Løsningsforslag/Sensorveiledning – IN3240/In4240 – vår 2022'

1 Multiple Choice

Which test values should you choose if you are going to test a speed control program with the following specifications:

- If you drive 50 km/h or less, nothing happens.
- If you drive faster than 50 km/h, but not more than 55 km/h, you will receive a warning.
- If you drive faster than 55 km/h, but not more than 60 km/h, you will be fined.
- If you drive faster than 60 km/h, you will lose your driver's license.

NB! The speed is measured in whole numbers.

Select one alternative:

- 49, 50, 54, 55, 60, 62
- 50, 51, 55, 56, 60, 61 ✔
- 0, 49, 50, 54, 59, 60
- 50, 55, 60

Maximum marks: 1

2 Mulitple Choice

Which of the following is an advantage of independent testing?

Select an alternative:

- Independent testers sometimes question the assumptions behind requirements, designs ar implementations. ✔
- Programmers can stop worrying about the quality of their work and focus on producing more code.
- The others on a project can pressure the independent testers to accelerate testing at the end of the schedule.
- Independent testers don't have to spend time communicating with the project team.

Maximum marks: 1

3 Multiple Choice

At what test level does the validation of a program take place?

Select one alternative:

- Integration test level
- Unit test level
- System test level
- Acceptance test level ✔

Maximum marks: 1

4 Multiple Choice

Which of the following test activities is least suitable for automation?

Select one alternative:

- Functional testing
- Regression testing
- Beta testing ✔
- Exploratory testing

Maximum marks: 1

5 Matching/Pairing

ISTQB distinguishes between the terms test type and test level.
Determine what is a test type and what is a test level.

Please match the values:

	Test level	Test type
Structure based testing	○	○ ✔
Non-functional testing	○	○ ✔
Acceptance testing	○ ✔	○
Functional testing	○	○ ✔
Unit testing	○ ✔	○
System testing	○ ✔	○
Testing related to changes	○	○ ✔
Integration testing	○ ✔	○

Maximum marks: 4

6 Multiple Choice

At what test level does the verification of a program take place?

Select one alternative:

- ○ System test level ✔
- ○ Acceptance test level
- ○ Integration test level
- ○ Unit test level

Maximum marks: 1

7 Matching/Pairing

Determine which test strategy is used in each of the cases.

Please match the values:

	Analytical	Consultative	Model based	Methodical
The tests are based on the systematic use of a predefined set of test conditions.	○	○	○	○ ✔
The tests are based on a state transition diagram of the product.	○	○	○ ✔	○
The tests are designed and prioritized based on risk level.	○ ✔	○	○	○
The tests are based on the views of business experts.	○	○ ✔	○	○

Maximum marks: 3

8 Matching/Pairing

The specification-based (black-box) test techniques can be used in several of the areas below. However, you must match them so that all the techniques are applied in the most suitable area.

	Use case testing	Decision table	EP and BVA*	State transition testing
Testing the interaction between user and program in a mobile application.	○ ✔	○	○	○
Testing the functionality of a check-in machine at an airport.	○	○	○	○ ✔
Testing of a bonus system based on several logical conditions.	○	○ ✔	○	○
Testing of an application that calculates the size of speeding tickets. The size of the fine depends on the size of the speeding.	○	○	○ ✔	○

** EP and BVA: Equivalence partitioning and boundary value analysis*

Maximum marks: 3

9 Multiple Choice

What is the main difference between a *walkthrough* and an *inspection*?

Select an alternative:

- An inspection is led by the authors, while a walkthrough is led by a trained moderator.
- A walkthrough is led by the author, while an inspection is led by a trained moderator. ✔
- Authors are not present during inspections, while they are during walkthroughs.
- An inspection has a trained leader, while a walkthrough has no leader.

Maximum marks: 1

10 Matching/Pairing

Determine which test activities are primarily performed by the tester and which are primarily performed by the test leader.

Please match the values:

	Tester	Test leder
Handle test automation tasks	✔	
Evaluate the test results against the exit criteria		✔
Obtain and prepare test data	✔	
Collect test results and report on test progression		✔

Maximum marks: 3

11 Matching/Pairing

What test basis is used at the different test levels? If you mean there is more than one option, you should choose the option that is used to a lesser extent at other test levels.

	Integration testing	Unit testing	System testing	Acceptance testing
Functional and non-functional requirements	○	○	○ ✔	○
Design documents for the system architecture	○ ✔	○	○	○
User requirements and user stories	○	○	○	○ ✔
Source code	○	○ ✔	○	○

Maximum marks: 3

12 Multiple Choice

Which of the following measurements is not suitable as an exit criteria?

Select one alternative:

- ○ Number of undetected errors ✔
- ○ Statement coverage
- ○ Fault density
- ○ Number of errors that have been detected but not yet corrected

Maximum marks: 1

13 Multiple Choice

Which of the following specification-based (black-box) test techniques is often used in combination with equivalence partitioning?

Select one alternative:

- Decision table testing
- Use case testing
- Boundary value analysis ✔
- State transition testing

Maximum marks: 1

14 Multiple Choice

Static analysis is used primarily ...

Select one alternative:

- in the implementation phase ✔
- during integration testing
- in the planning phase
- during acceptance testing
- during system testing

Maximum marks: 1

Terskelverdier:

A: 90
B: 80
C: 60
D: 50
E: 40
F: 0

Beslutningstabeller

a)

1. Bestem de ulike utfallene av programmet. (Aksjonene)

2. Bestem de binære betingelsene som ligger til grunn for utfallene.

Svar: 3 poeng. *1,5 for aksjonene, 1,5 for betingelsene*

1.
   - RI: Restriksjonsfri innreise
   - KH: Karantenehotell
   - HK: Hjemmekarantene
2.
   - GVP: Gyldig vaksinepass
   - PT: Positiv test
   - Barn: Personer under 18 år

b)

Sett opp en beslutningstabell som inneholder alle tenkelige kombinasjoner av de binære betingelsene. Tabellen skal inneholde en handlingsdel (aksjonsdel) som viser utfallet av hver kombinasjon.

Svar: **2 poeng,** *1 for alle kombinasjoner, 1 for aksjoner under riktig kolonne*

	R1	R2	R3	R4	R5	R6	R7	R8
GVP	true	true	true	true	false	false	false	false
PT	true	true	false	false	true	true	false	false
Barn	true	false	true	false	true	false	true	false
Aksjoner	HK	KH	RI	RI	HK	KH	RI	KH

Til sensor. Hjelp til å sjekke at tabellen over er riktig

R1: Barn med vaksinepass og positiv test må i hjemmekarantene.

R2: Voksne med vaksinepass og positiv test må på karantenehotell

R3: Barn med vaksinepass og med negativ test får innreise.

R4: Voksne med vaksinepass og med negativ test får innreise.

R5: Barn uten vaksinepass, men med positiv test må i hjemmekarantene

R6: Voksne uten vaksinepass, men med positiv test må på karantenehotell

R7: Barn uten vaksinepass og negativ test får innreise.

R8: Voksne uten vaksinepass og negativ test får karantenehotell.

c)

Reduser antall regler (kolonner) ved å forenkle (rasjonalisere) beslutningstabellen, uten å miste noen beregninger. Begrunn hvorfor du foretar forenklingen.

Svar: 3 poeng. *1 poeng for hver reduksjon, men 1 poeng trekk for manglende begrunnelse*

Reduksjonen kan gjøres på flere måter, så det er flere løsninger. Se alternativ løsning.

Alternativ 1:

	R1	R2	R3	R4	R5	R6	R7	R8
GVP	~~true~~	~~true~~	true	true	~~false~~	~~false~~	false	false
PT	true	true	false	false	true	true	false	false
Barn	true	false	~~true~~	~~false~~	true	false	true	false
Aksjoner	HK	KH	RI	RI	HK	KH	RI	KH

R3 og R4 kan slås sammen. Alderen spiller ingen rolle for utfallet.

R2 og R6 kan slå sammen. Hvis testen er positiv og den reisende er over 12 år, spiller det ingen rolle om man har vaksinepass eller ikke.

R1 og R5: Kan slås sammen. For barn spiller det ingen rolle om man har vaksinepass eller ikke, hvis testen er positiv. Resultatet blir uansett hjemmekarantene.

Alternativ 1:

	R1/5	R2/6	R3/4	R7	R8
GVP	-----	-----	true	false	false
PT	true	true	false	false	false
Barn	true	false	-----	true	false
Aksjoner	HK	KH	RI	RI	KH

Alternativ 2:

R6 og R8: Voksne får karantenehotell hvis de ikke har vaksinepass, uavhengig av om testen er positiv eller negativ.

R3 og R4 kan slås sammen. Alderen spiller ingen rolle for utfallet.

R1 og R5: Kan slås sammen. For barn spiller det ingen rolle om man har vaksinepass eller ikke, hvis testen er positiv. Resultatet blir uansett hjemmekarantene.

	R1/5	R2	R3/4	R6/8	R7

GVP	------	true	true	false	false
PT	true	true	false	------	false
Barn	true	false	-----	false	true
Aksjoner	HK	KH	RI	KH	RI

d) 2 poeng

Vurder følgende testtilfeller:

1. Voksen med gyldig vaksinepass og negativ test.
2. Voksen med gyldig vaksinepass og positiv test.
3. Barn med negativ test.
4. Barn med positiv test.

Trenger vi flere testtilfeller for å dekke forretningsreglene i beslutningstabellen? Hvis ja, spesifiser testtilfellene(e).

Svar: Her kan de enten bruke tabellen fra punkt b) eller punkt c).

Hvis utgangspunktet er b) blir svaret:

1. Voksen med negativ test uten vaksinepass. (regel 8)
2. Voksen med positiv test uten vaksinepass. (regel 6)

(Disse kan imidlertid slås sammen til: Voksen uten vaksinepass, uavhengig av testresultat.)

Hvis utgangspunktet er punkt c) dvs. en redusert tabell, kan svaret bli:

Alternativ 1: Vi mangler et testtilfelle for R8: Voksen med negativ test uten vaksinepass.

Alternativ 2: Vi mangler et testtilfelle for R6/8: Voksen uten vaksinepass, uavhengig av testresultat.

(Disse kan også slås sammen til: **Voksen uten vaksinepass, uavhengig av testresultat.)**

Forklaring:

1. Alternativ 1: dekker R3/4, alternativ 2: dekker R3/4
2. Alternativ 1: dekker R2/6, alternativ 2: dekker R2
3. Alternativ 1: dekker R3/7, alternativ 2: dekker R7
4. Alternativ 1: dekker R1/5, alternativ 2: dekker R1/5)

State transition diagram

a) Skriv ned den korteste stien gjennom tilstandsdiagrammet som samtidig er innom flest mulige tilstander. Hva er tilstandsdekningen til stien?

Dette spørsmålet er dessverre dårlig formulert, og noen kan ha misforstått det. Tanken var at man skulle finne den stien som passerte flest mulig tilstander. Hvis det var flere som passerte like mange, skulle man velge den korteste. Noen har imidlertid trodd at man skal velge den korteste stien gjennom grafen. Hvis så er tilfelle, må de får riktig på det og bruke den stien de har valgt som utgangspunkt forberegning av tilstandsdekning: Eks. Korteste sti: S1, S3, S4, S5, S9, S10, S12. Tilstandsdekning: 7/12 = 0, 58 = 58%.

NB 2! S5 kan byttes ut med S6 og/eller S12 kan byttes ut med S11 og gi like høy tilstands- og overgangesdekning.

Svar: 2 poeng, *1 for stien og 1 for beregning av tilstandsdekning.*

Det er 12 tilstander og 19 overganger. Stien som går gjennom flest antall tilstander:

S1, S2, S1, S3, S4, S5, S7, S8, S9, S10, S12.

Tilstandsdekning: 10/12 = 0,8333 = 83,3%

NB 1! Mange hadde ikke kalkulator på eksamen, men det er greit å skrive svart som en brøk eller et regnestykke.

NB 2! S5 kan byttes ut med S6 og/eller S12 kan byttes ut med S11 og gi like høy tilstands- og overgangesdekning.

b) Hver sti beskriver et brukstilfelle. Hvor mange brukstilfeller (ulike stier) trenger du for å oppnå 100% tilstandsdekning?
NB! Skriv ned stiene du trenger.

Svar: 2 poeng

Trenger 2 brukstilfeller for å oppnå 100% tilstandsdekning.

1. S1, S2, S1, S3, S4, S5, S7, S8, S9, S10, S12.
2. S1, S3, S4, S6, S7, S8, S9, S10, S11.

NB! Her finnes det flere alternative løsninger. Sjekk at stiene til sammen passerer alle tilstandene.

c) **Skriv ned den korteste stien gjennom tilstandsdiagrammet som samtidig passerer flest mulige overganger. Hva er overgangsdekningen til stien?**

Svar: 2 poeng

S1, S2, S1, S3, S4, S1, S3, S4, S5, S7, S8, S9, S10, S12.

Overgangsdekning: 11/19 = 0.579 = 57,9%

NB! S5 kan byttes ut med S6 og S12 kan byttes ut med S11 med like høy overgangsdekning.

d) Hver sti beskriver et brukstilfelle. Hvor mange brukstilfeller (ulike stier) trenger du for å oppnå 100% overgangsdekning? NB! Skriv ned stiene du trenger.

Svar: 4 poeng

Overgangsdekning: Trenger 6 brukestilfeller for å oppnå 100% overgangsdekning.

1. S1, S2, S1, S3, S4, S1, S3, S4, S5, S7, S8, S9, S10, S12.
2. S1, S3, S4, S6, S7, S9, S10, S11.
3. S1, S3, S4, S6, S8, S9, S10, S12.
4. S1, S3, S4, S6, S9, S10, S12.
5. S1, S3, S4, S5, S8, S9, S10, S12.
6. S1, S3, S4, S5, S9, S10, S12.

NB! Fint om dere sjekker at stiene er riktige

Oppgave: entry og ecit-kriterier, maks 6 poeng

Spørsmål	Svar	Poeng
Definer og forklar formålet med entry- og exit-kriterier i programvaretesting.	Entry-kriterier: **Avgjøre nå testing kan starte**, dvs. når alt er som er nødvendig er ferdigstilt slik at testingen kan starte. Exit-kriterier: Avgjør nå testingen kan avsluttes. Viktige med målbare kriterier, f.eks. ulike typer testdekning.	2
I hvilke(n) fase(r) av testprosessen bestemmes entry- og exit-kriteriene?	**Planleggingsfasen** (men kan redefineres underveis hvis spesielle forhold tilsier det)	1
Hvem er ansvarlig for dem?	**Testlederen**	1
Hvor og når blir de sjekket og av hvem?	Selv om testerne sjekker om testene passeres er det **testlederen** sjekker at exit-kriteriene er oppfylt ved avluting av hver testfase/testnivå/sprint.	2

Oppgave: Testskript, maks 6 poeng

Når vi skal kjøre tester med testutførelsesverktøy må vi skrive testskript.

Spørsmål	Svar	Poeng
Hva er et testskript?	**Et skript er serien av instruksjoner som trenges for å utføre en test,** enten testen skal utføres av (mennesker eller av) en maskin. Instruksjonene må være skrevet på et språk som utøveren forstår, f.eks. et programmeringsspråk hvis testen er automatisert	1
Hva er fordelene med generiske skript kontra lineære skrips?	Et lineært skript kan kun brukes på et enkelt testtilfelle. **Generiske skript kan brukes på flere testtilfeller**, f.eks. ved at samme skript mottar forskjellig parametere som dermed danner ulike testtilfeller.	2
Hva kjennetegner henholdsvis *datadrevet* skripting og *nøkkelorddrevet* skripting? Hva er forskjellen mellom de to skriptingteknikkene?	**Datadrevet skript**: Skiller testdata fra skriptet: Bruker samme skript med forskjellige inputverdier til å lage ulike testtilfeller **Nøkkelorddrevet skript**: Skriptet tar imot nøkkelord og eventuelle inputverdier som parametere. Dette muliggjør at man ikke bare kan variere inputverdiene (slik som datadrevet), men også kan utføre ulike type tester ut fra nøkkelordene.	3

- **Data-driven approach**: separates out the test inputs (the data) and uses a more generic script that can read the test data and perform the same test with different data.

- **In a keyword-driven approach:** the spreadsheet contains keywords with the actions to be taken (also called action words), and test data. Testers can then define tests using the keywords.

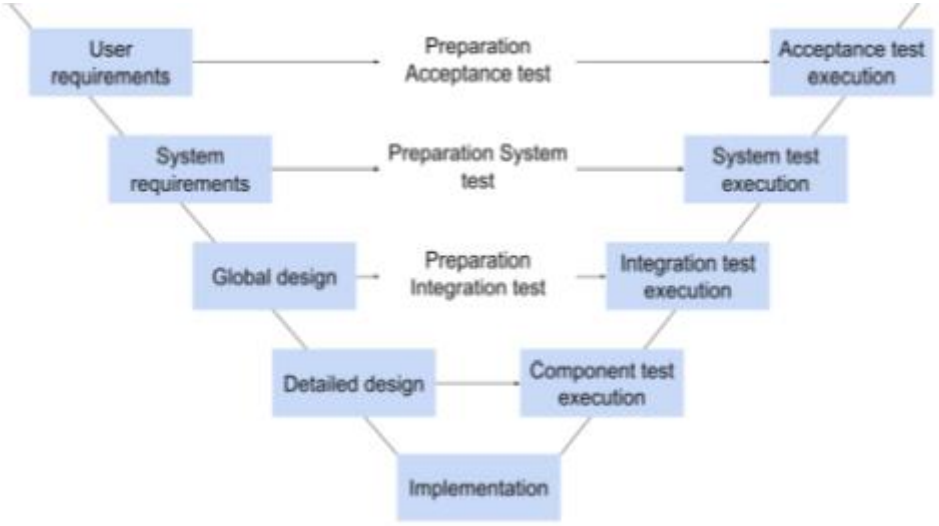
Oppgave: Statiske og dynamiske testteknikker, maks 6 poeng

Spørsmål	Svar	Poeng
Forklar hovedforskjellen mellom statiske og dynamiske testteknikker	Statisk testteknikker: Brukes til å utføres tester som **ikke krever at programmet under test kjøres**. Dynamiske testteknikker: Brukes til å lage tester som **krever at programmet under test kjøres**.	2
Hvor og når i utviklingsprosessen brukes de?	Statiske: • Reviewer: i planleggings- og designfasen, dvs. før implementasjonen • Statisk analyse: Under implementasjon av koden. Dynamiske: • Under enhetstesting/implementasjonsfase • Integrasjonstesting/implementasjonsfase • Systemtesting • Akseptansetesting	2
Er det tilstrekkelig å bare velge en av teknikkene, eller bør vi velge begge?	**Teknikkene er komplementære**. Dvs. de finner forskjellige typer feil og man bør derfor utføre både statisk og dynamisk testing.	2

NB! Svarene skal begrunnes

Oppgave: V-modellen, maks 6 poeng

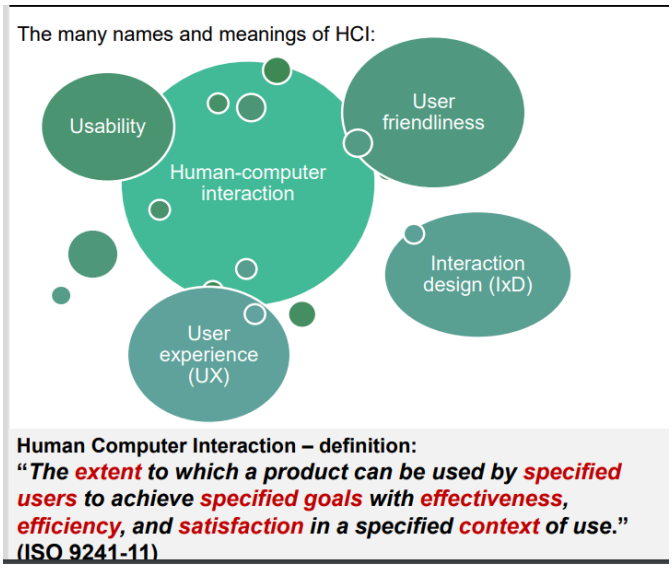
REF: Boka kapittel 2.1 side 37 Tegn en figur av **V-modellen** som viser de forskjellige testaktivitetene og de ulike testnivåene.



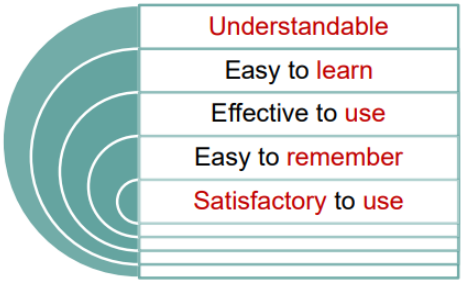
Spørsmål	poeng
Riktig venstreside	2
Riktig høyreside	2
Riktige horisontale piler	2

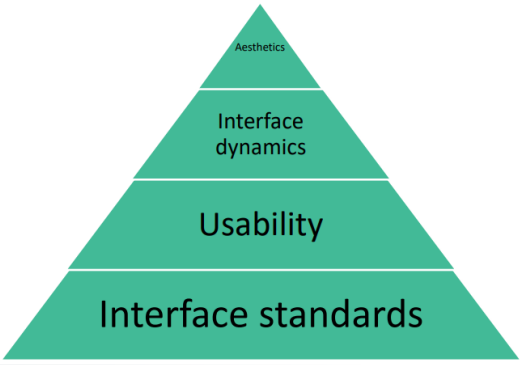
Oppgave: HCI

Spørsmål	Svar	Poeng
Hva står forkortelsen HCI for?	**Human Computer Interaction** Menneske maskin interaksjon	1
Hva er hensikten med HCI og hvordan er begrepet definert?	Def se under. Trengs ikke å kunnes ordrett.: *I hvilken grad et produkt kan brukes av spesifisert brukere for å oppnå spesifiserte mål med effektivitet, effektivitet og tilfredshet* ***i en spesifisert brukssammenheng."*** **Hensikten:** Øke kvaliteten på programvaren ut fra brukerens perspektiv. • Forståelig • Lett å lære • Effektiv å bruke • Lett å huske • Tilfredsstillende å bruke	2,5
Gjør rede for de ulike delene av **HCI-rammeverket**.	• Grensesnittstandarder • Brukervennlighet • Grensesnittdynamikk • Estetikk De trenger ikke å skrive så mye om hver.	2,5



The purpose of HCI testing is to make a software system:





Interface standards are constituted by:

- Best practices
- Consistent behavior and design
- Decrease work load
- Faster development

These standards have to be followed for both user interfaces and also APIs.

Usability means:

- Effectiveness
- Efficiency
- Satisfaction

Note: The key is to understand the target users and their needs and create a user-centric design.

An interface (whether visual or API) has to be designed in such way that it is:

- Responsive and fast
- Adaptable to the users needs and context
- Empowering the user
- Captivating

Aesthetics

- Responsible for the first impression
- Modern, fresh, appealing design
- Recognition of a company's applications
- A company's graphical profile

Essay:
ISTQB's test principles, maks poeng 25

For hvert prinsipp

Svar	Poeng
Navn på prinsippet (nr ikke viktig)	1
Forklare prinsippet ut fra • bakgrunn • hensikt	1
Redegjøre for hvilke **konsekvenser** prinsippet får for ulike **testaktiviter** knyttet til testplanlegging og testgjennomføring gjennom programmets livssyklus.	1

3*7 = 21

Hensikten bak alle prinsippene er å bevisstgjøre testeren om testingens muligheter og begrensninger og andre forhold som har betydning for testutøvelsen.

Prinsipp 1. Testing kan ikke bevise at koden er feilfri

Testing shows the presence of defects, not their absence

Forklaring: Selv om du tester koden aldri så mye, ja helt til du ikke lenger finner flere feil, er det likevel ikke noe **bevis for at koden er feilfri**! Sannsynligvis er det uavdekkede feil der som du ikke har oppdaget ennå.

Konsekvens: Din oppgave er å redusere sannsynligheten for at feil oppstår og ikke minst minimere konsekvensene de eventuelt vil få. Vær derfor forberedt på at feil kan dukke opp lenge etter at vår jobb er ferdig, også etter at programmet er tatt i bruk. Som følge av dette kan vi implementere rutiner som fanger opp feil som sluttbrukerne oppdager også etter at programmet er tatt i bruk.

Prinsipp 2. Fullstendig testing er umulig

Exhaustive testing is impossible

Forklaring: Å teste et program i alle dets tilstander er i praksis en uoverkommelig oppgave. Selv med ubegrensede ressurser og nok av tid vil det, bortsett fra helt trivielle programmer, være umulig å teste alt! Årsaken er at det oppstår det vi kaller kombinatorisk eksplosjon. Antall tilstander et program kan befinne seg i henger sammen med antall variabler i programmet og de ulike verdiene disse kan ha. Dette antallet trenger ikke være spesielt høyt før det i praksis er umulig å teste alt.

Konsekvens: Løsningen ligger i å velge den mest hensiktsmessige delmengden av alle eksisterende testtilfeller. En god tommelfingerregel er å **prioritere** områder med høyest risiko. I tillegg er det lurt å bruke systematiske testdesign teknikker, f.eks. spesifikasjonsbaserte testteknikker (black-box-teknikker) slik at man sikrer seg beste mulig testdekning i forhold til de ressursene man har til rådighet.

Prinsipp 3. Tidlig testing sparer tid og penger

Early testing saves time and money

Forklaring:
Hvis vi får fjernet flest mulig feil tidlig i utviklingssyklusen vil vi spare både tid og kostnader forbundet med å avdekke og rette dem opp på et senere tidspunkt. Dette er både veldig **tids- og kostnadseffektivt** samtidig som det øker kvaliteten på programmet. Totalt sett avdekkes flere feil og mangler enn tilfellet er hvis vi starter testingen på et senere tidspunkt.

Konsekvens: Ved å bruke statiske testingsteknikker, som f.eks. ulike typer reviews, kan vi kvalitetssikre programmets kravspesifikasjon og andre dokumenter som danner grunnlaget for implementasjonen, og derved forhindre at feil og mangler forplanter seg til programmet.

3.4 Prinsipp 4. Feil samler seg i klynger

Defects cluster together

Forklaring: Programmeringsfeilene er ikke spredd jevnt utover koden. De har imidlertid en stygg tendens til i å samle seg klynger. Undersøkelser har vist at omtrent 80% av feilene befinner seg i ca. 20% av koden. Hva årsaken til dette er, kan vi bare spekulere i, men det vi vet er at gitte moduler er utviklet under samme betingelser. De kan være utviklet under samme tidspress, samme kompetanse eller mangel på kompetanse, samme motivasjon osv. Det faktum at enkelte moduler er mer kompliserte enn andre, spiller også inn.

Konsekvens: Vissheten om at feilene har en tendens til å samle seg i klynger kan hjelpe oss **å fokusere testinnsatsen om de modulene der vi allerede har oppdaget feil**. Et statisk analyseverktøy, som analyserer koden uten å kjøre den, vil kunne være et nyttig hjelpemiddel i jakten på områder med dårlig kodekvalitet og/eller unødig høy kompleksitet. Slike områder bør vi gå nærmere etter i sømmene.

3.5 Prinsipp 5. Pass på sprøytemiddel-paradokset

Beware for the pesticide paradox

Forklaring: Begrepet sprøytemiddelparadokset ble introdusert på 1990-tallet av amerikaneren Boris Beizer. Han observerte likheter mellom programvaretesting og bruk av sprøytemidler i frukt- og grønnsaksproduksjon. I begynnelsen er gjerne sprøytemiddelet effektivt i kampen mot uønskede skadedyr. Men brukes det jevnlig over tid, vil effekten avta, enten fordi skadedyrene tilpasser seg

middelet (blir immune), skadedyrenes naturlige fiender også blir rammet, eller fordi det kommer nye skadedyr til som middelet ikke virker mot. Det samme fenomenet opplever vi i programvaretesting. **Bruker vi de samme testene om og om igjen, vil de etter hvert finne færre og færre feil, for til slutt ikke å finne noen**. Dette kan skape falsk trygghet. Vi forledes til å tro at koden er feilfri, mens sannheten er at vi ikke lenger har tester som er i stand til å finne feilene.

Konsekvens: Skal vi overvinne pesticid-paradokset, **er det viktig å variere testene, testverdiene og ikke minst hva som testes.** Det finnes mange ulike testingsteknikker som komplementerer hverandre. Det er derfor viktig å variere mellom de ulike teknikkene, samt kombinere dem slik at den samlede effekten gir best mulig testdekning.

3.6 Prinsipp 6. Testing er avhengig av konteksten

Testing is context dependent

Forklaring: Hvor stor feiltetthet kan vi akseptere når programmet tas i bruk? Det vil avhenge av hva programmet skal brukes til. Når liv står på spill, kan du ikke overlate til sluttbrukeren å oppdage feilene. Der risikoen er høy, som i programvare for fly eller medisinsk overvåkingsutstyr, må testingen utføres så grundig som over hode mulig. Prosjektet må tilføres de nødvendige ressurser i form av penger, tid og ikke minst menneskelig kompetanse for å sikre at ikke liv går tapt. For en mobilapp, som lett lar seg oppdatere, forholder det seg annerledes. Her kan du til en viss grad la sluttbrukeren av programmet avdekke de siste feilene, så lenge gevinsten ved å ha programmet i drift er større enn risikoen knyttet til feilene. Å teste alle prosjekter på samme måte uten å ta hensyn til risikoen, vil være feil bruk av ressursene.

Konsekvens: Testaktivitetene og omfanget av testingen må tilpasses programmets kontekst som igjen bestemmes av

- prosjektets størrelse
- prosjektets seriøsitet
- risiko knyttet til prosjektet
- utviklingsmetodikk
- ressurser
- …

Et lite studentprosjekt som skal utføres i løpet av et semester, vil testes forskjellig fra et stort prosjekt i milliardklassen som skal gå over flere år.

3.7 Prinsipp 7. Feilslutning vedrørende fravær av feil

Absence-of-error is a fallacy

Forklaring: Selv om et program er aldri så grundig testet og alle avdekkede feil er rettet, vil det aldri bli en suksess, med mindre det tas i bruk etter hensikten og oppfyller brukerens behov og forventninger. Den avgjørende testen gjøres av den eller de som skal bruke programmet.

Det er nærliggende å tro at et program som er testet og funnet i overensstemmelse med de krav og spesifikasjoner som er lagt til grunn er en suksess. Men det er ikke nok. **Det er til syvende og sist sluttbrukerne som avgjør om programmet er vellykket**. Om programmet ikke blir tatt i bruk, kan vi

neppe betegne det som en suksess. Det er mange produkter som av ulike årsaker aldri ser dagens lys. Det kan være produkter som brukerne ikke etterspør lenger, som er for vanskelig å bruke, som tiden har løpt fra og eller produkter som har priset seg ut av markedet. Vi har sett eksempler på dataprogrammer som det har tatt så lang tid å utvikle at når de endelig lanseres har nye teknologiske løsninger og plattformer overtatt.

Konsekvens: God kontakt gjennom hele utviklingsforløpet med markedet**, sluttbrukeren** og miljøet programmet skal brukes i, kan vi bidra til at programmet blir en suksess istedenfor en fiasko.