Final team reflection | 5.25" Floppy Disk

https://github.com/Alburrito18/FloppyDisk

Notera:
Formatet på texten nedan är på A: detta projekt, B: ett framtida projekt och C: funktionen för att gå från A→B

Customer Value and Scope

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

Vårt scope var redan från början baserat på det projektförslag som vår PO/kund presenterade och som gick ut på att ersätta en existerande, analog och manuell orderdata insamling via papper och penna med en digital lösning som även autogenererar vissa delar. Därigenom skulle man slippa tyda handskriven text och slippa upprepade inmatningar av den handskrivna informationen i flera olika system (orderläggning, CRM-system och datalagring).

Eftersom vår PO behövde åka ut till sina kundföretag för att göra fysiska avgjutningar, och i samband samlade in kunddatan, ansåg PO sig behöva en lösning som karaktäriserades av hög mobilitet. Genom diskussioner med PO avfärdades således ett datorprogram till förmån för en app, eftersom en mobil/surfplatta är enklare att ta med ut i fält. Då gruppen hade erfarenhet av att programmera i Java, föll valet på en android app. Eftersom inmatning dessutom förenklas av en stor skärm avgränsades applikationen i detta projekt initialt till en surfplattas format. Detta förenklade även bort dynamiska anpassningar till diverse olika skärmstorlek.

Värdeskapandet har huvudsakligen utgått från vår PO, i övrigt har endast begränsade försök gjorts att tillåta flera användare med samma tjänsteroll att använda appen parallellt (exempelvis genom att undvika duplikation av ordernummer vid multipla användare). Medan det kan ha varit en översikt, med tanke på att de som faktiskt använder appen är POs kunder, gör den enkla grafiska lösningen att vi inte upplevde ett stort behov av att inkludera dessa användares syn på värde.

Under de första ~ 4 veckorna hade vi ett starkt fokus på att inkludera alla datainmatningsfält samt att få datalagringen att fungera. Samtidigt hade vi då ännu inte börjat involvera PO i planering av user stories eller prioriteringen av features. Vi använde endast sprint reviews till att presentera sprintens resultat, utan att diskutera och revidera *product backlog* eller vägen mot *product goal*.

Under de sista 2 veckorna började vi dedikera mer tid till utvecklingen av övriga features som behövdes för en komplett MVP (i synnerhet skapandet av digitala kuponger och ordersammanställningar för utskrift). Samtidigt började vi involvera PO i prioriteringen av olika user stories och features, med resultatet att genereringen av filterkoder i synnerhet (en bokstavs- och sifferkombination som sammanfattar en order) prioriterades. Tidigare hade vi sett det som en mindre

viktig byråkratisk detalj, men att kunna inspektera den visade sig ha stort värde för PO som ett sätt att kontrollera huruvida ordern var korrekt.

Medan den sena involvering av PO delvis dröjde på grund av vår bristfälliga implementering av den agila arbetsmetodologin, så reflekterade vi även över att vi upplevde det svårare att involvera PO i prioritering av features i ett tidigt utvecklingsstadie. Produkten kändes i det stadiet så pass basal att det inte riktigt gick att tala om några riktiga features, utan det var mer fokus på att få grundfunktioner att komma på plats för att kunna visa upp en app överhuvudtaget.

Att utvecklingen av dokument-generationen påbörjades så pass sent, i kombination med att de visade sig vara något invecklade features, medförde att dessa färdigställdes först sista sprinten. Därmed fick PO inte dessa presenterade för sig förrän i slutprodukten och en värdefull möjlighet för feedback och diskussion om featureutformning gick förlorad.

En av ansträngningarna för att säkerställa att det utvecklingsarbete vi bedrev faktiskt genererade värde till PO var att substituera KPI:n “antal buggar rapporterade” till ett NKI (nöjd kund index), vilket också kan ha hjälpt oss prioritera de mest värdeskapande funktionerna under de sista 3 sprintarna.

B: I det här projektet upplever vi att vi kan ha fokuserat för mycket på den lösning som PO/kund efterfrågade, i stället för att identifiera de bakomliggande behoven och möjligheterna associerade med en digitaliseringslösning. Därigenom blev den resulterande produkten väldigt mycket en digital version av det befintliga systemet, vilket inte nödvändigtvis är den lösning som skapar mest värde. Vi hade velat undersöka möjligheten att inte bara överföra datainsamlingen till ett digitalt format, utan att anpassa affärsprocessen till utnyttjande av digitala tekniker. Kanske insamlingen av kunddata överhuvudtaget inte bör ske på plats?

Vi hade även velat få feedback på alla våra features som ingick i en MVP, vilket krävde att arbete på samtliga dessa påbörjades i ett tidigare skede av projektet.

Slutligen hade vi till ett framtida projekt velat ha en större förståelse för den inbördes prioriteringen av olika features samt vilka features som behöver ingå i en MVP i ett tidigare skede.

C: För att möjliggöra digitalisering av affärsmodellen, och inte bara göra datainsamlingen digital, bör man diskutera behov och olika lösningsalternativ grundligt i ett tidigt skede. Om man vill kunna redigera i fyllda kunduppgifter kanske det är bättre att investera i ett suddgummi än en android app? Som PO är det lätt att vara hemmablind för de processer man alltid använt, och det kan vara värt att undersöka alternativa lösningar som medför mer djuplodande förändringar av orderläggnings- och inköpsprocesserna.

För att kunna få feedback på alla features i god tid, så bör vi vid nästa projekt inte fokusera lika mycket på att färdigställa de mest väsentliga featuresen. I stället borde vi sannolikt haft ett approach av att utveckla en enkel version av kunddatasidan och en simplistisk version av dokumentgenerationen i ett tidigare stadie, och sedan utvecklat dessa efter features. Det är såklart även mer i linje med ett agilt arbetssätt, men vi halkade in att vilja färdigställa de påbörjade funktionerna, snarare än en fullständig och komplett MVP-version med samtliga funktioner i minsta möjliga form. Där bidrog sannolikt det faktum att vi ersatte ett befintligt, fysiskt dokument, så det var hela tiden tydligt vad slutresultatet borde innefatta och väldigt synligt om man hade en ännu ej färdig lösning.

För att få en större förståelse för vilka features som är viktiga, och hur de bör utformas, i ett tidigt stadie behöver PO definitivt involveras i planering av arbete och revidering av product backlog i ett tidigare skede av processen. En anledning till att det inte skedde var att de första veckornas arbete upplevdes som abstrakt och programmeringsteknisk, varför den oinsatte förväntades ha svårt att sätta sig in i valen som stod till hands. Emellertid borde vi vågat ta oss tid att förklara vad vi gjorde, vad som behövdes göras, och "kostnaden" i form av tid associerat med olika alternativ redan från början. Detta försvårades i det här projektet av att vi själva inte bottnade i dessa frågor, eftersom ingen hade tidigare erfarenhet av apputveckling.

A: 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)

Vi påbörjade projektet med ganska ospecificerade framgångskriterier. Vi ville lära oss agil mjukvaruutveckling, apputveckling för mobiler och Android Studio. Vi hade även en uttalad ambition om att kunna skapa någon form av påtagligt värde för vår PO innan kursens slut. Under kursens gång har vi delvis uppnått dessa framgångskriterier, och känner oss mer införstådda i vad såväl apputveckling som agil utveckling innebär. Framförallt känns det som att vi har byggt en grund att bygga vidare på hemma på kammaren om man så önskar.

Våra mål angående lagarbete var att vara snälla och förlåtande mot varandra och att sträva mot att hålla en kamratlig och gemytlig stämning oss sinsemellan. Genom att dessutom uppmuntra socialisering om ämnen utöver själva projektet och kursen hoppades vi på att bygga en god gruppkultur och sammanhållning. Samtidigt noterade vi vikten av att ha högt i tak och kunna säga till om det blev för flamsigt. De målen bedömer vi att vi lyckats hålla, och trots vissa betungande motgångar (framförallt många timmars bråkande med GitHub) så har stämningen alltjämt varit god.

Gällande ansträning hade vi en en uttalad ambition om att vidhålla en relativt jämn arbetsbörda och att skjuta in tillräckligt med ansträngning för att leverera en bra slutprodukt. Kanske var detta ett trubbigt satt mått, för tidigt i arbetet var effektiva sprint-kapaciteten tämligen låg för att sedan öka markant de sista veckorna när pressen på att sammanställa projektet ökade.

Inga nämnvärda revideringar av framgångsfaktorerna gjordes efter första veckan.

B: Det hade kunnat vara värdefullt att bilda sig en djupare, kontinuerlig förståelse sinsemellan för vad olika gruppmedlemmarna ville lära sig och åstadkomma, för att säkerställa att de fick arbeta på det. Emellertid fanns det redan frihet att efter intresse välja vilka user stories och features man arbetade med.

Eventuellt hade det kunnat finnas ett värde av att ett jämnare tempo under projektets gång. Emellertid var det bekvämt att kunna vara mer flexibel med när man la ner tiden, särskilt eftersom tidsintensiviteten hos andra åtaganden varierade mellan lagmedlemmar och veckor.

C: För att bilda större förståelse för individuella målsättningar kan återkommande och mer djuplodande diskussioner om dessa ämnen vara värdefullt. Nu hamnade det bara i de individuella reflektionerna.

Om ett jämnare tempo efterfrågas, kan ett fixt minimiantal timmar varje gruppmedlem bör investera i projektet per vecka specificeras.

A: 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

I början av projektet formulerade vi user stories som inte var särskilt vertikala och vars arbetsbörda var alldeles för omfattande. Den första veckan upplevde vi dessutom estimeringen och formuleringen av user stories så pass tidskrävande, och var så ivriga att komma igång, att vi negligerade att formulera user stories enligt formen "As an X, I want a Y, because of Z". Efter lite handledning blev våra user stories däremot mer väldefinierade, avgränsade och vertikala och landade i en standard form. Vi upplevde också tydligt hur det underlättade vårt arbete, genom tydliga och nåbara mål.

Effort estimation skedde till en början genom att uppskatta antal arbetstimmar med hjälp av fibonacci serien. Även om reduceringen av user stories storlekar underlättade estimeringen fann vi emellertid att det var utmanande att göra bra estimeringar. Detta primärt eftersom ingen var bekant med apputveckling eller android studio innan, varför stor osäkerhet angående tidsåtgång rådde. Vi var även osäkra på hur vi skulle estimera tidsåtgången vid parprogrammering (om två personer arbetade under 3h, var det då 6h eller 3h?). Valet aktualiserades i samband med att vi ibland använde parprogrammering och ibland singelprogrammering.

Även om serien var fördelaktig genom att tvinga fram lite större skillnader mellan user stories vilket underlättade prioriteringen fanns det även nackdelar. Ett problem som uppstod när vi uppskattade den gemensamma arbetstiden för två personer med fibonacci var att serien drog iväg ganska kraftigt efter 13, vilket kanske inte motsvarar den mer linjära verkligheten. Därav höll vi estimeringen huvudsakligen inom intervallet 1-13.

Efter hand blev det uppenbart att vissa gruppmedlemmar var skickligare än andra på vissa områden, och därmed skilde sig den individuella *sprint velocityn* mellan gruppmedlemmar. I ett försök att adressera detta försökte vi därför mäta någon mer abstrakt form av "story points" än åtgångna timmar. Detta tillät olika lagmedlemmar att ha olika "churn rates" (takt att för att färdigställde en given arbetsbörda). I samma veva bytte vi även estimeringssätt till t-shirt storlekar, då vi upplevde estimeringen i absoluta tal som svår och opraktiskt.

Användandet av t-shirt storlekar upplevdes som mer trivsamt då det blev mycket enklare och mer baserat på en intuitiv magkänsla, eftersom vi saknade tydliga definitioner av vilken arbetsbörda som motsvarade de olika storlekarna. Försök till att definiera de olika storlekarna utifrån olika referens-scenarion gjordes när gruppen var oense kring en enskild user story och behövde diskutera denna. Svårigheten med t-shirt-skalan bestod i att uppnå en samstämmig definition av vad de olika t-shirt storlekarna innebär i termer av ansträngning, i kombination med svårigheten att utvärdera hur väl estimeringen stämde med den verkliga arbetsbördan. Då antalet timmar estimerades var det tydligt visualiserat hur fel estimeringen blev om man estimerade 8h och det tog 3h. Det utvecklades mot att man sa "den känns lagom svår, det är nog M", och feedbacken uteblev. Därigenom försvårade det trubbiga ansträngningsmåttet att bli bättre på estimering.

Acceptanskriterier inkluderades i varje user story som en checklista med funktionalitet som krävdes för att varje user story skulle vara uppfylld. Denna checklista fungerade även som våran task breakdown. Kriterierna kunde se ut som “Ladda in sparad data vid öppnandet av appen”, “Fixa så all data sparas till samma fil” och “Navgraf ska kopplas från FirstFragment till BusinessView” och var således av varierande abstraktions- och detaljgrad med fokus antingen på funktionalitet eller implementation. Vi upplever att acceptanskriterierna och task breakdown fungerade bra. När man väl började arbeta med diverse user stories var det ofta att man insåg att andra tasks/kriterier var nödvändiga, och kanske av större betydelse, men det kändes som en oundviklig effekt av vår avsaknad av erfarenhet.

B: Till nästa projekt skulle vi vilja använda oss av en metod för effort estimation som ger kvantitativa resultat som är enkla att utvärdera mot utfall. Därigenom hoppas vi på att kunna bli kontinuerligt bättre på att estimera storleken på våra user stories och därmed minska risken för gruppens kompetens och resurser används suboptimalt.

Dessutom hade vi velat hitta ett sätt att specificera gruppmedlemmarnas inbördes kapaciteter, så att estimering kan göras oberoende av vem som ska utföra user storyn. Så blev inte fallet nu, där något kunde bedömas som small om person X gör det, för X har koll på input/output, men M-L om någon annan gör det”.

Det hade kunnat finnas ett värde i att undersöka om mer strukturerade acceptanskriterier gav bättre resultat.

C: Vi skulle vilja testa att använda abstrakta story points, och sedan ge varje person en individuell hastighet (story points/h). Sedan skulle vi återigen vilja använda en version av *planning poker* baserat på fibonacci serien för att estimera user stories. Medan estimeringen blir mer krävande, tror vi att feedbacken det ger i långa loppet skulle ge mer träffsäker och användbar estimering. En user story med 10 poäng bör ta en gruppmedlem med hastighet 1 poäng/h 10h att slutföra. Om det exempelvis tar 50 eller 2 så blir det uppenbart att estimeringen fungerat dåligt. Sedan tror vi det är viktigt att diskutera varför utfallen blev som det blev på sprint retrospective för att möjliggöra kollektivt lärande och analysera siffrorna bättre.

Använda acceptanskriterier på formen “Given, When, Then”. Känns dock som att det i synnerhet hade skapat värde senare i utvecklingsprocessen, då arbetet är mer UX fokuserat.

A: your acceptance tests, such as how they were performed, with whom, and which value they provided for you and the other stakeholders

Då målet med vår applikation var att ersätta en befintlig process och produkt (fysiska formulär och kuponger), blev de tidiga acceptans testerna något omvända. PO gav oss kopior och exempel på hur dessa fysiska exemplar såg ut. Sedan blev acceptans-testen att jämföra sitt resultat med dessa befintliga dokument och exempel för att säkerställa att ingen information eller funktionalitet försvunnit och därigenom blev POs önskemål ständigt integrerade i processen.

Dessutom gjordes veckovis presentationer av nytillkommen funktionalitet för PO på sprint review:n. Då denne hade begränsat med tid och appen bara gick att köra på emulatorn i Android Studio över Zoom så blev feedbacken och testningen gentemot acceptanskriterier kraftigt begränsade. Emellertid erhölls därigenom relativt bra feedback på UI. Presentationerna var inte på user story nivå, utan mer på produkt/sprint nivå.

Slutligen så använde vi peer-kodgranskning där varje user story jämfördes med acceptanskriterierna innan den kunde godkännas.

B: Till ett nästa projekt inom mjukvaruutveckling hade vi gärna involverat vår PO mer i faktiska testandet av applikationen i enlighet med vad vi gjorde senare i detta projekt. Det begränsade formatet att visa upp applikationen över Zoom och endast på en emulator hade säkerligen med fördel för *acceptance testing* kunnat bytas ut mot en nedladdningsbar version på en riktig tablet. Vår PO borde få mer att säga till om i processen att utvärdera UX och den sammantagna upplevelsen av att använda den digitala lösningen som ny version och substitut för den gamla metoden.

C: Att verka för att PO skulle kunna testa appen själv. Så länge det inte går är det svårt verkligen ge nyanserad kritik, då presentationerna via emulatorn och genom någon annans vägledning ger en mycket begränsad känsla för hur det är att faktiskt använda appen.

Att föra en mer utförlig och nyanserad dialog med PO angående vilka acceptanskriterier som är relevanta för de funktionaliteter som PO anser vara prioriterade att implementera.

A: the three KPIs you use for monitoring your progress and how you use them to improve your process

Först valde vi att använda oss av tre KPI:er: stressnivåer, känslan av mening (värdeskapande ingick som en del här) och samarbete i gruppen. Dessa valdes i samråd med hela gruppen och kändes som rimliga mått för att undersöka gruppen välmående, drivkrafter och värdeleverans. Initialt försökte vi därutöver även logga den individuella tidsåtgången för varje sprint och räkna antalet buggar vi stötte på. Brister på dessa två var att tiden inte alltid motsvarade värdeleveransen som varje individ gett upphov till, och att buggar av naturliga skäl förekommer i större utsträckning senare i projektet och därav får en skev fördelning. Vecka 6 ersatte vi “antal buggar rapporterade” mot ett NKI som mått (nöjd kund index) vilket innebar att vår PO fick uppskatta sprintens generade värde mellan 0-10. Bytet skedde med anledning av att appen bedömdes befinna sig i ett för experimentellt skede för att buggrapportering skulle vara ändamålsenligt; inga buggar rapporterades nämligen. Således hoppades vi på att vår nya KPI mer framgångsrikt skulle fånga värdeskapandet av varje sprint i vår PO:s ögon. Även en egen uppskattning av värdeleveransen infördes för att ge en referenspunkt för ett mått som beror på en extern aktör, till skillnad från andra KPI:er som enbart beror på den interna uppfattningen och prestationen.

Emellertid kommunicerade vi kanske inte våra förväntningar med måttet tillräckligt väl, och fick första veckan en 10:a mer baserat på att PO upplevde god kommunikation, punktliga leveranser och allmänt bra arbete från vår sida. Följaktligen valde vi nästkommande vecka att revidera måttet till “uppskatta sprintens värdeskapande i förhållande till föregående sprints”, där 5 var likvärdig prestation, för att föra bort resultaten från “gratis 10:or”. Det bedömdes ge önskvärda resultat, och KPI:erna landade därmed i en standard som vi höll kvar vid.

För att samla in KPI:er användes Flourish, en chatbot som automatiserade insamlingen genom att ställa frågorna i Messenger-chatten och dessutom fick de tillfrågade ge motiveringar till de numeriska insamling-värdena de angett och ge förbättringsförslag för att främja en positiv utveckling. Programmet fungerade väl, men medförde att sammanställningen av resultaten utgjorde en administrativ börda för den KPI-ansvarige och försvårade synliggörandet och tillgängliggörandet av resultaten gentemot övriga laget. Följaktligen hade laget dålig koll på utfallet av mätningarna, och vi misslyckades konsekvent med att implementera några direkta förändringar baserat på

förbättringsförslagen till nästa sprint planning (dvs. uppföljningar gjordes, men användes inte fullkomligt). Även om Flourish har en överblickande vy för hur laget presterar i samtliga KPI:er är verktyget gjort för en mer långsiktig användning och denna vyn kunde därav inte användas för att ge en rättvis bild av gruppens korta sprintar. Till följd av detta summerades KPI:erna för hand i textdokument som delades med övriga gruppmedlemmar (se exempel i mappen “KPIer” i vårt repo).

Den bristfälliga implementeringen av KPI:erna i vårt arbetssätt kan sannolikt inte enbart hänföras till tekniska och administrativa faktorer, utan är nog ett resultat av att vi inte upplevde mätningen som så värdeskapande. Medan det kan bero på att dåliga index användes, upplevde vi nog snarare att förbättringsförslag som Flourish samlade in var mer värdeskapande än de numeriska måtten och att det i en så liten grupp var tillräckligt enkelt att diskutera hur man upplevde stress etc sinsemellan och därigenom få en mer nyanserad bild än vad som kan kvantifieras i en siffra.

B: I ett framtida projekt hade vi behållit förbättringsförslagen kopplade till KPI:erna och sett till att reflektera mer över vår upplevelse av stress, värdeskapande och andra relevanta aspekter av arbetet i samband med sprint retrospective. Medan det är enkelt att förstå värdet av KPI:er i stora projekt, tycker en del av gruppen att det i framtida, små projekt vore bättre att slopa mätningen av KPI:er i syfte att frigöra tid och minska den administrativa bördor som inte genererar konkret värde. Samtidigt förespråkar en annan del av gruppen utveckling av måtten i stället för avveckling, och pekar på att de skulle behöva göras enklare att involvera i arbetet. I sådana fall måste de uppmätta värdena vara enklare att visualisera för alla gruppmedlemmar.

C: Antingen avveckla KPI:er eller visualisera uppmätta resultat tydligare (genom att exempelvis gå igenom dem på varje sprint retrospective eller skriva upp resultaten för varje vecka på scrum boarden). Dessutom undersöka möjligheterna att samla in resultaten på ett sätt som var mindre arbetsintensivt för den som sammanställer resultaten.

Social Contract and Effort

A: your social contract (Links to an external site.), 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

Efter kontraktets upprättande har endast en revidering förekommit, som gjordes för att addera ytterligare roller (PO- och KPI-ansvarig) och förändringar av de KPI:er (från antal buggar till NKI). I övrigt har vi inte heller gått tillbaka och kollat på det sociala kontraktet under projekttiden. Vi väljer emellertid att se det som ett tecken på att samarbetet i gruppen fungerat friktionsfritt och att inga sociala konflikter uppstått. Det kan delvis vara en konsekvens av att vi genom upprättandet av kontraktet diskutera sociala normer, regler och förväntningar således lyckades förebygga konfliktsituationer.

Vi tycker att det sociala kontraktet var ett värdefullt verktyg och tror att det var nyttigt diskutera hur vi ville arbeta med varandra, hur möten skulle ske och dylika förväntningar för att skapa en ömsesidig förståelse och undvika samarbetesrelaterade missförstånd och konflikter.

B: I ett nästkommande projekt hade vi uppskattat att använda det sociala kontrakten primärt som ett verktyg. Då vi ändå känner att det är bra att stämma av detaljerna som finns i kontraktet i ett tidigt skede är vi positivt inställda till att även använda det i framtida arbeten. Det är säkert givande att kontinuerligt återvända till kontraktet också för att få en uppfattning av vad som var den initiala avsikten och hur man ligger till just där och då. Genom denna jämförelse kan även insikter kring utvecklingen vinnas och nya vägval prioriteras.

C: Eftersom det som ansågs som mest värdefullt när det kom till det sociala kontraktet var just den diskussionen kring hur man beter sig, förväntningar osv. känns det som att man kan ta bort de delar från det sociala kontraktet som har större risk att förändras t.ex. KPI:er och vilka roller folk har. Det är i sig viktiga delar av projektet men kändes som separata punkter från den sociala biten. Dessa kanske passar bättre i separat dokumentation som revideras oftare. Där sociala kontraktet mer blir något man återvänder till när problem uppstår rent socialt.

A: 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)

Under kursens gång så har vi hållit koll på hur många timmar vi arbetat på projektet genom en kontinuerlig tidrapportering. Detta gjordes eftersom det annars är svårt att få en uppfattning av hur mycket tid diverse uppgifter tagit och det är svårt att uppskatta tidsåtgången i efterhand. Tidsrapporteringen gjordes i ett Google Spreadsheets ark där varje individ skrev in sin tidsinsats efter varje gång man jobbat på något. Utöver tiden i timmar och minuter lades även datum och kommentar till om vad man gjort. Därefter användes en summeringsfunktion för att hålla koll på hur mycket tid varje individ investerat i projektet. Även om vi försökt att ha gemensamt system för hur vi loggar och vilka tillfällen som ska loggas, reserverar vi oss för att somliga tillfällen har glömts bort i loggen. Särskilt de första veckorna och även de sista två är troligtvis drabbade av att några tillfällen fallit bort. Tidsrapporteringen avsåg aldrig att ha någon särskilt hög precision, utan skulle tjäna som ett ungefärligt mått för att man skulle kunna reflektera över tiden diverse uppgifter tog och kunna se så ingen enstaka individ avviker för starkt från resterande gruppen. Sett till den totala tiden som loggats skulle man kunna argumentera för att gruppen inte till fullo använt sig av de 20h/vecka som kursen i teori skulle göra anspråk på. En anledning till detta kan vara att ingen i gruppen var särskilt anförtrodd med utvecklingsmiljön *Android Studio* sedan tidigare, vilket kan ha lett till att vi överskattat tidsåtgången för en del user stories. Detta tillsammans med att inte alla tillfällen loggats och vi ändå har haft en enhetligt uppfattning av att de planerade målen uppfyllts och motsvarande värde levererats gör att gruppen ändå menar att detta är godtagbart.

En annan problematisk del när det gäller tid är att det var svårt att balansera den här kursen med att man har en annan kurs eller kandidatarbete. På något sätt så ville man lägg all sin tid när man satt och estimerade user stories, men sedan inser man att all ens tid inte är med i viktningen. Även några mindre buffertar planerades medvetet in de veckorna när andra åtaganden var särskilt påtagliga.

Det var flera veckor då vi kände att vi hade lagt mindre tid än det som förväntades, men vi var ändå nöjda med våra resultat och arbetsbördan. Vissa veckor försökte vi lägga mer tid för att öka velocity och komma längre med projektet, men slutligen föll vi ändå tillbaka till vår vanliga arbetstakt. De sista två veckorna la vi mest tid. Detta berodde främst på att vi såg ljuset i tunneln och verkligen såg att vi hade möjlighet att färdigställa ett bra resultat som skulle göra vår PO nöjd.

B: Produkten som vi levererade efter projektet var verkligen MVP, och hade vi lagt mer tid så hade vi säkerligen fått en bättre produkt. Därför så är det något att sträva efter i framtiden, särskilt om det skulle ske på en arbetsplats. Samtidigt så är vi nöjda med det som blev, och hade vi gjort denna kurs igen så hade vi varit nöjda med att vi fick ett bra resultat och att ingen blev för stressad under arbetet. Fördelaktigt hade varit om man enbart fick lägga tid på ett projekt som detta, istället för att dela sin tid mellan två olika projekt.

C: Under ett framtida projekt hade det varit bra att hitta ett smidigare sätt att tidrapportera, kanske kan man göra det i sin IDE på något sätt. Det hade underlättat eftersom det är lätt att glömma bort att logga.

Kanske hade även tydligare schemaläggning av timmar hade gjort att man verkligen satt de timmar som man borde. Nu var arbetet väldigt fritt och man gjorde det när man behagade. Är man schemalagd är man inte lika fri, men man får antagligen mer gjort vissa pass och mindre andra.

Även bättre estimering av stories hade hjälpt tidsåtgången varje sprint. Eftersom vi verkligen ville klara de stories som vi ålade oss varje vecka blev det ofta att vi planerade vår velocity i underkant och därmed spenderade mindre tid vissa veckor när vi fortfarande hade arbetskapacitet kvar. Andra veckor så tog vi på oss alldeles för mycket och fick då också problem med att våra stories kändes övermäktiga, då blev inte heller så mycket gjort. Att till nästa projekt estimera mer precist hade därmed hjälpt för bättre tidskonsumtion.

Design decisions and product structure

A: ***Design Decisions***
Tidigt i projektet fick vi av vår PO i uppgift att utveckla applikationen på en mobilplattform, eller rent specifikt en surfplatta. Detta gjorde att vi var tvungna att välja en utvecklingsmiljö utefter det. Innan hade vi varit inställda på utveckla applikationen i IntelliJ eller Eclipse men p.g.a. att det var begränsat till en surfplatta för Android var vi tvungna att tänka till. Vi valde att utveckla i Android Studio för att det använder Java vilket alla i gruppen kände sig bekväma med och för att var så pass populärt att det kändes som ett tryggt alternativ eftersom det är specificerat för utveckling till Android plattformar.

Ett annat viktigt steg i processen var hur vi valde att lagra all data. Eftersom applikationen till stor del handlar om just lagring var det nödvändigt att detta fungerade i tidigt skede. Ingen i gruppen hade någon större erfarenhet när det kom till lagring vilket var en stor anledning till att det ej valdes att bygga en intern databas istället sparades datan i textfiler. Detta för att det ansågs som enkelt och efter samtal med handledare stärktes det eftersom han var positiv till att det var så pass simpelt. Detta ansågs också som att jobba agilt eftersom det var en prioritet som behövde vara klart tidigt, därför prioriterades en fungerande lösning framför en möjligtvis bättre men mer komplicerad lösning. Eftersom vi heller inte hade någon tidigare erfarenhet kring databaser kändes det riskabelt att göra något svårare än nödvändigt. Detta kunde både dra tid från att leverera viktiga delar av projektet samt riskerar att inte ens vara bättre om vi gjort fel. Det sågs också som fördelaktigt om man behövde föra över data mellan enheter då bara är att kopiera filerna till en annan. Ett problem var däremot att bristen på erfarenhet gjorde att det krävdes flera user stories för att göra slutgiltiga produkten vilket resulterade att koden som var skriven ofta skrevs om.

Genomgående är dock att inga större medvetna val har gjorts kring arkitekturen av koden. Mycket av valen görs i stunden och det fanns inte särskilt mycket övergripande över hur vi ska ta oss dit. Det beror delvis på ovanan i Android Studio och därmed lades mycket tid ner på att ens förstå det. Det ansågs också som att andra saker behövde prioriteras utöver det, att lägga ner tid på att få struktur på projektet ansågs inte som högsta prioritet när man har tidsbrist och behöver lära sig en ny miljö att utveckla i, speciellt i samband med att värde behöver synas för kunden kontinuerligt. Istället valdes det att läggas fokus på det som skapar direkt värde för kunden, t.ex. få funktionalitet som lagring, inskrivning av data, generering av ordernumber mm. Projektekt blev också hindrat av ovanan att jobba agilt vilket gjorde att man behövde använda trial-and-error för att ens få koll på effektiva user-stories. Därmed vad det svårt att göra upp en plan både för nuvarande veckan och för framtida veckor.

Några aktiv val som gjordes var exempelvis att mängden anställda syns mellan varenda gång man lägger till en. Redan i mock-upen gjordes det ett val mellan dessa två varianter där den andra är att så fort man är klar med en anställd går man direkt till att skriva in information kring den nästa. Det valdes dock att gå på förstnämnda vägen för att ge översikt från användarens perspektiv när de ska lägga till en ny. Det blir då enklare för användaren att förstå när de lägger till en ny och hur många som lagts till.

Det valdes också att läggas mer fokus på att få en MVP, alltså minimal viable product, istället för att göra programmet visuellt tillfredsställande eller ge en unik känsla över vårt program.Genomgående för projektet är därför ett högt fokus på att få funktionalitet framför det grafiska. Även andra saker som designmönster har nedprioriteras för att allt ska vara användbart.

B: Nästa projekt
I nästa projekt skulle det vara bra om programmet fungerade även för andra mobila plattformar. I nuläget är programmet rätt hårdkodat till just den plattform som användes i simulatorn men om användaren exempelvis tänkt byta plattform skulle programmet eventuell inte fungera riktigt lika bra. Detta går även in på nästa punkt vilket är att designen generellt behövde lite mer kärlek. I ett annat projekt skulle det vara bra om det lades lite mer energi på den så att användarvänligheten höjs och blev mer tilltalande.

Även för arkitekturen skulle det i framtida projekt behövas planering för hur strukturen av själva koden ska se ut. Det skulle vara bra om man tidigt involverade fler activities kopplat till fragments på ett mer grundligt sätt så allt kod inte hamnar på samma ställe. Detta skulle göra det enklare att uppdatera koden och göra det enkelt att återvända till det utan att vara förvirrad. En mer grundlig genomgång och djupare planering underlättar också mot att inte behöva skriva om kod, vilket gör att mindre tid läggs på att skriva om gammal kod som inte håller längre.

C: A->B
För att lyckas få en kod som har lite mer struktur och design överhuvudtaget skulle det vara bra om man redan innan själva kodningen börjar lära sig lite mer om Android Studio generellt. Under projektet stötte vi på flera saker såsom en Fragment Manager som vi aldrig lyckades implementera för att vi inte hade någon kunskap om vad den gjorde. Det verkade vara någon sorts standard att använda sig av det och vissa problem skulle säkert undvikits om det implementerats. Redan nu efter man gjort ett projekt har man lärt sig en del och därför hade det varit enklare att sätta upp en mer effektiv bas i framtida projekt men för att undvika technical debt hade en grundlig planering hjälpt för att

implementera dessa rätt. Detta skulle även underlätta för hur man implementerar för flera plattformar än bara en surfplatta så att värden inte blir hårdkodade. Istället för att göra User Stories som man vet behöver återvända till, gör man mer permanenta lösningar som varar länge och inte behöver göras om längre fram i projektet. Detta görs enklast också med en tidig djup planering där man tänker flera steg fram istället för att bara få görbara user stories för veckan. Nu när man lärt sig mer om agilt arbete och vet mer om user stories är det säkert också enklare att ge sig in i liknande projekt i framtiden, för vad som är effektfulla user stories.

När det kommer till den ytliga designen hade det varit praktiskt om vi gjort mer omfattande user stories, som täckte in lite mer visuella element och användarvänlighet. Det hade t.ex. Läggas större krav på detta i definition of done att saker ska se bra ut och vara tilltalande till en viss gräns, så att en viss mängd energi läggs på det i varje iteration men att undvika att inte alltför mycket tid slösas på detaljer. I nuläget las väldigt lite fokus på detta och nästan allt fokus lades på funktionalitet, genomgående kändes det som att om bara lite mer tid lades på det visuella skulle det gjort väsentlig skillnad. Det skulle också vara bra för projektet som helhet om man varit mer medveten om designmönster att man kanske tog något möte för att påminna sig själv om vissa mönster som kan komma till användning.I vårt projekt blev det mycket på känsla, istället för medvetna val kring vilka designmönster som används, vilket kan göra att man glömmer vissa detaljer.

A: ***Technical Documentation***

För att dokumentera vår kod har vi främst använt JavaDoc för att kommentera funktionen på alla “public” metoder i programmet. På privata metoder har vanliga kommentarer använts när det ansetts nödvändigt, t.ex. om det är en väldigt komplicerad metod, men detta är långt ifrån alla och det finns många ställen som hade mått bra av att kommenteras mera. Just nu är vår kodbas relativt svårläslig, särskilt om man tittar på den som utomstående.

I början av projektet användes hemsidan figma.com för att rita upp en prototyp av hur programmet skulle se ut i slutändan. Denna prototyp finns ännu kvar och slutprodukten blev väldigt lik prototypen.

Annan sorts dokumentation såsom UML-diagram har vi inte använt i projektet. Detta för att i början hade vi mycket fokus på att få igång någonting av värde att kunna visa upp i slutet av sprintarna. Detta ledde till att dokumenteringen nedprioriterades. Vårt arbete har trots detta flutit på bra då vi hade mycket god intern kommunikation inom gruppen, vi hade flera regelbundna möten i veckan då vi gick igenom vad vi arbetade med och gick igenom koden man arbetat på med varandra. Vi dokumenterade även en del på vårt scrum-board på trello där vi på de individuella user story-korten kommenterade eventuella detaljer som var bra att veta.

Bristen på tydlig dokumentering såsom UML-diagram och liknande har lett till att vår programkod är svår att tolka om man inte själv är väldigt insatt i den. Det innebär också att vi har svårt att avgöra när programstrukturen är ohållbar, och programmet innehåller en del så kallad “technical debt” som så småningom hade behövts göras någonting åt om man önskade fortsätta jobba på det en längre framtid efter kursen. Dock har vi ännu inte känt av detta så mycket och har inte tyckt att det är ett problem värt att lägga vår tid på eftersom kursen är klar i detta läget, och vi lade i stället vår tid på att få ett fungerande program.

B: Nästa projekt önskar vi lägga mer vikt på dokumentation för att koden ska vara lättare att tolka och bygga vidare på. Om man gör ett program till ett företag eller liknande som ska arbetas på länge är det viktigt att undvika technical debt som man lätt missar när man först implementerar en funktion, men som sedan visar sig om man behöver ändra funktionen eller bygga vidare på den. Detta görs enklast genom att tydligt kommentera sin kod så att den är lätt att sätta sig in i, samt att samla alla klasser och filer som används i ett UML diagram så att man undviker så kallade "circular dependencies" osv.

Den muntliga interna kommunikationen var mycket bra under projektet, så under vidare projekt önskar vi ha kvar samma nivå och bygga på det.

C: I början av ett projekt är det viktigt att understryka vikten av dokumentering till alla som arbetar på projektet. Det är mycket enklare att planera ett programs uppbyggnad med interaktion- och klassdiagram innan man börjar skriva kod, som man sedan kan jobba utifrån under tiden man utvecklar. Det är också mycket enklare att alla delaktiga utvecklare kommenterar sin kod tydligt medans de skriver den så att det fortfarande finns fräscht i minnet, äldre kod är många gånger mer arbete att dokumentera.

A: ***Code Quality Assurance***
Näst intill all kodning som utförts under projektet skedde i grupper om två eller tre. Detta ledde till en relativt jämn kodkvalitet över hela programmet eftersom man nästan alltid programmerade tillsammans med någon som jobbat på en annan funktion än en själv föregående vecka. För att se till att inga stora uppenbara missar i koden slank med införde vi efter några veckor peer review av koden, dvs. innan man mergade sin bit kod man arbetat på med resten av programmet gick man igenom den tillsammans med någon annan i gruppen som inte varit med och arbetat på den. Vi har inte använt oss av tester av koden utöver dessa reviews.

Som nämnts i stycket ovan fungerade detta för oss men vi började i slutet känna av lite technical debt, och om det hade varit ett större mer omfattande program hade vi behövt börja omformatera delar av koden som inte höll standarden.

B: För att hela programmet ska ha en hög kodstandard krävs det att alla gruppmedlemmar är medvetna om god kodpraxis samt att man skriver tester till sin kod samtidigt som man skriver koden. Detta leder till mycket lättare att hitta eventuella buggar och buggar som kan uppkomma i sk. edge cases, som inte ofta uppkommer under körning av programmet.

C: För att uppnå det som nämns i B ovan krävs en planering av hur kodstrukturen och kvalitén ska se ut innan man börjar skriva programmet. Man bör också vara överens om hur man ska testa sina funktioner. Vi var nöjde med den interna granskningen av koden vi hade innan varje merge så tillsammans med dessa åtgärder kan man försäkra sig om en kod av högre kvalité som är lättare att bygga ut och/eller testa.

Application of Scrum

A: the roles you have used within the team and their impact on your work

Vi var väldigt försiktiga när vi skulle tillsätta roller i början. Det enda som egentligen var givet var att vi skulle ha en Scrummaster och en sekreterare. Med tiden växte även rollerna PO-”i projektet” och KPI-mästare fram. PO-”i projektet” skapades eftersom det var en av medlemmarna som hade en naturlig koppling till PO:n och vi ville sära på det ansvaret från Scrummastern. PO-”i projektet” ansvarade för att möten bokades in med vår PO, och han höll även kontakten med PO:n genom sprintsen om gruppen hade några snabba frågor. KPI-mästaren hade ansvar för att alla medlemmar loggade sina svar för KPI-erna på Flourish, och även att sammanställa resultatet av dessa. KPI-master medförde att det bara var en person som hade koll på utfallet av KPI-er. Att rollen behövdes indikerar sannolikt att KPI-er med fördel hade mäts på ett delvis annorlunda sätt för att minska behovet av admin. Scrummasterns roll har istället varit att leda planeringen under sprint-plannings och sprint-reviews samt föra gruppens röst under vissa möten. Sekreterarens roll innebar i början att man skulle skriva ner allt som sades under sprint-plannings och sprint-reviews. Men detta blev mer löst med tiden, och tillslut fanns det inte riktigt några normer för vem som skulle skriva. Om man ska sammanfatta användningen av roller i projektet så har det varit ytterst löst. Lösa roller kändes naturligt då behovet av koordination var begränsat och vi upplevde det som positivt att alla fritt kunde ta initiativ oberoende av roller och hierkier. De flesta individer i gruppen har fyllt samma roll, och det har inte funnits något behov av att ha en tydlig ledare.

En annan del som har påverkat vårt arbete är de olika gruppindelningar vi har gjort genom projektet, när vi exempelvis parprogrammerat. Den första veckan delade vi upp oss två och två. Det fungerade dåligt på grund av dåligt formulerade User Stories som dessutom var alldeles för stora. Därför testade vi veckan därpå att några arbetade enskilt och några i grupp. Det gick inte heller så bra, eftersom de som var själva inte hade koll på hur de skulle lösa sina problem och för att gruppen inte hade möten tillräckligt ofta. Under projektets gång testade även att jobba tre och tre och att dela upp oss med en IT-student och en I-student i varje programmeringspar.

Slutligen kan man säga att programmeringspar fungerat bäst, men att det också har funnits grupper som fungerat bättre än andra. Rotationen av att olika medlemmar hamnat på olika uppgifter varje vecka har passat sämre för några som känt att det blivit en uppförsbacke att ställa om, men det har även gett fördelar i att problem som sträckt sig över flera sprints till sist kunnat lösas, då nya ögon gav nya idéer och slutligen lösningar.

B: I framtida projekt så kan roller definitivt vara viktigt, men då bör de vara definierade efter gruppmedlemmars färdigheter. I detta projekt har vi varit väldigt jämna i vår kunskapsnivå men vi har även haft kravet att delta jämnt i de olika arbetsuppgifterna. Hade man delat upp gruppen i olika ansvarsområden, exempelvis design och utveckling så hade var och en kunnat fokusera mer på sin roll. Om en roll för en individ är tydligt definierad med vad man gör och inte gör, så blir det lättare att fokusera på sitt arbete. I verkligheten hade exempelvis Scrummastern inte haft rollen som utvecklare, utan istället fokuserat på hur arbetet skulle genomföras och varit mer av en supporter för de som utvecklade. Dessutom hade arbetet inom gruppen lättare kunnat göras vertikalt, om man delat upp gruppen på ett designteam och ett utvecklarteam. Nu har det istället blivit det ena eller det andra.

När det gäller parprogrammering så behövde vi nog gå igenom denna process för att veta vilka grupper som fungerar bättre och sämre. I ett framtida projekt så kommer nog samma procedur behövas, eftersom det då kommer vara andra människor man arbetar med, och det är svårt att veta vilka team som funkar innan man testat.

C: För att få en god uppfattning om medlemmarnas kunskap och erfarenheter så hade man kunnat diskutera hur det har gått till i tidigare projekt och prata om förväntningar och önskningar inför det nya projektet. Folk som känner varandra har bättre förutsättningar för att arbeta tillsammans, så genom att umgås och teambuilda med en ny grupp skapar man goda förutsättningar för ett framtida arbete. Att behålla grupperingar från tidigare kan kännas enkelt och självklart till en början, men det kommer hindra teamet från att bli just ett team. För att etablera roller kan man även behöva testa några olika roller, för att se hur gruppen fungerar bäst.

A: the agile practices you have used and their impact on your work

De agila praktiker som använts under kursens gång är: Sprint planning, product-backlog, kontinuerlig integration, parprogrammering, user stories, estimering av user stories, refaktorering samt sprint review och retrospective.

Sprint planning har varit grunden för projektets planering. Under dessa skrevs stories, prioriterades, estimerades samt delades ut till gruppmedlemmarna. Utan sprint planning så hade ingenting fungerat. I början var det däremot väldigt trögt eftersom det kändes svårt och tidsödande att både skriva stories och sedan även estimera dem. Ingen av de stories som skrevs första veckan blev klara utan fick skickas tillbaka till backloggen och brytas upp mer, då de var alldeles för stora. Problemet löste vi till veckan efter genom att börja estimera våra stories och vi lyckades få till några user stories som kunde läggas till i done. I början estimerade vi genom fibonacci-serien, där ett tal motsvarade antalet timmar vi förväntade oss att storyn skulle ta. Vi insåg snabbt dock att det blir konstigt att estimera för parprogrammering, eftersom antalet personer som läggs till att lösa uppgiften inte minskar tidkonsumtionen linjärt. Därav gick vi sedermera över till att använda tröjstorlekar som lösning i vår planning poker. Vår sprint planning blev också bättre efter att vi började fråga PO på våra sprint reviews om vad som han helst såg som nästa viktiga funktion i applikationen, och prioriterade backloggen efter det.

Under våra sprint retrospective reflekterade vi över sprinten som varit och skrev ned det i vår team reflection. Det hjälpte oss att utvärdera arbetet under den gångna veckan och på så sätt ta reda på vad vi vill ta med oss in i nästa sprint och vad vi behövde förbättra.

Ett konstant problem som vi kände med våra stories var att de var svåra att få vertikala, speciellt i kombination med att de skulle vara independent. Att skapa en vy för en sida som är en uppföljning till en tidigare, samt se till att det kan navigeras till den nya sidan, blir väldigt beroende av den tidigare sidan, och ifall navigationen inte är klar skapas inget värde av storien heller. Det är essensen i flera de problem vi stött på. Dessa problem minskade däremot med tiden. Ju större applikationen blev, desto mer isolerat kunde arbetet ske.

En annan utmaning med utformningen av vertikala user stories, var att vi emellanåt behövde göra omfattande arbete i backend. Exempel på det är när lagringen av data skulle implementeras, när algoritmen för generering av filterkod skulle skrivas, och då word-dokument motsvarande kuponger och order skulle genereras. Att då även inkludera betydande front-end element, i syfte att få vertikala user stories, hade resulterat i för stora user stories i stället. I de fallen tummade vi på kravet om vertikalitet, eller inkludera endast minimala frontend inslag (såsom tillägget av en knapp för att generera dokumentet).

Sedermera var just värde någonting vi tyckte var krångligt. Att hela tiden leverera värde skapar ett tvång av att allt som implementeras skall ge konkret funktionalitet. I kombination med sprintarnas

korta tidsrum prioriterades ofta en minimalt fungerande lösning, före en bra lösning. Detta ledde slutigen till att spaghettikoden som skrivits började stelna och klumpa ihop sig. Vi hade tillfällen då en ändring på ett ställe orsakade fel någon annanstans, vilket var väldigt jobbigt. Därför behövde vi göra en lite större refaktorering av koden för att känna oss bekväma med att arbeta med den igen.

Som tidigare nämnt tyckte vi också att det var svårt att få till independent user stories. När det inte lyckades försökte vi lösa problemet genom kontinuerlig integration för att således undvika konflikter vid merge. Vid ett tillfälle efterföljdes inte principen och vi fick stora problem vilket resulterade i att vi förlorade en del kod. Den slutgiltliga lösningen blev att köra en reset till en tidigare commit och därefter manuellt kopiera över kod från branchen in i master, vilket tog mycket tid utöver den veckans specificerade user stories.

Sedan användes även parprogrammering som tidigare beskrivits.

B: En förändring vi hade kunnat göra i ett framtida projekt är att träffas på plats. Det var inte optimalt under detta projektet, men om vi hade träffats på plats hade antagligen fokuset varit bättre. Det kan vara lite komplicerat att lära sig en ny metodik när man sitter i Zoom, för man fastnar i att titta på hur de andra i gruppen reagerar på det man säger. Ser alla trötta ut, så stressar man lätt vidare till nästa punkt på agendan, i stället för att verkligen se till att alla har fattat innebörden.

Hade vi exempelvis suttit i samma rum när vi skrev våra första user stories, så hade de antagligen varit bättre än det som blev. En annan sak som hade förbättrats av mer fysiska möten är att vi hade kunnat träffats för daily sprint meetings och att vi då hade fått en bättre översikt över vad de andra i gruppen höll på med. Det är bekvämt att köra via Zoom, men ibland är det för bekvämt.

Slutligen ser vi att en fördel med att sitta tillsammans är att det är lättare att ta del av varandras kunskap. Sitter man bredvid en gruppmedlem och har en fråga om något den har koll på så är det en låg tröskel till att be denne om hjälp. Att i stället kontakta någon digitalt tenderar att bli mer onaturligt, omständigt och drar ut på tiden, varför det i många fall nog inte blir av. Således kan arbete på plats underlätta etablerandet och utnyttjandet av en gemensam kunskapsbank.

Själva designen av programmet hade gynnats av att vara mer domändriven, alltså mer strukturerad och lättnavigerad. Även vi som kodare har ibland haft svårt att hitta vilken kodbit som står vart. Vi gjorde ett försök mitt under projektet att refaktorera, men förstod ganska snabbt att det var något vi borde ha haft med oss redan från början. Vi fick PO:ns affärsmodell tydligt beskriven och hade med hjälp av den kunnat bygga en tydlig domän.

C: För att kunna använda de agila metoderna på rätt sätt krävs is i magen. Det tar tid och är inte alltid kul, men genom att planera och strukturera varje steg på vägen kan man visa värde och även få en produkt som fortsätter vara relevant för en lång tid, eftersom den alltid kan uppdateras.

A: 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 your DoD? Did the feedback change your way of working?)

I vårt projekt har vi haft Jörgen Hedquist på Hear Nordic AB som vår Product Owner (PO). Vi har i princip träffat honom varje fredag innan veckans sprint retrospective. Under våra möten med PO:n

ställde vi frågor kring hans förväntningar på vår produkt, samt att vi ställde de frågor som dykt upp för utvecklingsteamet under veckan. Efter en tid började vi även be Jörgen att ge hos respons i form av en KPI, för att vi lättare skulle kunna mäta hans feedback.

Vi har haft tur att få en PO som har varit väldigt tillmötesgående och positiv. Vi fick alltid bra reaktioner på det vi gjort, och det uppstod aldrig några problem. Därför så fick vi sällan feedback som vi kunde arbeta med. Vi behövde inte heller ändra om i prioriteringen med våra user stories, utan allt gick bara som vi hade tänkt.

Detta har vi sett som både något dåligt och något bra. Vi är väldigt glada att vi har fått en god uppfattning om den produkt som vår PO efterfrågade. Men samtidigt så känns det som vi har varit lite bortskämda. Det känns som om verkligheten är flera steg bort, och vi kommer ställas inför högre krav i framtiden. En anledning till detta är att det arbete vi gjort nu har varit gratis, och att vi också varit tydliga till PO:n i början på projektet att det inte var säkert att vi skulle hinna bli klara. Hade projektet skapats under en stram tidsplan och även kostat mycket för kunden så hade vi antagligen stött på fler problem i kommunikationen.

B: I framtida projekt av liknande karaktär så hade det varit bra att tidigt förmedla för en potentiell PO att vi hade velat ha mer kritik och högre krav, för att göra arbetet mer likt "verkligheten". Syftar man istället på ett projekt med verklig karaktär, så är det viktigt att göra en tydligare sprint review där man inkluderar sin PO och förutom att mäta hur väl arbetsgruppen tycker att den har lyckats, även mäter hur väl PO:n tycker att arbetsgruppens känslor och åsikter stämmer överens med PO:ns. Överhuvudtaget hade PO:n varit mer inkluderad om projektet varit "på riktigt". Nu kunde vi inte med att ta den tiden från vår PO, eftersom vi inte var säkra att vi skulle kunna leverera en färdig produkt på slutet, och eftersom vi bett honom medverka, och inte själva blivit anlitade.

C: För att göra allt så tydlig som möjligt för både PO och arbetsgruppen hade man behövt att träffas och göra upp en plan som innefattar såväl PO:ns som arbetsgruppens idé av en MVP och DoD. Genom att definiera dessa från start är kraven mer specificerade från början, och man har något att gå tillbaka till om saker skulle bli oklara vid ett senare skede.

A: 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 som vi använde oss av under kursen var: Android Studio, GitHub, GitHub Desktop, Trello, Google Drive, Figma, messenger, google calendar, Zoom och Flourish. Tanken var från början att använda oss av IntelliJ istället för Android Studio, men när vi blev överens om att en app skulle utvecklas bytte vi till Android Studio som IDE.

Verktygen som vi hade till kommunikation, alltså Messenger, Zoom och Google Calendar, hade alla i gruppen redan tidigare erfarenhet med. De verktyg som några i gruppen var mer ovana vid var just GitHub och GitHub Desktop. Anledningen till att vi valde GitHub Desktop var för att det är en lägre tröskel än att gå via terminalen. Däremot är konceptet av version control inte helt enkelt och därför försökte de som använt Github Desktop sen tidigare att förklara och hjälpa de andra. Gruppen hade exempelvis ett möte där vi gick igenom grunderna i Git för de som aldrig använt det förr. Att hjälpa varandra har också överlag varit ett bra sätt lära sig. Sedermera var vi ändå inga experter på verktygen sedan innan, vilket ledde till att vi lärde oss ny funktioner. T.ex. möjligheten att via GitHub Desktop

få upp merge konflikter i Android Studio, för att jämföra branchen med main, sida vid sida, och acceptera eller förkasta ändringar. Sådana funktioner hittade vi exklusivt genom att googla om programmet och i olika forum, främst StackOverflow.

De verktyg som ingen använt tidigare var Flourish och Android Studio. Flourish användes för att hålla koll på våra KPI:er genom en chattfunktion som kan integreras via Messenger, Slack eller köras direkt i webbläsaren. Som användare är det väldigt enkelt att förstå verktyget eftersom det går att på att svara på frågor, det som kan varit lite klurigare var att sätta upp frågorna och sedan sammanställa resultaten, vilket vår KPI-master, Kailash, ansvarade för.

Att förstå sig på Android Studio visade sig vara mycket svårare. Själva programmet och dess GUI är väldigt likt det som återfinns i IntelliJ, vilket inte är konstigt med tanke på att programmet är baserat på just IntelliJ. Eftersom flera i gruppen hade arbetat tidigare med den IDE:n så var inte det något problem, problemen uppstod ofta istället i själva programmeringen emot android API:n. Vi valde att bygga programmet i språket Java då alla jobbat med det tidigare. Att bygga en app i Android Studio är ganska likt ett Java-program med fxml för att visa vyer. Det svåra var att förstå hur android biblioteken fungerade med varandra. Det var mycket nytt att förstå och många var förvirrade, t.ex. skillnaden på fragment och activity. Likt de utmaningar vi stötte på med GitHub desktop så använde vi oss främst av Google för att hitta lösningar. Det ledde ofta även här till StackOverflow. Ibland när vi hade specifika idéer som vi ville implementera var tutorials på YouTube också väldigt hjälpsamma.

B: I nästa stora projekt hade det också varit bra att använda en blandning av verktyg som arbetsgruppen är erfarna med men också några som de inte är. Det gör att det inte blir alltför mycket nytt att lära sig, samtidigt som det ska användas. Däremot hade själva inlärningsmetoderna kunnat förbättras. Som beskrivet användes Google mycket för att lösa svårigheter som uppkom. Problemet med det är att lösningarna som hittas oftast är alldeles specifika och egentligen inte särskilt upplysande. Att hitta ett stycke kod på StackOverflow och sedan kopiera rakt in i Android Studio är inte särskilt givande. Sedermera om den givna koden inte heller fungerar är det också otroligt svårt att försöka felsöka. Vad vi eftersöker till nästa projekt är att ha mer tålamod för att få en mer grundlig förståelse för verktygen och därigenom bygga en större grund för problemlösning.

En annan sak som hade underlättat när man kört fast, är om det hade funnits en handledare att kontakta när som helst. Vi träffade vår handledare en halvtimme 8:30 på måndagsmorgonen, och det var många gånger vi kände att vi kom på frågor mitt i veckan som vi var tvungna att vänta med. Detta gjorde att vi fick lösa problem på det sättet vi trodde var det bästa. Men hade det funnits någon att ha ett snabbt möte med när man kört fast så hade det underlättat. Även en handledare i programmering hade gjort stor skillnad, eftersom de kanske hade kunnat styra en åt rätt håll när man fastnat. Men vi förstår även att detta är svårt, när alla gör så olika projekt.

En ytterligare tanke är att använda välanpassade verktyg till uppgiften och inte ha några överflödiga heller. Ju mindre system som används desto lättare blir de att hålla koll på. Vi använde oss av Google Drive för olika typer av textdokument och dylikt. Eftersom vi ändå lade till flera av dem till GitHub repot, såsom team reflections, hade vi kunnat lägga upp allt där istället, och helt skippat Drive. Angående välanpassade verktyg så ju mer lämpade de är för den specifika uppgiften, desto lättare är det att förstå funktionerna och hur det fungerar.

C: För att välja välanpassade verktyg krävs det att ha en bra koll på både sitt scope, men också vad verktygen faktiskt erbjuder för funktioner. Alltså, granska de verktyg man sätter upp, och därigenom

förstå mer vad de erbjuder och hur de fungerar. På det sättet märks även ifall ett hjälpmedel blir överflödigt i kombination med de andra och kan tas bort.

För att få en mer grundlig förståelse för verktygen, specifikt de mer tekniska, hade det varit väldigt nyttigt att första kontakten med dem inte är i själva arbetet, utan att man även har inlärningsfas för dem. Att kolla någon snabbkurs eller tutorial om grunderna i Android Studio innan man hoppar rakt in i arbetet hade hjälpt mycket för att förhindra den initiala förvirringen som uppstod när vi inte visste i början hur vi skulle göra. Även att läsa dokumentationen av biblioteken som används hade varit väldigt nyttigt för att få en större grundkunskap.

A: relation to literature and guest lectures (how do your reflections relate to what others have to say?)

Vi i gruppen var överens om att vi inte känt att det funnits tillgång till litteratur eller gästföreläsningar som hjälpt oss genom projektet. Den litteratur vi fick tillgång till var den innan projektets start, och även om den gav en introduktion, så gav den oss inte så mycket när vi höll på med projektet. Vi kände att mycket av teorin behövde “upplevas” och arbetas med för att förstå vad vi egentligen skulle göra.

De vi framförallt saknade var mer handledning och tillgång till något att bolla med när man fastnar i olika problem. Vi hade önskat att kursen hade erbjudit fler handledarmöten, och kanske även någon som man hade kunnat ställa frågor till när man fastnade med sin kod. Vi har varit flitiga användare av Google, och vi har lyckats lösa våra problem, men ibland är det skönt att ha en guide som ställer rätt frågor på vägen.

B: Till nästa projekt hade vi gärna läst på mer innan hur man faktiskt arbetar enligt arbetssättet. Det kanske är meningen, men första veckan blev verkligen bara ett famlande i mörker, speciellt också när första handledningstillfället inte var förrän början av nästa sprint. Förhoppningsvis kommer vi alla nu att ha mer erfarenhet inom ämnet, från den här kursen, men det finns fortfarande självklart mycket kvar att lära.

Sedan hade vi gärna också relaterat mer till litteratur, eftersom det blev något i projektet vi ganska mycket glömde bort. Problemet vi såg där var att det finns mycket litteratur inom ämnet med mycket olika åsikter om vad som bör och inte bör göras. Speciellt när vi var nya inom området blev det svårt att välja vad vi ska tro, det blir lätt att man nöjer sig med första bästa. Därför tänkte vi att till nästa projekt, när vi är lite varmare i kläderna, blir det lättare att relatera till vad man tycker fungerar dåligt och bra med arbetssättet, och på så sätt faktiskt kunna relatera till litteratur.

C: För att bli mer insatta i det agila arbetssättet inför nästa projekt hade det behövts mer förberedande studier. Som tidigare nämnt har vi alla nu mer kunskap om ämnet, vilket också gör att tröskeln för att lära sig via litteraturen därav är lägre innan. Dock krävs det disciplin för att arbeta med ett projekt innan det ens har startat, men det tror vi att vi klarar.

Sedan hade gästföreläsningar också varit något vi gärna hade gått på för att lära oss, om möjligheten finns nästa gång.