




DAT255 VT2016

REFLEKTIONSRAPPORT

BUTANSWERSDO

VINCENT BRIGEL
ANTON KÄRRMAN
ANTON OTTOSSON
AXEL REBNER
OLOF WIREKLINT
DAVID ÅSENHIELM



Innehållsförteckning

Innehållsförteckning	2
1 Introduktion	3
2 Arbetsgång och använd metodik	4
2.1 Roller	4
2.2 Socialt kontrakt	4
2.3 Stand-up meetings	5
2.4 Parprogrammering	5
2.5 Distribution av tid	6
2.6 Effort och velocity	7
2.7 Product-backlog	8
2.8 Sprintar	9
2.8.1 Trello	9
2.8.2 Arbetsprocessen	10
2.8.3 Sprint retrospective	11
2.8.4 Sprint review	13
2.9 Testning	13
3 Reflektioner kring D1 och D2	15
4 Sammanfattande tankar och reflektioner	17
Appendix A - projektvision	18
Appendix B - Resultat av sprint retrospectives	19
Appendix C - Resultat av sprint reviews	21

1 Introduktion

Android-applikationen *55 problems - reporting made easy* utvecklades i kursen Software Engineering Project (DAT255) på Chalmers Tekniska Högskola av utvecklingsteamet ButAnswersDo bestående av Vincent Brigel, Anton Kärrman, Anton Ottosson, Axel Rebner, Olof Wireklint och David Åsenhielm. Utveckling skedde under kursens gång under april till och med början av juni 2016. Gruppen har använt sig av programvarorna Android Studio, GitHub, GitHub Desktop, Google Docs, Google Spreadsheet och Trello. Applikationen utvecklades i syfte att underlätta felrapporteringen för busslinje 55 som går i Göteborgs busstrafik. Applikationens utformning har baserats på en tydlig vision som tagits fram av projektgruppen och som varit underlag för utvecklingsprocessen. Visionen återfinns i appendix A. I rapporten har "Scrum and xp from the trenches: how we do scrum." av Kniberg (2015) använts som referens för resonemang knutna till Scrum.

2 Arbetsgång och använd metodik

I detta projekt har ett agilt arbetssätt anammats och mer specifikt har utvecklingsmetoden Scrum applicerats. I detta kapitel följer en redogörelse för gruppens arbetsmetodik med Scrum samt hur gruppen i övrigt har arbetat, tillsammans med en reflektion av valda arbetssätt.

2.1 Roller

Gruppen har i projektet valt att arbeta med olika ansvarsroller och valde att arbeta med en Scrum-Master, Product Owner, Github-ansvarig och Trello-ansvarig. Gruppen valde att inte ha en dedikerad roll för testare, utan testning blev istället kort i product-backlog lite senare under projektets gång. Detta för att inte knyta all testning till en person, så att alla kunde involveras i arbetet och lära sig. Scrum-mastern har varit ansvarig för att boka möten och programmeringssessioner. Dessa möten involverar sprint plannings, sprint demos och slutligen sprint retrospectives. Detta följer de riktlinjer som Kniberg (2015) tar upp.

En intern Product Owner valdes ut med syfte att ha en person som förde dialog med intressenterna och ansvarade också för att acceptanstest planerades in med dem. Github-ansvarig skulle se till att varje merge till gruppens huvudgren var buggfri, vilket säkerställdes genom bland annat FindBugs. Gruppen har i GitHub använt en sidogren där majoriteten av commitments kopplade till koden har gjorts. Trello användes under arbetet då en person i gruppen hade stor erfarenhet av detta verktyg sedan tidigare och enkelt kunde lära ut till övriga gruppmedlemmar. Denna person blev således Trello-ansvarig och hade i uppgift att strukturera upp den board som användes i projektet.

Arbetet med ansvarsområden medförde initialt att gruppen fick en bra struktur gällande användandet av de verktyg som varit centrala i projektet. Detta gjorde att det tidigt etablerades ett gemensamt arbetssätt för hur de olika verktygen skulle skötas, vilket gruppen tror varit anledningen till att arbetet med de olika verktygen fungerat relativt smärtfritt. Rollerna medförde inte mycket extra arbetstid för de personer som var ansvariga, men medförde att det fanns ett ansvar kring att saker och ting sköttes korrekt. I framtida projekt hade gruppen förmodligen försökt att fördela olika ansvarsområden igen, då det endast upplevts positiva följder med detta arbetssätt.

2.2 Socialt kontrakt

För att skapa förutsättningarna för ett effektivt samarbete inom gruppen skapades ett socialt kontrakt. Tanken var att sätta upp förhållningsregler kring arbetet i projektet, samt hur personerna i gruppen skulle förhålla sig till varandra. För att säkerställa att riktlinjerna i kontraktet efterföljdes fanns det även förbestämda påföljder om kontraktet inte upprätthölls. Gruppen anser att det sociala kontraktet varit ett gott initiativ och att det har fungerat som ett stöd under arbetet. Då alla i gruppen kände varandra väl sedan innan tenderade det dock att uppstå störningsmoment under gruppstillfällena genom att annat än relevanta saker för projektet diskuterades. För att kringgå detta hade gruppen i efterhand velat använda fler påföljder för tillfällen då kontraktet inte efterlevdes fullt ut för att få gruppen att fokusera bättre.

Överlag tycker gruppen att lagarbetet har fungerat bra. Inga uppenbara konflikter har uppstått och andra i gruppen har på ett effektivt sätt kunnat täcka upp ifall personer inte kunnat medverka på gemensamma träffar. Det har funnits en förståelse för att personer i gruppen har olika scheman och att det därför funnits tillfällen där alla inte kunnat medverka, även om det i

största mån eftersträvats. En lärdom är att acceptera att folk har olika scheman och istället hitta ett arbetssätt som är anpassat för aktuella förutsättningar.

2.3 Stand-up meetings

På grund av sprintarnas korta längd samt projektmedlemmarnas olika scheman var det svårt att hålla daily-Scrum-möten. Däremot bestämdes att det skulle hålla ett sprint planning-möte varje måndag, helst i anslutning till föreläsningen, samt att göra sprint review och retrospective fredagar varje vecka. Eftersom gruppen till stor del avsatte gemensamma pass för att arbeta med applikationen fanns det inte ett jättestort behov av daily-Scrum-möten för att uppdatera varandra om status på olika delar i projektet, detta skedde istället kontinuerligt under arbetspassen. I ett mer omfattande projekt eller i ett projekt med mer arbete på egen hand hade förmodligen det funnits större behov av möten för att uppdatera varandra.

2.4 Parprogrammering

I projektet ämnade gruppen att använda parprogrammering då samtliga medlemmar hade goda erfarenheter av detta från tidigare programmeringskurser. Även gruppens handledare (Andreas från MicroTube) rekommenderade att arbeta i par då det hade krävts mer detaljerad nerbrytning av uppgifterna för att kunna arbeta på egen hand, något som skulle vara svårt att hinna med på grund av projektets tidsram. Tanken var att under varje sprint bilda tre stycken par som skulle arbeta tillsammans under sprinten. Dessa team skulle sedan roteras mellan sprintarna för att nå maximalt kunskapsutbyte. Att frekvent rotera paren vid arbete med parprogrammering är något som bland annat Kniberg (2015) förespråkar. Utfallet under projektet blev något annorlunda, även om parprogrammering eftersträvades i högsta möjliga mån under hela projektets gång.

Eftersom ingen i gruppen hade någon tidigare erfarenhet av Android-programmering bestod mycket av det initiala arbetet av att sätta sig in i och förstå upplägget av de verktyg som användes. Detta visade sig vara väldigt tidskrävande initialt och för att försöka få alla på samma kunskapsnivå valde gruppen att arbeta med två grupper om tre personer de första två sprintarna. Gruppen frångick därmed den initiala tanken om parprogrammering. Kunskapsnivån höjde sig däremot snabbt och gruppen övergick till den initiala tanken om parprogrammering.

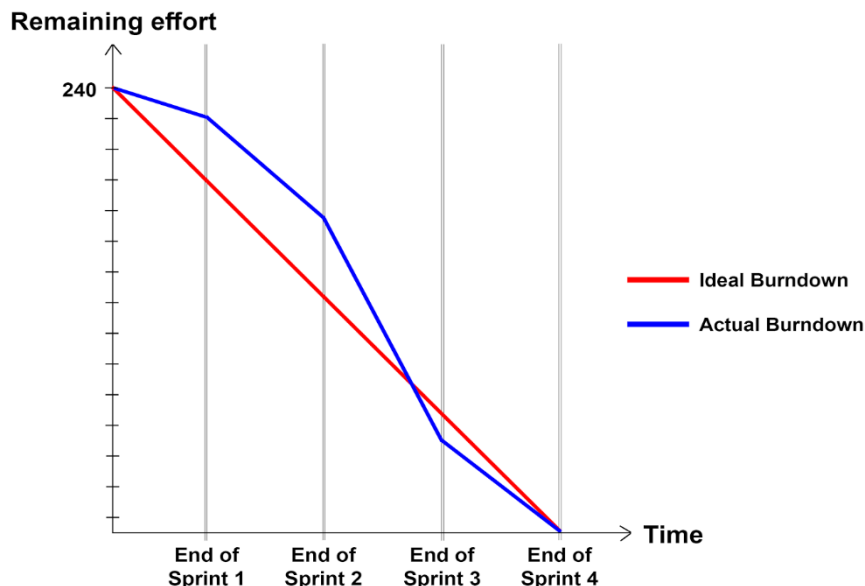
Roteringen i teamen för att utbyta kunskap försvårades något av att två personer hade datorer som hade svårt att köra Android-emulatorn och behövde alltid jobba med någon som hade en fungerande dator. Rotation bland programmeringsparen frångicks även i slutet av projektet. Flera av uppgifterna som genomfördes då var av liknande karaktär som de som tidigare lösts och gruppen valde därför i flera fall att låta samma personer genomföra uppgifterna för att nå en snabb lösning. Detta eftersom gruppen i slutskedet prioriterade en fullt fungerande prototyp framför totalt kunskapsutbyte.

Gruppen anser att arbetet med parprogrammering har varit givande för projektets framfart och hade vid återupprepning av projektet valt att anamma det igen. Däremot finns viss förbättringspotential med det arbete som gjorts och exempelvis hade gruppen i efterhand försökt utnyttja parprogrammering under hela processen för att möjliggöra mer effektivt arbete samt sprida kunskapen ytterligare.

2.5 Distribution av tid

Tidsfördelningen mellan medlemmarna har varit relativt jämn. Rollfördelningen som gruppen hade innebar inte så mycket extra arbetstid för de berörda. Det som behövde mest tid var Scrum-master som varje vecka behövde boka in möten, men även den tiden bedöms som relativt låg.

Figur 1 illustrerar hur arbetet i projektet har fortskridit. Gruppen arbetade snabbare i slutet av projektet, vilket till stor del beror på den inläringströskel som fanns och som tog mer tid än förväntat. Sett till effort gjordes de viktigaste delarna av applikationen under en förhållandevis kort tid under 1-2 sprintar i slutet av arbetet. Under de första sprintarna gick mycket tid åt att lära sig och hitta tekniker som skulle kunna passa för applikationens syfte. När kunskapen sedan föll på plats gick arbetet fort, mycket på grund av att det kunde parallelliseras.



Figur 1: Illustration över kvarvarande effort och tidsåtgången i arbetet i form av en Burndown chart. Den röda linjen illustrerar den effort-tid kurva som erhållits om allt effort utförts med samma takt. Den blå linjen illustrerar den verkliga effort-tid kurvan i arbetet.

Uppgiften som tog längst tid var att hitta en fungerande lösning för att skicka data mellan applikationen och en extern enhet. Gruppen spenderade mycket tid på att söka efter möjliga lösningar och la även mycket tid på att försöka sätta upp en egen databas. Gruppen fick inte databasen att fungera och fann efter ett tips från en områdeskunnig plattformen Firebase, en lösning som sedan användes i projektet.

Att så mycket tid lades på att försöka få till en databas som visade sig inte fungera kan dels relateras till vad John Lantz från Volvo pratade om relaterat till att våga pröva olika lösningar för att kunna lyckas. Även Michael Öhman på Spotify pratade om att arbeta med misslyckanden som en aktiv del av strategin. Trial and error samt learning by doing är i det avseendet relevanta begrepp och dessa har varit centrala under projektets gång. Dels på grund att de låga förkunskaperna innan projektets start, men även som en medveten strategi för att hitta lösningar till de problem som uppstod längs vägen.

Utifrån trial and error-strategin var en av de viktigaste lärdomarna att vara öppen för input från andra. I jakten på fungerande lösningar snabbades processen upp ytterligare de gånger gruppen rådfrågade kontakter med bättre kunskap inom de berörda områdena. En av anledningarna till att arbetet med att hitta en teknik för att skicka data ifrån applikationen tog så lång tid var att gruppen till en början inte rådfrågade andra för att hitta en lösning. Detta var en lärdom som gruppen tog med sig och i senare tillfällen i projektet, till exempel vid utformningen av unit tests, rådfrågades andra i ett tidigare skede vilket gjorde att gruppen inte körde fast i samma utsträckning.

2.6 Effort och velocity

Generellt var gruppen till en början dålig på att matcha velocity och effort. Antingen hann gruppen inte genomföra alla kort som planerats under en sprint, eller så lyckades fler än planerat att genomföras. Detta korrelerar mycket med inlärningskurvan. Samtidigt gjorde avsaknad av tidigare erfarenheter av applikationsprogrammering att det var svårt att uppskatta vad som var svårt respektive lätt att genomföra.

Dessa aspekter blev bättre mot slutet och sista sprinten lyckades gruppen matcha effort och velocity. Detta berodde till stor del på ökade programmeringskunskaper från gruppens sida. Till en början var uppgifterna små och det som krävde tid var att söka information för att hitta en lösning på problemet. Kodningen var i detta fall endast några rader eller enstaka metoder. Tiden för att söka information var svårastimerad, men när uppgifterna blev mer programmeringstunga blev också estimaten lättare och bättre. Detta eftersom att en mindre del av estimaten bestod av att uppskatta tidsåtgången för att söka information och en större del åt att uppskatta tid för kodning. Samtidigt blev gruppmedlemmarna snabbare vilket försvårade uppskattningarna. Detta eftersom tidigare utfört arbete inte rakt av kunde användas som referens för att uppskatta liknande moment.

En annan anledning till att estimeringarna blev sämre i början var att gruppen inte bröt ner user stories-korten tillräckligt mycket. Detta medförde alldeles för omfattande kort som gruppen i vissa fall inte uppnådde. Ett exempel på detta är länken som skulle möjliggöra att skicka data till en extern part som blev underestimerad pga att user-storien inte var tillräckligt nedbruten. Förmågan att bryta ner user-stories förbättrades under projektets gång. Vid framtida projekt hade faktumet att tydligt bryta ner user-stories varit en av de viktigaste lärdomarna att ta med sig, för att underlätta estimeringen av effort för korten i produkt-backloggen.

Gruppen bestämde inför första sprinten att en normalvecka innebar 60 i velocity. Inför varje sprint gjordes sedan nya uppskattningar av effort för aktuella user stories baserade på ökad förståelse för tidsåtgången. För att illustrera förändring av effort och nedbrytning av ett kort exemplifieras i Figur 2 nedan samma kort i ett tidigare och i ett senare skede av projektet.

ID	Name	Description	Importance	Estimate (Velocity=60)	Acceptance Criteria	How to demo
6	Skicka information om feltyp vid felrapportering	Som ledningscentral behöver jag veta vilken typ av fel rapporterad buss har så att jag kan vidta rätt åtgärd	50	35		Trycker på ett fel och rapporterar in. Feltypen i bussen registreras i en databas som visas upp.

ID	Name	Description	Importance	Estimate (Velocity=60)	Acceptance Criteria	How to demo
6	Skicka information om feltyp vid felrapportering	Som ledningscentral behöver jag veta vilken typ av fel rapporterad buss har så att jag kan vidta rätt åtgärd	55	60	-Möjlighet finns att definiera vilket typ av fel bussen har	Trycker på ett fel och rapporterar in. Feltypen i bussen registreras i en databas som visas upp.
6.1	Skicka med vilken buss vi befinner oss i	Som trafikledning måste jag känna till vilken buss som felet uppstår i.	55	20	-Vilken buss man sitter i skickas med till databasen	Trycker på ett fel och rapporterar in. Bussens namn finns med i felmeddelandet som registreras i databasen.
6.2	Geografisk position skickas med automatiskt varje gång ett fel rapporteras.	Som verkstadsarbetare vill jag veta var bussen var när felet rapporterades. Som chaufför vill jag inte lägga in positionen där bussen befinner sig manuellt.	55	25		Trycker på ett fel och rapporterar in. GPSen i bussen registrerar platsen i en databas som visas upp.
6.3	Skicka information om tid vid felrapportering.	Som ledningscentral behöver jag veta vid vilken tid ett fel rapporterades in.	55	10		Trycker på ett fel och rapporterar in. Tiden registreras i en databas som visas upp.
6.4	Välja allvarighet av fel	Som chaufför måste jag kunna klassificera hur allvarligt felet är så att ledningscentralen vet vilken åtgärd de ska vidta	55	10		Går att trycka på ett fel och sedan välja en färg för att beskriva allvarligheten av felet

Figur 2. Kort 6 från första product-backlogen (övre bild) respektive kort 6 från tredje product-backlogen, nu nedbrutet i mindre delar (nedre bild).

2.7 Product-backlog

Prioriteringen i Product-backlogen gjordes till en början enbart baserat på gruppens värdering för hur viktiga olika user stories var. Vid första handledningen fick gruppen feedback från handledare om vad som var viktigt att tänka på vid utformningen av en product-backlog. Detta ledde till vissa omprioriteringar av kort, samt att vissa user stories bröts ner mer till mindre deluppgifter. Inför varje sprint gjordes även omprioriteringar av product-backlogen baserat på insikter kring olika user stories relevans och även nya user stories tillkom. Efter handledningsmöte två med Andreas från MicroTube applicerades ett tankesätt för prioritering av product-backlogen baserat på att diskutera vilka element som är “must have”, “should have” respektive “nice to have”. Detta ledde till att gruppen prioriterade om vissa element. Till exempel bedömdes att skicka GPS-koordinater i felmeddelandet vara “nice to have” eftersom applikationen avses att användas på någon av ändstationerna och således tillför GPS information endast ett begränsat mervärde. Således fick kortet en lägre prioritet än innan.

Efter acceptanstester med busschaufförer tillkom ett antal efterfrågade funktioner som blev till nya user stories och som fick hög prioritet. Feedbacken från chaufförerna var av stort värde och kopplades tydligt till “stakeholder value”. I efterhand anser gruppen att det hade varit bra att få denna typ av feedback tidigare under projektet och även få liknande feedback från andra intressenter. Busschaufförerna var de mest lättåtkomliga av de olika intressenterna vilket bidrog till att endast de rådfrågades. Hade även till exempel trafikledningen tillfrågats kring funktionalitet hade färre beslut och prioriteringar behövts göras baserat på gruppens bedömningar. Ett exempel är funktionaliteten för att kunna klassificera kritikaliteten av felen som rapporteras. Det var en funktion som gruppen bedömde kunde vara önskvärd men med mer intressentinteraktion hade det kunnat visa sig att detta inte var efterfrågat, eller att det fanns annat som borde prioriterats högre.

Vid framtida projekt hade förmodligen intressentinteraktionen varit en av de tydligaste förändringarna kopplat till product-backlogen. Ökad interaktion med intressenterna ger en produkt som bättre svarar mot kundvärde och att få den feedbacken även tidigt i projektet hade kunnat bidra till mer värdeskapande arbete från gruppen. Att detta endast skedde i begränsad omfattning berodde dels på svårighet att kunna interagera med vissa av intressenterna, men även att gruppen till en början inte helt insåg värdet i denna typ av kommunikation.

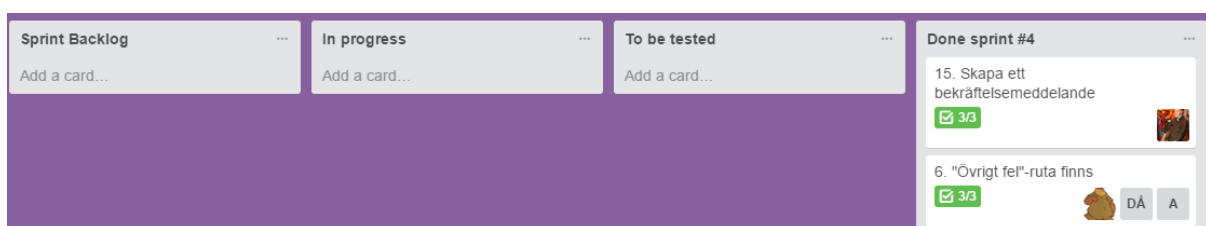
Att inte kunna interagera med intressenterna var något som John Lantz beskrev som ett problem på Volvo där hela avdelningar kunde ha ansvar för små delområden och endast leverera vidare sina resultat internt med en stor distans till slutkund. Han beskrev hur hans enda sätt att veta om det han gjorde var bra var i fall någon kom och klagade på hans arbete. Det visar på att samma problematik som gruppen upplevt kring svårigheter i att uppskatta vad kunden vill ha även är närvarande ute i industrin. Ständig interaktion med användare kan vara svårt att åstadkomma, särskilt om det rör sig om stora projekt och det finns flera intressenter, både internt och externt. För user stories har både andra gruppmedlemmar, busschaufförer, trafikledning, depån och examinators kunnat ses som intressenter och som beskrivits tidigare har det varit svårt att få feedback från vissa av dessa kontinuerligt.

2.8 Sprintar

Scrum-Mastern gjorde upp ett schema enligt Knibergs (2015) rekommendationer. Detta innebär att det fanns tid för att slappna av mellan sprintarna för att gruppen skulle kunna vila och lägga tid på andra aktiviteter. Sprintarna löpte över fem dagar (mån-fre). Denna process återupprepades fyra sprintar, varav sprint fyra pågick under två veckor. Anledningen till detta var att kandidatinlämningen låg samtidigt som en femte planerad sprint och tre av de fem arbetsdagarna skulle försvinna, då samtliga medlemmar arbetade fulltid med kandidaten. För att kompensera bortgången sprint fem ökades velocityn för sprint fyra.

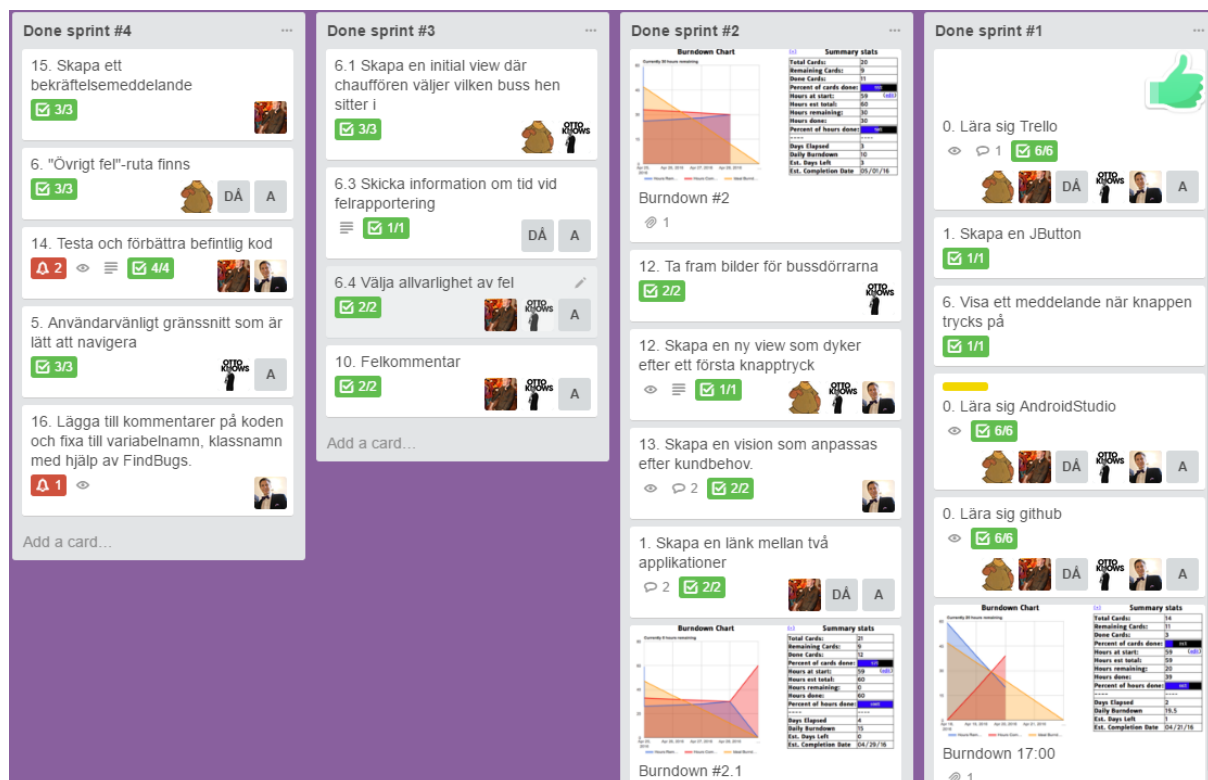
2.8.1 Trello

Gruppen har använt sig av programmet Trello för att på ett effektivt sätt strukturera arbetet med Scrum och user stories. Trello uppdaterades ständigt inför, under och efter sprintarna. Sprintkorten delades upp i fyra kategorier: *Sprint-backlog* (innehöll nedbrutna sprintkort med ursprung i product-backlog), *In progress* (de sprintkort som för tillfället arbetades med), *To Be Tested* (de kort som var redo för demo) och slutligen *Done* (de kort som klarat demon och ansågs helt klara), se Figur 3. Demo av korten gjordes med minst tre gruppmedlemmar närvarande för att säkerställa konsensus kring kortets fullbordan. Strukturen som använts med flera kategorier följer i det närmsta den modell Kniberg (2015) förespråkar, men skiljer sig då gruppen använt sig av en extra kategori i form av "To be tested". Detta användes då gruppen tyckte det var mer intuitivt och mer visuellt att ha en egen kategori för demo.



Figur 3. Innehåller de olika stadierna i Trello som ett kort kunde befinna sig i.

Gruppen valde att göra en "Done"-kategori för varje vecka vilket syns i Figur 4. Detta för att ha översikt över arbetets gång samt inte glömma av något som arbetats med. De gröna rutorna visar antalet acceptance-criterias som checkades av vid varje demo. Att använda sig av acceptance-criterias är något som bland annat Kniberg (2015) rekommenderar. Gruppens definition av klar bestämdes vara då ett kort uppfyller de acceptance-criteria som satts upp. I Figur 5 visas ett kort från Trello som på ett mer detaljerat sätt illustrerar hur gruppen formulerade sprint-korten och acceptance-criterias.



Figur 4. Visar de fyra sprintarna innehållande färdiga sprintkort, burndown charts och annat.



Figur 5. Figuren visar ett kort i Trello med beskrivning av user storien och acceptance-criterias.

2.8.2 Arbetsprocessen

Gruppen arbetade med sprint retrospectives efter varje sprint vilket bidrog till kontinuerlig förbättring av arbetet. Eftersom ingen i gruppen tidigare arbetat enligt Scrum var det till en början en viss inlärningskurva för de olika momenten vilket innebar att processen förfinades efterhand. Efter första sprinten beslutades det att mer detaljerat specificera de olika sprintkorten då de uppfattades som otydliga av gruppmedlemmarna och därför försvårade

bedömningen av ifall dessa var avklarade i sprinten. Detta hanterades genom att skriva in acceptance-criteria för varje enskilt sprintkort. Acceptance-criteria hade tidigare bara utformats för korten i product-backloggen vilket försvårade att bedöma de olika sprintelementens status då ett kort i product-backloggen ofta innehöll flera sprintkort. Förändringarna gjorde det lättare att uppskatta tiden samt att avgöra vilka element som kunde klassas som färdiga. Detta underlättade arbetet i de efterföljande sprintarna.

Efter andra sprinten gjorde gruppen ytterligare lärdomar som användes för att förbättra processen. Det beslutades att i framtiden arbeta två personer i stället för tre med samma uppgift, då det uppmärksammats att arbeta tre ledde till sämre fokusering. Att koda två personer samtidigt medförde senare i projektet att mer tid gick till merga ihop kod eftersom fler grupper kodade parallellt. Tack vare att fler av elementen i slutet av projektet inte var renodlade koduppgifter möjliggjordes dock att dela upp gruppen i tre par och uppnå bättre fokusering samtidigt som problemet med merges undveks. Som tidigare beskrivits rådfrågades under sprinten områdeskunniga för att hitta lösningar och detta togs med som en lärdom för framtida sprintar. I efterhand visade det sig att förändringen att tidigt ställa frågor till områdeskunniga inom ämnet effektiviserade arbetet, då inte lika mycket tid krävdes för att färdigställa de olika sprint elementen efter förändringen.

Efter den tredje sprinten förändrades arbetet med acceptance-criterias. Vid utformning av acceptance-criteria för att kunna ange kritikalitet på de fel som rapporterades skrevs kriterierna så att lösningen var tvungen att innefatta att kritikaliteten illustrerades genom olika färger. Det begränsade gruppens möjlighet till att utvidga funktionaliteten. Därför beslutades det att aktivt tänka på hur kriterierna i framtiden formulerades för att ge plats åt nya kreativa lösningar inom ramen för det problem som varje sprint element ämnade att lösa.

2.8.3 Sprint retrospective

För samtliga av sprintarna har gruppen genomfört sprint retrospectives under gemensamma möten i samband med respektive sprints avslutande. Enligt Kniberg (2015) bör sprint retrospectives innehålla en diskussion med gruppens medlemmar om vilka delar i sprinten som karaktäriserats av följande tre aspekter:

- Good: Om vi hade gjort om samma sprint igen skulle vi göra dessa delar på samma sätt igen
- Could have done better: Om vi hade gjort om samma sprint igen skulle vi förändrat arbetet med dessa delar
- Improvements: Konkreta idéer och tankar om vad vi kan göra bättre i framtida sprintar

I gruppens fall har detta tankesätt använts genom att tabeller byggts upp med en kolumn för varje aspekt. På raderna under kolumnerna beskrevs sedan delar av sprinten som av gruppen ansågs höra till respektive aspekt. I Tabell 1 illustreras anteckningarna från första sprintens retrospective.

Tabell 1. Sammanställning av retrospective för Sprint 1

<i>Good</i>	<i>Could have gone better</i>	<i>Improvement</i>
Vi har kommit igång med projektet på ett tillfredsställande sätt.	Vi underskattade problematiken med att skapa en länk mellan två enheter och bröt därför inte ner uppgiften tillräckligt detaljerat.	Skapa tydliga acceptance-criteria för varje kort i sprint-backloggen.

Vi fick en bra överblick över programmen.	Vi kunde varit tydligare med hur GUI skulle se ut. Det var otydligt huruvida vi hade gjort det eller ej.	Omvärdera tidsåtgången för user-stories i product-backlog.
AndroidStudio är ett smidigt program för att bygga basfunktionalitet.	Tidsuppskattningen var en aning sneddriven.	Bryt ner korten mer detaljerat i funktionalitet.
Vi har fått en bra bild av vad som är väsentligast i produkt utvecklingen.	Vi kunde haft tydligare namn på sprintkorten som var mer förklarande.	Skriv mer tydliga namn på sprintkorten

I den första sprinten missförstod gruppen hur anteckningarna i retrospective skulle göras och flera kommentarer i de olika kolumnerna blev därför felaktigt utformade jämfört med definitionen ovan. Anteckningar skrevs exempelvis på formen “vad som gått bra/dåligt resultatmässigt” snarare än “vad vi gjort bra/dåligt” och möjligheten att dra lärdomar utifrån dessa anteckningar, om sprinten skulle göras om, förvinns därför. Däremot kunde gruppen dra lärdomar och utnyttja de anteckningar som skrevs i improvement-kolumnen inför kommande sprintar.

Kvaliteten på anteckningarna förbättrades efter första sprinten och gruppen blev kontinuerligt bättre på att konkretisera anteckningarna allt eftersom det framgick hur anteckningarna borde skrivas. Detta kan illustreras med retrospective från sprint 3, vilken beskrivs i Tabell 2, där anteckningarna tydligare följer den definition från Kniberg (2015) som beskrivits ovan. Samtliga retrospectives återfinns i Appendix B.

Tabell 2: Sammanställning av retrospective för Sprint 3

<i>Good</i>	<i>Could have gone better</i>	<i>Improvement</i>
Intern workshop där vi gick igenom product-backlog för att kontrollera ifall vi saknade viktig funktionalitet. När vi gjorde detta kom vi på funktionalitet som vi under veckan kunde hitta en helhetslösning på, vilket gjorde att vi kunde bocka av flera kort än planerat.	Tidsplaneringen, denna gången överskattade vi tiden det skulle ta att utföra veckans kort och vi lade därför till extra kort i efterhand.	Tydligare variabelnamnsättning och kommentering av kod. Ibland svårt att förstå andras kod då variabelnamn, metodnamn, kodkommentarer etc. är otydliga.
Anammande tankesättet “must have”, “should have” och “nice to have” för prioritering av kort vilket gav större fokus på kundvärde.	Vi specificerade våra acceptance-criteria för hårt under denna sprint. Tidigare har vi haft för dålig detaljnivå, men denna gång specificerade vi för mycket i detalj att det gjorde att vi låste kortet till en specifik lösning. För att förtydliga skrev vi “Välja allvarlighet via en färg” och låste då oss till att göra lösningen med hjälp av färger, men vi använde i själva verket textbeskrivning. “Välja allvarlighet genom olika alternativ” hade varit bättre.	Var inte för specifika när vi skriver acceptance-criteria. Måste hitta en balans mellan att inte vara för otydlig, men samtidigt inte låsa oss vid en lösning som sedan visar sig vara suboptimal.

Utnyttjade GitHub desktop för att koda parallellt i olika delar av koden. Fungerade sedan smärtfritt att merge samman koden.		I högre grad utnyttja möjligheten att konsultera personer med högre områdeskunskap för att hitta rimliga lösningar till problemen. Även för att skapa en förståelse för vad som är rimligt att klara av under den begränsade tidsramen, utifrån de kunskaper vi besitter.
--	--	---

2.8.4 Sprint review

Enligt Kniberg (2015) bör sprint review genomföras efter varje sprint för att se till så att sakerna faktiskt blir helt klara. För samtliga av sprintarna har gruppen därför genomfört sprint review under möten i samband med respektive sprints avslutande. Dessa reviews är utformade efter en tabell med kolumnerna:

- Sprint element: Vilket sprintelement som åsyftas.
- Acceptance criteria: De acceptance-criterias som ska uppfyllas för att klassificera elementet som klart.
- Fulfilled: Om acceptance-criteria är uppfyllda eller ej.
- Comments: Eventuella kommentarer på om elementet är färdigt eller ej.

I Appendix C redovisas tabellerna som gruppens review-möten resulterade i.

Eftersom möjlighet till kontakt med stakeholders var begränsad under projektet satte gruppen inte acceptance-criteria där stakeholders bestämmer huruvida dessa är uppfyllda eller ej. I stället sattes acceptance-criteria där gruppen själva tog beslutet om de är uppfyllda och sedan utfördes när möjlighet gavs kontroller med stakeholders i form av bland annat acceptanstest. Dessutom genomfördes kontinuerligt diskussioner med handledare (Håkan och Andreas) som ledde till att gruppen omprioriterade user stories baserat på vilket värde dessa bedömdes ge stakeholders. Ett exempel på resultat från en sådan diskussion är att i början av projektet övervägde gruppen att prioritera bort länken mellan applikation och databas eftersom stora svårigheter uppstått vid arbetet med att implementera länken. Gruppen resonerade att det skulle vara möjligt att demonstrera en applikation utan att kunna skicka data men när detta diskuterades med Håkan framförde han att en länk är nödvändig för att visa värdet av gruppens produkt för produktägaren. Efter detta beslutades att i nästa sprint fortsätta arbetet med att skapa länken. Under slutpresentationen visade sig detta vara ett bra beslut eftersom de olika produktägarna generellt var mycket intresserade av att se vad som hände när en felrapport skickats iväg med applikationen och verkade sätta stort värde vid detta.

2.9 Testning

För att säkerställa att applikationen fungerade som tänkt har olika tester genomförts. Testerna har dels undersökt att programmet levererar värde till stakeholders på ett bra sätt och dels testat så att koden fungerar som den ska. Testerna som är genomförda på programmet är: acceptanstester, enhetstester, integrationstester samt FindBugs. En mer ingående beskrivning av respektive testtyp återfinns i githubrepot i mappen prototypeComments.

Noterbart är att testerna inte är uttömmande och kan således inte garantera att programmet fungerar i alla möjliga situationer. Med det sagt är testerna istället utformade för att påvisa att programmet fungerar som det ska i de olika typer av situationer som anses vara de kritiska och relevanta för kursens syfte. Det hade gått att göra fler tester för att ytterligare styrka applikationens tillförlitlighet men då projektet hade en begränsad tidsrymd prioriterade

gruppen att lägga tid på andra saker istället. Den avvägningen gjordes eftersom gruppen ansåg att de utförda testerna var tillräckliga medans det fortfarande fanns mer funktionalitet att bygga ut för att leverera ytterligare kundvärde.

3 Reflektioner kring D1 och D2

Jämförs gruppens vision som finns idag med den som lades upp i projektets start har den utvecklats till att fokusera mer på vad det är för värde som skapas och går djupare in på värdet för varje intressent. Detta har tillkommit naturligt under projektets gång då gruppen fått större förståelse för vad för värde som skapas, hur det skapas och varför det skapas. Genom att iterativt gå tillbaka till visionen löpande under sprint retrospectives har gruppen kunnat styra arbetet efter detta. Vidare har även product-backloggen uppdaterats i slutet av varje sprint. Till exempel har kort tagits bort, lagts till samt omprioriterats till följd av ökad förståelse kring värde. Förändringen illustreras i Figur 7.

ID	Name	Description	Importance	Estimate (Velocity=60)	Acceptance Criteria	How to demo	Notes
0	Installera och bekanta sig med verktyg	Som kodare måste vi känna till verktygen innan vi börjar koda	60	20	- Alla i gruppen har installerat GIT, Android Tools, Trello	Trello ska ha en tydlig struktur som alla förstår. Programmen visas visuellt inför gruppen och går igenom.	
1	Skicka och ta emot felmeddelande	Som chaufför vill jag kunna rapportera att det är något fel på bussen utan att behöva prata i radio. Som trafikledare eller busskontrollerare vill jag veta när det är fel på bussen.	55	40			
1.1	Skapa en knapp för att skicka att något är fel med bussen	Som chaufför vill jag kunna meddela att bussen har ett fel så att ledningscentralen kan göra något åt det	55	15	- En knapp finns som skickar ett anrop till en annan app via någon anslutning	Öppna applikationen, trycka på knappen	
1.2	Ta emot ett meddelande att något är fel med bussen	Som ledningscentral behöver jag få reda på när det är fel på någon buss så att jag kan vidta åtgärder för att lösa problemet	55	25	- Kan ta emot ett meddelande från en anslutning -Visar ett meddelande att en buss har fått fel	Öppna applikationen, ta emot meddelande	
2	Sortera fel efter högst rapporteringsfrekvens	Som chaufför vill jag kunna sortera felen efter förekomstfrekvens så att jag enkelt kan komma åt de vanligaste felen	10	90		Öppna applikationen, navigera till en felkategori och tryck på en sorteringsknapp som automatiskt sorterar fram de fel som rapporterats flest ggr.	

ID	Name	Description	Importance	Estimate (Velocity=60)	Acceptance Criteria	How to demo	Notes
13	Vision	Som programutvecklare måste jag tydligt se helheten så jag vet vad jag utvecklar mot	70	5	Samtliga gruppmedlemmar godkänner visionen och den innehåller värdet som skapas för samtliga stakeholders.	Visionen presenteras på ett daily scrum två dagar i rad för att låta den smälta.	
0	Installera och bekanta sig med verktyg	Som kodare måste vi känna till verktygen innan vi börjar koda	60	20	- Alla i gruppen har installerat GIT, Android Tools, Trello	Trello ska ha en tydlig struktur som alla förstår. Programmen visas visuellt inför gruppen och går igenom.	Färdig i Sprint 1
14	Testa och förbättra befintlig kod	Som nya kodare behöver vi säkerställa att koden vi har skrivit hittills håller måttet och inte innehåller exempelvis några buggar. Som användare av applikationen vill jag veta att denna stabilt fungerar på rätt sätt	60	20	- Findbugs accepterar koden -Verifieringstest/blackboxtest (det som kommer ut i databasen är vad som förväntas av oss själv) -Acceptens test med stakeholders	-Kör koden med findbugs -Rädså med en busschaufför	
1	Skicka och ta emot felmeddelande	Som chaufför vill jag kunna rapportera att det är något fel på bussen utan att behöva prata i radio. Som trafikledare eller busskontrollerare vill jag veta när det är fel på bussen.	55	35			

Figur 7. Delar ur product-backloggen från sprint ett (övre bild) och sprint fyra (undre bild). Det som prioriterades högt i sprint ett har fått en annan värdering över projektets tid.

Från Lego Scrum övningen tog gruppen med sig många lärdomar. Gruppen tog framförallt med sig tre insikter efter övningen som ämnades utnyttjas i projektet vilka visas i Figur 8.

Insikt 1

Tillämpade inte planeringsfasen innan vi påbörjade att bygga respektive user-story i första sprinten.

Insikt 2

Hade ingen löpande kommunikation med produktägaren förutom under review-faserna.

Insikt 3

Felaktig prioritering av user-stories.

Figur 8. De tre viktigaste insikterna från Lego-övningen som gruppen tog med sig in i projektet.

Den första förändringen mot arbetet gruppen gjorde under Lego-övningen var att genomföra en fullständig planeringsfas inför varje sprint. Detta gjordes under det Scrum-möte som hölls på måndagar, där det bland annat bestämdes vilka user-stories som skulle genomföras under sprinten, hur gruppen ville att produkten skulle se ut i slutet av sprinten och hur processen för att nå dit skulle se ut.

Den andra insikten som erhöles under Lego-övningen var att gruppen inte kommunicerade tillräckligt med produktägaren, vilket medförde att gruppen byggde saker som produktägaren i slutändan inte ville ha. I projektet har det varit två olika produktägare i form av dels Keolis samt Volvo och dels examinator på kursen. För att försöka överkomma problematiken med bristande kommunikation har gruppen pratat med bussförare under de acceptanstester som beskrivits tidigare samt utnyttjat handledningstillfällen och föreläsningar för att konsultera examinator i frågor om produkten. För att se till att kommunikationen skulle ske kontinuerligt tillsattes en intern produktägare vars ansvar var att se till att detta skedde.

Den sista insikten handlade om att gruppen gjorde en felaktig prioritering av user-stories och valde user-stories som matchade dess velocity snarare än att gruppen valde stories som var av hög prioritet för produktägaren och således var av större värde. I projektet ämnades denna problematik lösas genom en gemensam prioritering av de user-stories gruppen tagit fram samt att gruppen efterhand projektet fortlöpte gjorde omprioritering av produkt-backlog allt eftersom förståelsen av värdet för kunden ökade. Prioriteringen låg sedan till grund för vilka user-stories som genomfördes under respektive sprint. Omprioriteringarna gjordes efter konsultation med handledare samt efter konsultation med busschaufförer då nya insikter kring vad som var viktigast framöver erhöles.

Övriga lärdomar från Lego-övningen var att inte ändra på både effort och velocity samtidigt samt att alla ska ha kunskap nog att kunna ta över någon annans arbete om någon person är borta. Ett tydligt exempel på detta är att gruppen gått igenom källkoden vid varje demo för att alla ska förstå koden. Även de som fick ansvaret för Trello & Github hade som uppgift att kontinuerligt se till att gruppen förstod verktygen. "En deadline är en deadline" är ett tankesätt som också applicerats oavsett om det gällt produkten eller processen.

Halvtidsrapporten belyste framförallt problematiken kring kunskap. Det tog längre tid att lära sig saker än förväntat och detta påverkade processen tydligt då gruppen i början inte hann klart med alla kort gruppen tog på sig. Lärdomen har varit att räkna med mer tid för att lära sig och söka kunskap, men lite senare i sprint två insåg gruppen även att det skulle underlätta att söka efter extern hjälp mer frekvent. Då projektet hade en tidsbegränsning jobbade gruppen med att få hjälp av områdeskunniga så mer fokus kunde läggas på vad som skapade värde i produkten. Detta innebar att mindre tid behövde läggas på programmeringsproblem och mer tid kunde läggas på diskussion om produkten och dess värde. Detta visade sig under resten av processen vara väldigt givande.

4 Sammanfattande tankar och reflektioner

Överlag tycker gruppen att projektet har varit en utmaning då det funnits många osäkerheter. Medlemmarna har behövt söka upp mycket information på egen hand vad det gäller tekniska detaljer samtidigt som de behövt arbeta efter ett nytt arbetssätt. Osäkerheter har uppstått då det inte funnits några tydliga ramar för hur arbetet skulle se ut. Samtidigt är gruppen nöjd med resultatet av arbetet under den begränsade tid som projektet har utspelat sig, samtliga medlemmar har arbetat ordentligt och tagit ett stort ansvar för resultatet.

Då gruppen saknade tidigare erfarenhet från Android-programmering gick arbetet till en början ut på att söka information för att komma igång med arbetet. Bristen på erfarenhet gjorde det svårt att värdera olika lösningar på problem och bedöma vilka som var mest lämpliga för projektet. Detta ledde till att designval gjordes tidigt i arbetet som sedan försvårade arbetet med att skala applikationen. Störst problem medförde utformningen av hur fel skulle presenteras, där en activity skapades för varje fel som skulle kunna rapporteras. Om applikationen i framtiden ska skalas kommer det därför krävas en mängd activities. När insikten gjordes blev det en avvägning mellan att prioritera lösningens skalbarhet, och därmed göra om koden från grunden, eller att fokusera på stakeholder value under slutpresentationen. Valet föll i det aktuella fallet på att fokusera på prototypens förmåga att generera värde till stakeholders under presentationen framför att vara långsiktigt skalbar. Detta skulle förmodligen medföra svårigheter om gruppen valde att arbeta vidare på applikationen i framtiden.

En lärdom från detta är att det är viktigt att hela tiden tänka igenom de beslut som tas då de kan få konsekvenser för en lång tid framöver, något som även John Lantz på Volvo tog upp som en viktig faktor i hans arbete. Även gruppens handledare Andreas beskrev hur en stor del av hans arbete gick ut på att planera och strukturera koden han skulle skriva och att förstå vikten av detta är något gruppen tar med sig genom erfarenheterna från projektet och som förmodligen skulle ges mer tid i ett framtida projekt.

Gruppens prioriteringar av arbetsuppgifter beslutades genom diskussioner kring vad som genererade värde och hur det kopplades till visionen. Detta gjordes med grund i det material som var tillgängligt, exempelvis felrapporter från trafikledningen. Gruppen märkte att mycket värdefull input kunde erhållas från acceptanstesterna gällande de delar som gruppen tidigare diskuterat fram själva. I efterhand hade gruppen gärna sett att de utfört acceptanstesterna i ett tidigare stadie under projektet så att testerna kunde fungerat mer som vägledning. Det hade även varit givande med tester med trafikledningen och depån för att säkerställa att den funktionalitet som utformades faktiskt var funktionalitet som efterfrågades av dessa intressenter.

Appendix A - projektvision

55 Problems - reporting made easy är en applikation utvecklad för Göteborgs Energi, Keolis och Volvo som söker en lösning på de problem som följer med dagens felrapportering av 55:ans bussar. Syftet med 55 Problems är att kringgå språkbarriärer och underlätta felrapporteringen vilket gynnar busschaufförerna, trafikledningen och i förlängningen depån. Till skillnad från det befintliga radiobaserade rapporteringssystemet som förlitar sig på flera led av kommunikation erbjuder 55 Problems en lösning där busschauffören navigerar genom ett gränssnitt som tydligt påvisar vanliga fel som finns på bussen. Denna processen är enkel, snabb och smidig vilket möjliggör att fler fel rapporteras än i dagsläget. Samtidigt får trafikledningen konsekvent information och undviker risker för kommunikationsmissar. 55 Problems är framtagen genom samarbete med intressenterna och är anpassad för att lösa svårigheterna som de upplever. Det är en unik lösning som minskar tröskeln för chaufförerna att rapportera fel samtidigt som felen rapporteras på ett enhetligt sätt vilket underlättar för både trafikledningen och i slutändan depån.

Appendix B - Resultat av sprint retrospectives

B.1: Retrospective för sprint 1

<i>Good</i>	<i>Could have gone better</i>	<i>Improvement</i>
Vi har kommit igång med projektet på ett tillfredsställande sätt.	Vi underskattade problematiken med att skapa en länk mellan två enheter och bröt därför inte ner uppgiften tillräckligt detaljerat.	Skapa tydliga acceptance-criteria för varje kort i sprint-backloggen.
Vi fick en bra överblick över programmen.	Vi kunde varit tydligare med hur GUI skulle se ut. Det var otydligt huruvida vi hade gjort det eller ej.	Omvärdera tidsåtgången för user-stories i product-backlog.
AndroidStudio är ett smidigt program för att bygga basfunktionalitet.	Tidsuppskattningen var en aning sneddriven.	Bryt ner korten mer detaljerat i funktionalitet.
Vi har fått en bra bild av vad som är väsentligast i produktutvecklingen.	Vi kunde haft tydligare namn på sprintkorten som var mer förklarande.	Skriv mer tydliga namn på sprintkorten

B.2: Retrospective för sprint 2

<i>Good</i>	<i>Could have gone better</i>	<i>Improvement</i>
Vi gjorde en tydlig ansvarsfördelning där vi tog oss an de olika korten. På så vis kunde flera uppgifter fortgå samtidigt.	Vi kunde skannat efter flera lösningar från start och inte sikta in oss direkt på vissa områden som var mer komplexa än andra.	Jobba med att rådfråga folk med mer expertis i starten av sprintarna för att öka sannolikheten att landa rätt från start.
Vi tog in extern hjälp och fick tips på vilket lösning som skulle vara bäst lämpat för vårt arbete.		Jobba pair programming och undvika tre. Det blir lätt att en tappar fokus och drar med sig gruppen.
Då vi tillslut fick länken att fungera har vi lagt grunden för framtida sprinter. Detta underlättar planeringern.		

B.3: Retrospective för sprint 3

<i>Good</i>	<i>Could have gone better</i>	<i>Improvement</i>
Intern workshop där vi gick igenom product-backlog för att kontrollera ifall vi saknade viktig funktionalitet. När vi gjorde detta kom vi på funktionalitet som vi under veckan kunde hitta en helhetslösning på, vilket gjorde att vi kunde bocka av flera kort än planerat.	Tidsplaneringen, denna gången överskattade vi tiden det skulle ta att utföra veckans kort och vi lade därför till extra kort i efterhand.	Tydligare variabelnamnsättning och kommentering av kod. Ibland svårt att förstå andras kod då variabelnamn, metodnamn, kodkommentarer etc är otydliga.
Anammande tankesättet "must have", "should have"	Vi specificerade våra acceptance-criteria för hårt under denna sprint.	Var inte för specifika när vi skriver acceptance-criteria. Måste hitta en

och “nice to have” för prioritering av kort vilket gav större fokus på kundvärde.	Tidigare har vi haft för dålig detaljnivå, men denna gång specificerade vi för mycket i detalj att det gjorde att vi låste kortet till en specifik lösning. För att förtydliga skrev vi “Välja allvarlighet via en färg” och låste då oss till att göra lösningen med hjälp av färger, men vi använde i själva verket textbeskrivning. “Välja allvarlighet genom olika alternativ” hade varit bättre.	balans mellan att inte vara för otydlig, men samtidigt inte låsa oss vid en lösning som sedan visar sig vara suboptimal.
Utnyttjade GitHub desktop för att koda parallellt i olika delar av koden. Fungerade sedan smärtfritt att merga samman koden.		I högre grad utnyttja möjligheten att konsultera personer med högre områdeskunskap för att hitta rimliga lösningar till problemen. Även för att skapa en förståelse för vad som är rimligt att klara av under den begränsade tidsramen, utifrån de kunskaper vi besitter.

B.4: Retrospective för sprint 4

<i>Good</i>	<i>Could have gone better</i>	<i>Improvement</i>
Genomförande av integrationstest, vilket påvisade ett antal fel som snabbt kunde åtgärdas.	Fick problem när vi skulle merga koden vilket medförde vissa komplikationer som var tidskrävande att åtgärda.	Se till att merga kod mer kontinuerligt för att undvika att hamna i en situation där väldigt mycket kod behöver mergas samtidigt.
Genomförde acceptanstest, vilket gav väldigt värdefull feedback ur ett stakeholder-perspektiv.		

Appendix C - Resultat av sprint reviews

C.1: Sprint review 1

Sprint element	Acceptance Criteria	Fulfilled?	Comment
Lära sig Trello	Alla gruppmedlemmar har skapat ett Trello konto	Ja	
Lära sig github	Alla gruppmedlemmar har skapat ett git-konto och installerat en fungerande version av github desktop på sin dator	Ja	
Lära oss Android Studio	Alla gruppmedlemmar har installerat en fungerande version av Android Studio på sin egen dator	Ja	Alla lyckades installera programmet utom en gruppmedlem (detta pga hårdvaruproblem). Det beslutades att i projektet kommer gruppmedlemmen i fråga programmera tillsammans med någon annan i gruppen och sprintelementet flyttades till "Done"
Skapa en JButton	En klickbar knapp finns i busschaufförens applikation	Ja	
Visa ett meddelande när knappen trycks på	Ett meddelande visas när man trycker på knappen	Ja	
Skapa någon form av länk	En länk som kan länka samman två applikationer finns	Nej	Denna måste brytas ner i mer funktionalitet.
Skapa ett GUI		Nej	Väldigt oklart nedbrytet kort från start som även saknade acceptance-criteria. Vi har ett trivialt GUI men vi har inte skapat det själva. Vi tar därför bort detta kortet tills vidare.

C.2: Sprint review 2

Sprint element	Acceptance Criteria	Fulfilled	Comment
13. Skapa en vision som anpassas efter kundbehov.	1. Visionen accepteras av alla i gruppen. 2. Innehåller vilket värde som skapas för samtliga stakeholders		Då halva gruppen idag saknas är det svårt att uppnå kriterie 1
12. Ta fram bilder för bussdörrarna	1. Det finns tre bilder som indikerar vänster, höger och mitten i den andra vyn. 2. Det finns tydligt en basbild på en dörr	Ja	
1. Skapa en länk mellan två applikationer	1. Mottagarenheten visar att ett meddelande har tagits emot 2. Det går att skicka ett meddelande mellan två enheter	Ja	Det tog ett tag innan vi fann rätt lösning men sista dagen på sprinten fick vi ihop det, därav ser burndown charten ut som den gör.

12. Skapa en ny view som dyker efter ett första knapptryck	När jag trycker på dörrknappen så ska det tydligt dyka upp en ny meny där det finns tre alternativ som jag kan trycka på. Alternativen ska vara vänster, mitten eller höger dörr och när jag tryckt på denna så dyker det upp ett bekräftelsemeddelande.	Ja	
--	--	----	--

C.3: Sprint review 3

Sprint element	Acceptance Criteria	Fulfilled	Comment
6.3 Skicka information om tid vid felrapportering	När jag rapporterar ett fel dyker tiden för felrapporteringen upp i databasen med felmeddelandet.	Ja	Gruppen programmerade själv fram en metod som rapporterade tiden med millisekunders noggrannhet. Men detta ansågs överflödigt och gruppen lånade därför en metod som hittades på nätet för att skicka en mindre noggrann tid.
6.4 Välja allvarlighet av fel	Går att trycka på ett fel och sedan välja en färg för att beskriva allvarligheten av felet	Ja	Val av kritikalitet löstes inte med färger utan med en textbeskrivning. Därav uppfylldes inte kriteriet fullt ut, men samma funktionalitet återfinns.
8. "Övrigt fel"-ruta finns	Man klickar på övrigt rutan och får fram ett fält där en sträng kan skrivas in. Denna inskrivna text skickas med i felmeddelandet..	Ja	
14. Testa och förbättra befintlig kod	-Findbugs accepterar koden -Acceptens test med stakeholders -Verifieringstest/blackboxtest (det som kommer ut i databasen är vad som förväntas av oss själv) Unit test är skrivet	Nej	Allt utom unit test klart. Problematik med att förstå hur vi ska göra unit-testet. Gruppen har adresserat "områdeskunniga" för att få råd kring hur unit-testet skall utformas. Detta kort läggs över i nästa sprint-backlog.
6.1 Skicka med information om vilken buss vi befinner oss i	-Vilken buss man sitter i skickas med till databasen	Ja	

C.4: Sprint review 4

Sprint element	Acceptance Criteria	Fulfilled	Comment
15. Skapa ett bekräftelsemeddelande	-Rutan dyker upp när man klickar på skicka rapport -Rutan visar felträdet som har registrerats -Det ska finnas alternativ om man ska skicka felmeddelandet eller ej	Ja	
6. Övrigt fel- ruta finns	-Finns en ikon som säger Övrigt när man ska välja typ av fel -Går att klicka på ikonen och sedan förs man vidare till en ruta där man skriver en egen kommentar om felet	Ja	Blivit knasigt med namngivandet av detta kort.

	-Kommentaren skickas med i felmeddelandet		
5. Användarvänligt gränssnitt som är lätt att navigera	-Finns enligt gruppen lättförståelig grafik för menyknapparna -Finns ett enligt gruppen snyggt gränssnitt med fina färgkoder -Finns en bild på förstasidan i appen som enligt gruppen hör samman med appens syfte	Ja	
14. Testa och förbättra befintlig kod	-Findbugs accepterar koden -Acceptens test med stakeholders -Verifieringstest/blackboxtest (det som kommer ut i databasen är vad som förväntas av oss själv) Unit test är skrivet	Ja	