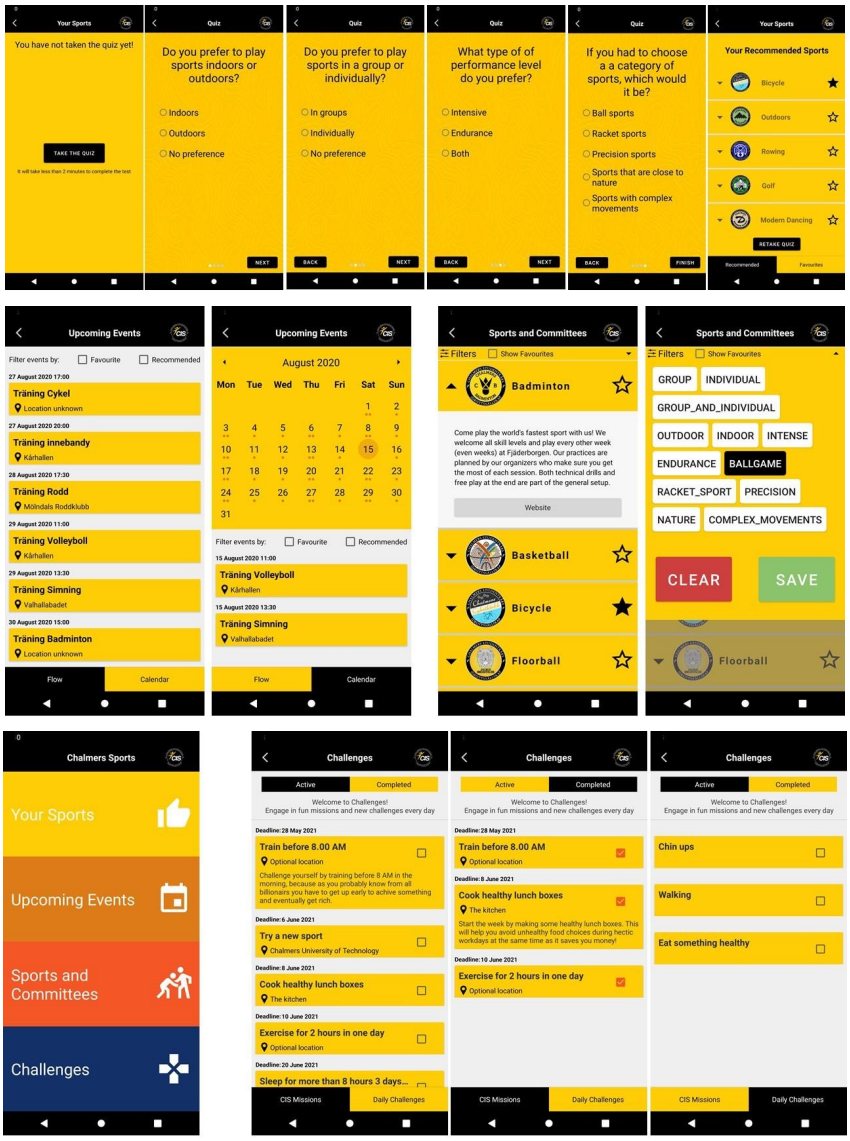
IBM System/360 Final Report

2021-05-31

Oscar Forsyth (IT), Oscar Johansson (IT), Victor Koch (I), Simon Larsson (IT), Simon Schuster (I), Hannes Svahn (IT), Mantas Trakanavicius (I)



Customer Value and Scope

Applikationens omfattning, prioritering av funktionalitet och värdeskapande

A: Under projektet valde vi att samarbeta med CIS som extern aktör, vilket innebär att de är de som efterfrågar värdet av applikationen. Viktigt att ha i åtanke är dock att det inte är CIS som är de faktiska slutanvändarna av applikationen, utan studenter på campus. I och med detta blev det ett dilemma mellan vad vi trodde att användarna ville ha, vad CIS ville ha och trodde att användarna ville ha, samt vad användarna faktiskt efterfrågade och värdesatte. Även om vi i gruppen till viss mån kan agera som slutanvändare påverkas vi av att vi också är de som måste implementera den efterfrågade funktionaliteten.

Under projektets gång gjordes tillfredsställande prioritering av user stories, vilket medförde att appens “key features” implementerades i ett tidigt skede av utvecklingen. Detta gjorde det möjligt att iterativt förfina dessa under kommande sprintar. Appens viktigaste funktion och den del som enligt CIS skapar mest värde användaren är quizet. Därmed önskade CIS ha detta implementerat allra först, men som nämnt i vår första gruppreflektion var quizet beroende av andra delar i appen vilket gjorde att vi som utvecklare fick förklara att en något annorlunda prioriteringsordning var optimal, vilket CIS då gick med på.

Sett till projektet som helhet innefattade det ett rimligt scope, där vi under de 6 sprintarna hann med att producera det innehåll som diskuterades med vår externa aktör i uppstarten av projektet.

B: I framtida projekt är det återigen viktigt att alltid ha fokus på den externa aktörens prioriteringar och samtidigt ha i åtanke vilka utvecklingsmöjligheter som finns, och hur olika user stories hänger ihop och måste prioriteras. Likaså är det i ett framtida projekt viktigt att ha ett stort fokus på den faktiska slutanvändaren och hela tiden fokusera på vad de värdesätter och efterfrågar. Genom att testa produkten på slutanvändaren har man möjlighet att täcka de behov som ej är självklara för en utomstående, och samtidigt få en bättre bild av vad som efterfrågas utan att påverkas av faktorer som att vi själva är de som måste implementera funktionaliteten.

A -> B: För att åstadkomma ett större fokus på slutanvändaren bör en utomstående part, i detta fall en student, ge feedback under projektets gång. På så sätt kan tankar från slutanvändarens perspektiv erhållas vilket kan göra att slutprodukten ger ett ännu större värde. Insamling av utomstående perspektiv ska helst göras flera gånger under projektets gång enligt oss, utvecklare, i gruppen. Dessutom skulle man kunna göra ett flowchart över user stories och dess prioriteringar inklusive dess beroenden för att tydligare kunna se

vilka stories som bygger på varandra. Det skulle underlätta för externa aktören att förstå följden av user stories, men samtidigt också ge utvecklare mer grund för att ta ett taktisk beslut som resulterar i högsta möjliga värde.

Det är även viktigt att hela målgruppen representeras i intervjuerna så att inte specifika behov från en viss del av målgruppen försummas. För detta kan även personas användas. Under projektets gång hade vi alltid slutanvändarna i baktanke, men det hade kunnat vara fördelaktigt att explicit ta fram personas för slutanvändarna för att få en ännu bättre bild av deras förväntningar.

KPI:er och deras betydelse för projektet

A: Under projektets gång har KPI:er använts flitigt. Vi använde tre KPI:er som representerar *produktivitet*, *välmående* och *kodkvalitet*. I början av projektet var vi ganska dåliga på att uppdatera dessa, men efter en gruppgemensam reflektion bestämde vi att uppdatera regelbundet för att kunna följa trender på ett bättre sätt. Kodkvalitet var den KPI som var mest otydlig. Gruppen lyckades inte bilda en gemensam bild för hur vi skulle bedöma kodkvaliteten, och därför blev inte denna KPI så representativ till slut. De resterande KPI:erna, produktivitet och välmående, fungerade bra och gav nyttig information till gruppen.

B: Inför framtida projekt skulle vi vilja använda KPI:erna mer kontinuerligt från start, dvs. fylla i dem varje dag i veckan för att hålla bästa möjliga koll på arbetstimmarna. Detta gjordes först efter de två inledande sprintarna i detta projekt.

Som nämnt under A var vår KPI för kodkvalitet otydlig och högst subjektiv. Denna borde ha varit tydligare och mer objektiv, så att den kunde vara direkt jämförbar mellan både sprintar och individer.

I ett framtida projekt vill vi kunna vara agila och omfördela arbetsbelastningen inom teamet mellan sprintar genom att använda sig av de KPI:er vi definierat. Exempelvis skulle en person som lagt ned många timmar under en sprint och känt sig väldigt stressad kunnat få en mindre belastande user story den kommande sprinten.

A -> B: En förbättring kopplad till omfördelning av arbetsbelastningen hade kunnat åstadkommas genom en kontinuerlig uppföljning, exempelvis dagligen, av KPI:erna. Således hade ett större värde kunnat erhållas från dem, likt det vi nämnde i vår första gruppreflektion. Dessutom borde ett nytt ramverk tas fram för att bedöma kodkvalitet. Det kan göras genom att konsultera med andra grupper och hur de har gjort för att bedöma kodkvaliteten. Man skulle även kunna ta kontakt med erfarna utvecklare och

konsultera med dem om hur de skulle tackla samma utmaning. Eventuellt kan det undersökas huruvida det finns ett sätt att göra det automatiskt, till exempel genom någon form av statistik från den IDE som använts.

Mål och lärdomar

A: Under uppstarten av projektet och det sociala kontraktet definierade vi ett antal mål. Bland dessa var ett att lära sig Android Studio och hur man bygger en Androidapplikation. Nu när projektet är över är detta något vi lyckats med, och har numera en grundläggande förståelse för dessa aspekter, och hur utvecklingen kan gå till. Likaså gäller för det agila arbetssättet och användningen av "repon", där delar av gruppen (framförallt I-studenterna) inte haft någon tidigare erfarenhet av detta.

B: Inför framtida projekt hade det varit fördelaktigt med ytterligare kunskap i Android Studio innan själva programmerandet startar. Detta för att reda ut vilka resurser (API:er och bibliotek) som finns tillgängliga och därmed kunna optimera koden.

A -> B: I framtida projekt tänker vi oss att det är bra att man konsulterar med några som har gjort mjukvaruutvecklingen i den miljö man tänker jobba i. I vårt fall kunde vi kontaktat äldre kursare som har programmerat en Androidapplikation och fråga om tips och råd för att undvika vanliga nybörjarmisstag. Sådana insikter är värdefulla för att snabbt kunna starta upp ett projekt.

User stories, uppgifter och acceptanskriterier

A: I början av projektet lyckades gruppen väldigt väl med att formulera user stories, uppgifter och acceptanskriterier samt att dela upp uppgifterna bland gruppmedlemmarna på ett sätt som undvek beroende, något som även tas upp i tidigare gruppreflektioner.

Acceptanstesterna genomfördes genom att funktionaliteten testades i emulatorn i Android Studio samt genom godkännande från CIS att det skapade värdet överensstämde med deras förväntningar. När det kommer till "effort estimation" lyckades vi generellt göra väldigt träffsäkra estimeringar, något som dels beror på vår relativt enkla men effektiva implementation av "Fibonacci Agile Estimation", men även gruppmedlemmars tidigare erfarenheter vid tidsestimeringen i KPI:erna.

B: I ett framtida projekt vill vi från start ha en tydlig struktur och tanke kring hur applikationen kommer att fungera och vara uppbyggd med exempelvis klasser och beroenden (slicing the cake).

Liknande gäller för de acceptanskriterier som definierats i respektive user story. Dessa ska vara tydliga och tillräckligt omfattande, vilket nämnts i tidigare reflektioner, så att de först blir uppfyllda när hela user storyn är klar och fungerande.

A -> B: I framtida projekt ser vi utvecklare ett stort värde i att lägga upp någon struktur för hur klasser och kod ska se ut. Detta skulle kunna åstadkommas med UML-diagram över applikationen där man tydligt kan se hur olika klasser hänger ihop. UML-diagram bortprioriterades dock denna gång på grund av kursens begränsade tid. Det skulle även underlätta att skapa uppgifter som inte är beroende av varandra. När det kommer till user stories så är målet att försöka göra dem självständiga så mycket som möjligt. Ett sätt att åstadkomma detta är genom en mer långsiktig planering med insyn i vilka user stories som bygger på varandra.

Nästa steg: Vad händer med applikationen framöver?

A: Gruppen hade initialt planer på att lansera appen på Google Play Store, men av flera skäl relaterade till bl.a. säkerhet och underhåll kommer appen istället distribueras som en APK-fil till CIS. Dessutom har gruppen gjort en “technical description” som är tänkt som ett stöd åt CIS då den förklarar all funktionalitet i appen och hur den underhålls. Vid projektets slut är det nu CIS som kommer att få ta över ansvaret för applikationen. Under detta projekt lades huvudfokus på att maximera värdet för CIS och slutanvändaren, vilket till viss del blev på bekostnad av kodkvaliteten eftersom tidsramarna för projektet var begränsade.

B: Vi vill i framtida apputvecklingsprojekt att det ska vara möjligt publicera applikationen på Google Play Store. Appen ska även vara lätt att underhålla och förstå sig på av någon som inte varit med under utvecklingsfasen. Dessutom kan det vara värt att tänka lite extra på kodkvaliteten eftersom denna är extremt viktig för att möjliggöra ett smidigt underhåll av appen.

A -> B: För att kunna ta med sig ovanstående aspekter, bör utvecklare ha dessa i åtanke redan i början av projektet. Ska applikation underhållas, speciellt av någon annan än utvecklarna, bör detta tas hänsyn till. Dessutom bör utvecklare då vara tydliga och noggranna med koden, så att de inte lämnar utrymme för missförstånd. Det är även viktigt att i framtiden ha i åtanke att applikationen kommer vara publik, och således vara säker på att API-keys och liknande är dolda eller på något sätt krypterade för att kunna garantera säkerheten vid publicering på Google Play Store.

Social Contract and Effort

Socialt kontrakt och samarbete

A: Ansvarsuppdelningen har fungerat bra i gruppen. Det har t.ex. varit väldigt smidigt att en person alltid haft ansvar för att starta och skicka länk till Zoom-mötena samt att en annan varit huvudansvarig för kontakten med CIS. Gruppen har haft regelbundna möten på fasta dagar vilket underlättat planering och skapat en kontinuitet i arbetet. Dessutom är vi nöjda med samtliga gruppmedlemmars förmåga att hålla tider.

Skillnader i kompetenser och erfarenheter mellan sektionerna är något vi har kunnat dra nytta av. Gruppmedlemmarna har kompletterat varandra på ett effektivt sätt, där studenterna från Informationsteknik oftast haft en djupare förståelse för de programmeringstekniska aspekterna medan studenterna från Industriell Ekonomi tagit lite extra ansvar för mötesstruktur, planering och kontakten med vår externa aktör.

Det faktum att flera gruppmedlemmar inte kände varandra sedan tidigare gjorde att vi kände ett ansvar att göra ett gott intryck vilket bidrog till ett positivt arbetsklimat. Gruppdynamiken hade dock kunnat förbättras ytterligare. Coronapandemin medförde att i princip alla möten hölls digitalt över Zoom, i de flesta fall utan videokamera.

B: I framtida utvecklingsprojekt skulle ett ännu mer proaktivt agerande kunna underlätta och effektivisera möten och koordinering av arbetsuppgifter. Dessutom skulle rollfördelningen kunna bli ännu tydligare bland gruppmedlemmarna, vilket skulle ge mer individuellt ansvar som förmodligen leder till högre arbetseffektivitet. Rollfördelningen borde i sådana fall baseras på vad man egentligen är bra på.

A -> B: För att åstadkomma en bättre gruppdynamik skulle man kunna ha teambuilding aktiviteter under uppstarten av projektet så att man lär känna varandra bättre. Detta skulle även minimera stelheten under möten. Ytterligare en aktivitet som kunnat förbättra gruppdynamiken är en check-in under varje möte som innefattar mer än att bara fråga gruppmedlemmarna hur de mår. För att öka fokus i gruppen och göra mötena mer personliga hade det möjligen varit positivt om videokameran varit på. I en tid utan pandemi hade fysiska möten varit att föredra, åtminstone en del av mötena.

Gruppens prestation

A: Under projektets gång har vi i gruppen upplevt att belastningen mellan medlemmarna har varit ganska jämn. Det kunde förekomma skillnader veckovis, eftersom alla inte

studerade samma program, exempelvis inlämningsuppgifter i andra kurser som man önskat lägga mer tid på. Trots veckovisa skillnader har belastningen jämnats ut i längden.

Dessutom försökte vi hålla en ganska flexibel och anpassningsbar miljö i gruppen beroende på studier. Vi såg inga problem i att en gruppmedlem tog på sig en enklare uppgift om denne hade en i övrigt hög arbetsbelastning. Vissa gruppmedlemmarna var redan i början av projektet tydliga med vilka veckor den totala arbetsbelastningen till följd av kandidatarbete och andra inlämningar skulle vara högre, vilket underlättade planeringen av projektet.

Vi har lyckats ha en hög effektivitet, dvs. vi har skapat ett högt värde under den tid vi tillägnat projektet. Det faktum att vi varit sju personer i gruppen har möjliggjort att vi klarat av att leverera det vi gjort i enlighet med vårt ambitiösa scope. De uppgifter som delegerades i början av veckan togs med allvar och vid slutet av veckan var nästan alltid allting gjort.

B: Vi vill i framtida projekt se till att alltid vara tydliga och öppna med vad som försiggår i andra kurser så att teamet alltid är medvetna om varandras arbetsbelastning för att smidigt kunna anpassa sig och planera arbetet så att belastningen blir jämn, rättvis och hållbar. Vi ser även att det är viktigt att alltid vara proaktiv och meddela förändringar i tid, samt anpassningsbar då teamet är relativt stort och det kan uppstå vissa meningsskiljaktigheter.

A -> B: För att uppnå målet ser vi att det är mycket viktigt att lära känna varandra tidigt i arbetet så att man känner sig bekväm med varandra och trygg i att be om hjälp och meddela förändringar. Detta går delvis in i det som diskuterats i föregående fråga, där teambuilding och liknande är viktiga aktiviteter för att lära känna varandra och lägga grunden till öppenhet, transparens och ärlighet. Dessa aspekter är också viktiga för att alla ska känna att de har möjligheten att framföra sina åsikter. Alla dessa aspekter är något som tagits upp i det sociala kontraktet.

Design decisions and product structure

Kodkvalitet och kodningsstandarder

A: När projektet började satt vi och planerade samt lade mycket tid och fokus på hur appen skulle se ut och vilken funktionalitet appen skulle ha men inte mycket tid lades på hur vi faktiskt skulle programmera den. Vi började med att skapa en mockup på hur vi ville att appen skulle se ut. Vi hade flera olika färgscheman och kopplade de olika sidorna till varandra. Efter att vi gjort klart alla sidor bestämde vi oss för att börja koda funktionalitet med hänsyn till mockupen.

Som sagt tidigare så saknades kunskap inom Android Studio, detta ledde till att vi ville få saker och ting att "fungera" först för att sedan fixa till det senare under projektets gång. Det blev inte så att vi fixade det men istället fortsatte vi arbeta med koden som var temporär. Detta ledde till att kodkvaliteten blev lidande. Många problem uppstod och det som följde var att endast en liten andel klasser kunde samarbeta, fler klasser implementerade samma kod, komplexiteten ökade och bug-fixing blev svårt. Ju längre in i projektet vi kom, desto värre det blev. En anledningen till att kodkvalitet hamnade i andra hand var för att vi istället fokuserade på funktionaliteten för att maximera värdet för CIS och slutanvändaren.

När vi sedan kollade tillbaka till vår mockup insåg vi att det fanns lite från det ursprungliga mockupen kvar i vår app. Funktionalitet hade lagts till och tagits bort lite hur som helst, men mockupen uppdaterades inte för att undvika dubbelarbete. Under projektets gång bestämde vi oss för att granska varandras kod efter varje vecka men det blev sällan gjort. Om appen fungerade så gick vi vidare. I slutet av projektet bestämde vi oss att vi måste satsa mer tid och energi på att granska kod eftersom vi visste att vi inte hade gjort det riktigt innan. Det var för sent att börja fixa koden i den här fasen av projektet. Pelarna som höll upp koden var problemet men de satt fast och det fanns ingen tid kvar att omstrukturera basen.

B: I framtida projekt vill vi lägga mer fokus på kodkvalitet redan från början och använda oss utav Android Architecture-mönster som MVVM. En annan viktig skillnad till nästa projekt är att faktiskt ha granskning av kod efter varje vecka där gruppmedlemmar går igenom commits från andra gruppmedlemmar och ser till att hög kodkvalitet uppnås. Till och med innan granskning av koden sker vill vi att gruppmedlemmen analyserar sin egen kod. Dessutom skulle det vara bra med en tydligare plan kring användandet av mockup.

I framtida projekt vill vi också följa så kallad "best practice". För att planeringen ska vara så bra som möjligt vill vi använda oss av ett klassdiagram men också att få ett mer fullständig

mockup som representerar slutprodukten. Vi vill i framtida projekt driva kodningsstandarder genom att använda till exempel, “CERT Coding Standards” som bas för vår utveckling av projektet.

A -> B: För att få bra kodkvalitet och uppnå vårt mål måste vi redan i början av projektet diskutera våra erfarenheter och kunskaper i gruppen. Om vi inte är kunniga inom det område som det framtida projektet kräver måste vi ägna mycket tid åt att testa och utforska ämnet. En stor anledning till varför vi hade dålig kod var för att vi inte var kunniga inom Android Studio.

Vi måste också bestämma som grupp vilken Coding Standard vi ska följa. Det kan vi göra genom röstning. För att uppnå “best practice” måste vi dokumentera, kommentera, indentera, följa ett konsekvent namngivningsschema och undvika så kallad “Deep Nesting”.

För att få rutiner i arbetet är ett förslag att granskningen av kod alltid utförs på en måndag med en partner.

Användandet av API:er och externa bibliotek för att skapa kundvärde

A: För att uppfylla de krav CIS ställde på applikationen var användandet av diverse API:er och externa bibliotek nödvändigt. Nedan följer de tre delar av applikationen där vi använt oss av APIer för att skapa värde.

Då CIS redan använder en kalender för att lägga in kommande aktiviteter ville de att applikationen skulle hämta information om aktiviteter därifrån. De använde Google Kalender, så följaktligen nyttjades tillhörande API. Tillämpningen av API:n var däremot aningen inflexibel då en API-nyckel användes för att hämta en URL till kalendern i JSON-format. En metod skapades sedan för att konvertera innehållet till aktivitets-objekt i applikationen. Troligtvis finns det externa bibliotek eller liknande som hade kunnat läsa in kalendern på ett bättre sätt, men då flera gruppmedlemmar hade tidigare erfarenhet av att arbeta med JSON samtidigt som ingen hade använt API:er förut, var detta en lösning som fungerade bra för applikationen i denna kursen.

Ytterligare uttryckte CIS intresse för att spara rekommenderade och favoritmarkerade sporter, samt avklarade utmaningar. För att tillgodose detta krävdes att information kunde sparas mellan olika användningstillfällen. Därför användes SharedPreferences API, vilket sparar enkla nyckel-värde par, vilket var det vi ville uppnå.

Kalendern i aktivitetsvyn var också ett viktig supplement till applikationen och då Android Studios standardkalender saknade mycket av funktionaliteten som efterfrågades

användes CompactCalendarView, som är baserad på kalendern i Android Studio, för att enklare kunna lägga till och visa event i kalendern.

B: I ett framtida projekt skulle det vara användbart att utforska hur man kan använda externa resurser mer effektivt. Att använda API:er och färdiga lösningar i ett projekt är tidseffektivt och kvaliteten på den externa koden håller generellt god kvalitet då den vanligtvis har använts och testats i flertalet likartade situationer tidigare. Det är därför ett perfekt alternativ för att fort skapa värde för kunden.

A -> B: För att kunna använda externa resurser mer effektivt i framtiden krävs det att man använder externa resurser oftare. Då alla gruppmedlemmar har uttryckt att de inte hade någon större erfarenhet inom området krävs det att man börjar utforska API:er och externa bibliotek, samt hur de implementeras i projekt mer praktiskt. Detta projektet var därför ett perfekt första steg, men för att effektivt kunna använda detta i framtiden krävs det att man gör fler liknande projekt; på egen hand, i skolan eller i arbetslivet.

Val av utvecklingsmiljö och konsekvenser för vår externa aktör

A: När gruppen fått bekräftat att CIS var positiva till utvecklandet av en mobilapplikation, utsågs två i gruppen för att undersöka vilka utvecklingsmiljöer som passade bäst. Valet föll på Android Studio och att skriva applikationen i Java. Detta då vi i tidigare projekt skrivit Java-kod och använt IntelliJ och Eclipse, där Android Studio är baserat på IntelliJ, men även för att Android Studio är en väldigt populär utvecklingsmiljö med mycket dokumentation online vilket är bra för oss med mindre förkunskap av apputveckling.

För CIS blev då konsekvensen att applikationen bara skulle fungera för Android-telefoner, men vi ansåg det inte rimligt att under den här kursen ta fram en applikation för både iOS och Android, vilket de hade förståelse för. Alternativ som att istället använda Flutter hade kanske kunnat lösa detta problemet, men det ansågs vara för olikt det vi tidigare använt för att kunna leverera ett bra slutresultat.

B: Valet av utvecklingsmiljö kan ha stora konsekvenser på slutresultatet då olika typer av miljöer fungerar olika bra för olika applikationer. Att därför utse personer med uppdrag att undersöka vilken utvecklingsmiljö som passar bäst är därför något som vi vill bevara i framtida projekt. Det hade dock varit fördelaktigt att arbeta i en miljö vi var mer familjära med för att kunna arbeta mer effektivt. Med stor sannolikhet blir även produkten som levereras bättre desto bättre vi kan utnyttja utvecklingsmiljön, vilket gynnar framtida kunder.

A -> B: För att kunna arbeta i familjära utvecklingsmiljöer, måste man ge sig tid till att bli familjär med nya miljöer. Användandet av Android Studio var således inget misslyckande, utan snarare ett väldigt viktigt steg för att kunna nå B. För att nå hela vägen till B krävs det därför att man gör fler liknande projekt i framtiden och inte är rädd för att testa eller lära sig nya saker.

Testning

A: I Definition of Done finns ett kriterium angående att koden ska testas innan en task är avklarad. Det finns ett test-dokument som beskriver olika sätt som applikationen har testats på. Primärt har testningen skett genom en emulator och exempelvis JUnit tester har inte använts i samma utsträckning. Detta beror framförallt på att vi i gruppen ansåg att JUnit tester fyller ett större syfte när applikationen behöver underhållas under en längre tid framöver. I det här projektet hade vi inte det här i lika stor åtanke, utan istället låg fokus på hur slutprodukten fungerade och vilket värde den gav CIS. Detta var enklare att testa genom en emulator.

I slutet av projektet var dock tanken att fler JUnit tester skulle skrivas, men väl då insåg vi att koden var utformad så att den var svår att testa på ett enkelt sätt. Det skulle därför vara ett stort arbetet att skapa givande tester vilket inte ansågs vara värt det, då vi redan hade testat all funktionalitet på andra sätt tidigare.

B: I framtida projekt av den här karaktären skulle det vara fördelaktigt att skriva kod som är enklare att testa, men testning hade troligtvis prioriterats på ett liknande sätt som under det här projektet, men i större projekt med fler inblandade är det betydligt viktigare att testa koden på ett sådant sätt att en utomstående programmerare enkelt kan ta del av eller använda testerna.

A -> B: För att koden ska kunna testas är det bra att man tänker på testerna direkt medan koden skrivs och sedan skriver testerna i samband med detta. Då kan man också märka om metoden gör för mycket eller är onödigt hårdkodad, vilket kan vara positivt ur ett kodkvalitativt perspektiv. I större projekt är det också viktigt att avsätta mer tid åt testning under varje sprint för att säkerställa att det blir av.

Vanlig och teknisk dokumentation

A: Under projektet gång dokumenterades många idéer, förslag, och förklaringar. Det fungerade relativt bra att skapa dokumentationen men att sedan uppdatera och

upprätthålla den gjordes inte i tillräcklig utsträckning. Dokumentation som skapades under projektet innefattar mockups, CIS-sports information, kodgranskning, tester, teknisk beskrivning, CIS-quiz frågor, handlednings- och mötesanteckningar, project scope, business model canvas och sist men inte minst javadoc.

Vi använde mycket dokumentation som var fördelaktig för projektet. Mockups gav oss en bra bas för designen och gjorde så att vår externa aktör fick bra visuell återkoppling. Handlednings- och mötesanteckningar gav oss inblick i de uppdrag och den funktionalitet som skulle implementeras samt strukturera arbetsflödet. CIS-quiz och sports dokumentation gav vår externa aktör direkt inflytande på det som skulle implementeras.

Vi lyckades dokumentera vanlig information samt teknisk information ganska bra, men vi saknade ändå dokumentation på båda sidorna. Teknisk dokumentation som vi saknade var klass- och komponentdiagram, domänmodell och “use cases”. Även om vi hade bra vanlig dokumentation så saknades kvaliteten och strukturen på den dokumentationen för att göra den enklare att läsa och följa. En stor anledning till varför vi inte hade mycket teknisk dokumentation samt bra struktur på vanlig dokumentation var att vårt projekt är relativt litet och inte behöver underhållas efter projektet är slut. Därför kände vi att det kunde bli lite överflödigt.

B: I framtida projekt vill vi se mer teknisk dokumentation och bättre struktur på vanlig dokumentation. Det beror så klart på hur stort projektet är, men om vi antar att det är ett stort projekt skulle vi ägna mer tid åt uppdatering av dokumentation så att dokumentationen uppdateras samtidigt som koden och funktionaliteten. Detta leder till att alla hänger med och är i fas under projektet samt minskar missförstånd.

I framtida projekt vill vi ägna tid åt “use cases” som berättar för oss hur vår app bör fungera samt fundera på vad som kan gå fel. Det ger oss en lista på vad som måste implementeras och utifrån listan vet vi ungefär hur lång tid projektet kommer att ta. Vi kan till och med utifrån listan diskutera vilken funktionalitet som behövs och inte behövs .

A -> B: För att uppnå vårt mål kopplat till dokumentation behöver vi lägga undan tid varje vecka för att uppdatera dokumentationen. Detta bör göras regelbundet. Vi behöver ägna mer tid i början av projektet till att skapa teknisk dokumentation (“use cases”, klassdiagram mm). Dokumentation ska granskas av gruppmedlemmar och produktägaren för att alltid hålla oss inom ramarna till det ägaren vill. Detta ger oss en stabil bas att fortsätta vidare på samt en bra utvecklingsprocess.

Application of Scrum

Projektroller i gruppen

A: Redan i början av projektet skapade vi tydliga roller inom gruppen. Dessa roller omfattar då Scrum master, Product owner och Zoom-ansvarig. Dessa roller har gjort det möjligt att lätt kunna komma igång med arbetet samt att alla möten har haft en bra struktur med hjälp av till exempel en dagordning som Scrum mastern har bestämt innan mötet. Trots detta har rollerna blivit lite som ett grupparbete där den som hade rollen fick huvudansvaret medan gruppen hjälpte till då och då.

B: I framtida projekt skulle det vara bra om dessa roller utnyttjas bättre, där personerna med respektive roll får mer väldefinierade arbetsuppgifter. Alternativt kan man ha roterande ansvar för dessa roller.

A -> B: Dessa problem uppstod främst pga. okunskap. Vi delade ut dessa roller i början när vi inte hade bra koll på vad de innebar. För att lösa detta skulle de som tog på sig rollen behöva ha full koll på vad det innebär innan den blir spikad. Om då ingen vill ha en specifik roll, blir det då bättre att rotera den mellan de andra gruppmedlemmarna. Detta kräver då i framtiden en bättre planering av rollerna i ett projekt. Erfarenheterna från detta projekt bidrar mycket till att lösa detta problem.

Agila arbetssätt i projektet

A: I detta projekt har vi använt oss av en Scrum board i Trello mycket för att strukturera vårt arbete. Vi har delat upp denna Scrum board i Epics, User Stories and Tasks.

Epics användes för att dela upp projektarbetet i tre större kategorier, plus en extra för admin-uppgifter. Dessa tre kategorier bestämdes tillsammans med vår externa aktör över vad som behövde vara med för att uppnå appens syfte. Utifrån dessa epics skapades user stories över mer specifikt arbete som behövde göras. Dessa följer syntaxen "As a … I want ... ". Detta gjordes för att följa riktlinjerna för ett agilt arbete enligt vår handledares exempel. Dessa user stories fick därefter en prioriteringsetikett samt en svårighetsetikett för att kunna planera och strukturera vilka user stories som skulle hanteras varje sprint.

Prioriteringsetiketterna har en färg och en siffra som visar i vilken ordning varje user story behövde prioriteras i för att kunna fortsätta arbetet (1 - Röd, 2 - Gul, 3 - Grön). Dessa bestämdes i samband med vår externa aktör med hänsyn till vad som ansågs viktigast samt av oss utvecklare som behövde påbörja vissa saker innan andra.

Svårighetsetiketterna har en färg, en svårighetsgrad samt tre siffror som är representativt för den svårighetsgraden (ljusblå - Lätt (0,1,2), blå - Mellan (3, 5, 8), mörkblå - Svår (13, 20, 40)). Färgen har ingen större relevans än att vi upplevde att det var bra att vi hade en nyans av samma färg som inte var med i prioriteringsetiketterna. Siffrorna kommer ifrån Fibonacci Agile Estimation. Detta var för att få till ett känt estimations system i vår Scrum board. Vi som grupp delade ut arbetsbelastningen för varje user story beroende på hur komplicerad den var.

I början av projektet sattes dessa etiketter ut på de user stories som skapades. När de första med prio 1 och delvis prio 2 var klara delades det ut nya prio 1 klassificeringar. Detta var för att omstrukturera vad som behövde läggas fokus på då vi hade kommit en bit in i projektet.

Dessa user stories hade utöver etiketterna även en lista med acceptanskriterier. Dessa acceptanskriterier skapades främst som egna anteckningar för oss själva för att sedan kunna skapa tasks utifrån dem. Dessutom bidrog vår externa aktör med sin åsikt. När alla acceptanskriterier var klara kunde vi acceptera user storyn som klar.

Tasks skapades utifrån respektive user story. Dessa tasks är väldigt simpla i sin formulering över vad som behövde göras. Varje task har däremot en livscykel genom fyra steg: Tasks, In Progress, Testing och Done. Vid testningsfasen testades tasken av en annan utvecklare i gruppen för kolla att all kod fungerade som den skulle.

Definition of Done användes för att definiera när tasks och user stories var klara. Denna följdes bra under projektets gång men ibland togs mindre genvägar för att få vissa saker klara snabbare. Dessa genvägar omfattar då att acceptanskriterierna uppfylldes med tillit till de som arbetat på dem och att dessa utfört sin uppgift korrekt, vilket inte helt följer den formella kontrolleringsprocessen.

Våra user stories var numrerade till vilken epic de tillhörde och i vilken ordning de skapades, till exempel 2.3 var under epic två och var den tredje user storyn skapad. Detta format fortsatte i tasks där till exempel task 2.3.1 var den första tillhörande user story 2.3.

I början av varje sprint bestämde vi i gruppen vem som skulle göra vad. När man tog/fick ett kort att arbeta på tilldelade man sig själv det kortet. Detta gjordes för att kunna se vad som behövde arbetas på samt vilken tillfällig grupp man var i för parprogrammering, något som användes mycket under projektets gång.

B: Vi känner oss som grupp nöjda med hur vi har använt olika praxis inom vårt agila arbetssätt. Men det finns några saker som vi definitivt skulle kunna förbättra.

I våra user stories hade vi endast "As a … I want ... " men den borde egentligen ha varit "As a … I want ... because… ". Detta är något att ta med sig till framtida projekt.

Någonting som skulle behövas i ett framtida projekt är ett sätt att visa vilka buggar som finns i programmet. Upptäckta buggar dokumenterades inte, mer än att antalet räknades i KPI:erna, vilket ökade risken för att bli bortglömda och olösta.

A -> B: För att uppnå denna vision inför framtida projekt behövs en bättre planering av Scrum boarden samt erfarenhet. För att underlätta hanteringen av buggar hade en bug-kolumn i vår Scrum board kunnat användas. Detta skulle underlätta genom att man lätt ser buggar som måste åtgärdas. De främsta av de problem som vi upplevde uppstod pga. att vi saknade kunskapen om vad som egentligen behövdes eller att vi rusade igenom viktiga delar. Med lärdomarna från detta projekt har vi dock skaffat oss erfarenheten att göra det bättre till nästa projekt.

Sprintgenomgången och dess betydelse

A: Efter varje sprint hade gruppen en dag tillägnad administrativt arbete där den gångna sprinten utvärderades och nästa planerades, både internt och tillsammans med externa parter. Anledningen till den okonventionella tidsrymden för sprintarna var att majoriteten uttryckte en vilja att arbeta även under helgen.

Dagen inleddes vanligtvis med en intern reflektion över sprintens arbetsprocess, vilken KPI:erna fungerade som underlag för. Under dessa tillfällen kunde varje gruppmedlem uttrycka sina tankar om sprinten och delvis utifrån detta anpassades nästa sprint. Dessutom gjordes en initial estimering av user-stories samt utformandet av en mötesagenda med viktiga punkter att behandla inför genomgången med CIS. Baserat på den temporära estimeringen valdes det också ut ett antal user-stories som en potentiell plan för den nästkommande sprinten.

Den andra delen av sprintgenomgången skedde tillsammans med den externa aktören, CIS, och den internt utsedda produktägaren hade ett större ansvar att leda dessa. Under mötena demonstrerades veckans resultat för att därefter diskutera huruvida de levde upp till förväntningarna eller ej, samt anledningar till detta. Med avseende på utfallet fördes diskussion kring hur ett resultat av liknande värde kan reproduceras eller undvikas, vilket påverkade utformningen av nästa sprint.

Efteråt fanns det möjlighet för CIS att föreslå eventuella förbättringar eller utökningar, som skulle vara värdefulla för dem. På så sätt var deras tankar och idéer alltid en central del av utvecklingsprocessen, vilket medförde att varje sprint kunde ta en värdeskapande

inriktning. I samband med detta presenterades också Scrumboarden för att ge en visuell representation av arbetets kvarvarande uppgifter. Tillsammans valdes det även ut ett antal user-stories för nästa sprint baserat på den estimerade svårighetsgraden att implementera samt värdet som den skulle medföra.

Efterhand märkte vi att produktiviteten i sprintgenomgången ökade. I början av projektet tog denna genomgång minst en hel dag, vid något tillfälle en och en halv. Mot slutet hade vi vant oss vid arbetssättet och byggt upp en bättre struktur med dagordning och liknande vilket gjorde att tidsåtgången minskade drastiskt till endast några timmar.

Mot slutet av arbetet blev dock själva värdeskapandet aningen mindre betonat. Orsaken kan förklaras med att det finns en balansgång mellan att bygga något av värde, att göra det snabbt, och att göra det på rätt sätt. Det sistnämnda hamnade på bakfoten under de första veckorna eftersom gruppen var mån om att leverera resultat. På grund av detta uppstod komplikationer med att utöka viss funktionalitet senare i projektet. Som en del av Definition of Done var kodkvalitet relativt bortprioriterat, och det var god tid att lägga ytterligare resurser på det.

B: Överlag har arbetets sprintgenomgångar varit produktiva sessioner och det finns mycket som skulle kunna tas med till framtida projekt. Dock har det lämnat lite att önska i vissa avseenden. Ett av de mest iögonfallande var planeringen av hur klandervärda saker skulle åtgärdas. Till exempel var kodkvalitet med på agendan av sådant som behövde förbättras i flera sprintar, men det utformades inte en stabil plan för att åtgärda detta förrän långt senare.

Ytterligare något att begrunda var att user-stories som temporärt lades in i Sprint backlog under det interna mötet delades upp i tasks. När Scrumboarden sedan presenteras för intressenten kan detta indirekt influera denne att följa gruppens dåvarande planering av sprinten. Detta eftersom ett annat val av user-stories skulle göra det nyligen gjorda arbetet med tasks meningslöst.

A -> B: För att påskynda agerandet till upptäckta problem kan det vara lämpligt att investera mer tid och fokus i att specificera en plan och tilldela uppgifter. Dessutom hade det varit fördelaktigt med mer struktur kring de olika mötenas tidsramar eftersom det då skulle kunna införas fler pauser. Brist på dessa kan ha varit en orsak till att planeringarna inte var optimalt utformade. Detta eftersom de ofta formulerades mot slutet av mötena när energinivån var låg.

Framöver bör det inte heller läggas vidare arbete på att utveckla synliga uppgifter som egentligen bara anbelangar utvecklarna inför möte med externa aktörer. För övrigt hade en större mängd punkter som tas upp internt kunnat uttryckas under mötet med externa

aktören istället. Detta hade eventuellt kunnat skapa en bättre diskussion och förståelse för varandras sidor.

Hjälpmedel och inlärningsprocessen

A: För att utveckla applikationen har en mängd olika verktyg utnyttjats, varav flera inte använts tidigare. Ett verktyg som använts kontinuerligt genom hela arbetet var GitHub samt GitHub Desktop. Till skillnad från I-studenterna, var gruppmedlemmarna från IT välbevandrade i versionshantering sedan tidigare projekt. Därför fördelades arbetet under den första sprinten inom mindre grupper, med åtminstone en person från vardera program. På så sätt kunde alla lära sig om GitHub, men också varandra. Gruppindelningen kulminerade i en produktiv första sprint, och lade grunden för ett effektivt samarbete.

Utvecklingsmiljön, Android Studio, var främmande för hela gruppen och därför fick varje individ i uppgift att läsa på och lära sig om Android Studio på egen hand i början av projektet. Alla lär sig på olika sätt så en flexibel inledande arbetsuppgift fungerade bra. För att utöka gruppens förståelse ytterligare fanns det möjlighet att dela med sig av lärdomarna under nästkommande möte. Dessutom tillämpades parprogrammering i början, med syfte att kunna bolla idéer och komplettera varandras kunskaper om Android Studio för att lösa uppgiften. Samtidigt kunde flera uppgifter genomföras parallellt då det fanns flera grupper istället för en enda. Allt eftersom uppgifter betades av från “Product Backlog” blev gruppen bekvämare med Android Studio, och det kunde ske en gradvis övergång till mer individuellt arbete.

Ett annat verktyg som nyttjades frekvent var Scrumboarden i Trello. Majoriteten av gruppen hade använt redskapet tidigare, men då följdes vattenfallsmodellen istället. Dock etablerades en grund för agil användning av Scrumboard under föreläsningar och övningstillfällen, men i planeringsfasen av den första sprinten var osäkerheten fortfarande hög. Därför rådfrågades gruppens handledare och kritik välkomnades. Scrumboardens utformning samt dess innehåll blev dock godkänt och därefter fortsatte arbetet med denna utan större hinder.

I början av projektet användes även Miro för att skapa en mockup för applikationen. Likt en lågnivåprototyp kunde potentiell funktionalitet demonstreras för intressenten, men också design och omfattning. Inlärningskurvan för detta verktyg var förhållandevis låg och i kombination med det intuitiva grafiska gränssnittet krävdes inte mycket tid till att börja använda det för vårt ändamål. Efter varje passerad vecka blev dock mockupen allt mer utdaterad på grund av svårigheter att underhålla.

B: Om man besitter kunskap och har en bred repertoar av verktyg kommer framtida projekt underlättas eftersom det i sådant fall finns möjlighet att välja det bäst lämpade för situationen. Framöver är därför förhoppningarna att förstå och kunna använda ytterligare hjälpmedel som nyttjas för mjukvaruutveckling. Exempelvis var Mockup-verktyget godtyckligt för detta projekt, men det hade varit dåligt anpassat för längre arbeten med mer fokus på design eftersom det var invecklat att uppdatera den. Dessutom sker det ständiga uppdateringar och förändringar av existerande verktyg, vilket innebär att man aldrig är färdiglärd. Betydelsen av att vara villig att lära sig nya saker är således hög.

A -> B: För att utöka befintliga kunskaper kan man fokusera på att göra inlärningsprocessen intressant. Detta är dock högst subjektivt, vilket medför att man måste finna sådant som fungerar för en själv. Genom att utforska nya områden och aktivt bearbeta och reflektera över inlärningsprocessen är det möjligt att upptäcka mönster mellan sådant som fungerat bra respektive dåligt. På så sätt kan man hitta de sätt som man personligen lär sig bäst på.

Användning av föreläsningar och litteratur

A: Under detta projekt har vi som grupp varit med på de olika föreläsningarna och lyssnat på innehållet i dem. Efter vissa föreläsningar har vi haft möten och därmed reflekterat tillsammans över dessa. När det kommer till litteraturen på kurshemsidan har vi främst läst om hur man utför kodgranskning i ett mjukvaruutvecklingsprojekt.

B: Vi i projektgruppen upplever att vi har använt Scrum effektivt. I framtida projekt kommer vi troligen inte göra några större förändringar. Om det skulle dyka upp nya tekniker eller strategier för att använda Scrum kommer vi ta del av dessa. Dessutom skulle uppstartstiden kunna minimeras genom inläsning av litteratur om vald utvecklingsmiljö. Detta skulle även förebygga misstag som man gör när man testar sig fram ovetandes.

A -> B: För att uppnå detta kommer vi arbeta som vi har gjort under detta projekt. Alternativt, om det skulle vara någonting nytt som vi har missat eller någonting nytt som har hänt med Scrum sedan sist, får vi ta del av relevant litteratur för att lära oss att applicera det i framtida arbeten.