Chalmers tekniska högskola
DAT255 - Software Engineering Project
HT 2017

Reflektionsrapport

Författare: Miranda Bånnsgård, Hanna Carlsson, Ludvig Ekman, Jonathan Gildevall, Peter Gärdenäs, Sara Kitzing, Madeleine Lexén, Tobias Lindgren, Erik Magnusson, Elina Olsson, Julia Ortheden, Johan Wennerbeck

Inledning

I arbetslivet är det vanligt att arbeta agilt, bland annat genom att använda sig av scrum. Det är en viktig kompetens att ha som förbereder en inför arbetslivet. Därför har vi i samband med ett projekt för att skapa kolonnkörning (*platooning*) till en MOPED använt oss av *scrum* i arbetet med att utveckla detta. Att jobba med *scrum* har många fördelar, t.ex. ger det en struktur på arbetet som gör att det går relativt lätt och snabbt att byta riktning på projektet och leverera färdiga saker till en produktägare. Visionen är att genom att arbeta med scrum kunna utveckla en adaptiv farthållare och ett sätt att få fordonet att följa efter i sidled.

Applicering av *scrum*

Att ta på sig ett nytt arbetssätt har inte varit enkelt, men under kursens gång blev arbetet successivt bättre. Ett bra exempel på detta är *daily scrum*.

Under hela projektets gång försökte vi träffas minst en gång varje dag för *daily scrum*, där vi snabbt gjorde en runda runt bordet med frågorna *Vad har du gjort*, *Vad ska du göra idag* och *Behöver du hjälp*. Den sista frågan ändrade vi efter ett tag till att man skulle beskriva ett aktuellt problem man har, och på så sätt kunde man svara på den frågan, och få hjälp, trots att man kanske inte hade svarat att man behövde hjälp om frågan var ställd som den var från början.

Syftet var att alla i gruppen skulle vara uppdaterade om hur det går samt samarbeta och hjälpa varandra om någon kört fast på en uppgift. De första veckorna bestämde vi att man inför varje *daily scrum* skulle skriva ner vad man skulle säga, så att den som var borta kunde läsa på vad som händer. Med tiden insåg vi att det snarare var ett hinder än något som underlättade, eftersom det krävde en liten bit extra jobb varje dag. Fokus ska ligga på att ta med sig något konstruktivt från mötet till resten av arbetsdagen, och om man är sjuk kan man ändå inte jobba den dagen. Ett annat skäl till att skriva ner vad vi skulle säga var att vi skulle kunna följa vad vi arbetat med, för att kunna göra en rejäl utvärdering, men Trello fungerade redan ganska bra till detta.

Kursboken[1] tipsade om att använda *burndown chart* under *daily scrum*, men eftersom vi inte hade en och samma plats för våra möten var det svårt att fixa något fysiskt. Mot slutet diskuterade vi om man kunde använda Trello även för detta, men kom aldrig fram till någon lösning.

Eftersom vi träffats varje dag har vi snabbt kunnat hjälpa varandra komma vidare genom att svara på frågor, omgruppera eller hänvisa till andra grupper som löst liknande problem. Men då vi är en stor grupp där alla inte läser samma kurser är det svårt att hitta tillfällen där det passar för alla att träffas. Det som var mest effektivt var då att ha *daily scrum* under rasten på kursens föreläsningar. Detta eftersom alla är där och det bidrar till bra sammanhållning för gruppen. Det var lite svårare att få till *daily scrum* de dagar när det inte varit gemensam

[1] Kniberg, Henrik. Scrum and XP from the Trenches. 2 uppl. C4Media, 2015. (s.56)

föreläsning, men eftersom alla var måna om att samarbetet och projektet skulle flyta på så löste sig de mötena oftast. Någon gång var vi tvungna att ha möte via Skype, men vi kände att det inte gav lika mycket som att träffas i person, och det är enkelt att förstå kurslitteraturens resonemang med att i möjligaste mån undvika en geografiskt utspridd grupp.

Roller, samarbete och det sociala kontraktet

Samarbete

Magin med scrum finns i det nära samarbetet varje dag och det är bra att så ofta som möjligt sitta tillsammans och jobba.[2] Det var dock inte alltid lätt att hitta en tid och plats när alla i gruppen kunde ses samtidigt och därför delade vi ofta upp oss i mindre grupper som fokuserade på varsin *user story*. Vi blev dock bättre på att jobba alla tillsammans under kursens gång, och framförallt de sista två veckorna såg vi till att alltid ha ett grupprum bokat där alla gruppens tolv medlemmar fick plats. Även när någon pluggar en annan kurs kunde man kort avbryta och fråga något om varför personen implementerat en lösning på ett visst sätt. Det nära arbetet har gjort allt enklare, t.ex. att fråga om hjälp, delge problem och komma med lösningar.

Både *daily scrum* (se föregående rubrik) och att dela grupprum har också förbättrat gruppdynamiken, och vi känner att det gjort oss mer sammansvetsade.

Vi samarbetade även mycket med andra grupper, t.ex. under handledningstillfällena då vi gick runt till andra grupper och pratade om våra tekniska lösningar och jämförde med deras. T.ex. i början av utvecklandet av vår pythonkod-lösning för ACC, berättade en annan grupp som kommit längre att lösningen var alldeles för långsam och det gjorde att vi valde att arbeta vidare med en annan lösning istället. Hjälpen vi fått av andra grupper och samarbetet mellan de olika grupperna under handledningen har varit av högt värde för oss. Den slutgiltiga lösningen vi använde oss av för ACC är baserad på en annan grupps kod som vi fick använda oss av. Samarbetet mellan grupperna fortsatte även vidare på en gemensam slack-kanal, mer om detta står under rubriken “Strategier för nya verktyg och teknologier”.

Socialt kontrakt

Vi skrev ett socialt kontrakt inför första sprinten men sedan lämnade vi det orört och öppnade det inte förrän i slutet av projektet. Mycket av det som vi stod i vårt sociala kontrakt var självklart för alla (som att t.ex. komma i tid och kommunicera mycket) och det har inte uppstått situationer som gjort att vi behövt ta hjälp av kontraktet. Det har underlättat arbetet då all kraft kunnat läggas på att komma framåt i projektet och inte lösa konflikter.

[2] Kniberg, Henrik. *Scrum and XP from the Trenches*. 2 uppl. C4Media, 2015. (s.156)

Roller

I början av projektet tänkte vi rotera rollen *scrum master* inom gruppen. Syftet med det var att flera personer skulle få testa på rollen, att vi skulle få nya perspektiv på rollen och att inte en person skulle göra samma sak hela projektet. Ytterligare en orsak till att vi roterade *scrum master* var att gruppen skulle bli mer *cross-functional* eftersom det i *scrum* är bra om flera personer kan täcka samma område. Ökad *cross-functionality* bidrar till att gruppen kan fokusera på det som behöver prioriteras och att alla inte endast är låsta till sitt ansvarsområde. Eftersom olika ansvarsområden kan vara mer eller mindre omfattande under olika delar av projektet är det bra att kunna omfördela gruppmedlemmarna. Om någons ansvarsområde för tillfället är litet kan den personen hjälpa någon som har ett större område.[3]

Scrum masterns område var dock alltid aktuellt och dennes ansvarsområde gick aldrig överstyr så att någon annan person behövde hoppa in och hjälpa till. Därför behöver inte mer än en person kunna agera *scrum master*. Vi upplevde även att det var väldigt svårt att komma in i rollen när man bara hade en vecka på sig. Dessutom blev det svårt för den nya *scrum mastern* att plötsligt behöva skaffa helikopterperspektiv över gruppen. Detta gjorde att vi halvvägs in bestämde oss att utse en och samma *scrum master* för resten av projektet. Vi upplevde att detta fungerade mycket bättre eftersom det bland annat bidrog till att gruppen fick en bättre struktur och att det fanns någon med övergripande koll.

Ett till misstag som gjordes vid projektets start var att gruppens uppgiftsbeskrivning för *scrum master* inte var tydlig eller utbredd nog. I början var *scrum masters* roll endast att vara ordförande på interna möten och delta på *scrum of scrums*. Vi insåg senare att *scrum master* inte hade så bra koll på vad övriga medlemmar i gruppen gjorde, och kunde därför inte alltid följa med på diskussionerna i *scrum of scrums*. Detta åtgärdades genom att ta ett gemensamt beslut om att *scrum master* skulle lägga tid på att hålla koll och förstå bättre vad alla arbetade med, så att hen kunde delta bättre i de tekniska diskussionerna.

En annan sak vi kom på saknades var någon som hade ansvar för att skapa och uppdatera ett dokument som hade med strukturen av projektet att göra, alltså någon som fixade *tech stories*. Denna uppgift föll på ett enkelt sätt på *scrum master*, men hade lika gärna kunnat ligga som ansvar för någon annan i gruppen. För att tackla detta problem bättre nästa gång hade vi behövt uppmärksamma problemet tidigare, och sett till att utse ansvariga under *sprint planning*.

När man jobbar med *scrum* är det viktigt att produktägaren bland annat är med på *sprint plannings* för att till exempel bestämma vilken prioritetsordning *user stories* ska vara i.[4] Eftersom vi inte hade så mycket kontakt med vår produktägare utsåg vi en produktägar-*proxy* till våra *sprint plannings* för att kunna hjälpa oss fatta beslut. Vi använde oss dock knappt alls av detta, dels för att vi tyckte uppgiften var tydlig från början och hade ganska lätt att själva avgöra prioriteten på våra *user stories*. Vi hade även mycket

[3] Kniberg, Henrik. *Scrum and XP from the Trenches*. 2 uppl. C4Media, 2015. (s.116)
[4] Kniberg, Henrik. *Scrum and XP from the Trenches*. 2 uppl. C4Media, 2015. (s.17)

problem med t.ex. hårdvaran i början av projektet som hindrade oss från att leverera något av värde, vilket gjorde att *proxyn* inte hade så mycket att agera efter. Den riktiga produktägaren ändrade dessutom inte så mycket krav, utan målet var alltid detsamma.

För att vara helt säkra på att vi prioriterade *user stories* enligt produktägarens önskemål, hade vi till exempel kunnat komma med färdiga beskrivningar på dessa till honom, och pratat med honom om dem. Att be honom direkt prioritera dem hade varit farligt, eftersom gruppens åsikter, tolkningar och input inte ska förbises, de är för viktiga.

Strategier

Vi arbetade mycket med parprogrammering för smidigare problemlösning och samarbete. Genom att sitta och programmera på en gemensam dator blir den gemensamma förståelse för koden större samt att man kan dra nytta av varandras kunskap på ett smidigt sätt. Nackdelen med parprogrammering upptäckte vi dock var att de var lätt för en person som kunde mer att hen skriver koden och den andra inte hänger med. Detta försökte vi undvika genom att hela tiden turas om med vem som skriver koden.

En annan strategi vi använde oss av var att boka grupprum där hela gruppen fick plats och kunde sitta och arbeta tillsammans, detta nämns tidigare under rubriken “Samarbete”. Under kursens gång hade vi ett gemensamt grupprum ett par gånger i veckan, förutom de två sista veckorna då vi insåg hur mycket det tillförde att samarbeta med direkt kontakt. Då bokade vi grupprum varje dag, och detta hade vi gjort varje dag under hela projektet om vi vetat bättre i början. Som nämns under rubriken *“Daily scrum”* la vi även stor vikt vid våra *daily scrum* vilket underlättade samarbetet och arbetet genom att uppdatera alla på hur det går och kunna få hjälp med problem.

Distribuering av tid

Under *sprint planning* delades *user stories* ut till mindre grupper, med gruppstorlek som berodde på hur omfattande uppgiften var. Uppgifterna och grupperna valdes utefter intresse och förkunskaper. Att distribuera tiden och ansvaret ansåg vi viktigt redan från början av projektet, därför valdes ett KPI som gick ut på att försöka fördela arbetet så lika som möjligt på alla mindre grupper. Ju mindre skillnad desto bättre.

Det har även varit svårt att få en rättvis arbetsfördelning då olika uppgifter visat sig vara olika svåra att lösa. Att få kod att köra på MOPEDen var en flaskhals större delen av projektet, och de som arbetade med detta problem lade ofta ner mer timmar än andra. Vi försökte med lösningar på detta under projektets gång, med delvis rotation av den tyngre uppgiften. En hel rotation, att alla som får uppgiften inte tidigare arbetat med den, hade varit ännu jobbigare eftersom de hade behövt sätta sig in i uppgiften från början. Det optimala hade varit att hitta en smidig lösning på detta problem, men det är snarare önsketänkande än något konkret. Det vi tar med oss är att vara ödmjuka inför problemen. När vi insåg att arbetsbördan blev ojämn började vi föra tidslogg för att synliggöra problemet, vilket vi borde gjort från början.

Effort, velocity och *task breakdown*

I första *sprint planning*-mötet delade vi upp uppgifter i *user stories* och uppskattade vår *effort* i antal timmar. Under projektets gång insåg vi dock att det är bättre att estimera vår *effort* i en relativ skala. Detta eftersom det är svårt att veta exakt hur lång tid en uppgift kommer ta, medan det är lättare att uppskatta hur stor en *user story* är jämfört med en annan.

Det har också varit lite svårt att få till *user stories* som följt INVEST (speciellt den första delen "*Independent*") eftersom *user stories* varit beroende av andra. Vi valde att prioritera "*Small*" aspekten över "*Independent*" i INVEST då stora oberoende stories inte tillät fler arbeta samtidigt än vid små stories med beroenden, medan de små fortfarande var lättare att estimera, prioritera och slutföra. Men det har därför varit knepigt att jobba med vissa *user stories* eftersom man inte kunnat gå vidare med dem förrän andra *user stories* är klara, och helt enkelt jobba på en annan uppgift tills det man väntat på löst sig. En entydig lösning på att kunna skapa självständiga *user stories* är svår att hitta, förutom att gruppen helt enkelt behöver vänja sig mer vid *scrum* och helt enkelt bli bättre på det med tiden. Vi hade kunnat lägga ned mer tid på planeringen, eller avsätta uppgiften att hitta på självständiga *user stories* som en egen *tech story*.

Vid de senare sprintarna lyckades vi bättre med få våra stories oberoende då flera flaskhalsar antingen löstes eller så skapades det strategier för att komma runt dem, såsom att skapa en *mockup* i simulatorn tills den slutgiltiga utvecklingsmiljön var klar. De *user stories* vi inte lyckades göra helt fristående fick smågrupperna ansvar för att definiera de gemensamma aspekterna såsom in-/utdata vilket tillät grupperna att arbeta parallellt trots att varje *user story* inte var fristående.

Vår *velocity* har varierat ganska mycket mellan olika sprintar. Det beror bland annat på att vi hade svårt att skapa små vertikala *user stories* i början, antingen blev varje *user story* för stora eller så blev de beroende av andra. Det, tillsammans med oförutsedda problem såsom hårdvarufel, gjorde att vi hade svårt uppnå någon *velocity* alls de två första veckorna. Istället kom väldigt mycket *velocity* under *sprint* 3-4 när vi slutförde det vi påbörjade i *sprint* 1-2. Det är nog alltid svårt att skapa värde i de första *sprintarna* under ett projekt, eftersom en stor del i början är att lära sig vad man ska göra och hur allt fungerar. Om vi hade gjort uppgiften *att förstå* i sig till en *user story* hade vi skapat värde till gruppen, och på så sätt genomfört värdefulla *sprintar*. Dessa *user stories* innefattar en hel del arbete, men hade däremot inte varit vertikala, så det kanske inte hade varit en bra idé ändå eftersom de inte levererat värde till produktägaren.

Det kunde även vara svårt att veta i vilken sprint något faktiskt var slutfört eftersom våra *definition of done* (DoD) många gånger kunde tolkas på flera sätt. För att minska det problemet och samtidigt göra våra *user stories* mindre delade vi upp de *user stories* som hade en tolkningsbar DoD i flera *user stories*. Till exempel kunde "Hitta cirkeln med kameran på MOPEDen ofta" delas upp i två *user stories* med DoD "hittar cirkeln 60% av gångerna" och en annan med "hittar cirkeln 90% av gångerna".

Reflektion av *sprint retrospective*

Det optimala för våra *sprint retrospectives* hade till att börja med varit att alla medverkade på alla möten, helst även produktägaren. Detta eftersom det som tas upp ofta är värdefull information och därför nyttig för alla att ta del av[5]. Just i vårt fall var detta svårt att lösa eftersom vi inte arbetar med projektet på heltid vilket medför att det är svårt att hitta tider då alla gruppmedlemmar är lediga samtidigt. En lösning på avsaknaden var att utse en *proxy* i gruppen. Mer om detta finns i "Roller, samarbete och socialt kontrakt".

Det var först i fjärde *sprint retrospective* vi genomförde mötet som man ska enligt *scrum*. Boken tipsar om att man ska ha en avslappnande, tillåtande och mysig miljö. Man kollar på backloggen och velocity och summerar sprinten. Runda bordet är bra, med vad som gått bra, dåligt och förslag på förbättring. Man diskuterar lösningar, och prioriterar vad man ska fokusera på att förbättra tills nästa gång, eftersom man kan inte göra allt. Mot slutet konkretiserar *scrum master* vad som ska tas med från mötet, vad vi ska förändra för att nästa sprint ska bli bättre.[6]

Detta har inte riktigt följts för vår del, främst eftersom vi åtminstone i början av projektet var dåliga på att strukturera upp våra möten utefter *scrum*. Detta beror delvis på att vi helt enkelt inte hade tillräckligt bra koll på vad ett *sprint retrospective*-möte skulle innehålla. Vi utformade det så bra som vi kunde för att sedan uppdatera strukturen när vi hade bättre koll. I *sprint retrospective* 1-3 hade vi runda bordet där alla fick säga sin åsikt om *sprinten*. Därefter hade vi *sprint planning* direkt, och på detta sätt kom lärdomarna från vår utvärdering med till nästkommande sprint. Så det främsta som skiljde sig från våra möten, jämfört med hur de borde varit, var en tydlig struktur där man sammanfattar vad vi åstadkommer, utvärderar detta och sedan knyter samman det som tagits upp innan man fortsätter med nästa punkt.

Efter varje *sprint retrospective* är det bra att ha en paus. Man borde alltså inte börja med *sprint planning* direkt efteråt då man dels behöver vila för att orka prestera men också för att hinna bearbeta alla tankar och saker man tog upp under sprint retrospective innan man börjar planera nästa sprint[7]. Detta var inget som vi följde till en början, främst för att vi tyckte att det var svårt som det var att hitta tider då så många som möjligt kunde närvara. Det kändes också som att vi förlorade för mycket tid på det då våra sprintar var rätt korta som de var och de då skulle bli ännu kortare med vila emellan. Sen hade vi från början inte heller tillräckligt bra koll för att veta att man helst inte skulle köra *sprint planning* och *sprint retrospective* direkt efter varandra.

Mot slutet av projektet införde vi dock detta i vilket fall. Det är trots allt en del av *scrum* och vi kände att vi ville applicera *scrum* på ett så bra sätt som möjligt. En bidragande faktor var också att vi inte riktigt hann lägga den tiden vi borde på varje enskild del under mötet när vi

[5] Kniberg, Henrik. *Scrum and XP from the Trenches*. 2 uppl. C4Media, 2015. s. 85-86.
[6] Kniberg, Henrik. *Scrum and XP from the Trenches*. 2 uppl. C4Media, 2015. s.85.
[7] Kniberg, Henrik. *Scrum and XP from the Trenches*. 2 uppl. C4Media, 2015. s. 92.

enbart hade två timmar för det. Vi ändrade då så att *sprint retrospective* hölls torsdag eftermiddag och *sprint planning* under fredag förmiddag. På så sätt fick vi mer tid till varje del och samtidigt en kväll för att bearbeta det som diskuterat.

Det hade gynnat oss att ha en scrum-fascist under hela projektet. Denna person skulle då varit noggrann att påpeka så fort vi inte gjorde det vi borde för att se till så att *scrum*-konceptet efterföljs[8]. Sen kanske vi inte hade valt att följa scrum till punkt och pricka ändå eftersom det inte alltid fungerar för alla grupper[9]. Detta hade kunnat varit något som underlättat för att vi skulle kunna ha ännu bättre *sprint retrospectives*, och troligtvis gjort att vi strukturerat upp våra *sprint retrospective*-möten tidigare. Vi skaffade oss en scrum-fascist efter gästföreläsningen med *8 dudes in a garage*.

En sak som skulle kunna ha hjälpt oss att komma en längre bit på vägen hade varit att ha en ansvarig för att medverka på andra gruppers *sprint retrospectives*. Detta är något som är bra att använda sig av i arbetslivet eftersom man delar med sig av varandras slutsatser och förbättringsförslag på det sättet, så att alla kommer längre på det[10]. För vår del hade det kunnat vara nyttigt att få höra från andra grupper vad som fungerat och inte. På så sätt hade vi kunnat utveckla vår applicering av *scrum* utifrån deras åsikter också då vi själva inte hade tid att testa så många olika metoder under denna korta tid. Gästföreläsnigarna fyllde delvis detta syfte, men att höra från våra kursare hade troligtvis gett ännu mer. Detta eftersom just *sprint retrospective* är den viktigaste delen i scrum, utan den riskerar man att göra samma misstag igen[11].

Reflektion av *sprint review*

Våra *sprint reviews* var en demo för produktägaren, då möjligheten gavs att komma med anmärkningar och kommentarer på vad som visades upp. Vi berättade för PO om hur vi prioriterat denna *sprint* och vad vi jobbat med för att se vad produktägaren tycker om riktningen vi går, även om vi inte hade något konkret att visa upp. *8 dudes in a garage* nämnde att det är bättre att visa upp något halvfärdigt än inget alls då du i tidigt skede kan få respons på ifall det är värt att fortsätta utveckla produkten, och det gjorde att vi fortsatte att redovisa halvfärdiga saker. Under demonstrationen kunde produktägaren också säga till om nya krav eller omprioriteringar som vi skulle ta hänsyn till.

Detta tog vi sedan med oss till vårt sprintmöte (då vi hade både *sprint planning och sprint retrospective* under samma möte) vi hade direkt efter. Där gick vi även laget runt och sa ifall vi var klara med vår *user story*, hur långt vi kommit med den och vad som fanns kvar att göra. På så sätt såg vi hur nära vårt sprintmål vi kommit, hur många *user stories* vi klarat av och hade kvar. Detta tog vi hänsyn till under *sprint planning* inför nästa sprint.

[8] Föreläsning: 8 Dudes in a Garage.
[9] Föreläsning: 8 Dudes in a Garage.
[10] Kniberg, Henrik. *Scrum and XP from the Trenches*. 2 uppl. C4Media, 2015. s. 87.
[11]Kniberg, Henrik. *Scrum and XP from the Trenches*. 2 uppl. C4Media, 2015. s. 84.

De första veckorna uppfyllde vi inte vårt sprintmål, och hade i därför princip samma mål i tre veckor. I början klarade vi inte mycket i vår *product backlog* men ju längre in i projektet vi kom desto mer hittade vi vår nivå och började leverera mer värde till produktägaren samt uppfylla våra sprintmål.

I början av projektet var vi flitiga med att göra *user stories,* men efter ett tag tappade vi *scrum*-tänket och fokuserade mer på MOPEDen. Första gången vi skrev *user stories* tog det väldigt lång tid. Följden av detta var att vi inte ville lägga vår värdefulla mötestid på att endast definiera *user stories*, så nästa möte blev det mer att vi delade ut uppgifter/ansvarsområden än färdiga *user stories* med tillhörande *tasks*. Detta ledde till att man själv inte hade lika bra koll på exakt vad man behövde leverera innan sprinten var slut. Mötena gick fortare genom att göra på det här viset, men svårare att utvärdera i *sprint retrospective* då vi inte hade klart för oss exakt vad vi skulle göra och ha gjort. En orsak till att vi frångick konceptet var att vi tyckte de var väldigt svårt att estimera *effort* för de olika uppgifterna eller *user stories*. Detta märkte vi redan första sprinten då vi la ner mycket tid på att tillsammans göra uppskattning och skriva *user stories*. Under sprinten märkte vi ganska snabbt att det var väldigt mycket svårare och krångligare än vi först trott eftersom vi inte var insatta i hur krävande projektet var. Ett exempel på detta var att vi satte en låg estimering på *user storyn* "att få igång och använda kameran", vilket visade sig ta flera *sprintar*.

Eftersom det tog väldigt lång tid att skriva och estimera alla *user stories* i storgrupp bestämde vi att varje smågrupp skulle komma på sina egna *user stories* med estimering och *tasks*. Det sparade väldigt mycket tid under planeringsmötet, dock glömdes detta ofta av och många gånger hade vi inga *user stories*. I slutet av sprinten i *sprint review* var det därför svårt att veta vilka *user stories* som var avklarade och hur många poäng de genererat. Eftersom vår grupp var så stor var *user stories* väldigt svårt, vi var tvungna att väga kvalité av en *user story* gentemot att alla skulle få vara med och skapa den, och förstå vad den faktiskt innebar.

För att återgå mer till *scrum* började vi under sprint fem skriva *user stories* i storgrupp igen för att dela ut till olika smågrupper på *sprint planning*. Vi upplevde att det var mycket lättare att estimera *effort* på de olika *user stories* på detta möte än det första sprintmötet, främst eftersom vi då var mer insatta i projektet. Vi valde dock att delegera ut till varje liten grupp att själva skriva *tasks* för *user storyn* de blev tilldelade. Vi hittade ett mellanting där vi tillsammans skrev *user stories* men skrev *tasks* själva. Det gjorde att varje grupp hade bra koll på vad dom skulle göra och fick själva ansvara för att dela upp arbetet i *tasks*. Detta gjorde också det mycket lättare att i *sprint review* se ifall vi uppnått vårt sprintmål samt se ifall vi uppnått vår *velocity*. Inför sprint fem återinförde vi också användandet av en *sprint backlog* med hjälp av Trello vilket gjorde det lättare att hålla koll på vad vi var klara med och inte.

Strategier för att använda nya verktyg och teknologier

Angripa samma problem från flera håll

Det var en stor tröskel för oss att ta oss över innan vi kunde använda oss av *plugins* på MOPEDen. Det gjorde att vi inte kunde testa och lära oss hur MOPEDens hårdvara fungerade. Därför började en delgrupp programmera funktionalitet i Python för att snabbt kunna leverera värde till både oss och produktägaren. Även om vi inte skulle använda Python-koden i den slutgiltiga koden så kunde vi med hjälp av den testa hårdvarans begränsningar i ett tidigare skede i processen jämfört med om vi endast använt oss av plugins. Genom att ha testat Python-koden upptäckte vi t.ex. att sensorn inte alltid genererade korrekta värden.

I och med alla tekniska problem som gjorde att vi inte kunde testa på den faktiska MOPEDen, utvecklade vi en mockup som vi körde i simulatorn. Mockupen var en samling av förbestämda, påhittade värden (de agerade in-data från sensorer) som gjorde att vi kunde testa vår kod i simulatorn. Det var väsentligt att ha ett system där vi kunde testa vår kod oberoende om MOPEDen fungerar eller inte så att vi kunde gå vidare.

Reflektion och utvärdering

Varje gång vi tillämpar nya verktyg och teknologier är det viktigt att vi reflekterar och utvärderar dem. Bara för att de fungerar bra för någon annan kanske det inte fungerar bra för oss i detta sammanhang. Detta har vi gjort varje vecka genom t.ex. sprint retrospectives, handledarmöten på Lindholmen och nu under reflektionsrapporten och har hjälpt oss att hela tiden förbättra våra strategier.

Slack

Vi har haft en intern slack inom gruppen med diverse olika kanaler som vi har använt för att planera, diskutera och informera om saker inom projektet. Vi har b.la. haft kanaler för olika user stories. Eftersom gruppen var så stor så underlättade detta kommunikationen då man kunde skapa en kanal med endast de som var inblandade i en specifik *user story*. Man kunde dessutom lätt byta grupp och ansluta till en ny kanal om man grupperade om. Vi har även delat med oss av sammanfattningar från kurslitteraturen på slack så även de som inte har läst boken kan ta del av fakta om scrum.

Vi har även haft en gemensam slack för kursen där vi har kommunicerat med andra grupper. Det har varit till stor nytta eftersom man har kunnat få och ge hjälp, dela och ta del av andras

lösningar, ställa frågor till lärare samt varit ett verktyg för att distribuera MOPEDer och laddare.

Drive

Vi har använt Google Drive för att samla dokumentation från b.la. föreläsningar, möten, inlämningsuppgifter samt vår *product backlog*. Vi ansåg att detta var den bästa och enklaste lösningen på hur vi ska dela projektrelaterade filer då alla redan har tillgång till Drive och det är enkelt att skapa nya och ändra befintliga dokument.

Vi har även haft en gemensam drive för alla projektgrupper där vi delat våra kontaktuppgifter och git-repon för att kunna dela med oss av kod.

Trello

Vi har använt oss av Trello för att registrera närvaro på våra *daily scrum*-möten. Det har varit till hjälp när vi ska utvärdera vårt KPI som mäter mötesnärvaro. Vi tänkte att det var smidigt eftersom det finns en inbyggd checklista som man kan använda för att markera närvaron för varje person och dag. I efterhand insåg vi att vi hade kunnat registrera närvaron på ett smidigare sätt, t.ex. i excel. Där vi också kunnat skriva en formel för att automatiskt räkna ut KPI:t i stället för att manuellt överföra information från olika ställen för att göra beräkningarna.

I början tänkte vi även använda Trello som en form *scrum board* där vi kunde lägga in våra *sprint backlogs*. Detta glömdes dock bort i de första sprintarna och först i sprint 5 la vi till en *sprint backlog*. Det var svårt att i början följa scrums alla delar till punkt och pricka, och *sprint backlog* var en del som helt enkelt prioriterades bort för att hinna med annat. Det bättre alternativet kanske hade varit att lägga ännu mer tid på scrum i början, eller iallafall följa upp detta under den andra veckan.

Samtidigt som *sprint backlog* infördes skapade vi även en *backlog* för de uppgifter vi hade utöver det som ska göras i sprintarna då vi insåg att det var ett bra och smidigt sätt att få en uppsikt över vilka uppgifter som ska göras, vilket stadie de är i och vem som ansvarar för att göra klart uppgifterna. Denna *backlog* användes mycket i slutet, så att man kunde göra något när man var klar med sin *user story* utan att behöva vänta på nästa *sprint planning* eller ens *daily scrum*.

Git

Vi har använt oss av git för att dela kod med varandra samt för att samla all kod på samma ställe. Att använda oss av git som versionshanteringsprogram var ett medvetet beslut då alla i gruppen har tidigare erfarenheter av och kan hantera git. Vi valde att installera git på

MOPEDen vilket gjorde utvecklandet av kod smidigt när alla kan skriva på sina egna datorer. Användandet av git gjorde det smidigare och lättare att jobba ihop.

Vi kom inte igång med kodandet förrän sent i kursen, och när vi väl gjorde det var det mycket arbete direkt. Detta ledde till att vi inte gick igenom ett gemensamt förhållningssätt till git, vilket ledde till problem i slutet. Då buggfixade vi hela dagen innan redovisningen, genom att ta bort eller förbättra kod som programmet *bugfix* sa var dålig, och *pushade* allting direkt till *master*. När MOPEDen inte längre ville starta fick vi jobba hela kvällen med att hitta den *commit* som orsakade felet.

Vi borde tagit tag i att komma fram till ett förhållningssätt tidigt, även om det inte kändes nödvändigt just då. Detta kommer dock ofta upp, att inte ta tag i saker tidigt som sedan leder till mer arbete i längden. Vi borde haft en bättre plan till hur alla av oss hela tiden skulle använda oss av olika *branches* och att vi inte inte *pusha* upp något förrän det testats att koden verkligen fungerar. Och detta skulle vi bestämt tidigt, redan när hela gruppen samlades först första gången. Då hade vi sluppit dessa tidskrävande felsökningar.

Reflektion på relationen mellan prototyp, process och intressent

Process - Prototyp
Vi trodde till en början att vi direkt skulle kunna komma igång och bygga på en prototyp som levererade värde för PO. Men istället var vi tvungna att lägga mycket tid under de första sprintarna på att lösa tekniska problem med plattformen. Det var inte förrän de sista sprintarna som vi kunde leverera den sortens värde vi först hade räknat med. I en perfekt värld syns processens arbete direkt i prototypen, men en sådan utvecklingsmiljö har man sällan utan det finns flera hinder man måste förbi först.

Prototyp - Intressent
Vi hade tänkt att intressenten skulle ge oss löpande input på prototypen och att vi sedan styrt utvecklingsarbetet utefter det. Eftersom det tog många sprintar innan vi kunde uppnå några av intressentens krav för prototypen fick vi inte jättemycket konstruktiv feedback varje vecka. Efter varje sprint visade vi ändå upp det vi hade för intressenten vars input sedan påverkade nästa sprints process. Men det var sällan intressenten tyckte vi skulle ändra något eller tillägga någon funktion, utan det var mest uppmuntran att fortsätta i samma riktning. Vi hade velat få fram en liten *shipment* av något med värde efter varje sprint så man snabbt får reda på om man är på rätt väg eller inte, detta är mycket viktigt när man jobbar agilt så att man snabbt kan överge arbete som inte är av intresse för intressenten.

Process - Intressent
Enligt scrum så bör intressenten vara den som lägger upp prioriteringarna i *sprint backlog*. I detta projektet hade vi inte möjlighet att göra det och vår första lösning på problemet var att använda en *proxy*. *Proxyns* uppgift skulle vara att agera intressent under våra möten, i praktiken blev det inte riktigt så eftersom det var en svår roll att ha. Att för en person snappa

upp vad intressenten vill under korta möten med honom. Istället försökte vi gemensamt prioritera vår *backlog* efter vad vi trodde intressenten hade velat, baserat på de svar vi fått när vi visat prototypen.

Ju längre in i projektet vi kom desto mer samarbete blev det grupper sinsemellan, detta är en utveckling av processen för att bättre kunna nå intressentens förväntningar. Eftersom allt intressenten ville se var flera MOPEDer som kör i *platooning*, genom att dela med oss av våra lösningar mellan grupperna gick utvecklingen mot detta mål mycket snabbare. Som kontrast mot detta kan man jämföra med legoövningen vi gjorde i början av kursen. Där vi tävlade mellan grupperna och inte samarbetade utan samlade på sig så mycket lego som möjligt för den egna gruppen, vilket höll tillbaka alla gruppernas gemensamma arbete.

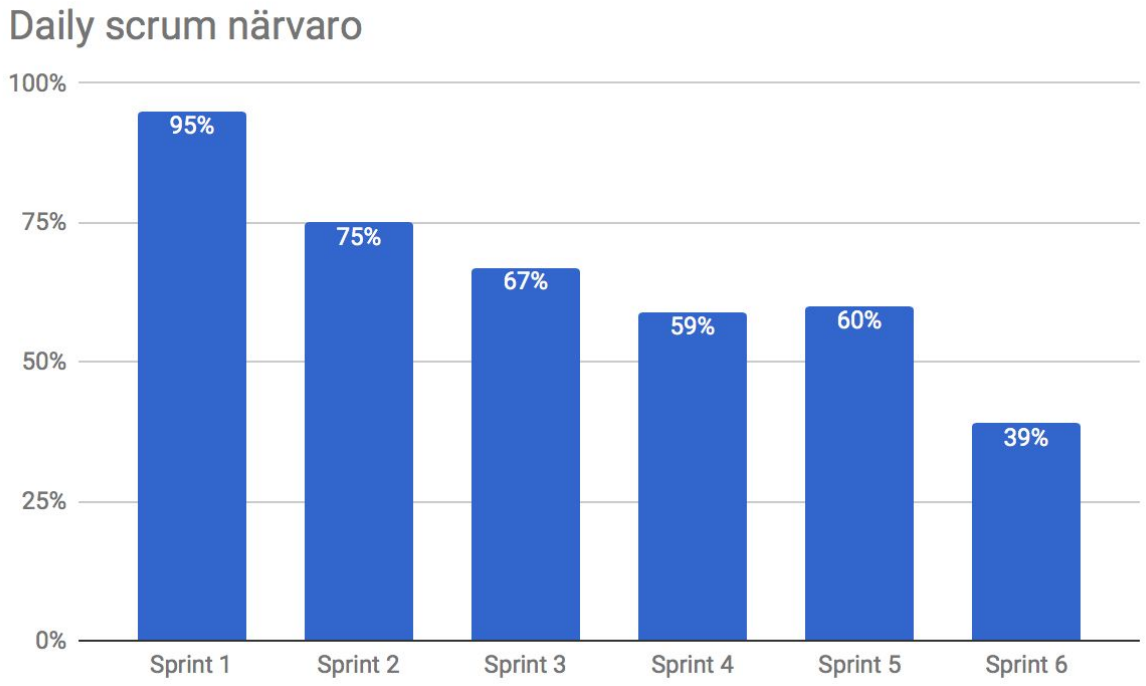
Utvärdering av D1 - D4

D1

I början av projektet hade vi en dålig uppfattning om vad projektet innebar och hur arbetet skulle se ut. Detta ledde till att inte alla strategier vi tog till D1 var ett bra sätt att arbeta med uppgiften.

Strategi 1 - *Daily scrum*-möten

Det visade sig att strategin att ha *daily-scrum* möten gav väldigt bra feedback kring hur arbetet fortskred. Dessa gav snabbt en bild av vad alla i gruppen höll på med, om gruppen behövde omfördela resurserna under sprinten för att någon *user story* visat sig svårare än trott, eller att någon behövde hjälp med något.



Vårt första KPI skapades för att mäta deltagandet på våra *daily scrum* möten. Man kan se en tydlig nedgång men den är inte representativ för alla fall, framförallt under sprint 6. Under den sprinten föll det bort två möten utöver de möten som föll bort eftersom vi de dagarna redan hade två möten med varandra, då vi delade upp *sprint retrospective* och *sprint planning*. Vi borde gjort det möjligt att byta antalet möten per vecka för att få mer representativ data.

I efterhand skapar detta KPI även en grund för att utvärdera mer än bara deltagandet, som var tanken från början, om deltagandet börjar minska kanske man behöver ändra mötena. Till exempel hur de är strukturerade, byta tid eller så kan det visa på att gruppen har för mycket annat som gör att det inte hinner med dagliga möten. Det mesta av detta är dock av ett mer spekulativt slag och blir mer grund för en vidare utredning istället för ett konkret förbättringsområde.

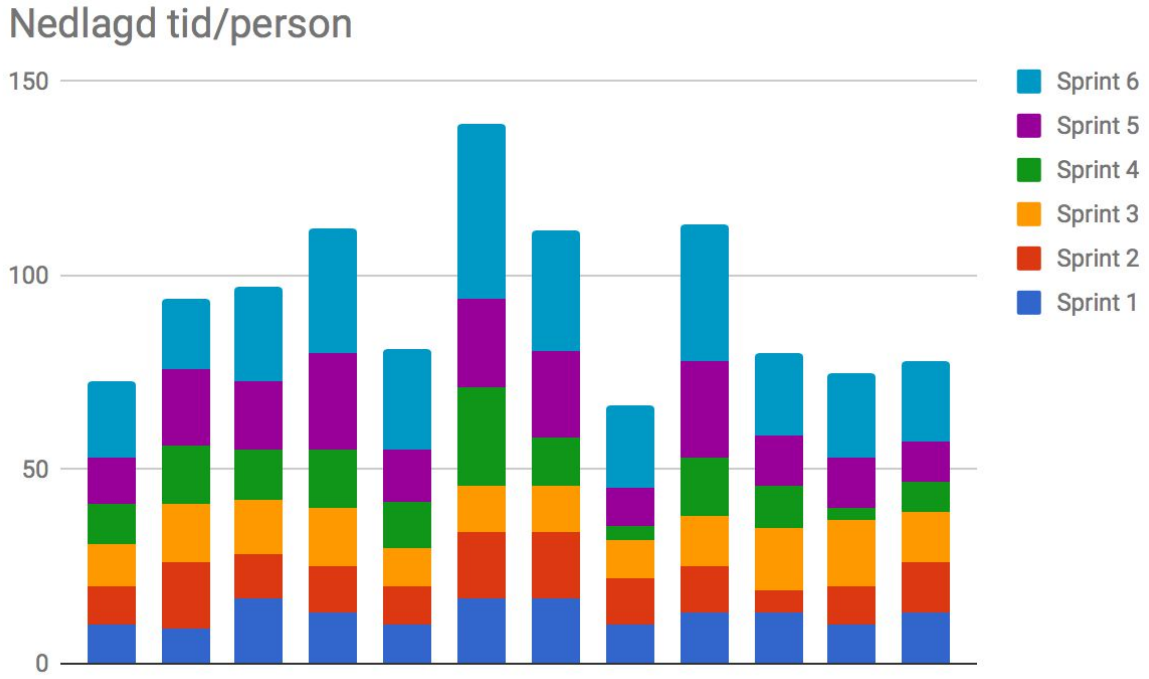
Strategi 2 - Bättre att göra rätt och bra från början

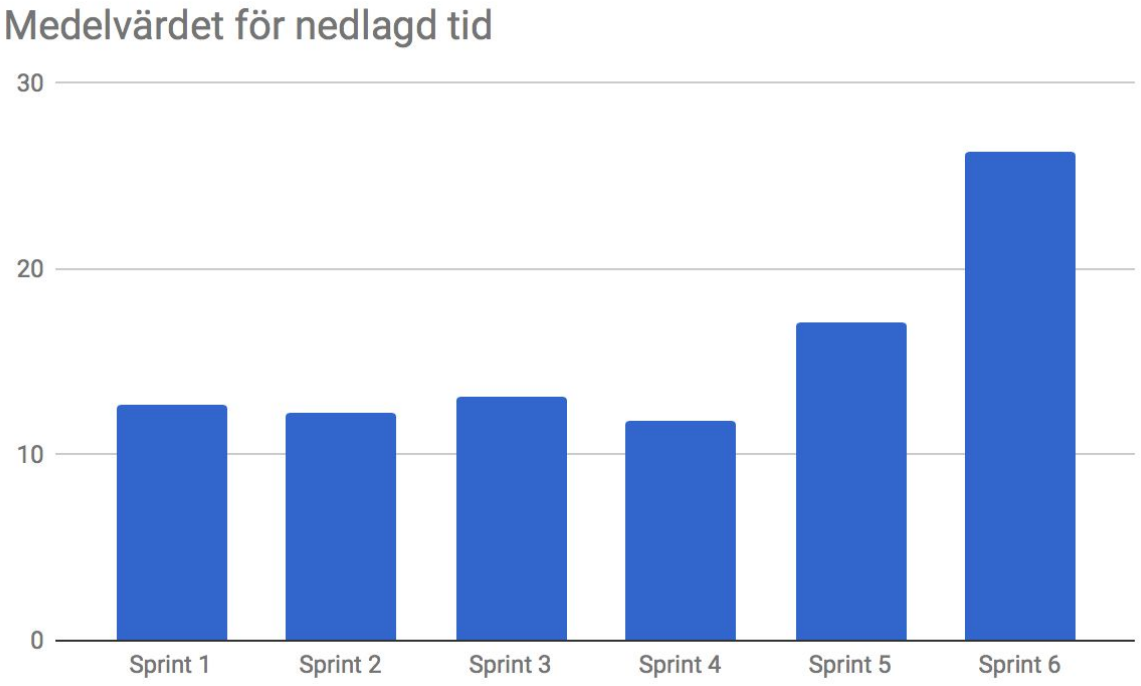
En annan strategi vi tog fram gick ut på att vi skulle till största möjliga mån försöka skriva bra strukturerad kod från början, så att vi närmare slutet av projektet inte behövde lägga mycket tid på att refaktorera. Denna strategi kom från hur vi hade arbetat på lego-övningen, där för stort fokus på kvantitet bara gjorde att uppgiften tog längre tid. Dock gjorde komplexiteten på projektet det omöjligt att arbeta på det sättet. Mycket av det som utvecklades var experimentellt och det var osäkert om det fungerade alls och om det i slutändan skulle visa sig nödvändigt eller tas bort. Det fungerade inte så bra i detta projekt men hade troligtvis varit av större värde i ett annat projekt. Efter lite mer analys av detta KPI kan det bidra till att man hamnar i vattenfallstänk. Med andra ord att man blir rädd för att skriva kod för att testa som man sedan kommer behöva ändra eller ta bort. I scrum-projekt är det snarare bra att testa sig fram på olika sätt med olika kod och olika språk eftersom vi

från början inte visste vilket sätt som skulle fungera bäst. Så ett KPI på tidsåtgång av refaktorering var ingen bra måttstock för hur effektivt arbetet faktiskt gick, baserat på hur uppgiften faktiskt var utformad. Men med lite anpassning hade det kunnat ge siffror som gett underlag till framtida förbättring.

Strategi 3 - Fördela arbetet jämnt mellan gruppmedlemmar

Under lego-övningen hände ofta att arbetsuppgifter blev snedfördelade, där vissa gruppmedlemmar inte hann med sin uppgift och andra inte kunde göra något. Denna ineffektiva tidsutnytjning hade kunnat minska om vi bara lagt lite tid på planering innan vi satte igång, istället för att alla bara rusade på en uppgift. Det visade också på hur dålig planering leder till olika fördelning av arbetet mellan gruppmedlemmar. Det var detta problem vi ville försöka lösa med vår tredje strategi. Denna gick ut på att försöka mäta hur mycket tid varje delgrupp la varje sprint för att se till att ingen la för mycket eller för lite tid. Dock var KPI:t för att mäta detta baserat på att vi lyckades uppskatta tidsåtgången på våra *user stories*, vilket i efterhand visat sig att vi ofta hade problem med. Detta leder då till frågan om ett KPI baserat på denna uppskattning kanske inte var helt genomtänkt för att mäta framgången för denna strategi.





De två diagrammen ovan visar på två olika sätt resultatet av vårt tredje KPI. Det blir lite missvisande då det inte användes ordentligt i början och således blev användningen av de tidiga siffrorna knappt befogad. Man kan se en tydlig skillnad i nedlagd till mellan gruppens medlemmar, vilket visar att vi inte fick ut vad vill ville av KPI:t. Det hade behövts mer fokus på det från början så att man redan från sprint 2 hade kunnat börja vidta åtgärder för att jämna ut arbetsfördelningen.

D2

Vår vision: *Att på sex veckor utveckla en mjukvaruprototyp som är startskottet i en revolution av transportindustrin, detta genom att demonstrera kolonnkörning med en MOPED. Mjukvaran kommer när den appliceras under lastbilstransporter öka kostnadseffektiviteten, säkerheten och minska miljöpåverkan.*

Det var svårt att formulera en bra vision, som inte blev alltför flytande. Detta var eftersom det inte var ett riktigt projekt. Det ledde till att visionen inte var något som vi jobbade mot, eller ens hade i åtanke under projektets gång. Med tanke på hur kort projektet var påverkade det inte så mycket, eftersom det inte fanns tid för projektet att driva bort från visionen. För att kunna arbeta mer mot visionen hade vi kunnat skriva en vision som siktade lägre, det hade troligtvis gjort det lättare för oss att relatera till vårt arbete och vårt projekts slutmål.

När vi skrev vår *product backlog* för första gången och uppskattade hur stora och komplexa uppgifterna var gjorde vi alldeles för låga uppskattningar, eftersom vi trodde att många delar skulle vara mycket lättare att lösa än de visade sig vara. Det visade sig också att vi delade ut för stora och diffusa uppgifter. Detta ändrade vi på till senare sprintar, då vi gjorde mycket

mindre *user stories* och gav dem en tydligare *definition of done*. Vi insåg på vägen att genomtänkta DoD gjorde uppgiften mer tydlig. Detta underlättade för gruppen att veta vad som krävdes av dem under en sprint och när de var klara.

D3

I början av projektet låg mycket av fokusen på MOPED-uppgiften och hur vi skulle lösa den, och gruppen ägnade inte mycket tid till att utveckla arbetssättet. Detta ändrades från mitten av projektet då gruppen började fokusera mer på hur vi arbetade med scrum, hur vi kunde använda oss av olika scrumverktyg för att arbeta effektivare. Bland annat använde vi Trello mer och delade upp *sprint planning* och *sprint retrospective*, som nämnts tidigare.

Under stora delar av projektet var många i gruppen hindrade av att det fanns stora tekniska bekymmer. I början var gruppen sämre på att arbeta runt dessa problem, vilket gjorde att vissa inte kände att de kunde bidra med något vissa sprintar beroende på vilken user story de hade. Detta blev gruppen bättre på senare i projektet, då vi utvecklade metoder för att arbeta kring de problemen, vilka gjorde att fler kunde arbeta med sina stories. Exempel på dessa metoder är mockupen, och att vi testade kamerakoden med simulerade bilder när vi inte hade tillgång till fungerande MOPED.

En annan grej som gruppen blev bättre på under senare delar av projektet var att göra något annat när det visade sig att ens *user story* inte gick att genomföra denna sprinten, som att utveckla och se över hur vi arbetade med *scrum*, och strukturera upp det mer. Det hjälptes också av att vi valde att ha en och samma *scrum master* som kunde lägga en del av sin tid varje sprint på att hålla koll på vad andra jobbade med, och hur *scrum* fungerade.

D4

Inför redovisningen hade vi testat vår MOPED, och den kunde helt följa efter en annan MOPED. Tyvärr innebar inte detta att MOPEDen klarade av redovisningen lika bra, vi lyckades få MOPEDen att följa den framför i tio meter som bäst. Skälet till att redovisningen inte var lika lyckad som vi hoppades ligger i tre tekniska problem vi upptäckte under denna sista dag.

Det första problemet berodde på ljusförhållandena, som ändrade vilken färg kameran tolkade vår röda cirkel som. Vi hade bara testat i vanlig salsbelysningsmiljö och ljuset under redovisningen var mycket vitare. Timmarna innan redovisningen satt vi och finstämde inom vilket spektrum kameran skulle leta efter, men anpassade vi bara till ljusflödet åt ena hållet, och inte både fram och tillbaka. Vi borde haft koll på att en annan typ av lokal har ett annat typ av ljus, men utöver det tacklade vi problemet så väl som vi kunde. Vi samlades på Lindholmen klockan nio under demonstrationsdagen, just för att tackla möjliga problem som dyker upp.

Nästa problem som dök upp var beslutet om det gemensamma avstånd till bilen framför. Vi hade ställt in MOPEDen på att göra mjuka och fina inbromsningar optimerade för ett avstånd på ca 60 cm till nästa MOPED. På grund av kommunikationsproblem med de andra grupperna bestämdes avstånd först sista dagen, och tyvärr för vår grupp bestämdes 40 cm som det avstånd vi skulle förhålla oss till. Resultatet av att 60 cm bromssträcka plötsligt blev 40 cm var logiskt; vi körde frekvent in i MOPEDen framför. Det gemensamma avståndet borde bestämts tidigare, vilket vår grupp också nämnde vid *scrum of scrums* veckan innan utan större bifall från de andra - vilket är förståeligt eftersom många antagligen hade vitala funktioner som högsta prioritet, och inte finlir.

Denna typ av kommunikationsproblem är antagligen frekvent förekommande - även i arbetslivet - men det finns antagligen ingen enkel lösning. Vi själva visste inte hur viktigt det var att ha ett gemensamt avstånd innan redovisningen, utan drev frågan för att vi gillar framförhållning. Att ha det som enda argument kommer man inte långt på. Istället hade vi behövt diskutera frågan inom vår grupp för att komma med bättre argument, vilket i sig kräver en ny nivå av framförhållning. Eller så hade vi kunnat försöka köra över de andra grupperna och sagt att ett visst avstånd gäller, men inte heller det låter särskilt lockande.

Det sista problemet upptäcktes när MOPEDerna körde. MOPEDen längst fram kunde bara svänga åt ett håll i taget. Den var tvungen att stanna helt innan den kunde svänga åt det andra hållet. Vi hade anpassat vår moped efter en mjuk körning, utan kraftiga svängar. När ledar-MOPEDen stannade hela tiden blev körningen i hela kolonnen väldigt ryckig, vilket i samband med föregående problem ledde till att vår MOPED till slut inte klarade av de skarpa svängarna, utan körde upp vid sidan av MOPEDen framför. Eftersom målet var kolonnkörning - och ryckig körning inte är kolonnkörning - hade vi inte kunnat göra något annorlunda. Vi arbetade under förutsättning att vi skulle köra i en kolonn, för det var så vi hade tolkat uppgiften av PO. Om vi hade gjort om projektet hade vi fortfarande tolkat PO på samma sätt och gjort samma prioriteringar.

Källhänvisning:

http://www.ub.umu.se/skriva/skriva-referenser/referenser-oxford
Fotnot, -> ha allt på samma sida, alltså ingen

Referens till boken:
Kniberg, Henrik. *Scrum and XP from the Trenches*. 2 uppl. C4Media, 2015.