Pålitelighet og Tilgjengelighet i Programvaresystemer

Tor Stålhane

IDI / NTNU

Mål og midler

Husk:

Det er safety som er målet.

Pålitelighet og tilgjengelighet er bare midler til å nå målet.

Hva er pålitelighet – 1

Fra IEEE Standard Glossary – Std 610:

Reliability. The ability of a system or component to perform its required functions under stated conditions for a specified period of time.

Fra IEC 61508:

Safety. Freedom from unacceptable risk

Hva er pålitelighet – 2

Viktig å være klar over at pålitelighet er relatert til:

- Krav – ikke behov. Du får det du ber om
- Spesifiserte omgivelser – ikke i alle omgivelser. Endrede omgivelser kan endre påliteligheten
- En spesifisert tidsperiode – ikke «for alltid»

Hva er tilgjengelighet – 1

Fra IEEE Standard Glossary – Std 610:

Availability. The degree to which a system or component is operational and accessible when required for use. Often expressed as a probability.

Enkelt sagt: Hva er sannsynligheten for at systemet er tilgjengelig på et gitt tidspunkt?

Hva er tilgjengelighet – 2

$$MTTF = \frac{total\ brukstid}{antall\ \ feilhendelser}$$

To feil på et år gir
MTTF = 4380 timer

$$MTTR = \frac{total\ reparasjonstid}{antall\ reparasjoner}$$

To reparasjoner på hhv. 12 og 3 timer gir MTTR = 7.5 timer

$$\text{Tilgjengelighet} = \frac{MTTF}{MTTF + MTTR}$$

Tilgjengelighet = 0.998
Tilsvarer 17.5 timer nedetid pr. år

Hvorfor er pålitelighet og tilgjengelighet viktig – 1

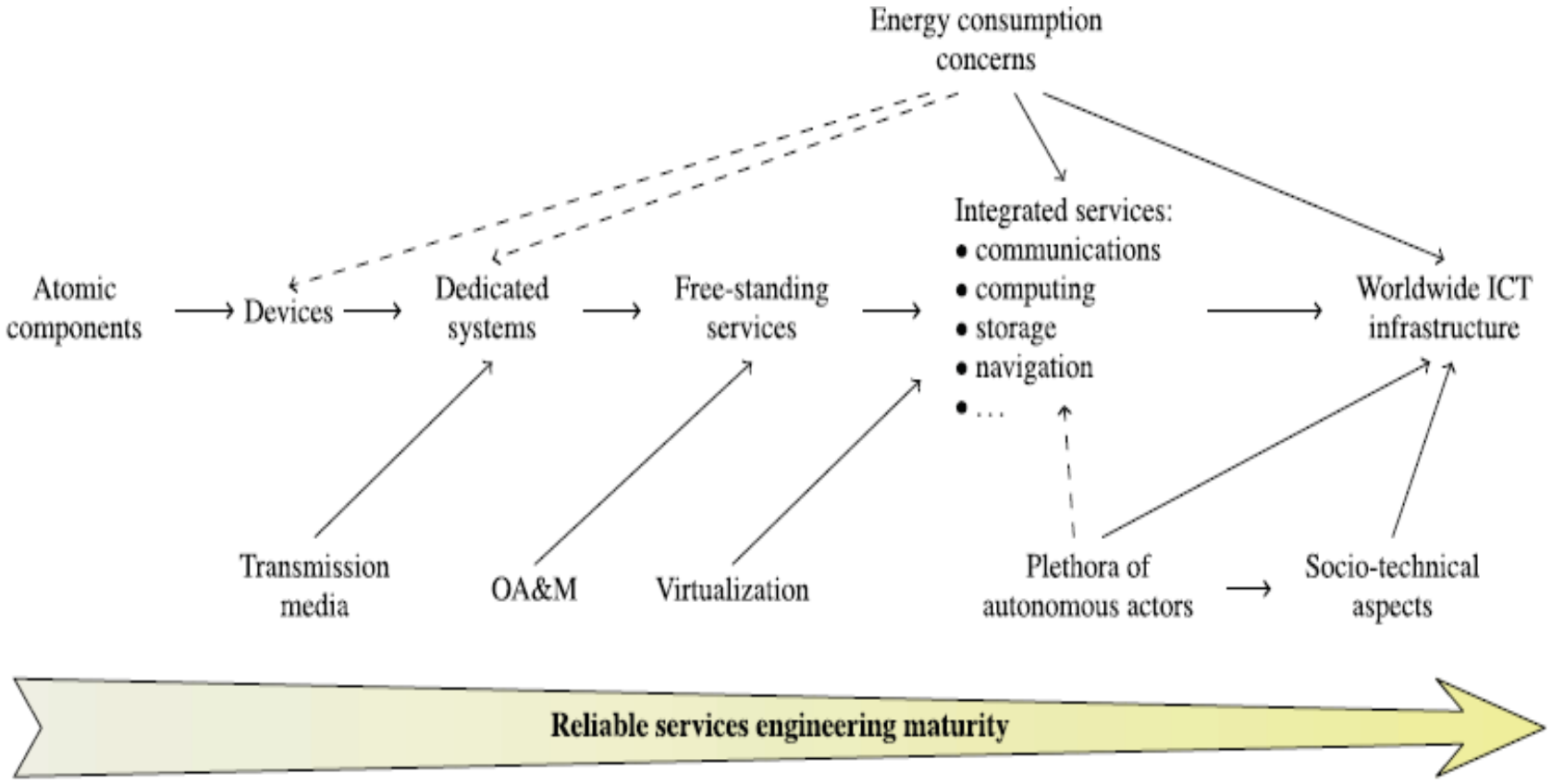
- Pålitelighet er viktig for at vi skal få et system som virker slik vi vil at det skal virke
- Tilgjengelighet er viktig fordi systemet må være tilgjengelig når vi har bruk for det.
- Utilgjengelig eller upålitelig system =>
  Manglende eller feil informasjon =>
  Feil beslutninger =>
  Fare for liv og helse

Hvorfor er pålitelighet og tilgjengelighet viktig – 2

Det er ikke bare de systemene som gjør spesifikke oppgaver – e.g. analyse av blodprøver – som er kritiske.

All informasjon som påvirker medisinske beslutninger er kritisk

Hvor gode er vi på hva



WELL-INTEGRATED:
- good engineering practice
- integrated and targeted research

'FUMBLING' REGION:
- poor understanding of underlaying phenomena
- no firm engineering practice
- research is scarce
- efforts not targeted

Hvor er problemene – 1

Datanett er – isolert sett – et relativt robust medium for kommunikasjon.

Det er når vi ser på ***totalsystemet*** at tallene blir litt uheldige.

Mange system-komponenter => mange muligheter for feil.

- DnBs nettløsning har MTTF på 0.18 år så langt i 2014
- Norsk Helsenett hadde en MTTF på
  - 0.20 år i 2008
  - 0.06 år i 2007

Hvor er problemene – 2

Viktige problemer:

- Plunder og heft – stress for personalet
- Manglende data, feil data

=> feil beslutninger => fare for liv og helse

Etter at systemet er opp igjen må man finne ut

- Hvorfor systemet gikk ned
- Hva man kan gjøre for at det ikke skal gjenta seg

Komponenter vs. Systemer – 1

$$\lambda = \sum_{i} \lambda_i$$

$$\lambda = \frac{1}{\mathrm{MTTF}}$$

Antar konstante feilrater og ingen duplisering av komponenter =>

- Feilraten (λ) er det inverse av MTTF
- Feilraten for hele systemet er lik summen av feilratene til de enkelte komponentene

Komponenter vs. Systemer – 2

ComputerWorld – Oddbjørn Schei:

- «Det er helt nødvendig med redundans. Derfor skal vi ha tre fiber til hvert senter»
- Alle tre forbindelsene vil imidlertid terminers i samme rom og ha felles strømforsyning og kjøleanlegg => gode muligheter for fellesfeil

Like enheter uten reparasjon:

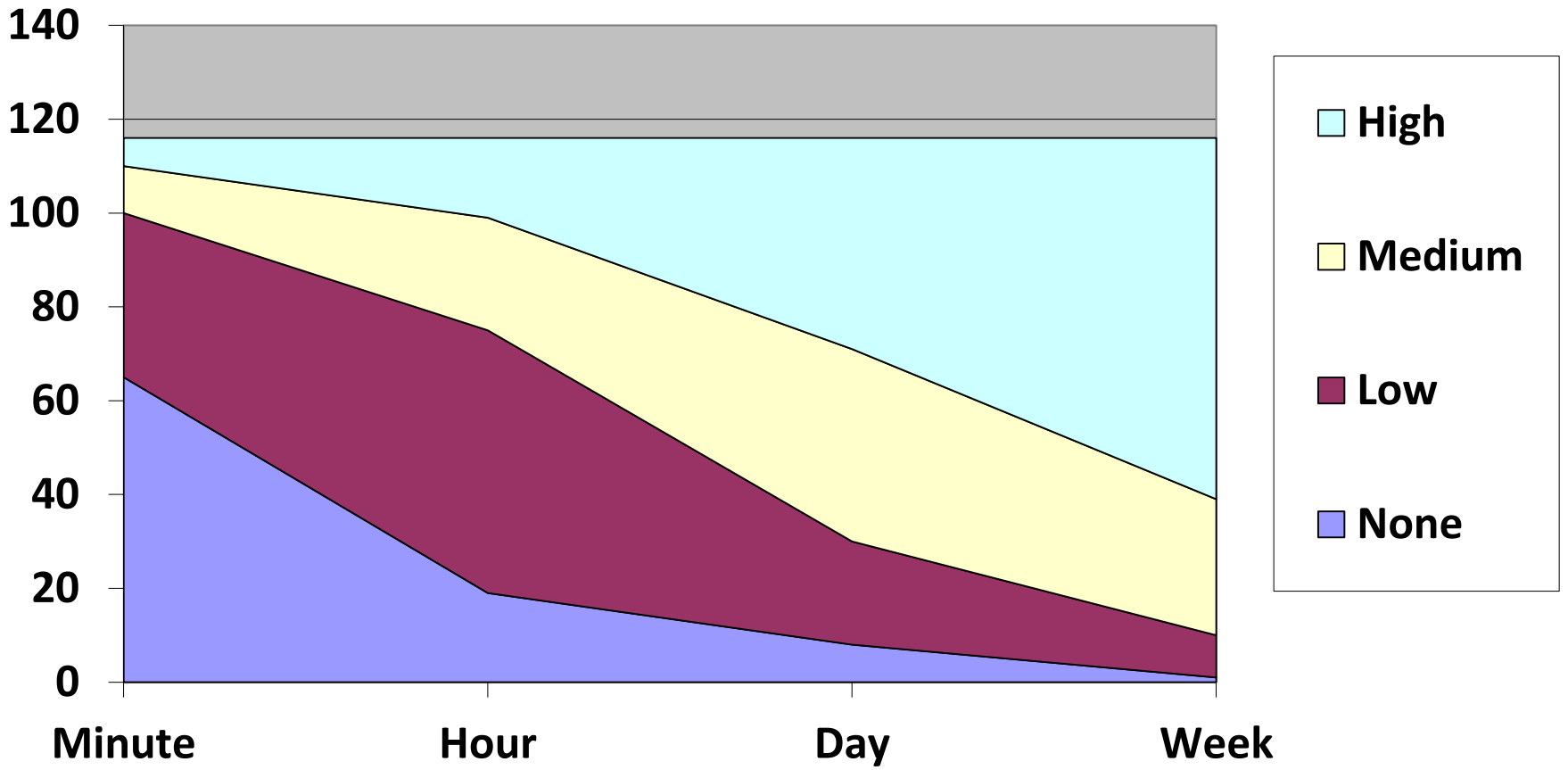
$$\mathrm{MTTF}_{\mathrm{tot}} = \frac{1}{\lambda}\sum_{i=1}^{N}\frac{1}{i} = \mathrm{MTTF}\sum_{1=1}^{N}\frac{1}{i}$$

Pålitelighet, tilgjengelighet og standarder

ISO 62304 og ISO 14971 nevner hverken pålitelighet eller tilgjengelighet. Vi må derfor:

- vurdere hva som kan skje viss systemet er nede
  - 10 minutter
  - 1 time
  - Ett døgn
  - Mer enn et døgn
- sette krav til system / leverandør / utviklere i henhold til konsekvensene

Eksempel på kritikalitetsanalyse



Krav til tilgjengelighet og pålitelighet – 1

- MTTR => hvor lang tid skal det ta å
  - Få all komponentene i gang igjen
  - Gjenopprette databaser
  - Gjenopprette alle forbindelser
- MTTF => systemet må
  - Være robust – ikke gå ned ved feil input e.l. =>
    - Defensiv programmering
    - «Graceful degradation»
  - Ikke ødelegge data, hverken i PC, servere eller i DB

Krav til tilgjengelighet og pålitelighet – 2

For å få leverandørene til å levere det vi trenger må vi:

- Stille **kontrollerbare** krav til produkt og prosess
- Følge opp **prosessen** – vi må kontrollere at de gjør det vi ble enige om – QA
- Sjekke **resultatet** – har vi fått et produkt med de egenskapene vi ba om – FAT og SAT

Vær nøye på hva du ber om – 1

- **Ikke si**: Systemet skal ha høy tilgjengelighet
- **Si**: Systemet skal ha en tilgjengelighet bedre enn 0.9999

$$\text{Tilgjengel ighet} = \frac{MTTF}{MTTF + MTTR}$$

OBS: Det er to måter å få høy tilgjengelighet på :

- Høy MTTF - bra
- Lav MTTR – ikke fullt så bra

Vær nøye på hva du ber om – 2

- Alt. 1: MTTF = 24 timer, MTTR = 1 minutt => Tilgjengelighet = 0.9993
- Alt. 2: MTTF = 1 år, MTTR = 6 timer => Tilgjengelighet = 0.9993

De to eksemplene har samme tilgjengelighet, men de fleste vil nok foretrekke alternativ 2.

Derfor: sett krav til MTTF og MTTR

Vær nøye på hva du ber om – 3

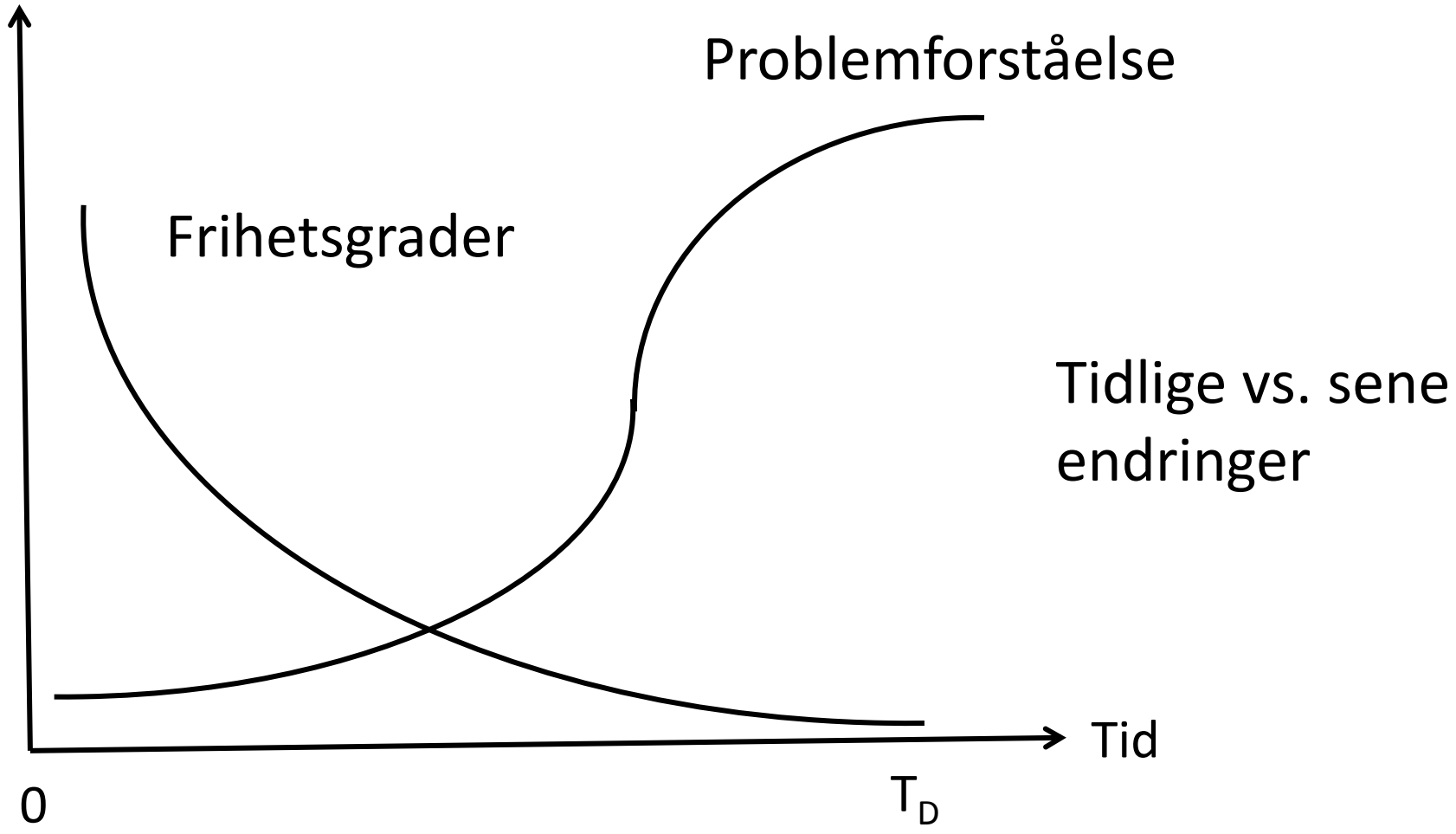
Mange prosjekter har en lang "hale" der vi endrer produktet

FRA: det vi ba om da det startet

TIL: det vi egentlig har bruk for

Slike sene endringer kan ødelegge både pålitelighet og tilgjengelighet =>
Bruk en smidig (agile) utviklingsprosess

Vær nøye på hva du ber om – 4



Hvordan kan vi følge opp kravene – 1

Det viktigste vi kan gjøre er å sørge for at krav til MTTR og MTTF er med i planene fra dag 1. Vi kan ikke legge det på til slutt i prosessen.

To spørsmål ved viktige beslutninger:

- Vil denne beslutningen påvirke MTTR eller MTTF?
- Viss «Ja», hvordan skal vi kontrollere produkt og prosess slik at
  - Det blir gjort slik vi var enige om
  - Systemet oppfører seg som planlagt

Hvordan kan vi følge opp kravene – 2

Testing er viktig. Vi må teste at

- Riktig input gir riktige resultater
- Feil input ikke «dreper» systemet, men
  - Gir informative feilmeldinger
  - Ikke ødelegger eksisterende data
  - Gir brukeren mulighet til å rette opp feilen og fortsette med oppgaven

Hvordan kan vi følge opp kravene – 3

I tillegg til programvarene er også viktig å teste

- Driftsorganisasjonen – e.g. reaksjonstid, service og reparasjoner (SLA)
- «Stand by» løsninger for maskinvare og nettverk
- Hele systemet i en pseudo-reell driftsituasjon – scenariotesting

Kan vi estimere pålitelighet

Vi vil raskt se på noen metoder:

- Flere, uavhengige testgrupper for å estimere
  1. Antall feil basert på testresultatene
  2. MTTF basert på estimat for feiltetthet
- Estimere basert på tester
  - Feilsannsynlighet
  - MTTF

Flere, uavhengige testgrupper

To uavhengige testgrupper.

- Den første gruppa finner M feil
- Den andre gruppa finner n feil
- m feil blir funnet av begge grupper.

$$\mathbf{m / n = M / N_0 \Rightarrow N_0 = M*n / m}$$

$$\mathbf{Feiltetthet = N_0/KLOC}$$

Dette kan brukes til å gi et estimat av MTTF

Estimering av pålitelighet basert på feiltetthet

Error content N_0 / KLOC	Approximate MTTF
> 30	1 minute for common operations
20 – 30	4 – 5 minutes for common operations
10 – 20	1 hour for common operations A few minutes in unusual situations
5 – 10	Several hours for common operations
2 – 5	25 hours for common operations
1 – 2	Several days for common operations
0.5 – 1	1 month for common operations
0.0 – 0.1	Indefinite. Continuous operation possible

Alternativer – mye testing

Estimere feilsannsynlighet

- Feilsannsynlighet: p
- Antall tester uten feil: M
- Signifikansnivå: α

$(1 - p)^M = \alpha \Rightarrow$

$M = -\ln(\alpha) / p$

Eksempel: $\alpha = 0.05$, $p = 0.001$
=> M = 3000

Estimere MTTF

- Total test-tid: TTT
  (test-tid * antall testere)
- Antall feil før testing: N_0

MTTF = e TTT / N_0

Eksempel: N_0 = 10, TTT = 100
=> MTTF = 27 dager

Kan det bli HELT sikkert

Det kan aldri bli helt sikkert, men det finnes alltid strategier for å gjøre det sikrere.

- Dupliser alle nettforbindelser og maskiner med dupliserte løsninger på forskjellige steder – i det minste i forskjellige bygning.
- Ha alternative løsninger – e.g. mobiltelefon / fasttelefon og manuelle rutiner. For at en alternativ løsning skal fungere må det øves – ofte.

Hvordan kan NTNU bidra

NTNU har flere miljøer som kan gi nyttige bidrag til pålitelighet, tilgjengelighet og safety:

- IDI – Programvare pålitelighet og sikkerhet
- NSEP
- Service kvalitet (QoS)
- Fyrtårnprosjektet OADE (Open Autonoumous Digital Ecosystems)

Noen konklusjoner

- Det kan aldri bli helt sikkert
- Feilraten for programvare vil dominere
- Målet er safety. Pålitelighet og tilgjengelighet er bare midler for å nå målet.
- Tilgjengelighet og pålitelighet er to sider av samme sak
- Krav må
  - stilles til tilgjengelighet (produkt og prosess) basert på risikoanalyse
  - følgens opp under hele utviklingsløpet
- Må definere hva vi vil gjøre når systemet er utilgjengelig – alternative løsninger.