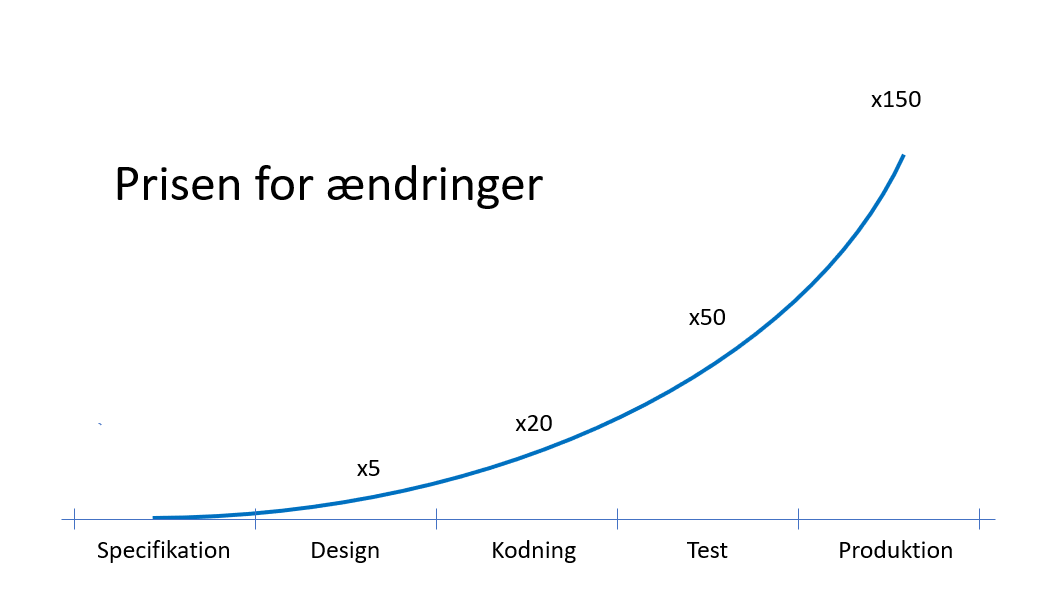
# Test og kvalitetssikring

Udarbejdet af Lars Skjærbæk, UCH

Når vi udvikler software, er det uundgåeligt at vi laver fejl. Finder vi først fejlene når produktet er kommet på markedet, er de meget dyre at rette. Vi risikerer ovenikøbet at få utilfredse kunder, som synes kvaliteten af vores produkt er dårligt. Derfor handler det om at finde fejlene og få dem rettet.



Finder vi en fejl i specifikationen, mens vi stadig arbejder med den, koster det ikke ret meget at rette fejlen. Finder vi først denne fejl under testen, skal vi tilbage og rette specifikationen. Så skal vi måske også rette designet og koden, og så skal vi i øvrigt teste igen. Derfor er fejlen meget dyrere at rette. Finder vi først fejlen, når produktet er kommet på markedet, bliver det rigtig dyrt. Så husk:

Fejl skal findes så tidligt i processen som muligt!

## Reviews

En stor del af softwareudviklingen består i at lave dokumentation. Også her vil vi gerne finde fejlene så tidligt som muligt, og det kan vi gøre ved at holde reviews. Vi skelner typisk mellem to typer af reviews:

* Peer review (walk through – inspections)  
  Formålet er at finde fejl i dokumentet og få dem rettet, så kvaliteten i dokumentet øges. (formativ test)
* Godkendelses review  
  Formålet er at få andres (ledelsens eller kundens) accept af et dokument

Et review er ikke bare en hurtig uformel gennemgang, men en struktureret proces som omfatter en formel rollefordeling (review leder, referent, forfatter), en struktureret gennemgang af dokumentet, afsnit for afsnit, samt dokumentation af problemerne. Man skal i den sammenhæng også lære noget om at give konstruktiv feedback.

## Debugging

Når vi skriver kode, tester vi jævnligt af om den fungerer. Hvis programmet ikke virker, finder vi fejlen, og retter den. Målet med denne test er, at få tingene til at fungere. De fejl vi finder, er typisk fejl i den måde vi har skrevet koden på. Det kan være variable eller løkker, der ikke er defineret rigtigt, eller fejl i data.

En god editor kan hjælpe med at påpege syntaks fejl, som f.eks. at vi har glemt et ”;” eller en ”}”. Den kan også indikere, hvis variable eller funktioner er defineret forkert. Andre fejl skal vi selv finde frem til, og det kan til tider være svært. Som hjælp til dette kan man afvikle programmet trinvist ved at indsætte break points, eller man kan udskrive debugging informationer (eks. indholdet af en variabel) som hjælpe en med at spore fejlen.

## Accept test

En accept test bruges til at verificere, at systemet overholder de specificerede krav. Denne test er en ”black box” test. Det vil sige, at man ikke behøver at kende noget til, hvorledes softwaren er opbygget, for at kunne specificere testen. Man skal bare kende til kravene. I en accept test skal man sikre, at man **systematisk** **kommer igennem alle programmets funktioner**. Derfor skal testen struktureres og dokumenteres rigtigt. Dette gøres gennem brug af test cases (se nedenfor – feltet til højre udfyldes under udførelse af testen med et OK eller en reference til en problemrapport)

Allerede når testen specificeres finder vi uklarheder i specifikationen (i eksemplet her: hvad skal der ske, når man indtaster et negativt tal). Derfor bør specifikationen af testen gennemføres som en del af kravs-fasen, hvor der er nemt at rette i specifikationen.

Udførelsen af testen kan først gennemføres, når produktet er færdigt, og først når testen er gennemført uden fejl, kan produktet accepteres af kunden (heraf navnet accept test). Laves der en ny release, bør hele accepttesten gennemføres igen.

Til større projekter defineres der mange testcases - en til hvert funktionalitetsområde.

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Input** | **Forventet resultat** | **Resultat af testen Dato: 1/12-19** |
| 1 | Start Kasseapp | Kasseapp vises. Det skal tydeligt fremgå at der ikke er indtastet data | OK |
| 2 | Scan vare (Indtast 12.75 og tryk ”Tilføj vare”) | Man skal nu se at sum er 12.75 | OK |
| 3 | Indtast ugyldigt input (space) og tryk ”Tilføj vare” | Input ignoreres | Fejl #1 |
| 4 | Indtast negativt tal (-456.78) | ? |  |
| 5 | Tryk ”Betal med kort” | Man skal kunne se summen, og at der skal betales med kort | OK |
| 6 |  |  |  |

### Problem rapportering

Fejl og mangler dokumenteres i et problemrapportsystem. Finder man under testen (eller ved anden brug af produktet) et problem, skal det indrapporteres i systemet. Det gør man ved at udfylde den øverste del af rapporten. Senere kan udviklerne analysere problemet, og finde frem til årsag og løsning. Til sidst registrerer man, når fejlen er rettet. Har man mange problemer, kan man vælge at give dem en kategori, således man kan prioritere indsatsen for fejlrettelserne. Således kommer fejlrapporten f.eks. til at se således ud:

Problem rapport nr.:

Problem vedrørende:

Problem beskrivelse:

Årsag og løsning:

Kategori:

Problem rettet:

1

Indtastning af ugyldige input

Når man ikke indtaster noget, eller hvis man indtaster et ugyldigt input (bogstav eller tegn) og trykker ”Tilføj vare” går app’en i baglås.

Kun tal kan adderes til sum. Der skal derfor testes for, om input er et gyldigt tal, inden det lægges til sum.

A

1/12-19 af LSK

Problem fundet:

1/12-19 af LSK

I forbindelse med udvikling af større systemer har man database baserede løsninger til at holde styr på problemerne. Desuden kan man ud fra metrikkerne i systemet udtale sig om kvalitet og fremdrift i udviklingsprocessen. Når man releaser en ny software, kan man også nemt dokumentere, hvilke problemer der er løst siden sidste release, og hvilke problemer der endnu ikke er løst.

## Brugertest

Brugertesten (bruger evalueringer) er en vigtig del af den agile udviklingsmetode. Når man har lavet en prototype, kan vi bruge denne test til at inddrage brugeren, og sikre at vi får en god vel dokumenteret feedback.

En brugertest er en formativ test, hvor man sætter rigtige brugere (eller bruger repræsentanter) til at afprøve og evaluere en løsning. En god måde at lave brugertest på, er at følge metoden for ”Usability test”, da denne er vel struktureret, dokumenteret og afprøvet. Metoden omfatter følgende step:

* Lav en plan for hvad, hvordan og hvornår du vil teste
* Inddrag rigtige brugere i testen
* Lær selv produktet at kende
* Definer konkrete opgaver, der skal løses i testen
* Gennemfør interview før og efter testen
* Optag testen på video - få brugeren til at tænke højt under testen
* Observer hvorledes opgaverne løses, og noter problemerne
* Udarbejd forslag til løsninger – nye prototyper

Til brugertesten er der udviklet et selvstændigt forløb som du kan finde her (Link: EMU Brugertest forløbsbeskrivelse.docx)