOCKochFIN, Jesper Bergquist aka BQ-QB, Jonathan Stigson aka J.Stigson och Viktor Wirén aka viktorwcfc

Innehållsförteckning

1. Customer Value and Scope

1.1 The chosen scope of the application

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

I början var scopet hela applikationen och början av sprintarna fokuserade först på att skapa en mockup för att sedan implementera kärnfunktionerna och bli van med Android Studio. Vi tog ej i åtanke om vad som skulle vara klart innan kursen är över och därmed tänkte vi inte heller på att vi ska ha skapat en viss nivå av värde för vår externa aktör. Däremot efter några tips från handledaren började vi fokusera på att göra klart en *minimal viable product* vars funktioner gav mest värde till externa aktören. De funktioner var att användarna ska kunna logga in, ha en startsida, skapa och se detaljer om en utmaning samt uppdatera ens utveckling.

Det vi borde ha gjort är att utifrån mockupen fråga vår externa aktör vilka funktioner hen vill ska vara färdig i slutet av kursperioden. Detta kommer då att bli *minimal viable product* som innehåller de mest viktigaste kärnfunktionerna som externa aktören anser viktigast och ger mest värde. Detta kommer inte heller bara göra det lättare för oss att planera hela projektet inom en viss tidsfrist, utan även låta oss fokusera på att leverera en färdig produkt som nödvändigtvis inte behöver vara komplett, men fungerar.

För att göra detta så är det viktigt att vi tidigt skapar en mockup som externa aktören är nöjd med och detta då kommer att vara målet vi strävar mot. Sedan får vi stämma av med vår externa aktör vad hen tycker ger mest värde och främst vill ha klart. Detta kommer att ge oss det första scopet som vi minst ska vara färdiga med och detta gör det möjligt för oss att bryta ner i user stories som vi sedan kan arbeta med.

1.2 Success criteria

“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)”

Ett av våra mål var att varje sprint bli klara med de user stories vi tagit oss an med lika mycket ansträngning varje vecka, det vill säga att vi skulle ha ett så jämnt arbetsflöde som möjligt över varje sprint. Det är något som är viktigt för att vi ska uppnå det agila arbetssättet *sustainable pace*. Vi insåg snabbt att ett av våra mål också blev att skapa så självständiga *user stories* som möjligt för att vi inte skulle få flaskhalsar i projektet, där vi behövde vänta på varandra. Genom att dela upp dem ytterligare behövde vi inte invänta någon annan och på så sätt var det lättare att bli klar i god tid. Det kan också ses som ett delmål till att varje sprint bli klara med de *user stories* som vi tar oss an, då självständiga *user stories* är ett måste för att vi skulle kunna uppnå det målet.

Senare under projektet blev ett av målen att bli klara tidigare för att vi skulle kunna hitta problem tidigare. Det var viktigt för att vi skulle få så bra kodkvalité som möjligt och slippa att lägga massor av tid senare i projektet på att fixa problem och buggar som kunde åtgärdats tidigare.

Vi hade först som mål att uppnå *minimal viable product* och sedan jobba vidare mot *preferable viable product*. Till nästa projekt hade det varit bra om redan i början av projektet sätter tydliga gränser för vad *minimal viable product* är samt *preferable viable product*, då det diskuterades fram senare i projektet.

Ett ytterligare mål var att alla i projektet skulle förstå och kunna använda sig av de hjälpmedel som vi haft under projektets gång. Det är något som vi successivt jobbat med hela tiden under projektets gång. Till nästa projekt hade det varit bra om alla delat med sig av sina individuella mål för projektet som vi skrivit *individual reflection*, så att alla kunde vara med och stötta varandra att även nå de personliga målen.

1.3 User stories

"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."

Acceptance criteria, *task breakdown* och *effort estimation* är något som kändes lite oklart för gruppen i de första sprinterna, då vi inte var fullt klara med hur vi skulle dela in *tasks* och vad exakt vi skulle sätta som *acceptance criteria* och *effort estimation* så gjorde det att vårt arbete tidigt var ostrukturerat och vi hann inte klart med allt som skulle göras på sprinten, efter att vi fick detta problem upplyst så lade vi större fokus på att tydligt fixa *acceptance criteria* och *effort estimation* innan varje sprint genom att notera dessa i vår *Scrum board*. Detta ledde till ett mer strukturerat arbete och vi blev bättre på att bedöma de båda ju mer tid det gick. *Task breakdown* var något som gjorde stor skillnad för oss. Att vi blev bättre på att bryta ner *tasks* gjorde att vi fick färdigt mer och i de senare sprintarna hann vi med allt som skulle göras på sprinten. Detta gav värde för vår externa aktör då vi fick mer färdiga delar av applikationen klara i varje sprint vilket gjorde att vi hade mer att visa upp. Det gav också värde för oss då vi kunde jobba på ett mer effektivt och mindre stressande sätt där arbets indelningen var mer realistisk för vad som skulle hinnas med.

I framtida projekt är det av värde att tidigt lägga stort fokus på att tydlig strukturering för *acceptance criteria*, *task breakdown* och *effort estimation*, så att gruppen tidigt kan komma in i en bra takt med sprintarna där alla har en bra arbetstakt.

1.4 Acceptance test

"Your acceptance test, such as how they were performed, with whom, and which value they provided for you and the other stakeholders."

Vår externa aktör var inte närvarande varje vecka, vilket gjorde att vi var lite förvirrade över vad som var minimal viable product. Halvvägs in behöver vi därför kontakta henne för att förtydliga vad som var absolut nödvändigaste. Detta gav oss en plan vilket gjorde att vi kunde planera sprintarna bättre, och göra klart det som gav värde först. Genom projektet testade vår externa aktör och gav oss specifika förbättringsområden för oss att utföra, dessutom vad hon ville ha härnäst. I en av våra sista sprintar skrev vi ett testprotokoll att följa för att navigera runt alla olika funktioner. Vi hann tyvärr inte testa om detta ledde till färre buggar, men i framtida projekt är detta något som borde testas i början.

Förutom att vår externa aktör testade vår applikation använde vi oss av *pull request* innan vi slog ihop utvecklingen till vårt inkrement. Vi satte ett krav på att minst två godkännanden skulle komma från vårt team innan get godkändes. Detta blev stressigt i början då *pull requests* skickades in precis i slutet på sprinten och utvecklingen av programmet hann inte riktigt godkännas innan sprinten var över. Detta ledde till att vi satte en gräns på att skicka in *pull request* två dagar innan sprinten var över för att ge tid till gruppen att godkänna och kolla igenom varje *pull request* mer noggrant.

I framtida projekt kommer definitivt reflektionen kring hur *pull requests* ska lösas tidigare. Detta kommer möjliggöra en del buggar som senare eventuellt kommer visa sig och öka energin lagd på projektet.

Även att den externa aktören är mer närvarande och hela tiden ger feedback på vad som ger mest värde och testar så att de funktioner vi hittills implementerat funkar som önskat. Detta kommer leda till mindre jobb i slutändan då vi redan i detta projektet behövde ändra utseende och vissa funktioner efter att det hade visats upp via vårt inkrement, vilket ledde till att vissa av oss behövde sitta senare sprintar och korrigera applikationen.

1.5 KPI:er

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

A: Projektet som varit

Till en början valde vi de tre KPI:erna:

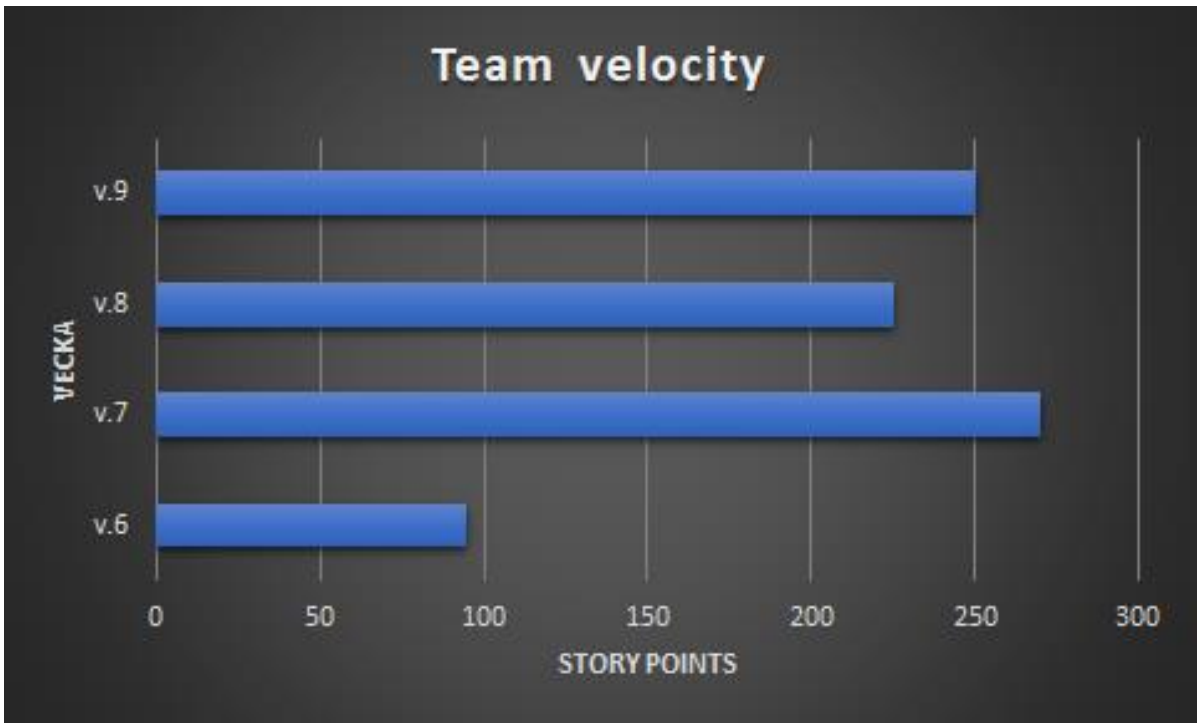
1. *How close to our goal are we? Sprint wise? Project wise?*
2. *Stress indicator (stress, workload)*
3. *Lines of code divided by the time spent (hours)*

Efter första sprinten insåg vi däremot att det var svårt att mäta hur nära vårt mål vi var enligt KPI nr.1, eftersom vi inte hade något sätt att mäta hur nära målet vi var. Efter diskussion med vår handledare, så bytte vi ut den till *team velocity* för att kunna få en mer mätbar KPI som nr.1. *Team velocity* mättes istället i story points, där antalet *story points* representerade hur mycket värde varje enskild user story innebar för att ta oss närmare vårt slutmål och *minimal viable product*. Nu kunde vi får mer svart på vitt hur mycket närmare vårt mål vi kom efter varje sprint. KPI 2 och 3 behöll vi genom hela projektet, då båda gick att räkna på och mäta. Målet var att hålla KPI:erna så jämna som möjligt över sprintarna, hitta en nivå som alla i gruppen klarade av samt hålla arbetsflödet jämnt för att vi skulle orka med det under en längre period och därmed använda det agila arbetssättet *sustainable pace.*

De slutliga tre KPI:erna för projektet:

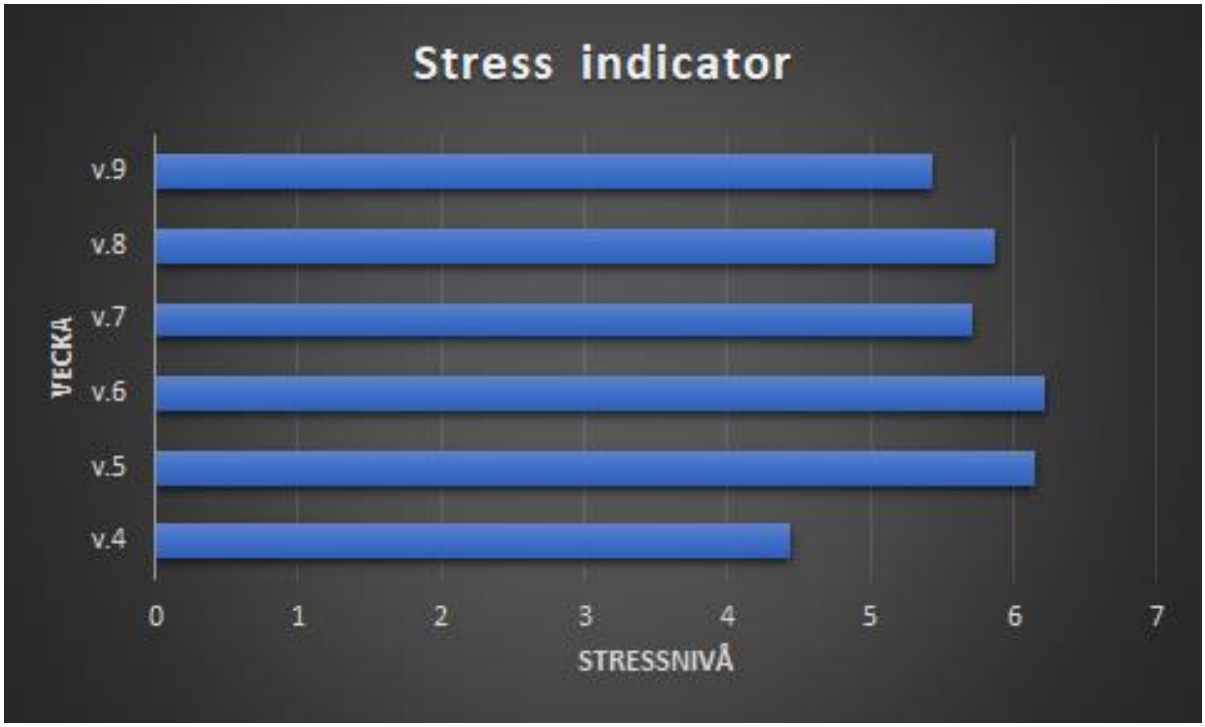
1. *Team velocity*
2. *Stress indicator (stress, workload)*
3. *Lines of code divided by the time spent (hours)*

Följande diagram visar på team velocity för projektet:



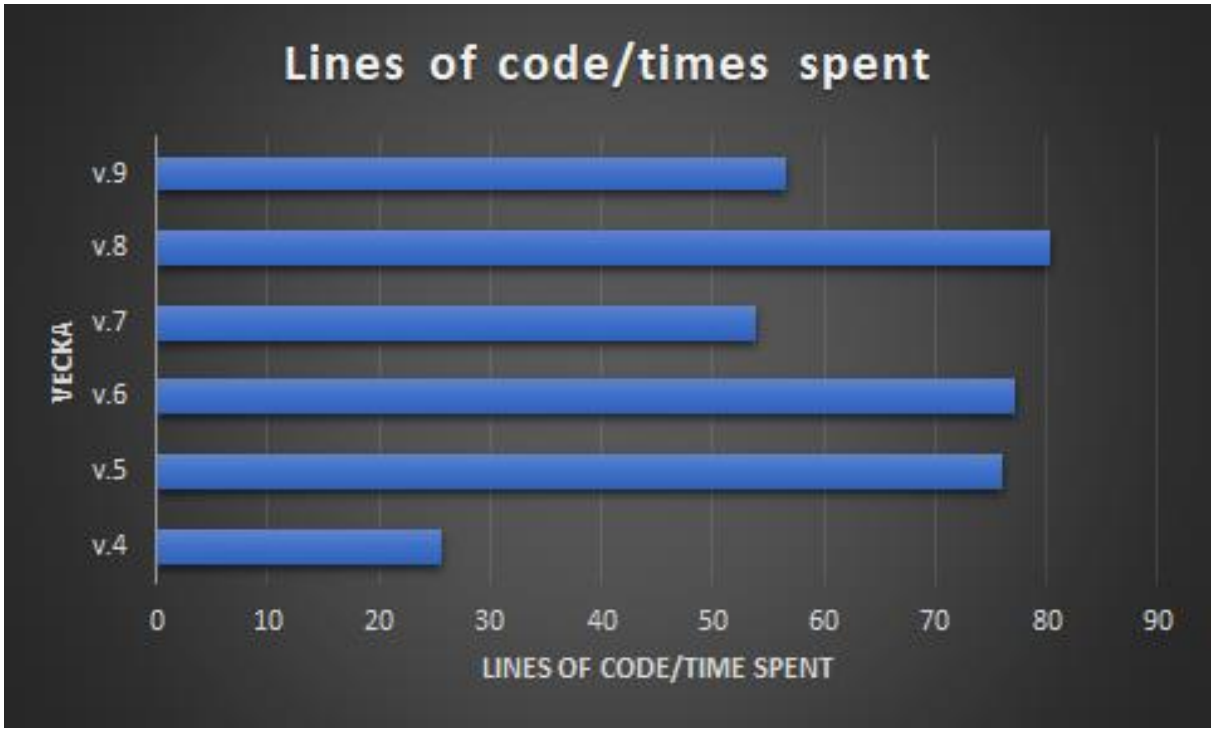
Vecka 4 och 5 saknas, eftersom vi hade en annan KPI innan som vi bytte ut. Varför vi hade lägre *story points* vecka 6 var för att vi i början av projektet var dåliga på att bryta ner *user stories* tillräckligt mycket, så det uppstod flaskhalsar och vi blev inte helt klara med en del *user stories*, vilket gjorde att vi inte fick de poängen. Det förklarar också varför vi veckan efter har fler *story points* än de andra veckorna. Allt eftersom projektet gick blev vi bättre på att bryta ner user stories:arna och på så sätt undvek vi att få flaskhalsar, vilket gjorde det enklare att slutföra *user stories*.

Följande diagram visar på stress indicator för projektet:



Första vecka ligger klart lägst i stressnivå, men sedan ökar och minskar den mellan ca. 5,5 till straxt över 6. Den håller sig alltså hyfsat jämn över veckorna, men varierar lite beroende på hur stressigt vi haft i de kurser vi haft vid sidan av denna. Vi tyckte att det var viktigt att ha med stressnivå som en KPI för att det hänger ihop med vår förmåga att prestera under varje sprint.

Följande diagram visar på gruppens totala antal rader kod/tiden som alla lagt på projektet under sprinten:



Rader kod/tid spenderad varierade från vecka till vecka, vilket dels kan ha berott på att rader kod inte tar med design aspekten eller det faktum att alla i gruppen skriver kod olika snabbt, då vi har varierande kunskap. Det är svårt att få en så inkluderande KPI som möjligt som tar med alla aspekter, men vid projektets början valde vi denna för att den var mätbar. Kanske finns det någon bättre KPI som tar med design och varierande kunskapsgrad som vi kan använda i nästa projekt för att få ett mer jämnt och ett mer tillförlitligt diagram.

2. Social Contract and Effort

2.1 Social contract

"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)”

Vårt gruppkontrakt var något som skapades tidigt och gjordes ordentligt. Vi satte upp tydliga tider och riktlinjer för möten. Dessa har följts under samtliga veckor och har skapat en stabil grund för vilket vårt arbete kretsat kring. Vi fastslog kommunikationsriktlinjer för hur och var vi kontaktade varandra. Dessa är något som skapat trygghet i vår kommunikation och alla vet vart de skall vända sig för att kommunicera. Attityder och gruppdynamik var en rubrik i vårt gruppkontrakt där sju regler om hur vårt sociala bemötande skulle gå till i grupparbetet, dessa regler har gjort den sociala kontakten i vårt arbete öppen och tillmötesgående och de är inte något som har behövts tas upp under arbetets gång.

Deadlines är något som gruppen ville hålla och skrev om i gruppkontraktet, detta gällde sprintar men också andra internt uppgjorda deadlines. Tidigt i arbetet gjorde en del oklarheter kring uppdelning av arbetet så att alla i gruppen inte alltid hann färdigt i alla sprintar, även olyckor och sjukdom förhindrandet gruppen från att hålla deadlines, senare i arbetet var arbetsfördelningen finslipad och gruppen höll sina deadlines.

I gruppkontraktet hade vi även med en övrig punkt för att få med regeln: *”Det är förbjudet att ändra någon annans kod utan beslut från personen som skrivit den ursprungligen.”* Denna punkt skrevs då personer haft upplevelser i tidigare projekt som inte varit positiva kring detta. Regeln bröts då personer behövde ändra i andras kod för att få applikationen att fungera korrekt och personen vars kod behövde ändras ej svarade på kontakt i tid. I framtida projekt skulle en regel som denna behöva omformuleras om vilka kodförändringar som är ok och inte. Vi känner i efterhand att ändringar i annans kod kan vara nödvändiga och de kommer fortfarande synas i *pull request.*

Gruppkontraktet är något som har fungerat bra för vår grupp och det har underlättat och tydliggjort vårt arbete och arbetsdynamik under projektets gång. Något som har saknats och som skulle kunna förfinas till nästa projekt är hur gruppen skall reagera och fördela när olyckor, sjukdom och dylikt gör att personer i gruppen ej kan närvara och missar arbete. När detta inträffade i en sprint missades den frånvarande personens arbete att plockas upp av gruppen. Vi lärde oss av detta och jobbade senare med omplaneringsmöten, men att tidigt tydliggöra regler och riktlinjer kring omplanering i nästa projekts gruppkontrakt är något som kommer göra arbetet än mer effektivt och finslipat.

2.2 Time spent and deliveries

"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)."

Antalet timmar spenderade per sprint steg allt eftersom kursen fortgick, med undantag för ett par sprintar på grund av andra kurser som kom emellan. Det skulle kunna bero på att hela arbetssättet var nytt för samtliga projektmedlemmar och särskilt att *estimation of effort* inte var särskilt precis inledningsvis. Under de första sprintarna underskattades antalet timmar som behövde läggas ner för att färdigställa alla *user stories*, vilket resulterade i att inte allt som planerats för sprinten hann bli klart. Under de senare sprintarna däremot lades fler timmar ner samtidigt som mängden arbete också ökade.

Detta skulle kunna bero på att ett flertal projektmedlemmar aldrig använt utvecklingsmiljön Android Studio innan men blev mer bekväma under projektets gång och kunde därför ta sig an större uppgifter som kanske tar mer tid.

Det handlar helt enkelt om erfarenhet och inledningsvis bristen på det. När samtliga vant sig med den agila utvecklingsprocessen började saker och ting komma naturligt och gå smidigare vilket som tidigare nämnts resulterade i förmågan att leverera mer på kortare tid.

Något som skulle kunna förbättras för att minska risken för att något liknande ska inträffa vid ett framtida projekt är att, ifall det märks att *estimation of effort* är fel, kalla till ett omplaneringsmöte eller liknande och korrigera det som märkts kommer ta mer eller mindre tid. Det är något som började implementeras under den senare delen av projektet.

3. Design decision and product structure

3.1 Design decisions

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

I detta projekt hade vi nära kontakt med vår externa aktör, och det absolut första vi gjorde var att skapa en mockup i Figma för att visa vår design. Vi använde Figma på

grund av att vissa i gruppen redan hade erfarenhet av Figma och det är ett enkelt sätt att dela våra tankar med den externa aktören.

Från tidigare erfarenhet av att bygga en Android applikation var valet lätt att välja MVVM som program arkitektur. Modellen behöver vi för att ha all data och logik, detta ska inte blandas ihop med andra delar av programmet. Vyn innehåller allt som visas på skärmen, vilka komponenter som syns och vilka användaren kan interagera med. Vymodellen är till för att manipulera vyn beroende på data och vad vyn innehåller, vilket fungerar som en brygga mellan vyn och modellen, utan att det ska finnas beroenden mellan dem.

Tack vare få beroenden och separering av olika delar av applikationen kodmässigt, var det lätt att bygga en del och sedan lägga till olika önskemål från vår externa aktör. För att skapa värde för både oss och vår externa aktör skapade vi till exempel en komponent för att kunna mäta poäng i en utmaning. När en komponent fungerade som den skulle var det lätt att lägga till flera andra komponenter utan mycket ansträngning. Dock följdes inte detta fullt då vi hade endast en *viewModel* och om detta varit ett större projekt hade det blivit väldigt komplicerat att bygga vidare på funktioner och det skulle behöva skrivas om. Även andra småfel runt arkitekturen finns, men vi valde att lägga mer fokus på värde till kunden istället för kodkvalitet.

I kursen gick det smidigt med besluten, men en förbättring skulle kunna vara att ha en diskussion med den externa aktören för att kunna välja ett verktyg som hon skulle kunna ha erfarenhet av. Detta skulle kunna ge mer värde för henne då det utförs i hennes önskemål och underlätta för hennes framtida användning av projektet.

3.2 Technical documentation

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

Vad gäller dokumentation har det gjorts med hjälp av Trello, Figma och text dokument. Trello användes för en *Scrum Board* att uppdatera utvecklingen av user stories, uppgifter och buggar. Figma användes för grafisk design och text dokument för att dokumentera möten och regler kring kodkvalitet och testning.

För ett liknande projekt i framtiden skulle det nog vara fördelaktigt att ha en kontinuerligt uppdaterad *domain model* eller liknande för att enkelt kunna få en överblick av hela projektet.

3.3 Use and update of documentation

"How you use and update documentation throughout the sprints."

Genomgående i arbetsprocessen har användandet och uppdateringen av gruppens dokumentation fungerat relativt väl. Om problem har uppstått så har gruppmedlemmar inte tvekat på att hjälpa till med problem eller svarat på frågor. Programmet Trello har använts för att tillhandahålla en Scrum Board med bland annat dokumentera *user stories*, *story points* och de framtagna värdena på *estimation of effort* dokumenterades inför varje sprint. Google Drive har använts för att spara all dokumentation från möten. Här inkluderas även gruppens *team reflections*, sprintplaneringar och sprintutvärderingar, vilka sedan laddades upp på GitHub. Tillvägagångssättet och användningen av dokumentationen har blivit bättre och bättre under projektets gång. Från att delar av gruppen varit osäkra på hur man ens laddar upp filer till GitHub eller hur man skapar en ny *task* i Trello, till att man gör det utan problem. Goda råd från handledare angående vad som är relevant att ha med eller angående vart fokus bör läggas har också varit hjälpsamt, sett till arbetets dokumentation.

Om ett liknande projekt hade gjorts i framtiden med erfarenheterna som erhållits under projektets gång kan slutsatsen dras att en mindre startsträcka gällande

programmen hade varit att föredra. Om alla i gruppen hade haft erfarenhet av exempelvis GitHub sedan innan, så hade arbetet kunnat vara mer effektivt från början. Allt som har dokumenterats har kanske inte varit helt relevant för arbetets gång. En del kan ses som överflödigt och en del kan dessutom saknas i dokumentationen, men i och med att detta var första gången som gruppen varit delaktiga i ett agilt projekt så är det dock inte helt oväntat. Om projektet hade utförts igen hade gruppen haft en tydligare bild om vad som ska vara med i dokumentationen.

För att lyckas bättre med agilt projekt i framtiden anses en tydlig dokumentation från grunden vara en vital del. Om man tidigt i arbetsprocessen identifierar vart fokus ska ligga kan dokumentationen bli bättre. Vidare kan dokumentationen bli tydligare och mer strukturerad om alla i gruppen har erfarenhet av programmen där dokumentation utförts sedan tidigare.

3.4 Code quality and coding standards

"How you ensure code quality and enforce coding standards."

Kodens kvalité har förändrats mycket under projektet gång. Från början låg fokus mer på att leverera en produkt till vår externa aktör, men allt eftersom projektet gick skiftade fokus mer mot god kod kvalité. Tidigt bestämde vi oss för att alla *branches* skulle sammanställas med main branch:en med hjälp av GitHub *pull requests*. Dessa *requests* skulle granskas och godkännas av minst två andra gruppmedlemmar för att *requesten* skulle gå igenom. Vad vi från början missade att göra var att inte bestämma hur koden skulle granskas. Vi hade inte bestämt vad som skulle anses vara bra respektive dåligt och inte heller vad som kunde accepteras och vad som behövdes ändras. Detta gjorde att beroende på vilka som granskade en specifik *request* kunde resultatet bli olika. Detta förstärktes av att skillnaden i erfarenhet med Java samt Android var stor mellan oss. Resultatet blev att de med mindre erfarenhet inte blev tillfrågade att granska särskilt många *request* då tanken oftas var att vi bad de erfarna inom området att granska *requesten* istället. Då projektet gick framåt och mer av vårt

fokus låg på just kod kvaliteten gjorde vi ett flertal ändringar. Först och främst gjorde vi kod granskningsregler som alla kunde följa för att göra kodkvaliteten så konsekvent som möjligt. Efter två sprintar kom vi fram till följande regler som vi sedan använde i resten av projektet.

- Readability
The code should be either easy to understand by the use of variable names, method names etc, or that the code is explained using comments or javadocs.
- Dependencies
There should not be any useless dependencies, and dependencies that can be removed using in some way, ex: observer pattern. The reviewer should have an idea of how to do this refactoring.
- Testability (modellen ska kunna testas)
All the logical code, i.e the model, should either have tests or should have the ability to be easily tested in the future.

För att reglerna ska vara applicerbara för alla medlemmar är de skrivna på ett sådant sätt att erfarenheten på granskaren ska ha så lite betydelse som möjligt. Exempelvis behöver inte granskaren förstå exakt hur en del kod fungerar, utan då ska det istället finnas kommentarer eller *javadocs* som förklarar den koden istället. Vi började också skriva upp alla medlemmar som granskare så att alla medlemmar kan granska och inte bara dem med mest erfarenhet. Även om detta inte var ett perfekt system då alla nu själva hade ansvaret att granska kod, valde dem med minst erfarenhet istället att inte granska något utan valde istället att vänta på att någon annan gjorde det. Detta för att dem inte kände att de hade kunskap nog för att granska koden och valde då istället att inte granska alls. Detta ledde till att vi i det sista två eller tre sprintarna valde att de med mer erfarenhet granskade kod tillsammans med någon med mindre erfarenhet för att lära ut hur hen gör för att alla medlemmar i framtiden ska kunna granska kod självständigt.

I ett framtida projekt skulle vi framförallt göra två saker tidigare. Vi skulle redan från början skriva objektiva och tydliga kodgranskningsregler för alla *requests*, både små och stora, för att få så bra och konsekvent kod som möjligt. Vi skulle även börja granska kod tillsammans direkt. Detta skulle innebära att alla medlemmar så snabbt som möjligt skulle kunna granska koden självständigt och ha en övergripande

förståelse för hur koden fungerar. Det innebär även att samtliga skulle kunna underhålla och utveckla alla delar av koden.

4. Application of Scrum

4.1 Roles and their impact

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

Rollerna som använts under projektets gång är som följer: *Product Owner*, V*ice Product Owner*, *Scrum Master*, programmerare samt GitHub-ansvarig. Alla i gruppen har varit programmerare, men utöver det har rollerna tydliggjort vem som har det yttersta ansvaret för faktorerna. *Product Owner* har kommunicerat med vår externa aktör i form av att exempelvis presentera vad som åstadkommits under veckan samt godkännande av det som utförts. *Vice Product Owner* övertog ansvaret om *Product Owner* inte kunde närvara vid exempelvis möten. *Scrum master* har delvis använts i projektet genom att personen med ansvarsområdet har fokuserat på värdeskapande

för kunden, men det har främst varit ett gemensamt arbete med Scrum som har använts. GitHub-ansvarig har främst varit ansvarig för att hjälpa medlemmar i gruppen med användning av GitHub. Exempelvis med hur man skickar *pull requests*.

Vid nästa agila projekt hade det varit att föredra om en roll som GitHub-ansvarig inte behövde lägga så mycket tid på att lära ut. Om samtliga medlemmar kunde använda programmen som krävs från start hade tid och energi sparats och arbetet kunnat flyta på smidigare. I och med att alla hade minimala erfarenheter av *Scrum* vid projektets start kändes det naturligt att alla hjälptes åt med användandet. I framtiden kan det vara att föredra att en medlem med djupare kunskaper inom Scrum kunde stötta och tydliggöra arbetet ytterligare. Genomgående hade arbetet kunna vara effektivare om baskunskaperna hos medlemmarna gällande relevanta program eller tillvägagångssätt varit större och om projektet hade gjorts om ännu en gång hade således utfallet troligen blivit bättre.

4.2 Agile practices

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

Den agila processen som vi använt för projektet är Scrum, där de agila arbetssätten som vi använt är följande:

- *User stories*
- *Pair programming*
- *Customer value*
- *Sustainable pace*

Varje sprint har varit från måndag till måndag, där vi haft återkommande möten varje måndag och fredag.

Vi har varje vecka skrivit *user stories* som vi gett story points och *estimation of effort*, som vi sedan delat upp mellan oss i gruppen. Vi har jobbat på *user stories* i mindre

grupper eller enskilt beroende på svårighetsgrad och kunskap kring de olika uppgifterna.

I och med att alla i gruppen hade olika erfarenheter av att programmera i Android Studio och använda versionshantering i GitHub, så var *pair programmin*g givet under projektets gång. Under första sprinten valde vi därför att i gruppen ha fokus på att alla skulle komma igång och få bättre koll på programmen. Vi parade därför ihop oss i grupper om två eller tre personer, med någon som kunde programmen med de som behövde lära sig mer. Vi tror att det var ett bra arbetssätt för att alla skulle komma på banan och lära av varandra. Vi har märkt under projektets gång att de som inte jobbat med det alls innan känt sig mer och mer säkra på hur de jobbar i programmen. Elina och Viktor W har kört pair programming genom större delen av projektet med andra gruppmedlemmar och med varandra. Vi tror att det har varit det bästa sättet för att få dem att utvecklas, där vi successivt kunnat ge dem större och mer utmanande *user stories*. Det har varit viktigt för oss inom projektet att hjälpas åt och lära av varandra, där det alltid funnits någon att fråga om vi fastnat på något eller inte vet hur vi ska lösa ett problem. Till nästa projekt hade det varit en bra idé att köra pair programming med varje person i gruppen för att på så sätt lära från olika personer samt att vi lär känna varandra i gruppen bättre.

Det var viktigt för oss att jobba med *customer value*, vilket är varför vi tidigt under projektet bytte en av våra KPI:er till *team velocity*, där varje *user story* fick *story points* efter hur viktig den var för slutprodukten. Allt för att vi i projektet slut skulle uppnå *minimal viable product.*

Något som vi jobbade med genom hela projektet var att försöka uppnå en *sustainable pace*, vilket vi gjorde genom att ha med KPI:n *stress indicator* som skulle ge oss en riktlinje kring hur gruppen upplevde arbetsbelastningen samt få ett genomsnitt på stressnivån i gruppen. Till nästa projekt hade det varit intressant att komplettera gruppens genomsnitt med de individuella stressnivåerna, eftersom det kunde skilja sig ganska mycket från vecka till vecka. Det möttes dock upp med att de personer

som var stressade fick mindre arbetsbelastning och de som inte kände så hög stress drog ett större lass.

4.3 Sprint review

"The sprint review and how it relates to your scope and customer value (Did you have a PO, if yes, who?, if no, how did you carry out the review? Did the review result in a re-prioritisation of user stories? How did the reviews relate to you DoD? Did the feedback change your way of working?)"

Vår *PO* var Felix O. och vår *sprint reviews* utfördes på samma dag innan handledningstiden. Vi gick igenom hur det gick för alla, om alla blev klara eller om det uppkom några komplikationer eller buggar samt gick vi även igenom om det är något mer som behövde fixas till nästa sprint. Under våra *sprint reviews* insåg vi att vi oftast inte kunde leverera de *user stories* och *tasks* under de flesta sprintar, utan det fanns alltid något kvar. Detta ledde till att våra *user stories* och *tasks* aldrig uppfyllde vår *DoD*.

Efter ett möte med handledaren, insåg vi att meningen med sprinten är att alla *user stories* och *tasks* som ska göras under sprinten ska vara klara i slutet och inget ska finnas kvar att göras. Det vill säga att det är vårt löfte till externa aktören att *user stories* och *tasks* ska bli *done, done, done* (*coded*, *pull request approved*, accepterad av den externa aktören). Därmed lärde vi oss att vi kunde omplanera och om någon inte hinner med sin del eller får förhinder så ska man alltid försöka delegera om uppgifterna eller meddela den externa aktören att det vi ej kommer att kunna uppfylla den delen av löftet. Efter mötet ändrades arbetsupplägget och sprintarna därefter blev fullständigt klara (*done, done, done*) inom tidsfristen.

Något vi tar med oss från detta är att i framtiden kan det underlätta att komma överens med externa aktören om *scopet* är möjligt att fullgöra inom nästa sprint så att vi ej lovar för mycket. Att alla user *stories* och *tasks* kommer vara vår främsta

prioritering och om det skulle vara så att det uppkommer förhinder under sprinten så kommer vi omplanera eller meddelade externa aktören om ändrade planer.

4.4 Learning and using new tools and technologies

"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)."

För att utveckla vår app använde vi oss av IDEn Android Studio då den är speciellt utvecklad för just android och gör det väldigt enkelt att både arbeta med backend och frontend i samma utvecklingsmiljö. Som tidigare nämnts har vi i gruppen haft olika mycket erfarenhet av Android och dess utvecklingsmiljö. Vi har därför jobbat mycket med att dela med oss av våra tidigare erfarenheter och kunskaper till resten av gruppen för att alla till slut ska kunna utveckla och underhålla alla delar i appen.

Vi valde Java istället för Kotlin som man även kan använda för Android då alla medlemmar har arbetat med det tidigare. Detta gjorde att vi kunde fokusera på att dela med oss av just Android kunskaperna. Vi valde att använda GitHub för versionshantering då flera av medlemmarna redan arbetat med det. Dock var det två stycken som aldrig hade arbetat med någon typ av versionshantering innan samt bara två som använt sig av *pull requests* innan. Vi började därför med att dela med oss av våra kunskaper om just GitHub i de första sprintarna och mot slutet hade alla ganska bra koll på hur det fungerar. För att vissa inte hade så bra koll från början bestämde vi oss för att ha en *GitHub-master* (Felix) som hade ansvar för att sammanställa de *pull requests* som under en sprint skapas och se till att alla *merge conflicts* togs hand om. I början var flera medlemmar med när han gjorde det för att lära sig, men då det ansågs som komplicerat lades huvudansvaret på en person istället.

Vi använde oss av Trello för vår Scrum board. Kunskapen av Scrum boards var ganska låg från början så vi fick tillsammans förbättra denna med varje sprint. Från

början var informationen och värdet som det skapade för gruppen ganska lågt men i de sista sprintarna var de mycket mer information och mycket mer värde som vi som utvecklare kunde få ut från det.

Exempelvis ger både våra acceptance criteria oss en bra överblick om vad som ska göras för att en *user story* ska vara klar och också vad andra gruppmedlemmar håller på med just nu.

Vi tycker att verktygen vi använt fungerat bra under projektets gång. Även om det fanns relativt låg erfarenhet av dessa verktyg i gruppen lyckades vi dela med oss av tillräckligt mycket för att vi alla skulle hamna på ungefär samma nivå. I ett annat projekt skulle det vara bra om alla redan hade kunskap om alla verktyg som ska användas så att tiden som läggs på att lära ut minskas. I verkligheten tror vi dock att det är mer eller mindre omöjligt och sättet vi arbetar för att dela med oss av vår kunskap fungerade bra.

4.5 Literature and guest lectures

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

Relevant litteratur för det agila projektet som har utförts beskriver hur projektet underlättas genom användandet av user stories. Närmare bestämt hur man bryter ner ett projekt i betydligt mindre beståndsdelar. Adzic (2012) beskriver den så kallade hamburgarmetoden som ett tillvägagångssätt där projektet som helhet liknas med en hamburgare där mindre omfattande tasks liknas vid tillbehören, exempelvis ost eller sallad. Hamburgarmetoden har använts flitigt av gruppen. När projektet inleddes var det svårt att identifiera *tasks* som var av rimlig magnitud. Uppgifterna gav mycket värde för kunden, men var samtidigt väldigt omfattande. Efter diskussion med handledare lades fokus på detta och våra *user stories* blev mindre och fler ju längre projektet fortskred. Detta kommer vi även ta med oss till potentiella framtida agila projekt.

Adzic (2012) beskriver även signifikansen att identifiera en *minimal viable product* . Detta gjordes tillsammans med projektets externa aktör. Basfunktioner som gav mest värde identifierades och skillnader mellan funktioner som var på *Minimum Viable* snarare än på *preferable* basis valdes ut. Även detta är ett koncept som tas med i framtida objekt.

Källförteckning

G, Adzic. (2012) *Splitting user stories -- the Hamburger Method.* Tillgänglig på: https://gojko.net/2012/01/23/splitting-user-stories-the-hamburger-method/