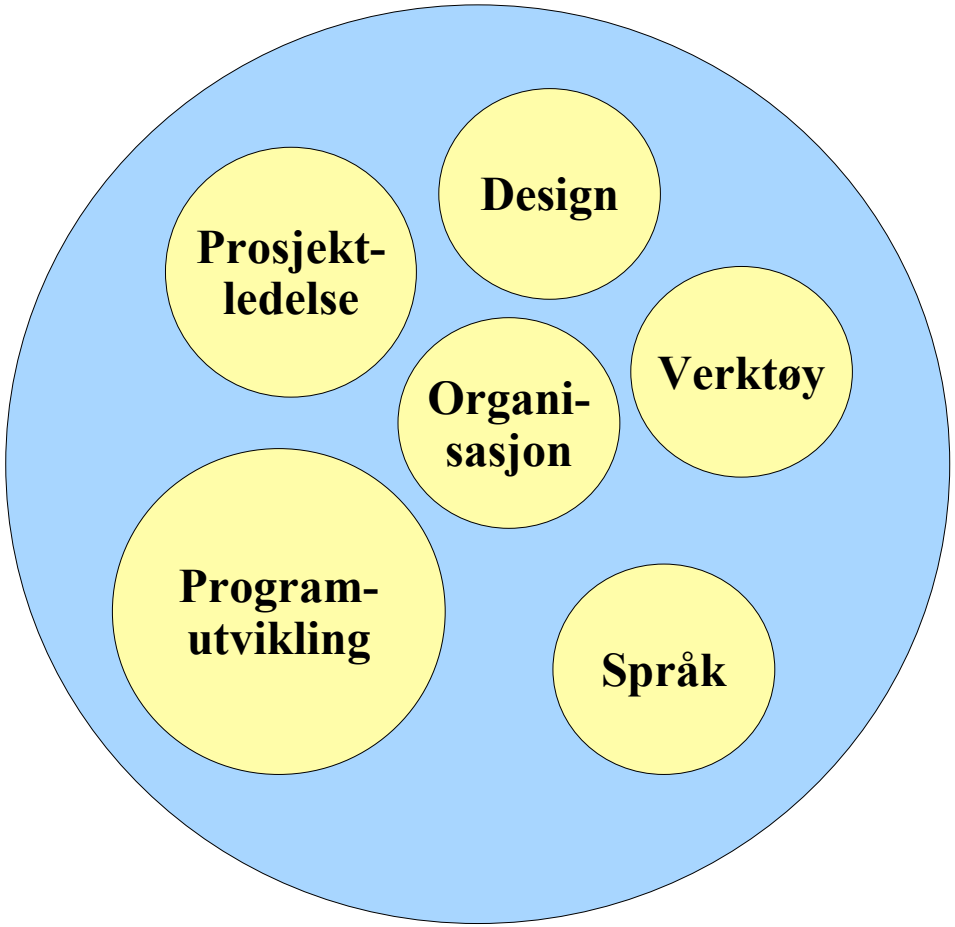
# Kvalitet og programvareutvikling

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Kurset



Universitetet i Stavanger

# Lærebok og forelesninger

- Kvalitet og programvareutvikling
  - Tor Berg Hansen og Grete Hjerttø
  - Gyllendal  ISBN 82-05-31143-9

- Forelesingene
  - Jeg foreleser noe
  - Det legges litt opp til diskusjoner
  - Dere foreleser noe
- Oppgaver
  - Sammen prøver vi å løse noen av oppgavene i læreboken

# Timeplan

- Forelesninger
  - Mandag kl 12.15 – 14.00 D - 205
  - Onsdag kl 10.15 – 12.00 D - 206
  - Fredag kl 08.15 – 10.00 E - 354

# Læreboka er delt i 6 hovedeler

- Del 1 kap. 1 – 3 Kvalitetsteori
- Del 2 kap. 4 – 6 Standarder for kvalitetsystemer
- Del 3 kap. 7 – 8 Prosess og prosessforbedring
- Del 4 kap. 9 – 15 Livsløpsprosesser og livsløpsmodeller
- Del 5 kap. 16 – 20 Støtteprosesser
- Del 6 kap. 21 – 22 Hjelpemidler for prosessforbedring

# Kan kvalitet måles?

- Kvalitetsprodukt
  - Fornøyde med produktet
  - Vi er glade for å ha funne fått tak på produktet
  - Hvor mange er fornøyde med kvaliteten på mobilen

Def. av kvalitet

- Produktet har egenskaper jeg liker
- Produktet er robust, uten feil, har lang levetid.
- Produktet er luksuriøst lekkert

# ISO om kvalitet

**ISO 8402 1994**

Helhet av egenskaper og kjennetegn ved et produkt eller tjeneste, som vedrører dets evne til å tilfredsstille faste krav eller behov.

**ISO 9000:2000**

**Kvalitet** – i hvilke grad en samling iboende egenskaper oppfyller krav

- Hvilke krav har utviklingsmiljøet?
- Hvilke krav har brukerne?

***Kvalitet betyr å oppfylle krav!***

Universitetet i Stavanger

# Behov og forventninger

- Behov
  - Fravær  av noe som er krevd, ønskelig og nyttig
  - Objektivt definerbart
  - Uformell, samtale, skisse
- Forventninger
  - Håp, formodning, antagelser om framtiden
  - Subjektivt og av og til underforstått, ikke erkjent
  - Uformell, verbal, ofte usagt

Universitetet i Stavanger

# Krav

- Krav
  - Et utsagn som beskriver funksjonelle egenskaper, ytelseparametere og andre egenskaper ved produktet. Formell beskrivelse av behov
  - Fastsatte egenskaper ved produktet fra brukerens synsvinkel
  - Formelt dokument

Universitetet i Stavanger

# Spesifikasjon

- Spesifikasjon
  - Formell beskrivelse av alle utsagn om krav. Plan for realisering av krav
  - Fastsatte egenskaper ved produktet fra leverandørens synsvinkel
  - Formelt dokument

# Kunden og miljøer rundt

- Kunde
  - En som bruker produkter
- Norge finnes det lover og foreskrifter som stiller krav til et produkt
  - Kjøpsloven
  - Lov om produktkontroll
- Produsentens krav
  - Arbeidsmiljø
  - Produksjonsprosessen
  - Vedlikehold
  - Krav til inntjening

Universitetet i Stavanger

# Hva er god nok kvalitet

- 0 feil?
- Akseptabelt antall feil?
- I kvalitetsteorien er 0 feil målet

# Hvem har ansvaret

- Ledelsen har hovedansvaret
  - Hver ansatt har ansvar for å gjøre en god nok jobb
- Forutsetninger for god kvalitet
  - Nok ressurser
  - Riktig verktøy
  - Tilstrekkelig opplæring
  - God motivasjon
  - Klare ansvarsforhold

Universitetet i Stavanger

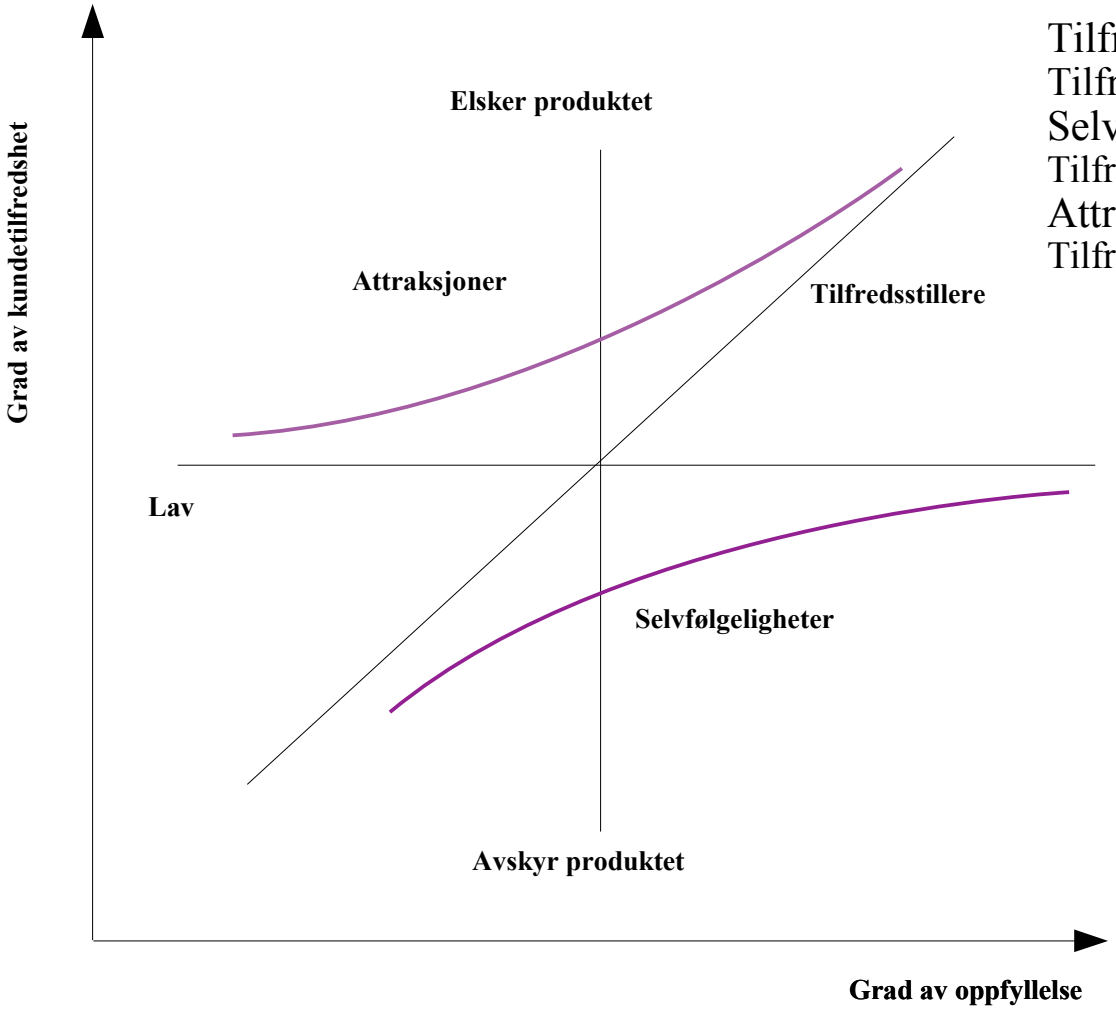
# Er kvalitet lønnsomt?

- **Regnestykke for hva økt kvalitet koster**
  - Kostnad til opplæring
  - Kostnad til evaluering og forbedring av produksjonsprosessen
  - Kostnad til oppfølging og overvåking
- **Regnestykket ved ikke å gjøre noe**
  - Kostnad til kontroll om feil oppstår
  - Kostnad til å gjøre ting om igjen
  - Lavere produktivitet (ineffektive prosesser)
  - Misfornøyde kunder (tap av kunder)

# Er kvalitet lønnsomt?

- **Investering i bedre produktegenskaper betyr:**
  - Investering i kundeundersøkelser og konkurrentundersøkelser
  - Investering i innovasjon og nytenkning
  - Eventuelt investering i nye produksjonslinjer

# Kano - modellen




Tilfredsstillere:
Tilfredsstiller krav som kunden har
Selvfølgeligheter
Tilfredsstiller behov men er selvsagte
Attraksjoner
Tilfredsstiller ubevisste krav

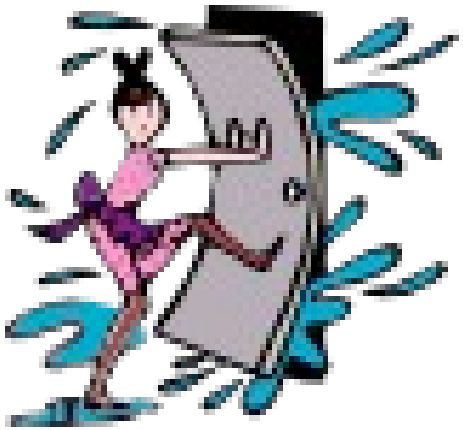
Universitetet i Stavanger

# Strategi for å oppnå kvalitet

- Kontroll av ferdig produkt
- Reparere feil eller produktet kastes
- Forebygge / Kvalitetsikring
  - Krav til organisering av arbeidet
  - Krav til kompetanse
  - Krav til dokumentasjon og formell gjennomgang
  - Krav til milepæler og sjekkpunkter
  - Krav til arbeidsmåte
  - Krav til testing av produktet opp mot spesifikasjoner

# Strategi for å oppnå kvalitet

- **Reparasjonstrategien**
  - Undersøker sluttproduktet og sjekker
  - Tester for å finne feil
  - Utbedrer feil
  - Kaster produktet dersom det ikke kan brukes



# Strategi for å oppnå kvalitet

- **Kvalitetssikringsstrategien**
  - Dette er styrt og prioritert av ledelsen
  - Tilrettelegger for at kvalitet kan oppstå
  - Kvalitet er å tilfredsstille kundene
  - Kvalitet er en del av bedriftens organisering
  - Kundefokus
  - Ledelsen har kvalitetsmål

# TKL

Terje Kårstad

Universitetet
i Stavanger

# Total kvalitetsledelse

- **Hvorfor ledelsesfilosofi?**
  - **Kvalitet lønner seg**
- **Medarbeiderne må beherske**
  - **Teknikker**
  - **Verktøy**
  - **Kjenne beste praksis**
- **Ledelsen må**
  - **Stake ut kurs**
    - **Fastsette strategi**

# Ledelse

- General
- Mennesket er maskiner
- Samarbeidsorienter
  - Medarbeideransvar
  - Alle arbeider sammen mot et felles mål

# Total kvalitetsledelse

- TKL er en ledelsesfilosofi
  - Startet i 1980, men går tilbake til 1920
  - Hett tema rundt 1990
    - Europa, USA såg at Japan var best
- Eks
  - Biler
  - Klimaanlegg
  - Osv
- Hva var det Japan gjorde
  - Ledelsesfilosofien kom fra USA

# TKL

- **Noen navn på kvalitetsguruer**
  - **William Edwards Deming (1900 –1993)**
  - **Joseph Juran  (1904 -**
  - **Armand V. Feigenbaum**
  - **Kaoru Ishikawa**
  - **Genichi Taguchi**
  - **Shigeno Singo**
  - **Masaaki Imai (Kaizen)**

# Hva er de enige om

- Toppledelsen rolle er viktig
  - Ledelsen er ansvarlig for 80% av problemene
- Det er en kvaliteskrise i den vestlige verden
- Kvalitet er viktig
- Kontinuerlig forbedring er viktig
- Behov for kunnskap basert på fakta
- Finne årsakene til problemene
- Aviser kampanjer
- Opplæring og invistering i framtiden
- Aktiv deltagelse av de ansatte

# Demmings 14 punkter

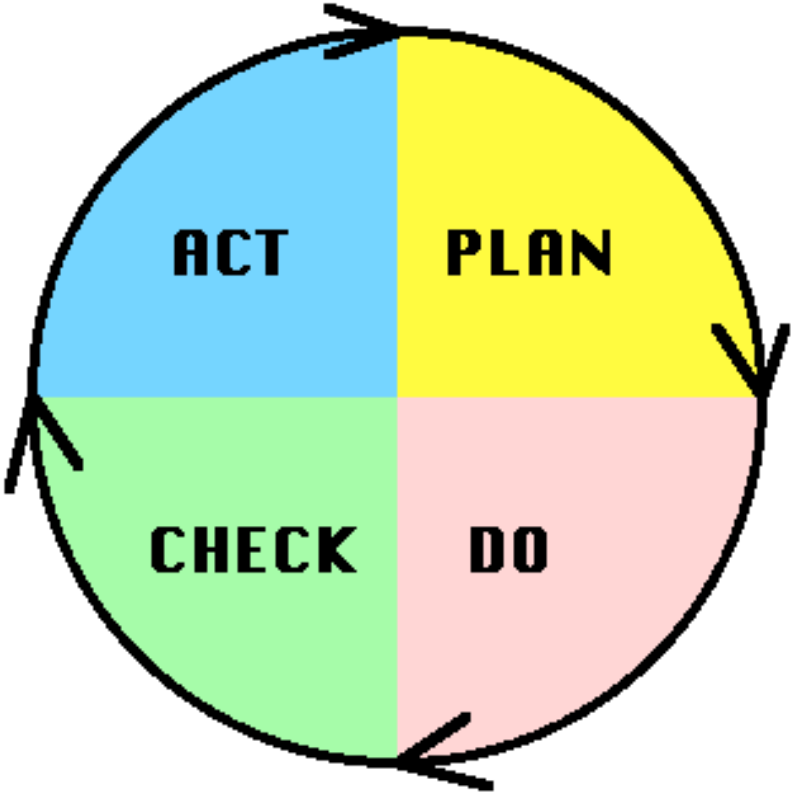
- **Demmings 14 punkter**

1) **Sørg for kontinuerlig forbedring av produkter og tjenester**
2) **Ta i bruk den nye filosofien slik at det oppnøs økonomisk stabilitet**
3) **Slutt med å være avhengig av sluttkontroll for å oppnå kvalitet**
4) **Slutt med å godta laveste anbud**
5) **Utfør kontinuerlig forbedring av produksjonssystemet, og tjenestene som ytes**
6) **Sett opplæring på jobben i system**
7) **Ta i bruk moderne metoder for veiledning og lederskap**
8) **Fjern frykt**
9) **Bryt ned barrierer mellom avdelinger og personer**
10) **Slutt å bruke slagord, formaninger og plakater**
11) **Fjern arbeidsstandarder og kvoter basert på tall**
12) **Fjern hindringer som gjør at arbeiderne ikke får anledning til å vise yrkesstolthet**
13) **Sett i gang et omfattende opplæringsprogram**
14) **Ledelsen må øyeblikkelig gå til aksjon og sørge for at kvalitet og produktivitet kontinuerlig forbedres**

# Demming

- **Dødelige sykdommer**
  1) **Mangel på vedvarende forpliktelse**
  2) **Kortsiktig profitt**
  3) **Prising av prestasjon**
  4) **Jobb-hopping**
  5) **Kun vektlegging av synlige tall**

# Shewhart-Demming hjulet




- **Hjulet**
  1) Planlegg
  2) Utfør
  3) Vurder
  4) Iverksett

# Juran

- **Filosofien**
  1) **Sett opp mål som skal nåes**
  2) **Lag planer som viser hvordan målene skal nåes**
  3) **Fordel ansvar**
  4) **Gi belønning**
- **Kvalitetstriologi**
  - **Planlegging**
    - **Hvem er kunden**
    - **Formidl kundens behov i et språk som alle forstår**
    - **Optimaliser produktet ved stadig å forbedre prosessen**
  - **Kvalitetskontroll**
    - **Oppdag og kontroller avvik fra det optimale**
    - **Sløsing må opphøre**
  - **Kvalitetsforbedring**
    -

# Juran

- **Kvalitetstriologi**
  - **Planlegging**
    - **Hvem er kunden**
    - **Formidl kundens behov i et språk som alle forstår**
    - **Optimaliser produktet ved stadig å forbedre prosessen**
  - **Kvalitetskontroll**
    - **Oppdag og kontroller avvik fra det optimale**
    - **Sløsing må opphøre**
  - **Kvalitetsforbedring**
    - **En leder skal**
      - **Ta ansvar for forbedringer**
      - **Forstå hvilke tiltak som må settes i verk**
      - **Anvende den universelle rekken**

# Juran

- **Den universelle rekke**
  - **Skaffe bevis for behov**
  - **Finn frem til prosjekter**
  - **Skap en organisasjon som stimulerer til forbedring**
  - **Still diagnose**
  - **Gå til aksjon**
  - **Bryte gjennom barrierer som hindrer forbedring**
  - **Ta vare på gevinsten**

# Armand V. Feigenbaum

- Oppdaget av japanerne tidlig på 1950 - tallet
- Boken: «Total Quality Control  TQC»
  - Kvalitetskontrollen må gjelde hele organisasjonen
  - Kvalitetskontroll er en metode å drive foretning etter
    - Fokus på menneskelige relasjoner i organisasjonen
  - Ansvaret ligger hos ledelsen den må:
    - Sette standard for kvalitet
    - Finne ut om en bruker disse standardene
    - Ta affere hvis de ikke følges
    - Planlegge forbedringer av standarden
- Kvalitet er det viktigste for å oppnå susess

# Japanere

- **Kaizen - Forbedring (Masaaki Imai)**
  - Kaizen kontinuerlig forbedring ved hjelp av små skritt
    - Kvalitet skal prege alt
    - Alt skal forbedres ved hjelp av små kontinuerlige skritt
    - Kaizen er viktigere enn innovasjon
- **Kaizen er gitt innhold av**
  - Kaoru Ishikawa
  - Genchi Taguchi
  - Shigeo Shingo

# Kaoru Ishikawa

- **Bedriftsomfattende kvalitetskontroll**
  - Alle må læres opp i kvalitetsteori
    - Kvalitetssirkler
      - Bidra til forbedring
      - Utvikle samvirke mellom mennesker
      - Utnytte det beste i hver enkelt
      - Eksempel på slike grupper
        - Sikkerhetsgruppe
        - Produktivitetsgruppe
        - Null-feil gruppe
        - Kvalitetskontrollgruppe
        - Minitenkeboks

# Kaoru Ishikawa

- **Hans ledelsesfilosofi**
  - **Menneskelige delmål**
    - **Glade ansatte, fornøyde kunder, tilfretse eiere**
  - **Kvalitative delmål**
    - **Riktig kvalitet, null feil**
  - **Økonomiske delmål**
    - **Pris, kostnad, fortjeneste**
  - **Kvantitative delmål**
    - **Mengde, leveringstidspunkt**

# Genichi Taguchi

- **Kvalitetstapfunksjonen**
  - Tap påføres produsenten fordi
    - Ting må gjøres om igjenn
    - Ting må skrapes
    - Vedlikehold fører til tap
- **Funksjonen kan brukes til måling**
  - Blir brukt av Tyota

# Shigeo Shingo

- **Prosesser må stoppes når det oppstår feil**
  - **Årsaken finnes**
  - **Produksjonssystemet må ha måleinstrumenter slik at**
    - **Feil som er i ferd med å utvikle seg, blir oppdaget**
- **Poka – yoke prinsippet (null feil)**

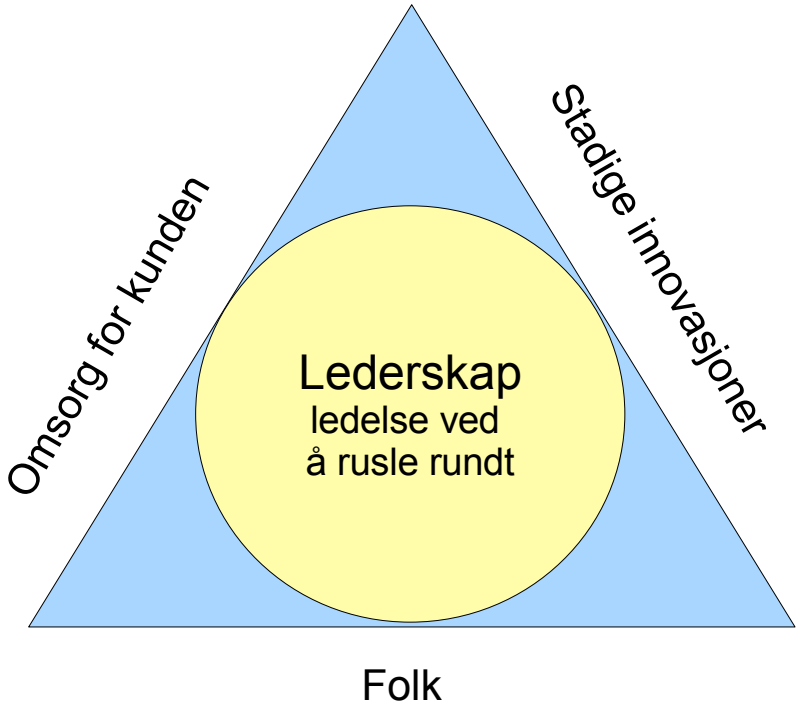
# Ny bølge

- **Philip B. Crosby**
- **Tom Peters**

# Philip B. Crosby

- **Gjøre ting rett første gangen**
- **Kunden må være tilfreds**
- **Fire absolutter for kvalitetsledelse**
  - **Kvalitet er tilfredstille kravene**
  - **Kvalitetsystemet må forhindre feil**
  - **Statistisk prosesskontroll må brukes for å få insikt i produksjonen**
  - **Kvalitet må måles og utrykkes i finansielle termer**
- **Prosessen med kvalitetsforbedring stopper aldri**
  - **Forbedringen skjer i 14 trinn**

# Tom Peters




- **Bok «In Search of Excellence»**
  - **Stadige innovasjoner**
  - **Eksepsjonell omtanke for kunden**
- **Management => Leadership**
- **Lederskap**
  - **Binder sammen**
- **En leder må like forandring**
- **Kvalitetsforbedring tar aldri slutt**

# Claus Møller

- **Firma**
  - **Time manager International**
- **Brukt av EU**
- **Han vektlegger**
  - **Effektivisering av administrative prosesser**
  - **Kunden må tilfredstilles**
  - **De som produserer må inspireres**
  - **Bedriften må kontinuerlig endres til det bedre**
  - **Den enkelte arbeidstager må utvikles**
- **Bruker begrepet kvalitetsorganisasjon**

# Har man lykkes med TKL

- **USA – EUROPA**
  - **Mange kastet seg på bølgen fra 1990**
  - **Ingen vidunderkur, en må satse langsiktig**
  - **Hva er tilstanden i IT bransjen**
  - **Det er kommet en mer realistisk holdning til TKL**
  - **Lede bedrifter slik at kvalitet blir vektlagt**
- **I Japan er dette en del av bedriftskulturen**

# Standarder og kvalitetsmodeller

- **ISO 9000 familien**
  - **Kundefokus**
  - **Lederskap**
  - **Personellets engasjement**
  - **Systemtankegang ved styring**
  - **Kontinuerlig forbedring**
  - **Beslutninger basert på faktiske hendelser**
  - **Gjensidig fordelaktig samarbeid med leverandører**
- **EFQM «Europan Fondation for Quality Management»**

# Programvareindustrien

- **TKL passer**
  - **Kundefokus**
  - **Lederskap**
  - **Personellets engasjement**
  - **Systemtankegang ved styring**
  - **Kontinuerlig forbedring**
  - **Beslutninger basert på faktiske hendelser**
  - **Gjensidig fordelaktig samarbeid med leverandører**
- **I Norge**
  - **SPIQ**

# Kvalitet og programvare
# Kapittel 3

Terje Kårstad

Universitetet
i Stavanger

# Programvarekvalitet og samfunnet

- **Programvare eksisterer over alt**
  - **Banker**
  - **Telefoner**
  - **Lønnssystem**
  - **Biler**
  - **PC-er**
  - **Overvåkningssystem**
  - **Opplæringssystem**
  - **O.s.v**

# Kvalitetsbegrep for programbare

- **ISO 9000:2000 kvalitet – i hvilken grad en samling iboende egenskaper oppfyller krav**
  - **Krav er framsatt av ulike inntressepartnere**
- **Å utvikle programvare som har kvalitet betyr:**
  - **Programmet *oppfyller de fremtidige brukeres behov og forventninger***
  - **Programmet er *uten alvorlige* feil**
  - **Programmet har egenskaper som gjør det *egnet* for kunden**

# Vi vil nå se på følgende begrep

- **Kvalitetsbegrepet**
- **Kunder å andre inntressepartnere**
- **Kvalitet og økonomi**

# Klassifisering av programkvalitet

- **Hensiktsmessig å skille mellom begrepene**
  - **Prosesskvalitet**
    - **Utviklingsløpet**
  - **Produktkvalite**
    - **Egenskaper ved systemet sett fra brukeren**
  - **Ekstern kvalitet**
    - **Synlig for brukeren**
  - **Intern kvalitet**
    - **Synlig for utvikler**

# Kvalitetsattributter

Skal vi oppnå kvalitet, må vi vite hva vi mener med kvalitet. Det nytter ikke med generelle termer, man må vite hva som menes med kvalitet helt spesifikt for det systemet som utvikles. En snakker om systemets kvalitetsatributter.

# Kvalitetsatributter

- **Hva vil kunden ha, hva er viktig?**
  - **Korrekt, pålitelig og robust**
  - **Ytelse og effektivitet**
  - **Brukervennlighet**
  - **Vedlikeholdbart**

# ISO og Kvalitetsatributter

- **ISO og kvalitetsatributter ISO 9126**
  - **En ISO stander er retningsgivende, den sier ikke at alle systemer som lages skal ha de atributtene som er spesifisert i standarden, men den er nyttig som veiviser når en diskuterer kvalitetsatributter for et system.**

# ISO 9126

- **Funksjonalitet (systemet skal møte spesifiserte funksjonelle behov/krav)**
- **Pålitelighet ( systemet skal kunne opprettholde sitt ytelsesnivå under spesifiserte betingelse og i et spesifisert tidsrom)**
- **Brukbarhet (systemets brukeregenskaper)**
- **Effektivitet (responstid, ressursbruk)**
- **Vedlikeholdbart og utvidbart**
- **Flyttbart ( kan installeres i mange miljøer)**
- **Erstatningsegenskaper (kan systemet erstatte andre system)**

# Kvalitet slik kunden ser det

- **Det er i dialog med kunden / brukerne at et system med god kvalitet kan utvikles**
- **Det er ikke mulig for noe system å være optimaliser for alle attributtene i ISO standarden**
- **Måler er å lage et system som mest mulig presist oppfyller de attributtene som er viktige for akkurat dette systemet.**
- **For hvert stikkord i listen må vi avgjøre om dette er relevant.**
- **Systemutvikleren vil ofte lage et gullforgylt system, men er det kunden vil ha?**

# Hvem er kunden

- **Vi utvikler programvare for oss selv**
- **Vi utvikler et kommersielt produkt**
  - **systemet skal selges , vi må avdekke krav og behov**
- **Utvikler systemet for en bestemt kunde**
  - **Oppdragsgiver er den som skal betale, hva ønsker han**
  - **Brukerne hva ønsker de**

# Programvare og lovverk

Lov om arbeidsvern, arbeidsmiljø, lov om behandling av personopplysninger, rammeavtale mellom NHO og LO om teknologisk utvikling av datamaskinbaserte systemer

# Kvalitet slik utvikleren ser det

- **Kunden er ikke på jakt etter det nye og framtidstetta, han vil tilfredstille dagens behov. Han vil ikke ha noe ønske om å tenke framover**
- **Kun kundekrav vil frata utvikleren yrkesstolthet, han blir fabrikkarbeider**
- **Kunden bør nok ha et viktig ord med i laget, når et system spesifiseres, men systemutvikleren må virke som rådgiver.**

# Kvalitet økonomi og programvare

- **Kvalitet økonomi for kunden**
  - **Driftsstans og feilretting koster mye**
  - **System som er enkelt å bruke øker produktiviteten**
- **Kvalitet og økonomi for produsenten**
  - **Det lønner seg for produsenten å framstille det riktige systemet mest mulig effektivt**
    - **Riktig første gang?**
    - **Utvikling av programvare er menneskeintensivt**
    - **Forståelse av kvalitet er i stadig utvikling**
    - **Det går lang tid fra en investerer i forbedring av utviklingsprosessen til en ser resultater i form av bedre programvare**

# Hva koster egentlig kvalitet

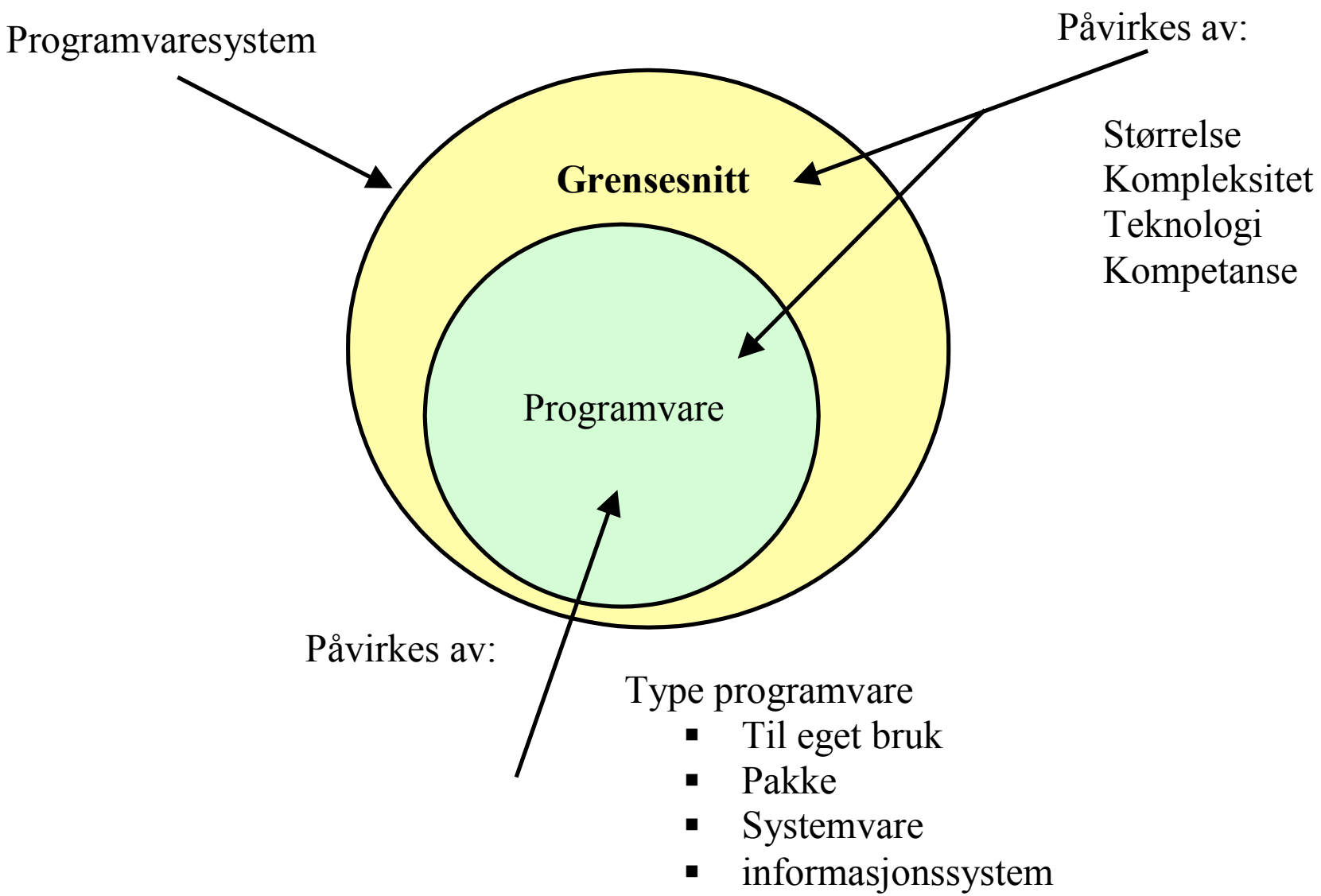
- Mangel på kvalitet har følgende kostnader
  - Finne feil
  - Rette feil
  - Sende ut nye versjoner
  - Renome
  - Tap av markedsandeler

# Det er viktig å finne feil tidlig

- Fasen fra behov til krav
  - Kostnad 1
- Fase fra krav til ferdig system
  - Kostnad 10 - 150
- Etter installasjon
  - Kostnad 1000 og oppover

# Programvareutvikling/Systemutvikling

- Systemutvikling – programvareutvikling og litt til
- Systemutvikling er:
  - Utvikle programvare
  - Definere og avgrense grensesnittet mot omgivelsene
  - Grensesnitt mot annen programvare


Programvaresystem
Påvirkes av:
Størrelse
Kompleksitet
Teknologi
Kompetanse
Grensesnitt
Programvare
Påvirkes av:
Type programvare
Til eget bruk
Pakke
Systemvare
informasjonssystem

# Etikk for systemutvikling

- Systemutvikling skal være i offentlighetens inntresse
- Systemutvikling skal ivareta inntressene til kunde og arbeidsgiver
- Produktet  skal ha høyst mulig standard
- Systemutvikler skal opprettholde integritet og dømmekraft
- Oppmuntre til etiske holdninger under utviklingen av systemet
- Fremme integritet og omdømme til profesjonen
- Rettferdig overfor andre kollegaer
- Delta i livslang læring

# Vi avgrenser oss til å se på

- **Utvikling av større informasjonssystemer som skal støtte arbeidsoppgaver i en bestemt bedrift**
- **Informasjonssystem:**
  - **Et system der programvaren er det primære, men som også har med seg arbeidsrutiner, utstyr og opplegg for brukerstøtte.**
  - **Systemet skal tilfredstille kunden**
  - **Systemet er skreddersøm**
  - **Kunden betaler utviklingen**
  - **Arbeidet organiseres som et prosjekt**
  - **Systemet utvikles av systemutviklere, men kunden deltar i prosjektet**

# Standarder for kvalitetssystemer
# Kapittel 4

Terje Kårstad

Universitetet
i Stavanger

# Kvalitetssystem

- **Kvalitetssystemet er de styringselementer som ledelsen innfører i en bedrift for å ha kontroll med produksjonen, og dermed kvaliteten på produktene**
- **Liten bedrift, ikke noe formelt system (slik gjør vi det her)**
- **Når bedriften vokser, må en skrive ned noen styringsprinsipp**
  - **Jobbeskrivelse**
  - **Noen standarder må innføres**
  - **Jo større bedrift, dess mer systematikk**

# Standarder for kvalitetssystem

- **ISO 9000:2000 serien**
  - **Styringssystem for å rettlede og styre en organisasjon når det gjelder kvalitet.**
- **En bedrift som følger ISO standarden må forholde seg til**
- **Åtte overordnede prinsipper for kvalitetsstyring**
- **Krav til kvalitetssystemets struktur og innhold**
- **Krav til kontinuerlig overvåkning og forbedring av systemet**

# Kvalitetssystem for hvem

- For ledelsen
- For de ansatte (info om hvordan ting skal gjøres i bedriften)
- Skal demonstrere for omverden at her er det orden i sysakene

# ISO 9000:2000 styrker kundefokus

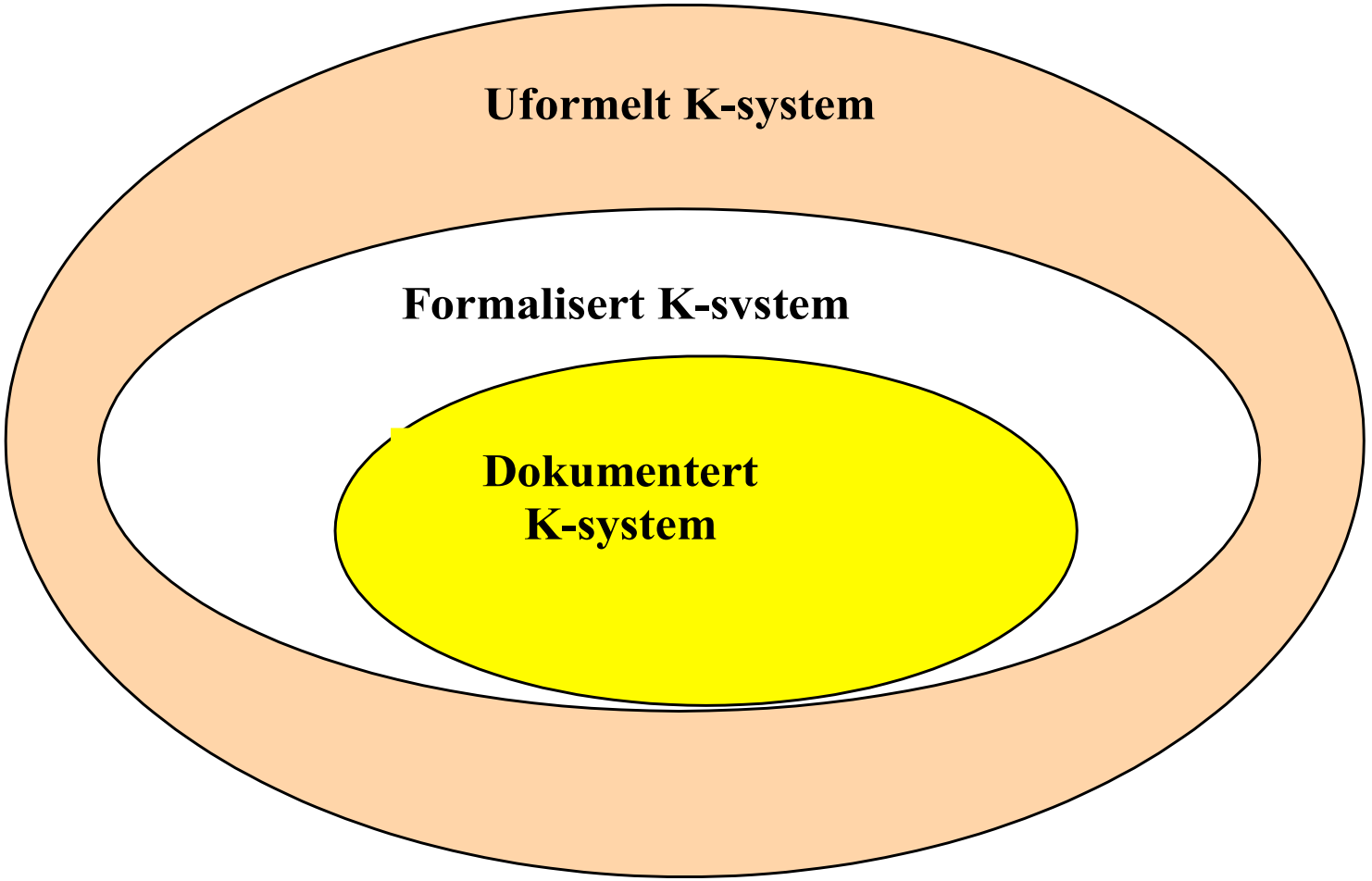
- Anvendelse av et system for kvalitetstyring fører til følgende
  - oppmuntrer organisasjonen til å analysere kundekrav
  - fastsette prosesser som fører til produkt som er akseptable for kunden
  - holder styr på prosesser i bedriften
  - kan være et rammeverk for kontinuerlig forbedring
  - gir organisasjonen og kundene tillit til at bedriften kan framskaffe produkter

# Kvalitetsystem

Kunden myndighetene  kan vurdere

- Hvilke prosesser har bedriften
- Hvordan styres disse prosessene
- Hvordan ledes bedriften
- Hvem har ansvar for hva
- Hvordan reguleres forholdet til kundene

# Kvalitetssystemet



# ISO 9000:2000 serien

- **ISO 9000:2000 Grunntrekk ved systemer for kvalitetsstyring og terminologi**
- **ISO 9000:2001 Krav til system for kvalitetsstyring, kan brukes internt eller til sertifisering**
- **ISO 9004:2000 Veiledning utover kravene i ISO 9001**
  - **Hovedtanken i disse standardene er generelle og allmenngyldige**

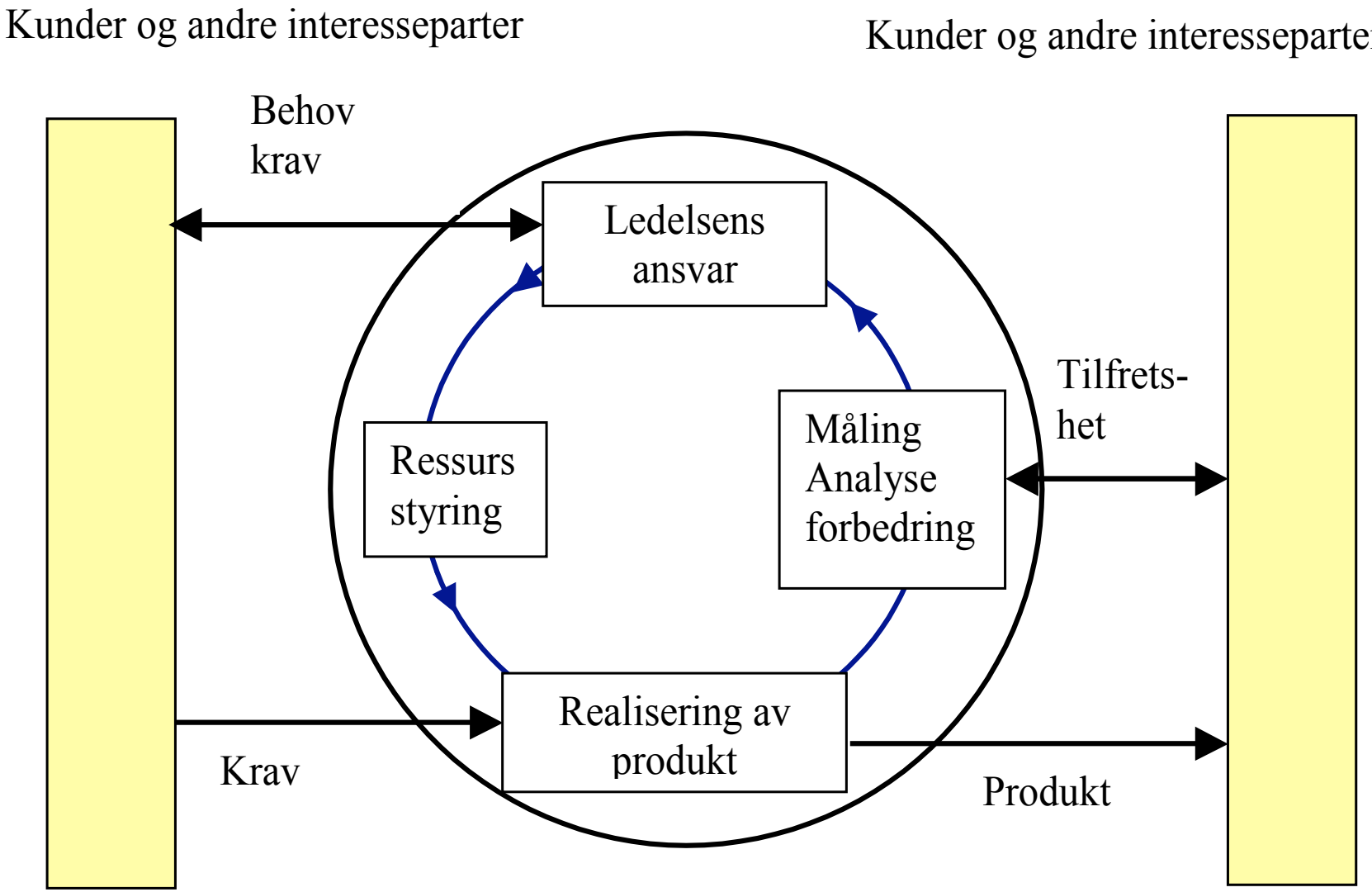
# ISO 9000:2000 vektlegger 8 prinsipper

- kundefokus
- lederskap
- prosesstanken
- personellets engasjement
- systemtankegang ved styring
- kontinuerlig forbedring
- beslutninger basert på faktiske hendelser – måling
- gjensidig fordelaktig samarbeid med leverandør

# ISO 9000:2000 seriens 4 hovedprosesser

- **ledelsens ansvar**
- **ressursstyring**
- **realisering av produkt**
- **måling analyse, forbedring**

# ISO 9000:2000 og kontinuerlig forbedring



# Ledelsens rolle

- **Gjennom lederskap og tiltak kan ledelsen skape et miljø som engasjerer de ansatte i kvalitetsforbedring. Ledelsens rolle er som følger:**
  - **Etablere og holde ved like organisasjonens kvalitetspolitikk og kvalitetsmål**
  - **Fremme anvendelser av kvalitetspolitikken og kvalitetsmålene gjennom hele organisasjonen, øke bevissthet og engasjement og motivasjon**
  - **Fokusere på kundens krav gjennom hele organisasjonen**
  - **Sørge for at hensiktsmessige prosesser er tatt i bruk for å gjøre det mulig å oppfylle krav fra kundene og andre inntresseparter for å oppnå kvalitet.**
  - **Sørge for at et virkningsfullt og effektivt system for kvalitetstyring blir etablert, iverksatt og holdt ved like for å oppnå kvalitetsmålene**
  - **Sørge for at nødvendige ressurser er tilgjengelig**
  - **Ha periodisk gjennomgang av kvalitetstyringssystemet**
  - **Beslutte tiltak vedrørende kvalitetspolitikken og kvalitetsmålene**
  - **Beslutte tiltak for å forbedre systemet for kvalitetstyring**

# Bedriftens kvalitetsmål

- *Noe som forsøkes oppnådd, eller som det siktes mot, når det gjelder kvalitet.*
- Eks:
  - For 2006 skal andeler reklamasjoner på produkt X reduseres med 20%, sammenlignet med antall reklamasjoner i 2005. Antallet reklamasjoner måles som antall reklamasjoner delt på totalt antall leverte varer av type X

# Bedriftens kvalitetspolitikk

- **Def**
  - **Organisasjonens overordnede hensikter og retning angående kvalitet slik dette formelt er uttrykt av den øverste ledelsen.**
- **Kvalitetskolikken må være praktisk rettet, den må ikke være slagordspreget. Den må være:**
  - **Konsis og klar**
  - **Angå alle ansatte**
  - **Sette klare standarder for prestasjoner og kvalitet**
  - **Være signert at toppledelsen**

# Eksempel 1:

- Kvalitet er å tilfredsstille kundens behov
- Alle har en kunde – vi har også interne kunder
- Kvalitet dreier seg om å tilrettelegge arbeidet slik at feil og mangler forhindres
- Alle enheter skal ha kvalitetssystemer som gir tillit til at kvalitet kan oppnås
- Kvalitet på hver enkelts arbeid og i samspill med andre legger grunnlaget for å oppnå kvalitet
- Alle har ansvar for kvalitet på eget arbeid, men ledelsen har et spesielt ansvar for å tilrettelegge slik at kvalitet kan oppnås

# Eksempel 2

- **Vårt firma vil levere feilfrie produkter i rett tid og som oppfyller kundens forventninger ved å**
  - **Forstå kundens behov**
  - **Forutse og arbeide for å møte kundens framtidige behov**
  - **Kontinuerlig forbedre oss og forhindre feil**
  - **Tiltrekke oss og sørge for opplæring av kvalifisert personell**

# Standardens krav til kvalitetstyring

- **Standarden stiller krav og gir veiledning i styring av:**
  - **Menneskelige ressurser**
  - **Infrastrukturer**
  - **Arbeidsmiljø**
  - **Leverandør og partnerskap**
  - **Naturresurser**
  - **Økonomiske ressurser**

# Noen trender

- Ansatte får delegert myndighet
- Kontrollrutiner blir fjernet, satser på veiledning og opplæring
- Kontrollen skal kun avdekke forbedringsmulihhete
- Alle som påvirker kvalitet, skal ha frihet til å:
  - Iverksette tiltak for å forhindre problemer
  - Identifisere og dokumenter problemer
  - Ta initiativ til å løse problemer
  - Verifisere løsningen på problemer
  - Sørge for oppfølgning til det er funnet tilfredstillende løsninger

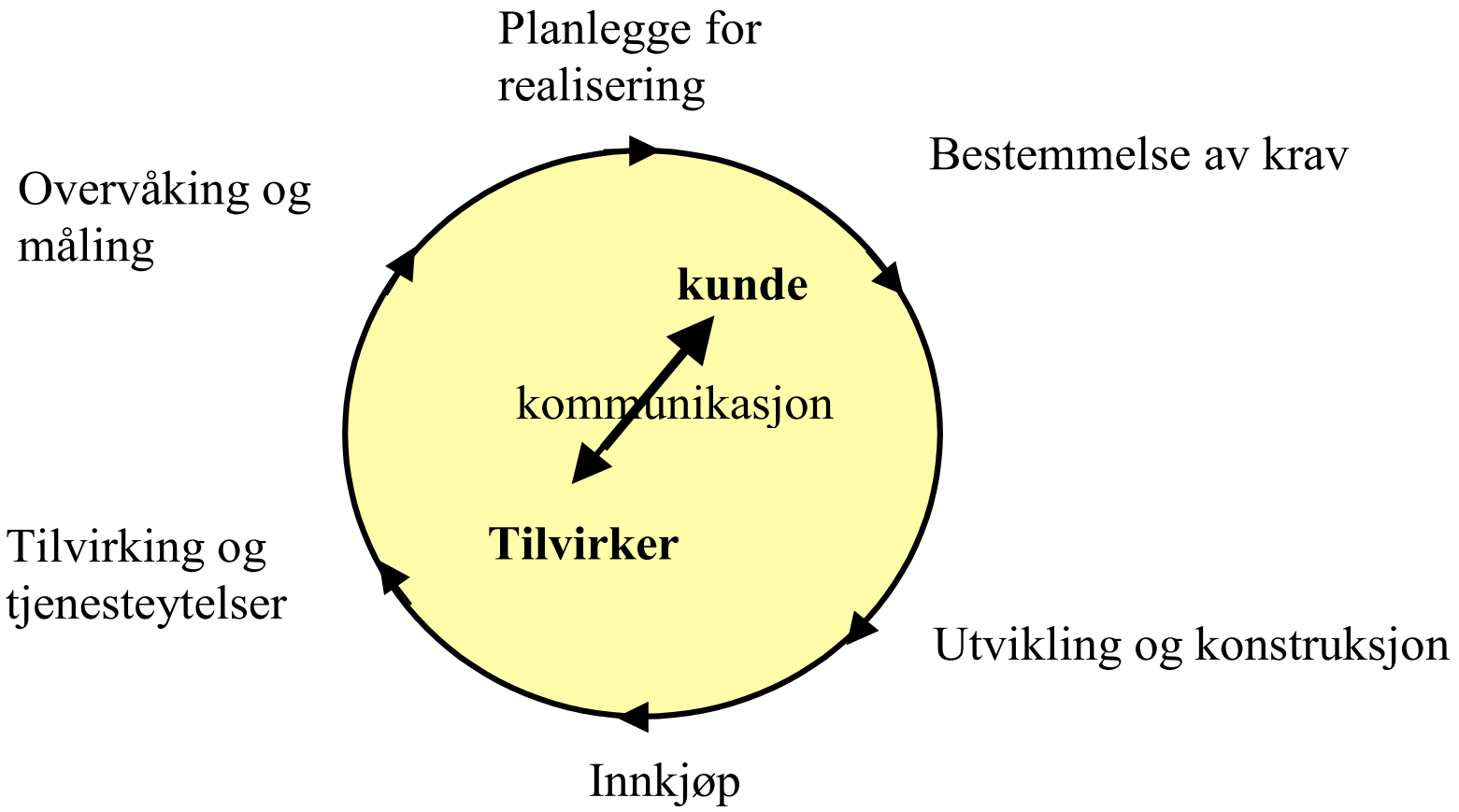
# Egne kvalitetsfunksjoner

- **En bedrift hvor kvalitet styres etter standard prinsipper, vil ikke ha en stor kvalitetsavdeling, men et stort antall medarbeidere praktiserer kvalitet.**
- **Ingen medarbeidere har lenger ansvar for kun å kontrollere, men det kan være behov for en kvalitetsleder. Denne lederen ivaretar visse oppgaver på vegne av ledelsen**
- **Kvalitetslederen bør ha myndighet til å treffe beslutninger og disponere over ressurser**
- **Kvalitetslederen bør rapportere direkte til ledelsen**

# ISO 9000:2000 serien er sterkt prosess orientert

- **Prosesser:**
  - **En samling av beslektede eller samvirkende aktiviteter som omformer tilført grunnlag til resultater**
- **To typer prosesser**
  - **Prosesser for realisering, resulterer i produkter**
  - **Støtteprosesser**

# En bedrift må planlegge,organisere og utvikle de prosesser den har behov for



# Standarden har krav til innhold i hver av deleprosessene

- **Eks: Under planlegging av utvikling og konstruksjon skal organisasjonen bestemme:**
  - **Utvikling og konstruksjonstrinnene**
  - **Gjennomgåelse, verifikasjon, validering som er aktuelt for hvert trinn i utviklingen og konstruksjonen**
  - **Ansvar og myndighet for utvikling og konstruksjon**

# Utforming av prosesser ISO 9004:2000

Dokumentasjonens omfang og natur bør være passende for organisasjonen og tilfredstille krav i kontrakter, lover, forskrifter, behov og forventninger hos kunder og andre inntressepartnere. Dokumentasjonen kan være i enhver for eller på et medium som passer for organisasjonens behov.

# Dokumentasjon av kvalitetsystemet

- **Det dokumenterte systemet vil en finne i en kvalitetshåndbok**
- **Innhold:**
  - **Ledelsens mål for kvalitetspolitikken**
  - **Dokumenterte prinsipper for kvalitetsarbeidet, samt ansvarsforhold. Dette rammeverket for de detaljerte instruksene på neste nivå**
  - **Detaljerte instrukser for hvordan arbeidet skal foregå. Her finnes prosedyrer og standarder**

# Hvordan organisere slike håndbøker

- Lagres elektronisk intranett
- Tilgang via en nettleser
- Alltid oppdatert

# ISO standard og krav til dokumentasjon

- Dokumentasjon av kvalitetsystemet skal omfatte
  - Dokumenterte utsagn om kvalitetspolitikken og kvalitetsmål
  - En kvalitetshåndbok
  - Dokumenterte prosedyrer som kreves i standarden
  - Dokumenter som organisasjonen trenger for å sørge for effektiv drift og kontroll av prosesser
  - Registreringer som kreves i denne standarden
- ISO har også krav til godkjenningsprosedyre i forbindelse med utgivelse, endringsrutiner og versjonskontroll

# Prosedyrer utgjør en sentral del av kvalitetsystemet

- En dokumentert prosedyre angir vanligvis
  - Hensikt og omfang
  - Målgruppe – hvem gjelder prosedyren for
  - Startbetingelser
  - Beskrivelse av framgangsmåte – når og hvordan den skal gjennomføres
  - Sluttbetingelser
  - Hvilke dokumentasjon som skal brukes
  - Krav i forbindelse med endring og godkjenning av prosedyren
  - Forbindelser til andre prosedyrer

# Advarsel

- **Vær kritiske til prosedyrer, de kan sementer hele organisasjonen.**
  - **Er sjekklister et alternativ til prosedyrer**
  - **Prosedyren bør utarbeides av dem som skal bruke den**
  - **Ansvar og vedlikehold bør ligge hos dem som bruker prosedyren**
  - **Alle prosedyrer bør underlegges en form for versjonskontroll.**

# Revisjon av kvalitetssystemet

- **Kvalitetsystemet er dynamisk, det utvikler seg i takt med skiftende tider og omstendigheter**
- **Bak må ligge et systematisk opplegg for**
  - **Måling og tolking av resultater**
  - **Revisjon og oppfølging for å avdekke forbedringsmuligheter**
  - **Systematisk forbedringsarbeid**

# ISO 9000:2000

- **Ledelsens gjennomgang**
- **Måling og analyse av kundens tilfredshet, kvalitetssystemet, prosesser, produkter**
- **Forbedring, korrigering, tiltak**

# Revisjon

- **Hensikt**
  - **Overvåke kvalitetssystemets tilstand og å finne forbedringsmuligheter. Revisjon må foregå på en ryddig måte. Hensikten er å forbedre kvalitetssystemet ikke å overvåke de ansatte!**

# Ulike typer revisjon

- **Definisjoner**
  - **Gjennomgang: Gjennomføres for å bestemme hensiktsmessigheten, tilstrekkeligheten og virkningen for det aktuelle emnet når det gjelder å oppnå etablerte mål**
  - **Revisjon: systematisk og uavhengig dokumentert prosess for å framskaffe revisjonsbevis og bedømme det objektivt for å bestemme i hvilke grad kriterier for revisjon er oppfylt**
  - **Revisjonsbevis: registrering, angivelse av faktiske forhold eller annen informasjon som er relevant for kriteriene for revisjon og som kan verifiseres.**
  - **Kriterier for revisjon: samling av politikk, prosedyrer eller krav brukt som referanse**

# Intern revisjon

- Ulike typer revisjon som bedriften gjennomfører selv
  - Prosjektevaluering
  - Oppfølging av praksis
  - Ledelsesgjennomgang

# Ekstern revisjon

- **Konsulterende eksternrevisjon**
- **Overvåkende eksternrevisjon**

# Prosjektevaluering

- **Alle prosjekter bør evalueres etter avslutning. Dette kan føre til**
  - **Endringer i kvalitetssystemet**
  - **Oppbygging av en erfaringsbank**

# Oppfølging av praksis

- **Kontroll om prosjekter og annen produksjon foregår i henhold til gjeldende prosedyrer og retningslinjer gitt i kvalitetsystemet (”quality audit”)**
  - **Bør gjennomføres regelmessig**
  - **Må være formalisert**
  - **De som gjennomfører kontrollen er uavhengige**
  - **Innenfor et rimelig omfang**

# Ledelsesgjennomgang

Kvalitet er ledelsens ansvar. Ledelsen må derfor jevnlig foreta grundig gjennomgang av kvalitetssystemet. Fungerer kvalitetssystemet etter hensikten? All tilgjengelig informasjon benyttes

- Resultater av intern revisjon
- Rapport fra prosjektevaluering
- Resultater fra kundeundersøkinger
- Analyse og måling av produksjonen
- Informasjon om endringer i teknologi

# Konsulterende ekstern revisjon

Hensikten er å la utenforstående vurdere kvalitetssystemet

Dette er en type revisjon som utføres i samarbeid mellom bedriftens folk og eksterne konsulenter. Denne revisjonen er varslet

# Overvåkende revisjon

Målet er å avdekke om bedriftens kvalitetssystem fungerer slik det skal. Denne revisjonen utføres av eksterne revisorer, på oppdrag fra noen utenfor bedriften.
(oppfølging etter at bedriften har blitt ISO sertifisert)
I denne revisjonen har bedriften og revisjonen ulike mål.

# ISO-organisasjonen og sertifisering av kvalitetssystemer
## Kapittel 5

Terje Kårstad

Universitetet
i Stavanger

# Hva er ISO

- **ISO – ISOS latin betyr likt**
- **ISO – International Standardisation Organisation**
- **Internasjonalt nettverk for nasjonale standardiriseringsorganisasjoner**
- **Hovedkvarter i Genève**
- **Utgir internasjonale standarder**
- **Ca. 130 medlemsland medlemmene må drive med nasjonalt standardiseringsarbeid**
- **Norge er representert ved Norges Standardiseringsforbund**

# Standarder utgitt av ISO

- Mer enn 11000 standarder
- En standard kan være noen få sider eller mange hundre sider
- Eks
  - Grafiske symboler som brukes i biler
  - Standard for ISBN-nummer (International Standard Book Number)
  - Standard for bolter og skruer
  - Standard for kreditkort

# Hvordan blir en ISO-standard en standard?

- ISO-standarder er tilbud, det er frivillig å ta dem i bruk
- En nasjon eller de som produserer varer må frivillig ta standarden i bruk
- En ISO-standard gjøres til norsk standard gjennom Norges Standardiseringsforbund. Den vil da bli merket som ”Norsk Standard”. Den får da et NS-nummer
- Er ISO-standarden tatt i bruk som europeisk standard, vil den bli merket med ”Europeisk Standard,” EN-nummer
- Eks NS-EN-ISO 9000-3

# Arbeidet med en standard

- En ISO-standard er markedsdrevet
- Når en standard skal utvikles, sette det sammen en gruppe av eksperter som utvikler standarden
- Når en standard utgis dokumenterer den hva ekspertene har kommet fram til skal være den nye standarden
- Utviklingen av standarden kan ta mange år
- En ISO-standard må være nyttig og ikke representere foreldede løsninger
- En ISO-standard revideres hvert femte år

# Hva finner vi i en standard?

- Tekniske spesifikasjoner eller andre presise kriterier
- Dette er
  - Forskrifter
  - Retningslinjer
  - Definisjoner
  - Det er viktig at en standard er presis, derfor mange tørre definisjoner

# Hvilke myndighet har ISO?

- ISO har ingen myndighet
- Arbeidet med en standard er behovdrevet

**Av dette følger:**

- **ISO pålegger ingen å følge en standard**
- **ISO følger ikke opp hvordan standarden brukes**
- **ISO foretar ingen sertifisering**
- **ISO fortar ingen godkjenning av de som driver ISO sertifisering**

# ISO-9000-serien og andre aktuelle standarder

- **ISO 8402 Kvalitetsledelse og kvalitetsikring – Terminologi (1986)**
- **ISO 9000-serien (1987) (Norsk standard 1988)**
- **ISO 9000:2000 omarbeidet ISO 9000 standard (2000)**
- **Den siste standarden består av:**
  - **ISO 9000:2000 System for kvalitetsstyring. Grunntrekk og terminologi**
  - **ISO 9001:2000 System for kvalitetsstyring. Krav til et slikt system**
  - **ISO 9004:2000 System for kvalitetsstyring. Retningslinjer for prestasjonsforbedring**
- **Følgende standarder er også aktuelle for bedrifter som driver systemutvikling**
  - **ISO 9000-3:1998 Retningslinjer for kvalitetssikring av programvare**
  - **ISO 9004-2 Kvalitetsledelse. Retningslinjer for tjenester**
  - **ISO 9004-8 Kvalitetsledelse. Retningslinjer for prosjektstyring**
  - **ISO 10005:1995 Retningslinjer for kvalitetsplaner**
  - **ISO 10007:1996 Retningslinjer for konfigurasjonsstyring**
  - **ISO 10011:1997 Retningslinjer for revisjon**

# Sertifisering av kvalitetssystemer

- *Def.*
  - En uavhengig gjennomgang og vurdering av kvalitetssystemet mot en akseptert standard
  - En garanti for at kvalitetssystemet oppfyller visse minimumskrav
  - Ingen garanti for kvaliteten på produktet

# ISO-sertifisering har positiv virkning

- Ansvaret for kvalitet blir plassert hos leverandøren
- Leverandørens dokumentasjon av prosedyrer har blitt bedre
- ISO-9000 sertifisering blir sett på som et konkurransefortrinn
- En ISO sertifisert bedrift får ofte redusert sine kostnader p.g.a. feil ved produktet
- Kundene har ikke lenger behov for egne kvalitetsrevisjoner

- Sertifiseringshysteri
  - Sertifisering brukes for å overbevise kundene
  - Dette ble brukt mye av leverandører til oljeindustrien på 1990 talet
- Hva koster sertifisering
  - Finnes ingen svar

**Gode argumenter for:**

- Kundene krever det
- Bedriften har behov for et objektivt og samlende mål for et kvalitetsforbedringsprogram
- Den regelmessige oppfølgingen av kvalitetssystemet fører til at systemet holdes ved like

**Gode argumenter mot:**

- Sertifisering er dyrt og binder ressurser i lang tid framover
- Sertifisering betyr kun at kvalitetssystemet oppfyller visse generelle minimumskrav
- Sertifisering kan ta fokus bort fra det verkelige kvalitetsproblemet
- Sertifisering gir ikke nødvendigvis konkurransefortrinn
- Sertifiserte bedrifter leverer ikke nødvendigvis bedre produkter

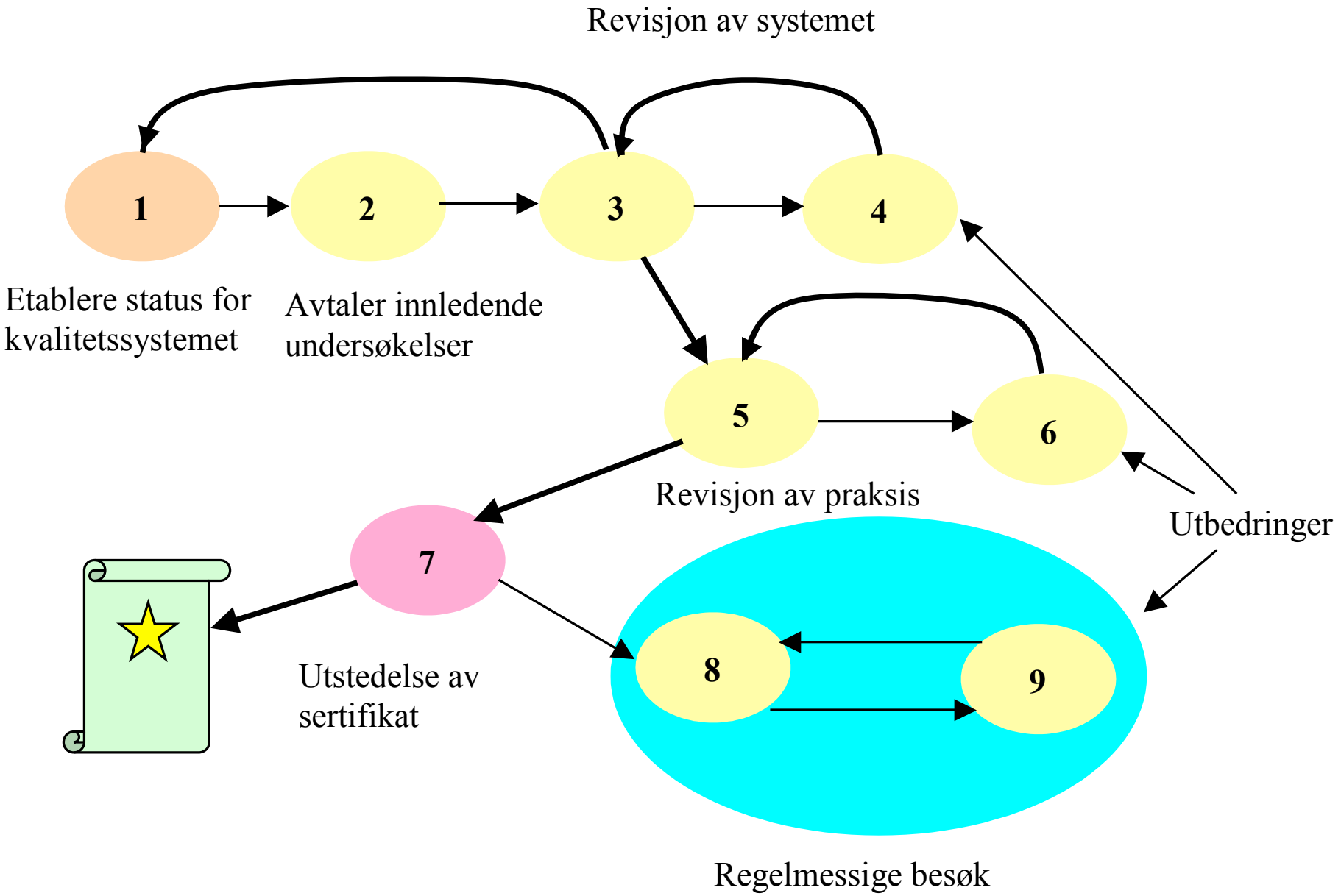
# Sertifisering skal utføres av **tredjepar**

- Her er noen eksempler på ISO begrep
  - Første parts revisjon
    - Bedriftsintern revisjon
  - Andre parts revisjon
    - Ekstern revisjon utført av kunder
    - Tredje parts revisjon
    - Ekstern revisjon i en bedrift utført av en uavhengig organisasjon etter anmodning fra bedriften

# Sertifisering i Norge

- Sertifisering er en kommersiell aktivitet som kan utføres av hvem som helst. Det stilles krav til dem som utfører sertifisering
- Akkrediteringsorganer godkjenner dem som sertifiserer
- Noen bedrifter som utfører sertifisering
  - Det Norske Veritas
  - Norwegian Certification System
  - Norsk Systemsertifisering

# Sertifiseringsprosessen



# Avtaler – innledende undersøkelser

- Hvilke standarder skal ligge til grunn
- Hvilke sertifiseringsfirma skal benyttes
- Hvilke kvalifikasjoner har de ulike firma
- Sertifiseringsselskap kontaktes
- Beskrivelse dokumenter og kvalitetsystem oversendes
- Besøk av firmaet og diskuterer
  - Pris
  - Utarbeides avtaler om pris og tidsramme

# Revisjon av systemet

- Sertifiseringsfirmaet går gjennom det dokumenterte kvalitetssystemet
- Bedriften får påbud om forbedringer
- Bedriften foretar nødvendige forbedringer

# Revisjon av praksis

- Sertifiseringsystemet sjekker om bedriften etterlever det dokumenterte kvalitetssystemet
  - Sjekker utvalgte prosjekter opp mot srandard
- Sertifiseringsfirmaet kommer med kommentarer til praksis
- Nødvendige korreksjoner i praksis foretas
- **Utstedelse av sertifikat**

- **Regelmessige besøk**
  - **Et sertifikar varer ikke evikt, sertifiseringselskapet kommer jevnlik på besøk for å sjekke bedrifte,**
- **Resertifisering**
  - **Et sertifikat gjelder i tre år. Da må kvalitetssystemet resertifiseres. En resertifisering følger samme prosedyre som sertifisering.**

# Innføring av et kvalitetssysten for systemytvikling
# Kapittel 5

Terje Kårstad

Universitetet
i Stavanger

# Et Styringsystem for systemutvikling

- Hva bør et kvalitetsystem for programutvikling inneholde?
- Hva bør en bedrift gjøre for å få på plass et slikt system?

# *Standarder som utgangspunkt*

- **ISO 9000:2000-serien**
- **ISO 9000-3 kan være nyttig**
- **Andre standarder**

SO-standarden er nyttig for å få på plass kvalitetsystemets overordnede struktur og innhold, men vi må støtte oss på andre standarder for å utforme de enkelte kravene i standarden

# *Kvalitetsystemets elementer*

- **Hovedprosessen for produksjon deles ofte i to**
- **Livsløpsprosessen**
  - **Aktiviteter vi gjennomfører fra kunden utrykker et behov, til systemet taes i bruk**
- **Støtteprosesser**
  - **Prosesser som støtter opp om livsløpsprosessen**
  - **Testing**
  - **Konfigurasjonstyring**
  - **Prosjektstyring**
  - **Innkjøp**
  - **etc**

# Kvalitetsystemets elementer

I et kvalitetssystem kan en forvente å finne følgende elementer

**Prossmodeller:** **overordnet rammeverk – viser overordnet framgangsmåte for en prosess**

**Metoder:** **detaljert ftamgangsmåte for å løse et problem, kan bli brukt i flere prosessmodeller, på flere steder i modellen**

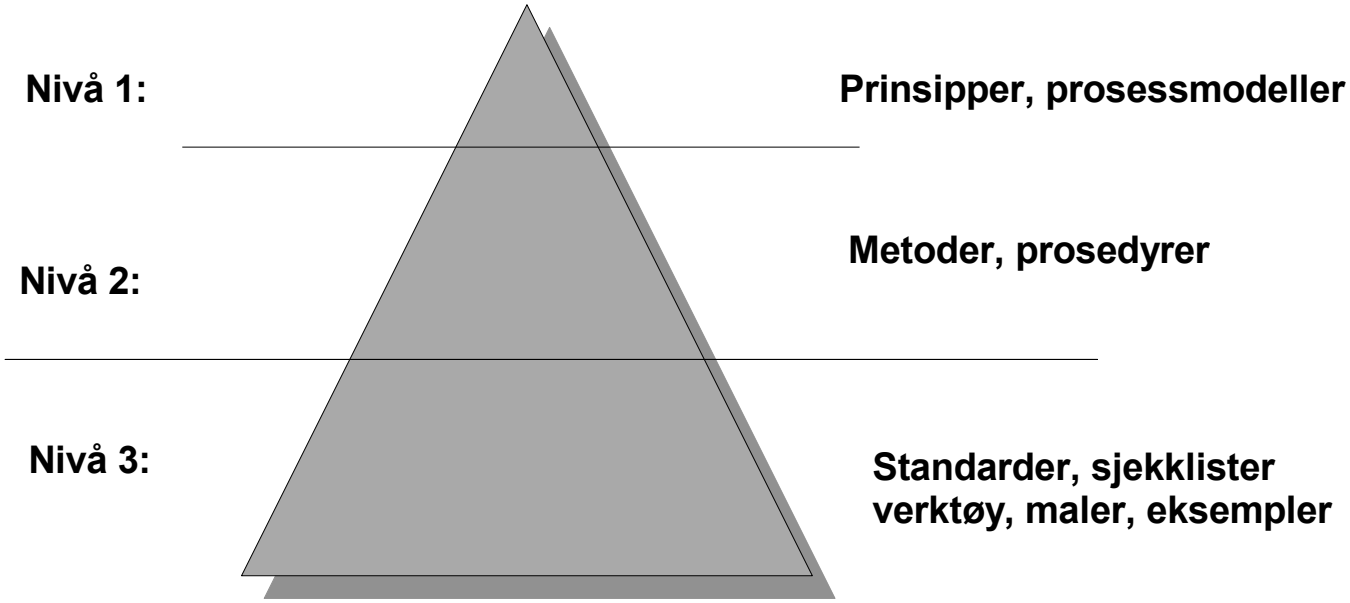
**Prosedyre og teknikker:** **viser detaljert framgangsmåte for en oppgave, kan støtte flere metoder**

**Verktøy:** **gir støtte til å gjenomføre metoder og teknikker**

**Standarder:** **skal følges i arbeidet, for eksempel programmeringsstandarder eller navnessandarder**

**Maler, eksempler, sjekklister:** **gir støtte til arbeidet på ulike måter**

# Kvalitetsystemets struktur




Eksempel på kvalitetsystemets hierarkiske oppbygging

# *Innføring av et kvalitessystem*

- **Et kvalitetsystem kan ikke kjøpes ferdig**
  - **En oppskrift på hvordan en skal jobbe**
  - **Uttrykker bedriftens krav til metoder og prosedyrer**
- **Ved å kjøpe systemet kan en risikere å få et A4 system som ikke passer inn i bedriften**

# *Innføring av et kvalitessystem*

- Et kvalitetsystem bør ikke kjøpes ferdig, men elementer kan anskaffer ved kjøp dersom:
  - Behovet for kvalitetsystem er erkjent både av ledelse og ansatte
  - De overordnede prinsippene og holdningene utformes av bedriften selv
  - Bedriften har definert sine krav til de elementene den ønsker å kjøpe
  - Bedriften tilpasser de kjøpte elementene slik at de passer inn i egen bedriftens kultur
  - Kvalitetsystemet innføres på samme måte som om det var egenutviklet

# *Innføring av kvalitetsystem betyr endring*

- Et nytt kvalitetsystem betyr:
  - Nye prinsipper
  - Nye standarder
  - Nye metoder
  - Nye teknikker
  - Nye verktøy

## *Informasjon, informasjon og mer informasjon*

- Skal en lykkes med endring må en informere. Det betyr:
  - Informasjon før en setter igang
  - Informasjon underveis
  - Riktig informasjon
  - Nok informasjon

# *Fellestrekk ved dårlige omstillingsprosesser*

- Leder av prosessen neglisjerer informasjon
- Ingen har ansvar for informasjonsvirksomheten
- Deltakerne i prosjektet vet lite om hvor viktig informasjon er
- Informasjonstiltakene er ikke planlagte. Resultatet er mange ad hok tiltak for å stoppe en ryktebørs
- Lederen vil vente med informasjon til alt er klart. Da har han noe å informere om
- Lederen informerer for tidlig og for mye. Ingen gidder å høre på masse svada

# *Hva bør en informere om*

- Motivere til forandring
- Folk må få lov til å være skeptiske. Det er umulig å instruerer folk til å være entusiaster
- Noen grunnleggende spørsmål som folk vil ha svar på:
  - Hva slags endring skal det bli?
  - Hvilke effekt kommer det til å ha?
  - Hva vil endringen bety for meg?
  - Er jeg i stand til å klare den nye situsjonen

# *Medvirkning*

- Det er viktig å få med seg de rette personene. Her er noen typer som bør være med:
  - **Helter**
  - **Sponsorer**
  - **Endringsagentene**
    - Disse skal gjøre jobben. Her bør en velge de rette personene

# Endringsagentene

- De må være entusiastiske i forhold til det som skjer
- De må ha kompetanse både på problemet og løsningene
- De må være respektert i organisasjonen
- De må ha toppledelsens tillit
- De må representere et tverrsnitt av organisasjonen

# *Utholdenhet*

- Innføring av et kvalitetsystem krever utholdenhet. Det er en prosess som tar lang tid, og det tar enda lenger tid å se effekten av det vi har gjort
  - Innføring 6 mnd.
  - Måle effekt 12 mnd etter innføring

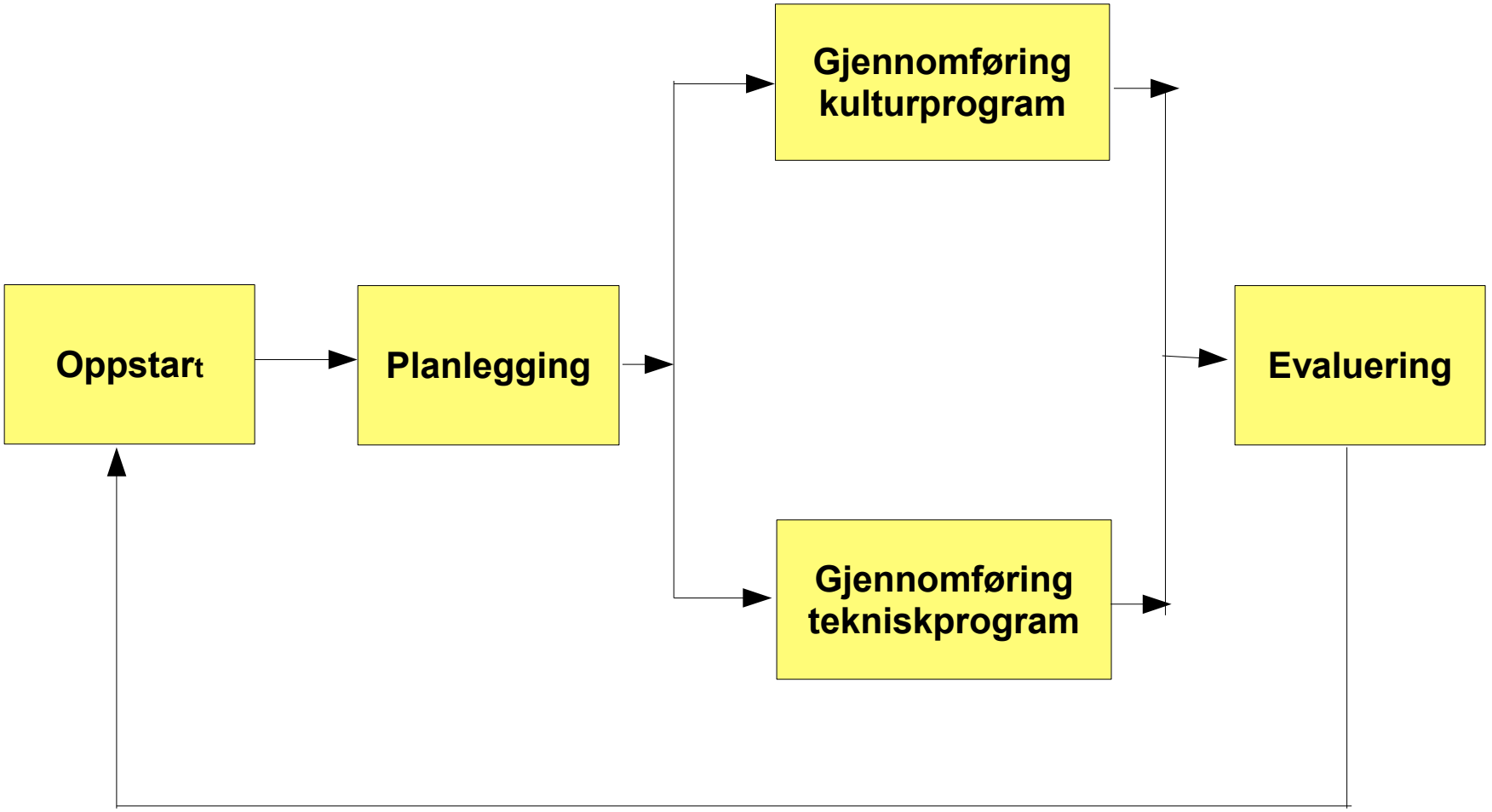
# *Kvalitetsystemet må vedlikeholdes*

- Arbeidet må fortsette etter innføringen
- Vi må måle effekten av systemet
- Gjennomføre forbedringer
- Dette er en vedvarende prosess som krever jevn tilførsel av ressurser

# *Utholdenhet svikter*

- Ledelsen satte i gang uten å vite omfanget av jobben
- Ledelsen satte igang uten egentlig å være overbevist om at dette var riktig
- Det tar lang tid før resultatene viser seg
- Motstanden i organisasjonen er for stor
- Ledelsen skiftes ut
- Ledelsen skifter fokus etter en tid

# *Hovedaktiviteter*



# *Oppstart*

- Ledelsen formulerer en kvalitetspolitikk for IT-avdelingen
- Kvalitetspolitikken må gjøres kjent for alle
- Ledelsen oppretter en støtte organisasjon for innføringen
  - Denne gruppen skal arbeide aktivt med innføringen og må derfor kunne representere ledelse og alle kategorier av ansatte

# *Planleggingen*

- Planleggingen starter ved:
  - Skaffe seg oversikt over status
  - Det gamle (uformelle) systemet kan vurderes opp mot en standard (ISO 9000)
  - Formulering av mål og krav (kan ta utgangspunkt i ISO 9000:2000)

# *Gjennomføring av kulturprogram*

- Skal innføringen lykkes må hele organisasjonen stå bak. Kulturprogrammet prøver å få dette til. Dette er et arbeid der en prøver å endre folks holdninger. Målet er:
  - Øke bevisstheten angående kvalitet
  - Få alle engasjert i arbeidet
  - Sikre at alle har eierskap til det som skjer
- Dette arbeidet består i informasjon og opplæring. Alle må være med på å sette kvalitetsmål for sin egen avdeling.

# *Teknisk program*

- Her er det viktig å få på plass:
  - Krav arbeidet i avdelingene
  - Retningslinjer for arbeidet i avdelingene
  - Standarder for arbeidet i avdelingene
  - Verktøy som skal brukes
  - etc.
- Dessuten må en innføre en livsløpsmodell:
  - ***Den overordnede strategi og rammeverket for hvordan produksjonsprosessen – systemutviklingen – skal gjennomføres.***

# *Livsløpsmodell*

- ***Den overordnede strategi og rammeverket for hvordan produksjonsprosessen – systemutviklingen – skal gjennomføres.***
- ***De andre elementene må støtte opp under livsløpsmodellen.***
  - ***Noen deler av arbeidet trenger prosedyrer (detaljerte beskrivelser)***
  - ***Innfører et begrenset utvalg av verktøy***
  - ***Innfører et måleprogram***
  - ***Gjennomføre opplæring***

# *Evaluering*

- *Et kvalitetsystem må evalueres og følges opp kontinuerlig.*
- *I kap 4 omtalte revisjon av kvalitetsystet*
- *Revisjon gir nyttig informasjon.*
- *Når målesystemet er på plass må vi ha:*
  - *Et opplegg for analyse av data*
  - *Et opplegg for å kunne benytte konklusjonene til noe nyttig*

# *Kontinuerlig forbedring av kvalitetssystemet*

- Kvalitetsystemet bør inneholde (ISO 9000:2000)
  - Prosess for ledelse
  - Prosess for ressursstyring
  - Prosess for produksjon
  - Prosess for måling og forbedring

# *BOHICAS og andre problemer*

- **BOHICAS (Bend Over, Here I Come Again – Stupid)**
- **Noen grunner til at IT avdelinger ikke innfører et kvalitetsystem**
  - **Kostnadene**
  - **Løpet fullføres ikke**
  - **Innføringen saboteres**
  - **Kutter i svingene**
  - **Ledelsen ønsker kun et formelt system**
  - **Man går for grundig til verks**
  - **Systemet må gi effekt med en gang**
  - **Det foretas ingen målinger av de ulike tiltakene**
  - **IT avdelingen får ny ledelse og legger ned hele greia**

# Kapittel 7
# Prosesser og prosessforbedring

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Hovedtese

- *Det er en sammenheng mellom prosessen som fører fram til produktet og kvaliteten på produktet*
- Oppgaver som utføres er prosesser
  - Medarbeidere kan samarbeide om å utvikle deler av et produkt, dette er også en prosess
- Alle prosesser har kunder og levrandører

Universitetet i Stavanger

# *Utvikling av programvare og prosesstankegang*

- Programvare blir utviklet
- Det gjøres i en eller annen form for utviklingsprosess

Universitetet i Stavanger

# Hva er en prosess

- En prosess er noe som skjer trinnvis over tid.
- En prosess er en logisk kjede av logisk sammensatte og repeterbare aktiviteter som utnytter virksomhetens ressurser til å foredle et objekt med det formål å oppnå spesifiserte og målbare resultater/produkter til internt eller eksternt bruk.

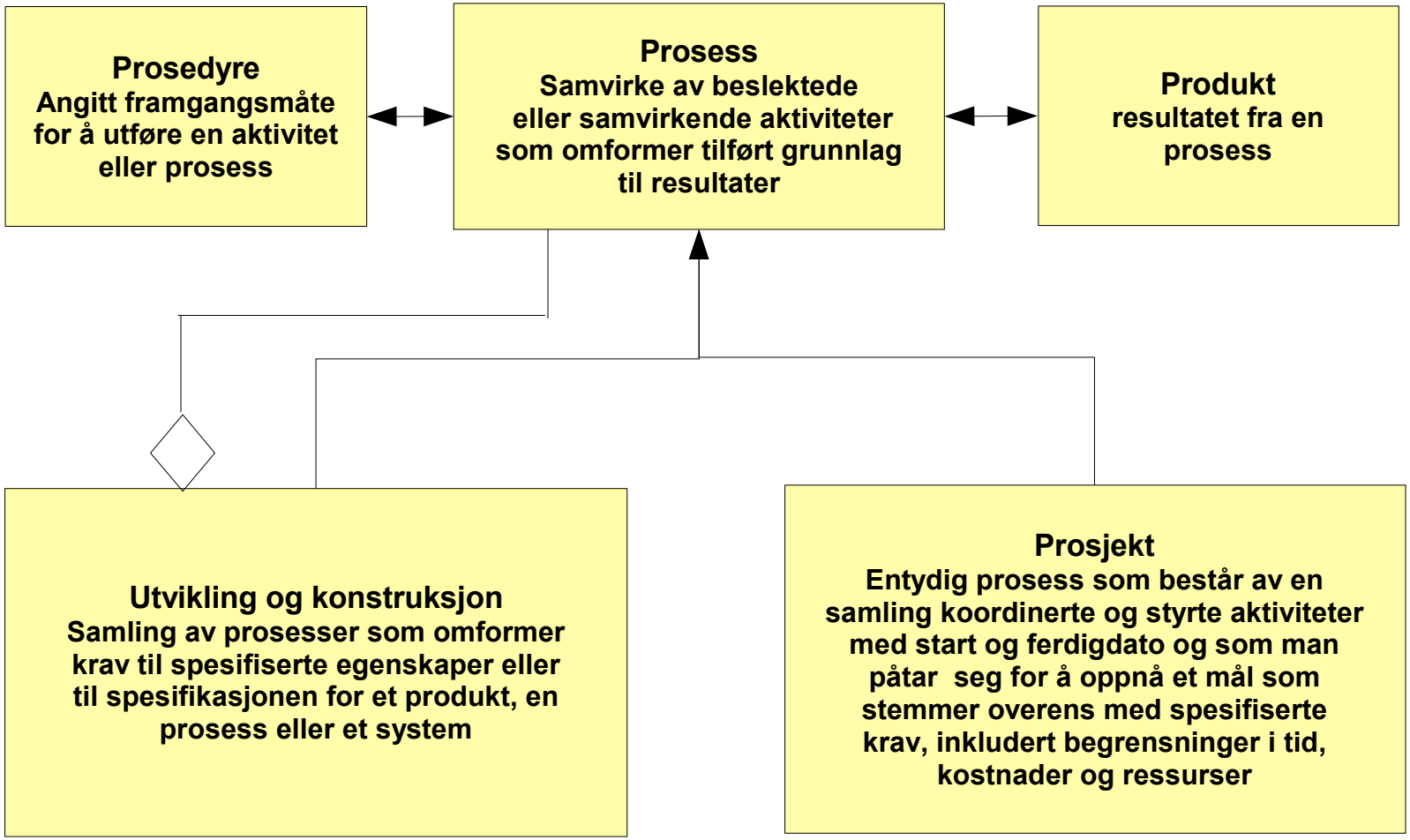
Universitetet i Stavanger

# *Programvareprosess*

- *Alle oppgaver som er støttet av verktøy, standarder, metoder og praksis og som inngår i produksjon og evaluering av programprodukter gjennom deres livsløp*
- *Programvareprodukter er:*
  - *Prosjektplan*
  - *Kravspesifikasjon*
  - *Dokumentasjon av design*
  - *Dokumentasjon av kode*
  - *Tester*
  - *Brukerdokumentasjon*

Universitetet i Stavanger

# *Sammenheng mellom prosessbegreper*



Universitetet i Stavanger

# *Livssyklusprosesser.*

- *Primær livssyklusprosesser*
- *Støttelivassyklusprosesser*
- *Organisasjonslivsyklusprosesser*

Universitetet i Stavanger

# *Primære livsyklusprosesser*

- Primære livsyklusprosesser
  - Anskaffelsesprosessen
    - Definerer aktiviteter ved anskaffelse av programvaren
  - Leverandørprosessen
    - Definerer aktiviteter til leverandøren av programvare
  - Utviklingsprosessen
    - Definerer aktiviteter ved utvikling av programvare
  - Forvaltningsprosessen
    - Definerer aktiviteter ved forvaltning og drift av programvare
  - Vedlikeholdsprosessen
    - Definerer aktiviteter ved vedlikehold og modifikasjoner av programvare

Universitetet i Stavanger

# *Støttelivsyklusprosessen*

- Dokumentasjonsprosessen
  - Definerer aktiviteter for å ta vare på informasjon som produseres i en livsløpsprosess
- Konfigurasjonsprosessen
  - Definerer konfigurasjonsaktiviteter
- Kvalitetssikringsprosjekt
  - Definerer de aktivitetene som skal sørge for objektivt å sikre at produktet og prosessen er i samsvar med spesifiserte krav, og at planer følges
- Verifikasjonsprosessen
  - Definerer aktiviteter for å verifisere programvaren.

Universitetet i Stavanger

# *Støttelivsyklusprosessen*

- Valideringsprosessen
  - Definerer aktiviteter for å valider programvare
- Inspeksjonsprosessen
  - Definerer aktiviteter for å evaluere status til produkter og aktiviteter
- Revisjonsprosessen
  - Definerer aktiviteter for å finne ut om ting gjøres i samsvar med krav, planer og kontrakter
- Problemløsningsprosessen
  - Definerer aktiviteter for å analysere og fjerne problemer

Universitetet i Stavanger

# *Organisasjonslivssyklusprosessen*

- Ledelsesprosessen
  - Definerer aktiviteter for ledelse av livsløpsprosessen
- Infrastrukturprosessen
  - Definerer aktiviteter for å få på plass den underliggende struktur for en livsløpsprosess
- Forbedringsprosessen
  - Definerer aktiviteter for å få på plass måling, kontroll og forbedring av livsløpsprosessen
- Opplæringsprosessen
  - Definerer aktiviteter som skal sørge for at personalet får skikkelig opplæring

Universitetet i Stavanger

# *Andre modeller*

- Primærprosesser (verdiskapningen)
- Støtteprosesser  (nødvendig støtte for primærprosessene)
- Utviklingsprosesser (sørger for at organisasjonen overlever på lang sikt)

Universitetet
i Stavanger

# *Huphrey 1989*

**Et viktig første steg på veien mot løsningen av problemene med programvare er å håndtere hele oppgaven som en prosess som kan kontrolleres, måles og forbedres.**

Universitetet i Stavanger

# *En prosess leverer et produkt*

En prosess leverer et produkt, men produktene fra samme prosess er aldri helt like.

Eksempel:

- Turen til jobben/skolen
  - Bruker ulik tid, selv om transportmiddelet er det samme

Universitetet i Stavanger

# *Egenskaper ved prosesser*

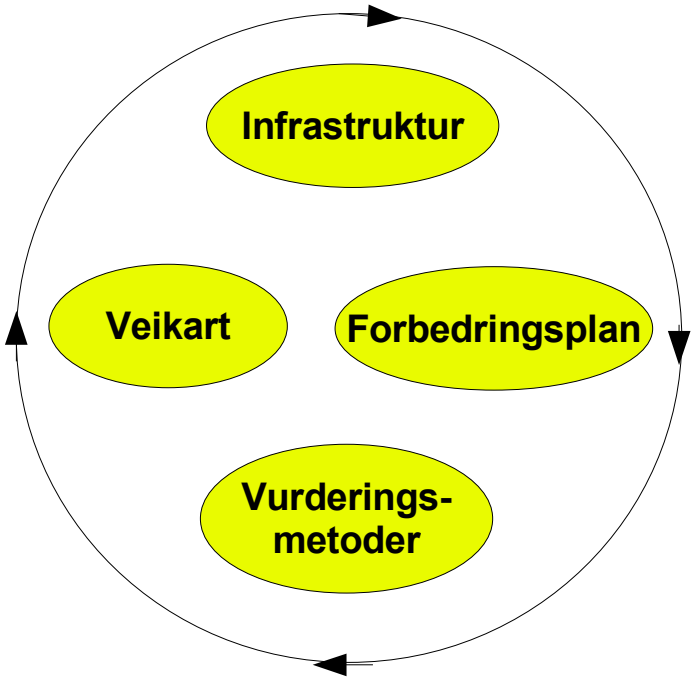
- En prosess viser variasjon
- Prosesser som viser naturlig variasjon er stabile
- En prosess har ytelse og dugelighet
  - Ytelse: Forteller hva prosessen kan levere
  - Dugelighet: Kan prosessen levere innenfor kravene som stilles

Universitetet i Stavanger

# *Prosessforbedring krever rammeverk*

- Prosessforbedring krever at en har en metodikk/rammeverk som kan brukes for å styre forbedringstiltakene. Målet er å få på plass en programvareprosess som:
  - Gjør det mulig å gjenta tidligere suksesser
  - Gjør det mulig å finne årsaken til feil når feil oppstår, og å hindre at feil blir gjentatt
  - Gjør det mulig å innføre et aksepterte statistisk rammeverk som gjør det mulig å forutsi varighet og ressursbruk i utviklingsprosjekter som følger prosessen
  - Er veldefinert, og som kan tilpasses alle typer utviklingsprosjekt
  - Er kjent av alle
  - Blir fulgt av alle

# *Et rammeverk for prosessforbedring*




- Fire elementer:
  - Infrastruktur
  - Forbedringsplaner
  - Vurderingsmetoder
  - Veikart

Universitetet i Stavanger

# *Infrastruktur*

- Infrastrukturen er fundamentet en bygger prosessen på
  - Organisasjonskulturen
  - Standarder
  - Opplegg for opplæring
  - Verktøy
  - Retningslinjer

Universitetet i Stavanger

# *Infrastruktur Prosessforbedringsgruppen*

- Organisatorisk infrastruktur
  - Mennesker som holder prosessforbedringen i gang
- Roller
  - Sponsorer:
    - Representanter for toppledelsen
  - Helter:
    - Drivkreftene som initierer prosessen
  - Endringsagentene:
    - Dette er de som gjør jobben

Universitetet i Stavanger

# *Infrastruktur*
# *Gruppens hovedoppgaver*

- Initiere
- Støtte

- Dette innebærer at den må:
  - få på plass en prosessstandard
  - vedlikeholde en prosessdatabase
  - sørge for introduksjon av ny teknologi
  - sørge for grunnleggende opplæring
  - være konsulenter
  - vurder om forbedringsmålene er nådd, og rapportere status

Universitetet i Stavanger

# *Infrastruktur*
# *En programvareprosess bør endres trinnvis*

- Identifisere problem
- Prioritere
- Lage handlingsplaner
- Oppnå enighet om målene
- Sette folk i arbeid
- Sørge for opplæring og veiledning
- Sette i gang gjenomføringen
- Følge opp framgang
- Løse problemer som dukker opp

Universitetet i Stavanger

# *Teknisk infrastruktur*

- Den tekniske plattformen
  - Datamaskiner
  - Verktøy
  - Ulike typer hjelpemiddel

Universitetet i Stavanger

# *Vurderinger*

- Hjelper til i forpedringsprosessen
- Tar utgangspunkt i:
  - En prosessmodell og et veikart / hvilke tilstand er virksomheten i?
  - Har ledere og utviklere den nødvendige opplæring?
  - Har ledelsen forpliktet seg på å følle opp prosessen?
  - Finnes det et system for måling av resultater?
  - Finnes det et system for tilbakemelding?
- Skal finne kritiske problemer og foreslå forbedringer

Universitetet i Stavanger

# *Veikart*

- Et skal hjelpe oss å finne fram!
- Det er nødvendig å ha en modell å støtte seg på når veien framover stakes ut.
  - CMM modellen har:
    - Fire trinn
    - Fem modenhetsnivåer
  - ISO/IEC 1504 har:
    - En rekke prosesser som befinner seg på forskjellig modenhetsnivå
    - Ved å se på hver enkelt prosess kan en lage en modenhetsprofil for organisasjonen
    - Denne profilen sier noe om organisasjonens totale modenhet

Universitetet i Stavanger

# *Planlegging*

- Nøkkelen til suksess er:
  - Planlegging
  - Gjennomføring
  - Kommunikasjon
- Prosessen må gå som et skikkelig prosjekt. Hovedhensikten er å gjennomføre endring. Prosjektet må ha mål og milepæler som er klart uttrykt.

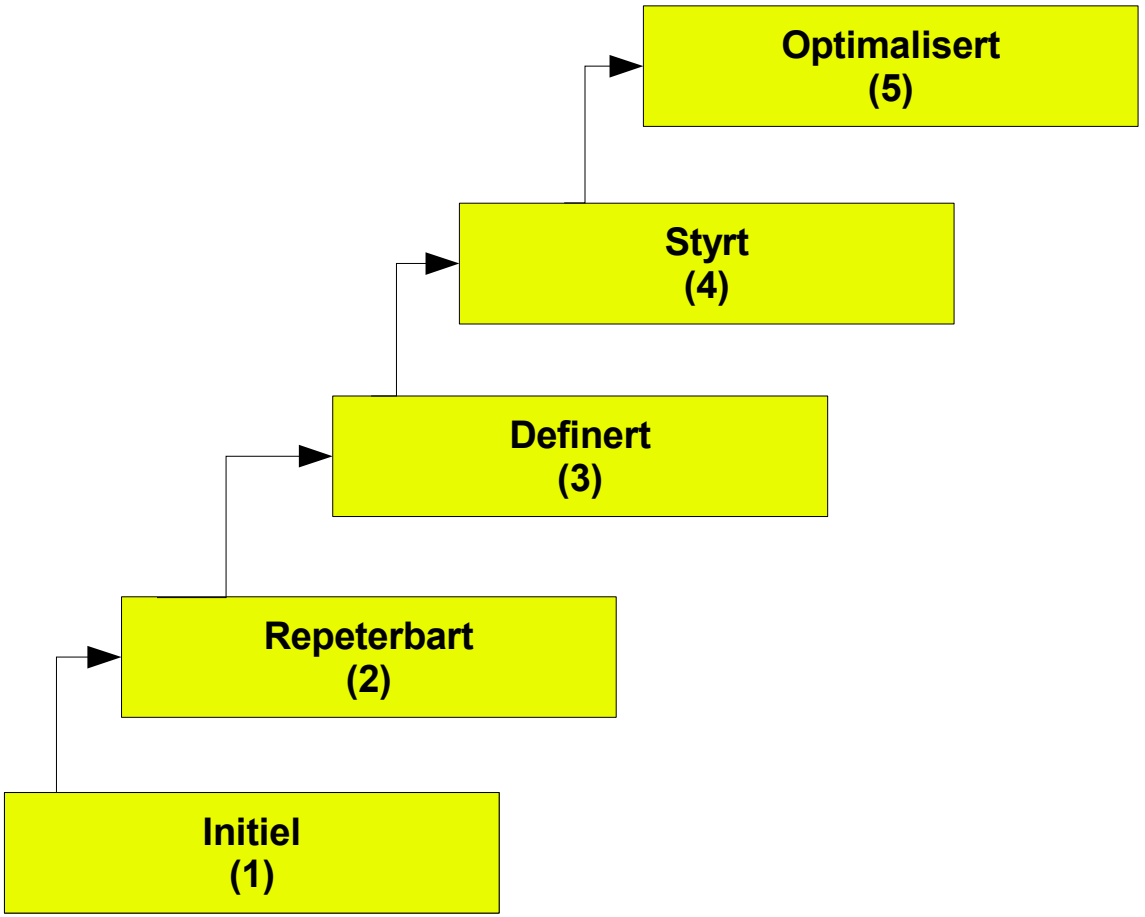
Universitetet i Stavanger

# *Noen modeller*

- *CMM*
  - *Stammer fra det amerikanske forsvarsdepartementet*
  - *Software Engineering Capability Maturity Model (SE-CMM)*
- *ISO/IEC 15504*
  - *Dette er et alternativ til CMM (ikke ferdig)*
  - *Europeisk inspirert*

Universitetet i Stavanger

# *CMM modellen*



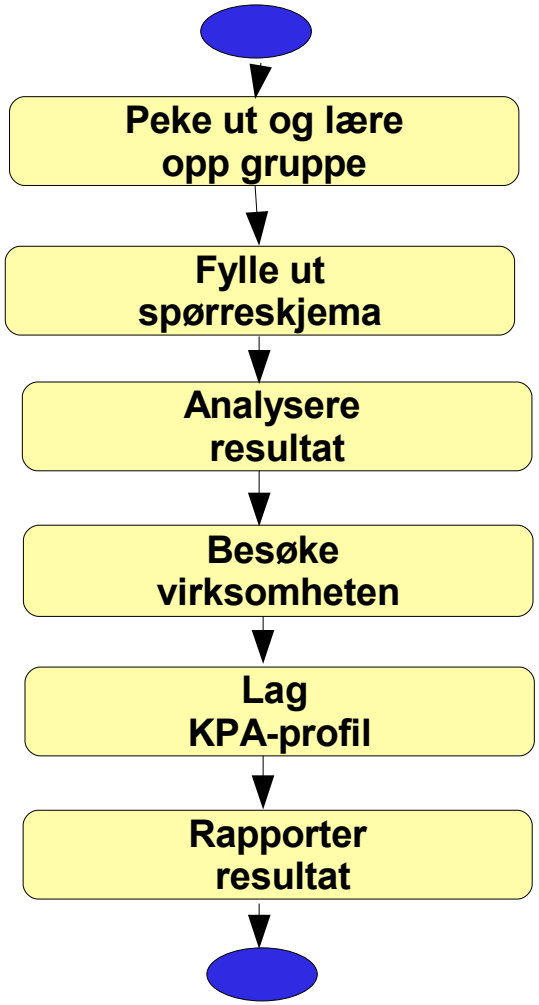
Universitetet i Stavanger

# *De ulike nivåene*

- ***Initielt nivå***
  - ***Ingen formelle prosedyrer***
  - ***Bruk av verktøy er tilfeldig***
  - ***Det som foregår er ad hoc***
  - ***Toppledelsen forstår ikke problemet***
- ***Repeterbart nivå***
  - ***Kontroll over prosjektplanlegging og oppfølging***
  - ***Har et opplegg for kvalitetsikring og endringskontroll***
  - ***Kostnad og tidsforbruk estimeres***
  - ***Tidligere suksesser kan gjentas***
- ***Definert***
  - ***En definert prosess er dokumentert og standardisert***
  - ***Den er integrert i organisasjonen, kan tilpasses enkelte prosjekt***
- ***Styrt nivå***
  - ***Man kan måle kvaliteten av prosesser og produkter***
  - ***En prosessdatabase er på plass***
  - ***Nødvendige ressurser til å gjennomføre forbedringer er på plass***
- ***Optimalisert nivå***
  - ***Nå kan kontinuerlig forbedring gjennomføres***
  - ***Årsakene til feil blir funnet og fjernet***
  - ***Analyse av innsamlede data kan gjøres automatisk***
  - ***Ledelsen har forstått viktigheten av å analysere prosesser.***

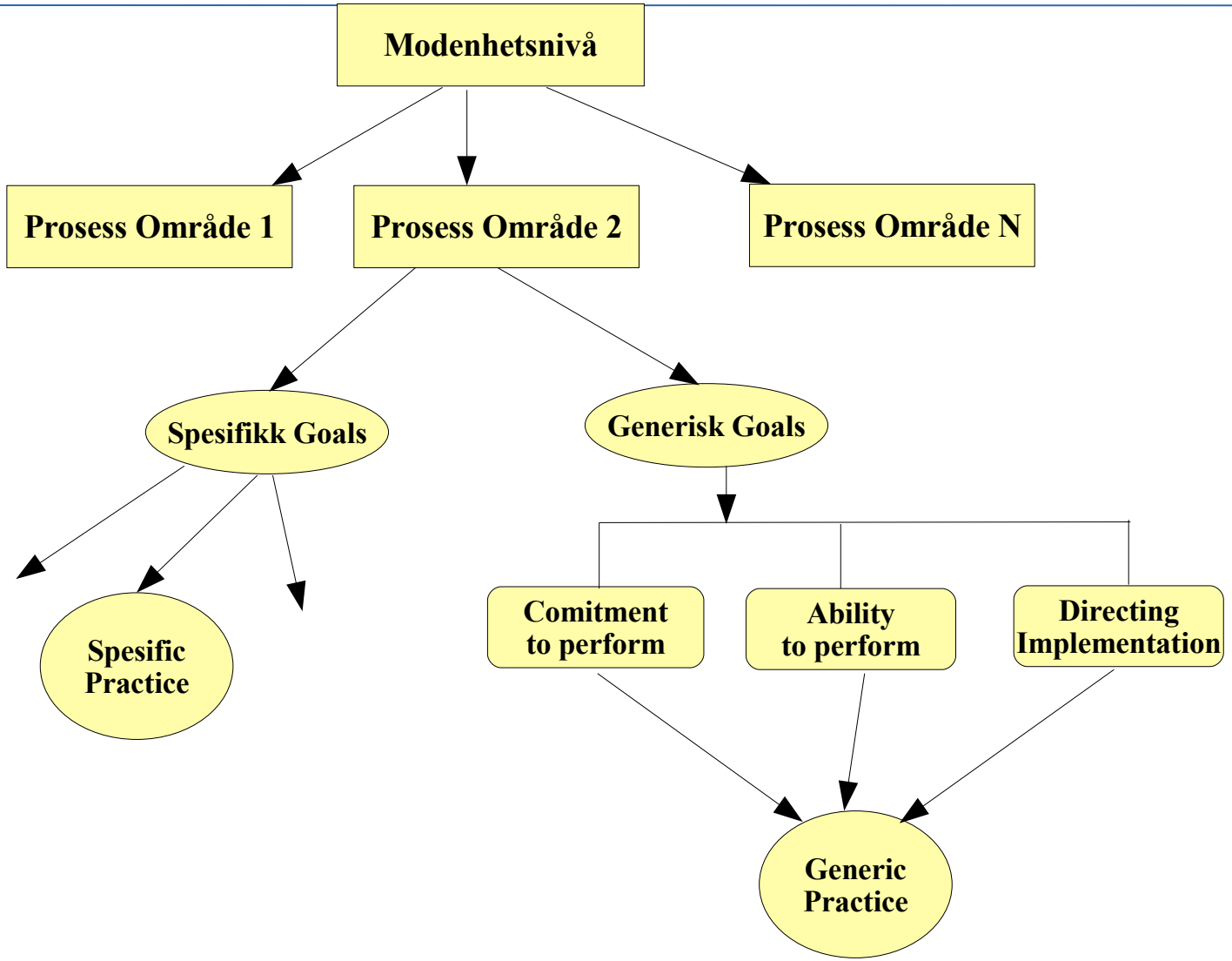
Universitetet i Stavanger

# *Prosessvurdering CMM*



Universitetet i Stavanger

# *Prosessvurdering CMM*



Universitetet i Stavanger

# *Fellestrekk ved nivåene i CMM modellen*

- Hvert nivå har en del prosesser som skal gjennomføres
- Hver prosess har noen spesifikke mål og noe generiske mål
- De generiske er kalt generiske for de er felles for mange prosesser
- Spesifikke mål gjelder kun for enkelte prosesser
- En spesifikk praksis er noe som er viktig for å oppnå et spesifikt mål
- Generisk praksis er metoder for å kunne gjenta prosessene

Universitetet i Stavanger

# *ISO/IEC 15504 modellen*

- Dette er et alternativ til CMM, men den er inspirert av CMM. Arbeidet med denne standarden starter i 1991, en pilotversjon ble sluppet i 1998. Denne modellen er Europeisk inspirert, den er ikke ferdig.
- Standarden kan brukes til:
  - å bestemme dugelighet
  - å forbedre prosesser
  - Selvvurdering

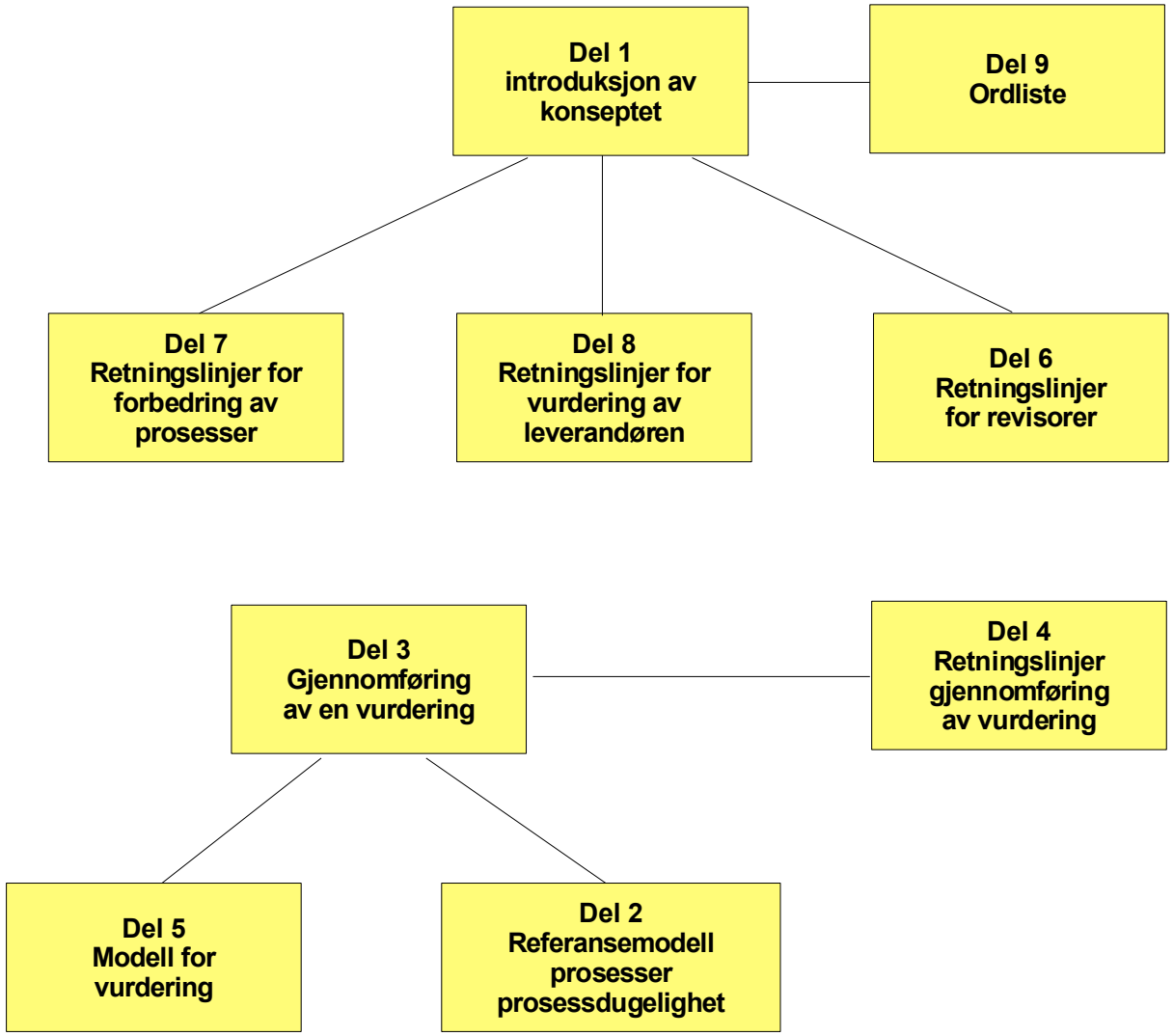
Universitetet i Stavanger

# *ISO/IEC 15504 modellen*

- Aktuelle brukere av standarden er:
  - innkjøpere som vil bestemme dugeligheten til en leverandør
  - programleverandører som vil gjøre vurdering av egen prosess
  - utviklingsgruppe som ønsker et verktøy for kontinuerlig forbedring
  - ledere som vil sikre at prosessen bidrar til at virksomheten når sine overordnede mål
  - revisorer som har behov for et rammeverk for å gjennomføre vurderinger

Universitetet i Stavanger

# *ISO/IEC 15504 modellen*



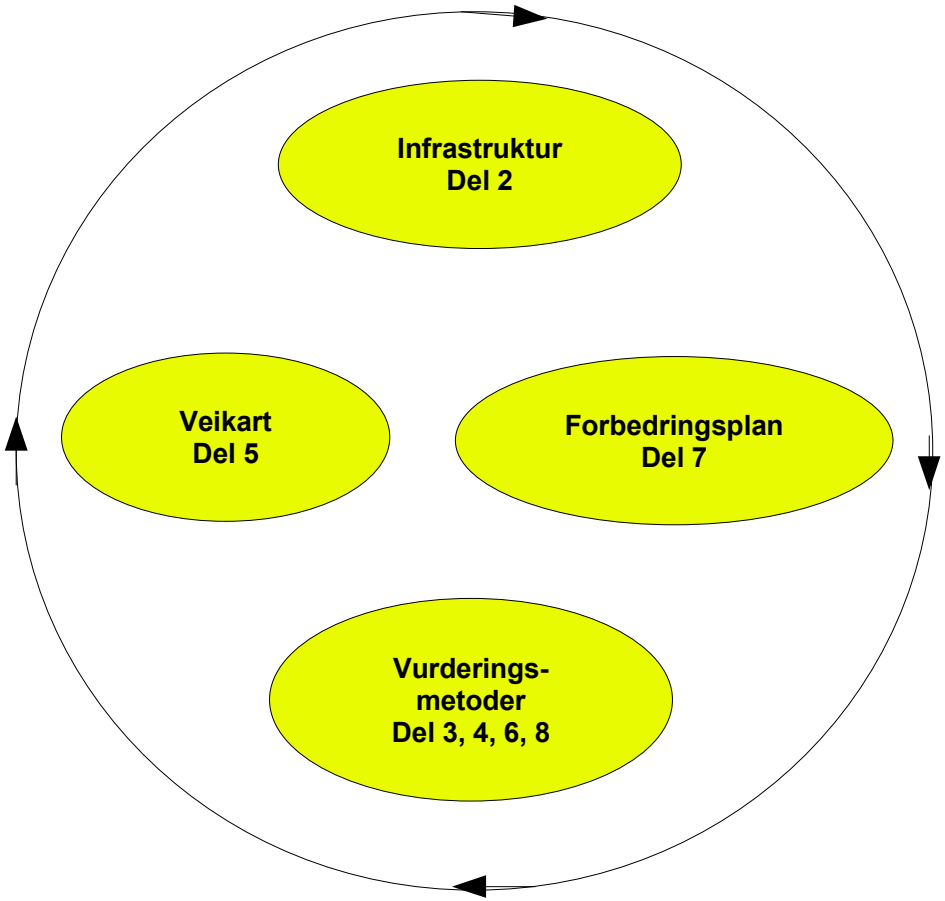
Universitetet i Stavanger

# *ISO/IEC 15504 modellen*

- Del1: Introduserer konsepter. Forklarer hvordan ting henger sammen
- Del2: Her beskrives referansemodellen
  - dim1: beskriver deleprosessene
  - dim2: modenhetsnivåene til prosessene
- Del3: Tar for seg gjennomføring av vurdering
- Del4: Retningslinjer for gjennomføring av vurdering
- Del5: Modell for gjengføring av vurdering
- Del6: Kvalifisering av revisorer
- Del7: Bruk av data for å forbedre prosesser
- Del8: Bruk av data for å vurdere dugelighet.
- Del9: Ordliste

Universitetet i Stavanger

# *ISO/IEC 15504 rammeverk for prosess forbedring*



Universitetet i Stavanger

# *Første dimmensjon Prosessdimmensjonen Totalt 29 prosesser*

- CUS (Custom Supplier Process) Prosesser knyttet direkte mot kunder
- ENG (Engeneering Processes) Spesifikasjon, implementasjon, vedlikehold av system og programvare, brukerdokumentasjon
- SUP (Supported Processes) Nødvendige støtteprosesser
- MAN: (Management Processes) Generiske praksis som kan brukes av ledere ved utvikling og vedlikehold ac programvare
- ORG (Organization Processes) Setter virksomhetsmål, fremskaffer produkter og ressurser

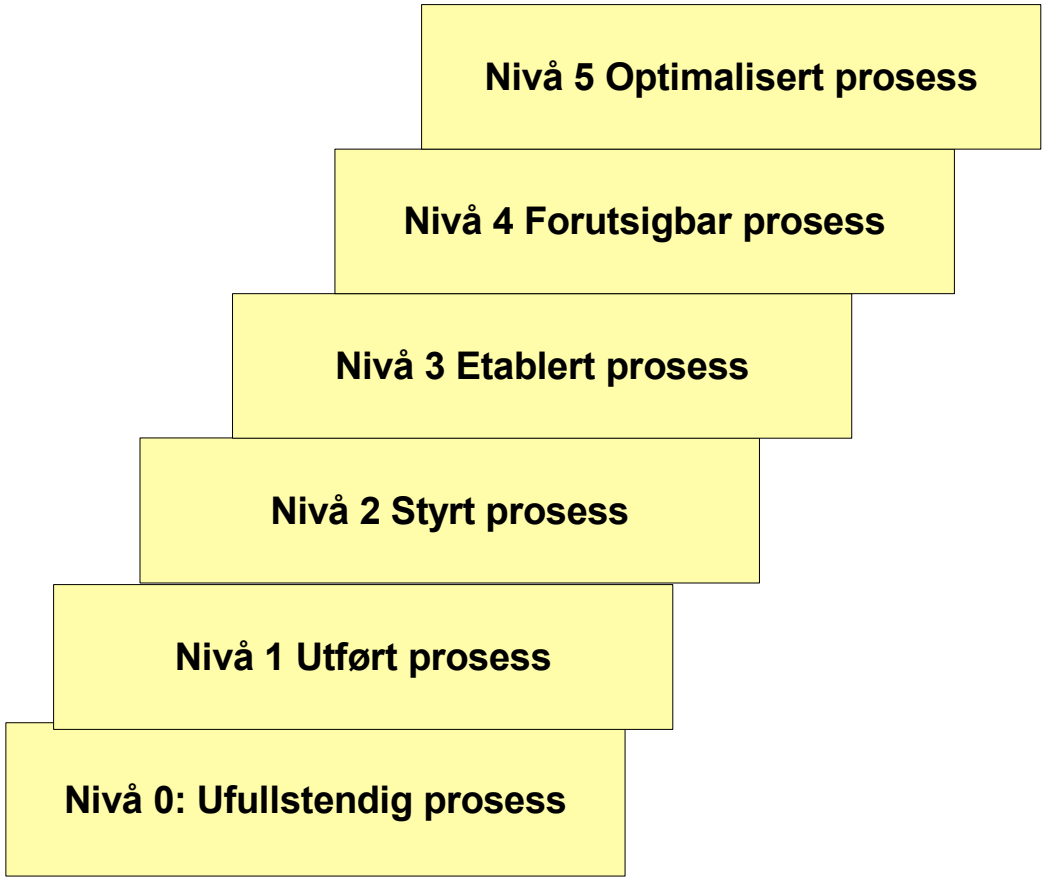
Universitetet i Stavanger

# *Andre dimmensjon modenhetsnivå*

- En prosess kan være på seks ulike duglighetsnivå
  - Nivå 0: oppnår sjelden det en ønsker
  - Nivå 1: Alt er ikke nøye planlagt, men oppnår som regel det en ønsker
  - Nivå 2: Produserer produkter av akseptabel kvalitet, innenfor tid og kostnad. Prosedyrer følges
  - Nivå 3: Her brukes definerte prosesser basert på sunne prinsipper for programutvikling
  - Nivå 4: Ytelsen ligger alltid innenfor definerte kontrollgrenser. Ytelsen måles og analyseres.
  - Nivå 5:  Prosessen er optimalisert med hensyn på framtidige behov.

Universitetet i Stavanger

# *ISO/IEC 15504 modenhetsnivåer*



Universitetet i Stavanger

# *Rangering av prosesser*

- ISO/IEC 15504 rangerer prosesser etter en gitt skala
  - N: ikke oppfylt
  - P: Delvis oppfylt
  - L: Stort sett
  - F: Fullstendig oppfylt
- Ved å se på alle prosessene får en en slags modenhetsprofil for hele organisasjonen

Universitetet i Stavanger

# *SPIQ – en norsk tilpasning til prosessforbedring*

- Både ISO modellen og CMM modellen er omfattende og krevende
- Det viktigste med CMM er målene ikke at en har oppnådd alle modenhetsnivåene.
- CMM kan tilpasses norske bedrifter,
- Det samme gjelder ISO/IEC modellen.
- SPIQ er en norsk modell.

Universitetet i Stavanger

# *SPIQ*

- SPIQ: Software Process Improvement for better Quality
- Prosjektet startet i 1997 og ble avsluttet i 1999
- SINTEF, NTNU og Universitetet i Oslo
- Målsetting:
  - Spre kunnskap om prosessforbedring til norsk IKT-industrien

Universitetet i Stavanger

# *SPIQ*

- Kvalitet på programprodukter må komme som et resultat av en forbedring av de prosesser som brukes ved utvikling av programvare
- Dette forutsetter:
  - Formalisme knyttet opp mot prosesser
  - Læring
    - Metodehåndbok
    - Temagrupper
    - kurs og seminarer
    - undervisning ved universiteter
    - Informasjonsbank (http://www.geomatikk.no/spiq)

Universitetet i Stavanger

# *Hovedelementer i SPIQ*

- SPIQ baserer seg på TKL
- SPIQ er faktabasert hovedredskapet er «Goal–Question-Metrices» (GQM)
- Det er ikke nødvendig at en prosess er stabil før en begynner og forbedre den
- Det legges ikke vekt på at bedriftene skal kunne gå fra et modenhetsnivå til neste
- Virksomhetene må plukke ut ting som virker for dem

Universitetet i Stavanger

# *Plukke ut prosesser for forbedring*

- Hvor er vi?
  - Identifisere virksomhetens prosesser
  - Finne sterke og svake sider ved disse prosessene
- Hvor vil vi?
  - Sette mål
- Hvordan kommer vi dit? (Veikart)
  - Identifisere og gjennomføre forbedringstiltak
  - Vurdere effekten av tiltakene
  - Institusjonalisere tiltak som har den ønskede effekt

Universitetet i Stavanger

# *Hovedelementer i SPIQ*

- Metodehåndbok
  - Rammeverk for forbedring
    - baserer seg på etablerte metoder for prosessforbedring
    - løpende utviklingsplaner
    - langsiktig forretningstrategi mot forretningsmessige mål

# Forbedringstiltak må drives kontinuerlig

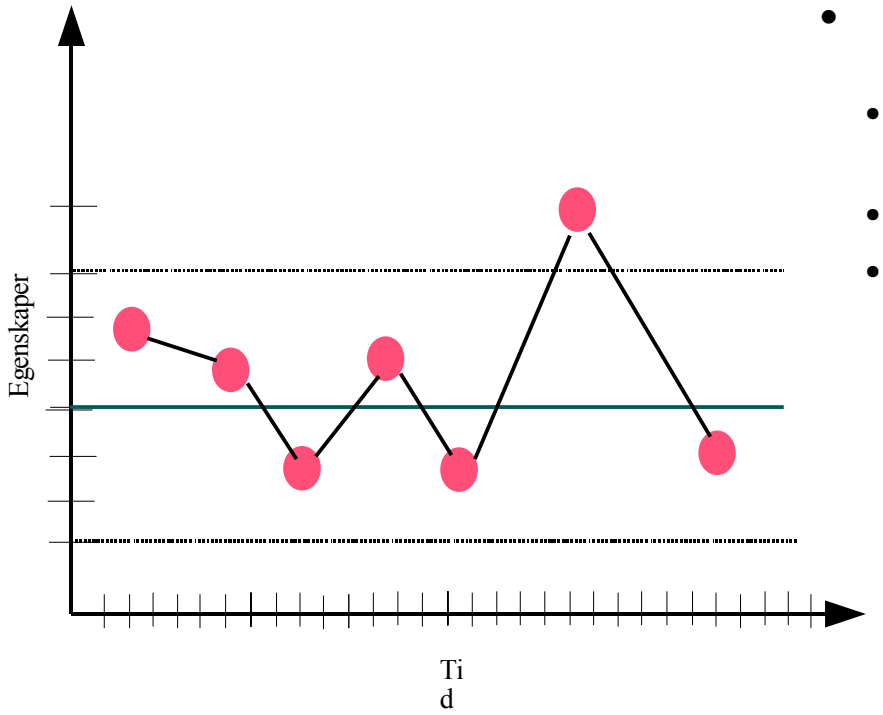
- To nivåer
  - Prosjektnivå
    - pilotprosjekter, positive resultater overføres til virksomhetsnivået.
  - Virksomhetsnivå

Universitetet i Stavanger

# Statistisk prosesskontroll

- TKL sin hovedtese er bort fra sluttkontroll, man må hindre at feil oppstår! En må fokusere på produksjonsprosessen
- Tese:
  - Kvaliteten på prosessen avgjør kvaliteten til produktet.
- Det er viktig å oppdage problemer tidlig, da er de lettest å rette
- Kunnskap om prosesser må fremskaffes ved hjelp av vitenskaplige metoder
- Fakta må framskaffes ved hjelp av matematikk
- SPC er et redskap til å finne ut hva en prosess er i stand til å produsere

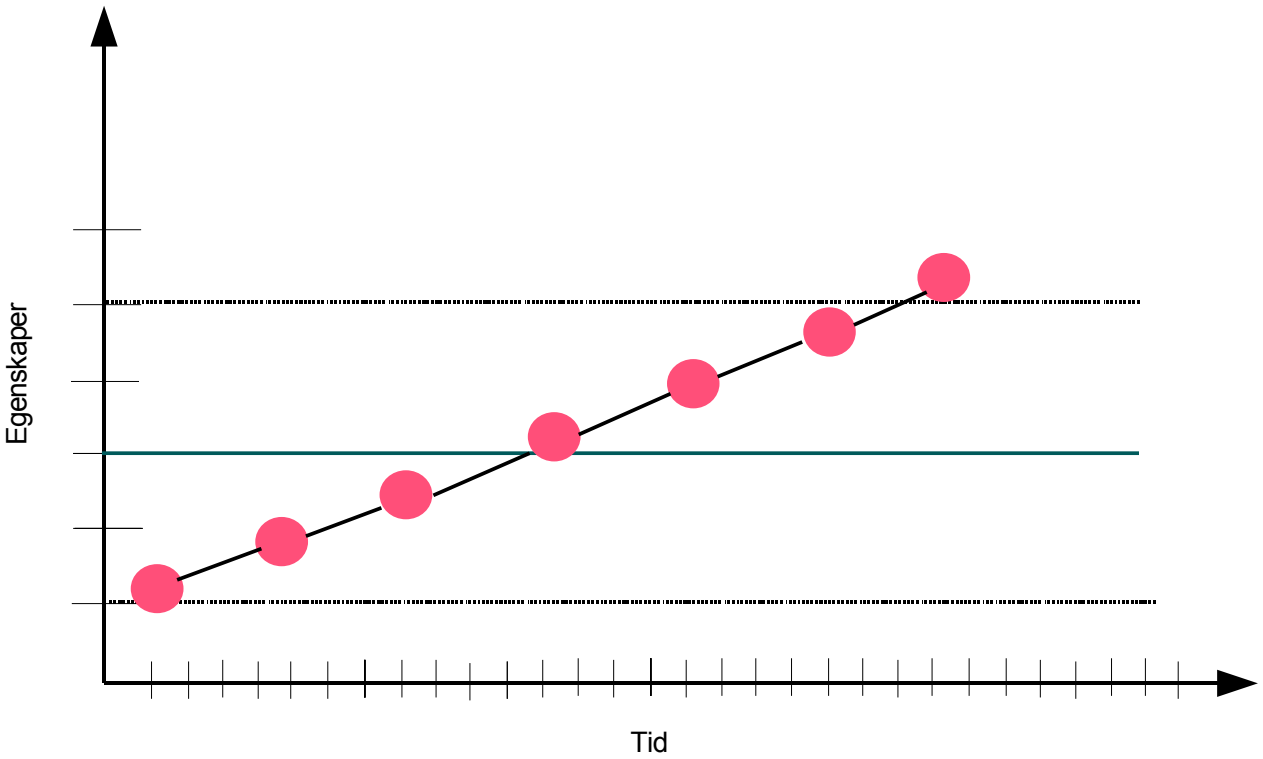
# Prosess




- **Stabil / Ustabil**
- **Et kontrolldiagram kan avdekke om det er problemer med en prosess**
  - **Langs Y-aksen er en eller annen egenskap ved prosessen**
  - **Langs X–aksen plottes tiden**
  - **Tre horisontale linjer**
    - **Sentrallinjen er statistisk middel**
    - **De to øverste representere spredningen. Grensen fastsettes som regel som**
      - **tre standardavvik**
    - **Dersom noen stikkprøver faller utenfor, tyder det på at noe spesielt har hendt**

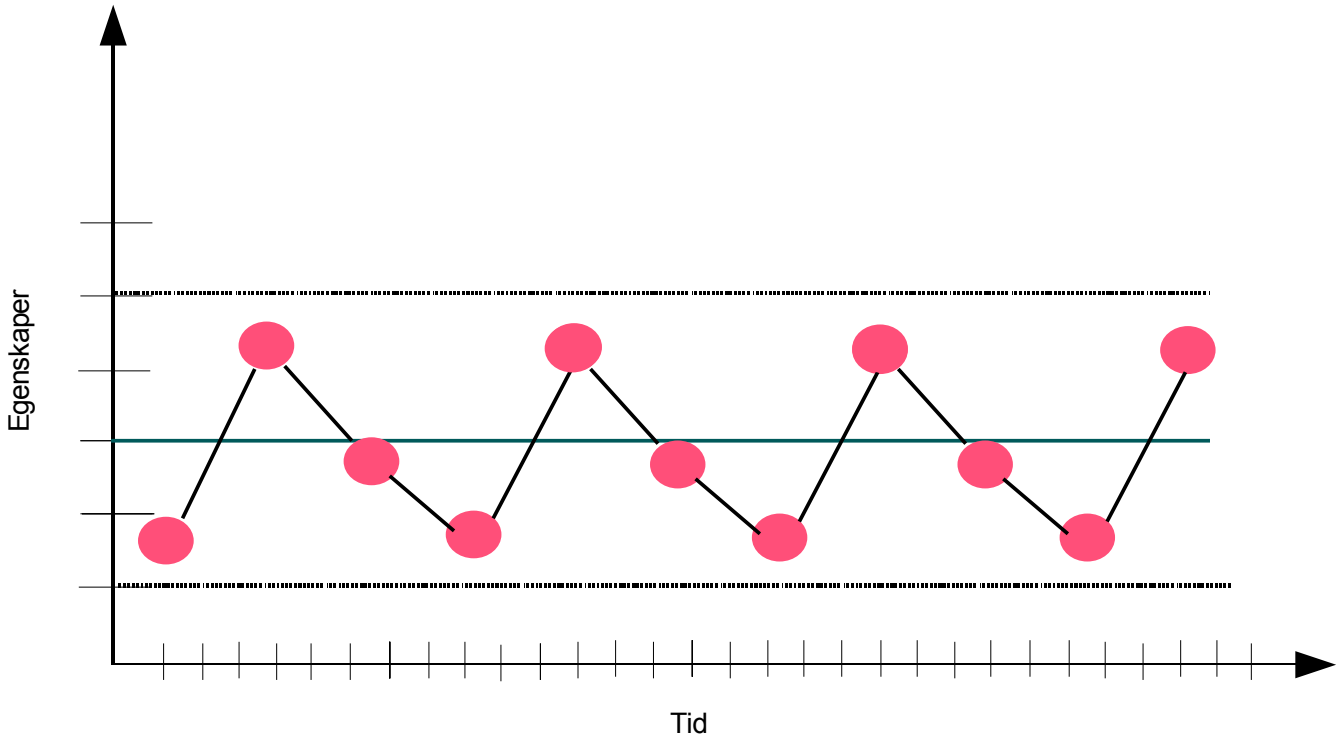
Universitetet i Stavanger

# Trend



Universitetet i Stavanger

# Syklisk



Universitetet i Stavanger

# Hvordan tegnes kontrolldiagrammet

- Variabelkontrolldiagram når dataene er målbare, må velge hvilke fordeling som benyttes
  - Lengde
  - Diameter
  - Vekt
- Atributtkontrolldiagram, må velge hvilke fordeling som benyttes
  - Antall defekter
  - Antal suksesser

Universitetet i Stavanger

# Er SPC alltid svaret

- Dette er en omstridt teknikk innenfor programvareutvikling.
- CMM argumenterer sterkt for SPC
- Utvikling av programvare er ikke produksjon
- Utvikling av et programsystem er som regel en prosess under stadige endringer, det gjør at SPC har begrenset verdi

# Kapittel 8
# Best praksis

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Best praksis

*Definisjon*

*Best praksis er: en arbeidsmåte som er velprøvd i industrien (i mange prosjekter), og som gir gode resultater*

Best praksis vil

- Gjøre jobben ved systemutvikling enklere

Universitetet i Stavanger

# *Hvordan finne best praksis*

- Høre på andre og prøve
- Systematis innsamlig av data for å finne ut hva som fungerer
- Det finnes firma som systematisk samler inn data fra mange ulike utviklingsmiljø

# *Kriterier som blir brukt for å finne ut om noe er best praksis (Caper Jones)*

- Har gitt resultater som produktivites og kvalitetsmessig er blant de 5% beste av resultatene i industrien
- Er minst 15% bedre enn resultater vi oppnår uten gitt framgangsmåte
- Er vurdert som nyttig av dem som bruker dem
- Har ikke fororsaket større problemer i noen prosjekter
- De praksiser som Caper Jones definerer som de beste har vist seg å holde, de er velprøvde.

# *Caper Jones velger best praksis ut fra et omfattende tallmatriale*

- En vurdering av firmaets framgangsmåte når det utvikler systemer
- En vurdering av de resultater de oppnår
- Hvilke framgangsmåter benytter de som får det best til?

# *For å kunne forbedre må en ha noe å vurdere mot*

- ISO 9000 sertifisering er en form for vurdering
- Vurdere ut fra CMM modellen
- Vurdere ut fra ISO/IEC 15504 modellen
- Det som trengs er:
  - Dokumentasjon av den praksis som følges
  - Noe å samenligne praksis mot som vi mener gir best praksis ( en standard)
  - Formelle metoder for å gjøre samenligningen
  - Noen som kan gjenomføre samenligningen og som er kompetente til å konkludere og gi anbefalinger

# Vurdering

- En kvalitativ vurdering gjennomføres ofte som:
  - Konsulterende revisjon
- Vanlig framgangsmåte:
  - Ekstern konsulent besøker firmaet som skal revideres
    - Samler inn data
    - Interjuer
    - Spørreundersøkelse
    - Undersøker hva som faktisk skjer
- Sluttprodukt
  - En beskrivelse av situasjonen
  - Anbefaling om videre arbeid med å forbedre praksis

Universitetet i Stavanger

# *Kvantitativ vurdering av produktivitet og kvalitet*

- *En kvantitativ vurdering prøver å sette pålitelige tall på en IT-avdelings produktivitet og på kvaliteten på det produktet som produseres.*
- *En samler inn:*
  - *Fakta om en avdeling*
  - *Karakteristika for prosjekter*
- *En vanlig metode som benyttes er benchmarking.*

Universitetet i Stavanger

# Benchmarking

- *Et firma som driver systemutvikling velger å delta i benchmarking*
- *Det vil si at de gir egne data til et benchmarking-firma*
  - *Gartner Group*
  - *Meta Group*
  - *Dr. Howard Rubin's annual benchmak report*
  - *...*

Universitetet i Stavanger

# Benchmarking

Selskapet selv samler inn data, eller får hjelp til å samle inn data. Firmaet får tilbake en rapport som viser egne resultater på en rekke definerte områder. Dessuten får en vite hvordan en ligger ann i forhold til andre.

Det er viktig at slike resultater behandles konfidensielt.

Siden firma rapporterer selv, kan de jukse!

Universitetet i Stavanger

# *Det er størrelsen det kommer an på*

- I forbindelse med kvantitativ vurdering må vi kunne måle størrelsen på produktet – på IT-systemet
- Vanlige måter er:
  - Antall funksjonspoeng (FP)
  - Antall kodelinjer (LOC)
  - OO-metrikk (objektorienterte mål)

Universitetet i Stavanger

# *Det vanligste målet er å telle funksjonspoeng*

- *Her er noen eksempler på funksjonpoeng for noen typer applikasjoner*
  - *web-applet 50FP*
  - *dataspill 3000FP*
  - *databasesystemer 7500FP*
  - *informasjonsystem 10000-25000FP*
  - *operativsystem 75000FP*

Universitetet i Stavanger

# *Prosjektfaktorer som har betydning*

- *Best praksis for et lite prosjekt trenger ikke være best praksis for et stort. Skal vi kunne trekke konklusjoner ut av en best praksis studie må vi vurdere:*
  - *type programvare*
  - *Kompleksitet*
  - *Rammebetingelser*
  - *type prosjekt*
  - *prosjektdeltakernes kvalifikasjoner*

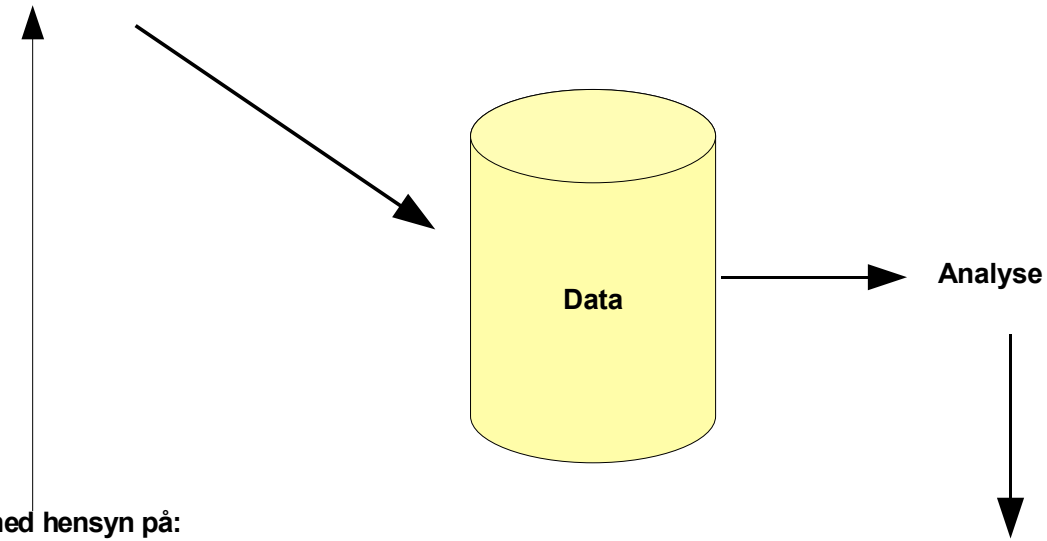
Universitetet i Stavanger

# *Hvordan måles produktivitet og kvalitet*

- *Dette kan gjøres ved:*
  - *produktivitet – i funksjons poeng / månedsverk*
  - *medgått tid – i funksjons poeng / måned*
  - *kvalitet – i antall feil og mangler rapportert i en periode etter i driftsetting vurdert mot antall feil og mangler funnet før i driftsetting*

# Grunnlag for å definere beste praksis

**Kvalitativ vurdering av praksis mot standarder f.eks ISO, CMM**
**Kvantitativ vurdering av resultat**




**Gruppert med hensyn på:**
- Type prosjekt
- Prosjektspesifikke faktorer = risiko

**Hva gjør de beste? = Best praksis**
**Bruk de 5% beste**
**Gir minst 15% bedre resultat enn ikke-bruk**
**Aldri farlig**

# Er ISO sertidisering beste praksis

- 30% av firma som har et kvalitetsnivå over gjennomsnittet er ISO sertifisert i tillegg benyttes
  - formelle inspeksjoner
  - formell testing
  - har egne kvalitetsavdelinger
- 50% av firma som er ISO sertifisert har gjennomsnittresultat
- 20% har resultat under gjennomsnittet
- Konklusjon
  - ISO-sertifisering vurderes som nøytral praksis uten signifikant påvirkning av kvalitet eller produktivitet.

# *Er bruk av formell inspeksjon best praksis*

- Alle prosjekter i øvre sjikt på «defect removal efficience» benyttet formell inspeksjon
- Alle dårlige prosjekt benyttet ikke fomell inspeksjon
- Prosjekt som benytter formell inspeksjon har i gjenomsnitt 15% bedre resultat på «defect removal efficience» enn prosjekt som ikke benytter formell inspeksjon.
- Alle prosjekter med mer enn 1000 fuksjonspoeng og som benytter formell inspeksjon har høgre produktivitet enn prosjekter som ikke benytter formell inspeksjon. For prosjekter mellon 100 og 500 funksjonspoeng har formell inspeksjon ingen betydning.
- Konklusjon
  - Bruk av formell inspeksjon er best praksis for alle prosjekter over 1000 funksjonspoeng.

# *En uformell tilnærming til best praksis*

- Caper Jones har en systematisk tilnærming til best praksis, dette er et solid utgangspunkt for å bestemme om noe faktisk er god praksis.

- Skal vi slutte å snakke om best praksis dersom vi ikke har dette store datagrunnlaget?
  - Må se på bedriftens erfaringer
  - De ansatte sine erfaringer

# Kapittel 9
# Systemutviklingens utfordringer

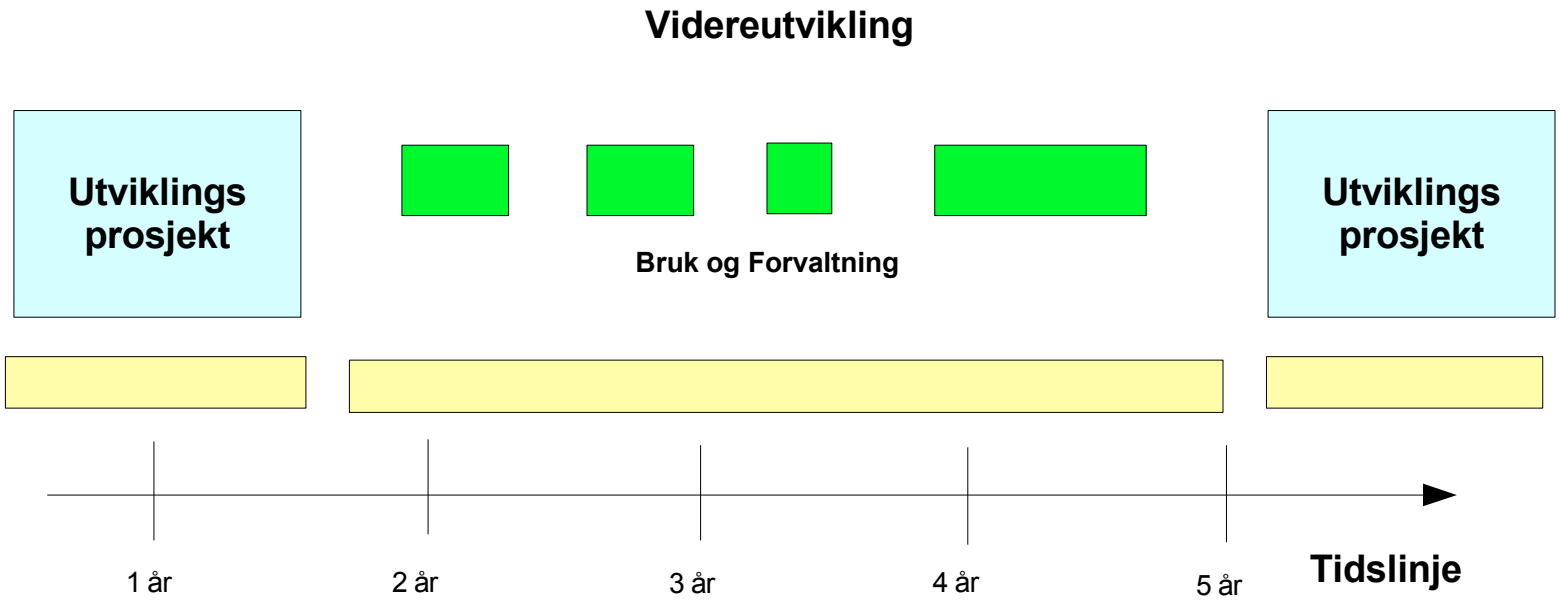
Terje Kårstad

Universitetet
i Stavanger

# Best praksis

*Definisjon*

**Livsløpet til et programvaresystem starter med ideen om systemet og slutter den dagen systemet taes ut av bruk.**



Universitetet i Stavanger

# *Hva er beste system for utvikling og forvalting av programvare*

- *Hva er beste framgangsmåte for å utvikle og å vedlikeholde et høykvalitets programsystem*
- *En bank innfører et nytt kundesystem*
  - *Systemet skaper frustrasjon i månedsvis*
- *En bedrift innfører et nytt lønnssystem*
  - *Grøss og gru*

Universitetet i Stavanger

# Hvor mange lykkes

- Eksperter påstår at mindre enn 50% av programvareprosjektene lykkes ved å:
  - Møte kundens behov
  - Har akseptabel kostnad
  - Ferdig innenfor en gitt tidsramme
  - Er relativt feilfrie

***Edsger Dikstra (Communication of the ACM «The end of computer science»)***

***«The average customer of the computer industry has been served so poorly that he expects his system to crash at all time, and we witness a massive worldwide distribution og bug-ridden software for which we should be depeeply ashamed»***

Universitetet i Stavanger

# Hva har gått galt

- *Dårlig forståelse av krav i prosjektgruppen*
- *Stadig endringer av krav underveis i prosjektet*
- *Systemet utvikles med dårlig vedlikeholdbarhet*
- *Alvorlige mangler i spesifikasjonen oppdages for sent*
- *Systemet har mange feil når det innføres*
- *Systemet utvikles med dårlig ytelse*
- *Systemansvarlig har dårlig kontroll over de ulike versjonene og lite kontroll over leveranser*

Universitetet i Stavanger

# Hva er vanskelig i systemutvikling

- *Systemutviklingsprosessen består av*
  - *Fra behov og forventninger til spesifikasjon (**Planlagt kvalitet**)*
    - *Må samarbeide med kundene*
    - *Finne ut hvilke kvalitet en planlegger for*
  - *Spesifikasjonen er ferdig det skal bygges kvalitet inn i systemet (**Den utviklede kvalitet**)*
  - *Fra ferdig system til et godt arbeidsredskap (**Den opplevde kvalitet**)*
  - *Kunden må oppleve at systemet fungerer*
    - *Systemet oppfyller kravene*

Universitetet i Stavanger

# Tre ulike faser med ulikt kvalitetsmål

- *Planlagt kvalitet*
- *Produsert kvalitet*
- *Den opplevde kvaliteten*

Har vi planlagt og produsert det rette systemet, kan opplæring løse floken. Har vi planlagt det rette systemet, men produsert det med feil og mangler da har vi et problem!
Hva dersom vi har planlagt og produsert feil system, uten en eneste feil?

Universitetet i Stavanger

# Fra behov og forventninger til spesifikasjon

- I denne delen av arbeidet skal vi bestemme:
  - Hvilke egenskaper skal systemet ha
- Det vi prøver å få fram er:
  - Kundens behov
    - Uttalte behov
    - Ikke uttalte behov
- Kundens behov og forventninger er grunnlaget for å sette opp
  - Kravspesifikasjon til systemet
- Kravspesifikasjonen er arbeidstegningen for utviklingen!

***Brooks***

***«No other part of the conceptual work is as difficult as establishing the detailed tecnical requirements... no other part of the work so cripples the resulting system if done wrong»***

Universitetet i Stavanger

# *Viktige spørsmål*

- *Hvem er kunden?*
- *Vet kunden hva han vil ha?*
- *Arbeidet i denne fasen krever arbeidsteknikker og arbeidsprosesser! Det er mulig at et nytt IT system kan snu opp ned på avdelinger og vante arbeidsrutiner.*
- *Arbeidet krever kompetanse og ferdigheter som vi kanskje ikke har lært*
  - *Forstå kunden*
  - *Høre hva han sier*
  - *Forstå organisasjonen systemet skal inn i*

Universitetet i Stavanger

# *Systemutvikling er underlagt lover*

- *Myndighetskarv*
- *Personopplysninger*
- *Arbeidsmiljø*
- *Uskrevne lover*

Universitetet i Stavanger

# Fra spesifikasjon til ferdig system

**I denne fasen skal systemet utvikles, systemet bygges i henhold til kravspesifikasjonene. I prinsippet er denne delen av jobben enklest å håndtere. Det er nå kun å følge arbeidstegningen og så må en ikke gjøre feil.**

- Systemutvikling er vanskelig det stiller store krav til:
  - Analytiske evner
  - Nøyaktighet
  - Dokumentasjon
  - Sporbarhet

Universitetet i Stavanger

# Systemet skal inn i en bedrift

Det nye systemet skal fungere i en bedrift med en gitt bedriftskultur og i samspill med andre systemer. Dette kan gi begrensninger i valg av løsninger. Dessuten må prosjektet gjøre seg kjent med eksisterende infrastruktur og gjeldende krav og standard.

Teknologidelen som benyttes er ofte komplisert og den utvikler seg hele tiden. I alle prosjekter må en regne med å ta i bruk ny og uprøvd teknologi.

Universitetet i Stavanger

# Det er brukeren som avgjør om systemet fungerer

- *Et programsystem må funger for brukeren i det miljøet han/hun arbeider. Opplevd kvalitet krever:*
  - *De riktige egenskapene*
  - *Brukeren får den støtte og opplæring som trengs*
- *Opplæring må planlegges under utviklingsperioden, det er ikke noe som legges på plass når systemet er ferdig. Brukerstøtten kan omfatte:*
  - *Brukerdokumentasjon, inkludert online hjelpetekster*
  - *Opplegg for distribusjon og installasjon*
  - *Opplegg for forvaltning og videreutvikling*
  - *Opplæring*
  - *Helpdesk*
  - *Opplegg for feilrapportering og feilretting*

Universitetet i Stavanger

# Det er brukeren som avgjør om systemet fungerer

- *I tillegg kan vi i denne fasen få ekstra utfordringer*
  - *Viktige sider ved systemet bel ikke godt nok tatt vare på under utviklingen*
  - *Driftsikkerhet og vedlikehold kan ha fått for liten oppmerksomhet*

Universitetet i Stavanger

# Driftsfasen/igangsetting

- *I tillegg kan vi i denne fasen få ekstra utfordringer*
  - *Viktige sider ved systemet ble ikke godt nok tatt vare på under utviklingen*
  - *Driftsikkerhet og vedlikehold kan ha fått for liten oppmerksomhet*

Universitetet i Stavanger

# Hvordan overvinne vanskene

- Systemutvikling blir av mange sett på som sort magi
- Systemutvikling er en engangsoppgave «skreddersøm». Det er en kompleks oppgave, og det finnes ingen naturlover å støtte seg til. Vi må stole på fagfolkenes kompetanse
- Holdningen over er farlig selv om den er delvis sann. Utvikling av programvare er:
  - Menneskeintensiv
  - Kompetanseintensiv
- Mange tror at dersom en har ansatt de beste er problemene løst, men statistikk viser noe annet.

Universitetet i Stavanger

# Silver bullet

- Frustrerte systemutviklere har lett for
  - Kaste seg over teknologiske nyvinninger
    - Høynivåspråk
    - Objekt orientert programmering
    - Kodegenerering – CASE-verktøy
    - Verifikasjon «bevis» for at design og kode er korrekt

Universitetet i Stavanger

# Nye teknikker løser ikke problemet

- Det egentlige problemet er ikke teknologisk. Det egentlige problemet er:
  - Programvarens kompleksitet
  - Kravet om samsvar med omgivelsene, både med bruker og andre systemer.
  - Kravet om endring når omgivelsene endres
  - Programvarens usynlighet, vi jobber med programvare på et abstrakt nivå

> Det finnes ingen vidunderkur dersom den ikke angriper det egentlige problemet. Ny teknologi gir positive bidrag, men den kan ikke eliminere de egentlige problemene. Det å mestre oppgaven å:
>
> - Definere hvilke programvare vi skal bygge.

Universitetet i Stavanger

# Noen kommentarer

***Brooks for 15 år siden***

*Vi må slutte å bygge programvare og istedet la den vokse fram.*
*Kan*

- *Prototyping*
- *Interaktiv utvikling*

*være løsningen?*

Dette var nok et steg i riktig retning, men det var ikke den endelige løsningen

Universitetet i Stavanger

# Noen mener

- Systemutvikling kun kan lykkes dersom
  - Vi klarer å definere den riktige prosessen som gir oss kontroll over alle trinn i utviklingen
- Målet er å optimalisere prosessen. *(Managing the Software process Humphry 1989*)
- Ut fra dette har det vokst fram ulike krav og standarder for systemutvikling. Felles for slike krav og standarder er:
  - Krever orden
  - Krever disiplin
- Dette er noe som mange systemutviklere misliker sterkt. Av og til har dette ført til at for å skrive 100 linjer kode måtte en skrive et 1000 siders dokument som dokumenterte systemet.
- Programvarekrisen er frisk og oppegående

Universitetet i Stavanger

# Smidige teknikker

- Dagens diskusjon:
  - Tungvektsprosesser
  - Lettvektsprosesser
- Tungvektsprosess:
  - Utviklingsprosessen er veldefinert og kontrollert
    - Fossefallsmetoden (Kapittel 10)
- Lettvektsprosess
  - Design og utvikling under veis, interaktiv utvikling
    - eXtreme Programming (kapittel 15)

Her møtes to religiøse leirer og har skikkelige slag.

Merk!
Også lettvektsprosesser krever systematikk og orden, det en gjør er å utvikle skrittvis i små skritt, men det må være system i hvordan skrittene settes.

Universitetet i Stavanger

# Konklusjon

- Viktige sider ved et kvalitetsystem for systemutvikling
  - Vi må ha tro på at systemutvikling kan styres
  - Vi må ha tro på at det finnes en beste praksis
  - Vi må ha tro på at prosessen kan kontrolleres
  - Vi må tro at det er mulig å utføre systematisk forbedring

# Kapittel 10
# En generisk livsløpsmodell

Terje Kårstad

Universitetet
i Stavanger

# Livsløpsmodeller

**I det øyeblikk livsløpsprosessene er beskrevet og definert, har vi fått en livsløpsmodell. Modellen viser hvordan prosessene kan gjennomføres.**

Modellen inneholder:

- Flere aktiviteter
  - Beskriver hvordan aktivitetene kan organiseres
  - Beskrivelse av hvordan aktivitetene kan gjennomføres
  - Beskrivelse av resultater som produseres av de ulike aktivitetene
  - Kan inneholde krav og retningslinjer for de ulike aktivitetene

**Det er vanlig å organisere aktivitetene slik at modellen består av et visst antall faser som skal gjennomføres.**

Universitetet i Stavanger

# Livsløpsmodellen

**L**ivsløpsmodellen er en av hjørnesteinene i et kvalitetssystem for systemutvikling. Den uttrykker:

- **Overordnet utviklings strategi**

Alle andre element i kvalitetssystemet må støtte opp under denne modellen. Valg av modell for lisløpsprosessene er viktig og det er mange å velge mellom.

Universitetet i Stavanger

# ISO 9000-3

## En livsløpsprosess skal være sentrum i kvalitetsystemet

Det er flere modeller. De ulike modellene er et uttrykk for ulikt syn på hva som er den beste strategien for å gjennomføre systemutvikling. Det er mulig grovklassifisere modellene i to klasser

- Fossefalsmodellen
- Ulike evolusjonære modeller

Sentralt i alle livsløpsmodeller står:

- De ulike fasene og faseinndelingen

Universitetet i Stavanger

# Livsløpsmodellen

Fasene definerer en måte å arbeide på. Når en går fra en fase til en annen konsoliderer en stillingen og har som regel nådd en milepæl.

I denne delen av forelesningene ser vi på en generisk livsløpsmodell og hvilke faser den bør inneholde.

Universitetet i Stavanger

# Noen definisjoner

- Inkrement – en liten bit
- Inkrementell utvikling
  - Å arbeide slik at systemet for stadig mer funksjonalitet ved å legge til ett og ett inkrement i gangen
- Iterere
  - Å gjenta samme aktivitet
- Iterasjon
  - En runde med en serie aktiviteter. Resultatet er at et inkrement blir lagt til
- Evolusjon
  - En prosess med gradvis utvikling eller forbedring

Universitetet i Stavanger

# ISO/IEC TR 1571 Information technology

Inkrementell

- Starter med å bestemme kravene og arkitekturen
- Deretter bygges systemet i inkrementer
  - Et inkremet består av et bygg
  - Første inkrement realiserer en del av kravene
  - Neste inkrement realiserer et ny del av kravene
  - O.s.v

Evolusjonær

- Kravene utvikles inkrementelt
- Systemet utvikles inkrementelt

Man finner ikke kravene først, men underveis samtidig som systemet utvikles. Begge måtene er inspirert av Sheward/Deming syklusen for kontinuerlig forbedring.

Universitetet i Stavanger

# Klassifikasjon av modeller

Viktig å merke seg er:

- Ingen modell gjør krav på å være den eneste riktige som kan brukes med samme hell på alle typer utvikling.

*Artefakta:*

*En fellesbetegnelse som brukes om alt som produseres i løpet av et utviklingsprosjekt. (Kildekode, modeller, dokumenter)*

Universitetet i Stavanger

# Klassifikasjon av modeller

**Tungvektsprosess**

- Mange artefakta. Dette er resultat av mye formalisme og byråkrati
- Lite rom for fleksibilitet, mye kontroll
- Arbeidskrevende detaljplanlegging i starten
- Predikativt framfor adaptiv

**Lettvektsprosess**

- Må få opp et kjørbart system så fort som mulig
- Ned toner kravet til formell dokumentasjon
- Det utarbeides ikke detaljplaner for mer enn neste fase
- Kundene deltar aktivt i utviklingsgruppen
- Utviklingsgruppene er små (ikke mer enn 10 personer)

Universitetet i Stavanger

# Klassifikasjon av modeller

**Predikativ prosess**

- Prøver å forutsi og planlegge aktiviteter og ressurser i detalj tidlig i prosjektet

**Adaptiv prosess**

- Aksepterer at endring er normalt. Det er endringene som driver prosessen

**Fortung prosess**

- Det brukes mye tid i tidlig fase fram til kodestart og relativt mindre tid til koding og testing

**Balansert prosess**

- Tidsforbruket er jevnt fordelt

Universitetet i Stavanger

# Eksempel på modeller

Følgende modeller vil bli omtalt i dette kurset

- «code-and-fix» - den aller første modellen
- fossefalsmodellen – alle livsløpsmodellers mor
- evolusjonære modeller – å gro systemet fram
  - RAD – modeller  Rask systemutvikling
  - UP – modeller Objektorientert tilnærming i iterasjoner
  - XP – for det helt ekstreme

Universitetet i Stavanger

# En generisk modell

- mål
- rammer
- kost/nytte
- planer og ressurser

- behov/problemer
- brukerkrav
- valg av løsning
- systemkrav

- arkitektur
- database
- programmering
- testing
- dokumentasjon

- opplæring
- endre rutiner
- endre organisasjon
- installere
- ta i bruk


Skal prosjektet gjennomføres?
Hva slags prosjekt skal gjennomføres?

Vil den spesifiserte løsningen tilfredsstille brukerne?
Er kostnadene akseptable

Er løsningen realisert i henhold til spesifikasjonene

Selv om dette er en generisk modell kan det ha mening å diskutere best praksis for de enkelte fasene. Dette gjøres i resten av dette kapittelet.

Universitetet i Stavanger

# Forstudiet

I denne fasen forankres prosjektet. Hensikten er å skaffe til veie et godt grunnlag for å kunne beslutte om prosjektet skal gjennomføres eller ikke. Resultatet fra denne fasen kan sees på som en kontrakt mellom kunden systemutvikleren.

De viktigste resultatene fra fasen:

- Mål og rammer
- Kost/nytte analyse og risikoanalyse
- Kvalitets plan og gjennomføringsplan
- Prosjektorganisasjon

Universitetet i Stavanger

# Analyse av behov og løsninger

Hensikten med denne fasen er å lage en logisk beskrivelse av den løsningen som skal realiseres i neste fase. Resultatet av dette arbeidet er grunnlag for å kunne bestemme om den spesifiserte løsningen ser ut til å dekke behovet, og om kostnaden ved å realisere løsningen er akseptabel.

Universitetet i Stavanger

# Analyse

Dette gjøres ved å:

- Forstå problemområdet slik at vi ser mulige løsninger
- Få innsikt i og beskrive brukernes krav til en løsning
- Trinnvis spesifisere de funksjonene, de dataelementene og de egenskapene som skal realisere systemløsningen
- Trinnvis spesifisere nødvendige endringer i organisasjonen og i arbeidsrutinene for at systemløsningen skal fungere

Universitetet i Stavanger

# Resultat av analyse

Fasens viktigste resultater:

- Behov og problembeskrivelse
- Kravene til løsning
- Vurdering av alternative løsninger
- Spesifikasjon av den valgte løsning
- Estimere kostnader for realisering
- Tids og ressursplaner for realisering
- Revidert kost/nytte
- Revidert risikoanalyse

Denne fasen deles gjerne inn i flere del faser, og den blir ofte utført i iterasjoner.

Universitetet i Stavanger

# Design, koding og testing

Hensikten med denne fasen er å utforme systemet logisk og fysisk, gjennom en system-arkitektur, og deretter å lage selve systemet.

De viktigste fasene er:

- Systemarkitekturen
- Fysisk datamodell og fysisk database
- Ferdig testede og dokumenterte programmoduler
- Integrert og ferdig testet system
- Brukerdokumentasjon, forvaltnings dokumenter og drifts dokumenter.

Universitetet i Stavanger

# Innføring av systemet

Hensikten med denne fasen er å innføre systemet i brukerorganisasjonen slik at:

- Det er samsvar mellom organisasjonen, arbeidsrutinene og systemet
- Brukerne kan bruke systemet
- Brukerne stoler på systemet
- Det er etablert et støtteapparat som tar seg av brukerproblemer, retter feil og tar seg av daglig drift
- Det er etablert et sikkerhetsnett i overgangsfasen, slik at brukerne kan få utført sine daglige rutiner og oppgaver selv om det oppstår problemer.

Universitetet i Stavanger

# Forvaltningsfasen

Hensikten er at brukerne hver dag skal oppleve at de har et system som oppfyller brukerbehovene og som fungerer uten alvorlige feil og mangler.

For å oppnå dette må vi sørge for:

- Ha kontaktpersoner som brukerne kan spørre hvis de lurer på noe
- Gi hjelp å opplæring til dem som trenger det
- Ta imot feilmeldinger og rette feil og mangler i systemet etter hvert som de oppdages
- Videreutvikle systemet når forutsetningene endres
- Utføre driftsoppgaver som sikkerhetskopiering og lignende på en tilfredsstillende måte.

Denne fasen tar ikke slutt, den varer så lenge systemet er i drift.

Universitetet i Stavanger

# Beste praksis uansett livsløpsmodell

Systemutvikling er en menneskeintensiv og kreativ aktivitet

Utfordringen er:

- Er det mulig å lage en formalisme og og presisjonsnivå som det også er mulig å jobbe kreativt innenfor.
- Når kan en la tankene fly og når må en presisere og dokumentere, verifisere og validere

Presise dokumenter er langt å foretrekke framfor en uformell beskrivelse av systemet.

Universitetet i Stavanger

# Hver sak for seg

Å utvikle et programsystem er en vanskelig oppgave og der en mengde med problemer som må løses. Vi må ha en strategi for hvordan problemene skal løses. Vi må prøve å ta en ting av gangen.

Ulike måter å skille saker på:

- Skille oppgaver i tid, slik at oppgavene blir delt inn i sekvenser med handlinger.
  - Vi starter med det første så det neste o.s.v.
- Vi kan skille ulike kvaliteter fra hverandre
  - Design av database
  - Databaseutforming
  - Optimalisere database med hensyn på effektivitet
- Vi kan se en sak fra ulike synsvinkler
  - Funksjonell synsvinkel
  - Dataorientert synsvinkel
- Dele et problem i mindre biter og behandle hver bit for seg

Universitetet i Stavanger

# Modularitet

Modularitet betyr:

Dele systemet inn i mindre biter – moduler

Prinsippet er brukt av ingeniører til alle tider. De viktigste fordelen med modularitet er:

- Kan konsentrer oss om en modul om gangen
- Systemet konstrueres ved å sveise sammen moduler

Modularitet har mange praktiske fordeler

- Hver enkelt liten bit kan fordeles mellom personer
- Vi kan gjenbruk av biter fra et annet system
- Vi kan forstå systemet ved å forstå biter
- Vi kan gjøre endringer ett sted

«Isolere først, overvinn etterpå»

Universitetet i Stavanger

# Viktig for å oppnå god modularitet

For å ha suksess er det viktig at modulene og grensene mellom modulene er grundig gjennomtenkte.

- Lav kobling mellom modulene
- Ting som er tett koblet høre til samme modul

Universitetet i Stavanger

# Abstraksjon

Dersom vi ikke kan løse det konkrete problemet er det ofte abstraksjon en ypperlig strategi. Det får fram kjernen i problemstillingen, dessuten hender det at løsningen vi finner er generell og kan brukes i andre sammenhenger.

Abstraksjon gjør det mulig å:

- Få fram det vesentlige
- Velge bort detaljer

Universitetet i Stavanger

# Generalitet

Prinsippet går ut på at hver gang vi prøver å løse et problem så forsøker vi å løse et som er mer generelt.

Generelle løsninger er lettere å bruke i andre system. Gjenbruk av kode er også et viktig prinsipp som øker produktivitet og kvalitet.

Universitetet i Stavanger

# Litt om gangen

Ved inkrementell utvikling utvikler en systemet gradvis i små skritt. Vi starter med å utvikle en bit, tester denne biten og fortsetter med neste. Nye inkrement gjør at vi stadig kommer nærmere målet.

Universitetet i Stavanger

# Utvikle for endring

De eneste system som er ferdigutviklet er system som ikke brukes. Programsystem som er i bruk endres kontinuerlig. Det gjør at det er ekstremt viktig at et system er vedlikeholdbart. Alle systemer må utvikles slik at videreutvikling er mulig.

Hvordan kan dette gjøres:

- Ved modularitet
- Ved generalisering
- Ved presist utformet og vedlikehold dokumentasjon
- En gjengarbeid og forståelig systemarkitektur.
- En gjennomarbeid datamodell

Universitetet i Stavanger

# Framover

I de neste kapitlene tar vi for oss de ulike fasene i den generiske livsløpsmodellen. Vi kommer til å se på beste praksis for de ulike fasene. Vi vil forsøke å relatere denne praksis til de standardene vi alt har sett på:

- CMM
- ISO

# Kapittel 11
# Forstudiet

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Hva skjer i forstudiet

Hensikten med forstudiet er å finne ut om det skal gjennomføres et systemutviklingsprosjekt, i så fall – på et helt overordnet nivå – hva slags system skal utvikles.

Forstudiet er:

- Innledningen
- Orienteringsfag

Vi prøver å:

- Skaffe oss oversikt
- Avgrense omfanget
- Finne ut om prosjektet er lønnsomt

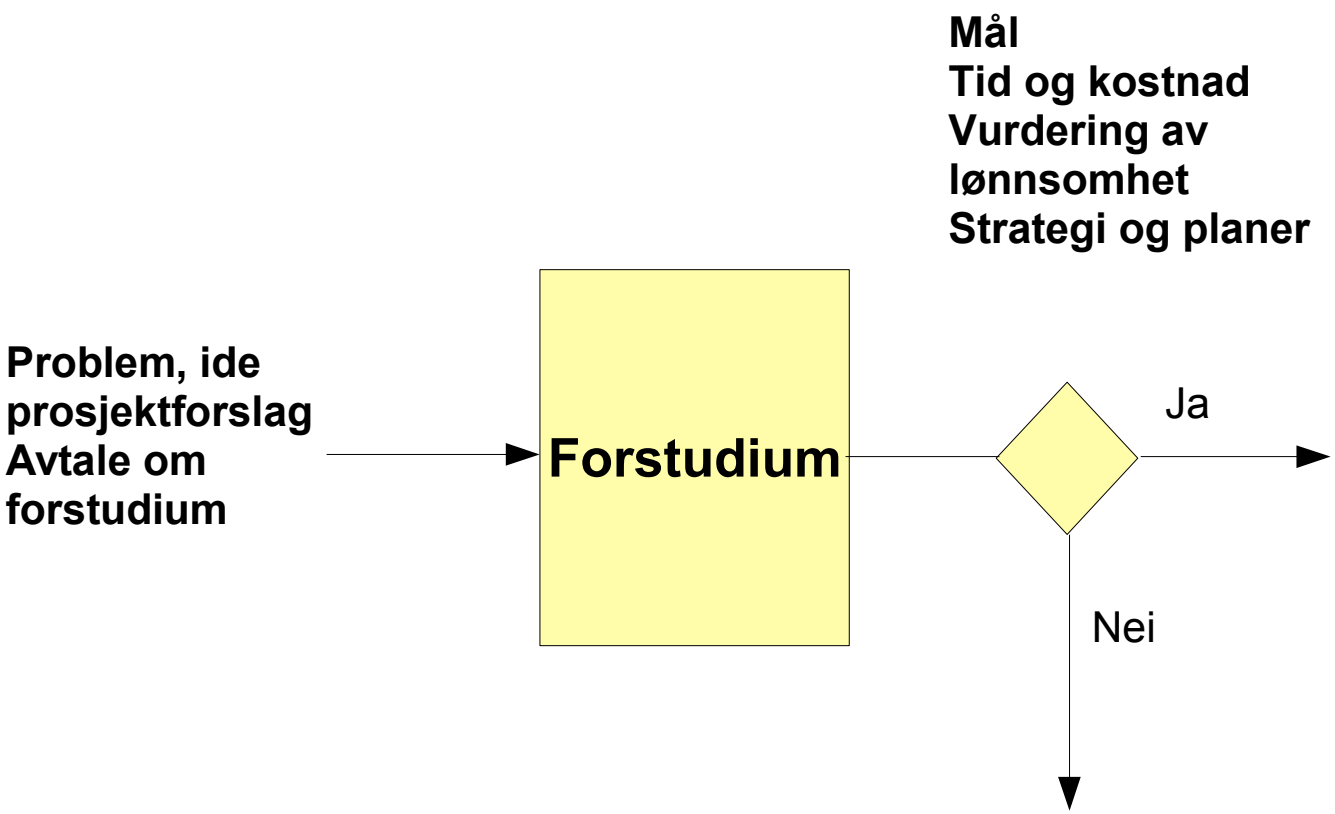
Universitetet i Stavanger

# Resultat av forstudiet

- Prosjektets mål
- Lønnsomheten til prosjektet
- Eventuelle trusler som prosjektet må beskytte seg mot
- Strategi og overordnede planer
- Planer for tid og ressursbruk
- Bemanning og behov for kompetanse

På dette grunnlaget bestemmer en om prosjektet skal gjennomføres eller stoppes.

# Forstudiet


Mål
Tid og kostnad
Vurdering av lønnsomhet
Strategi og planer
Problem, ide prosjektforslag
Avtale om forstudium
Forstudium
Ja
Nei

Universitetet i Stavanger

# Grunnlage for å starte forstudiet

- Et behov
- En oppgavebeskrivelse
- En avtale om gjennomføring

Oppgavebeskrivelser kan være mer eller mindre klart formulert. Noen i kundens organisasjon har identifisert et problem, et behov eller for muligheter som må utnyttes. Dette kan være:

- Et prosjektforslag
- En IT strategien
- En ide eller en tanke

Universitetet i Stavanger

# Forstudiet

Arbeidet i forstudiet går ut på å utdype problemstillingen, gjennomføre intervjuer, dokumentlesing, ide dugnader og lignende.

I tillegg må vi få til:

- En presis avtale mellom kunde systemutviklerne
  - Hva skal gjøres i forstudiet
  - Hva skal det koste
  - Hvor lang tid skal brukes
  - Hvem skal delta

Universitetet i Stavanger

# Hvem gjør hva

I denne fasen må vi samarbeide tett med kunden. Det er kunden som sitter på det meste av informasjonen.

Systemutvikleren sin rolle blir å:

- Stille de riktige spørsmålene
- Analysere
- Systematisere
  - Sørge for at mål rammer og forventet nytteverdi av prosjektet blir kartlagt og dokumenterte
  - Foreslå en overordnet strategi for prosjektgjennomføringen
  - Estimere kostnad, tid og ressursbehov
  - Foreslå prosjektorganisasjon

Universitetet i Stavanger

# Hvem gjør hva

Kunden må:

- Gi ramme for forstudiet (økonomi og tid)
- Fremskaffe informasjon og bakgrunnsmateriale
- Sørge for at de riktige brukerne blir knyttet opp mot prosjektet
- Formulere forventet nytte
- Sørge for at resultatet av forstudiet blir verifisert av personell som har riktig kompetanse

Sammen skal systemutvikler og kunden sikre at prosjektet forankres på riktig måte i organisasjonen.

Universitetet i Stavanger

# Best praksis for forstudiet

De viktigste fasene for å sikre kvaliteten i forstudiet er:

- Forankring av prosjektets
- Organisering av prosjektets
- Bruk av prosjektmodeller
- Håndtering av usikkerhet
- Kvalitets plan
- Risikoanalyse
- Vår evne til å se mer enn en løsning av problemet

Universitetet i Stavanger

# Organisering av prosjektet

Organisering betyr å definere hvem som gjør hva, sørge for at de rette personen er på plass i prosjektet. Det er vanlig å ha:

- En styringskomité
- En kvalitetsrådgivende gruppe
- En prosjektgruppe
- En prosjektleder

Universitetet i Stavanger

# Best praksis for organisering

- En styringskomité som minst består av en representant for kunden og en representant for egen ledelse
- En erfaren prosjektleder for forstudiet
- En eller flere faste kvalitetsrådgivere knyttes til prosjektet. Dette er diskusjonspartnere for prosjektleder.
- Brukerne må ha tilstrekkelig representasjon i prosjektorganisasjonen
- Definer alle roller i prosjektorganisasjonen slik at det går klart fram hvem som gjør hva og hvem som har ansvar for hva.
- De som får en oppgave må få vite
  - Mål
  - Kriterier for resultatene
  - Tidsfrister og timeanslag
  - Kontroller forståelsen
- Gjennomfør lagbygging tidlig i prosjektet
- Ikke aksepter at viktige personer skal arbeide mindre enn 50% i prosjektet.

Universitetet i Stavanger

# Livssyklus modellen er utgangspunktet

I forstudiefasen planlegger vi resten av prosjektet. Bruk alltid livssyklusmodellen som utgangspunkt for planene. Vurder hvilke modell som passer best.

En modell med inkrementell leveransestrategi er i de aller fleste tilfellene langt å foretrekke. Men dette krever nøye planlegging. Hver liten bit/leveranse må planlegges nøye.

Universitetet i Stavanger

# Usikkerhet

Alle systemutviklingsprosjekt er forbundet med usikkerhet og risiko. Tidlig i prosjektet kan usikkerhetene i tid å kostnad være opp mot +/- 80 – 90%

Etter hvert som prosjektet får mer kunnskap reduseres usikkerheten i estimatene. Kostnaden tidlig i prosjektet kan vi se på som en investering for å få vite mer og å oppnå større sikkerhet.

Universitetet i Stavanger

# Kvalitetsplanen

En kvalitetsplan viser hvordan

- Generelle standarder
- Generelle krav
- Generelle retningslinjer

som ligger i kvalitetssystemet skal brukes av prosjektet.

***Råd: Lag en Kvalitets plan for prosjektet.***

Følg de retningslinjene som ligger i kvalitetsystemet. Få en erfaren prosjektleder til å vurdere kvalitetsplanen.

Universitetet i Stavanger

# Risikoanalyse

En risikoanalyse har som mål å avdekke om det er trusler mot prosjektgjennomføringen, og hva som kan gjøres for å beskytte seg mot disse. Risikoanalysen gjøres første gang i forstudiet.

*Gjennomføre en risikoanalyse og sørg for å oppdater den jevnlig gjennom hele prosjektet.*

Universitetet i Stavanger

# Søk etter flere løsninger

I forstudiet må en tenke utradisjonelt. Prøv å få på bordet mange mulige løsninger. Vurder disse løsningene og ranger dem. Velg ut de beste og prøv og gjennomføre risikoanalyse.

# Kapittel 12
# Analyse, design, kode, testing

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Systemutviklingens kjerneområder

Denne delen av kurset er dekket av egne kurs, dette er den delen av kurset der en systemutvikler skal føle seg hjemme.

Det som inngår er:

- Systemering
- Datamodellering
- Databasedesign
- Arkitekturen
- Programutvikling
- Testing
- etc.

Stoffet er dekket i andre kurs.

Universitetet i Stavanger

# Analysefasen

Hensikten med analysen er å bestemme og beskrive i detalj hva slags system som skal lages.

Vi må bestemme

- Hva systemet skal gjøre
- Hvordan systemet skal virke
- Hvordan systemet skal se ut

Beskrivelsen må være på en slik form at

- Brukerne kan forsikre seg om at dette er systemet som oppfyller deres behov
- Designer og programmerer har et entydig arbeidsgrunnlag

Den logiske beskrivelsen kan bestå av

- tekst
- programbiter
- formelle modeller

Analysen er kanskje den mest krevende delen av systemutviklingen.

# Grunnlaget for start av Analysefasen

**Grunnlaget for start av analysefasen er resultater fra forstudiet**

- Resultatmåla for prosjektets
- Problembeskrivelse
- Detaljerte planer for analysen
- Fordeling av roller og oppgaver
- Rammebetingelser som gir absolutte føringer
- Risikoanalyse som viser forhold vi må være spesielt oppmerksom på

Universitetet i Stavanger

# Resultat av Analysen

Resultatene av analysen er grunnlag for flere viktige beslutninger. Kunden må gi svar på følgende

- De dokumenterte kravene er i samsvar med kundens egentlige behov
- De kriterier vi har funnet fram til, er de riktige for å velge løsning
- Den anbefalte løsningen er den som best oppfyller brukerkravene
- De estimerte kostnadene er innenfor rammene av det som kan aksepteres

Analysen foregår trinnvis, kunden må fatte mange beslutninger underveis. Systemutvikler må passe på at kunden får nok beslutningsunderlag.

Universitetet i Stavanger

# Viktige oppgaver i Analysen

Kunden skal:

- Hente fram den informasjonen systemutvikleren trenger
- Sikre at de riktige personene stiller til interju
- Sørge for at resultatene av analysen blir verifisert av personell med rett kompetanse
- Ta beslutninger i rett tid

Systemutvikleren skal:

- Sørge for at brukerkrav blir kartlagt og dokumenterte
- Utrede alternative løsninger
- Få fram og dokumentere kriterier for valg av løsningene
- Gjennomføre evaluering av de ulike alternativene
- Estimere kostnad, tid og ressursbruk for gjengføringen av resten av prosjektet

Universitetet i Stavanger

# Best praksis for analysen

Hvordan sikre at vi spesifiserer det rette systemet? Vi har følgende utfordringer

- Å avdekke hva brukerne vil ha
- Å forme og spesifisere et system som oppfyller disse kravene

Det finnes ingen enkle løsninger som garanterer et vellykket resultat, men her er noen enkle råd:

Universitetet i Stavanger

# Arbeidsmetoder analysen

Benytt

- I møte med kundene bruk konkrete og visuelle teknikker
  - prototyping
- Bruk ISO 9126 som utgangspunkt for å bestemme systemets egenskaper
- Trinnvis kvalitetsplanlegging for å dokumentere krav
- Bruk anerkjente framgangsmåter for utvikle spesifikasjonene
  - ER modeller
  - DFD diagrammer

Universitetet i Stavanger

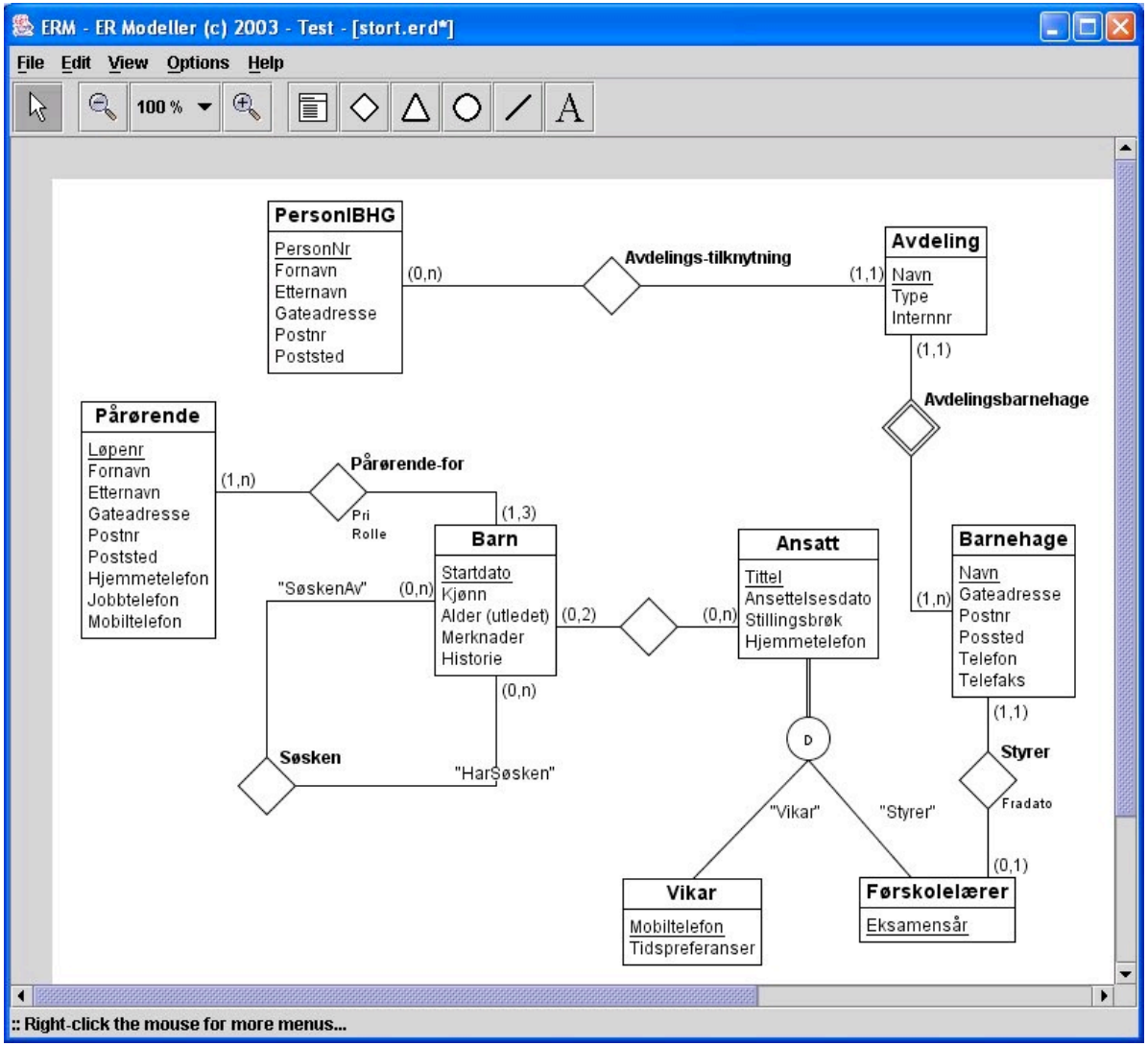
# Kort og konsis framstilling

Et stabilt og komplett sett av brukerkrav er et godt mål for arbeide i analysefasen. Men det er sjelden en god ide å satse på at vi skal ha dette fullstendig ferdig før vi i det hele tatt har utviklet noe. En bedre strategi er:

- Sikt mot å beskrive komplett og presist en avgrenset sentral delmengde av brukerkravene
- Realister denne biten
- Erfaringer fra denne biten gjør at en går videre og spesifisere en ny bit, så utvikling

Universitetet i Stavanger

# ER-modeller



Universitetet i Stavanger

# Arbeidsmetoder analysen

Benytt

- I møte med kundene bruk konkrete og visuelle teknikker
  - prototyping
- Bruk ISO 9126 som utgangspunkt for å bestemme systemets egenskaper
- Trinnvis kvalitetsplanlegging for å dokumentere krav
- Bruk anerkjente framgangsmåter for utvikle spesifikasjonene
  - ER modeller
  - DFD diagrammer

Universitetet i Stavanger

# Bruk modeller for dokumentasjon av kravene

Modellene skal være egnet til kommunikasjon med brukerne, men de må kunne vise viktige sider ved systemet.
Brukerkravene må omfatte

- Krav til funksjoner
- Krav til data
- Krav til egenskaper

Systemspesifikasjonen må omfatte

- Automatiserte funksjoner, funksjonslogikk, inndata og utdata
- Manuelle rutiner
- Skjermbilder
- Den logiske datamodellen og alle dataelementene
- Systemegenskaper
- Et opplegg for brukerstøtte og forvaltning

Universitetet i Stavanger

# Bruk modeller for dokumentasjon av kravene

Legg vekt på sammenhengen mellom brukerkrav og systemkrav og sporbarhet. Spesifikasjonene må ikke inneholde detaljer om design eller implementasjon. De skal si hva som skal lages ikke hvordan

Spesifikasjonen dokumenteres i et **formelt dokument** som er underlagt versjonkontroll og **formell godkjenning**.

Universitetet i Stavanger

# Still strenge krav til resultater fra analysen

Analysen skal være så god at

- Vi kan overlate til andre å utvikle systemet
- Det er en entydig kontrakt mellom oppdragsgiver og systemutvikler

Universitetet i Stavanger

# IEEE Std 830-1998

1. **Utvetydig:** Hvert utsagn i spesifikasjonen skal ha kun en tolkning
2. **Komplett:** Alle signifikante krav må være med. Dette gjelder
    1. data
    2. egenskaper ved systemet
3. **Verifiserbare:** Det må være mulig i ettertid å kontrollere at det ferdige systemet oppfyller kravene i spesifikasjonen
4. **Konsistente:** Ingen deler av spesifikasjonen må være i konflikt med andre deler.
5. **Modifiserbare:** Strukturen og stilen må gjøre det lett å endre deler av spesifikasjonen
6. **Sporbarhet:** Vi må sørge for at vi kan spore et brukerkrav fra den brukeren eller gruppen som kom med kravet, via spesifikasjonene til de delene i det ferdige systemet hvor kravet er oppfylt. En systemegenskap må kunne spores tilbake til kravene
7. **Brukbare for drift og forvaltning:** Spesifikasjonen inneholder viktig informasjon om systemet. Denne informasjonen trengs også til drift og forvaltning av systemet.

Universitetet i Stavanger

# Design, koding og testing

Design koding og testing er de tekniske aktivitetene i utviklingsprosessen. Disse aktivitetene utføres alltid, uavhengig av utviklingsmodellen som følges. I den generiske modellen starter fasen når kravspesifikasjonen er ferdig.

I en virkelig prosess vil de naturlig gjentas en rekke ganger. Under koding kan en oppdage at design burde vært annerledes. Da må en gjøre endringer i designet. Under testing kan det dukke opp feil som fører til endring både i design og koding.

Fasen er ferdig når vi har et testet, kjørbart programvaresystem klart for innføring.

Universitetet i Stavanger

# Hva er design

Design kan bety flere ting alt etter sammenhengen det brukes i.

- Et mønster eller en tegning
- Formgiving
- I systemutvikling
  - En prosess og resultat av den prosessen
    - Resultatet er designet av systemet

Universitetet i Stavanger

# Design

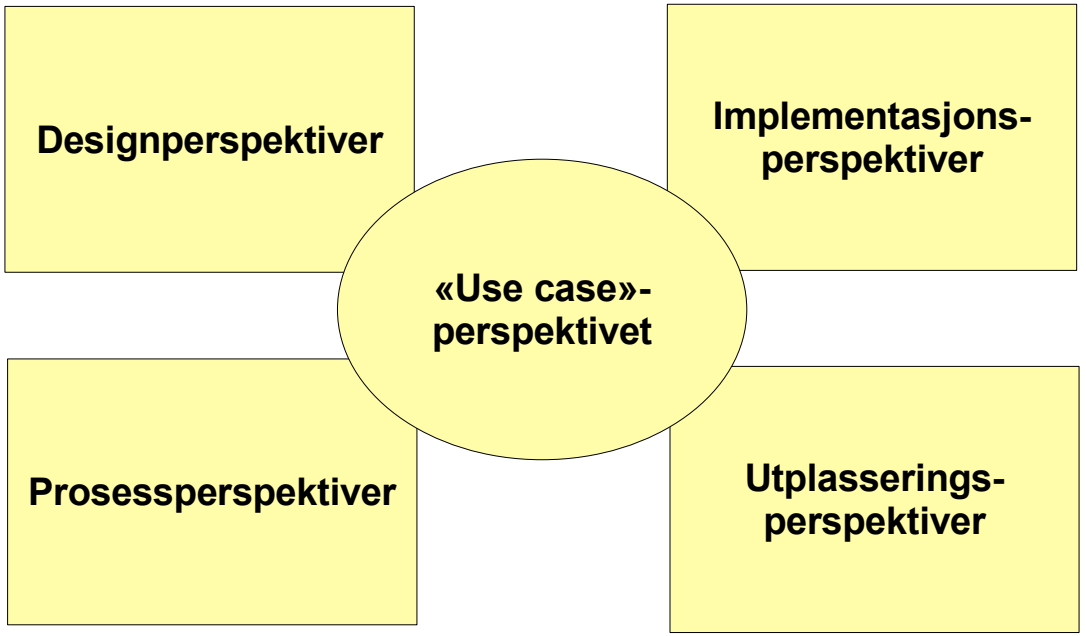
Designprosessen krever, erfaring, kreativitet og systematisk arbeid den skal føre fram til

- En detaljert beskrivelse av hvordan systemet er delt opp i logiske delsystemer, moduler og enheter
- En detaljert beskrivelse av hvordan de logiske delene er fordelt på prosesser og fysiske prosesseringenheter
- En detaljert beskrivelse av topologien til systemet
- En detaljert beskrivelse av grensesnittet mellom enheter, moduler og delsystemer
- En detaljert beskrivelse av datastruktur og hvordan de skal representeres
- En detaljert beskrivelse av algoritmene for funksjonene i systemet

**Resultatet skal være alt som er nødvendig for at en skal kunne skrive kode.**

Universitetet i Stavanger

# Systemet har en struktur




Strukturen og sammenhengen kan betraktes fra ulike perspektiver. Hvert perspektiv viser en side ved systemet. Et gitt perspektiv kan vise en logisk side ved systemet, et annet den fysiske siden til systemet.

Universitetet i Stavanger

# Perspektiver

- Designperspektiv
  - Her vises hvordan systemet er delt opp i programvareenheter, og hvordan disse samspiller.
- Prosessperspektivet
  - Her vises systemets kjørbare enheter. Det dreier seg om tråder og prosesser.
- Implementasjonsperspektiver
  - Dette viser hvordan programmet er delt opp i fysiske filer
- «Use-case» perspektiver
  - I dette perspektivet ser en systemet ut fra brukeren sin synsvinkel. Dette viser hvorfor systemet er blitt som det er blitt.

Universitetet i Stavanger

# Designprosessen

Designprosessen skal lede fram til

- Arbeidstegningene

Resultatet er modeller og dokumenter so skal evalueres. Dette gjøres for å finne ut om systemet som er under utvikling, vil kunne tilfredsstille alle brukerkravene.

Designprosessen er delt i to faser. I NS-ISO/IEC 12207 Informasjonsteknologi. Programvarens livsyklusprosesser

- Toppnivå/grovdesign
- Detaljert design

Universitetet i Stavanger

# Programvare-arkitektur

Shaw and Garland:

> Programvarearkitektur dreier seg om beskrivelser av elementene som systemet bygges opp av, samspill mellom disse elementene, mønstre som gir retningslinjer for hvordan komponentene skal settes sammen og rammer for disse mønstrene.

Universitetet i Stavanger

# Programvare-arkitektur

Poenget er:

- Arkitekturen definerer komponenter. Det omfatter informasjon om hvordan komponentene skal samspille.
- Et system kan ha mer enn en struktur. Strukturen avhenger av perspektivet vi ser systemet ut fra.
- Alle systemer har en arkitektur. Et system kan alltid deles opp i komponenter og relasjoner mellom komponenter.
- Atferden til hver komponent er en del av arkitekturen. Diagram med bokser og linjer, er ikke systemets arkitektur. Det kan vise sider ved arkitekturen.

Arkitekturen til et system kan være god eller dårlig. God arkitektur fører til:

- Systemet tilfredsstiller alle krav
- Systemet kan utvides

Universitetet i Stavanger

# Arkitektoniske stiler

Systemutviklere har en verktøykiste som består av et utvalg programvarestrukturer som har vist seg å fungere i praksis. Det er nå gjort forsøk på å dokumentere og systematisere slike arkitektoniske stiler.

- Klient-tjener
- Objektorientering
- Lagdeling

Det som karakteriserer en arkitektonisk stil, kommer frem gjennom svar på disse spørsmålene (Shaw & Garland 1996)

- Hva er typen av komponenter og forbindelsen mellom komponentene?
- Hva er de tillatte strukturelle mønster
- Hvilke beregningsmodeller ligger i bunnen
- Hva er det uforanderlige i stilen
- Finnes det noen kjente eksempler på bruken av stilen
- Hvilke vanlige spesialiseringen finnes.

Universitetet i Stavanger

# Arkitektoniske stiler

**Rør og filter**

Hver komponent har et sett med inndata og et sett med utdata. En komponent leser en strøm med inndata på inngangen og produserer en strøm med data til utgangen

**Objektorientering**

Hovedpoenget er data-abstraksjon og inn kapsling av data og implementasjonsdetaljer.

**Implisitte kall**

De involverte komponentene er frikoblet. En komponent kringkaster/annonserer en hendelse Alle komponenter i systemet kan registre seg som interesserte til å håndtere denne hendelsen. Det gjør de ved å registre en prosedyre for hendelsen Systemet kan da kalle den registrerte prosedyren.

Universitetet i Stavanger

# Arkitektoniske stiler

**Lagdelte systemering**

Systemet er organisert hierarkisk. Hvert lag sørger for tjenester til laget over og er klient til laget under. Kommunikasjonsprotokoller bruker denne stilen.

**Repositorier**

Her er to helt forskjellige komponenter.

- En sentral datastruktur
- En samling uavhengige komponenter som opererer på denne strukturen

**Andre stiler**

**Prosesskontroll**

Brukes i sanntidssystemer. I et slikt system, er en tjener en prosess som yter tjeneste til en annen prosess.

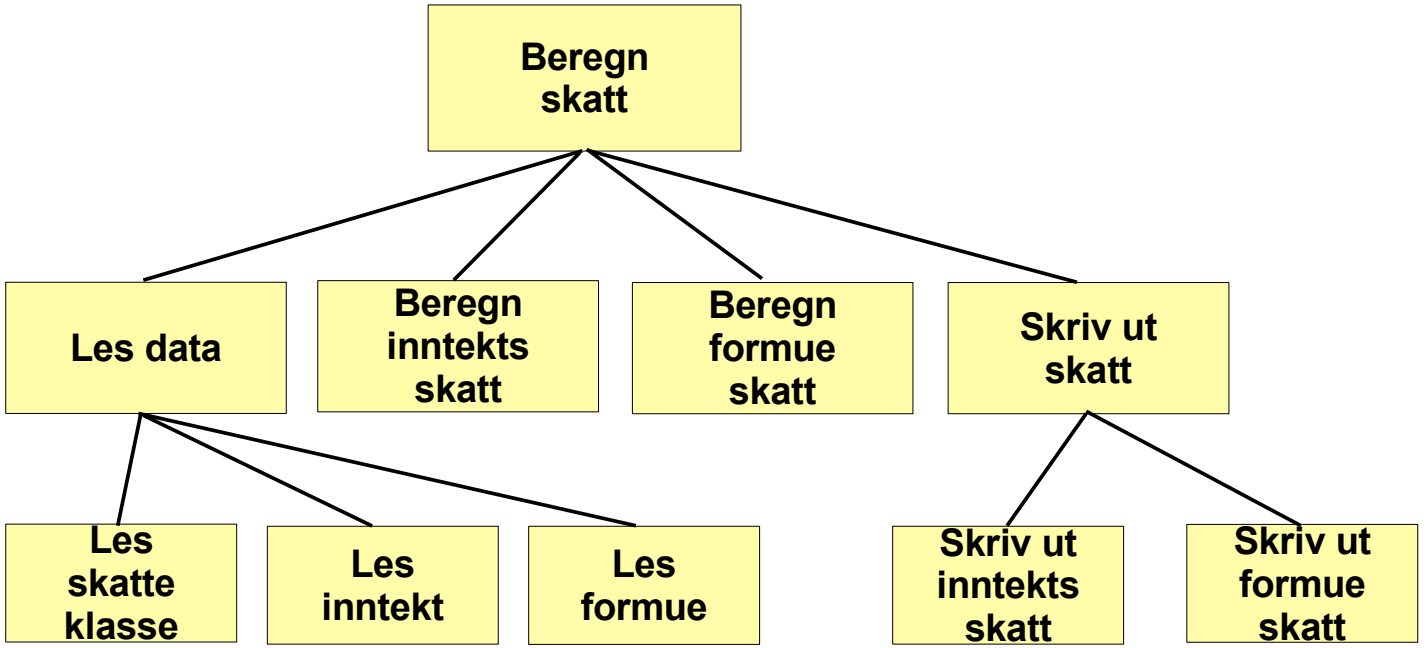
**Heterogen arkitekturen**

I større system er det en blanding av alle mulige arkitekturer.

Universitetet i Stavanger

# Designmetoder

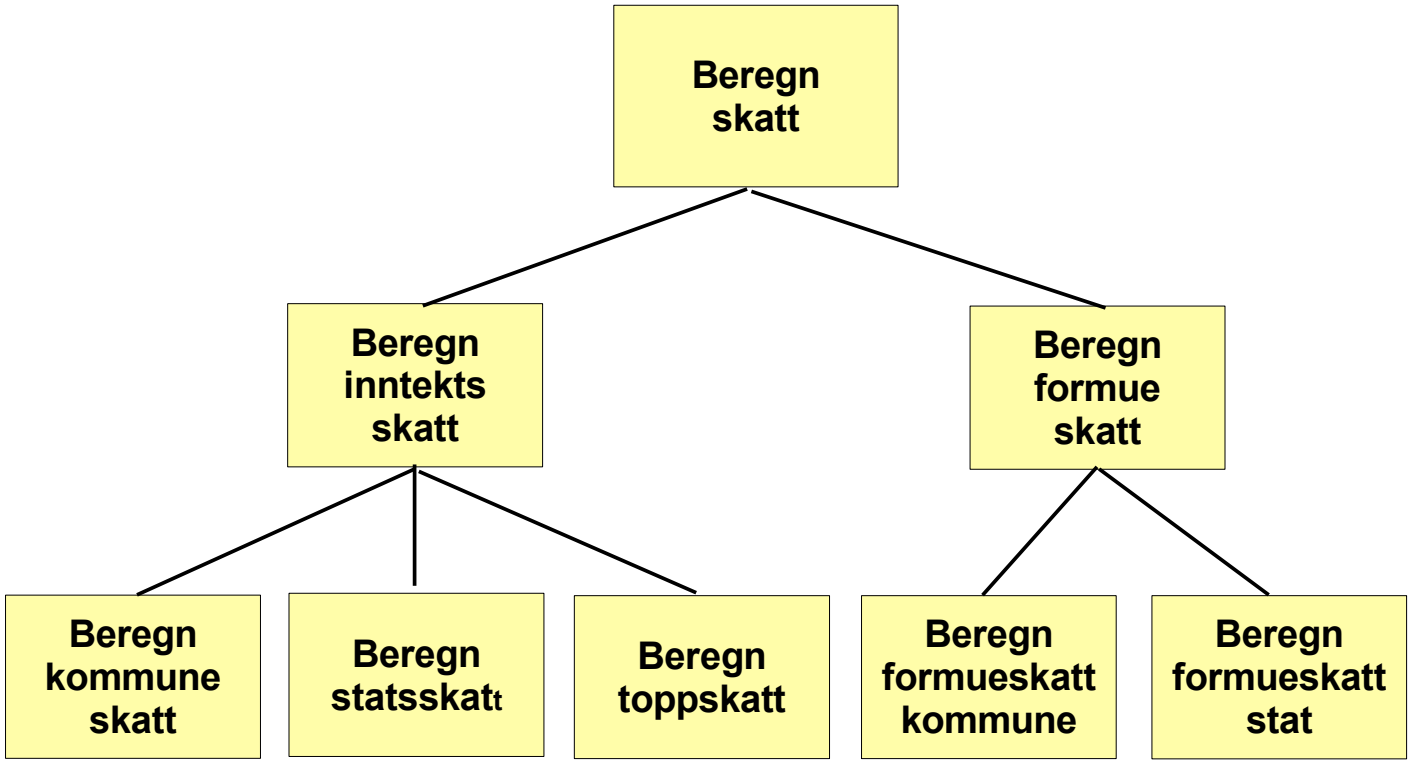
Dataflyt design



Universitetet i Stavanger

# Designmetoder

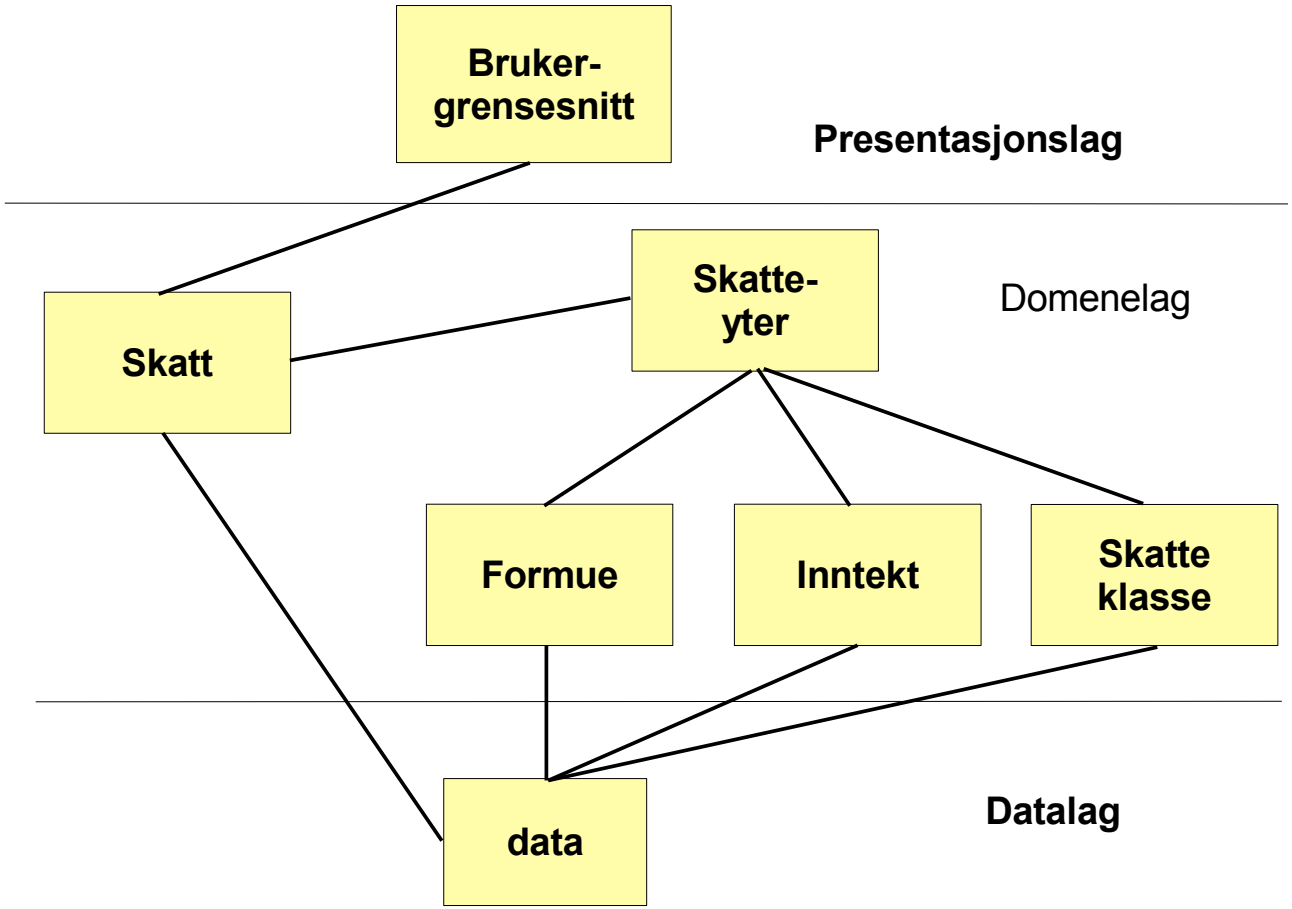
Datastruktur-design



Universitetet i Stavanger

# Designmetoder

Objektorientert design



Universitetet i Stavanger

# Best praksis for design

Modularitet

- Modulene i systemet har smale grensesnitt mot resten av systemet
- Strukturert design bruker begrepet høy/lav khoesjon
  - En modul har høy khoesjon dersom en modul den utfører et lite avgrenset og beslektede sett av oppgaver
- Modulene er løst koblet dersom modulen utveksler informasjon med ideelt sett bare en annen modul.
- Løst koblede og khoesive moduler har disse fordelene
  - Lett å forstå
  - Lett å vedlikeholde
  - Lett å utvide

Etter første design går vi kritisk gjennom resultatet. Vi undersøker om elementene er khoesive og løst koblede. Vi kan beregne metrikker for å få et mål på hvor god den foreslåtte design virkelig er. Er vi ikke fornøyde må vi gjøre endringer.

Universitetet i Stavanger

# Koding

Koding er den andre av de tekniske bitene i utviklingsfasen. Nå skal systemet realiseres. Dette er høydepunktet for de fleste utviklere, men resultatet blir sjelden bra uten grundig forarbeid.

Selv om en kan ha god hjelp av kodegeneratorer, må en skrive kode selv. Hvilke språk som skal brukes er ofte gjenstand for diskusjon. Velg det som passer best til prosjektet og som utviklerne kan. (det finnes mer enn 2000 programmeringsspråk)

De viktigste kriteriene for valg av språk er:

- Hvilke applikasjon er det snakk om
- Hvilke språk har utviklerne kjennskap til
- Er det behov for samspill med tidligere utviklede systemer
- Kan språket bidra til å øke produktiviteten
- Er det lett å kommunisere ideene i design over til utviklerne

Universitetet i Stavanger

# I NS-ISO/IEC 12207

- Utvikleren skal utvikle og dokumenteres
  - hver programvareenhet og database
  - testprosedyrer og data for å teste hver programvareenhet og database
- Utvikleren skal teste hver programvareenhet og database for å forvisse seg om at de tilfredsstiller kravene. Testresultatene skal dokumenteres.
- Om nødvendig skal utvikleren oppdatere brukerdokumentasjonen.
- Utvikleren skal oppdatere kravene til og tidspunktene for integrasjonstesting

Universitetet i Stavanger

# I NS-ISO/IEC 12207

- Utvikleren skal evaluere koden og testresultatene i forhold til kravene nedenfor
  - Sporbarhet til kravene og design av programvareenheten
  - Eksternt samsvar med kravene og design av programsystemet
  - Internt samsvar mellom krav til enheter
  - Testdekning av hver enhet
  - Hensiktsmessighet av de kodemetoder og standarder som er brukt
  - Egnethet for programvareintegrasjon og testing
  - Egnethet for drift og vedlikehold

Resultatene av evalueringen skal dokumenteres.

Universitetet i Stavanger

# Hvilke tester skal gjennomføres

Jo tidligere en feil blir oppdaget jo letter er det å rette feilen.
Testingen starter med

- Testing de enkelte enheter

Enhetene kobles sammen (integrasjontesting)

- De sammen-koblede enhetene testes

Systemet er ferdig

- Nå utføres ulike typer systemtester
  - Er kravene til systemet oppfylt

Ulike typer testing taes opp i kapittel 19

# Kapittel 13
# Innføring

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Innførings-fasen

Hvem er det som har ansvaret for innføringen av systemet. Er det

- De som utvikler systemet?
- Er det kunden?

Ønsker vi at systemet skal ha kvalitet, må vi ha en gjennomtenkt plan for hvordan systemet skal innføres i kundens arbeidsmiljø. Alle har vi hørt.

- Vi har fått nytt datasystem og det fungerer ikke

Alle som får er nytt system å forholde seg til er ypperlige testere de vet ikke hvordan systemet skal virke, det gjør at de bruker det på alle mulige utradisjonelle måter.

- Takler systemet en slik bruk?

Universitetet i Stavanger

# Målet med denne fasen

Målet med med alle nye system er å gjøre systemet **usynlig.** Et godt system skal være:

- Et nyttig verktøy
- Velfungerende
- Et anonymt hjelpemiddel
- Systemet skal være der når jeg trenger det

Dersom systemet får all oppmerksomheten så er det noe galt med systemet. Målet med rutinene vi lager for innføringen av systemet er at vi aldri får høre setningen.

«Det er litt rot og kaos i dag, men vi har fått nytt datasystem.»

Universitetet i Stavanger

# Hensikten

Hensikten med systeminnføringen er:

- Brukerne kan systemet
- Brukerne stoler på systemet
- Brukerne har nytte av systemet

# Når starter innføringen

Innføringen starter når prosjektet starter. Innføringsaktiviteter gjennomføres gjennom hele prosjektet. Vi sørger for

- Informasjon til brukerorganisasjonene
- Vi passer på at arbeidsrutiner endres
- Vi sørger for at utstyr kommer på plass

Når vi går inn i siste fase av prosjektet, har innføringen foregått lenge. Innføringsfasen blir det endelige punktum i et innføringsløp som har gått parallelt med alle de andre aktivitetene i prosjektet. Nå er det bare å sørge for at alt det nye begynner å virke sammen og kan erstatte de/det gamle systemet.

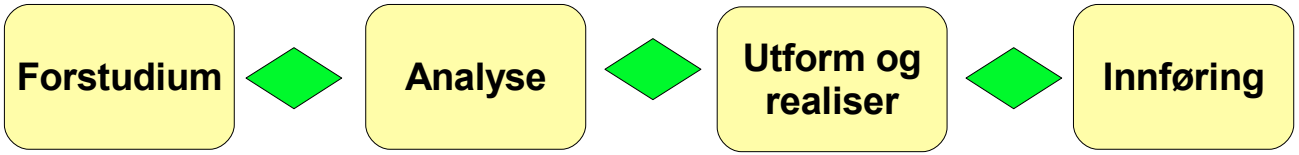
Universitetet i Stavanger

# Hvem har ansvaret

Systemutviklingsprosjekter har som regel flere representanter fra kundens organisasjon som aktive prosjektdeltakere, disse kan fungere som bindeledd inn i kundens organisasjon. Det er nødvendig at prosjektet:

- I planene tar med nødvendige aktiviteter i kundens organisasjon og tidfester disse.
- Sjekker at disse aktivitetene gjennomføres som planlagt og at resultatene holder mål.
- Om nødvendig støtter gjennomføringen

# Innføringsaktiviteter




1 Innformasjonsvirksomhet

2 Planlegging av innføring

3 Brukeropplæring

4 Endring av arbeidsrutiner og organisasjon

5 Overføring til driftsmiljø

6 Utvikling av permanent dokumentasjon

7 Etablering av støtteapparat

8 Akseptsanse

Universitetet i Stavanger

# Aktiviteter

- Informasjonsvirksomhet
  - Den informasjonen som prosjektet gir til kundes organisasjon
- Planlegging av innførings-fasen
  - Dette er prosessen fra strategi for innføring til en komplett og detaljert plan for innførings-fasen
- Brukeropplæring
  - Kapittel 13.2.2
- Endring av arbeidsrutiner og organisasjons
  - Dette er nødvendige endringer for å få systemet til å passe (Kapittel 13.2.4)
- Overføring til driftsmiljø
  - Dette er måten vi jobber på for å sørge for at systemet ikke ødelegger driftsmiljøet (Kapittel 13.2.6)
- Utvikling av permanent dokumentasjon
  - Samtidig som vi utvikler systemet må vi sørge for å utvikle en god og tjenlig dokumentasjon. Den skal støtte både brukerne og dem som retter feil og holder systemet i drift. (Kapittel 14.2.4)
- Etablering av støtteapparat
  - Støtteapparatet må være på plass den dagen brukerne starter å bruke det nye systemer. (Kapittel 13.3.3)

Universitetet i Stavanger

# Planlegging av innføring

Innføringsfasen er hektisk. Mye skal gjøres og mye skal på plass. Dette må skje fort, det har innvirkning på arbeidssituasjonen til mange mennesker.

- Sørg for at overgangen foregår på en kontrollert og strukturert måte
- Sørg for at systemet introduseres i organisasjonen til rett tid

For å få dette til må en planlegge. Utearbeid en innførinsstrategi som godkjennes av brukerne. Strategien skal beskrive:

- Rekkefølgen på delsystemene som innføres
- Rekkefølgen på innføringen i ulike deler av kundens organisasjon
- Nødvendig samordning med andre hendelser
- Mulighet for å ha systemet på prøve
- Opplæringsbehov for brukergrupper og strategi for opplæring
- Er det behov for spesiell assistanse
- Opplegg for parallellkjøring og andre overgangsordninger

Universitetet i Stavanger

# Plan innføring

- Plan for opplæring og informasjon
  - Ansvarsforhold
  - Oversikt over hvilke opplæringsmateriell og brukerdokumentasjon som skal lages
  - Oversikt over hvem som skal ha hvilke opplæring
  - Spesifikasjon av alle nødvendige kurs eller informasjonsopplegg, med kostnader, tidsplaner og avhengighet til andre hendelser
- Plan for datakonvertering og initialisering med alle nødvendige aktiviteter, hvem har ansvar, tidsplaner

Innførings-planene må revideres flere ganger, helt til alle detaljer er på plass. Alle planene må diskuteres med kunden.

Universitetet i Stavanger

# Brukerne må få opplæring

For at innføring av et nytt system skal bli vellykket må brukerne:

- Forstå systemet
- Beherske systemet

Brukeropplæring kan være vanskelig for systemutviklere, de tenker ofte annerledes enn vanlige brukere.

Brukeropplæring må planlegges. Viktige ingredienser er:

- Analyser de ulike brukergruppers opplæringsbehov
  - Superbruker treng en type opplæring
  - Vanlige brukere treng en annen type opplæring
  - Langvarig kursing av mange brukere er urealistisk
- Opplæringsmateriell må være tilpasset brukerne ikke systemet og systemutviklerne
- Planlegg med at opplæringen skjer i flere trinn
- Brukeropplæringen kan starte tidlig, men noe av systemet må nok være på plass i arbeidsmiljøet

Universitetet i Stavanger

# Støtteapparatet må på plass

Før systemet settes i drift må det være etablert et

- Støtteapparat som har ansvar for å hjelpe brukerne og for å holde systemet i drift.

Disse personene må utpekes så tidlig som mulig, senest før avslutningen av analysen. Disse personene må få nødvendig opplæring og den må være ferdig før jobben skal utføres. Det ideelle er at de er med i prosjektgruppen og deltar i utviklingen av systemet.

Det utarbeides en avtale for brukerstøtte, drift og forvaltning, og at de nødvendige rutinene beskrives. Avtalen må være godkjent, og rutinene må være aksepterte av brukerne.

Universitetet i Stavanger

# Støtteapparatet må på plass

Har organisasjonen en helpdesk, pass på at de som sitter der vet om systemet, og at de har kompetanse til å svare på spørsmål.

Sørg for tilstrekkelige reserveløsninger, parallellkjøring mellom gammelt og nytt system. Dette kan være nødvendig dersom en alvorlig feil oppstår.

Universitetet i Stavanger

# Forbered organisasjonen

Det nye systemet er ikke et isolert system. Det må:

- Integreres med arbeidsrutinene i organisasjonen

Dette kan bety endring i rutiner og slikt må integreres i prosjektet. Vanlig framgangsmåte er:

- I forstudiet finner vi omfanget av de endringene som må gjøres
- I analysen arbeider vi først med systemet. Deretter utvikler vi modeller for hvilke deler at systemet som skal totalautomatiseres og hvilke deler som skal være manuelle. Vi definerer nødvendige endringer av roller og arbeidsoppgaver. Videre utvikler vi detaljspesifikasjon. Disse må omfatte
  - Arbeidsrutiner
  - Roller
- Realiseringen sørger for å gjennomføre endringene. Vi utpeker personer til de ulike rollene og gjør dem kjent med ansvar og arbeidsoppgaver

Universitetet i Stavanger

# Forbered organisasjonen

Før systemet innføre små vi forsikre oss om:

- Roller og ansvar er kjent i organisasjonen av alle som har med innføringen av systemet og gjøre
- Arbeidsoppgavene er kjent i organisasjonen av alle som har med innføringen av systemet å gjøre

Til slutt må vi forsikre oss om:

- Alt nytt utstyr og programvare er anskaffet og på plass
- Kostnadsbildet for det nye systemet er kjent

Universitetet i Stavanger

# Informasjon

Informasjon fra prosjektet er en av de viktigste brikkene i innføringen av systemet. Legg vekt på å gi:

- Regelmessig informasjon
- Målrettet informasjon

Planlegg som et minimum å gi:

- Generell informasjon til hele brukerorganisasjonen
- Spesiell informasjon til alle som er berørt av prosjektet
- Informasjon om prosjektet i all dokumentasjon som leveres til kunden
- Regelmessig informasjon til brukerne om status
- Spesifikk informasjon før systemet innføres om hva som skal skje

Universitetet i Stavanger

# Overføring til driftsmiljø

Et nytt system må ikke ødelegge for de systemene som allerede er i drift. Det vil si at vi må ha et utviklingsmiljø som er isolert fra driftsmiljøet. Vi må ha minst to adskilte fysiske miljø, helst tre.

- Et utviklings og testemiljø
- Et akseptansemiljø
- Et driftsmiljø

Universitetet i Stavanger

# Overføring til driftsmiljø

Med tre miljøer er dette vanlig arbeidsprosedyre

1. I god tid før utviklingen starter, etablerer vi et utviklingsmiljø. Dette miljøet inneholder alle nødvendige programmer og alt nødvendig verktøy. I dette miljøet utvikles og testes systemet i hele utviklingsperioden. Her integreres alle modulene og systemet ferdigstilles.
2. Mot slutten av analysen opprettes et akseptansemiljø. Akseptansemiljøet skal være mest mulig likt driftsmiljøet, men det er skjermet fra driftsmiljøet. I dette miljøet foregår følgende tester:
   - Systemtest. Den samlede testen av hele systemet (FAT factory acceptance-test)
   - Stresstest. Tester ytelsene under ekstreme forhold
   - Test av programvare mot de kriterier som gjelder for at det skal kunne overføres til driftsmiljøet.

Universitetet i Stavanger

# Overføring til driftsmiljø

3. Et driftsmiljø etableres i god tid før systemet settes i drift. Dette er det miljøet som tar seg av den daglige driften. Miljøet må ha
   - Nødvendig basisprogramvare
   - Nødvendig infrastruktur
   - Nødvendig kapasitet
   - Detaljerte driftsprosedyrer

System som har bestått akseptansetesten overføre til driftsmiljøet

# Kapittel 14
# Forvaltning

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Forvaltning

- Dette er tiden fra systemet settes i drift til det tas ut av drift.
- Dette er en periode som kan strekke seg over flere år.
- Normalt varer denne perioden i 3 – 5 år.

Universitetet i Stavanger

# Begreper

Her er noen viktige begreper

- Brukerstøtte er det vi gjør for å støtte brukeren i den daglige bruken av systemet. Det dekker
  - Sikrer at systemets funksjoner dekker brukernes behov
  - Definerer kvalitetskrav til systemet
  - Sørger for skikkelig opplæring
  - Svarer på henvendelse om bruk av systemet
  - Er et bindeledd mellom brukerne og de som drifter systemet
  - Deltar i planleggingen av videreutviklingen av systemet

Universitetet i Stavanger

# Begreper

- Forvaltningen er det vi gjør for å holde systemet i samsvar med spesifikasjonene. Det dekker
  - Å rette feil i systemet etter hvert som de oppdages
  - Å gjøre endringer i systemet for å tilpasse det nye driftsmiljø
- Drift er det vi gjør for å holde systemet i drift. Det omfatter
  - Installasjon av nye versjoner
  - Overvåking av ytelse
- Videreutvikling er det vi gjør for å endre systemets funksjonalitet på grunn av endrede krav fra brukerne

Universitetet i Stavanger

# Begreper

Det er vanlig å holde videreutvikling av systemet utenfor det vi her kaller forvaltningen. Videreutvikling gjennomføres i prinsippet på samme måte som utvikling. Arbeidet organiseres i et prosjekt, og det gjennomføres etter en livsløpsmodell.

Videreutvikling og forvaltning går inn i hverandre

- Endringsforslag og feilmeldinger behandles på samme måte
- Når gjennomførere videreutvikling retter vi også mindre feil

Universitetet i Stavanger

# Best praksis for forvaltning

Før vi setter systemet i drift må det være klinkende klart hvem som har ansvar for systemet i forvaltningsfasen.

Det er vanlig at en person utpekes til dette i en tidlig fase av prosjektet. Det er en fordel om denne personen deltar i prosjektet.

Forvalrningsarbeidet kan organiseres på mange måter, men det som er viktig er:

- Det er tilstrekkelig med ressurser
- Personene som forvalter systemet har nok kompetanse
- Ressursene er på plass den dagen systemet settes i drift

Universitetet i Stavanger

# Hvem har ansvaret

Mange systemutviklere fraskriver seg ansvaret for denne delen av prosjektet. Spesielt dersom det nye systemet krever endring i brukerorganisasjonen.

«Vi kan da ikke blande oss bort i kundens organisasjon!»

Men dersom ingen tar ansvaret for denne fasen er vi nesten garantert en brokete innføring av systemet. Vi må påta oss rollen som rådgivere. Dette betyr:

- Utrede alternativer
- Informere
- Påpeke konsekvenser

Universitetet i Stavanger

# Hvem har ansvaret

Systemutviklingsprosjekter har som regel flere representanter fra kundens organisasjon som aktive prosjektdeltakere, disse kan fungere som bindeledd inn i kundens organisasjon. Det er nødvendig at prosjektet:

- I planene tar med nødvendige aktiviteter i kundens organisasjon og tidfester disse.
- Sjekker at disse aktivitetene gjennomføres som planlagt og at resultatene holder mål.

Universitetet i Stavanger

# Forvaltning må organiseres

- Brukerfaglig ansvarlig
  - Har ansvar for brukerstøtte
- Systemansvarlig
  - Har ansvar for systemet og forvaltning av systemet
- Driftsansvarlig
  - Har ansvar for drift av systemet
- Helpdesk
  - Tar imot alle henvendelser fra brukerne

Universitetet i Stavanger

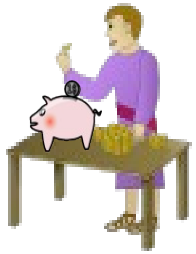
# Forvaltning må organiseres

- Brukerfaglig ansvarlig
  - Har ansvar for brukerstøtte
- Systemansvarlig
  - Har ansvar for systemet og forvaltning av systemet
- Driftsansvarlig
  - Har ansvar for drift av systemet
- Helpdesk
  - Tar imot alle henvendelser fra brukerne

Universitetet i Stavanger

# Forvaltning må organiseres





Universitetet i Stavanger

# En god avtale må inneholde

- Servicenivå
  - Hvilke nivå skal vi ha på serviceavtalen
  - Systemets prioritet
  - Oppetid
  - Systemvakt
  - Prosedyrer for kategorisering av feil
  - Maksimal utrykningstid
- Krav til sikkerhetskopiering og reetablering ved alvorlige feil
- Krav til oppbevaring av data
- Krav til opprettholdt kompetanse

# En god avtale må inneholde

- Bemanning med navngitte personer
- Kostnader og belastning av kostnader
- Spesifikasjon av rutiner som skal følges

Avtalen må legges på rett nivå, den må ikke bli for omfattende.

- Service koster!
- Skal det være kveldsvakt eller ikke

# Det må etableres rutiner for forvaltningen

- Sammen med systemet må det leveres beskrivelse av de rutinene som skal følges ved forvaltning av systemet.
- Rutinene kan være uformelle (telefon e post) Dette avhenger av antall brukere og størrelse på systemet
- For store organisasjoner må og mange brukere må en ha formelle rutiner
- De personen som skal gjøre jobben må være med å lage rutinene

# Endringahåntering

- Dette er rutinene som sørger for at feil blir rapportert og forslag til endringer blir behandlet
- Trinnene i denne prosessen er
  - Innmelding av feil eller endringsforslag
  - Vurdering av feil eller endringsforslag (systemansvarlig, brukerfaglig ansvarlig)
    - Feiler bør prioriteres
  - Retting av feil eller gjennomføring av endring
    - Bør utarbeide en spesifikasjon over endringene
    - Alvorlige feil bør rettes med en gang
    - Mindre alvorlige feil rettes i forbindelse med videreutvikling

Universitetet i Stavanger

# Endringahåntering

- En bør aldri ender den versjonen som er i drift
- Endringer bør foregå i utviklingsversjonen
- Godkjenning av nu versjon
  - En ny versjon må testes på vanlig måte
  - Virker det som har virket før
- Til slutt settes den nye versjonen i drift
  - Den nye versjonen blir ny master

Universitetet i Stavanger

# Dokumentasjon

- Når systemet utvikles, lages det dokumentasjon som gjør det mulig å holde systemet i drift
  - Permanent dokumentasjon
    - Brukerdokumentasjon
    - Forvaltningsdokumentasjon
    - Driftsdokumentasjon
- Dokumentasjonen er ofte mangelfull
  - Tidspress
  - Ikke nok resurser
  - Dokumentasjonen forvitrer
- Krav til god dokumentasjon øker når nye tar over

Universitetet i Stavanger

# Dokumentere eller ikke

- Dess mer dokumentasjon dess bedre?
- Velg ut det som skal dokumenteres
- Kost/nytte analyse for dokumetasjon
- Still følgende spørsmål
  - Hva koster det å utvikle dokumentasjon
  - Hva koster det å forvalte systemet uten dokumentasjon
  - Kan dokumentasjon genereres automatisk
  - Kan dokumentasjonen vedlikeholdes
  - Kan dokumentasjonen oppdateres automatisk
- Sats på å utvikl den minste mengde dokumentasjon som vi kan klare oss med

Universitetet i Stavanger

# Brukerdokumentasjon

- Fører til korrekt og konsis bruk av systemet
- Kan inndeles slik
  - Formål, bruksområde, ansvarlige
  - Oppstilling av aktuelle lover,forskrifter, regler og retningslinjer for området som systemet omfatter
  - Overordnet beskrivelse av systemet
    - Dialogstruktur, skjermbilde, rapportstruktur på og avloggingsprosedyre

Universitetet i Stavanger

# Brukerdokumentasjon

- Brukerrettet beskrivelse av systemet
  - Bruksområde
  - Grensesnitt mot andre systemer
  - Skjermbilder med forklaring av datafelt og regler for utfylling
  - Menyer og kommandoer
  - Funksjoner og behandlingsregler
  - Regler for retting av feilregistrerte data
  - Feilmeldinger
  - Manuell kontroll og avstemmingsrutiner
  - Manuelle rutiner
  - Driftsmessige forhold av betydning
  - Lagringsprosedyrer

Universitetet i Stavanger

# Brukerdokumentasjon

- Oversikt over rapporter
- Rutiner for forvaltning
- Retningslinjer vedrørende sikkerhet
- Ordliste

Universitetet i Stavanger

# Forvaltningsdokumentasjon

- Formål, bruksområde og ansvarlige
- Oppstilling av aktuelle lover,forskrifter, regler og retningslinjer for området som systemet omfatter
- Systembeskrivelse
  - Grensesnitt mot andre systemer
  - Systemarkitektur
  - Database beskrivelse
  - Datafelt beskrivelse
  - Systemets funksjoner
  - Adgangskontroll og autorisasjon
- Sikkerhetskrav

Universitetet i Stavanger

# Forvaltningsdokumentasjon

- Driftsmessige krav og ressurser
- Standarder som systemet benytter
- Rutiner for systemforvaltning
  - Rutiner for forvaltning
  - Rutiner for testing, testdata, forventede testresultater
  - Rutiner for oppdatering av dokumentasjon
  - Rutiner for informering av berørte
  - Bibliotekrutiner
- Begreper
- Programdokumentasjon
- Ordliste

Universitetet i Stavanger

# Driftsdokumentasjon

- Formål, bruksområde og ansvarlige
- Ressurser, maskinvare, programvare, utstyr, kommunikasjon
- Driftsrettet beskrivelse av systemet
  - Avhengighet av andre systemer
  - Oversikt over driftsoppdrag med tilhørende rapporter og avhengigheter mellom oppdrag
  - Beskrivelse av driftsoppdrag med tilhørende rapporter og parametre
  - Bestillingsrutiner for oppdrag
  - Beskrivelse av driftsplanleggingssystem

Universitetet i Stavanger

# Driftsdokumentasjon

- Driftsrutiner
  - Kjøreplan
  - Prioritet ved driftsforstyrrelser
  - Ressursforbruk
  - Manuelle kontroller
  - Konsollmeldinger og hvordan disse behandles
  - Oversikt over driftsfeilmekdinger
  - Rutiner i forbindelse med driftsfeilmeldinger
  - Rutiner i forbindelse med feilmldinger
  - Loggføring og behandling av logger
  - Rutiner for sikkerhetskopiering
  - Rutiner for omstart og oppretting
  - avbruddsrutiner

Universitetet i Stavanger

# Driftsdokumentasjon

- Rapporter
- Oppfølging og driftsvedlikehold av sysemet
- Rutiner for forvaltning, behandling av endringsønsker, feilmeldinger og reklamasjon
- Begreper
- Retningslinjer for sikkerhet
- Ordliste

# Kapittel 15
# Ulike livsløps modeller

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Ulike livsløpsmodeller

- Code-and-fix
- Fossefallsmodellen
- Evolusjonære modeller

Universitetet i Stavanger

# Code-and-fix

Slik drev en også systemutvikling i dataalderens barndom.

*«Fundamental of Software Engineering»*

- «Code-and-fix»
  - Programmerer en systembit
  - Endrer kode og retter feil

Universitetet i Stavanger

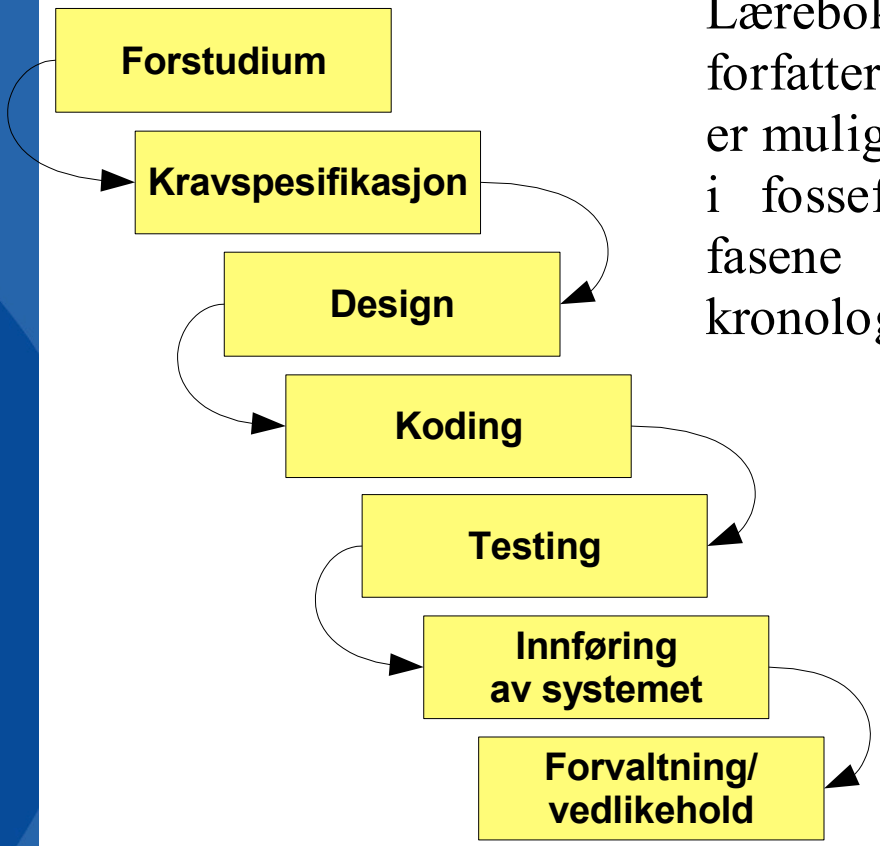
# Code-and-fix

Denne metoden fører til systemer som er uforutsigbare og nærmest umulige kontrollere.

Resultatet var at tanken om systemutvikling som ingeniørdisiplin oppsto, og en prøvde å lage en modell for systemutvikling.

Det ente med at en gikk fra det ene ekstreme til det andre. På 80 tallet ble fossefalsmodellen utviklet og brukt.

# Fossefallsmodellen




Fossefallsmodellen ble svært utbredt. Dette er en modell alle har hørt om. Læreboka har tatt med en fase til, som forfatterne har kalt prosjektvurdering. Det er mulig å legge så mange faser en vill inn i fossefallsmodellen, det viktige er at fasene følger etter hverandre i en kronologisk rekkefølge.

Universitetet i Stavanger

# Fossefallsmodellen

For hver fase er følgende definert:

- Hensikt
- Startbetingelser
- Et sett av resultater som er grunnlag for arbeidet (resultatene fra forrige fase)
- Arbeidet som skal omforme arbeidsgrunnlaget til de definerte resultatet
- Et sett av resultater som kommer ut av fasen
- Sluttbetingelser

Universitetet i Stavanger

# Fossefallsmodellen

Formalismen og kravene til kontroll kan variere mye i de ulike variantene av modellen. Det er stor forskjell i fleksibilitet og anvendbarhet mellom en *resultatorientert modell* som først å fremst styrer ved å definere krav til resultatene i fasen, og en *prosedyreorientert modell,* gir detaljerte instruksjoner for alle trinn og aktiviteter i fasene.

Felles for alle modellene er:

- Modellen er linær
- Modellen er fasedelt
- En fase må være avsluttet før den neste starter
- Hele systemet må gjennomføres i et gjennomløpe
- Modellen er dokumentdreven

Universitetet i Stavanger

# Fossefallsmodellen

Vannet kan ikke renne oppover og det kan en i prinsippet ikke i fossefallsmodellen heller. Dette er ikke riktig, det er mulig å stoppe opp å gå tilbake. Dette har blitt gjort stadig vekk i de prosjektene som har benyttet fossefallsmodellen som livsløpsmodell.

Universitetet i Stavanger

# Fossefallsmodellen

Styrke

- Mye bedre enn «Code-and-fix»

Svakheter

- Krever at vi analyserer oss fram til et komplett og stabilt sett med spesifikasjoner og brukerkrav
- Modellen er monolittisk, den fordrer at vi arbeider på hele systemet
- Den produserer haugevis med dokumentasjon (papper)

Når systemet er ferdig, ender vi kanskje opp med kommentarene

- Det tok lang tid
- Det ble dyrt
- Det er gammeldags

Universitetet i Stavanger

# Fossefallsmodellen -ubrukelig

- Modellen er fullt mulig å bruke, men den må brukes med fleksibilitet
- Modellen er velegnet for et oversiktlig og ikke altfor stort system
- Modellen kan sees på som en generisk modell, den sier noe om fasene i et utviklingsprosjekt

Konklusjonen er:

I mange sammenhenger burde fossefallsmodellen erstattes med andre livsløpsmodeller.

Universitetet i Stavanger

# Evolosjonære modeller

Brooks 1975 og Brooks 1995:

- «Do it twice»

En systemutvikler bør se den første utgaven av systemet som en prøve, noe vi lager for å få erfaring, for senere å kaste det å lage det virkelige systemet. Dette er ideen bak:

- Bruk og kast «prototyping»

Lag en systembit, prøv om det virker, kast hele greia og lag det virkelige systemet,

Denne metoden eliminerer et vesentlig problem med fossefallsmodellen, kravet om et komplett sett av brukerkrav, brukerne vet ikke hva de vil ha før de har sett systemet.

Universitetet i Stavanger

# Evolosjonære modeller

Vi har fått et nytt problem:

I den tiden det tar å utvikle prototypen endrer kravene seg, og når vi er ferdig kan vi ha laget feil system.

Dette kan rettes på ved å levere systemet bitvis.

Hva om vi stabiliserte noen krav, kanskje de mest sentrale, utvikler en bit av systemet som oppfyller disse kravene? Vi lar så brukerne prøve denne delen av systemet. Dette kan gi oss nyttig tilbakemelding og vi tar en ny runde, finner fram til andre krav og fortsetter utviklingen.

Denne måte å utvikle på kaller vi

- En evolusjonær prosessmodell

Universitetet i Stavanger

# Evolosjonære modeller

Det finnes mange ulike evolusjonære prosessmodeller, felles for alle er at at arbeidet skjer inkrementelt. Det er ikke et krav i alle modellene at systemet settes i drift bitvis, selv om utviklingen av systemet skjer i bitvis.

Vi vil nå se på tre ulike evolusjonære modeller

1. RAD-modellen
2. UP-modellen
3. XP-modellen

Universitetet i Stavanger

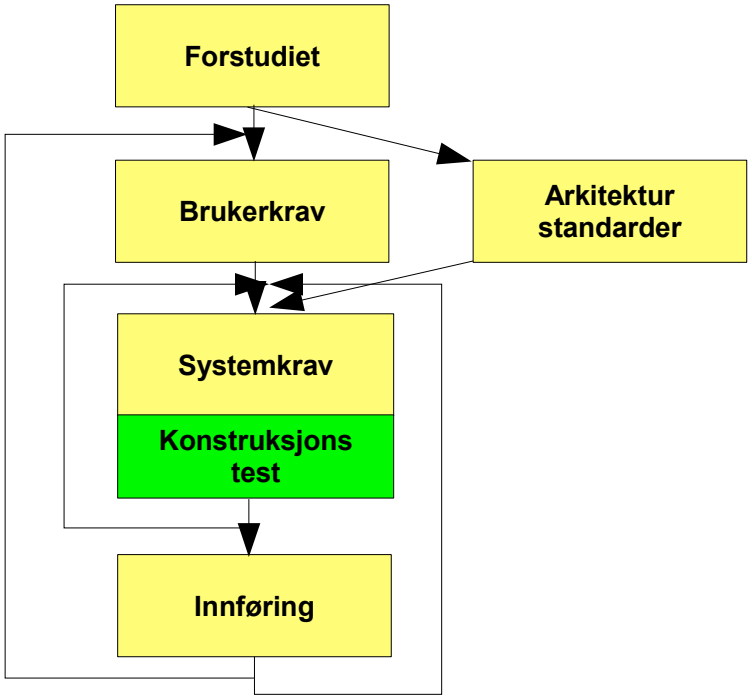
# RAD-modellen

Arbeidsteknikker

- Intensive arbeidsmøter med kreative teknikker som idedugnat og lignende
- Visuelle teknikker, modellbygging, prototyping
- Bruk av spesialistteam som jobber tilnærmet 100% i prosjektet
- Gjenbruk av programmoduler
- Tidsboksing, det settes opp faste tidsrammer for ulike oppgaver, og man gjør så mye en rekker av oppgaven innenfor rammene. Når tiden er over, erklæres oppgaven som avsluttet.

Universitetet i Stavanger

# RAD-modellen



Universitetet i Stavanger

# Vurdering RAD-modellen

RAD-modellen har mange kvaliteter

- Gjenbruk av kode
- Rask utvikling
- Kundene er aktive i utformingen
- Bitvis utviklingen
- Mindre papper

Problemer med metoden

- Planlegger for dårlig
- Det hele går for fort, ting får ikke modnes
- Gjenbruk er vanskelig å få til
- Det er vanskelig å få til konsentrert innsats både av kunder og utviklere

Universitetet i Stavanger

# Vurdering RAD-modellen

Konklusjon:

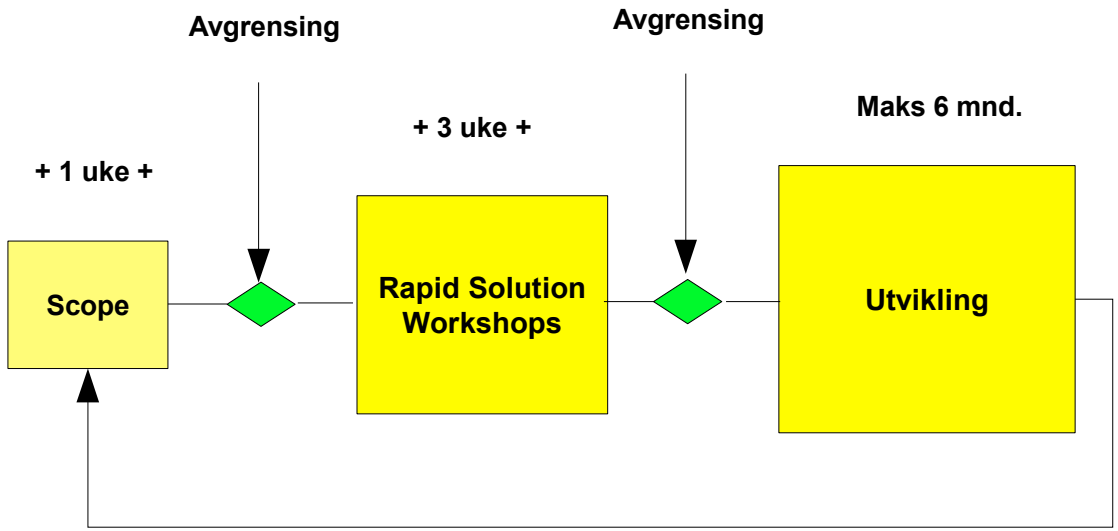
- RAD-prosjekter krever nøye planlegging og tett oppfølging.
- Det nytter ikke å utvikle biter av systemet i iterasjon hvis ikke hver bit er nøyaktig beskrevet i forhold til helheten.
- Det må finnes en overordnet plan slik at hver enkelt leveranse vet nøyaktig hva som skal gjøres i neste omgang
- Det nytter ikke å gå til et intensivt arbeidsmøte, dersom det ikke er planlagt i detalj
- Utviklingen underveis i møtene må følges opp kontinuerlig

Hvordan kan RAD-modellen benyttes samtidig som en lar problemstillinger modnes?

- Ikke gjøre for mye på en gang
- Erfaring fra utviklede løsninger tas med i neste iterasjon. Har vi gjort feil får vi gjøre ting på nytt.
- Rad-modellen krever mye av kundene, de må være aktive. Både beslutningstakere og brukere.

# Eksempel RAD-modellen

Eksempelet er hentet fra et firma som heter «Cambridge Technology Partners» (CTP)
I utviklingen blir tid og pris holdt konstant, det som reguleres er omfanget. Den faste prisen gjelder en definert bit av systemet.



Universitetet i Stavanger

# Eksempel RAD-modellen

Modellen har tre faser

- Scope
  - Bestemmer mål og ramme, dette avgrenses til noe som virker realistisk
- Rapid Solutions Workshops
  - Utforme løsningen funksjonelt ved å bygge en prototype
  - De aktuelle personene sitter sammen og jobber i lag
  - Løsningen avgrense ytterligere dersom rammene sprekker
- Utvikling
  - Her foregår en tradisjonell utvikling av de bitene en er blitt enige om skal med

Legg merke til at denne modellen ikke har noen fase for innføring.

Universitetet i Stavanger

# Unified Process

Denne modellen er resultat av den «objektorienterte bølgen» Tre av pionerene

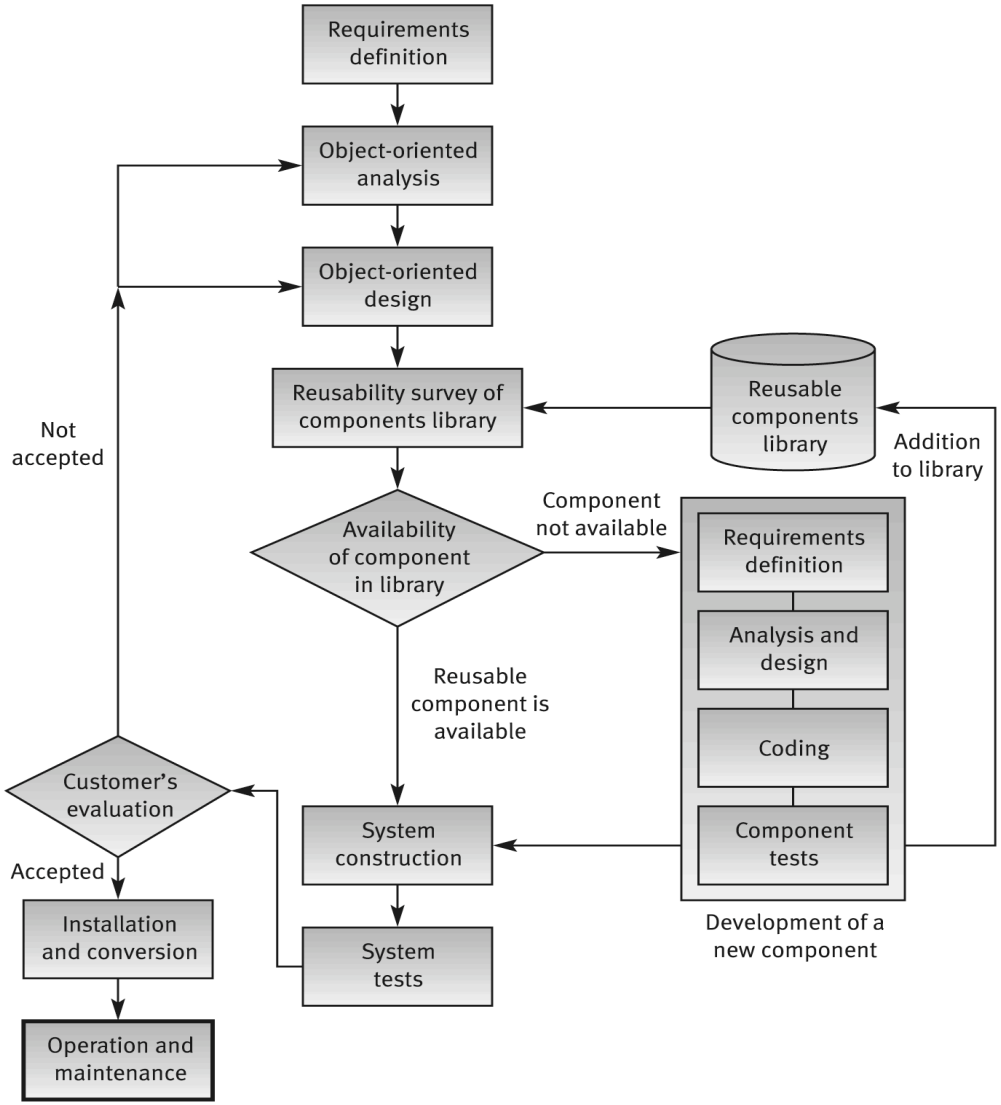
- James Raumbaug
- Grady Booch
- Ivar Jacobsen

Arbeidet fra disse tre har ført til det vi kjenner som UML diagram. UML er blitt et alment anerkjent notasjonsspråk. UML er også utvidet med ideer fra andre.

UP er blitt en prosessmodell for objektorientert utvikling

Universitetet i Stavanger

# Unified Process



Universitetet i Stavanger

# Unified Process

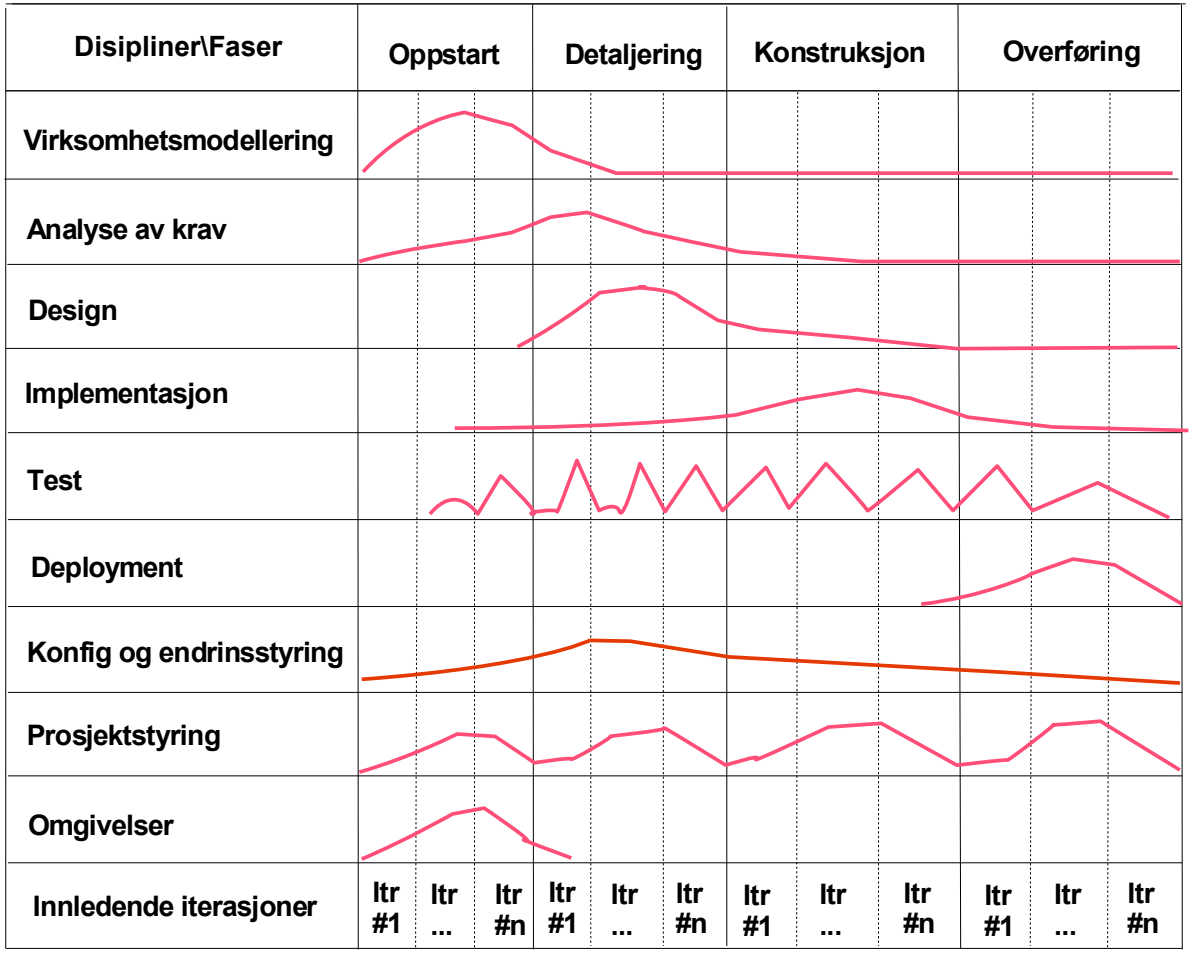
Målet med modellen er å legge til rette for produksjon av programvare av høy kvalitet, som tilfredsstiller brukerkravene, og som leveres innenfor forutsigbare tids og budsjettrammer. Prosessen skal kunne skreddersys til å passe mange forskjellige prosjekter og organisasjoner.

I UP skal en produsere og vedlikeholde modeller ikke dokumenter. Modellene er ulike sider ved et objektorientert system.

Modellene skal vise forskjellige sider ved en objektorientert struktur. UP er arkitektursentrert. Det betyr at en tidlig i prosessen må lage en grunnleggende arkitektur. Den arkitekturen skal være robust og må kunne utvides.

Universitetet i Stavanger

# Unified Process




Unified Process – faser og disepliner og iterasjoner

Universitetet i Stavanger

# Unified Process fire faser

Oppstart(Inspection)

- Grunnlaget for prosjektet legges.
  - Finner fram til suksesskriterier
  - Vurderer risiko
  - Estimerer resursbehov
  - Utformer en overordnet tidsplan med faser og milepæler

Prototyping er nyttig i denne fasen, denne fasen kan bli sett på som et forstudium. Ut fra informasjonen som graves fram avgjør en om prosjektet skal fortsett

Universitetet i Stavanger

# Unified Process fire faser

Detaljering (Elaboration)

- Problemet analyseres
- Får på plass arkitektur
- Detaljerer prosjektplaner
- Prøver å eliminere risiko
- Hovedkravene til systemet beskriver
- De viktigste use case realiseres

Universitetet i Stavanger

# Unified Process fire faser

Konstruksjon

- Her framstilles systemet gjennom en rekke iterasjoner
- Krav og akseptsansekriterier spesifiseres svært nøyaktig
- Etter at systemet er tatt i bruk kan nye krav og problemer dukke opp, da må livsløpet starte på nytt

Overlevering (Transition)

- Systemet leveres til brukerne. Dette er ofte en betaversjon
- Etter at systemet er tatt i bruk kan nye krav og problemer dukke opp

Universitetet i Stavanger

# Unified Process

Slutten av hver fase ender i en milepæl. Hver fase i prosessen kan deles i mange iterasjoner. Hver fase er et fullstendig utviklings løp som skal resultere i et frislipp (releas) som enten er internt eller eksternt. Vert slikt frislipp er en kjørbar, men ufullstendig del av systemet som er under utvikling. Det nye systemet blir gradvis komplett etter hver iterasjon. Hver iterasjon kan bli sett på som et mini fossefallsløp, men iterasjonene i hver fase har ulikt fokus

Diseplinene representerer ting som må gjøres. Hver disiplin har et sett av artefakta. Et poeng er at hver disiplin inneholder nesten alle aktiviteter, men med forskjellig omfang og man arbeider med samme artefakta. I starten vil artefaktaene stadig endre seg, men etter hvert konvergerer de mot en stabil tilstand.

Universitetet i Stavanger

# Unified Process fire faser

Konstruksjon

- Her framstilles systemet gjennom en rekke iterasjoner
- Krav og akseptsansekriterier spesifiseres svært nøyaktig
- Etter at systemet er tatt i bruk kan nye krav og problemer dukke opp, da må livsløpet starte på nytt

Overlevering (Transition)

- Systemet leveres til brukerne. Dette er ofte en betaversjon
- Etter at systemet er tatt i bruk kan nye krav og problemer dukke opp

Universitetet i Stavanger

# Unified Process fire faser

Konstruksjon

- Her framstilles systemet gjennom en rekke iterasjoner
- Krav og akseptsansekriterier spesifiseres svært nøyaktig
- Etter at systemet er tatt i bruk kan nye krav og problemer dukke opp, da må livsløpet starte på nytt

Overlevering (Transition)

- Systemet leveres til brukerne. Dette er ofte en betaversjon
- Etter at systemet er tatt i bruk kan nye krav og problemer dukke opp

Universitetet i Stavanger

# Disipliner innen UP

- Virksomhetsmodellering. Her modelleres foretningsprosesser. Målet er å få fram kunnskap om det virksomhetsområdet som det skal utvikles programvare for
- Analyse og krav. Målet er å få fram hva systemet skal gjøre. Man skal fram til funksjonelle krav uttrykt ved use case. Det utarbeides en visjon for systemet.
- Design. Her beskriver en hvordan systemet skal realiseres. Det fastlegges en arkitektur, hvilke objekter programvaren skal bestå av, eventuell database, nettverk o.s,v.
- Implementasjon. Dette omfatter aktiviteter for å realisere systemet. Koding av klasser utføres. Systemet organiseres i delsystemer og komponenter. Enheter testes

Universitetet i Stavanger

# Disipliner innen UP

- Deployment. Dette er aktiviteten for å produsere et frislipp og levere programvare til sluttbruker. Det inkluderer
  - Planlegging
  - Gjennomføring av betatester
  - Konvertering av eksisterende programvare og data
- Konfigurasjons og endringskontroll. Denne delen skal sikre at de ulike arkefakta ikke er i konflikt med hverandre. Man må ha kontroll med oppdateringer, endringer, versjoner, varianter og frislipp av programvaren.
- Prosjektstyring. Gjennomgående kaktiviteter der en følger opp og planlegger progresjonen i prosjektet.
- Omgivelsesdisiplinen. Aktiviteter for å få på plass det som trengs for å støtte gjennomføringen av prosjektet. Man skal finne fram til og tilrettelegge nødvendig verktøy og tilpasse prosessen til det aktuelle prosjektet.

Universitetet i Stavanger

# Vurdering av UP

UP gjør bruk av disse beste praksiser:

- Objektorientert teknologi
- Tidlig håndtering av risiko
- Tidlig kjørbart system
- Kontinuerlig engasjement av brukerne
- Tidlig testing
- Visuell modellering av systemet

Ulemper

- Tungt og byråkratisk
- Produserer mange arkefakter

Universitetet i Stavanger

# *XP – eXtreme Programming*

«eXtreme Programming (XP)» er en prosessmodell som har vind i seilene, og som mange snakker om. Mange nye systemer utvikles for Internett, disse preges av:

- Hyppig endring i teknologi

For slike systemer er lettvektsprosesser enset mulige utviklings løp. I XP drives beste praksis til det ekstreme. De praksiser som over tid har vist seg å føre til gode resultater er grunnlaget. Hovedpersonen bak XK er Kent Beck.

Universitetet i Stavanger

# XP – eXtreme Programming

Kent Beck om XP

- Hvis koderevisjon er bra, må det gjøres hele tiden. To programmerere må arbeide sammen hele tiden (Parprogrammering)
- Hvis testing er bra, skal alle teste hele tiden, også kunden
- Hvis design er bra, må alle gjøre det hver dag
- Hvis enkelhet er bra, skal systemet være i enklest form som støtter ønsket funksjonalitet
- Hvis arkitektur er viktig, skal alle arbeide med å definere og raffinere arkitekturen
- Hvis integrasjonstesting er viktig, skal man integrere og teste mange ganger hver dag
- Hvis korte iterasjoner er bra, gjør man iterasjonene virkelig korte – sekunder, minutter, timer, ikke uker og måneder og år.

Universitetet i Stavanger

# XP – eXtreme Programming

Beck legger vekt på at XP er en morsom måte å utvikle på. Den skiller seg fra andre prosesser ved:

- Tidlig, konkret og kontinuerlig tilbakemelding fra korte iterasjoner
- Inkrementell planlegging som tidlig frambringer en grov overordnet plan som deretter oppdateres stadig gjennom livsløpet
- Fleksibel implementasjon av funksonalitet i takt med endrede behov
- Bruk av autmatiske tester skrevet av programmerer og kunde for tidlig å fange opp defekter
- Muntlig kommunikasjon, tester og kildekode for å kommunisere struktur og hensikt
- Bruk av en evolusjonær design prosess som varer så lenge systemet varer
- Tett samarbeid mellom programmerere med gjennomsnittlige ferdigheter
- Praksiser som programmerere instinktivt mener er riktige i det daglige, og som tjener prosjektet på lang sikt.

Universitetet i Stavanger

# XP – eXtreme Programming

Det som er innovativt med XP, er at alle gjennomprøvde praksiser settes under en paraply, at man sikrer at praksisene gjøres så nøye som mulig og at man sikrer at alle som er involvert, støtter hverandre på best mulig måte.

Universitetet i Stavanger

# XP – eXtreme Programming

I XP har en 4 hovedaktiviteter:

1. Koding
2. Testing
3. Lytting
4. Design

Universitetet i Stavanger

# XP – eXtreme Programming

Beck vektlegger de fire verdier

- Stadig kommunikasjon med kundene og mellom medlemmer i utviklingsprosjektet
- Enklest mulig løsning
- Hurtig tilbakemelding ved hjelp av enhetstester og integrasjonstester
- Mot til å håndtere problemer i forkant

I XP er endring normalt!

Universitetet i Stavanger

# XP – Planleggingsspillet

- Deltakere:
  - Utvikling og virksomhet
- Brikkene er kartotekkort
  - På kortene skrives «strories» Et par setninger som beskriver en del av systemet
- Virksomheten legger fortellingene i tre bunker
  - De som er nødvendige
  - De som er mindre viktige
  - De som er kjekke å ha

Universitetet i Stavanger

# XP – Planleggingsspillet

- Utviklerne legger fortellingene i tre bunker
  - Fortellinger som kan estimeres ganske nøye
  - Fortellinger som kan estimeres rimelig nøye
  - Fortellinger som ikke kan estimeres i det hele tatt
- En iterajon planleges med det antall fortellinger som kan realiseres innen en bestemt tid, og inkrementet utvikles. Virksomheten bestemmer tidspunktet for når et resultat må foreligge, og Utvikler estimerer hvor mange fortellinger som det er mulig å realisere innen den tiden. Det vanlige er å ha to uker som tidshorisont for et frislipp av en bit av systemet.

Universitetet i Stavanger

# XP – Vurdering

XP er omstridt og er neppe vidunderkuren som kurerer alle problemer knyttet til programutvikling. XP passer neppe for alle typer programvare, men XP har fått en entusiastisk tilhengerskare.

Programmerere er tiltrekkes av

- Mindre byråkrati
- Mindre papper

Ledere skremmes av

- Ekstra kostnader
- Mindre kontroll

XP virker fornuftig for en del typer systemutvikling og må vurderes som en mulig livsløpsmodell.

# Kapittel 16
# Kvalitetsplanlegging

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Kvalitetsplanlegging

Kvalitetsplanlegging betyr:

- Kvalitet er målet og veien planlegges slik at målet blir nådd

Resultatet av planleggingsprosessen beskrives i en **kvalitetsplan**.

Det finnes ulike standarder for kvalitetsplaner og i denne delen av forelesningene vil vi se på innholdet i noen av disse standardene.

# Kvalitetsplan

Hva er en kvalitetsplan?

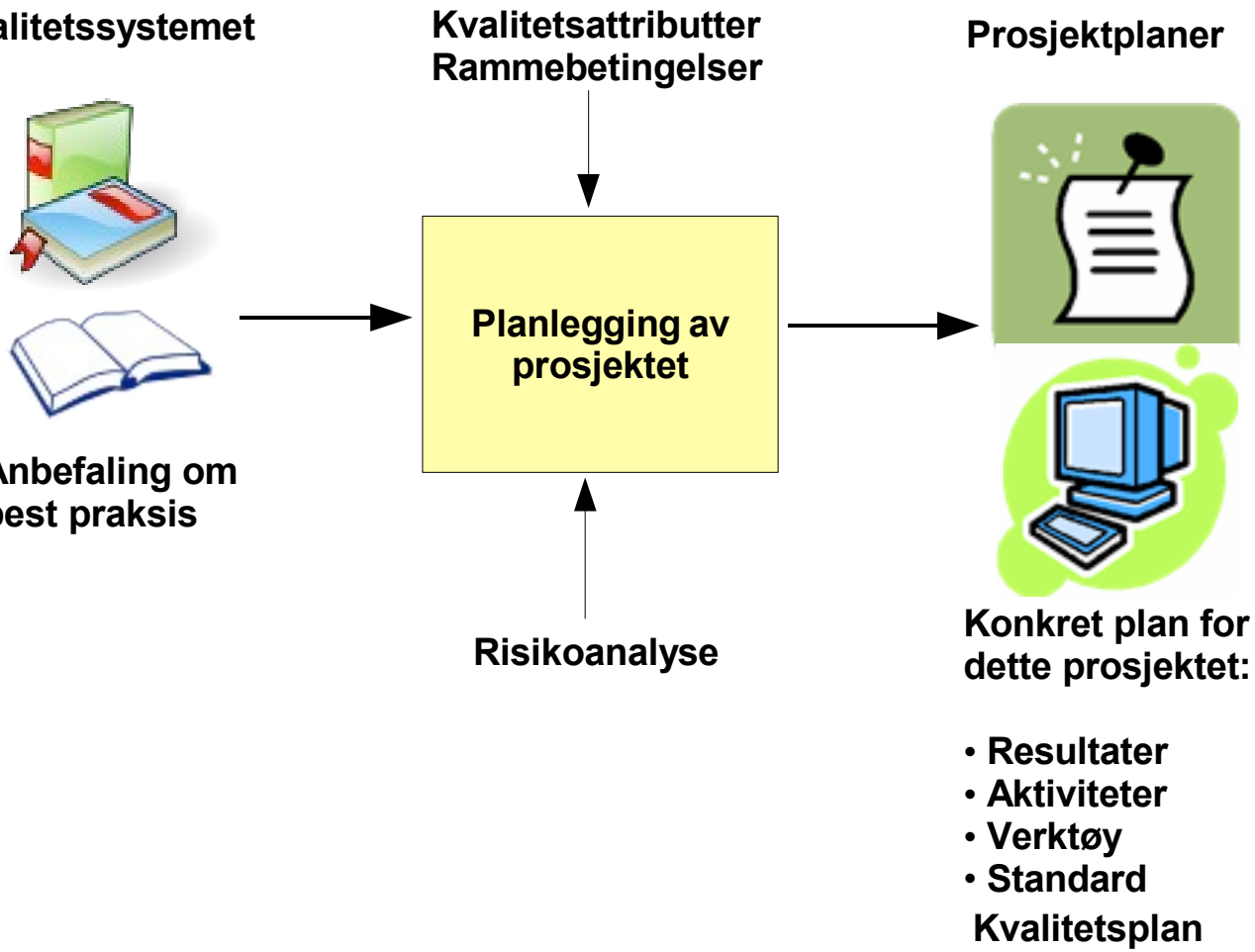
> En kvalitetsplan er et dokument som viser hvordan kvalitetstyringen skal foregå for en bestemt ordre eller et bestemt prosjekt. Kvalitets planen viser hvordan vi tilpasser bedriftens kvalitetssystem til det konkrete prosjektet og den konkrete ordren.

*For produksjon etter samlebåndsprinsippet gjelder*

- All produksjon skal ha de samme egenskaper
- Ønskelig med minst mulig variasjon
- Produksjonen må gjennomføres mest mulig likt

I et kvalitetsystem for produksjon må en ha detaljerte prosedyrer for framstilling av produktene. Produksjonen må følge prosedyrene til punkt og prikke.

# Kvalitetsplan



Universitetet i Stavanger

# Kvalitetsplan

Prosjekter og ordrer av spesielle produkter er engangs-oppgaver. Her gjelder andre regler. Kvalitetssystemet kan ikke ha prosedyrer som passer til alle engangs-oppgavene. I kvalitetsystemet satser en på gode generelle krav og retningslinjer, disse tilpasses til den aktuelle engangs-oppgaven/prosjektet.

Når prosjektet planlegges. Er det viktig å tilpasse de generelle retningslinjene til det konkrete prosjektet. Utgangspunktet er de spesielle utfordringen som prosjektet har, ut fra disse skreddersyr vi en gjennomføringsplan. Denne skreddersømmen beskriver vi i Kvalitets planen. I denne Kvalitets planen viser vi konkret hvilke kvalitetsikringstiltak som skal gjennomføres i prosjektet.

Universitetet i Stavanger

# Kvalitetsplan

Bedriftens kvalitetsystem er utgangspunktet for kvalitets-planen, men systemets kvalitets-attributter og rammebetingelsene for prosjektet er viktige når retningslinjene i det generelle kvalitetssystemet skal tilpasses i en kvalitetsplan. I tillegg må en ta hensyn til resultatene fra risikoanalysen.

Utvikling av programvare er en typisk engangsoppgave. Hvert prosjekt er unikt, og selv om vi finner støtte i kvalitetsystemet, må vi planlegge gjennomføringen slik at vi tar hensyn til det som er spesielt for prosjektet. Det bør derfor alltid utarbeides en kvalitetsplan. Følger bedriften ISO 9000:2000 standarden er dette et krav.

Universitetet i Stavanger

# Kvalitetsplan

Kvalitets planen bør være kort og lettfattelig. Humphery 1989 foreslår følgende innhold:

1. Produktbeskrivelse
   - En overordnet beskrivelse av det fremtidige systemet, hvem som skal bruke det og hensikten med å utvikle systemet
2. Overordnet prosjektplan
   - Milepæler og leveranser og beskrivelse av hvem som har ansvar for hva. Dette kan enten være i selve planen eller en referanse til et annet dokument
3. Beskrivelse av livsløpsprosessen og støtteprosessene (testing, konfigurasjonsstyring etc.) som prosjektet skal følge.
   - Her refereres til kvalitetssystemet og til konkrete standarder som følges. Eventuelle avvik fra kvalitetssystmet må dokumenteres

Universitetet i Stavanger

# Kvalitetsplan

4. Kvalitetsmål
   - Kvalitetsattributtene for systemet, eller henvis til dokument
   - Prosessen prosjektet skal følge for å verifisere at kvaliteten er oppnådd dokumenteres.
   - Godkjenningsprosessen dokumenteres med:
     - Framgangsmåte
     - Kriterier for godkjenning
     - Ansvar for godkjenning
     - Framgangsmåte dersom systemet ikke godkjennes
5. Resultatet av risikoanalysen
   - Selve analysen eller henvising til dokumentet som inneholder analysen

Universitetet i Stavanger

# *ISO 9000:2000 om kvalitetsplaner generelt*

Standarden forlanger at det skal utarbeides en skriftlig kvalitetsplan for

- prosjekter med nye produkter, tjenester eller prosesser

Retningslinjer for å utarbeide kvalitetsplaner finnes i ISO 10005:1995, dette er en utdypning av ISO 9000:2000, men den er ikke revidert etter at den nye ISO serien kom. Her er de mest sentrale elementene i planen:

# ISO 9000:2000 om kvalitetsplaner generelt

1. Hva gjelder planen for
   - Produkt/Prosjekt
2. Fordeling av ansvar og myndighet hos levrandøren (oppdragsgiver)
3. Referanse til spesielle prosedyrer, metoder og arbeidsinstrukser i kvalitetssystemet som skal anvendes
4. Eventuelle tillegg til prosedyrer, arbeidsinstrukser i kvalitetssystemet når det er nødvendig

# ISO 9000:2000 om kvalitetsplaner generelt

5. Hensiktsmessige kontrollrutiner for
   - Gjennomgang av kontrakten
   - Kontroll av design
   - Dokument og datakontroll
   - Kontroll med produkter levert av kunden
   - Kontroll av innkjøpte produkter
   - Kontroll med leveranser
   - Sporing og endringskontroll
   - Prosesskontroll

# ISO 9000:2000 om kvalitetsplaner generelt

6. Oversikt over planlagte inspeksjoner og tester, inkludert kontroll av utstyr til inspeksjon og testing
7. Opplegg for endring og modifikasjon av Kvalitets planen
8. Andre tiltak som er nødvendige for å nå kvalitetsmålene

Universitetet i Stavanger

# ISO 9000-3: kvalitetsplanen systemutvikling

- Kvalitetsmål, hvis mulig i målbare størrelser
- Fasemodellen som følges
- Start og resultatkriterier for hver fase
- Fastsetting av gjennomganger, tester, verifisering, validering
- Identifikasjon av prosedyrer for konfigurasjons styring
- Detaljerte planer også tidsplaner for
  - Konfigurasjonsstyring
  - Verifikasjon og validering av alle resultatene
  - Avvikskontroll
  - Endringskontroll
  - Oppfølging av Kvalitets planen
- Spesifikt ansvar for kvalitetsaktiviteter

Universitetet i Stavanger

# IEEE: kvalitetsplanen bør inneholde

Kravene er:

1. Hensikten med planen, hva gjelder den for
2. Oversikt over dokumentene som refereres i planen
3. Beskrivelse av ansvarsforhold
4. Oversikt over dokumenter som skal utvikles, rutiner for dokumentkontroll. Standarden inneholder minimumskrav til hvilke dokumenter som skal utvikles. Der gjelder
   - Kravspesifikasjon
   - Designdokumentasjon
   - Plan for verifisering og validering
   - Rapport fra verifikasjoner og validering
   - Brukerdokumentasjon
   - Plan for konfigurasjons styring

Universitetet i Stavanger

# IEEE: kvalitetsplanen bør inneholde

5. Standarder som skal følges. Det gjelder
   - Dokumentasjonstandard
   - Standard for logisk strukturer
   - Kodestandard
   - Standard for kommentering i koden
   - Testestandarder
6. Kvantitative kvalitetsmål som det skal måles mot

Universitetet i Stavanger

# IEEE: kvalitetsplanen bør inneholde

7. Kontroller og gjennomganger. Her har standarden med minimumskrav til følgende gjennomganger:
   - Gjennomgang av kravspesifikasjon
   - Gjennomgang av overordnet design
   - Gjennomgang av detaljert design
   - Gjennomgang av plan for verifisering og validering
   - Sluttkontroll av funksjonalitet og av systemets indre struktur
   - Løpende kontroll underveis av samsvar mellom krav, design, arkitektur, kode
   - Gjennomgang av plan for konfigurasjonsstyring
   - Erfaringsinnsamling etter prosjektslutt

Universitetet i Stavanger

# IEEE: kvalitetsplanen bør inneholde

8. Identifikasjon av alle tester
9. Prosedyre for feilrapportering og retting av feil
10. Bruk av spesielle verktøy, teknikker og metoder for å støtte Kvalitets planen
11. Metoder og utstyr for å lagre, sikre dokumentere versjoner av programvaren
12. Kontroll med utstyr
13. Kontroll med programvare levert av andre som inngår i produktet
14. Prosedyre i forbindelse med utvikling og oppbevaring av kvalitetsplandokumenter
15. Nødvendig opplæring
16. Risikoanalyse

# Hva er tilstrekkelig kvalitetssikring

Målet for et utviklingsprosjekt er:

*Sikre en effektiv gjennomføring og oppnå gode resultater*

Gjennomføringen sikres ved:

*Gjennomføringsplanen*

Kvaliteten skires ved:

*Kvalitetsplanen*

Det er viktig at tiltakene i kvalitets-planen samsvarer med prosjektet. For mange og omfattende tiltak koster både tid og ressurser og det går ut over effektiviteten. For få tiltak går ut over kvaliteten.

# *Hva er tilstrekkelig kvalitetssikring*

Det viktige spørsmålet blir

***Hva er tilstrekkelig kvalitetssikring i mitt prosjekt?***

Noen hjelpemidler for å finne svar:

- Bedriftens kvalitetssystemet
- Generelle retningslinjer som bør følges
- Bedriften har dokumentert erfaringer fra tidligere prosjekt
- Vi kan få støtte råd av andre i bedriften

Universitetet i Stavanger

# Hvorfor tilpassinger

Her er noen grunner til å gjøre spesielle tilpasninger

- Prosjektet har behov for å utnytte spesielle metoder og teknikker eller verktøy på en best mulig måte.
  - Verktøy for prototype
  - Verktøy for modellering
- Vi har behov for å forenkle gjennomføringen når risikoen er liten
- Vi har behov for å innføre ekstra kvalitetsikringstiltak når risikoen er stor.
- Vi har behov for å gjennomføre andre aktiviteter eller levere andre typer dokumenter enn dem som kvalitetssystemet beskriver.

Universitetet i Stavanger

# Noen huskeregler

Hvordan skal prosjektet organiseres?

- Hvem skal ta beslutningene som er nødvendige?
- Hvem må få uttale seg?
- Hvem kan kvalitetssikre resultatene?
- Hvem har den kompetansen vi trenger?

Ut fra svarene på disse spørsmålene fordeles roller og ansvar. Ta utgangspunkt i den anbefalte måten å organisere prosjekter på, det skal finnes i kvalitetssystemet. Gjør de tilpasninger som er nødvendige og pass på at et lite prosjekt ikke blir topptungt. Men ta ikke bort

- Styringskomiteen
- Kvalitetskontrollen

Universitetet i Stavanger

# Noen huskeregler

Hvilke faser skal prosjektet deles inn i?
Spør heller

Hvilke beslutninger trengs underveis?

Definer viktige milepæler eller hendelser i prosjektet der det er nødvendig med en beslutning.

Eksempel

- Prioritering av krav

Del prosjektet i faser slik at en får et beslutnings punkt på disse stedene. Utgangspunktet er kvalitetsystemet, men vi må vurdere om vi trenger færre eller flere faser enn de som står beskrevet i kvalitetssystemet.

Universitetet i Stavanger

# Noen huskeregler

Hvordan skal beslutningsprosessen være?
For mange prosjektledere er styringskomitémøtene og kvalitetskontrollmøtene en plage. De krever ressurser og tar tid. Det hender at prosjektet forsinkes mens en venter på et slikt møte. Samtidig er det viktig å sikre at sentrale resultater i prosjektet får god nok behandling. Dette gjør at det er god investering å planlegge nøye beslutningsprosessene i prosjektet.

Er det mulig å bruke

- Telefonkonferanse
- Videokonferanse

Er det mulig å unngå store formelle møter, men likevel få tilstrekkelig tyngde i beslutningsprosessen?

Universitetet i Stavanger

# Hvilke resultater skal kontrolleres?

Uavhengig gjennomgang av prosjektresultatene er nyttig, og brukes altfor sjelden.

Når prosjektet planlegges bør de resultatene det er viktig å validere (er gode nok) og verifisere (er korrekte) plukkes ut.

Deretter må en finne personene som kan utføre gjennomgangen og hvilke prosedyrer som skal følges.

For å gå gjennom brukerkrav trenger en representanter fra brukerne og en erfaren systemutvikler. Dersom det er datamodellen som skal kontrolleres, vil vi benytte en ekspert på datamodellering.

Denne type gjennomgang kan gjøres formelt eller uformelt.

Råd:

> Bruk uavhengig gjennomgang så mye som mulig, gjør heller kravet til formalisme mindre.

Universitetet i Stavanger

# Hvilke resultater skal kontrolleres?

Uavhengig gjennomgang av prosjektresultatene er nyttig, og brukes altfor sjelden.

Når prosjektet planlegges bør de resultatene det er viktig å validere (er gode nok) og verifisere (er korrekte) plukkes ut.

Deretter må en finne personene som kan utføre gjennomgangen og hvilke prosedyrer som skal følges.

For å gå gjennom brukerkrav trenger en representanter fra brukerne og en erfaren systemutvikler. Dersom det er datamodellen som skal kontrolleres, vil vi benytte en ekspert på datamodellering.

Denne type gjennomgang kan gjøres formelt eller uformelt.

Råd:

> Bruk uavhengig gjennomgang så mye som mulig, gjør heller kravet til formalisme mindre.

# Hvilke aktiviteter må vi ha med i planen?

Vi må ha med de resultatene som er nødvendige for å produsere de resultatene vi har planlagt å lage.

# Er dette en god plan?

Dette er det viktigste spørsmålet. Sørg for å få

- En vurdering av gjennomføringsplanen og kvalitetsplanen
- En godkjenning av gjennomføringsplanen og kvalitetsplanen

Når planen er godkjent, er det planen vi skal måles mot og ikke generelle retningslinjer.

Vurdering av planen før godkjenning er lurt, en styringskomité er ikke alltid den rette instansen til å avgjøre kvaliteten på planen.

# Kapittel 17
# Risikoanalyse

Terje Kårstad

Universitetet
i Stavanger

# Å beskytte seg mot uønskede hendelser

Et viktig element når vi planlegger for kvalitet, er å vurdere om det finnes trusler som kan hindre oss i å nå målene. Hensikten er å forstå det landskapet vi skal gi oss inn i, slik at vi kan være best mulig forberedt. Og ikke minst finne mottiltak før det skjer noe alvorlig.

En slik vurdering gjør vi ved hjelp av **risikoanalyse**. Selv om risikoanalysen er en del av kvalitetsplanleggingen, blir den omtalt her.

# Risiko

Hva er risiko?

- Risiko er sannsynlighet for at en uønsket hendelse intreffer og konsekvensene dersom det skjer.

Hva er risikoanalyse?

- Å finne frem til uønskede hendelser (risikofaktorer) som prosjektet må beskytte seg mot
- Bedømme sannsynligheten for at de vil inntreffe
- Analysere hvilke konsekvenser det vil få hvis de inntreffer
- Vurdere virkemidler som kan hindre at risikofaktorene oppstår, eller som kan minske konsekvensen hvis det skjer

Universitetet
i Stavanger

# Risiko

Universitetet i Stavanger

# Prosjektets risikofaktor

En risikofaktor er en hendelse som skaper en uønsket situasjon i prosjektet hvis den oppstår. Vi vil ha et tap forbundet med hendelsen: tap av tid, penger, kvalitet, kontroll, forståelse og lignende.

Et eksempel kan være at en ekspert som prosjektet er avhengig av slutter i firmaet. Vi vil da få tap i kompetanse, men i tillegg vil vi sannsynligvis også få tap i tid, av penger, kanskje av kvalitet.

Universitetet i Stavanger

# Prosjektets risikofaktor

Det er vanlig å skille mellom ulike kategorier av risikofaktorer

- De generelle
  - Dette er risikofaktorer som alle prosjekter har. Eksempel:
    - Misforstår kravene
    - Mister nøkkelkompetanse
    - Vi får for dårlig tid til å teste
- De prosjektspesifikke
  - Dette er risikofaktorer som er spesielle for vårt prosjekt. Eksempel:
    - Vi er avhengig av et annet prosjekt eller en leveranse

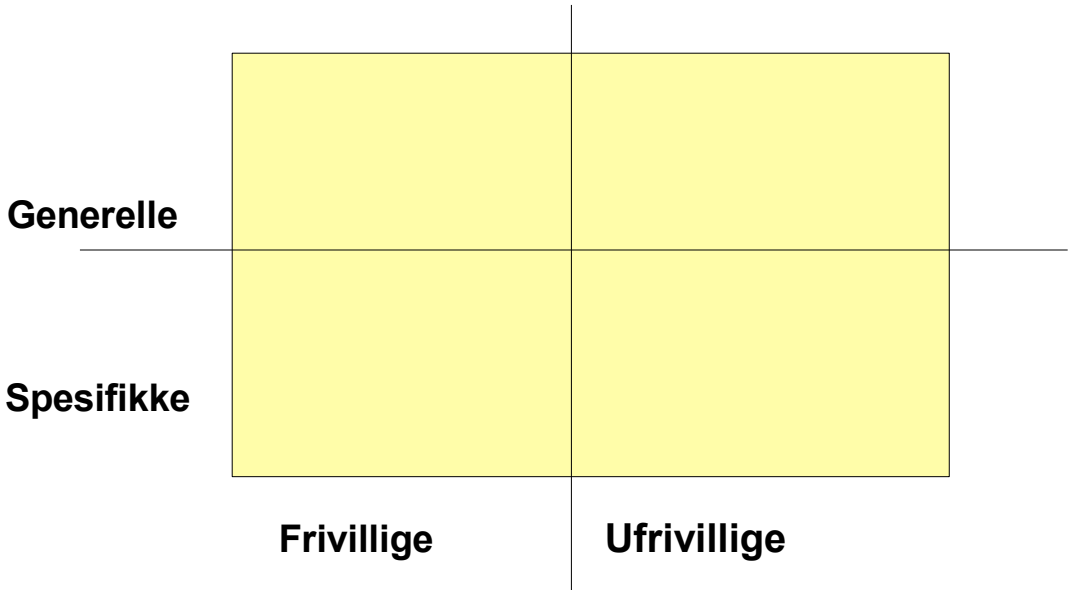
Universitetet i Stavanger

# Prosjektets risikofaktor

- De ufrivillige
  - Dette er risikofaktorer som påføres av omgivelsene rundt oss. Eksempel:
    - En nøkkelperson blir syk eller slutter
- De frivillige
  - Dette er risikofaktorer vi velger å ta. Eksempel:
    - Vi velger å bruke et verktøy vi aldri har brukt før
    - Vi prøver å utvikle noe vi vet ikke er gjort før

Universitetet i Stavanger

# Ulike kategorier risikofaktorer



Universitetet i Stavanger

# Risikofaktor

Barry W. Boehm har i «Software Risk Management» (Boehme 1991) en liste over ti-på topp risikofaktorer som de fleste prosjekter bør vurdere i risikoanalysen. Her er listen:

1. Mangel på kvalifisert personell eller problemer knyttet til personell, f.eks samarbeidsproblemer. Dette gjelder personell både i prosjektgruppa og brukerorganisasjonen
2. Urealistiske tidsplaner eller budsjetter
3. Utvikling av feil system (det var ikke dette systemet brukerne trengte)
4. Utvikling av feil brukergrensesnitt. Det fungerer av ulike årsaker ikke for systemets framtidige brukere
5. «Gold platting» (forgylling – vi utvikler systemet med unødvendige detaljer eller funksjoner eller mer avansert enn det det er behov for)

Universitetet i Stavanger

# Risikofaktor

6. Hyppige endringer av kravene underveis
7. Dårlig kvalitet på eller forsinkelser i systemkomponenter som utvikles av andre
8. Dårlig kvalitet eller forsinkelser i systemkomponenter som kjøpes inn
9. Kapasitetsproblemer i det ferdige systemet
10. Bruk av ny og uprøvd teknologi, eller bruk av teknologien på grensen av det den er beregnet til

Universitetet i Stavanger

# Risikofaktor

Lærebokforfatteren har lagt til noen for egen regning

- Vi kjenner ikke datakvaliteten i eksisterende database, det kan bety problemer når vi skal laste inn data
- Det kan oppstå problemer i et prosjekt vi er avhengig av på en eller annen måte.

Disse listene er nyttige som et utgangspunkt for å identifisere og vurdere risikofaktorer, men vi bør alltid gjøre en selvstendig vurdering av de faktiske risikofaktorene i vårt prosjekt. En idédugnad i prosjektgruppen kan være til hjelp og støtte.

Universitetet i Stavanger

# Sannsynlighet og konsekvenser

Etter at vi har identifisert prosjektets risikofaktorer, må vi gjøre en vurdering av sannsynligheten for at de hendelsene de representerer inntreffer.

Dette kan gjøres ved hjelp av kompliserte statistiske metoder, men det kan også gjøres enkelt og uformelt. En kan oppnå god forståelse ved at prosjektgruppen tar fatt i hver hendelse og forsøker å klargjøre så mye som mulig om

- Når
- Hvorfor
- Hvor

Universitetet i Stavanger

# Sannsynlighet og konsekvenser

Vurderingen dokumenteres og vi graderer hver enkelt risikofaktor ved å gi dem en verdi mellom 0 og 10, eller skalaen lav, middels, høy.

Det som er viktig er at alle som deltar i risikoanalysen har samme forsåelse av den skalaen som brukes.

Når vi har vurdert sannsynligheten for at en hendelse inntreffer, må vi vurdere konsekvensene av hendelsen. Det kan vi gjøre ved å stille oss spørsmål som:

- Er dette en ufrivillig eller frivillig risikofaktor?
- Er dette knyttet til en spesiell type teknologi?
- Hvem blir berørt hvis den inntreffer?
- Er det katastrofalt hvis den inntreffer?

Universitetet i Stavanger

# Sannsynlighet og konsekvens

Disse og lignende spørsmål har som mål å gi oss bedre forståelse av konsekvensene. På samme måte som for sannsynlighetene må vurderingen dokumenteres. Konsekvensene kan kvantifiseres (i krone og øre, tid) dersom det er naturlig.

Etter å ha vurdert sannsynlighet og konsekvenser er vi i stand til å gi hver risikofaktor en prioritet. Den beregnes ut fra en kombinasjon av sannsynligheten for at risikofaktoren inntreffer, og konsekvensene hvis den inntreffer. Risikofaktorene som får høy prioritet, er de som har høy sannsynlighet for å inntreffe, og store konsekvenser hvis de inntreffer.

# Sannsynlighet og konsekvens

| *Sannsynlighet/ Konsekvens* | *Høy* | *Middels* | *Lav* |
|---|---|---|---|
| **Høy** | **Høy** | **Høy** | **Middels** |
| **Middels** | **Høy** | **Middels** | **Lav** |
| **Lav** | **Middels** | **Lav** | **Lav** |

Universitetet i Stavanger

# Sannsynlighet og konsekvens

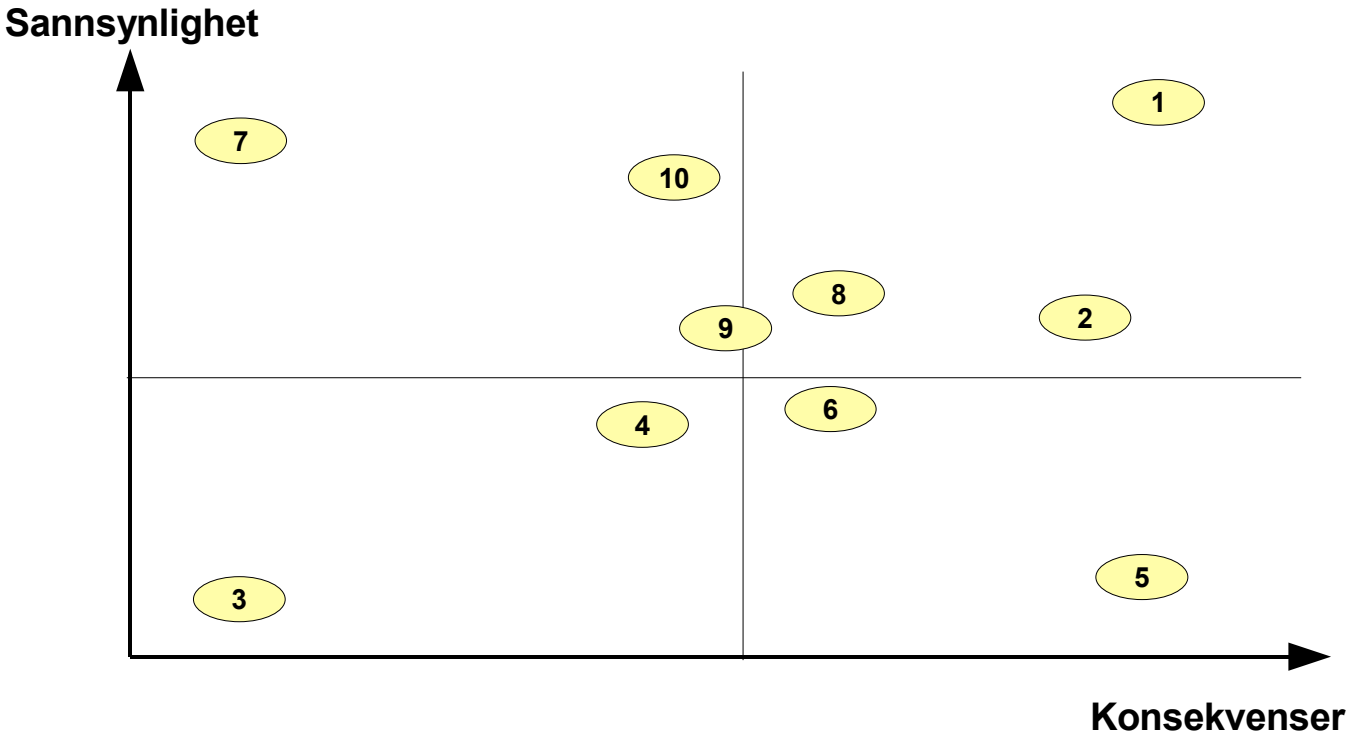
En annen måte er å beregne en risikoindeks, som en får ved å multiplisere sannsynligheten og konsekvensen.

Eksempel:

- Sannsynlighet 8 (sannsynlighet mellom 0 og 10)
- Konsekvens 6 (konsekvens mellom 0 og 10)
- Risikoindeks 8*6 = 48

# Sannsynlighet og konsekvens

En tredje mulighet er å plotte sannsynlighetene og risikofaktorene



Universitetet i Stavanger

# Sannsynlighet og konsekvens

Vi gir prioritet til hver risikofaktor ganske enkelt for å kunne konsentrere innsatsen om de viktigste – de med risikofaktorer som har høy prioritet.

Enden vi bruker avansert statistisk verktøy eller enklere metoder er det viktig for prosjektet at denne type vurdering gjennomføres.

Universitetet i Stavanger

# Tilltak

For hver risikofaktor – og spesielt de med høy prioritet – må vi vurdere hva vi kan gjøre for å hindre at de hendelsene de representerer inntreffer, og hva vi kan gjøre for å minimalisere effekten hvis de skulle inntreffe.

Det er grovt sett tre strategier for å vurdere risiko i prosjektet, og alle må vurderes.

Vi kan fjerne risikoen ved å **endre rammebetingelsene**. Vi kan velge en annen teknologi hvis risikoen er knyttet til teknologien, eller vi kan bemanne prosjektet på en annen måte hvis risikoen er knyttet til prosjektgruppen.

Universitetet i Stavanger

# Tilltak

Vi kan **overføre risikoen**. Vi kan ta inn i kontrakten med en leverandør at vi skal ha kompensasjon hvis leverandøren er for sen med en komponent. Eller vi kan ta innn i kontrakten med oppdragsgiver at prosjektets planer er avhengig av at brukerorganisasjonen stiller de riktige ressursene til disposisjon.

Vi kan a**kseptere risikoen** og sette i verk tiltak for å kontrollere den. Boehm har ikke bare en liste over risikofaktorer, men han har foreslått mulige tiltak

Universitetet i Stavanger

# Tilltak

1. Personell
   - Kun høyt kvalifiserte folk
   - Flere med samme kompetanse
   - Kontraktfesting av ressurser
   - Preallokering av ressurser
2. Tid og kostnad
   - Estimater fra feler kilder
   - Tidsboksing
   - Faste kostnadsrammer
   - Inkrementell utviklingen
   - Gjenbruk av kjente komponenter

Universitetet i Stavanger

# Tilltak

3. Galt system
   - Analyse av arbeidsprosessen og organisasjonen
   - Brukerundersøkelser
   - Prototyping
4. Galt brukergrensesnitt
   - Prototyping
   - Scenarioanalyse
5. «Gold – plating»
   - Kost – nytte analyse
   - Prototyping
   - Kategorisering og prioritering av brukerkrav

Universitetet i Stavanger

# Tilltak

7. Mangler i eksternt utviklede komponenter
   - Sjekk referanser
   - Belønningskontrakter
   - Inspeksjoner
   - Lagbygging
   - Prototyping
8. Mangler i innkjøpte komponenter
   - Benchmarking
   - Inspeksjoner
   - Sjekk av referanser
   - Sammenlignende analyse (mot konkurrenter)

Universitetet i Stavanger

# Tilltak

9.Kapasitetsproblemer

- Simulering
- Benchmarking
- Modellering
- Prototyping
- Tuning

10.Ny teknologi eller teknologi på grensen

- Teknisk analyse
- Kost-nytte
- Prototyping
- Sjekk av refranser

# Tilltak

Når vi har identifisert mulige tiltak for å kontrollere risiko må vi gjøre en ting til. Vi må vurdere hva det vil koste å gjennomføre tiltakene. Og her er en enkel huskeregel.

**Denne regelen gjelder generelt**

Kostnaden forbundet med å hindre at en uønsket hendelse inntreffer, skal ikke overstige de kostnadene som påløper hvis hendelsen inntreffer.

# Dokumentasjon av risiko

Figur vi startet med viser en enkel og oversiktlig måte å dokumentere risikofaktorer på.

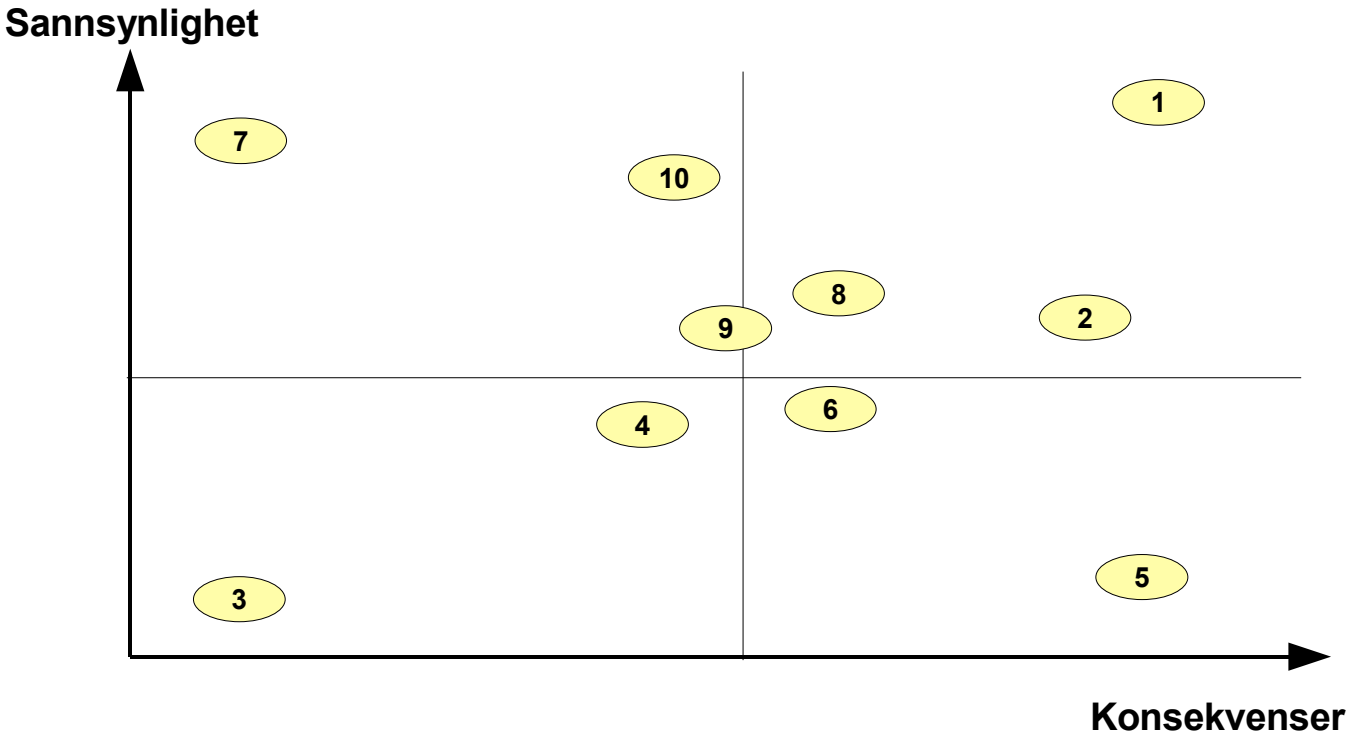
De risikofaktorene vi trenger å bekymre oss for ligger i øvre høyre hjørne, høy sannsynlighet og store konsekvenser. De vi ikke trenger å bekymre oss over ligger nederst i venstre hjørne.

Sammen med dette plottet beskriver vi kort for hver risikofaktor

- Hva ligger i risikofaktorene
- Hva som avgjør om den inntreffer eller ikke
- Hvilke konsekvenser vi forutser
- Hvilke tiltak vi har planlagt for å kontrollere risikoen

# Sannsynlighet og konsekvens

En tredje mulighet er å plotte sannsynlighetene og risikofaktorene



Universitetet i Stavanger

# Oppsummert

En risikoanalyse består av følgende trinn:

1. Identifisering av prosjektet risikofaktorer
2. Identifisering av sannsynligheten for at risikofaktoren inntreffer
3. Identifisering av konsekvensen hvis risikofaktoren inntreffer
4. Prioritering av hver enkelt risikofaktor
5. Identifisering av mulige tiltak for å redusere risikoen
6. Kostnadsvurdering av hvert enkelt tiltak
7. Dokumentasjon av risikoanalysen på en enkel form

En slik enkel risikoanalyse bør gjennomføres med jevne mellomrom i prosjektet. Analysen vil da vise utviklingen av de enkelte faktorene og er et glimrende utgangspunkt for å diskutere viktige forhold i prosjektet mellom prosjektleder og styringskomité. Dette viser at prosjektleder har kontroll.

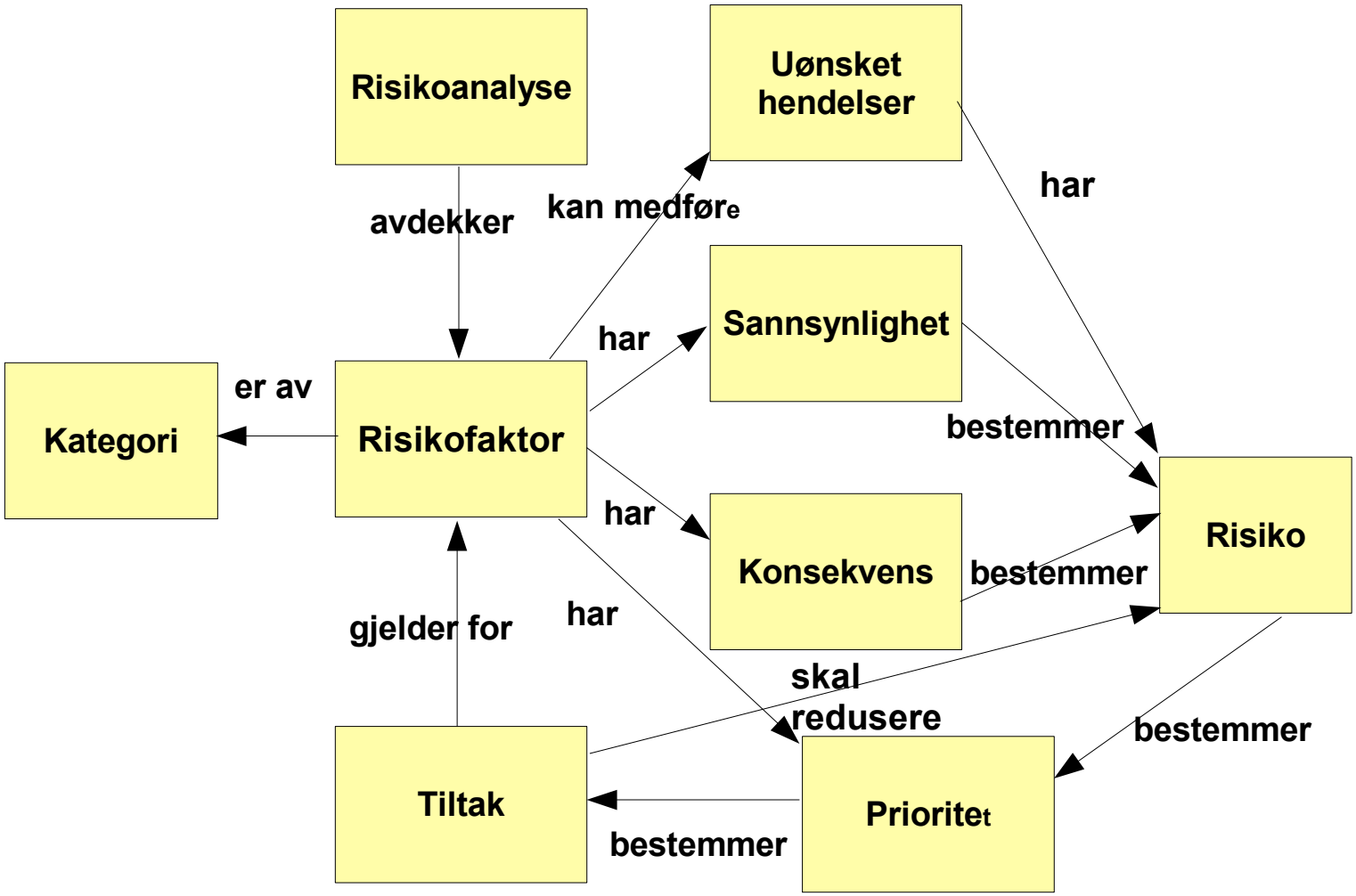
Universitetet i Stavanger

# Oppsummert

Vi har følgende relasjoner

- Risikoanalyse avdekker risikofaktorer
- Risikofaktorer kan medføre uønskede hendelser
- Risikofaktorer har sannsynlighet for å inntreffe, og konsekvenser hvis de inntreffer
- Sannsynlighet og konsekvens gir risiko for risikofaktorene
- Risiko bestemmer prioriteten for risikofaktorene
- Prioriteten bestemmer hvilke tiltak vi vil iverksette
- Tiltakene skal redusere risikoen for en risikofaktor

# Oppsummert


Risikoanalyse
Uønsket hendelser
avdekker
kan medføre
har
Sannsynlighet
Kategori
er av
Risikofaktor
bestemmer
har
Risiko
Konsekvens
bestemmer
har
gjelder for
skal redusere
bestemmer
Tiltak
Prioritet
bestemmer

# Kapittel 18
# Konfigurasjonsstyring

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Konfigurasjonsstyring er å kontrollere endringer

Konfigurasjonsstyring er en helt sentral mekanisme for å styre utviklingen og videreutviklingen av et system.

Sitat:

> «Uten konfigurasjonstyring, eller med dårlig konfigurasjonsstyring, er vi garentert at prosjektet vil bli hjemsøkt av kaos, uforståelige og av og til uopprettelige feil, lav produktivitet og uhåndterlig systemevolusjon»

- Hvorfor trenger vi konfiurasjonsstyring?
- Hva er konfigurasjonsstyring?

Universitetet i Stavanger

# Konfigurasjonsstyring er å kontrollere endringer

*Konfigurasjonstyring er å ha kontroll på endringer endringer i systemet!*

Endringer kommer av at

- Systemet og systemets komponenter er under utviklingen
- Det oppstår feil eller behov for endring etter at systemet er satt i drift
- Det er ulike versjoner av samme system i ulike brukermiljø

Universitetet i Stavanger

# Konfigurasjonsstyring er å kontrollere endringer

På grunn av endringer vil vi ha flere versjoner av systemet og av systemets komponenter. Konfigurasjonskontroll er de prosedyrer, standarder og verktøy som skal sørge for at vi har dette under kontroll. Målet er at vi alltid vet at vi arbeider på riktig versjon av komponenten, og at vi alltid kan gå tilbake til en tidligere versjon hvis det oppstår feil.

Vi vil i denne delen av forelesingene se på:

- Hvorfor konfigurasjonsstyring
- Begreper og mekanismer
- Nødvendige prosedyrer
- Verktøy for konfigurasjonskontroll
- Planlegging av konfugurasjonskontroll

Universitetet i Stavanger

# Mangel på konsjonsstyring skaper problemer

Vi trenger konfigurasjonsstyring fordi systemet endrer seg. Det endrer seg i alle faser av utviklingen og det endrer seg etter at systemet er satt i drift. Hvis vi ikke kontrollerer endringene på en strukturert måte vil vi oppleve problemer som:

- Et kapittel i brukermanualen blir oppdatert og vi går en hissig telefonsamtale fra en kollega som lurer på hvorfor vi har slettet den siste endringen han utførte i dokumentet.
- I en hektisk innspurt forsvinner arbeid vi er sikre på er utført. En feil som er rettet dukker plutselig opp igjen. En ny flott systemegenskap som var testet og med i systemet er der plutselig ikke lenger.
- I en viktig demo viser det seg at flere av skjermbildene opptrer i «gammel versjon»

Universitetet i Stavanger

# Mangel på konsjonsstyring skaper problemer

- Etter å ha brukt dager på å programmere datautvekslingen mellom systemet og databasen, oppdager vi plutselig at noen i mellomtiden har endret datamodellen
- Vi har forvaltningsansvar for et system, men greier ikke å finne årsaken til en feil. Etter mye slit oppdager vi at den logiske datamodellen og den fysiske datamodellen ikke stemmer overens.
- Vi leverer en ny versjon til en kunde, som klager fordi han får en brukermanual som hører til en annen versjon av systemet.
- Mens vi videreutvikler systemet, retter vi en feil i den versjonen som er i drift. Når vi setter i drift den nye versjonen, er den gamle feilen der igjen.

Universitetet i Stavanger

# Mangel på konsjonsstyring skaper problemer

Humphry:

> *«The most frustrating software problems are often caused by poor configuration management. The problems are frustrating because they take tame to fix, they often happen at worst time, and they are totally unnecessary.»*

Universitetet i Stavanger

# Mangel på konsjonsstyring skaper problemer

Humphry:

> *«The most frustrating software problems are often caused by poor configuration management. The problems are frustrating because they take tame to fix, they often happen at worst time, and they are totally unnecessary.»*

# Konfigurasjonstyring styrer endringsarbeidet

Konfigurasjonsstyring skal sørge for at vi mestrer følgende situasjoner

- **Forskjellige personer jobber med ulike deler av systemet – samtidig.**
- **Endring av godkjente systemkomponenter**

Universitetet i Stavanger

# Konfigurasjonstyring styrer endringsarbeidet

- Endringen er fornuftig og ønskelig
- Vi endrer rett versjon
- Vi passer på å endre andre komponenter som berøres av endringen
- Vi koordinerer endringsarbeidet med annet arbeid på systemet
- Vi kan gå tilbake til tidligere versjoner av systemet hvis det oppstår feil etter endringen

**Flere versjoner av et system**

Av og til er det nødvendig å utvikle og vedlikeholde flere versjoner av et system. Da må vi ha flere gyldige versjoner av ulike komponenter av systemet.

Universitetet i Stavanger

# Hvor i livssyklusen trenger vi konfigurasjonsstyring

Konfigurasjonsstyringen starter når prosjektet starter, og den er slutt først når systemet ikke lenger er i drift.

Universitetet i Stavanger

# Mekanismer og begreper i konfiurasjonsstyring

I dette avsnittet beskrives mekanismer som er selve grunnlaget for konfigurasjonsstyringen. I neste avsntt vil vi vise hvordan vi kan bygge opp prosedyrer for endringskontroll og vesjonskontroll basert på disse mekanismene.

Universitetet i Stavanger

# Konfigurasjonsenheter og baselines

Konfigurasjonsenheter og baselines er to sentrale begreper i konfigurasjonsstyringen. Disse begrepene defineres slik i ISO 10007:1995

**Configuration baseline:**

«Configuration of a product, formally established at a spesific point in time, which serves as a reference for futhure activities»

**Configuration item:**

«Aggregation of hardware, software, processed materials, services or any of its discrete portions, that is designated for configuration management and treated as a single entity in a configuration management process»

Universitetet i Stavanger

# Konfigurasjonsenheter og baselines

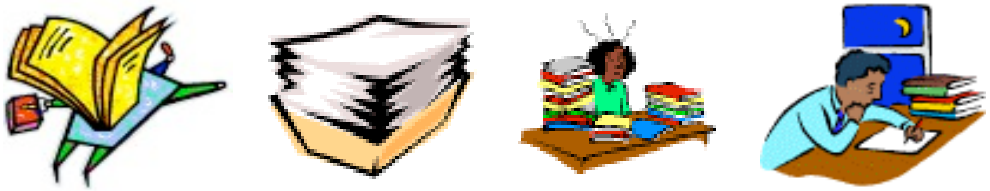
En litt mindre formell definisjon:

- En baseline er en formelt godkjent leveranse fra prosjektet som er grunnlad for videre arbeid i prosjektet
- En konfigurasjonsenhet er en bit av systemet; det kan være programvare, dokument, testopplegg etc. eller samling slik, som vi har bestemt skal underlegges konfigurasjonsstyring, og som vi behandler som en enhet.

Universitetet i Stavanger

# Valg av konfigurasjonsenheter

Å utvikle et stort system betyr å skrive hundrevis av sider med dokumentasjon, utvikle utkast etter utkast til datamoder og funksjonsmodeller, programvare, et stort antall kodemoduler, produsere stabler med planer og revisjoner av planer, utarbeide en rekke med testbeskrivelser og tilhørende testdata.



Universitetet i Stavanger

# Valg av konfigurasjonsenheter

Vi må tidlig i prosjektet ta stilling til hvordan vi vil dele inn disse prosjektresultatene i passe store konfigurasjonsenheter. Vi ønsker en balanse hvor vi ikke for mange og for små konfigurasjonsenheter og heller ikke for få og for store.

Ulempen med mange og ulempen med store og få konfigurasjonsenheter er at arbeidet med dem kan bli lite smidig. Når en person jobber med en stor enhet, låser han en stor del av systemet mot andre endringer. Fordelen er at det er lett å ha oversikten, det er lett å koordinere arbeidet og det medgår færre ressurser til selve konfigurasjonsstyringen.

Vi bør ikke la produkter bli definert som konfigurasjonsenheter hvis det strengt tatt er unødvendig å kontrollere endringen i dem. Det er ikke nødvendig å konfigurasjonsstyre uformelle prosjektdokumenter som skisser, arbeidsnotater, møteinnkallinger etc.

Universitetet i Stavanger

# Kriterier for valg av konfigurasjonsenheter

Når vi bestemmer hvilke konfigurasjonsenheter vi skal ha, må vi vurdere

- For hvilke resultater vil vi kontrollere endringer og versjoner

Noen eksempler prosjektresultater som det er naturlig å konfigurasjonsstyre

- Planer
  - Prosjektplaner, aktivitetsplaner, ressursplaner, testplaner, kvalitets planer etc.
- Dokumenter
  - Kravdokument, designdokument, brukerdokumentasjon, testdokumentasjon
- Testskript og datakvaliteten

Universitetet i Stavanger

# Kriterier for valg av konfigurasjonsenheter

- Basisprogramvare
  - kompilatorer og operativsystem
- Kildekode
- Eksekverbar kode
- Lenkeoppsett
- Databaseskript

Deretter må vi vurder om resultatet skal være en egen konfigurasjonsenhet eller inngå i en større enhet.

Universitetet i Stavanger

# Kriterier for valg av konfigurasjonsenheter

Deretter må vi vurder om resultatet skal være en egen konfigurasjonsenhet eller inngå i en større enhet.

Nyttig regel for å velge egnede konfigurasjonsenheter.

1. Enhver komponent som kan blir endret, er en kandidat til å være egen konfigurasjonsenhet. Vi må vurdere:
   - Hvor stor og kompleks er enheten
   - Hvor ofte vil enheten bli endret, hva er kostnaden med å gjøre endringer
   - Forventet gjenbruk av enheten
   - Er komponenten kritisk med hensyn til sikkerhet/ytelse
   - Er komponenten teknisk vanskelig
   - Hvilke rolle har komponenten i arkitekturen av systemet
   - Er komponenten selvstendig med hensyn til kompilering, installasjon, utførelse

Universitetet i Stavanger

# Kriterier for valg av konfigurasjonsenheter

2. Oppdelingen må være konsistent med systemarkitekturens oppdeling i moduler, og oppdelingen må være konsistent med fordeling av ansvar i prosjektet.
3. Oppdelingen må ta hensyn til hvilke resultater som må kontrolleres sammen fordi de naturlig utgjør en enhet og er avhengig av hverandre. En kodemodul må kontrolleres sammen med dokumentasjonen. Kodemodulen i et delsystem må kanskje kontrolleres sammen med spesifikasjonene, testbeskrivelsen, testdata og dokumentasjonen
4. Ethvert prosjektresultat som har betydning for bruk og forvaltning av systemet, skal inngå i en konfigurasjonsenhet, og ethvert mellomresultat som ikke har betydning for bruk og forvaltning av systemet, bør holdes utenfor konfugurasjonsstyringen.

Universitetet i Stavanger

# Kriterier for valg av konfigurasjonsenheter

Det er vanlig å definere et hierarki av konfigurasjonsenheter i prosjektet. Et produkt som på et stadium i prosjektet er en egen konfigurasjonsenhet, kan på et annet stadium i prosjektet være en del av en større konfigurasjonsenhet.

Kriteriene for valg av konfigurasjonsenheter bestemmes tidlig i prosjektet og beskrives formelt i prosjektets plan for konfigurasjonsstyring.

# Hva betyr det at et resultat er en konfigurasjonsenhet

Et resultat blir en konfigurasjonsenhet dersom:

- Enheten blir entydig identifisert
- Det vedlikeholdes dokumentasjon som viser sammenstningen av konfigurasjonsenheten, relasjoner til andre enheter og endringshistorikk.
- Enheten styres med hensyn til hvilke endringer som kan utføres, hvem som kan utføre endringene og når de kan utføres

I prinsippet kan kun en person jobbe med en konfigurasjonsenhet i et gitt tidspunkt. Dersom flere ønsker å gjøre endringer i en gitt enhet, må sekvensen styres.

Skal en konfigurasjonsenhet endes må den sjekkes ut av systemet og når endringen er utført må den sjekkes inn i systemet.

# Identifikasjon og dokumentasjon av konfigurasjonenheter

Universitetet i Stavanger

For en konfigurasjons enhet skal følgende dokumenteres

**Entydig identifikasjon**

- Det må være mulig å introdusere nye enheter uten at det er nødvendig å endre identifikatoren for de allerede eksisterende enhetene
- Identifikasjonen må kunne identifisere ulike typer produkter – dokumenter, filer datasett, skript, planer, kodemoduler

**Informasjon om relasjoner**

- Lenker sammen konfigurasjons enheter som har en hierarkisk relasjon til hverandre
- Viser hvilke deler en konfigurasjons enhet består av
- Viser hvilke baselinje en konfigurasjons enhet skal inkluderes i

# Identifikasjon og dokumentasjon av konfigurasjonenheter

**Endringshistorikken for konfigurasjonsenheten**

- Hva er endret
- Hvem har endret
- Hvem godkjente endringen
- Når ble enheten endret

Universitetet i Stavanger

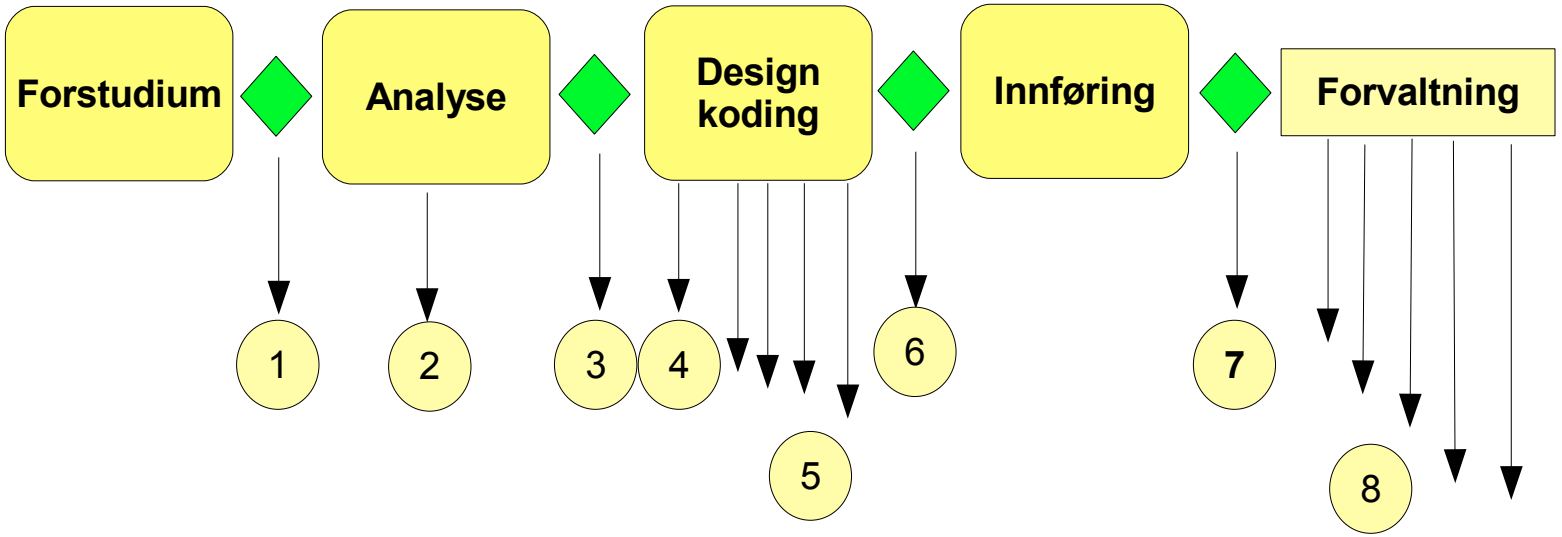
# Hva er baselines

En baseline er et formelt godkjent resultat fra prosjektet. Når vi har etablert en baseline, har vi etablert et referansepunkt, et stabilt, godkjent startpunkt som all videre utvikling tar utgangspunkt i.

En baselinje består av konfigurasjons enheter, av og til bare en, men vanligvis flere. Hvor mange konfigurasjons enheter en baselinje består av, kan være dynamisk. Baseline vokser etter hvert som nye konfigurasjons enheter klargjøres, godkjennes og integreres i baseline. For å kunne integreres i en baseline, må konfigurasjonsenheten gjennomgå en formell godkjenning.

# Typiske baselines i et prosjekt

Hvilke baselines vi har i et prosjekt, avgjøres i stor grad av hvilke livssyklusmodell vi følger. Det er naturlig å ha minst en baselinje for hver fysisk fase i et prosjekt. En baselinje blir en typisk leveranse fra fasen.




Generiske baselines i et prosjekt

Universitetet i Stavanger

# Typiske baselines i et prosjekt

1. Forstudieplanen og tilhørende prosjektplaner for videreføring av prosjektet. Forstudierapporten er ikke alltid underlagt kinfigurasjonsstyring
2. Brukerkrav
3. Systemkrav, inkludert logisk datamodell. Hvis analysen gjennomføres som en fase, kan systemkrav og brukerkrav være intehrert i en felles baseline
4. Systemarkitektur, inkludert fysisk database design
5. Kodemoduler: en eller flere baselines for selve systemet. I de tilfellene vi har flere delsystemer er det naturlig å ha en baseline for hvert delsystem
6. Systemet: En baselinje som omfatter hele systemet ferdig til drift med tilhørende dokumentasjon. Hvis det er naturlig, kan vi ha flere baselines – en for hvert delsystem
7. Systemet etter godkjenningstest. Dette kan være en ny baselinje eller være identisk med 6, avhengig om godkjenningen fører til endringer
8. Flere versjoner av systemet. En baselinje for hver ny versjon av systemet etter at det er satt i drift.

Universitetet i Stavanger

| ***Baseline*** | ***Typiske aktivite-ter*** | ***Formell godkjenning*** | ***Typisk konfigura-sjonsenheter som inngår*** |
|---|---|---|---|
| Bruker-krav | Analyse av bruker-krav | Kravspesifikasjoner; formell godkjenning av krav | Krav til funksjoner, data og egenskaper; utviklingsplan; plan for konfigura-sjonsstyring, kvalitetsplan |
| System-krav | Detaljert krav-analyse; datamodellering; funksjons-modellering | Inspeksjon av system-krav; formell godkjen-ning av systemkrav | Logisk funksjonsmodell; datamodell; spesifikasjon av egenskaper; grensesnitt-beskrivelse; spesifikasjon av manuelle rutiner; godkjente endringer i baseline brukerkrav |
| Arki-tektur | Utforming system-arkitektur; fysisk databasedesign; implementasjons planlegging | Inspeksjon av arkitek-tur; inspeksjon av fysisk databasedesign; formell godkjenning av arkitektur og database | Arkitektur; fysisk data-base; planer for program-meringsfasen; testplaner; godkjente endringer i base-line systemarkitektur |
| Kode | Programmering; testing | Modultest; integrasjonstest; regresjonstest systemtest | Kildekode; kjørbar kode; testplaner; testbeskrivelser; testrapporter; testdata; installasjonsplaner; data for overføring; genererte data-base; basisprogramvare; godkjente endringer i baseline arkitektur |
| System | Integrasjon, utforming bruker dokumentasjon og for-valtningsdokumentasjon | Akseptansetest | Kjørbart system, bruker-dokumentasjon, forvaltnings-dokumentasjon; installert database |

Universitetet i Stavanger

# Dokumentasjon av baseline

For hver baseline må vi dokumentere

- En entydig identifikasjon
- Strukturen og hvilke konfigurasjons enheter som inngår
- Endringshistorikk
- Hvilke prosedyrer som skal følges for å endre og godkjenne endringer i konfigurasjonsenheten i baseline

Universitetet i Stavanger

# Endringer og godkjenning av konfigurasjonsenheter og baselines

En baseline kan endres, og det er normalt at en baseline blir endret. Men enhver endring i en baseline må gjennomgå en formell endrings kontroll, og enhver endring i en baseline fører til en ny versjon av en baseline. Integrering av en godkjent konfigurasjons enhet i en baseline fører til en ny versjon av baseline.

Hvis vi endrer en allerede godkjent konfigurasjons enhet, så får vi en ny versjon av både konfigurasjonsenheten og av den baseline hvor konfigurasjonenheten inngår.

Universitetet i Stavanger

# Endringer og godkjenning av konfigurasjonsenheter og baselines

En baseline kan endres, og det er normalt at en baseline blir endret. Men enhver endring i en baseline må gjennomgå en formell endrings kontroll, og enhver endring i en baseline fører til en ny versjon av en baseline. Integrering av en godkjent konfigurasjons enhet i en baseline fører til en ny versjon av baseline.

Hvis vi endrer en allerede godkjent konfigurasjons enhet, så får vi en ny versjon av både konfigurasjonsenheten og av den baseline hvor konfigurasjonenheten inngår.

Universitetet i Stavanger

# Endringer og godkjenning av konfigurasjonsenheter og baselines

Et eksempel:

Utvikleren tester modulen til den er «feilfri». Modulen lenkes deretter opp sammen med delsystemet. Så gjennomføres en integrasjonstest og regresjonstest av delsystemet. Hvis dette går bra, blir kodemodulen formelt godkjent, og vi får en ny versjon av delsystemet.

I dette tilfellet ser vi ar

- Kodemodulen er en konfigurasjons enhet
- Delsystemet er en baseline
- Formell godkjenning og akseptanse av konfigurasjonsenheten skjer ved at baseline(delsystemet med kodemodulen integrert) gjennomgår integrasjonstest og regresjonstest
- Dette er samtidig en formell godkjenning av den nye versjonen av baseline

Universitetet i Stavanger

# Bibliotek for konfigurasjons enheter og baselines

Vi må lagre konfigurasjons enheter og baselines og dokumentasjonen av dem på en kontrollert og strukturert måte. De bibliotekene vi etablerer for dette, blir sentrale når vi utarbeider rutiner for konfigurasjons styring.

Det er vanlig å ha flere biblioteker for konfigurasjons enheter og baselines i ulike stadier av utviklingen. Hvor mange, og hvilke biblioteker vi etablerer, varierer fra prosjekt til prosjekt og fra bedrift til bedrift. Det er mange faktorer som spiller inn. Ulike bedrifter har ulik standard. Prosjekter er forskjellige og har ulikt behov for kontroll.

Universitetet i Stavanger

# Bibliotek for konfigurasjons enheter og baselines

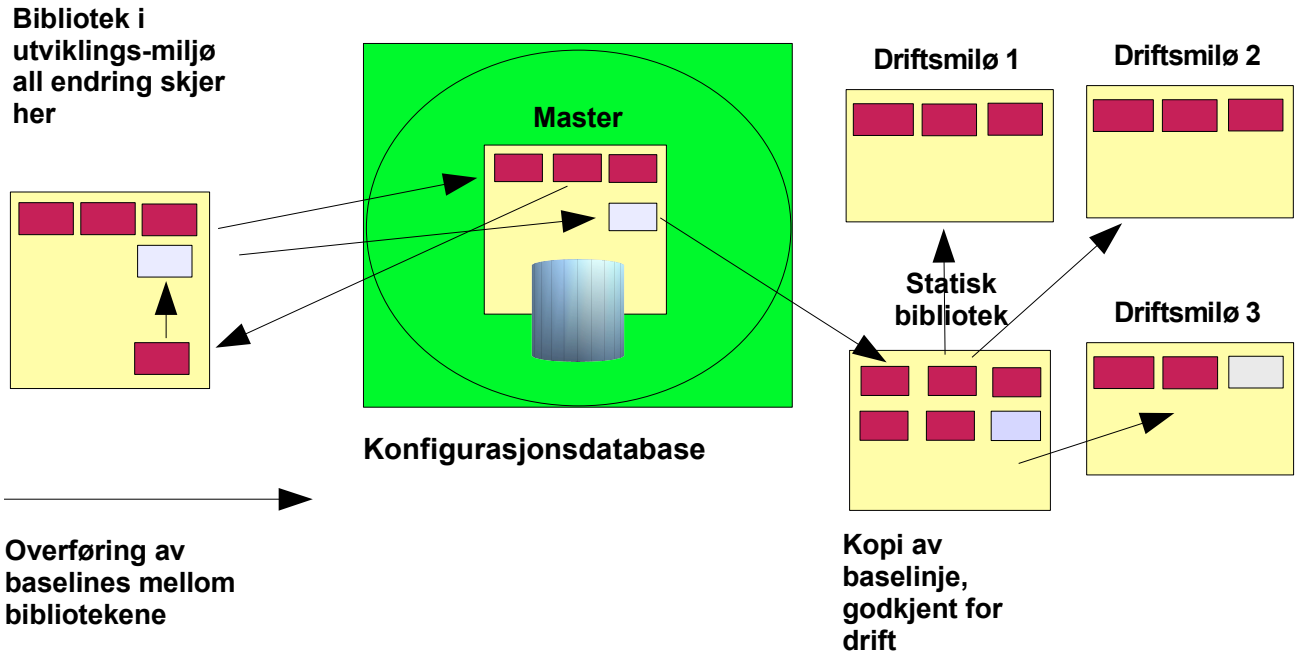
Det er imidlertid vanlig med fem typer bibliotek, I de fleste bedrifter som har innført konfigurasjons styring, finner vi en eller annen form for:

- Ett eller flere bibliotek i utviklingsmiljøet
- Det kontrollerte biblioteket – Master
- Konfigurasjonstatabasen
- Det statiske biblioteket
- Driftsmiljøer

# Bibliotek for konfigurasjons enheter og baselines

Vi vil se nærmere på disse bibliotekene, hva de inneholder, og hvilke regler som gjelder.

Neste figur viser en skisse over typiske bibliotek som inngår i konfigurasjonstyringen. Hvert bibliotek omtales nedenfor.



Universitetet i Stavanger

# Bibliotek i utviklingsmiljø

Utviklingsmiljøet er arbeidsplassen til den enkelte systemutvikler. Alle arbeidskopiene av de konfigurasjonsenhetene som er under utvikling, finner her. I dette miljøet utvikles og testes de ulike modulene helt til de er klare til å godkjennes og integreres i baseline.

Når vi skal endre en konfigurasjonsenhet som er akseptert og integrert i baseline, hentes de fra det kontrollerte biblioteket og over i utviklingsbiblioteket.

Universitetet i Stavanger

# Det kontrollerte biblioteket - Master

I det kontrollerte biblioteket lagres alle baselinjer, og dermed alle konfigurasjonsenheter som er godkjent og akseptert som baseline.

I dette biblioteket lagres alle versjoner av baseline. Det gjelder alle gyldige versjoner og alle tidligere versjoner. Med utgangspunkt i dette biblioteket kan vi dermed reetablere en hvilke som helst versjon av systemet på et hvile som helst stadium i utviklingen.

Når en konfigurasjonsenhet skal integreres og aksepteres i en baselinje, vil den bli overført til dette biblioteket. Når det skal gjøres endringer i en konfigurasjonsenhet som er integrert og akseptert i en baseline, må den sjekkes ut fra dette biblioteket.

Universitetet i Stavanger

# Det kontrollerte biblioteket - Master

Når en konfigurasjonsenhet er endret, må den sjekkes inn, testes og aksepteres på nytt. Da lagres en ny versjon av konfigurasjonsenheten. Det medfører at det også lagres en ny versjon av den baselinje som konfigurasjonsenheten inngår i.

Det kontrollerte biblioteket må ha flere nivå av adgangskontroll med mulighet for å spesifisere restriksjoner på hvem som får lese hva, og med sterkt begrenset tilgang til lagring og sletting. Det må også ha funksjoner for lagring, fremhenting og sletting av ulike versjoner av baseline og konfigurasjonsenheter. Og det er viktig at det er funksjonalitet for låsing av konfigurasjonsenheter for endring og/eller for koordinering og samtidig endring i en konfigurasjonsenhet.

Universitetet i Stavanger

# Konfigurasjonsdatabasen

Dette er databasen som inneholder all relevant informasjon om konfigurasjonsenhetene og baseliner. Denne informasjonen benyttes til

- Å vurdere konsekvensene av endring
- Å skaffe oversikt over status på ulike konfigurasjonsenheter og baseliner
- Å skaffe statisk informasjon om endringshistorikk, ressursbruk og lignende
- Å svare på spørsmål angående systemets konfigurasjon, for eksempel
  - Hvilke kunder har hvilke versjon av systemet
  - Hva slags infrastruktur er nødvendig for en bestemt versjon av systemet
  - Hvor mange versjoner av systemet er etablert, og når ble de etablert
  - Hvilke systemversjon må endres hvis en bestemt konfigurasjonsenhet endres
  - Hvor mange endringskrav som står igjen for en gitt versjon
  - Hvor mange rapporterte feil eksisterer for en gitt versjon

Universitetet i Stavanger

# Konfigurasjonsdatabasen

Denne databasen bor ideelt integreres i det kontrollerte biblioteket og en del verktøy har funksjonalitet for dette. Med en integrasjon får vi støtte for å kontrollere sammenhengen mellom informasjon og status for konfigurasjonsenheten. Og vi kan finne sammenhenger mellom ulike konfigurasjonsenheter når det skal gjøres endringer. Det er imidlertid fullt mulig å ha konfigurasjonstyring selv om vi ikke benytter avansert verktøy. Da må vi vedlikeholde informasjonen i en egen database.

Universitetet i Stavanger

# Statisk bibliotek

I det statiske biblioteket lagres baselines som er klar til bruk og frigitt for bruk. Hovedkopien av de systemversjonene som er i drift finnes her.

Universitetet i Stavanger

# Driftsmiljøet

I driftsmiljøet finner vi systemet i daglig drift. Et system kan være kopiert til mange driftsmiljø. Og det finnes mange versjoner av systemet i ulike driftsmiljø.

Universitetet i Stavanger

# Ansvarsforhold

I forbindelse med konfigurasjonsstyring er det viktig å ha klare ansvarsforhold.

**Ansvar for endringskontroll**

- Den som har ansvar for endringskontrollen, er ansvarlig for å vurdere og godkjenne om foreslåtte endringer i en baseline skal gjennomføres. Dette kan være en person eller en gruppe personer. Det avhenger av prosjektfasen og hvor stort prosjektet er, og muligens andre forhold. Tidlig i prosjektet er det kanskje prosjektleder som har ansvaret. I programmeringsfasen i et stort prosjekt kan det være flere grupper som er ansvarlige for hver sine delsystem.

Universitetet i Stavanger

# Ansvarsforhold

**Konfigurasjonsansvar**

- Den som har Konfigurasjons ansvar, har det overordnede ansvar for konfigurasjons- styringen i prosjektet. Vedkommende har spesifikt ansvar for integrasjon og formell godkjenning av alle endringer og baseliner. Vedkommende har også ansvar for alle inn – og utsjekkinger av konfigurasjons enheter fra det kontrollerte biblioteket. I et lite prosjekt er det kanskje prosjektleder som har dette ansvaret. I et stort prosjekt er oppgaven så stor at det må utpekes en egen person.

**Endringsansvar**

- Når en endring er godkjent, utpekes det en ansvarlig for gjenomføringen. Det er den som er ansvarlig for endringskontrollen, som velger hvem som skal være ansvarlig.

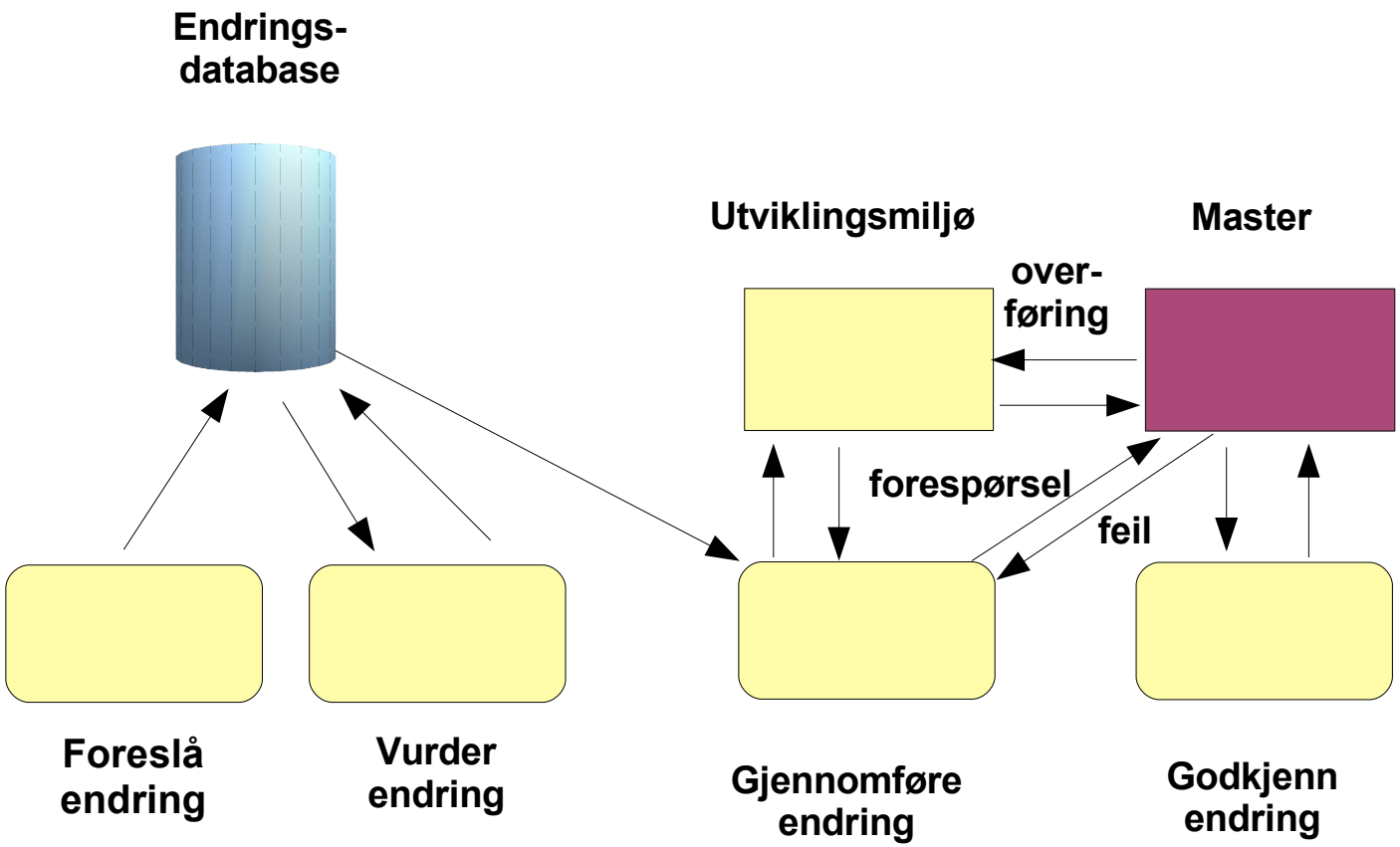
Universitetet i Stavanger

# Prosedyre for endringskontroll

Endring under konfigurasjonstyring betyr endring i baseliner. Det vil si i godkjent versjon av en konfigurasjonsenhet. Figuren under viser hovedtrekkene i behandling av endringer i baselines. Nedenfor beskrives en generell prosedyre for hvordan dette kan gjøres. Den eksakte utformingen av prosedyren varierer fra bedrift til bedrift og fra prosjekt til prosjekt. Krav til formalisme avhenger av prosjektets størrelse og kompleksitet. Endringer og feilmeldinger behandles i prinsippet likt.

Universitetet i Stavanger

# Prosedyre for endringskontroll



Universitetet i Stavanger

# Trinn1: Foreslå endring

Det etableres en database for å lagre feilmeldinger og endringsforslag, og for å kunne følge med i behandlingen av dem. Det etableres et standardformular for å melde inn feil eller for å foreslå en endring.

Et endringsforslag eller feilmelding kan ha mange årsaker.

- Det oppstår problemer med et system i drift
- Brukerne ønsker å forbedre en funksjonalitet
- Det oppdages feil eller mangler

Enhver som oppdager feil eller mangler må kunne fylle ut og sende inn en feilmelding/endringsforslag.

Universitetet i Stavanger

# Trinn1 : Foreslå endring

Et ferdig utfylt formular er et prosjektdokument. Formularet må inneholde tilstrekkelig informasjon til at forslaget kan evalueres. Et minimum vil være:

- Navn på den som har meldt inn forslaget
- Forslag til hva som skal endres, og hvilke konfigurasjonsenhet(er) det skal endres i
- Vurdering av hvor kritisk endringen er
- Eventuelt en referanse til utdypende dokumentasjon, f.eks feilrapport

# Trinn2 : Behandling av endringsforslag

Et hvert forslag må få en formell vurdering som enten fører til at forslaget blir avvist, eller at forslaget blir godkjent. Et forslag som er godkjent for gjennomføring, vil bli prioritert og overlatt til en endringsansvarlig for gjennomføring.

Det er ansvarlig for endringskontroll som vurderer og godkjenner forslagene. I vurderingen må en svare på spørsmålene

- Er endringen nødvendig
- Er konsekvensene akseptable
- Er endringen tilstrekkelig dokumentert
- Er planen for gjennomføring tilstrekkelig
- Når skal endringen gjennomføres

Universitetet i Stavanger

# Trinn2 : Behandling av endringsforslag

**Klassifisering**

- Klassifiseringen skal være til støtte i prioriteringen av endringsforslaget. Klassifiseringen kan være avhengig ag
  - Viktighet
  - Type endring
  - Type konfigurasjonsenhet som skal endres
  - Nødvendig forarbeid
  - Andre forhold som må vurderes som relevante av prosjektet

Universitetet i Stavanger

# Trinn2 : Behandling av endringsforslag

**Vurdering av konsekvensene av endringen**

- Planer, kontrakter, kostnad
- Systemregnskaper
- Metode og verktøy som er i bruk
- Grensesnitt mot andre systemer

Resultatet av alt dette kan bli at endringen godkjennes, avvises eller sendes tilbake for bearbeiding. Hvis den blir godkjent, utpekes en ansvarlig. Endringen får en prioritet, og det blir gitt en ramme for å gjennomføre endringen. Endringsforslaget oppdateres, slik at det viser resultatet av vurderingen.

Universitetet i Stavanger

# Trinn3 : Gjennomføring av endring

For å gjennomføre endringen må endringsansvarlig få overført den aktuelle konfigurasjons-enheten(ene) fra Master til utviklingsmiljøet. Anmodning om dette går til kofigurasjonsansvarlig.

Etter at endringen er gjennomført, blir resultatet testet i utviklingsmiljøet til endringsansvarlig. Når endringsansvarlig er fornøyd med testene blir kofigurasjonsenheten(ene) overlatt til konfigurasjonsansvarlig og blir overført til Master for integrasjon og godkjenning.

Universitetet i Stavanger

# Trinn4 : Godkjenning

For enhver konfigurasjonsenhet og enhver baseline skal det være etablert en formelle prosedyre og kriterier for godkjenning. Den konkrete utformingen er avhengig av hva slags konfigurasjonsenhet og hva slags baseline det dreier seg om.

Mår konfigurasjonsenheten er en kodemodul og den skal integreres i en baseline som er et delsystem, vil den formelle godkjenningen typisk bestå av **integrasjonstest** og en **regresjonstest**.

Når konfigurasjonsenheten er et dokument, kan den formelle godkjenningen bestå av kommentarrunde etterfulgt av et formelt godkjenningsmøte.

**Konfigurasjonsansvarlig** er ansvarlig for godkjenningen. En konfigurasjonsenhet som ikke er godkjennes, tilbakeføres til utviklingsmiljøet for utbedring.

Universitetet i Stavanger

# Rutiner for integrasjon av endringer i baseline

Det må utvikles rutiner for når, og på hvilke måte, en nyutviklet (eller endret) konfigurasjonsenhet skal overleveres til konfigurasjonsansvarlig for integrasjon og godkjenning.

Det er ulike strategier for dette. Hvis utviklingen følger en tradisjonell, sekvensiell livsyklusmodell (fossefalsmodellen), er det vanlig at konfigurasjonsenheten overføres når de er ferdig utviklet og testet. Det skjer gjerne mot slutten av en fase. Alt som er produsert i fasen, overleveres fra utviklerne til konfigurasjonsansvarlig, integreres, testes og godkjennes nærmest som en samlet enhet.

Universitetet i Stavanger

# Rutiner for integrasjon av endringer i baseline

Hvis utviklingen følger en evolusjonær modell med inkrementelle leveranser, kan det være hensiktsmessig å gjøre det på en annen måte. Noen utviklingsmiljø praktiserer hyppige, kanskje daglige, integrasjoner av nye leveranser (versjoner av) konfigurasjons enheter. Alle utviklerne leverer det de har klart. De nye enhetene integreres i baseline, og testes. Alle enheter som ikke består testen, returneres til utviklerene. De andre blir del av den nye versjonen av baseline. Leveransen vokser da gradvis fram gjennom hele perioden.

Fordelen med denne måten å gjøre det på, er at baseline hele tiden viser virkelig status for prosjektet, og at ingen jobber for lenge med feil utgangspunkt. Vi får tidlig testet hvordan komponentene virker sammen, og kan dermed tidlig oppdage problemer.

Universitetet i Stavanger

# Versjonskontroll ved idriftsetting

Versjonskontroll betyr kontroll av at vi setter de riktige versjonene av systemet i drift i riktig driftsmiljø.

Vi må samarbeide med brukermiljøet for å planlegge når en ny versjon skal settes i drift. Vi må ta hensyn til to forhold

- Brukerne ønsker et stabilt miljø, uten for mange endringer. Det betyr at det er ikke ønskelig å skifte versjon for ofte.
- De ønsker at feil og mangler rettes/fjernes så raskt som mulig. Det betyr at vi må sette i drift en ny versjon så snart den er ferdig.

Planleggingen er kort og godt å vurdere disse faktorene opp mot hverandre, og finne ut hva som er best for brukerne.

Universitetet i Stavanger

# Versjonskontroll ved idridtsetting

Det neste er at vi må bygge en riktig versjon av systemet

Dette er en versjon som stemmer med det aktuelle driftsmiljøet og som inneholder riktig versjon av alle komponentene til systemet. En ny systemversjon er mye mer enn bare eksekverbar kode. Vi skal levere

- Datafiler
- Installasjonsprogram
- Systemdokumentasjon
- etc.

Universitetet i Stavanger

# Versjonskontroll ved idridtsetting

Alle disse komponentene må ha rett versjon. Konfigurasjonsansvarlig har ansvaret. Prosedyren som følges er:

1. Alle komponenter som utgjør den nye versjonen, må identifiseres og samles. Dette er:
   - Kodemoduler
   - Datafiler
   - Dokumenter
   - Installasjonskript
   - Manualer
   - etc
2. Den nye versjonen må dokumenters. Hensikten er at akkurat denne versjonen må kunne gjenskapes i fremtiden. Det er derfor nødvendig å dokumentere nøyaktig identifikasjon, versjonsnummer etc. for alle komponentene som inngår, hvor den er satt i drift og når den er satt i drift

Universitetet i Stavanger

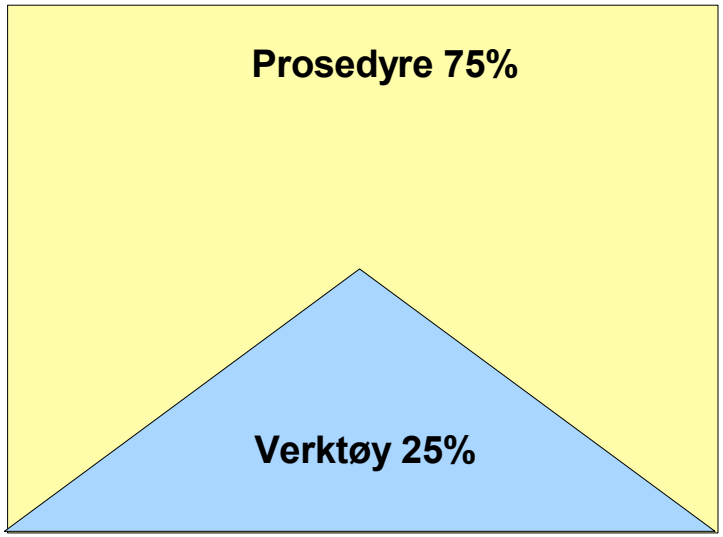
# Versjonskontroll ved idridtsetting

3. Systemet bygges. Dette er prosessen med å kompilere og å lenke alle kodemodulene og gjøre dem om til et eksekverbart program som kan kjøres i et spesielt driftsmiljø. T denne prosessen må vi vite
   - At vi har riktig versjon av alle komponentene
   - At alle datafiler er tilgjengelige, og at referansene til datafilene i programmene er de samme som er brukt i driftsmiljøet
   - At driftsmiljøet har rett versjon av alle basisprogrammene

Det er vanlig å bruke verktøy i systembyggingen. Konfigurasjonsansvarlig utarbeider da et script som identifiserer de komponentene som skal inngå i systemet, så gjør verktøyet resten.

# Verktøy for konfigurasjons styring

Konfigurasjons styring er så komplekst og så fullt av detaljer som skal holdes styr på at det er nærmest umulig å klare seg uten verktøystøtte. Men vi må være klar over at verktøyet skal støtte de prosedyrene vi legger på plass – et verktøy kan aldri gjøre jobben for oss. Forholdet mellom prosedyrer og verktøy han beskrives som 75% prosedyre og 25% verktøy.




Figuren understreker hvor viktig det er å legge på plass gode arbeidsrutiner, retningslinjer og standarder for konfigurasjons styringen. Uten dette er verktøy til liten nytte. Men vi trenger verktøy til støtte for rutinene. Det finnes en rekke verktøy på markedet. Ikke alle har funksjonalitet for alle aspekter av konfigurasjons styringen. Det kan av og til være nødvendig å benytte flere verktøy i kombinasjon

Universitetet i Stavanger

# Planlegging av konfigurasjons styring

I avsnittene foran vi beskrevet de generelle mekanismene og prosedyrene som konfigurasjons styringen bygger på. I dette avsnittet skal vi se nærmere på planen for konfigurasjons styring.

Når vi planlegger konfigurasjons styringen er det viktig at alle parter er enige om ambisjonsnivået. Vi må finne svar på hva konfigurasjonsstyringe skal omfatte og hva som er målet. Med alle parter menes

- Prosjektleder
- Den ansvarlige for konfigurasjons styringen
- Den ansvarlige for systemarkitekturen
- Eventuelle kunder

Universitetet i Stavanger

# Planlegging av konfigurasjons styring

Deretter utarbeides det en konkret plan for konfigurasjons styringen. Denne planen skal beskrive

- Ansvarsforhold
- Standarder
- Prosedyrer

Planen skreddersy konfigurasjons styringen til prosjektets gjennomføringsplan (livssyklus- modell), og de kravene og rammebetingelsene som er beskrevet i kontrakten.

Konfigurasjons styringen starter når prosjektet starter, den slutter når systemet ikke lenger er i drift. Planen utarbeides helt i starten av prosjektet, og den revideres for hver ny fase i prosjektet.

# Innhold i en plan for konfigurasjons styring

Når vi lager en plan for konfigurasjons styring bør ta utgangspunkt i en standard som definerer hva planen skal inneholde.

Et eksempel på en slik standard er: *IEEE 828 – 1998, IEEE Standard for Configuration Management Plans*

En annen standard er ISO 10007. I ISO 1007 anbefales det at planen i størst mulig grad refererer til organisasjonene generelle standarder og prosedyrer, slik at planen blir så enkel som mulig, og slik ar man unngår å dublisere beskrivelser som allerede finnes andre steder.

# Innhold i en plan for konfigurasjons styring

ISO10007 anbefaler følgende kapitler i en plan konfigurasjons styring.

1. Introduksjon og oversikt
   - Beskrivelse av systemet
   - En plan som viser de viktigste hendelsene knyttet til konfigurasjons styringen
   - Hensikten og omfanget av konfigurasjons styringen
   - Referanse til andre aktuelle dokumenter
2. Prinsipper og prosedyrer i konfigurasjons styringen
   - Overordnede prinsipper for konfigurasjons styringen
   - Ansvarsforhold som angår konfigurasjons styringen
   - Kriterier for og utvelging av konfigurasjons enheter
   - Avtalt rapportering angående konfigurasjons styringen
   - Terminologi

# Innhold i en plan for konfigurasjons styring

3. Identifikasjon av konfigurasjons enheter
   - En hierarkisk oversikt over hvilke enheter en har
   - Hvilke nummereringskonvensjon som skal benyttes
   - En oversikt over hvilke baseline som skal etableres, og når de skal etableres
   - Prosedyrer for godkjenning
4. Endringskontroll
   - Beskrivelse av hvordan endringskontrollen er organisert, hvem som er ansvarlig, hvem de rapporterer til
   - Prosedyre for behandling av endringsforslag helt fram til at endringen er integrert og godkjent i en baseline

# Innhold i en plan for konfigurasjons styring

5. Statusrapportering
   - Prosedyre for å registrer og å rapportere alle nødvendige data angående status for konfigurasjons styring
   - Definisjon av alle data som skal inngå i statusrapportering fra konfigurasjonstyringen
6. Kontroll av konfigurasjons styringen
   - En oversikt over hvilke kontroller som skal gjennomføres av konfigurasjonsstyringsaktivitetene og resultatene
   - Prosedyrene som skal følges, hvem som skal delta, og hvem som ahar myndighet
   - Beskrivelse av hvilke rapporter som skal utarbeides

# Kapittel 19
# Validering og verifisering

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# V & V – for å sikre at vi lager riktig system

- Verifisering og validering er prosessen for å bestemme om kravene til et system eller en komponent er komplette og korrekte
- Om produktet fra hver fase tilfredsstiller kravene eller betingelsene pålagt fra forrige fase
- Om det ferdige systemet eller komponenten oppfører seg i samsvar med de spesifiserte kravene.
- Verifisering og validering foregår gjennom hele utviklingsprosessen.
- Validering skal gi svar på om vi utvikler rett system,
- Verifisering skal gi svar på om man utvikler systemet riktig.

Universitetet i Stavanger

# V & V – for å sikre at vi lager riktig system

- Vi kan dele V & V aktivitetene i to hovedgrupper
  - Testing
  - Gjennomgåelse (revisjon)

- Hensikten med testing er å avdekke feil og mangler ved det produktet man lager.
- Programsystemet eller deler av det kjøres, med et sett testdata og en observerer om programmet oppfører seg som man forventer.
- Man sier derfor at en test er dynamisk.

Universitetet i Stavanger

# V & V – for å sikre at vi lager riktig system

- Gjennomgåelse er en form for statisk testing av programmet.
- En prøver å finne feil i programmet uten å teste.
- En kollega lese gjennom koden, en gjøre det selv.
- Andre former for gjennomgåelse er mer formelle og strukturerte i sitt opplegg.
  - Kodeinspeksjon
- Alle produkter som lages, kan underkastes gjennomgåelse.
- Det betyr at man kan begynne å lete etter feil og årsaker til feil før man har et kjørbart program.
- Dess tidligere i utviklingsprosessen feil og mangler finnes, jo lettere og billigere er de å rette.

Universitetet i Stavanger

# V & V – for å sikre at vi lager riktig system

- Hvilke V & V aktiviteter som gjennomføres,er knyttet til den utviklingsprosessen som følges.
- I fossefallsmodellen er poenget at hver fase gjøres ferdig før neste starter.
  - Produktene fra en fase må derfor være godkjent før oppstart av neste fase.
- En slik godkjenning krever at produktene er verifisert og validert.
- I de tidlige fasene, før en har et kjørbart program, benyttes forskjellige former for gjennomgåelse
- Når et kjørbart program foreligger, gjennomføres tester som grunnlag for godkjenning.

Universitetet i Stavanger

# V & V – for å sikre at vi lager riktig system

- Unified Prosess (UP) og eXtreme Ptogramming, interaktiv og inkrementell utvikling.
- Poenget å få til noe kjørbart så fort som mulig.
  - Rask tilbakemelding fra brukerne.
- Risikoen for at brukerne får noe de ikke ønsker reduseres.
- Testing blir derfor gjennomført under hele prosessen.
- I XP er dette en av de tingene som blir drevet til det ekstreme.
  - En gjør fortløpende integrasjon og testing (Se kapittel 15)

Universitetet i Stavanger

# V & V – for å sikre at vi lager riktig system

Vi vil se på ulike V & V aktiviteter.

- V & V aktivitetene behandles uavhengig av valg av utviklings-prosess.
- For testing er det naturlig å starte med enhetstest
- Deretter se på tester som legges til etter hvert som systemet bygges sammen.
- Følger en fossefallsmodellen, kan en si at man følger et lineært løp
- UP og XP legger opp til et gjentakende sirkulært løp.
- Uavhengig av prosess gjør man de samme aktivitetene om enn med ulik vekt.

Universitetet i Stavanger

# Testing

Dynamisk testing.

- *(Software Engineering. Theory and practice (Pfleeger 1998) og Software Engineering (Sommerville 2001))*
- Hensikten med testing er å finne så mange feil før systemet tas i bruk.
- Mange tror at hensikten med testing er å vise at programmet er feilfritt, men det er det umulig å få til ved testing.
- Testing kan bare avdekke feil

Skal testing vise at programmet er rett må en

- Kjøre programmet på alle mulige datasett
- Dessuten må vi vite resultatet som programmet skal

Universitetet i Stavanger

# Testing

- Produktet som kundene får, må oppleves som nyttig.
- Testing er et nødvendig middel for å sikre kvaliteten på systemet. Men kvaliteten må bygges inn i programmet gjennom skikkelig arbeid med
  - definisjon av krav, og bruk av sunne prinsipp og anerkjent praksis gjennom analyse, design og implementering.
- Testing kan ikke erstatte dette
- Testing tar tid og tid er penger.
- Planleggingen av testingen er derfor viktig.
- En må ha tester som med stor sannsynlighet finner de fleste og vesentligste feilene

Universitetet i Stavanger

# Testing

- Ikke lag for få tester
  - Programmet har mange feil som kunden må slite med
- Ikke lage feil tester
  - Finner de feilene og det fører til frustrerte kunder
- Ikke lage for mange tester
  - En unødvendig arbeid
- En ikke finner alle typer feil gjennom **dynamisk testing**.
  - systemet som skal utprøves kan kjøres.
- Det betyr at testingen først starter etter at en har begynt å implementere.
- Mange feil kan være introdusert gjennom spesifikasjons og designfasen.
- Oppdages en feil seint, er den dyr å rette.

Universitetet i Stavanger

# Testing

- Dynamisk testing går betyr programmet arbeide med data og vi observere resultatet.
- Blir resultatet som forventet, sier vi at programmet har passert testen
- Testing har i tre trinn
  1. Planlegging
  2. Finne et sett med testdata
  3. Gjennomføre testen og evaluere resultatet
- Sentralt er valg av testdata.
- Poenget er å avdekke feil
- Vi må derfor være grundige i valg av testdata.

Universitetet i Stavanger

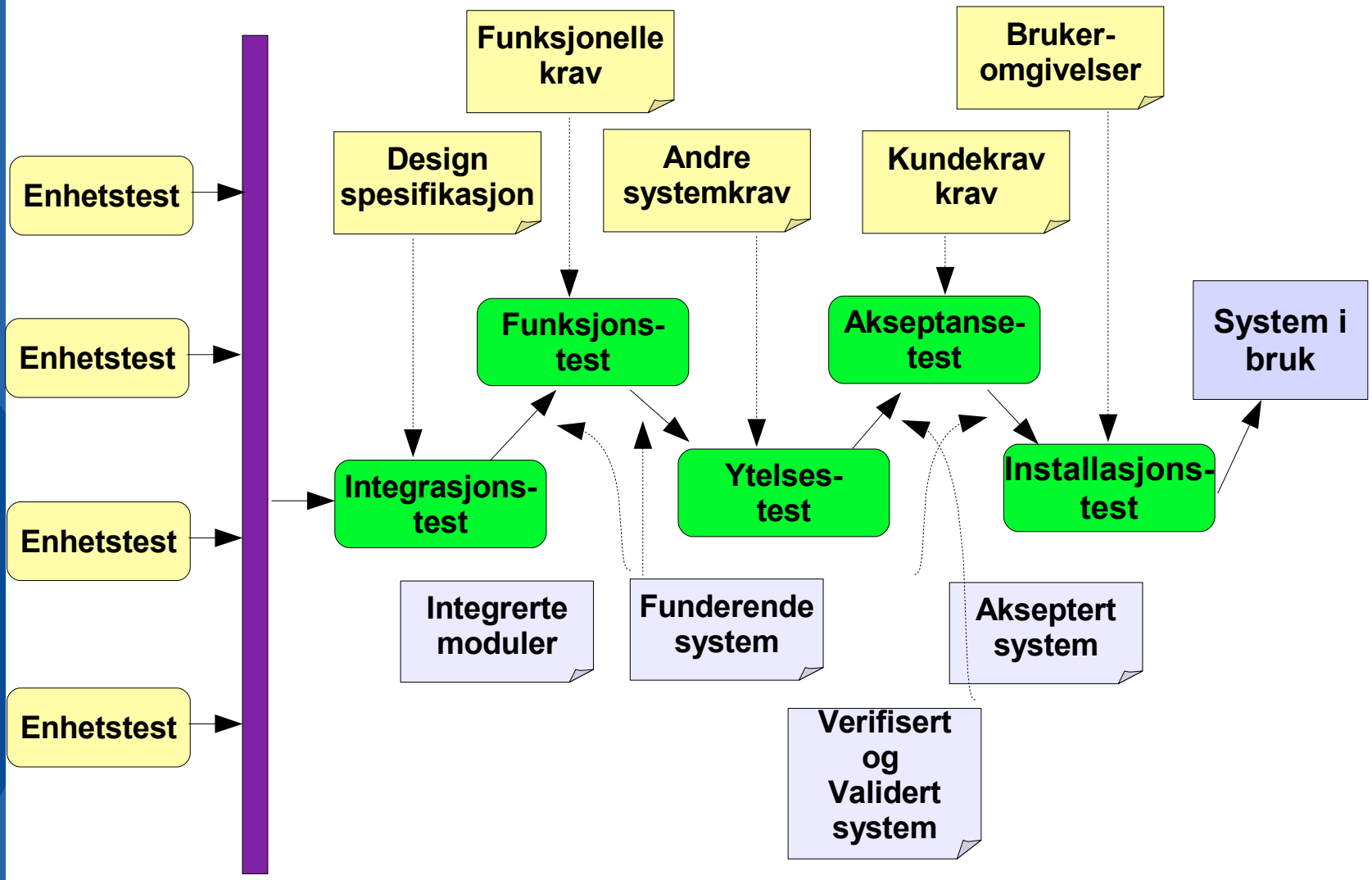
# Ulike typer tester

Det er mange typer tester. Tester kan karakteriseres på ulikt vis. En måte er å se på trinnene programmet gjennomløper. Da støter vi på begrepene

- Enhetstest
- Integrasjonstest
- Systemtest
- Akseptansetest

Universitetet i Stavanger

# Ulike typer tester



Universitetet i Stavanger

# Ulike typer tester

- Figuren over er ikke en beskrivelse av fossefalsmodellen.
- Figuren viser en rekkefølge av testaktiviteter.
- Denne gjennomløpes en gang eller flere ganger, etter hvilke utviklingsprosess som følges.
  - Vi starter med å teste de enkelte modulene.
  - Deretter integrasjons-tester.
  - Forskjellige typer av systemtester følger deretter.
  - Er systemene store, følger vi de samme trinnene og bygger opp subsystemer og tester disse.
  - Subsystemene sammen til et totalsystem.

Universitetet i Stavanger

# Ulike typer tester

En avklaring av begrep

- **Enhet**

  Det kan være en prosedyre eller en klasse, men det kan også være en modul eller en gruppe moduler

- **Modul**

  En samling av enheter som naturlig hører sammen. Modulene utgjør byggeblokkene i systemet. En modul ligger gjerne på en fil og kompileres separat. Man kan også finne begrepet komponent brukt.

- **System**

  Den totale samlingen av moduler som sammen utfører de funksjoner som vi vil at programsystemet skal utføre. Store systemer er flærfunksjons-systemer og er bygd opp av subsystemer

- **Subsystem**

  En samling moduler som utfører en eller et begrenset antall funksjoner

Universitetet i Stavanger

# Ulike typer tester

Andre aktuelle begreper er:

- Alfatester
- Betatester
- Regresjonstester
- Ytelsestester
- Etc.

Tester kan også karakteriseres etter den strategien som velges for testen

- Svart boks testing
- Hvit boks testing
- Ovenfra og ned
- Nedenfra og opp

Universitetet i Stavanger

# Enhetstester

- Testen vi gjennomfører etter at vi har kodet ferdig en bit av programmet
- Vi snakker i den forbindelse om to typer tester
  - Svart boks testing
  - Hvit boks testing
- Svart boks er kjent fra fysikken, og brukes når man betrakter et system fra utsiden.

  Ved hvitbokstesting har vi full kjennskap til den interne strukturen

Universitetet i Stavanger

# Hvitbokstesting

- Ved hvitbokstesting har vi full kjennskap til den interne strukturen.
- I prinsippet har vi full oversikt over alle kombinasjoner og veier i programmet.
- Vi kan dermed lage et testsett og gjennomføre tester slik at programmet gjennomløper alle disse veiene.
- La oss se på et eksempel

Universitetet i Stavanger

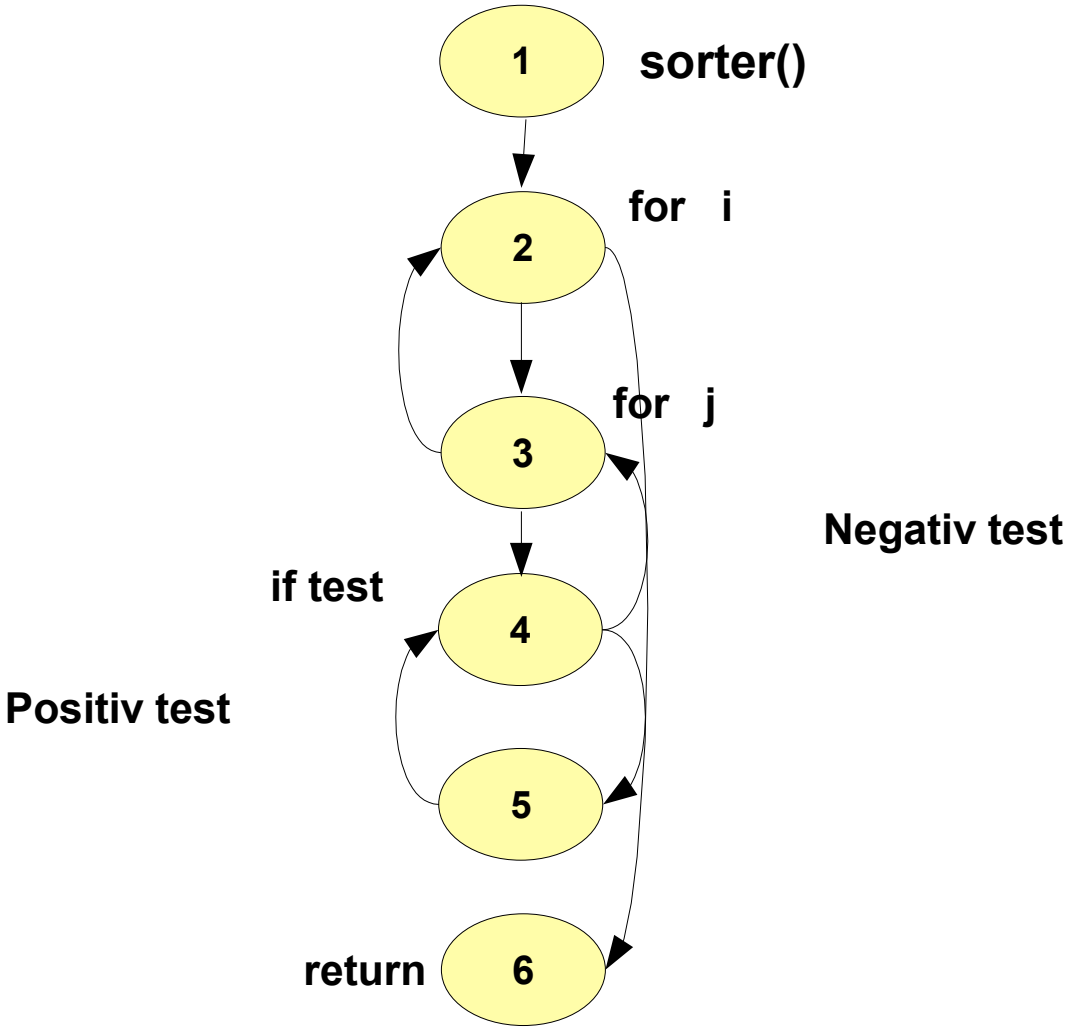
# Enhetstester

```
public static void sorter( int[] a ) {
    for ( int i = 1; i < a.length; i++ ) {
        for ( int j = 0; j <= i; j++ ) {
            if (a[i] < a[j]) {
                int tmp = a[j];
                a[i] = a[j];
                a[j] = tmp;
            }
        }
    }
}
public static void main( String[] args ) {
    int[] a = {5,3,2,6,2,11,0,13,18,7};
    sorter(a);
    for( int i = 0; i < a.length; i++ ) {
        System.out.print(a[i] + ", ");
    }
}

}
```

Universitetet i Stavanger

# Hvitbokstesting



Universitetet i Stavanger

# Enhetstester

Eulers formel: $V = E - N + 2$

- V: antall veier
- E: antall kanter
- N: er antall noder

I den grafen vi ser på er det 6 noder og 8 kanter. Dette gir 4 mulige veier.

1. Går ikke inn i første for løkke
2. Går inn i første for løkke, men ikke inn i neste
3. Går inn i begge for løkken, men ikke inn i if testen
4. Går inn i begge for løkkene og inn i if testen

Universitetet i Stavanger

# Enhetstester

- Vi må lage 4 tester som tester alle mulige uavhengige veier.
  - Når programmet vokser, vil antall veier eksplodere.
- Noen av nodene er løkker, det de gjennomløpes N ganger.
- Tar vi hensyn ti dette får vi at vi må ha mer enn N(N-1) tester, for å sortere N tall.
- Testene må lages slik at vi får alle typer gjennomkjøringer. Dette er helt umulig å gjennomføre i praksis.
- Men ut fra kjennskapen til strukturen i algoritmen kan vi velge strategier og tilhørende datasett.

Universitetet i Stavanger

# Enhetstester

1. Test av setninger. Vi sørger for at hver setning blir utført minst en gang
2. Test av forgreninger. Vi sørger for at hver gren blir utført minst en gang
3. Test av uavhengige veier. Vi sørger for at hver uavhengig vei blir utført minst en gang
4. Test av vei fra definisjon til bruk. Alle veier fra definisjonen av en variabel til dens bruk blir testet minst en gang
5. Test av all bruk. Testen må føre til at minst en vei fra hver definisjon fører til at all bruk av definisjonen nåes
6. Test av alle predikater og noen beregninger. For hver variabel og hver definisjon av variabelen må testen inkludere minst en vei fra definisjon til bruk i hvert predikat. Hvis noen variable ikke blir dekket, inkluderes vei fram til bruk.
7. Test av all bruk og noen predikater. For hver variabel og hver definisjon av variabelen må testen inkludere minst en vei til hver bruk av variabelen. Hvis noen variabel ikke blir dekket på denne måten, inkluderes veier fram til predikater.

Universitetet i Stavanger

# Enhetstester

Faren med hvitbokstesting

- Ser seg blind på den indre strukturen
- Glemmer andre viktige poeng.
- Disse blir tatt hensyn til i svartbokstesting.

Universitetet i Stavanger

# Svartboks-testing

- Kjenner vi ikke de interne detaljene og må velge testdata ut fra andre kriterier.
- Vi må sikre oss at de testdata vi velger, virkelig er i stand til å avdekke mulige feil.
- I fagkretser kalles det å finne testdatasett som dekker så mange feil som mulig.
- Det er ingen enkel oppgave. Det er mange måter å evaluere testdataene på er beskrevet i faglitteraturen.
- **Mutasjon.**
  - Det går ut på at en lager seg varianter av det programmet som skal testes, og legger inn kjente feil i det. Hvis testdatasett finner disse feilene, så aksepterer vi testdatasettet.

Universitetet i Stavanger

# Svartboks-testing

Et prinsipp går ut på:

- Gruppere testdata i spesielle klasser. En klasse representerer en bestemt bruk av programmet.
- Vi velger et representativt sett med testdata fra hver klasse, og konkluderer med at hvis programmet fungerer for disse dataene, så vil det fungere for alle andre data i samme klasse.
- Vi vil teste sorter()
- Vi kan spesifisere testdatasettet omtrent samtidig med at vi fastsetter de funksjonelle egenskapen til funksjonen.
- Testdataene kan brukes til å teste alle funksjoner som sorterer heltall. Grensesnittet mot funksjonen er:
- sorter()

Universitetet i Stavanger

# Svartboks-testing

- **public static void sorter( int[] a )**

Vi lager oss testdata som representerer det ekstreme, og det typiske.

1. Tabellen er tom
2. Tabellen inneholder kun et element
3. Antall element i tabellen er oddetall
4. Antall element i tabellen er partall
5. Antall element i tabellen er oddetall, og to av tallene er like
6. Antall element i tabellen er partall, og to av elementene er like
7. Antall element i tabellen er oddetall, og to av tallene er like. De tallene som er like er det første og siste tall
8. Antall element i tabellen er partall, og to av tallene er like. De tallene som er like er det første og siste tall

Universitetet i Stavanger

# Svartboks-testing

| *Klasse nr* | *Testdata for tabell* | *Antall element* | *Forventet resultat* |
|---|---|---|---|
| 1 | Tom tabell | 0 | Feilindikasjon |
| 2 | 10 | 1 | 10 |
| 3 | 12, 500, 3, 20, 34 | 5 | 3,12, 20, 34, 500 |
| 4 | 32, 100, 56, 1000, 123, 50 | 6 | 32, 50, 56, 100, 123, 1000 |
| 5 | 6, 8, 8, 10 | 5 | 6, 8, 8, 10, 23 |
| 6 | 76, 53, 27, 70, 27, 1 | 6 | 1, 27, 27, 53, 70, 76 |
| 7 | 1, 2, 8, 5, 1 | 5 | 1, 1, 2, 5, 8 |
| 8 | 23, 19, 50, 42, 90, 95, 23 | 8 | 10, 19, 23, 23, 42, 50, 90, 95 |

Universitetet i Stavanger

# Svartboks-testing

- Første klasse representerer problemet med at en prøver å bruke metoden før datasettet er initialisert.
- Denne måten å velge testdata på kalles *ekvivalenspartisjonering*
- Tankegangen er programmet oppfører programmet seg som forventet på disse dataene, så føles det intuitivt sannsynlig at det oppfører seg normalt på alle andre lignende datasett.
- Inndata klassifiseres i et sett av klasser. Klassene har felles egenskaper.
- Når man har identifisert klassen, finner man fram til data for hver klasse.

Universitetet i Stavanger

# Svartboks-testing

- Det er viktig å velge de rette testklassene.
- Antall klasser er normalt ikke ubetydelig, selv for enkle funksjoner.
- Vi er nokså sikre på at funksjonen vil sortere en uordnet tabell hvis den klarer disse testene.
- Men kan det være andre typer feil i funksjonen. Disse kan gi problemer for programmerere som skal bruke funksjonen i et eller flere systemer.
- Mange organisasjoner har som standard at alle funksjoner skal returnere en boolean i selve funksjonen og ha retur-parametere som indikerer en status etter at funksjonen er ferdig med oppgaven sin.

Universitetet i Stavanger

# Svartboks-testing

Det betyr:

- At alle programmerere vi samarbeider med, følger denne standarden.
- At en programvareenhet ikke følger standarden, kan føre til at klientprogrammet feiler. Denne type feil vil kanskje ikke oppdages før vi kommer langt ut i prosessen med å integrere systemet. Da blir det mye dyrere å rette.
- Det er viktig at alle programmerere som deltar i et utviklingsprosjekt, kjenner detaljert til hvilke standard som brukes.
- Man supplerer testingen med gjennomgåelse av koden for å finne feil av den typen vi har diskutert her.

Universitetet i Stavanger

# Integrasjons-testing

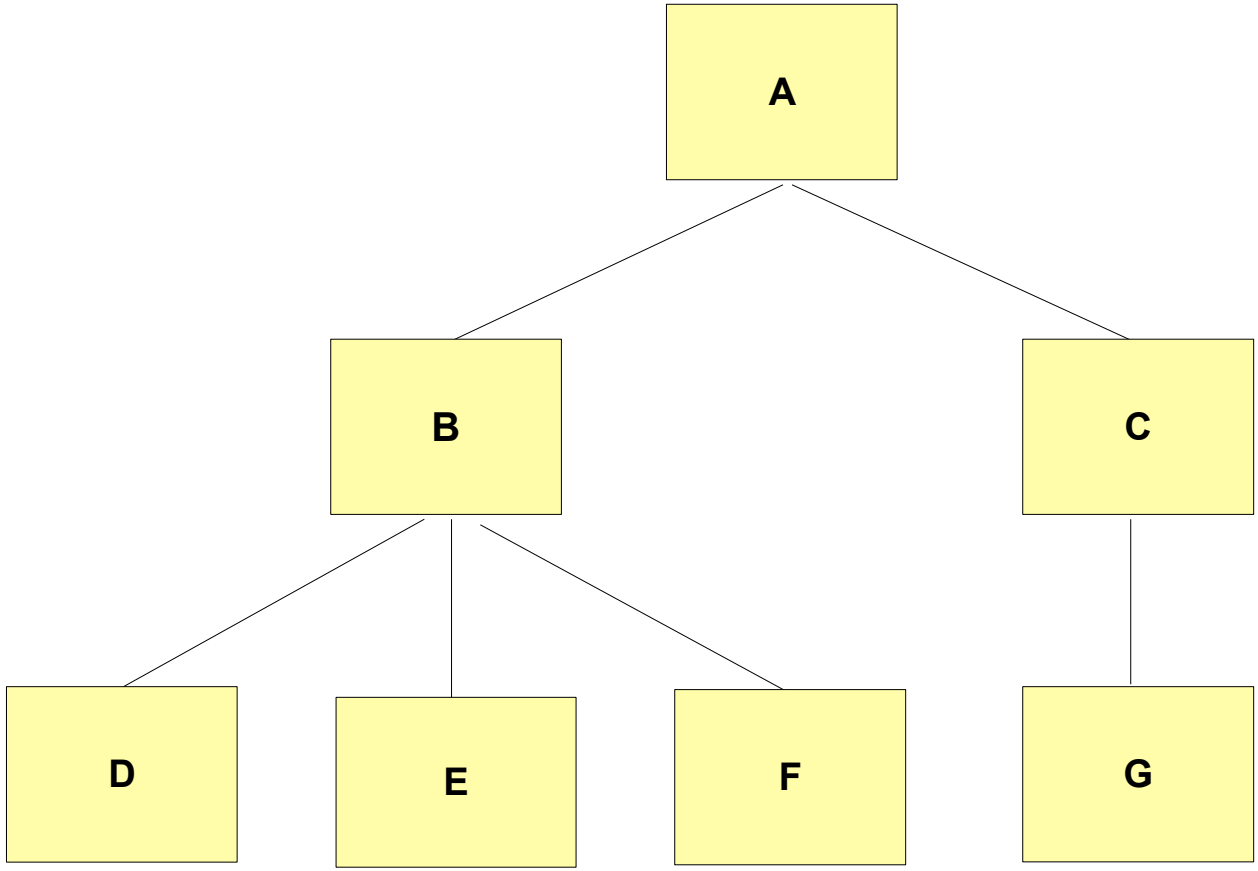
- Integrasjon er det arbeidet en gjør når forskjellige enheter eller moduler
- Etter hvert som en bygger sammen systemet, tester en for å se om enhetene fungerer som de skal. Vi vil finne ut om systemet oppfyller de spesifiserte design.

Integrasjon og testing kan organiseres som en trinnvis prosess. En kan ha ulike strategier:

- Fra bunnen og opp (bottom up)
- Fra toppen og ned (top down)
- Kombinasjon
- Det store smellet (big bang)

Universitetet i Stavanger

# Integrasjons-testing



Universitetet i Stavanger

# Integrasjons-testing

- Hvert lag inneholder moduler som blir mer og mer spesialiserte jo lenger ned i lagene vi kommer.
- Tradisjonelt er dette et resultat av strukturert design.
- Et objektorientert system vil ha mer preg av et nettverk av samarbeidende objekter,
- Innenfor hvert lag vil objektene danne et nettverk. Det er fullt mulig å tenke seg at boksene inneholder objekter/klasser.

Universitetet i Stavanger

# Fra bunnen og opp

- Starte på lavest mulig nivå, teste alle enheter på dette nivået for så å arbeide seg oppover i lagene.
- Med nedenfra og opp strategien starter en å teste modulen D, E, F og G.
- Deretter  tester en modulen B og C. For til slutt ende opp med å teste modul A.

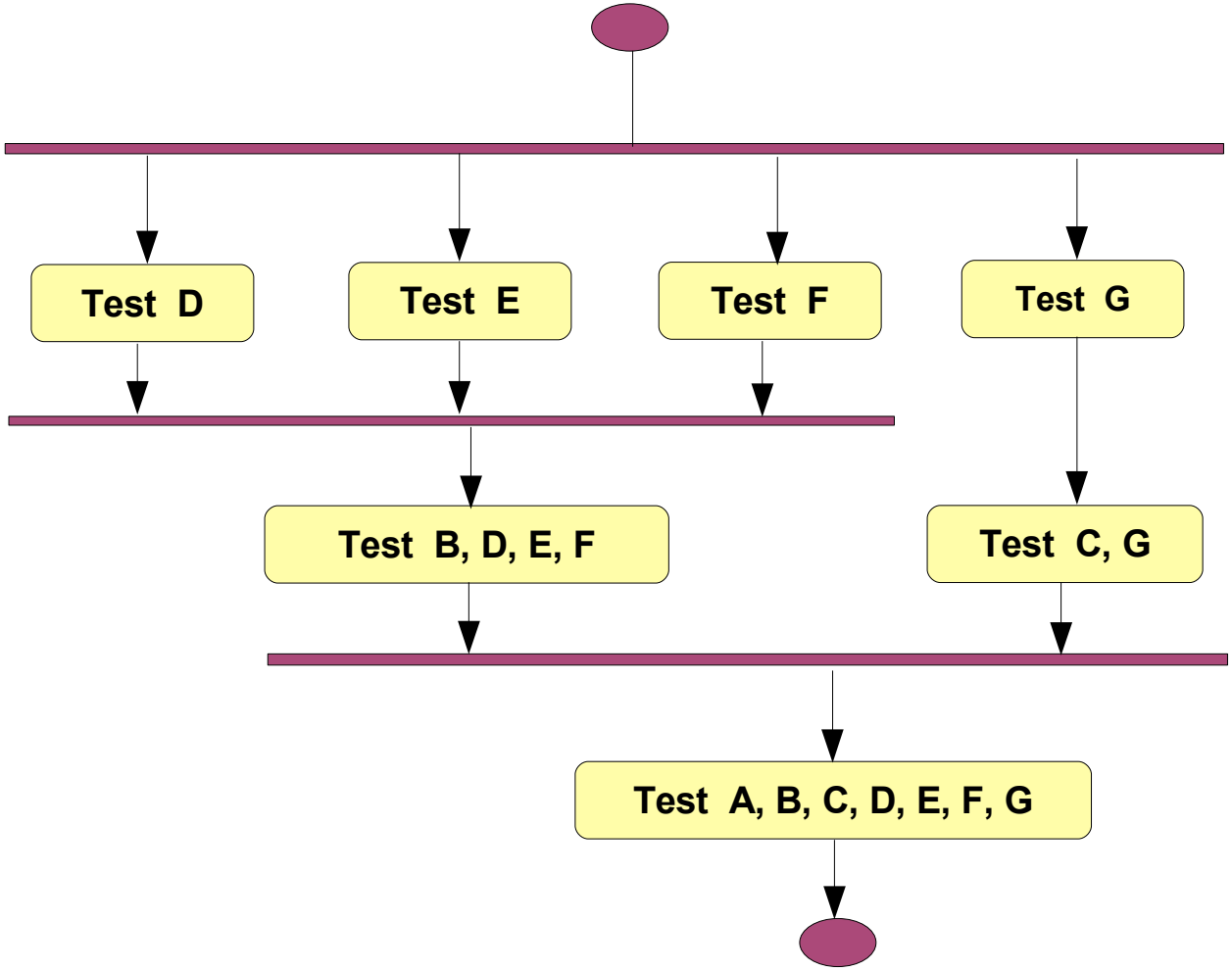
Universitetet i Stavanger

# Fra bunnen og opp

- Strategien krever testdrivere.
- Dette er programbiter som erstatter modulene på nivået over
- Det betyr at vi må skrive en god del kode som ikke brukes i det ferdige programmet.
- Dette er en svakhet med metoden, men dette er helt nødvendig for objektorientert utvikling der systemet bygger opp av klasser/objekter.
- Hver klasse må testes av testdrivere. Også når en bygger opp systemet av ferdige komponenter er dette en hensiktsmessig strategi.

Universitetet
i Stavanger

# Fra bunnen og opp



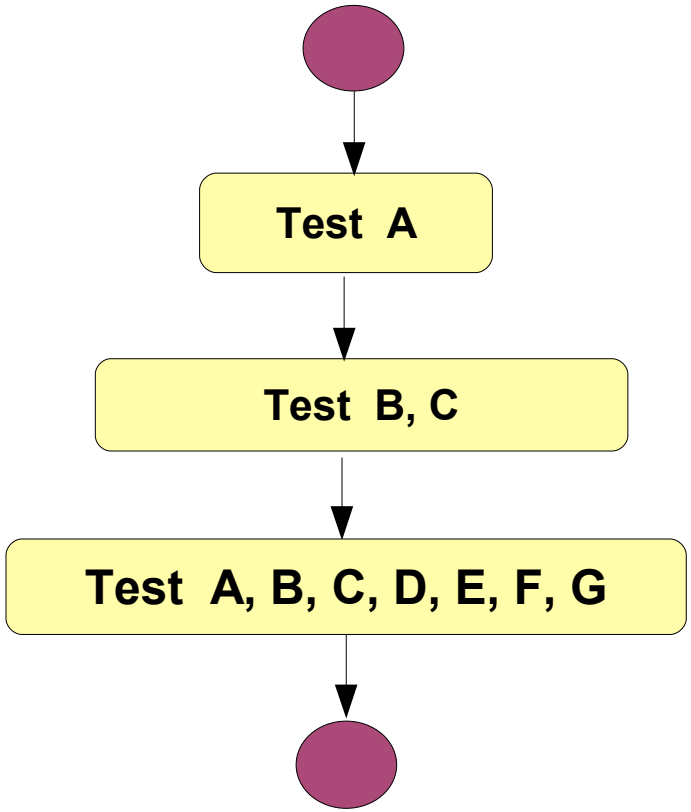
Universitetet i Stavanger

# Fra bunnen og opp

- En annen ulempe med denne nedenfra og opp-integreringen, er at en ikke får testet funksjonaliteten sett fra kundene og brukerne før hele systemet er ferdig.
- Dette er viktig ettersom misforståtte krav bør avdekkes så tidlig som mulig.

Universitetet
i Stavanger

# Fra toppen og ned



Universitetet i Stavanger

# Fra toppen og ned

- Starter en på toppen og tester modulen på øverste nivå først.
- Denne strategien krever at en erstatter de underliggende modulene med stubber.
- Dette er programbiter som simulerer oppførselen til underliggende moduler.
- I store system kreves det svært mange stubber. Dette er en ulempe med ovenfra og ned testing.
- En oppdager svært seint at modulen på laveste nivå ikke lar seg realisere. Det kan føre til at en må designe hele systemet på nytt.

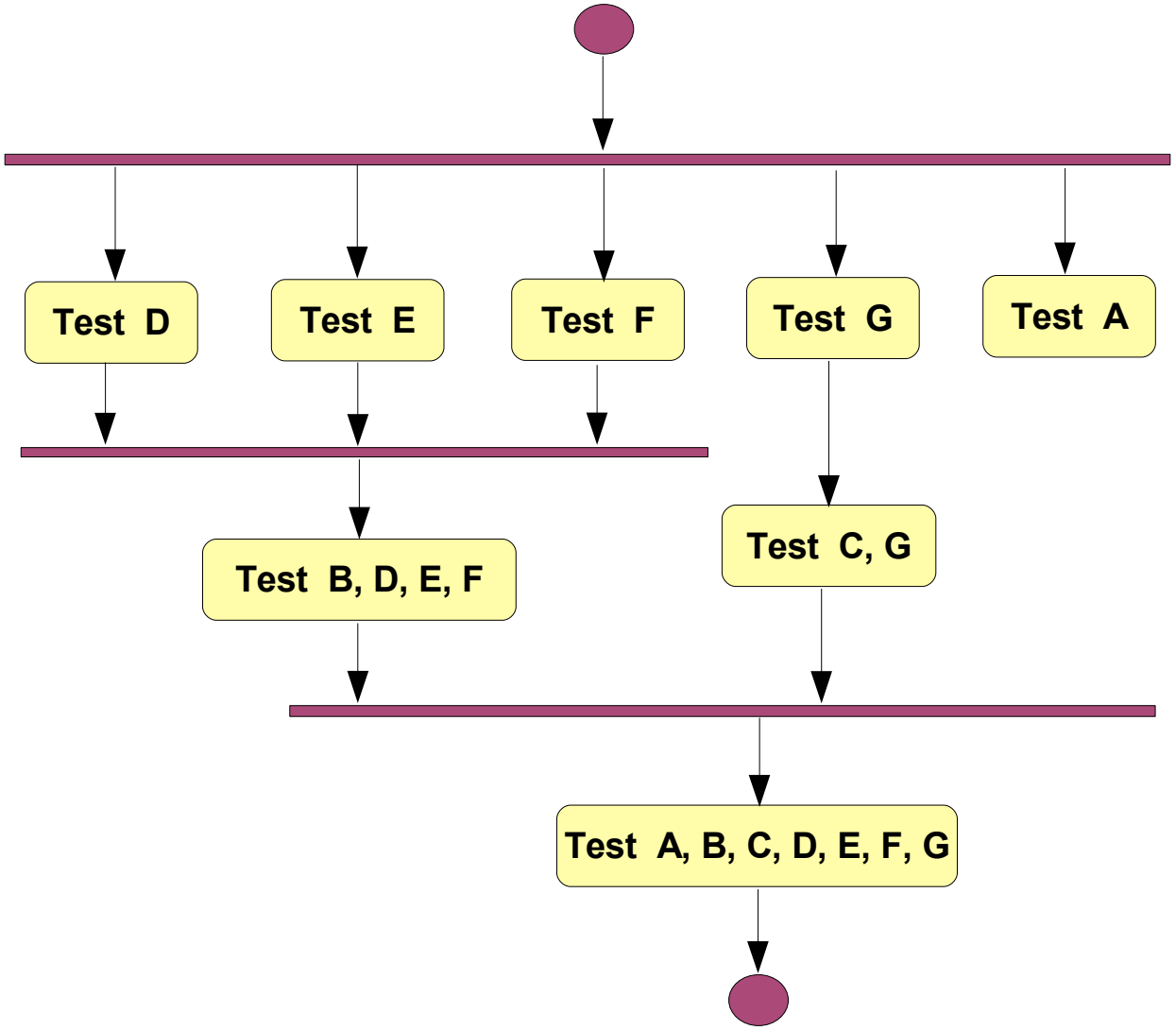
Universitetet i Stavanger

# Fra toppen og ned

- Fra toppen og ned ikke er noen ideell strategi som gjelder for alle typer systemer.
- En må gjøre et valg, og ofte kan en kombinasjon av disse to strategiene være hensiktsmessig.
- En kan ta for seg modulene på øverste og nederste nivå. Da må en lage både testdrivere og stubber.
- Systemet betraktes som om det består av tre lag. Det øverste behandles fra toppen og ned
- Det nederste laget, som kan bestå av standard komponenter, biblioteks-rutiner og lavnivå klasser
- Nivået i midten blir målet en arbeider mot.

Universitetet i Stavanger

# En kombinasjon



Universitetet i Stavanger

# Big bang

- Det siste alternativet er big bang.
- Systemet settes sammen og testes som et hele.
- Denne strategien er ikke å anbefale ettersom det blir vanskelig å lokalisere moduler som er kilde til eventuelle problemer.
- Med de andre strategiene får en på en ordnet måte først prøvd de enkelte modulene og forvisset seg om at de spiller sammen med sine nærmeste samarbeidspartnere.
- En avdekker og lokaliserer feil som skyldes samspillet mellom moduler.

Universitetet i Stavanger

# Hvor lenge skal en teste

- Umulig å lage tester som avdekker alle feil.
- Vi vet ikke hvor mange feil programmert inneholder, og det er umulig å gjennomføre tester med alle tenkelige fremtidige kombinasjoner av inndata til programmet.
- Det er nødvendig å ha en viss tillit til at de testene vi gjennomfører, avdekker de viktigste feilene i programmet.
  - *konfidens*.
- Vi ønsker å kunne anslå med en gitt sannsynlighet hvor mange feil som gjenstår, og hvor hyppig programmet dermed kan komme til å feile.
- Vi og/eller kundene kan vurdere om dette er akseptabelt eller om ytterligere testing må gjøres.

Universitetet i Stavanger

# Hvor lenge skal en teste

Såing av feil er en metode. Ideene at en legger inn et antall feil i programmet. Så utfører en tester. Under testingen vil en avdekke kjente og ikkekjente feil. Hvis en antar at vi er like effektiv til å oppdage kjente og ukjente feil, kan vi sette opp ligningen:

N = Totalt antall ikke sådde feil

n = Totalt antall oppdagende ikke sådde feil

S = Totalt antall sådde feil

s = Totalt antall oppdagende sådde feil

$$\frac{s}{S}=\frac{n}{N}\Rightarrow N=\frac{S\cdot n}{s}$$

Antall gjenværende ikke sådde feil er da:

$$n_R=N-n=(\frac{S\cdot n}{s})=n\cdot(\frac{S}{s}-1)$$

Universitetet i Stavanger

# Hvor lenge skal en teste

En må ta stilling til om dette er noe en kan leve med.

Den forutsetningen vi har gjort, er at de sådde feil er av samme type som de ikke sådde feil, og at vi er like effektive i å oppdage sådde som ikke sådde feil. Etter som vi på forhånd ikke kan vite hva som er typisk for de ikke sådde feil, er det ikke gitt at forutsetningen holder. Erfaring kan være med å øke vurderingsevnen.

Problemet med såing av feil er at en vanskelig kan vite hva som er typiske feil, og dermed kunne så slike feil.

# Hvor lenge skal en teste

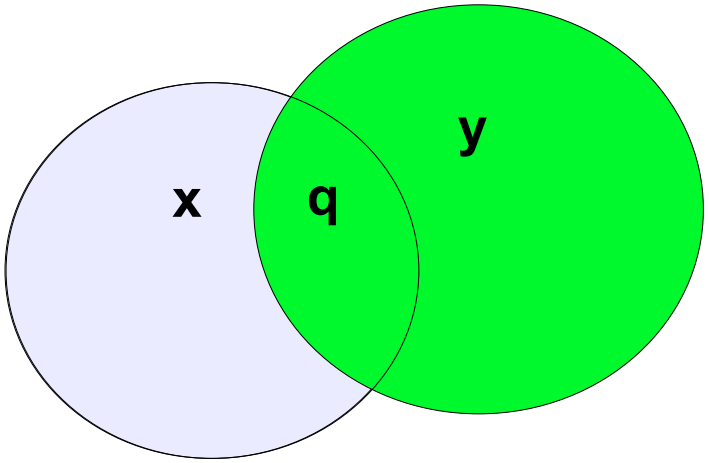
En annen alternativ er å bruke to uavhengige testgrupper. De to gruppen vil oppdage forskjellig antall feil, men noen feil er felles

n = Totalt antall feilbegrep

x = Totalt antall feil oppdaget av gruppe 1

y = Totalt antall feil oppdaget av gruppe 2

q = Antall feil som er oppdaget av begge grupper



Universitetet i Stavanger

# Hvor lenge skal en teste

Generelt har vi:

$$q \leq x \wedge q \leq y$$

Sannsynligheten for at en feil blir oppdaget av gruppe 1 er:

$$P_1 = \frac{x}{n}$$

Sannsynligheten for at en feil blir oppdaget av gruppe 2 er:

$$P_{(1,2)} = \frac{q}{n}$$

Sannsynligheten for at en feil blir oppdaget av begge gruppene er:

$$P^{(1,2)} = (1 - P_1) \cdot (1 - P_2)$$

Universitetet i Stavanger

# Hvor lenge skal en teste

Litt triksing med formlene gir:

$$P_{(1,2)} = P_1 \cdot P_2 \Rightarrow \frac{q}{n} = \frac{x}{n} \cdot \frac{y}{n} \Rightarrow nq = xy \Rightarrow q = \frac{xy}{n}$$

$$n = \frac{xy}{q}$$

Universitetet i Stavanger

# Hvor lenge skal en teste

Eksempel:

- x = 24, y = 36, q = 16

$$n = \frac{24 \cdot 36}{16} = 54$$

De to gruppene har oppdaget:

- 24 + 36 – 16 = 44 antall feil

Antall feil som gjenstår er:

- 54 – 44 = 10

Spørsmålet blir om dette er noe vi kan leve med, eller om det må utføres mer testing

# Kapittel 19
# Validering og verifisering

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Systemtest/Integrasjonstest

- Systemtester skal finne ut om programmet fungerer slik designeren vil.
- Testen skal vi finne ut om systemet oppfører seg slik brukere og kunder vil.
- Systemtester er storskalatester. De utføres gjerne over litt legere tid for å se om det totale systemet oppfører seg slik som det skal.
- Testsituasjonen bør ligne mest mulig på brukssituasjonen som programsystemet vil møte når det er i operativ drift.
- Den eneste praktiske måten å utføre systemtester på er etter svartboksprinsippet.
- Systemtestene bør utføres av en uavhengig gruppe som samrår seg med kunde/brukere og utviklere.

Universitetet i Stavanger

# Systemtest

Systemtestene er de fire siste testene som er vist i figur 19.1

- Funksjonstest
- Ytelsestest
- Akseptansetest
- Installasjonstest

Universitetet i Stavanger

# Funksjonstest

- Dette er den første testen som utføres på hele systemet.
- Den skal forvisse seg om at de funksjonelle kravene er tilfredsstilt.
- Utgangspunktet er kravspesifikasjonen
  - Har høy sannsynlighet for at feil blit oppdaget
  - Gjennomføres av en uavhengig testgruppe
  - Bruker både gyldige og ugyldige data
  - Ikke modifiserer systemet for å gjøre testingen lettere
  - Har stoppkriterier
- En tester en funksjon fra kravspesifikasjonen av gangen.
- Det betyr at testingen kan starte før hele systemet er ferdig.

Universitetet i Stavanger

# Ytelsestest

- Ytelsestestene skal vi finne ut om systemet tilfredsstiller de ikke-funksjonelle kravene.
- Disse kravene finner en også i kravspesifikasjonen.
- Flere ytelsestester  kan utføres.

*Stresstest*

Testen skal avdekke i hvilke grad systemet er i stand til å tåle å bli utsatt for ekstreme påkjenninger.

Et visst antall samtidige brukere.

Kjøretid.

*Volumtest*

Vi skal teste om systemet tåler store datamengder

*Konfigurasjonstest*

Skal finne ut om forskjellige konfigurasjoner fungerer som de skal

Universitetet i Stavanger

# Ytelsestest

*Kompatibilitetstest*

Kan systemet kan samspille med andre systemer?

*Regresjontest*

Kan system erstatte et annet?

*Sikkerhetstest*

Tilfredsstiller systemet krav til tilgjengelighet, integritet og konfidensialitet med hensyn til data og tjenester

*Timing-test*

Her undersøkes responstiden. Slike tester gjennomføres gjerne i forbindelse med stresstester for å se om kravene til responstid holder under ekstreme påkjenninger

*Omgivelsetester*

Skal vise om systemet oppfører seg som spesifisert på forskjellige installasjoner

Universitetet i Stavanger

# Ytelsestest

*Kvalitetstest*

Har systemet den nødvendige pålitelighet, tilgjengelighet og vedlikeholdbarhet

*«Recovery» - test*

Kan systemet ta seg inn etter forskjellige former for avbrekk eller feilsituasjoner

*Vedlikeholdstest*

Er behovet for diagnostiske hjelpemidler er møtt.

*Dokumentasjonstest*

Er de nødvendige dokumentene skrevne

*Menneskelige faktorer*

Er brukergrensesnittet som spesifisert

Universitetet i Stavanger

# Ytelsestest

Testing av ytelse er å få svar på om systemet tilfredsstiller kravene til pålitelighet, tilgjengelighet og vedlikehold-barhet

*Pålitelighet*

Sannsynligheten for at systemet vil fungere gjennom en tidsperiode uten feil. Er denne sannsynligheten stor er systemet pålitelig

*Tilgjengelighet*

Sannsynligheten for at systemet vil fungere i henhold til spesifikasjonene ved et gitt tidspunkt. Et system som er fullstendig opp å gjøre på et gitt tidspunkt, har høy tilgjengelighet.

*Vedlike holdbarhet*

Sannsynligheten for at en vedlikeholds-aktivitet kan bli utført innenfor et angitt tidsrom

Universitetet i Stavanger

# Ytelsestest

- MTBF: midlere tid mellom feil
- MTTF: Midlere tid til neste feil
- MTTR: Midlere tid til reperasjon
- R: Pålitelighet
- A: Tilgjengelighet
- M : vedlike holdbarhet

$$MTBF = MTTF + MTTR$$

$$R = MTBF/(1 + MTBF)$$

$$A = MTBF/(MTBF + MTTR)$$

$$M = 1/(1 + MTTR)$$

Universitetet i Stavanger

# Ytelsestest

- Tallene estimeres ved å inn data under testing og drift av systemet.
- I en periode registrerer tiden mellom oppdagelse av feil, gjennomsnittet for alle disse tidene er MTTF.
- Tiden fra det øyeblikket en feil er reparert og systemet kjører til det igjen feiler
- Denne tiden er en stokastisk variabel, og påliteligheten er knyttet til betraktninger omkring sannsynlighetsverdier for denne variabelen. Et enkelt estimat for forventet tid til neste feil kan vi gjøre ved å se på tiden mellom feil.

# Ytelsestest

Vi har gjort følgende registrering

| $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ |
|---|---|---|---|---|---|
| 3 | 30 | 113 | 81 | 115 | 9 |

Et estimat for tiden til neste feil blir:

$$\frac{(3+30+113+81+115+9)}{6}=58,5$$

Tar en de to siste får en:

$$\frac{(9+115)}{2}=62$$

Universitetet i Stavanger

# Ytelsestest

- I dette tilfellet er ikke forskjellen stor.
- Er vi nøye med vår testing og vår reparasjon, vil vi se at estimatene over tid vil nærme seg hverandre,
- Trenden vil være at tiden mellom hver feil øker. Det vil si at påliteligheten øker.

Universitetet i Stavanger

# Akseptanstest

- Testen utføres av de som skal ha systemet.
- Det har mye til felles med systemtesten, men det er kundene som setter premissene.
- Systemet blir testet med virkelige data. Det vi si data hentet fra miljøet hvor systemet skal brukes.
- I avtalen som inngås mellom utvikler og kunde, vil det være en klausul om at program-systemet må klare en akseptanse-test før kunden aksepterer systemet og betaler for det.
- Det er flere typer tester som kan defineres under denne overskriften.

Universitetet i Stavanger

# Akseptanstester

*Benchmark – test*

Utføres for å sammenligne flere funksjonelt like systemer

*Alfatest*

Utføres hos utvikleren, men simulerer brukermiljøet. Brukeren tar systemet i bruk som om det var i normal drift.

*Betatest*

Utføres hos kunden. Dette er vanlig dersom systemet skal leveres til et stort antall kunder. Leverandøren velger ut en representativ gruppe kunder som får systemet til utprøving. Testen gjennomføres før det ferdige systemet sendes ut på markedet. Det er vanlig å utføre betatester på et system som er beregnet på et massemarked. Eksempler

- Kompilatorer
- Teksbehandlingssystemer
- Regneark
- etc.

Universitetet i Stavanger

# Akseptanstest

- Både alfa og betatester beregnes som pilottester.
- De er kjennetegnet ved at testene utføres gjennom noe som ligner daglig bruk.
- «Benchmark» - tester, på den andre siden, utføres med nøye valgte data for å sjekke spesifikke egenskaper.
- Akseptansetester kan også foregå parallelt med bruk av eksisterende systemer slik at man kan kontrollere at man oppnår forbedring gjennom bruk av det nye systemet.
- En gradvis overgang til det nye systemet kan dermed gjøres.

Universitetet i Stavanger

# Installasjonstest

- Testen utføres når systemet er installert i sitt rette driftsmiljø.
- Hvis akseptanse-testen har gått bra, er det ikke alltid at vi trenger denne testen.
- Av og til ønsker man en siste test på at systemet oppfører seg fornuftig i driftmiljøet.
- En lager, sammen med brukerne, tester for å se om alt fungerer som det skal også etter installasjonen.
- Den daglige bruken vil være en pågående installasjonstest, og systemet vil trolig feile over tid.

Universitetet i Stavanger

# Regresjonstest

- Etter endringer i et program, må en forvisse seg om at det ikke er introdusert nye feil i programmet.
- Vi kjører om igjen de testene programmet ikke feilet på før. Hvis vi oppdager nye feil, blir programenheten avvist. Vi gjør regress.
- Regresjons-tester kan automatiseres.
- Testdata kan legges på en fil. Forventet resultat legges på en annen fil. Vi lar programenheten vi skal teste, hente data fra inn-filen og sende resultatene til ut-filen.
- Innholdet på utdatafilen sammenlignes med innholdet på filen med forventet resultat, og avvik rapporteres.
- Det finnes avansert verktøy for regresjons-testing og for andre typer tester.

Universitetet i Stavanger

# Feilfinning er ikke testing

- Etter at programmet feiler, må feilen lokaliseres og rettes.
- Dette er ingen triviell oppgave.
- Særlig er dette vanskelig hvis det er andre som skal lokalisere feilen, enn de som har utviklet koden.
- En et stykke på vei med intelligent gjetning.
- Registrer under hvilke betingelser programmet feiler.
- Eventuelle feilmeldinger fra operativsystemet studeres, og med kjennskap til hvilke type av feil som mest hyppig forekommer, kan man begynne å søke i bestemte moduler og funksjoner som man mistenker inneholder kilden til feilen.

Universitetet i Stavanger

# Feilfinning er ikke testing

- For ytterligere å innskrenke området for hvor feilen kan ligge, og ekskludere andre deler av systemet, legger en inn utskrifter med meldinger om hvor langt kjøringen av programmet har kommet.
- Ved å studere rekkefølgen på disse utskriftene etter at programmet har feilet, vil vi etter hvert kunne snevre inn området for hvor feilkilden ligger i programmet. Dette er den eneste systematiske måten å lokalisere feilkilder på. Det har lenge vært brukt av programmere som det eneste effektive hjelpemiddelet.
- Dette kan være svært tidkrevende og krever stadige ompompileringer. For å slippe for mange rekompileringer kan man i C++ legge slike utskrifter inn i koden. Utskriftene slås på eller av ved å sette en opsjon  til kompilatoren.

Universitetet i Stavanger

# Feilfinning er ikke testing

Etter hvert har en fått feilfinningsverktøy som programmererens verden lettere. På engelsk kalles dette for «debugger» Slikt verktøy har vanligvis disse egenskapene:

- Eksekverer en programline av gangen
- En kan sette stopp-punkter i programmet
- Det er mulig å se hvordan innholdet i variabler endres under eksekvering
- Het et mulig å se innholdet i en variabel og /eller endre innholdet
- Det er mulig å få en liste over programsetninger som er gjennomløpet inntil programmet stopper ved et stopp-punkt

Universitetet i Stavanger

# Feilfinning er ikke testing

I et integrert utviklingsmiljø vil feilfinningsverktøyet være koblet sammen med editoren slik at når feil lokaliseres av verktøyet, så vil det bli markert hvilke kodelinje feilkilden ligger. Samtid kommer en forklarende tekst i et annet vindu.

Universitetet i Stavanger

# Feilbegreper

Begrepet er brukt uten å spesifisere hva som menes. Vanligvis framgår det av sammenhengen. I det følgende vil vi avklare noen begreper. (Humphrey, 1989)

- Error – feil gjort av mennesker
- Defekter – Uheldige egenskaper i programmet. Det skyldes som regel en error. Ikke alle feiltagelser fører til defekter. En feil i en kommentar vil ikke føre til en defekt
- Bugs – en defekt i programmet som kommer til syne under bruk. Begrepet kan gi assosiasjoner til noe som kommer inn i programmet utenfor kontroll. Og det er ikke slik, det er et resultat av en error.
- Failures – Betyr at et installert program tidvis ikke oppfører seg slik det skal. Det kan skyldes bug, feil installasjon, maskinvarefeil, kommunikasjons-problemer
- Problemer – er brukeropplevd. De kan skyldes «failures», feil bruk eller misforståelser. Problemer er menneskelige hendelser i motsetning til «failures», som er systemhendelser.

Vi har stort sett brukt begrepene feil og feiling. Sammenhengen de brukes i, vil som regel indikere hvilke av disse begrepene det dreier seg om.

Universitetet i Stavanger

# Klassifisering av feil

Klassifisering og kategorisering av feil er nyttig.

Ved å samle data kan en finne ut hvilke kategorier av feil som hyppigst opptrer.

Da har man et redskap til å sette inn tiltak for å eliminere årsaken til feilene.

En måte å klassifisere feil på finnes hos Grafy (Grady 1997).

- Hvor ble feilen introdusert ( i hvilke fase i utviklingen)
- Hvilke kategori tilhører feilen
- Hvilke modus er feilen i
    1. Noe som mangler
    2. Noe er uklart
    3. Noe er galt
    4. Noe er endret
    5. Noe kan gjøres bedre

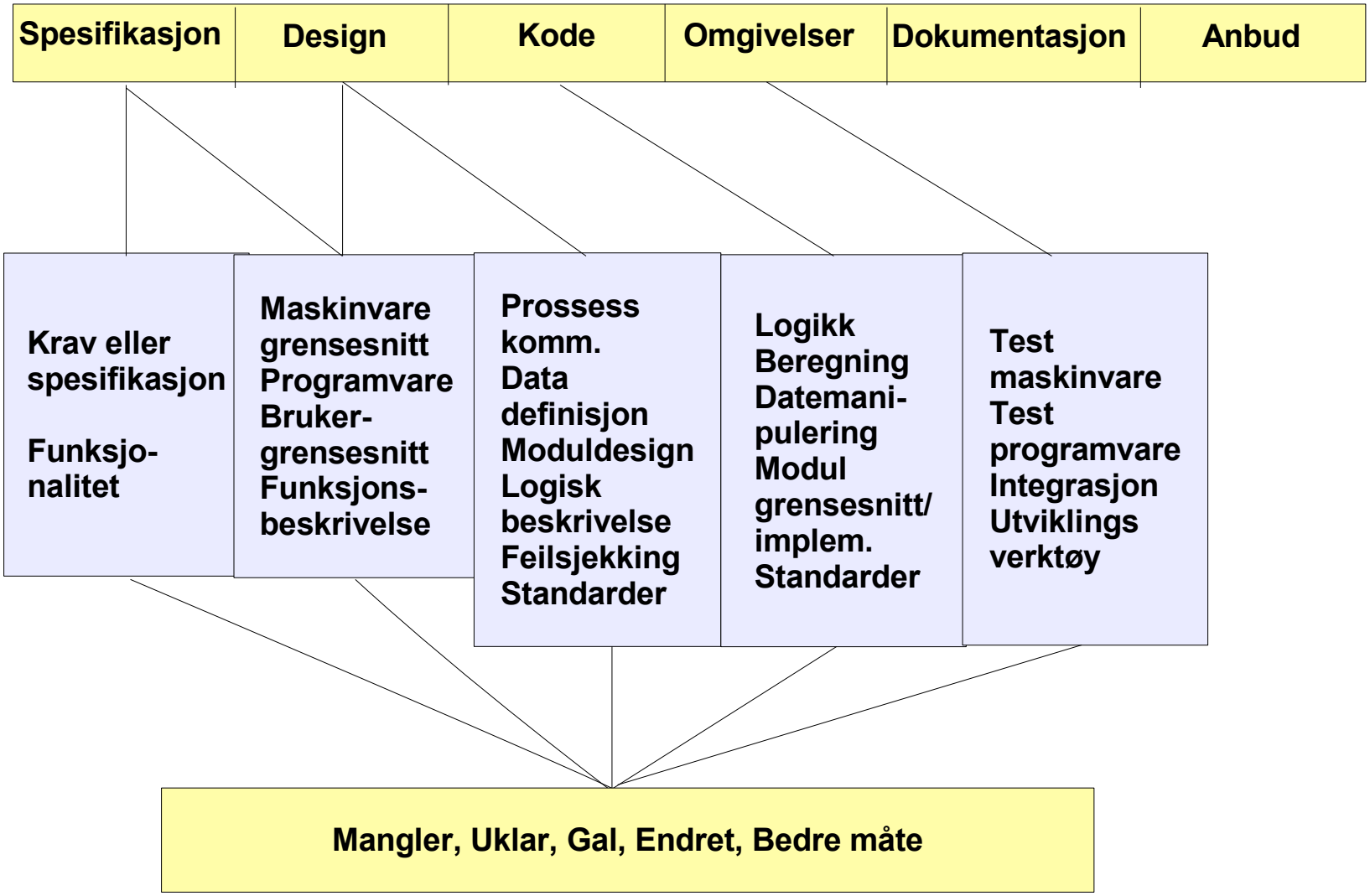
Universitetet i Stavanger

# Klassifisering av feil

- Med en slik klassifisering kan man ved hjelp av verktøy lage forskjellige oversikter som viser hyppigheten av bestemte feil, hvor de introduseres, hva feilen skyldes og årsaken til feilen.
- Å bli enig om en klassifisering som alle i organisasjoner forstår, kan være vanskelig. Uten en slik felles forståelse blir verdien av innsamlede data dårligere. Konklusjoner trukket ved å sammenligne prosjekter, blir mindre pålitelige.

Universitetet i Stavanger

# Klassifisering av feil



Universitetet i Stavanger

# Gjennomgåelser

Hva er gjennomgåelse? Vi starter med en ISO 9000:2000 definisjon

- *Gjennomgåelse (review) – aktivitet som gjennomføres for å bestemme hensikts- messigheten, tilstrekkeligheten og virkningen av det aktuelle emnet for å oppnå etablerte mål*
- *Revisjon (audit) – systematisk, uavhengig og dokumentert prosess for å framskaffe revisjonsbevis og å bedømme objektivt, i hvilke grad kriterier for revisjon er oppfylt.*
- *Inspeksjon (inspection) – bedømmelse av samsvar ved observasjon og vurdering understøttet når det er aktuelt med måling, prøving eller tolkning.*

Universitetet i Stavanger

# Gjennomgåelser

- Vi har sett på hva begrepene innebærer i forbindelse med vurdering av kvalitetssystemer.
- Alle typer feil kan ikke finnes ved testing.
- Testing er noe vi gjør ved å kjøre programmer, eller deler av programmet med et datasett og registrerer oppførselen.
- Dette er kun mulig å gjøre atter at en er kommet ganske langt i utviklingsprosessen.
- Erfaring viser at jo lenger i utviklingsprosessen vi er kommet, jo vanskeligere og dyrere er det å finne og reparere feil.
- Undersøkelser viser at mange av årsakene til at programmer feiler, er å finne i mangelfull kravspesifikasjon.
- Det er avgjørende for et godt resultat at en legger inn V & V – aktiviteter med jevne mellomrom utover i hele utviklingsprosessen. Det er uavhengig av hvilke prosess vi utvikler.

Universitetet i Stavanger

# Gjennomgåelser

- Ulike typer gjennomgåelse har vist seg å være effektive i å avdekke feil tidlig i utviklingsprosessen.
- Men det er også effektivt under koding.
- Det er ulike typer gjennomgåelse.
- Felles for alle typer gjennomgåelse er at man går kritisk gjennom det produktet som er laget.
- Hensikten kan være å avdekke feil og mangler, og/eller forvisse seg om at produktet oppfører seg i samsvar med spesifikasjoner og standarder.
- Gjennomgåelser kan være formell eller uformell.
  - Kollegagjennomgang
  - Mer formelle gjennomgåelser

Universitetet i Stavanger

# Kan bli underlagt gjenomgåelse

- Planer
- Spesifikasjoner
- Testopplegg
- Kodeinspeksjon
- Dokumentasjon
- Design

Universitetet i Stavanger

# Gjennomgåelser

Grovt kan en dele gjennomgåelsen inn i tre typer:

- Ledelsesgjennomgåelse
- Teknisk gjennomgåelsen
- Inspeksjon
- Gjennomganger (walk fhrough)
- Revisjon (audits)

Hensikten er å forbedre kvalitet og produktivitet.

Universitetet i Stavanger

# Ledelsesgjenomgåelse

- Skal gi ledelsen innsyn slik at framgangen i prosjektet sikres.
- Det kan føre til at forbedring tiltak settes i verk og å sikre god ressursutnyttelse.
- Denne type gjennomgåelse utføres av to eller flere personer fra ledelsen og de som har ansvar for gjennomføringen av prosjektet.
- Resultatet av aktiviteten er en gjennomgåelserapport.

Universitetet i Stavanger

# Teknisk gjennomgåelse

- Målgruppen er ledelsen.
- Skal sikre at det man gjør, er i henhold til planer og spesifikasjoner.
- Skal sikre at endringer gjøres på en ordnet måte.
- Den tekniske gjennomgåelsen utføres av tre eller flere fra den tekniske ledelsen av prosjektet med sjefingeniøren som leder. Utviklere kan delta. Målet for gjennomgåelsen er:
  - Er programmet i henhold til tekniske spesifikasjoner
  - Er programvaren er i samsvar med de standarder, retningslinjer, planer og prosedyrer som skal følges i prosjektet
  - Å sikre at endringer blir skikkelig utført
  - Å sikre god kommunikasjon mellom kundens og utviklernes tekniske stab

Universitetet i Stavanger

# Inspeksjoner og gjennomganger

- Gjennomgåelsene skal hjelpe de som utfører arbeidet, til å gjøre en bedre jobb
- Uavhengige inspektører går gjennom det arbeidsstykket som er laget for å se om det inneholder feil eller mangler.
- Baserer seg på prinsippet, programmering uten høy ego-faktor. Godt resultat ikke er en persons ansvar
- Ingen skal henges ut, det er hjelp til selvhjelp.
- Folk fra ledelsen skal ikke delta i slike gjennomgåelser. Det som i hovedsak skiller inspeksjon fra gjennomgåelser, er graden av formalisme.
- En gjennomgang er uformell.
- To eller flere kolleger går gjennom for eksempel programkode for å se etter mulige feil eller mangler.
- En gjennomgang er et forum læring. Inspeksjoner er formelle og strukturerte. De utføres av tre til seks personer med klart definerte roller og ansvar.

Universitetet i Stavanger

# Inspeksjoner

- Det er viktig å gjennomføre inspeksjoner i alle utviklings- prosjekter. Skal bidra til å motivere til bedre arbeid. Hensikten er å finne feil så tidlig som mulig. Skal også sikre at det er enighet om hva som skal lages, og at avtalte retningslinjer blir fulgt. Inspeksjoner brukes spesielt på kode.
- Dette er hovedprinsippet for inspeksjoner. «Managing the software process»(Humprey 1989)
- En inspeksjon er formell. Det er regler som skal følges. Det skal brukes sjekklister, og deltagerne har bestemte roller.
- Sjekklister og standarder lages for hver type produkt som skal inspiseres. Om nødvendig skreddersys disse til et spesielt prosjekt.
- Deltakerne må være godt forberedt, de må kjenne sine roller.
- Hensikten er å finne problemer. Disse løses ikke under inspeksjonen, men i ettertid av den eller de som har laget produktet som inspiseres.
- Inspeksjoner utføres av utviklere for utviklere. Ledelsen deltar ikke.
- Det samles inn data om inspeksjons-prosessen og produktet, dette grunnlag for å forberede både inspeksjons-prosessen og produktet.

Universitetet i Stavanger

# Inspeksjoner

**Det er fire roller**

- *Moderatoren:* Personen som leder inspeksjonen
- *Produsenten:* En eller flere personer som har laget produktet
- *Inspektørene:* De som skal gjennomgå produktet for å finne ut om det er potensielle problemer med det som er laget. De må ha god erfaring med utvikling av de aktuelle typer produkter og er gjerne kolleger av produsentene.
- *Referenten:* Person som skal sørge for at all relevant data om inspeksjonen blir notert og tatt vare på. Under inspeksjonen skal det noteres alle problemområder som dukker opp, hvem som har påvist problemet, og hvem som er ansvarlig for å løse problemet.

Universitetet i Stavanger

# Inspeksjoner

Inspeksjons prosessen består av trinnene

- Forberedelese
- Gjennomføring av selve inspeksjonen
- Etteraktiviteter

Starter med at ledere og produsenter avgjør at et produkt er klart for inspeksjon.

Inspeksjons-gruppen settes sammen, og moderator utpekes. En holder et innledende møte. Her framlegges det materialet som skal brukes under inspeksjonen. Målene med inspeksjonen klargjøres. Deltagerne går hver til seg og forbereder seg. De setter seg inn i standarder og sjekklister og leser gjennom produktet. Mulige problem og feil noteres i loggskjema. Tiden som brukes skal også registrenes. Dataene leveres til moderator før eller ved åpningen av inspeksjons-møtet.

Universitetet i Stavanger

# Inspeksjoner

- Inspeksjonen starter med at moderator forsikrer seg om at alle deltakerne er forberedt. Hvis ikke, avlyses møtet og et nytt berammes. Det er viktig at alle er forberedt, slik at det ikke spilles tid under inspeksjonen.
- Produsenten går gjennom rapporterte feil og problemer. Dette gjøres for å avklare om det er virkelige feil eller kun en misforståelse. Data om feilene blir notert. Det som noteres er, hvor ble feilen funnet, hvilke kategori av feil, årsaken til feilen hvis det lar seg avgjøre raskt.
- Etter at de viktigste feil er gjennomgått og diskutert, leser man gjennom produktet igjen for å avdekke eventuell mindre feil og mangler. Også disse noteres.

Universitetet i Stavanger

# Inspeksjoner

- Inspeksjonen avsluttes med at moderator oppsummerer og forvisser seg om at alle deltagerne er blitt hørt, og at de har fått lagt fram sine synspunkter.
- Moderator avklarer videre tiltak, datoer for når problemene skal være løst og feil rettet opp noteres.
- Alle skjema fylles ut. Ut fra en vurdering av problemer og feil som er funnet, avgjøres det om det er behov for en ny inspeksjon.

Universitetet i Stavanger

# Inspeksjoner

- Etter inspeksjonen må det gjøres etterarbeid.
- Produsenten må rette feil og løse de problemene som er påvist.
- Etter en vurdering av resultatene, avgjør moderator om det er behov for ny inspeksjon. Dette kan bli nødvendig selv om tidligere avgjørelser tilsa det motsatte. Det gjelder særlig dersom endringene som er gjort, ble mer omfattende enn opprinnelig antatt.
- Det siste moderator gjør, er å sørge for at resultatene for inspeksjonen kommer på plass i databasen for inspeksjoner.
- Ledelsen blir informert om resultatene og at inspeksjonen var vellykket.

Universitetet i Stavanger

# Revisjon

- En revisjon er en uavhengig evaluering av en prosess og/eller et produkt.
- Revisjonen gjøres mot en etablerte standarder. Hensikten er å sikre at en prosess virkelig følger etablerte standarder., og at nødvendig dokumentasjon blir laget, og at rapporter forteller hva som blir gjort i de forskjellige aktivitetene.
- Revisjoner kan være planlagt eller holdes etter behov

Universitetet i Stavanger

# Revisjoner kan ha ulikt fokus

- Rettet mot gjennomføringen av prosjektet
- Rettet mot deler av et prosjekt
  - Har programvaren nødvendig funksjonalitet
- Revisjoner kan holdes etter at et prosjekt er gjennomført.
- Det gjøres for å forvisse seg om at det som er avtalt, er utført, om lover og regler er fulgt og for å sikre at både gode og dårlige erfaringer blir tatt vare på. Slik kan de komme til nytte i fremtidige prosjekter.
- Revisjoner kan utføres av interne eller eksterne uavhengige revisorer.

Universitetet i Stavanger

# Effekten av gjennomgåelse og tester

Dette avsnittet viser noen data om effektiviteten av forskjellige teknikker vi bruker for å oppdage feil. Dataene er hentet fra *«Software engineering. Theory and practice» (Pfleeger, 1998)*

| ***Teknikk*** | ***Krav*** | ***Design*** | ***Koding*** | ***Dokumentasjon*** |
|---|---|---|---|---|
| Prototyping | 40 | 35 | 35 | 15 |
| Kravgjennomgåelse | 40 | 15 | 0 | 5 |
| Designgjennomgåelse | 15 | 55 | 0 | 15 |
| Kodeinspeksjon | 20 | 40 | 65 | 25 |
| Enhetstest | 1 | 5 | 20 | 0 |

Universitetet i Stavanger

# Effekten av gjennomgåelse og tester

- Tabellen er ordnet etter fasen som genererer feilen.
- Det vil si at feil i krav som blir oppdaget under enhetstest, blir plasser i kolonnen for krav. Tabellen viser andel feil som ble oppdaget i de ulike fasene med de forskjellige teknikkene. For eksempel ser vi at feil introdusert under koding, blir 65% oppdaget ved inspeksjon av koden og 20% ved enhetstest.
- Neste tabell viser hvor egnede de ulike teknikkene er til å finne feil i typiske produkter fra utviklingsprosessen.

Universitetet i Stavanger

# Effekten av gjennomgåelse og tester

| *Teknikk* | *Feil i krav* | *Feil i design* | *Feil i kode* | *Feil i dokumentasjon* |
|---|---|---|---|---|
| Gjennomgåelse | Brukbar | Utmerket | Utmerket | God |
| Prototyping | God | Brukbar | Brukbar | Ubrukelig |
| Testing | Dårlig | Dårlig | God | Brukbar |
| Bevis av korrekthet | Dårlig | Dårlig | Brukbar | Brukbar |

Egnede teknikker for oppdagelse av feil

Vi ser at gjennomgåelse er utmerket for å oppdage feil i design og kode, og at prototyping er best egnet for å oppdage feil i krav. Bevis av korrekthet er ikke omtalt i denne boken. Det dreier seg om å føre matematiske bevis for at programmet er korrekt.

# Kapittel 20
# Anskaffelser

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Anskaffelser og systemutvikling

- I systemutvikling er anskaffelser og anskaffelses-kontrakter et sentralt tema.
  - Systemutvikling ofte utføres av en part på oppdrag dra en annen part
  - De fleste systemutviklingsprosjekter har elementer av anskaffelser i seg
- Anskaffelser er et område der de fleste firma vil ha formelle krav, for eksempel til hvordan en anbudsinnbydelse skal foregå.
- I det følgende beskrives noen overordnede prinsipper, og en mulig framgangsmåte skisseres. Det er imidlertid viktig å sjekke hvilke regler som gjelder i det firmaet en arbeider for.

Universitetet i Stavanger

# Anskaffelsesavdelingen

- De fleste firma av en viss størrelse vil ha folk som er ansvarlig for firmaets anskaffelser, eller kan bistå i en anskaffelsesprosess.
- I beskrivelse av anskaffelsesprosessen har vi forutsatt at slike finnes og kan støtte utviklingsprosjektet. Vi har derfor skissert en arbeidsdeling mellom prosjektgruppen og anskaffelses avdelingen, denne arbeidsdelingenkan bli betraktet som beste praksis.
- En anskaffelses-avdeling vil da være profesjonelle på den kommersielle siden, mens system-utviklerne er profesjonelle tekniske fagfolk. En arbeidsdeling er utmerket.
- Finnes ikke en slik avdeling eller person, må en likevel prøve å gjennomføre anskaffelser på en ryddig måte, følge regler og spesielt passe på prosjektets uavhengighet i forhold til ulike leverandører.

Universitetet i Stavanger

# Sjekkliste for IT-kontrakter DND

Den Norske Dataforening har utgitt en sjekkliste for IT-kontrakter. Den ble til gjennom et prosjekt i Den Norske Dataforening, Østlandet våren 1996 og er laget som et hjelpemiddel for ansvarlig utforming og oppfølging av IT-kontrakter.

Sjekklisten anbefales som et hjelpemiddel ved utarbeidelse, kvalitetsikring og oppfølging av IT-kontrakter. Den inneholder :

- Oversikt over kritiske suksessfaktorer
  - Momenter som bør være tilfredsstilt før utarbeidelse kontrakten
- En oversikt over anskaffelsesprosessen
- Oversikt over det viktigste innholdet i en IT-kontrakt

Denne sjekklisten kan anskaffes ved å henvende seg til DND.

Universitetet i Stavanger

# Typer anskaffelser

Det er hensiktsmessig å skille mellom to typer anskaffelser

- **Enkle anskaffelser**

  Enkle anskaffelser forekommer hyppig i alle typer utviklingsprosjekt. Det dreie seg om

  - kjøp av verktøy
  - kjøp av konsulenttjenester

  Dette er en underordned del av prosjektarbeidet. I slike tilfeller må vi også ha kontrakter, og når vi skal utforme disse, bør vi søke bistand hos anskaffelses avdelingen. Denne type anskaffelser ligger utenfor det som videre behandles.

Universitetet i Stavanger

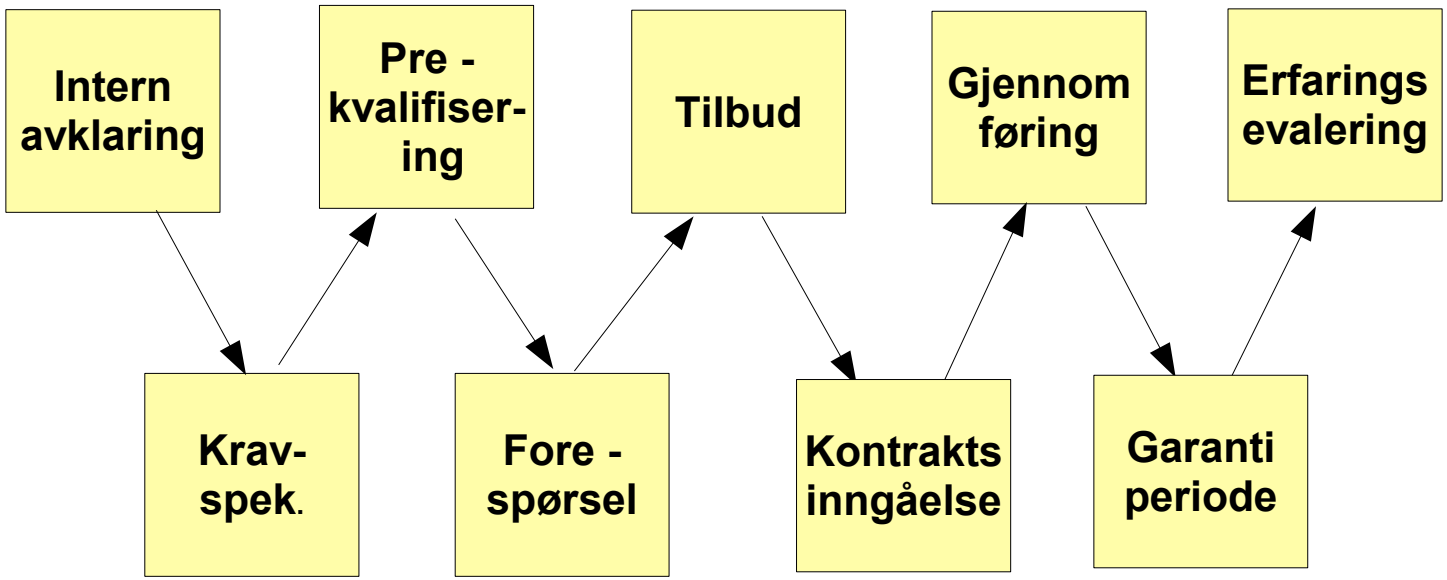
# Typer anskaffelser

- **Anskaffelser av produkter som skal utvikles/tilpasses i prosjektperioden**

  Eksempel på dette er når en leverandør utenfor prosjektet skal levere en komponent til systemet. Dette kan være en større eller mindre komponent, det kan til og med være et helt system. Uansett hvor stor del av systemet som anskaffes, allerede finnes i markedet, vil vi følge noenlunde samme prosesser fra vi starter arbeidet med anskaffelsene til vi aksepterer den ferdige leveransen. Denne prosessen er beskrevet i kap. 20.2

# En livsløpsprosess for anskaffelser

Målet for anskaffelsesprosessen er å sikre det riktige produktet anskaffes og at det har rett kvalitet. I tillegg skal vi ivareta de økonomiske interessene hos begge parter og sørge for å fremstå som profesjonelle og ryddige.




DND sin sjekkliste skisserer følgende faser og hovedaktiviteter i en anskaffelsesprosess.

Universitetet i Stavanger

# En livsløpsprosess for anskaffelser

- Et utviklingsprosjekt som skal anskaffe større eller mindre deler av løsningen, må gjennomføre et livsløp med tilnærmet de samme fasene som om hele systemet skal utvikles av prosjektgruppen.
- Alt arbeidet vi normalt gjør i et forstudium, må vi regne med å gjennomføre når (deler av) systemet skal anskaffes og det samme for innføringen.

Universitetet i Stavanger

# En livsløpsprosess for anskaffelser

- Det som blir annerledes, er arbeidet i analysefasen og arbeidet med utviklingen.
- Selv om komponentene vi skal kjøpe egentlig er ferdig utviklet, må vi regne med å spesifisere og gjennomføre noen tilpasninger for at systemet passe inn i eksisterende infrastruktur sammen med andre systemer.
- Vi kan aldri sløyfe disse fasene i arbeidet. I tillegg må vi utarbeide en del spesielle resultater som direkte angår anskaffelsen. For eksempel avtaledokumenter. Til gjengjeld kan vi forenkle eller sløyfe en del resultater som direkte angår utviklingen av den komponenten som skal kjøpes inn, for eksempel detaljert kravspesifikasjon, kode etc.

Universitetet i Stavanger

# En livsløpsprosess for anskaffelser

- Prosjektet bør planlegges med utgangspunkt i en livsløpsmodell.
- Vi må imidlertid gjøre nødvendige tilpasninger for å ta vare på anskaffelses-aktivitetene.

# Noen anskaffelses-aktiviteter

**Markedskartlegging**

- Før vi vurderer å anskaffe en vare eller tjeneste, er fornuftig å få en oversikt over hva som finnes i markedet.
- Det får vi gjennom markedskartlegging.
- Det er viktig at markedsundersøkelsen gjennomføres slik at ikke bedriften taper noen fordeler før en eventuell forhandlingsrunde.
- Det er derfor god praksis at en eventuell anskaffelses-avdeling gjennomfører markedskartleggingen.
- Hvis den gjennomføres av prosjektleder, bør anskaffelses- avdelingen godkjenne alle brev som går ut.

# Spesielle aktiviteter i et anskaffelses-prosjekt

- **Forstudium**
  - **Markedskartleggingen**
  - **Strategi for anskaffelser**
- **Analyse**
  - **Plan for anskaffelser integrert med prosjektplan.**
  - **Krav til produkt og leverandør**
  - **Tilbyderliste**
  - **Forespørsel til leverandør**
  - **Teknisk evaluering av tilbud**
  - **Kommersiell evaluering av tilbud**
  - **Avtaledokumenter – kontrakter**
  - **Resultat akseptansetest**

Universitetet i Stavanger

# Spesielle aktiviteter i et anskaffelses-prosjekt

- **Design**
  - **Oppfølging av leverandør**
  - **Plan for tilpassing og integrasjon med andre deler**
  - **Dokumentasjon av tilpasninger og integrasjon mellom produktet og resten av systemet eller infrastrukturen**
  - **Godkjenning av tilpasninger og integrasjon**
  - **Resultat akseptansetest**

# Spesielle aktiviteter i et anskaffelses-prosjekt

- Innføring
  - Erfaringsevaluering
- Forvaltning
  - Eventuell avtale om forvaltning og videreutvikling
  - Oppfølging i garantiperioden

Universitetet i Stavanger

# Strategi for anskaffelse

En anskaffelses-strategi skal avklare viktige overordnede prinsipper og valg knyttet til anskaffelsen. Den bør inneholde

- Strategi for valg av leverandør
- Anbefaling om kontrakts-form
- En vurdering av kompleksitet og risiko knyttet til anskaffelsene
- En beskrivelse av fremtidig bruk av produktet
- Ønsket detaljeringsgrad på de kravene som skal være grunnlag for å velge leverandør

# Strategi for anskaffelse

Skal utvikles i prosjektperioden,

Strategien bør si noe om eierskap til produktet

Eventuell videreutvikling av produktet og eventuell kommersialisering av produktet.

Vanlig framgangsmåte for valg av leverandør ved større anskaffelser er:

1. Det sendes en forespørsel til minst tre leverandører
2. En teknisk evalueringsgruppe vurderer tilbudene ut fra tekniske krav uten å kjenne pris
3. En kommersiell evalueringsgruppe vurderer pristilbudene og lager en endelig innstilling til prosjektgruppen om hva som bør velges (etter en samlet vurdering basert på den tekniske innstillingen)
4. Prosjektgruppen godkjenner valget. Det er vanlig å ha en regel om at en velger det tilbudet som har lavest pris blant de som tilfredsstiller de tekniske kravene.

Universitetet i Stavanger

# Strategi for anskaffelse

En slik framgangsmåte forutsetter:

- Det finnes tre aktuelle leverandører
- Vi er sikre på hva slags produkt vi ønsker

Vi bør ikke sende ut en forespørsel til leverandører vi vet ikke er aktuelle. De koster betydelige beløp for leverandøren å lage et tilbud, og det er derfor «uetisk» å sende forespørsel til leverandører som ikke vil komme i betraktning, bare fordi vi vil oppfylle kravet om minst tre leverandører.

Universitetet i Stavanger

# Strategi for anskaffelse

- Hvis vi ikke er sikre på hvilke produkt vi ønsker, kan vi ta utgangspunkt i produktet som leverandørene har, la leverandøren være med på å diskutere anvendelsene og mulige tilpasninger av disse.
- Hvis vi velger denne strategien, må vi være forsiktige slik at vi ikke for tidlig blinker oss ut en leverandør og mister muligheten til å forhandle om pris.
- Det beste er å innlede forhandlinger og diskusjoner med flere leverandører. Men da må vi spille med åpne kort slik at leverandøren vet hva som foregår. Hvis flere leverandører legger ned et betydelig arbeid i diskusjonene rundt produktet, må vi kanskje regne med å betale for dette.

# Krav til leverandøren og produktet

Kravene til leverandøren og til produktet som utvikles, er grunnlag for valg av leverandør, og de vil inngå i kontrakten som grunnlag for avtalen med leverandøren. Kravene må spesifiseres nøye

1. De er grunnlag for produktets kvalitet.
2. Mangelfulle spesifikasjoner av krav kan medføre at vi taper retten til å reklamere på kvaliteten til produkt
3. Mangelfulle spesifikasjoner kan også medføre at vi må inngå dyre tilleggs-kontrakter for å få det vi ønsker
4. Det er et vanlig at vi velger det tilbudet som har lavest pris og som oppfyller kravene. Hvis kravene er gale eller ufullstendig spesifisert, kan vi risikere å måtte velge bort det beste tilbudet, fordi et annet også oppfyller kravene og i tillegg har lavest pris.

# Krav til leverandøren og produktet

- Det kan være like galt å spesifisere kravene for detaljert.
- Det er et faktum at kravene til et IT-system både må og skal utvikles underveis.
- Hvis vi vil ha et godt produkt, må vi sørge for å gjøre rom for en slik utvikling også når vi anskaffer systemkomponenter.
- Vi må ikke spesifiserer kravene  så rigide og detaljerte innledningsvis at vi går glipp av det kreative. Men vi må spesifisere dem slik at vi får det vi vil ha.

Humphrey anbefaler at vi spesifiserer de fundamentale kravene

- ytelse, pålitelighet, brukervennlighet, brukerstøtte, funksjoner, grensesnitt mot andre systemer osv.
- presise krav det som skal dokumenteres.

Det kan være aktuelt å pre-kvalifisere leverandørene

# Tilbyderliste, forespørsel og evaluering av tilbud

- Det utarbeides en liste over leverandører som skal inviteres til å gi tilbud (tilbyder).
- Grunnlaget for denne lista er markedsundersøkelsen og pre kvalifiseringen av leverandørene.
- Denne listen bør godkjennes av en eventuell anskaffelses-avdeling. Det er vanlig å sende forespørsel til mellom tre og seks leverandører.

# Tilbyderliste, forespørsel og evaluering av tilbud

- Forespørselen til leverandørene utarbeides basert på kravene som er spesifisert.
- En forespørsel er et formelt dokument som bør utarbeides og sendes ut av anskaffelses- avdelingen.
- Det er god praksis å utarbeide en instruks til tilbyderne om hva hvordan tilbudet skal se ut. Og hvordan de skal forholde seg i tilbuds-fase, for eksempel hvem de skal kommunisere med.
- For offentlig sektor er det et krav at alle tilbyderne skal behandles likt. For privat sektor kan dette vurderes, men dette er den ryddige framgangsmåten.

# Tilbyderliste, forespørsel og evaluering av tilbud

- Etter at forespørselen er sendt ut, bør all kontakt med tilbyderne enten skje skriftlig, og med anskaffelses avdelingen etter faste prosedyrer.
- Alle spørsmål fra tilbydere skal stilles skriftlig, og alle spørsmål med svar distribueres til alle. Hensikten er å sikre at ingen av tilbyderne får fordeler fremfor de andre.
- I denne perioden er det god praksis at de som er involvert i utviklingsprosjektet unngår enhver kontakt med tilbyder.
- Det er vanlig at en bedrift har spesielle regler og prosedyrer knyttet til mottak av og åpning av tilbud. Dette håndteres av en eventuell anskaffelses-avdeling.

# Tilbyderliste, forespørsel og evaluering av tilbud

- Så tidlig som mulig bør det utpekes komiteer for teknisk og kommersiell evaluering.
- Det er god praksis å gjennomføre evalueringen så formelt og ryddig som mulig.
- Dette styres av anskaffelses avdelingen, og all kontakt med tilbyderne i evaluerings-perioden, for eksempel for avklaring, bør håndteres av denne avdelingen.

# Tilbyderliste, forespørsel og evaluering av tilbud

Oppsummert:

- Det er god praksis at prosjektleder og andre fagpersoner **ikke** tar imot invitasjon til lunsj, middag eller lignende fra tilbyderne
- Ikke innhenter eller diskuterer pris med noen av tilbyderne utenom den formelle forespørselen
- Ikke forteller noen av tilbyderne hvordan det ligger an med evalueringen
- Ikke forteller noen av tilbyderne hvilke andre tilbydere eller alternativer som vurderes

# Tilbyderliste, forespørsel og evaluering av tilbud

**Neppe best praksis**

Universitetet i Stavanger

# Oppfølging av anskaffelser

- En hensikt med anskaffelses kontrakter er at de skal være et godt instrument i gjennomføringen av anskaffelser.
- Den bør derfor avklare på hvilke måte anskaffelsen skal følges opp, og hvem som har ansvaret for oppfølgingen.

Universitetet i Stavanger

# Sjekkliste DND

I sjekklisten til DND finner vi følgende sjekkpunkter

- Er deltakerne kjente med alle relevante punkter i kontrakten
- Benyttes kontrakten løpende i gjennomføringen
- Foretas all gjennomføring og endringshåntering i henhold til kontrakten
- Er det utpekt ansvarlig for oppfølging av kontrakten
- Finnes det prosedyrer som skal styre oppfølgningen av kontrakten
- Håndteres avviksituasjoner i henhold til kontrakten

Universitetet i Stavanger

# Sjekkliste DND

I sjekklisten til DND finner vi følgende sjekkpunkter

- Følges arbeidet opp med hensyn på personell og roller
- Følges arbeidet opp med håndtering av problemer og behov
- Følges arbeidet opp med hensyn på avviksituasjoner
- Etableres resultater i henhold til akseptansekriterier
- Avklares uoverensstemmelser og løses disse fortløpende
- håndteres avtalen regelmessig i statusmøtene med hensyn på avviksituasjoner
- Er kontrakten oppfylt for begge parter

Universitetet i Stavanger

# Erfaringsevaluering

Det er god praksis å gjennomføre en evaluering av erfaringer med en anskaffelse. Det er spesielt følgende forhold som bør evalueres

- Hvor egnet var kontrakten – er det noe som bør endres ved neste anskaffelse
- Hvor egnet var anskaffelsesprosessen – er det noe som bør endres
- Hvordan var samarbeidet med denne leverandøren, og hvordan var leverandørens evne til å levere

Universitetet i Stavanger

# Erfaringsevaluering

I DND sin sjekkliste anbefales at følgende punkter evalueres.

- Er erfaringer fra kontraktsarbeidet oppsummert
- Er kravene oppfylt
- Hva er det økonomiske resultatet
- Hva er det kompetansemessige resultatet
- Hva er erfaringen fra konkurransen mellom tilbyderne
- Hvem fikk kontrakten og hvorfor
- Hva kan virksomheten gjøre for å informere samarbeidspartnere

Universitetet i Stavanger

# Erfaringsevaluering

- Er erfaringer fra gjennomføringen og garantiperioden oppsummert
- Er det forbedringsaktiviteter som bør gjennomføres
- Er formålet med IT-kontrakten oppfylt
- Hva kan forbedres i anskaffelsesprosessen
- Kan vi overføre erfaringene til andre internt
- Hvem skal delta i evalueringen – kunde, leverandør, ledelse, brukere etc.
- Finnes det sjekklister for evaluering

Universitetet i Stavanger

# Anskaffelseskontrakter

- ***Et spørsmål om tillit***

Tillit er grunnlaget for ethvert godt og effektivt kontraktsforhold. Spør vi en jurist vil han helt sikkert si at om vi ikke stoler på leverandøren, bør vi ikke inngå kontrakt med han. Et kontraktsforhold som ikke bygger på gjensidig tillit lykkes svært sjelden.

Universitetet i Stavanger

# Anskaffelseskontrakter

***Men er det tillit som preger IT-kontrakter? Hvordan vil vi oppføre oss som kjøper om vi ikke stolte på leverandøren. Antagelig ville vi***

- Insistere på en svært presis beskrivelse innledningsvis av hva som skal lages
- Insister på at arbeidet skal følge detaljerte standarder
- Insistere på hyppige og detaljert oppfølging underveis
- Legge mye arbeid i å definere detaljerte godkjennings-kriterier
- Presse leverandøren til å gi en fast pris på arbeidet
- Legge inn straffeklausuler i kontrakten i tilfelle leverandøren ikke klarer å levere som avtalt

Universitetet i Stavanger

# Anskaffelseskontrakter

Mange systemutviklingsprosjekter er mislykket. Det er lett å forstå at kjøperen av systemutvikling, mener han har behov for å beskytte seg selv og behov for å sikre at han faktisk får det han vil ha. Men hva er den beste måten å sikre seg på.

Vi husker at TKL filosofen Deming i sine 14 punkter tar opp forholdet til leverandørene. Her heter der:

- Slutt å godta det laveste anbudet
- Etabler en form for forpliktende samarbeid
- Bykk på tillit, ikke mistro

Universitetet i Stavanger

# Anskaffelseskontrakter

Vi finner også denne holdningen i ISO 9000:200 serien. Standarden sier:

> En organisasjon og dens leverandører er avhengig av hverandre, og et gjensidig fordelaktig samarbeid forbedrer begge parter sin evne til å skape verdier

Men kan dette gå bra? La oss se på hva som skjer hvis vi har en IT-kontrakt med detaljerte spesifikasjoner, avtale om fast pris og en stram tidsplan og straffeklausuler i tilfelle leverandøren ikke klarer å levere som avtalt. Hvordan ville vi oppført oss hvis vi var på leverandørsiden i en slik kontrakt? Vi ville satse på å slippe unna med så lite som mulig – gjøre akkurat det som var spesifisert i kontrakten og ikke en tøddel mer. Vi ville ikke ha noe incentiv til å levere et best mulig produkt.

Og så er det slik med programvare at vi sjelden vet på forhånd helt presist hva vi vil ha. Detaljerte krav til systemet må utformes underveis i prosessen, gjennom prøving og feiling, i et samarbeid mellom den som skal bruke systemet, og den som skal lage systemet.

Universitetet i Stavanger

# Hvordan bør da IT kontrakter utformes (Humphery 1989)

1. Vi bør ha tillit til at leverandøren er til å stole på og har tilstrekkelig kompetanse. Hvordan kan vi få det
   - Gjennom erfaring
   - Studere hva han har gjort før
   - ISO-sertifisert
2. Vi bør ha teknisk kompetanse til å følge opp og vurdere kvaliteten av det arbeidet som gjøres av leverandøren
3. I utformingen av kontrakten bør vi forutsette gjensidig tillit, og vi bør erstatte straffeklausuler med klausuler om belønning av godt arbeid
4. Vi bør legge vekt på å beskrive hva som skal gjøres, en plan for å gjøre det og akseptsanse-kriterier for å godkjenne det som er gjort.

Universitetet i Stavanger

# Hvordan bør da IT kontrakter utformes (Humphery 1989)

5. Vi aksepterer at de detaljerte kravene vil endre seg underveis og gir rom i kontrakten for at dette kan påvirke omfang, pris og leveranseplan.
6. Vi legger vekt på prosessen som leverandøren skal følge, spesielt gjennomgang og kvalitetskontroller.
7. Vi legger vekt på rutinene for samarbeid mellom leverandøren og oss som oppdragsgiver, spesielt en plan for dokumentasjon, rapportering og godkjenning.

Universitetet i Stavanger

# Hva inneholder en IT-kontrakt

Til slutt et eksempel på innholdet i en IT-kontrakt. Det er hentet fra et større firma i Norge, og gjelder for det som kalles entrepriseoppdrag. Det er oppdrag der leverandøren skal utføre et større selvstendig arbeid under egen ledelse og uten daglig tilsyn av kunden.

# Hva inneholder en IT-kontrakt

Denne kontrakten består av følgende dokumenter:

1. Bestilling, inneholder følgende
   - Navn på kundens kontaktperson
   - Arbeidsbeskrivelse. Her stilles det opp krav til planlegging og oppfølging av arbeid
   - Kompetansekrav til leverandørens personell
   - Omfang med timeramme som ikke må overstiges uten godkjennings
   - Arbeidssted
   - Navngiving av personell som er godkjent til å arbeide med oppdraget.
   - Faktureringsopplysninger
   - Signaturer

# Hva inneholder en IT-kontrakt

2. Kontraktsformularet Det har følgende punkter
   - Parter i kontrakten
   - Oversikt over kontraktens dokumenter, med prioritet i tilfelle uoverens-stemmeleser mellom dem
   - Leverandørens plikter
     - Planlegging og oppfølging
     - kvalitet
     - personellets kompetanse
     - rapportering
     - plikt til å holde seg informert om og følge lover og foreskrifter
     - plikt til å hindre misbruk av kundens IT-ressurser
   - Betaling

Universitetet i Stavanger

# Hva inneholder en IT-kontrakt

- Arbeidsbeskrivelser med henvising til de enkelte bestillingene
- Priser
  - timepriser og hva det inkluderer
  - normal arbeidstid
  - regler for reiser
  - grunnlag for avregning
- Kontraktsperiode og mulighet for utvidelse
- Avbestillingsregler
- Signering

# Hva inneholder en IT-kontrakt

3. Generelle kontraktsvilkår, med følgende punkter
   - Definisjoner
   - Kvalitetssikring
   - Helse, miljø og sikkerhet
   - Betaling
   - Opphavs og eiendomsrett
   - Hemmeligholdelse av informasjon
   - Skadesløshetsholdelse
   - Mislighold
   - Forsikring

Universitetet i Stavanger

# Hva inneholder en IT-kontrakt

- Engasjement av underleverandør
- Force majeure
- Godtgjørelse fra tredjepart
- Tillegg
- Endringer
- Transport av kontrakten
- Varsler
- Rett og tvister

# Kapittel 21
# Måling

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Hvorfor måle

Ingeniører måler for å skaffe seg kunnskap. En programvare-ingeniør ønsker svar på følgende spørsmål:

- Hvor stort er programmet?
- Hvor godt er programmet?
- Hvor lett er systemet å vedlikeholde?
- Hvor mange feil er det i programmet?
- Hvor lenge bør en teste?
- Hva koster testingen?
- Hvor lang tid vil det ta å utvikle et lignende system?
- Hva vil det koste å utvikle programmet?
- Hvor mange personer trenger en til utviklingen?

Universitetet i Stavanger

# Hvorfor måle

Ingeniører er vant til å foreta målinger, men dette er lite vanlig innenfor programutvikling. Dette er i ferd med å endre seg. Også programvareingeniøren har behov for å kvantifisere sin forståelse av prosesser, produkter og organisasjoner.

Det er vanskelig å kvantifisere hva som skal måles under programutvikling. Dessuten kan det være vanskelig å utnytte målte størrelser. Interessen for dette er økende og i dette kapittelet vil vi se på:

- Hva kan måles
- Hva kan ikke måles
- Hva bør måles
- Hvordan kan man måle?
- Hva skal resultatene brukes til?

Universitetet i Stavanger

# Måleteori

Ved måling prøver en finne et tall som uttrykker en egenskap ved en ting eller et objekt. Vi har

- Fartsmåler
- Termometer

Målingen gir oss informasjon som vi kan reagere på. Farten kan justeres, PC-en kan få mer kjøling osv.

Vi assosierer altså erfarte egenskaper med et tall, eller egentlig med et symbol. Tallet blir et symbol på den erfarte egenskapen. Dette kan vi formulere abstrakt, og det er det måleteorien gjør.

I matematikken beskrives assosiasjoner ved hjelp av funksjoner/relasjoner. Relasjoner beskriver assosiasjoner mellom elementer i mengder.

Universitetet i Stavanger

# Måleteori

**Definisjon 1:**

*La A være en mengde av fysiske eller empiriske objekt. La B være en mengde av formelle objekter (tall). En måling er en en til en avbildning av A inn i B*

$$\mu : A \rightarrow B$$

Universitetet i Stavanger

# Måleteori

Et fysisk eller empirisk objekt kan være:

- En person
- Programsystem
- Prosess

I en mengde av fysiske objekter kan flere objekter ha samme egenskap.

*Måling er prosessen med å knytte symboler, vanligvis tall, til en egenskap vi tillegger en gjenstand.*

Eksempler

- Måler hastighet på biler som passerer
  - Mengden **A** er biler som passerer
  - Mengden **B** er alle hastighetene til bilene som passerer.

Universitetet i Stavanger

# Måleteori

I stedet for å observere hastigheten kan vi observere

- fargen til bilene, da blir mengden B en mengde av farger

Som programvareingeniører kan vi være interessert i kompleksiteten til forskjellige programvaremoduler.

- Ved måling kan vi observere og erfare at forskjellige moduler har ulik kompleksitet ved at vi har en regel for hvilke tall som uttrykker kompleksiteten til programvaremodulen.
- Vi kunne også gradere kompleksiteten etter en alfabetisk skala. Q er lav kompleksitet mens R, S, T kunne være uttrykk for økende kompleksitet.

Vi må bruke en regel når vi tilordner symboler. Vi må vite hvordan vi skal velge de symbolene som skal representere en egenskap. Det sikrer objektivitet og gir resultater som kan sammenlignes med resultater fra lignende målinger.

Universitetet i Stavanger

# Måleteori

- **Nominalskala**. Hvis det eneste vi kan observere. Er om to er like eller ikke, da kan vi bare klassifisere.
  - Mengden A er kildekode
  - Mengden M er type programmeringsspråk, C++, Java, Pascal eller Fortran
- **Ordinalskala.** Vi er kun i stand til å observere at det er forskjell mellom objekter.
  - Ved å bruke ulike metoder kan vi si noe om modulariteten til et program. Men dersom vi ikke kan si hvor mye mer modulært et program er enn et annet, da måler vi langs en ordinal skala. Vi er istand til å rangere programmer langs denne skalaen etter modularitet.

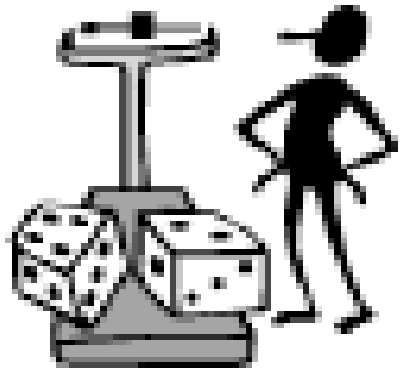
Universitetet i Stavanger

# Måleteori

- **Intervallskala.** Når en måling gjør oss i stand til å observere intervallet mellom målinger, måler vi langs en intervallskala. Hvis vi klarer å uttrykke forskjellen i modularitet mellom programmer, bruker vi en slik skala.
- **Forholdsskala.** Hvis vi i tillegg til å kunne si noe om størrelsen på et intervall også kan si noe om forholdet mellom intervallene, måler vi langs en forholdsskala. For modularitet betyr dette at vi kan si at en modul er dobbelt så modulær som en annen, som igjen kan være tre ganger så modulær som en tredje modul. For at dette skal være mulig, må skalaen ha et absolutt nullpunkt
  - Temperaturskala i Celsius og Fahrenheit er intervallskala
  - Kelvins temperaturskala er en forholdskala

# Måleteori

- **Absoluttskala.** Her er det bare en måte å gi en måling på. Når en teller linjer i et program, så kan man (hvis er enig om definisjonen av linje) bare komme fram til ett resultat.



Universitetet i Stavanger

# Måleteori

**Definisjon 2**:

Et relasjonssystem er et ordnet tuppel

$$(S, rel_1, \ldots, rel_n, op_1, \ldots, op_m)$$

hvor

S er en mengde med objekter

$rel_1, \ldots, rel_n$ er relasjoner på objektene i S

$op_1, \ldots, op_m$ er binære operasjoner på objektene i S

Universitetet i Stavanger

# Måleteori

Noen eksempler:

Vi ser på et utvalg av programvaremoduler og observerer kompleksiteten.

- Mengden S er programvaremodulene
- Vi kan se ulike relasjoner mellom objektene
  - $rel_1$ En binær relasjon mellom to objekter
  - $rel_2$ En terner relasjon mellom tre objekter
- I dette eksempelet kan vi sammenligne to moduler og finne ut om de er like komplekse
- Vi har et sett av operasjoner
  - $op_1$ kan være en binær operasjon som kombinerer to moduler og lager en ny modul som har høyere kompleksitet. Denne kompleksiteten er det mulig å observere og den har sin plass i observasjons-mengden

Universitetet i Stavanger

# Måleteori

Eksempelet

- S er samling med planker
- Vi har ulike relasjoner
  - $rel_1$ Vi kan si om to planker er like lange
  - $rel_2$ Vi kan si om den første planken er større enn den siste
- Vi har et sett med operasjoner
  - $op_1$ Sett sammen to planker
  - $op_2$ Halvere en planke

Universitetet i Stavanger

# Definisjon 3

$$A=(S_A, relA_1, \ldots, relA_n, opA_1, \ldots, opA_m)$$

Et relasjonsystem med fysiske eller empiriske objekter.

$$B=(S_B, relB_1, \ldots, relB_n, opB_1, \ldots, opB_m)$$

Et relasjons system med formelle objekter (f.eks tall)

$\mu : S_A \rightarrow S_B$ Er en måling

$(A, B, \mu)$ Er en skala hvis og bare hvis

$$relA_i(a_{(i1)}, \ldots, a_{(ik)}) \Leftrightarrow relB_i(\mu(a_{(i1)}), \ldots, \mu(a_{(ik)}))$$

$$\mu(a\, opA_j\, b) = \mu(a)\, opB_j\, \mu(b)$$

for alle verdier av $i$ og $j$ og for alle

$$a, b, a_{(i1)}, \ldots, a_{(ik)} \in S_A$$

Universitetet i Stavanger

# Måleteori

Definisjonen sier følgende:

- For hver relasjon definert på fysisk objekt er det en tilsvarende relasjon definert mellom formelle objekt
- Relasjonene på fysisk og formelle objekt er ekvivalente
- Utføres en operasjon på fysiske objekter så kan en utføre tilsvarende operasjoner på formelle objekter og omvendt.

Universitetet i Stavanger

# Måleteori

For eksempelet med planker betyr dette:

- Finner en at en planke er lenger enn en annen planke ved observasjon, skal det samme komme til uttrykk ved tallene som representerer målingen. Tallet som representerer den største planken er større enn tallet som representerer den minste planken. Hvis plankene er like skal tallene være like.
- Skjøter vi sammen to planker og måler lengden, skal tallet for lengden være det samme som tallet vi får dersom vi måler de to plankebitene og adderer disse tallene.

Universitetet i Stavanger

# Måleteori

- Tekststrenger kan sammenlignes på samme måte som planker
- Dersom vi slår sammen to moduler er kompleksiteten til den sammenslåtte modulen ikke nødvendigvis summen kompleksiteten til de to modulene vi startet med.
- Her er noe av problemet knyttet måling av programvare, hvordan finne målinger som kan brukes.
- Det vi har beskrevet er unikhets-prinsippet for skalaer.
- Vi kan foreta transformasjoner av skalaer til nye skalaer.
- Hvis en slik transformasjon gjør at unikhets-prinsippet fremdeles holder, så er begge skalaene like gode.

# Eksempel på transformasjon mellom skalaer

Eksempel:

Gå mellom Celsius og Fahrenheit

$$\overline{M} = \alpha M + \beta \wedge \alpha > 0$$

Universitetet i Stavanger

# Måleteori

Generelt har vi:

- For nominalskalaer er lovlig transformasjon å bytte symboler
- For ordinalskalaer er alle monotone økende funksjoner lovlige skalaer
- For intervallskalaer fører alle transformasjoner av lineære grupper, som vist i eksempelet foran, til lovlige skalaer
- For forholdskala holder denne transformasjonen

  $\overline{M} = \alpha M \wedge \alpha > 0$
- For absolutt kala er ingen transformasjon mulig.

Universitetet i Stavanger

# Definisjon 4

$(A, B, \mu)$ Er en skala hvor B er mengden av de reelle tall.

$\gamma(A)$ Er mengden av alle reelle tall som er en måling av objekter A.

Avbildningen:

$$t : \mu(A) \rightarrow B$$

En gyldig transformasjon hvis og bare hvis triplettene

$(A, B, t \circ \mu)$ Er en skala.

Universitetet i Stavanger

# Definisjon 5

$(A, B, \mu)$ Er en skala hvor B er mengden av de reelle tall.

Et utsagn om målingene

$$\mu(A)$$

Er meningsfull hvis og bare hvis sannhetsverdien bevares etter anvendelse av en gyldig transformasjon.

*Målinger av lengde i meter og tommer tilfredsstiller disse kravene.*

Universitetet i Stavanger

# Målinger og metrikk

- Uformelt kan en si at måling er å sette tall på egenskaper til fysiske gjenstander, hendelser og prosesser.
- Vi bruker også begrepet måling om resultatet av prosessen med å måle.
- I den sammenheng er en måling (resultatene) vanligvis tall, *et måltall*, som angir hvor mange enheter av en måleenhet, målingen (prosessen) har resultert i.
- Hvis vi har målt distansen mellom to steder til å være 6 km, er *6 måltallet km måleenheten*.
- Vi kan si at det er en relasjon mellom de to stedene, og distanse er en egenskap ved relasjonen mellom dem.

Universitetet i Stavanger

# Målinger og metrikk

- I faglitteraturen om programvare og måling støter vi på begrepet metrikk.
- Mange forskere og fagfolk bruker metrikk som samlebegrep for alt som kan måles i forbindelse med programvareprodukter og prosesser.
- Men måleteoretisk er en metrikk en spesiell gruppe målinger som tilfredsstiller neste definisjon:

Universitetet i Stavanger

# Metrikk

$R^p$ Mengden av ikke negative reelle

Avbildning

$m : A \to R^p$ er en metrikk hviss:

$m(x,y) = 0 \Leftrightarrow x = y$

$m(x,y) = m(y,x) \quad \forall x, y \in A$

$m(x,z) \leq m(x,y) + m(y,z) \quad \forall x, y, z \in A$

Universitetet i Stavanger

# Målinger og metrikk

Intuitivt sier definisjonen

- To like objekter har ingen avstand
- En får samme avstand om en måler fra x til y eller fra y til x
- Avstanden mellom to steder er mindre eller lik avstanden en oppnår når en legger turen om et tredje sted.

En metrikk er en spesiell måling hvor resultatet er en distanse, eller lengde.

I litteraturen om programvare og måling finnes det flere forskjellige definisjoner på hva metrikk er. I «The project manager's guide to software engineering's best practices» (Christnsen & Thayer, 2001) har vi denne definisjonen.

Universitetet i Stavanger

# Definisjon 7

*En metrikk er en beregnet eller komponert indikator basert på to eller flere målinger*

Etter denne metoden er en måling noe som gjøres direkte, men en metrikk er en beregning basert på målingene.

Eksempel:

Vi kan telle antall feil vi finner i en programvaremodul, og vi kan telle antall kildekodelinjer modulen består av. Dette er målinger fordi vi gir direkte tall til egenskaper ved modulen. Disse tallene kan brukes til å gi en indikasjon på kvaliteten til modulen ved å beregne en metrikk som vi kaller feiltetthet, og som beregnes som antall feil pr tusen kildekodelinje.

En annen definisjon finner vi i «Innsamling og validering av måledata» ( Wedde, 1997)

Universitetet i Stavanger

# Definisjon 8

*En metrikk defineres som en karakteristikk av et produkt eller en prosess*

- Selv om metrikk og måling har forskjellige definisjoner, brukes de i praksis om hverandre og som synonymer.
- Vi møter metrikk som både er målte og en beregnede egenskaper ved programvaren.
- Det kan ernyttig å klassifisere metrikker i :
  - Produktmetrikker
  - Prosessmetrikker

Universitetet i Stavanger

# Metrikker

- Ved hjelp av en produktmetrikk prøver en å finne et uttrykk som sier noe om kompleksiteten og modulariteten til produktet knyttet opp mot produktivitet
- Metrikker for produktivitet kan brukes til å si noe om hva en er i stand til å prestere, og til estimering.
- Ikke alle metrikkene er like aktuelle i dag, men de er tatt med for å vise utviklingen til programvaremetrikker

Universitetet i Stavanger

# Produktmetrikker

- Antall kildekodelinjer
- Antall kildekodelinjer kan brukes som metrikk for størrelsen til et program. Dette er en metrikk som er lett å måle og målingen kan automatiseres. Med utgangspunkt i antall kildekodelinjer kan vi beregne andre metrikker:
  - Produktivitet
    - Kildekodelinjer pr månedsverk
- Tar en vare på slike data kan de brukes til å estimere ressursbehovet for framtidige utviklingsprosjekter, eller man kan sammenligne produktiviteten i forskjellige prosjekter og ved å bruke forskjellige programmeringsspråk.

Universitetet i Stavanger

# Produktmetrikker

Det er utviklet flere estimeringsmodeller som bruker antall kildekodelinjer som utgangspunkt. Den mast kjente er COCOMO. Standardverket er «Software engineering economics» (Boehm, 1981).

Det er flere problemer knyttet til denne metoden

- Hva skal telles?
  - Alle linjer
  - Eksekverbare linjer
  - Skal kommentarer telle
- Kan ulike programmeringsspråk sammenlignes med denne metoden?
  - Blir mest produktiv ved lavnivå-språk

Universitetet i Stavanger

# Produktmetrikker

**Produktivitetsparadoks**

Et program skrevet i assembly resulterte i 5000 kildekodelinjer. Det tok 28 uker å utvikle programmer. Et program med samme funksjonalitet skrevet i et høynivåspråk resulterte i 1500 kodelinjer. Utviklingen tok 20 uker. Produktiviteten blir:

- Assembly
  - 714 kildekodelinjer pr månedsverk
- Høynivå-språk
  - 300 kildekodelinjer pr månedsverk

Men kostnaden ved å utvikle programmet i et høynivåspråk er lavere, og utviklingstiden kortere.

Universitetet i Stavanger

# Produktmetrikker

- Antall kildekodelinjer er dårlig egnet for å sammenligning mellom språk.
- Kan brukes til å sammenligne produktivitet mellom prosjekter hvor samme språk er bruk
- Bruk av kildekodelinjer som basis for å kunne estimere tid og ressurser for et nytt prosjekt kan være problematisk.
- Antall kildekodelinjer er først kjent etter at en er ferdig
- Antall linjer kode en tror systemet vil ende opp med må estimeres.
- En er på jakt etter metrikker som det er mulig å finne i starten av et utviklingsprosjekt
- Den mest kjente og antagelig mest brukte er *funksjonspoeng*. Denne metrikken kommer vi tilbake til

Universitetet i Stavanger

# Produktmetrikker

- Halstead var en av de første som formulerte metrikker for programvare.
- Hans viktigste bidrag hevdes å være at han problemstillingen fram.
- Nytten av de metrikkene han definerte, er omstridt,
- Halstead tar utgangspunkt i størrelser som det er lett å trekke ut av kildekoden til et program
  - $n_1$ = antall forskjellige operatorer
  - $n_2$ = antall forskjellige operander
  - $N_1$ = det totale antall operatorer
  - $N_2$ = det totale antall operander

Universitetet i Stavanger

# Produktmetrikker

Operatorer er språkets nøkkelord, mens operandene er variabler og konstanter i programmet.

Vokabularet til programmet defineres som:

- $n = n_1 + n_2$

Lengden til programmet defineres som:

- $N = N_1 + N_2$

Programmets volum defineres som:

-

Universitetet i Stavanger

# Produktmetrikker

- Denne formelen er et uttrykk for programmets kompleksitet.
- Måleenheten er bit.
- Dette er de grunnleggende ligningene. Halstead mente det var mulig å finne estimater for $n_1$ og $n_2$ allerede under spesifikasjonen av programvaren.
- Ved å trekke inn erfaringsdata utledet han flere ligninger. Disse mente han kunne brukes til å beregne ressursene som trengtes for å utvikle et program, og tiden det ville ta.

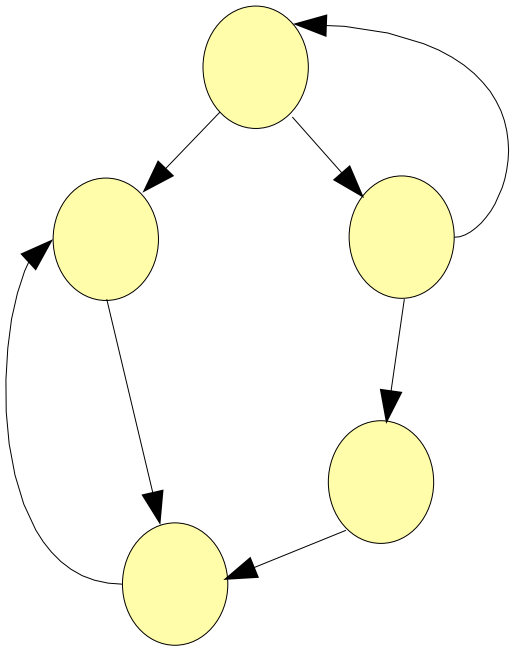
Universitetet i Stavanger

# Produktmetrikker

- Halstead ser kun på operatorer og operander.
- Mange hevder at det ikke gir et godt nok uttrykk for kompleksiteten til et program, fordi det sier ingenting om strukturen i programmet.
- McCabe tar for seg grafen til programmet og beregner det syklomatiske kompleksitetstallet ved:
  - V(G) = antall kanter – antall noder + 2

Universitetet i Stavanger

# Produktmetrikker



Et syklomatisk kompleksitetstall som er 10 eller større, er en indikasjon på at en programmodul er for kompleks og har stor sannsynlighet for å inneholde feil.

Hewlett Packard har et automatisk opplegg for å finne kompleksitetstallet og bruker det systematisk i sin kvalitetssikring.

Alle programmer med et syklomatisk komplejsitetstall høyere enn 10 blir betraktet som potensielt problematiske og blir vurdert og undersøkt nærmere.

Det syklomatiske kompleksitetstallet er også utgangspunkt for å finne testsituasjoner ved hvitboksing.

Universitetet i Stavanger

# Produktmetrikker

Funksjonspoeng

- Denne metrikken gir en språkuavhengig karakteristikk av programmer.
- Den ble lansert av Allen Albrect og hans kolleger i IBM på slutten av 1970 tallet etter at de hadde fått i oppdrag å se på måling og metrikker. De formulerte en metrikk knyttet til funksjonaliteten til programmer. Funksjonaliteten uttrykkes i;
  - Antall funksjons poeng (FP)
- Den beregnes ved hjelp av tall som en kan finne før programmet er implementert.
- FP kan derfor brukes til å estimere utviklingstiden og ressursbehovet når man har nok erfaringsdata som viser eventuelle sammenhenger mellom FP og det man vil

Universitetet i Stavanger

# Produktmetrikker

Funksjonaliteten til et program er knyttet til følgende elementer, som kan telles:

- **Antall bruker-inndata:** Inndata er en samling fakta eller tall som behandles i en transaksjon. Et eksempel er samlingen av data i et skjermbilde.
- **Antall bruker-utdata:** Utdata er en samling fakta eller tall som er resultatet av transaksjonen. Et eksempel er en rapport som blir produsert.
- **Antall brukerforespørsler:** En brukerforespørsel er en direkte forespørsel fra en bruker og som gir umiddelbar svar fra programmet i form av utdata i for eksempel et skjermbilde.
- **Antall interne logiske filer:** En intern logisk fil inneholder logisk relaterte data eller kontrollinformasjon. Et eksempel kan være en tabell i en relasjonsdatabase som blir oppdatert av programmet.
- **Antall eksterne grensesnittfiler:** En ekstern grensesnittfil er en gruppering av logisk relaterte data eller kontrollinformasjon som brukes av programmet, men som vedlikeholdes av en annen applikasjon. Et eksempel er en tabell i en relasjonsdatabase som bare blir lest av programmet, og som oppdateres av andre program.

Universitetet i Stavanger

# Produktmetrikker

- Det finnes detaljerte regler og retningslinjer for hva som skal telle, og hvordan man skal telle. «The Internal Function Point User Group» (IFPUG) gir ut og vedlikeholder disse.
- En vekt for kompleksiteten assosieres med hvert element. Den kan ha tre verdier. Verdien bestemmes etter en vurdering av hvor kompleks elementet er. Elementet kan være enkelt, middels eller komplekst. Verdiene som brukes er vist i tabellen under.

Universitetet i Stavanger

# Produktmetrikker

| *Parameter* | *Enkel* | *Middels* | *Kompleks* |
|---|---|---|---|
| **Antall inndata** | 3 | 4 | 6 |
| **Antall utdata** | 4 | 5 | 7 |
| **Antall forespørsler** | 3 | 4 | 6 |
| **Antall filer** | 7 | 10 | 15 |
| **Antall eksterne grensesnitt** | 5 | 7 | 10 |

Universitetet i Stavanger

# Produktmetrikker

I tillegg trekkes inn 14 faktorer som sier noe om de generelle egenskapene til det systemet som utvikles. Hver faktor har en bestemt vekt bestemt av dens betydning. Vektfaktorene er vist i de to neste tabellene.

| *Ingen innflytelse* | *Liten innflytelse* | *Moderat innflytelse* | *Middels innflytelse* | *Signifikant innflytelse* | *Essensiell innflytelse* |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

Universitetet i Stavanger

# Produktmetrikker

1. Datakommunikasjon. Sendes eller mottas data over en kommunikasjonslinje
2. Distribuert databehandling. Skal systemet bestå av distribuerte komponenter som utveksler data?
3. Ytelse. Er ytelse kritisk?
4. Datamaskinressurser. Må programmet lages for å kjøre på maskinvare med begrensede ressurser?
5. Transaksjonshastighet. Må det taes spesielt hensyn til mengden av transaksjoner som kan komme i visse perioder.
6. Online inntasting av data. Kreves det online inntasting av data?
7. Sluttbrukereffektivitet. Er det spesielle faktorer i forbindelse med brukervennlighet som det må tas hensyn

Universitetet i Stavanger

# Produktmetrikker

8. Online oppdatering. Blir interne logiske filer oppdatert online?
9. Prosesseringskompleksitet. Er det spesiell vanskelig prosesseringslogikk som det må tas hensyn til?
10. Gjenbrukbarhet. Er det krav til at programmet må utvikles slik at det kan gjenbrukes av andre applikasjoner?
11. Installasjonsvennlighet. Påvirker overgangen fra tidligere systemer utviklingen av dette systemet?
12. Driftsvennlighet? Er det spesielle krav til oppstart, sikkerhetskopiering og rekapitulasjon som påvirker utviklingen av systemet?
13. Flyttbarhet. Må systemet lages for å kunne kjøre på forskjellige steder og gor forskjellige bruksorganisasjoner?
14. Endringsvennlighet. Er det krav at det må tas hensyn til at prosseseringslogikk og datastruktur kan bli endret?

Universitetet i Stavanger

# Produktmetrikker

Formelen som gir funksjonspoengene er:

$$FP = \left(\sum_{i=1}^{5} Tall_i \cdot Vekt_i\right) \cdot \left(0{,}65 + 0{,}01 \cdot \sum_{j=1}^{14} F_j\right)$$

Her er $Tall_i$ antallet som er talt opp for et element. $Vekt_i$ et tall for den vekt elementet har og $F_j$ graden av påvirkning fra de enkelte generelle systemfaktorene.

Universitetet i Stavanger

# Produktmetrikker

Objektpoeng  en veiet sum av

- Antall separate skjermbilder
- Antall rapporter som produseres
- Antall moduler skrevet i et tredjegenerasjonsspråk og som trengs i tillegg for å få programmet til å fingere

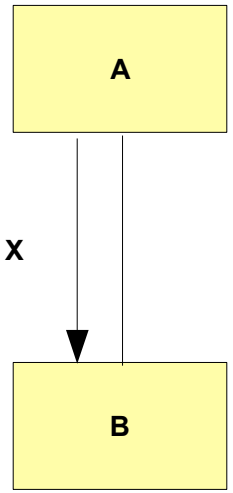
Brukes for å karakterisere programmer som i hovedsak settes sammen av komponenter ved bruk av et aplikasjonsverktøy.

De forskjellige tallene gis en vekt etter en vurdering av kompleksiteten.

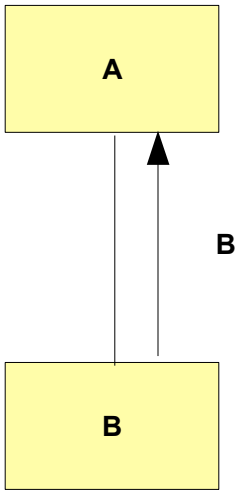
- enkle skjermbilder 1 poeng,
- moderate 2
- komplekse 3 poeng.
- rapporter gis tilsvarende 2,5 eller 8
- tredjegenerasjonsmodul teller 10 poeng

# Metrikk for informasjonsflyt, IF4

Det er flyt hvor en modul kaller en annen og overfører argumenter. Lokal flyt har vi også når en modul returnerer et resultat. (se figuren under)




Modul med argument

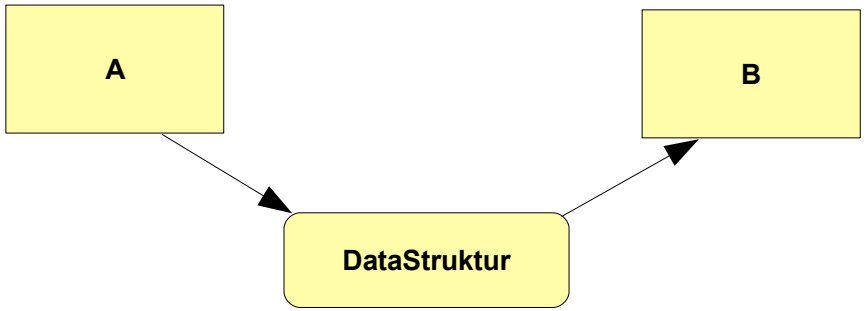



Modul med resultat

- <A,X,B>
- <A,B,B>

# Metrikk for informasjonsflyt, IF4

Global informasjonsflyt mellom moduler via en global datastruktur er vist på neste figur

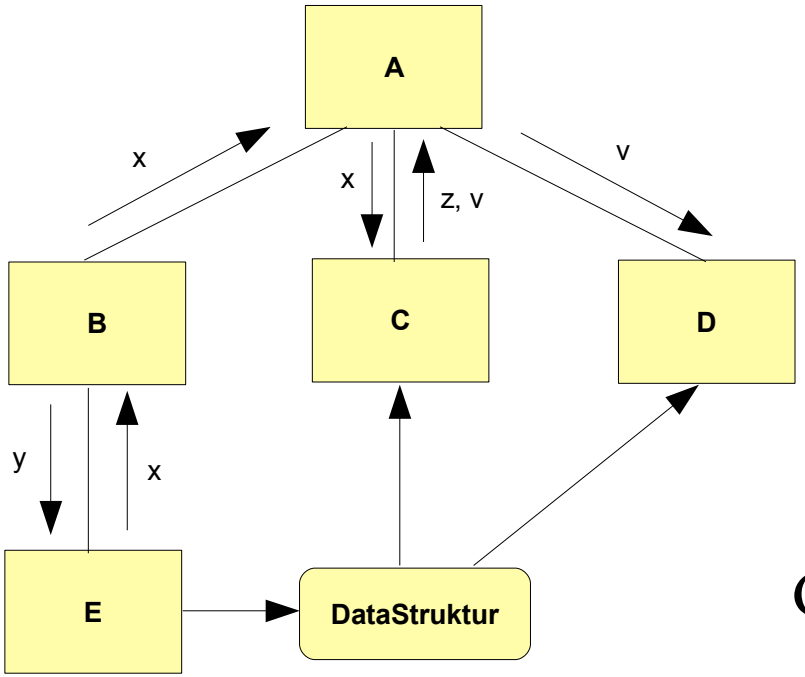



Dette kan skrives: <A, DataStruktur, B>

- **Fan in:** Dette er antallet informasjonsflyt som går inn i en gitt modul
- **Fan out:** Dette er antallet informasjonsflyt ut av en gitt modul

# Metrikk for informasjonsflyt, IF4

IF4 finnes med utgangspunkt i antallet fan out og fan in.
Neste figur forklarer.




Lokal informasjonsflyt:

- <A, x, C>
- <A, v, D>
- <B, x, A>
- <B, y, E>
- <C, z, A>
- <C, v, A>
- <E, x, B>

Global informasjonsflyt

- <E, DataStruktur, C>
- <E, DataStruktur, D>

Universitetet i Stavanger

# Metrikk for informasjonsflyt, IF4

Resultatene settes opp i en tabell

| *Modul* | *Fan in (FI)* | *Fan out (FO)* | *FI*FO* | $IF4_i$ |
|---|---|---|---|---|
| A | 3 | 2 | 6 | 36 |
| B | 1 | 2 | 2 | 4 |
| C | 2 | 2 | 4 | 16 |
| D | 2 | 0 | 0 | 0 |
| E | 1 | 3 | 3 | 9 |
| **Total** | | | | 65 |

Universitetet i Stavanger

# Metrikk for informasjonsflyt, IF4

Det vi har gjort er å

- Liste opp antall informasjonsflyt inn Fan in og antall informasjonsflyt ut Fan out
- Beregne produktet FI*FO
- Beregne IF4 = $(FI*FO)^2$
- IF4 for hele systemet er summen av alle $IF4_i$

Universitetet i Stavanger

# Metrikk for informasjonsflyt, IF4

- IF4 skal fortelle noe om kompleksiteten til systemet og enkeltmoduler.
- Høy IF4 indikerer problemer.
- Enkeltmoduler med høy IF4 er kandidater for nærmere undersøkelse og redesign.
- Ved å kvadrere produktet av fan in og fan out blir moduler med problemer lettere å peke ut.
- Målingen er ordinal.
- Den sier ikke noe om graden av kompleksitet, bare at modul A er den mest komplekse og at D er den minst komplekse i innbyrdes ordning:
  - D < B < E < C < A

Universitetet i Stavanger

# Metrikk for informasjonsflyt, IF4

- IF4 kan brukes til å stille kritiske spørsmål til design. Er det nødvendig med så mange forbindelser til enkelte moduler?
- Moduler med mange forbindelser gjør sannsynligvis mye. Det strider mot prinsippet om god design.
- Prinsippet sier vi skal søke å lage et system som består av løst koblede og enfoldige moduler som hver ideelt sett bare gjør en ting.
- Enfoldige moduler er lettere å forstå, lettere å endre og kandidater for gjenbruk.

Universitetet i Stavanger

# Prosessmetrikker

- I det foregående har vi sett eksempler på Produkt metrikker,
- Vi er også interessert i metrikker som kan brukes til å karakterisere prosessen vi bruker for å framstille produktene. Det trenger vi når vi skal bestemme dugeligheten til prosessen, og når vi skal vurdere effekten av forskjellige forbedringstiltak.
- Det er en rekke egenskaper med prosessen vi kan være interessert i å måle.
- Noen egenskaper måles direkte. Andre måler vi indirekte ved å måle på produktene fra prosessen.
- Tabellen under har noen eksempler på prosessmetrikker.
- De mest vanlige metrikkene har med tid, kostnad, ressurser, produktivitet og kvalitet å gjøre.

Universitetet i Stavanger

# Prosessmetrikker

| *Metrikk/indikator* | *Definisjon/måleenhet* |
|---|---|
| Utviklingstid<br>• planlagt: totalt og pr. fase<br>• virkelig: totalt og pr. fase | Timer, dager, uker, måneder, år |
| Ressursbruk<br>• planlagt: totalt og pr. fase<br>• virkelig: totalt og pr. fase | Timeverk, dagsverk, ukeverk, månedsverk, årsverk |
| Produktivitet | kildekodelinjer/måmed,<br>kildekodelinjer/dagsverk,<br>funksjonspoeng/uke<br>funksjonspoeng/årsverk |
| Kvalitet | feil/kildekodelinje, feil/funksjonspoeng<br>IF4/objekt |
| Testkvalitet | Antall tester godkjent |
| Problemer | Antall problemrapporter<br>Antall uløste problemer |
| Kundetilfredshet | Andel tilfredse kunder, Antall nye kunder |

Universitetet i Stavanger

# Om data

- Dataene representerer faktaene vi trenger.
- Metrikkdata er derfor de verdiene vi knytter til en metrikk. Til daglig bruker vi begrepene om hverandre.
- Data kan klassifiseres som
  - målbare
  - tellbare
- Målbare data avleser vi ved hjelp av et måleinstrument.
  - tid brukt i designfasen.
- Slike data er aldri helt eksakte
- Men forskjellige personer kan få tilnærmet samme resultat.
- Tellbare data er eksakte og resulterer i heltall.
- Ulike personer gjør tellingen vil de komme fram til samme resultat,
  - forutsatt at de er enige om hvordan de skal gjøre tellingen.

Universitetet i Stavanger

# Om data

- Data kan også være objektive eller subjektive.
- Når en måling av objektive data utføres av forskjellige personer, vil de få samme resultat.
- Hvis det er målbare data blir ikke resultatet eksakt det samme, men vil falle innenfor forutsigbare statistiske grenseverdier.
- Subjektive data krever vurdering. Resultat vil variere fra person til person.
- Er subjektive data har noen verdi.
- Det har de, spesielt hvis flere eksperter brukes til å komme fram til enighet om vurderingen.
- Subjektive data samler vi inn når det ikke er mulig å foreta en direkte måling fordi den er for dyr eller vanskelig å gjennomføre.

Universitetet i Stavanger

# Om data

- Data er ***valide*** når de uttrykker det vi har til hensikt å måle.
- Data er ***pålitelige*** hvis samme data blir resultatet av gjentatte målinger under samme betingelser.
- Validitet og pålitelighet er ikke det samme.
- Data kan godt være pålitelige uten å være valide, fordi målingen gjentas med den samme systematiske feil. Se på ligningen:

$$M = T + s + e$$

- M er observert verdi
- T er virkelig verdi
- s er systematisk feil
- e er er tilfeldig feil

Universitetet i Stavanger

# Om data

- 368
- 0.475
- 8.21
- 0.00982
- 20.4

Når man skal telle antall signifikante siffer starter man fra venstre og hopper over null frem til første siffer som ikke er null. Null mellom andre siffer telles. Det spiller ingen rolle hvor desimaltegnet er plassert. Hvorfor treller vi inn dette med signifikante siffer? Det er fordi det får betydning i beregningene. Et resultat er aldri mer nøyaktig enn usikkerheten i det siste signifikante siffer. Hvis man bruker takk med forskjellig antall signifikante siffer i beregninger, er det tallet med færrest signifikante siffer som bestemmer nøyaktigheten.

Universitetet i Stavanger

# Om data

- Den systematiske feilen kan skyldes feil innstilling av måleinstrumentet eller at den eller de som bruker måleinstrumentet eller teller data gjør en feil.
- Tilfeldige feil skyldes egenskaper ved måleinstrumentet som brukes. Middelverdien er vanligvis null. Feilen kan reduseres ved å gjenta målingen og beregne middelen.
- Tall som er et resultat av målinger (med instrumenter), angis med antall signifikante siffer. Usikkerheten ligger i det siste signifikante siffer. Disse tallene har alle tre signifikante siffer

Universitetet i Stavanger

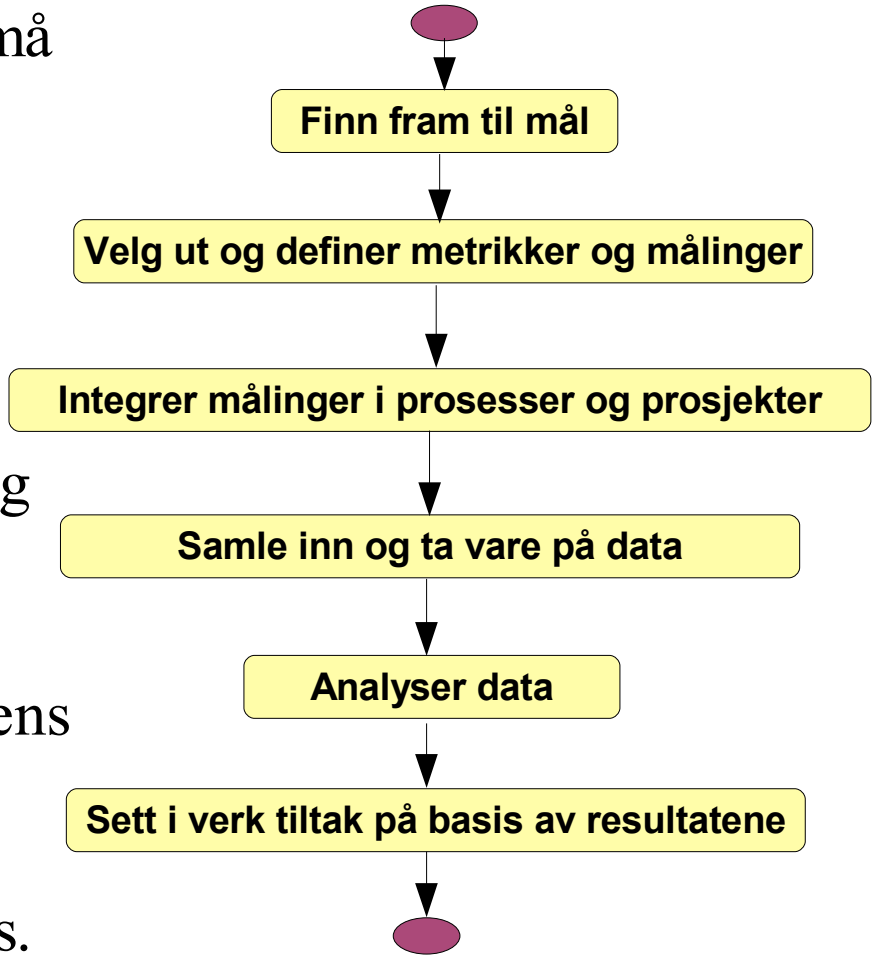
# Om data

Disse reglene gjelder:

- Ved addisjon eller subtraksjon av tall med forskjellig antall signifikante siffer runder man av til det antall signifikante siffer som er i det tallet med færrest signifikante siffer
- Ved multiplikasjon må alle tall rundes av slik at antall signifikante siffer for alle tall blir det samme som det med færrest signifikante siffer. Dette gjøres før regneoperasjonen starter.
- Hvis det i beregningen inngår både eksakte og ikke eksakte tell, er det de ikke eksakte tallene som bestemmer antall signifikante siffer.

# Måleplaner

- Alle seriøse virksomheter må ha opplegg for måling.
- Det gjelder uansett bransje fordi overlevelse på sikt krever stadig økende produktivitet og tilfredsstillelse av kunder og andre interessepartnere.
- Måling må etableres med utgangspunkt i virksomhetens overordnede mål og inkluderes i alle prosjekter. En måleprosess må på plass.



# Kapittel 22
# Verktøy

Terje Kårstad

Universitetet
i Stavanger

Universitetet i Stavanger

# Verktøy

Gode verktøy er en forutsetning for å kunne gjøre godt arbeid. Arbeid med prosessforbedring er ikke noe unntak. Det dreier seg ikke om fysisk redskap, men om nyttige teknikker for

- Å identifisere, analysere og forstå problemer
- Å analysere og presentere data
- Å velge ut og prioritere forbedringstiltak

Vi vil nå se på et utvalg av verktøy. Det er verktøy det er nyttig å ha i verktøykassen når vi arbeider med prosessforbedring. Dette er et begrenset utvalg, det er mange flere å velge mellom. De fleste verktøyene har sitt opphav i Japan. De er konkrete resultater av TKL-bevegelsen.

# Verktøy

***To grupper verktøy***.

Begge gruppene inneholder syv verktøy hver.

Tallet syv har en spesiell symbolsk betydning i Japan.
Samuraiene var utstyrt med syv redskaper når de drog i strid.

- Gruppe 1
  - De syv kvalitetskontroll verktøyene

    årsak-virkning-diagram, tids-diagram, flytskjema, Pareto-diagram, sprednings-diagram, histogram, kontroll-diagram
- Gruppe 2
  - De syv styrings og planleggingsverktøyene

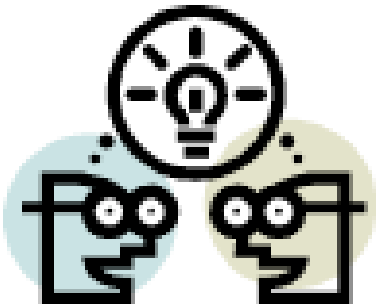
    slektskaps-diagram, relasjons-diagram, tre-diagram, presentasjons-matrise, matrise-prioritets-diagram, prosess-beslutnings-diagram, nettverks-diagram

Universitetet i Stavanger

# Kreative Verktøy

- Gode ideer er nødvendige i alt forbedringsarbeid.
- Vi trenger ideer for å forstå problemer og å finne løsninger.
- Mange av verktøyene vi skal bruke, og som beskrives i dette kapittelet har gode ideer som utgangspunkt.
- Arbeid med kvalitet og forbedring starter med ideer. Det betyr igjen at vi må få fram ideer.
- Kreativitet er nødvendig, men kreativitet må stimuleres.
- Til det finnes det flere verktøy.

# Idé dugnad



- Idédugnad er en gruppeteknikk.
- En gruppe mennesker skal i et positivt og tillitsfullt miljø stimulere hverandre til å få fram mange gjerne ville ideer på kort tid.
- Ingen kritikk eller diskusjoner av ideer er tillatt under genereringen. Det gjøres etterpå. Da skal ideene systematiseres og prioriteres.

Universitetet i Stavanger

# Regler Idé dugnad

- Personene som skal delte, må ha kunnskap og erfaring innen problemområdet. De må kunne delta aktivt.
- Det er ikke tillatt å kritisere eller diskutere ideer mens genereringen foregår. Spørsmål for å oppfatte om en idé er forstått, er tillatt.
- Legg inn pauser
- Kvantitet er viktigere enn kvalitet.
- Noter ned alle ideer.
- Ideer kan kombineres. Det er lovlig å bygge på andres ideer.
- Ideene tilhører gruppen.

Universitetet i Stavanger

# Trinnene i Idé dugnad

1. Forklar reglene
2. Definer problemet. Det kan gjerne formuleres som et spørsmål
3. Sett i gang idégenereringen.
   - Strukturert

     Hver enkelt presenterer etter tur en idé. Det kjøres flere runder, og det er lov å stå over.
   - Ustrukturert

     Det er fritt fram. Den som får en idé lanserer den umiddelbart
4. Dokumenter ideene. Man kan bruke klistrelapper som en fester på en tavle eller vegg. Alternativt kan ideene skrives på en tavle eller flippover. Sørg for at alle ideene blir notert.
5. Gå gjennom ideene for å sikre at alle har forstått hva de betyr, og om noen egentlig uttrykker det samme og kan kombineres eller fjernes.
6. Grupper og prioriter ideene. Avhengig av hensikten med idédugnaden vil ideene kunne være input til andre verktøy.

Universitetet i Stavanger

# Idé dugnad

- Idédugnad er også kjent under begrepet
  - ide-muldring, ide-spruting og engelsk «brainstorming»
- En ustrukturert Idédugnad kan lett føre til at noen få personer dominerer.
- Strukturert Idédugnad gi færre og ikke så ville ideer.

# Skriftlig Idé dugnad trinnene

I store trekk følger man de samme trinnene som for Idédugnad:

1. Definer problemstillingen. Skriv på en tavle, eller hvis det brukes kort, skrives det på hvert kort.
2. Deltakerne skriver ideer på kort eller tavler
3. Deltakerne kan føye til ting på andres kort eller på tavlen. Poenget er å få fram nye ideer eller videreutvikle ideer gjennom kombinasjonseffekter
4. Sorter og prioriter ideene.

**Kommentar**

Skriftlig Idédugnad «brainwriting», har den fordelen at det er lettere å få fram detaljerte ideer.

Universitetet i Stavanger

# Skriftlig Idé dugnad trinnene

- Av og til kan problemstillingen som skal belyses være av en slik art at det er nødvendig at deltakerne sikres en viss anonymitet.
- Problemstillingen kan være kontroversiell og konfliktfylt.
- Det er viktig å belyse problemet og få fram gode ideer til løsninger.
- Men kreativiteten kan lett bli hemmet.
- I slike situasjoner kan en bruke Crawford slip-metoden.

# Idé dugnad Crawford slip-metoden

- Dette er en variant av skriftlig Idédugnad.
- Metoden sikrer a flest mulig deltar i idégenereringen, ikke bare de mest fremfuse og taleføre.
- Metoden bruker små kort.
- Kortene sirkulerer ikke mellom deltakerne for tilføyelser.
- Ingen åpen vurdering gjøres til slutt.
- Kortene samles  inn, og en person lager en rapport.
- Deretter drøftes ideene for å oppnå enighet om hva som videre skal gjøres.

**Kommentar**

Metoden kan også brukes når store mengder informasjon blir generert.

# Nominell gruppeteknikk

**Trinnene**

1. Problemene defineres.
2. Ideer genereres og skrives på kort. Disse kalles idékort. Hver idé får sitt kort.
3. Ideene listes opp på en flippover og gies en identifikator. Det kan være en bokstav
4. Ideene diskuteres kort for å avdekke uklarheter og å fjerne like eller beslektede ideer.
5. Hver deltaker avgir stemme for ideen. Det skjer ved at hver enkelt gir fem ideer poeng fra 1 til 5. Den ideen som vurderes som best eller viktigst, får 5. Hver deltaker har et rangeringskort hvor de rangerte ideene føres opp med identifikator og poeng.
6. Rangeringskortene samles inn. Poengene føres opp på flippoveren.
7. Poengene summeres og vinneren er den ideen som har høyest poensum.
8. Det åpnes for en diskusjonsrunde hvor nye momenter kan trekkes inn, hvor det rett og slett dykker opp nye ideer.
9. Endelig rangering gjøres ved en ny runde gjennom trinnene 5,6 og 7

Universitetet i Stavanger

# Nominell gruppeteknikk

**Kommentar**

- Det finnes en variant hvor avstemmingen gjøres etter en slags utslag-metode. Hver deltaker får et stemmekort.
- Det førers opp så mange ideer som en vil.
- Kortene samles inn, og stemmene for hver idé registreres.
- Så plukkes de ideene som har fått en stor del av stemmene ut.
- Det kan være de som har fått mer enn 50%.
- Nye runder kjøres inntil en tilslutt ender opp med et lite antall ideer, to til fire stykker.

Universitetet i Stavanger

# Slektskapsdiagram

- Slektskapsdiagram er resultatet fra en metode som brukes for å gruppere store en mengde med informasjon.
- Denne er kommet fram ved ideedugnad. Det dreier seg om å gruppere og å sortere slike ting som
  - ideer
  - meninger
  - brukerkrav
  - problemer
- Grupperingen skjer etter naturlige relasjoner.
- Antallet grupper må ikke overstige ti.
- Arbeidet med slektskapsdiagram skal stimulere til kreativ tenkning slik at nye sider av et problem kommer fram.

Universitetet i Stavanger

# Slektskapsdiagram

**Trinnene**

Man starter med å samle en gruppe kompetente mennesker i et rom med en tavle. Deretter følger en disse trinnene

- Problemet som skal analyseres, skrives øverst på tavlen. En kort diskusjon avklarer om alle har forstått problemstillingen.
- Idédugnad brukes til å fram rikelig med ideer til løsning av problemet. Ideer skrives på selvklebende lapper. Disse festes på tavlen i tilfeldig orden. Avklar misforståelser og rett opp uklare utsagn.
- Nå skal lappene med ideer sorteres i naturlige grupper. Det gjøres i stillhet ved at deltakerne rusler rundt og flytter lapper. Etterpå diskuterer en resultatet og gjør eventuelle korreksjoner. Grupper av ideer som hører sammen, rammes inn. Deretter lages en overskrift over hver gruppe. Piler mellom gruppene kan brukes til antyde sammenhenger mellom gruppene.
- Diagrammet analyseres, og videre arbeid med ideene planlegges. Diagrammet gir den nødvendige oversikt for planleggingsarbeidet.

Universitetet i Stavanger

# Slektskapsdiagram

**Kommentarer**

- Slektskapsdiagrammer er kjent under flere navn
  - KJ-diagram (Jiro Kawakita), affinitets-diagram
- Slektskapsdiagram inneholder gjerne et femti-talls ideer, men det er ikke uvanlig med flere.
- Antall grupper er fem til ti.
- En kan i planleggingsarbeidet rangere gruppene etter viktighet.
- Det er vanlig å bruke en tredelt skala.

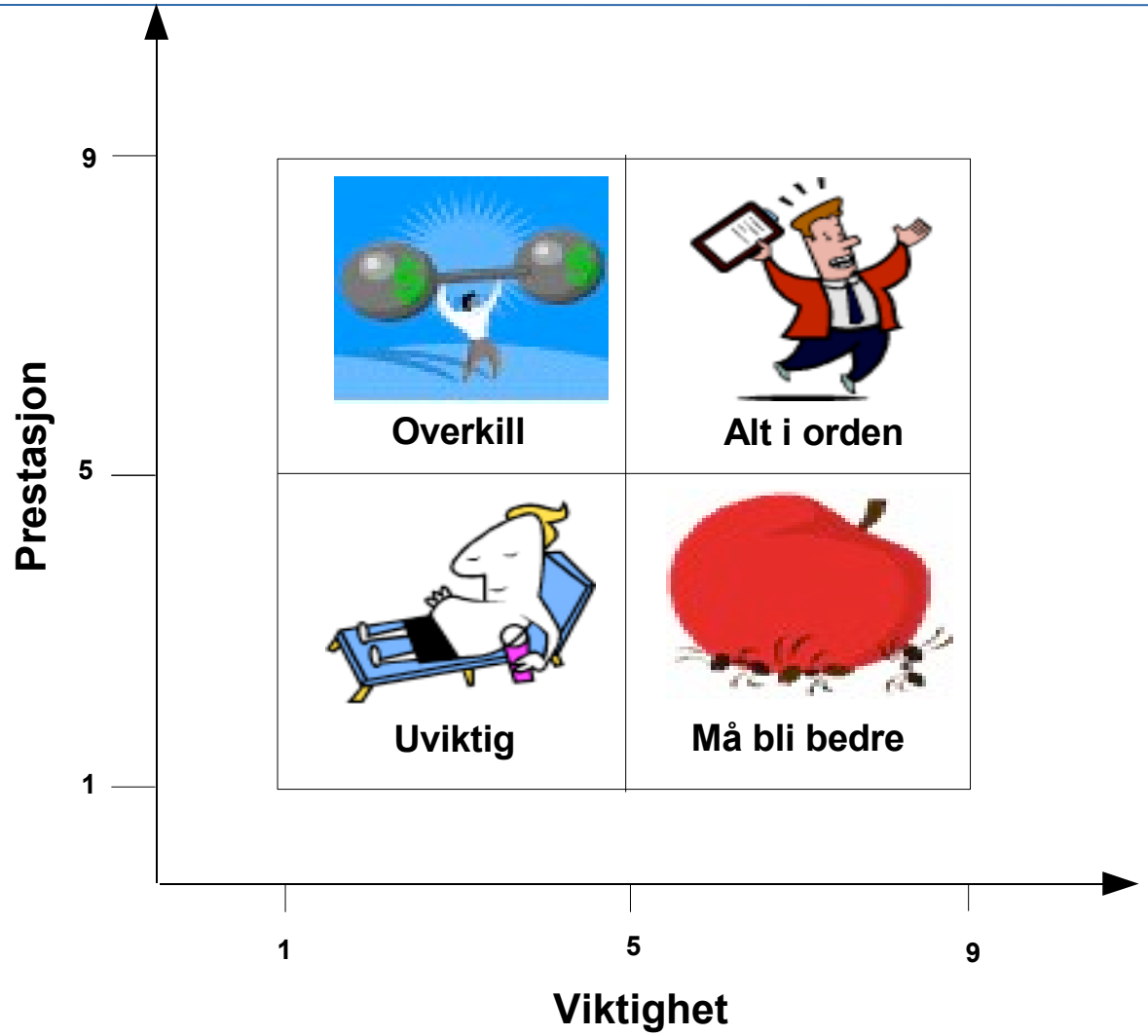
Universitetet i Stavanger

# Verktøy for problemanalyse

## *Matrisediagrammer*

- Matrisediagrammer finnes i ulike former.
- Det brukes for å vise sammenhenger mellom ting og som hjelpemiddel når tiltak skal prioriteres.

# Prestasjonsmareise

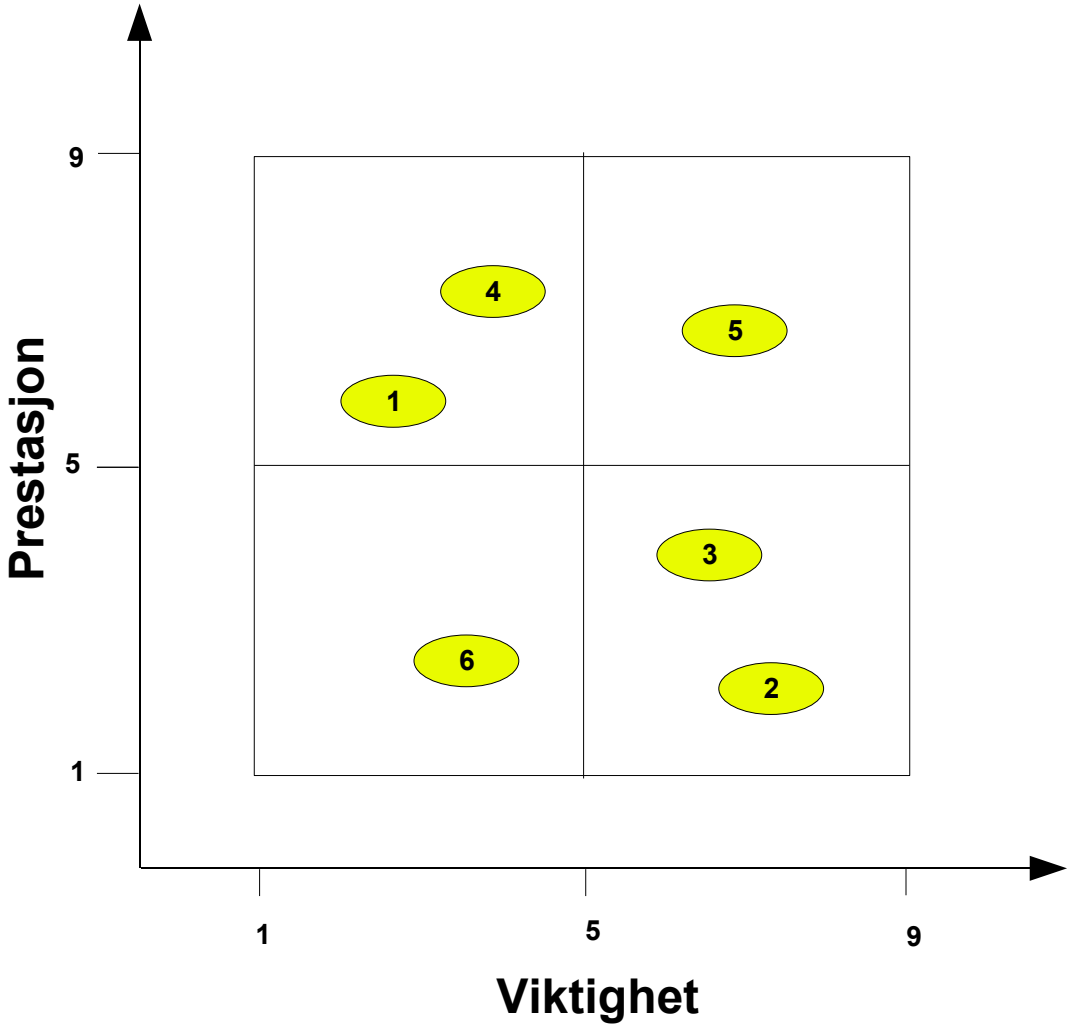


# Prestasjonsmatrise

Hver akse har en gradering for betydning. Skalaen går fra 1 til 9. De fire rutene i matrisen betyr:

1. Uviktig. Prestasjonen er lav, men har heller ikke særlig betydning
2. «Overkill» Skyter spurv med kanoner. Prestasjonen er høy, men har liten betydning
3. Må bli bedre. Området vurderes som viktig, men prestasjonen er dårlig. Det betyr at i dette området må det satses forebyggende
4. Alt i orden. Høy viktighet god prestasjon. Etter som betydningen er stor, må området voktes, slik at ikke prestasjonen blir dårligere. Også her kan en gjøre forbedringer. En skal aldri være helt fornøyd. Men dette området prioriteres etter område 3

# Risikoanalyse

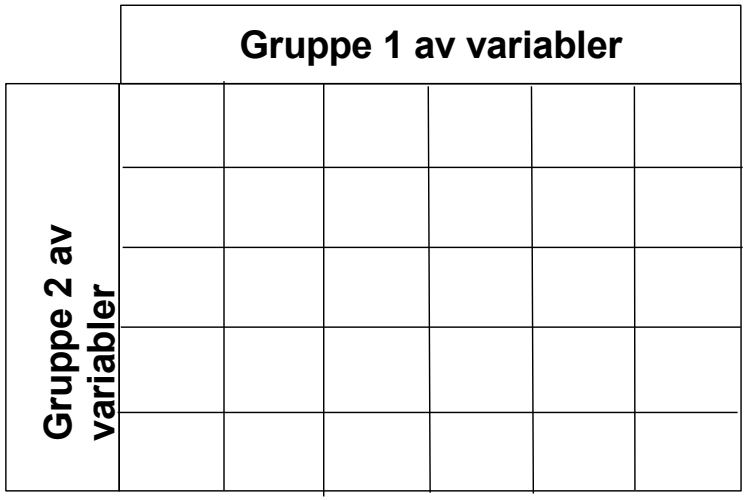


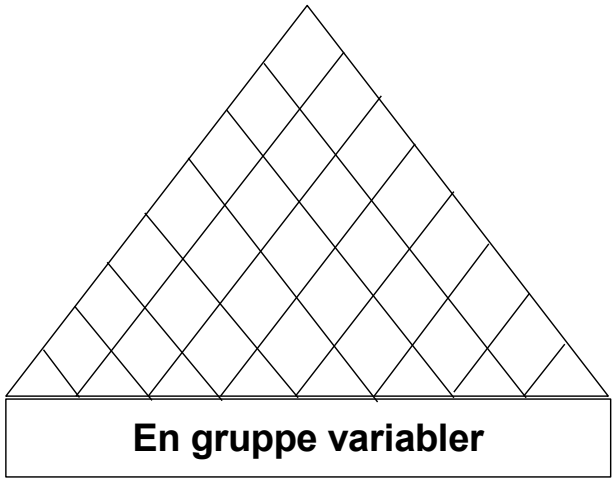
Universitetet i Stavanger

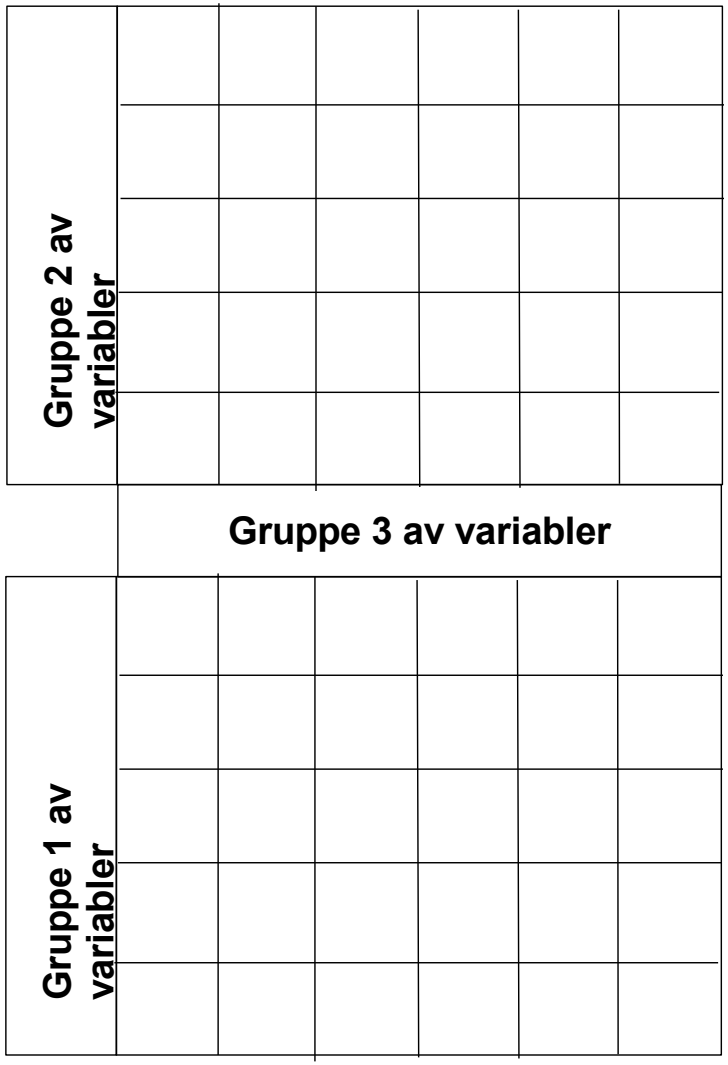
# Sammenlignings-matriser

- Denne typen matriser brukes når vi ønsker å vise forbindelser mellom ting.
- Forbindelsene kan dreie seg om årsaks-sammenhenher eller konflikter.
- Matriser finnes i forskjellige former.
- Formen bestemmes av hvor mange ting som skal sammenstilles eller settes opp mot hverandre.
- Neste figur viser tre typer.
- Takform gir egen-relasjon mellom gruppe variabler.
- L form relaterer to grupper og variable,
- T form tre grupper variabler.

# Sammenlignings-matriser



# Eksempel L form

| | Egenskap 1 | Egenskap 2 | Egenskap 3 | Egenskap 4 | Egenskap 5 | Egenskap 6 |
|---|---|---|---|---|---|---|
| Bruker 1 | 1 | | | | | |
| Bruker 2 | | | 3 | | | |
| Bruker 3 | | | 9 | | | |
| Bruker 4 | | | | | | |
| Bruker 5 | | | | | | |

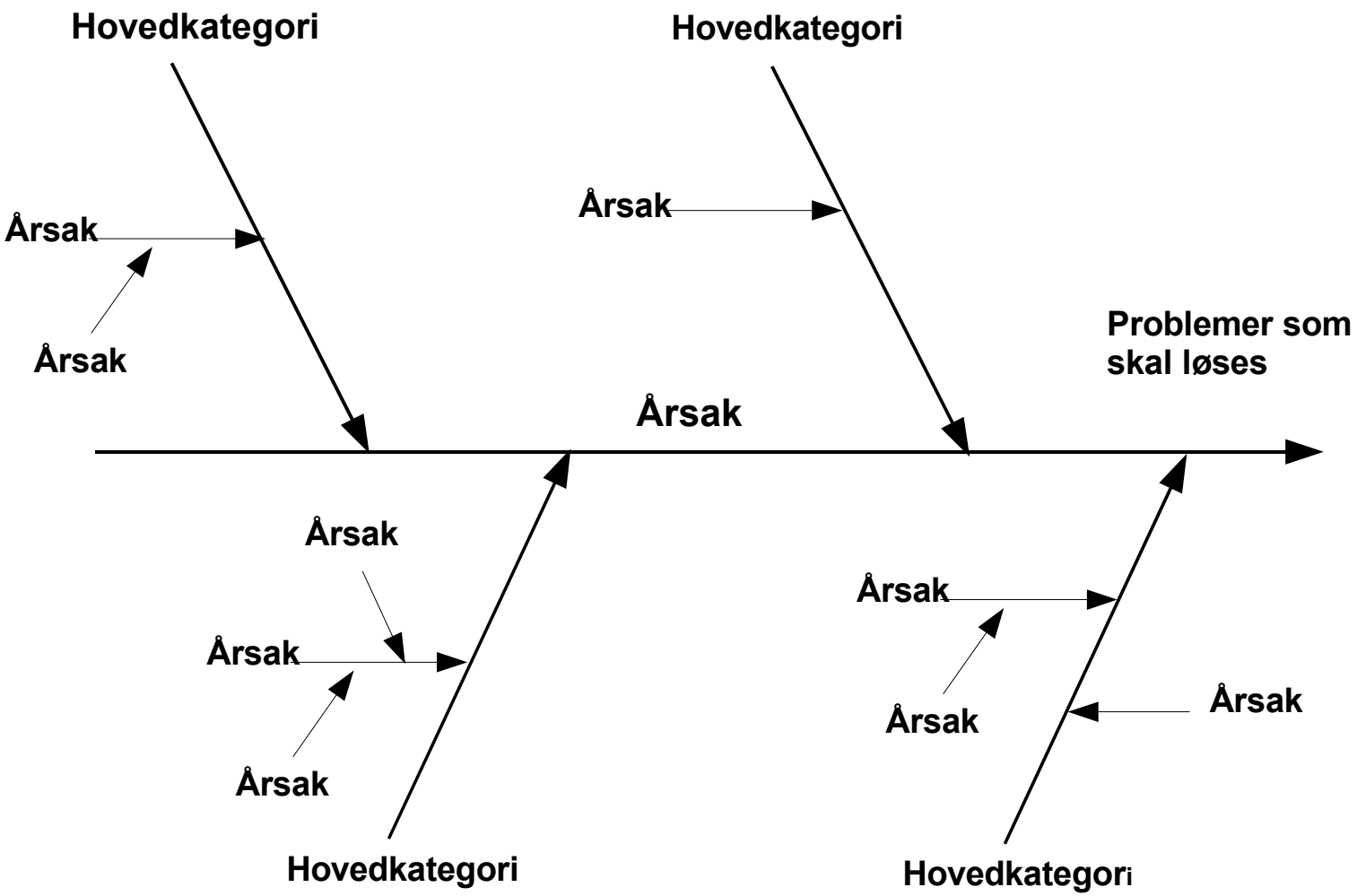
- Egenskapene 2, 4, 5 og 6 tilfredsstiller ingen brukerkrav
- Bruker 4 og 5 får ingen krav tilfredsstille

Universitetet i Stavanger

# Årsak virkning diagram

- Diagrammet brukes sammen med en teknikk for å finne fram til mulige årsaker til et problem eller en hendelse.
- Et ledd i denne prosessen er en ide-dugnad.
- Nytten størst når arbeidet gjøres i en gruppe med personer som har kunnskap om det problemområdet som skal undersøkes.
- Men teknikken og diagrammet kan også brukes av enkeltpersoner.

# Årsak virkning diagram



Universitetet i Stavanger

# Framgangsmåte

- En måte er å starte med å identifisere hovedkategoriene
- Finne nye årsaker under hver kategori.
- Idédugnad brukes for å finne hovedkategorier og årsaker under hver hovedkategori.
- En annen måte er å starte med idédugnad for å få fram mange ulike årsaker.
- En finner fram til hovedkategorier,
- Årsakene grupperes under hovedkategorier og på nivåer under.
- Til slutt tegnes diagrammet.

# Framgangsmåte

**Trinnene**

Uansett hvilke metode en bruker for å tegne diagrammet, følges disse trinnene:

1. Klarlegg problemet eller hendelsen dere skal finne fram til
2. Finn fram til årsaker ved hjelp av idédugnad
3. Tegn på en flippover, eller på en stor tavle.
4. Analyser diagrammet og finn fram til de viktigste årsakene som dere vil prioritere å finne løsning på

Universitetet i Stavanger

# Kjent barn har mange navn

- Årsak-virkning-diagram er også kjent under navnet fiskebeinsdiagram.
- Et annet navn er Ishikawa-diagram

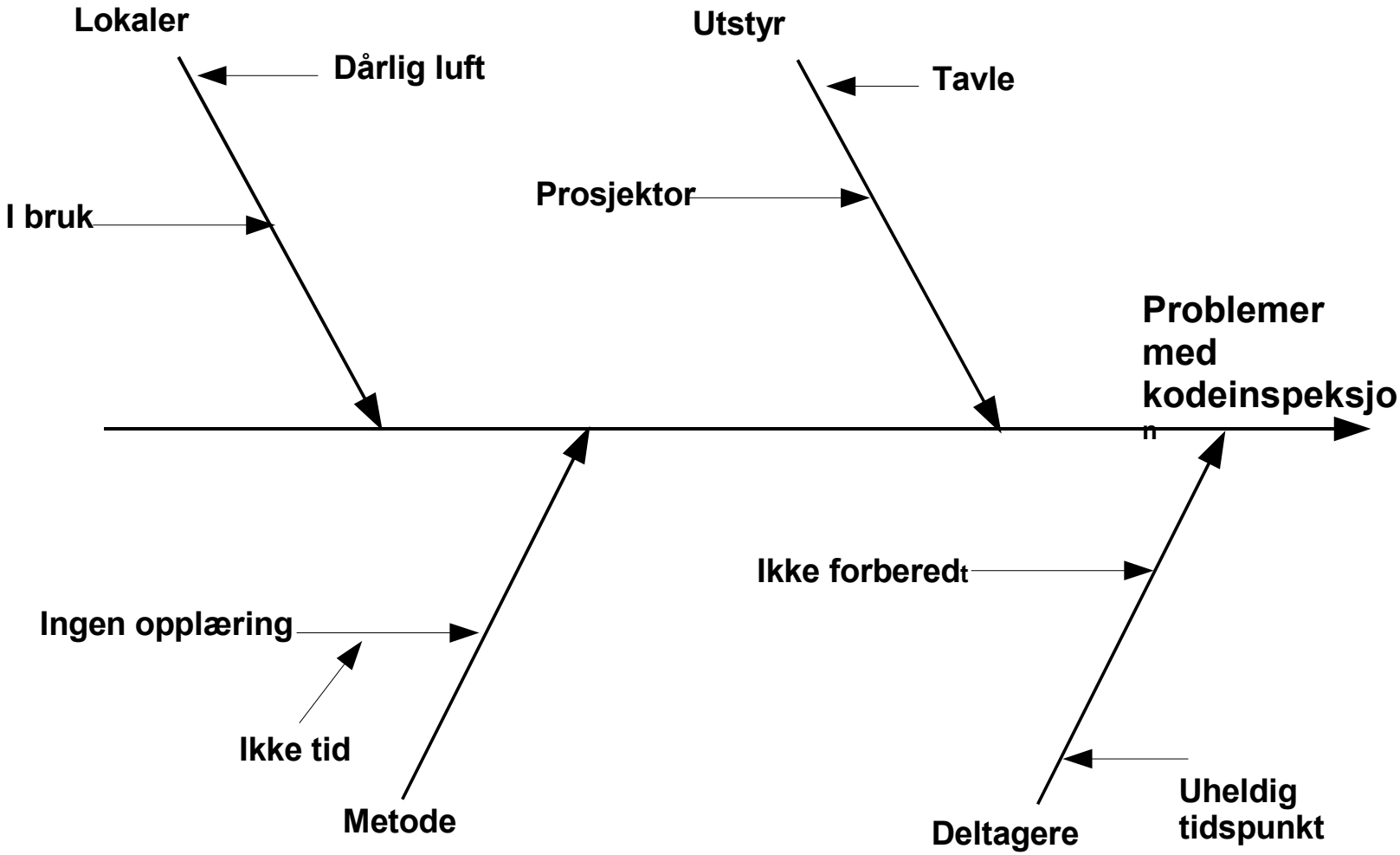
Årsak-virkning-diagrammet kan brukes både ved analyse av opplevde problemer og for å kunne forutsi fremtidige utfordringer.

En prosess med flere trinn eller faser kan analyseres ved at man tegner årsak-virkning-diagram for hvert prosesstrinn.

En samlet vurdering til slutt kan avdekke de problemene som har størst betydning for totalprosessen.

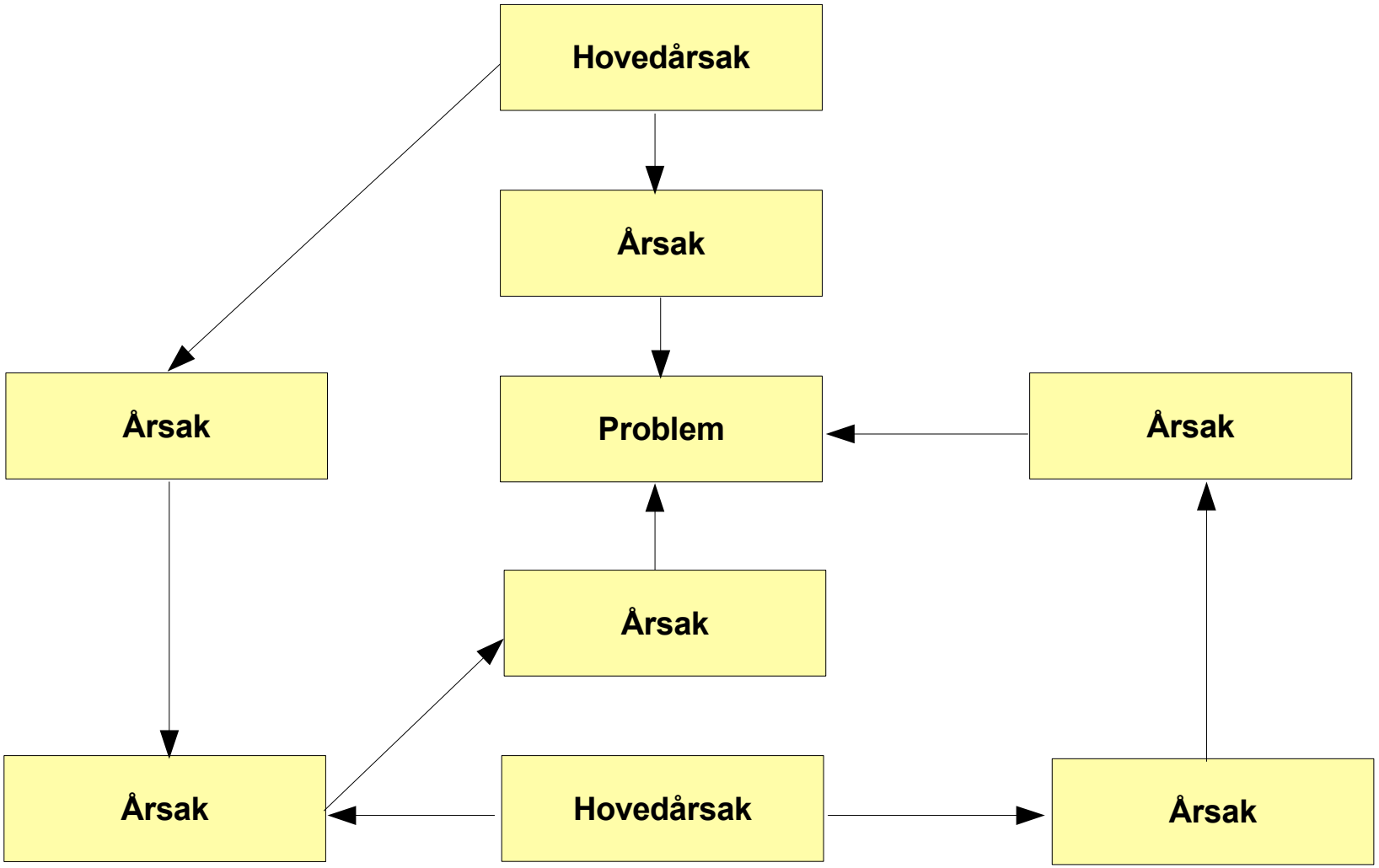
Neste figur er et konstruert eksempel på problemer med kodeinspeksjoner:

# Problemer med kodeinspeksjon



Universitetet i Stavanger

# Relasjonsdiagram



Universitetet i Stavanger

# Relasjonsdiagram

- Relasjons-diagram brukes for å vise sammenhenger mellom problemer og årsaker.
- Sammenhengene vises med piler mellom bokser.
- Figuren over viser en prinsippskisse av oppbygningen til et relasjons-diagram.
- En starter med et problem, finner årsakene til problemet.
- En pil tegnes mellom en årsak som utløser en annen.
- Teknikken er nyttig når en vil avdekke virkelige årsaker og ikke bare symptomer på at det er et problem.

Universitetet i Stavanger

# Relasjonsdiagram

**Trinnene**

1. Formuler problemstillingen
2. Identifiser alle faktorene som antas å ha en eller annen sammenheng med problemet. Dette kan gjøres ved idédugnad
3. Formuler faktorene og tegn bokser. Man kan skrive på klisterlapper som kan henges opp på en vegg eller tavle
4. Identifiser årsak-virkning forhold og tegn piler
5. Analyser diagrammet og finn hovedårsaker. Faktorer med mange piler ut er kandidater til å være hovedårsaker.

Universitetet i Stavanger

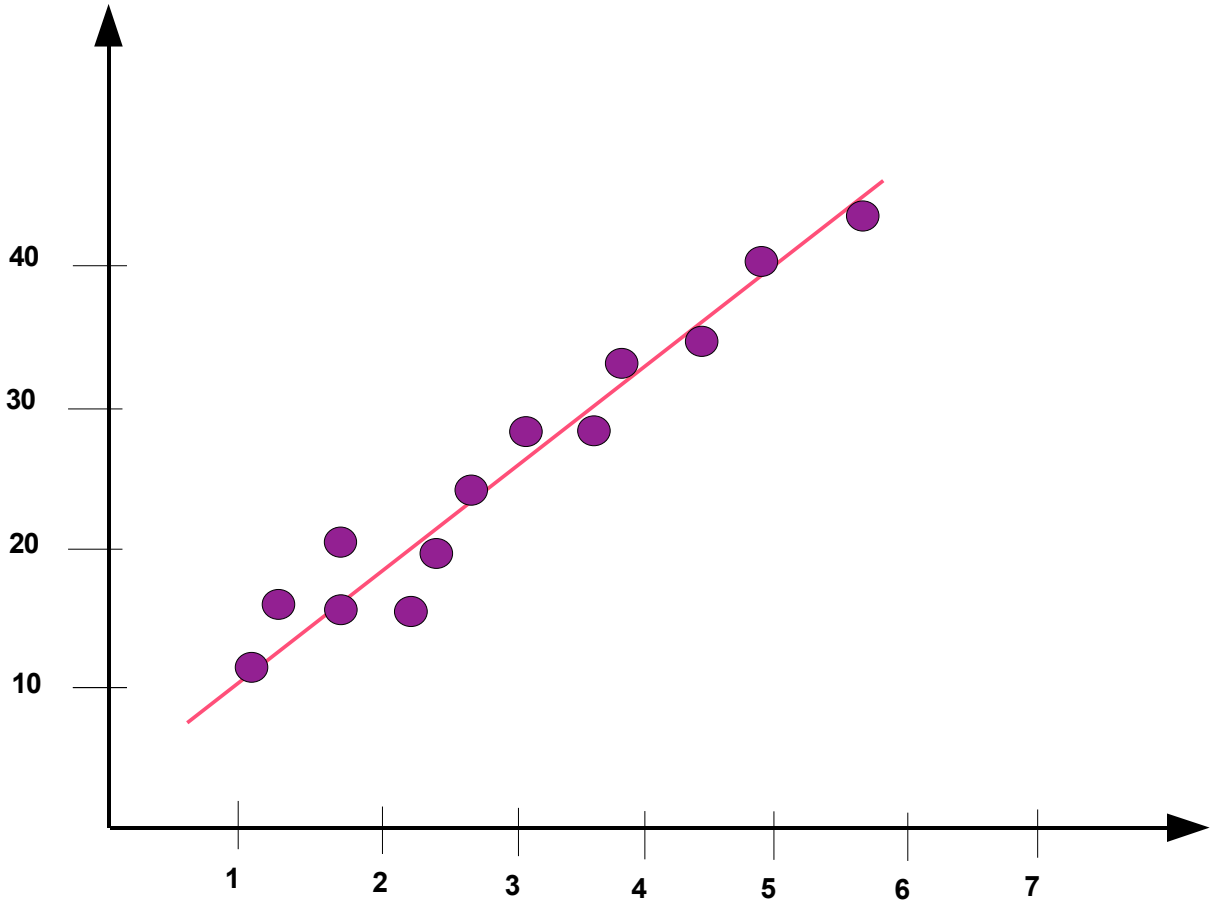
# Relasjonsdiagram

**Kommentarer**

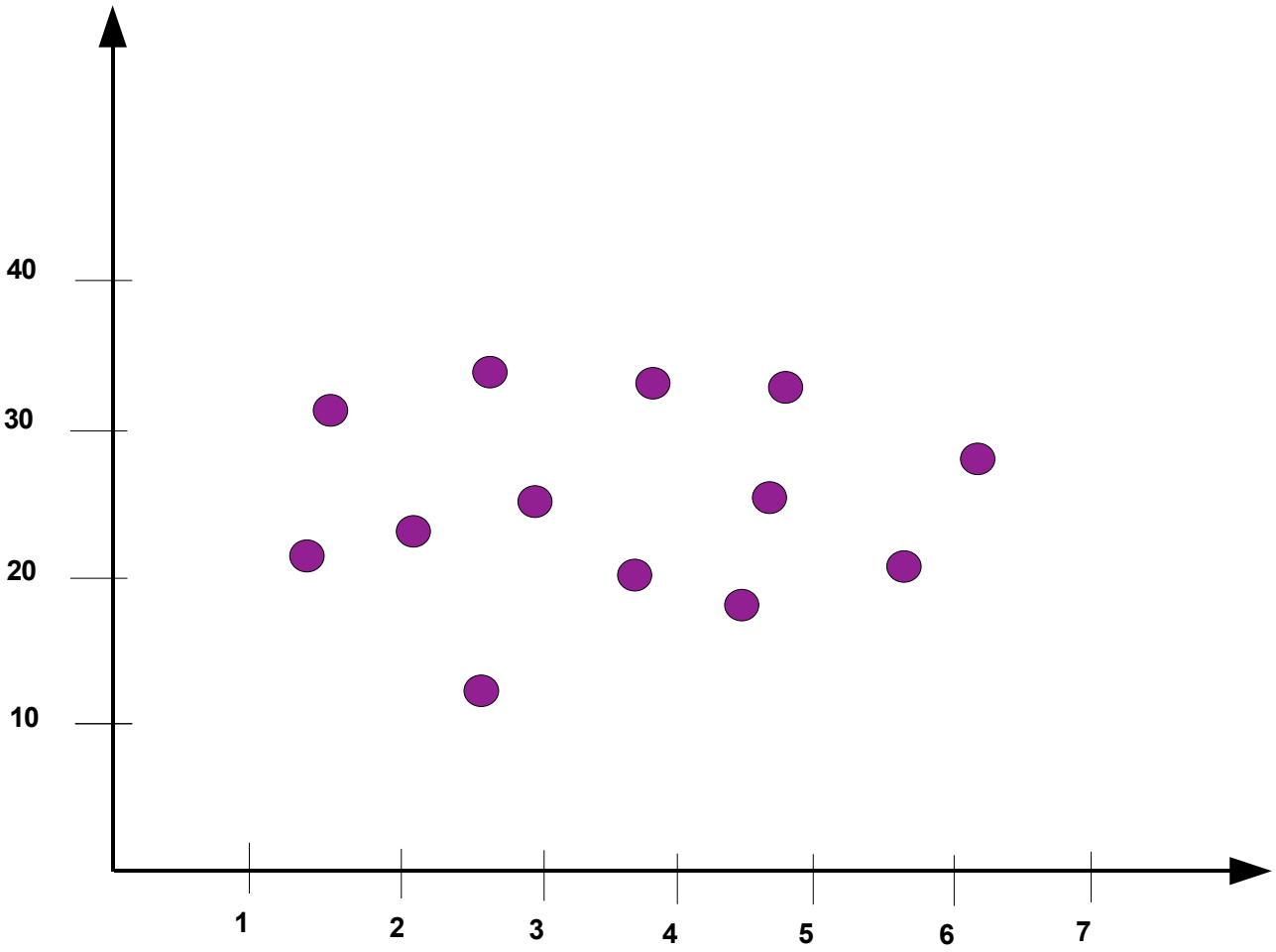
- Relasjons-diagram har på mange måter samme funksjon som årsak-virkning-diagram.
- Også i dette diagrammet tegner vi piler fra ide eller årsak som utløser en annen.
- Hva man bruker kan være en smakssak, men relasjons-diagram er bedre egnet for kompliserte problemstillinger.

Universitetet i Stavanger

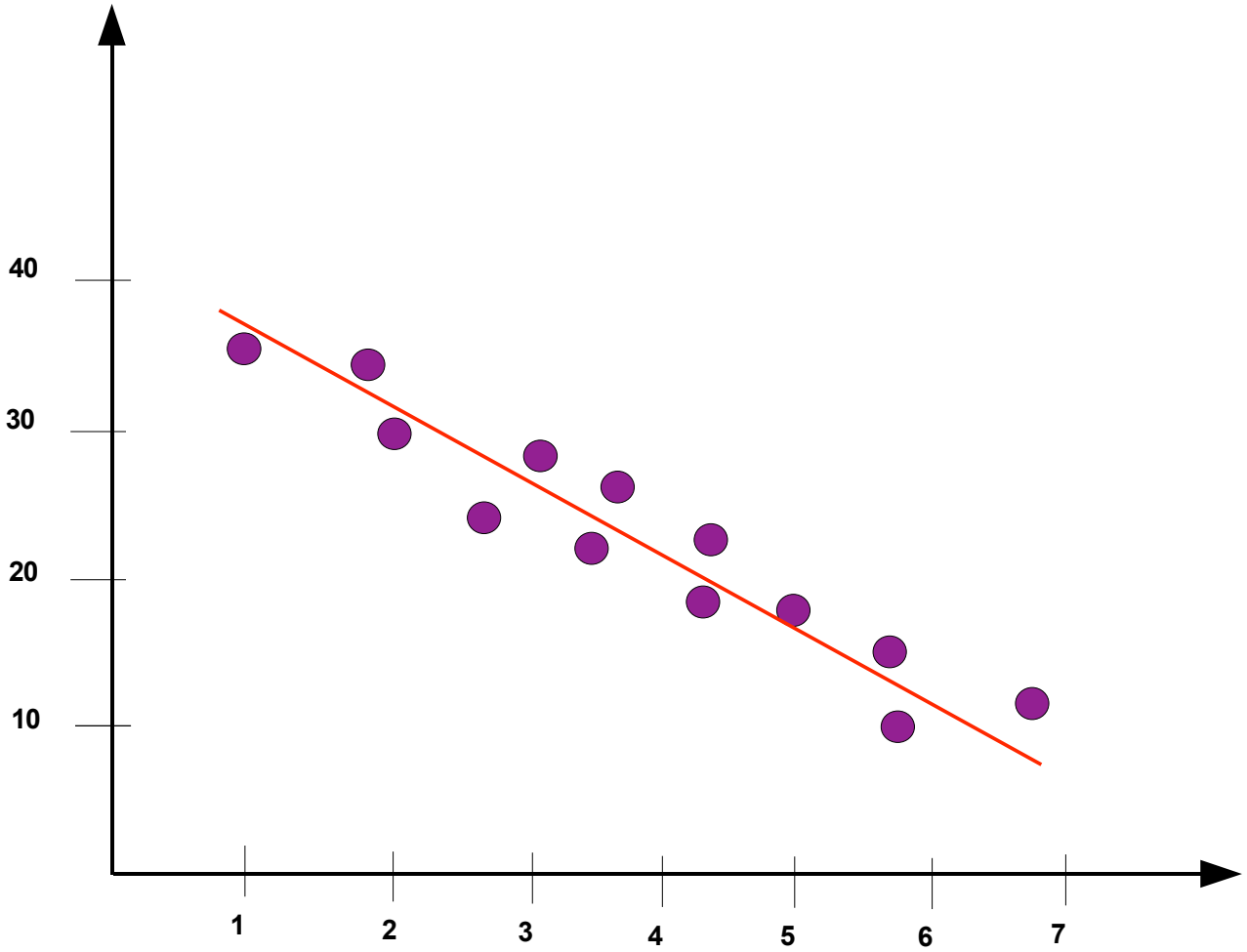
# Sprednings-diagramer positiv sammenheng



# Sprednings-diagramer
# Ingen sammenheng



# Sprednings-diagramer
# Negativ sammenheng



Universitetet i Stavanger

# Sprednings-diagramer

**Kommentarer**

Ting som typisk framstilles i et sprednings-diagram, er sammenhengen mellom størrelsen på et program og faktorer som påvirkes av størrelsen.

Størrelsen kan uttrykkes i antall kildekodelinjer, funksjonspoeng, eller antall objekter.

Eksempler på faktorer som kan tenkes å bli påvirket av størrelsen, er utviklingstid, ressurser som trengs til utvikling, antall feil som introduseres under utviklingen.

COCOMO-modellene (kap 21) for estimering baserer seg på at det er en sammenheng mellom størrelsen på programvaren og ressursene som trengs for å utvikle programvaren.

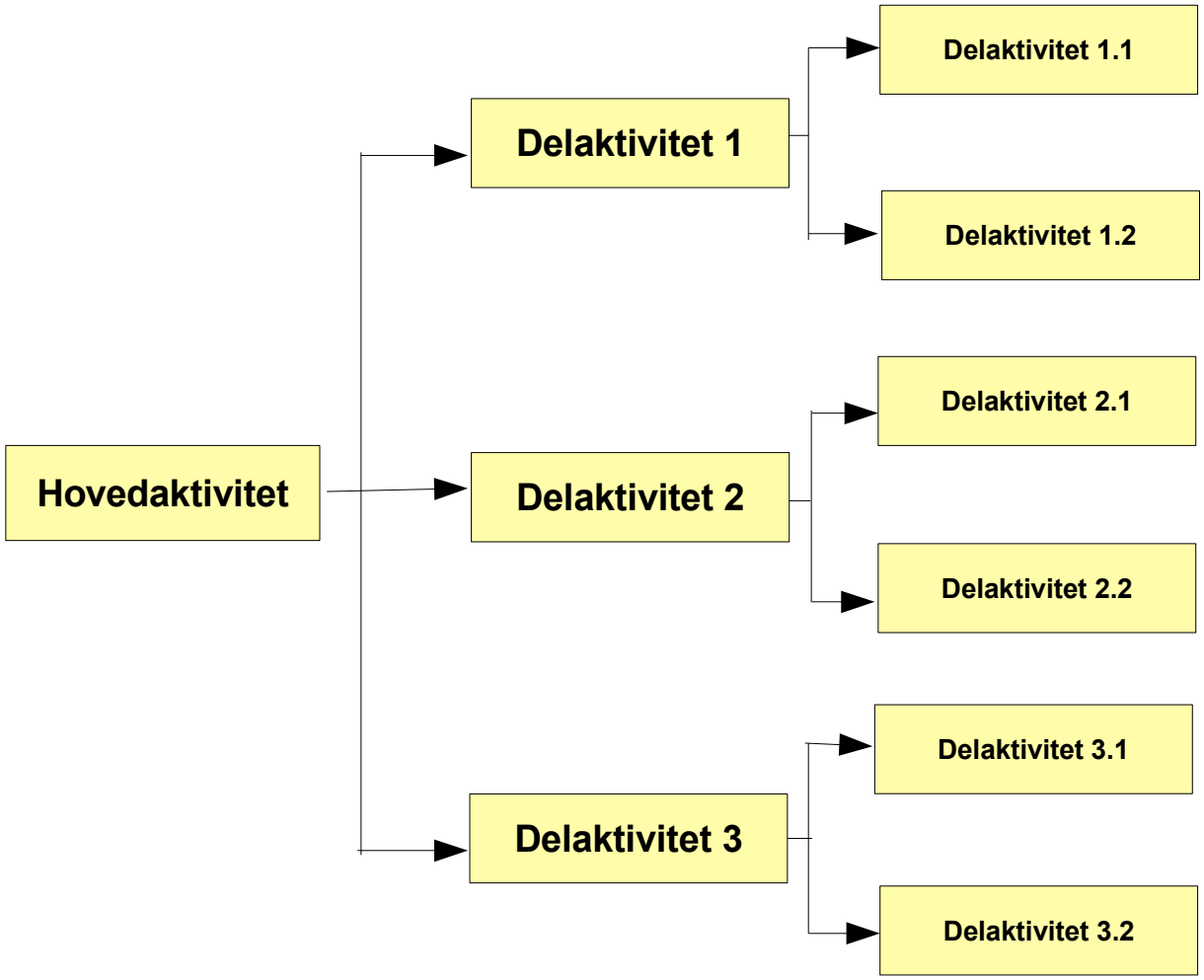
Data framstilt i sprednings-diagrammer kan analyseres ved hjelp av statistiske teknikker som korrelasjonsanalyse. Det gir kvantitative estimater for styrken i relasjoner mellom variabler.

Vi har her lansert sprednings-diagrammer blant verktøy for problemanalyse.

Det er gjordet for  å illustrere sammenhengen mellom ting. Generelt kan en si at et sprednings-diagram er en grafisk framstilling av data. Sprednings-diagram er en form for plotte-teknikk.

Universitetet i Stavanger

# Tre-diagram



# Tre-diagram

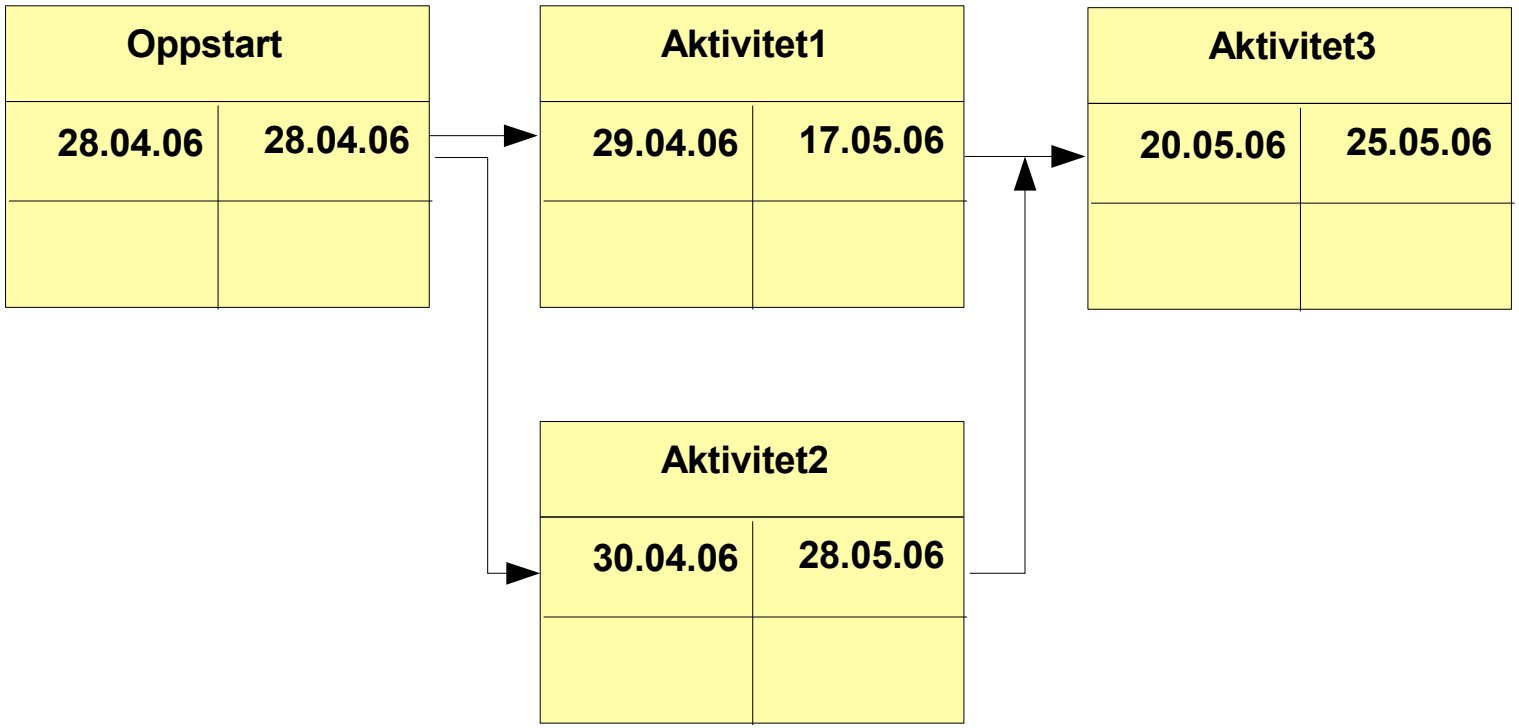
- Diagrammet brukes for å bryte ned ting eller for å vise alternative veier når beslutninger skal tas.
- I figuren er diagrammet brukt til å bryte en hovedaktivitet ned i del-aktiviteter.

**Kommentar**

- Tre-diagrammet har ulike navn alt etter hva det skal brukes til.
- Et prosessbeslutningadiagram viser hvordan risikofaktorer kan deles opp og ende i aksjoner for å håntere risikoen.
- Den generelle betegnelsen tre diagram brukes gjerne når en tar utgangspunkt i et mål og bryter det ned i aktiviteter for å nå målet, som vist på figur 22.10.

Universitetet i Stavanger

# Nettverks-diagram



Universitetet i Stavanger

# Nettverks-diagram

- Nettverks-diagram brukes under prosjektplanlegging. Det viser trinnene i et prosjekt og avhengigheten mellom dem.
- Figuren viser et PERT diagram.
- Boksene kan ha forskjellig innhold. I figuren har vi brukt navnet på aktiviteten, startdato og sluttdato. Det kan også lages plass for å angi varighet og slakk, virkelig startdato og virkelig sluttdato.
- Nettverks-diagram kalles ofte pil-diagram.

Universitetet i Stavanger

# Intensive arbeidsmøter

Ideen bak slike møter er enkel – vi samler alle parter i en sak i et rom og slipper dem ikke ut før de er blitt enige.

Gjennom slike arbeidsmøter ønsker en å oppnå to ting:

- En effektiv bruk av ressurser
- En dramatisk forbedring av tidsforbruket sammenlignet med vanlig saksgang

Intensive arbeidsmøter er en god ting, men det er en rekke forutsetninger som nå være på plass hvis de skal bli vellykkede.

# Intensive arbeidsmøter

Valg av deltagere er viktig

- Ikke for mange, men alle som er part i saken må være med
- Deltagerne må vite hvem de representerer og de må ha besluttende myndighet

Tre sentrale roller

1. Møtelederen

   Skal sørger for at tidsfrister overholdes, at regler følges, at alle kommer til ordet. Skal ikke være part i saken og samarbeider tett med neste rolle:

2. Fasilitator

   En viktigste rollen. Har ansvar for at møtet når målet. Må hele tiden følge med i det som skjer, oppsummere, stiller spørsmål, parkere tema når en ikke kommer lenger, ber om time-out når det er nødvendig, osv. Skal ikke være part i saken. En god løsning er å bruke en utenforstående med erfaring. En krevende rolle

3. Sekretariat

   Dokumentere det som skjer. Det må skje på en slik måte at et foreløpig referat kan legges fram når som helst. Dette betyr at referatet skrive ved hjelp av en PC.

# Intensive arbeidsmøter

Møtet må **planlegges svært nøye**. Før møtet starter må en sørge for:

- Målet for møtet er klart, kjent og akseptert.
- Det foreligger en fast agenda med faste tidsfrister for de ulike sakene og planen er kjent og akseptert.
- Alle som deltar er forberedt og kjenner saken godt.
- Møterommet er godt utluftet og har nødvendig utstyr.

Det er nødvendig, men ikke tilstrekkelig, å sende ut informasjon om møtet på forhånd. En sjekke at alle har gjort de nødvendige forberedelsene. Det kan være en ide å ha et formøte med de involverte partene. Før møte starter bør en sjekke med hver enkelt deltaker om han er godt nok forberedt. Hvis en deltaker ikke er forberedt kan han bli oppfordret til å forlate møtet.

# Intensive arbeidsmøter

**Møterom og utstyr**

Møteformen stiller store krav til møterommet.

- Behagelig å oppholde seg i
- God ventilasjon
- God plass
- Rikelig plass til deltakerne
- Rikelig med vegger til gule lapper
- Flippover
- PC-verktøy
- Kanon
- PC-nettverk

# Intensive arbeidsmøter

**Kreative teknikker og verktøy for problemløsning**

- Det naturlig å ta i bruk kreative teknikker

Et intensivt arbeidsmøte må ha **faste regler** og disse må være kjente og aksepterte.

- Reglene må håndteres strengt.
- Målet med reglene er å sikre at alle kommer til ordet
- Alle synspunkter blir tilstrekkelig belyst,
- Sakslisten kan gjennomføres som planlagt,
- Alle saker får nødvendig behandling.

Universitetet i Stavanger

# Intensive arbeidsmøter

**Eksempler på regler:**

- Fem-minutters-regelen

  Alle deltakerne kan be om fem minutters pause når de måtte ønske det. Hensikten er å sikre at alle er opplagte og klare nok i hodet til å bidra konstruktivt

- Tildelt taletid

  Hver deltaker får utdelt en viss mengde taletid, møteleder tar tiden, og deltakere som har brukt sin del får ikke ta ordet. Hensikten er å sørge for at alle kommer til ordet, og at ikke enkelte deltakere dominerer hele møtet.

- Parkeringsplass

  Det opprettes parkeringsplass for problemer som ikke løses, og tema det ikke oppnås enighet om. Det er vanlig å gå systematisk gjennom parkeringsplassen på slutten av dagen og bestemme hva som skal gjøres med hvert enkelt tema som finnes der. Stemme over det, behandle det neste dag (dersom møtet varer over flere dager), utsette det til et sener møte, utrede det i mellomperioden, osv.

# Intensive arbeidsmøter

**Gjennomføringen**

- Det viktigste er at møtet gjennomføres etter den planlagte agenda.
- Hvis det oppdages at:
  - noen av deltakerne ikke er tilstrekkelig forberedt,
  - noen ikke har tilstrekkelig mandat til å ta beslutninger
  - det hender noe uforutsett som hindrer møtet i nå målet
- Arbeidet bør oppsummeres og møtet avsluttes.

# Intensive arbeidsmøter

**Etterarbeid**

- Etterarbeidet er like viktig som forarbeidet.
- Fasilitator sammen med referenten utarbeider en dokumentasjon over det som ble vedtatt på møtet Dokumentasjonen sendes til alle deltakerne, de får en frist til å kommentere referatet og til å komme fram med eventuelle motforestillinger.
- Siden et intensivt arbeidsmøte er krevende og hektisk, kan det bety at enkelte deltakere i ettertid kan komme fram med vesentlige innspill eller innvendinger som ikke kom fram under møtet. Det må være rom for at disse blir hørt.

# Intensive arbeidsmøter

**Forutsetninger for å lykkes**

Et intensivt arbeidsmøte er god arbeidsmåte for å oppnå enighet om brukerkrav eller andre saker. Suksessen av møtet står og faller med organiseringen av møtet. Vi kan ikke lykkes uten å ha

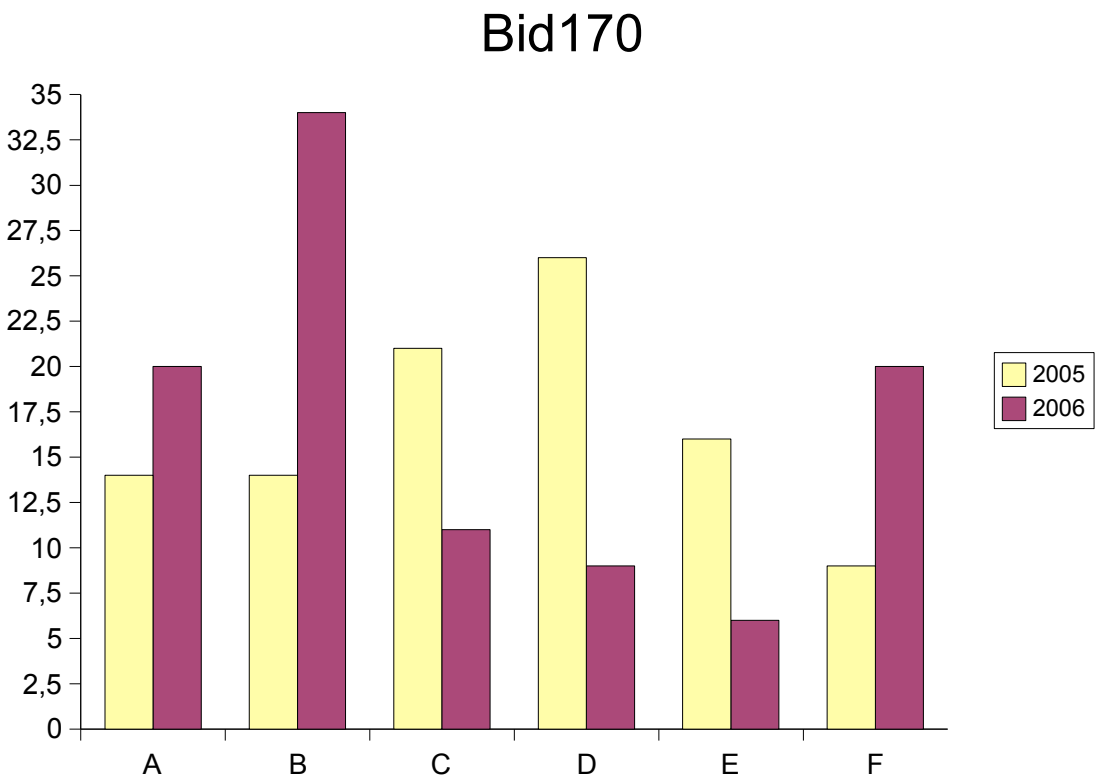
- klare mål
- de riktige deltakerne, vel forberedte og med mandat til å treffe beslutninger
- en klar plan for møtet
- en dyktig fasilitator
- en dyktig sekretær
- støtte av EMS-verktøy
- et grundig forarbeid

Universitetet i Stavanger

# Verktøy for presentasjon

- Teknikker for grafisk framstilling av data.
- Grafisk framstilling er nyttig når vi analyserer.
- En god grafisk framstilling hjelper til å avsløre problemer og kan avdekke årsakene til disse problemene.
- Sånn sett er dette også verktøy for problemløsning.

# Verktøy for presentasjon



# Verktøy for presentasjon


| *Minutter pr. Kildekodelinje* | *Antall* | *Frekvens* |
|---|---|---|
| Fra 4 opp til 5 | 5 | 0,23 |
| Fra 5 opp til 6 | 2 | 0,09 |
| Fra 6 opp til 7 | 7 | 0,32 |
| Fra 7 opp til 8 | 3 | 0,14 |
| Fra 8 opp til 9 | 3 | 0,14 |
| Over 9 | 2 | 0,09 |
| ***Total*** | *22* | *1,00* |

Universitetet i Stavanger

# Verktøy for presentasjon

**Kommentarer**

Et sentralt tema i forbindelse med histogrammer er antall av og bredden på søylene. Her er noen generelle retningslinjer:

- Det er best at alle celler er like brede
- Midtpunktet i hver klasse markerer midten av en søyle
- Bruk ikke for mange søyler. Husk at histogrammet skal være et sammendrag av en større mengde data. For mange søyler vil ikke gi et slikt inntrykk. Men det bør ikke være for få celler heller, da blir samendraget så konsentrert at dataene blir skjult. Her er en anbefaling på antall søyler.

Universitetet i Stavanger

# Verktøy for presentasjon

Tabellen kan brukes som en indikasjon på hvor mange søyler som er passe

| *Antall observasjoner* | *Antall søyler (klasser)* |
|---|---|
| Mindre enn 50 | 5 - 7 |
| 50 - 100 | 6 - 10 |
| 100 - 250 | 7 - 12 |
| Over 250 | 10 - 20 |

# Verktøy for presentasjon

**Pareto-diagram er et søylediagram**
Søylene er ordnet etter størrelse fra venstre mot høyre. Den høyeste søylen kommer lengst mot venstre. Hensikten er å skille ut de viktigste problemområdene.

Diagrammet har navnet etter Paerto.

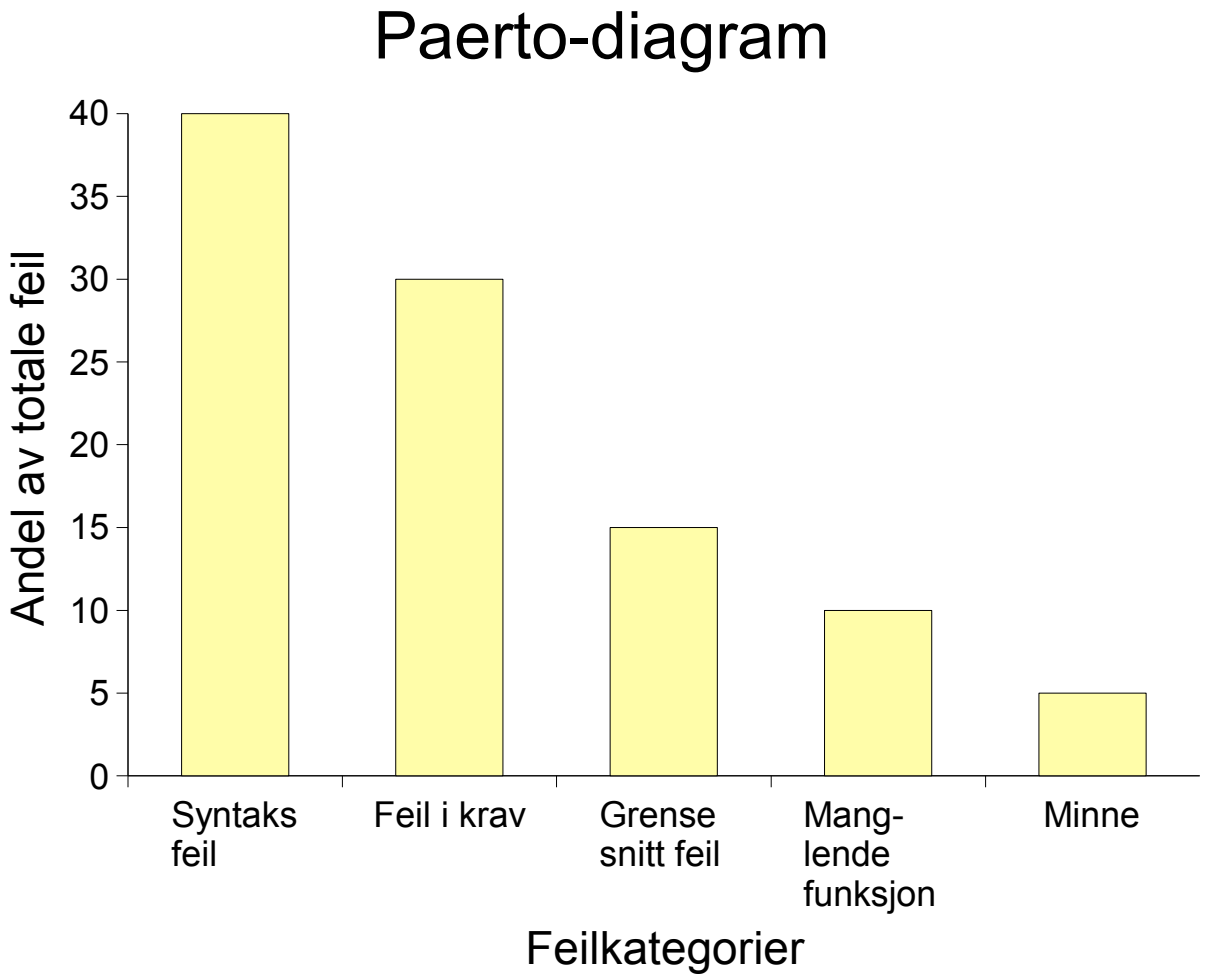
- Perto-prinsippet. 80/20 – regelen.
- 80% av alle problemer skyldes 20% av alle årsakene.

Diagrammet kan hjelpe til å svare på følgende:

- Hvilke feil er viktigst å finne?
- Hvilke deler av prosessen bidrar mest til de problemene vi har?

Det gjelder å finne de 20% viktigste årsakene og konsentrere oppmerksomheten om dem.

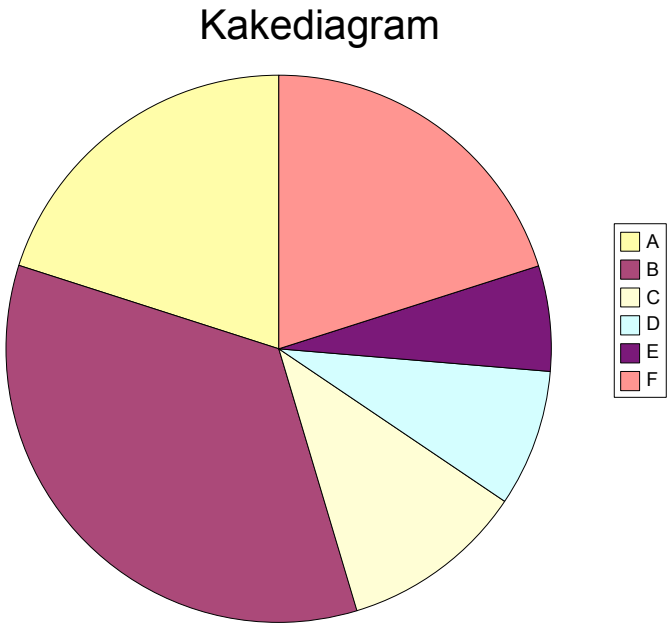
# Verktøy for presentasjon



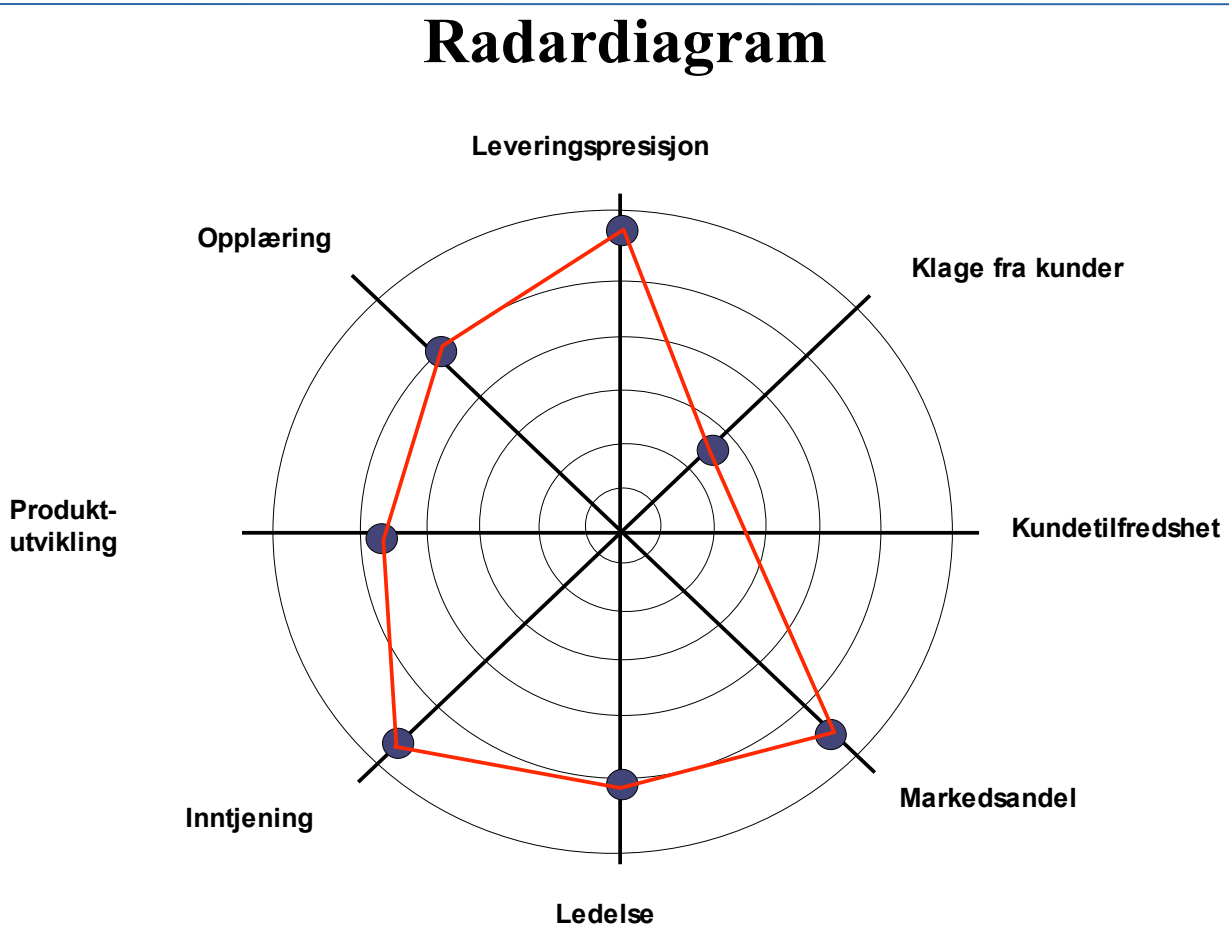
# Verktøy for presentasjon

## *Kakediagram*

Kakediagram brukes for å vise en prosentvis fordeling av et eller annet. Diagrammet gir et godt bilde av størrelsesforhold når kravet til presisjon ikke er stort.



# Verktøy for presentasjon



# Verktøy for presentasjon

**Raderdiagrammer** bruker vi når vi vil sammenligne prestasjoner eller egenskaper ved en prosess eller et produkt.

Diagrammet har fått navnet sitt fordi det ligner plottet av en radarskjerm.

Det har stråler (eiker) som går ut fra sentrum i diagrammet.

Hver stråle representerer en egenskap eller en prestasjon. Disse gis en verdi.

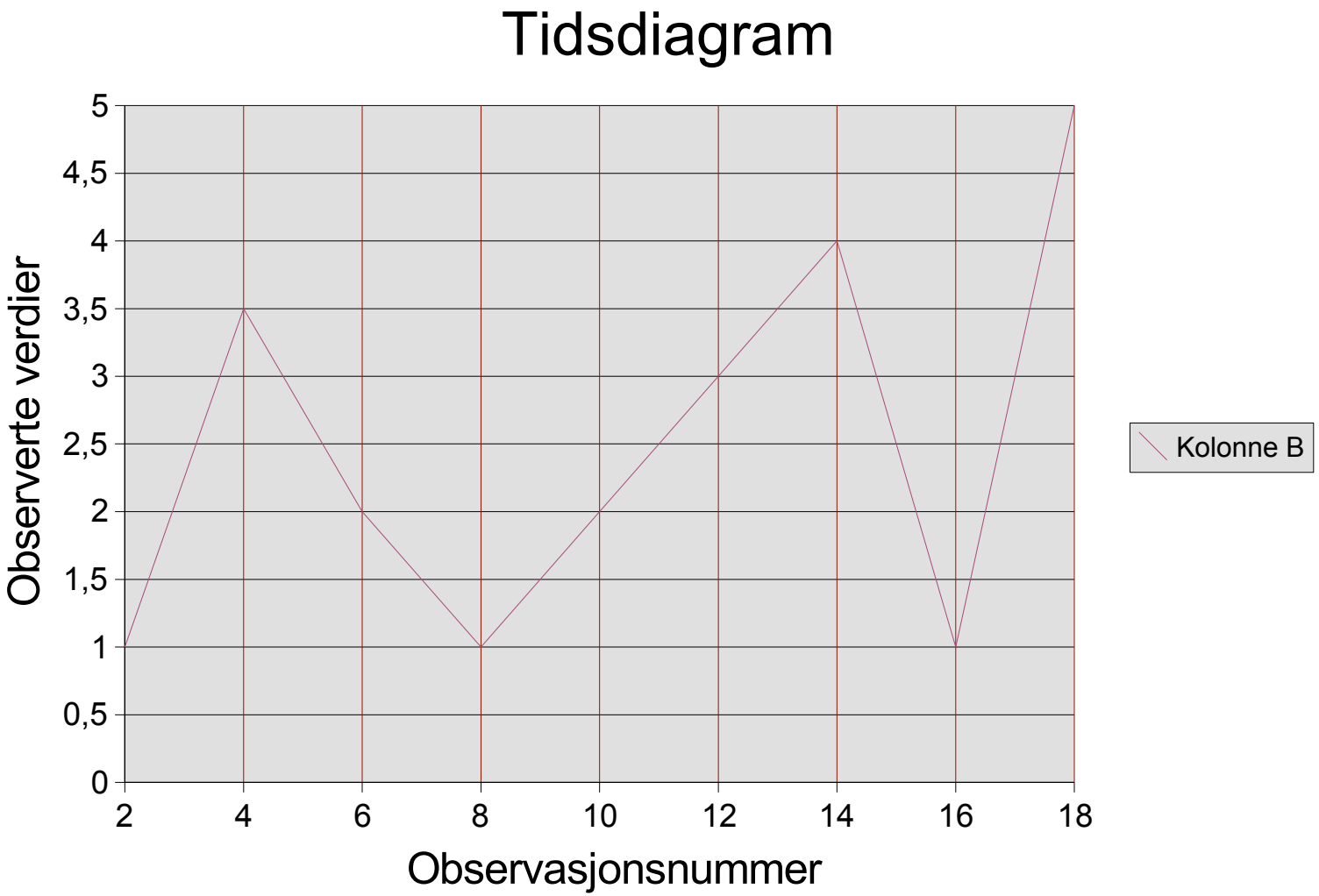
I diagrammet over har vi ikke gitt noen målestokk langs strålene. Den behøver ikke være den samme for hver egenskap. Men verdien øker ut fra sentrum i diagrammet. Det vil si at prestasjonen er bedre jo lenger ut på strålen den er plassert. Vi kan bruke målte verdier eller subjektivt vurderte verdier.

**Kommentarer**

Diagrammet er velegnet for å vise en prestasjons-profil. Det kan også brukes når en vil sammenligne egen profil med andres profil.

Universitetet i Stavanger

# Verktøy for presentasjon



Universitetet i Stavanger

# Verktøy for presentasjon

**Hva er et tidsdiagram?**

- Et tidsdiagram viser variasjonen av data over tid.
- Kan brukes til å se på trender eller variasjonsmønster.
- I diagrammet kan vi se hvilke retning ting utvikler seg. Ser en uønskede trender, må man gripe inn og gjøre endringer.
- Flere datasett kan plottes i samme diagram. Slik kan en se sammenhenger og hvordan forskjellige forhold utvikler seg over tid.

Universitetet i Stavanger

# Verktøy for presentasjon

**Kommentarer**

Tidsdiagrammer er et spesialtilfelle av spredningsdiagrammet, men nå er den ene aksen alltid er en tidsakse.

Tidsdiagrammet kan brukes til å

- Se hvordan prosessen oppfører seg over tid
- Se hvordan ressursbruk endrer seg over tid
- Se trender i bemanningen
- Se hvordan en klarer å unngå feilrapporter
- Se hvordan ting utvikler seg i forhold til planer eller estimater

Bruk av diagrammer og estimater kan overdrives. Noen trender er naturlige svingninger, og da er det ikke nødvendig å sette inn tiltak.

# Trinnvis kvalitetsplanlegging (QFD)

## Trinnvis kvalitetsplanlegging

## «Quality Function Deployment»(QFD).

QFD er et rammeverk. Det inneholder et sett av metoder og verktøy og er et av de viktigste redskapene vi har fått fra TKL bevegelsen. Kundefokusering er viktig prinsipp i TKL. Dette fordi organisasjoner er avhengig av kunder og må sørge for at kundene er tilfredse. Det er ikke nok bare å tilfredsstille uttrykte krav. Kano-modellen sier at vi må også sørge for at selvfølgeligheter og attraksjoner er med i leveransen. QFD er et redskap som skal sørge for at alle krav blir tatt hensyn til. Metodene og verktøyene skal sørge for at kundens stemme blir hørt gjennom hele utviklingsprosessen. Alle intresseparter skal sammen, gjennom bruk av QFD, sørge for at produktet som skal utvikles, blir best mulig.

QFD er utviklet i Japan. Det ble først tatt i bruk i industrien, men verktøyet og metodene er også velegnet for tjenesteytende virksomhet. Det betyr at hovedideen bak QFD kan tilpasses omtrent alle typer virksomhet. I det siste har QFD fått økende interesse innenfor utvikling av programvare.

Flere store virksomheter hevder at de benytter QFD, Toyota, HP og IBM

Universitetet i Stavanger

# Trinnvis kvalitetsplanlegging (QFD)

**Verktøyene i QFD**

Den overordnede filosofien bak QFD er at det må legges opp til en trinnvis utviklingsprosess hvor kundetilfredshet er i fokus.

Utviklingsgruppen må være tverrfaglig for å oppnå dette. Den må ha representanter fra forskjellige kundegrupper og fra alle avdelinger i virksomheten som kan bidra positivt til at produktet blir slik kundene vil ha.

I tillegg inneholder QFD et sett av verktøy som skal lette arbeidet. I tradisjonell japansk ånd må antall verktøy være syv.

Universitetet i Stavanger

# Trinnvis kvalitetsplanlegging (QFD)

Vektøyene er:

- relasjonsdiagram (22.3.3)
- slektsdiagram (22.2.5)
- trediagram (22.3.5)
- matrisediagram (22.3.1)
- matrise prioriteringsdiagram, som er en type sammenligningsmatrise (22.3.1)
- prosessbeslutningsdiagram (22.3.5)
- nettverksdiagram (22.3.6)

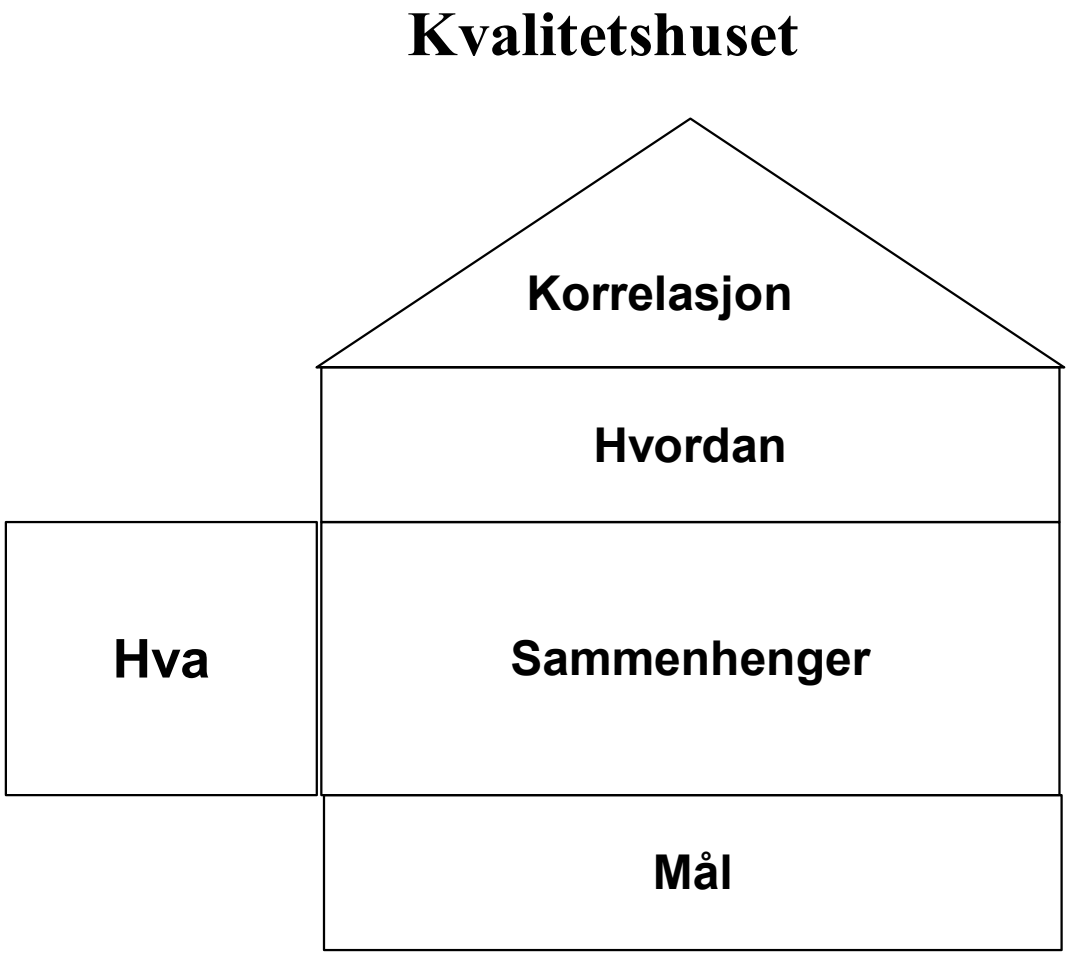
Universitetet i Stavanger

# Trinnvis kvalitetsplanlegging (QFD)

**Kvalitetshuset**

Selv om det er syv verktøy som utgjør basisen for QFD, er det ett som har fått større oppmerksomhet enn de andre. Det er det såkalte kvalitetshuset som er en sammenstilling av flere matrisediagrammer. For mange er det dette verktøyet som er QFD. Det finnes også mange varianter tilpasset ulike behov. Hovedideen i de fleste variantene er vis på neste figur.

# Trinnvis kvalitetsplanlegging (QFD)



# Trinnvis kvalitetsplanlegging (QFD)

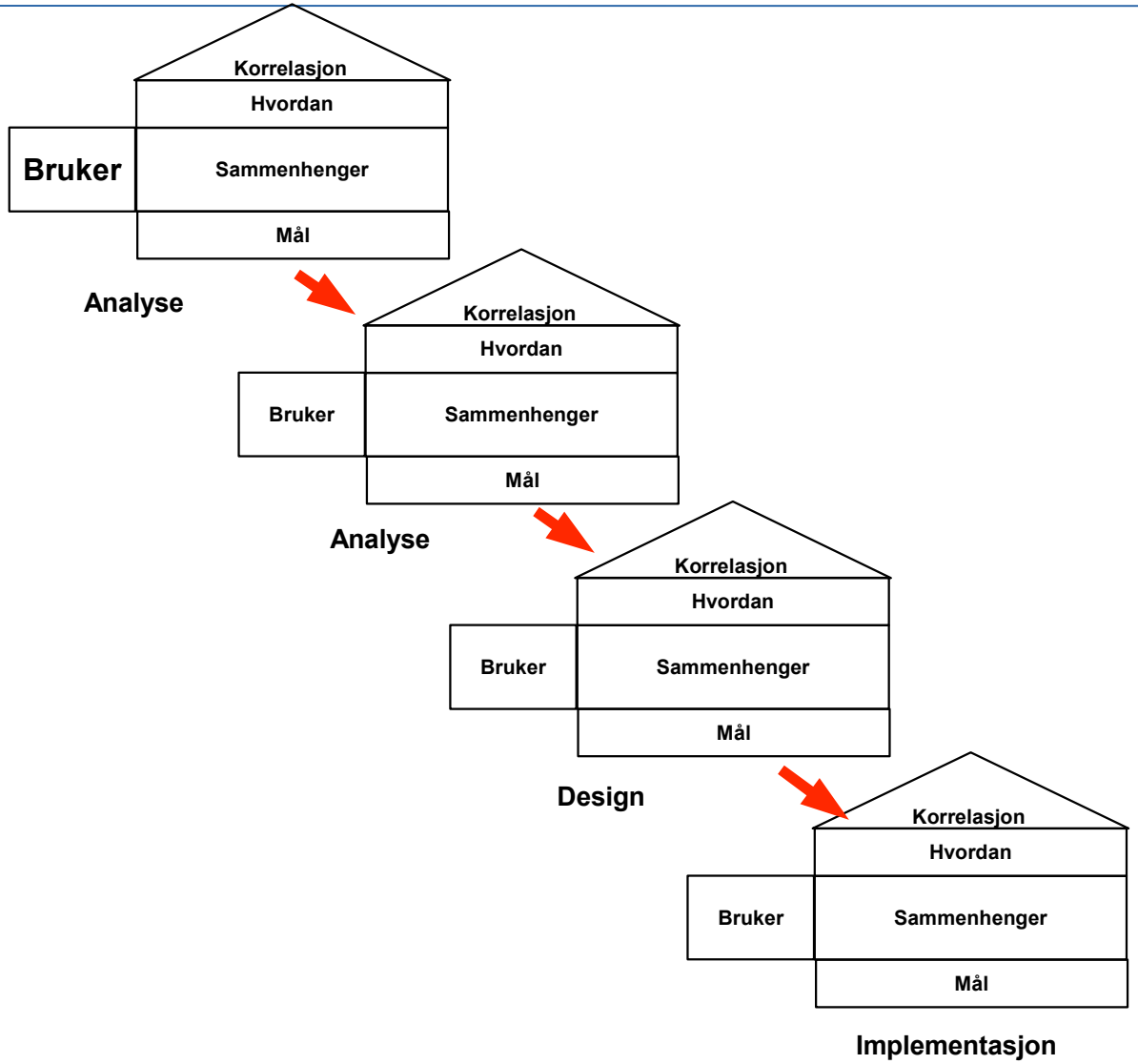
«Etasjene» kvalitetshuset er satt sammen av flere matriser. Sammenhenger er en matrise som viser relasjoner mellom hva man ønsker å oppnå, og hvordan man vil nå det. Taket er en matrise der en indikerer positive eller negative relasjoner mellom faktorene i Hvordan. «Kjelleren» kalt mål, tar en med dersom faktorene i hvordan har kvantifiserbare mål knyttet til seg.

QFD legger opp til en trinnvis prosess hvor en hele tiden passer på at kundens stemme blir hørt. Dette gjøres ved at det lages et kvalitetshus for hvert trinn i prosessen. Hvordan fra ett trinn blir hva i trinnet som følger etter.

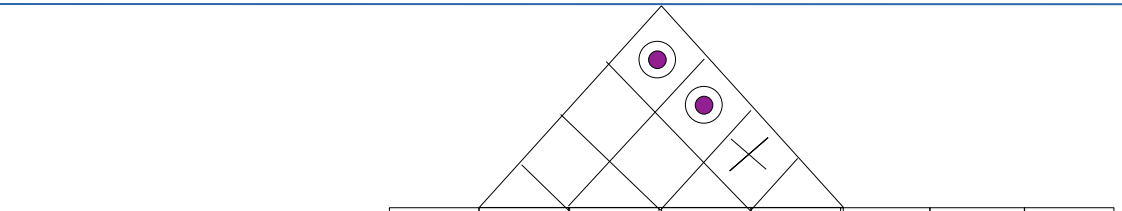
Figuren under viser hvordan vi kan tenke oss å bruke en kjede av kvalitetshus i en generisk utviklingsprosess. Vi starter med å identifisere brukerne og deres krav, og konstruerer Kvalitets huset for disse. Konstruksjonen av Kvalitets huset i de etterfølgende trinn sikrer at det er sammenheng mellom hva vi skal gjøre, og hvordan vi løser oppgaven. Det er hele tiden mulig å spore tilbake til brukerne og deres krav.

Kvalitets huset kan også brukes når vi vil prioritere prosess-

# Trinnvis kvalitetsplanlegging (QFD)



# Trinnvis kvalitetsplanlegging (QFD)



| Kundekrav til produktprosessen | | Viktighet | Kravgjennomgåelser | Kodeinspeksjon | Komponentbasert utvikling | XP som utviklingsmodell | Vår prosess | Konkurrentens prosess | Benchmark |
|---|---|---|---|---|---|---|---|---|---|
| | Feilfritt produkt | 5 | 9 | 9 | 9 | 9 | 3 | 4 | 5 |
| | Lav pris | 4 | 3 | 3 | 3 | 8 | 2 | 3 | 4 |
| | Kort utvikling | 5 | 3 | 3 | 9 | 3 | 1 | 3 | 4 |
| | Lett å bruke | 5 | 1 | 1 | 3 | 3 | 4 | 2 | 5 |
| | Absolutt viktighet | | 74 | 74 | 108 | 104 | | | |
| | Relativ viktighet | | 21% | 21% | 30% | 29% | | | |

Sterkt positivt

Svakt negativt

Universitetet i Stavanger

# Trinnvis kvalitetsplanlegging (QFD)

I Hva delen finner vi brukernes krav til produktet og dermed hva prosessen må kunne levere.

I Hvordan delen har vi en liste over noen mulige forbedringstiltak. Videre har vi en kolonne for viktighet. Verdien er fra 1 til 9, der 9 er høyeste verdi.

Til høyre er tre kolonner som viser hvordan vi, konkurrenten og markedsleder presterer. Vi ser at vår prosess er dårligere enn markedsleder på alle områder, den er også dårligere enn konkurrent på flere områder.

I taket vises grad av korrelasjon mellom tiltakene. Den er stekt positiv mellom XP kravgjennomgåelse, tilsvarende for kodeinspeksjon, men den er svakt negativ mellom XP og komponentbasert utvikling.

Kjelleren inneholder summer for viktighet og beregnet relativ viktighet. Det beregnes en sum for hvert tiltak. Summeringen beregnes slik:

For hver rute multipliseres rekkens tall for viktighet med den vekten brukerne har gitt ruten og alle disse verdiene summeres.

# Trinnvis kvalitetsplanlegging (QFD)

For hver rute multipliseres rekkens tall for viktighet med den vekten brukerne har gitt ruten og alle disse verdiene summeres.

For kravgjennomgåelse blir dette: $5 \cdot 9 + 4 \cdot 3 + 4 \cdot 3 + 5 \cdot 1 = 74$

Relativ viktighet blir: $\frac{74}{(74 + 74 + 108 + 104)} = 0{,}21$

Universitetet i Stavanger

# Trinnvis kvalitetsplanlegging (QFD)

Kvalitets huset viser at tiltakene Komponentbasert utvikling og XP er det som har relativt høyst viktighet, og disse tiltakene bør derfor prioriteres.

Studerer en taket i Kvalitets huset ser en at disse tiltakene har svakt negativ korrelasjon.  Grunnen til det er at XP legger opp til å lage minimumsløsninger og ikke utvikle for framtidig gjenbruk. Men komponenter som allerede er utviklet må fortsatt kunne gjenbrukens selv om en satser på XP. Her må det nærmere undersøkelser til.

Hovedpoenget med eksempelet er å vise at Kvalitets huset er et utmerket verktøy til å sammenstille informasjon. Denne informasjonen skal hjelpe oss til å ta rette beslutninger.

**Kommentar**

T dette eksempelet har vi brukt et kvalitetshus som er tilpasset våre behov. I litteraturen kan en støte på mange varianter av Kvalitets huset.

Universitetet i Stavanger

# GQM («Goal Question – Metrics»)

**Hva er GQM?**

GQM skal hjelpe oss til systematisk å bestemme metrikker. Det starter med formulering av mål. I tilknytning til målene kan det stilles en rekke spørsmål som må besvares. Svarene forteller om målene er nådd. For å kunne gi svar må vi måle. Det krever igjen at vi definerer hensiktsmessige metrikker. GQM viser hvordan dette kan gjøres.

«SPIQ har valgt GQM som metode for å definere, samle inn og analysere data innenfor utvikling av programvare. Dette gjelder både for produkt og prosess. Det er viktig allerede fra starten å være klar over at metodens styrke ligger i å samle inn, organisere, dokumentere og nyttiggjøre seg den kunnskap og erfaring som allerede finnes i bedriften. Bakgrunnen for å bruke GQM er erkjennelsen av at det som bruker en prosess er de virkelige ekspertene»

Universitetet i Stavanger

# GQM («Goal Question – Metrics»)

**Trinnene**

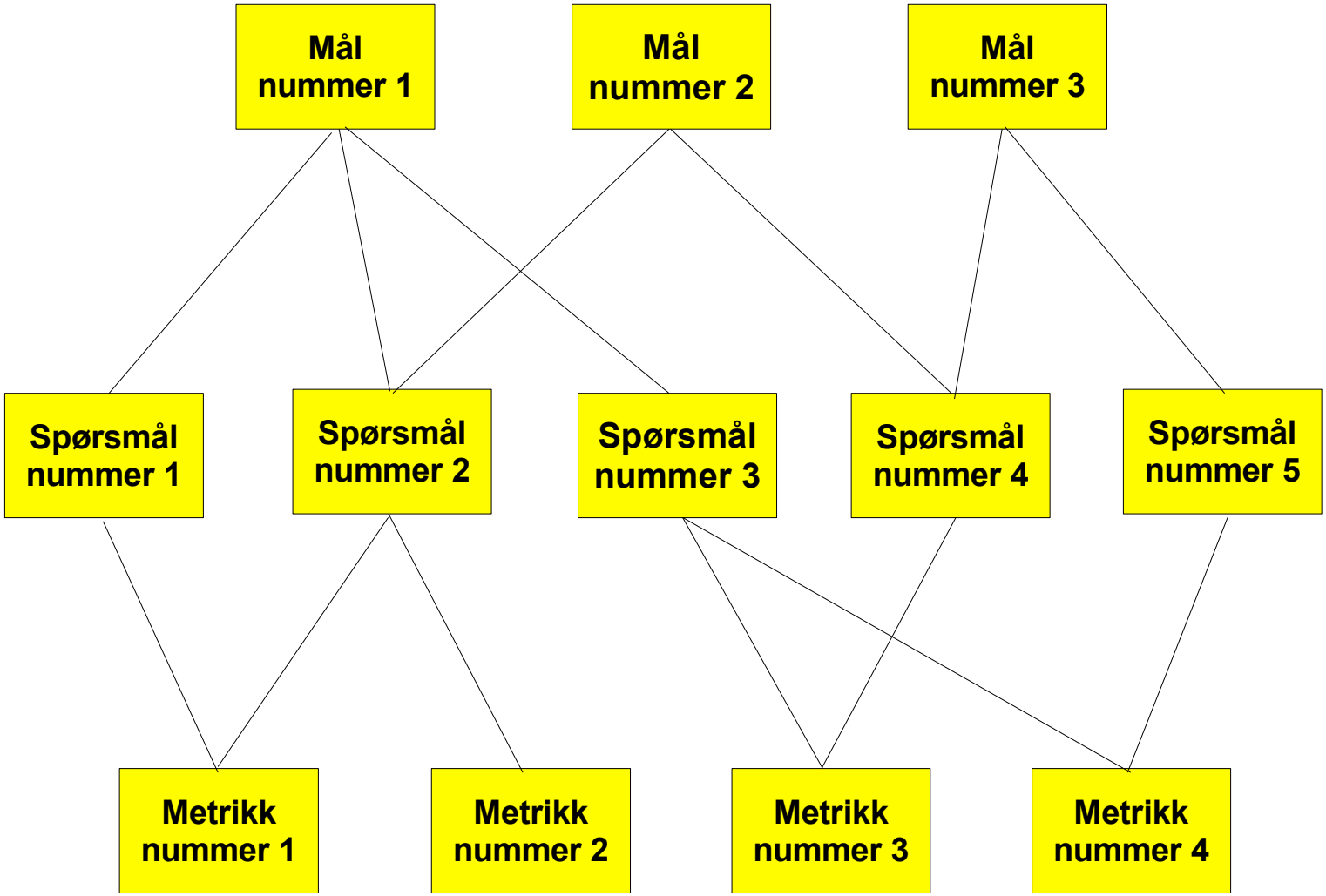
GQM tar utgangspunkt i prosjektmåle. Deretter følger disse trinnene

1. Formuler mål med utgangspunkt i prosjektmålene
2. Formuler spørsmål, som når de besvares, forteller om målene er nådd.

   Spørsmålene må formuleres presist, svarene må være kvantitative
3. Finn fram til metrikker som må samles inn for å gi svar på spørsmålene
4. Lag måleplaner
5. Samle inn, valider og verifiser data
6. Analyser dataene og vurder hvilke svar de gir på spørsmålene som er formulert. Indikerer svarene at målene er nådd?
7. Foreslå eventuelle forbedringer

Universitetet i Stavanger

# GQM («Goal Question – Metrics»)



# GQM («Goal Question – Metrics»)

**GQM – syntaksen og arbeidsarket**

Utgangspunktet er at en har mål som skal nås. Et slikt mål kan være at en ønsker å **redusere antall feil som når fram til akseptansetesten**. Et annet mål kan være **økt produktivitet**. I GQM formulerer en målene på en spesiell måte slik at det skal bli lettere å finne fram til de riktige spørsmålene. De formuleres etter **«GQM – syntaksen»** (Se Praktisk kvalitetsforbedring (Dybå 2002)) . Malen ser slik ut:

Analyser <*objekt*> for å <*hensikt*> <*fokus*> sett fra <*perspektiv*> i omgivelse <*omgivelse*>

Universitetet i Stavanger

# GQM («Goal Question – Metrics»)

Analyser <*objekt*> for å <*hensikt*> <*fokus*> sett fra <*perspektiv*> i omgivelse <*omgivelse*>

Ordene i klammeparentes betyr:

*Objekt* – det som skal analyseres. Det kan være et prosesstrinn eller et produkt.

*Hensikt* – det som skal oppnås. Det kan være at en ønsker å forstå noe eller forbedre noe

*Fokus* – hva det er ved prosessen, produktet eller tjenesten vi ønsker å forstå eller forbedre

*Perspektiv* – fra hvilke ståsted analysen skal foretas. De kan være fra kundens, utviklers, ledelse eller intresseparters

*Omgivelser* – i hvilke omgivelser det dreier seg om. Eksempel på omgivelser er organisasjon, avdeling og prosjekt

Etter at QGM målet er formulert ved hjelp av QGM syntaksen, må vi skaffe oss oversikt over og få dokumentert ting knyttet til målformuleringen. Arbeids arket hjelper oss med den nødvendige struktur og dokumentasjon. Arbeids arket har disse delene

# GQM («Goal Question – Metrics»)

| | |
|---|---|
| **Mål**<br>Mål formulert etter QGM syntaksen | |
| **Fokus**<br>Hovedformål for undersøkelsen | **Omgivelser**<br>Ting vi tror kan påvirke hovedmålet |
| **Hva vi tror**<br>Hva vi tror og vet om hovedformålet | **Påvirkning**<br>Hva vi tror og vet om hvordan omgivelsene påvirker resultatene |
| **Kommentarer**<br>Ideer som kan dukke opp, og som ikke hører hjemme i de andre rutene | |

Utforming av arbeidsark

Universitetet i Stavanger

# GQM («Goal Question – Metrics»)

**Kommentarer**

Grunnlaget for å kunne formulere spørsmålene i arbeids arket er interjuer med personer som har kunnskaper om det målingen dreier seg om. Personene er de som tilhører *Perspektiv*, og som har kunnskap om *Objekt*. Hvis perspektiv er kunder må vi intervjue kunder. Hvis Perspektiv er utviklingsavdelingen må vi intervjue utviklere. Spørsmålene i arbeidsarker skal lede til forståelse når de blir besvart. Dette må det være enighet om. Derfor er det endelig utfylte arbeids arket resultatet av mange runder med diskusjon. Disse fører til ny innsikt og kunnskap. Det er styrken til arbeids arket.

GQM – prosessen skal ende opp med en måleplan. Den skal fortelle hvilke metrikker som er viktige, og data som må samles inn. Hvem som skal samle inn dataene, og hvordan dette skal gjøres.

Universitetet i Stavanger

# GQM («Goal Question – Metrics»)

Et eksempel

Vi er ansatt i et firma som utvikler programvare på oppdrag, firmaet ProgVirk. Konkurransen er stor og det er viktig å kunne

- levere til rett tidsdiagram
- lever til rett pris

Universitetet i Stavanger

# GQM («Goal Question – Metrics»)

Dette betyr at nøyaktige estimater er viktig. For lave anslag fører til at vi får prosjekter, men taper penger. For høye anslag vil føre til at vi ikke får oppdrag.

Estimerings-nøyaktighet er derfor svært viktig. Det vi estimerer, er hvor lang tid som trengs for å gjennomføre et utviklingsprosjekt, og hva det vil koste. Tiden angir vi i antall måneder. Kostnaden beregner i antall månedsverk. Estimerings-nøyaktighet definerer vi som forholdet mellom virkelig medgått tid eller kostnad og estimert tid eller kostnad.

Etter intervjuer med prosjektledere og utviklere dukker en rekke problemstillinger opp. Disse danner utgangspunktet for å formulere mål i GQM etter GQM – syntaksen og utfylling av arbeids arket. Vi tar med noen få spørsmål og metrikken for å vise prinsippet. Neste figur viser det utfylte arbeidsarker

# GQM («Goal Question – Metrics»)

| | |
|---|---|
| **Mål** | |
| Analysere utviklingsprosjekter for å finne faktorer som påvirker estimeringanøyaktigheten sett fra prosjektleder i utviklingsavdelingen i ProgVirk | |
| **Fokus** | **Omgivelser** |
| S1: Hva er estimeringsnøyaktigheten i dag?<br>S2: Hva er sammenhengen mellom omfanget av kravene og kostnadene ved utviklingen av systemet? | S3: Påvirker utviklernes erfaring utviklingskostnadene?<br>S4: Påvirker prosjektleders erfaring estimeringsnøyaktigheten |
| **Hva vi tror** | **Påvirkning** |
| S1: Mellom 0,8 til 2,0 med 1,5 som mest vanlig<br>S2: En ikke linær sammenheng som i COCOMO-modellen | Økt erfaring blant utviklerne reduserer utviklingskostnadene.<br>Økt prosjektledererfaring øker estimeringsnøyaktigheten |
| **Kommentarer** | |
| Det må undersøkes om COCOMO II er en egnet estimeringsmodell for ProgVirk | |

Universitetet i Stavanger

# GQM («Goal Question – Metrics»)

Med arbeids arket utfylt og enighet om innholdet kan vi finne egnede metrikker. Her er noen forslag.

Hvis vi med estimerings-nøyaktighet menes;

$$\frac{\text{virkelig kostnad}}{\text{estimert kostnad}}$$

Kan vi bruke disse til å besvare:

Universitetet i Stavanger

# GQM («Goal Question – Metrics»)

Besvare S1:

- M1: Virkelig kostnadene
- M2: Estimert kostnadene

For å besvare S2:

- M1: Virkelig kostnad
- M3: Antall objekter i domenemodellen

Besvarer S3:

- M1: Virkelig kostnad
- M4: Gjennomsnittlig antall år utviklingserfaring for utviklerne i utviklingsgruppen

Besvarer S4:

- Virkelig kostnadene
- Estimert kostnad
- Antall år som prosjektleder

Universitetet i Stavanger

# Statistiske verktøy

I ISO 9000:200 påpekes det at statistiske metoder er viktige for å forstå variasjoner i produkter og prosesser.

Statistisk analyse kan bidra til å avdekke problemer, finne årsaker til problemer, analysere ytelse, evaluere virkninger av endringer og forutsi ytelse.

Det finnes noen enkle statistiske verktøy som er velegnet for bruk i kvalitets og forbedringsarbeid.

De statistiske metodene som har vist seg å være mest nyttige er:

Universitetet i Stavanger

# Statistiske verktøy

- Lineær regresjon, kan gi svar på sannsynligheten for om det er sammenheng mellom datasett
- T-testen, kan gi svar på om det er forskjeller som ikke kan forklares ved naturlig variasjon
- ANOVA, kan gi svar på om forskjeller kan henføres til grupper av egenskaper
- kji-kvadrattesten, kan gi svar på om det er sannsynlig at forskjeller fra gang til gang er så store at det ikke skyldes tilfeldigheter

Det finnes støtte for disse i Excell. I boka «Bruk av enkle statistiske metoder i prosessforbedring» (Stålhane, 1998) er det eksempler på bruk av disse metodene.

## *Pensum MID130*

Lærebok: ( Gyldendal )

Kvalitet og programvareutvikling

Tora Berg Hansen og Greta Hjertø

ISBN 82-05-31143-8

Hele læreboka er pensum

# *Firma programutvikling*

## Sjekkliste for krav spesifikasjons rapport

Prosjekt navn:

**Gjennomlest dokument:** **Versjon:**

| Punkt nr. | Område | Ja | Nei | I. A. | Kommentar |
|---|---|---|---|---|---|
| **1** | **Dokumentet** | | | | |
| 1.1 | Utført og laget ut fra overordnede krav til prosessen | | | | |
| 1.2 | Strukturen er i samsvar med overordnede krav | | | | |
| 1.3 | Dokumentet er komplett | | | | |
| 1.4 | Har de nødvendige referanser til andre dokument | | | | |
| **2** | **Seleve spesifikasjonen** | | | | |
| 2.1 | De nødvendige funksjoner var utførlig dokumentert | | | | |
| 2.2 | Design innput samsvarer med krav til utdata | | | | |
| 2.3 | Kravet til programmet samsvarer med kravene til produktet | | | | |
| 2.4 | Grensesnittet mot eksterne pakker og eksternt miljø er klart og veldefinert | | | | |
| 2.5 | GUI grensesnittet er fullstendig og klart definert | | | | |
| 2.6 | Ytelseskrav: responstid, kapasitet til å ta inndata, lagrings kapasitet er korrekt og fullstendig beskrevet | | | | |
| 2.7 | Alle feilsituasjoner og systemets respons er korrekt definert og klart beskrevet | | | | |
| 2.8 | Datagrensesnitt mot eksisterende og planlagte programpakker er korrekt og klart spesifisert | | | | |
| 2.9 | Testprosedyrer for å teste at kravene er oppfylt er definerte og lette å forstå | | | | |
| **3** | **Prosjektets gjennomføring** | | | | |
| 3.1 | Er de spesifiserte kravene mulig å gjennomføre, tatt i betraktning prosjektresurser, budsjett timer osv. | | | | |
| 3.2 | Er det mulig å oppfylle kravene i 2.6, tatt i betraktning begrensninger påført av andre system og komponenter og eksterne system som samarbeider med dette systemet. | | | | |

Kommentar:

Navn: Dato: Signatur

# Timeplan

# MID130 – Kvalitetsikring i programvare 2006

| | Mandag | Tirsdag | Onsdag | Torsdag | Fredag |
|---|---|---|---|---|---|
| **8:15** | | | | | E-354 |
| **9:15** | | | | | E-354 |
| **10:15** | | | D-206 | | |
| **11:15** | | | D-206 | | |
| **12:15** | D-205 | | | | |
| **13:15** | D-205 | | | | |
| **14:15** | | | | | |
| **15:15** | | | | | |
| **16:15** | | | | | |