Post Mortem

Team BusGen

Kristoffer Knutsson
Alexander Kloutschek
Daniel Wassbjer
Michel Folkemark
Johan Iversen

1. Arbetssätt och upplägg

(Svara på fråga 1 och 3)

1.1 Upplägg

Denna underrubrik kommer lite lätt gå in på en av de första besluten vi anser vara ett misstag, och det beslutet var att göra en ganska hård avgränsning mellan androidutveckling och serverutveckling. Mer om varför detta anses som ett dåligt beslut kommer under rubrik 3.

På det första mötet angående vad som skulle erbjudas i applikationen fick alla komma med förslag på User-Stories, vilken lades upp i en lista på Trello. Denna lista bearbetades sedan och "översattes" till faktisk funktionalitet och därmed arbetsuppgifter varpå de sorterades efter prioritet. Denna lista gallrades och modifierades något vid ett senare tillfälle efter att vår enkätundersökning visade ett svalt intresse i vissa funktioner vi tänkte ha med.

1.2 Kanban

Med en lista på uppgifter så ansågs Kanban vara den mest passande metoden för parprogrammering där ingen naturlig produktägare fanns med i bilden, och arbetet kunde fortlöpa genom att beta av denna lista.

Innan ett programmeringsprojekt kan all funktionalitet ses som ett fiktivt berg av uppgifter att genomföra, och detta berg kan kännas oöverkomligt. En styrka med Kanban (och Scrum) är att detta berg delas ner i småbitar som är lättöverkomliga. Givet situationen med uppdelningen var Kanban ett effektivt arbetssätt som tillät oss att snabbt komma ur startgroparna och få fram en kod som vi var nöjda med, det ska dock tilläggas att mer tid eventuellt hade lagt på organisering om vi "delat upp elefanten helt horisontellt".

I Kanban finns det inte roller, vilket är både dess styrka och dess svaghet. För projektets räkning var det en nackdel att inte ha en produktägare då det aldrig kom ett läge att stanna upp och fundera på vad det är man utvecklar, samt att en eventuell produktägare inte får ett läge att ändra sig, lägga till eller ta bort funktionalitet. Detta leder till att man i slutändan kan stå med en produkt inte är vad kunden tänkt sig eller helt enkelt inte är så bra som den kan bli. Därav kommer vi till slutsatsen att Kanban lämpar sig bra när funktionalitet kan/bör släppas kontinuerligt, till exempel vid vidareutveckling av projekt. Svagheten med Kanban kommer när Scrum eller hybriden Scrumban är att föredra, till exempel i projekt där man utvecklar något åt en kund och önskemålen kan komma att förändras mycket under projektets gång

1.3 Parprogrammering

Parprogrammering användes under både utvecklingen av servern och klienten. Eftersom det var flera som utövade detta under olika omständigheter har vi valt att beskriva hur de olika

parterna upplevt hur det har varit att använda sig utav parprogrammering utifrån deras situation.

En fördel som de uppmärksammade är att man genom detta kan producera bättre kod på ett mer effektivt sätt än om båda hade suttit och kodade var för sig, förutsatt att man inte redan i förväg har bestämt i detalj hur koden ska se ut och vara strukturerad, något de inte hade gjort. Detta tack vare möjligheten att diskutera koden samtidigt som den skrivs, vilket också minskar svårigheten för att tackla nya problem som uppstår, till exempel om det är någon ny funktion som ska läggas till.

Eftersom båda personerna arbetar med samma kod minskas även “truckfaktorn”. Skulle en person till exempel bli sjuk är det inga som helst problem för den andra personen att fortsätta arbetet på egen hand. Dessutom får koden en viss standardstruktur utan att man behövt bestämma en sådan i förhand.

Den allra största nackdel som kunde hittas är att parprogrammering möjligen är mindre effektivt i den aspekt att man inte producerar lika mycket kod som man hade gjort om man arbetat separat. Däremot kräver ett sådant arbetssätt antingen att man jobbar på större separata delar, eller att man redan innan till viss del har bestämt hur allting bör sitta ihop. Det kräver också ett visst beslutsfattande i frågor där man har flera likvärdiga lösningar på problem, eller när man gör en temporär lösning för att återkomma till det senare - ibland är ett dåligt beslut bättre än inget beslut.

Hur effektivt det egentligen är med parprogrammering kan diskuteras, men som nämnt kan det ge en mer välskriven kod med mindre risk för att refaktoreras än om man arbetar skilt, vilket är den definition av “effektivitet” som vi har.

I framtida projekt skulle vi främst nyttja parprogrammering i fall där man arbetar med delar av projektet som ligger väldigt nära varandra, eller som kan vara svårt att dela upp överhuvudtaget. När det är möjligt att dela upp ett projekt i större, mer avgränsade delar föredrar vi dock att inte använda oss utav parprogrammering.

Fördelen med tekniken var att det av uppenbara skäl blev mycket lättare att anpassa den ena personens kod till den andras när båda var närvarande. En nackdel var att det trots allt blev en hel del komplikationer och mycket tid fick läggas på felsökning, men där kan man nog inte helt lägga skulden på tekniken utan man kanske istället skulle arbetat exklusivt på detta sätt direkt från början.

1.4 Stand-up Meetings

Under projektets gång hade vi så kallade stand-up-möten där vi på några minuter gick igenom vad varje gruppmedlem hade åstadkommit och eventuella problem som måste lösas. Fördelarna med detta var att man fick en bra överblick av hur långt projektet i helhet, och varje individuell del, hade utvecklats, även om man arbetat nästan helt separat. Skulle ett problem uppkomma upptäcks detta relativt snabbt, och om detta skulle hindra utvecklingen går det snabbt att se till att detta löses.

Med tanke på att ett stand-up-möte enbart tar några få minuter att genomföra är det en väldigt effektiv metod man kan använda för att ge alla inom gruppen en klarar bild på hur utvecklingen av produkten går. Med anledning av detta har vi också svårt för att se någon gång det skulle tänkas vara negativt att använda sig av stand-up-möten.

1.5 Möten

Under kursens gång hölls helgruppsmöten en till två gånger i veckan. Då gruppen delat upp arbetet i två mindre arbetsgrupper, server och klient, användes dessa möten för att hålla varandra uppdaterade om varandras åtaganden. Tekniska lösningar både inom och mellan grupperna diskuterades, eftersom det var viktigt att kommunikationen mellan server- och klientdelen av applikationen fungerade. Även det faktum att ha tillgång till en whiteboard gav mycket när det gällde att skissa på idéer och få översikt. Ett typiskt möte bestod i en halvtimmes diskussion som berörde hur klient och server borde se ut för att bäst kunna samspela, och sedan ytterligare en halvtimme där de två grupperna planerade mer inom sin egen grupp. Detta fungerade bra, särskilt eftersom de inte hölls för ofta, så det fanns alltid något nytt att gå igenom. Däremot användes aldrig konceptet med en “produktägarhatt”, dvs att någon gruppmedlem antog rollen som produktägare och ställde frågor utifrån det perspektivet, vilket var tanken från början.

2. Arbeta som grupp

(Svara på fråga 2, 6 och 9)

Alla i gruppen är överens om att man i snitt lagt cirka 25-30 timmar per vecka.

Namn	Möten	Tävling	Föreläsning	Kodning	Inlärning	Dokumentation	Total
Daniel	25	10	13	150	25	25	261
Johan	25	15	17	165	35	5	262
Michel	25	15	10	150	30	15	245
Alexander	25	10	17	170	25	5	252
Kristoffer	25	15	10	150	20	25	245
Total	**125**	**65**	**67**	**785**	**135**	**75**	**1265**

Vem som fick gå på vilken workshop bestämdes i princip genom att varje gruppmedlem sa vilken/vilka workshops de helst ville gå på. Det råkade bli så att de flesta hade ganska olika önskemål så alla fick som de ville.

Initialt var planen att medverka på alla workshops, men det blev så att två skippades. Dessa två tyckte gruppen både att tiden skulle vara bättre investerad på att arbeta med själva projektet samt att temat för workshopsen kändes irrelevant för hur projektet hade utvecklats vid den tidpunkten.

Workshopsen gruppen medverkade i var “24h med bussen” samt “Från koncept till affär”. Just “24h med bussen” visade sig vara intressant men tyvärr inte ge något som kunde appliceras på projektet.Gällande “Från koncept till affär” så var även det intressant och vi lärde oss mycket om hur man säljer och skyddar en idé eller ett koncept. Men just för detta projekt var det väl snarast tips om hur vi skulle presentera vår applikation inför juryn på tävlingen.

3. Reflektion på projekt

(Svara på fråga 4, 5, 7 och 8)

3.1 Vad fungerade under projektet?

I början av projektet delades arbetet upp i tre separata grundkomponenter, något som tycktes falla sig naturligt i detta fall; server, klientlogik och GUI. Detta ledde till att man fördelaktigt kunde arbeta på de olika komponenterna utan att behöva ta hänsyn till vad de andra delarna höll på med. Alltså kunde alla jobba på i sitt egna tempo och behövde aldrig stanna för att vänta in någon annan dels nya kod för att läggas in och byggas ihop med resterande delar för en ny version av prototypen.

Tidigt under kursen fick vi tips på en föreläsning att ta planera in tid för att lära oss arbeta med och bli bekväma med nytt material. Denna gång var det för vår del i form av Androids API och WebSocket. Att faktiskt planera in tid för inlärning tog bort en stor del av stressfaktorn med att behöva lära sig nytt material innan man började jobba, då man fick känslan av att bidra till projektet även om man inte skrev en enda rad kod. I efterhand inser man att detta var mycket värdefullt då man inte gått in i varje liten fälla man annars troligtvis hade.

Att undgå att gå i fällor är generellt en bra idé, och att organisera sig i början med User-Stories översatt till små munsbitar av arbetsuppgifter gjorde att man också fick en mycket klar bild inom gruppen för vad som skulle utvecklas. Detta tar inte särskilt lång tid och potentiell nytta är Mycket stor vilket gör det till en ovärderlig del av många, om inte alla, framtida projekt.

.

3.2 Vad fungerade sämre?

Som det nämndes i avsnitt *3.1* delades arbetet upp i mindre komponenter. Dessa komponenter slogs dock inte samman förrän väldigt sent i projektet, något som egentligen bör ske i ett tidigt stadie när man använder sig utav agil systemutveckling. Detta ledde till att det krävdes en hel del arbete när komponenterna väl skulle slås ihop, som annars hade delats upp i mindre omgångar. Ännu viktigare är att vi av denna anledning inte lägger fokus på att ha en produktprototyp tillgänglig så tidigt som möjligt. Vi tog inte hänsyn till att en eventuell produktägare bör ha möjligheten att utifrån en tidig prototyp kunna ändra eller lägga till saker som hen vill att produkten ska innehålla.

Ett annat problem med att sent koppla samman de olika delarna var att man då kan stöta på komplikationer, vilket vi även gjorde även om det var i en mild form. Faktum kvarstår dock att i ett annat projekt bör man se till att minimera denna risk genom att tidigt få ut en prototyp och inte riskerar att fastna med skägget i brevlådan.

Uppdelningen ledde även till att gruppen förlitade sig onödigt mycket på att varje gruppmedlem skulle vara frisk och kunna koda kontinuerligt, och en sjukdom skulle vara en stor förlust för gruppen, en så kallad hög *truck-factor*.

3.3 Lärdomar inför kommande projekt

Det allra viktigaste vi tar med oss till framtida projekt är att se till att så tidigt som möjligt få ihop något som går att visa för en produktägare, och stegvis arbeta vidare med de olika komponenterna, samtidigt som de hålls i fas.

I fallet med servern och klienten gjordes de allra flesta funktioner, som att skicka meddelande och gå med i chattrum, möjliga innan klienten sattes upp för att kunna kommunicera med servern. Istället borde vi skapat ett slags “skelett” för både servern och klienten, kopplat samman dessa, och betat av user stories i samma takt. När det görs möjligt att skapa en användare på servern, görs det samtidigt möjligt att på klienten skapa en användare. Sedan hade man gått vidare till att skapa chattrum på servern, och sett till att användaren kan gå med i ett chattrum, innan man fortsätter med nästa user story.

Något som möjligen bidrog med att sammankopplingen skedde så pass sent som den gjorde är att vi saknade någon som agerade produktägare. Hade vi haft någon eller några som då och då agerade produktägare hade vi kanske insett att vi behövde få ihop en tidig prototyp för att det skulle vara möjligt för produktägaren att ha något att se.

I början av projektet bestämde vi att vi under möten skulle hålla alla gruppmedlemmar uppdaterade med hur koden såg ut inom de olika delarna av projektet, och på så sätt minska truckfaktorn. Tyvärr glömdes detta av med projektets gång, och även om vi lyckligtvis aldrig direkt påverkades av detta inser vi att vi bör ha fullföljt detta då vi annars hade kunnat få stora problem, särskilt framåt slutet.