Tekniker som användes vid utvecklingen av ChalmersOnTheGo

Under arbetets gång utvecklade gruppen en kombination av Extreme Programming, XP, och Scrum-metodologi; en inte helt ovanlig utveckling i dylika projekt. Skeendet underbyggdes därtill av att kunskapen om dessa metodologier ökade löpande under kursens gång. Just det faktum att gruppen lärde sig mer om verktygen GitHub och Pivotal tracker under projektets gång gjorde att dessa verktyg inte utnyttjades ultimat i början – startsträckan var ett par veckor åtminstone. Detta löste sig emellertid senare och verktygen utnyttjades flitigt därefter. Det är svårt att säga hur detta skulle kunna göras bättre då det ju hölls en tutorial i början av kursen – kanske att den skulle ligga som allra första moment, kanske flera gånger, vara mer inriktad på ”Klicka på knapp X, sedan knapp Y…” eller utformas som genomgång per grupp och inte för hela klassen. Resurskrävande, men kanske givande.

Så gott som alla element i Scrum-metodologin utnyttjades från projektets första dag. Gruppen hade Scrum-möten måndag, onsdag och fredag varje vecka. Måndagsmötet skedde i samband med måndagsföreläsningen, onsdagsmötet skedde via Facebook-grupp och fredagsmötet skedde i efter kurshandledarmöte. Måndagsmötet var också det möte då den veckolånga sprinten planerades, liksom den tidigare sprinten avslutades, varför detta möte var extra långt. ”Färdig kod” definierades under dessa sprintavslutningsmöten som ”fungerande kod”. Denna definition hade kunnat vara tydligare, särskilt i en potentiell framtida grupp där medlemmarna har olika ambitionsnivå. Även andra regler hade kunnat definieras, såsom backlog-etikett med mera, men befanns inte nödvändiga under detta projekt. Det hade emellertid kunnat bidar till en mer städad backlog. Under Scrum-mötena fick varje person berätta vad han eller hon gjort sedan det sista mötet, beskriva eventuella problem, samt berätta om sin kommande planering. Nya insättningar av User stories eller uppgifter i Backlogen i verktyget Pivotal Tracker gjordes av respektive berörd gruppmedlem eller av någon annan under mötet eller efteråt.

Mötesstrukturen för Scrum (1. Vad har jag gjort, 2. Vad ska jag göra, 3. Vad hindrar mig) hölls vid varje Scrum-möte. Emellertid fungerade mötena också som ett tillfälle för gruppmedlemmarna att hjälpa varandra med olika problem eller reda ut riktningar för utvecklingsarbetet. Sålunda kunde diskussionen sväva ifrån Scrummandet, men när problemen väl lösts återfördes fokus till Scrum. Detta fungerade mycket bra i just denna grupp och detta projekt, där alla medlemmar var drivande för att reda ut problem och komma vidare i utvecklingen. Som en reflektion i efterhand förvånas jag över hur lite mötestid som faktiskt lades på annat än projektspecifik diskussion - i en mindre initiativtagande, men ändå lyssnande, gruppkultur hade vårt ganska fria mötesupplägg inte ha fungerat. Det hade troligtvis inte heller fungerat i om gruppen haft två eller fler medlemmar extra. Övrig kommunikation och diskussion skedde genom en Facebook-grupp som uppdaterades med nya inlägg så gott som varje dag. Detta kommunikationssätt möjliggjorde snabb feedback, liksom olika arbetstider inom gruppen vilket passade gruppmedlemmarna mycket bra.

Gruppens samarbete och diskussionsklimat fungerade bra från början, varför varken Product owner eller Scrum master utsågs; alla gruppmedlemmar iklädde sig dessa roller. Detta kan jämföras med den mer informella rolluppdelningen som kan göras i XP,

där rollerna kan variera mellan gruppmedlemmar beroende på tillfälle. I ett framtida större projekt eller med en annan typ av grupp, hade jag emellertid gärna sett att de olika rollerna fanns definierade. Gruppmedlemmarna agerade även kunder och tillbringade tid i början av projektet med att formulera User stories som sedermera diskuterades på ett måndagsmöte för att bestämma riktning med projektet. Det huvudsakliga verktyget här var Pivotal tracker vars Icebox, innehållandes utkast till User stories, fungerade som ett bollplank och diskussionsunderlag, medan Product backlog och Sprint backlog fylldes på efter diskussioner och under måndagsmöten. Product backlog prioriterades inte i någon nämnvärd utsträckning, men sprinterna planerades ändå med utgångspunkt i att få en fungerande kärnfunktionalitet fortast möjligt. Just när det gäller Pivotal tracker hade arbetet kunnat styras upp bättre för att på bästa sätt kunna utnyttja funktionerna i Pivotal tracker – som det var blev Pivotal tracker både ett verktyg och ett mål i sig. Först när det bara är ett verktyg som hela tiden utnyttjas och uppdateras kan det utnyttjas tillfullo.

Sprinterna som gruppen planerade hade mer XP- än Scrum-karaktär i det att de dels sträckte sig över endast en vecka (vilket Scrum visserligen också kan göra, men vanligare är längre) och dels att den planerade sprinten kunde förändras. Om en gruppmedlem exempelvis åtagit sig en uppgift, ägnat tid åt att lösa den och sedan kommit på att den inte var till nytta för projektet, kunde uppgiften diskuteras i gruppen och sedermera bytas ut mot en annan uppgift. Gruppen eftersträvade att den kod som laddades upp till repository i verktyget GitHub skulle vara fungerande vilket så gott som hela tiden vidmakthölls; gruppens Increment var vanligtvis redo för eventuell release vid slutet av varje sprint. Sprint-upplägget fungerade bra i gruppen, kanske främst på grund av att gruppmedlemmarna tog initiativ, både till att åta sig uppgifter i sprinten, och till att erbjuda hjälp eller fråga efter nya uppgifter när den egna väl var klar. Koden ägdes kollektivt i enlighet med XP – varje gruppmedlem var fri att gå in i någon annans kod och optimera eller kommentera; alla tog ansvar för att all kod skulle fungera. Även refactoring skedde både av den som skrivit koden och andra i gruppen.

Gruppen utnyttjade inte Burn down chart annat än för att visa på utvecklingsprocessen i efterhand. Dessa diagram hade emellertid kunnat utnyttjas bättre för att låta gruppen hålla sig informerad om hur utvecklingsprocessen fortskred. I ett framtida projekt skulle jag utnyttja detta, då det både är tidseffektivt (om övriga delar av Pivotal tracker utnyttjas ordentligt) och informativt.

User stories var som nämnt den främsta utgångspunkten för att kommunicera önskad funktionalitet. Gruppen producerade knappt 40 User stories som alla låg till grund för minst ett requirement. Gruppen arbetade därtill utifrån den initiala produktvisionen ”A Complete Chalmers Experience” som också kan betraktas som en metafor i enlighet med XP-metodologi. Det var denna metafor som hela tiden låg i grunden för alla User stories – de jämfördes mot visionen och bedömdes tillföra mer eller mindre värde till just produktvisionen. Det var mycket nyttigt att ha en sådan gemensam grund att stå på och utgå ifrån och jag skulle gärna använda det i ett framtida projekt.

Gruppen hade initiala intentioner till att utnyttja Test Driven Development som XP definierar det under projektet. Det blev emellertid inte riktig TDD (där ingen kod ska skrivas förrän ett test först har skrivits), men istället testintensiv kodning. Ingen funktion pushades till repository utan att vara testad och fungerande, men testerna

skrevs alltså i efterhand. Jag är osäker på om jag skulle vilja applicera TDD i ett framtida projekt av liknande karaktär; jag förstår fördelarna med att säkra att koden fungerar eller genom testskrivande bedöma om funktionen alls är till nytta, men om testen skrivs i efterhand kan det vara lättare att få en dynamisk programmeringsprocess. I mer omfattande projekt kan jag emellertid se att utbytet för tid-nytta vid TDD skulle ge bättre utdelning.

I detta projekt använde gruppen både unit tests och acceptance tests. Databasklasserna unit testades främst ”to pass” vilket hade kunnat utvidgas, särskilt om det ska användas som en grund för vidare utveckling. Även Acceptance tests utfördes, minst ett per User story, vilket föranledde mer än femtio tester. De blev alla manuellt utförda då automatiserade tester inte bedömdes ha gott tid-nytta-utbyte; det skulle ta längre tid att skriva testen än att skriva och utföra dem manuellt. Hade projektet varit mer omfattande hade automatiserade tester fått en mycket större nytta; det hade dessutom blivit lättare att få en god översikt över branch coverage.

Gruppen tog mer eller mindre automatiskt fasta på att kommunicera, ge feedback, skapa feedback genom tester, ha respekt för andras åsikter och kod liksom för sina egna, samt att våga slänga kod eller våga optimera den – typiska XP-metoder som jag gärna också skulle se i ett framtida projekt. Exempelvis hade gruppen initialt och en bra bit in i arbetet planerat för att ha två lägen i applikationen; dag och natt. I enlighet med XP där designbeslut ska tas när de blir aktuella befanns denna initiala tanke vara överflödig och onödig varför gruppen bestämde att den skulle tas bort – vågade slänga kod. Dessa tekniker sparade både tid och kraft som annars skulle ha lagts på att hitta omvägar eller diskutera. Gruppen var öppen för förändring och även olika medlemmar ville försvara ”sin” kod, förblev diskussionsklimatet bra och logiska argument slog subjektiva – en mycket tillfredsställande kultur som jag tror kan vara extremt viktig just i programvaruutveckling.

Gruppen gjorde två releases, en alpha och en final. Detta går i linje med XP där små releases släpps kontinuerligt. Kanske hade gruppen kunnat släppa ännu en release, men på grund av projektets utbildande moment och storlek blev det endast två. Dessutom fick gruppen mycket sent reda på att den skulle släppa flera versioner, varför gruppen inte hade möjlighet att planera så mycket för den första; då lades tiden istället på att fortast möjligt få ihop en fungerande kärnapplikation. I en framtida kurs eller projekt skulle jag gärna jobba mer med många små releases, dels för att det är vedertagen metodik, och dels för att det verkar vara både tidseffektivt och utvecklande för programmeraren, men i så fall med mer framförhållning.

Sammanfattningsvis fungerade det upplägg som löpande mejslades fram mycket bra i gruppen; utvecklingsprocessen fick många karaktärsdrag som liknade agile development. Gruppen hade tidigt en fungerande produkt, interaktion mellan gruppmedlemmar skedde frekvent, det fanns ett förändringsvänligt klimat och fungerande kod var en, åtminstone informellt, mätare på framgång. I ett framtida projekt skulle jag emellertid vara konstruktivt kritisk till delar av vårt projektupplägg då dess effektivitet dels berodde mycket på att projektet inte var så omfattande, och dels på att gruppkulturen var både öppen, hjälpsam och initiativrik (trots att endast två av medlemmarna arbetat ihop tidigare). Jag anser också att visionen uppnåddes på ett bra sätt genom flexibel och öppen programvaruutveckling.

Tid som lades ned

Olika gruppmedlemmar spenderade det mesta av sin tid på olika saker, främst med utgångspunkt i respektive persons styrkeområden, samt vad som behövde göras för applikationen skulle nå den initiala visionen. Exempelvis arbetade Anders mycket med navigation, Niklas med GUI, Rene med datainsamling och blandad kodning, Fredrik med databasen och jag själv med dokumentation. Samtidigt hade alla gruppmedlemmar något finger med i så gott som alla områden – som ett sätt att kvalitetskontrollera, vara bollplank eller hjälpa varandra.

Gruppmedlemmarna la ner olika mycket tid under olika skeden i projektet på grund av att olika medlemmar hade olika tung arbetsbörda i parallella kurser. Medlemmarna registrerade sina respektive tider i en intern arbetsdagbok. Arbetstidsfördelningen fungerade bra och de relativa insatserna jämnade ut sig under den totala projekttiden som uppgick till ca 375 timmar.

Vad som fungerade bra och mindre bra i gruppen

Som nämnt i ovanstående texter fungerade samarbetet över förväntan bra. I och med gruppmedlemmarnas olika styrkeområden, men ändå lika stora ambitionsnivå, blev gruppen så bra jag personligen skulle kunna begära. Gruppmedlemmarna tog initiativ, även till uppgifter som inte betraktades som ”roliga”, men som ändå måste göras. Jag tror att våra olika bakgrunder och erfarenheter, förutom ambition och öppenhet, var våra främsta fördelar. Innan detta projekt hade jag gissat att ett gäng duktiga IT-studenter skulle göra detta projekt allra bäst, men med tanke på alla de olika aspekter som finns i ett projekt av denna typ, har jag lärt mig att gruppdynamik och olika referensramar kan övertrumfa sådan expertis. Exempelvis kunde jag själv ägna mig åt att utforska och planera dokumentation, samtidigt som jag följde kodutvecklingen för att kunna vara ett bra bollplank vid diskussioner. På detta sätt skapade jag maximalt med dokumentationsvärde, samtidigt som övriga gruppmedlemmar kunde diskutera dokumentation med mig, jag deras kod med dem, och de koncentrera sig på att skriva värdeskapande kod, presentation eller datainsamling. Jag är, med grund i resonemanget ovan, osäker på om gruppens slutprodukt hade blivit bättre om alla gruppmedlemmar hade befunnit sig på samma kunskapsnivå inom programmering. Med tre data-studenter och två industriella ekonomer blev denna projektkurs mycket tillfredsställande.

Efter att ha funderat länge kan jag inte komma på några stora nackdelar med gruppens samarbete. Det enda moment som ibland kunde inverka negativt var en liten grad av språkförbistring då alla helgruppsdiskussioner fördes på engelska för att inkludera tyskspråkige Rene. Eventuella missförstånd reddes emellertid snabbt ut, och inte heller Renes annorlunda skolerfarenhet medförde några problem i arbetet.

Att göra annorlunda i framtida projekt

I ett framtida projekt skulle jag vilja engagera mig mer i att framställa kod, främst för att öka min kompetens inom kodning i större projekt. (Jag kommer också göra det privat utanför Chalmers för att utveckla Bliss-appen som var ett av våra tre initiala projektförslag, men med C# och för Windows-plattform istället för Java och Android).

I ett framtida projekt skulle jag också söka utnyttja och registrera händelser i Pivotal tracker till fullo från allra första början, nu när jag känner till funktionerna och vet hur de kan användas. Gällande GitHub skulle jag absolut utnyttja det om jag arbetade i grupp, men om gruppen inte kommer överens om bra användningsregler skulle jag vilja avstå från att utnyttja Branch-verktyget och endast pusha rakt till mastern med färdig kod. Detta för att undvika merge-konflikter och tidsödande manuell rättning av kod.
Jag skulle också gärna se att agile development fick komma till sin rätt bättre i form av continous developing pace. Detta skulle kräva bättre planering och troligtvis även bättre schemaöverensstämmelse hos gruppmedlemmarna än vad som var fallet i detta projekt. Därmed inte sagt att utvecklingsfarten var mycket olika, men det blev ibland punktinsatser som hade kunnat omvandlas till kontinuerliga, mindre insatser.

Fler enheter att testa koden på skulle jag nog sätta som ett krav inför framtida projekt. Emulatorerna var både svåra att få att fungera och extremt långsamma när de väl körde. Som det var fanns det turligt nog några olika Android-telefoner, men inte tillräckligt många för att alla gruppmedlemmar skulle kunna köra appen. Detta ledde till utdragna testprocesser och osäkerheter i vad som faktiskt fanns att tillgå.

Kommentar

Jag vill avsluta med att ge Max cred för att han hade möte med oss varje fredag och verkligen la tid på att förstå våra problem och hjälpa oss med både kod och projektinriktning.