Post Mortem

Milestone av Tobias Andersen, David Bergström, Joakim Berntsson,
Tim Kerschbaumer, Erik Nordmark & Simon Takman.

Projektets utformande

Volvo SICS Innovation Challenge blev för oss ett viktigt projekt. Å ena sidan hade vi en kurs att klara av, å andra sidan hade vi en kund att tillfredsställa. Vi bestämde oss direkt för att anta utmaningen och skapa en applikation med värde för vår kund.

Projektet är utformat efter en agil utvecklingsmodell där vi arbetat med flera iterationer. Vi har använt oss av en projektteknik som heter Scrum och tagit hjälp av Taiga - ett Scrumanpassat verktyg. Vi har även använt oss av Git för smidig versionshantering av koden.

Arbetsprocessen har bestått av noggranna förundersökningar och systemförberedelser, en agil Scrumbaserad utveckling och välutformad testning och dokumentation.

Hela projektets utformande, med antal dagar för varje del och vem som utfört vad kan ses i detalj med tillhörande gantt-diagram i filen “development-plan.pdf”.

Agil systemutveckling

Vi gjorde valet att arbeta i enlighet med agil systemutveckling i vårt projekt snarare än en vattenfallsmodell. Den största orsaken var att vi inte visste hur mycket vi skulle hinna implementera inom den satta tidsramen. I den agila systemutvecklingen ser man till att ha fungerande programvara tidigt i processen, vilket sätter mindre press på att hinna färdigt med allt man har tänkt. Detta hjälpte oss att begränsa projektet till några enstaka funktioner för att sedan planera tillägg i framtida utveckling. Att arbeta agilt hjälpte oss också att justera våra metoder under projektets gång. Sättet vi planerat första iterationen på förändras till nästa efter att vi utvärderat hur vi utfört vissa delar.

Det enda som kändes svårt med att arbeta agilt var avsaknaden av kontakten med kund. Det hade varit ännu bättre om vi efter varje iteration, och även inför, kunde fört diskussioner med kunden och på så sätt fått input i vår process. Vårt försök till att möta detta var att simulera kunden inom vår grupp.

I framtida projekt kommer vi att använda oss av agil systemutveckling så långt det går. Vi tycker om konceptet att frekvent kunna släppa färdiga produkter. Det gör också att man lätt kan hänga med i utvecklingen och ligga i framkant med den senaste tekniken.

Scrum

Scrum var det agila projektverktyg vi valde att använda oss av. Ingen av oss hade tidigare arbetat med det, men med tanke på dess fördelar tyckte vi att det verkade vettigt för denna uppgiften.

Fördelarna med att använda Scrum för Volvo SICS Innovation Challenge var flera. För det första så underlättade verktyget för att ganska snabbt komma igång med själva utvecklingen. För det andra tyckte vi om konceptet av Product Backlog med User Stories eftersom det samlade allt man tänkt med applikationen på ett ställe i samma format. Utöver detta gillade vi tanken på att ha ett litet möte varje morgon för att ha koll på vad som händer och se om någon behöver hjälp med något. Det var även skönt att en person tog ansvar för det administrativa så att inte alla behövde tänka på det samtidigt som man var djupt nedsjunken i svåra uppgifter. Fördelarna var nog egentligen fler än vad vi hann upptäcka under detta projektet, men första bekantskapen med denna agila systemutvecklingsteknik gav oss god försmak.

En av nackdelarna med Scrum i det här projektet var att vi inte helt och hållet gick in i alla roller som fanns. Det kändes konstigt att ha en person som bara sköter det administrativa, en person som agerar produktägare och ett gäng utvecklare som skriver all kod. Allihopa ville vara delaktiga och bidra till koden. Vi hamnade i något mellanting som kändes mer i enlighet med uppdraget. Man kan säga att vi tog inspiration från Scrum-rollerna och mötte dessa med en allmän delaktighet från vår sida.

Vi försökte relativt tidigt läsa på om Scrum och hur man använder det i mjukvaruprojekt. Vi implementerade arbetssättet och hade sprintplaneringsmöten, retrospektiva möten, försökte få in så många dagliga möten vi kunde och hittade ett riktigt bra verktyg för att kunna utnyttja Scrums möjligheter (se nästa avsnitt om Taiga).

Idag är det många företag som arbetar med Scrum och det känns bra att nu ha fått en inblick i vad det handlar om. I detta projekt hamnade alla roller och uppgifter i samma lilla grupp på sex personer, vilket kändes lite svårt. I större sammanhang finns det istället stora möjligheter att dela upp arbetet och Scrum blir här ett passande verktyg.

I ett framtida projekt av liknande storlek skulle vi absolut kunna tänka oss att använda Scrum. Vi hade förmodligen modifierat arbetssättet så att det passar en mindre grupp, men vi tycker att det är ett bra sätt att driva mjukvaruprojekt på ett agilt sätt.

Taiga - *tree.taiga.io/project/milestone*

När vi hade bestämt oss för att använda Scrum letade vi direkt efter användbara verktyg för att enkelt kunna strukturera upp projektet. Vi provade flera stycken som vi tyckte verkade bra men när vi hittade Taiga så var vi sålda. Även om flera veckor redan passerat så försökte vi i största möjliga mån föra över allt arbete till Taiga.

Taiga medförde flera fördelar. För det första så var det helt och hållet Scrum-anpassat. Man kunde skapa User Stories, sätta Story Points, skapa Sprints, lägga till uppgifter i varje sprint för olika User Stories och mycket mer. Verktyget blev ett sätt att hantera allt man behövde för projektet på ett och samma ställe. Detta var särskilt positivt när vi arbetade med projektet på olika håll eftersom vi lätt kunde se vem som arbetade med vad och när. Det fanns även möjlighet att ge olika roller till projektmedarbetarna (UX, Project Leader, Frontend, Backend etc). Ifrån de User Stories man skapat och de Story Points man satt ut kunde personerna i de olika rollerna se hur mycket jobb de hade framför sig för varje sprint. Ett fint tillägg var att Taiga försåg oss med ett Burndown Chart som visade alla Sprints och de Story Points vi satt och sammanfattade detta i ett fint diagram.

Den största nackdelen med Taiga för detta uppdraget var att det tog ett tag att sätta sig in i alla funktioner som fanns att tillgå. Tiden var vårt största hinder till att fullt ut använda verktyget.

Vi skulle absolut använda Taiga i framtida projekt, särskilt de av större skala med flera inblandade. Verktyget verkar vara anpassat för en större grupp där det är viktigt att ha en bra kommunikationskanal som samlar hela projekthanteringen på samma ställe. Men för vilken projektgrupp som helst som väljer Scrum som tillvägagångssätt så är Taiga ett klockrent verktyg.

Git & Gitflow

Att valet om versionshanteringsprogram föll på Git kom naturligt av flera anledningar. Dels eftersom att vår grupp bestod av sex medlemmar och vi hade för avsikt att implementera flera nya funktioner parallellt inom gruppen. Men också på grund av hur enkelt och resurssnålt det är att skapa nya brancher i Git kontra andra liknande system.

Även om Git i teorin kan lösa de flesta problem som annars kan uppstå när många utvecklare ansvarar över samma kod, krävs ett bra arbetsflöde. Annars är det lätt hänt att man spenderar mer tid på att lösa så kallade "*merge conflicts*", och fundera över vad som gått fel, än på att faktiskt utveckla nya funktioner. Vi bestämde oss tidigt för att arbeta enligt arbetsflödet Gitflow.

Gitflow bygger till stor del på så kallade *"feature branches"* – det vill säga brancher som motsvarar enskilda och isolerade funktioner. Denna inkapsling gör det enkelt att implementera nya funktioner utan att bryta fungerande kod i huvudbranchen. Gitflow och feature branches gick också hand i hand med hur vi arbetade med User Stories, Tasks och liknande i Scrum.

Att använda Git tillsammans med Gitflow har fungerat över all förväntan när det gäller kodhantering och har samtidigt tvingat oss att planera på ett bra sätt inför den faktiska implementeringen.

Att hitta några direkta nackdelar med Git och hur vi arbetat med det är svårt och vi kommer definitivt att göra på samma sätt i framtida projekt. Det bör dock påpekas att det kräver viss förståelse om Git och framförallt hur exempelvis remotes, brancher och repositories (lokala vs centrala) fungerar.

Utförande

I förundersökningen gjorde vi studier kring lastbilschaufförers vanor under färd genom att intervjua ett par förare. Dessutom tog vi kontakt med företag som själva gjort liknande undersökningar och fick därifrån värdefull information. Vi studerade även diverse tekniker vi visste skulle behövas i projektet. Parallellt med förundersökningen arbetade vi med att ta fram en produktvision samt en primär och sekundär persona.

När vi avslutat förundersökningen och hade en konkret idé om produkten skapade vi modeller för hur mjukvaran skulle utformas. Vi gjorde detta i form av ett abstrakt UML-diagram (analysmodell) samt ett mer detaljerat diagram (designmodell). Även andra diagram, såsom flödesdiagram, skapades för att lättare få en bild över hur utvecklingen skulle struktureras och delegeras.

Med en klar bild över projektets grund fyllde vi vår Backlog med flera User Stories. Vi bestämde hur vår första sprint skulle se ut och påbörjade utvecklingen. När första sprinten var testad och klar såg vi över hur mycket tid vi hade och bestämde vilka funktioner som var rimliga att ta med i sprint nummer två.

Efter att sprint nummer två avslutas utfördes ytterligare testning. Vi utförde bland annat en kognitiv genomgång, testning av systemets helhet och analyser för vår kodstruktur. Samtidigt som detta skapade vi ytterligare dokumentation av projektet i form av javadoc, presentation, utvecklardokumentation och instruktionsbok om hur applikationen används.

Förstudier

För att kunna skapa en säker applikation som lastbilschaufförer vill använda krävdes det noggranna förundersökningar. Vi började tidigt med att ta kontakt med ett par lastbilschaufförer för att få en uppfattning om vad de behöver när de utför sina uppdrag. Detta gav oss några idéer men vi ville bredda vårt perspektiv ytterligare. Vi ville höra vad Volvo egentligen är ute efter som kund och var ivriga efter kreativa idéer.

Vi tog därför kontakt med reklambyrån Forsman & Bodenfors. F&B har skapat flera reklamfilmer för Volvo Lastvagnar och har en inblick i företaget och dess målgrupp. De gav oss flera tips om hur man skapar sig en bild av sin målgrupp och berättade om egna fältstudier de gjort i samband med reklamfilmerna. När vi tog upp en av våra första idéer som hade att göra med en navigationsapplikation tipsade de oss om en hemsida – World Trucker.

World Trucker visade sig att vara ett Community för lastbilschaufförer. Dessa förare hade rankat tusentals rastplatser, restauranger, bensinstationer och mycket mer över hela världen. Detta var exakt den grund vi behövde för att vår idé skulle fungera som bäst. Vi upptäckte även att Communityt drevs av Volvo Lastvagnar vilket vi hoppades bara skulle vara till vår fördel. Vi tog kontakt med World Trucker och företaget bakom – Dear Friends.

Under vårt första möte med Dear Friends presenterade vi vår idé och ställde frågan om vi kunde samarbeta med dem på något sätt. De var otroligt hjälpsamma och tillmötesgående. Vi fick tillåtelse att använda deras API för att hämta information från deras databas med alla lastbilsstopp. API:et var relativt enkelt att arbeta med och där fanns allt från restauranger till servicepunkter speciellt utvalda av lastbilschaufförer över hela världen. Detta var ett gynnsamt samarbete och vi var ganska säkra på att vi hade något unikt i vårt projekt.

Dear Friends blev vårt bollplank under projektets gång. De lyssnade på idéer och gav feedback för förbättring. Fortsatt försåg Dear Friends oss med relevant information från deras egna förundersökningar inför skapandet av World Trucker. De fungerade även som stöd inför vår slutliga presentation.

Git

Vi hade alla viss erfarenhet av Git från tidigare projekt, men hade långt ifrån utnyttjat dess fulla kapacitet. Vi hade till exempel haft all kod i huvudbranchen och inte arbetat efter något speciellt arbetsflöde. Därför bestämde vi oss den här gången för att ta vårt sätt att arbeta med Git till nästa nivå. Vi kände till att resurssnåla brancher var en av de stora fördelarna med Git och såg därför till att unyttja detta.

För att minimera risken för merge conflicts och onödigt hoppande mellan olika brancher inledde vi utvecklingen av varje ny funktion med att noggrant planera hur denna skulle implementeras och vem som skulle göra det. Först därefter skapade vi en dedikerad branch för funktionen i fråga. Typiskt var att denna branch levde lokalt hos de gruppmedlemmar, oftast 1-3 stycken, som var med och utvecklade funktionen. Först när funktionen var testad och klar fördes den in i huvudbranchen genom en så kallad *“merge”* och togs därefter bort.

En fullständig beskrivning av vårt arbetsflöde finns i “developer-documentation.pdf”.

Scrum

Utvecklingsstadiet i vårt projekt bestod av fyra veckor som var uppdelade i tvåveckors sprints, en för applikationens stillastående läge och en för dess rörliga. I dessa ingick även testning och dokumentation. Den första sprinten blev aningen överbelastad då mycket av det som behövdes i rörligt läge även behövdes i stillastående. Men detta åtgärdades snabbt genom att dela upp arbetet ytterligare och flytta en del uppgifter till nästa sprint.

Arbetet för att ta fram en behaglig design utfördes i iterationer där de första skisserna gjordes på whiteboard. De utvärderades och gjordes om och följdes upp med detaljerade skisser i Photoshop. Till slut hade vi en implementerad version där vi kunde utföra kognitiva genomgångar och andra tester.

Implementation

Vi har genomgående utgått från User Stories i Scrum som övergått till funktioner utvecklade i dedikerade "feature branches". En funktion utvecklades till stor del i par, men krävde ibland flera gruppmedlemmar. Detta fungerade överlag bra förutsatt att kommunikationen var tydlig.

För att få en tydlig struktur med utsatta roller skilde vi tidigt frontend från backend. Detta skapade också en bra balans mellan områden man kunde fördjupa sig i och områden man kan kunde överlåta till andra.

Eftersom att vi samtidigt var beroende av varandras arbete såg vi till att kontinuerligt föra samman funktioner områdena emellan. Detta gjordes på planerade "merge-möten" där fokus låg vid att föra in feature branches i huvudbranchen. När alla funktioner blivit införda provkörde vi gemensamt applikationen och gick igenom eventuella frågor och oklarheter kring den nya funktionaliteten. Detta gjorde att alla, trots delegering av olika uppgifter och områden, hade bra uppfattning över projektets utveckling.

Gruppdynamik

Som grupp har vi fungerat mycket bra. Att vi kände varandra redan innan projektets start var en stor fördel och innebar att vi kunde hoppa över de första gruppdynamiska stadierna och snabbt hitta våra roller. Särskilt bra fungerade uppdelningen av arbetet. Med allas olika intressen var det naturligt vad var och en skulle ta sig an för uppgifter.

En vanlig vecka inleddes oftast med ett möte med diskussioner kring vad som skulle göras i veckans sprint. Arbetet delades upp och var det något beslut man var osäker på diskuterades det här medan mindre beslut överläts till var och en. Beslutsfattandet i gruppen gick smidigt och vid meningsskiljaktigheter löstes dessa genom demokrati. En av fördelarna av att vi kände varandra var att vi inte hade några problem med att ta en diskussion och reda ut oklarheter direkt när de uppstod. Vid veckans slut sammanfattade vi arbetet och utvärderade det som gjorts.

En viktig pusselbit i ett projekt av den här typen är naturligtvis individuellt ansvarstagande. I en grupp på sex personer blir det svårt att få alla samlade vid varje enskilt tillfälle. Det leder till att var och en måste ta eget ansvar för att få sprinten klar till slutet av veckan. För oss var det tydligt att när man väl åtagit sig en uppgift såg man till att få den gjord. Eftersom detta

fungerade från start byggdes ett förtroende och vi vågade lita på att var och en drog sitt strå till stacken.

Vi upptäckte snabbt hur viktigt det var med tydlig kommunikation. När vi sågs i slutet av första veckan hade till exempel två personer utfört i stort sett samma arbete. Vi lärde oss av detta och blev tydligare i våra arbetsuppdelningar. Samtidigt insåg vi att arbetet blev enklare då vi satt tillsammans, när tillfälle gavs.

Något annat vi lärde oss under projektets gång var att tydligt sätta upp tider då vi förväntades jobba tillsammans. Till en början satte vi endast ut en tid då vi skulle starta. Sex olika liv innebär sex olika tider att förhålla sig till. Detta ledde ibland till att irritation uppstod då någon i gruppen var tvungen att gå hem tidigare. Lärdomen blev att det även var viktigt att bestämma sluttid.

Reflektioner kring speciella beslut

Designat för Nexus 7

När utvecklingen sattes igång behövde vi en enhet att testa på. I emulatorn fanns flera val men vi ville utveckla mot en speciell enhet som vi även kunde testa i verkligheten. Dessutom hade vi inte tiden att anpassa gränssnittet dynamiskt till flera olika skärmstorlekar och upplösningar.

Vi valde Nexus 7 som utvecklingsenhet på grund av dess omodifierade Androidversion som även AGA har byggts utifrån. Nexus 7 med dess 7-tumsskärm ansåg vi vara en realistisk storlek på den skärm som i framtiden kan monteras i en lastbil. Dessutom har vi, ur ett säkerhetsperspektiv, valt att inte fokusera på enheter med små skärmar.

Gränssnittskomponenter och layout har blivit speciellt framtagna för Nexus 7 och det är endast på denna enhet vi kan garantera att vår applikation uppfyller de krav som ställts.

Inga instrumenttester

Vår applikation var aldrig tänkt att fungera eller leva som en traditionell Androidapplikation. Vi ville därför inte prioritera att skriva tester för att få den stabil på en telefon för dagligt användande. Vi bestämde oss därför för att hoppa över instrumenttester och istället satsa på att enhetstesta så mycket som möjligt. Att vi dessutom endast hade runt fyra veckor att implementera funktioner gjorde beslutet enklare.

Allt gick inte att enhetstesta

Vårt mål har varit att skriva enhetstester till alla nya funktioner innan de förts in i huvudbranchen. Trots detta har vissa klasser visat sig vara svåra att enhetstesta. Det har till exempel varit svårt att testa de modeller och hjälpklasser som är beroende av data via

HTTP-requests. Även klassen som ansvarar för kommunikation över HTTP var problematisk då den beror av många yttre faktorer, som till exempel servern den kommunicerar med.

För att testa de delar som använt AGAs JAR-filer krävs värden från en simulator. Dessa värden finns inte att tillgå förrän efter testerna terminerat och därför har enhetstestning varit svårt att utföra.

Slutord

Volvo SICS Innovation Challenge blev en spännande utmaning för oss. Det kändes utmanande och samtidigt roligt att få vara först ut med AGA och nya tekniker för lastbilar. Med tanke på att varken riktlinjer eller begränsningar gavs, bortsett från säkerhetsaspekten, var det upp till oss att skapa något av värde. Vi blev själva ansvariga för hela processen – allt från förundersökningar till färdig produkt. Vi gav oss inspirerade ut på okänd mark.

I och med att vi ville skapa något av värde valde vi att ta förundersökningen på största allvar. Vi ville skapa något nytt men samtidigt undvika att återuppfinna hjulet. Därför blev vi positivt överraskade när vi kontaktade stora företag och de tog sig tid att hjälpa oss. Tack vare detta förbättrades våra förutsättningar vilket motiverade oss att ta fram en bra produkt.

Aldrig tidigare hade vi gjort ett projekt med så många medarbetare och vi märkte att krav på tydlighet och struktur blev viktigare än någonsin. Tack vare bra verktyg och god gruppdynamik kunde vi använda gruppstorleken till vår fördel.

Vi har fått insyn i hur en helt ny produkt utvecklas och hur mycket man kan få gjort med bra arbetsmetodik. Med tanke på tidsramen och de fria tyglarna är vi mycket nöjda med vår slutprodukt Milestone och lämnar projektet med positiva intryck.