Reflektionsdokument

SE Project, Grupp 30
ht 2012

+Reflektion Doc
Alla föredrog olika metoder, och vi provade allihop till och från under projektets gång. Vi har använt bl.a. Google Docs, rena textdokument, latex. I vanligaste fallet var det 1-2 personer som skrev på varje dokument.

Google docs var både positivt och negativt, möjligheten att göra spreadsheets är jättebra, men om någon annan börjar ändra i ett dokument de tagit från repot (istället för att ändra på google docs) så kan man göra en hel del dubbelarbete. Den stora fördelen vi såg var dock “multiplayer”-möjligheten, att kunna skriva på samma dokument och i realtid se vad andra gör för ändringar.

Version control, för de dokument som vi **inte** hade i google docs så var de nästan uteslutande rena textfiler som vi versionshanterade. Vad gäller google docs-dokumenten finns där ingen möjlighet att versionshantera eftersom pdf/odt behandlas som binärer, så här har det ibland blivit lite problematiskt. Generellt sett har vi haft google docs-dokument som färdigställts under en sprint och sedan pushat pdf-kopior till repot, detta har inneburit att dessa inte alltid har varit aktuella och reflekterat kodens/projektets tillstånd.

Om vi hade börjat från början kanske vi hade hållit oss till versionshanterade textfiler rakt igenom, om vi inte känt ett trängande behov av spreadsheets.

+Reflection SE
Test Driven Development.
Med inledande dålig kunskap om programmering till Android var det svårt att bygga en fast stomme till applikationen redan från början. Då hade mycket arbete blivit dubbelt då vi hade fått skriva om testerna när vi gjorde större ändringar i arkitekturen. Såhär i efterhand med mer “kött på benen” och större koll på SDK:t hade det varit lättare att bestämma sig för en viss uppbyggnad och börja skriva tester direkt och köra testdriven utveckling.

GIT-användning.
Detta var i början av projektet ett av de största problemen för oss, då ingen av oss hade någon erfarenhet av git, och majoriteten av gruppen inte hade använt sig av någon form av versionshantering innan. I början då alla tyckte det svårt att använda git blev det att vi väntade för länge mellan varje gång vi la upp kod och det blev då ofta konflikter. Mot slutet har vi varit bättre på att använda git, men vi borde kanske ha branchat mer och på det sättet bättre ha utnyttjat fördelarna med git. Github har på det stora hela fungerat bra, t.ex. buggrapporteringen där har vi verkligen haft nytta av. Att man kan få notifications på mejl gör att man snabbt och

enkelt kan diskutera en bug och dess fix utan att behöva sätta sig ner vid en dator, och att ett flertal lösningar kan diskuteras innan man implementerar den slutgiltiga.

Vad tyckte vi om SCRUM-utveckling.
Att börja med att skriva User Stories var intressant, och det blev lättare att prioritera de olika uppgifterna som vi behövde utföra. En annan positiv sak med scrum som blev central för vårt artbete var sprint backlogs, att ha ett dokument där man hela tiden kan se hur sprintens mål betas av och vad som kvarstår att göra har varit jättebra. Vi ser alla fram emot att testa att jobba med scrum på mer konventionellt sätt, där rollerna blir tydligare, i framtiden. Att rollerna blivit dubbla för två personer har inte varit något större problem, men arbetet blir säkert något annorlunda när produktägaren inte också är en utvecklare.

Vad tyckte vi om ant/EMMA.
Ant har fungerat jättebra för vårt huvudprojekt, lätt att sätta upp och har sedan dess fungerat kanon. Lite klurigare var att konfigurera och sätta upp testprojektet med ant/EMMA, då vi hade problem med hur emma-jarfilerna skulle importeras in i projektens respektive Build/Class path. Efter utförligt googlande och experimenterande lyckads vi till slut att få igång det utan att förstöra något annat. När denna konfiguration görs rätt (hur man gör detta står i developer guide) så fungerar att bygga och automattesta med ant/EMMA också problemfritt. Hade vi varit lite snabbare med att få igång ant, och om jenkins hade haft någon möjlighet att köra android jUnit så hade vi säkert haft nytta av den tjänsten. En bit in i vårt arbete med unit tester övervägde vi att dela upp testprojektet i ett android jUnit-projekt och ett standard jUnit-projekt (för de tester som inte behövde något androidspecifikt). Vi ansåg inte att automatbyggen med jenkins i sig var värt mängden arbete för att skapa ett nytt projekt och flytta en massa kod, när vi ändå kan automattesta lokalt.

Övrigt.
Vi utforskade möjligheterna att använda oss av Pure Data (http://puredata.info/), vilket är en grafisk programmeringsmiljö speciellt inriktad på digital signalbehandling och ljud, och lade en ansenlig mängd tid på att leka med/lära oss hur detta fungerade. Vi beslutade att inte använda oss av PD eftersom det dels skulle krävt en rätt stor refaktorisering av vår kodbas, och då en hel del av koden skulle bestå av Pure Data-patches (grafiskt konstruerade moduler), vilket hade dragit ner vår mängd betygsgrundande javakod. Vid eventuell vidareutveckling av appen kan ett skifte till PD vara väldigt intressant.

+Reflection Team
Att jobba i ett team kräver betydligt mer planering och klarare strukturer för att fungera, att bara börja koda i var sin ände och hoppas att man ska få ihop det fungerar inte. Att köra scrum har hjälpt oss att undvika att arbeta oplanerat och behöva reda upp en massa problem i efterhand. Även om versionshantering med git är bra så hjälper det föga när flera personer sitter och löser samma problem fast på olika sätt och den som pushar först vinner.

Fördelar finns givetvis, främst i att den gemensamma kunskapen och erfarenheten är större än för en enskild individ. Någon är riktigt duktig på att bygga databasimplementeringar, någon

annan bra på att sätta upp ett intuitivt GUI, etc. Detta vägde i mångt och mycket upp diverse problem med att jobba i team. Det hade varit bättre att ha ett centralt scrum-verktyg där man kan ha projektplanering, sprints och backlog. Det har inte vi vilket gjort det svårt att få en överblick på vad som är gjort snabbt.