**Arkitektur - Model View Controller**

Kompleksitet

Allerede på sprint 1 diskuterte gruppa hvordan vi skulle takle komplekstiten i kodebasen. Siden ingen hadde erfaring med større prosjekter fra tidligere var ingen spesielt opplyst på strategier for å takle den slags. Vi hadde fått hint fra student assistent om at Model View Controller var hensiktsmessig, så vi bestemte oss for at alle medlemmer skulle lese seg opp på temaet MVC. Det var dog ikke før sprint 4 at vi endte opp med å implementere MVC – siden behovet for struktur ikke meldte seg før vi skulle implementere db med grensesnitt.

Øivind fant ut at MVC kunne implementeres på forskjellige måter i java. Hensikten til MVC er å systematisk strukturere kodebase i 3 deler, slik at programflyt og logikk er i controller, grafisk grensnitt befinner seg i view og all data som programmet bruker ligger i model. Derfor gir faglitteratur heller ingen absolutt mal for hvordan dette skal implementeres, men foreslår hvordan den kan implementeres. Vi syntes at faglitteraturens metode 2 Og poengterer at vi står fritt i valg av hvordan implementasjonen gjøres avhengig av programmets krav og programmeringsspråkets egenskaper. MVC er kort sagt et arkitektonisk mønster som strategisk takler kompleksiteten, men programmereren står fritt likevel fritt til å velge en taktikk for hvordan dette skal implementeres.

Implementasjon

De tre første sprintene ble brukt til å prototype database, grafisk grensesnitt og gjøre oss kjent med verktøyene. Ikke før vi begynte å sette sammen løsningene meldte behovet seg for MVC. Hvordan skulle vi elegant separere GUI-kode fra databasen? Gruppen hadde nå blitt delt inn i 2 fraksjoner – GUI og database. Dette gjorde at behovet for å strukturere kode i henhold til MVC ble enda mer trykkende. Vi valgte derfor å lage mappestrukturen i henholdsvis tre java pakker, model, view og controller. Pakken ’model’ inneholdt all database-spesifikke filer, forretnings-logikk og domene-data ’view’ inneholdt alle GUI filer. Pakken ’controller’ hadde på dette tidspunktet ingen filer.

Vi så oss nødt til å lage controlleren etter hvert som grensesnitt-sidene ble ferdig-lagd.

Database-gruppen som bestod av to personer besluttet at den ene skulle lage kontrolleren til view, mens den andre skulle lage modellen. Siden forståelsen av hvilke krav modellen forandret seg underveis, ble det til at vi formet modellen parallelt med at vi lagde kontrolleren. Ved å benytte oss av å parogrammere for raskt ku. Da kunne Sigurd oppdatere modellen parallelt med utviklingen controlleren. Dette resulterte i

Den ansvarlige inntok da rollen som model-ekspert og veileder for den som lagde controlleren.

Erfaringer vi gjorde oss var at hvis en person programmerer alene og mangler dokumentasjon på kildekoden man programmerer opp mot, blir programmeringen betydelige mer ineffektiv og frustrerende. Kanskje kunne vi hatt som regel å ikke flette inn kode inn i master før javadoc er skrevet. Altså, skriv javadoc før man deler kode på github. Forsinkelser i sprint 5 fra gui-gruppa førte til at vi måtte avvente med å ferdigstille controller inntil alle hadde lagd ferdig grensesnittene. Sigurd brukte sprint 5 og 6 til å oppdatere deler av model, og controller etterhvert som vi ble bedre kjent med behovene i grensesnittene.

**Parprogrammering**

På oppfordring av student-assistent og senere vit.ass. ble vi oppfordret til praktisere disiplinen parprogrammering.