Grupp Jingis

Post Mortem Rapport DAT255

SiKKr

2014-10-31

Eric Bjuhr
Armand Ghaffarpour
Oskar Jönefors
Emilia Nilsson
Jesper Olsson
Joel Tegman

CHALMERS
Chalmers University of Technology
Informationsteknik

Arbetssätt

Som så många gånger förr, startade vi det här projektet med ett väldigt formellt och strukturerat arbetssätt, för att sedan övergå i ett slarvigare och mer diffust arbetssätt. Vi försökte applicera scrum och agile de första två veckorna av utvecklingsprocessen. Den första iterationen sattes upp med hjälp av user stories och issues på Waffle. Vi satte upp realistiska mål för iterationen under ett gruppmöte och försökte sedan följa upp processen med dagliga scrum-möten. Dagliga är väl en sanning med modifikation, men ett par gånger i veckan följdes processen upp. Vi delade även ut ansvarsområden, som projektansvarig, designansvarig och backendansvarig.

Dessa uppföljningsmöten var oftast i form av stand-up-meetings, där var och en fick berätta vad man gjort, vad man håller på med, vad man planerar att göra och eventuella hinder. Om man var klar med sina uppgifter fick man antingen ta på sig nya uppgifter om några väldefinierade sådana fanns. Om inte, kunde någon annan i gruppen som hade lite mer kvar att göra be om personens hjälp. Mötena försöktes hållas så korta som möjligt.

Det mesta av utvecklingen skedde tillsammans, där allt som oftast hela gruppen var närvarande. Vid dessa tillfällen utövades en del parprogrammering. Även om det kanske inte var uttalat att vi skulle använda oss av parprogrammering, så föll det ganska naturligt när vi satt tillsammans i gruppen. Om man behövde hjälp, satt man ofta två och två vid en dator och försökte lösa problemet tillsammans.

Vi satte inget konkret slutdatum för iterationen, vilket skulle visa sig vara början till slutet av vår korta scrum-session. Följden av uteblivet slutdatum ledde till att iterationen aldrig riktigt följdes upp eller utvärderades när den var klar, vilket i sin tur ledde till att ingen ny iteration sattes upp. Detta betydde att våra requirements aldrig blev uppdaterade efter den sista iterationen. Arbetssättet blev istället något som mer liknade vattenfallsmetoden.

Vi slutade nästan helt att ha återkommande stand-up-meetings, med undantag för enstaka tillfällen. Backlogen hölls förvånansvärt nog ganska bra uppdaterad, trots bristande strukturerad kommunikation inom gruppen. Detta gjorde att vår omedvetna övergång från scrum till vattenfall inte märktes av lika tydligt som man kan tro.

Sammanfattningsvis kan man säga att arbetssättet till en början utgjordes av scrum, med stand-up-meetings och parprogrammering, för att sedan mer övergå i vattenfalls-metoden med inslag av parprogrammering.

I kursen talades det om arbetssättet “Semat”. Detta var dock någonting vi dessvärre inte testade på. Dels för att vi inte helt greppade tillvägagångssättet och dels för att vi till en början enbart ville komma igång med kodandet och sedan aldrig kände något naturligt behov av det.

I ett framtida projekt hade det varit intressant att testa på “Semat”-metoden och utforska dess för- och nackdelar. Vår gissning är att det mycket väl kan fungera på lite större projekt som rör sig över längre tid.

Stand-up-meetings

Fördelen som vi upptäckte med att använda detta arbetssättet var att man ständigt var informerad om vad alla gjorde och man hade nästan dagligen chansen att ta upp saker för diskussion, be om hjälp och så vidare. I och med den ständiga uppdateringen man fick, kunde man bättre tidsplanera sina individuella uppgifter utefter hur långt de andra i gruppen hade kommit. På så sätt eliminerade man problemet med att behöva vänta på att vissa komponenter skall vara klara innan man kunde börja jobba och dylikt.

I efterhand är det svårt att se några konkreta nackdelar med det här arbetssättet, mycket eftersom vi bara utövade det i ungefär två veckor. En sak som man dock började märka mot slutet av vecka två var att mötenas kvalité sjönk en aning. Koncentrationen var inte på topp, informationen och innehållet var inte lika bra och mötena gav inte lika mycket som de hade gjort i början. En möjlig orsak till detta är att vi tyckte att dessa möten blev ganska tråkiga och enformiga, och på så sätt tappade fokus. Så om man ska hitta någon nackdel är det väl i så fall just det; att mötena kan bli tråkiga i längden, och tappa sin effekt och verkan.

Eftersom mötets längd ofta var väldigt kort, sällan över tio minuter, anser vi det här arbetssättet vara väldigt tidseffektivt med tanke på hur mycket vi fick ut av det. Tio minuter per dag som genererar god struktur och goda förutsättningar för arbete får anses som väldigt bra.

Detta arbetssätt tror vi lämpar sig bäst för ett projekt med ett ganska kort tidsspann. Detta eftersom man inte hinner tröttna på mötena på samma sätt och de kan hålla hög kvalitet genom hela projektet. På grund av detta skulle vi nog inte använda oss utav detta arbetssätt för ett projekt som skall pågå under en lång tid. Om man trots allt vill använda sig av detta arbetssätt för ett projekt med ett långt tidsspann, kan det nog krävas att man på

något sätt förändrar mötenas upplägg med jämna mellanrum, så att de inte ser likadana ut hela tiden. Detta måste dock göras utan att förändra mötets innehåll, som ju är det centrala, vilket kan vara ganska svårt.

Om man skall jämföra den perioden som vi använde oss av denna metod med den period som vi inte gjorde det, kan man hitta en del skillnader. Den allra största var informationen som man hade om andras arbete och strukturen på hela projektet. Med den här metoden fick vi en väldigt god struktur på vårt arbete och utan den fick vi det motsatta, ett ganska stökigt och oorganiserat projekt.
Så den slutsatsen man kan dra är att vårt projekt mådde bättre när vi använde oss av den här metoden.

Parprogrammering

Fördelarna som vi upptäckte med detta arbetssättet var framförallt att problem som uppstod under utvecklingen gick mycket snabbare att lösa om man satt i par. Sitter man själv är risken oftast stor att man snöar in sig på ett problem och tappar rationellt tänkande. Den risken blir mycket mindre när man sitter i par, eftersom man ofta tänker på problemet på olika sätt och lättare kan komma fram till en lösning. Pardynamiken gör att båda utvecklarna konstant antingen lär sig ett koncept, eller får lära ut ett koncept, eftersom någon hela tiden måste driva arbetet framåt. Parprogrammering reducerar även problemet med att man som utvecklare beskyddar sin kod, och blir defensiv när någon vill ändra i den. Det blir inte lika personligt när man är två som har utvecklat, vilket är en stor fördel.

Om man skall hitta någon nackdel med den här metoden så är det väl att produktiviteten kan bli lidande, eftersom båda personerna inte kan programmera samtidigt. Men man måste samtidigt väga den produktivitet man förlorar mot den produktivitet man hade kunnat förlora om båda suttit var för sig, insnöade på varsina problem. En annan möjlig nackdel är att man lätt kan bli irriterade på varandra, hamna i dispyter som i vanliga fall inte hade uppstått, och på så sätt skapa dålig stämning i gruppen.
I vårt fall får vi anse parprogrammeringen vara ganska tidseffektiv. Oftast blev resultaten goda och oftast blev problemen lösta snabbare än vad de i vanliga fall hade blivit.

Denna metod skulle vi kunna tänka oss att använda i projekt där gruppen har en hög mångfald. Då kan man på bästa sätt utnyttja varandras olikheter och applicera det på de problemen man ställs inför. Om man istället har en grupp där medlemmarna är ganska lika och har ungefär samma kompetensnivå, kanske det inte är läge att använda sig av den här metoden då man inte med säkerhet kommer att få maximal utdelning av den.

Vattenfalls-metoden

Eftersom vi i vårt projekt inte fullföljde alla de metoder vi började med (scrum, agile...) resulterade det i att vi till slut använde oss av något som mest liknade vattenfalls-metoden.

Fördelarna med denna metod är väl ganska få. I vårt projekt är det ganska svårt att säga eftersom vi inte planerade att använda oss av denna metod. Sen är det väl inte helt sanningsenligt att säga att vi faktiskt använde oss av denna metod, men om man ska säga att vi jobbade utefter en modell de sista veckorna så är väl vattenfalls-modellen det närmsta man kan komma.

De sista tre veckorna i projektet handlade i princip bara om att färdigställa en produkt utifrån de de krav vi hade ställt upp i ett tidigt stadium. Allt fokus låg på utvecklingen och vi tog oss aldrig riktigt tid att utvärdera det vi gjorde, eller följa upp det på något sätt. Om man skall hitta något positivt under dessa tre veckor så är det väl just det faktum att vi fick väldigt mycket gjort. Det fanns hela tiden arbetsuppgifter åt alla, och alla jobbade hårt. Vi behövde inte lägga ner någon tid på möten eller liknande, utan alla bara jobbade på.

Det negativa med detta var dock det faktum att vi egentligen inte hade någon möjlighet att ändra på de krav vi hade från början. Vi var tvungna att utgå från dessa, utan att ha chansen att ändra något. Nu blev ju resultatet inte så illa som man hade kunnat tro, men det här arbetssättet är inget som vi ser någon fördel att jobba efter i framtiden. Behovet att kunna ändra på krav och dylikt samt att jobba flexibelt är alltför stort för att vattenfalls-metoden skall vara effektiv.

Detta fungerade bra

Det som fungerade bra i projektet var uppdelningen av arbetet. Det gick att placera varje projektmedlem till en specifik uppgift utan att man störde varandras arbete. Man kunde jobba på var sin vy, utility-klass, modellklass o.s.v. och det hände därför ofta att man jobbade i en egen branch med det man ville implementera.
Det fanns också alltid något man kunde göra, tack vare att backlogen höll sig hyfsat uppdaterad. Med aktiva projektmedlemmar var det också enkelt att få hjälp och stöd när man fastnade i något problem. Att mötas och prata om vad man arbetade med just då och

ha möjlighet att ta upp problem eller be om hjälp kändes som ett gott stöd. Mycket av det som gjorde att projektet gick bra låg i att det var strukturerat från början. Det fanns en bra grund och trots det mindre organiserade arbetet mot slutet var det inget som egentligen förvärrades.

Vi hade fått en hel del dokumentation att läsa igenom (SICS, NTHSA, Volvo rekommendationer, mm), och fick vi tipset att låta en person läsa igenom en dokumentation istället för att låta alla läsa allt. Senare tog den som läst igenom dokumentet på sig ansvaret att redovisa informationen som han eller hon tyckte var viktigt som alla i gruppen kunde ta del av. På så vis har man sparat tid och ändå fått med allt som behövs.

Under utvecklingsfasen då ny funktionalitet implementerades genomfördes regelbundna test av appen för att undersöka att den hela tiden uppförde sig som vi ville. Detta genom att skicka den byggda appen till någon av gruppmedlemmarnas telefon och grundligt gå igenom just den berörda delen.
Att felsöka på detta sätt var initialt svårt då meddelandet “SiKKr har stoppats” var det enda som skrevs ut då programmet krashade.
En logg-klass skrevs för att kunna skriva ut intressant data till telefonen som sedan kunde undersökas för att ta reda på varför applikationen slutat fungera. Detta fungerade bra i många fall men ibland kunde det vara svårt att veta exakt vad som behövde loggas för att hitta felet.

Detta fungerade mindre bra

Något som hade kunnat fungera bättre var diskussionerna kring designbeslut. Det var lätt att fastna i små detaljer, hamna utanför det relevanta och inte kunna ta hand om det absolut viktigaste först. Detta var inget stort problem, men hade kunnat undvikas och på så sätt kunnat gå vidare snabbare och få mer tid över. Givetvis var det inga oviktiga diskussioner, men med tanke på hur kort tid man har på sig skulle man behövt dela upp och prioritera arbetet för att effektivisera diskussionerna och avbryta när de började tappa sin relevans.

Just för att vi tappade bort våra stand-up möten blev det lätt att man blev sittande själv med sina problem istället för att ta tag i gruppen och be om hjälp till exempel.
Hade vi fortsatt med mötena hade alla problem lyfts var dag och det hade blivit naturligt att be eller erbjuda hjälp. Detta var inte något stort problem men det var något som fungerade mindre bra längre fram i projektet.

Övriga reflektioner

Trots avsaknad av stand-up meetings följdes arbetet upp ett fåtal gånger. Där reflekterade vi över funktioner vi hade samt funktioner vi saknade, där de funktioner som saknades blev prioriterade i en viss ordning. Detta gjorde att vi inte jobbade med fel saker och kunde bli klara med det viktigaste först och på så sätt far inte projektet illa lika mycket av det oorganiserade upplägget.

Ingen av oss var bekanta med Android-programmering sedan tidigare. Tid gick åt till att lära sig grunder och hur enkla kommandom utfördes. Detta gjorde att det tog tid till en början innan man kunde förverkliga de idéer man hade. Detta medförde även att vi inte riktigt hann färdigställa allt som vi hade velat.

Hade mer tid funnits hade vi implementerat ytterligare en del saker. Något vi hade ändrat på är designen på vissa av vyerna. Vissa känns lite för tråkiga och behöver finputsas. Vår tanke från början var att kunna skapa en egen skärm för pågående samtal. Detta var dock något som vi uppfattade som komplicerat och sköt upp det. Det kom ej med i vår slutversion. I slutprodukten använder vi oss inte av AGAs funktionalitet någonting. Framtida förbättringar skulle vara att anpassa applikationen utefter hurvida lastbilen körs eller står stilla. När lastbilen står stilla skall det exempelvis finnas möjlighet att läsa och skriva textmeddelanden, slå telefonnummer och göra diverse inställningar. Vi hade väl även tankar om att knappen för röststyrning och även inspelning av röstmeddelanden skall befinna sig någonstans på ratten, för att minska tiden som man behöver titta bort från vägen.

Sammanfattningsvis får vi ändå säga att vi är väldigt nöjda med resultatet av applikationen. Tittar man tillbaks på de krav och den vision som vi skrev i början av projektet, ser man att den funktionalitet vi betonade då finns med i slutprodukten.

Hur vi jobbade som grupp

Som grupp fanns det bra kommunikation och samarbete. Större delen av tiden satt vi tillsammans och jobbade i samma rum. En stor fördel med att alltid ha sin grupp nära tillhands var att man kunde undvika konflikter i koden (merge conflicts), eller om de skedde ändå, kunna lösa dem smidigare. Förutom det kunde det alltid komma upp frågor, antingen rent programmeringstekniska, hjälp med designval eller bara att få lite input eller

feedback. Följden av detta blev en bättre studieatmosfär som ökade arbetsmoralen, något som kan vara svårt att uppnå om man sitter hemma.

I övrigt hade varje utvecklare chans att få jobba med den nivån på programmeringen som kändes bra och fastnade man i svårare programmering hade man antingen gruppen att fråga eller den man parprogrammerade med.

Tiden som varje projektmedlem la på projektet varierar, men totalt jobbade vi alla i stort sett varje dag, flera timmar om dagen, där nästan alla var med varje gång och bidrog aktivt med att ta projektet framåt. Vissa har lagt fler timmar än andra, men alla har jobbat hårt med att få projektet att fungera. Över alla veckor som projektet gått skulle ett snitt (per person) ligga på cirka sex timmar per dygn.

Vad vi skulle göra annorlunda nästa gång

Nästa gång vi skulle genomföra ett liknande projekt skulle vi försöka ha mer kontinuerlig genomgång av vart vi befinner oss just nu. Till exempel dagliga scrum-möten eller uppföljningsmöten efter varje iteration och utvärdera hur det gick och vad nästa steg är.

Det hade varit intressant att testa på Semat och undersöka om det hade gett resultat.

Trots att vi hade utsett en designansvarig så gick alla designbesluten genom hela gruppen. Att låta sex personer försöka få sin vilja igenom var inte effektivt och ledde sällan till några konkreta beslut. Att hitta ett bättre sätt hade varit oss till fördel. Skulle man helt släppa yttersta beslutet till en enskild person eller kanske ett par bestämma? Skulle man initialt i gruppen försöka samsas om någon enkel grund och sedan låta detaljerna bestämmas av den som faktiskt ska implementera det?
Detta är saker att reda ut tidigt i ett nästkommande projekt.