# Test af pristjek220

For at lave et godt produkt, og sikre at produktet lever op til product owner’s forventninger, bliver produktet udsat for nogle forskellige former for test. Dette er illustreret på Figur 1, hvor der kan ses de forskellige teststadier, som er unittest, integrationstest, systemtest, accept test. Gennem udviklingen af Pristjek220 er der brugt FDD, hvilket giver den fordel at man kan udvikle en feature til programmet, hvorefter der kan bestemmes om den skal sættes i produktion. Derefter kan der opstilles tests, således at den lever op til kravene. På den måde spares der en masse tid på de funktioner, som product owner beslutter sig for, han ikke vil have. TDD er modsat, der skrives test før man laver features, hvilket giver den fordel, at koden lever op til testenes kravene med det samme.



Figur 1: V-model for udvikling af produkt.[[1]](#footnote-1)

## Unit Test

Unit testning er det første teststadie, som programmet kommer igennem, da det tester de individuelle klasser isoleret fra resten af systemet. Det vil sige at når en klasse får tilføjet en funktion, kan den testes uafhængigt af, hvor langt udviklingen af resten af systemet er. Gennem unit testning sikres der, at hver klasse individuelt opfører sig på den forventede måde ved de givne inputs, og derved bliver programmet kvalitetssikret, så det lever op til forbrugernes forventninger. Til unittest af Pristjek220 er der blevet benyttet Nunit frameworket og Nsubstitute, til at kunne opstille de forskellige testscenarier, og lave unittestene automatiske. Nsubstitute er også et framework, som bruges til at substituere de forskellige klasser, som klassen, der testes, har relationer til. Derved isoleres klassen fra resten af systemet. Det at automatisere testene giver den fordel at programmøren ikke skal bruge tid på at teste manuelt flere gange. Derudover giver automatiske test muligheden for, at koden kan optimeres efterhånden. Denne optimering kan så blive kørt igennem de automatiserede tests, uden at skulle bruge for lang tid på at teste manuelt, at funktionaliteten stadig fungere, som den skal. Automatiske unittests er dog ikke den gyldne hammer, der løser alle problemer, da de automatiserede tests kun tester, det de er lavet til at teste, hvilket vil sige, at nogle ting, kan blive overset. Derudover er der nogle ting som unittests ikke kan teste, så som brugervenlighed og hvordan den enkelte klasse opfører sig, når den sættes sammen med resten af systemet.

## Integrations test

1. SWT PP – 07.1 Integration Test Pattern slide 2. [↑](#footnote-ref-1)