Agile Endor - Final Report

ALMROTH JOHAN, Department of Computer Science and Engineering
BERG PHILIP, Department of Computer Science and Engineering
IBRAHIM MIRAI, Department of Computer Science and Engineering
JONSSON MAGNUS, Department of Computer Science and Engineering
LYSHOLM ALEXANDER, Department of Computer Science and Engineering
THALL JOHANNA, Department of Computer Science and Engineering

Contents

Authors' addresses: Almroth Johan, Department of Computer Science and Engineering, gustalmjok@student.gu.se; Berg Philip, Department of Computer Science and Engineering, gusberpha@student.gu.se; Ibrahim Mirai, Department of Computer Science and Engineering, gusibrmi@student.gu.se; Jonsson Magnus, Department of Computer Science and Engineering, gusjonmaef@student.gu.se; Lysholm Alexander, Department of Computer Science and Engineering, guslysal@student.gu.se; Thall Johanna, Department of Computer Science and Engineering, gusthalljo@student.gu.se.

1 INTRODUKTION

Projektets syfte var att lära deltagarna att arbeta med **agila** metoder inom utveckling. Ett vanligt och populärt ramverk inom agila utvecklingsprojekt är **scrum**. Gruppen valde att arbeta med scrum delvis på grund av dess popularitet inom IT-branschen och då dess metoder är lämpliga samt applicerbara för ett digitalt samarbete. På grund av rådande omständigheter med COVD-19 pandemin har gruppen tvingats till att hitta nya mötes-plattformar och lösningar. Kommunikationskanalen **Discord**[1] användes för möten samt daglig kommunikation, **Jira**[3] och **Github**[2] användes som de två utvecklingsplattformarna för projektets kodbas och planering.

Målet med produkten i projektet var att sträva efter och bidra till ett eller flera av Förenta Nationernas (FN) 17 hållbarhetsmål [6]. Det här projektets fokuserade primärt på mål 1 *No poverty* och mål 12 *Sustainable Consumption and Production* samt delvis mål 6 *Clean water and sanitation.*

För att leverera så mycket användarnytta som möjligt under de veckor projektet pågick och öka kännedomen inom olika språk beslutade sig gruppen för att skapa en hemsida med **React**[7]. De flesta medlemmarna hade ingen tidigare erfarenhet av webbutveckling och introduceras under projektets gång till flera språk och ramverk så som **HTML**, **CSS**, **JavaScript**.

Då huvudsyftet med projektet var att lära sig jobba agilt lades fokus på utvecklingsmetoden snarare än resultatet av utvecklingen. I rapporten beskrivs hur olika agila utvecklingsmetoder applicerades, varför dessa valdes samt hur dessa påverkade arbetet. Den går även in i detalj för de olika metoderna och redogör hur de styrde projektet samt hur gruppens förståelse för dessa utvecklades under tid.

2 CUSTOMER VALUE AND SCOPE

2.1 Målet med appen & Prioritering av funktioner

Målet med produkten var att skapa en informativ hemsida som förmedlar kunskap på ett interaktivt, nyanserat och roligt sätt. Då kunskap lägger grunden för en djupare förståelse prioriterades att representera data på ett roligt och mottagligt sätt för användaren. Målgruppen var bred och projektet hade inga konkreta **personas** med målsättningen att hemsidan skulle vara estetiskt tilltalande och enkel för alla användare.

För att uppnå så hög kvalité som möjligt lades fokus initialt på ett fåtal men viktiga funktioner. Ett färgschema skapades för att passa det generella temat på hemsidan vilket, givet de 3 valda FN-målen, kretsade kring vatten och vattenanvädning. Vidare inleddes utvecklingen av 3 sidor, ett Quiz för att göra hemsidan lekfull, en nyhetssida kopplad till reddit för att låta användaren ta del av det senaste inom ämnet, samt en tabell med ren data med vattenanvändning per land. Datan som visades ansågs lättolkad men svårmottaglig för användaren vilket föranledde en vidareutveckling av en interaktiv karta. Målet med kartan var att ytterligare visa på olika länders användning och tillgång till vattenresurser.

I retrospekt hade utvecklingen underlättats om en, eller flera, personas tagits fram i kombination med konkreta målsättningar för applikationen. Det var stundtals svårt att avgöra vilka funktioner som var viktigast då målen och tillvägagångssättet var tvetydiga. Likaså blev designen varierande och beslut fattades ofta baserat på utvecklarnas egna åsikter. En tydligare planering med konkreta mål och en plan för att nå målen hade underlättat skulle projektet göras om.

2.2 Personliga mål

I början av projektet fastställdes gruppens mål både med kursen och applikationen. Ambitionsnivån sattes relativt högt och samtliga var nyfikna på att bekanta sig med agila metoder och React/Javascript. Det var med en ödmjuk inställning projektet påbörjades och stort fokus lades på de agila detaljerna. Gruppen värderade ett bra samarbete vilket speglas i **KPI**:erna som togs fram tidigt och återfinns i sektion *KPIs*.

2.3 Användning av user stories

User Stories låg till grund för hela utvecklingsprocessen i projektet. Efter att ha fastställt vad som skulle göras i projektet påbörjades en idealiseringsfas för att ta fram idéer som skulle kunna bidra med värde till användaren. När idéerna fastställts och konkretiserats konverterades dessa till User stories i mjukvaruverktyget Jira. Jira har funktioner så som **sprint**, **backlog**, **story points**, **tasklists** med flera för user stories. Dessa funktioner användes i den utsträckning gruppen ansåg nödvändig och rimlig givet kursens mål.

Dessa framtagna user stories tilldelades en prioritetsordning baserat på hur mycket värde de skapade för användaren samt hur viktiga de var för produkten i sin helhet. Då en del user stories var beroende av andra vägde även detta in i prioriteringsordningen för varje sprint.

För att estimera arbetsbördan i varje user story tilldelades de poäng inom skalan fibonacci-nummer. Fibonacci-skalan används för att göra en relativ estimering. Användningen av skalan tvingar utvecklarna att tänka efter och motivera sitt beslut istället för att gå på känsla. Skalan användes i kombination med **planning poker** för att ytterligare främja individuell reflektion. Efter att varje medlem uppvisat sin estimering ombads de mest avvikande att motivera sina tankar. På så vis kunde gruppen identifiera och reflektera över eventuella svårigheter eller enkelheter.

Användningen av user stories fungerade till stora delar väl. De flesta kunde utföras enskilt och undvek således beroenden som stoppade utvecklingen på olika fronter. Estimering av svårighetsgraden var dock svår då många nya språk och verktyg introducerades i projektet. Vissa user stories tenderade att bli för stora och skulle egentligen ha brutits ned i flera mindre. På så sätt hade planeringen varit enklare då vissa user stories inväntande andra för att kunna bli påbörjade.

2.4 Acceptanskriterier för applikationen

Hemsidans stabilitet och kompatibilitet över flera plattformar prioriterades högt då det ansågs bidra till en bättre användarupplevelse samt öka hemsidans tillgänglighet. User stories ackompanjerades med tekniska krav för att fungera korrekt för olika tänkbara skärmupplösningar, specifikt telefoner och hög-upplösta widescreen-skärmar.

För att säkerställa grafisk stabilitet användes Reacts/JEST bibliotek[8] för så kallade **snapshot tester**. Dessa verifierar att hemsidans utseende beter sig som förväntat. Det var viktigt då den skulle fungera på åtskilliga skärmar för en bättre användarupplevelse.

2.5 KPIs

(A).

KPIerna i början av kursen. De första KPIerna vi skrev innehöll **velocity**, **happiness** och antalet **buggar** vi hittade i koden.

Vi valde att mäta velocity för att undersöka om något påverkade utvecklingshastigheten i projektet. När **unit testing** introducerades i vår **definition of done** blev det uppenbart att dessa hade en avsevärd negativ inverkan på hastigheten. En bidragande orsak till detta var att ingen i gruppen hade tidigare erfarenhet av de introducerade testing-verktygen **Jest** och **react-testing-library**.

Happiness KPIn. Varje vecka fyllde vi i happiness, stress och productivity på en skala 1-5 för att mäta gruppens välmående med hänsyn till de mått som tagits fram i happiness. Det användes till exempel för att justera fördelningen av antal story points under varje sprint.

Debug var en KPI som mätte hur många buggar som kom igenom **code reviewn** och testningen. Idén var att basera denna på **github issues** i vilken gruppen skulle skapa issues när buggar upptäcktes

- Velocity
  - Estimeringar av user stories. Hur: Mät med story points.
  - Poäng avklarade av total poäng varje sprint.
- Happiness
  - Hur lycklig känner du dig? Från 1-5
  - Hur stressad känner du dig? Från 1-5
  - Hur produktiv känner du dig? Från 1-5
- Debug
  - Använd ‘issues’ funktionen på Github

2.5.1 Uppdaterade KPIer. Halvvägs in i projektet bytte vi ut debug KPIn som höll koll på issues till **estimation accuracy** då den inte ansågs värdefull. Estimation accuracy räknades ut genom att dela de avklarade story points varje sprint, med vad vi ursprungligen estimerade att vi skulle klara under sprint planning, vid start av sprinten. Vi fick då ut vår estimation accuracy som en procent och kunde på så sett se hur bra våra estimat hade varit för varje sprint.

- Estimation Accuracy
  - Hur duktiga är vi på att nå våra mål?
  - Hur väl träffar vi vår velocity? Avklarade story points / totala antal story points.

Exempel på användbar data vi får ut av KPIerna. Några exempel på hur KPIerna kan ge oss intressant information är 1 och 2 där vi ser att stressen gick upp under sprint 3. En bidragande orsak till detta tros vara uppdateringen av definition of done som lade till att koden skulle vara testad. Ingen i teamet hade någon erfarenhet av Jest, react-testing-library, snapshot testing eller testning av front-end kod.

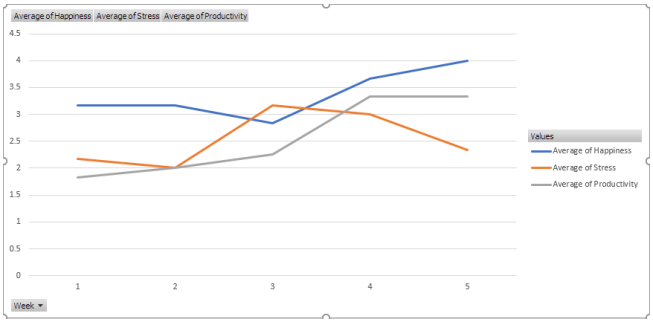

Fig. 1. teamets snittvärden varje sprint

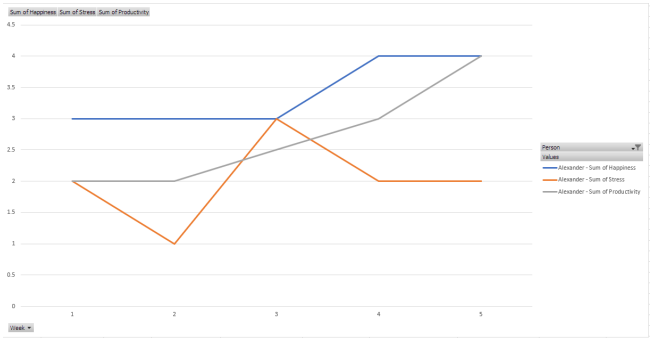

Fig. 2. Alexanders värden varje sprint

Vi kan även se i 3 att velocityn gick ner väldigt mycket när vi la till testning som ett krav. Detta hade en negativ inverkan på stressen eftersom det nu kändes som att vi inte fick något gjort. Men produktiviteten, stressen, och velocityn blev successivt bättre efter några sprintar när vi började förstå hur allt fungerade.

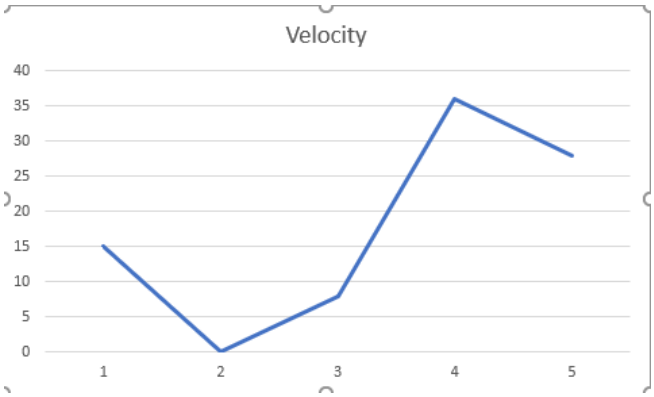

Fig. 3. Velocity för varje sprint

(B). Framöver så skulle vi vilja hitta och byta KPIer som inte ger något värde tidigare i projekt. Att samla in data om KPIer som inte används är bara slöseri med tid som istället hade kunnat spenderats på mer värdefulla KPIer.

(A → B). Ett sätt att uppnå B är att lägga ner mer tid på att analysera processen under våra **retrospectives**, vilket är en viktig ceremoni inom scrum som vi inte la tillräckligt med fokus på under vårt projekt. Om vi hade organiserat riktiga retrospective möten istället för att ibland slänga in en kort retrospective vid samma tillfälle som vi hade sprint review och skrev våra reflections så hade vi haft ett tillfälle att analysera vårt arbetssätt.

3 SOCIAL CONTRACT AND EFFORT

3.1 Social Contract

- Var öppen för kritik.
- Vara trevlig.
- Gör din del.
- Våga fråga.
- Tala om ifall du inte kan delta på något möte.
- Du är viktig.
- Skriva på svenska.
- En sprint går från Måndag 10:00-Måndag 08:00.

Alla personer i gruppen har levt upp till det sociala kontraktet som vi framställde i början av kursen och visade engagemang redan från start. Vid behov av hjälp ställde man frågor och alla gjorde sitt yttersta för att vara så hjälpsamma som möjligt. Det skapade en trygg stämning i gruppen som gjorde det enklare att leva upp till det sociala kontraktet.

Nya ändringar förekom i kontraktet under kursens gång vid behov. Ett exempel var att vi bestämde oss för att skriva så mycket vi kunde på svenska med undantag av kod, kommentarer, agila termer och vissa individuella reflektioner.

Ännu en förändring i det sociala kontraktet var tiderna för våra möten. Vi insåg under de första veckorna att våra sprints blev för korta för att hinna med våra önskemål. Därav flyttades ett av våra möten som var på Fredagar till Måndagar efter helgen istället, det gjorde att man även hade hela fredagen samt helgen att lägga ner tid på arbetet. Att förlänga våra sprints var ett väldigt bra val eftersom att det minskade stressen.

3.2 Effort

Vi har lagt ca 780 timmar på detta projekt. Fördelningen på dessa 780 timmar är:

- Möten: ca 180 timmar sammanlagt.
- Egna arbeten: ca 600 timmar sammanlagt.

(A). I början av projektet hade vi inte en klar idé om vad för slags applikation vi ville göra eller vad den skulle innehålla/förmedla, så i början gick det åt mer timmar till möten för att dela med sig av sina idéer. Desto längre in i projektet vi kom minskade behovet av längre möten då vi hade en klarare bild på vad vi skulle göra och litade på att vi kunde uppnå målen vi hade satt upp den veckan.

Första veckorna etablerades vad för projekt vi ville skapa, hur vi skulle gå tillväga samt materialet vi behövde för att åstadkomma det. Då de flesta i gruppen inte hade någon tidigare erfarenhet av att arbeta av materialen vi hade valt, gick de första veckorna ut mycket på att bekanta sig med de nya redskapen. En i gruppen som hade tidigare erfarenhet av att arbeta med React, hjälpte de resterande i gruppen att förstå grunderna genom att hålla i en föreläsning med demos och exempel. Vid fördelning av user stories, tog de mer erfarna gruppmedlemmarna de svårare uppgifterna så resterande gruppmedlemmarna kunde lättare komma in i materialet och utveckla sin kunskap i sin egen takt, samtidigt som vi bibehöll vår velocity.

Resultatet av de första veckorna var bra men effektiviteten sänktes halvvägs in i projektet när vi började öka antalet story points vi tog oss an i våra sprints, samtidigt som vi introducerade tester när majoriteten av gruppmedlemmarna aldrig tidigare hade skrivit några tester i React. Produktiviteten sänktes då väldigt drastiskt och vi fick problem med att avsluta våra user stories den sprinten. De sista veckorna var vi som mest produktiva, vilket syntes på antalet totala poäng gruppen avklarade, vilket kan ha berott på att vi var klara med de mest tidskrävande delarna av projektet, men även pågrund av att vi hade ackumulerat en större förståelse om hur bl.a. React, Jest och Material UI funkade.

(B). Vi vill ha en jämn nivå av produktivitet varje vecka där vi avslutar majoriteten av våra user stories. Vi hade satt upp vissa mål(user stories) som vår applikation inte uppfyllde mot slutet av projektet, men tog beslutet att inte utföra dem då det hade krävts att vi skulle ha skrivit en server. Vi tog beslutet att inte göra en user story som involverade en server eftersom det då hade tagit för lång tid för majoriteten av gruppmedlemmarna att bekanta sig hur man jobbar med servrar och det i sin tur hade medfört en sänkt velocity. Detsamma gäller testningen. Vi ville kunna leverera mer funktionalitet och inte spendera mer tid än nödvändigt på att skriva tester.

(A → B). Då vi fick problem halvvägs in i projektet när vi ökade antalet user stories så kunde vi istället ha färre, viktigare user stories. Det skulle bibehållit vår produktivitet och inte sänk vår velocity.

Eftersom testerna var en stor anledning varför vår effektivitet sänktes under en sprint, så hade en lösning kunnat vara att färre personer i gruppen fokuserade på att lära sig och förstå testning i React, och därefter sprida vidare kunskapen till de övriga medlemmarna i gruppen, då testningen fortfarande är en stor och viktig del utav ett fullt fungerande projekt. En annan lösning på problemet hade kunnat vara att vi förberett oss bättre genom att studera testning i början av kursen så att man lär sig det mer successivt under veckorna, vilket skulle leda till en bättre fördelning av inlärning och en förminskning av stress.

Vad gällandes user stories som inte fullföljdes, hade man kunnat införa dem löpandes under tidens gång, då man har en större kontroll över vad tiden räcker till. Samtidigt kan man ha kvar dessa user stories som mål att sträva efter, vid mån av tid.

4 DESIGN DECISIONS AND PRODUCT STRUCTURE

4.1 Design Decisions and customer value

(A). I detta projekt har vi valt oss att använda det Javascript biblioteket React[7] och material UI[5] vilket bygger på React. React är ett bibliotek skapat för att enkelt kunna designa interaktiva webbapplikationer, och material UI ett bibliotek för som innehåller en del färdiga UI komponenter som är byggda med hjälp av React.

Vi valde dessa bibliotek främst med kunden i åtanke då det hade tagit längre tid att skapa hemsida utan dem och därmed blivit dyrare för kunden. Båda biblioteken är populära och är aktivt underhållna vilket innebär att kunden inte kommer behöva oroa sig för att utvecklingen på dem upphör.

(B). Vad för typ av designmönster, bibliotek, verktyg, etc man använder sig är väldigt beroende på projektet man håller på med. Det är dock väldigt viktigt att lägga tid på att diskutera med både utvecklare och kunden vad som passar bäst projektet bäst. Det man väljer kan mycket väl hänga med under hela projektets livslängd, och om det man väljer visar sig vara ett dåligt val kan väldigt mycket kod behövas skriva om.

(A → B). Ha bra förståelse för hela projektet och vad som kan behövas, och spendera mycket tid på att hitta de rätta verktygen.

4.2 Technical Documentation

(A). Vi började arbeta utan en formell teknisk dokumentation och hade en diskussion om det halvvägs in i projektet. Vi kom fram till att det fanns större behov till att fortsätta lägga tid på de språk, verktyg och bibliotek vi använde. Att påbörja teknisk dokumentation halvvägs in i projektet kändes svårt och tidskrävande, vilket var något vi inte hade tid med då. Vi visste också att vi var de enda personerna som skulle jobba med projektet och beslöt oss då för att skippa det.

(B). Att använda teknisk dokumentation är något vi har nu insett är mycket värdefullt även om projektet är "litet nog" för utvecklarna att ha bra koll på i huvudet. För kunden så är detta enormt värdefullt, speciellt för att förstå de olika huvudena i projektet, och även för andra framtida utvecklare som möjligtvis fortsätter på projektet. Utan teknisk dokumentation blir det mycket svårare att underhålla. Lite teknisk dokumentation är mycket bättre än ingen alls.

(A → B). Man ska börja med dokumentationen så tidigt som möjligt. Det är också bra att undersöka vad för olika typer av teknisk dokumentation som funkar bäst för det projekt man arbetar med. Skriver man objektorienterad kod så kan **UML** diagram vara intressanta, skriver man imperativ kod kanske flödesdiagram passar bättre. Vissa språk har även verktyg för att hjälpa till med skapande av dokumentation.

4.3 How documentation is updated

(A). Om man bortser från kommentarer i koden så har vi inte uppdaterat någon teknisk dokumentation

(B). Dokumentationen ska uppdateras iterativt, i samband med att projektet utvecklas. På så sätt minskas arbetet för sig själv och man glömmer inte detaljer. Detta kan man göra på olika t.ex. att man uppdaterar dokumentationen i samband med att man gör user stories, eller i samband med slutet på sprinten.

(A → B). Man bestämmer som ett team att dokumentationen uppdateras iterativt. Detta tror vi bäst och enklast görs genom att lägga till det som ett krav i DoD.

4.4 Code quality and enforcing coding standards

(A). Vi har använt oss av en funktion i Github där vi låser vår **master branch** för direkta **commits**. Detta har varit väldigt användbart för att förhindra dålig kod att förekomma, utan att andra märker det. Vi arbetar istället med våra user stories i egna branches. När en user story betraktas vara färdig av dem som jobbat med den har en **pull requests** till master branchen skapats, där andra medlemmar i gruppen kan granska den nya koden och sen godkänna den ifall allt ser bra ut. Det krävs godkännandet av två medlemmar i gruppen, för att personen som skapade pull requesten, ska tillåtas att **merga** in till master.

(B). Att alla i gruppen håller sig till bestämda designmönster är väldigt användbart då det gör hela kodbasen mycket mera konsistent och enklare att upprätthålla. Att alla använder sig av samma formatteringsstansard är också viktigt så man inte håller på och formatterar om varandras kod varje commit.

(A → B). Innan man börjar projektet så bestämmer gruppmedlemmarna hur de vill strukturera upp koden, t.ex. om man kanske vill att varje nytt objekt kommunicerar med andra objekt genom interfaces. Man får också bestämma sig av en specifik kodformatterare, och ladda upp formatterarinställningar till Github så koden blir konsistent. Detta kommer förebygga att man formatterar andras kod.

5 APPLICATION OF SCRUM

5.1 The roles within our team and the impact it had

(A). I projektet har vi inte haft någon uttalad **produktägare**. Det är en uppgift vi alla har tagit på oss och beslut har tagits gemensamt under möten. Avsaknaden av en dedikerad produktägare att gå till har gjort vårt övriga arbete långsammare då vi alla har behövt diskutera kraven istället för att en produktägare snabbt har kunnat förtydliga krav när det behövts.

I början pratade vi om att utse en **scrum master**, men då vi bara kunde läsa om vad en sådan skulle göra och inte visste i praktiken hur arbetsrollen fungerar, är även detta en roll vi gemensamt har tagit på oss. Detta ledde till vi inte hade någon att vända oss till om något blockerade framsteg. Istället tog alla det ansvaret och ett möte behövde organiseras varje gång vi hade problem.

Utvecklarens roll i projektet är att utföra arbetet enligt sprinten. Samt att bryta ner stories till tasks. Även att estimera effort av stories. Utvecklararna bygger upp sprint backloggen baserat på målet av sprinten och velocityn dom uppskattar klara.

(B). Det är absolut önskvärt att ha en produktägare i framtiden som man kan gå till med frågor om kraven för att förtydliga dem. Produktägaren skulle även vara användbar att ha till att prioritera produktbackloggen istället för att ha långa möten varje vecka när alla var produktägare.

Självklart ska ett projekt som använder sig av scrum ha en scrum master. En person som är duktig på scrum och kan identifiera och lösa saker som blockerar teamets arbete. Scrum mastern kan även organisera och ta upp problem med arbetssättet under retrospective mötet. Detta är något som förhoppningsvis kommer finnas i alla framtida projekt där vi jobbar med scrum.

(A → B). I nästa projekt ser vi till att ha en utsedd produktägare som har koll på målet med applikationen och sprintarna som kan förtydliga kraven när det behövs så vi slipper våra långa möten där alla är den som bestämmer.

Det som är viktigt för oss nu som utvecklare, är att ta med det agila arbetssättet och scrum till nästa projekt. För att inte ha ett metodiskt arbetssätt leder till ineffektivt och rörigt arbete.

Alla i grupp Endor har genom denna kurs fått gedigna kunskaper i området och skulle kunna ta på sig rollen som utvecklare i ett scrum team.

5.2 The agile practices we used and their impact

(A). I detta projekt har vi försökt använda så många agila aspekter som möjligt. Vi har bland annat använt oss av parprogrammering vilket har hjälpt oss att snabbare komma igång med webbutveckling, som har varit helt nytt för nästan alla medlemmar.

Vi använde oss utav user stories för att skriva ner specifikationen av applikationen. User stories ger oss ett systematiskt arbetsätt där alla på ett tydligt sätt ser vad som behöver göras.

Då scrum var den agila metoden som läraren kunde ge handledning i blev det vårt val. Ifrån scrum så använde vi sprints, sprint reviews, sprint planning, daily scrums och sprint retrospective. Vi organiserade tiden i sprints som började med sprint planning och avslutades för det mesta med en sprint review eftersom vi inte höll retrospectives varje sprint. Sprint planning var användbart så alla i teamet visste vid början av varje sprint vad alla som skulle göras under sprinten. Våra sprint reviews hjälpte teamet att vara synkade så alla visste vad som hade gjorts under sprinten. Våra **daily scrum** möten (som inte var varje dag) gav möjlighet till medlemmar som fastnat med något att få hjälp av andra medlemmar, vilket ökade produktiviteten. När retrospectives hölls så hjälpte det oss att förbättra vårt arbetssätt, t.ex. så ändrade vi längd på sprinten vilket var användbart. Problemet med våra retrospectives var att vi inte alltid hade dom vid slutet av varje sprint. Ibland hade vi diskussioner mitt i veckan som snarare hade passat som en retrospective.

(B). Avse ett möte speciellt för sprint retrospective. Detta var något vi försökte med men det blev inte riktigt av i slutändan, och istället så blandades det in med sprint reviewn. Genom att utse en speciell tid då man bara reflekterar över processen vi använde under sprinten och inget annat, så kan man snabbare och bättre förbättra framtida sprints. Förutom att retrospectives inte blev av så fungerade arbetssättet väl.

(A → B). Bestäm en specifik tid för sprint retrospective. Att ha en schemalagd tid skulle hjälpa med att se till att mötet blir av.

5.3 Our sprint review and the relation to our scope and customer value

(A). I vårt fall utsåg vi inte en speciell produktägare, istället så hade alla medlemmar en liten roll i det. Vi började vår sprint review med att gå igenom alla user stories som var tilldelade under sprinten och lät varje utvecklare berätta om hur det hade gått. Sprint reviewn ledde t.ex. till att utöka funktionaliteten av vårt water usage table genom att skriva en ny user story. Varje user story ansågs officiellt avklarad när de andra medlemmarna var säkra på att det följde vår DoD som innehöll saker som t.ex. koden ska vara testad, mergad och reviewad av minst två andra utvecklare.

Vår roll som produktägare framkom som mest under sprint planeringen, det var då vi bestämde oss för vad som vad som skulle göras beroende på vad som var viktigast för användaren. Till exempel så började vi med en user story som skrev ut vattendata i en tabell på hemsidan. Interaktivitet och grafiken kom i senare user stories då det inte var en prioritet i början.

(B). Att utse en person som kan ta sig an rollen som produktägare kan göra det tydligare, vad exakt som har högre prioritering, jämfört med att alla medlemmar försöker bestämma vad som ska prioriteras, vilket kan vara svårt då vissa medlemmar kan tycka olika. Att utse en produktägare är även användbart för att ha någon som entydigt kan svara på frågor om att förtydliga specifikationer.

(A → B). Utse en person till att vara produktägare före projektets gång som ansvarar för målet av applikationen. Produktägaren ska också vara den som kan förtydliga specifikationerna eftersom det är deras ansvar vad målet med applikationen ska vara.

5.4 Best practices for learning and using new tools and technologies

(A). Det var mycket nytt att lära då endast en av oss hade tidigare erfarenhet av att göra webbapplikationer men det löste vi genom att gruppmedlemmen fick hålla en genomgång på en timme som

var mycket uppskattad. Efter genomgången hade vi tillräckligt med kött på benen att börja försöka själva. Nästa stora hinder var när vi började med tester. Tester har vi gjort innan, men tester fungerar annorlunda när det är design som skall testas och inte funktioner. När vi lade till i definition of done att all kod skulle vara testad tänkte vi att det här skall inte vara något problem, men vi alla fastnade och ingen lyckades klara av sina tasks på grund av detta. En vecka gick där alla satt på sitt håll och försökte komma på hur man gör. Visst vi hade möten där vi konstaterade att vi satt fast, men vi kom inte vidare. Detta var något vi frågade om på handledningsmötet och vi fick hjälp hur vi skulle göra. Lösningen var att det inte ger någon kundnytta att vi inte löser våra tasks, så vi skall minska ner på våra tester.

(B). Vi alla vet hur man lär sig nya saker, men i detta fallet handlar det om en kurs i agil utveckling och inte om webbapplikationer så vi vill lära oss agil utveckling från grunden och upp. När vi lärt oss om webbaplikationer har vi mer försökt lära oss hur man gör och inte lika mycket hur allt fungerar.

(A → B). Det finns många sätt att få ny kunskap några bättre än andra. Det fungerar att bara bestämma sig för vad man vill lära sig och sedan försöka tills man fastnar och då lösa det problemet för att gå vidare. Men det är inget hållbart sätt få en ingående kunskap i ämnet. Vi som har programmeringskunskap behöver inte lära oss grundläggande Javascript utan kan hoppa över dom bitarna och gå direkt på de språkspecifika delarna för att snabbare komma in i ett nytt språk.

5.5 Relation to literature and guest lectures

(A). Boken 'scrum and xp from the trenches' [4] är lättläst och hjälper till förståelsen inför ämnet och genom att efterlikna hur de gör i boken, har vi snabbt förstått hur vi skall göra t.ex sprint retrospective. Vi har även haft nytta av de andra länkarna, speciellt de som handlar om hur man jobbar hemifrån. Detta är då något som lätt gör att man får dåliga vanor om man inte har disciplin och kan styra upp sin arbetsdag. Någon gästföreläsning har vi inte haft.

(B). Att ha en sprint retrospective är viktig för utvecklingen av gruppens användning av scrum.

(A → B). Våra sprint retrospectives var inte så strukturerade och det var lätt att vilja hoppa över dom. Samtidigt var ämnena vi tog upp självklara, för att de handlar om problem som har identifierats tidigare under veckan. När man tittar i boken står det att det är många som hoppar över sprint retrospectives för de tycker att det är onödigt att lägga tid på när de redan är stressade till att producera kod.

6 SAMMANFATTNING

I denna projektkurs har vi haft en större frihet än vad vi brukar ha i andra kurser. Vi har haft ett ramverk som vi måste följa men utöver det kunde vi göra mer eller mindre vad vi ville. Friheten kan dock leda till att mer fokus läggs på produkten än arbetssättet, vilket inte var kursens huvudsyfte. Vi valde att göra en webbapplikation, dels för att vi var nyfikna på vad det var eftersom vi inte har gjort något sådant innan, men också även för att koden vi skriver snabbt genererar synligt resultat. Detta underlättade när vi pratat kundnytta och för att identifiera potentiella förbättringar.

Det agila arbetssättet har gjort att alla i gruppen kunde vara delaktiga då det är lätt att hjälpa till ifall någon halkar efter, då det oftast finns någon i gruppen som har svar på frågorna som ställs i stunden. På samma sätt syns det ifall någon behöver större utmaningar så att de inte blir uttråkade.

Det har varit en underhållande kurs som utvecklat våra agila kunskaper och format oss till bättre programmerare.

REFERENCES

[1] Discord. *Discord software*. 2020. URL: https://discord.com/ (visited on 10/28/2020).
[2] Github. URL: https://github.com.
[3] Atlassin Jira. *Jira Software*. 2020. URL: https://www.atlassian.com/software/jira.
[4] Henrik Kniberg. *Scrum and XP from the trenches*. URL: http://www.infoq.com/minibooks/scrum-xp-from-the-trenches-2.
[5] MaterialUI. *MaterialUI*. URL: https://material-ui.com/.
[6] United Nations. *THE 17 GOALS*. 2020. URL: https://sdgs.un.org/goals (visited on 10/22/2020).
[7] React. *React*. URL: https://reactjs.org/.
[8] React. *React testing*. URL: https://reactjs.org/docs/testing-recipes.html#snapshot-testing.