Velkommen del 2 av IN1030

- Introduksjon til systemutvikling
- Prosesser og prosessmodeller

UNIVERSITETET I OSLO

Dag Sjøberg og Gunnar Bergersen

About Me

- Current position: Professor at University of Oslo
  - Software Process Improvement,
    Agile and Lean Methods, Software Quality,
    Empirical Research Methods
- Education:
  - MSc, University of Oslo, 1987
  - PhD, University of Glasgow, Computing Science, 1993
- Work experience
  - University of Oslo since 1993
  - SINTEF ICT (20 %) 2014–2019
  - Simula Research Laboratory, 2001–2009
  - University of Oslo 1993–2000
  - Statistics Norway (SSB) 1987–1990
  - National University Hospital (Rikshospitalet) 1985–1986
- Startup
  - Member of steering committee and co-owner of four startup companies

About me

- Current:
  - Associate professor II at Programming & Software Engineering (20%)
  - Chief Product Officer Greps (skill testing of developers)
- Education
  - MSc (2001) and PhD (2015) from University of Oslo
  - PhD thesis: “Measuring programming skill”
- Prior work experience
  - Programmer
  - IT Project leader two companies
  - CEO three companies

Pensum 1: Lærebok

Ian Sommerville: Software Engineering, 10th ed., Pearson.

I dag:

Relatert til kap. 1–3

Pensum 2: Lysark fra forelesningene

- Inkluderer pensum som utfyller læreboka
- Undervisningen på universitetet skal være ”forskningsnær”. Derfor presenteres forskningsresultater som ikke står i læreboka
- Tekstbaserte, dvs. godt egnet for lesing til eksamen

Næringslivsperspektiver
Innspill til ifi's strategiprosess

Bjørn Taale Sandberg, Forskningsdirektør Telenor ASA

The Ideal candidate

- Has technical breath and depth and knows how to do modern development (cloud, agile, the 'right' languages and frameworks)
- Knows enough about Cybersecurity to write safe and secure applications
- Understands the domain (i.e. business insight) and how SW can be used to transform business, including 'sorting out legacy'…
- Has practical experience 'working in the real world'
- Can build solutions by writing own code, using microservices and platforms
- Has 'power skills' (used to be 'soft skills'):
    - Can talk to business managers and customers
    - Can communicate (present / public speaking)
    - Has a growth mindset
    - Understands how to work with others, locally and in an ecosystem

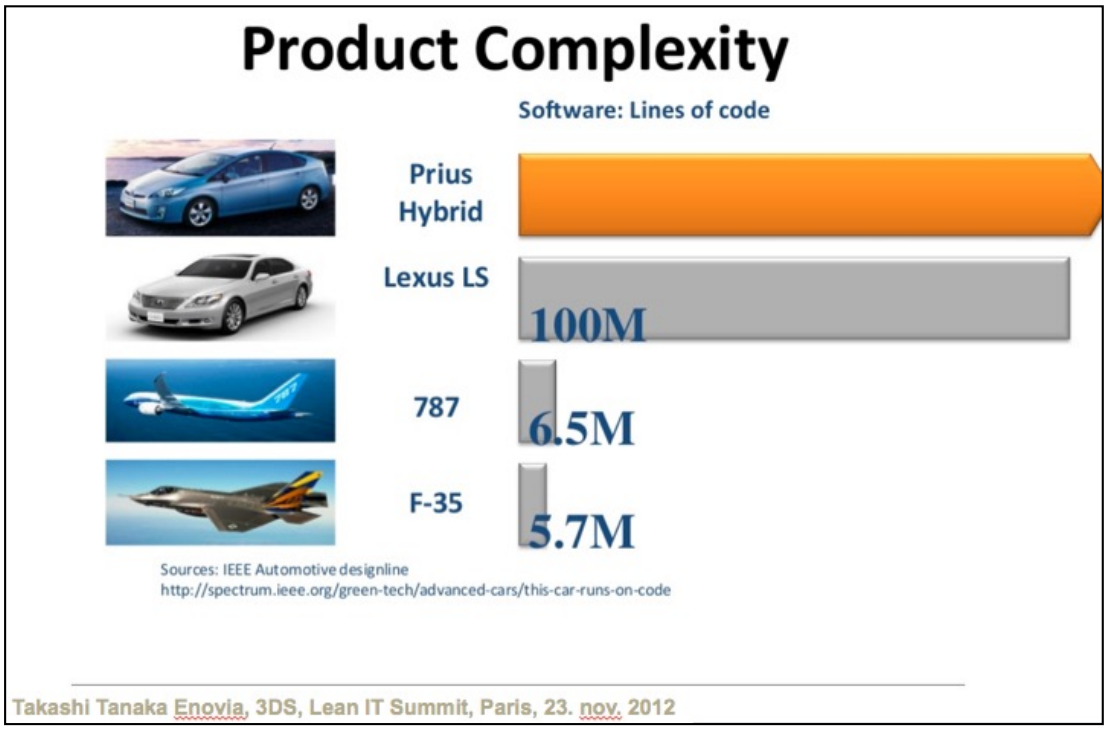
Plan for forelesningen

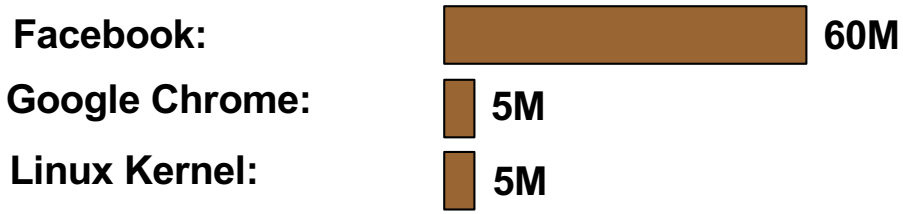
- Læremidler
- Kap. 1: introduksjon
  - IT-systemer og systemutvikling
  - Eksempel på systemutvikling
- Kap. 2: Systemutviklingsprosessen – hvordan jobbe smart i IT-prosjekter
  - Prosessmodeller (modeller for systemutviklingsprosesser)
  - Fossefallsmodellen
- Kap. 3: Smidige metoder
  - Tidsboksbasert (Scrum)
  - Flytbasert (Kanban)
  - Eksempler på praksiser ved smidig utvikling

Programvare finnes nærmest overalt!

Variasjon i størrelse og kompleksitet

- Programvaren
  - Fra noen få tusen til mange millioner linjer kode
- Utviklingsteamene
  - fra enkeltpersoner til over 1000 utviklere (MS Windows)
- Kostnad utvikling og vedlikehold
  - fra noen tusen kroner til flere milliarder

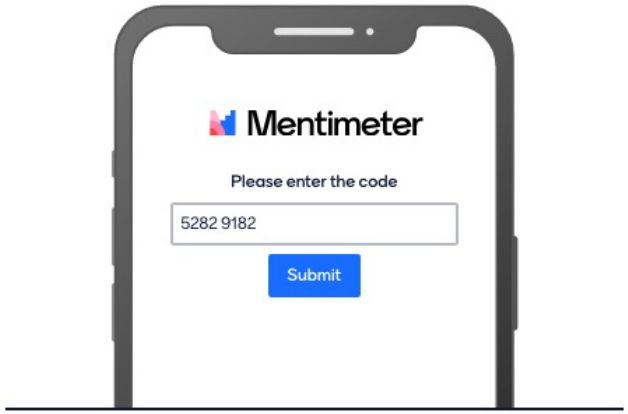




Hva med deg?

Go to

www.menti.com



Enter the code

5282 9182

Or use QR code

IT-skandaler: Aftenposten

Utviklingen går ikke så fort som folk tror.
IT-skandaler i det offentlige må man regne med i mange tiår fremover, advarer fagfolk.

Skandalene er kommet for å bli

IT OG POLITIKK

HALVOR HEGTUN

Dataskandalene har i tur og orden hjemsøkt trygdeetaten, skatteetaten, utenrikstjenesten, sporveien, Nav, politiet.

Det aller første Gro Harlem Brundtland gjorde da hun høsten 1996 ga seg som statsminister, var å melde seg på PC-kurs på Stortinget sammen med den avtroppende finansministeren Sigbjørn Johnsen.

Så sent som for 16 år siden var det altså mulig å styre Norge uten dataskjerm på kontorpulten. Men de store dataskandalene var nettopp på denne tiden på full fart inn i politikkens verden.

Stortingets aller første offentlige høring, i mai 1996, dreide seg om hvordan trygdeetaten hadde sløst bort hundrevis av millioner kroner på datasystemet Tress-90. Trøbbelprosjektene skulle komme på løpende bånd, ofte med tunge politiske etterspill.

Trøbbel overalt

– Helt siden jeg kom inn på Stortinget i 1997, har vi holdt på med dataproblemer. I alle komiteer jeg har vært, utbrøt justiskomiteens leder Per Sandberg (Frp) da 22. julikommisjonen hadde avlevert sin rapport. «IKT-infrastrukturen i politiet må karakteriseres som lite hensiktsmessig», fastslo kommisjonen. Frp har foreslått en egen havarikommisjon for dataskandaler. Skal det da aldri ta en slutt?

Svaret er nei. I hvert fall når spørsmålet stilles i det nye og staselige informatikkbygget på Blindern.

Naiv optimisme

– Det er lett å tro at den teknologiske utviklingen går så rasende fort på alle områder. Du kan høre folk si at «om ti år er ikke dette med data noe problem lenger». Men det er fryktelig naivt, sier professor Dag Sjøberg.

– Men utviklingen går da rasende fort?

– Kapasiteten og hastigheten i datamaskiner har doblet seg annethvert år. Men den menneskelige IQ utvikler seg ikke like fort. De fundamentale problemene vedvarer: Hvordan man lager IT-systemer som løser brukernes behov og samtidig håndterer kompleksiteten, sier Sjøberg.

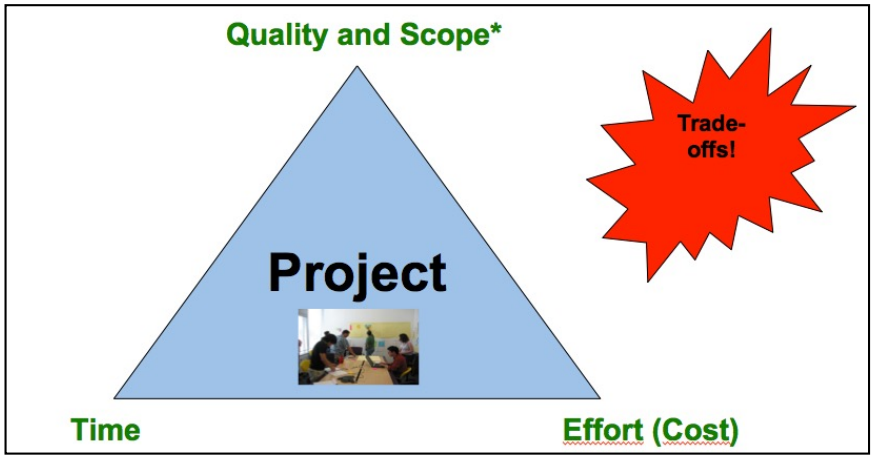
– En liten smarttelefon har visst større datakapasitet enn Apollo-fartøyene som tok seg til månen?

– Ja, det er fantastisk hva du kan gjøre med telefonen din. Men det sitter ikke tusener av programmerere i Apple og Microsoft og utvikler styringssystemer for Oslo universitetssykehus. Det er voldsomt komplekse problemstillinger. Nye systemer skal fases inn og fungere i samspill med gamle systemer, som ikke kan hives ut over natten. Det ligger pasientopplysninger på flere hundre ulike datasystemer, anslår Sjøberg.

Aftenposten 8. mars 2021

- **"Riksrevisjonen setter direktorat under lupen. Skal granske all bruk av konsulenter. Nøyer seg ikke med å se på det omstridte IT-prosjektet Akson*."**
- "Direktoratet for e-helse har vært i hardt vær det siste året. De har hatt ansvaret for det mye omtalte Akson-prosjektet. Det skal gi fastleger, legevakter og helsestasjoner et felles journalsystem. Prosjektet er omstridt. Ulike aktører mener prosjektet er for **dyrt**, for **stort** og vil ta for lang **tid**."

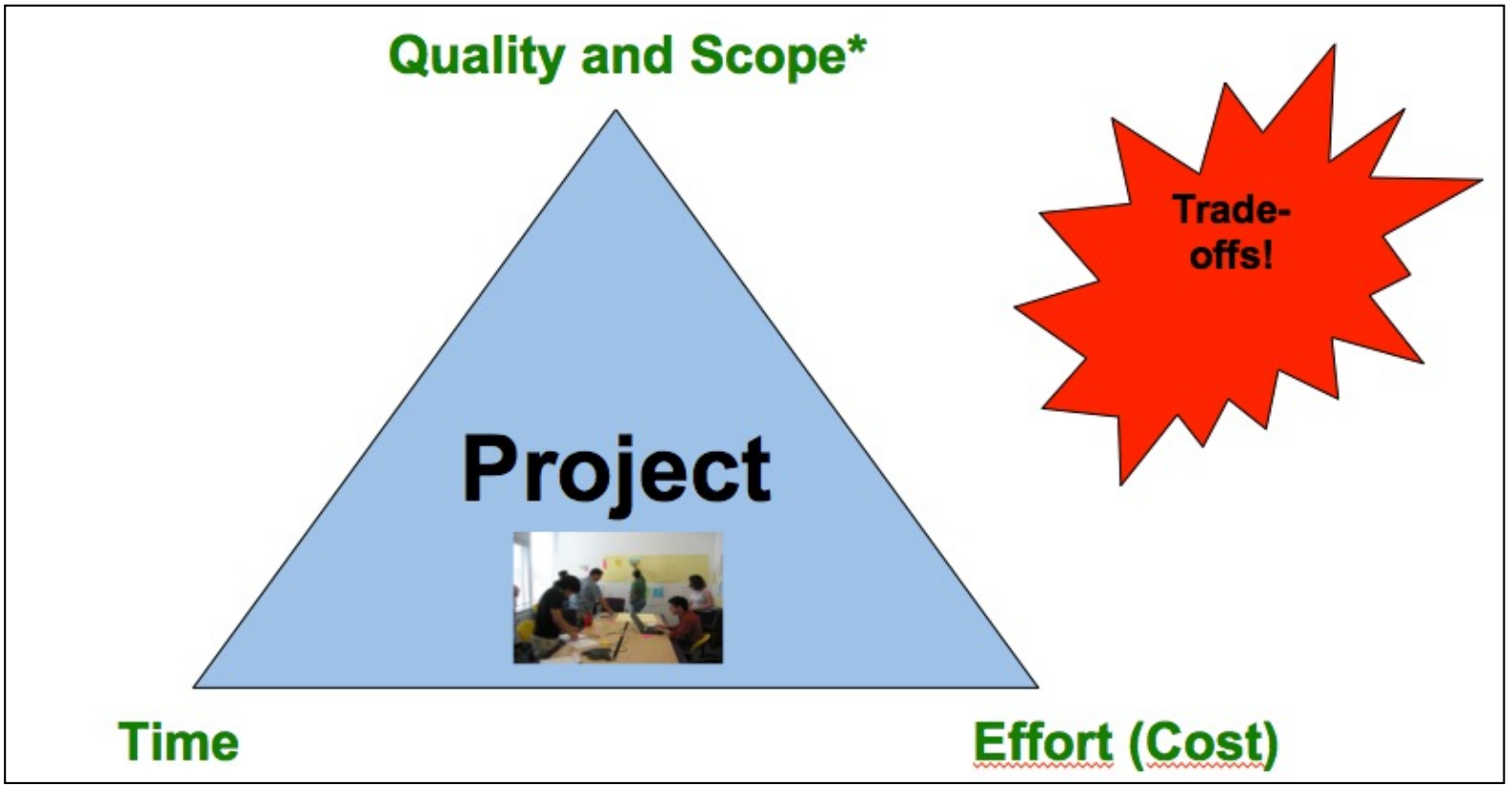
*Akson – felles kommunal journal og helhetlig samhandling



Stor usikkerhet i IT-prosjekter

- Få prosjekter blir skandaler, men
- mange prosjekter har stor usikkerhet når det gjelder
  - tid
  - kostnader
  - levert funksjonalitet og
  - kvalitet

”Magisk triangel”/prosjekt-triangel



*Scope: hvor mye funksjonalitet systemet omfatter

Definisjon: Systemutvikling (software engineering)

- – er læren om utvikling og forvaltning av IT-systemer av høy kvalitet innen gitte tids- og kostnadsrammer
- Viktige kvalitetsegenskaper er funksjonell egnethet «hva kan systemet gjøre», effektivitet, pålitelighet, brukskvalitet, vedlikeholdbarhet og sikkerhet
- Typiske aktiviteter er planlegging, kravinnsamling og kravanalyse, design, programmering/koding, testing, konfigurasjonsstyring og versjonshåndtering

Hvilke konsekvenser har det at systemer er av ulike typer?

- Ulike typer systemer har ulike egenskaper og stiller ulike typer krav
- Systemene må derfor utvikles på ulike måter
- Hva vil være forskjellig ved utvikling et fly/tog-kontrollsystem og web-system som gir oversikt over Ifi sine kurs?

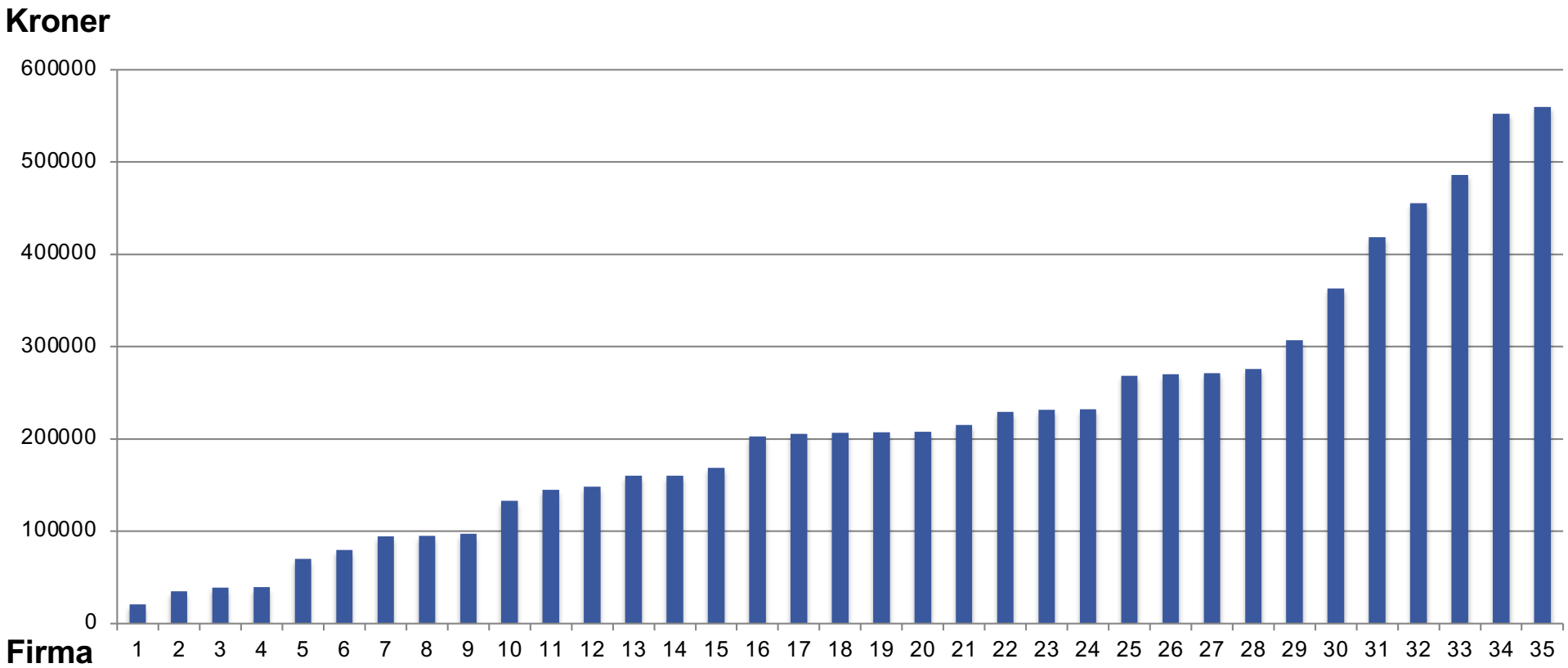
Plan for forelesningen

- Læremidler
- Kap. 1: introduksjon
  - IT-systemer og systemutvikling
  - Eksempel på systemutvikling
- Kap. 2: Systemutviklingsprosessen – hvordan jobbe smart i IT-prosjekter
  - Prosessmodeller (modeller for systemutviklingsprosesser)
  - Fossefallsmodellen
- Kap. 3: Smidige metoder
  - Tidsboksbasert (Scrum)
  - Flytbasert (Kanban)
  - Eksempler på praksiser ved smidig utvikling

Variasjon i systemutvikling

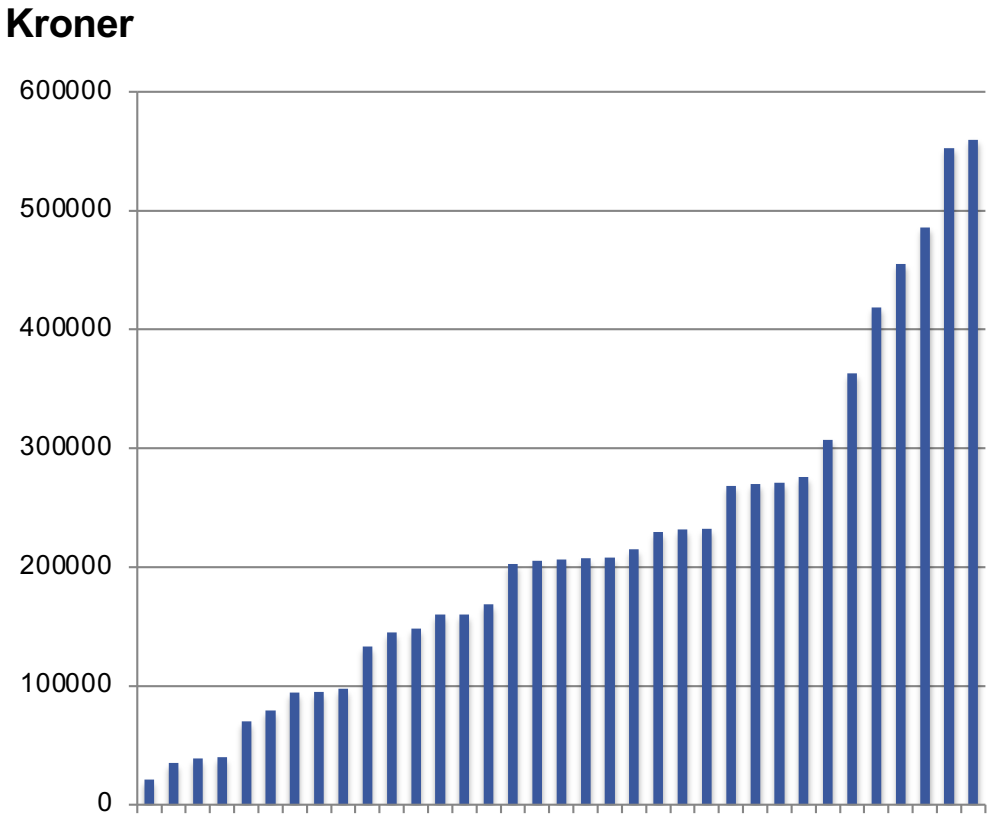
- 81 firmaer invitert til å komme med anbud på et lite web-basert informasjonssystem
- 35 firmaer la inn anbud
- Hvor stor variasjon i pristilbud?

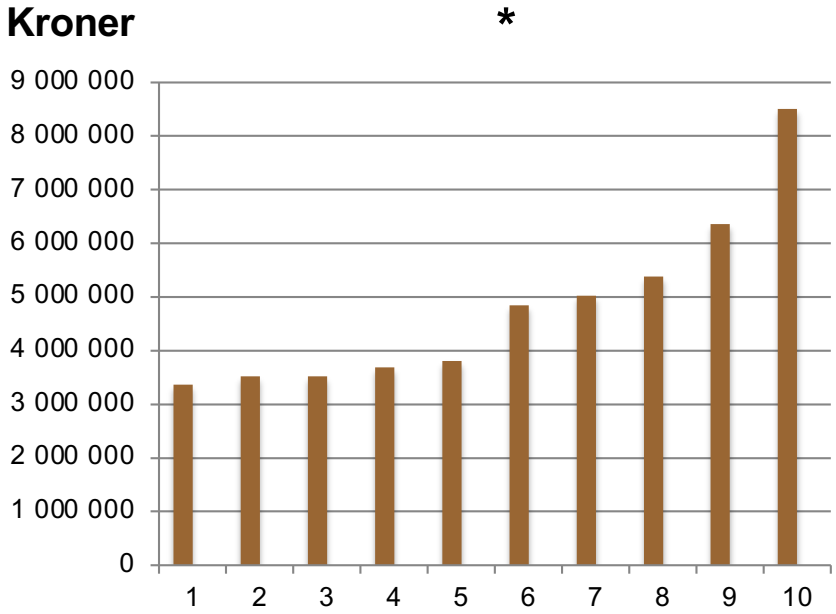
Fra kr. 21.000 til kr. 560.000 !



Variasjon anbud IT-prosjekter versus veiprosjekter

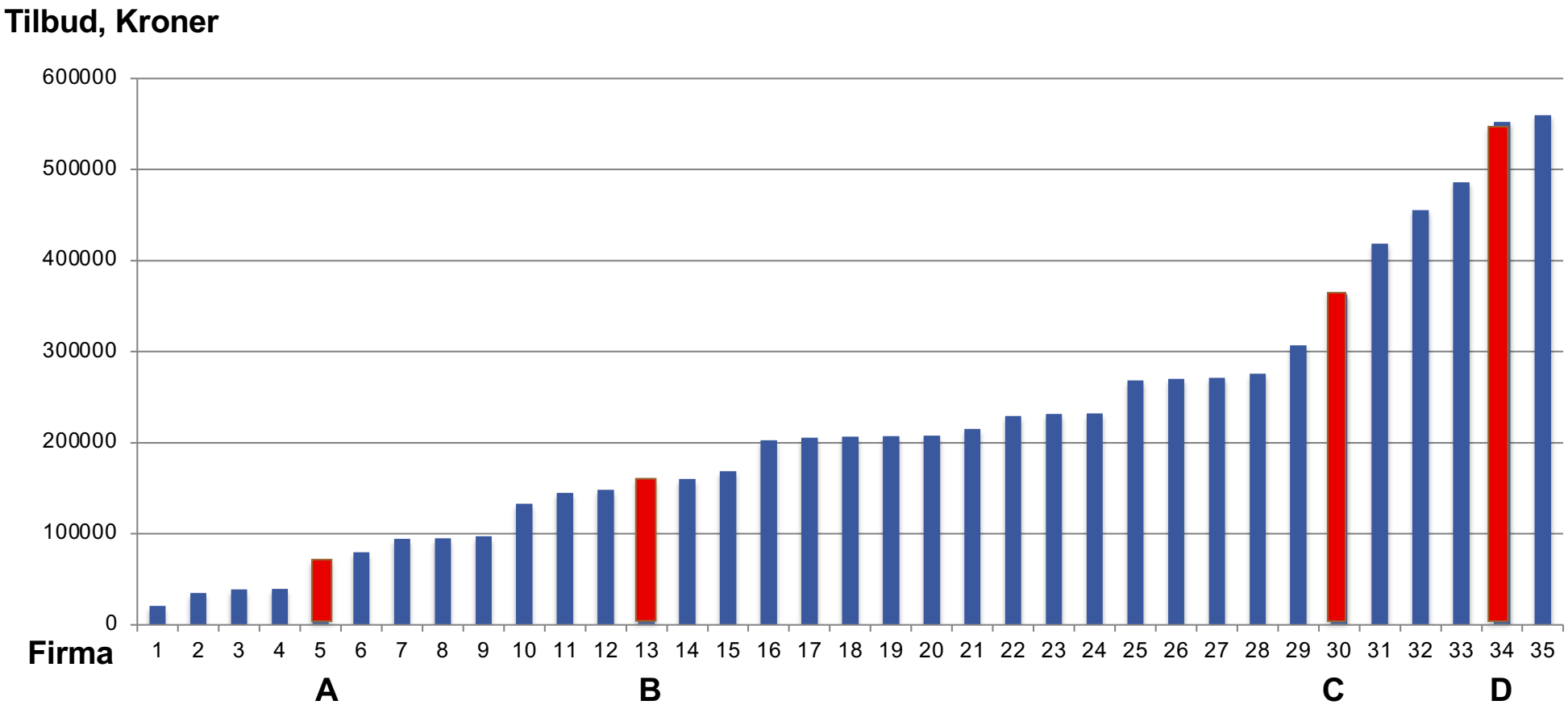
3 X





*H. Pedersen, "Tender Prices: Bridge, Tunnel, Electro and Road Building and Maintenance 1998-2006", Technology Report 2468, Norwegian Public Roads Administration, 2006

Spennende studie – valgte fire firmer istedenfor ett!



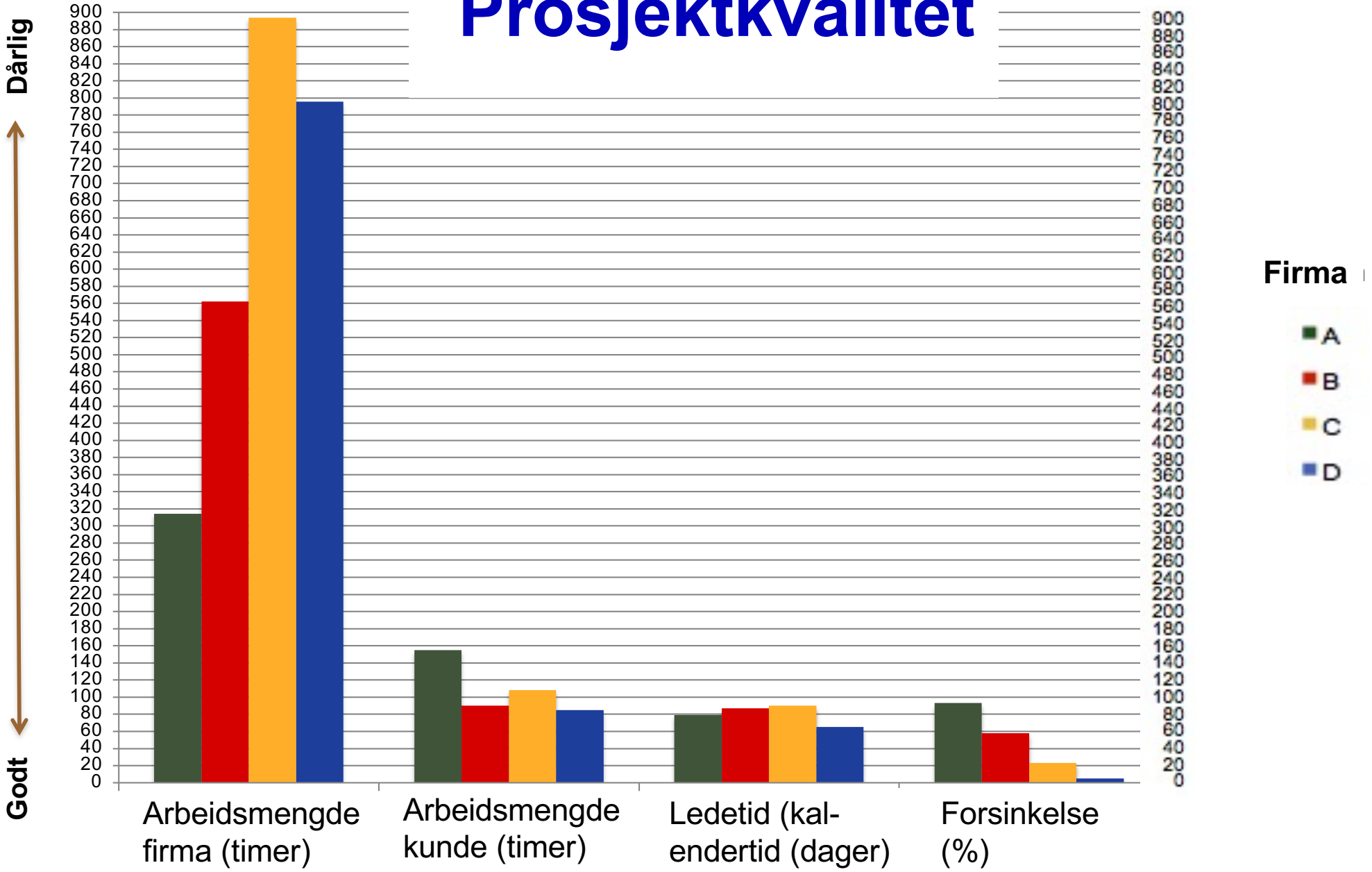
B.C.D. Anda, D.I.K. Sjøberg and A. Mockus. Variability and Reproducibility in Software Engineering: A Study of four Companies that Developed the same System, *IEEE Transactions on Software Engineering* 35(3):407-429, 2009

Hvilket firma gjorde det ”best”?

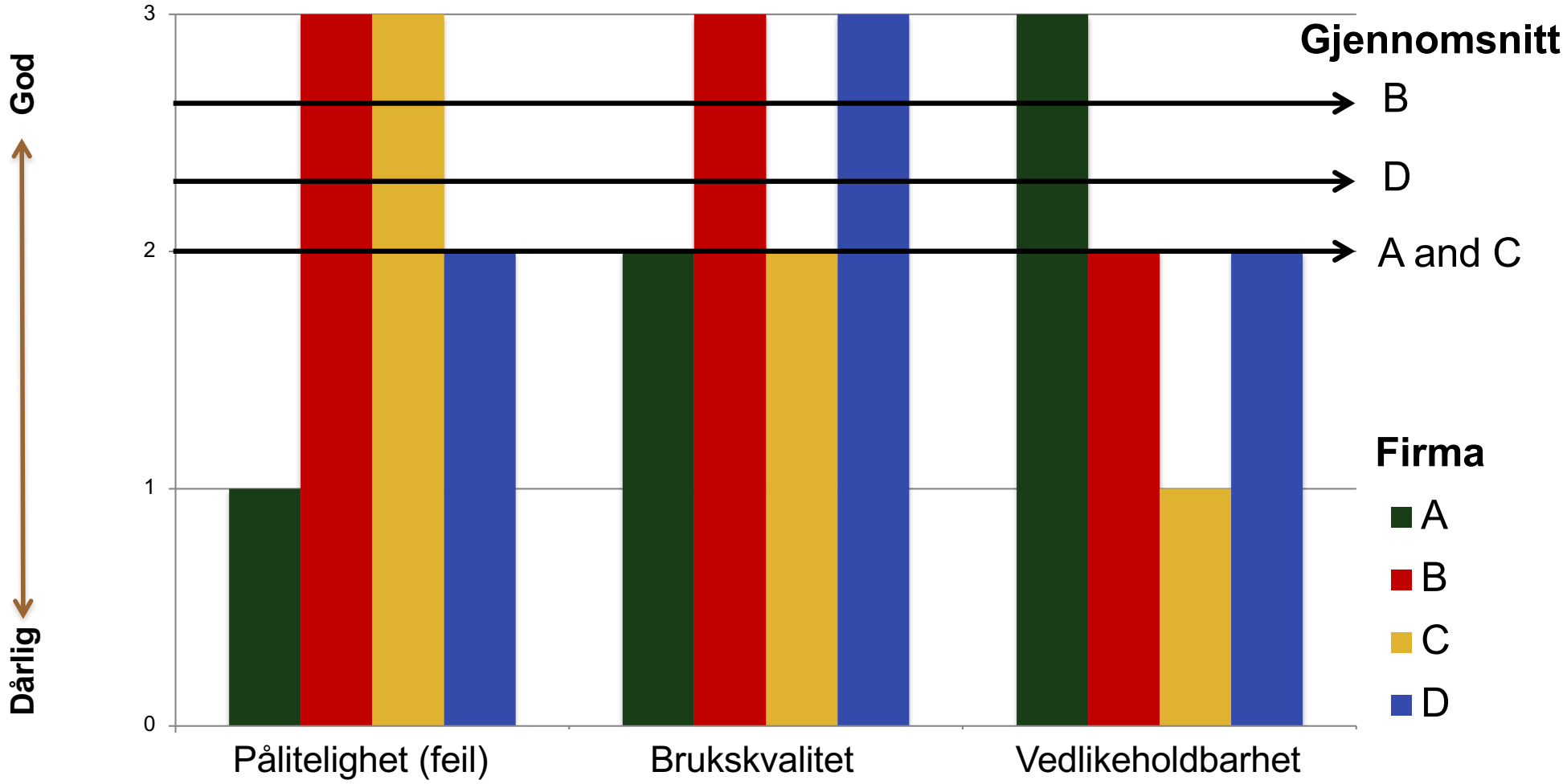
Hva betyr ”best”?

Sammenheng mellom pris/arbeidsmengde og kvalitet?

Prosjektkvalitet



Systemkvalitet



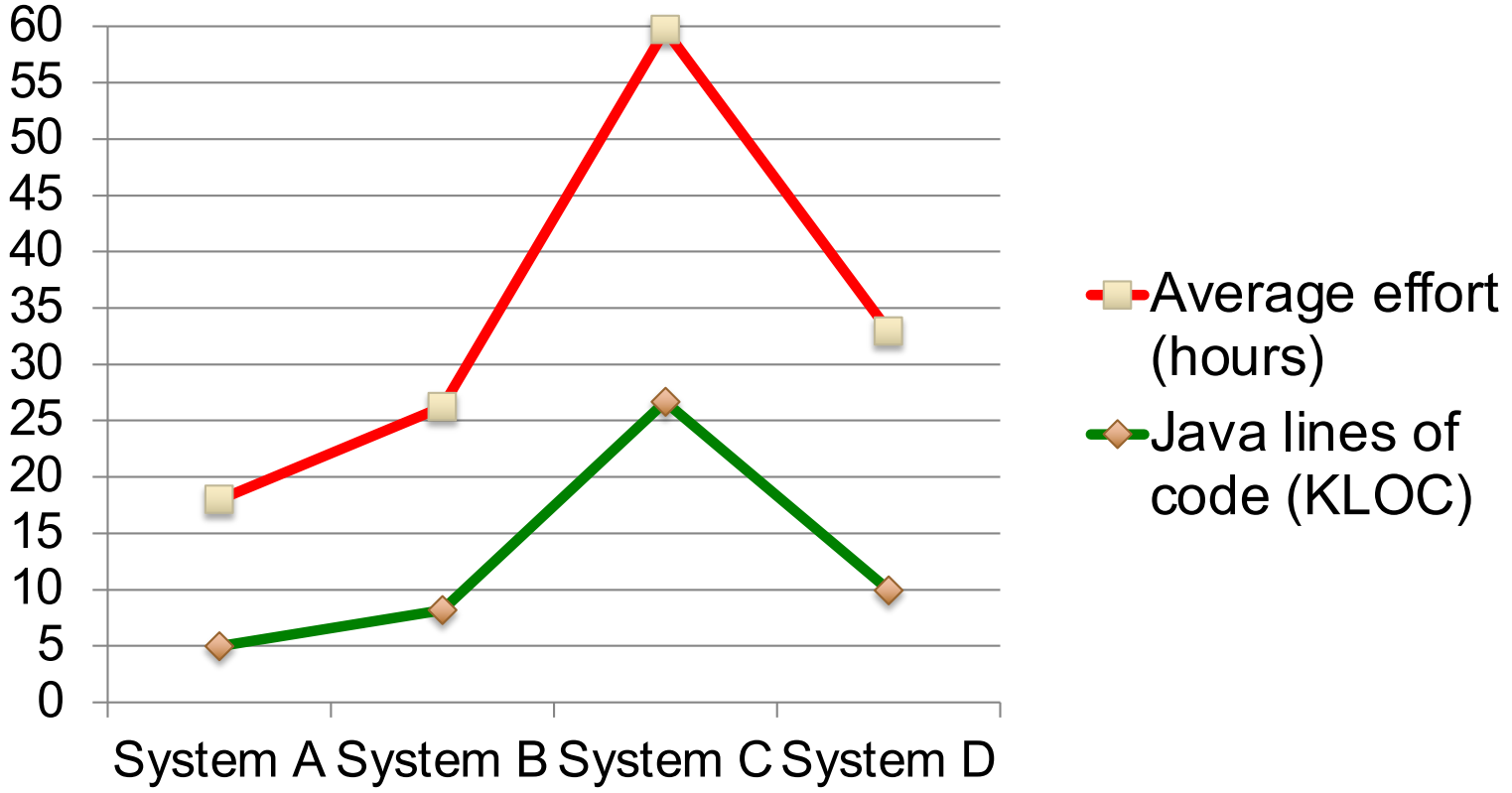
Vedlikehold (forvaltning)

- Vedlikehold av IT-systemer betyr ikke å bevare originalversjonen mest mulig slik som for bil, hus etc.
- Vedlikehold er alle endringer utført på et system etter at det er satt i drift
- Utgjør 50%–90% av kostnadene i levetiden til et system
- Stor andel av nedarvede (“legacy”) systemer: bank, forsikring, offentlige etater

Alle fire systemene satt i drift i parallell. Etter to år, behov for vedlikehold:

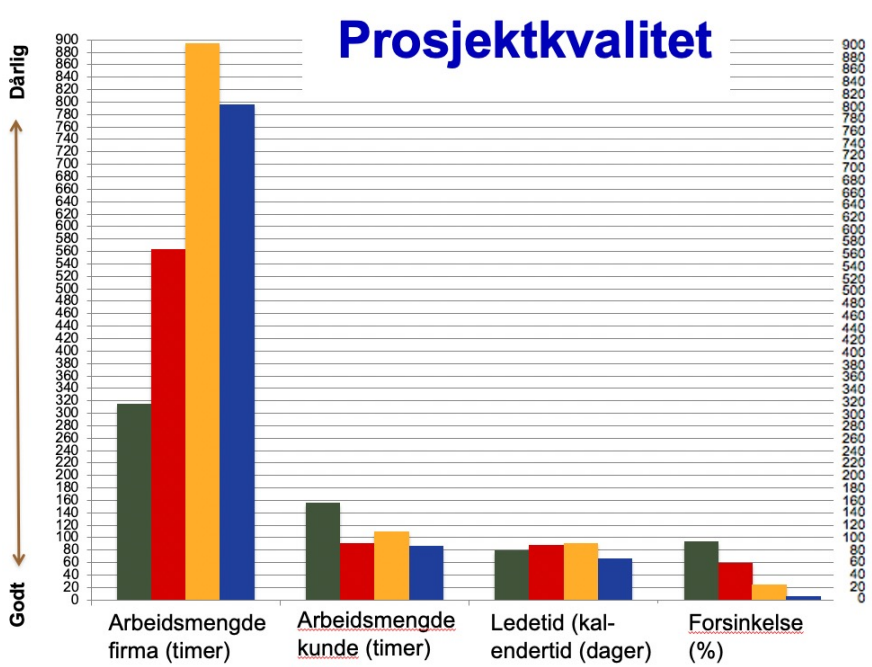
- Tre endringer
  - To tilpasninger etter plattformendringer
  - Ett ønske om ny funksjonalitet
- Leide inn 6 utviklere fra Tsjekkia og Polen som hver utførte vedlikeholdsoppgavene på 2 av systemene
  - Brukte ca. 3 uker hver på hvert system
  - kostnad kr. 400.000
- Tid brukt på hver fil ble automatisk registrert

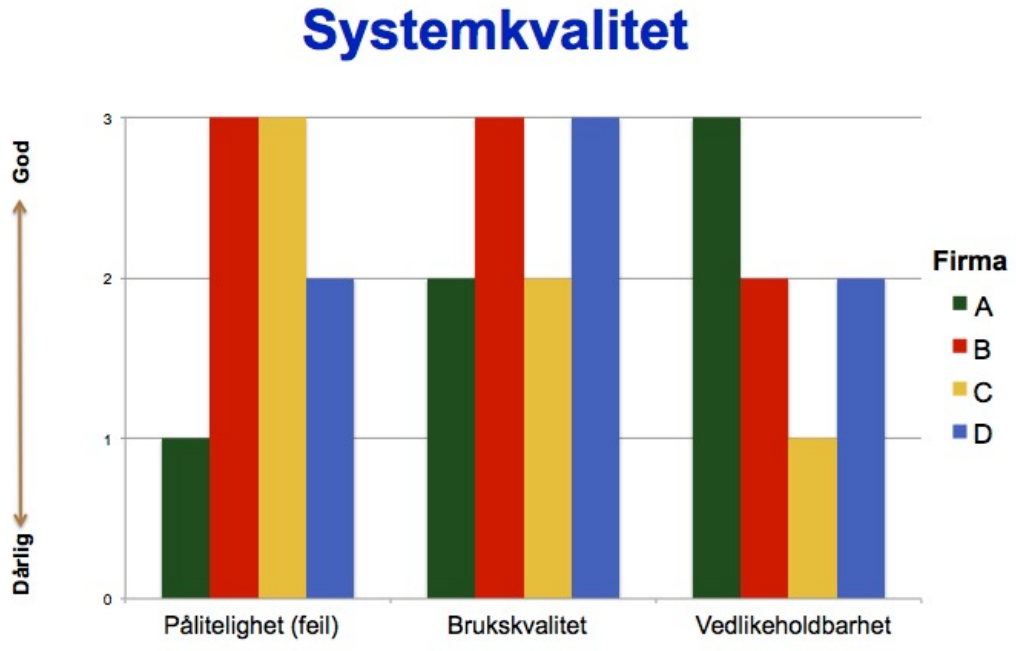
Størrelse versus vedlikehold



Programmeringstips: Etter at du har skrevet en kodebit, tenk igjennom om du kan skrive samme funksjonalitet på en kortere og mer forståelig måte

Hva var årsaken til forskjellene?





- Ulike måter å jobbe på, dvs. ulike systemutviklingsprosesser, gir ulike resultater
- Viktig å være bevisst hvilke kvalitetsegenskaper man ønsker å vektlegge

Plan for forelesningen

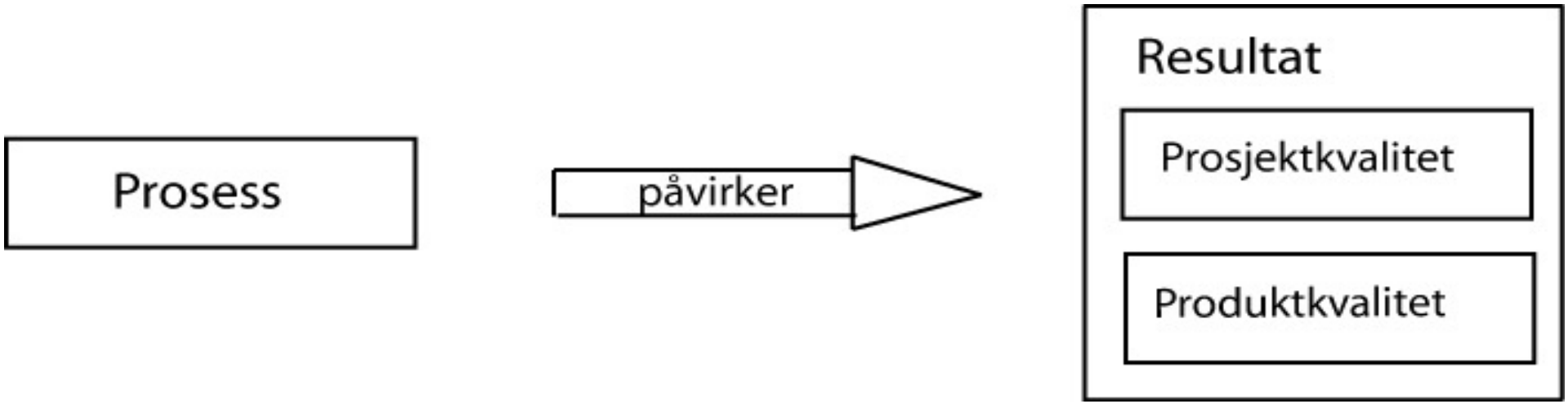
- Læremidler
- Kap. 1: introduksjon
  - IT-systemer og systemutvikling
  - Eksempel på systemutvikling
- Kap. 2: Systemutviklingsprosessen – hvordan jobbe smart i IT-prosjekter
  - Prosessmodeller (modeller for systemutviklingsprosesser)
  - Fossefallsmodellen
- Kap. 3: Smidige metoder
  - Tidsboksbasert (Scrum)
  - Flytbasert (Kanban)
  - Eksempler på praksiser ved smidig utvikling

Systemutviklingsprosess

- – er de aktivitetene som utføres for å utvikle et IT-system
- Aktivitetene varierer, men vil alltid ha elementer av
  - spesifisering av kravene, dvs. hva systemet skal gjøre
  - design av systemet (for eksempel lage en datamodell)
  - implementering av koden (programmering)
  - validering av at systemet gjør det kunden ønsker
  - endringer av systemet i forhold til nye og endrede krav hos kunden

Prosessen påvirker resultatet

- Systemutviklingsprosessen vil påvirke kvaliteten både på prosjektet selv og systemet som utvikles
- Måten man jobber på påvirker også arbeidsmiljøet (trivsel, motivasjon, kompetanseutvikling etc.) som igjen påvirker prosjekt- og produktkvalitet
- Din kompetanse og måten du og ditt team jobber på avgjør hvordan prosjektet og sluttproduktet blir!

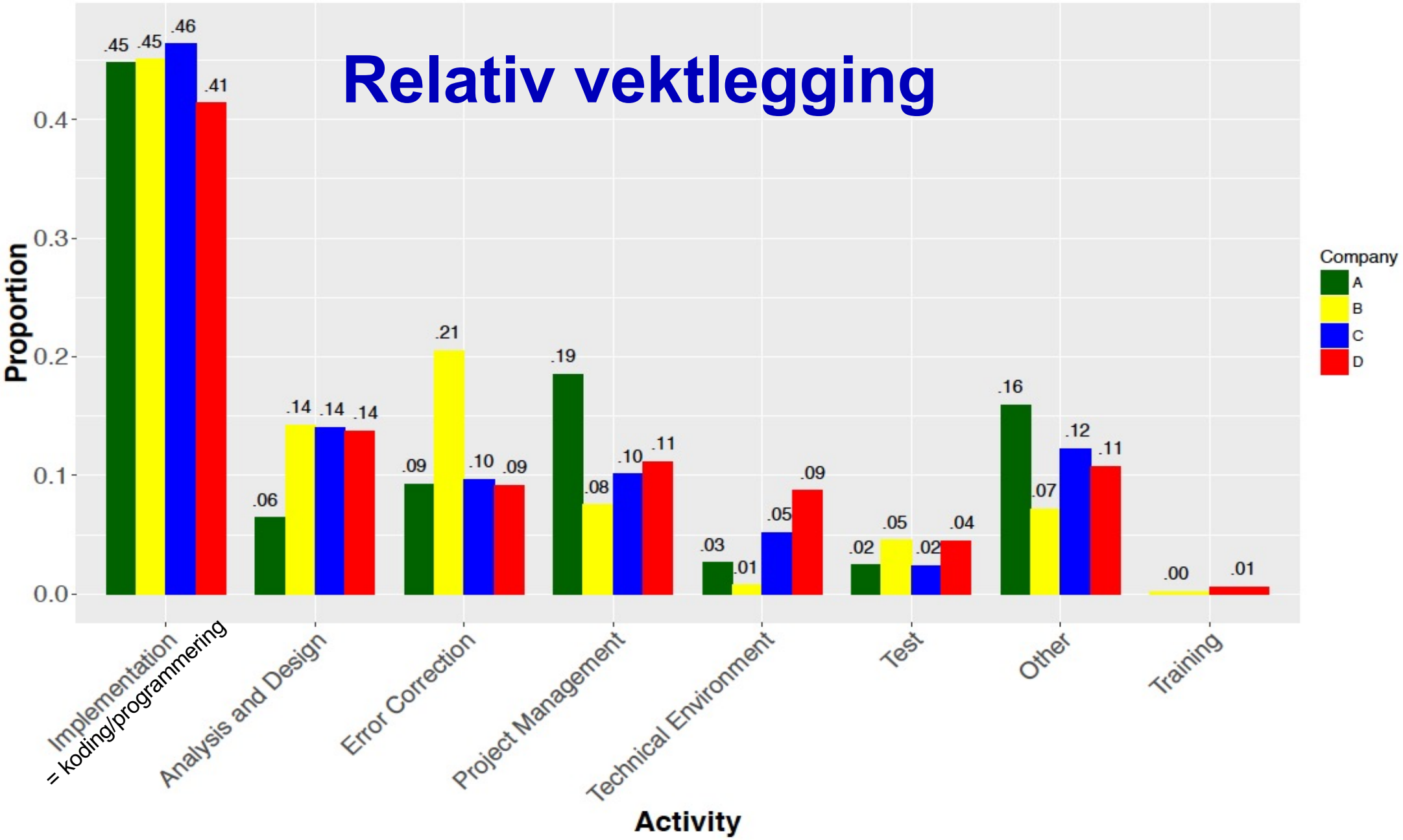


Prosess-egenskaper

- Hvilke aktiviteter inngår i prosessen?
- Hvor mye av hver aktivitet (absolutt og relativt i forhold til hverandre)?
- Når i utviklingsfasen gjøres (hvor mye) av hver aktivitet?
- Prosessbeskrivelser vil også kunne inneholde
  - delprodukter/resultater av en aktivitet
  - rollene til dem som er involvert i prosessen
  - hvordan teamene organiseres (man jobber sjelden alene)
  - metoder, verktøy og teknikker som brukes

Aktiviteter i de fire firmaene

Relativ vektlegging



Eksempel på roller

- Utvikler
- Vedlikeholder
- Arkitekt/system designer
- Grafisk designer
- Tester
- Prosjektleder
- Bruker-/kunderepresentant

Ikke trivielt å besette et prosjekt med den riktige kompetansen!

Eksempel på verktøy

Verktøy for:

- Utvikling (IDE – Integrated Development Environment)
- Konfigurasjonsstyring/endringshåndtering
- Testing
- Diagramkonstruksjon
- Prosjektstyring
- Feil- og problemhåndtering (bug & issue tracking)

Valg av verktøy er heller ikke trivielt!

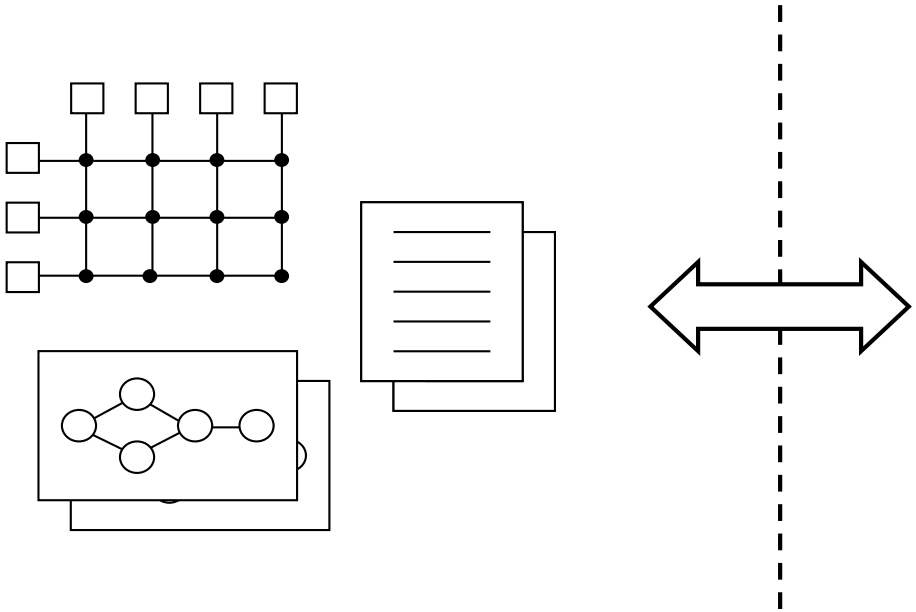
Plan for forelesningen

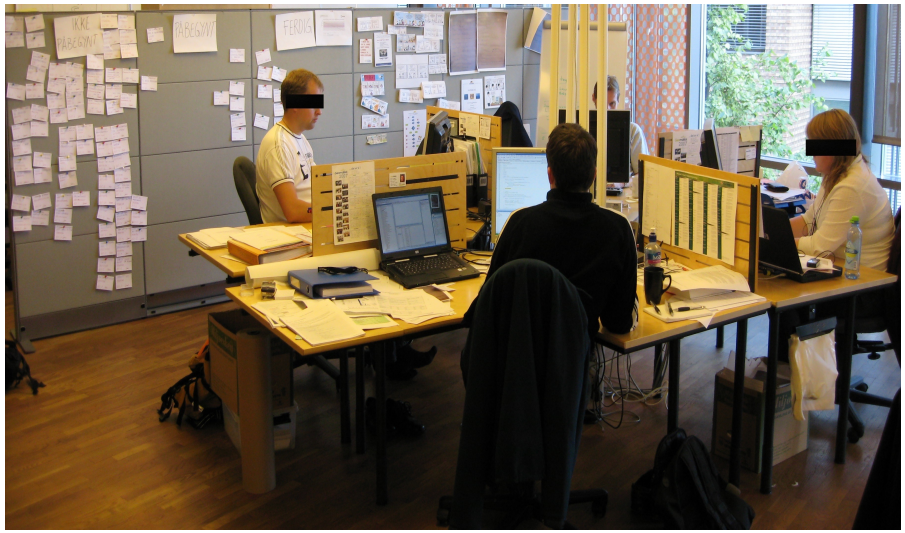
- Læremidler
- Kap. 1: introduksjon
  - IT-systemer og systemutvikling
  - Eksempel på systemutvikling
- Kap. 2: Systemutviklingsprosessen – hvordan jobbe smart i IT-prosjekter
  - Prosessmodeller (modeller for systemutviklingsprosesser)
  - Fossefallsmodellen
- Kap. 3: Smidige metoder
  - Tidsboksbasert (Scrum)
  - Flytbasert (Kanban)
  - Eksempler på praksiser ved smidig utvikling

Reell prosess versus modell av prosess

- Systemutviklingsprosess (= faktisk, reell prosess):
  - de aktivitetene som utføres i et utviklingsprosjekt
- Prosessmodell (=formell prosess) (kalles også gjennomføringsmodell)
  - En abstrakt, forenklet representasjon av en prosess
    - Normativ (preskriptiv)
      - beskriver en prosess slik noen mener den *bør* være (vanligste betydning), dvs.”oppskrift” på hvordan jobbe
    - Deskriptiv
      - beskriver en prosess slik vi mener vi utfører den

Formell versus reell prosess

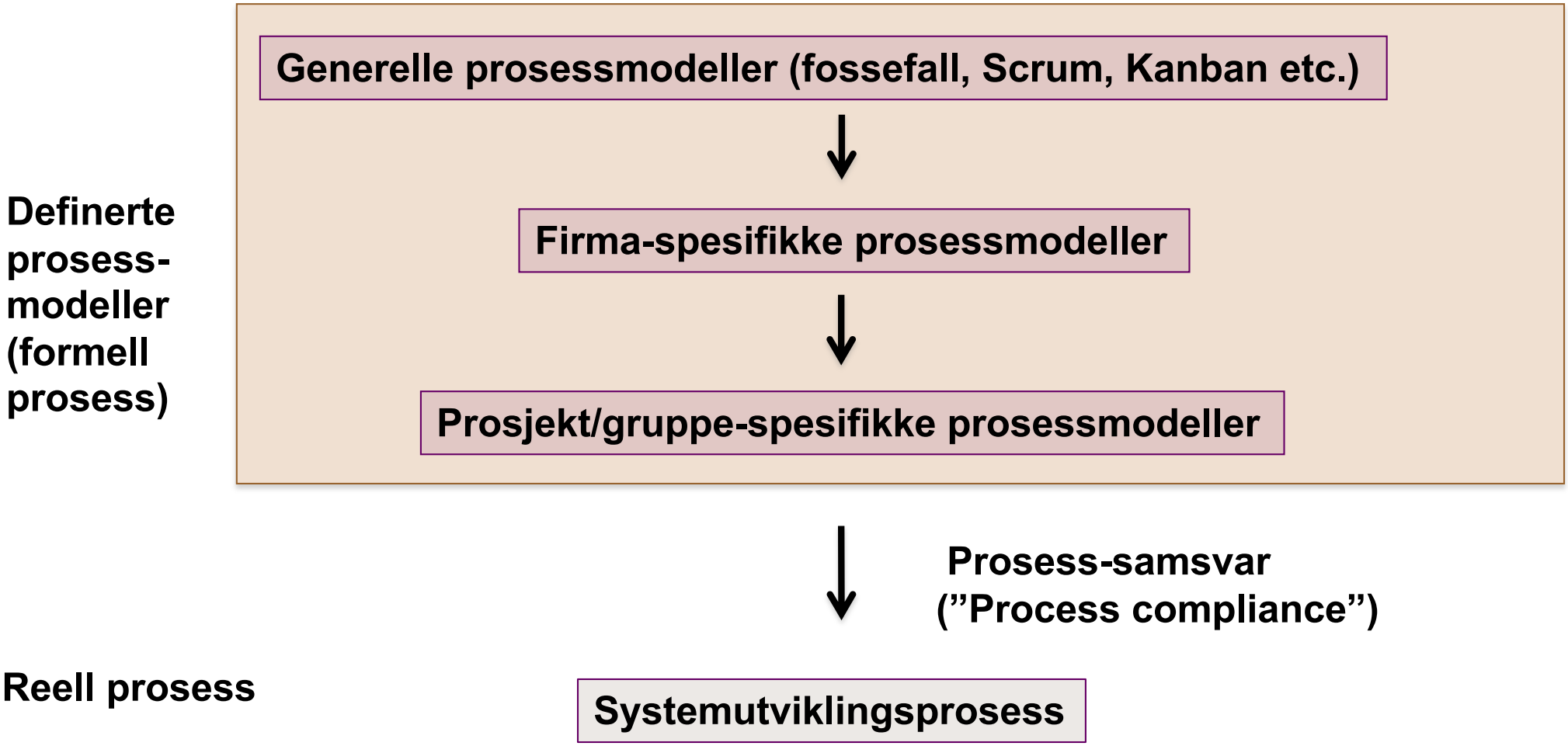




Prosessbeskrivelse

Prosessutførelse

Nivåer av prosessmodeller



En overordnet prosjektmodell

Digitaliseringsdirektoratet
Prosjektveiviseren

Søk

Hva er Prosjektveiviseren? | Prosjekttyper | Roller | Dokumentasjon | God praksis | Begreper

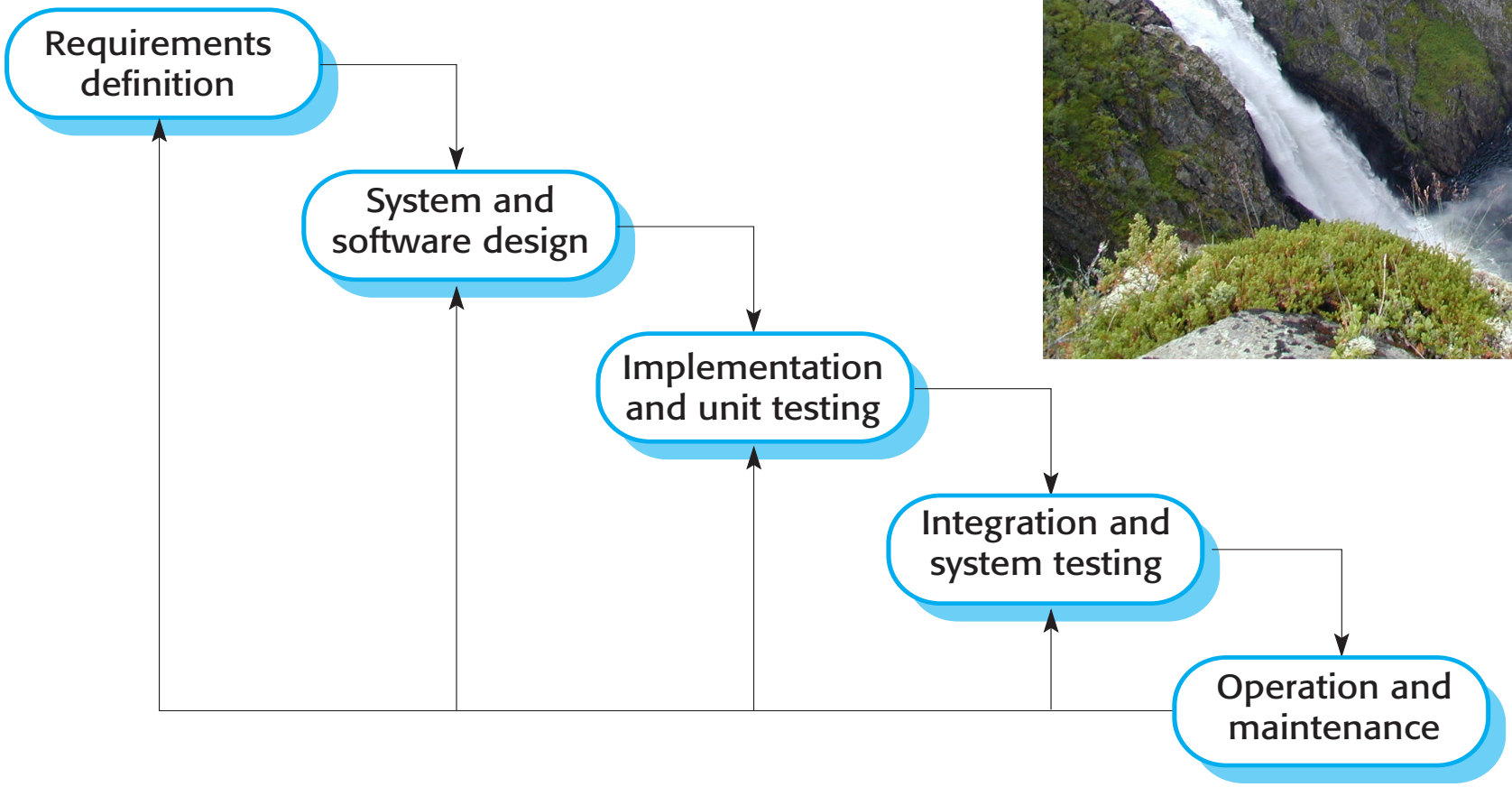
Hvordan tilpasse prosesser?

- Prosesser må tilpasses – ingen prosjekter er like
  - Mange faktorer påvirker prosessen
- Hva kan tilpasses?
  - Faser/aktiviteter, roller, ansvarsforhold, utforming/frekvens på rapporter og gjennomganger
- Hvordan tilpasse?
  - Identifiser prosjektomgivelser – utviklingsstrategi, risiko, krav, applikasjonsområde, type kunde etc.
  - Innhent synspunkter fra utviklere, brukere, kunder
  - Definer prosesser, aktiviteter og roller
  - Begrunn og dokumenter tilpasningene

Plan for forelesningen

- Læremidler
- Kap. 1: introduksjon
  - IT-systemer og systemutvikling
  - Eksempel på systemutvikling
- Kap. 2: Systemutviklingsprosessen – hvordan jobbe smart i IT-prosjekter
  - Prosessmodeller (modeller for systemutviklingsprosesser)
  - Fossefallsmodellen
- Kap. 3: Smidige metoder
  - Tidsboksbasert (Scrum)
  - Flytbasert (Kanban)
  - Eksempler på praksiser ved smidig utvikling

Fossefallsmodellen



I prinsippet går man ikke tilbake til tidligere hovedaktiviteter før systemet er satt i drift.

Fossefallsmodellen – klassisk ingeniørtilnærming

- Sivilingeniørprosjekter (bygg & anlegg) er *plandrevet*, dvs. relativt mye tid på planlegging og utviklingen dokumenter som styrer prosjektet
- Separate faser
- Vanskelig å tilpasse endringer i krav underveis

Plan for forelesningen

- Læremidler
- Kap. 1: introduksjon
  - IT-systemer og systemutvikling
  - Eksempel på systemutvikling
  - Kap. 2: Systemutviklingsprosessen – hvordan jobbe smart i IT-prosjekter
  - Prosessmodeller (modeller for systemutviklingsprosesser)
  - Fossefallsmodellen
  - Kap. 3: Smidige metoder
  - Tidsboksbasert (Scrum)
  - Flytbasert (Kanban)
  - Eksempler på praksiser ved smidig utvikling

Behov for smidighet

- Klassisk ingeniørtilnærming vist seg ofte å ikke være egnet
- Derfor er “smidige metoder” nå blitt det normale
  - Vektlegging av kode fremfor (omfattende) design og dokumentasjon
  - Vektlegging av samarbeid med kunde fremfor kontraktsforhandlinger
  - Raskere levering av kode og raske endringer tilpasset endrede brukerkrav

En samling software-guruer i 2001: Agile Manifesto – 12 prinsipper*

1	Kundefokus	Prioriter å tilfredsstille kunden gjennom tidlige og kontinuerlige leveranser av programvare med verdi
2	Positiv til endringer	Ønsk endringer i krav velkommen, selv sent i utviklingen. Bruk endringer til å skape konkurransefortrinn for kunden
3	Lever hyppig	Lever fungerende programvare med et par ukers til et par måneders mellomrom. Jo oftere, jo bedre
4	Kunde-involvering	Forretningssiden og utviklerne må arbeide sammen daglig gjennom hele prosjektet
5	Stol på den enkelte	Bygg prosjektet rundt motiverte personer. Gi dem og miljøet støtten de trenger. Stol på at de får jobben gjort
6	Ansikt-til-ansikt kommunikasjon	Mest effektivt å formidle informasjon til og innad i et utviklingsteam ved å snakke ansikt til ansikt
7	Kode er hovedproduktet	Fungerende programvare er det primære målet på fremdrift
8	Stabile omgivelser	Smidighet fremmer bærekraftig programvare-utvikling. Sponsorene, utviklerne og brukerne bør kunne opprettholde et jevnt tempo hele tiden
9	Teknisk kvalitet	Kontinuerlig fokus på fremragende teknisk kvalitet og godt design fremmer smidighet
10	Enkelhet	Enkelhet – kunsten å maksimere mengden arbeid som ikke blir gjort – er essensielt
11	Selvstyrte team	De beste arkitekturer, krav og design vokser frem fra selvstyrte team
12	Kontinuerlig prosessforbedring	Med jevne mellomrom reflekterer teamet over hvordan det kan bli mer effektivt og så justerer det adferden sin deretter

***http://agilemanifesto.org og http://agilemanifesto.org/iso/no/principles.html**

Plan for forelesningen

- Læremidler
- Kap. 1: introduksjon
  - IT-systemer og systemutvikling
  - Eksempel på systemutvikling
- Kap. 2: Systemutviklingsprosessen – hvordan jobbe smart i IT-prosjekter
  - Prosessmodeller (modeller for systemutviklingsprosesser)
  - Fossefallsmodellen
- Kap. 3: Smidige metoder
  - Tidsboksbasert (Scrum)
  - Flytbasert (Kanban)
  - Eksempler på praksiser ved smidig utvikling

Prosessprinsipp:
Timeboxing versus “task-boxing” / “task-flyt”

Tidsbokser (Scrum)

- Velg noen prioriterte oppgaver og jobb med dem i faste tidsintervaller (tidsbokser*) med definerte oppstarts- og avslutningsaktiviteter

Flyt av oppgaver (Kanban)

- Definerer et sett med oppgaver eller “features” som skal lages, og lever så snart man er ferdig. Oppgaver skal ”flyte” uten avbrudd gjennom de nødvendige aktivitetene til de er ferdige (”oppgaveboksing”)

*Tidsboks: en fast tidsperiode som et gitt arbeid skal være ferdig innenfor

Scrum – analogi fra rugby

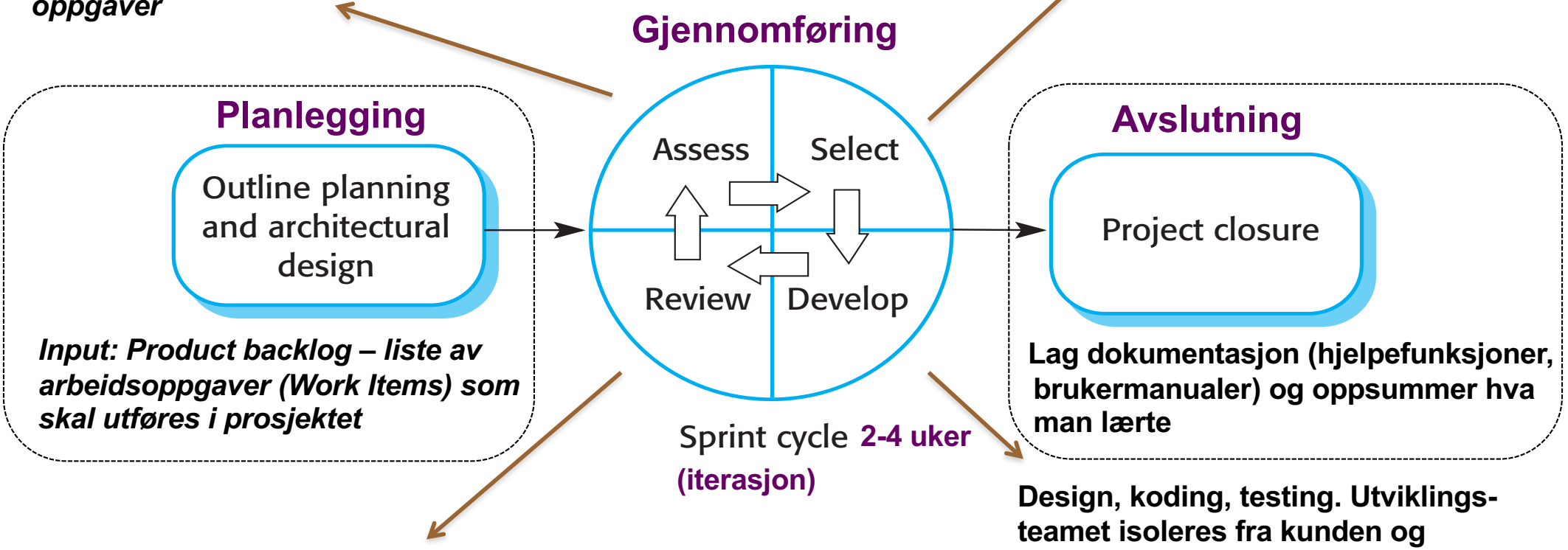
Scrum – tre faser

- **Planleggingsfasen**: overordnede mål for prosjektet etableres og programvarearkitekturen designes
- **Gjennomføringsfasen**: en serie med iterasjoner (kalt ”sprint”), der hver iterasjon leverer et inkrement av systemet
- **Avslutningsfasen**: nødvendig dokumentasjon som hjelp-funksjoner og brukermanualer fullføres, og man oppsummerer hva man har lært i prosjektet

Scrum

I sprint-planleggingsmøtet evalueres oppgavelisten (sprint backlog) som er en samling av brukerhistorier. Mål for sprinten settes inkl. prioriteter og risiko. Kunden kan sette nye krav el. gi nye oppgaver

Utviklingsteam og kunde velger egenskaper og funksjonalitet som skal utvikles i sprinten



Input: Product backlog – liste av arbeidsoppgaver (Work Items) som skal utføres i prosjektet

Sprint cycle **2-4 uker (iterasjon)**

Lag dokumentasjon (hjelpefunksjoner, brukermanualer) og oppsummer hva man lærte

Resultatene og teamarbeidet evalueres mot målene som ble satt i sprint-planleggingsmøtet ("retrospective"), og ferdigstilt funksjonalitet presenteres for kundene ("demo")

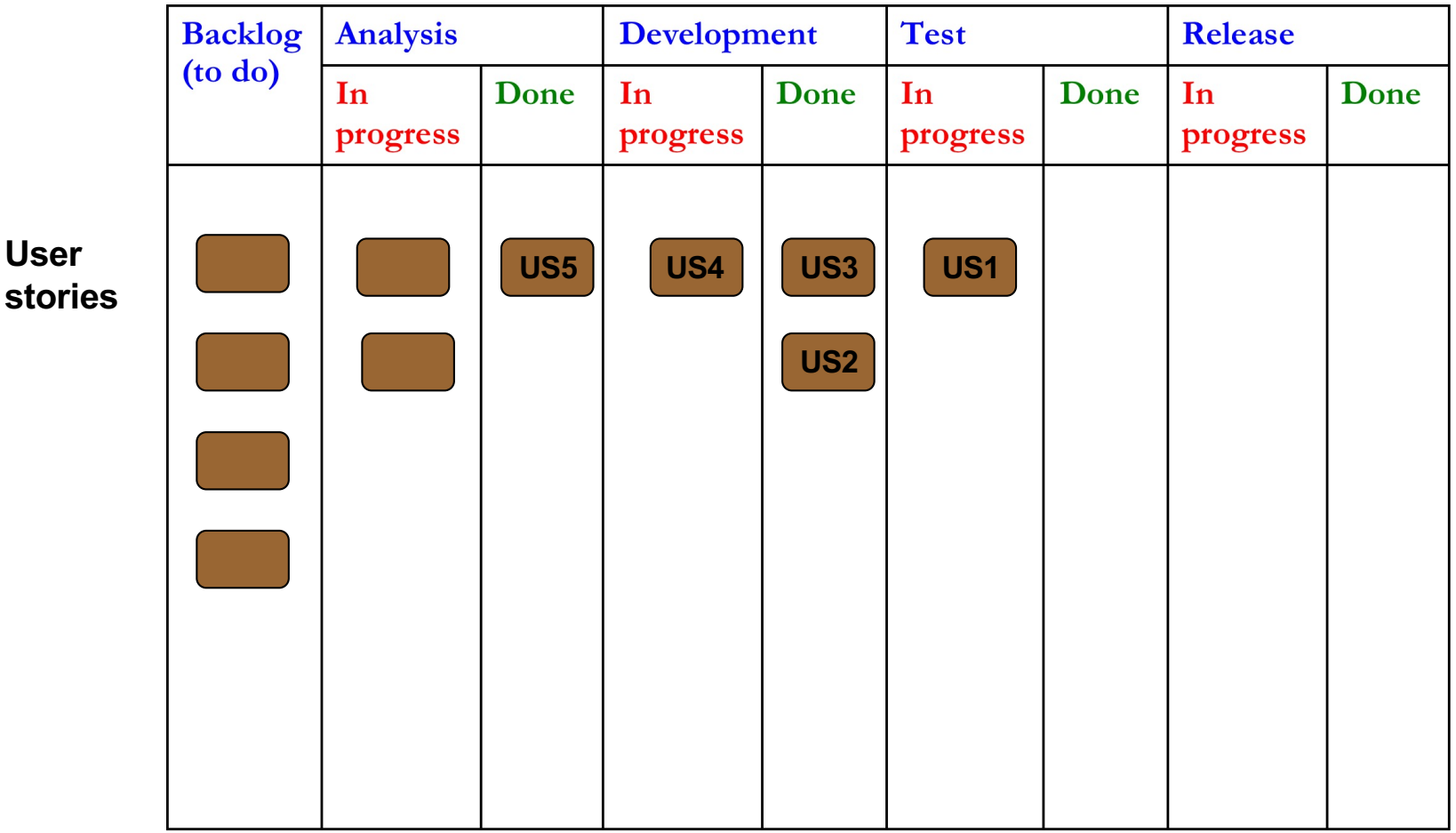
Design, koding, testing. Utviklings-teamet isoleres fra kunden og organisasjonen, dvs. all kom-munikasjonen skjer via Product Owner/Scrum Master for å unngå forstyrrelser

Brukerhistorie (user story)

- Én eller flere setninger som beskriver hva brukeren av et system ønsker å få ut av systemet på formen:
  - ”Som en <rolle> ønsker jeg <funksjon> for å oppnå <verdi>”
- Kort beskrivelse, passer på et kort eller gul lapp

Visualisering

Noter en oppgave eller arbeidspakke på en gul lapp og sett den på en tavle



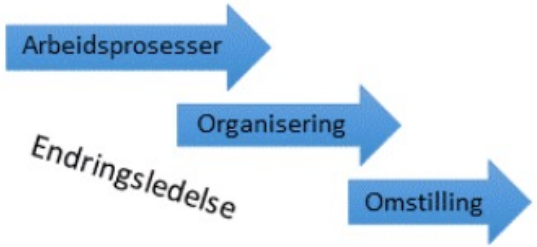
(Antatte) fordeler ved Scrum

- Systemet blir delt opp i en mengde forståelige og håndterbare deler
- Ustabile krav hindrer ikke progresjon i prosjekt-gjennomføringen
- Hele teamet observerer hva som skjer i prosjektet, og kommunikasjon innen teamet blir god
- Kundene får inkrementer levert fortløpende og kan gi tilbakemelding på hvordan deler av systemet fungerer
- Tillit mellom kunder og utviklere kan etableres tidlig
- Kryss-funksjonelle team (kompetansene på ulike områder finnes innen teamet) sikrer framdrift og reduserer risiko

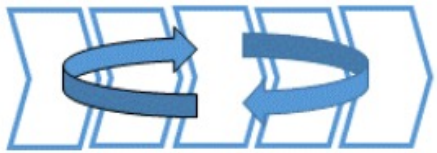
Hovedprosjekt iht Prosjektveiviseren

Styre produktleveranser

Leveranseprosjekt 1
Organisasjonsutvikling

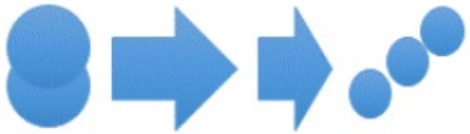


Leveranseprosjekt 2
Programvare-utvikling



F.eks. Scrum

Leveranseprosjekt 3
IT-anskaffelse



Anskaffelsesprosessen

Plan for forelesningen

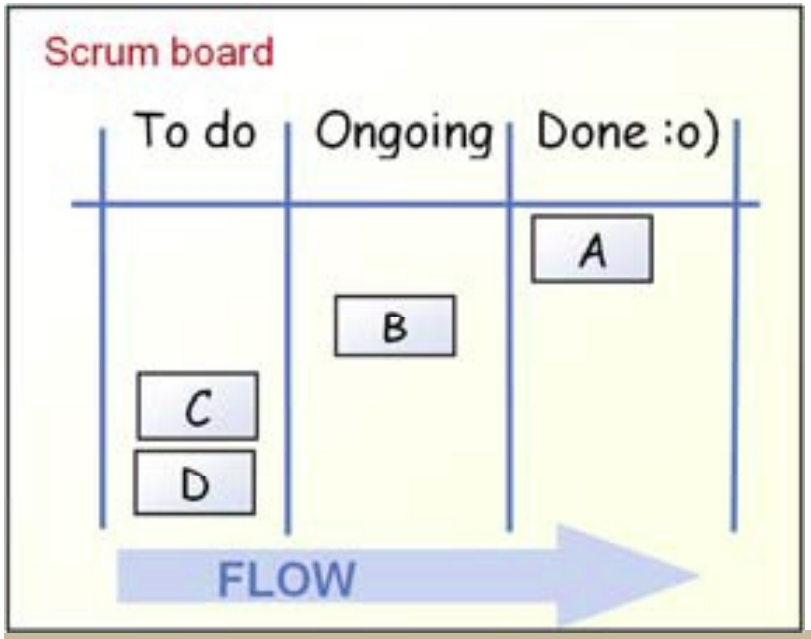
- Læremidler
- Kap. 1: introduksjon
  - IT-systemer og systemutvikling
  - Eksempel på systemutvikling
- Kap. 2: Systemutviklingsprosessen – hvordan jobbe smart i IT-prosjekter
  - Prosessmodeller (modeller for systemutviklingsprosesser)
  - Fossefallsmodellen
- Kap. 3: Smidige metoder
  - Tidsboksbasert (Scrum)
  - Flytbasert (Kanban)
  - Eksempler på praksiser ved smidig utvikling

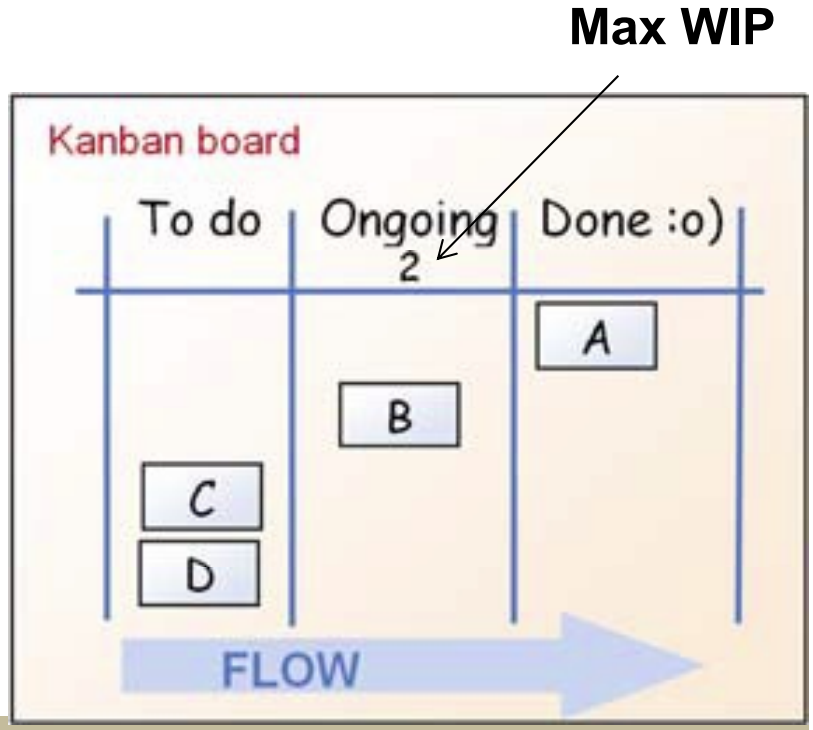
Kanban

- Fokus på å få oppgaver (arbeidspakker) raskt utført = antall brukerhistorier (features) implementert per tidsenhet
- Begrense antall arbeidspakker som det jobbes med i parallell (WIP = Work In Progress) for å hindre flaskehalser
- Antakelse: Jo høyere WIP, jo saktere flyter arbeidspakken gjennom arbeidsprosessene
- Når en pakke er ferdig, kan man etterspørre en ny som man begynner å jobbe med (pull)
- Mindre fokus på estimering av tid og kostnader

* De spesielt interesserte kan lese om WIP i et norsk firma (ikke pensum): Sjøberg, Dag IK. An empirical study of WIP in kanban teams. In: *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. Oulu, Finland, ACM, 2018.

Scrum board versus Kanban board





From: Kanban and Scrum - making the most of both by Henrik Kniberg and Mattias Skarin

Fordeler ved Kanban

- Flaskehalser i prosessen blir synlige
  - Fokus på å bli ferdig med oppgaver som hindrer gjennomstrømning fremfor å begynne på flere oppgaver som vil hope seg opp
- Kan drive smidig utvikling uten å bruke “tidsbokser”
  - Spesielt for drifts- og supportoppgaver og vedlikeholdsoppgaver vil veldefinerte “sprinter” ofte ikke gi mye mening
- Gunstig der det er svært vanskelig å estimere oppgavene

*** Sommerville diskuterer ikke flytbasert utvikling (Kanban). Temaet er likevel viktig**

Plan for forelesningen

- Læremidler
- Kap. 1: introduksjon
  - Hva er systemutvikling og hvorfor lære om det?
  - Eksempel på systemutvikling
- Kap. 2: Systemutviklingsprosessen – hvordan jobbe smart i IT-prosjekter
  - Prosessmodeller (modeller for systemutviklingsprosesser)
  - Fossefallsmodellen
- Kap. 3: Smidige metoder
  - Tidsboksbasert (Scrum)
  - Flytbasert (Kanban)
  - Eksempler på praksiser ved smidig utvikling

Table 2. Agile principles.

	Mean	Standard Deviation
Continuous integration	4.5	0.8
Short iterations (30 days or less)	4.5	0.8
"Done" criteria	4.5	0.8
Automated tests run with each build	4.4	0.9
Automated unit testing	4.4	0.9
Iteration reviews/demos	4.3	0.8
"Potentially shippable" features at the end of each iteration	4.3	0.9
"Whole" multidisciplinary team with one goal	4.3	0.8
Synchronous communication	4.4	0.9
Embracing changing requirements	4.3	0.8
Features in iteration are customer-visible/customer-valued	4.3	0.8
Prioritized product backlog	4.4	0.9
Retrospective	4.2	1.0
Collective ownership of code	4.2	0.9
Sustainable pace	4.2	0.8
Refactoring	4.2	1.0
"Complete" feature testing done during iteration	4.1	0.9
Negotiated scope	4.1	0.9
Stand up/Scrum meeting	4.1	1.1
Timeboxing	4.1	1.1
Test-driven development unit testing	4.0	1.0
Just-in-time requirements elaboration	4.0	1.0
Small teams (12 people or less)	4.0	1.1
Emergent design	4.0	1.0
Configuration management	4.0	1.2
Daily customer/product manager involvement	3.9	1.0
Release planning	3.9	1.1
Test-driven development acceptance testing	3.8	1.0
Team documentation focuses on decisions rather than planning	3.8	1.2
Informal design; no big design up front	3.7	1.0
Co-located team	3.6	1.1
Team velocity	3.6	1.1
Requirements written as informal stories	3.6	1.1
10-minute build	3.6	1.3
Task planning	3.5	1.2
Coding standard	3.5	1.2
Kanban	3.4	1.6
Acceptance tests written by product manager	3.4	1.2
Pair programming	3.3	1.2
Burndown charts	3.3	1.3
Code inspections	3.2	1.3
Design inspections	3.3	1.3
Planning Poker	3.1	1.4
Stabilization iterations	3.0	1.5

Hvilke praksiser er essensielle for at et team skal være smidig? (1=ikke viktig; 5=essensielt, et team er ikke smidig hvis det ikke utfører denne praksisen)

[L. Williams, *Communications of the ACM*, vol. 55, no. 4, april 2012]

Refaktorering (forbedring)

- Forbedre koden selv om ikke umiddelbart behov for dem
- Koden mer forståelig og enklere å endre, og mindre behov for dokumentasjon (mer vedlikeholdbar)
- Noen endringer krever at arkitekturen omstruktureres (kostbart)
- Eksempler på refaktorering
  - Reorganisering av klassehierarki for å fjerne duplisert kode
  - Forbedre navn på attributter og metoder
  - Erstatte kode med kall til metoder i et programbibliotek
- Dårlig kode vil ha ”teknisk gjeld”

Parprogrammering

To programmerere utvikler kode sammen:

- Fører
  - skriver på tastaturet
- Navigatør
  - observerer arbeidet til føreren og ser etter feil og svakheter
  - ser etter alternativer
  - noterer ting som må gjøres
  - slår opp referanser
- Kan brukes uavhengig av smidige metoder

Hva er kost-nytten ved parprogramming?

- Sommerville skriver i boka s. 70:
  - "However, studies with more experienced programmers (Arisholm et al., 2007*; Parish et al., 2004) did not replicate these results. They found that there was a significant loss of productivity compared with two programmers working alone."

Verdens største eksperiment i software engineering med 300 profesjonelle utviklere som deltakere

***E. Arisholm, H.E. Gallis, T. Dybå and D.I.K. Sjøberg. Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise, *IEEE Transactions on Software Engineering* 33(2):65-86, 2007**

Effekten av å bruke parprogrammering "kommer an på"

Programmerings-ekspertise	Oppgave-kompleksitet	Bruke PP?	Kommentar
Junior	Lett	Ja	Gitt at hovedmålet er god kvalitet
	Vanskelig	Ja	Gitt at hovedmålet er god kvalitet
Mellomnivå	Lett	Nei	
	Vanskelig	Ja	Gitt at hovedmålet er god kvalitet
Senior	Lett	Nei	
	Vanskelig	Nei*	

*** Med mindre oppgaven er svært for vanskelig, selv for en senior**

Oppsummering

- Software engineering er læren om utvikling og forvaltning av programvaresystemer av høy kvalitet innen gitte tids- og kostnadsrammer
- Finnes mange ulike kriterier for prosjekt- og systemkvalitet
- Ulike prosessegenskaper vil påvirke prosjekt- og systemkvaliteten
- Valg av prosess vil avhenge av hvilke kvalitetsaspekter man ønsker å vektlegge
- Det å jobbe smart, dvs. ha gode og effektive arbeids-prosesser vil alltid være et aktuelt tema

Takk for i dag!