Reflektionsrapport – Magic Bus

CrispIT

Andreas Andersson

Gustav Blide

Felix Ehrnberg

Fredrik Hansson

Måns Löfås

Mikael Sjöblom

Innehåll

Förklaring till färger i höger marginal:

- Application of scrum
- Reflection on the sprint retrospectives
- Reflection on the sprint reviews
- Best practices for new tools and technologies
- Reflection on the relationship between *prototype*, *process* and *stakeholder value*

1 Introduktion

I juni 2015 började Buss 55 gå mellan Chalmers Johanneberg och Lindholmen Science Park. Buss 55 utvecklades av Volvo som ett konceptfordon med syftet att testa ny miljövänlig teknik för att öka kollektivtrafikens attraktionskraft. För att höja passagerarupplevelsen är bussarna utrustade med eluttag och wi-fi. Detta gör att busschaufförerna är uppkopplade mot Internet och på så vis kan överföra information om bussen till de som behöver den. De intressenter som främst har nytta av att kunna ta emot information från bussarna är i nuläget trafikledningen, som hanteras av Keolis, samt Volvos serviceavdelning för Electricity-bussarna.

1.1 Bakgrund

Utifrån ett besök på Volvo Bus Experience Center och en diskussion med representanter från Keolis och Volvo identifierades ett problem i kommunikationen mellan busschaufförer och resterande parter. I dagsläget sker all problemrapportering från bussarna via telefon till trafikledningen. I de fall bussar behöver repareras vidarebefordrar trafikledningen informationen till servicecentret. Problemet med dagens upplägg uppges delvis vara att busschaufförerna inte har möjlighet att kontakta trafikledningen under färd utan måste vänta till en ändhållplats. Detta innebär en fördröjning mellan att ett fel uppkommer och den tidpunkt då det rapporteras in. Vidare betraktas förmågan till verbal kommunikation mellan busschaufför och trafikledning vara begränsad till följd av språkliga hinder. På grund av att information flödar i flera led för att nå servicecenter får dessa inte alltid en korrekt bild av situationen.

1.2 Problemformulering

Med utgångspunkt i bakgrunden togs beslutet att ta fram en lösning för att underlätta kommunikation mellan de olika parterna och öka transparensen mellan trafikledning och servicecenter. Lösningen är tänkt att bestå av två olika applikationer varav den ena kan skicka felrapporter (felrapportering) och den andra kan ta emot och behandla felrapporter (informationshantering). Målet är att en konstruera applikationen för felrapportering på ett sätt som tillåter busschauffören att skicka en rapport i samband med stopp vid hållplats. Applikationen som tar emot information är ämnad att nyttjas av trafikledning såväl som servicecenter både vid planering och för framtida förebyggande underhåll samt felsökning.

1.3 Metod

Hela projektet har utförts i enlighet med Scrum-metodologin. Scrum grundar sig i en tro på fördelarna med att jobba i ett lag (Scrumguides, 2016). Tanken är att laget presterar som bäst om de blir tilldelade mål snarare än specifika arbetsuppgifter och därmed har möjlighet att arbeta flexibelt inom satta ramar. Pichler (2010) beskriver Scrum som en dynamisk arbetsprocess där nya funktionaliteter arbetas fram genom kundåterkoppling. Scrum är en agil arbetsmetod där en produkt utvecklas stegvis i så kallade sprintar. Sims och Johnson (2012) menar att Scrum metodologin identifierar tre primära roller i form av produktägare, Scrum master och lagmedlemmar. Arbetet avvek till viss del från konventionella mjukvaruutvecklingsprojekt då det utfördes som en del av en kurs och i brist på en direkt överordnad produktägare betraktades Keolis och Volvo gemensamt som en produktägare.

En viktig princip i Extreme programming (XP), en agil arbetsmetodik inom mjukaruutveckling, är enkel design. Enkel design syftar till att tillräckligt utformning genomförs för att möta dagens krav och inte mer (Sommerville, 2011). Denna princip har präglat gruppen genom hela processen. Dagens krav hos produktägaren betraktades av gruppen som någonting som huvudsakligen prioriterar en funktionell lösning framför ett noggrant genomarbetat grafiskt användarsnitt (exempelvis färgtema eller ikoner). Av denna anledning lades större fokus vid att utveckla en applikation som fyllde en viktig funktion, felrapportering. Samtidigt följdes även principen om inkrementel design, det vill säga att utformandet av applikationen hålls enkelt från början och förbättras kontinuerligt med små förändringar (Kniberg, 2015).

2 Sprintar

Varje sprint inleddes med ett sprintmöte som förlades på måndag förmiddag och varade i en och en halv timme. Under sprintmötena deltog samtliga gruppmedlemmar. Syftet med dessa möten var att planera näskommande sprint. Tillfällena inleddes med en diskussion om nytillkomna user storys samt en potentiell omprioritering av de user storys som låg i projektets produktbacklog. Här togs även beslut om effort och velocity. Utifrån prioriteringen och förhållandet mellan effort och velocity placerades user storys in i nästkommande sprints backlog.

Inför varje sprint satte gruppen en lista på de aktiviteter som innefattades i varje user story och var menade att skapa det verifierbara värdet i applikationen som user storyn speglade, någonting som Panchal (2008) beskriver som *definition of done*. För att i slutet av varje sprint sedan avgöra om en user story färdigställts ur ett kvalitetsperspektiv använde sig gruppen åter av denna lista. En user story ansågs inte vara avklarad enbart för att rätt funktion utvecklats. Utöver funktion ställdes genomgående krav på att kodstrukturen möjliggjorde integration med övriga delar av projektet, att koden klarade av tester, att koden var kommenterad samt att det grafiska användargränssnittet motsvarade framtagna kravspecifikationer. För att denna metod skulle fungera ställdes alltså höga krav på de riktlinjer som togs fram på sprintplaneringsmötena. Det ledde dock till att mindre tid behövde läggas på att anpassa olika komponenter till varandra.

Innehållet i sprintens backlog delades upp mellan tre lag om två personer som sedan ansvarade för att arbetet blev utfört. Lagen ansvarade sedan för den tilldelade uppgiften under hela sprinten. Efter varje sprint hölls en *sprint review* där färdiga produkter visades upp. Avslutningsvis behandlades kvalitén hos processen genom en så kallad *sprint retrospective*.

2.1 Effort och Velocity

Effort är ett sätt att mäta de relativa skillnaderna i ansträngning och storlek av uppgifter (Pichler, 2010). Från början var tanken att använda avrundade Fibonaccital vid tilldelning av effort för varje user story. Dock kände vi ganska snabbt att det var lättare att använda en egen lite modifierad skala för att lättare kunna översätta det till arbetstimmar och planera våra sprintar. Därför har våra user storys tilldelats 10, 20, 30 eller 50 poäng, där varje poäng ska motsvara antalet arbetstimmar, en uppdelning som motiverades av det specifika fallets omfång och typ av user storys. Detta förfarande gjorde det tydligt för oss vid planering och samtidigt underlättade det uppföljning under veckan för att

se hur arbetet fortskred i förhållande till planen. Därför sattes också en fast velocity på 60 arbetsimmar per vecka, då det var den tid som enligt plan fanns tillägnad produktutveckling, se avsnitt 3.1 Tidsdisponering.

Eftersom velocityn var satt från början, var det effort som var den parameter vi kunde ändra på och experimentera med. Från det att gruppen började arbeta med en produktbacklog bestämdes en user storys effort utifrån magkänsla. Kniberg (2015) nämner detta som ett lämpligt sätt att estimera för mindre grupper och där sprintarna är korta, vilket stämmer överens med upplägget för detta projekt. Gruppen upplevde svårigheter i att avgöra effort för olika user storys vilket gav upphov till vissa problem vid sprintplanering och fördelning av arbete. Problemet härleddes främst till gruppens oförmåga att uppskatta komplexiteten hos koden som krävdes för att uppfylla olika user storys. I de första sprintarna resulterade detta i att vissa arbetsuppgifter slutfördes snabbare än beräknat medan andra tog längre tid. Att använda avrundade fibonaccital hade till viss del kunnat göra approximeringen lättare, då poängen kunnat efterlikna den faktiska tidsåtgången bättre. Däremot skulle fortfarande problemet av att approximera i okunskap om lösningen på user storyn finnas kvar. Efter första sprinten togs beslut om att ändra effort på några av de user storys som fanns i projektets produktbacklog. Detta i och med att gruppen insåg att inlärningskostnaden skulle bli mindre under andra sprinten och dessutom insett att komplexiteten av kodandet hade blivit större. Valet att ändra effort i stället för velocity var att några arbetsuppgifter, till följd av vad vi lärt oss, omvärderades i förhållande till varandra. Velocityn hölls intakt för att göra det enklare att jämföra olika sprinter med varandra och för att kunna behålla den tydliga velocityn uppdelad på arbetstimmar.

Omvärdering av effort för en user story gjordes i flera fall där det var nödvändigt vid sprintplanering och för att ytterligare förbättra vår uppskattning av effort utökades sprintplaneringsmötena inför de två sista sprintarna med en kort diskussion om varje user story och hur den skulle hanteras rent programmeringstekniskt. Då framkom det tidigt huruvida det fanns uppenbara lösningar eller om uppgiften skulle kräva fördjupning och därmed ta längre tid.

2.2 Sprint reviews

Arbetsprocessens veckolånga sprintar har alla avslutats med bland annat en sprint review, som enligt Kniberg (2015) är ett tillfälle där sprintlagen visar upp sina framsteg samt får respons från intressenter och övriga sprintlag. I enlighet med vad Pichler (2010) skriver skapade dessa möten en möjlighet för oss att reflektera över produktens utveckling och hur projektet bör fortgå. Som följd av reflektioner

mynnade varje sprint review delvis ut i förslag på nya user storys som placerades in i projektets produktbacklog. Dessa user storys diskuterades ej på djupet under sprint review tillfällena, utan i nästkommande sprintplaneringsmöte.

Pichler (2010) menar att en sprint review är ett tillfälle då produktägaren kan jämföra produkten mot user storys och definition of done. Att avgöra om produkten är helt färdig var av stor vikt då värdet av någonting som är halvgjort i en sprint, enligt Kniberg (2015), är noll. Inom gruppen hände det vid ett tillfälle att ett lag stod med en halvgjord uppgift då sprinten var slut. Detta hanterades då genom att denna user story återplacerades i projektets produktbacklog. Efter diskussion på nästkommande sprintplaneringsmöte beslutades att samma lag skulle fortsätta med denna user story med samma beräknat effort utan hänsyn till det halvfärdig arbete som gjorts.

På grund av att projektet utfördes som en del av en kurs och inte på uppdrag av en arbetsgivare fanns det en geografisk såväl som administrativ distans till de tilltänkta produktägarna, Keolis och Volvo, vilket ledde till att sprint review tillfällen ej kunde genomföras tillsammans med samtliga intressenter. Av denna anledning och som följd av den ytliga relationen till intressenter, var det arbetsgruppen som stod för stor en andel av framtagna user storys. Gruppen gick därmed miste om åsikter från de slutgiltiga användarna vilket anses vara en svaghet i projektets upplägg.

Ries (2011) menar att produktutveckling som ej bygger på kundåterkoppling leder till icke värdeskapande aktiviteter som borde undvikas. Då grundidén till projektet byggde på ett behov som identifierats i samråd med slutkunden betraktas dock det tilltänkta värdeerbjudandet som väl underbyggt. En mer kontinuerlig dialog, i form av sprint review tillsammans med övriga intressenter, hade dock kunnat leda till en process där en bredare bild av värdet för användarna kunnat uppmärksammas. Kniberg (2015) menar däremot att det, som gruppen gjorde, finns fördelar med att även använda sig av en intern sprint review där sprintlagen ger en djupare beskrivning av utmaningar och viktiga tekniska beslut. Detta då det kan vara ett bra tillfälle att sprida kunskap mellan sprintlagen. Sprint review-tillfällena anses därför ha bidragit positivt till processen trots frånvaro av bland annat produktägare.

I början av samtliga sprintar delade gruppen upp sig i tre olika lag. I slutet av varje sprint hölls sedan en intern sprint review där de olika lagen visades upp sina resultat i form av kod och funktion i själva applikationen. Under dessa tillfällen visades uppdateringar upp och de mest framstående besluten som tagits under arbetsgången förklarades och motiverades. Vid dessa tillfällen diskuterades även

huruvida user storys och definition of done var uppfyllda. Sims och Johnson (2012) menar dock att beslut om huruvida produkten uppfyller kravspecifikationer ej bör tas under sprint review möten utan snarare behandlas vid ett tidigare tillfälle. Valet att ändå göra detta byggde på att samtliga sprintar avslutades och påbörjades under måndagar och att det ansågs överflödigt att först ha en diskussion om sprintens backlog för att sedan ha en sprint review. Istället ansåg gruppen att sprint review genomgången skapade underlag för att ta beslut om huruvida user storys var uppfyllda eller inte. Detta underlättades även av att dessa möten endast involverade de personer som ansvarade för att ta beslut om user storys var uppfyllda eller ej.

2.3 Sprint retrospectives

Varje sprint review har följts av en sprint retrospective då gruppen ser tillbaka på och utvärderar processen under sprinten. Enligt Kniberg (2015) bör sprint retrospectives huvudsakligen lägga fokus på frågan "Vad kan vi göra bättre nästa sprint?" och det är med detta syfte som gruppens retrospectives har utförts genom hela projektet.

Under arbetets gång har sprint retrospectives utförts varje måndag i samband med sprint review möten och sprintplaneringsmöten. Våra retrospectives inkluderade de viktigaste punktena som Kniberg (2015) tar upp i form av sammanfattning av sprinten och förslag av scrum mastern, undersökning av velocity samt "Rounds". I våra "Rounds" fick varje medlem i gruppen berätta för gruppen vad i processen som har gått bra och mindre bra samt vad som de anser hade kunnat göras bättre i nästkommande sprintar. Efter att varje medlem ostört fått lägga fram sina tankar och funderingar diskuterade vi i grupp hur vi skulle fortsätta och tog konsensusbeslut om eventuella förändringar som skulle genomföras. Där konsensus inte nåtts har gruppen fört diskussioner. Processen var visserligen tidskrävande och i vissa fall behövde kompromisser göras, men genom att kontinuerligt och grundligt lösa alla problem och skillnader i åsikter tidigt har gruppen tillåtits arbeta mycket mer enat än om diskussionerna lämnats.

Via retrospectives lyckades gruppen förbättra sitt arbetssätt i stor utsträckning då eventuella problem som uppstått under veckan kunde diskuteras och förslag på förbättringar kunde delas med hela gruppen. I och med att vi inte hade några dagliga Scrummöten och att retrospectives utfördes först när sprinten var slut så löstes problem som uppstått under sprintens gång inte förrän sprinten var slut. Detta ledde i sin tur till att vissa sprintlag arbetade mindre effektivt under vissa sprintar. Enligt

Schwaber och Sutherland (2013) kan dagliga Scrummöten vara till hjälp för att undvika problem som nämns ovan och att implementera det inom gruppen hade således kunnat förbättra vår process.

Gruppen försökte vid ett tillfälle att införa dagliga Scrummöten för att kunna ta itu med problem som uppstod, men eftersom alla medlemmarna hade väldigt olika scheman var det svårt att genomföra och hålla dessa vid liv. Dessutom insåg gruppen att många av problemen som uppstod inte krävde hela gruppens kompetens och diskussioner kunder därför hindra processens fortlöpande. Med utgångspunkt i detta togs ett beslut att istället föra en öppen dialog i gruppens kommunikationskanaler vid de tillfällen då gruppmedlemmar behövde hjälp. Från och med sprint tre lades mer ansvar på de individuella sprintlagen genom att de tilläts ta egna beslut. Detta involverade beslut om egna tillägg, så länge det kunde motiveras för under nästkommande sprint review. Ökad kommunikationen mellan lagen hjälpte även att undvika överlappande arbete. Använda kommunikationskanalerna var främst textmeddelandetjänster där regelbundna uppdateringar om framsteg förmedlades. Utöver detta så sattes dessutom ett extra veckomöte in för att kunna synkronisera gruppen och ta beslut om det fortsatta arbetet.

Den ökade kommunikationen och sprintlagens förmåga att lösa problem på egen hand och kommunicera dessa var till enorm nytta de sista två sprintarna. Att ha en mer öppen kommunikation och sprintlag med större eget ansvar gjorde att alla lagen lyckades jobba mer effektivt och lösa problem som uppstod utan att behöva samla hela gruppen.

2.4 Process

Projektet har bestått i fem sprintar som varat en vecka vardera. Inför varje sprint har user storys flyttats till en sprintbacklog för att sedan fördelas mellan tre olika sprintlag om två personer. Gruppens mål var att, efter varje sprint, kunna visa upp framstegen för produktägarna på ett sätt som tydligt visade att projektet rörde sig i rätt riktning. Därför användes vertikala user storys som enligt Burden[1] innebär att ett mindre antal user storys behandlas på samtliga nivårer så som databas, grundläggande kod och grafiskt användargränssnitt. Tanken bakom detta var att det skulle bli lättare för både övriga gruppmedlemmar och övriga intressenter att se konkret bevis på framstegen som gjorts, i form av en applikation och inte endast av till exempel en databas vars funktionalitet är mer svårvisualiserad. De user storys som behandlats under sprint ett till sprint fem finns redovisade i bilaga 1.

[1] Håkan Burden (Universitetslektor, Chalmers tekniska högskola) Föreläsning den 11 april 2016.

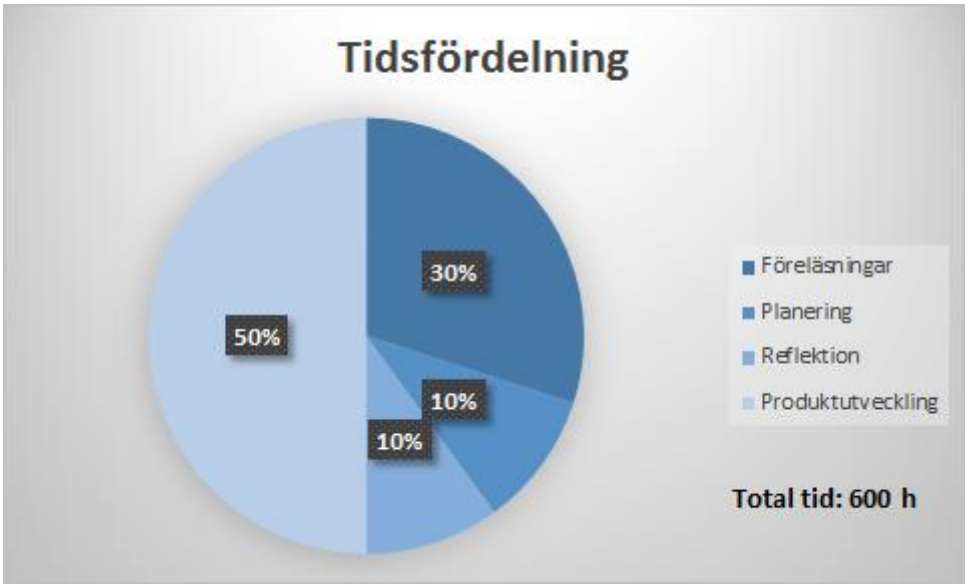
3 Arbetsfördelning

I inledningsfasen av projektet upprättades ett socialt kontrakt mellan samtliga gruppmedlemmar i syfte att skapa samförstånd om projektets roller och ansvarsfördelning. Det har under projektets gång aldrig uppstått en situation där en gruppmedlem ansetts ha brutit mot kontraktet. Huruvida detta berodde på kontraktets existens eller en allmänt god arbetsmoral är svårt att avgöra. Däremot gjorde upprättandet av det sociala kontraktet att gruppen tidigt enades om höga mål i projektet, därför rådde aldrig något tvivel om engagemang.

Inom Scrum är "Scrum master" den som är ansvarig för att metodiken Scrum förstås och efterföljs (Schwaber & Sutherland, 2014). Tidigt i arbetet tog en av medlemmarna på sig ansvaret att vara Scrum master inom gruppen. I denna roll höll han bland annat koll på den backlog vi hade inom gruppen, drev diskussionen av både sprint retrospectives och sprint reviews samt hade koll på att möten gick framåt. Då idéen bakom Scrum enligt Sommerville (2011) är att hela gruppen ska känna sig befogade att göra beslut så däremot fördelades övriga ansvar, som normalt tillskrivs en Scrum master, inom gruppen.

En person fick ansvaret att sköta den huvudsakliga kontakten med produktägare och övriga intressenter utanför teamet. Ytterligare en person hade förkunskaper inom områdena Android Studio och Trello och blev därför ansvarig för att dessa plattformer utnyttjades på bästa sätt. För att ingen uppgift skulle underprioriteras ansågs det väldigt viktigt att samtliga ansvar fördelades och tillägnades någon i gruppen. Avsaknad av en formell ledare kan ha begränsat den positiva effekt som Scrum master rollen är menad att tillföra. Däremot valdes vårt upplägg utifrån gruppens storlek och med hänseende till att alla besatt olika kompetenser som inte återfanns individuellt samlade i en individ. Ingen i gruppen var heller certifierad Scrum master.

3.1 Tidsdisponering



Figur 1. Tidsfördelning mellan aktiviteter under projektet.

Projektet varade i fem veckor och arbetsveckorna bestod i tjugo timmar nedlagd tid av varje gruppmedlem. Tiden var tänkt att fördelas mellan föreläsningar, sprintplanering, produktutveckling samt reflektion i enlighet med Figur 1. På grund av sjukdom och parallella skolaktiviteter, såsom förpliktelser i kandidatarbeten, uppstod vissa individuella avvikelser från denna plan. Detta var bland annat en följd av parprogrammingsmetodiken, som gjorde gruppmedlemmarna beroende av varandras scheman. Fördelningen av nedlagd tid uppskattas ha sett liknande ut för samtliga gruppmedlemmar. Under föreläsningar, sprintplanering och reflektion deltog, då det var möjligt, samtliga gruppmedlemmar. Under produktutvecklingstimmarna delades gruppen upp i mindre sprintlag.

3.2 Parprogrammering

Med hjälp av parprogrammering kunde arbetet genomföras mer effektivt och med djupare samarbete, där alla medlemmar blev delaktiga i arbetets process. Fler uppgifter kunde skötas samtidigt på olika håll och med färre personer involverade, till skillnad från om hela gruppen suttit tillsammans. Från tidigare erfarenheter av grupparbete ansåg gruppen att de flesta uppgifter går att lösa av endast två personer. Parprogrammering visades sig vara ett lämpligt tillvägagångssätt vid exempelvis utveckling av databasen, då det endast krävdes att ett par personer byggde upp kunskaperna kring hur uppgiften skulle lösas. Paren fokuserade således under sin tid under varje sprint

på att gå djupare i olika uppgifter. Eftersom att paren själva sågs nästan varje dag så fördes en daglig kommunikation åtminstone med en del av gruppen och behovet av dagliga möten med hela gruppen blev mindre nödvändigt.

Med hjälp av parprogrammering kunde kunskaper kompletteras, till skillnad mot om personer jobbat individuellt. Då samtliga medlemmar kom in i gruppen med olika kunskaper så gjorde ett närmare samarbete inom lag av två att kunskapen lättare kunde spridas internt. För att kunskapen inom varje lag inte skulle stanna där samt för att varje medlem skulle få chansen att utvecklas i samarbete med fler personer så bytte gruppen lag efter varje sprint.

Många user storys kunde kopplas till vad som gjorts inom föregående sprintar. Av denna anledning passade det bra att en person från varje lag fortsatte inom det område denna arbetat på. Den person från varje par som inte stannade bytte till att istället arbeta på någon ny aspekt av applikationen efter varje sprint. I linje med vad Öhman[2] berättade om varför Spotifys utvecklare själva får ansvara för underhållet av sin egen kod så resonerade vi att detta skulle skapa större ägandekänsla och bredare engagemang runt kodens användande. För den person i laget som stannade så blev koden den precis skrivit omhändertagen, inte enbart i underhållssyfte utan även i syfte att vidare använda och utveckla den.

Att gruppen förändrade uppsättningen av lagen ställde krav på att alla inom gruppen kom överens. Detta visade sig väldigt snabbt när personer som var mindre nära varandra fick ingå i ett par. Däremot ställde detta aldrig till några större problem, då vi förde öppna dialoger. De konflikter som uppstod löstes genom att ha en diskussion i hela gruppen. På så sätt kunde vi tillsammans försöka komma överens om gruppens åsikter i frågan. I och med att vi skiftade uppsättning av lagen så krävdes det även att alla kompletterade varandra i processen. Detta underlättades av att vi alla var relativt nya i umgänge med varandra och även hade väldigt olika förkunskaper.

Stundtals i utvecklingen bestod arbetet endast av att skriva kod utan större krav på efterforskning eller diskussion. Detta uppfattades då som en uppgift som endast krävde en person vilket gjorde den andra lagmedlemmen överflödig. Detta gav inte några förödande effekter, eftersom denna situation endast uppkom när laget hade kommit på en fullständig lösning tillsammans och således visste vad de skulle göra. Det som istället hände då var att den andre personen, som inte kodade, tog ansvaret att se över

[2] Michael Öhman (Spotify) Föreläsning den 13 maj 2016.

hur funktionen skulle representeras eller hur applikationens övriga grafiska användargränssnitt kunde förbättras. Detta då det prioriterades efter funktion.

4 Relationer med aktörer

En mycket stor utmaning som gruppen stått inför under hela projektets gång har varit avsaknaden av en konventionell produktägare. För att möta denna utmaning lades stor ansträngning i arbetets tidiga skede på att fastställa ett tydligt värdeerbjudande utifrån situationen som observerades vid projektets början och de intervjuer som utfördes då. För att se till att värdeerbjudandet varit relevant och efterfrågat av de olika parterna har gruppen kontaktat serviceavdelningen och trafikledningen för feedback samt testat applikationen mot busschaufförerna.

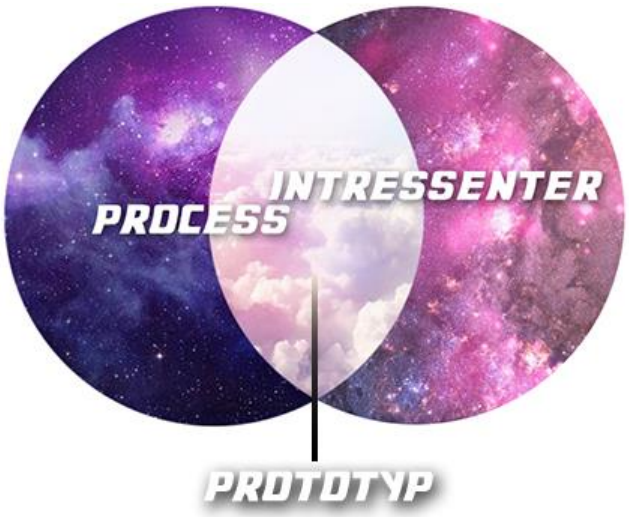
När viktiga beslut skulle tas i prototypens utformning diskuterades dessa i gruppen för att se till att besluten skapade rätt värde. En återkommande risk var att individuella gruppmedlemmar tog beslut som inte låg i linje med det tilltänkta värdeerbjudandet. Därför var vi noggranna med att alla beslut som låg utanför ramarna av en user story togs gemensamt i grupp. Detta fick dock en viss effekt på flexibiliteten då många beslut försköts till nästa gruppmöte vilka endast skedde en gång i veckan. I de sista sprintarna, när sprintlagen gavs större befogenheter att ta beslut utan konsensus, var det ytterst viktigt med kommunikation för att inte individuella lag skulle ta beslut som minskade prototoypens värde.

Alla tester som utförts har haft de grundläggande kundvärdena i åtanke och endast om ändringar resulterat i ökat kundvärde har förändringar implementerats. Applikationen testades grundligt vid varje ny implementering av nya funktioner eller ändringar i koden. Detta för att kontrollera att de nya förändringarna faktiskt gav det kundvärdet som efterfrågades samt för att ge sprintlagen en möjlighet att pausa i arbetet och analysera huruvida processen speglat det värde som efterfrågats av kunden.

Enligt Lantz[3] så är utvecklare inom Volvo väl medvetna om att produktspecifikationer och mål kan komma att ändras under projektens gång. Därför har de en öppen syn på de beslut som fattas och tillåter, om det är nödvändigt, ändringar av tidigare beslut. Detta anammades av gruppen genom att tillåta ändringar, även om konsensusbeslut tagits om någonting, om en ändring skulle innebära ökat kundvärde. Ett exempel på detta var att gruppen vid ett tillfälle i processen tog beslutet att tillverka två separata applikationer för trafikledningen och servicecenter för att öka anpassninsgraden. Baserat

[3] Jonn Lantz (VCC) Föreläsning den 25 april 2016.

på feedback från produktägaren togs dock, i nästkommande sprint, ett nytt beslut om att återgå till den ursprungliga planen om att endast tillverka en gemensam applikation för informationshantering.



Figur 2. Beskrivande figur över samspelet mellan process, protoyp och intressenter.

Gruppens uppfattning om samspelet mellan prototyp, process och intressenter kan liknas med Figur 2 ovan. Endast med en djup förståelse av intressenter skapas en väl utformad prototyp. Alla aktiviteter som utförts i processen har gjorts i syfte av att skapa värde för kunden i slutändan. Genom kontinuerliga testningar och demonstrationer av det som skapas med en protoyp, kan det säkerställas att värde skapas för kunden. I de fall då prototypen avviker mot kundens förväntningar måste prototypen ändras och eventuellt även processen för att i framtiden tidigare kunna säkerställa att rätt värde skapas från början. Likt vad Brunnegård[4] föreläste om så ansåg gruppen att det var viktigt att jobba i korta cykler och stämma av kontinuerligt mot kund, för att kunna göra förändringar i både arbetssätt och produkt på ett sätt som skapar värde.

I applikationen för felrapportering togs ett stort beslut att ta bort möjligheten att radera skickade felrapporter, eftersom trafikledningen uttryckt i samtal att det är väldigt sällan som busschaufförer vill ta bort en skickad rapport. Möjligheten finns istället att busschaufförerna kan kommentera rapporten utifall problemet har lösts på egen hand. Återkoppling från busschaufförer visade vidare att en full historik av alla rapporter de skickat som är åtgärdade inte var önskvärt. Den funktionaliteten ersattes i felrapporteringsapplikationen med att enbart visa rapporter som inte åtgärdats.

[4] Viktor Brunnegård (Chalmers Ventures) Föreläsning den 27 april 2016.

Layouten på applikationen för felrapportering gjordes om i stor utsträckning efter att ha undersökt hur busschaufförer använde den. Dels genom att de olika stegen som busschauffören behövde ta för att färdigställa en felrapport tydliggjordes med siffror på vänster sida, men även genom att inte använda ord som kunde vara svåra att förstå för chaufförerna med begränsade språkkunskaper i svenska. Förändringarna gjordes efter det iakttagits att busschaufförerna vid användning inte visste vilken knapp de skulle börja klicka på och att ordet "symptom" inte var helt lätt att förstå för dem samt att det var tvetydigt vad det syftade på.

Vid presentationen av applikationen påpekades ett fåtal områden där mer arbete hade varit önskvärt för att efterleva kundens önskemål. Dessa områden fokuserade dock huvudsakligen på det grafiska användargränssnittet (GUI) vilket var förväntat eftersom gruppen, som tidigare nämnt, prioriterat funktionalitet. Det grafiska användargränssnittet var utformat att vara så användarvänligt som möjligt och använde därför inte avancerade grafiska verktyg. Under demopresentationen visade det sig att personerna som fick testa applikationen snabbt kunde orientera sig genom processen, vilket bekräftade att användarvänlighet uppfyllts. För mer erfarna android-användare kunde däremot enkelheten te sig svår i det att de förväntade sig mer grafiska verktyg än vad som erbjöds.

På startskärmen i applikationen för informationshantering hade knapparna kunnat fylla upp skärmen, främst i bredd, då det inte fanns någonting annat i designen som hindrade detta. I enlighet med vad Sjölie[5] påstått hade detta uppfyllt användarvänligheten bättre. I applikationen för felrapportering fanns det olika betydelser för färgerna på två olika delar, vilket kan uppfattas som missvisande. Detta ansågs i efterhand vara en följd av att gruppen fokuserat väldigt mycket på att göra varje delfunktion logisk och därav inte sett över den logiska samhörighet som bör finnas mellan olika funktioner i applikationen. I övrigt anses listan över möjliga fel som användaren kan välja mellan varit för lång. Detta kan göra att tiden för en ovan användare att gå igenom felrapporterings-processen ökar. Eftersom att applikationen är utformad för att användaren ska komplettera rapporten i efterhand så kom gruppen på att detta problem kunnat lösas genom att huvudkategori på problemet väljs i första skedet och specifieringen av det senare. Detta skulle minska antalet alternativ vid första valet radikalt.

[5] Daniel Sjölie (Universitetslektor, Chalmers tekniska högskola) Föreläsning den 29 april 2016.

5 Verktyg

Tre primära vektyg har utnyttjats i detta arbete. All kodning utfördes i Android Studio som valdes på grund av att programmet är anpassat för utveckling av androidapplikationer. För att samordna kodningen och möjliggöra en sammansättning av kodsegment som producerats på olika enheter användes GitHub. Produktbacklog, sprintbacklog samt färdigställda user storys visualiserades i programmet Trello. Här publicerades även övrig administrativ information som kunde kopplas till arbetsprocessen.

Android Studio

Valet av Android Studio som utvecklingsmiljö byggde på dess höga anpassningsgrad för utveckling av Android applikationer. Endast en av gruppmedlemmarna hade förkunskaper om programmet vilket innebar att denne person fick ansvar att hjälpa övriga gruppmedlemmar att förstå grunderna i programmet. Nackdelen med detta var att personen i fråga inledningsvis fick mindre tid över till produktutveckling. Gruppen uppfattade det dock som mer tidseffektivt att övriga gruppmedlemmar slapp söka efter information på egen hand. I senare delar av projektet skedde all informationssökning om användning av Android Studio via internet.

GitHub

På rekommendation av kursansvarig användes GitHub, ett webbhotell, som plattform för lagring av mjukvara och versionshantering. Ingen i gruppen hade tidigare erfarenhet av att använda GitHub som ett verktyg vilken innebar att det krävdes en viss inlärningstid. Användningen av GitHub blev således mer effektiv under arbetets gång. Exempel på problem som uppstod i uppstartsfasen var att källkod lagrades i olika projekt vilket försvårade sammanslagning av kod. Vidare uppstod ett problem med att namn på klasser förändrades vid hämtning av kod. Först i slutskedet gjordes upptäckten att detta berodde på en felaktig namngivning i Git.

Då gruppens arbetsmetod byggde på att olika sprintlag utvecklade olika delar av produkten fanns ett behov av ett forum för att dela kod. Ny kod laddades endast upp då en ny funktion hade skapats och saknade kompileringsfel. Tanken bakom detta var att de andra gruppmedlemmarna inte skulle kunna ladda ner projekt som inte gick att köra. Baksidan med denna strategi var att den interna transparensen minskar och arbetet blir mer individuellt vilket inte går i linje med scrum-metodikens fokus på lagarbete.

Trello

Det webbaserade programmet Trello användes för att visualisera de deluppgifter som projektet bestod i. Här publicerades även övrig information som ansågs vara relevant för hela gruppen. Trello användes för att skapa en sorts kanban-struktur där produktbacklog, sprintbacklog och avklarade user storys redovisades. Inför varje sprint flyttades user storys från produktbacklog till sprintbacklog. När en user story uppfylldes enligt definition of done sorterades den som färdig. Funktionerna i Trello gjorde det även möjligt att visuellt bryta ner varje user story i deluppgifter vilket var till stor nytta vid planering av sprinter.

Användning av Trello skapade struktur och gjorde det möjligt för samtliga gruppmedlemmar att följa arbetets gång. Projektets produktbacklog kunde enkelt uppdateras med nya user storys vilket innebar att nya idéer kunde samlas på samma ställe för att sedan diskuteras. Vi hade en person som var ansvarig för att föra över färdiga user storys. Syftet bakom detta vara att ha stor kontroll på vilka delar som ansågs avklarade för att på så vis undvika att halvfärdiga projekt kategoriserades som avklarade.

6 Referenslista

Kniberg, H. (2015) *Scrum And XP From the Trenches.* C4Media.

Pachal, D. (2008) What is Definition of Done (DoD)?. *Scrum Alliance.* *https://www.scrumalliance.org/community/articles/2008/september/what-is-definition-of-done-(dod) (2016-05-26)*

Pichler, R. (2010) *Agile Product Management With Scrum.* Boston: Pearson Education Inc.

Ries, E. (2011) *The Lean Startup*. UK: Portfolio Penguin.

Schwaber, K. & Sutherland, J. (2013) *The Scrum Guide*. Hämtad: http://www.scrumguides.org/download.html (2016-05-28)

Sims, C. & Johnson, H.L. (2012) *Scrum: A Breathtakingly Brief and Agile Introduction.* [Elektronisk]. Dymaxicon.

Sommerville, I. (2011) *Software Engineering*. Boston: Pearson Education Inc.

7 Bilagor

Bilaga 1 - user storys:

De user storys som har använts under projektets gång finns redovisade i tabellen nedan.

ID	User Story	Effort	Acceptanskriterier	Kommentarer	Efterfrågare	Sprintnr
1	Skicka felrapport	50	Kunna skicka en felrapport, under 30 sekunder och med max 4 tryck.	Felrapporten ska sparas i en databas när den skickas. För att busschaufförен ska slippa vänta på att felrapporten skickas, kan en preliminär rapport skickas som kompletteras senare.	Busschaufför	1
2	Läsa felrapport	50	Kunna läsa en felrapport som inkommit, innehållande bussid, symptom, kommentar, gradering, status och all tillgänglig data från bussen.	Alla tillgänglig data från bussen ska sparas med felrapporten vid tillfället då den skickas. Det kan göras med hjälp av plattformen.	Trafikledning/Service	2
3	Kompletter a rapport	30	Kunna vid senare skede lägga till kommentar, ändra	Uppdatering ska endast kunna göras om något	Busschaufför	3

			gradering och symptom till felrapporten.	förändrats.		
4	Se skickade rapporter	20	Kunna se tidigare skickade rapporter för aktuell buss som inte är lösta och se om de måste kompletteras med kommentar.	En lista över ej lösta rapporter för aktuell buss där man vid klick kommer in i uppdaterafunktionen.	Busschaufför	3
5	Livefeed	30	Kunna i realtid se alla icke lösta felrapporter i en lista sorterad efter tidpunkt och med möjlighet att sortera efter grad, där symptom, gradering, kommentar och tidpunkt visas. Det ska vara tydligt vilken gradering felrapporten har.	En lista med den viktiga datan representerad från felrapporterna.	Trafikledning/Service	4
6	Markera rapporter som lösta	30	Kunna markera skickade felrapporter som lösta, att status på dem ska uppdateras och att de inte längre visas där endast olösta	Se till att status i databasen uppdateras och att de inte längre visas där endast olösta rapporter ska visas.	Trafikledning/Service	4

			rapporter ska visas.			
7	Busslista	30	Kunna överskådligt se status på bussar och kunna se de icke lösta felrapporterna kopplade till den specifika bussen.	En lista med alla bussar som visar med två olika färger dess status. (Med eller utan några olösta rapporter) Kunna klicka sig vidare till felrapporterna därifrån.	Trafikledning/Service	5
8	Historik	20	Kunna se historik på alla bussar och söka bland dem.	En lista med bussar där man kan klicka sig vidare för att se alla lösta felrapporter för specifik buss.	Trafikledning/Service	5
9	Ändra gradering	10	Kunna ändra gradering på en felrapport.	En funktion för att ändra gradering i detaljerade felrapporterna.	Trafikledning/Service	5

Kommentar: Vi har inte använt importance, som nämns av Kniberg (2015), men vi har ändå prioriterat genom att vid varje sprintmöte omvärdera och omprioritera ordningen. Det hade kanske varit värdefullt att använda importance för att lättare kunna väga hur viktiga olika stories var i förhållande till varandra, men vi ansåg ändock att vårt tillvägagångssätt skapade en välordnad prioriteringen. Från början var det svårt att lägga user stories och acceptanskrav på lämplig nivå, då det var lätt att väva in tekniska lösningar eller gå för djupt in på problemen. Efter feedback och retrospectives ändrades dock detta och en bättre nivå på acceptanskraven kunde väljas.