Testkvalitet (coverage og BVA)

```
Se  p  Eksemplet herunder.
int foo (int x, int y)
{
    int z = 0;
    if ((x>0) && (y>0))
    {
        z = x;
    }
    return z;
}
```

1 Hvad forstås ved testkvalitet?

2 Coverage

Er en structual testing tecnique (glass- or white box, kræver kendskab til implementationen) bruges både i unit og intergationstest.
Der findes fire typer coverage:

2.1 Function coverage

Ser om alle funtioner har kørt

2.2 Statement/Line coverage

Er alle statements blevet ekskveret? (return, call, assignments)

$$100 * \frac{\text{Source lines reached by test}}{\text{Source lines in code}}\%$$

Figure 1: udregning af statement/line coverage

Ulempen ved denne er at vi får en whitebox test og at den er insensitiv overfor kontrolstrukturer (tjekker ikke om begge scenarier af en is-statemens er kørt)

2.3 Branch/Decision coverage

Er alle branches blevet kørt? Er alle Booleans blevet evlaueret for både true og false i fx while, switch, if? Denne inbefatter også "asynkrone branches" såsom interrupt handlers, exceptionhandlers og switch-statements

- Hvis foo bliver kaldt under eksekvering er function coverage opfyldt.
- bliver foo kald med parameterene 1,1 (foo(1,1)) vil alle linjer være kaldt. Statement coverage er opfyldt.
- kald foo(1,0) og foo(1,1) vil opfylde branch coverage
- kald foo(1,1), foo(1,0) og foo(0,0) For at opfylde branch/decision coverage. Det er her nødvendig at teste tre gange da de to første gør x condition true, mens den trejde er false. y er kun true i den første.

Functional testing kigger kun på hvad programmet opnår og ikke hvad der sker inde i programmet (blackbox)

3 BVA

BVA, eller Boundary Value Analysis er et begreb anvendt i tests til at bestemme med hvilke værdier der bør testes. Har vi eksemplet ovenfor, ønsker vi at sætte vores tæstværdier så tæt på skillelinjen som muligt.

Det vil sige at vi vil tæste foo :

- foo(0,0) Det vil give false for begge.
- foo(1,1) Hvilket pga. det er ints er det tætteste vi kommer på en boundary value som giver true.

3.1 EP

En ep (equivalence partition) er et område for hvilket hvis anvendt, det samme resultat kan forventes.

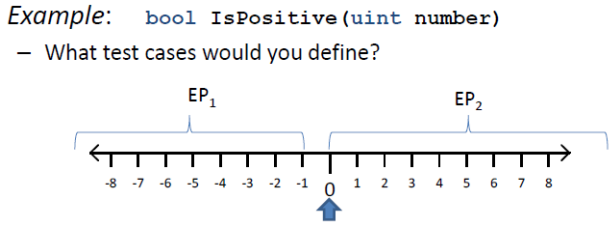

Figure 2: EP example

EP for vores foo eksempel vil således være true {1... infinit} & false{0... -infinite}