TDP003 Projekt: Egna datormiljön

Reflektionsdokument

Författare
Maximilian Bragazzi Ihrén, maxbr431@student.liu.se

Höstterminen 2014
Version 1.0

1. Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Skapad inför inlämning.	140908
1.1	Skrivit mer om min upplevelse med Emacs, vilka fördelar och nackdelar det kan ha. Skrivit om hänvisningen under punkt 3.3 till korrekt format, utvecklat mina tankar inför framtiden. Lagt till referenslista.	150104

2. Inledning

Jag jobbade tillsammans med Markus Lewin (marle943).

När vi jobbade med kodning så blev det mest jag som skrev själva koden, men Markus satt alltid bredvid och diskuterade vad det var vi gjorde. Vi skrev inte något utan att diskutera det först, så att båda av oss blev nöjda med lösningen. Om det var något som en av oss inte förstod så frågade vi och fick det förklarat.

Däremot, när vi jobbade med alla dokumenten man skulle skriva efter koden så bytte vi plats, så att Markus var den som skrev för det mesta. Denna uppsättning var inte något vi hade planerat på (både för kod och för dokument), men det blev så rent naturligt. Jag tyckte systemet fungerade relativt bra, även om man blev lite uttråkad om man inte var den som skrev.

3. Erfarenheter

Under arbetet har vi haft många positiva erfarenheter och väldigt få negativa. Det som var negativt var även väldigt små saker, så som att vi blev oroliga att en ändring vi hade gjort i koden nyligen skulle förstöra något när vi skulle redovisa (även om det aldrig inträffade).

3.1. Projektplanering

Det första vi gjorde i projektet var att skissa upp generella bilder på hur vi ville att hemsidan skulle se ut med hjälp av GIMP, så vi visste mer exakt vad vi behövde veta och hur vi skulle kunna uppnå det. Det hjälpte ganska mycket när vi skulle designa hemsidan senare (det hjälpte även med LoFi-prototypen, då vi redan i början av projektet visste den generella designen av sidan).

När vi hade ritat bilderna så använde vi dem för att skriva upp vad vi skulle behöva kunna om HTML och Flask för att kunna uppnå detta (så som hur man får in data från Flask till HTML-sidan, hur man använder ”form”-taggen i HTML, etc), vilket vi senare utvecklade till projektplaneringen. Personligen tyckte jag att detta var en bra metod, och det verkar Steve McConnell (2004, 23-55) hålla med om, då han skriver att, likt en byggarbetare som jobbar på ett hus, så är bra planering mer effektivt på alla sätt, dvs. att planera väl sparar in mycket tid och resurser i slutet av ett projekt, t.ex. om man har en massa buggar eller om man har glömt någon punkt. Om man planerar väl i förtid så skulle inte sådant hända.

3.2. LoFi-prototyp

När vi gjorde LoFi-prototypen började vi med att göra en HTML-mall av vår hemsida. Vi hade först tänkt oss att vi skulle rita dem på papper, men vi kom fram till att det skulle vara enklare att göra det direkt i HTML, plus att vi skulle ha tillgång till en färdig mall när själva hemsidan skulle byggas.

Vi försökte oss på att använda en hemsida som hette GoMockingBird, men vi förstod inte så mycket av hur man skulle använda den på de ca 10 minuterna vi försökte, så vi bestämde oss för att skriva allt för hand i Emacs. Det verkade som det skulle bli svårt först, men när man hade gjort klart CSS:en så blev allt mycket enklare.

Emacs var ett väldigt bra program att använda, då det kunde öppna alla filtyper som vi hade kodat i, det är väldigt enkelt att använda när man väl har satt sig in i alla kortkommandon. Kortkommandon i Emacs är både dess störta fördel, men även dess största nackdel. Ofta så skiljer sig kortkommandon i Emacs från vad de flesta är van vid, t.ex. så är kortkommandot för att klistra in satt till Ctrl + Y istället för Ctrl + V. När man väl har satt sig in i dessa kortkommandon är dock Emacs ett av de bästa verktygen att jobba med i programmering, och inte bara för att jobba med Python/Flask.

Dock så saknar Emacs några av de funktioner som programmerare idag är relativt vana vid när de jobbar med ett IDE, t.ex. att programmet föreslår variabelnamn medans du skriver. Dock så är detta bra just för oss som elever då vi behöver tänka på allt sådant, vilket i sin tur gör att man lär sig bättre.

När man satt där i början av projektet tyckte jag att vi kanske hade spenderat för mycket tid på LoFi:n, men nu i efterhand kan jag säkert säga att det var värt det (speciellt det där med att ha en färdig mall till den ”riktiga” hemsidan).

3.3. Datalager

När vi började med datalagret gick allting snabbt framåt. Med hjälp av API:n så blev vi klara med allt förutom sökfunktionen på ett labbtillfälle! Såklart så var alla de funktionerna inte lika bra som de vi tyckte var ”release”-värdigt, men de fungerade (vi hade t.ex. ingen felhantering alls vid det laget).

Sökfunktionen var dock en helt annan grej. Vi hade stora problem med den, speciellt när det kom till sorteringen. Den första knappt fungerande versionen hade vi fyra funktioner (om man inte räknar med sorteringen), och de var uppbyggda så att man hade en funktion för att söka i allt, en funktion för att söka i tekniker, en för att söka i fields, och en sista för att söka i både tekniker och fields. Det var mindre effektivt än det låter, till och med (som jag skrev i dagboken: ”Skrev klart sökfunktionen i datalagret, blev lika mycket kod som alla de andra funktionerna kombinerat.”). Steve McConnell (2004, 43-55) skriver mycket om hur viktigt det är att ha en utlagd design-arkitektur för att ha ett väl fungerande program. Han tar t.ex. upp att du utan bra arkitektur skulle kunna ha rätt problem men fel lösning, vilket var precis vad vi hade för problem i koden beskriven ovan.

Till slut kortade vi ner det till tre funktioner (igen, förutom sorteringen), vilka var själva sökningen och även en koll för tekniker och en koll för fields (vilka båda bara returnerade True eller False).

Nu i efterhand kan jag säga att det var bra att vi blev klara tidigt med det grundläggande i funktionerna, då det gav oss mer tid till att testa felhantering, och hur vi skulle kunna angripa det på bästa sätt. Jag kommer även att börja planera väl i förväg till framtida projekt, så att jag inte får en

så ful kod som ”Search”-funktionen var till att börja med.

3.4. Flask (och Jinja i mindre grad)

Vi började kolla på hur Flask fungerade och så ett par dagar innan själva föreläsningen vi skulle ha på det, men vi förstod nästan ingenting av det som stod på Flasks hemsida.

Efter föreläsningen var dock allt väldigt klart för oss hur Flask fungerade. Vi gjorde klart allt förutom söksidan på en eftermiddag, men vi sade båda att det skulle kunna förbättras design-mässigt (även om vi inte jobbade något mer på designen förutom att vi lade till att knapparna i menyraden skulle ”fade-a” ut i bakgrunden, men vi skyller på att vi är programmerare, inte designers).

Söksidan tog lite extra tid eftersom ingen av oss visste hur man använde ”form”-funktionen i HTML, men när vi väl kom på hur det fungerade, och hur man gjorde checkboxar, drop-down-menyer och liknande gick det snabbt framåt.

Som jag sa om datalagret så tycker jag att det var bra att vi skrev klart det mesta av den grundläggande koden så snabbt som möjligt, eftersom vi behövde testa felhantering och hur man skulle kunna göra det bäst.

3.5. Dokumentinlämningar

När vi skulle skriva installationsmanualen och användarmanualen så bestämde vi oss helt enkelt för att skriva en var. Jag tog installationsmanualen för att jag trodde att det skulle vara enklast (pga. målgruppen var enklare), men det visade sig att det var raka motsatsen. Detta eftersom att man behövde förklara allt in i minsta detalj så att en person som inte har använt Linux så mycket skulle förstå. Till exempel kunde man inte bara skriva att man skulle navigera till en viss mapp, utan man behövde förklara att man skulle använda ”cd”-kommandot och hur det fungerade. Plus att det ingick att man skulle lägga upp sin portfolio på en server och ha den körandes där. Ingen av oss visste hur man gjorde det om man inte ville ha en terminal öppen på sin egen dator hela tiden så fick vi kolla upp det, och det tog lite tid (det visar sig att det var väldigt enkelt, finns saker (”tmux”) inbyggda för sådant).

När vi började med systemdokumentationen så jobbade vi tillsammans igen, men även fast vi gjorde det så blev det mycket mer komplicerat än vi hade tänkt oss. Vi tyckte att det skulle vara mycket enklare att förklara koden om vi hade sekvensdiagram för alla delar vi skulle förklara, men att skriva alla sekvensdiagrammen tog väldigt lång tid (det var inte svårt dock, förutom att man blev rätt så uttråkad efter ett tag, då det blev mycket repetition). Vi fick dock ”MVG”/”G++” på det dokumentet, så jag kan säga att det var väl värt det!

Testdokumentationen skrev vi efter att ha haft ett projektdemonstrations-seminarium, där alla gick omkring och försökte sabotera alla andras system. Pga det fick vi mycket saker vi kunde använda i testdokumentationen, så det var inte lika jobbigt som de dokument vi hade innan. Dock så var det ett krav på att man skulle bevisa att alla kraven som ställts på hemsidan hade uppfyllts, och det var inte alltid allt vi hade skrivit ner från seminariet täckte kraven ordentligt.

3.6. Samarbetserfarenheter

Personligen har jag inte jobbat tillsammans med någon annan på ett programmerings-projekt (om man inte räknar att man har bett någon att göra några få bilder), så jag tyckte att det var bra träning

inför eventuella framtida jobb.

Dock tror jag det hade fungerat mycket bättre om vi hade haft ett större projekt utan krav på att vissa saker ska vara färdiga för inskickning vid vissa datum, och fler personer i gruppen. Då skulle man ha kunnat simulera något som är mer liknande till ett projekt ute i arbetslivet. Men eftersom det var det första projektet i utbildningen kan jag förstå att man vill introducera oss elever till koncepten runt projekt innan man kastar in oss i ett större samarbete.

Men själva samarbetet i sig har fungerat väldigt bra. Det var bra att kunna ha någon att prata med och bolla idéer.

3.7. Inför framtida projekt

De enda tips jag skulle kunna ge mig själv skulle vara att följa det generella jag har gjort i detta projekt:

1. Planera noggrant innan du börjar
2. Bli klar med själva kodningen så snabbt som möjligt, så att feltestning och felhantering kan ta upp det mesta av tiden
3. Prata mycket med dina arbetskamrater och se till att man förstår vad den andra gör.

4. Egna tankar

Generellt så skulle jag säga att jag är nöjd med projektet. Koden i resultatet är jag väldigt nöjd med, till skillnad från designen, men jag är som sagt inte designer, så vad gör det? Speciellt så är jag nöjd med hur sökfunktionen ser ut nu, jämnfört med hur den såg ut förut.

Jag har lärt mig mycket under projektets gång, allt från hur man använder Flask till hur det är att arbeta i grupp, och det känns som det kommer vara något bra att ha med sig i framtiden. Att lära sig om hur Subversion fungerar har också varit väldigt intressant (men från vad jag har läst så tror jag att jag kommer föredra Git).

5. Referenslista

McConnell, Steve. 2004. *Code Complete Second Edition*. Microsoft Press.