Post Mortem Report - Determinator

Grupp: PINOMG
Kurs: DAT255
Handledare: Morgan Ericsson
Chalmers Tekniska Högskola, VT 2015

Björn Agaton
Ebba Mannheimer
Martin Lindehammar
Olle Lindeman
Patrik Olsson
Philip Xu Cederhill

Introduktion
Denna rapport beskriver det software-projekt som genomfördes våren 2015 på Chalmers i kursen Software Engineering Project, DAT255. Projektet har inneburit att utveckla en Android app där olika tekniker och metoder för sådana projekt har använts. Visionen för den app som gjordes i projektet var:

"En app för att fatta *snabba beslut* i vardagen"

Appen möjliggör för användaren att ställa binära frågor till andra användare och snabbt få ett entydigt svar tillbaka. Finessen är att ett beslut alltid erhålls eftersom appen slumpar svaret om svarsalternativen från lika många röster. Det grafiska gränssnittet på appen påminner om snapchat för att användaren skall kunna känna igen logiken och snabbt förstå hur appen skall användas. Meningen med appen är att snabba beslut skall kunna fattas utan att långa diskussioner uppstår. Att frågorna är binära bidrar till enkelheten och frågor som kräver fler svarsalternativ än så får ställas i andra forum.

I dagens samhälle med många olika sociala forum och olika chatt-grupper tenderar enkla beslut, till exempel att bestämma lunchställe, att ta väldigt lång tid att komma överens om. Ofta uppstår diskussioner som är helt onödiga och i vissa fall är snabbheten i beslutet essentiellt och om det inte fattas tillräckligt snabbt så faller planerna. Vi som utvecklat appen har ofta svårt att komma överens när beslut skall fattas. Det slutar ofta med att en omgång sten-sax-påse får avgöra, vilket känns väldigt gammalmodigt med dagens tekniska möjligheter. Dessa problem ämnar appen att lösa och en typisk användargrupp är ett kompisgäng som vill ta snabba beslut kring till exempel vad de skall äta till lunch eller vilken bar som de skall träffas på för en After Work. Andra tänkbara användargrupper är arbetskollegor som skall bestämma mötesrum eller tid för fikarast. Ett annat möjligt användningsområde för ett mer kommersiellt syfte är att genomföra marknadsundersökningar med appen. Detta skulle kunna ske genom att visa två bilder på olika produkter där användaren sedan får avgöra vilken de helst skulle köpa. Nedan beskrivs hur projektet har genomförts och resultatet av arbetet.

Metod
I detta avsnitt presenteras vilka metoder och tekniker som har använts under projektets gång. För varje avsnitt diskuteras för- och nackdelar samt hur användbart arbetssättet hade varit i framtida projekt av olika storlek.

Gruppen
Att arbeta i ett team inom programmering har varit en ny erfarenhet för flera av oss. Två personer har tidigare gjort programmeringsprojekt tillsammans och hade en del kunskap om hur det går till, medan det var helt nytt för de andra. Att de två hade kunskap om Git sedan innan var oerhört hjälpsamt för de andra fyra i början av projektet eftersom de kunde lära ut det till resten av teamet. Det gjorde att teamet snabbt kunde komma igång och börja koda. Att de två även har en mycket bredare kunskap om kodning var också till stor hjälp då de kunde instruera och coacha de andra fyra om någon fastnade med ett kodningsproblem. Gemensamt för alla är att projektets teamutformning har varit väldigt lärorik och gjort det möjligt att testa flera olika tekniker som brukar användas i programmeringsteam.

Den största skillnaden från hur vi alla jobbat med programmering tidigare är att allas individuella bidrag nu behövde passa ihop tillsammans i ett större projekt. Exempelvis behövde klasser som skickar objekt mellan sig ha matchande procedurer för att appen skall fungera. Tidigare har vi inte skrivit kod som har varit tvungen att passa ihop med kod som någon annan utvecklar. Ibland har detta gjort att någon fastnar med ett problem som annars hade varit väldigt enkelt. Detta har däremot förbättrats under projektets gång och det var framför allt i början som problemen kunde uppstå. Vi förstod från början att projekt av den här typen ställer

höga krav på kommunikationen mellan teammedlemmarna men har också lärt oss vikten av tydlig, kommenterad och välskriven kod.

Kommunikationen har från början varit enkel eftersom alla teammedlemmar är nära vänner sedan innan. Detta underlättade framför allt uppstartsfasen eftersom vi kunde komma igång med projektet snabbt då kommunikationen var väldigt smidig. Hade projektet utförts igen med en gruppsammansättning där personerna inte kände varandra sedan tidigare hade uppstarten varit trögare. Detta eftersom att alla behöver lära känna varandra för att kunna arbeta på samma effektiva sätt. Dessutom kunde vi i det här projektet snabbt enas om att vi hade en hög ambitionsnivå, något som alltid är känsligt att diskutera om det är delade åsikter och teammedlemmarna inte känner varandra så bra. En nackdel med att arbeta i en grupp där vi kände varandra väl var att stämningen ibland kunde bli väl avslappnad. Detta ledde till att produktiviteten under de gemensamma programeringstillfällena ibland sjönk. Detta var ofta till förmån för mer lättsamma och sociala aktiviteter som gjorde processen och arbetet roligt, men som i sig inte tillförde något produktivt.

Något som ytterligare förenklat kommunikationen är att fem av sex teammedlemmar vid sidan av projektet även har skrivit kandidatuppsats tillsammans. Därför passade alltid tider för att ses och jobba tillsammans väldigt bra och den enda svårigheten var att bestämma tider med den sjätte medlemmen. Detta har ändå fungerat bra då vi hade som upplägg att försöka koncentrera apputvecklingen till samma dag i största möjliga mån, vilket beskrivs nedan.

Vi bestämde från början att vi ville testa på att försöka göra så mycket som möjligt av kodningen tillsammans. Därför dedikerade vi en dag i veckan till ett så kallat “Hackaton”, alltså ett kodmaraton. Dessa dagar träffades vi tidigt och satt med varandra och arbetade på projektet hela dagen med gemensamma pauser för möten eller mat. Detta blev ett roligt inslag i veckan och dessutom mycket effektivare än om vi suttit och kodat på egen hand eftersom vi lättare kunde hjälpa varandra på plats. Att arbeta på det här sättet kan vara påfrestande och hade kanske inte fungerat lika effektivt i ett projekt där teammedlemmarna inte känner varandra. I det här projektet var det dock väldigt lämpligt då vi som grupp var mest effektiva dessa dagar. Självklart har individuellt arbete skett vid sidan av varje Hackaton, vilket varit nödvändigt men inte lika effektivt för de av oss som inte är lika vana programmerare. I ett större och längre projekt hade detta kanske inte varit hållbart i längden och det hade då varit bättre att inte arbeta så långa pass koncentrerade till en dag. Det passade dock bra för detta projekt med tanken på tidspressen och det fokus som varit på kandidatarbetet under våren.

I efterhand har vi fått uppfattningen att ett team på sex medlemmar är en lagom storlek för den här typen av programvaruutveckling. Att några av oss är bättre på att programmera än andra har inte varit problematiskt utan alla har kunnat hitta områden som passar deras nivå.

Uppskattad tidsåtgång

Vi har uppskattningsvis lagt ner ca 10 timmar per Hackaton under de fem sprinterna som genomförts under projektet. Det har även gått åt 4 timmar till att planera och utvärdera varje sprint. Slutligen har varje person i gruppen ägnat ungefär 15 timmar vardera åt dokumentation, testning och rapportskrivande. Utöver detta har alla medlemmar lagt ner 5 timmar individuellt arbete per vecka, med viss differens mellan oss på +/- 2 timmar. I efterhand så har denna tidsallokering känts bra och vi hade troligtvis lagt upp det på ett liknande sätt igen.

Scrum

Ingen av oss i gruppen har tillämpat Scrum i tidigare utvecklingsprojekt. I detta projekt har vi använt delar av Scrum-metodiken efter egen anpassning. De delar av scrum vi valde var de som ansågs vara lämpliga för storleken på projektet samt de som var enkla att applicera då projektets tidshorisont var relativt kort. De vi framförallt använt är de olika aktiviteterna inom Scrum, exempelvis:

- Sprints
- Sprint Review
- Sprint Planning
- User Stories

Vi har inte använt oss något av de roller som finns definierade inom Scrum. Vi valde bort dessa eftersom vi inte trodde att det skulle tillföra något när vi inte hade en riktig kund till appen. Vi hade ingen Scrum Master då ingen var tillräckligt införstådd i alla principer samtidigt som vi gärna ville vara fria att testa på olika roller under processens gång. I framtiden och med mer erfarenhet inom Scrum tror vi det är viktigt att utse en Scrum Master. Detta för att ha en person som lyfter blicken och kan se till att metodiken följs i den mån man beslutat om.

Att jobba iterativt med sprinter har varit lärorikt då vi tvingades att verkligen tänka efter ifall det vi gjorde var i linje med Sprint Planning samt tillförde värde till "kunden". Vid tidigare mjukvaruprojekt har vi aldrig arbetat särskilt strukturerat utan det har mestadels handlat om att lösa laborationsuppgifter. Vi kom alltså in med ett någorlunda öppet sinne och vi tycker den iterativa processen är fördelaktig och passade väl med projektet.

För liknande projekt där utveckling sker mot kund kommer vi fortsätta använda Scrum metodiken för att snabbt kunna leverera synligt värde. För stora globala projekt kan det antagligen vara svårt att synkronisera alla utvecklingsteam. För kritiska mjukvarusystem är det nog inte heller möjligt att arbeta lika iterativt utan där måste man samla in alla krav och specifikationer och dokumentera de för att säkerställa funktionaliteten Exempel på detta kan vara bromssystem i bilar.

Vi har även valt att använda vissa metoder från eXtreme Programming (XP) så som Pair Programming men också bortsett från andra metoder relaterade till XP. Diskussion och reflektioner kring pair programming tas upp senare i rapporten.

Sprint Planning

Efter handledningstillfället varje vecka genomfördes en sprint planning för kommande sprint. Varje sprint var en vecka lång och totalt genomfördes fem sprinter under projektet. Som underlag för sprint planningen fanns ett stort antal user stories som vi hade skrivit i början av projektet. Under projektets gång lades även nya user stories till, allt eftersom vi förstod vilken ytterligare funktionalitet som kunde vara användbar.

De första user storyn vi skrev var mycket övergripande och mer av typen "epics" än enkla user stories. Det gjorde att de första sprinterna var lite svåra att planera och vi förstod därför att vi behövde bryta ner de i mindre bitar med mer konkret funktionalitet För att planera sprinterna använde vi oss av verktyget Trello vilket beskrivs nedan.

En av de största utmaningarna med att planera sprinterna var att uppskatta hur stor arbetsbörda varje user story innebar. Vissa stories som vi trodde var svåra att genomföra gick mycket snabbt att klara av, medan vissa fick hänga med i några veckor innan de kunde bockas av. Under de första veckorna var uppdragen generellt enklare att programmera medan vi under de sista veckorna stötte på större tekniska utmaningar. Antagligen hade uppskattningen av tidsåtgången blivit bättre och enklare att göra med mer erfarenhet och större kunskap om programmering men viss osäkerhet finns nog alltid kvar.

Ett annat problem vi stötte på under de första sprinterna var att det var svårt att avgöra när en user story kunde anses som färdig. För detta gjorde vi checklistor för varje user story för att ta fram en "definition of done". Mer om hur detta gjordes beskrivs under avsnittet Trello.

Att vi gjorde sprint plannings under hela projektet var väldigt viktigt för att genomföra projektet på rätt sätt, samt undvika att programmera sådant som inte tillförde appen något. Dessutom var det viktigt för att kunna prioritera funktionaliteten rätt och göra saker i rätt ordning. Eftersom att vi var ett så litet team hade vi tät kommunikation hela tiden och det hände därför att vi gjorde saker i appen som inte var med på någon user story. På det sättet gick vi ibland ifrån planeringen, men i de fallen var det saker vi upptäckte som behövde fixas innan resten kunde genomföras. Sådan flexibilitet var bra för vårt team men är något som antagligen bör undvikas i största möjliga mån i större projekt.

Vi hade antagligen kunnat undgå att behöva göra saker som inte var planerade i sprinten genom att göra en mer ordentlig planering och verkligen tänka igenom hur allt skulle integreras innan, men detta hade så klart varit mycket mer tidskrävande och också krävt mer kunskap om android som inte fanns i början av projektet. I detta fall med storleken på teamet och projektets omfattning fungerade arbetssättet mycket bra, men i större projekt och större team bör planeringen göras mer omfattande och följas mer strikt.

Trello

För att organisera våra User Stories har vi använt projekthanteringsapplikationen Trello. Vi valde att använda Trello eftersom några gruppmedlemmar använt appen förut och för att det är byggt för mjukvaruutveckling och Scrum. Vi skapade fem stycken kategorier för att ordna våra kort: Product Backlog, Sprint Planning, Current Sprint, In Progress, In Testing. Vi skapade även en ny kategori efter varje sprint där klara user stories lades.

Fördelarna med appen var att det var lätt att visualisera alla user stories, samarbeta flera användare och se hur korten flödade genom utvecklingen. Man kunde även lätt se vem som jobbade på vad och hur långt personen kommit med detta. Detta förutsatte dock att man tog på sig ett kort och sedan flyttade det till In Testing när man var klar och ville att någon skulle testa ens arbete. I och med att vi alltid satt tillsammans i samma rum när vi utvecklade var det ofta lättare och smidigare att kommunicera detta muntligt. Detta ledde ofta till att korten i Trello var dåligt uppdaterade efter aktuell status. Det fanns även vissa oklarheter över när ett kort verkligen var klart och skulle flyttas till de färdiga korten. Vi införde därför en Definition of Done-lista på varje kort, vilken definierade när ett kort skulle flyttas. Efter sprint 2 laddade vi även hem ett plugin som gav extra funktionalitet som exempelvis möjligheten att poängsätta arbetsinsatsen som krävdes för varje user stories. Detta gav även en samlad summa av sprintens arbetskrav. Det var inte helt lätt att bedöma arbetsinsatsen som krävdes, vilket gjorde att vi bara använde oss av detta en gång.

I och med att vi var en så liten grupp som alltid jobbade tillsammans var tekniken inte alltid tidseffektiv. Den gav dock en bra dokumentation över våra user stories, vilket var nyttigt. Vi tror att appen eller liknande verktyg skulle vara nödvändig och mer användbar i större arbetslag men också för projekt som sträcker sig över längre tid, där det blir viktigt att dokumentera alla user stories. Vi hade inte använt Trello för projekt av mindre omfattning än detta.

Pair Programming

Vi utnyttjade oss av Pair Programming vid utvecklingen av de flesta user stories. För större och omfattande uppgifter använde vi oss alltid av detta medan mindre uppgifter som en person snabbt kunde klara av ibland gjordes ensamt. Vi flyttade runt bland paren så att det inte alltid var samma personer som arbetade tillsammans. Intresse och önskan om vad man ville utveckla, testa på och upptäcka styrde vilken uppgift man gjorde och vem man arbetade med. Ofta var det så att om det var någon som ville testa på något nytt parade han/hon ihop sig med en person som var mer erfaren på den uppgiften så att både kunde lära sig något. Detta ledden även till kunskapsspridning inom gruppen. Fördelarna med detta var att man hade någon att diskutera koden och utveckling med. Istället för att själv sitta och försöka lösa problem kunde man göra det tillsammans och undvika mycket fel, både logiska och syntax-fel. De tillfällen då det var

outforskat område för båda i paret löste vi det ofta genom att en sökte efter lösningar på nätet och den andra testade de allt eftersom för att se vad som funkade. Vi har använt oss mycket av utvecklardokumentationen för Android samt Stack Overflow. Eftersom erfarenheten och färdigheterna inom programmering skiftade i gruppen var detta en utmärkt lösning.

Nackdelar med tekniken var att det ibland kunde bli lite tidsineffektivt att två personer löste en uppgift som en person hade klarat. Fördelarna med högre kvalitet övervägde dock nackdelarna så vi fortsatte med arbetssättet genom hela projektet. Vi har lärt oss mycket genom att arbeta parvis, det var det som drev projektet framåt och kommer arbeta parvis i liknande projekt framöver. I framtiden och med större färdigheter och erfarenheter av Androidutveckling tror vi det är möjligt att arbeta ensam. Men den kvalitetssäkring och kunskapsspridning som pair programming för med sig kommer fortfarande vara argument nog för att metoden skall användas.

Stand up meeting

För att säkerställa att alla i gruppen var införstådda med vad som skulle göras vid varje programmeringstillfälle införde vi korta möten vid varje uppstart. Vi dessa möten sammanfattade vi hur vi låg till i sprinten. Framför allt pratade vi om vad som var gjort och hur det hade gått. Vi diskuterade också om arbete för att lösa eventuella problem fått några konsekvenser för det fortsatta arbetet. Detta gjorde att vi enkelt kunde kommunicera tagna beslut och arbeta mer effektivt genom att konflikterande lösningar och redundant arbete kunde undvikas.

En viktig funktion var också att stämma av vad som var kvar att göra under sprinten och hur det skulle prioriteras. Ibland var möten kortare och varade bara någon minut och ibland uppstod diskussioner som ledde till att mötet kunde vara upp till en halvtimme. En viktig fördel med mötena var att det var ett bra tillfälle att meddela ifall någon stött på problem och behövde hjälp. Vid de tillfällen vi planerade att fortsätta arbetet individuellt avslutades de gemensamma programmeringstillfällena med ett kort, sammanfattande möte för att stämma av vad som skulle göras.

Att ha korta semistrukturerade möten fungerade väldigt bra och hjälpte gruppen att gemensamt föra projektet framåt. Flera av oss hade tillämpat liknande möten i andra sammanhang eller andra projekt vilket ledde till att det var enkelt och effektivt att införa. Trots att metoden är generell och har flera användningsområden lämpade den sig väl för software-projekt och är något som vi kommer fortsätta att använda oss av i framtida software-projekt.

Google Drive

För skrivna dokument och tester använde vi oss av Google Drive. Fördelarna med detta var att alla kunde bidra och läsa alla dokument i realtid. I och med att vi arbetade med Scrum så skedde mycket av dokumentationen allt eftersom projektet fortskred. Därför var det bra att kunna gå in och läsa vad som redan dokumenterats och vad som behövde läggas till eller ändras. Om vi arbetat enligt vattenfallsmodellen tror vi att behovet av applikationer av den här typen minskar eftersom dokumenten då är mer statiska där allt är skrivet i förväg. Vi som grupp är väldigt vana vid Google Drive och kommer använda det i alla olika typer av projekt även i framtiden. En liten nackdel med att använda Google Drive var att det blev ytterligare ett ställe som vi hade information på och det fanns en viss risk för att information blev alltför fragmenterad. Vi gjorde dock bedömningen att risken för detta var liten och att fördelarna övervägde nackdelarna så vi fortsatte med systemet.

Git

Vi använde oss av Git som versionhanteringssystem med Github som repository. Det ställde till lite problem i början då endast två personer i gruppen tidigare använt Git. Vi spenderade en dag då de gick igenom grunderna i Git genom att vi skapade vårt repository samt gjorde våra första

commits. Om inte de två hade haft erfarenhet inom Git hade uppstartssträckan var mycket längre. Fördelarna med Git som versionshanteringssytem är uppenbara med den möjlighet det ger att utveckla separata funktioner för att sedan sammanfoga det. Utan det hade det inte varit lika smidigt att jobba på separata datorer.

Vi satte upp vissa grundregler i början där man inte fick arbeta mot master branchen utan alla nya funktioner gjordes i en ny branch som gick från develop. När man var klar med sin funktion gjorde man en pull request till develop och någon annan än paret som utvecklat funktionen godkände koden. Inför varje Client Meeting (handledningstillfälle) testade vi koden och slog sedan ihop den med master. De nackdelar vi upplevde var ofta förknippade med våra bristande färdigheter inom systemet då det ibland kunde ta lite extra tid när man stötte på problem och var tvungen att hämta någon annan som kunde reda ut det. Något som vi upplevde som lite frustrerande var att de kunde vara omständigt att göra en liten, trivial modifikation i koden ifall man inte arbetade i en aktiv branch. Det kunde då vara enklare att kommunicera ändringen som behövde göras till någon som satt bredvid.

En reflektion vi gjort angående vår branchnings-konvention är att det tenderade till att bli många små branches som alla inkrementellt resulterade i en feature. En anledning till dessa "mini-branches" var att vi tidigt i projektet behövde använda varandras kod, t.ex. behövdes klassen Poll för många features i början. Men vi har nu i efterhand insett ett bättre sätt att jobba med branches, där en branch i princip kan vara kopplad till en user story och bara mergas in i develop efter noggrann testning och codereview. Detta hade inte bara inneburit en snyggare historik i git utan även ett tydligare sätt att arbeta där develop-branchen alltid håller hög kvalitet.

I övrigt fungerade Git utmärkt och vi kommer använda det igenom inom både stora och små projekt.

Tester

Vi har parallellt och i efterhand konstruerat tester för vissa funktioner i appen. Det var inget vi gjorde från början utan skedde en bit in i utvecklingen. En av de grundläggande metoderna i XP är just testdriven utveckling och är alltså inget vi har använt oss av i någon större utsträckning. Vi märkte dock att tester kunde vara ett utmärkt sätt att upptäcka buggar, fel och misstag tidigt. Framförallt mindre betydande eller tydliga buggar upptäcktes med tester. Det innebär också att vi tänkte till innan vi kodade vilka funktioner som var nödvändiga. Ett exempel på detta är att vi genom att skriva några tester innan utveckling insåg att vi saknade stöd för att kunna backa i appen med bibehållen data. En nackdel med testerna var att det tog tid från själva kodandet och därför skrev vi endast test för vissa funktioner. Vi har inte heller skrivit några automatiska tester för det grafiska gränssnittet då vi saknade kunskap inom detta och eftersom flödet i appen är ganska simpelt.

Enklare enhetstest har skrivits för att testa metoden TDD, Test Driven Development, alltså att test skrivs innan själva koden implementeras. Detta var något som vi tyckte underlättade programmeringen eftersom man hade en väldigt tydlig definition of done, vilket var när testet gick igenom. I framtida projekt är just automatiska test något vi verkligen skulle vilja få mer kunskap om och erfarenhet av. I större projekt med mer tid och personer tror vi testdriven utveckling kan vara väldigt effektivt för att upptäcka brister och något vi i så fall hade använt oss mer av.

Resultat

Projektet och arbetsprocessen kring detta har överlag fungerat mycket bra. Eftersom vi hann genomföra fem stycken sprinter hann vi också testa på olika arbetsmetoder och utvärdera dessa varje vecka. Det som fungerade bra fortsatte vi att göra och det som inte gav någon effekt slutade vi med.

De metoder vi tyckte fungerade bra var pair programming, korta stand-up meetings, sprint planning och Git som versionhanteringssystem. Trello som användes för sprint planningen var lite onödigt tidskrävande för den hjälp det gav och därför gick vi ifrån det verktyget i slutet av projektet. Några av de metoder vi lärde oss om i kursen var mycket bra men antagligen är de ännu mer användbara i större team och större projekt. Eftersom vi var ett litet team och alltid programmerade tillsammans i samma rum blev kommunikationen mycket enkel, varför vissa verktyg och metoder då blev överflödiga.

Ibland var det svårt att hålla sig till planeringen för varje sprint och det hände att vi gjorde saker som inte var med i sprintplaneringen. Det gjorde att vi tappade viss överblick och ibland gjorde saker som var enkla men inte högsta prioritet och att andra svåra och krångliga user stories fick hänga med någon vecka till. Dock var det inget som försvårade kodningen eftersom att vi hela tiden kommunicerade vad vi gjorde till varandra men en lärdom är att vara mer konsekvent med planerna och utarbeta tydliga sprinter.

Diskussion

I ett liknande projekt i framtiden skulle vi utveckla sprintplaneringen för att även täcka ett lite mer långsiktigt perspektiv. Vi upplevde att våra planeringar som endast täckte en vecka var lite för detaljerad för att få en övergripande bild över de fem veckor långa projektet. Det hade varit bra att komplettera detaljplaneringen kring den nästkommande sprinten med en större grov planering över vad som skulle göras i de efterföljande sprinterna. Då presentationen i slutet av kursen utgjorde en tydlig deadline för när appens mest grundläggande funktionalitet skulle levereras och visionen skulle vara översatt i kod fanns även ett behov av en långsiktig planering för att säkerställa att detta kunde levereras. En sådan mer övergripande planering borde innefatta i vilken ordning epics och de större funktionerna skulle implementeras.
Vi misstänker att en grupp med mer erfarenhet av scrum och androidutveckling skulle kunna definiera user stories och värdera tidsåtgången på dem på ett sådant sätt att denna övergripande planering inte behövs. Vi upplevde att det var väldigt svårt att i förväg bestämma tidsåtgången för en user story eller avgöra ifall en user story kunde delas upp i flera mindre user stories för ett bättre resultat. Det gjorde att det var svårt att lita på att vår kortsiktiga sprintplanering skulle hålla och upplevde därför ett behov av mer övergripande planering.

Det har varit väldigt lärorikt att arbeta med scrum-metoder och agil programmering. Att praktiskt använda det har varit en nödvändighet för att förstå funktionen med det. Dock upplever vi att samordning och koordinering inte varit något större problem för ett projekt av den här storleken med endast ett utvecklingsteam. Det känns därför som att vissa resurser lagda på att använda scrum ibland kostat mer än vad vi har fått ut det. När vi befunnit oss på samma plats och arbetat på en relativt simpel app har mycket av koordineringen av gruppen kunnat skötas muntligt.

Användningen av Git tillförde mycket till vårt arbete efter den korta uppstartsträckan i början. Git är verkligen något vi kommer använda oss av i framtida projekt. Vi har även konstaterat att versionshanteringen tillhandahåller viss funktionalitet som vi har saknat vid skrivandet av kandidatrapporten, så vissa av oss kanske i femte årskursen istället skriver examensarbetet i Git.

Slutligen tycker vi alla att detta projekt har varit väldigt roligt och lärorikt. Tyvärr har den knappa tiden på fem sprints varit lite kort för att komma så långt i utvecklingen som vi velat men det bästa är att appen finns kvar och vi kommer fortsätta med utvecklingen av den. Detta gör vi inte bara för att det är kul och utvecklande, utan också för att vi verkligen är i behov funktionaliteten att förenkla alla de små vardagliga beslut som måste tas varje dag.