DAT257, Final Team Reflection, Group 3

Philip Tibom

Raoul Kent

Alfred Kjeller

Marcus Gedda

Joar Hellqvist

5 juni 2020

Innehåll

1 Kundvärde och omfång

1.1 Introduktion till kundvärde

I golf finns det ett antal olika sätt att räkna poäng, varav det allra vanligaste poängsystemet i Sverige kallas för Poängbogey. I golf finns det alltid en siffra på varje hål som indikerar hålets Par. I poängbogey får spelaren 2 poäng om spelaren slår samma antal slag som hålets Par-siffra indikerar. Exempelvis om ett hål har Par 3, och spelaren slår 3 slag, så får spelaren 2 poäng. Om spelaren slår ett slag över par (4 slag), så får spelaren istället 1 poäng, eller om spelaren slår 1 slag under par (2 slag) så får spelaren 3 poäng och så vidare. Om en spelare genomgående slår samma antal slag på varje hål som hålets par indikerar, kallas denna spelare för scratchspelare och det indikerar att spelaren har ett så kallat nollhandicap. Eliten inom golf spelar därför utan handicap och förhåller sig endast till hålens Par.

Den stora majoriteten av golfare är dock inte eliter och eftersom det finns en mängd olika skicklighetsnivåer inom sporten, så har varje spelare ett personligt golfhandicap. Dessutom finns det olika svårighetsgrader på olika banor, samt olika utslagsplatser som kallas för tee, där varje tee har en varierande svårighetsgrad. Spelaren kan alltså välja vilken tee den vill slå ut bollen ifrån beroende på sin egen förmåga och det handlar oftast om distansen mellan teen och hålet. I och med dessa olika svårighetsgrader som skiljer sig mellan banor, men även mellan olika tees, så har svenska golfförbundet upprättat 2 variabler som kallas för Slopevärde (slope rating) och Banvärde (course rating). Dessa två värden är unika mellan varje bana och tee för att spelaren skall kunna beräkna ett så kallat spelhandicap. Spelhandicapet är alltså det effektiva handicap som spelaren använder för att beräkna sina poäng på den specifika golfrundan. Exempelvis kan en spelare ha ett personligt handicap på 36, men det effektiva spelhandicapet på en något svårare bana kan bli 39.

Det effektiva spelhandicapet behöver beräknas vid varje spel och görs med följande formel:

$$\textit{Spelhandicap} = \text{Personligt handicap} \times \frac{\text{Slopevärde}}{113} + (\text{Banvärde} - \text{Banans totala par})$$

Att ha ett handicap innebär att spelaren får fördela ut extraslag på vissa hål, som spelaren får tillgodoräkna sig i sin uträkning av poängen. Med andra ord, spelarens personliga Par på hålen förändras beroende på spelarens handicap. Om en spelare har 36 i spelhandicap och spelar en 18-hålsbana så blir uträkningen relativt enkel, 2 slag fördelas ut på varje hål ($36/18 = 2$). Det blir dock mer problematiskt om exempelvis spelaren har 35 i spelhandicap och då antalet slag inte längre går jämt ut över antalet hål ($35/18 \approx 1.94$). Spelaren kan dessutom ha upp till en decimal i sitt handicap (ex. 35,3). Därför har varje hål ett ytterligare värde som kallas för Handicapindex som går från 1 till 18. Handicapindexet avgör i vilken ordning extraslag från spelarens handicap får delas ut på hålen. Första extraslaget delas alltid ut på hålet som har handicapindex 1, sedan index 2 och så vidare, hela vägen upp till 18. Handicapindexen skiljer sig dock från hålnumren, exempelvis kan första hålet ha handicapindex 6.

Slutligen behöver spelaren beräkna sitt resultat baserat på all ovanstående information, och med vår applikation görs allt detta automatiskt. Spelaren behöver endast mata in sitt personliga handicap och antal slag spelaren gjort på varje hål. Sedan sköter vår applikation alla uträkningar inklusive det effektiva spelhandicapet på vald bana och tee, samt spelarens personliga par på varje hål och slutligen poängräkningen. Förutom att användaren slipper göra en rad beräkningar, så finns det även ett kundvärde i att slippa bära på papper och penna. Spelaren slipper även memorera sina slag då applikationen kontinuerligt håller koll på varje slag för samtliga spelare i ronden.

1.2 Applikationens omfång

(A)

Applikationen skall innefatta såpass mycket funktioner och innehåll så att slutanvändaren kan använda den till det tänkta ändamålet. Projektgruppen har utvecklat applikationen så att slutanvändaren kan räkna slag på ett enkelt sätt. Slagen skall sedan presenteras i ett lättförståeligt scorekort där även poäng enligt poängbogey räknas ut. Slagräknandet i kombination med scorekortet för upp till fyra spelare anser gruppen vara projektets Minimal Viable Product (MVP). Detta är baserat på den information och behov som slutanvändaren har beskrivit.

(B)

I ett liknande framtida arbete hade gruppen kunnat etablera tydligare och mer konkreta funktioner som skulle ingå i projektets omfång. Gruppen etablerade till exempel att en slagräknare skulle ingå i applikationen men inte dess specifika utformning. En tydligare beskrivning av funktioner hade gjort det enklare och effektivare att programmera. Gruppen hade då kunnat uppnå en högre velocity om förarbetet hade varit gjorts mer noggrant. Hade gruppen, innan utveckling, undersökt hur UI:t skulle se ut, skulle även mindre jobb behövas göras i slutet av projektet för att uppnå ett bra UI.

($A \rightarrow B$)

För att kunna ta fram bättre funktionsbeskrivningar så behöver mer förarbete göras. Både i form av att undersöka hur användaren vill ha funktioner utformade och undersöka vad som är implementationsmässigt möjligt eller smidigast. Förklaringarna till berättelserna hade kunnat utvecklats vid sprint planeringen och inkluderats i uppgifterna. Högre krav kunde också ställts på intressenten för att förklara mer utförligt hur denne ville att saker och ting skulle utformas i applikationen. Detta hade kunnat resultera i att applikationen mer sannolikt kommer tillfredsställa slutanvändarens behov. Samtidigt är det svårt för gruppen som inte har jättemycket erfarenhet inom varken golf eller Android-programmering.

1.3 Framgångskriterier

(A)

Projektgruppen mäter hur lyckat projektet har varit utifrån hur mycket värde som skapats för slutanvändaren och intressenten samt hur väl projektgruppen lyckats applicera Scrum till arbetet. För att kunna mäta hur mycket värde som arbetet har skapat för användaren så behövs en god förståelse vad användaren vill ha. För att uppnå en god förståelse om vad användare vill få ut av applikationen, så har god kommunikation upprättats med slutanvändaren. Minst ett möte med slutanvändaren hölls varje sprint. Under mötena gicks applikationen igenom och efter respons från handledare började mer öppna frågor att ställas så att användaren fick förklara vad han ville att vi skulle ändra på, så att applikationen fungerade bättre som scorekort. Projektgruppen blev nöjd om vi kunde utveckla applikationen så att det blev enklare för användaren att räkna slag och poäng, samt sammanställa ett spelprotokoll över golfrundan.

Den andra saken som utgör huruvida projektgruppen har lyckats är hur väl gruppen har applicerat Scrum för att utveckla applikationen. Appliceringen av Scrum innebär att samtliga i gruppen deltar i arbetet och deltar på stand-up möten, samt upprätthåller god kommunikation med resterande gruppmedlemmar. Om samtliga i gruppen bidrar till arbetet och till användandet av Scrum så anser vi projektet vara lyckat. Detta kan till viss del mätas av de sociala KPI:erna. Gruppen strävar efter att nå så höga KPI:er som möjligt.

(B)

En förbättringsmöjlighet med arbetet är att kunden inte kvantifierade hur mycket värde som skapades. Kunden förmedlade att han var mycket nöjd med applikation och att den uppfyllde ett behov. Det kunden inte förklarade var hur mycket värde varje funktion skapade och vilka delar det var som skapade värde eller användning.

($A \rightarrow B$)

För att mäta värdet som kunden upplever hade gruppen kunnat kvantifiera en värdeskala så att kunden konstruktivt kunnat framföra vad som var viktigt. Som exempel hade kunden kunnat beskriva 1–10 hur mycket värde som varje funktion hade bidragit med. Hade en funktion då fått ett sämre värde från kunden kan gruppen undersöka vad det var som gjorde att just den funktionen inte var värdeskapande. En sådan förändring hade också säkerställt att varje sprint var värdeskapande för kunden.

1.4 Stories

(A)

Gruppen valde tidigt att använda sig av job stories istället för user stories, eftersom gruppen upplevde att job stories är mer specificerande för ändamålet. Job stories beskriver bättre när aktivteten utförs än user stories, vilket leder till att storyn blir mer specifik. Job stories har varit fördelaktiga för projektarbetet för att dem har varit beskrivande om vilka funktioner som ska ingå i sprinten för att möta behovet. Gruppen har gemensamt utformat job stories baserat på intressentens idéer och förslag. Gruppen har även hjälpt intressenten med att komma på idéer i de fall då intressenten inte alltid haft full koll på vad han vill ha. Exempelvis har skillnaden mellan att spara data lokalt eller remote förklarats, då intressenten inte har kunskap om skillnaderna. Dessa berättelser hamnar i arbetets backlog och prioriteras sedan utav gruppen (som agerat gemensam PO) för att sedan flyttas över till den aktuella sprinten. Efter att nuvarande sprint backlog har fyllts av prioriterade job stories så bryts de upp till uppgifter.

Därefter estimeras uppgiftens ansträngningspoäng och då sammanställs också berättelsens totala ansträngningspoäng som består av respektive uppgifter. Sammanfattningsvis jobbar gruppen med de uppgifter som bestämts i respektive sprint, där uppgifterna är anslutna till job stories som baserats på feedback från intressenten.

(B)

Ett förbättringsområde som gruppen hade i projektet var att job stories kunde varit mer förklarande. Job stories förklarade vad för funktion som gruppen ville utveckla men ingen förklaring på hur själva funktionen skulle utvecklas. Vilket även ledde till att tidsåtgången och svårighetsgraden på stories var svåra att bedöma på förhand. Dock korrigerades poängen på uppgifterna löpande under sprintens gång vilket ledde till att poängen ändå blev korrekta i slutändan.

Utformningen av berättelserna var främst baserad på det som intressenten specificerade. Samtidigt var projektgruppen som formulerade samtliga job stories. Vid vissa tillfällen föreslog projektgruppen vissa job stories till intressenten för att fråga om denne hade tyckt att det var bra att ha med den specifika berättelsen. Samspelet mellan intressenten och projektgruppen gjorde så att bra job stories kunde tas fram och samtidigt höja värdet av slutprodukten för användaren.

$(A \rightarrow B)$

Vad gruppen hade kunnat göra var att undersöka i ett tidigare skede hur svårt och eller hur lång tid varje job story egentligen tar. Genom att göra förarbetet och undersöka varje framtida uppgift kan också projektgruppen ge en bättre beskrivning av varje del. En tydligare beskrivning hade gjort det enklare att genomföra varje uppgift i respektive sprint. Samtidigt hade det varit väldigt svårt och tidskrävande att planera varje del mer noggrant, eftersom gruppen ej besitter tillräckligt med erfarenhet inom alla nödvändiga områden och ofta fungerade det bättre för gruppen att prova sig fram. Förarbetet hade även gjort det enklare för gruppen att planera sin tid mer effektivt och gruppen hade potentiellt kunnat undvika att planera in för mycket eller för lite arbete i respektive sprint.

1.5 Definition of done

(A) Projektgruppen hade inte en skriftlig definition of done i början av projektet. Utan hade en muntlig överenskommelse vad definition of done innebar. Den skriftliga definitionen skapades halvvägs genom arbetet och återfinns i GitHub repositoryt.

Gruppen använde sig av olika metoder för att säkerställa att koden var på en acceptabel nivå. Första metoden var att innan varje merge till master fick ske, så skall personen gå igenom samtliga delar av applikationen för att säkerställa att den nya koden inte bryter den existerande applikationen.

Dessa steg är utformade på följande vis:

1. Kör alla enhetstester (unit tests) för projektet.
2. Testa knappen ”Börja ny golfrunda”
3. Välj en bana
4. Lägg till nya spelare
5. Öka och minska antalet slag för en spelare.
6. Kolla så att siffrorna (slag och poäng) vissas korrekt i scorekortet
7. Återgå till huvudmenyn, undersök ifall rundan syns bland oavslutade rundor
8. Testa oavslutade rundor så att korrekt information visas.
9. Testa så att avslutade rundor visas på ett korrekt sätt med korrekt siffror

Allt eftersom funktioner läggs till förändras denna process för att täcka de testbehov som applikationen har. När ny funktionalitet lagts till har tester för de lagts till i listan ovan. Exempelvis tillkom det databastester som kollade att de nya funktionerna fungerade med databasen för banorna. Hela modellen och den lokala Android-databasen har 100% code coverage.

Införandet av tester, vilka presenterades ovan har varit framgångsrikt. Att alla både har kört de unit tester som konstruerats samt testat UI:t enligt den framtagna mallen har motverkat att buggar mergeas in i masterbranchen. I de fall då utvecklare har påträffat buggar vid genomförandet av dessa tester har dialog förts i mellan gruppmedlemmar för att identifiera och lösa problemet.

(B)

Problemet med den muntliga överenskommelsen var att det inte var säkert att alla hade samma syn på vad som tolkades som färdigt. Det är enkelt att glömma av vad som har sagts och hade det varit nedskrivet så hade projektgruppen alltid haft ett ramverk att förlita sig på.

Vid en optimal implementation av definition of done så borde alla utvecklare följa definitionen noggrant så att inte olika utvecklare producerar kod som är minimalt kompatibel med varandras kod. Några mindre problem uppstod pga. att dokumentet inte följdes noggrant, exempelvis hände det att när slagräknaren gjordes om så slutade scorekortet att fungera. Detta ledde inte till några allvarliga problem men resulterade i att arbete behövdes lägga på att göra koden funktionell. Arbetet hade undvikits om gruppen hade följt definition of done mer noggrant. Samtidigt hade gruppen en väldigt enad bild om vad som behövdes göras och kollas innan merge till master. Detta resulterade i att det aldrig blev några konflikter mellan gruppmedlemmarna, då få funktionella errors uppstod vid mergear.

($A \rightarrow B$) I framtida projekt hade gruppen i en tidigare fas kunnat upprätta en skriftlig definition av done. Ytterligaren en sak som bör förbättras är användingsfrekvensen av dokumentet som stöd vid merge till master. Efter att projektgruppen hade skapat den skriftliga definitionen så var det inte säkert att utvecklarna använde sig av dokumentet som stöd när utvecklarna mergade till mastern. Nästa gång skulle gruppen försöka göra dokumentet definition of done"enklare att komma åt så att fler utvecklar skulle följa stegen ännu mer noggrant.

En ytterligare förbättring hade varit att lägga till en kravlista på alla job stories för att säkerställa att gruppen var överens om vad job storyn innefattar. Dock hade det blivit väldigt tidskrävande att göra kravlistor vilket hade kunnat leda till minskad produktivitet.

1.6 Acceptanstester

(A)

Projektgruppen tolkade acceptanstester som tester som körs på koden innan en merge till master. Således bestod våra acceptanstester av de tester som definierats i sektion 1.5. Dock genomförde projektgruppen user acceptance tests, i form av användartester och intervjuer med intressenten, samt ytterligare användare, för att få feedback främst på användarvänligheten men även om den uppfyller den funktionalitet användaren fordrar. Denna typ av tester och kommunikation med intressenten genomfördes genomgående under projektet, men först mot slutet gjordes ytterligare användartester med andra parter. Testerna var dock inte formellt definierade eller utformade som acceptanstester. Acceptanstesterna var därför begränsade men gav ändå värdefull information hur applikationen kunde förbättras.

Ett exempel på resultatet av en användartest var att en användare som fortsatt med en tidigare "oavslutad runda" och sedan avslutat rundan, hamnade tillbaka i listan för oavslutade rundor vilket då var helt tom eftersom rundan flyttats till listan för "avslutade rundor". Detta skapade förvirring hos användaren. Tack vare denna feedback ändrade vi så att användaren istället skickas tillbaka till huvudmenyn vilket även var det som användarna förväntade sig skulle hända.

Efter vår sista sprint, så godkände vår intressent produkten och ansåg att den uppfyllde kraven för MVP. Intressenten uttryckte att den i vissa fall var bättre och mer användarvänlig än kommersiella applikationer och att han planerar att använda den för sitt eget golfspel i framtiden. Det finns givetvis fler funktioner att implementera i applikationen men intressenten anser att den uppfyller alla krav för att vara användbar.

(B)

Gruppen använde inte formellt user acceptance tests (UAT). Problemet med detta var att projektgruppen riskerade att missa det som var viktigt för kunden under testandet av koden. Risken finns då att koden funkar men inte för det ändamålet som kunden eller användaren har. Enligt [1] hade arbetet kunnat använda sig av user acceptence tester mer genomförligt för att bättre testa om applikationen kommer att nå upp till dem krav som användarna kommer att ställa. Genom att tillåta den tänkta användaren att använda applikationen för det exakta ändamålet i korrekt miljö säkerställer gruppen att applikationen tillgodoser det faktiska behovet [1].

($A \rightarrow B$)

I ett framtida arbete hade projektgruppen kunnat använda sig av UAT:er för att säkerställa att acceptanstesterna som gruppen använde stämmer överens med kundens ändamål. Detta hade krävt mer kommunikation med kunden samtidigt som det inte är säkert att det hade behövts vid utvecklandet av en sådan enkel applikation som den projektgruppen utvecklade. Det finns en risk att det hade krävt mer tid och energi samtidigt som det kanske inte hade skapat mer värde.

1.7 KPI

(A)

Tidigt i kursen valde gruppmedlemmen Hussein att hoppa av kursen eftersom han upplevde att han läste för många kurser samtidigt (3 stycken) och att han inte hann med sitt arbete i vårt projekt. Gruppen har kommunicerat med Hussein både innan och efter han annonserat sitt avhopp, för att se om det gick att underlätta hans situation på något sätt men han kände ändå att kursen skulle bli för tung samt att scheman mellan de olika kurserna kolliderade. Han ville därmed helt enkelt inte fortsätta. Som följd av detta beslutade gruppen att införa fem nya KPI:er i syfte att följa upp hur gruppmedlemmar upplever att samarbetet fungerar, hur de mår och hur projektet fungerar.

De fem KPI:erna, *kommunikation, arbetsbörda, stressnivå, arbetsprocess* och *tillfört värde* utvärderades därefter vid varje sprint review utifrån nedanstående skala:

- 1 = Mycket missnöjd
- ...
- 5 = Mycket nöjd

För vecka 18, 19, 20 och 21 presenteras genomsnittet av gruppmedlemmarnas utvärdering i tabell 1 nedan:

Tabell 1: Sammanställning av genomsnittliga KPI-värden för vecka 18, 19, 20 och 21. *(Högt värde är bra och lågt värde är dåligt)*

KPI	**Vecka 18**	**Vecka 19**	**Vecka 20**	**Vecka 21**
Kommunikation	3,4	3,6	2,6	4,6
Arbetsbörda	3,2	3,6	4	4
Stressnivå	2,8	3,2	3,6	4,4
Arbetsprocess	3	3,4	2,6	4
Tillfört värde	2,8	3,4	2,4	5
Genomsnitt	**3,04**	**3,44**	**3,04**	**4,4**

Utifrån tabell 1 ovan kan det observeras en förbättring från vecka 18 till 19, men vidare kan det konstateras att den genomsnittliga bedömningen sjönk till *OK* igen. *Kommunikationen* fungerade sämre under vecka 20 jämfört med tidigare veckor. Då gruppen diskuterade detta framgick att missförstånd i kombination med skilda fokus, i samband med deadline för tre gruppmedlemmars kandidatarbeten, låg till grund för detta. Missförstånd som uppstod vecka 20 kan förklara den lägre utvärderingen av *arbetsprocessen* och det skilda fokuset inom gruppen till följd av kandidatarbetet resulterade i ett lägre *tillfört värde* totalt sett. Gällande *arbetsbörda* skedde det en genomsnittlig förbättring från vecka 18 till vecka 20 och för *stressnivå* skedde förbättring till och med vecka 21. Respektive gruppmedlems genomsnittliga bedömning av dessa KPI:er framgår av tabell 2 nedan:

Tabell 2: Genomsnittlig utvärdering av KPI för respektive person för vecka 18, 19, 20 och 21. *(Högt värde är bra och lågt värde är dåligt)*

KPI	**Alfred**	**Joar**	**Markus**	**Philip**	**Raoul**
Kommunikation	3,25	4,25	3,5	3,5	3,25
Arbetsbörda	4	4	3,75	3,75	3
Stressnivå	4	2,75	3,25	3,5	4
Arbetsprocess	3,5	3,5	3,25	3	3
Tillfört värde	3,25	3,25	4	3	3,5
Genomsnitt	**3,6**	**3,55**	**3,55**	**3,35**	**3,35**

Det framgår av tabell 2 att gruppmedlemmarna i genomsnitt upplever att samarbetet fungerat något bättre än *OK*. Det bör dock beaktas att värdena i tabell 2 är just genomsnitt av hur respektive gruppmedlem utvärderat respektive KPI för de fyra veckorna. Vad som ej framgår är den spridning mellan olika gruppmedlemmars utvärdering av KPI:er som observerades under de tidigare veckorna där vissa gruppmedlemmar som haft fokus på kandidatarbete upplevt stress för att de inte bidragit som de önskar. Samtidigt har de som bibehållit ett större fokus på denna kurs har haft svårt att arbeta på grund av hinder som uppstått på grund av att saker inte blir gjorda i tid. Skillnaden mellan högsta respektive lägsta utvärdering av respektive KPI för varje vecka framgår av tabell 3 nedan:

Tabell 3: Skillnaden mellan högsta och lägsta utvärdering av respektive KPI för vecka 18, 19, 20 och 21.

KPI	**Vecka 18**	**Vecka 19**	**Vecka 20**	**Vecka 21**
Kommunikation	1	2	2	1
Arbetsbörda	1	1	3	0
Stressnivå	4	3	3	1
Arbetsprocess	2	2	3	0
Tillfört värde	2	4	3	0
Genomsnittlig spridning	**2**	**2,4**	**2,8**	**0,4**

Det kan utifrån tabell 3 ovan konstateras att spridningen för utvärderingarna ökade kontinuerligt från vecka 18 till 20. Detta reflekterar de ovan nämnda problemen kring skilda fokus och missförstånd som diskuterades av gruppen vid utvärdering av vecka 20. I samband med det beslutades att de gruppmedlemmar som föregående vecka haft ett annat fokus, istället ska lägga mer kraft och vikt vid kommunikation och arbetsprocessen i detta projekt. Vidare kom gruppen överens om att ventilera problem som uppstår i arbetsprocessen i ett så tidigt skede som möjligt. Troligtvis bidrog dessa

överenskommelser till att både den genomsnittliga utvärderingen av KPI:erna, och den genomsnittliga spridningen av utvärderingar från, vecka 21 var högre respektive lägre till skillnad från tidigare veckor. Den minskade spridningen i kombination med ökade grad av nöjdhet talar för en positiv utveckling i samarbetet där skillnaden mellan gruppmedlemmars upplevelser minskat samtidigt som den sammantagna graden av nöjdhet ökat.

(B)

Även om diskussion av utvärderade KPI:er vid sprint review endast genomfördes vid de två sista utvärderingarna var det något som samtliga gruppmedlemmar upplevde som positivt för samarbetet. Detta då det var en väg att lyfta problem och öka förutsättningarna för utformning av förbättrade arbetssätt. Möjligen hade det kunnat undvikas att en av gruppmedlemmarna valde att lämna projektet, ifall dessa KPI:er kommit på plats från början. Detta då det från början hade varit möjligt att anpassa arbetssätten så de passade alla gruppmedlemmar till större utsträckning.

Som ovan nämnt upplevde gruppmedlemmarna diskussion kring utvärderade KPI:er som framgångsrikt för samarbetet. Dock hade gruppen ingen direkt rutin för hur utvärderade KPI:er skulle diskuteras. De första riktlinjerna som verkligen förbättrade samarbetet i gruppen definierades först vid utvärdering av vecka 20, vilket reflekteras av den ökade spridningen från vecka 18–20 i tabell 3. Troligtvis hade varit fördelaktigt om likande struktur som användes vid dagliga standups, applicerades även vid samtal kring utvärderade KPI:er för att kunna lyfta skillnader och problem i samarbetet och projektet utifrån respektive utvärderad KPI.

$(A \rightarrow B)$

Under detta projekt har samtliga gruppmedlemmar, på grund av de rådande omständigheterna, aldrig träffat varandra. Att lyfta problem och ifrågasätta arbetssätt är något vi tror varit svårare på grund av dessa faktorer. Genom att ses på riktigt, hade det möjligen skapats bättre förutsättningar för att lyfta viktiga diskussioner som under detta projekt först kom i ett senare skede.

Vidare lärdom är att diskussion kring utvärderade KPI:er är essentiellt för att utvärdering av dem ska kunna skapa värde. Därtill bör dessa diskussioner genomföras med struktur så att alla får sin röst hörd.

Det är dock inte säkert att de KPI:er som upprättats är de som bäst hade kunnat utvärdera hur samarbetet fungerar och hur gruppmedlemmarna mår. Likväl som gruppen borde fokuserat på att diskutera utvärderade KPI:er, borde fokus även lagts på att diskutera vilka KPI:er som används. Möjligen hade dialog med andra grupper varit fördelaktigt i syfte att få inspiration kring vilka KPI:er som vi hade kunnat testa och diskutera.

2 Socialt kontrakt och arbetsinsats

I detta avsnitt följer resonemang kring hur gruppen förhållit sig till det sociala kontraktet samt hur kontraktet formats under projektets gång. Vidare diskuteras även arbetsinsats från gruppen samt de verktyg som användes för att mäta detta.

2.1 Socialt kontrakt

(A)

Allt eftersom projektet fortlöpte diskuterades nya förhållningsregler gruppmedlemmarna emellan. Däremot skrevs inte nya överenskommelser kring arbetssätt och rutiner ned i det sociala kontraktet. Snarare förblev överenskommelser muntliga som exempelvis riktlinjerna för kommunikation vilka togs fram vid utvärderingen av vecka 20 då det, som nämnt i kapitel 1.7 KPI, rådde friktion i kommunikationen på grund av skilt fokus från vissa gruppmedlemmar.

(B)

Under andra halvan av projektet upprättades dokument innehållande riktlinjer för utvecklingsprocessen med avseende på designbeslut, kodkvalitet, grafisk design och UI samt testning av kod. Att dessa fanns nedskrivna på papper gav positiva effekter på utvecklingsprocessen. På liknande sätt kan det tänkas att det hade kunnat skapas bättre förutsättningar för ett välfungerande samarbete i gruppen om muntliga överenskommelser kring samarbetet hade skrivits ned i det sociala kontraktet. Att skriva ner muntliga förhållningsregler kring exempelvis att komma i tid och att sitta i ett tyst rum under möten för att bidra till så goda förutsättningar för fokuserade möten som möjligt, skulle kunna leda till förbättringar.

($A \rightarrow B$)

För att kunna samtala kring nya förhållningssätt i ett tidigt skede krävs att frågan lyfts till ytan. Att diskutera samarbetet i gruppen borde därmed varit en punkt i dagordningen för varje sprint review. På samma vis borde även utvärdering av nedskrivna riktlinjer behandlas under sprint review för att bibehålla de arbetssätt som fungerar och ta bort de riktlinjer som inte bidrar till en positiv utveckling för gruppen.

2.2 Arbetsinsats

(A)

Samtliga gruppmedlemmar har deltagit i nästan alla dagliga standups, vilket har varit värdefullt. Våra standups innehöll korta genomgångar av vad som gjorts och vad som ska göras.

Att helgarbete har varit nödvändigt vissa gånger har nästan i samtliga fall berott på oförutsedda komplikationer som uppstått utanför kursen vilket har lett till mindre tillgänglig tid än planerat. För att då ta ikapp missad arbetstid har helgerna använts istället. Stressnivån har även ökat på grund av dessa oförutsedda komplikationer i kombination med att man fortfarande vill leverera det man utlovat till gruppen. I största möjliga mån har helgarbete försökt undvikas och i de fall det har skett låg fokuset på de allra viktigaste funktionerna i respektive sprint.

Git-inspector användes for mäta antalet mängden kod som gruppmedlemmarna bidragit med. Gruppen var dock medveten om att antalet mängden kod inte var representativt för hur mycket arbete som respektive deltagare faktiskt har bidragit med. Hänsyn togs till att varje person har olika velocity och att varje person endast skulle ta på sig så mycket arbete som denne klarar av inom en arbetsvecka. Därmed var det upp till respektive gruppmedlem att justera sin egen velocity inför varje sprint.

(B)

Sammantaget genom projektet har samtliga gruppmedlemmar bidragit med god arbetsinsats. Den har dock inte varit jämn från alla gruppmedlemmar genom projektet då de tre gruppmedlemmar från I-programmet lade mycket tid på deras kandidatarbeten. Möjligen hade arbetsinsatsen kunnat bli mer jämn över projektet ifall parprogrammering hade använts strukturerat. Genom att med andra gruppmedlemmar komma överens om *när* de ska arbeta tillsammans och *vad* de ska göra, hade det möjligen en jämnare arbetsinsats kunnat erhållas. Detta då det skulle blivit både roligare att arbeta samt då troligtvis färre uppgifter hade skjutits framåt till slutet av varje sprint.

Gruppen bokförde inte hur många timmar de lade på kursen, utan snarare har det förts en dialog inom gruppen vid genomförd sprint kring huruvida mängd nedlagd tid relaterar till respektive veckas åstadkomna arbete.

$(A \rightarrow B)$

För att jämna ut arbetsinsatsen hade det förmodligen varit fördelaktigt med tydligare planering av arbetet genom en mer strukturerad parprogrammering. Vidare hade det kunnat vara intressant att bokföra nedlagda timmar för att kunna utvärdera hur nedlagd tid relaterar till slutförda uppgifter, över längre tidsperioder än vecka för vecka. Detta då det hade varit möjligt att jämföra arbetsinsats och velocity över längre tidsintervall. Kanske hade även en tidslogg kunnat ge vägledning kring hur arbetsinsatsen skulle kunna koordineras mellan både gruppmedlemmar och veckor i syfte att erhålla ökad velocity med samma aggregerade arbetsinsats.

3 Designbeslut och projektstruktur

3.1 Hur våra beslut stödjer kundvärde

(*A*)

I vår första teamreflektion, så beskrivs det att vi haft en tydlig dialog med intressenten i syfte att i tidigt skede kunna lägga en grunddesign som kommer lämpa sig för användaren. Intressenten gav då förslag på funktioner som hade varit önskvärda i vår applikation. Dock hade gruppen en osäkerhet och tog i beaktning att en intressent kanske inte alltid lyckas specificera de funktioner hen egentligen vill ha. Därför var den initiala strategin att ta fram och implementera funktioner utöver det som intressenten redan specificerat, i syfte att underlätta för vederbörande att ge åsikter om vad användaren kan tänkas vilja ha. Som det även står i den första teamreflektionen, så kom gruppen till insikt att denna typ av tillvägagångsätt inte nödvändigtvis är önskvärd eftersom intressenten kan tvingas ta ställning till funktioner som hen kanske inte hade reflekterat över annars. Det kan också leda till att intressenten potentiellt skulle börja ge förslag på funktioner utefter den linje som lagts fram av utvecklarna, istället för att utgå ifrån sin egen erfarenhet. Planen var därefter att enbart fokusera på de förslag som intressenten haft.

Intressenten har haft en idé om vad den vill att applikationen skall åstadkomma. Som exempelvis att digitalisera ett scorekort, automatisk uträkning av poäng och så vidare. Intressenten har dock inte haft varken tekniska kunskaper eller insikt i gruppens arbetsresurser. Detta har inneburit att gruppen varit tvungen att prioritera och fokusera på de viktigaste funktionerna som varit genomförbara utifrån våra resurser, för att kunna åstadkomma en funktionell MVP som intressenten faktiskt kan ta med sig ut på golfbanan och ha nytta utav.

De funktioner som prioriterats har varit:

- Inmatning av spelare (initialer, handicap och tee)
- Inmatning av slag för upp till 4 spelare (enligt golfregler)
- Uträkning av spelarnas poäng baserat på handicap, på varje hål och totalt
- Permanent lagring så att spelare kan återuppta runder
- Historik (Lagring av föregående rundor)
- Inladdning av banor från fjärrserver så att banor kan uppdateras i realtid utan att användaren behöver göra något
- Ett användarvänligt och buggfritt gränssnitt i samtliga vyer

Funktioner som har prioriterats lägre och därmed inte tagits med i applikationen är:

- GPS-rangefinder där användaren kan mäta avstånd mha. kartan
- Statistik över spelarhistorik med grafer över prestation på varje bana, för varje spelare osv.
- Lagring av olika spelare
- Visning av fullständiga spelarnamn
- Delning av resultat
- Synkronisering mellan enheter i realtid (multiplayer)

Ovanstående funktioner har prioriterats lägre eftersom vi tillsammans med intressenten har kommit fram till att då funktionerna skulle vara väldigt användbara, så är de inte absolut nödvändiga för att applikationen skall fungera och därmed uppfylla en MVP. Prioriteringen har därför varit att fokusera på ett färre antal funktioner men att istället göra de valda funktionerna bra och med hög fokus på användarvänlighet, så att intressenten faktiskt kan använda applikationen och ha nytta av den när vi lämnat den ifrån oss i slutet av kursen. Exempelvis hade vår applikation ett fält för inmatning av hela spelarnamnet med planer för att namnet sedan skulle kunna användas i exempelvis historiska ronder. Intressenten upplevde dock att det var jobbigt att skriva in hela namnet varje gång och eftersom namnet ändå inte visades någonstans i applikationen, så togs beslutet att dölja inmatningen för spelarnamn i UI:t. Funktionaliteten finns dock kvar i koden och är tillgänglig för vidare utbyggnad i framtiden. Detta är eventuellt även någonting som skulle bli mer användbart när funktionaliteten att spara och återanvända spelare existerar.

Något som vi funderat på en hel del är hur man skall lägga till banor i applikationen, då användaren kan tänkas vilja spela på banor i hela Sverige. Idealt skulle man vilja ha tillgång till Svenska golfförbundets API, men det kostar ca 30 000 kr per år och är därmed inte möjlig att använda i denna kursen. Därför hade vi ursprungligen en idé som innebar att användaren själv kan mata in nya banor och därmed bli oberoende av utvecklarna. Men med vidare insikt innebar detta att användaren skulle behöva mata in ungefär 64 olika värden per bana, vilket både vi och intressenten tyckte blev väldigt jobbigt för slutanvändaren. Vi beslutade därför att skapa ett eget REST-API (fjärrserver) på banor med korrekta värden, som enkelt kan uppdateras av antingen en administratör eller av utvecklarna själva. Inledningsvis föreslog intressenten att enbart ta med banor i Göteborgsregionen, då det är där hen bor och spelar golf som mest. Vidare finns det möjlighet att successivt implementera samtliga banor i landet genom att lägga till banor i vår fjärrdatabas, och det kommer direkt att synas automatiskt i varje användares applikation utan behov av uppdateringar.

(B)

En del funktionalitet som vore användbar har upptäckts med tiden och var inte definierad vid start. Dock är en del av den funktionalitet som upptäckts väldigt användbar, men på grund av att man inte utvecklat applikationen med detta i åtanke, blir viss funktionalitet ett väldigt stort projekt att genomföra. Exempelvis är lagring och återanvändning av olika spelare något som teoretiskt borde vara väldigt enkelt, men blir svårt på grund av bristfällig framförhållning. Dels då det inte finns utrymme i vårt UI för att visa det fullständiga namnet någonstans, men också i den underliggande arkitekturen. Detta innebär att man behöver ändra hur tabeller, slagräknare, listor och så vidare ser ut rent grafiskt, för att kunna få plats med den typen av information. Det är även någonting som förändrar hur den lokala databasen ser ut.

$(A \rightarrow B)$

Man skulle idealt vilja ha större åtanke i hur en applikation kan komma att utvecklas i framtiden och bygga koden på ett sådant vis att den är lätt att bygga ut. Att planera och utveckla koden på ett sådant vis skulle vara gynnsamt för utveckling i längden. Dock skulle det kräva mer planering och tidsåtgång för att åstadkomma detta men det skulle troligtvis leda till mer effektiv utveckling i längden. Med den tidsperiod vi haft skulle vi troligtvis ha åstadkommit mindre i och med att ytterligare resurser då skulle lagts på planering istället för effektiv utveckling. Så det handlar i slutändan om man vill ha mer funktionalitet på kort tid, eller mer funktionalitet på sikt. Man skulle därför vilja göra en avvägning och planera mer för potentiell vidareutveckling av applikationen, men inte för mycket så att produktiviteten markant försämras på kort sikt.

3.2 Teknisk dokumentation (hur vi använder och uppdaterar)

(A)

Dokumentationen har varit en viktig del i projektet då vissa personer har varit väldigt involverade i vissa delar av applikationen och för att beskriva funktionalitet för övriga medlemmar i gruppen har dokumentation och diagram använts. Ett exempel på detta är flödesdiagrammet som visar hur helheten i applikationen skall se ut, hur användaren kan navigera mellan olika sidor och så vidare. Detta är viktigt för att kunna få ihop en helhet i slutprodukten, och låta utvecklarna koppla ihop sina delar så att det passar och blir genomtänkt.

Under projektets gång har dokumentation i mån av behov utvecklats. Dokumentationen kan hittas i vårt Github repository, under mappen "documents and deliverables".

Vår dokumentation

- UML-klassdiagram (på hela modellen)
- ER-diagram (för den lokala databasen)
- Designbeslut för kod och arkitektur
- Kodkvalitet och konventioner (innehåller även definition of done)
- Testningsprocedurer innan varje merge med master
- Grafisk design och UX beslut
- JavaDocs med beskrivning för hela modellen
- Flödesdiagram för navigering i applikationen

(B)

Vi har dokumenterat i princip allting fram till sista sprinten vilket har varit användbart för oss. Men man skulle även vilja dokumentera det sista som gjorts ifall projektet någongång skulle återupptas. Ett problem har varit att vi inte haft någon formell rutin på hur dokumentation skall genomföras. Istället har ett fåtal personer fått uppgiften (i form av scrumkort) att dokumentera vissa saker vid behov. Man skulle istället vilja att dokumentationen alltid var aktuell och att det fanns mer definierade rutiner på hur den skall skapas och uppdateras.

($A \rightarrow B$)

Dokumentation kan tyckas vara tråkigt att lägga tid på och bidrar inte direkt till kundvärde. Men det bidrar indirekt till kundvärde genom att bidra till högre kvalitet av slutprodukten då det hjälper utvecklarna att kommunicera en gemensam vision för vad som komma skall, och öka produktiviteten då det minskar inlärningskurvan av andras kod. Därför vore det bättre om varje person har som ansvar att dokumentera sin egen utveckling löpande under varje sprint, vilket också gör att det i slutändan inte blir en jättestor och jobbig uppgift.

3.3 Kodkvalitet och kodstandarder

(A)

Gruppen beslutade tidigt i projektet att använda sig av Googles Java-konventioner för att upprätthålla en enhetlig och lättläst kodbas. Android bygger även i grund på arkitekturen MVC där vyn representeras av XML filer och kontrollern representeras av en Activity. Utöver detta har vi valt att använda arkitekturen MVVN (Model–view–viewmodel) då den ingår i en utvecklarstacken Jetpack som tagits fram specifikt för Android, av Google. Vi har därmed valt att använda så mycket av Jetpack som möjligt vilket inkluderar databasbiblioteket Room som är en ORM för Android. Vi använder även Retrofit2 som också ingår i Jetpack-stacken, för att kunna skapa en REST-klient som kommunicerar med och hämtar golfbanor från vår egenbyggda REST-API.

(B)

Generellt sätt har våra val av kodstandarder och arkitekturer varit självklara och fungerat bra. Däremot skulle fler arkitekturer kunna användas. Det är dock svårt att införa då gruppen har varierande bakgrunder och kunskaper om olika designmönster. Vidare nämns förslag på förbättring i sektion 3.3.1, med hjälp av peer-review.

($A \rightarrow B$)

Man skulle kunna åsidosätta extra tid åt att utbilda gruppen inom relevanta designmönster som applicerats och som kan komma att nyttjas. Och på så vis låta samtliga i gruppen ta ikapp den kunskap som behövs för att på ett enhetligt sätt nyttja designmönster. Peer-review är också någonting som kan tillföra förbättring här och diskuteras vidare i sektion 3.3.1.

3.3.1 Peer-reviews

(A)

Gruppen valde tidigt i projektet att inte använda sig av peer-reviews för att spara tid och effektivisera arbetet. Exempelvis kan det vara så att en person upptäcker något litet som behöver ändras, och om då två andra personer måste godkänna ändringen innan den kan bli mergad så blir det väldigt tidskrävande för samtliga medlemmar. Gruppen har haft väldigt många commits med mindre ändringar och att inte använda oss av peer-review har varit väldigt tidsbesparande för samtliga i gruppen.

Istället för peer reviews, så har vi använt oss dels av unit-tester för 100% av modellen inklusive den lokala databasen, men också en checklista med funktioner som varje person måste testa manuellt i applikationen innan den får merga med master. Detta är någonting som har fungerat väldigt bra och gått fort att utföra (ungefär 1-2 minuter tidsåtgång för den som vill merga med master). Med hjälp av denna metodik har vi inte haft en enda commit som brutit funktionalitet i mastern under hela projektets gång.

I det stora hela så har vårt tillvägagångsätt fungerat bra. Men det finns avvikelser där valda konventioner och arkitekturer inte upprätthållits. Det finns även risk att en utvecklare glömmer av att köra igenom testerna. Därför föreslås det i framtiden att använda sig av kontinuerlig integrering senare i sektion 4.3.

(B)

Valet att inte använda peer-reviews men att istället använda unit-tester och en checklista har som sagt fungerat bra och bidragit till en hög produktivitet. Dock så har avsaknaden av peer-reviews låtit brister i kodkvaliten passera eftersom våra tester enbart kollar om funktionaliteten i modellen är korrekt. Exempel på sådana brister är att Googles kodstil och relevanta designmönster inte alltid följts på ett bra vis.

I en ideal värld vill man ha en perfekt arkitektur, där samtliga medlemmar följer kodkonventioner till punkt och pricka. Det är nog svårt att upprätthålla oavsett hur man gör då alla människor är olika och ingen är perfekt, men peer-review skulle troligtvis bidra en förbättrad nivå.

$(A \rightarrow B)$

Man kan tänka sig att användning av peer-review skulle låta gruppmedlemmar påpeka om ett designmönster inte använts, eller om det använts felaktigt i ett tidigt skede, och därmed förhindra problem längre fram. Speciellt om gruppen hade haft fler medlemmar eller om det funnits fler teams som jobbat på projektet. Det är svårt att avväga om det i vårt fall hade gjort en stor skillnad eller om det hade minskat eller ökat vår produktivitet. En korrekt arkitektur kan nämligen bidra till ökad produktivitet i längden. Ett konkret exempel på detta är att Scorecard-tabellen har byggts i ren Java vilket gör det tidskrävande att redigera designen jämfört med om den hade skrivits i XML. Men samtidigt kände utvecklaren sig mer bekväm med att lösa uppgiften på det viset som denne gjort. Så det kanske hade tagit längre tid initialt att göra uppgiften annorlunda, men minskat tidsåtgången att redigera tabellen i ett senare skede. Man kan tänka sig här att en peer-review i tidigt stadie kunnat påtvinga en annan lösning, men kanske hade lösningen blivit likadan ändå i slutändan. I vårt projekt skulle det nog ha varit optimalt att ha peer-review på större commits som exempelvis nya funktioner, men mindre optimalt på små korrigeringar som exempelvis buggfixar. Med detta i åtanke hade det varit av värde att ha en utvärderingsperiod där man nyttjar peer reviews för att undersöka ifall det tillför värde för utvecklingsprocessen. Nästa steg skulle därför vara att ha en utvärderingsperiod där man testar att använda peer-review enbart på större ändringar i applikationen, för att undersöka ifall det tillför värde till utvecklingsprocessen, samt ifall peer-review skulle behöva behövas på samtliga typer av commits.

4 Tillämpning av Scrum

4.1 Roller i teamet

(A)

Under kursens gång har vi haft statiska roller i form av kundkontakt och scrum master. Detta innebär att vi haft en fast scrum master under hela projektets gång, mycket likt som det skulle vara på en arbetsplats. Kundkontakten har naturligt fallit på de personer i gruppen som haft relation med slutanvändare redan innan projektet. Vi har inte haft möjlighet att hitta någon som ville ställa upp som PO då det kräver mycket jobb från PO:ns sida. Därför beslutades att samtliga medlemmar i gruppen gemensamt skulle ta sig an rollen som PO, med demokratiska beslut i hänsyn till intressentens behov. I bästa möjliga mån har det utförts backlog refinement utifrån vad som ger kundvärde i första hand, och inte det som är lågt hängande frukt från ett utvecklarperspektiv.

(B)

Å andra sidan kan man tänkas hamna i team med roterande scrum master. Fördelarna med detta är att alla har möjlighet att samlat få bättre insikt, ägandeskap och engagemang som annars kan ses som ”någon annans ansvar” istället för ”vårt ansvar”. Vanligtvis i projekt har man en faktiskt dedikerad PO, vilket skulle till skillnad från den kollektiva tillvägagångsätt vi hade ge en backlog som bättre representerar stakeholders behov.

($A \rightarrow B$)

Man skulle kunnat testa främst två saker; roterande scrum master och en dedikerad PO.

Det första, en roterande scrum master, skulle kunna ge bättre möjlighet för alla gruppdeltagande att få insikt i vad det innebär att vara scrum master och dra lärdom ifrån de utmaningar det för med sig. Denna erfarenhet skulle kunna skapa en starkare känsla av ett kollektivt ansvar samt öka empati och psykologisk säkerhet inom teamet. Detta är två parametrar för högpresterande team som identifierats under Googles interna undersökning Google Rework [2].

Det andra, en dedikerad PO, hade kunnat ge mer konkreta förbättringar i form av en mycket välprioriterad backlog som reflekterar kundbehov. Och det skulle dessutom kunna leda till konstruktiva konflikter mellan utvecklare och PO. I det fallet där man delar upp ansvaret av PO, bland just utvecklare, faller denna konflikt bort. Istället kan utvecklarna själva välja det som känns lätt att implementera, eller kanske vad de själva tycker ger mest kundvärde, vilket inte är det bästa för kunden.

4.2 Sprint review, planning, retrospective och deras kundvärdeskapande

(A)

Vi började med vår första sprint redan andra veckan i kursen, med fullständig sprint planering och sprint reviews kontinuerligt efter varje sprint. Vi har sett till att alla i projektgruppen varit närvarande och tillsammans utvärderat utfört arbete samt deltagit i planering av kommande arbete.

Under sprint reviews tidigt i projektet låg stort fokus på att se vad som är klart och hur det förhåller sig till kommande sprint. Dessa tidiga sprint reviews misslyckades dock att representera kundvärde, i stor utsträckning på grund av det arbete som utförts inte hade uppenbar koppling till uppenbart kundvärde. Svårigheten att mäta kundvärde med ett icke-fungerande gränssnitt är det svårt att få bra feedback när endast vissa delar av systemet fungerar, eller när saker som inte syns (modell)

har utvecklats. Senare sprint reviews, med ett grundläggande gränssnitt på plats, skulle komma att skifta fokus till att fokusera på förbättringar för användarupplevelsen baserat på intervjuer med intressenten.

För att kunna påbörja ett projekt krävdes en grund att utveckla utifrån, både tekniskt men även kunskapsmässigt. Utöver detta kunskapsgap ansåg vi att det inte skulle vara enkelt att få grundlig feedback på ett praktiskt taget tomt gränssnitt. Detta innebar inte att vi inte hade kundvärde i tankarna när vi utvecklade, utan snarare att vi ville skapa en prototyp för att få feedback på för att kunna driva framtida beslut om implementation, design och UX.

I våra tidiga sprints använde vi ett tidsestimat som poängsättning av uppgifter där en timma arbete motsvarade 1 poäng. Denna poängsättning sågs dock mer som riktlinjer och inte som absoluta planer. Det visade sig dock vara mycket svårt att uppskatta tidsåtgången för uppgifterna, men det tog inte särskilt lång tid att nyttja grundstenarna av den agila processen, även ifall det fanns ”vanor” från vattenfallsliknande arbetssätt, som skulle behöva ersättas med nya beteende.

I senare delen av projektet infördes sprint retrospective, som inte omedelbart gav kundvärde, men som var ett sätt att försöka återkoppla till hur teamet upplevde arbetet. Genom denna introduktion av mätbar återkoppling av sociala KPIer, kunde utvecklingsprocessen bättre modifieras för att kunna leverera mer kundvärde. Till en början kändes dessa KPI-mätningar inte lika informativa, eller hjälpsamma, som de gjorde i slutet. Med hjälp av retrospective hade gruppen chans att nyttja återkopplingen för att kunna agera mot förändring. I slutet av projektet var det mycket hjälpsamt och gav möjlighet att applicera förbättringar på vårt arbetssätt. Exempelvis, då kommunikation upplevdes bristfällig beslutades det att man skulle informera om ens tillgänglighet under kommande dagen vid standup. Detta resulterade i att medlemmar mer aktivt informerade varandra om deras tillgänglighet, men kanske inte till den grad som önskades och skulle med vidare återkoppling kunnat förbättrats ytterligare. Även detta kan återkopplas till Google Rework [2], att genom mätningar och återkopplingsprocesser försöka fostra en miljö som bidrar till ett produktivt team.

(B)

Något som saknats, som skulle kunna förbättra sprint reviews, är att man har en extern PO som har ”fingret på pulsen” när det gäller kundvärde och relaterade stories.

Genom att tidigt undersöka hur vi kan etablera standarder kring review, retro och planning skulle det kunna underlätta arbetet. Eftersom det inte var helt tydligt från början vad som skulle göras vid varje möte tog det tid att skapa rutiner som närmat sig ett korrekt utförande av scrum. En samlad genomgång av olika scrum möten kan genomföras för att försäkra att alla i projektet är medvetna om vad som fordras av dem vid varje möte. Särskilt viktiga är just review och retrospective, som med bättre förståelse kunnat öka förståelse och enhetlighet inom teamet.

$(A \rightarrow B)$

Genom att lyckas rekrytera en extern PO kan man också få en oberoende part som kan utvärdera stories och få mer precis återkoppling ifall det arbete som utförts uppfyller de krav som ställts. Detta skulle ha effekten att story prioritering och backlog refinement blir mycket mer representativt för intressentens behov. Än mer så om denna PO också har tid att lägga på att ha grundlig kundkontakt, med tydlig dokumentation som resultat av intervjuer. Detta hade kunnat skapa en mycket bra stöd för utvecklarna och skapat en starkare feedback-loop från slutkunden.

För att få mer effektiva möten, och skapa gemensamma förutsättningar, hade en standardisering av möten kunna utföras. Gemensam genomgång och etablering av standarder kan nyttjas för att få en effektivare arbetsprocess. Inte nog med att det hade kunnat vara närmre scrum standarder, men det ger möjlighet till teamet att utvärdera saker som uppfyller deras behov ifall det inte täcks av de rekommendationer som erhålls i litteratur om agila arbetsprocesser.

4.3 Agila metoder och dess påverkan på arbetsprocessen

(*A*)

Gruppen hade inledningsvis en väldigt begränsad erfarenhet av scrum vilket innebar att inlärningskurvan var hög. Vi började även använda scrum en vecka tidigare än planerat i kursens schema, vilket innebar att medan vi låg lite före planeringsmässigt, så hade vi ännu inte lärt oss alla väsentliga rutiner. Detta resulterade i att agila rutiner implementerades stegvis.

Tidigt implementerades grundläggande agila metoder; scrum möten (standup, planning, review), scrum board, job stories, iterativt arbete, en product backlog och i viss utsträckning agila roller. Det testades först att sätta upp ett projekt med Azure Devops som ett paket med mjukvarustöd till den agila processen, men ersattes med alternativa verktyg då Azure Devops hade kostnader associerat med produkterna vi ville nyttja. Främst av dessa är Trello som vi använt som vår digitala Scrum board [3]. Som tidigare nämnt ansattes det att ”PO rollen” skulle fördelas mellan gruppmedlemmar, det vill säga att alla utvecklare deltar i backlog refinement och prioritering av backlog.

Agila metoder som tillkom under kursens gång var: sprint retrospective och i viss utsträckning en kundorienterad approach. Sprint retrospective skulle komma att utföras ordentligt i en senare del av projektet. Detta kom att förbättra feedbackprocessen om hur vi förbättra utvecklingsprocessen. Att introducera retrospectives skulle komma att bli mycket hjälpsamt i att utvärdera förbättringsområden och försöka att förändra rutiner och arbetssätt. Det tog en till två retrospectives innan det gav märkvärt positiv inverkan på projektet genom att bidra med underlag till det iterativa förbättringsarbetet.

(*B*)

Agila metoder som inte nyttjas, men som kanske borde är: timeboxing, continuous integration (CI), automatiserad testning och parprogrammering.

Timeboxing hade kunnat ge mycket bättre återkoppling kring hur mycket tid som krävs för uppgifter att utföras. Det kan argumenteras för att detta skulle leda till mer planeringsoverhead, men det hade kunnat vara av värde att testa detta.

Med CI och automatiserade tester kan vi som utvecklare enklare säkerställa full funktionalitet och buggfrihet. CI:n kan skapa kort återkoppling och undvika större merge konflikter.

Parprogrammering hade kunnat nyttjas i större utsträckning för att bidra med kunskapsdelning och att minimera defekter i koden. En nackdel med parprogrammering är att det kan leda till långsammare utveckling, särskilt i början då majoriteten av utvecklarna inte var bekanta med Android-utveckling.

För att ha en bra återkoppling till velocity hade verktyg med stöd för burndown charts, såsom Azure DevOps [4], kunnat ge en god svit med verktyg. Azure DevOps har även verktyg för att sköta mycket av den annars ”manuella bokföring” som vi gjort. Att exempelvis ha bra story och task tracking, burndown charts, bug tracking och CI är flera features det har som skulle kunna stötta en agil utvecklingsprocess.

Mycket finns redan nämnt under (B) och $(A \rightarrow B)$ i sektion 4.2. Exempelvis hade en dedikerad PO kunnat delta i möten på ett sätt som skulle kunna skapa konstruktiva diskussioner kring vad som bör prioriteras utifrån intressentens perspektiv. Genom att ha ett användarnära perspektiv kan man få en mycket bättre chans att iterativt utveckla en produkt som ger kundvärde.

$(A \rightarrow B)$

Även enklare timeboxing skulle kunna testas genom att kort efter varje standup, ha en lös planering av vilka timmar som man kommer arbeta på projektet och hur fördelning av denna tid på uppgifter ser ut. Att planera samarbetstid är en svår administrativ uppgift då vi har haft 4 olika varierande scheman.

Det vi nyttjat är en manuell checklista, där både tester av kod för modell och databas skall köras samt att utföra vissa manuella tester av gränssnittet. Tester skulle istället för att manuellt köras kunnat nyttja CI med automatiserade tester. Vid senare undersökning visar det sig vara så att verktyg såsom Travis CI har stöd för android projekt, vilket skulle kunnat nyttjas tillsammans med de standarder för tester som satts. Beslutet att inte nyttja Azure DevOps [4] endast en kostnadsfråga i både monetära och arbetstimmar för att lära sig verktyget. En mindre utredning om vilka verktyg som skulle kunna nyttjas för att stödja en agil process hade kunnat bidra med verktyg som kunde stöttat och underlättat projektets overhead.

Gällande parprogrammering rekommenderar vi att testa parprogrammering, för att undersöka effekten på utvecklingstakt, defekter och buggar. Med utvärdering av denna parprogrammering kan vi enklare se ifall det är något man vill göra mer eller mindre av samt när man vill nyttja det.

Referenser

[1] S. T. Help, *What Is User Acceptance Testing (UAT): A Complete Guide*, 2020. Tillgänglig: https://www.softwaretestinghelp.com/what-is-user-acceptance-testing-uat/ (hämtad 2020-05-01).

[2] J. Rozovsky, *The five keys to a successful Google team*, 2015. Tillgänglig: https://rework.withgoogle.com/blog/five-keys-to-a-successful-google-team/ (hämtad 2020-05-28).

[3] *Simple Golf project Scrum board*. Tillgänglig: https://bit.ly/2ZTs8eG.

[4] *Azure DevOps*. Tillgänglig: https://azure.microsoft.com/en-us/services/devops/.