1 hvorfor integrationstest

Årsagen til vi laver integrationstest er for at sikre os at de testede moduler passer sammen. Fx skal legoklodserne passe sammen.
Dette gælder også for:

- Klasser
- Pakker
- Subsystemer

2 Før integrationstest

Før vi kan integrationsteste modulerne er der nogle ting, som skal være opfyldt.

- Unittest skal være laver på samtlige metoder i klasserne.
- Systemarkitekturen for systemet skal være kendt **(Dependency tree)**.
- Der ligger en klar plan for integrationstesten.

3 Big bang test

Big bang er en test, hvor det må bære eller briste.
Kryds finger og håb på det bedste.
Ulemperne er større end fordelene. Det virker oftes kun for små, ukomplexe systemer, samt de bugs, der findes ved denne test ligger sent i udviklingsforløbet.

4 Bottom-up test

Bottom-up testen starter i bunden af dependency træet og tester det nederste lag først. Der skrives en driver, som initerer testene, hvor hvert lag, indtil topniveauet er nået.

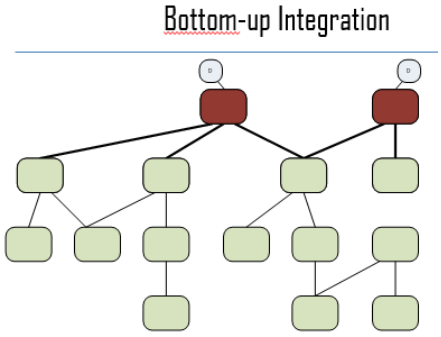

Figure 1: Bottom-up testing

Fordele:

- Ingen stubs
- Nemt at dække alle interfaces på alle niveauer.
- Kan afsløre design fejl tidligt i et forløb, da det er top niveau, som testes igennem.

Ulemper:

- Kræver mange drivers på flere niveauer
- Udskyder vigtige test af control interfaces **(Undlad at snake om, hvis muligt.)**

5 Top-down testing

Top-down testen er det modsatte af bottom-up. Her startes i toppen. Der skrives driver til topniveauet og alt under niveauet, man tester stubbes ud.

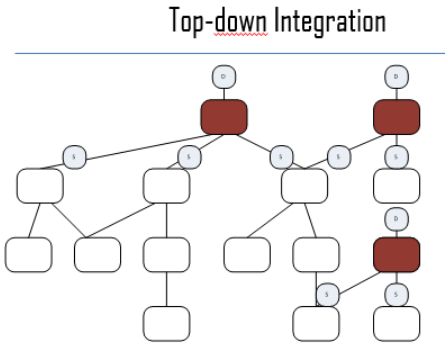

Figure 2: Top-down test

Fordele:

- Gør det nemmere med samtidig HW og SW udvikling.

- kan hurtig afgøre om kontrol komponenterne fungere, som de skal.

ulemper:

- Kræver mange stubs, men ok med NSubstitute test framework.
- kan være svært at teste

6 Collaboration testing

I collaboration test tager man en "gren" ad gangen i dependency træet.

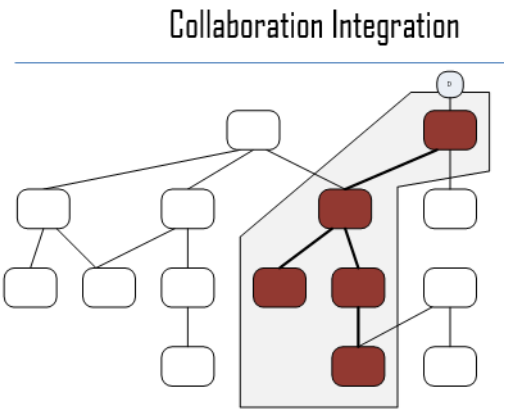

Figure 3: Collaboration integration

Fordele:

- Intuitivt, da det kan følge use-cases
- God til højniveau systemer
- Kan anvendes til iterativ udvikling

Ulemper:

- Alle deltager fra niveauerne kan blive testet seperat.

7 Sandwich testing

Sandwich test er, hvor man tester øverste lag, hvor man skriver drivere til klasserne, som skal testes og stub'er subklasserne. Dernæste tager man det nederste lag og skriver driver til inden man tester de midterste lag.

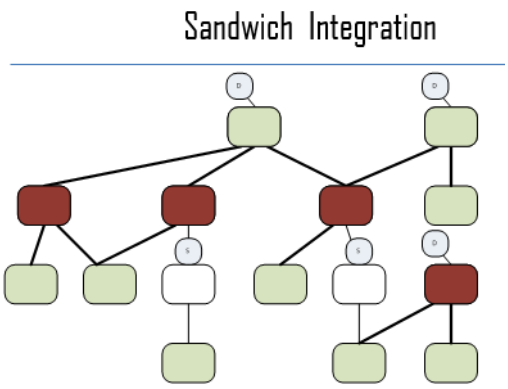

Figure 4: Sandwich integrationstest modellen

Fordele:

- God til store projekter med flere ressourcer
- Kombinerer fordele fra bottom-up og top-down.

Ulemper:

- Kan kræve mange drivers.
- Kan ikke anvendes på små systemer