

## World Class Test Management

Af Poul Staal Vinje

Certified Scrum Master

Certified Scrum Practitioner

[poul.vinje@gmail.com](mailto:poul.vinje@gmail.com)

[www.agile-metoder.dk](http://www.agile-metoder.dk)

[www.softwaretest.dk](http://www.softwaretest.dk)

### Artikel 2 om WCTM: Ingen testgæld

World Class Test Management.....	1
Artikel 2 om WCTM: Ingen testgæld .....	1
1. Fra den første artikel .....	2
2. Midler .....	2
3. Kort udviklingsperiode.....	3
4. Testbarhed .....	3
5. Testdrevet udvikling .....	4
6. Definition of Done.....	4
7. Strategi .....	4
8. Bundlinie.....	5

## 1. Fra den første artikel

Denne artikel uddyber det første punkt i definitionen på WorldClassTestManagement.

Citat fra den første artikel:

Ingen testgæld på noget tidspunkt i projektet. Testgæld er den mængde af analyse, design eller kode der er udviklet, men ikke er testet. Man skylder testen så at sige. Jo mindre testgæld, jo mere fremragende testledelse er der udvist. Hvis der ikke skyldes test undervejs overhovedet, er det verdensklasse.

Problemets størrelse afhænger af to ting:

- Mængden vi skylder, fx mængden af kode der mangler at blive testet.
- Periodens længde vi skylder testen i.

Hvis det er krav og ikke kode der mangler test? Og kravene først verificeres når vi releaser? Så er der en lang periode hvor vi øger testgælden, og alt efter hvor mange krav der er i projektet, er det også en stor mængde vi skylder.

Citat slut.

Testen skal altså holde trit med udviklingen. Lige fra kravspecifikation til kodning og integration. Være på omgangshøjde. Testdrevet udvikling er den optimale løsning. Men som minimum skal testen ikke sættes helt af. Og ikke komme sidst i mål.

Test omfatter:

- Funktionel test.
- Testen af non-funktionelle egenskaber. Brug for eksempel ISO 9126 som grundlag. Nogen hader den, andre elsker den, vælg eventuelt en anden standard/model.
- Testen af øvrige artefakter, for eksempel risikoanalyse og statusrapporter.

## 2. Midler

Midlerne til 0 testgæld er:

Kort udviklingsperiode, for at begrænse den totale mængde af test der kan skyldes, plus at det er mere overskueligt at holde nullet.

Øge testbarheden af krav, design og kode, for at gøre testen lettere.

Testdrevet udvikling, for at holde testen i spids.

Forudsætningerne er desuden klare "Definition of Done", og en velegnet udviklingsstrategi.

De følgende afsnit uddyber midlerne.

### 3. Kort udviklingsperiode

Kort tid fra krav til kode. I agil udvikling, og i iterativ udvikling i det hele taget, er der maksimalt 4 uger fra krav til kode. Større systemer leveres i mindre dele, og et system der er estimeret til en varighed på 18 måneder, leveres altså i 18 slices. Der hver for sig er testet, og hver for sig er udviklet med en løbende testgæld på 0.

### 4. Testbarhed

Ved at øge testbarheden af det der skal testes, bliver testen fundamentalt nemmere og billigere at holde på omgangshøjde.

De tre traditionelle artefakter vi ønsker at gøre mere testbare er krav, design og kode.

Lad os først se på de tre artefakters testbarhed, i et traditionelt projektførløb:

Krav forsynes med eksempler. Når et krav fremsættes, skal det ledsages af flest mulige konkrete eksempler. Hvert eksempel vil mindske testgælden. Eksempler på kravene kan udarbejdes på ethvert tidspunkt. De kan tilføjes løbende.

Et testbart design kan derimod kun opnås hvis designerne tænker test ind i selve designfasen. Testbarhed kan ikke klistres på. Testbarhed er et designkrav. Designerne skal stille sig selv spørgsmålet: Har jeg gjort hvad jeg kan for at lette testernes arbejde. Der er masser af checklister til godt design. Hvis designere bruger spørgsmålene i checklisten løbende, i stedet for på et review af det færdige design, kan testgælden holdes i kort snor.

Koden gøres testbar ved at udvikle simple moduler med få veje i hvert. Grunden er at testdækning kun beregnes pr. modul, og ikke i forhold til det samlede veje igennem et system med alle moduler set under ét.

Lad os dernæst se på testbarhed i agile projekter:

Kravene dokumenteres i Userstories. Testbarheden øges ved at formulere eksempler. Analogt til traditionelt. Men Userstories er simplere end for eksempel Use cases, og bruger simplere templates. Arbejdet med at gøre dem testbare er tilsvarende enklere. Desuden gennemgår produktejeren hver Userstory for teamet på SprintPlanningWorkshop, og det giver teamet mulighed for at stille spørgsmål og få uddybet eksemplerne.

Designet laves under vejs. Der er ingen designfase. Det er altså "Incremental design". Der er født testbart. Designet gror langsomt frem, og bliver testet løbende.

Koden udvikles testdrevet. Der er pr. definition 100% dækning, fordi koden skrives for at leve op til testcases.

## 5. Testdrevet udvikling

Enten "Classic TDD" ved at skrive alle testcases til en feature, før koden der implementerer featuren skrives.

Eller "Real TDD" ved at skrive få testcases ad gangen, skrive koden der lever op til dem, og fortsætte indtil en feature er færdig. Med Refactoring af koden under vejs.

I begge tilfælde er testen flyttet, således at vi ikke bare har indført mini-vandfald.

TDD udvides i reglen med automatisering.

Testdrevet udvikling bruges ofte sammen med andre Agile Engineering Practices, for eksempel PairProgramming, Continuous Integration, Whole Team. Incremental design, Refactoring, Collective Ownership, On-site Customer, Small Releases.

## 6. Definition of Done

Definition of Done, er en klar beskrivelse af hvornår et artefakt er klar til at gå videre i udviklingsforløbet. Klare og fælles kriterier er en effektiv vej til 0 testgæld. Uklare eller individuelle kriterier er vejen til en ukendt og overraskende testgæld.

Ikke mindst for kode er det vigtigt. Er koden færdig når den er skrevet? Eller når den er kompileringsfejlfri? Eller når den er unittestet? Eller når den er unittestet med en kendt dækning af koden? Eller når den er unittestet med en kendt og bestemt dækning af koden? Eller er den først færdig når unittesten er automatiseret, og koden kan regressions-testes automatiseret.

## 7. Strategi

Udviklingsstrategien spiller ind.

I en vandfaldsmodel kan testgælden blive høj, hvis udviklingen er nået langt, før testen starter.

I en V-model når testgælden stadig at blive høj. Kun reduceret med den test der implicit er resultatet af at testen planlægges tidligt.

I en W-model, med reviews fra start af projektet, holdes testgælden tilsvarende lavere end i en V-model. Men en W-model er stadig re-aktiv.

I en Testdrevet udvikling kan testen for alvor udføres løbende, og uden en løbende gældsætning. Først med denne type pro-aktive teststrategier kan der opnås verdensklasse.

## **8. Bundlinie**

I WCTM er testen en del af udviklingen. Test ER udvikling. Det er to sider af samme sag.

Kvaliteten skal være synlig og målelig på ethvert givet tidspunkt. Når gælden er nul (0) kan kvaliteten inspiceres, vurderes, måles, benchmarkes, kommenteres, diskuteres, godkendes/afvises.

Kan man ikke holde nullet, så må man holde gælden lavest mulig, og arbejde på at blive stadig bedre til ikke at skylde. Og løbende betale af på den der opstår.