Timeline

- Post Mortem Rapport

Victorious Secret - DAT255

David Hallén

Karl Lennartsson

Piotri Neto

Victor Nydén

Ludvig Sylvén

Innehållsförteckning

1. Introduktion

Hela världen rör sig mot ett allt större beroende av teknik. Än så länge har teknikutveckling i termer av bland annat digitalisering och ständig uppkoppling bara skrapat på ytan. I vår utveckling mot ett digitaliserat och ständigt uppkopplat samhälle får vi inte glömma bort att lära från det förflutna. Det är genom att inte göra om forna misstag som vårt samhälle utvecklas framåt. Det är genom kunskaper om historia som vi förstår nuet. Vi ska därför ständigt lära av vår historia, och utmana varandra i vem som kan den bäst.

1.1 Bakgrund

PISA-resultat och OECD-rapporter talar sitt tydliga språk. De svenska studieresultaten blir allt sämre. En bidragande förklaring är samhällets teknikutveckling. Hur motiveras en tonåring att studera när han eller hon på några sekunder kan hitta alla svar med några fingerklick på sin smartphone? Varför lägga tid på att lära sig när olika händelser ägde rum, när det endast tar några sekunder att hitta rätt svar på exempelvis Google? Allmänbildningen anses av vissa vara död. Men det menar vi är felaktigt. Allmänbildning är inte död, den har snarare tagit en ny form som är anpassad till vår moderna era. Kunskap handlar inte längre om att känna till specifika årtal eftersom många har tillgång till Google. Men för att använda Google, måste kunskap finnas om vilka sökord som ska användas samt hur sökresultaten ska tolkas. Det är viktigt att kunna sätta olika skeenden i relation till varandra. Det är viktigt att på en tidslinje kunna placera ut olika händelser före, eller efter varandra, eftersom det skapar en känsla för sammanhanget och en således en större förståelse som ej nås genom att enbart studera specifika årtal.

1.2 Gruppens bakgrund och inlärningsprocess

Alla gruppmedlemmar kände varandra sedan tidigare eftersom alla i tre års tid har gått i samma klass. Ingen i gruppen hade tidigare erfarenhet av att utveckla mobilapplikationer (härefter appar), eller annan typ av större programmeringsprojekt. Därför hade alla medlemmar i gruppen behov av att lära sig vissa verktyg som senare användes under projektet. Alla gruppmedlemmar hade dock erfarenhet av Java, XML och SQL.

Ett av verktygen som använts var Git. Några i gruppen använde sig av ett inlärningsverktyg för Git som kallas för *Learn Git Branching.* Det är ett visuellt och interaktivt inlärningsverktyg som syftar till att lära sig att förgrena koden samt slå ihop grenar i Git. De gruppmedlemmar som hade använt sig av Learn Git Branching ansåg inte att detta hjälpte dem nämnvärt för att kunna använda sig av GitHub. Det visade sig inte vara särskilt svårt att lära sig att hantera Git utan några guider. Istället tillämpades metoden *learning by doing*.

Trots att alla gruppmedlemmar hade tidigare erfarenhet av de båda språken Java, XML och SQL fanns det inte någon erfarenhet av hur dessa skulle hanteras tillsammans. För att lära sig grunderna inom Android-programmering använde sig samtliga av diverse undervisningsvideor från exempelvis YouTube. Även olika forum på Internet har använts i samma syfte. Det främsta exemplet på detta är forumet Stack Overflow där frågor och svar kan skrivas av medlemmar och visas sedan öppet för allmänheten. Dessa hjälpmedel har

också använts för att hantera mer avancerade tekniska svårigheter så som att bygga upp en databas för lagring och hämtning av frågor till spelet.

1.3 Vision

Gruppens vision har varit att skapa ett spel som både är underhållande och lärorikt. Spelet ska för användaren vara utmanande och engagerande, och användare ska också kunna mäta sin prestation samt ha möjligheten att utmana andra. Användaren ska med spelet kunna lära sig om hur vitt skilda händelser relaterar till varandra i den så kallade fjärde dimensionen: tid. Visionen har också varit att skapa ett spel som kan användas i utbildningsyfte och därför ska lärare samt andra intressenter få möjlighet att skapa sina egna frågor i detta syfte.

2. Metod

I detta kapitel kommer metoden bakom projektet att presenteras, och arbetsprocessen beskrivas.

2.1 Val av projekt

Idéfasen av projektet påbörjades snabbt då flera av oss redan bar på idéer innan projektets start. Därför ägnades de första veckorna åt utvärdering av befintliga och en del nya idéer. Gruppen genomförde en typ av långsiktig *stop and go brainstorming*, vilket innebar att gruppen sågs var och varannan dag, både formellt och informellt, för att diskutera idéer och däremellan fick var och en göra sina egna reflektioner. Tillslut sållades antalet idéer ner till tre: en app för att använda låtar från *Spotify* som ringsignal (idé 1), en app för att sortera aktier utefter av användaren valda sökkriterier (idé 2) och en app i form av ett spel som går ut på att lägga ut historiska händelser i kronologisk ordning (idé 3). Gruppen valde att förverkliga idé 3, och appen kom att kallas för Timeline.

I valet mellan de tre ovan beskrivna idéerna utvärderades framförallt gruppens tekniska möjligheter att skapa de olika apparna. Både idé 1 och 2 skulle kräva användning av API:er som gruppen saknade kunskap om. För idé 2 skulle det krävas tillgång till realtids börsdata vilket skapade en osäkerhet kring huruvida denna typ av tjänst skulle vara dyr eller kräva mycket kontakt med dess leverantör. För idé 1 fanns en hel del osäkerheten kring Spotifys API och huruvida appen överhuvudtaget var tillåten ur upphovsrättslig synpunkt. Ur teknisk synvinkel bedömdes idé 3 vara det bästa valet, då allting kunde skapas på egen hand utan beroende av några externa aktörer. Då Timeline dessutom bedömdes som den idé som med störst sannolikhet skulle gå att färdigställa med ett gott resultat på utsatt tid, behövdes ingen vidare valprocess. Ytterligare en anledning till att Timeline valdes, var att alla gruppmedlemmar trodde på idén och kände engagemang för den.

2.2 Sprintboardverktyg

Projektet genomfördes i enhetlighet med *Scrum*-metodik. Således användes en *product backlog* för att dokumentera vad som skulle göras. Initialt användes Microsoft Excel för att kategorisera *user stories* och för att planera för respektive *sprint*. Excelbladen delades mellan gruppmedlemmarna via Github. Relativt tidigt insågs dock det omständliga i att bladen ständigt behövde laddas ner, ändras och sedan laddas upp igen. Till följd av denna problematik övergicks till att arbeta i ett kalkylblad i Google Drive istället, vilket fungerade väl under resterande delar av projektet.

I sprintboarden sorterades user stories*,* mer övergripande *epics* samt mer abstrakta *fuzzy stories.* User stories rangordnades efter prioriteringsordning och den *effort* (ansträngning) respektive user story förväntades kräva uppskattades. Arbetsbelastningen per sprint baserades på hur stor total effort som lyckades färdigställas i föregående sprint.

2.3 Projektets gång

Projektet delades upp i fem olika sprints som alla var en vecka långa. Nedan redovisas vilka uppgifter som utfördes under respektive sprint.

2.3.1 Sprint 1

Projektets första sprint sågs som en inlärningsperiod med två mål. Det första målet var att få igång utvecklingsmiljön. Det andra målet var att snabbt få ut en första version av appen, för att visualisera konceptet och enklare se vad som behövdes göras, samt dela upp arbetet.

Under denna sprint användes tekniken *pair programming* och arbetet utgick från user stories. Tasks användes inte, men arbetet var ändå mycket effektivt. Vad som gick sämre i den första sprinten var sammanslagningen av kod, till stor del till följd av att ännu ej fungerande kod synkroniserades. Till följd av detta lades i följande sprints större fokus på kvalitetskontroll av kod innan synkronisering.

2.3.2 Sprint 2

Målet för nästa sprint var att utveckla en fungerande databas och en typ av poängfunktion. I denna sprint valde gruppen att arbeta med tasks. Detta bedömdes dock ej bidra till ökad effektivitet. De user stories som fanns var nämligen inte tillräckligt omfattande för att de på ett givande sätt skulle kunna brytas ned i vettiga tasks. Till följd av att det ständigt fanns dialog mellan gruppmedlemmarna bedömdes det fungera minst lika bra att enbart använda tydliga user stories.

Som nämnts ovan antogs från och med denna sprint ett mer noggrant förhållningssätt till kvalitetssäkring av kod innan den synkroniserades. Tekniken pair programming ansågs vara för omständlig, eftersom gruppens medlemmar trivs med att programmera på olika tider och platser samt att det var svårt att finna tider då alla kunde programmera tillsammans. Därför användes pair programming allt mindre och arbetet övergick till att mer ha formen av *pair negotiation,* vilket innebär att två personer blev ansvariga för en viss uppgift eller user story och samarbetade för att lösa denna.

2.3.3 Sprint 3

Den tredje sprinten syftade till att fortsätta utveckla funktioner kopplat till databasen, och utveckla spellogiken. Eftersom arbetet med tasks visade sig vara överflödigt i föregående sprint, valdes i denna sprint att återgå till att endast använda user stories. På grund av olyckliga schemakrockar gruppmedlemmarna emellan färdigställdes endast tre av fem user stories denna sprint.

2.3.4 Sprint 4

I sprint fyra lades mest tid på att refaktorera och kommentera koden, samt förbättra användarupplevelsen. Logiken i programmet effektiviserades. Eftersom arbetssättet med

user stories och pair negotiation visat sig framgångsrikt i tidigare sprints användes samma metodik även i denna sprint.

2.3.5 Sprint 5

Fokus i denna sprint lades på att addera funktionalitet för att tillåta för användare att lägga till och ta bort egenskrivna frågor. Det grafiska användargränssnittet färdigställdes och interaktionen mellan app och användare förbättrades. Eftersom det var den sista sprinten lades mycket tid på att sitta gemensamt i gruppen.

2.4 Testning av applikation

Då projektets medlemmar simultant arbetade med att utveckla olika typer av funktionalitet som därefter sammanfogades och synkroniserades var behovet av att kontinuerligt testa koden stort. Denna testning har under större delen av projekttiden bestått av manuell icke standardiserad testning genom användargränssnittet. Med icke standardiserad testning menas att de test som genomförts ej varit nedskrivna. Istället har gruppmedlemmarna på ett mer spontant vis sökt utvärdera koden genom att manuellt testa olika möjliga situationer i spelet. För att detta inte enbart skulle vara något som gjordes av tvång, strävades efter att nya frågor kontinuerligt skulle läggas till i spelet, för att på detta vis göra testprocess mer underhållande.

Under arbetets gång har dessutom stor vikt lagts vid att låta utomstående personer testa spelet. Det är lätt att man som utvecklare blir blind för vissa typer av fel och oklarheter i programvarans utformning, vilka personer utan insyn i utvecklingsprocessen lättare ser. Att dessutom inte enbart lyssna på dessa personers feedback utan att även faktiskt titta på när de spelar har varit viktigt i utvecklingen av en mer användarvänlig produkt. Ett exempel som kan nämnas är att scrollfunktionen på tidslinjen i spelet, tidig i utvecklingen, var otydlig för användare. Detta eftersom korten hade en sådan storlek att fyra stycken kort precis fyllde hela skärmen, vilket i sin tur gjorde det oklart att det faktiskt gick att scrolla. Detta löstes genom att storleken på korten ändrades, så att endast delar av korten på kanten av tidslinjen syntes, vilket gjorde scrollfunktionen mer uppenbar.

För att kunna genomföra den typ av testning som beskrivs ovan var det viktigt att tidigt i processen ha en fungerande produkt med ett grafiskt gränssnitt. I projektet kunde denna tidiga produkt färdigställas redan under den första sprinten, vilket således var positivt ur testningssynvinkel.

Explicit testdriven utveckling har inte använts under projektet. I projektet har dock mycket fokus legat på att bygga in olika former av test i koden för att minska risken för felaktigheter. Under sista delen av projektet har även standardiserade manuella test tagits fram. Dessa består av protokoll med beskrivningar av vad testaren ska göra under respektive teststeg tillsammans med en beskrivning av vad teststeget bör resultera i för konsekvens. Testaren jämför sedan det verkliga resultatet med det förväntade, och om dessa överensstämmer är testet lyckat. Behovet av denna typ av standardiserade test upplevdes som större under slutet av projektet då det med liten tid kvar av projektet upplevdes som att risken var större för att alla möjligheter ej skulle hinna testas genom mer spontan testning. De standardiserade *test-suits* som skrivits har fokuserats på att uppnå hög *bransch-* och

statement-täckning, dvs. täckning av alla olika möjliga utfall och vägarna till dessa. Då applikationen innehåller relativt stora mängder kod, fastslogs att det skulle vara tidskrävande att skriva utförliga test för all funktionalitet. Därför fokuserades dessa standardiserade test på områden som tidigare haft en del problem och där det är lätt att missa att testa vissa typer av förutsättningar genom spontan testning.

2.5 Riskhantering

Eftersom projektet haft en relativt kort tidsram fanns det ett antal risker som behövde hanteras för att säkerställa att en färdig produkt skulle kunna presenteras vid projektets slut. Först och främst arbetade gruppen intensivt i början av projektet. Det gjorde att det tidigt framkom en bild av vad som behövde göras, och hur lång tid det skulle ta, för att bli klara i tid. Detta gjorde att risken för att arbetet skulle bli allt för intensivt i slutet av projektet minskade.

Vidare var samtliga gruppmedlemmar ej tidigare bekanta med de program som användes i projektet. Till följd av detta fanns det en naturlig osäkerhet kring hur programmen skulle användas mest effektivt. Gruppen valde att inte arbeta med programmens mer avancerade och komplicerade funktioner, utan höll sig till basfunktionerna. Detta medförde att arbetet förmodligen blev något mindre effektivt än vad det potentiellt skulle kunna ha varit, men samtidigt kunde risken för att felaktig användning av programmen skulle försena processen minskas.

När applikationen fungerade enligt de kravspecifikationer som ansågs viktigast, valde gruppen att inte lägga till några fler avancerade funktioner. Istället lades fokus på att förbättra och effektivisera befintliga funktioner. På så vis kunde risken minskas för att appen inte skulle bli färdig i tid.

Som beskrivits i avsnitt 2.1 fanns det till en början idéer som byggde på att utveckla appar som skulle bero på externa aktörer. Då gruppen såg risker i att dessa typer av samarbeten skulle kräva extra tid och eventuellt pengar, valde gruppen att utveckla en app utan externa beroenden.

2.6 Gruppdynamik och roller

Gruppen valde att inte tilldela medlemmarna specifika uppgifter och ansvar. Istället hade alla medlemmar i början av projektet ett övergripande ansvar för allt. Allteftersom projektet började ta form och arbetet delades upp i sprints som i sin tur bestod av user stories blev en eller flera gruppmedlemmar ansvariga för att lösa specifika user stories. När en person redan hade börjat arbeta med exempelvis databasen och därmed redan visste hur den fungerade, blev det en naturlig följd att den personen fortsatte att arbeta med databasen. På så vis utmejslades rollfördelningen utifrån kunskap och intresse, se tabell 1.

En tanke med detta var att alla gruppmedlemmar skulle känna ansvar för helheten. En risk med en för tydlig rollfördelning är också uppgiftsområden som inte kunde förutses på ett tidigt stadium i ett projekt hamnar mellan stolarna och att ingen tar ansvar för dessa. Då ingen av gruppmedlemmarna hade tidigare erfarenhet av liknande projekt bedömdes denna risk som relativt hög.

Tabell 1: Antal timmar och andel av den totala tiden som ägnats till olika steg i projektet.

	Planning	***Coding***	***Documenting***	***Refractoring***	***Helping teammembers***	***Total***
Total Time (h)	**100**	**200**	**160**	**120**	**50**	**630**
Piotri Neto (%)	**20**	**25**	**5**	**35**	**15**	**100**
Karl Lennartsson (%)	**20**	**30**	**10**	**30**	**10**	**100**
David Hallén (%)	**20**	**15**	**30**	**10**	**25**	**100**
Ludvig Sylvén (%)	**20**	**15**	**25**	**15**	**25**	**100**
Victor Nydén (%)	**20**	**15**	**30**	**10**	**25**	**100**
	100	**100**	**100**	**100**	**100**	

Som nämns i avsnitt 2.3.2 Sprint 2 var det svårt att samordna tider för att hela gruppen kunde arbeta tillsammans under längre stunder. Försök gjordes för att överbrygga dessa problem genom att ha fler kortare möten tillsammans. En risk med att utveckla kod på olika platser är att lösningar på problem utvecklas som inte är kompatibla till den kod som resten av gruppen utvecklar. För att hantera denna risk strävade gruppen efter att i så hög utsträckning som möjligt synkronisera med GitHub så fort en funktionalitet var färdigutvecklad. Detta gjorde att övriga medlemmar kunde se vad som utvecklats och anpassa sig efter det, samtidigt som man själv såg vad övriga utvecklat och man kan anpassa sig efter vad de har utvecklat.

3. Resultat

I detta kapitel presenteras hur gruppen anser att den nuvarande funktionaliteten förhåller sig till den vision som fanns i ett tidigt stadium av projektet och hur denna vision har förändrats, samt vad detta skulle få för funktionella/tekniska konsekvenser. Vidare följer en reflektion över vilka metoder som har tjänat gruppen väl samt vad som borde ha gjorts annorlunda.

3.1 Återkoppling till vision

I kapitel 1 förklaras att visionen i ett tidigt stadium av projektet var att skapa ett underhållande och givande multiplayerspel, möjligt att anpassa efter intresse och behov. Denna vision kan anses vara uppfylld. Multiplayerfunktionen innebär i dagsläget att det går att spela flera spelare mot varandra på samma enhet. En vidareutveckling av mulitplayerfunktionen skulle vara att göra det möjligt att spela mot varandra på olika enheter. Fördelen med den befintliga lösning för multiplayerspel är att den möjliggör spel även då internetuppkoppling saknas eftersom den endast använder sig av lokal data. Det nämns ovan i avsnitt 2.7 Riskhantering att anledningen till varför vi valde att börja utveckla en lokal version var att säkerställa att en komplett app och inte riskera att en mer avancerad teknisk lösning inte skulle blivit färdig.

Det beskrivs även i avsnitt 1.1 Vision att spelet skulle kunna anpassas efter användarens intresse och behov. Även denna vision kan anses vara uppfylld då det är möjligt att välja att spela fördefinierade kategorier inom vilka det finns på förhand inlagda frågor. Det är också möjligt att själv lägga in de frågor och kategorier som man tycker är intressanta. Dessutom ges möjlighet till att ta bort de frågor som användaren av någon anledning inte längre vill ha kvar. En annan del av visionen var att kunna mäta sin prestation. Det är möjligt att mäta sin prestation i den mån att användaren har möjlighet att spara sina resultat i en highscorelista i vilken användaren kan se de fyra bästa resultat som sparats på enheten.

Under projektets gång upptäcktes flera områden som var önskvärda att utveckla vidare. Dessa områden var svåra att se från början, utan har utvecklats allteftersom gruppen har arbetat tillsammans och diskuterat möjliga applikationsområden och därefter tagit fram nya tankar om vidare utveckling av appens funktionalitet

En av de ursprungliga visionerna var att man som användare skulle ha nytta av appen vid inlärning, exempelvis via skola eller utbildning. En tanke var då att studenter tillsammans med lärare skulle ges möjlighet att skapa egna frågor för att användaren skulle få chansen att lära sig just det som är intressant. Ett problem med nuvarande lösning, att nya frågor och kategorier endast kan skrivas lokalt är att man förlorar chansen till skalfördelar som exempelvis en lärare kan ge till en klass. Därför vore det önskvärt om en exempelvis en lärare ges möjlighet att skriva frågor som sedan kan importeras automatiskt till studenternas enheter.

På samma vis som ett spel som kan spelas över flera olika enheter skulle en highscorelista som underlättar jämförelse av resultat vänner emellan öka känslan av att man utmanar varandra. Vidare hade det också varit önskvärt att ha utöka med en funktion som på ett mer

överskådligt vis skulle göra det möjligt att jämföra ens egna resultat över tid, exempelvis att användaren kan följa hur resultaten har utvecklat sig över tid. På så vis skulle användaren få feedback och lättare identifiera eventuella kunskapsframsteg.

Även om det vore önskvärt om flera funktioner läggs till, anser gruppen att det redan nu finns en nytta i den nuvarande applikationen som användare kan vara intresserade av att ta del av. Dock är applikationen inte redo för publicering då Google Plays kvalitetskriterier inte ännu uppfylls och en publiceringsprocedur måste först sammanställas.

3.2 Utvärdering av Tekniker

Detta avsnitt avser att utvärdera två av de tekniker som genomsyrat projektet, nämligen pair programming och Scrum.

3.2.1 Pair programming

Initialt var metodiken pair programming användbar för att få alla gruppmedlemmar delaktiga i arbetet. Då respektive gruppmedlem gick in i projektet med olika styrkor och förkunskaper, möjliggjorde pair programming för att alla snabbt kunde lära sig av varandra och att ingen fastnaden i basala problem så som användande av Android Studio och GitHub. I övrigt var pair programming användbart vid svårare och mer komplexa uppgifter som krävde mycket kunskapsinhämtning, exempelvis skapandet av den interna databasen. Det var ofta lättare för två personer att hitta sätt att lösa ett specifikt problem eftersom personerna kunde se problemet från skilda synvinklar. Pair programming gav också upphov till ökad *brainstorming,* vilket var användbart vid design av gränsnitt och gameplay.

Den största nackdelen med pair programming som gruppen upplevde var det behov av logistikupplägg som det medförde. Visserligen hölls veckovisa möten, men då alla i gruppen har haft mycket oregelbundna scheman har gemensam programmering inte alltid varit möjligt. För triviala uppgifter märktes ingen särskild skillnad i tidsåtgång eller kvalitet då pair programming användes jämfört med när programmerandet skedde på egen hand. I samband med exempelvis handledarmöten var tekniken dock effektiv, då gruppen redan var samlad.

Pair programming användes i viss utsträckning under hela processen men dock i allt mindre grad efter sprint 1. Att inte använda metodiken gav stora produktivitetsfördelar eftersom alla gruppmedlemmar kunde jobba åtskilt när det passade bäst. Trots att pair programming inte användes kontinuerligt har pair negotiation använts i stor utsträckning. Detta upplevdes som en bra kompromiss för att bibehålla fördelen i att problem angrips från olika synvinklar, samtidigt som inte samma behov av att alltid sitta tillsammans fanns.

3.2.2 Scrum

Då man har en produktägare som sätter krav på specifikationerna för applikationen är fördelen att man alltid får mål givna som gör att man kan arbeta lösningsorienterat än funktionsorienterat. Gruppen har därför ägnat sig åt att lösa de problem vi som enligt produktägaren bör lösas, och inte ägnat tid till att addera funktionalitet som egentligen inte behövs.

Tekniken Scrum har ställt nya krav på projektgruppen vad gäller planering jämfört med tidigare erfarenheter. I tidigare större projekt har allt planeringsarbete koncentrerats till tidigt i processen, medan planeringsarbetet nu istället har skett kontinuerligt i mindre faser under projektets gång. Tendenser till en viss frustration har uppstått bland projektmedlemmarna, på grund av ovana med löpande och frekvent dokumentation och planläggning.

Som nämnts ovan fanns viss frustration i att en del tid gick åt till dokumentation och lägga upp planer för det fortsatta arbetet. Dock är det svårt att säga vad resultatet av projektet hade blivit om gruppen inte investerat lika mycket tid i projektplanering. En farhåga är att mycket tid hade ägnats åt att addera funktionalitet som i slutändan inte behövs. Om Scrum inte hade använts hade förmodligen mer kod skrivits, dock hade det förmodligen producerats mindre kod som hade tillfört något värde till projektet. Att använda Scrum gjorde att gruppen arbetade med rätt saker. Därför anser gruppen att metoden Scrum har tjänat gruppen väl.

Flera av gruppens medlemmar är intresserade av att använda sig av tekniken även i projekt som inte har med IT att göra. Parallellt med detta projekt gjorde samtliga gruppmedlemmar ett kandidatarbete. Flera gruppmedlemmar ser fördelar i att använda Scrum även i ett sådant typ av projekt. Allmänt ser gruppen vissa fördelar med att använda sig av Scrum då en tydlig tidsgräns för projektet finns, eftersom Scrum lägger mycket vikt i att ständigt uppskatta hur lång tid olika uppgifter tar, vilket kan ge ökad kontroll i ett projekt. Om projektet inte hade kunnat brytas ner i mindre bitar och uppgifter hade Scrum förmodligen blivit en mer svårtillämpad metod.

4. Diskussion

I detta kapitel kommer diskussion föras kring arbetsprocessen i projektet i termer av rolluppdelning, användning av vissa tekniker och projektets artificiella natur.

4.1 Gruppdynamik i projektet

Programmering i grupp är något som skiljer sig från många andra typer av projekt när något ska skapas. När en grop ska grävas går det ofta snabbare ju fler som är med och gräver. När en mobilapplikation ska programmeras finns inte samma korrelation. Om många utvecklare arbetar på många olika delar till ett program finns det risk för hög komplexitet vid synkronisering av de olika delarna. I projektet var det endast fem personer vars kod behövde synkroniseras, men ändå blev det vissa problem vid ihopslagning av kodsegment.

Processen fungerade i sin helhet bra. Idéfasen var den minst problematiska och vi är nöjda med valet av projekt. Mycket av detta berodde på att medlemmarna hade idéer redan innan projektets start, om så inte hade varit fallet vore det lämpligt att använda sig av andra metoder för att genera idéer. Som sagt användes stop and go brainstorming för att sålla och utveckla befintliga idéer. Denna metod fungerade bra bland annat eftersom vi alla kände varandra sedan innan och går i samma klass, vilket gav upphov till flera spontana och informella möten.

Rolluppdelning i detta projekt kan ses som en avvägning mellan att få bra resultat och att lära sig mycket. Om en gruppmedlem kan någon del extra bra är det för resultatets skull klokt att denna person ansvarar just för den delen. Å andra sidan är det mer lärorikt om någon i gruppen som är sämre på den delen, får ansvaret. Till en viss grad ansvarade varje gruppmedlem för det som denne var duktig på. Om projektet varit längre hade möjligheterna till att rotera arbetsuppgifter varit större. I ett större projekt hade större fokus därmed kunnat läggas på att utveckla gruppmedlemmarnas kunskaper. Eftersom projektet hade en relativt kort tidsram var det svårt att ge alla gruppmedlemmar en både bred och djup teknisk förståelse.

4.2 Användning av backlog

Den backlog som använts för att strukturera upp arbetet användes i enlighet med vedertagen scrum-metodik. Det var första gången gruppen arbetade med user stories, epics och fuzzy stories. Till en början var det god balans mellan de tre olika typerna av applikationsspecifikationer, men i de senare sprinterna fanns bara user- och fuzzy stories. Huruvida detta vittnar om någon obalans i utvecklingsarbetet kan vi inte uttala oss om. Vi tror hur som helst att det beror på att vi under arbetets gång inte fick speciellt många nya krav från kund eller produktägare som hade kunnat användas för att skapa nya fuzzy stories, epics och user stories. Vi tar med oss att nästa gång arbeta mer med nedbrytning av fuzzy stories och epics, för att ha kontinuerligt kunna utveckla och förbättra projektet.

Avslutningsvis, kan vi konstatera att det var bra att snabbt få ut en produkt. Det gjorde att vi enkelt såg vad som behövde göras för att få fram en applikation i enlighet med

kravspecifikationen. I sprint 2 provade vi att arbeta med tasks, vilket var mer omständligt än givande. Samarbetet i gruppen var så pass bra att tydliga user stories räckte. Scrum var ett effektivt sätt att arbeta, för att med en grupp på fem personer kunna ta fram en applikation som eventuellt kan vara användbar för användarna och som samtliga gruppmedlemmar är stolta över.

4.3 Projektets artificiella natur

Efter fem sprints kunde en färdig produkt levereras. Arbetssättet har varit effektivt och lärorikt, vilket bidragit till ett bra slutresultat. Något som bör poängteras är projektets artificiella natur, och en intressant diskussion är hur det har påverkat det agila arbetsättet. Först och främst så finns det ofta i verkliga scrum-baserade projekt följande roller i utvecklingsarbetet: kund, produktägare och utvecklarteam. I detta projekt har utvecklarteamet även varit produktägare, och ibland kund. Gruppens handledare har både varit produktägare och kund. Dessa diffusa och flytande roller gjorde att kravspecifikationen för applikationen inte fluktuerat speciellt mycket på grund av kundens önskemål. Därför har inte önskemål om nya funktioner eller modifiering av specifikationer tillkommit under projektets gång vilket kan tänkas vara vanligt i verkliga fall.

Vidare bidrog avsaknaden av en tydlig kund till att det varit svårt att definiera när applikationen var färdig. Med en kund får utvecklarteamet tidigt acceptanskriterium, som visserligen kan ändras under projektets gång, men som ger tydliga direktiv. Vid prioritering av user stories försökte gruppen att sortera efter högst *business value*. Denna bedömning blir också svårare utan tydlig kravspecifikation från kund.

Integrering av kund i processen är en stor del i verkliga *scrum*-projekt, för att utvecklarteamet ska leverera det kunden vill ha. Vi tror att verkliga fall, med inblandning av kund, är lättare att hantera till en börja då det bör finnas tydliga direktiv i startfasen. Längre in i ett verkligt projekt tror vi dock att det blir mer komplicerat att ständigt anpassa sig efter nya krav från kunden. Detta hanteras effektivt av den agila metodik som ligger bakom *scrum*. Således bör *scrum* vara ett mer effektivt arbetssätt i verkliga fall. Eftersom kravspecifikationen i vårt projekt var i princip densamma genom hela processen hade den gamla metodikten Vattenfallsmodellen eventuellt kunnat användas. En nackdel hade dock varit att det skulle tagit längre tid innan vi fått ut en första version av spelet, vilket vi såg som en stor fördel i vår arbetsprocess. Att arbeta i korta sprints gav oss också möjlighet att tidigt identifiera vad som tog lång tid att göra och vilka initiala idéer för gameplay och design som inte fungerade. Således kunde vi i nästkommande sprint dra lärdom av detta och på ett bättre sätt uppskatta tidsåtgången för user stories samt förbättra gameplay och design.

Projektets artificiella natur kan även bidra till att förklara till varför testdriven utveckling (TDD) inte använts. Då user stories i detta projekt har härstammat från utvecklarna har de också kunnat förändras och justerats under sprinten, till följd av att nya idéer eller implementeringssvårigheter uppkommit. TDD är relativt närbesläktat *acceptance testing,* testning av huruvida koden uppfyller kundens krav. Denna typ av testning blir svår att genomföra då kraven och önskemålen kontinuerligt förändras även under en sprint. Det blir av denna anledning meningslöst att inleda en sprint med att skriva test som baseras på krav som förändras innan sprinten är avslutad. Så som user-stories har använts i projektet har de

främst gett utgångspunkten för ny funktionalitet och inte i samma utsträckning gett en bild av slutresultatet.