**Kap 11 Evaluering**

**Kap 11.1 Hva gikk bra**

*Bra sammerbeid*

Vi har hatt et utmerket samarbeid i gruppen. Som nevnt har det ikke vært noen konflikter eller missnøye. Vi har jobbet jevnt gjennom hele prosjektet og hatt et godt fungerende system med ansvarsområder. Hvis noen har trengt hjelp på enkelte områder har de ikke nølet med å spørre om hjelp. Vi har hatt gode møter og har som nevnt kommunisert godt og effektivt gjennom Facebook (kap 10.6).

*Alle var arbeidsvillige*

Det stod ikke på arbeidsviljen til folk under dette prosjektet. Alle har vist engasement og innitiativ og vært villige til å lære og jobbe. Vi har måttet lære nye ting både for å gjennomføre oppgavene våre (Qt Jambi, Dia, osv) og for å koordinere arbeidet (Git, IceScrum, osv), men alle har ønsket å lære det de trengte. Enkelte på gruppen har kunnet mer på noen områder enn andre, og disse har vært flinke til å lære bort til andre.

*Strukturerte i startfasen og kom fort igang*

Vi fikk en god start, både sosialt og prosjekt-messig. Alle virket engasjerte og innsatsvillige. Vi lagde Facebook-gruppe med en gang, delte ut ansvarsområder hurtig, og kom fort igang med planleggingen av systemet.

*Flinke til å bruke Scrum*

Fra første møtet av var vi klare på at en viktig del av dette faget var prosessmetoden, og at vi være bevisste på arbeidsmetodikk. Vi var kjappe med å sette oss inn i Scrum og komme til enighet om hvordan vi kunne implementere Scrum i prosjektet vårt.

*Tilpassing arbeidsmetodikk etter situasjon og utgangspunkt*

Scrum er laget spesielt med tanke på folk som jobber med et felles prosjekt på fulltid, og tar også en rekke andre forutsetninger som ikke stemmer med våres situasjon. Det betyr såklart ikke at Scrum ikke har vært til hjelp for oss, tvert imot, men vi har måttet tilpasse metoden etter vår situasjon, og mener vi har lykkes med dette. Vi mener f.eks. det var lurt å ha noen lange møter fremfor mange korte, samt kommunisere ia Facebook. Det betyr ikke at det ikke hadde vært fordeler med å gjøre alt slik som Scrum legger opp til, men selv om løsningene våre har vært langt fra perfekte har de de ofte ikke vært langt fra å være optimale så langt vi kan se.

*Romreservasjon*

Romservasjon (kap 10.17) hjalp oss veldig. Vi fikk fred og ro i møtene våre, noe som hjalp oss med å kommunisere og arbeide mer effektivt. Et lite problem som vi kanskje kunne gardert oss mot noen ganger var at noen av rommene var vanskelige å finne, noe som gjorde at noen av oss kom for sent til enkelte møter. Vi var flinke til å bestille rom vi hadde vært på før når de var ledige, men vi kunne kanskje ha bestilt rom tidligere, i og med at vi hadde faste Scrum-møter.

*IceScrum*

Selv om IceScrum ikke er et perfekt program er vi godt fornøyde. Det var godt å ha et program som gjorde så mye av tilpasningen til Scrum for oss. Lars fikk satt det opp på sin egen server, og det fungerte smertefritt.

*Debugging*

Folk har vært flinke til å forstå hva som foråsaker bugs med tanke på at det ikke bare dreier seg om kode vi har skrevet selv, men også koden i samspill med biblioteker og rammeverk. Acceptance testing er blitt brukt for å lete etter bugs, og det har vi hatt suksess med.

**Kap 11.2 Hva gikk dårlig**

*Alle endte ikke opp med å gjøre like mye*

Noen hadde mer kompetanse på sitt område enn andre, både på grunn av kompetansen de opparbeidet seg underveis i prosjektet og kompetansen de hadde fra før. Dette gjorde at de var i stand til å gjøre mange ganger så mye per tid på dette området enn andre. På flere områder ville det for mange av gruppens medlemmer tatt veldig mye arbeid bare å komme til det punktet hvor man kan lage noe i det hele tatt. Selv om det forekom at folk underveis i prosjektet satte seg inn i helt nye områder slik at flere kunne jobbe på dette området var det ikke bare å flytte vilt rundt på hvem som gjorde hva slik at arbeidet ble jevnt fordelt. For eksempel ble det til at Lars Erik gjorde mer arbeid enn snittet bl.a. siden han var den eneste som fra før hadde hatt kurs i databaser.  
Vi prøvde etter beste evne å fordele oppgaver og ansvarsområder slik at det ble jevnt, og gitt utfordringene med å fordele arbeidet jevnt synes vi ikke vi gjorde dette så aller værst. Noen har programmert mer enn andre, mens andre har laget flere diagrammer enn andre og jobbet mer med rapporten enn andre. Ingen har endt opp med å bruke mindre tid på prosjektet enn man typisk bruker på et “Gløshaugen-fag”. Vi er takknemmelige ovenfor de som til sammen har gjort mer enn snittet av gruppa, og kan ikke se noen måte det ville vært praktisk mulig med våre forutsetninger å gjøre det slik at alle gjorde like mye.

*Litt dårlig start med dokumentasjon av arbeid*

Det var flere på gruppen som hadde erfaring med Trello fra tidligere arbeid, så vi startet med Trello. Trello var utmerket for vårt prosjekt og vår arbeidsmetodikk - tenkte vi. Det ble imidlertid en del arbeid med å tilpasse Trello til Scrum, eksempelvis måtte vi generere våre egne burndowncharts (og div annet). Når vi fikk byttet til IceScrum og lært oss å bruke det ble ting bedre.

*Burndown-chartene kunne vært penere*

Dette har nok å gjøre med at ikke alle alltid har vært like flinke til å jobbe jevnt, men det er ikke bare det. I begynnelsen var ikke alle like flinke til å skrive ned timer når de skulle gjøre det, og derfor ble det til at man tok av mange av timene på en gang mot slutten av sprinten, uten at det betyr at alt arbeidet ble gjort da. En annen grunn er at at vi på noen av sprintene begynte å jobbe rett etter at sprinten hadde startet, og tok av timer samme dagen. Når vi gjorde dette ble det ikke registrert som arbeid vi hadde gjort, men heller som om vi hadde startet med en mindre oppgave forann oss, men dette ble vi først bevisst på senere. Dessuten er at helgene er med i sprintene, som fører til at grafen blir flatere til tider. En annen ting som har gjort burndown-chartene mindre penere enn vi hadde ønsket er at vi var ganske nybegynnere på å estimere timer, spesielt i begynnelsen. Vi estimerte for lite timer på enkelte oppgaver, og ble ikke ferdig med disse innen timesanslaget. Det hendte også at vi brukte betydelige mengder med tid på ting som ikke hadde blitt spesifisert som oppgaver i sprinten.

*Misting av arbeid på grunn av Git-tabbe*

Vi brukte en del timer i starten på å lære Git, men så gikk vi over i en fase hvor vi gjorde ting etter “learning by doing”-prinsippet. Det funket bra i starten og litt videre, helt til ble Git-repoet ødelagt fordi malware-infiserte filer hadde blitt lastet opp. Vi måtte da kjøre rollback en uke tilbake i tid og mistet verdifult arbeid som måtte gjøres om igjen. På dette punktet hadde vi ikke begynt med branching.

*Qt-jambi*

Vi antok at Qt-jambi ville støtte visning av alle kart. Dette viste seg å være mye mer komplekst enn vi trodde, og resulterte i at programmet vårt kun fikk kjøre på 32-bits Java i Windows. Dersom vi hadde visst dette på forhånd hadde vi kanskje valgt et annet API for GUI.

Qt-jambi har mye funksjonalitet og fleksibilitet, men det betyr også lengre opplæring. For prosjektets tidsramme ble det derfor noe tyngre enn hva vi var komfortable med. Her kunne det altså vært bedre med et enklere API. Samtidig har flere av gruppens medlemmer lært en teknologi de kan bruke senere.

Mye tid ble brukt til å kode en mulig løsning til grensesnitt, som så ble kastet bort. Grunnen var at klassene som var brukt ikke var egnet for det de ble brukt til. Dette kunne ha vært ungått om programmør hadde gjort en grundigere jobb med research før kodingen begynte.

*Møtte på utfordringer og ble ikke ferdig så tidlig som vi hadde tenkt*

Vi ble ikke helt ferdig med programmet så tidlig som vi ønsket. Vi støtte på flere uforutsette problemer som tok lenger tid å løse enn hva vi trodde var rimelig å anta. **/\*Dette må oppdateres med henhold på de nylige problemene vi har hatt!\*/**

**Kap 11.3 Lærdom til neste gang**

*Lære versjonskontrollsystemet bedre før vi starter prosjektet*

Vi lærte oss å bruke branching sent ut i prosjektet. Tanken var at vi ville komme fort i gang, men vi hadde tenkt til å lære branching før eller siden uansett, og hvis vi hadde startet med branching og sikkerhetskopiering med en gang kunne vi kanskje ha mistet mindre arbeid i forbindelse med Git-uhellet vårt.

*Tenke mer gjennom valg av prosessverktøy*

Vi startet med Trello uten å evaluere så mye som vi ellers hadde gjort. Hvis vi hadde startet med Trello med instillingen *”la oss undersøke om vi kan få dette til å fungere godt for våre formål”* fremfor ”*la oss komme i gang*” hadde vi trolig funnet ut før at Trello ikke var et passende verktøy for oss, og startet med IceScrum fortere.

*GUI*

Prosjektet har gitt oss økt forståelse for kompleksitet i byggingen av GUI. Vi har også fått mer forståelse for hva slags komponenter som brukes i dynamiske grensesnitt. Det kunne tidligere ha vært et fokus på interoperabilitet mellom GUI og andre komponenter. Dette kunne ha vært gjort gjennom grensesnitt, abstrakte klasser eller andre former for kontrakter. Da hadde det vært enklere å skrive kode som fungerte med andre komponenter. Nå ser vi enda bedre bruken av flere av teknikkene vi lærte om i *Objektorientert Programmering* (TDT4100)!

*Organisering av filer*

Fra bunnen var det enklest å operere med filstrukturer der alt var samlet i én kategori. Dette ble rotete etterhvert som prosjektet vokste. Dersom det tidligere hadde vært et fokus på mappestruktur ville vi fått et system som var lettere å sette seg inn i for de som ikke har laget det selv.

Vi kunne bbrukt mer tid i startfasen på å lære oss rammeverket vi skulle bruke litt bedre. Dette har påvirket at vi ikke ble ferdig så tidlig som vi skulle ønske. Så mot slutten lærte vi oss skikkelig “branching” så vi ikke skulle få noen problem med ting vi jobbet på selv og selve prosjektet i sin helhet. Vi byttet tidlig over fra trello til icescrum, her brukte vi også “learning by doing” prinsippet som viste seg å gi litt problematikk underveis. Vi lagde i starten av icescrum ikke opp i “stories” men hadde bare en hovedtask på hvert emne og hva som skulle gjøres. Dette viste seg å være vanskelig å estimere timer opp i store bolker, men utover prosjektet ble vi bedre. Vi begynte å dele opp i stories og lagde bolker med hovedemner og til slutt i tasks som vi estimerte timer i. Dette viste seg å være mye enklere og mye mer overskikkelig enn det vi hadde gjort tidligere. Man kan ikke lære seg rammeverket 100% uten og bruke det, men vi skulle brukt litt mer tid på det så hadde vi kanskje spart oss de ekstra timene vi brukte på å rette opp git. Samtidig som å gjøre opp arbeid vi allerede hadde gjort. Man ser også i slutten av prosjektet at teamet har kommet langt iforhold til scrum metoden og dens rammeverk rundt scrum. Vi har alle lært icescrum bedre, og kan si at vi føler oss trygg på å kunne bruke dette videre i en annen sammenheng.

Vi hadde 2 scrum møter hver uke i hele prosjektet, dette fungerte helt ok. Det som kanskje var noen av problemene vi møtte på, var at vi fikk store diskusjoner som egentlig kunne vært plassert i andre møter. Dette gikk utover scrum møte, men så hadde vi ikke scrum møter på 15-20 min hver dag. Vi satte av 2 timer til scrum møter, og i disse ingår diskusjoner og estimeringer mm. For hovedgrunnen med scrum møter er å si hvordan man ligger an, I forhold til det man har gjort og det man skal gjøre. På disse møtene skal man egentlig ikke diskutere problemer, men vi hadde satt av såpass tid at vi gjorde det også. Med andre ord, det er viktig å forstå bruken av scrum møter. Man skal ikke oppnå noe objektivt, men si ifra hvordan man ligger an iforhold til det man holdet på med.

For å kunne lage løsninger med høy kompleksitet har det i enkelte tilfeller lønnet seg å se på kildekode til andre programmer. Teknikken med å sette seg inn i andre sin kode og gjøre bruk av andres teknikker i egen programvare har vi blitt flinkere til etter å ha kodet GUI.