LINKÖPINGS UNIVERSITET

TDP003 Projekt: Egna datormiljön

Reflektionsdokument

Författare

Joakim Johansson, joajo229@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2016
Version 1.1

2016-12-18

1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.1	Skrivit klart alla punkter och exporterat pdf.	161218
1.0	Skapat dokumentet och börjat skriva lite på sammarbete och förra reklektionsdokumentet	161217

2 Föregående reflektionsdokument

I föregående projekts reflektionsdokument återgavs det vikten av att använda GIT redan från början i projektets, precis som vi gjorde i detta projekt. Jag måste säga att det gick mycket bättre med synkning och sammansättningen av versioner i detta projekt än i förra projektet.

Precis som det återgavs i förra reflektionsdokumentet fokuserade vi här på att producera en bättre projektplan. I detta projekt skapade vi en bättre design specifikation, vilket underlättade arbetet under projektets gång.

3 Vad har jag fått ut av kursen, vad har jag lärt mig?

3.1 Designspecifikation och klassdiagram

Vi gjorde lite annorlunda än vad andra grupper gjorde, då vi satsade mycket på klassdiagrammet och designspecifikationen och planerade hela spelet redan innan vi började skiva koden. Detta ledde till att vi hade koll på strukturen och uppbyggnaden av spelet tidigare jämfört de andra grupperna.

Den nackdel som vi noterade var att det stundtals blev svårt att planera ett spel som ska använda sig av SFML innan den är testad. Dock är det svårt att planera något som man har en begränsad insyn i, vilket vi under projektets gång åtgärdade genom att göra några mindre stickprovstester under arbetsgången.

Fördelen med att ha ett massivt klassdiagram och designbeskrivning är att när vi sedan skulle börja skriva all kod tog det bara några dagar och sedan hade vi alla klasser fullt fungerande, visserligen med några mindre buggar men det gick väldigt snabbt att skriva allt eftersom att klassdiagrammet skulle mer eller mindre bara kopieras utan någon större eftertanke.

3.2 Testdriven utveckling

Efter klassdiagrammet och designspecifikationen, när vi skulle börja skriva koden, valde vi att arbeta på ett testdrivet sätt. Detta gjorde att vi skapade flertalet testfall i Catch som vi sedan försökte få vår kod att klara sig igenom med hjälp av våra klasser och funktioner.

Jag har aldrig tidigare arbetat med testdriven utveckling i ett sådant här stort omfång tidigare men jag måste erkänna att jag uppskattade det. Det finns dock en nackdel. Vi jobbade mycket med våra testfall vilket gjorde att våra funktioner och classer skulle fungera perfekt. Detta var en aningen ostimulerande eftersom att man aldrig fick se spelet på skärmen under testerna. Inte förrän våra klasser gick igenom samtliga tester kunde vi började sätta ihop allt och skapa ”main”-loopen. De sista stegen jämfört med övriga moment i processen upplevdes däremot relativt enkla.

Jag tror att jag kommer jobba mer med testdriven utveckling i framtiden. Till en början låter arbetssättet tråkig men den är faktiskt ganska stimulerande efter man har testat på den.

3.3 Doxygen och dokumentation

När det var dags att exportera dokumentation från vårt projekt använde vi oss av ett program som heter Doxygen. Vi placerade dess fil i vår root-katalog och den började skapa en stor mapp men en mängder av HTML filer. Detta gjorde att vi fick gedigen dokumentation utav programmet i form av en hemsidan, vilket är riktigt smidigt.

Enda nackdelen med Doxygen i detta fall är att den läser kodfilerna och letar efter speciella kommentarer som den sedan läser in och lägger som en text på HTML-sidan. Detta gjorde att vi var tvungna att gå igenom koden och skriva in fler kommentarer med i den speciella stilen som Doxygen gillar. Det var ganska mödosamt men i slutet lyckat.

I framtiden tror jag att jag kommer försöka skriva kommentarerna på det sättet Doxygen gillar redan direkt medan man skriver koden för att sedan undvika extrajobbet med att skapa dokumentationen.

3.4 Granskning och förståelse av någon annans kod

När vi skulle granska vår beställargrupps kod var det riktigt intressant. De använde en helt annan metod jämfört med oss när de arbetade. Vi gjorde hela vår ”backend” men funktioner och klasser först och sedan implementerade det grafiska. Vår beställargrupp jobbade lite med båda samtidigt för att bygga mer och mer på spelet. Detta tycker jag har både nackdelar och fördelar. En av de positiva delarna med det är att vi lätt kunde se hur det går för dem i projektet enbart genom att testspela spelet eftersom att nästa all kod som var skriven påverkade spelets grafiska del direkt.

4 Samarbetet

Jag tycker att samarbetet med Alexander har går väldigt bra. Vi har jobbat mycket tillsammans bland annat genom att en av oss skriver kod och den andra kollar på och hittar buggar och vise versa. Vi har även försökt dela upp delar av projektet så att vi kan jobba parallellt för att spara tid.

Nackdelen är att det ibland blev problem med olika sammanslagningar på git. En gång fick vi inte ordning på det och det resulterade i att vi gjorde en rollback och sedan skickade den ändrade koden till den andra datorn för att lägga in den på rätt ställe i koden. Men jag antar att detta går att lösa genom att man skaffar sig lite mer erfarenhet med git och dess funktioner.

5 Annat

Sammantaget tycker jag att det har varit ett roligt och lärorikt projekt och jag kommer förmodligen arbeta med flera projekt av denna typ i framtiden.