

# Eksamenskurs ISTQB Foundation 31. august 2017

*Dette må du kunne for å stå på eksamen!*

# Kursholder

---

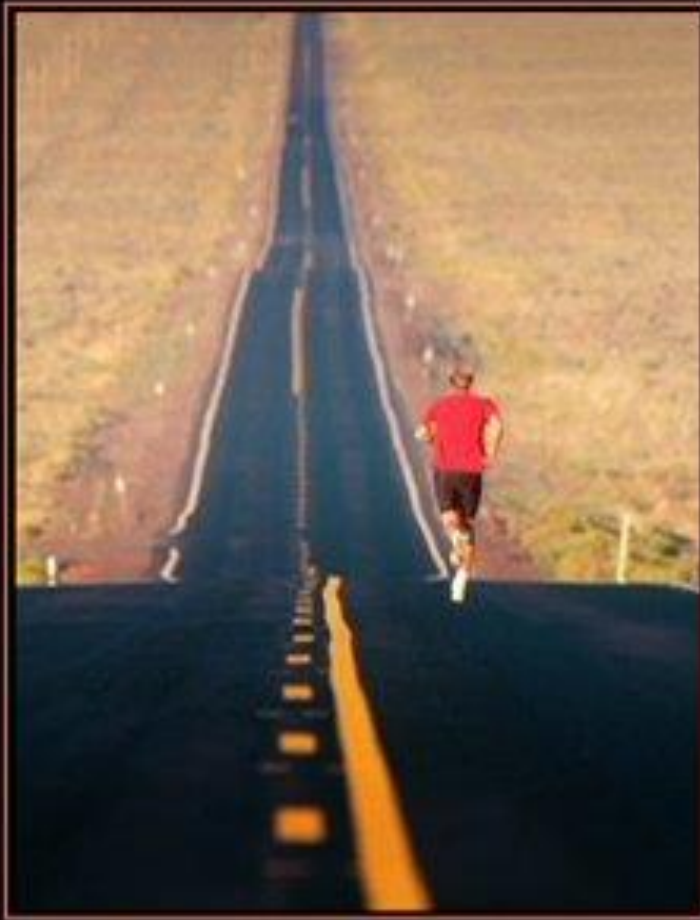


## **Marianne Øberg**

- Avdelingsleder for Quality Transformation

## **Audun Brekke**

- Manager BT



# QUALITY

THE RACE FOR QUALITY HAS NO FINISH LINE-  
SO TECHNICALLY IT'S MORE LIKE A DEATH MARCH.

[www.despair.com](http://www.despair.com)



# ISTQB Certified Foundation Level (CTFL)

- Engelsk – 60 min (75 min)
- K1 Remember - 50%
- K2 Understand - 30%
- K3 Apply - 10%\*
- K4 Analyze - 10%\*
- 40 multiple choice-spørsmål, kun ett svar pr spørsmål er korrekt
- Må ha 65% rett (26 spørsmål)
- Feil eller ingen svar gir 0 poeng

| Exam       | Number of questions per Chapter |       |       |       |       |       |       |
|------------|---------------------------------|-------|-------|-------|-------|-------|-------|
|            | Ch. 1                           | Ch. 2 | Ch. 3 | Ch. 4 | Ch. 5 | Ch. 6 | Total |
| Foundation | 7                               | 6     | 3     | 12    | 8     | 4     | 40    |

I følge ISTQB tar det 14,5 timer å pugge pensum.



# Endringer på eksamen og pensum

---

The goal is to make this version easier to read, understand, learn, and translate, focusing on increasing practical usefulness and the balance between knowledge and skills.

- This major release has made the following changes:
  - Fewer K1 Learning Objectives (LO) in general, – 15 LO in 2018 compared with 27 LO in 2011.
  - Less focus on chapter 5 Test Management, – 15 LO in 2018 compared with 24 LO in 2011.
  - More emphasis on review, a K3 LO has been added to chapter 3. – Static Analysis by Tools section is removed, and will be covered in other syllabi.
  - More emphasis on test techniques in chapter 4. – Section 4.1 of 2011 moved and merged with section 1.4 of chapter 1.
  - Agile is mentioned in the content of the syllabus. – But not included in the wording of any LO.
  - White-box techniques are downgraded. – K4 and K3 removed – they will be covered in other syllabi.



# AGENDA

08.30 – 09.30: Kapittel 1

09.45 – 11.00: Kapittel 2

11.00 – 11.45: Lunsj

11.45 – 12.30: Kapittel 3

12.45 – 14.15: Kapittel 4 – gruppeoppgaver

14.30 – 15.30: Kapittel 5

15.40 – 16.00: Kapittel 6



# Kapittel 1: Grunnlag for testingen

- Hvorfor er test nødvendig?
- Hva er testing
- Syv testprinsipper
- Den grunnleggende testprosessen
- Testingens psykologi





# WHY?

## ALTINN-RAPPORT

### Altinn avsløres i hemmelig rapport

Store mangler i den nye Altinn-plattformen. Det Norske Veritas setter spørsmålstegn ved kompetansen til dem som forvalter systemet.

Av [Eспен Zachariassen \(@ezach\)](#)

Publisert: 19. mars 2012 kl. 19:04 · Oppdatert: 26. mai 2012 kl. 10:58

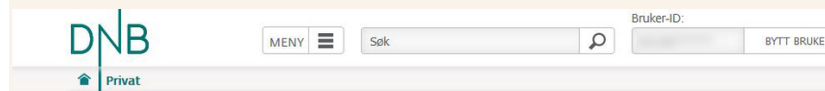


Aller Media ble rammet av et datainnbrudd. (Ill.: 199pema/Vikipedia)

**Persondata på avveie etter datainnbrudd i Aller Media**

## DNB vet ikke hvorfor kunder ikke kommer inn i nettbanken

Landets største bank sliter med å forstå egen nettbank.



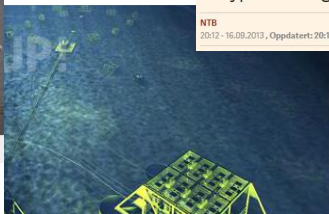
### PROFILERTE STILLINGER



### Norwegian Dreamline

Flyelskapet Norwegian av typen Boeing 787 Dreamliner

NTB  
20.12 - 16.09.2013 · Oppdatert: 20.13 - 16.09.2013





# Fra behov til endelig produkt



How the customer explained it



How the project leader understood it



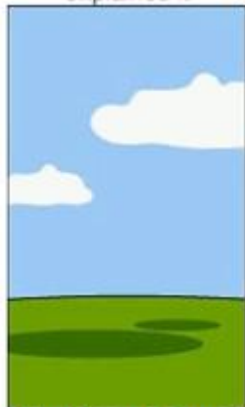
How the engineer designed it



How the programmer wrote it



How the sales executive described it



How the project was documented



What operations installed



How the customer was billed



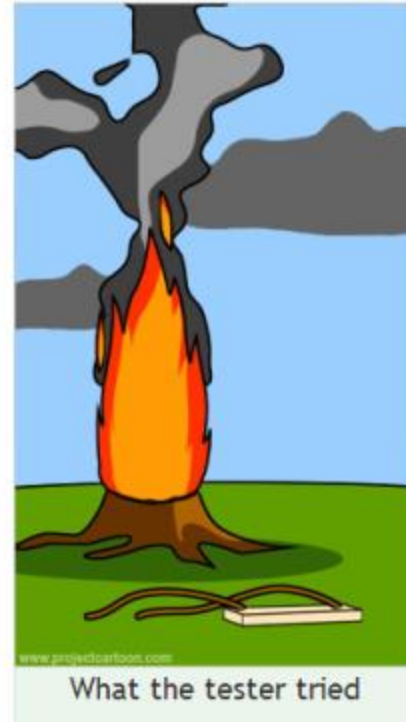
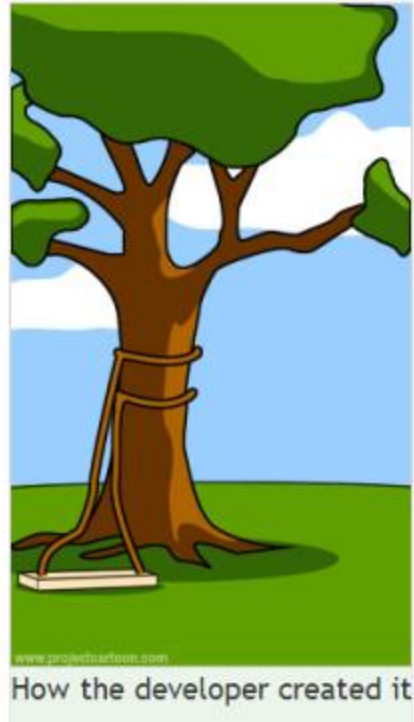
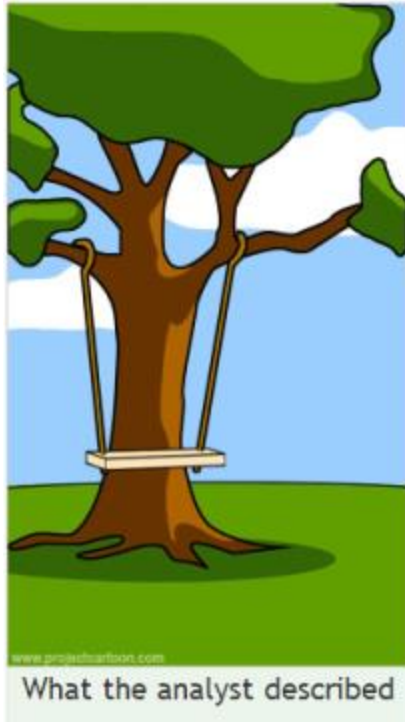
How the helpdesk supported it



What the customer really needed

## Fra behov til endelig produkt

---



# Forventningsstyring hos kunden

---



# Hva er viktigst for ditt produkt?




## Trege nettsteder dreper netthandel

Undersøkelse viser at nettshoppere er veldig utålmodige, orker ikke vente mer enn to sekunder på innlasting av nettsider.

[Hovedside](#) / [Om Statens vegvesen](#) / [Presse](#) / [Nyheter](#)

## Bruk Vegvesenets selvbetjeningsløsninger

 SKRIV UT

Har du mista vognkortet, skal selge bilen eller søke om førerkort? Nå kan du fikse det selv på nett. [23.06.2015]

I løpet av det siste året har Statens vegvesen innført flere selvbetjeningsløsninger for sine brukere. Flere av de oppgavene du før måtte til en trafikkstasjon for å få løst, kan du nå fikse selv på nettet.

### Logg deg inn på "Din side"

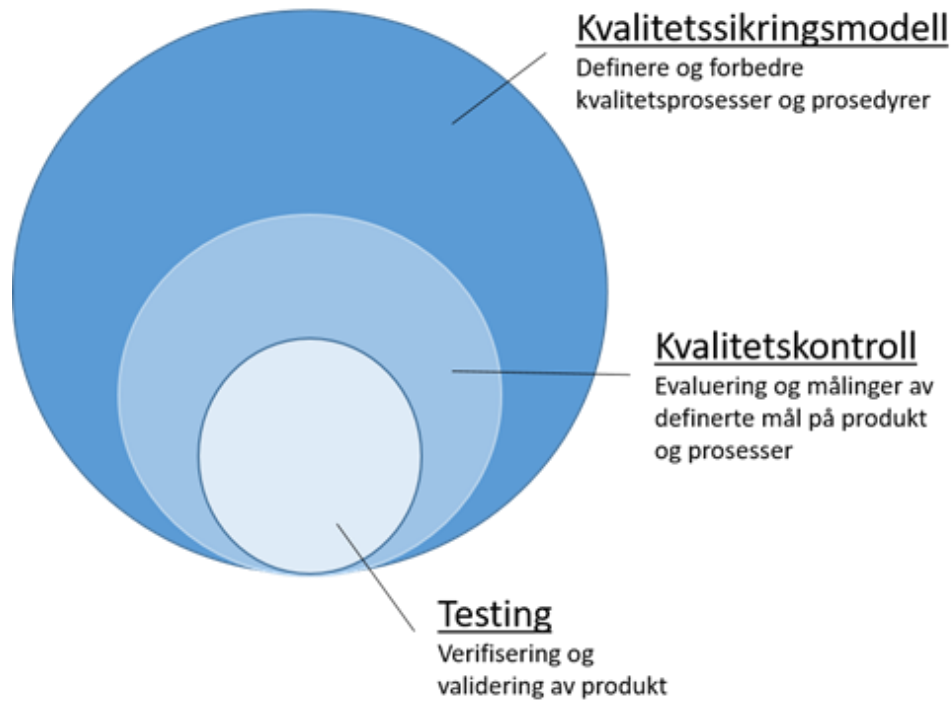
Du logger deg inn på «Din side» på samme måte som du leverer selvangivelsen eller logger deg inn i nettbanken.

- Dette gjør livet enklere for publikum, samtidig som det fører til at vi i Vegvesenet får brukt skattepengene vi forvalter mer effektivt, sier Kjetil Wigdel i Statens vegvesen.





# QA, QC og Test



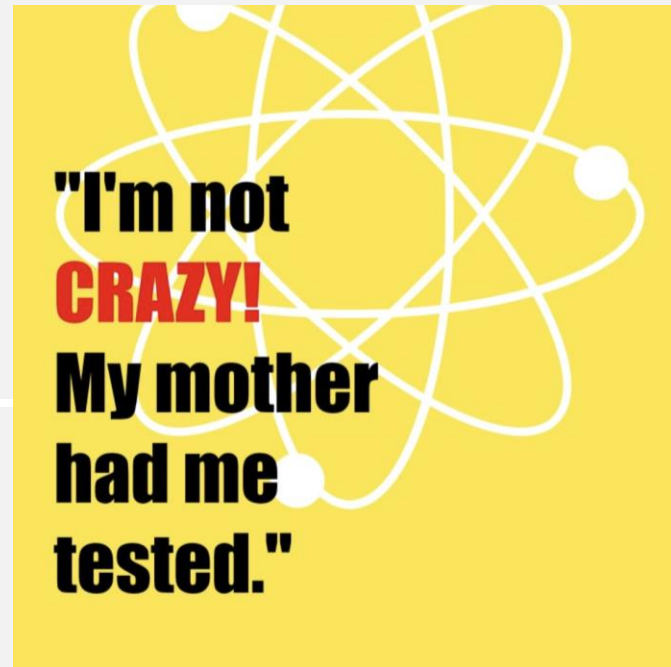
# Testformål

---

- Finne feil
- Forhindre feil
- Kontrollere produktkvalitet er i henhold til krav og spesifikasjoner
- Få tillit til kvalitetsnivået
- Gi informasjon for å skaffe beslutningsgrunnlag
- Risikostyring

**Verifisere** – Har programvaren blitt laget *riktig*?

**Validere** – Har *riktig* programvare blitt laget?



# Hvor mye testing er nok?

- Det kommer an på risikoen i systemet
  - Prosjekt og produkt risiko
- Begrensninger i prosjektet
  - Tid, kost og kvalitet
- Kunne gi rett informasjon til interessenter
  - Slik at de kan ta en god beslutning
- Avhengig av konteksten



# Test er avhengig av konteksten (nytt 2018)

Contextual factors that influence the test process for an organization, include, but are not limited to:

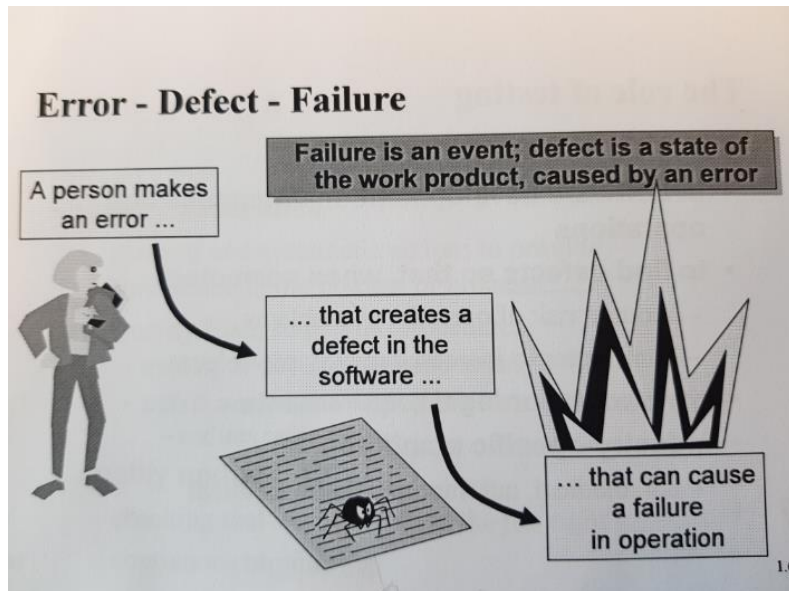
- Software development lifecycle model and project methodologies being used
- Test levels and test types being considered
- Product and project risks
- Business domain
- Operational constraints, including but not limited to:
  - Budgets and resources
  - Timescales o Complexity
  - Contractual and regulatory requirements
- Organizational policies and practices
- Required internal and external standards





# Begrepet «feil»

“A human being can make an **error**, which produces a **defect** in the program code, or in a document. If a defect in code is executed, the system may fail to do what it should do (or do something it shouldn't), causing a **failure**. Defects in software, systems or documents may result in failures, but not all defects do so.”



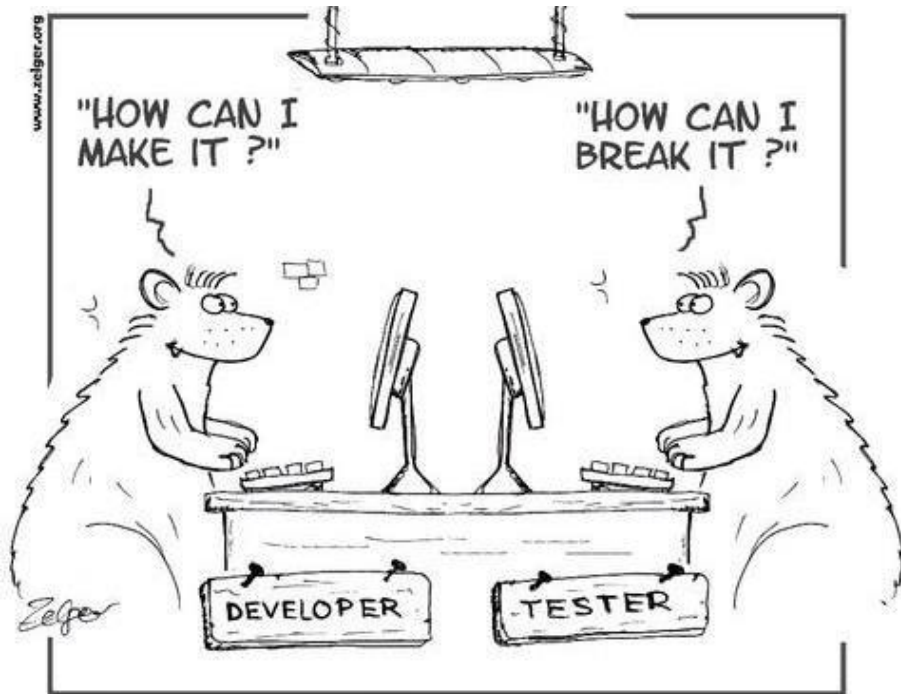
The **root causes** of defects are the earliest actions or conditions that contributed to creating the defects.

# Testingens psykologi

Ulik innfallsvinkel



Ulik oppfattelse



## When Tester reopens a Bug



Software Tester



Software Developer

Courtesy - Yogesh Khairnar

QA-QC Arena

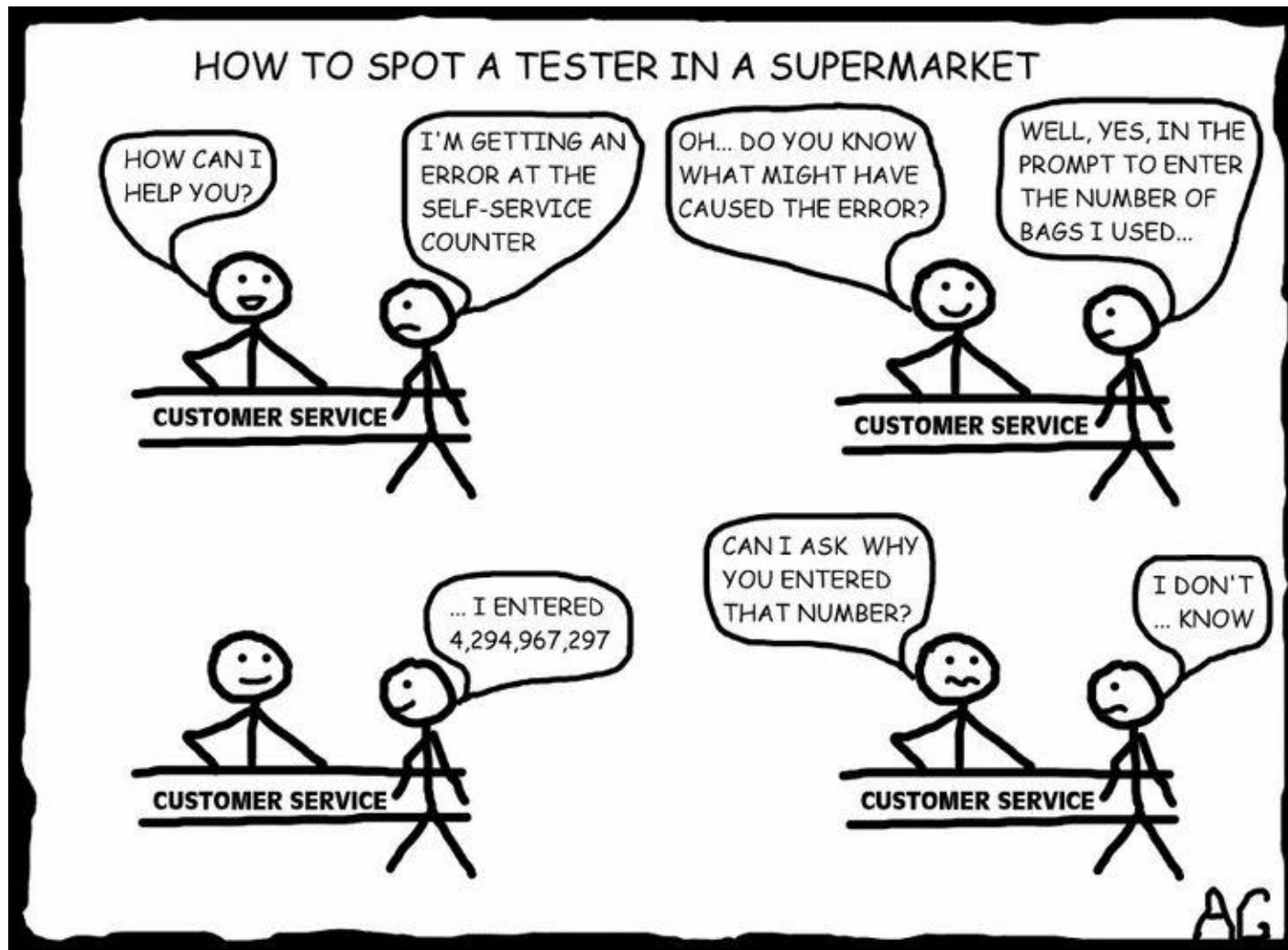
<http://qa-qcarena.blogspot.in>



# Testingens psykologi - respekter hverandres perspektiver



Men.. tenk som en tester – litt hver dag



# De 7 testprinsippene

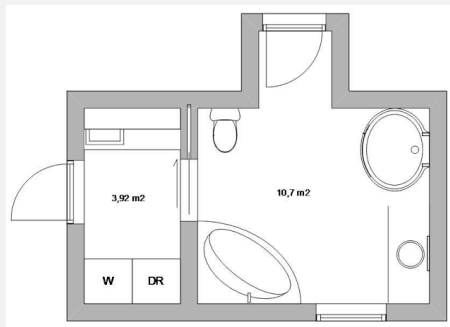
## 1. Avdekker feil, beviser ikke feilfri programvare



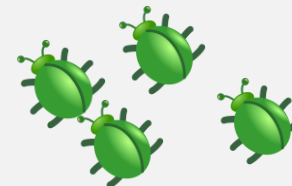
## 2. Komplettest umulig



## 3. Tidlig testing

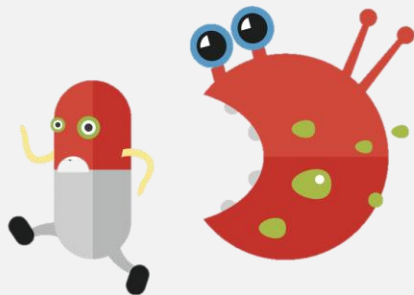


## 4. Klyngedannelse av feil



---

- 5 .«Pesticid-paradoks»



## 6. Avhengig av konteksten



## 7. Feiltakelse vedr. «fravær av feil»

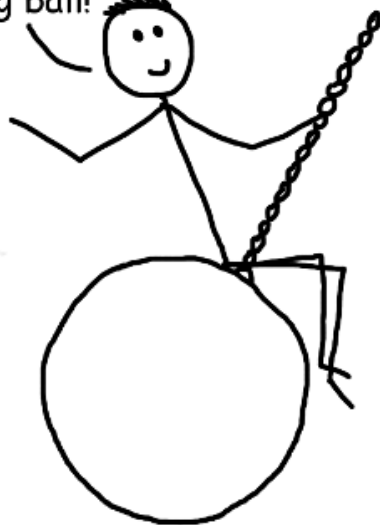
Hjelper ikke med feilfritt produkt hvis ikke det er det kunden ønsker



## Testing er ikke bare å kjøre tester

If testers break software...  
Then Miley Cyrus is the best tester ever...

I'm a wrecking ball!



# Den grunnleggende testprosessen (K2)

## The fundamental test process

---

- Den grunnleggende testprosessen består av følgende 5 hovedaktiviteter:
  1. Testplanlegging og teststyring - *test planning and control*
  2. Analyse og design - *test analysis and design*
  3. Implementering og gjennomføring - *test implementation and execution*
  4. Evaluering og rapportering - *evaluating exit criteria and reporting*
  5. Avslutningsaktiviteter - *test closure activities*

**Eksamen:**

**Husk de fem grunnleggende testaktivitetene og tilhørende oppgaver  
(oppdatert til K2 i 2018)**





# Den grunnleggende testprosessen

- Testplanlegging og teststyring – *test planning and control*
  - Spesifikasjon av aktiviteter for å oppfylle formål og oppdrag. Implementasjon av test policy/strategi, bestemme scope, risiko, og mål for testen (test levels, entry og exit criteria). Finne ressurser, lage tidsplan..
  - Teststyring: Kontinuerlig sammenligning av virkelig fremdrift i forhold til plan
- Analyse og design – *test analysis and design*
  - Granskning av testgrunnlaget (krav, risikonivå, rapporter fra risikoanalyse arkitektur, design, grensesnittspesifikasjoner)
  - Høynivå testtilfelle
  - Miljø, infrastruktur, verktøy, testdata
  - Sporbarhet
- Implementering og gjennomføring – *test implementation and execution*
  - Spesifisere testtilfeller inkludert testdata
  - Lager rekkefølge på prosedyrer og scripts, setter opp miljø, kjører tester.
  - Logging av resultat, sammenligne resultat, rapportering av avvik
  - Gjentakelse av nødvendige aktiviteter, f.eks retest



# Den grunnleggende testprosessen

- Evaluering og rapportering – *evaluating exit criteria and reporting*
  - Kontroll av testlogg mot exit criteria definert i testplan
  - Er testingen ferdig?
  - Utarbeidelse av rapport
- Avslutningsaktiviteter – *test closure activities*
  - Gjøres ved prosjektmilepæler osv.
  - Lukking av hendelsesrapporter/opprettelse endringsrapporter
  - Handover av testware til mottagelsesorganisasjon
  - Dokumentasjon på godkjennelse/underkjennelse



# Oppgave: match stegene i testprosessen med aktivitetene

1. Test planning and control
  2. Test analysis and design
  3. Test implementation and execution
  4. Evaluating exit criteria and reporting
  5. Test closure
- 
- a) *Check test logs against planned criteria, assess whether more tests are needed*
  - b) *Verify test mission and objectives, specify test activities, compare progress to plan*
  - c) *Review test basis, transform requirements into test conditions and test cases*
  - d) *Transform test cases into test procedures, create testware, set up environment, run tests*
  - e) *Check deliverables, archive testware, analyse lessons learned*



## Oppgave: match stegene i testprosessen med aktivitetene

1. Test planning and control
  2. Test analysis and design
  3. Test implementation and execution
  4. Evaluating exit criteria and reporting
  5. Test closure
- 
- a) *Assess whether more tests are needed*
  - b) *Decide what metrics to monitor*
  - c) *Create test inputs and expected results*
  - d) *Handover of testware to maintenance*
  - e) *Set up the test environment, compare actual with expected results*



## Testplanlegging og teststyring (planning and control)

---

- Testplanlegging er den aktiviteten som består av å definere testens formål og spesifisere testaktivitetene for å oppfylle formål og oppdrag.
- Teststyring er den kontinuerlige sammenligningen av virkelig fremdrift ift. planen, samt rapportering av status og avvik ift. planen. Det innebærer også å gjennomføre tiltak for å oppfylle prosjektets oppdrag og formål. For å kunne styre testingen, bør hele testaktivitetene overvåkes nøye gjennom hele prosjektet. Testplanlegging tar med i betraktning tilbakemelding fra overvåkings- og styringsaktiviteter.
- Testplanleggings- og styringsaktiviteter er nærmere definert i kapittel 5 i ISTQB- pensum.



## Testplanlegging og teststyring (planning and control)

---

- Testplanlegging er den aktiviteten som består av å definere testens formål og spesifisere testaktivitetene for å oppfylle formål og oppdrag.
- Teststyring er den kontinuerlige sammenligningen av virkelig fremdrift ift. planen, samt rapportering av status og avvik ift. planen. Det innebærer også å gjennomføre tiltak for å oppfylle prosjektets oppdrag og formål. For å kunne styre testingen, bør hele testaktivitetene overvåkes nøye gjennom hele prosjektet. Testplanlegging tar med i betraktning tilbakemelding fra overvåkings- og styringsaktiviteter.
- Testplanleggings- og styringsaktiviteter er nærmere definert i kapittel 5 i ISTQB- pensum.



## Testplanlegging og teststyring (planning and control)

---

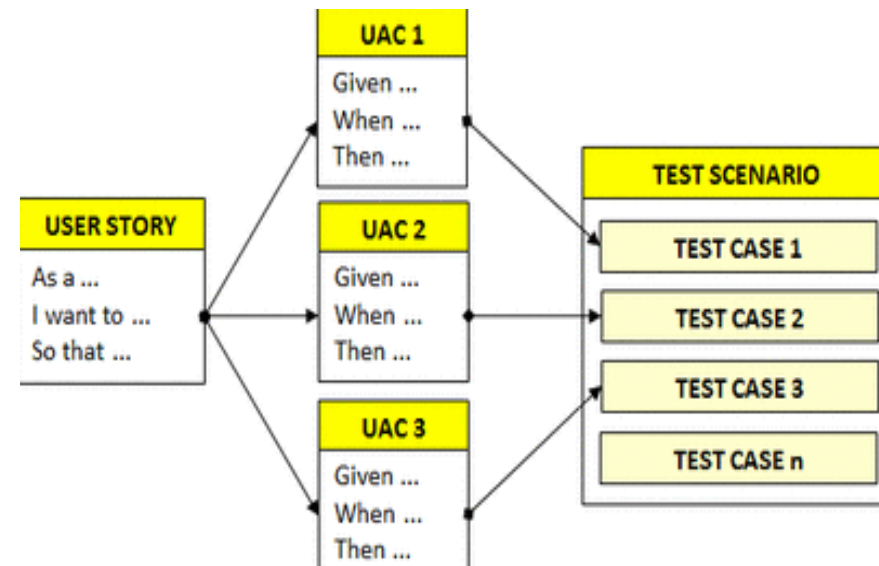
- Testplanlegging er den aktiviteten som består av å definere testens formål og spesifisere testaktivitetene for å oppfylle formål og oppdrag.
- Teststyring er den kontinuerlige sammenligningen av virkelig fremdrift ift. planen, samt rapportering av status og avvik ift. planen. Det innebærer også å gjennomføre tiltak for å oppfylle prosjektets oppdrag og formål. For å kunne styre testingen, bør hele testaktivitetene overvåkes nøye gjennom hele prosjektet. Testplanlegging tar med i betraktning tilbakemelding fra overvåkings- og styringsaktiviteter.
- Testplanleggings- og styringsaktiviteter er nærmere definert i kapittel 5 i ISTQB- pensum.



# Testanalyse og -design (analysis and design)

Testanalyse og -design er der de generelle testformålene blir omformet til konkrete testbetingelser og testtilfelle. Aktiviteten har følgende hovedoppgaver:

- Granskning av testgrunnlaget (slik som krav, kritikalitetsnivå (risikonivå), rapporter fra risikoanalyse, arkitektur, design, grensesnittspesifikasjoner).
- Evaluering av testbarheten av testgrunnlaget og testobjektene.
- Identifisering og prioritering av testbetingelsene basert på en analyse av testobjektene, spesifikasjonen, atferd og struktur av programvaren.
- Utarbeidelse og prioritering av høynivå testtilfelle.
- Identifisering av nødvendige testdata for å støtte testbetingelsene og testtilfellene.
- Design av testmiljøet og identifisering av nødvendig infrastruktur og verktøy.
- Å skape sporbarhet i begge retninger mellom testbasis og testtilfelle.





# Testimplementering og –utførelse (implementation and execution)

Testimplementering og -utførelse er aktiviteten der testprosedyrer og scripts spesifiseres ved å kombinere testtilfelle i særskilt orden og inkludere all annen informasjon som behøves for utførelsen av testen, og der testmiljøet settes opp og testene utføres. Aktiviteten har følgende hovedoppgaver:

- Ferdigstilling, implementering og prioritering av testtilfelle (inklusive identifikasjon av testdata).
- Utvikling og prioritering av testprosedyrer, utarbeidelse av testdata, og hvis det er behov, forberedelse av testrammer og skriving av automatiske testscripts.
- Produksjon av testsuiter fra testprosedyrene for å muliggjøre effektiv testutførelse.
- Verifisering av at testmiljøet har blitt satt opp på korrekt måte.
- Verifisering og oppdatering av tosidig sporbarhet mellom testbasis og testtilfelle.
- Utførelse av testprosedyrer, enten manuelt eller ved bruk av testverktøy, ifølge en planlagt sekvens
- Logging av resultater fra testutføringen og registrering av identiteter og versjoner av programvaren under test, testverktøy og testware.
- Sammenligning av faktiske resultater med forventede resultater.
- Rapportering av avvik som hendelser og analyse av disse for å identifisere årsak (f. eks. en feil i koden, i spesifiserte testdata, i et testdokument, eller en feiltakelse i måten testen ble utført).
- Gjentakelse av testaktiviteter som følge av tiltak som ble utført i forbindelse med avvik. For eksempel, gjennomføring av en test som tidligere ikke lyktes for å bekrefte en feilrettelse (re-testing), utførelse av en korrigert test og/eller utførelse av tester for å kontrollere at feil ikke har blitt introdusert i uendrede områder av programvaren eller at en feilretting ikke avslørte andre feil (regresjonstesting).



## Evaluering av sluttkriterier og rapportering (evaluating exit criteria and reporting)

---

Evaluering av sluttkriterier er den aktiviteten der testutførelsen blir vurdert i forhold til de definerte testformålene. Dette bør gjøres for hvert testnivå (se kapittel 2.2). Aktiviteten har følgende hovedoppgaver:

- Kontroll av testlogger mot sluttkriteriene som ble spesifisert i testplanleggingen.
- Vurdering om flere tester behøves eller om sluttkriteriene bør forandres.
- Utarbeidelse av testsluttrapporten for de involverte parter.

# Avslutningsaktiviteter (closure)

---

Her samles det inn data fra avsluttede testaktiviteter for å samle erfaring, testware, fakta og tall. Slike aktiviteter gjøres ved prosjektmilepeler for eksempel når et programvaresystem frigis, når et testprosjekt er ferdig (eller avbrytes), når en milepæl har blitt oppnådd, eller når en vedlikeholdsfrigivelse har blitt fullført. Testingsavslutningsaktiviteter omfatter følgende hovedoppgaver:

- Kontroll av hvilke av de planlagte oppgavene er utført og testartefaktene som har blitt levert.
- Lukking av hendelsesrapporter eller opprettelse av endringsregistreringer for de som fortsatt er åpne.
- Dokumentasjon på at systemet er blitt godkjent eller underkjent.
- Ferdigstilling og arkivering av testmaterialet, testmiljøet og testinfrastrukturen for fremtidig gjenbruk.
- Overlevering av testware til de som er ansvarlige for vedlikehold.
- Analyse av erfaringer for å bestemme endringer som behøves ved fremtidige utgivelser og prosjekter.
- Bruk av informasjonen som er samlet for forbedring av testmodenhet.



# Oppsummering kapittel 1

Testing er nødvendig fordi:

- Feil er dyre, kvalitet, «liv kan gå tapt»

Mål med test er:

- Gi informasjon og beslutningsgrunnlag, finne defects, få tillit til systemet, prosessforbedring

7 testprinsipper: (må pugges)

- Tilstedeværelse av feil, fullstendig test, tidlig test, klyngedannelse, pesticid-paradoks, kontekstavhengig, «fravær av feil»

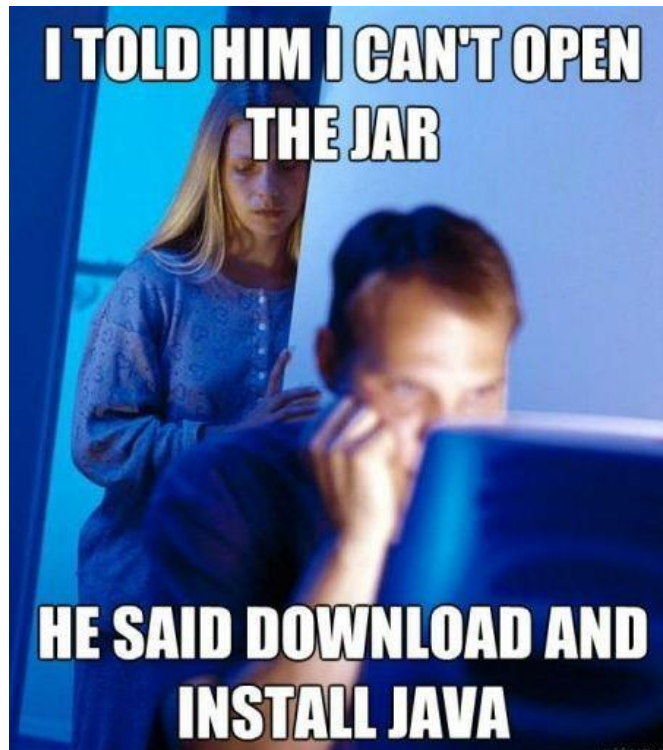
5 steg i testprosessen: (må pugges )

- Analysis and design, implementation and execution, evaluating exit criteria and reporting, closure

# Kahoot spørsmål fra kapittel 1



# Kaffepause!



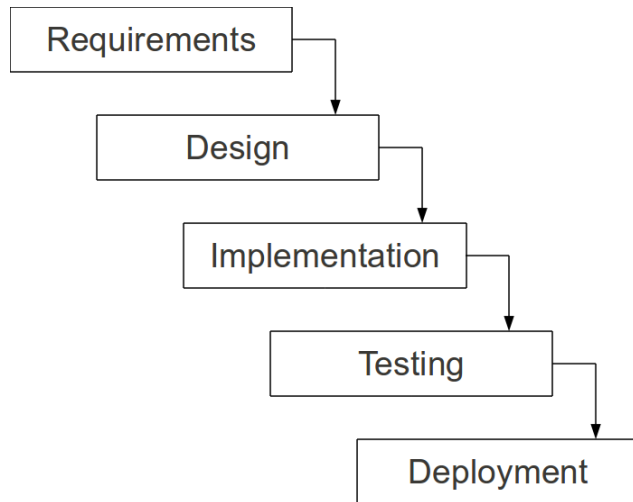
# Kapittel 2: Testing gjennom livssyklusen

- Utviklingsmodeller for programvare
- Testnivåer
- Testtyper
- Testing i vedlikehold



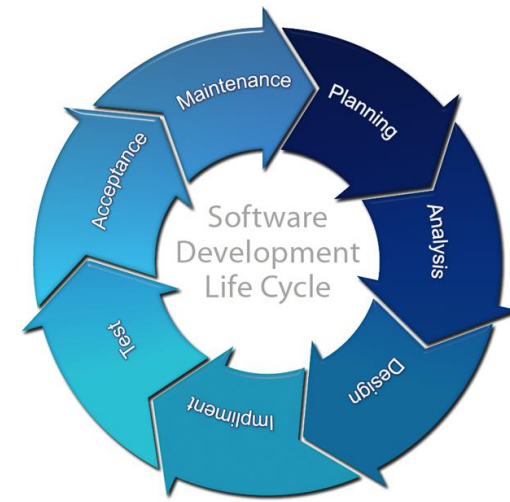
# Softwareutviklingsmodeller: sekvensiell eller iterativ-inkrementell

- Fossefall



- Fast scope – én leveranse
- Synliggjør testingen
- Tydelig fokus på testplanlegging
- Klare krav

- Smidig



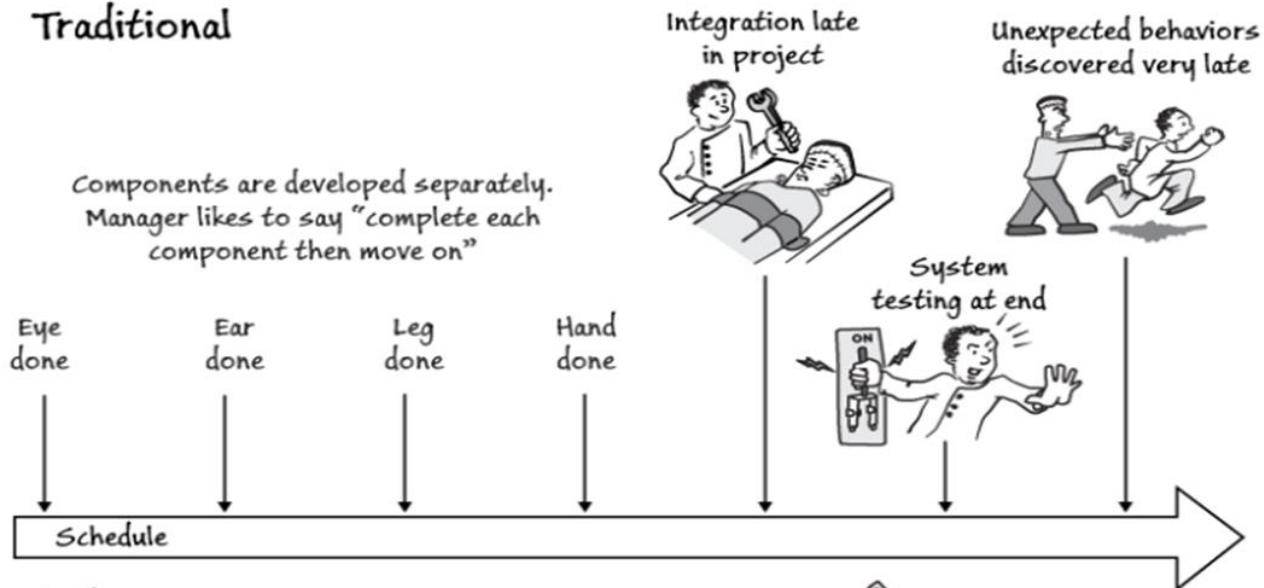
- Tilpasning underveis
- Det viktigste først
- Tidlig i test/produksjon
- Endre behov underveis



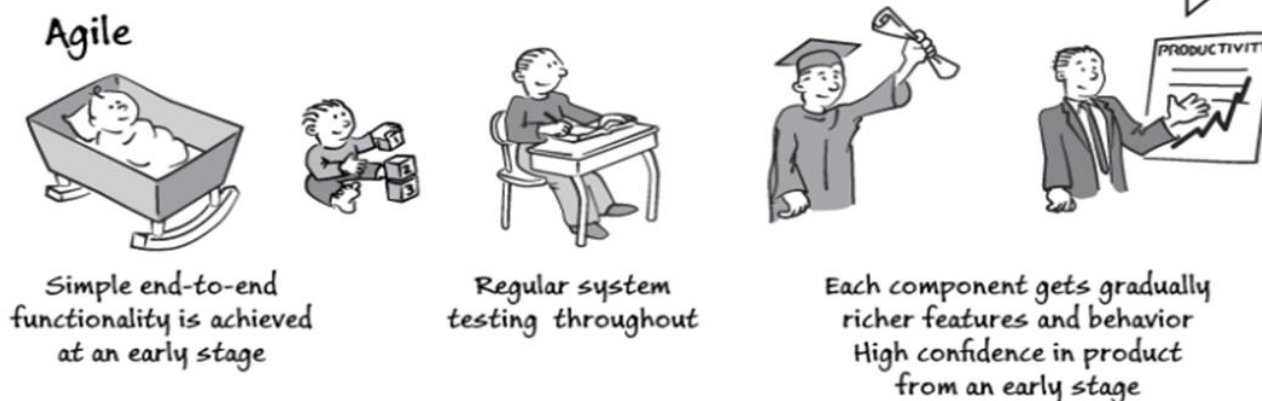


# Traditional vs. Agile Development

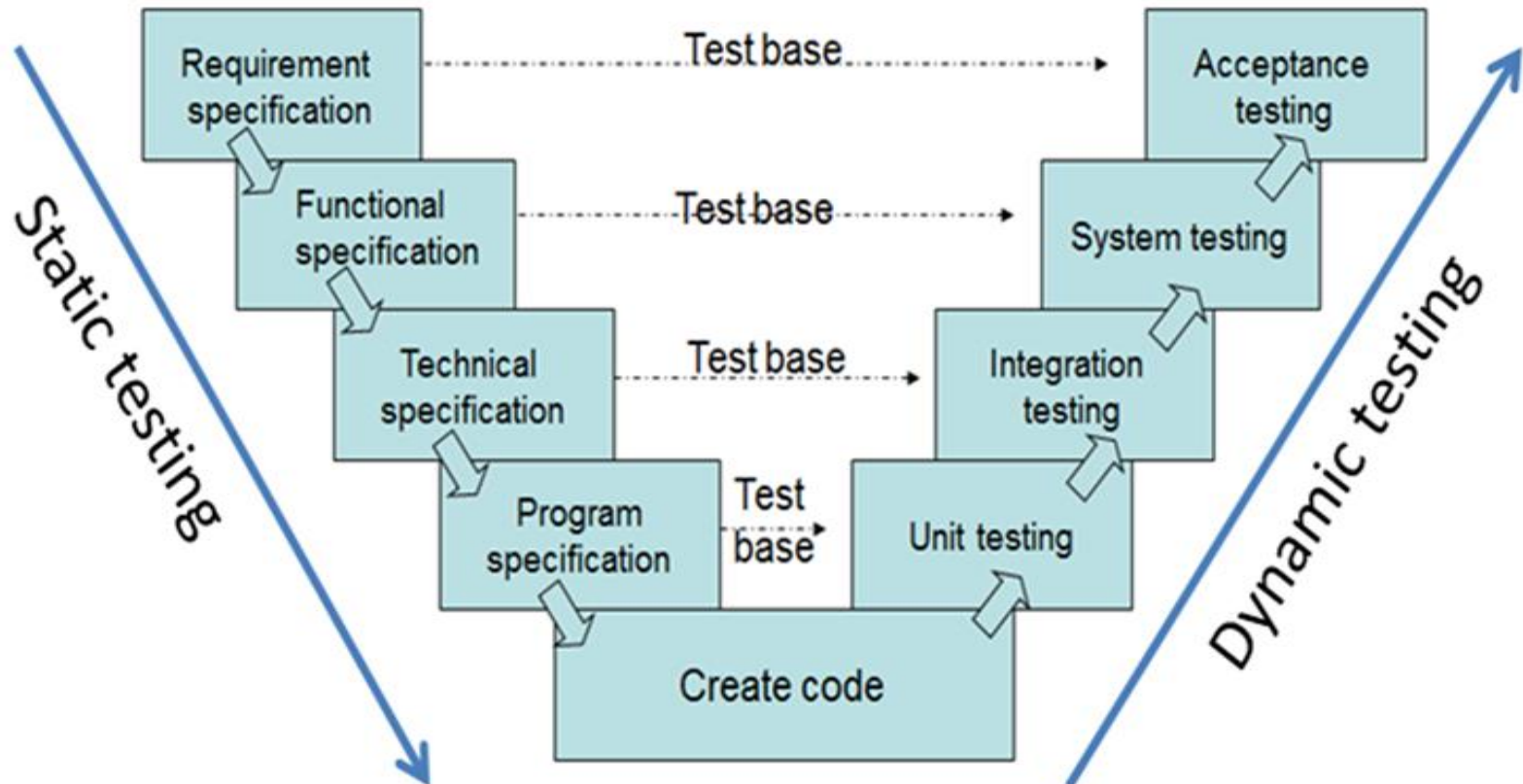
## Traditional



## Agile



## V-modellen (sekvensiell)



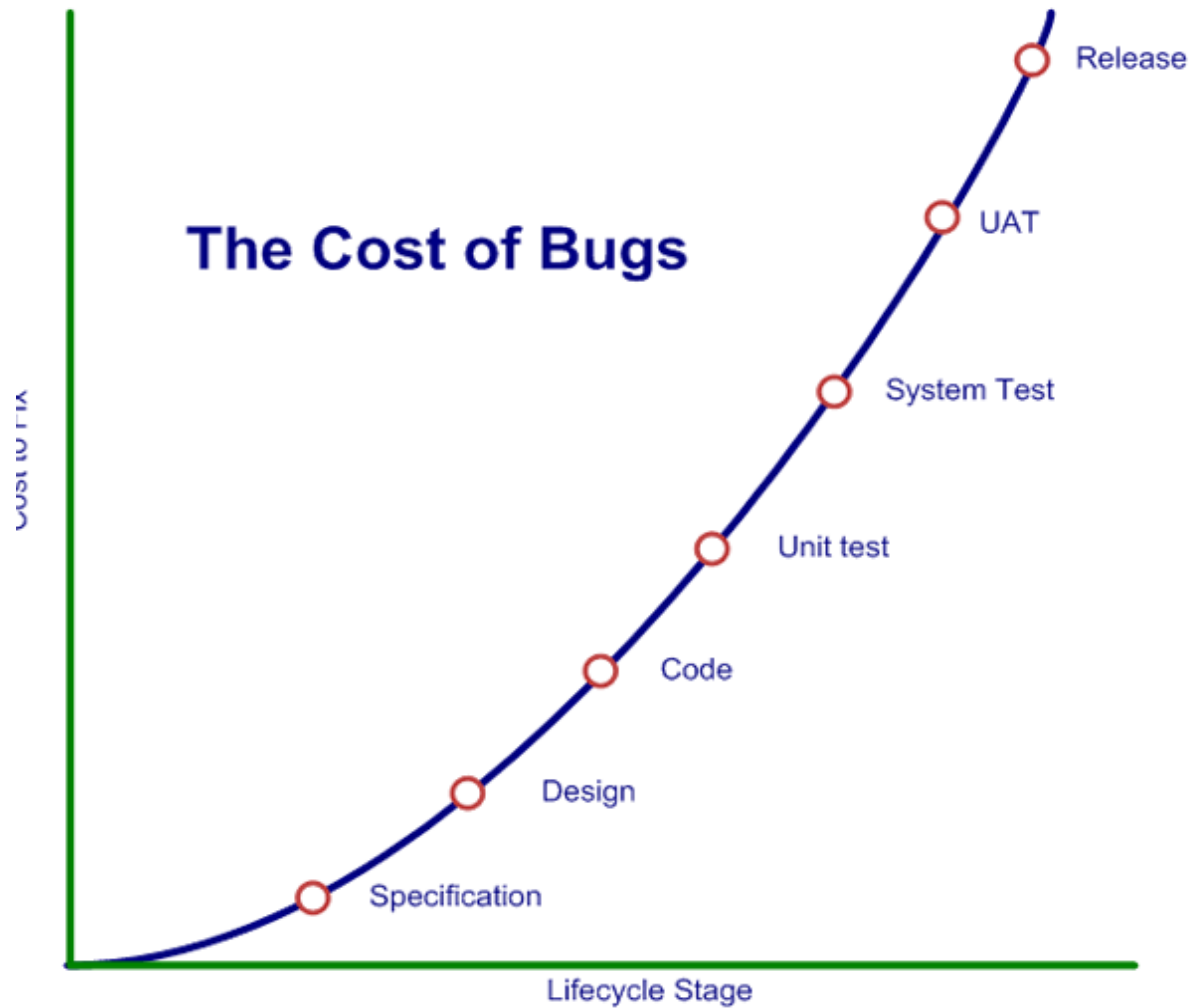
# Testing i en livssyklusmodell

---

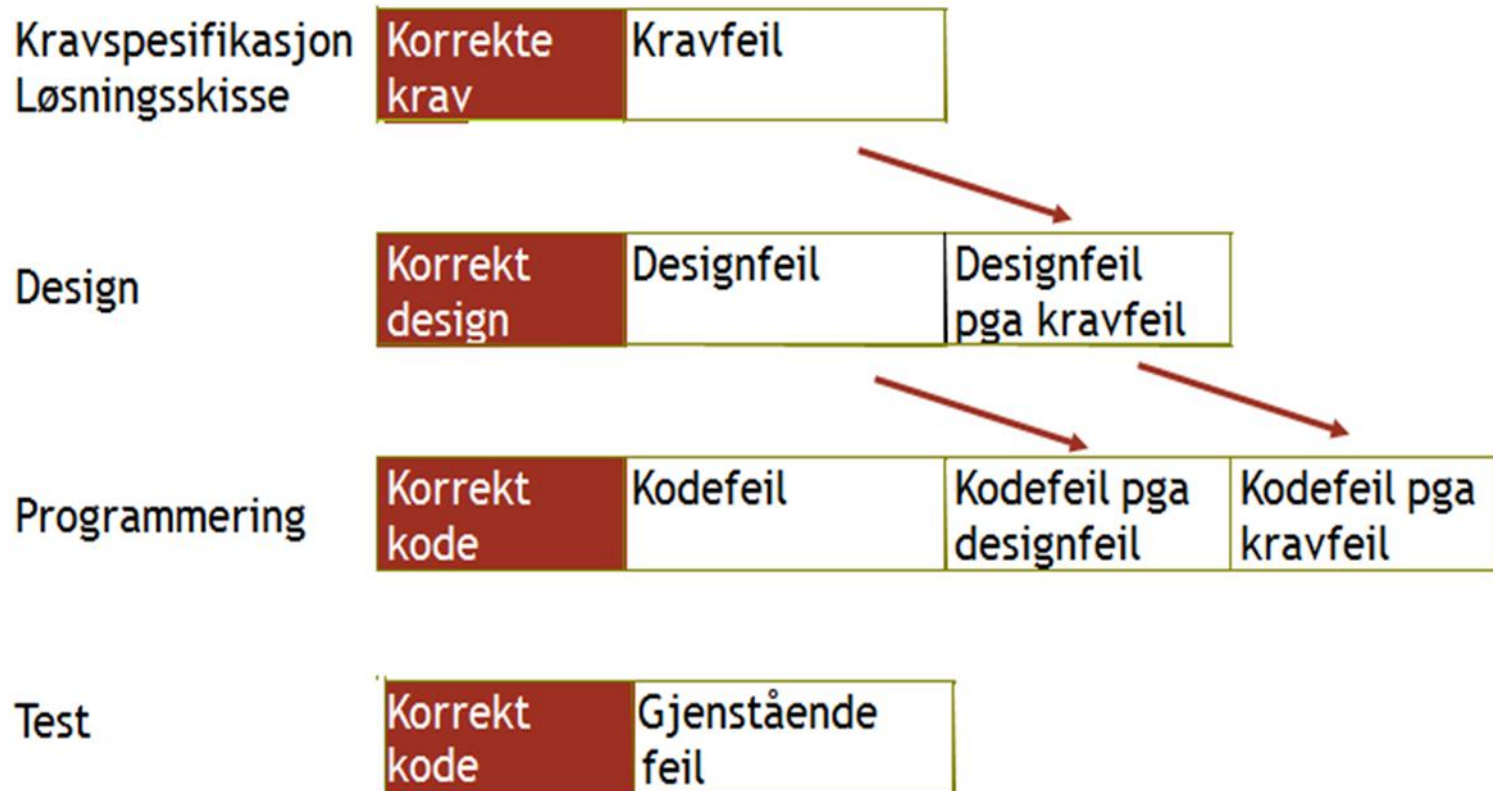
- I enhver livssyklusmodell er god testing karakterisert ved følgende:
  - For hver utviklingsaktivitet er det en tilsvarende testaktivitet
  - Hvert testnivå har bestemte testmål
  - Analyse og design av testene for et gitt testnivå bør starte når den tilsvarende utviklingsaktiviteten utføres
  - Testere bør delta i gjennomgang av dokumenter så snart utkast er tilgjengelig



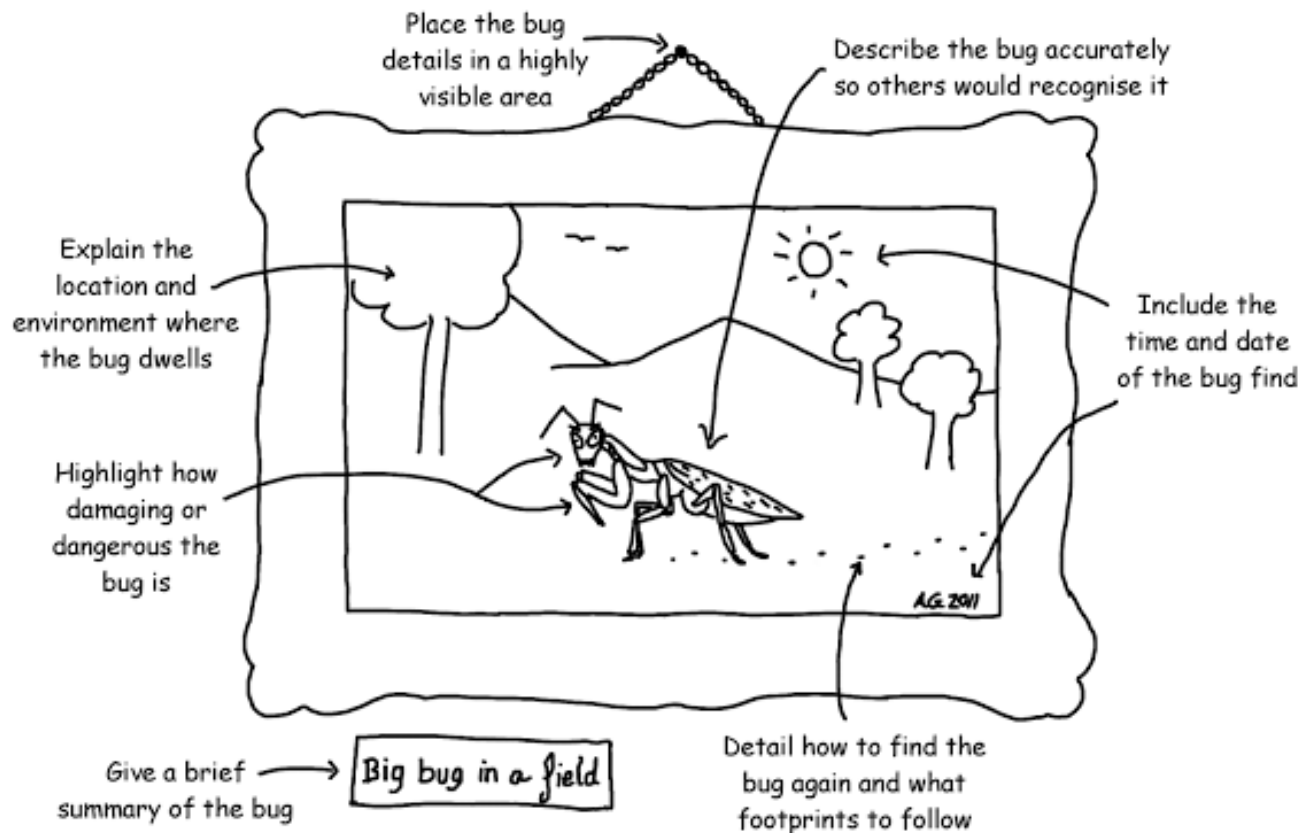
## Kostnader knyttet til bugs



## Hvor oppstår feilene



## The Art of Bug Reporting

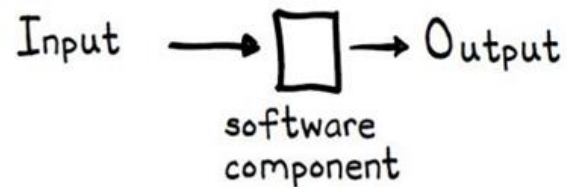


AG  
Andy Glover [cartoontester.blogspot.com](http://cartoontester.blogspot.com) © 2012

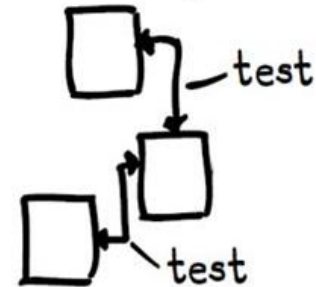
# Testnivåer (test levels) – viktig for eksamen!

## Test levels

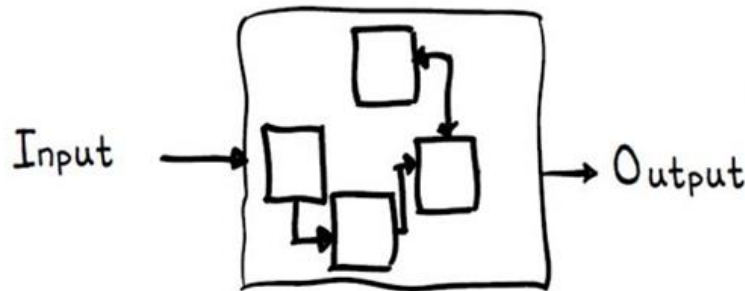
### Unit testing



### Integration testing



### System testing



### Acceptance testing



For hvert testnivå definerer man testbasis og testobjekter

---

### Testbasis

- Grunnlaget for planlegging av test
- Definere forventet resultat

**Eks:** Brukerhistorier, workflows, spesifikasjoner, diagrammer

### Testobjekt

- Den «fysiske» deler vi tester

**Eks:** Moduler, systemer, API'er, hardware





# Komponenttesting/enhetstesting

---

- Gjøres vanligvis av utviklerne
- Minste mulige testbare enhet
- Defects fikses løpende uten at de logges noe sted

## Testbasis:

- Krav til komponenten
- Detaljdesign
- Kode

## Testobjekter:

- Komponenter
- Programmer
- Datakonverterings-/migrasjonsprogrammer
- Databasemoduler



# Integrasjonstesting

---

- Tester grensesnitt mellom komponenter og kommunikasjon mellom ulike deler av systemet
- Flere integrasjonstestnivåer - avhengig av testobjekt/mål
- Bør integrere trinnvis fremfor «big bang»

## Testbasis:

- Software- og systemdesign
- Arkitektur
- Arbeidsflyt
- Use cases

## Testobjekter:

- Delsystemer
- Databaseimplementasjon
- Infrastruktur
- Grensesnitt
- Systemkonfigurasjon
- Konfigurasjonsdata



# Systemtest

---

- Gjelder hele systemets oppførsel, dvs omfanget av et prosjekt
- Testmiljøet bør ligge så tett opptil produksjonsmiljøet som mulig
- Bør omfatte både funksjonelle og ikke-funksjonelle krav til systemet

## Testbasis:

- Kravspesifikasjon (system)
- Use cases
- Funksjonell spesifisering
- Risikoanalyserapporter

## Testobjekter:

- System-, bruker- og driftsmanualer
- Systemkonfigurasjon og konfigurasjonsdata



# Akseptansetest

---

- Kundens ansvarsområde– er systemet klart for installasjon?
- 4 typer
  - User acceptansetesting
  - Operational acceptansetesting
  - Alfa og beta testing
  - Contractual and Regulatory acceptansetesting

## Testbasis:

- Brukerkrav
- Systemkrav
- Use cases
- Forretningsprosesser
- Risikoanalyserapporter

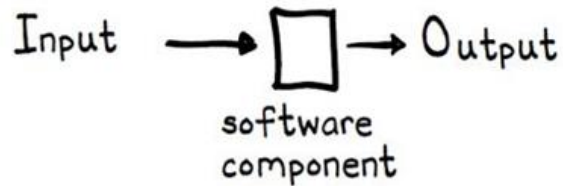
## Testobjekt:

- Forretningsprosess på det integrerte systemet
- Drifts- og vedlikeholdsprosess
- Bruksprosedyrer
- Rapporter
- Konfigurasjonsdata

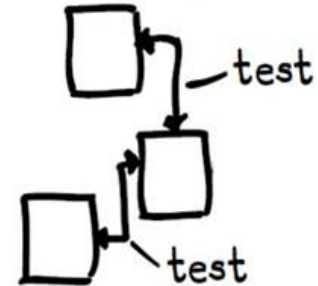


# Test levels

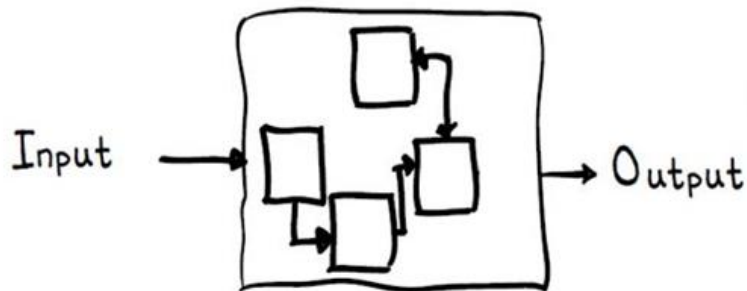
## Unit testing



## Integration testing



## System testing



## Acceptance testing



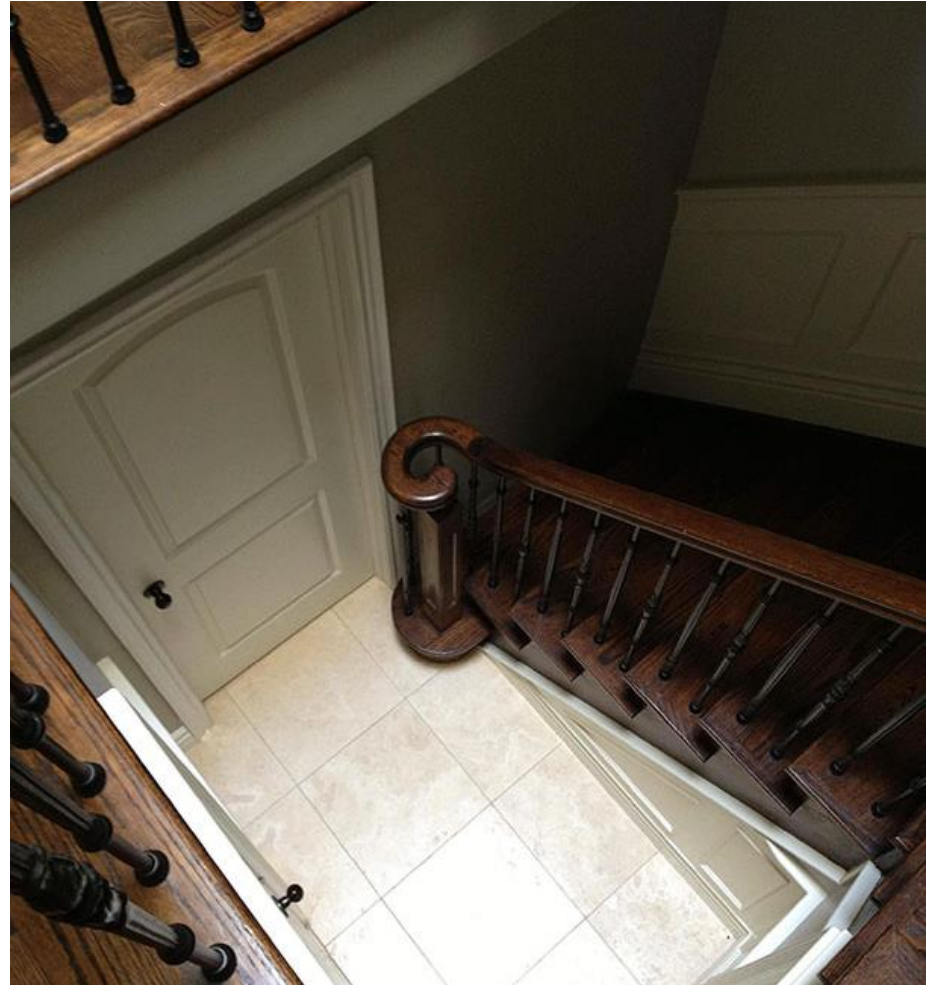
## 4 Testtyper (test types) – viktig for eksamen!

| Testtyper                     | Testformål  |
|-------------------------------|---|
| <b>Funksjonell test</b>       | Funksjonen programmet skal oppfylle   |
| <b>Ikke- funksjonell test</b> | Kvalitetsegenskaper som Performance, load test, stress test, maintainability, reliability, portability, usability |
| <b>White-box test</b>         | Hvordan er løsningen strukturert opp (tenk gjennomsiktig boks)  |
| <b>Endringsrelatert test</b>  | Teste selve endringen som er gjort, samt om den har påvirket andre deler av koden                                 |

# 1. Funksjonell testing

---

- Kan utføres på alle testnivåer
- Utleder testbetingelser og testtilfelle fra systemets funksjonalitet
- Fokus på «hva» systemet gjør
- Prosessflyt eller ulike tilstander av systemet
- Negativ test



## 2. Ikke-funksjonell testing

---

- Kan utføres på alle testnivåer
- Fokus på «hvordan» systemet virker
- Eksempler: performance, load, stress, maintainability, reliability, portability, usability
  - Ofte dårlig spesifisert..
  - Hvordan tester vi disse egenskapene?

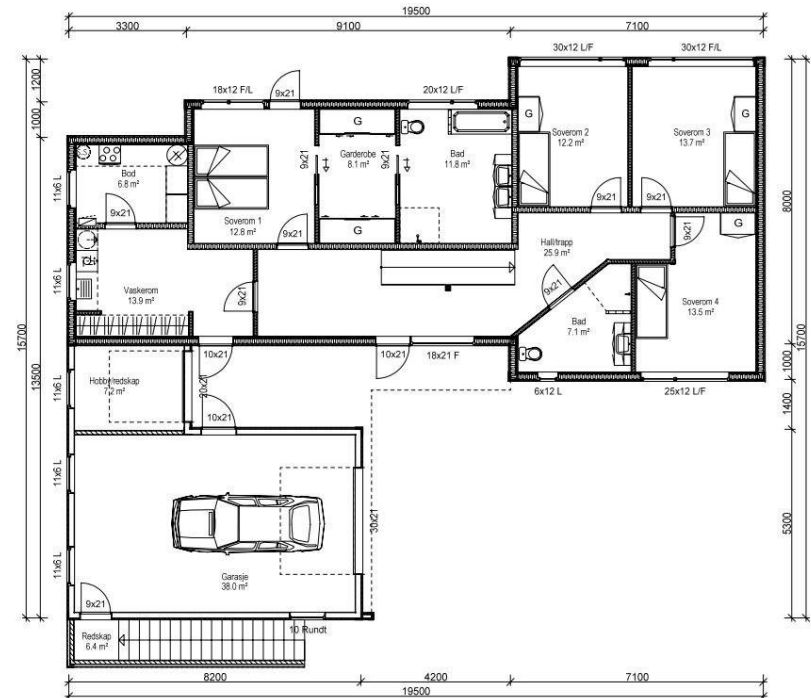




### 3. White-box

Test av programvarens struktur/arkitektur (strukturell testing/white box)

- Kan utføres på alle nivåer:
  - Komponenttesting: code coverage
  - Integration: module coverage
  - System: menu coverage
  - Acceptance: business model coverage
- Uttrykkes gjerne i prosent, kan også produsere flere tester for å øke dekningsgraden



## 4. Endringsrelatert test (retest og regresjonstest)

---

- Kan utføres på alle testnivåer
- Når det er innført endringer av systemet, f.eks defect fixes.
  - **Retest** av selve feilen
  - **Regresjonstest** har endringen påvirket resten av systemet?





99 little bugs in the code.  
99 little bugs in the code.  
Take one down, patch it around.  
127 little bugs in the code...

Regression:  
"when you fix one bug, you  
introduce several newer bugs."



# Testing i vedlikehold

## Maintenance testing

---

- Hvorfor?
  - Systemet har blitt **modifisert**
    - Patcher, fixes, forbedringer, ny versjon, miljøendringer
    - Liten endring trenger ikke bety liten test!
  - Systemet skal flyttes til en ny plattform (**migrasjon**)
    - Må sjekke at systemet fortsatt fungerer operasjonelt
  - Fases ut (**retirement**)
    - Arkivering av data og migrering
- Konsekvensanalyse



# Oppsummering kapittel 2

---

- 2 systemutviklingsmodeller:
  - Sekvensiell og iterativ-inkrementell
- 4 testnivåer:
  - Komponenttest, integrasjonstest, systemtest, akseptansetest

*4 typer akseptansetest:*

  - ✓ Brukerakseptanse, operasjonell, kontrakt, alfa/beta-testing
- 4 testtyper:
  - Funksjonell, ikke-funksjonell, whitebox, endringsrelatert

*7 typer ikke-funksjonell testing:*

  - ✓ Performance, load test, stress test, maintainability, reliability, portability, usability
- 3 typer vedlikeholdstesting (maintenance):
  - Modifications, migration, retirement of software



## Spørsmål: Match formålet med testtypen

- **Gjør de rette matchene:**

- 1) Funksjonell test
- 2) Ikke-funksjonell test
- 3) Strukturell test
- 4) Test av endringer

- A) Teste kode
- B) Bekreftelse og regresjonstest
- C) testing of quality attributes
- D) Tester basert på krav

1) 1A, 2B, 3D, 4C

2) 1D, 2B, 3A, 4C

3) 1D, 2C, 3A, 4B

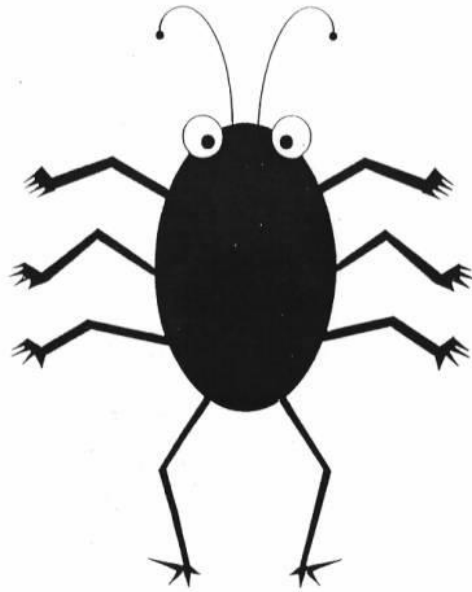
4) 1A, 2C, 3B, 4D



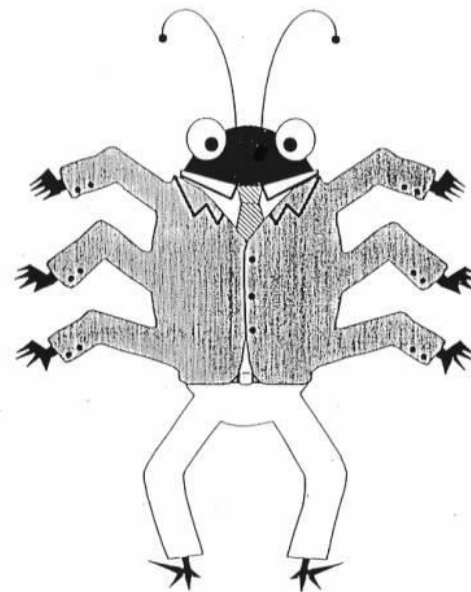
# Kahoot spørsmål fra kapittel 2



# Lunsj!



**BUG**



**FEATURE**



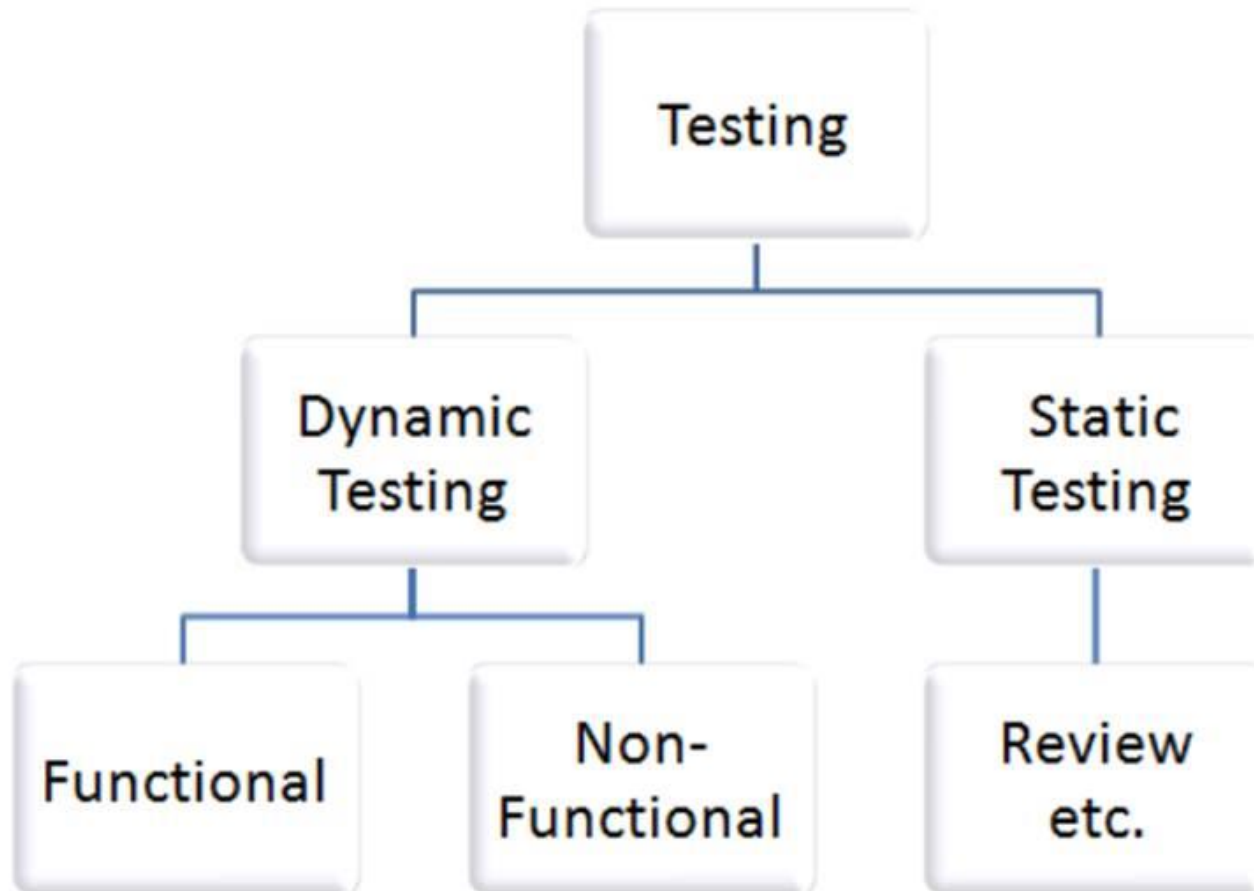
# Kapittel 3: Statiske teknikker

- 3.1 Statisk testing – hva er det og hvorfor bruker vi det?
- 3.2 Review prosessen



**Faser og roller i formelle granskninger: viktig for eksamen, pugg dette på egenhånd 😊**

*Nytt fra 2018: As focus on reviews is increase in the 2018 syllabus, tester should be able to summarize the activities in their own words (Updated from K1 to K2)*



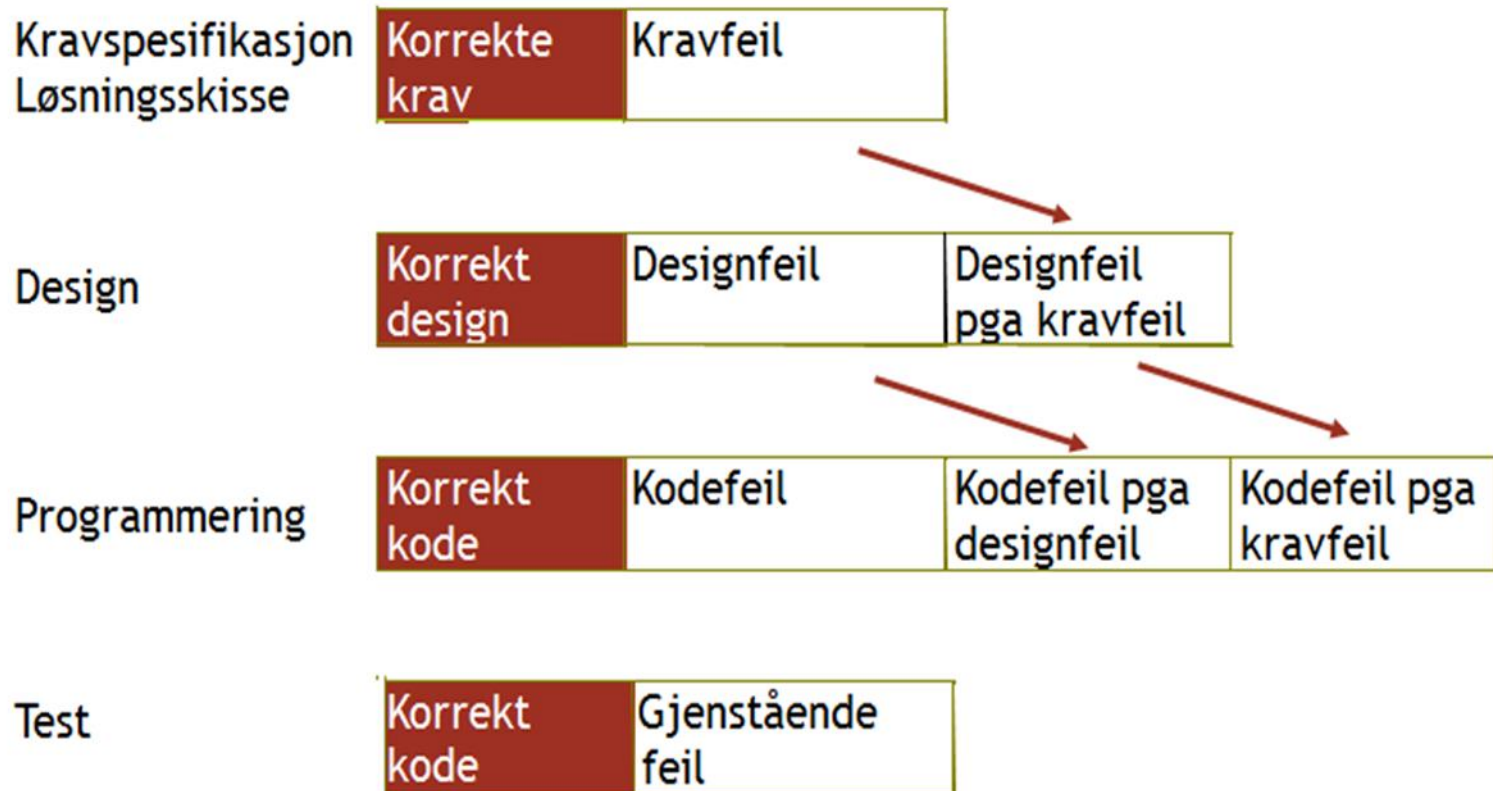
## Hva tester vi?

---

- Specifications, including business requirements, functional requirements, and security requirements
- Epics, user stories, and acceptance criteria
- Architecture and design specifications
- Code
- Testware, including test plans, test cases, test procedures, and automated test scripts
- User guides
- Web pages
- Contracts, project plans, schedules, and budgets
- Models, such as activity diagrams, which may be used for Model-Based testing



# Hvor oppstår feilene





## **THE PROBLEM ABOUT BEING A PROGRAMMER**

My mom said:

"Honey, please go to the market and buy 1 bottle of milk. If they have eggs, bring 6"

I came back with 6 bottles of milk.

She said: "Why the hell did you buy 6 bottles of milk?"

I said: "BECAUSE THEY HAD EGGS!!!!"

## Fordeler med bruk av statisk analyse

---

- Tidlig oppdagelse av feil før testutføring.
- Tidlig varsel om mistenkelige egenskaper ved koden eller designet ved å beregne måleverdier, som høy kompleksitet.
- Identifisering av feil som dynamisk testing ikke finner så lett.
- Oppdagelse av avhengigheter og selvmotsigelser i softwaremodeller
- Forbedret vedlikeholdbarhet av kode og design.
- Forebygging av feil, hvis en lærer av feilene under utvikling.



# Statisk testing kan identifisere defects tidligere og/eller som dynamisk testing ikke kan

---

- Requirement defects (e.g., inconsistencies, ambiguities, contradictions, omissions, inaccuracies, and redundancies)
- Design defects (e.g., inefficient algorithms or database structures, high coupling, low cohesion)
- Coding defects (e.g., variables with undefined values, variables that are declared but never used, unreachable code, duplicate code)
- Deviations from standards (e.g., lack of adherence to coding standards)
- Incorrect interface specifications (e.g., different units of measurement used by the calling system than by the called system)
- Security vulnerabilities
- Gaps or inaccuracies in test basis traceability or coverage (e.g., missing tests for an acceptance criterion)



# Statisk analyse med verktøy

---

- Mål: finne feil i kildekoden og i modeller (ikke mot krav, men mot kodestandards)
- Koden kjøres ikke, men analyseres (kontroll- og dataflyt)
  - Forhåndsdefinerte regler og programmeringsstandards: feil syntaks, sårbarheter..
  - Under komponent- og integrasjonstest, eller ved innsjekk av kode
- Kan måle testdekningen på ulike nivåer



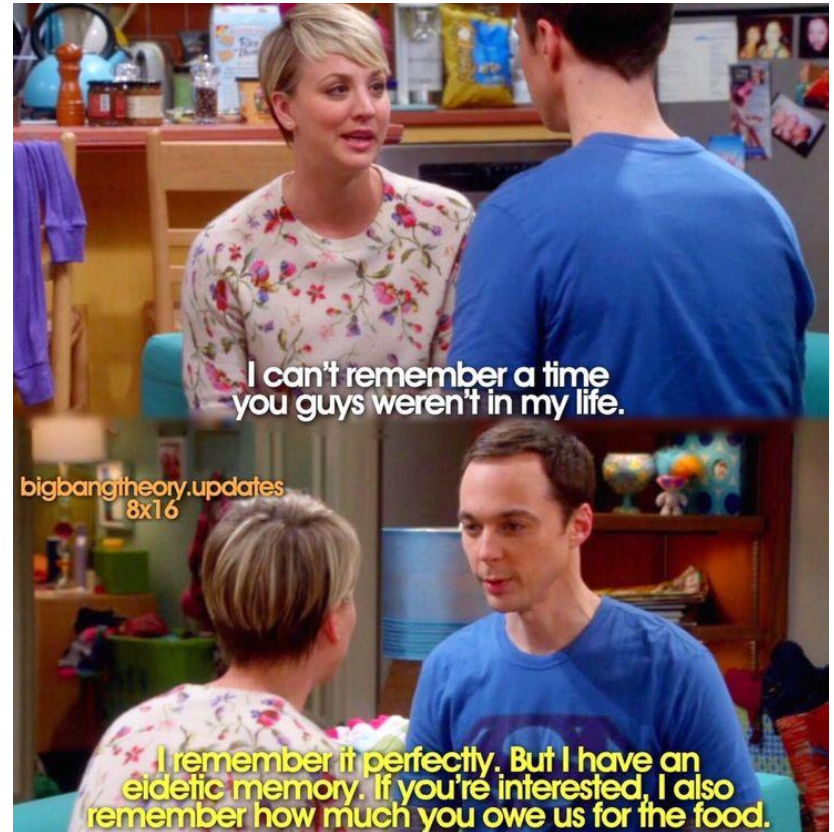
## 2. Reviewprosessen

---

- Varierer fra meget uformell (for eksempel ingen nedskrevne instruksjoner for de som gjennomgår) til meget formell (dvs. medvirkning fra hele gruppen, dokumenterte resultater og dokumenterte prosedyrer for å holde en granskning)
- Ulike mål: finne feil, forstå problemområde, opplæring av testere, diskusjon osv..

## 2. Review prosessen – huskeliste og oversikt

- 5 aktiviteter i prosessen
- 6 ulike roller
- 4 ulike typer review
- 5 ulike teknikker



## Aktiviteter

---

- 1. Planning** – definere mål, ressurser, og metode
- 2. Initiate review** – distribuere dokumenter/kode, forklare mål
- 3. Individual review (i.e., individual preparation)** - Gå igjennom dokument/kode
- 4. Issue communication and analysis** – Gå igjennom resultat og defects (Kategorisering)
- 5. Fixing and reporting** – Fikse defects, godkjenne produkt



# Typer statisk testing

## 1. Informal review

Samtaler med enkeltpersoner og forfatter

## 2. Walkthrough

Gjennomgang av dokumenter og forfatter

## 3. Technical review

Gjennomføres av tekniske spesialister  
Av tekniske dokumenter/kode

## 4. Inspection

Ofte kalt formal review  
Ledes av en «moderator»  
«Scriber» skriver referat



## Uformell granskning/Informal review (e.g., buddy check, pairing, pair review)

---

Hovedegenskaper:

- Ingen formell prosess;
- Dette kan være parprogrammering eller at en teknisk ekspert går gjennom design og kode;
- Kan (men må ikke) være dokumentert;
- Nytteverdien kan variere avhengig av hvem som gjennomgår;
- **Hovedformål:** En billig måte å få nyttig feedback.



# Strukturert gjennomgang (walkthrough)

---

## Hovedegenskaper:

- Møte ledes av forfatteren
- Scenarier, “tørrkjøring”, gruppe av arbeidskolleger
- Møtet har åpen dagsorden
- Mulighet for at deltakerne forbereder seg før møtet, og mulighet for granskningsrapport, liste av hva som er funnet og mulighet for en protokollant (som ikke er forfatter)
- Kan i praksis variere fra meget uformell til meget formell;
- **Hovedformål:** Finne feil i løsning, forbedre produktet, vurdere alternative implemteringer, sikre riktige standarder og løsninger
  - Brukes også som opplæring

# Teknisk gjennomgang/Technical review

---

- Dokumentert, definert prosess for å finne feil i produktet. Den inkluderer arbeidskolleger og tekniske eksperter.
- Kan utføres som en «peer review» uten at ledelsen deltar
- I ideelle tilfeller ledet av en opplært møteleder (moderator), ikke av forfatteren
- Reviewerne forbereder seg før møtet;
- Valgfri bruk av sjekklister
- Utarbeidelse av en granskningsrapport som omfatter en liste av hva som er funnet og utsagn om produktet oppfyller sine krav og, der det behøves, anbefalinger vedrørende de ting som er funnet.
- Kan i praksis variere fra meget uformell til meget formell
- **Hovedformål:** Å diskutere, gjøre beslutninger, evaluere alternativer, finne feil, løse tekniske problemer **og sjekke samsvar med spesifikasjoner, planer, forskrifter og standarder.**



# Inspeksjon/Inspection

---

- Ledet av en opplært møteleder (moderator), ikke av forfatteren
- Vanligvis sjekking ved hjelp av arbeidskolleger
- Definerte roller
- Samling og bruk av måldata
- Formell prosess basert på regler og sjekklister
- Spesifiserte inngangs- og utgangskriterier for å godkjenne produktet;
- Forberedelse før møtet
- Inspeksjonsrapport med liste av anmerkninger
- Formell oppfølgingsprosess (med valgfri bruk av data til prosessforbedring)
- Valgfri bruk av en leser
- **Hovedformålet er** detecting potential defects, evaluating quality and building confidence in the work product, preventing future similar defects through author learning and root cause analysis





## Roller (Endringer fra 2011)

- **Author** – eier av dokumentet
- **Management** - ansvarlig for planlegging
- **Facilitator (often called moderator)** – sørge for effektive møter
- **Review leader** – overordnet ansvar for review
- **Reviewers** – gjennomfører av review
- **Scribe (or recorder)** - sekretær



# Suksessfaktorer

---

- Granskningen har et klart og definert mål
- De rette menneskene er med
- Feilene som finnes er velkomne og de rapporteres objektivt
- Bruk av sjekklistor
- Deltakerne er opplært i granskningsteknikker, spesielt når det gjelder de mer formelle teknikker, som inspeksjon.



## 5 ulike statiske teknikker (NY i 2018)

**Ofte definert ut ifra hvilket perspektiv man har på testingen**

---

- **Ad hoc** - ad hoc 😊
- **Checklist-based** - **systematiske** reviews basert på definerte sjekklister
- **Scenarios and dry runs** – Definerer ulike senarioer for bruk av produktet. Mer spesifikk guidelines.
  - Eks: Releasedokumentasjon ; gå igjennom installasjonsprosedyren
- **Role-based** - produktet evalueres utifra ulike roller i organisasjonen eller ulike brukere
  - Eks: uerfarende, erfarende, unge, gamle brukere
  - Administratorer, Tester, produkteier
- **Perspective-based** – Reviewers tar ulike stalkholders viewpoint. Reviwers skal også «bruke produktet » til generere ønsket resultat.
  - Eks: Tester går igjennom brukerhistorier og «lager» akseptansetestsenarioer



# Oppsummering kapittel 3

---

- **Reviews/granskning – når og hvorfor brukes de?**
  - Finne defects og tvetydigheter tidlig i prosessen
  - Eksekverer ikke koden
- **4 typer av review:**
  - Informal review, walkthrough, technical review, inspection
- **5 aktiviteter i prosessen**
  - Planning, Initiate review, Individual review, Issue communication and analysis, Fixing and reporting
- **6 ulike roller**
  - Author, Manager, Facilitator, Review leader, Reviewers, Scribe (or recorder)
- **5 ulike teknikker**
  - Ad hoc , Checklist-based , Scenarios and dry runs, Role-based , Perspective-based



# Kahoot spørsmål fra kapittel 3



# Kaffepause!



# Kapittel 4: teknikker for testdesign

- Testutviklingsprosessen
- Kategorier av teknikker for testdesign
- Spesifikasjonsbaserte/black box-teknikker
- Strukturbaserte/white box-teknikker
- Erfaringsbasert teknikk
- Valg av testteknikker



# Testutviklingsprosessen

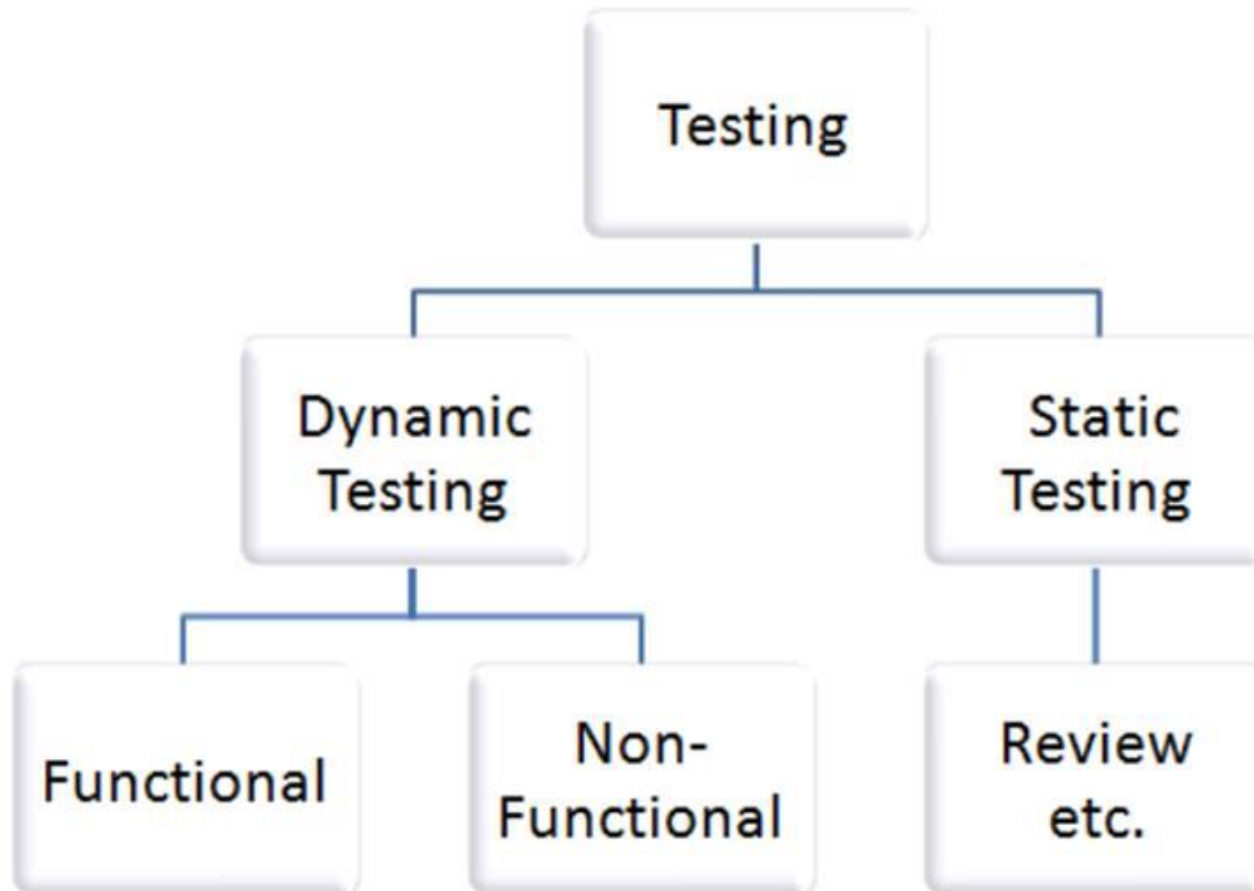
---

- Testutviklingsprosessen spenner fra å være meget formell til helt uformell uten dokumentasjon
  - Avhenger av kontekst og modenhet
- 1) Analyse av testgrunnlag for å identifisere testbetingelser
- 2) Identifisere testtilfelle
- 3) Designe og spesifisere testcases
- 4) Utvikle testprosedyre
  - Testtilfeller blir utviklet, implementert og prioritert
  - Samles i en kjøreplan





## Statisk og dynamisk testing



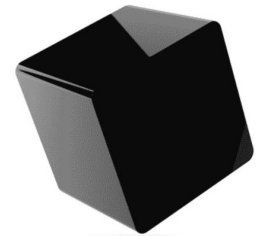
# Kategorier av teknikker for testdesign

---

Hensikt: å identifisere testbetingelser, testtilfeller og testdata

- **Spesifikasjonsbaserte teknikker (black box)**

- Modeller brukes spesifisering av problemet som skal løses, softwaren eller dens komponenter



- **Strukturbaserte teknikker (white box)**

- Info om hvordan softwaren er konstruert
- Enkelt å måle dekningsgrad



- **Erfaringsbasert teknikk (experience-based)**

- Kunnskap og erfaring om softwaren og hvor sannsynlige feil kan finnes seg



# Black box-teknikker

---

## BLACK BOX

### Spesifikasjonsbasert

- Ekvivalensklasseinndeling (equivalence partitioning)
- Grenseverdianalyse (boundary value analysis)
- Beslutningstabelltesting (decision tables)
- Tilstandsbasert testing (state transition testing)
- Brukstilfelletesting (use case)



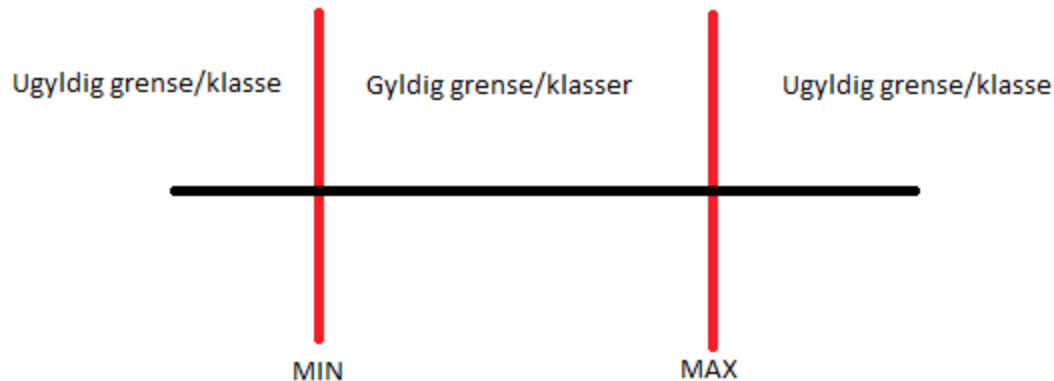
# Black box: Ekvivalensklasseinndeling

## Equivalence partitioning

---

Hvordan definere en ekvivalensklasse?

- Systemets inndata deles i intervaller
- En ekvivalensklasse er en datamengde der alle elementene behandles likt
- Ekvivalensklasser kan identifiseres fra spesifikasjonen
  - Tommelfingerregel - Velg data som ligger inne i ekvivalensklassen



# Black box: Grenseverdianalyse

## Boundary value analysis

---

- Vi designer tester til å teste/validere verdiene på grenser som er satt i systemet.
- Rundt grenseverdiene er det veldig mange feil/defects som «gjemmer seg»
- Feil på grenseverdiene kan skyldes misforståelse i kodingen fra utviklerne



# Black box: Grenseverdianalyse

## Boundary value analysis

---

Gyldige og ugyldige grenser:

- Gyldige grenseverdier er grenseverdier på grensen til ekvivalensklasser med gyldige verdier.
- Ugyldige grenseverdier er grenseverdier på grensen til ekvivalensklasser med ugyldige verdier.

### Eksempel

- Gyldig område [1 ... 100]
- Gyldige verdier = 1, 2..... 99, 100
- Ugyldige verdier = 0 og 101 f.eks

2 og 99 er mindre  
viktige å teste!  
(samme  
ekvivalensklasse  
som 1 og 100)

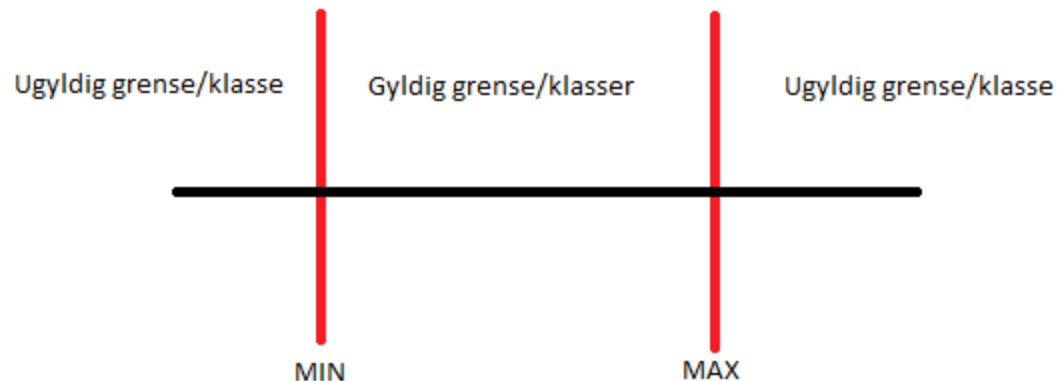
# Black box: Grenseverdianalyse

## Boundary value analysis

Hvordan finne grenseverdier?

Numeriske data på grenser:

- (1) Gyldig, ugyldig min
- (2) Gyldig, ugyldig max



# Black box: beslutningstabelltesting

## Decision tables

---

### Hva er beslutningstabeller bra for?

- Finne feil i logikk
  - AND - OR forvekslet
  - Er noen betingelser utelatt?
  - Betingelser feil vei (NOT)
  - Feil sammensetning av betingelser (parenteser)

### Hva brukes beslutningstabeller til?

- Brukt til å utforske ulike kombinasjoner basert på forhold som kan være riktige eller ikke.
- Hjelper testerne å sikre at alle kombinasjoner er tatt med i beregning.



# Black box: beslutningstabelltesting (1/2)

## Decision tables

---

### Oppgave: Uttak i minibank

#### Forutsetninger

1. Kun gyldige kort kan brukes, ugyldige kort skal avvises (men skal ikke slukes)
2. Du har 3 forsøk på å taste riktig PIN. Hvis du taster feil PIN på 3. forsøk, skal kortet slukes
3. Det er en forutsetning at du har penger på konto for å kunne ta ut penger.

# Black box: beslutningstabelltesting (2/2)

## Decision tables

| Tilstander | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Sjekke begge J og N? |
|------------|---|---|---|---|---|---|---|----------------------|
|            |   |   |   |   |   |   |   |                      |
|            |   |   |   |   |   |   |   |                      |
|            |   |   |   |   |   |   |   |                      |
|            |   |   |   |   |   |   |   |                      |
|            |   |   |   |   |   |   |   |                      |
| Resultater |   |   |   |   |   |   |   |                      |
|            |   |   |   |   |   |   |   |                      |
|            |   |   |   |   |   |   |   |                      |
|            |   |   |   |   |   |   |   |                      |
|            |   |   |   |   |   |   |   |                      |

# Black box: beslutningstabelltesting

## Decision tables

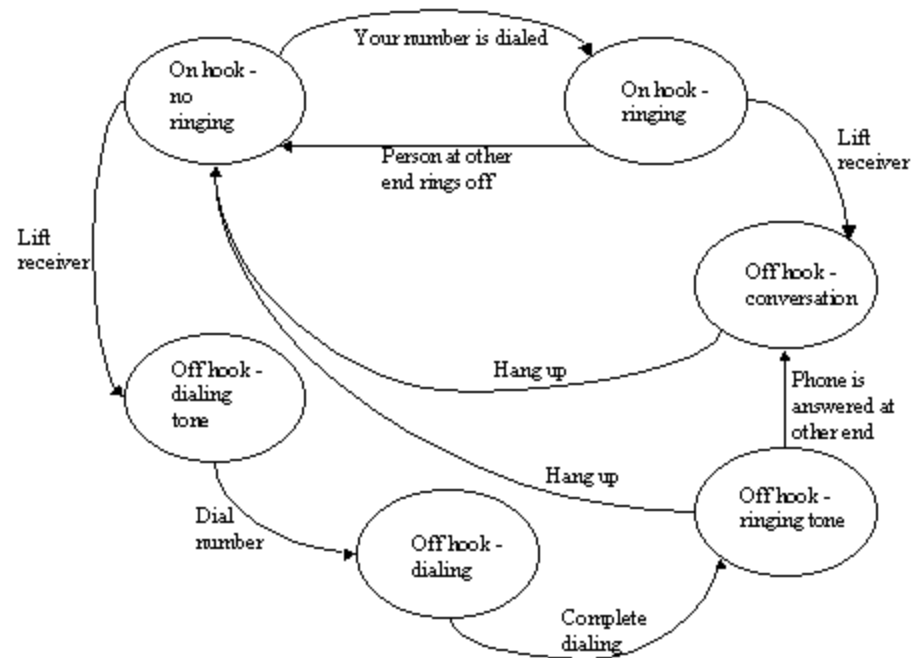
| Tilstander/Tiltilfeller | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Sjekke begge J og N? |
|-------------------------|---|---|---|---|---|---|---|----------------------|
| Gyldig kort             | N | J | J | J | J | J | J | OK                   |
| Første PIN riktig       | - | N | N | J | J | N | N | OK                   |
| 3 feil PIN              | - | N | N | N | N | J | J | OK                   |
| 1-2 feil PIN            | - | J | J | N | N | N | N | OK                   |
| Penger tilgjengelig     | - | N | J | N | J | N | J | OK                   |
| <b>Resultater</b>       |   |   |   |   |   |   |   |                      |
| Avvis kort              | J | N | N | N | N | N | N | OK                   |
| Prøv ny PIN             | N | Y | Y | N | N | J | J | OK                   |
| Sluk kort               | N | N | N | N | N | J | J | OK                   |
| Uttak penger            | N | N | J | N | J | N | N | OK                   |

# Black box: Tilstandsbasert testing

## State transition testing

Tester typiske sekvenser av tilstander.

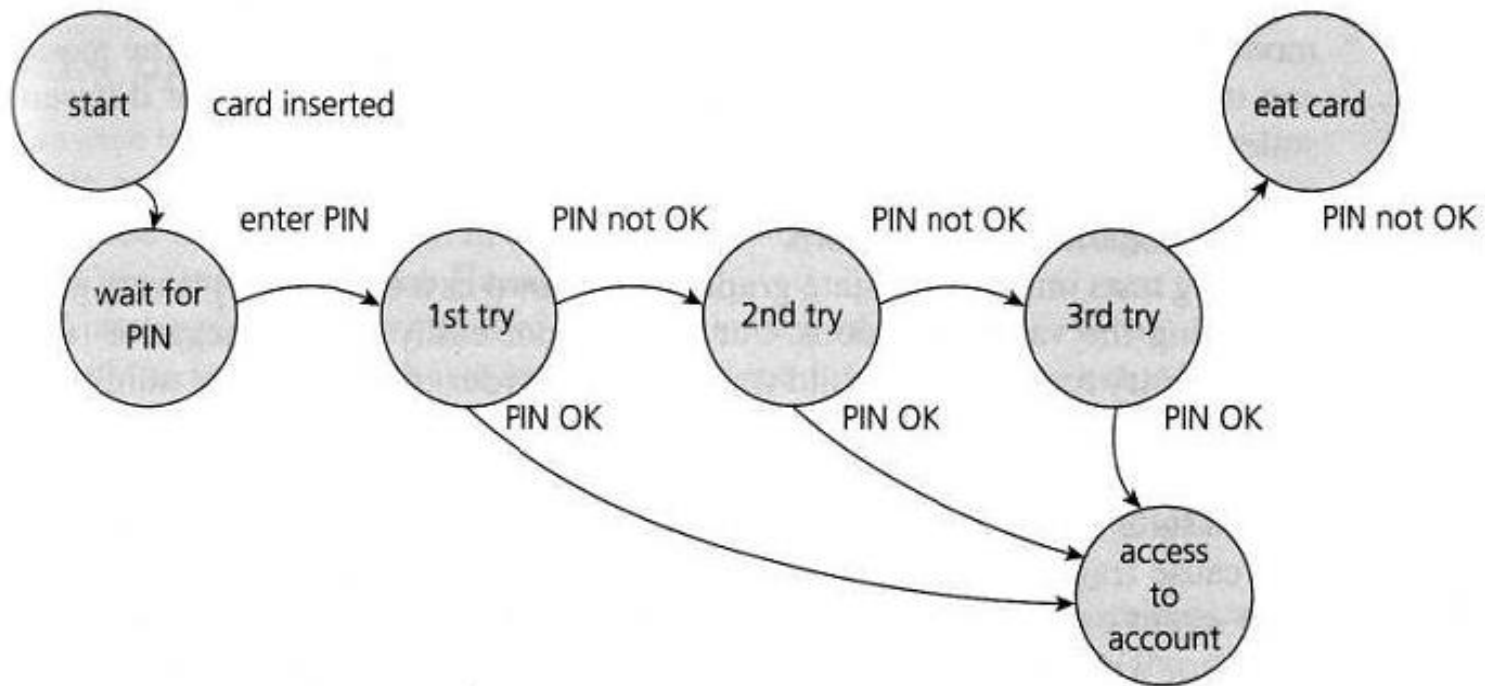
- Tester alle tilstander
- Tester alle overganger
- Tester alle sekvenser av overganger.
- Tester ugyldige overganger og tilstander.



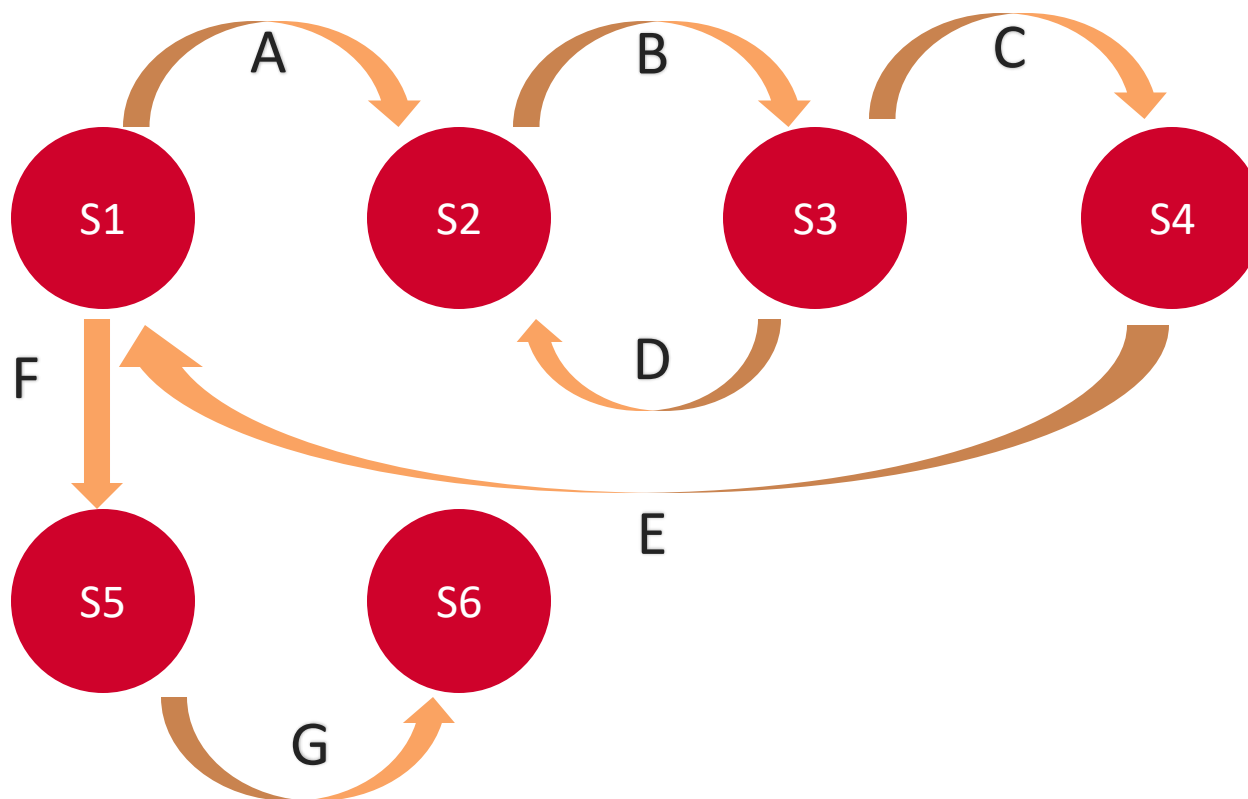
# Black box: Tilstandsbasert testing

## State transition testing

---



## Oppgave: state transition testing



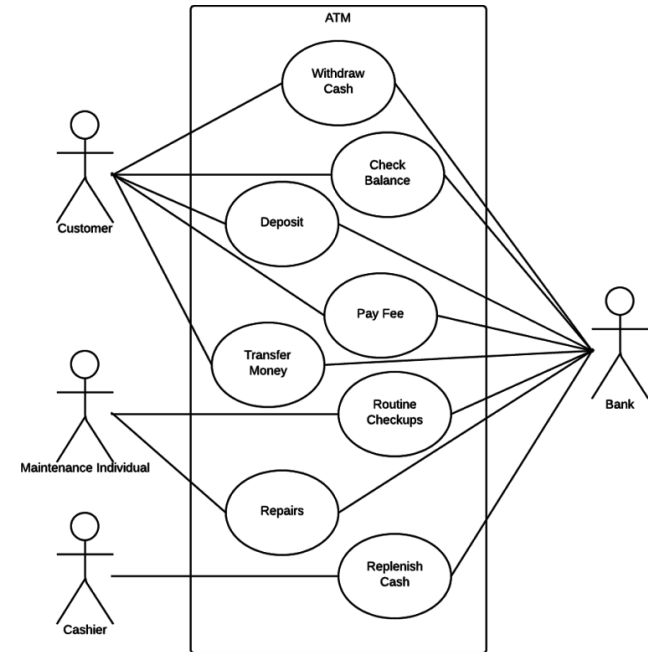
- Hvilken av de følgende dekker alle transitions?

- a) A B C D E F G
- b) A B D B C E F G
- c) A B C D E F G
- d) A B D B C E A B C E F G

# Brukstilfelletesting

## Use case

- Beskriver samhandling mellom deltagere (brukere og subsystemer)
  - Basert på faktisk sannsynlig bruk
- Har forhåndsbetingelser og avslutningstilstand
- Hovedflyt og alternativ flyt



# White box-teknikker

---

## WHITE BOX

### Strukturbasert

- Programinstruksjonstesting og dekning (statement testing and coverage)
- Forgrenings- /beslutningstesting og dekning (branch or decision testing and coverage)
- Andre strukturbaserte teknikker
  - Betingelsesdekning (condition coverage)
  - Sammensatt betingelsesdekning (multiple condition coverage)





# White box: Programinstruksjonstesting og dekning, forgrenings- /beslutningstesting og dekning

Statement testing and coverage, branch/decision testing and coverage

## Oppgave:

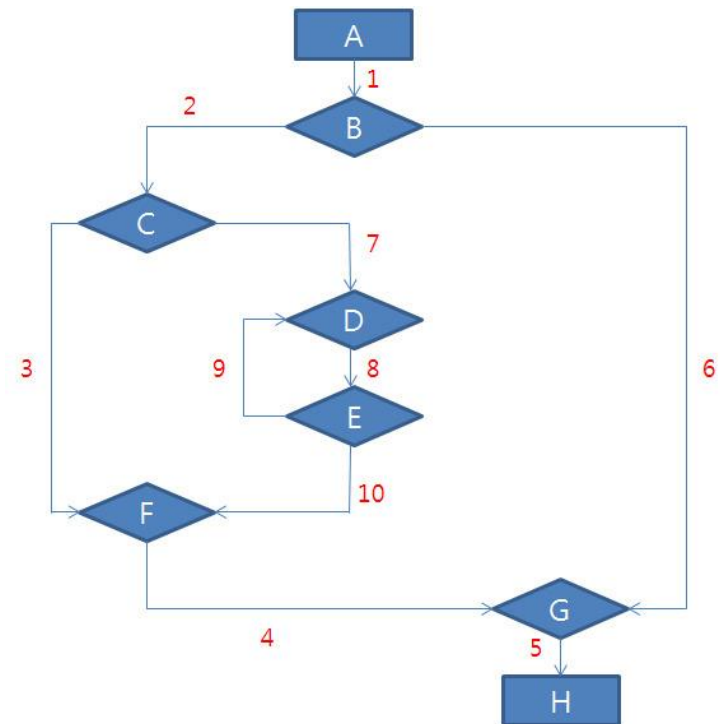
Statement coverage: *PacMan*

- Pacman skal spise alle boksene med færrest mulig forsøk.

Branch/decision coverage: *Bil*

- Bilen skal kjøre alle veiene

Hvor mange tester for statement og hvor mange for branch coverage får du?



# White box-testing: statement coverage og decision coverage

---

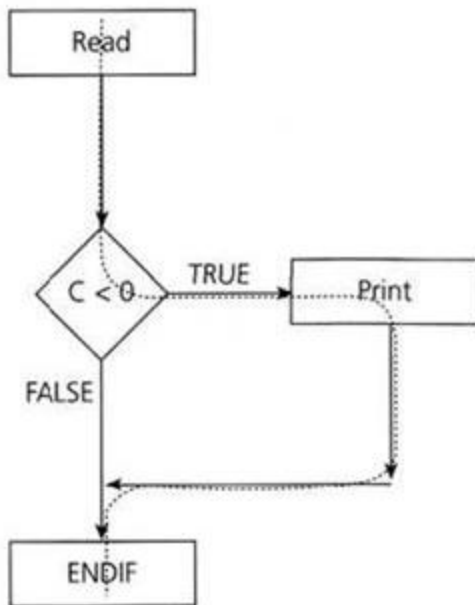
- Gitt følgende pseudokode:

**Read A**

**Read B**

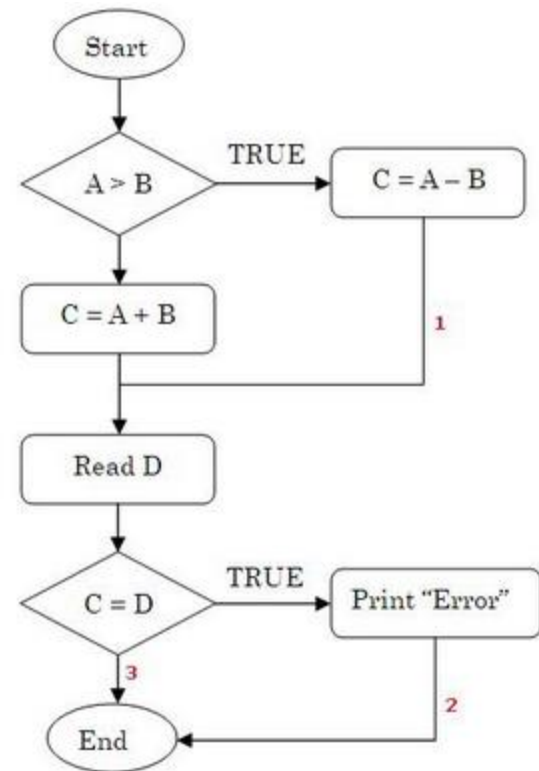
```
IF A+B > 100 THEN
    Print «Stort tall»
ELSE
    IF A+B > 50 THEN
        Print «Lite tall»
    ENDIF
ENDIF
```

1. Hva er minimum antall testcases for å oppnå 100% statement coverage?
2. Hva er minimum antall testcases for å oppnå 100% decision coverage?
3. Gi verdier for A og B som vil gi 100% statement coverage.
4. Vil de samme verdiene oppnå 100% decision coverage?



```

IF A > B THEN C = A - B
ELSE
  C = A + B
ENDIF
Read D
IF C = D Then Print "Error"
ENDIF
  
```



# Erfaringsbaserte teknikker

---

- Feilgjetting - *error guessing*
- Eksplorativ testing - *exploratory testing*



# Erfaringsbasert teknikk

---

| Error guessing                           | Exploratory testing                                   |
|--|---|
| Skrive testcase basert på erfaring.      | Timeboxed   |
| Bruke erfaring for å fremprovosere feil. | Brukes når det er begrenset tid.                      |
| Brukt etter formelle testteknikker.      | Brukes når det er mangelfull beskrivelse av systemet. |
| Strukturert tilnærming                   | Minimum med planlegging                               |
| Kan velge testcase eller ikke testcase   | Ingen testcase  |



# Valg av testteknikker

---

- Avhenger av mange faktorer: typen system, standarder, kontraktskrav, risikonivå, tid og budsjett osv.
- Noen teknikker mest egnet for spesielle situasjoner; andre benyttes på alle nivåer
- Bruker vanligvis kombinasjoner av strategier og teknikker



# Kahoot spørsmål fra kapittel 4



# Gruppeoppgaver til kapittel 4





# Ekvivalensklasser

---

- Egenandel for bilforsikring er basert på alderen til føreren. 17 til og med 25 har rate A, 26 til og med 50 er rate B, og over 50 er rate C.
- Hvilke verdier er innenfor gyldige ekvivalensklasser?
  - a) 19,23,44
  - b) 32,47,49
  - c) 17,25,55
  - d) 22,33,55

## Grenseverdier

---

- Prisen på nedlastning av filer kommer an på tiden det tar å laste ned. For mindre enn 2 minutter koster det 1NOK, filer som tar mer enn 2 og til og med 5 koster 2 NOK og nedlastningstid på lenger enn 5 minutter koster 3NOK.
- Basert på grenseverdier, hvilke tider bør bli testet?
  - a) 2:00, 2:01, 5:00, 5:01
  - b) 1:59, 2:00, 4:59, 5:00
  - c) 1:59, 2:00, 5:00, 5:01
  - d) 2:00, 2:01, 4:59, 5:00

# Grenseverdier

---

- En klubb gir bare de som er mellom 18 og 30 adgang. Når man bruker 3 grenseverdier, hvilke grenseverdier bør være med?
  - a) 18, 19, 20, 28, 29, 30
  - b) 16, 17, 18, 30, 31, 32
  - c) 17, 18, 19, 29, 30, 31
  - d) 18, 19, 21, 29, 30, 35

## Hva er IKKE RIKTIG for ekvivalensklasseinndeling?

- a) Alle ugyldige data for en input er alltid del av samme ekvivalensklasse.
- b) Inputdata blir delt i grupper der en forventer at data i en gruppe oppfører seg på lik måte.
- c) Ekvivalensklasseinndeling som teknikk kan brukes for å sikre seg at både input og output blir dekket med sin datavariasjon.
- d) Ekvivalensklasser kan finnes både for gyldige og for ugyldige data

Gitt er følgende pseudokode:

*Integer a;*

*If ( $a > 1$  AND  $a < 50$ )*

*Then ...*

*End if*

**Hvilket sett av testverdier er et resultat av en riktig utført grenseverdianalyse?**

a) 0, 1, 2, 50, 51

b) -1, 0, 1, 49, 50, 51

c) 1, 2, 49, 50

d) -32767, -1, 0, 1, 49, 50, 51, +32767

Hvilke verdier for å teste input til et felt for personnavn er et typisk resultat av feilgjetting?

---

- a) «Anna», «Åse»
- b) «George W. Bush», «Osama Bin Laden», «007»
- c) ingen verdi gitt, «Mona Åse Stöckheim-Ænesdal», et navn som er svært langt
- d) «Jon D. Olsen», et navn som er svært langt

Et felt på en skjerm skal fylles inn med et positivt heltall opp til to sifre langt. Hvilket resultat er en riktig utført ekvivalensklasseinndeling?

- a) **Ugyldige klasser** = ikke verdi gitt / ikke heltall / null eller negativt tall / over 99. **Gyldige klasser** = verdi gitt / heltall / verdi fra og med 1 til og med 99.
- b) **Ugyldige klasser** = kommatall / ikke tall / ledende nuller / ledende pluss / null eller negativt tall / over 99. **Gyldige klasser** = verdi gitt / heltall / verdi fra og med 1 til og med 99.
- c) **Ugyldige klasser** = ingen verdi gitt / kommatall / null eller negativt tall / over 99. **Gyldige klasser** = verdi gitt / heltall / verdi fra og med 1 til og med 99.
- d) **Ugyldige klasser** = kommatall / ikke tall / ledende nuller / ledende pluss / null eller negativt tall / over 100. **Gyldige klasser** = verdi gitt / heltall / verdi fra og med 2 til og med 99.

Et felt på en skjerm skal fylles inn med en tekst på opp til 10 tegns lengde. Hvilket resultat er en gyldig ekvivalensklasseinndeling?

- a) **Ugyldige klasser** = ikke gitt verdi / lengre enn 11 posisjoner. **Gyldige klasser** = verdi gitt / lengde mellom 1 og 10 tegn.
- b) **Ugyldige klasser** = ikke gitt verdi / lengre enn 10 posisjoner. **Gyldige klasser** = verdi gitt / lengde mellom 1 og 10 tegn.
- c) **Ugyldige klasser** = ikke gitt verdi / andre tegn enn a til z / lengre enn 10 posisjoner. **Gyldige klasser** = verdi gitt / lengde mellom 1 og 10 tegn / bare bokstaver fra a til z.
- d) **Ugyldige klasser** = ikke gitt verdi / lengre enn 10 posisjoner. **Gyldige klasser** = verdi gitt / en klasse for hver av lengdene fra og med 1 til og med 10



## Gitt er følgende programkode:

- *INFO: END FOR er en løkke. Alt i mellom de to stikkord utføres like mange ganger som det er elementer i tabellen som du som tester lar programmet lese).*

```
BEGIN
    INDEX := 0;
    FOR ALL ELEMENTS IN THE TABLE BELOW DO THIS:
        INCREMENT THE INDEX "INDEX" BY 1;
        IF (AMOUNT(TABLE(INDEX)) > 0)
            THEN PAY
        END IF;
    END FOR;
END
```

Hvor mange ganger må en minst starte dette programmet for 100% programinstruksjonsdekning (statement coverage)?

- a) 1
- b) 2
- c) 4
- d) mange

# Hvor mange ganger må en starte programmet minst for 100% stidekning (path coverage)?

---

- *Samme program som i forrige oppgave:*

```
BEGIN
  INDEX := 0;
  FOR ALL ELEMENTS IN THE TABLE BELOW DO THIS:
    INCREMENT THE INDEX "INDEX" BY 1;
    IF (AMOUNT(TABLE(INDEX)) > 0)
      THEN PAY
    END IF;
  END FOR;
END
```

- a) 1
- b) 2
- c) 4
- d) Mange



Gitt følgende programkode:

```
IF (A > B)
    THEN DO X;
END IF
IF (C IS True)
    THEN DO Y;
END IF
```

Hvor mange testtilfelle behøver en minst for 100% av følgende testdekninger?

- a) Programinstruksjonsdekning: 1, forgreningsdekning: 1
- b) Programinstruksjonsdekning: 2, forgreningsdekning: 2
- c) Programinstruksjonsdekning: 2, forgreningsdekning: 1
- d) Programinstruksjonsdekning: 1, forgreningsdekning: 2

# Billettpriser

---

Billettprisen er avhengig av høyden på en person: Under 1,10m er gratis. Fra og med 1,10m og inntil 1,40m er det halv pris. Over 1,40m er det full pris. Hvilket datasett dekker de gyldige ekvivalensklassene?

- a)  $0,9 - 1,2 - 1,4$
  - b)  $-1,0 - 0,9 - 1,2 - 2,0$
  - c)  $0,0 - 1,1 - 1,4$
  - d)  $1,0 - 1,2 - 2,1$
- Variant: Hvis grensene var halv pris fra og med 1,20m til og med 1,50 m, hva er rett svar da?

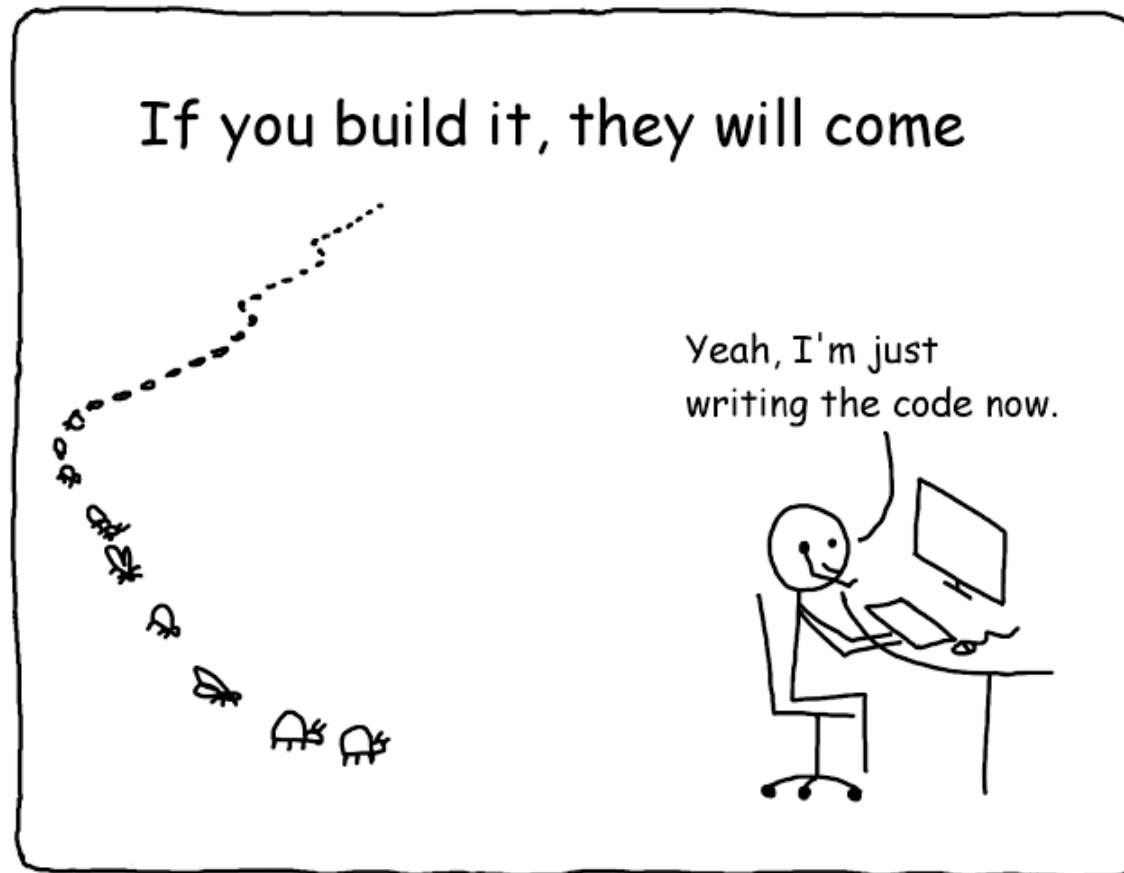
# Oppsummering kapittel 4

---

- 3 kategorier av testteknikker:
  - Black box (spesifikasjonsbasert), white box (strukturbasert), erfaringsbasert
- 7 black box-teknikker:
  - Equivalence partitioning, boundary value analysis, decision table testing, state transition testing, use case testing,
- 2 white box-teknikker:
  - Statement testing and coverage, decision testing and coverage.
- 2 erfaringsbaserte teknikker:
  - Exploratory testing, error guessing
- Hvordan velge testteknikk?
  - Kommer an på flere faktorer som risiko, systemtype, krav, modeller, kunnskap



# Kaffepause!



cartoontester.blogspot.com © 2013

# Kapittel 5: Testledelse

- Testorganisasjon
- Testplanlegging og estimering
- Overvåking og kontroll av testfremdrift
- Konfigurasjonsstyring
- Risiko og testing
- Hendelses- og feilstyring



# Testorganisasjon og uavhengighet

Ingen  
uavhengighet

Stor  
uavhengighet



Test

Utvikler  
tester på  
egen kode

Noen  
andre på  
teamet

Noen fra et  
annet team  
eller testteam

Noen fra en  
annen  
organisasjon



# Testplanlegging og estimering

---

- Testplanlegging: testleder lager overordnet testplan og i separate planer for de enkelte testnivå
  - Aktiviteter: bestemme omfang og risiko, testnivå, testobjekter, roller +++
  - Bestemme start- og sluttkriterier
- Testestimering: viktig del av testplanleggingen
  - Basert på måledata eller ekspertbasert
  - Omfanget avhenger av: produktegenskaper, utviklingsprosessens egenskaper og testresultatet.



# Overvåkning og kontroll av testfremdrift

---

- Oppfølging av testfremdrift
  - Vanlige måledata omfatter: prosent arbeid utført, informasjon om feil, testmilepæler, kostnader..
- Rapportering – hva skjedde under testing?
  - Antall feil som gjenstår
  - Eksisterende risiko
- Teststyring
  - Beslutning eller endring gjort på grunnlag av måledata

# Konfigurasjonsstyring

---

- Etablere og vedlikeholde integritet
  - Testmateriale er underlagt versjonskontroll
  - Sporbarhet mellom testobjekter
- Hva skjer ved dårlig konfigurasjonsstyring
  - Vi greier ikke reprodusere defects
  - Kan ikke gjøre rollbacks
  - Fiksede defects gjenoppstår
  - +++



# Risiko og testing

---

- Risiko: muligheten for at hendelse vil inntre samt dens uønskede konsekvenser = et mulig problem
- Kan slå ut i prosjektrisiko eller produktrisiko
- Testing kan hjelpe med å identifisere ny risiko, hvilke som burde reduseres og reduksjon av usikkerhet rundt eksisterende risiko
  - Risikobasert testing

# Hendelses- og feilhåndtering

---

- Faktiske resultater matcher ikke forventede resultater
  - Testmiljøproblemer?
  - Var forventede resultater feil?
  - Var testen gjort riktig?
  - Feil i softwaren?
  - Feil versjon av softwaren?
- Hendelsesrapporter
  - Lages under utvikling, review eller testing
  - Gi feedback
  - Gir testleder mulighet til å holde kontroll over kvalitet og fremdrift
- Verktøy: defect tracking, «bug tools»



# Oppsummering kapittel 5

---

- Hvilke nivåer av uavhengighet har vi:
  - Utviklere(?), tester på utviklingsteam, uavhengig testteam, forretningsrepresentanter, spesialister, noen fra en annen organisasjon
- To metoder for estimering:
  - Basert på måledata, eksperttilnærming
- Risiko – hva er det?
  - Mulighet for at en hendelse, fare, trussel eller situasjon vil inntre, samt uønskede konsekvenser = et mulig problem.
- To hovedtyper risiko:
  - Prosjektrisiko og produktorisiko



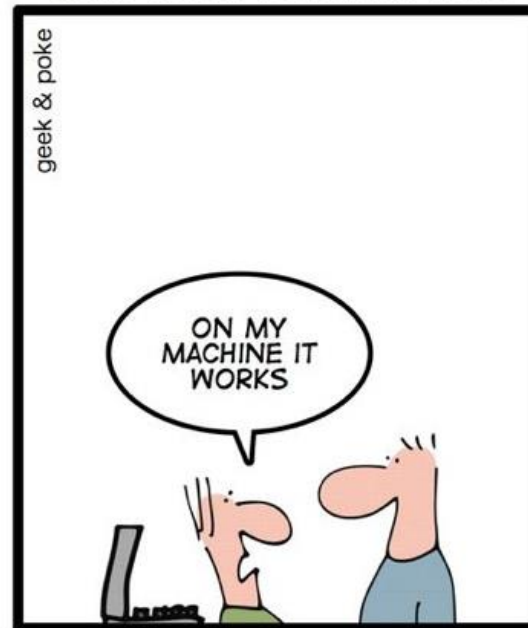
# Kahoot spørsmål fra kapittel 5



# Kaffepause!

*JUST IN CASE YOU'RE STILL NOT  
SURE WHETHER YOU'RE IN A  
SOFTWARE PROJECT*

*WAIT UNTIL YOU HEAR THIS:*





# Kapittel 6 – verktøystøtte til test

- Typer testverktøy
- Effektiv bruk av verktøy: mulige fordeler og risikoer
- Introduksjon av et verktøy i en organisasjon



# Hensikt

---

- Hensikten med testverktøy er å:
  - Forbedre effektiviteten av testing
  - Automatisere aktiviteter som er ressurskrevende manuelt
  - Automatisere aktiviteter som ikke kan utføres manuelt
  - Øke påliteligheten ved testing av komplekse manuelle oppgaver
- Noen verktøy støtter tydelig én bestemt aktivitet, andre kan støtte flere. Mange er samlet i en felles verktøypakke/suite.
- Omfatter også verktøy som brukes til monitorering (f.eks verktøy som overvåker filaktivitet i en applikasjon)
- Excelark 😊

# Klassifisering av testverktøy



# Mulige fordeler og risikoer

---

- Fordeler
  - Repetativt arbeid reduseres (f.eks kjøring av regresjonstest, oppgi testdata på nytt)
  - Bedre konsistens og repeterbarhet
  - Enkel tilgang til informasjon om tester og status
- Risiko
  - Urealistiske forventninger
  - Undervurdering av tid, kostnad, opplæring..
  - Innsats med tanke på testprosessen

# Introduksjon av et verktøy i en organisasjon

---

- Bør innføres som et pilotprosjekt
- Suksessfaktorer: trinnvis utrulling, tilpasse og forbedre arbeidsprosesser..

Hovedvurderingene i å velge verktøy omfatter:

- Organisasjons modenhet
- Proof-of-concept
- Evaluering av leverandør
- Evaluering av opplæringsbehov
- Kost-nytte

# Kahoot spørsmål fra kapittel 6



# Oppsummering kapittel 6

---

- Hva er de 6 kategoriene av testverktøy?
  - Administrasjon av test, statisk testing, testspesifikasjon, testutføring og logging, ytelse og overvåkning, spesifikke testbehov
- Fordeler ved å benytte et verktøy:
  - Redusere repeterende arbeid, konsistens, objektiv vurdering, enkel tilgang til informasjon om test
- Risiko ved bruk av verktøy:
  - Urealistisk forventning, undervurdering av innsats, for stor tillit, neglisjering av versjonskontroll
- Hva er hovedmålene med et pilotprosjekt?
  - Få erfaring, vurdere om verktøyet passer eksisterende prosesser

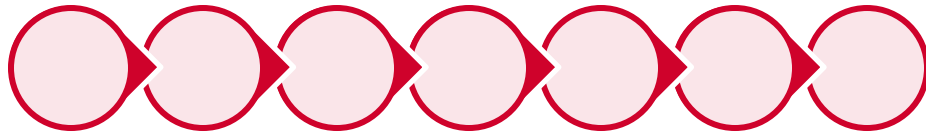
## Noen tips og råd helt til slutt..

---

- Husk at du ikke får minuspoeng for å svare feil! Det er bedre å gjette enn å ikke svare.
- Les spørsmålet nøye – vær obs på ordene *aldri* og *alltid* eller doble negasjoner
- Bruk kunnskapen du har lært for å fjerne helt uaktuelle svar først når du er usikker



# Hvordan alt henger sammen med alt



Testprinsipper (7)



Den grunnleggende testprosessen (5)

## NÅR

- Systemutviklingsmodeller (2)
- Test levels (4)
  - Akseptansetest (4)

Maintenance test (3)

## HVORDAN

- Test types (4) (ulike «targets»)

Testteknikker (3)

- BB (5)
- WB (2)
- Erfaringsbasert (2)

## HVA

- Ikke-funksjonelle egenskaper (7)

Risiko  
2 typer

# Eksamen

---

- Les dere opp på syllabus engelsk 2018 versjon .  
<https://www.istqb.org/downloads/category/51-ctfl2018.html>
- Gjør gamle eksamensoppgaver – men merk mindre endringer i 2018 pensum (Release version 2018 beskriver endringer)
  - 3 læringsmål oppdatert fra K1 til K2
  - 5 nye læringsmål (uten eksamensoppgaver)
- Satt opp i melkeveien 24 september kl 17.
  - Meld dere på liste i dag, eller send mail innen uka.
  - Info om eksamen sendt på mail fra Kristin Orskaug