Post Mortem Report

Inledning

Odin är en Androidapplikation som motverkar mobbning genom att kartlägga och visualisera mobbningsincidenter. Genom att tillhandahålla en plattform där personer som blivit utsatta för kränkningar enkelt kan rapportera incidenten skapas en möjlighet till kartläggning av mobbning som tidigare inte varit möjlig. Även personer som sett andra som blivit utsatta för mobbning kan rapportera incidenter för att hjälpa kartläggningen. Kartläggningen kan sedan användas i realtid för att geografiskt visualisera hur utbredningen av mobbning ser ut i olika områden. Detta kan vara till stor hjälp för myndigheter och hjälporganisationer som kan rikta stödmedel till specifika områden eller använda data till bättre forskning och rapportering. Det kan även hjälpa skolor, bostadsrättsföreningar och andra intressenter att skapa sig en bild av problemet i sin närhet. Slutligen är det framförallt ett stöd för personer som fallit offer för mobbning att kunna uttrycka sin situation anonymt och ta del av data kring var andra personer upplevt en liknande situation.

Följande punkter beskriver projektets epics (övergripande funktionalitet):

- Användaren ska kunna rapportera incidenter
- Användaren ska kunna se vart incidenter tidigare ägt rum på en karta
- Användaren ska kunna få tillgång till kontaktuppgifter till olika hjälp- och stödfunktioner

Varför är informationen intressant?

- ”Kartläggning [av mobbning] är grunden till allt trygghetsarbete och en av de viktigaste insatserna mot mobbning.” – stödorganisationen Friends
- Offer för mobbning kan se att de inte är ensamma i sin situation.
- Myndigheter, hjälporganisationer, skolor och föräldrar kan sätta in resurser där mobbning är vanligt förekommande.

Planering

Projektet inleddes med interna diskussioner kring syftet, målet och visionen med applikationen för att därigenom skapa en gemensam grund att stå på inför det fortsatta arbetet. Redan under första mötet formulerades ett antal preliminära user stories och målsättningar för applikationen. Stor vikt lades vid att skapa en gemensam bild av syftet och visionen med applikationen. För att få en tydlig struktur på våra tankar och idéer användes Trello, en molnbaserad tjänst för projektplanering. Med hjälp av Trello kunde gruppmedlemmarna tilldelas olika roller med tydliga ansvarsfördelningar. Några veckor in i projektet fasades Trello ut till fördel för Google Drive. Fördelen med att använda Google Drive istället för Trello var bland annat att user stories, dokumentation och projektplaneringen kunde samlas på ett ställe, samt att miljön ansågs mer användarvänlig, då alla var vana vid google-miljön.

Tidigt i projektet kontaktades också flera olika hjälporganisationer för hjälp och idéer så att applikationen faktiskt utvecklades efter användarens behov. Hjälporganisationen Friends bidrog med idéer och förslag till applikationen men hade tyvärr inte möjlighet att avstätta tid och resurser för projektet under våren. Dock tyckte de att det var en väldigt bra idé på applikationen och ville gärna ha återkoppling när projektet var klart.

Sprintplanering

Under projektets gång användes sprintplanering för att sätta veckovisa delmål för projektet. Delmålen formulerades under gruppmöten som ägde rum en gång i veckan. Utgångspunkten för dessa delmål var user stories, med viss anpassning om vad som verkade praktiskt genomförbart under en sprint. De olika user stories tilldelades olika nivåer av uppskattat arbetsåtgång, eller “effort levels”. Detta skapade en bra bild som hjälpte till att prioritera olika funktioner. I vissa fall framgick dock att dessa uppskattningar varit helt felaktiga, vilket gjorde att arbete las ner på funktioner som sedan visades vara för tunga för att implementera. Ett exempel på detta var en tidig ambition att implementera en “heat map” som visade starkare färger där det var mer vanligt försikommande med incidenter, vilket uppskattades som en “mycket enkel åtgärd”, men visade sig medföra för mycket arbete för att det skulle hinnas med.

Genom sprintplaneringen fick gruppen en bra överblick över vad som behövde göras under veckan och vem som var ansvarig för respektive uppgift. Nackdelen med denna typ av planeringsprocess var att arbetsbelastningen under vissa veckor blev ojämnt fördelad mellan projektgruppens medlemmar. Detta kunde till exempel handla om att arbetsbelastningen för den person som var ansvarig kopplingen mellan applikationen och servern var extra hög under en specifik fas i projektet. Sammanfattningsvis fungerade sprintplanering mycket bra för vårt projekt och detta är en teknik vi även kommer nyttja i framtida projekt.

Metodik

Pair programming

Parprogrammering har använts i viss utsträckning under projektets gång. Detta var speciellt användbart när ett mer komplext problem skulle lösas och två personers problemlösningsförmåga kunde kombineras. Parprogrammering var också användbart när vi experimenterade med olika lösningar och idéer. En fördel med parprogrammering är att man alltid har en annan person att bolla idéer och tankar med samtidigt som koden granskas av ytterligare en person vilket därmed kan höja kvaliteten på koden. En nackdel med parprogrammering är att man förlorar flexibilitet till viss del. I vårt fall tog sig detta uttryck i form av att det emellanåt var svårt att planera in gemensamma tider eftersom alla projektmedlemmar har olika scheman och läser olika kurser. Detta kunde emellanåt utgöra ett hinder för framsteg i projektet. Ytterligare en nackdel är att produktiviteten riskerar att sänkas vilket gäller speciellt för enklare programmeringsuppgifter där en person ensam kunde löst

uppgiften. Sammantaget kan konstateras att parprogrammering är ett effektivt verktyg om det används vid rätt tillfälle och för rätt typ av programmeringsuppgifter.

System för versionshantering - Git

Under projektets gång har Github använts som versionshanteringssystem (VCS) och repository för mjukvarukoden. För dokumentationen av projektet har främst Google Drive använts. Den främsta fördelen med att använda dessa system är möjligheten för olika användare att arbeta parallellt samt möjligheten att arbeta på olika versioner. Vid tillfällen då problem uppstått har det genom Githubs versionshantering, och funktionen rollback, varit möjligt att gå tillbaka till en fungerande version av koden och fortsätta arbeta vidare därifrån.

Nackdelen med att använda Github som versionhanteringssystem var den långa startsträckan i projektets inledande fas eftersom de flesta av projektmedlemmarna inte hade någon tidigare erfarenhet av Git. Mycket tid spenderades på att förstå hur systemet fungerade snarare än att faktiskt använda det vilket hade en negativ inverkan på produktiviteten i projektet. Med bättre förståelse för hur detta system fungerade hade branching and merging kunnat användas mer effektivt. Det kan också konstateras att fördelarna med att använda Git är större i projkekt med fler projektmedlemmar. I vissa avseenden var vår projektgrupp för liten och att systemet därför blev något svårarbetat.

Trots nackdelarna med att använda Git som versionshanteringssystem i detta projekt kan det konstateras att detta är ett mycket populärt verktyg och att ha lärt sig använda Github kommer vara en tillgång i framtiden. Troligtvis kommer Github eller liknande versionshanteringssystem användas i andra teambaserad mjukvaruprojekt som vi arbetar med i framtiden varför detta är en nyttig erfarenhet.

Delar av Scrum

Inledning

Under projektets gång använde vi ett antal beståndsdelar från Scrum som är en metodik för systemutveckling. Kursens utformning förutsatte användning av vissa delar i Scrum men det var också ett antal aspekter som inte användes eller som användes i mindre utsträckning på grund av olika anledningar.

Roller

Scrum utgår från de olika rollerna Produktägaren, Scrum Master och utvecklingsteamet. Handledaren för kursen var Produktägare och ansvarade för produkten och som i samband med oss bestämde vad som skulle prioriteras och vad som skulle göras under våra sprintar. Vi själva i gruppen fungerade som utvecklingsteamet som gjorde själva utvecklingen av produkten. Vi hade ingen uttalad Scrum Master för projektet vars roll skulle vara att coacha teamet och se till så att allt rullade på smidigt. Anledningen för detta var att vi inte kände att vi

behövde en specifik person för detta utan att vi tillsammans samarbetade mycket för att lösa de problem vi stötte på. Eftersom mycket i kursen var nytt för många av oss i gruppen, som exempelvis Git och Android Studio, så prioriterade vi att vi lärde oss det istället för att ha någon som skulle coacha oss. De olika rollfördelningarna under kursens gång fungerade bra även om vi inte hade någon uttalad Scrum Master. En anledning för detta skulle kunna vara att projektet var relativt litet och att fokus inte var på att koda effektivt utan snarare att lära sig om Git, Android Studio och Scrum. Dock kan det tillägas att gruppen inte arbetade särskilt aktivt med de olika rollerna eftersom alla vi studenter var utvecklare och vår handledare var vår Produktägare som guidade vårt arbete framåt. Fördelarna med att använda roller är att olika personer har olika ansvar och därför koll på att olika moment faktiskt genomförs. En nackdel med roller kan vara att man blir intvingad i strukturer som faktiskt inte passar arbetet, speciellt eftersom roller som behövs oftast utvecklas och förändras under tidens gång. I framtida projekt antas roller fungerar bra om det är många som är inblandade och ifall projektet är relativt stort. Om däremot projektet är litet och kort så lär roller vara mindre viktiga medans samarbete och flexibilitet för högre prioritet. Avslutningsvis kan det väl sägas att roller kan hjälpa till att strukturera och förenkla ett arbete så länge som personerna inte blir för låsta i sina roller.

Product Backlog

För de olika beståndsdelarna i Scrum så använde teamet en Product Backlog av olika User Stories som vi ville implementera under kursens gång. Dessa hade vi skrivit i Google Drive i början av projektet och vi kompleterade även med nya User Stories under arbetets gång. Det som inkluderades i våra User Stories var vad användaren ville ha samt varför. Vi hade också estimerat hur viktiga de olika User Stories var samt hur mycket tid det skulle krävas för att implementera User Story:n. Det som skildes från teorin om användningen av en Product Backlog var att det inte var Produktägaren, alltså vår handledare, som ägde och hanterade alla önskemål utan det var vi i utvecklingsteamet som ansvarade över dem. Det fungerade dock fortfarande bra då vi inte behövde använda Product Backlog så ofta utan mest under våra veckovisa möten med vår handledare där vi bestämde vad vi skulle arbeta med under veckan. Eftersom vår Product Backlog inte användes så aktivt utan mest för att bestämma vad vi skulle göra under våra sprintar så är det svårt att säga hur effektiv den var eller inte. Hursomhelst var det väldigt bra att från ett tidigt stadie rangordna och bestämma vad vi ville ha med i applikationen och att vi hela tiden kunde återkomma till den för att se vad nästa steg i utvecklingen av applikationen skulle vara. Tekniken är väldigt bra för att få överblick vad som behöver göras för ett arbete samt för att prioritera vad som är viktigast att göra. Därför skulle den kunna användes till många olika arbeten i framtiden med syfte att strukturera och prioritera arbetet, även om det bara exempelvis är för vad som ska göras under en rapportskrivning. Troligtvis skulle man inte använda tekniken för ett väldigt litet projekt där det inte behöver struktureras upp så mycket vad som ska göras eller ett projekt där många olika delar arbetas med parallellt. Men sammanfattningsvis en bra metod för att prioritera vilket arbete som för tillfället är viktigast samt vad det är som ska göras.

Sprint Backlogs

Vi använde också beståndsdelen Sprint Backlog under projektets gång. För oss var vår Sprint Backlog det den del i Product Backlog som vi i utvecklingsteamet skulle implementera under den kommande sprinten. Oftast hade vi en generell idé över hur vi skulle göra detta men eftersom vi inte alltid hade stenkoll på hur olika system fungerade så fick vi ibland börja sprinten med research. Fördelen med att ha en bestämd del av vad som skulle göras under sprinten var att det hjälpte oss att fokusera i utvecklingsteamet på vad vi skulle arbeta med under de närmaste dagarna. Nackdelen var att vårt programmerande då ibland blev kortsiktigt och bara löste det som skulle lösas för tillfället. Alltså utvecklade vi inte alltid koden till det som var optimalt för senare implementeringar. Dock har vi förstått att detta ändå är ett bra arbetssätt då det innebär att vi inte börjar utveckla saker som vi senare aldrig kommer använda utan att vi bara utvecklar det som för tillfället är relevant och som kommer användas. Det tog väldigt lite tid att använda en Sprint Backlog då det i princip utgick från vår Product Backlog och att vi bara bestämde under våra veckomöten med vår Product Owner (handledare) vad det var vi skulle prioritera under veckan. Därför var det en relativt effektiv metod som hjälpte oss att fokusera men som inte krävde så mycket tid. I framtida projekt så känns det som en bra teknik ifall det finns många olika utvecklingsområden men där det är viktigt att fokusera på mindre uppgifter åt gången. Den är kanske mindre optimal ifall utvecklingen av många beståndsdelar sker parallellt och inte i olika distinkta sprints. Eftersom vi använde Sprint Backlogs under hela perioden så går det inte att jämföra för oss vad skillnaderna var med att använda eller att inte använda Sprint Backlogs.

Inkrement

En tredje beståndsdel av Scrum som vi använde var det som kallades Inkrement, alltså det som skapas i varje sprint eller med andra ord en körbar produkt med nya tillskott. Det vi skapade gjorde vi med transparens mot varandra i gruppen så att alla hade koll på vad vi hade gjort, vad vi inte hade gjort samt vad som är bäst att göra härnäst. Genom att arbeta mot att få fram ett exekverbart program så var det lättare att se vad vi hade åstadkommit hittills samt testa för olika buggar. Det vi var mindre bra på var att visa hur vi rent kodmässigt hade skapat de nya funktionaliteterna för varandra. Vi var alltså relativt bra på att visa vad för ny funktionalitet som hade tillkommit men inte hur den hade implementeras. Detta gjorde det ibland svårt för andra i utvecklingsteamet att fortsätta arbeta på olika delar av applikationen då det enbart var en person som hade koll på koden. Det är hursomhelst en effektiv metod med Inkrement för att se hur långt arbetet har kommit samt vad som är kvar att göra. I framtida situationer är det en bra metod ifall det är mindre funktionaliteter som adderas till ett redan exekverbart program som visuellt visar vad den nya funktionaliteten är. Antar att det är en mindre bra metod för program som inte är körbara förrän en stor del av koden redan har skapats.

Sprints

Som tidigare nämnt använde vi också konceptet Sprints från Scrum. Våra sprints var en vecka långa och vi hann med 6 sprints under kursens gång. För oss började en sprint med ett möte med vår Product Owner (handledare) där det bestämdes vad som skulle göras under den närmaste veckan (Sprint Backlog) med utgångspunkt från vår Product Backlog. Sprinten avslutades sedan en vecka senare med ett möte med vår Product Owner (handledare) då de utlovade ändringarna som skulle ha gjorts granskades.

Daily Scrums

En sak som vi inte använde var Daily Scrums, alltså korta dagliga planeringsmöten för teamet. En förklaring till detta var att teamet inte sågs varje dag utan vi jobbade ofta på olika platser vid olika tillfällen. Det kan däremot nämnas att när vi väl sågs alla tillsammans, även om det inte var så ofta, så började vi ofta med att prata ihop oss om hur det gått den senaste tiden, vad vi har kvar att göra och vad vi skulle göra just då. Detta var inte på grund av att det var en del av Scrum utan på grund av att det är ganska naturligt med att få alla medlemmar i en grupp på samma plan när man väl träffas. Detta blir extra viktigt ifall gruppen inte träffas så ofta vilket var fallet för oss då många var upptagna med andra saker till och från. Vi har alltså svårt att säga fördelarna och nackdelarna med Daily Scrums då vi inte använde detta koncept. Det som dock kan sägas är att det generellt är bra om alla medlemmar i en grupp har koll på vad andra har gjort och vad andra har kvar att göra. Alltså ge en uppdatering på den senaste tidens arbete. Generellt sätt väldigt effektivt då det inte brukar ta så lång tid. En nackdel skulle väl kunna vara ifall det just tar lång tid att berätta vad man gjort den senaste tiden och vad man har kvar att göra om det inte är relevant för de andra och ifall det tar för lång tid att berätta. Slutligen kan vi säga att vi inte använt Daily Scrums men har gjort en linkande planeringsmöte de gånger vi träffats alla tillsammans.

Sprint Retrospective

Sprint Retrospective är en ytterligare del som vi använt lite från Scrum ramverket. Scrum Retrospective innebar att alla gruppmedlemmar och Produktägaren (handledaren) satt tillsammans för att lära sig från sprinten som gått. Detta gjordes inte jätteaktivt utan mer som en del av våra veckomöten där vi gruppmedlemmar träffade vår handledare. Fördelen med Sprint Retrospective är att gruppen kan identifiera förbättringar i arbetssättet och kan försöka förbättra dessa under kommande sprinter. Nackdelen blir om bara möjliga förbättringar identifieras men sedan inte implementeras. En möjlig förbättring från vår grupp skulle kunna vara att arbeta mer aktivt med att utvärdera hur vårt senaste arbete gått istället för att bara göra det om det kommer upp. Vi skulle också ha kunnat arbeta mer med att faktiskt försöka förbättra det vi insåg var brister i vår arbetsmetodik. Exempelvis märkte vi rätt tidigt i kursen att vi hade svårt att hitta tillfällen där alla gruppmedlemmar kunde träffas. Det blev bättre i slutet men vi kunde ha jobbat mer aktivt och försökt planerat in möten mer i förväg. Att lära

sig från sprinten som gått är väldigt värdefullt då de aldrig går helt perfekt. Det är därför väldigt värdefullt att göra detta i framtida projekt så länge som det inte tar alltför lång tid från programmering och liknande arbete. Det bör alltså inte läggas för stort fokus på hur vi arbetar jämfört med vad vi ska producera men det är fortfarande viktigt med någon sorts av balans. Slutligen kan vi säga att ifall man faktiskt vill lära sig av sitt arbete så är det bättre att arbeta mer aktivt med det än att bara köra på och hoppas att det blir bättre.

Summering

Avslutningsvis angående Scrum kan vi säga att ett antal beståndsdelar användes men att gruppen generellt sätt skulle kunna har arbetat mer aktivt och mer strukturerat kring metodiken. Ifall detta hade gjorts hade vi nog fått flera av fördelarna med Scrum vilket betyder att den slutgiltiga produkten hade blivit bättre. Dock kan det tilläggas att det ofta är svårt att lära sig ett helt nytt ramverk direkt utan det kräver ofta lite tid att lära sig de olika delarna. Därför lär vi kunna använda Scrum-metodiken bättre nästa gång vi använder den för ett projekt.

Spenderad tid

Till vårt förfogande har vi haft 20 timmar per vecka och person. Eftersom vi var sex personer i gruppen så hade vi alltså 120 timmar per vecka till förfogande.I dessa timmar ingår programmeringstid, föreläsningar, handledningsmöten, gruppmöten samt research. Även om de inte förts någon loggbok över antalet arbetade timmar i projektet kan det konstateras att vi vissa veckor arbetat mer och andra veckor mindre. Sammantaget uppskattar vi att vi på projektet arbetat 15-20 timmar per vecka och person. Fördelningen av tiden uppskattas var följande för hela projekttiden:

- Kodning: 40%
- Dokumentation: 25%
- Planering/strukturering/design: 15%
- Testning: 10%
- Kundkontakt (med hjälporganisationer): 5%
- Presentationsförberedlese: 5%

Samarbetet

Trots vissa svårigheter att gemensamt boka in tider för att arbeta på projektet så har samarbetet och kommunikationen i gruppen fungerat bra. Att träffas en gång i veckan och diskutera föregående veckas arbete, samt bestämma vad som behöver göras för kommande sprint, fungerade väl i vår projektgrupp. Detta var ett bra sätt för att hålla alla i gruppen informerade om hur projektet fortlöper men också ett bra sätt för att fördela arbetsuppgifter.

Förutom veckovisa möten har vi använt Facebook som främsta kommunikationskanal. Denna kanal har använts delvis för att planera in gruppmöten men också för att ställa frågor och bolla idéer med de övriga gruppmedlemmar. Sammantaget har samarbetet i vår grupp fungerat bra. Projektmedlemmarna har alla olika bakgrunder och förkunskaper och tillsammans har vi kompletterat varandra på ett bra sätt.

Trots detta har har arbetet varit ganska segregerat, då de olika gruppmedlemmarna haft problem att koordinera gemensamma arbetstider. Detta har stundtals skapat informationsklyftor mellan olika delar av arbetet, vilket sedermera byggts över enligt ovan, men det anses att klyftorna och segrationen gett upphov till merarbete och förlorad arbetstid.

Tankar inför framtida projekt

Planeringen av projektet kunde förmodligen effektiviseras genom att arbeta med mindre omfattade sprints samt mindre user stories. I vissa fall tenderade våra user stories att bli ganska omfattande vilket dels gjorde arbetsbelastningen i gruppen ojämn men ledde också till projektet blev svårare att överblicka då det var mycket “work in progress”. Med kortare sprints och/eller mer frekventa möten så hade det varit lättare att dela upp arbetet mer flexibelt eftersom projektet ständigt växte och utvecklades, och det kan ha hänt mycket mellan två möten.

Under projektets gång har delar av olika utvecklingsmetodiker, såsom Scrum, använts. Inför kommande projekt bör arbetsprocessen specificeras tydligare och anpassas till gruppen. Optimalt är förmodligen att skapa en egendefinierad arbetsprocess utifrån delar av Scrum, XP, etc. Genom att alla projektmedlemmar använder samma arbetsmetodik blir det lättare att överblicka projektet och styra utvecklingen.

I framtida projekt bör vi fokusera mer på testning av koden. I vårt projekt skedde detta löpande men kunde ha gjorts betydligt mer strukturerat och systematiskt.

Gruppmöten under projektets gång kunde också ha effektiviserats genom att göra en person ansvarig för att boka in möten såväl som att skapa en dagordning inför respektive möte. I vissa fall var våra veckomöten oproduktiva och ineffektiva eftersom det inte fanns någon tydlig agenda eller dagordning. Bättre struktur på mötena hade därmed bidragit till en effektivare arbetsprocess.

Ytterligare en del av utvecklingsarbetet som kunde gjorts effektivare är ökad modularisering av koden för att minska kopplingen mellan olika delar av applikationen. Att skapa väldefinierade syften för alla klasser och metoder hade haft en positiv inverkan på produktiviteten i projektet eftersom det då skulle bli enklare för övriga gruppmedlemmar att förstå kod som andra skrivit. I planeringsfasen borde vi också ha definierat ett tydligt UML-schema för alla klasser och komponenter. Detta hade sparat tid under projektets gång.