Gemensam reflektion

1 Gruppreflektion

Innehåll

1.1 Kommunikation

Eftersom vi alla oftast suttit i samma sal när vi jobbat i par så har kommunikationen gått väldigt bra. Det gick snabbt att kunna diskutera med de andra paren och klargöra eventuella missförstånd. Vissa missförstånd blev det ändå men det kunde blivit mycket värre. Vi använde Slack för att få tag på varandra när vi inte var i skolan och det fungerade också bra.

Den externa kommunikationen har fungerat mindre bra. Oklarheten över vad handledarens uppgift var i projektet försvårade den externa kommunikationen.

1.2 Samarbete

Här och i våra individuella reflektioner vill vi bli examinerade på [Y67]:
Det har gått relativt bra att samarbeta inom gruppen, de flesta har kunnat vara närvarande nästan varje dag och det har varit lätt att diskutera saker. Parprogrammeringen har fungerat bra, det blir självklart skillnader mellan olika par och vem man jobbar med. På grund av stressen i slutet av projektet använde vi oss inte alltid av parprogrammering de sista dagarna då mycket fokus låg på att få klart kod och vi inte alltid satt två och två, utan ibland nästan hela gruppen tillsammans för att lösa ett problem.

1.3 Koordination

Koordinationen har gått bra under projektet förutom för sista sprinten. Då var de flesta delsystemen klara och all fokus blev på GC:n. Samtidigt luckrades paren för parprogrammeringen upp pga. stressen och det blev mer godtyckliga par för varje dag.

1.4 Beslutstagning

Den första planeringen tog ganska lång tid, men det berodde inte på att vi var oense utan mest tidsbrist. Att ta beslut har varit en demokratisk process där alla har fått vara med och tycka till, men där de som hållit på med vissa moduler fått lite mer att säga till om då de kan delsystemet bättre.

1.5 Process

Här vill vi bli examinerade på [Y64]:
Vi valde att jobba med delar av Scrum och testdriven utveckling (TDD). De viktigaste delarna som vi jobbade med var dagliga morgonmöten där man beskrev hur vi låg till, uppdelning i sprints och planering runtom dessa, och att skriva tester innan implementation för att hjälpa till med design. Vi använde Trello för att planera mer i detalj vad som skulle göras under en sprint.

Vår planering för sprintarna låg på ganska hög nivå så att paren kunde se efter tidsbegränsningar och behov vad som skulle göras. Upplägget kan ses i vår burndown-chart.

I början följdes processen mer än den gjorde senare i projektet. Det kändes som om att den skulle vara givande men än en gång kom tidspressen i vägen och fick oss att tänka om. Vi har ändå haft mycket kontakt med nästan dagliga möten och försökt hålla alla i fas med vad som händer.

1.6 Kodstandard

Här vill vi bli examinerade på [Y65]:
Vi valde att använda GNU:s kodstandard ($https://www.gnu.org/prep/standards/html_node/Writing-C.html$). Den stora hjälpen med kodstandard har varit att allt ser ut på samma/liknande sätt, även fast standarden i sig inte var det viktigaste. Läsbarheten ökade och missförstånd minskade. Vissa delar av standarden vi valde var svåra att komma igång med i början. Vi bestämde tidigt att inte lägga ner så stor vikt vid att hitta den perfekta standarden, då det viktigaste som sagt var konsekvent tillämpande av någon standard.

1.7 Kodgransking

Här vill vi bli examinerade på [Y66]:
Vi valde att göra pullrequests (PR) i slutet av varje sprint. Dessa skulle kodgranskas i början av nästa sprint av ett annat par. I praktiken blev det att vissa ändringar behövdes mitt i sprinten, framför allt i slutet av projektet, och då gjordes det PR oftare. Då kodbasen blev stor väldigt fort blev kvalitén på granskningarna lidande, det fanns tyvärr inte tid att göra riktigt grundläggande granskningar på väldigt stora PR. Sammanlagt gjordes 14 pullrequests (tom 9 januari) och det var ett fåtal gånger som saker behövde ändras innan de fick mergas till master.

Alla par jobbade lika med PR och vi tror att det höjde kodkvalitén tack vare att andra behövde titta igenom koden. Det gjorde också så att tester kunde köras en extra gång innan allt hamnade i master branchen.

Jämfört med tidigare kodgranskningar under kursen kändes dessa mer relevanta då det rörde vår "egen"kod. De kändes även svårare då man samtidigt som man granskade koden behövde sätta sig in i hur den fungerade på ett helt annat sätt (det var inte längre bara en annorlunda skriven version av ett program man själv skrivit), vilket ibland kunde vara utmanande.

Om vi skulle starta om projektet idag skulle vi försöka göra frekventare PR då vi tror att granskningarnas kvalitet skulle öka om en mindre mängd kod behöver granskas vid varje PR.

1.8 Regressionstestning

Här vill vi bli examinerade på [Y69]:
Vi utvecklade enhetstester under hela projektet och använde dessa som bas för vår regressionstestning. Dessa kan köras genom kommandot make testsom endast kör testerna och make coveragesom sammanställer code coverage för testerna. Vi har konstant behövt underhålla testerna då designen ändrats eller nya funktioner tillkommit.
OBS! Vi har problem med att få fram code coverage för två moduler (gc.c och header.c). Mer information om detta finns i Dokumentation/Enhetstestning.md i projektets Git-repo.

1.9 Sammanställning av tidsloggar

Vi har sammanställt våra tidsloggar och kom fram till följande siffror:

Namn	Möte	Plan.	Design	Impl.	Test	Dok.	Post.	Upp.	Fakt.	Diff.
Adam	6:50	9:10	4:30	47:00	21:45	3:10	7:00	85:20	99:25	-14:05
Daniel	5:15	5:40	4:45	42:30	30:00	3:00	8:00	88:20	99:10	-10:50
Henrik	6:15	10:40	9:45	14:00	37:30	1:30	8:00	78:05	87:40	-9:35
Maria	4:15	7:10	4:30	32:36	20:25	7:00	5:00	71:25	80:56	-9:36
Robert	5:15	5:40	4:45	36:30	24:30	6:30	6:00	67:20	89:10	-21:50
Simon	4:15	8:40	9:30	26:40	26:15	5:30	8:00	78:10	93:20	-15:10
Totalt	32:05	47:00	37:45	199:16	160:25	26:40	42:00	468:40	549:41	-81:06