# Projektgennemførelse og metoder

## Projektstyring

Gruppen har valgt at tage elementer fra Scrum, således at projektet styres med en fornuftig, iterativ model.

Gruppen har kørt sprints på 1-3 uger. Herunder listes de elementer, gruppen har anvendt:

* *Task board*
  + Gruppen har brugt et digitalt Scrum board vha. en in-house web-applikation (*Redmine*), som skolen har stillet til rådighed. Her er det muligt at se hvilke opgaver, der skal laves under nuværende sprint, samt hvem de tilhører, og hvor meget tid der er sat af til opgaven.
* Stå-op-møde
  + Mandag, onsdag og fredag har gruppen holdt et møde på 5-10 minutter. Her kommer hvert gruppemedlem ind på 3 ting; Hvad har du lavet? Hvad skal du lave? Har du nogen forhindringer? Det har gavnet gruppen, da man hurtigt får feedback fra hvert medlem og hører om de er gået i stå, eller om der er nogen, der mangler tasks.
* Retrospekt-møde
  + Under retrospekt-mødet har gruppen afsluttet nuværende sprint og planlagt næste. Der ses på hvor meget gruppen nåede, og hvert gruppemedlem kommer med feedback til sidste sprint, samt eventuelle forbedringer til næste sprint. Herefter planlægges et nyt sprint med omhu, således at opgaverne og den normerede tid lyder realistisk ift. forhindringer.

Kort beskrivelse af hvert sprint:

|  |  |  |
| --- | --- | --- |
| **Iterationsnummer** | **Længde** | **Formål** |
| 1 | 4 dage | Kravspecifikation. |
| 2 | 2 uger | De første kernefunktionaliteter implementeres/påbegyndes. |
| 3 | 2 uger | De sidste kernefunktionaliteter (det første *use case*-udkast, gruppen lavede) laves færdigt. Nogle ting refaktoreres. |
| 4 | 3 uger | Udvidelser (nye use cases) påbegyndes. Unit tests sættes op. Databasen integreres med applikationen. |
| 5 | 2 dage | Et minisprint for at fastlægge hvad gruppen skal lave i næste sprint. Teknologiundersøgelse af MVVM og MVC pattern, samt vurdering af hvorvidt koden med fordel kan refaktoreres til et af disse. |
| 6 | 1 uge | Påbegyndelse af synkroniseringsprocessen og webapplikationen. |
| 7 | 1 uge | Sync laves færdig, og der arbejdes videre på webapplikationen. |
| 8 | 1 uge | Notifikations-*use casen* laves færdig, og Sync kobles på WPF-applikationen. *Controller* og *Model* arbejdes på i web-applikationen |
| 9 | 1 uge | Rapportskrivning. De enkelte *use cases* uddelegeres i web-applikationen. |

## UML

Til formidling af kravspecifikation og systemarkitektur, har projektgruppen valgt at anvende *Unified Modeling Language* (*UML*). Dette er valgt for at formidle systemet bedst muligt, da *UML* er industristandard, og desuden simpel og intuitiv at gå til for omverdenen.

## Tidsplan

Gruppen har fulgt en tidsplan (**bilag XX**), hvor der er forsøgt at sætte mere tid af til rapportskrivning end hvad vi gruppens medlemmer har afsat under tidligere semesterprojekter.

## Mødestruktur

Gruppe- og vejledermøder er blevet styret ved hjælp af en mødeindkaldelse (**bilag XX**), efterfulgt af et møde med dagsorden, dirigent og referent. De administrative roller er blevet fastlagt vha. en turnusordning (**bilag XX**), hvor de forskellige roller som referent og dirigent er skiftet fra møde til møde. Mødeindkalderen var fastlagt. Dette er gjort for at sikre at alle gruppemedlemmer, som måtte ønske det, har fået indblik i det administrative arbejde. For at sikre konsensus i dokumenter, er der udarbejdet skabeloner til mødeindkaldelser og referater. Møderne er blevet afholdt efter behov, med udgangspunkt i et møde ved et sprints begyndelse og afslutning. Referatet (**bilag XX**) fra forrige møde er blevet gennemgået og godkendt ved hvert møde.

## Continuous Integration

*Jenkins* er i det følgende beskrevet, på trods af at det ikke kom til at virke under dette projekt, da der er brugt mange timer på det, og der er kommet en masse erfaringer ud af det.

*Jenkins* har haft til formål at sikre en objektiv og løbende håndtering af de tests, der er skrevet i projektet. Det kobles sammen med et *git-repository*, og har derigennem adgang til den nyeste publicerede version af projektet. *Jenkins* kompilerer selv projektet og kører de *test-suites*, der er lavet ved hjælp af *NUnit*- og *NSubstitute*-frameworket. Det kan også sættes op til at udføre statisk analyse og *coverage*, hvis det ønskes.

I dette projekt er der blevet brugt lang tid på at få *Jenkins* til at fungere. Dette er dog ikke lykkedes efter hensigten. Det *git-repository*, der er brugt i projektet, kører på *GitHub*, hvilket viste sig at være et problem. Under opsætningen af *Jenkins*-projektet skal der specificeres, hvornår *Jenkins* skal bygge den *Visual Studio*-solution, der henvises til. Til omfanget af dette projekt, vil det være optimalt at bygge projektet hver gang der bliver lavet et nyt push. For at dette kan lade sig gøre med et *git-repository*, der kører på *GitHub*, skal der installeres et plugin på *Jenkins*-serveren. Det har ikke været muligt at finde en måde at gøre dette på, uden at prøve sig frem med opsætningen af et af de plugins, der findes til netop dette. Da projektgruppen ikke har direkte adgang til *Jenkins*-serveren, kunne det derfor ikke lade sig gøre.

Der blev senere udleveret et *git-repository* af skolen, som kører på *Gitswat*, og derfor ikke havde ovenstående problem. I forbindelse med udviklingen af projektets databasetilgang, blev der brugt et *modeling*-projekt i *Visual Studio*. Denne type projekt kræver at der er nogle bestemte filer i installationen af *Visual Studio*. Disse filer var ikke tilgængelige på *Jenkins*-serveren, og *Jenkins* kunne derfor ikke bygge systemets *solution*. Der blev forsøgt at omgå dette ved at lægge de manglende filer i en mappe i projektet, og manuelt ændre i en af projekternes *.csproject*-fil til at der skulle ledes på den nye lokation. Resultatet heraf blev at *Jenkins* kunne bygge projektet, men at det ikke kunne bygges lokalt i *Visual Studio*.

Da underviseren, der står for *Jenkins* lagde de manglende filer der, hvor *Jenkins* søgte efter den, efterspurgte *Jenkins* ikke længere de pågældende filer, men nu nogle andre. På dette tidspunkt var størstedelen af de ønskede test-suites skrevet og kørt, og grundlaget for at bruge *Jenkins* var ikke længere tilstrækkeligt til at det var tiden og indsatsen værd.

Det ovenstående beskriver processen med *Jenkins* i forhold til *WPF*-applikationen. Jenkins skulle efter planen også have været brugt til at teste web-applikationen. Her manglede *Jenkins*-serveren, ligesom ved *modeling*-projektet, en fil for at kunne bygge web-applikationer. Tidligere erfaringer viste, at det ikke kunne betale sig at forsøge at løse dette problem, uden selv at have adgang til serveren, og *Jenkins* blev derfor helt droppet.