# Specifikation og analyse

Da udviklingen af WPF-applikationen gik i gang, var det meget nyt. Gruppen lærte undervejs hvordan delene skulle programmeres, og der var ikke de store overvejelser omkring hvilken arkitektur programmet skulle bygges op efter. Dette har givet nogle problemer med at bevare overskueligheden, og med at holde koden let at vedligeholde. I slutningen af april blev der foretaget en analyse af *MVC*, *MVP* og *MVVM*, og om det kunne betale sig at refakturere koden, for at overholde disse arkitekturer. Gruppen fandt, at det ville have været en god idé at opbygge WPF-applikationen efter *MVVM*-arkitekturen, helt fra starten af. Hvis det var sket, ville mængden af kode i *code behind* have været reduceret væsentligt. Problemet med kode i *code behind* er, at det gør det vanskeligt at teste koden i form af *unit tests*. *MVC* og *MVP* blev også undersøgt. Grundlæggende har det vist sig at WPF-applikationen til dels følger MVC/MVP-arkitekturen. Der er et *View* i form af XAML, *presenter*/*controller* i form af *code behind*, *Business Logic Layer* (*BLL*)og *Data Access Layer* (*DAL*). Det ville have været en fordel at fjerne koden i *code behind*, og rykke det ud i et lag for sig selv. Fordelene ved at gøre dette kunne dog ikke gøre op for den tid det ville tage at skrive koden om. Gruppen har derfor besluttet at arbejde videre med koden som den var, men at lære af det og bruge den nye viden i forbindelse med udviklingen af *Web app*.

**Figur 1** MVC/MVP til 3-lags-arkitektur

En anden udfordring har været *mapping*’en mellem objekt-udgaven af en *entity* i koden, og dens relationelle database-udgave. Der er brugt lang tid på at få denne *mapping* til at være korrekt, og sørge for at udgaven i koden er den samme som den eksisterende i databasen. Løsningen på dette blev fundet gennem undervisningen i I4DAB. Grundet den fremskredne udvikling i WPF-applikationen blev det ligeledes besluttet at det ikke kunne betale sig at refaktorere koden, men at implementere det som en del af *Web app*.

I *Web app* er der også blevet indført et *GOF-pattern* ved navn *Facade*. *Facade*-mønsteret har i dette tilfælde til opgave at give et enkelt adgangspunkt til *DAL*. I forbindelse med *DAL* er der et *repository* for hver tabel i databasen, samt en databasekontekst. For at overskueliggøre dette er der lavet en facade, så man i koden, der skal bruge disse elementer, blot opretter en *Facade*, og ikke alle de forskellige elementer. Dette gør koden mere vedligeholdelsesvenlig, og overskueliggør koden, set fra et højere abstraktionsniveau.

For yderligere info om de trufne beslutninger, henvises til mødereferaterne (**Bilag XX**).