

Effektive samarbeidspraksiser for kravhåndtering

Hans Gallis
Symphonical

Kjetil Moløkken-Østvold
Conceptos Consulting

JavaZone, 18. september 2008

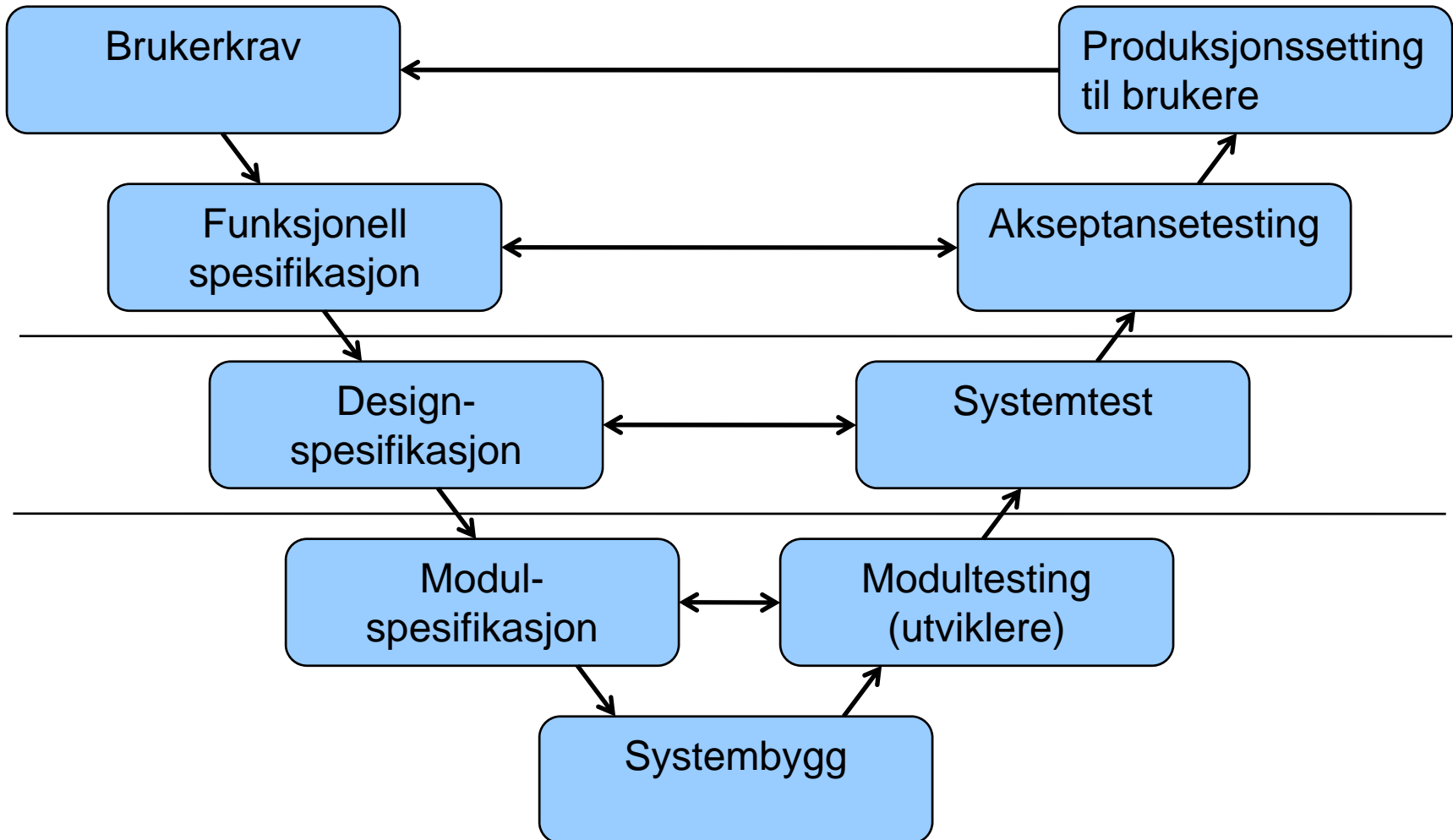
Viktige momenter ved denne sesjonen

- BOF = Diskusjonsbasert sesjon.
- Stor variasjon i bakgrunn og roller hos deltakere.
- Avbryt, spør og diskuter underveis.

Er kravhåndtering viktig?

- Kravprosesser (og brukertesting) krever:
 - Struktur og langsiktighet.
 - Godt samarbeid mellom kunde og leverandør.
- Mange sliter med å få kravprosesser og brukertesting smidig!
- Kravprosessen forut for en iterasjon er ofte undervurdert.
- Skal man gjennomføre en stor up-front kravprosess eller skal man gjøre den til del av hver enkelt iterasjon?
- Symphonical har fått et eget IFU-prosjekt gjennom Innovasjon Norge på dette temaet.

V-modellen



Oppgave: Forretningsmål

- Bakgrunn: Uavhengig om du er på leverandørsiden, kundesiden eller i en annen rolle.
- Spørsmål: Kan du beskrive hva som var forretningsmål(ene) i ditt siste prosjekt?

Dagens budskap



Effektive
Prosjekter

Bedre estimering,
kravhåndtering og
prosjektgjennomføring

Gode samarbeidspraksiser

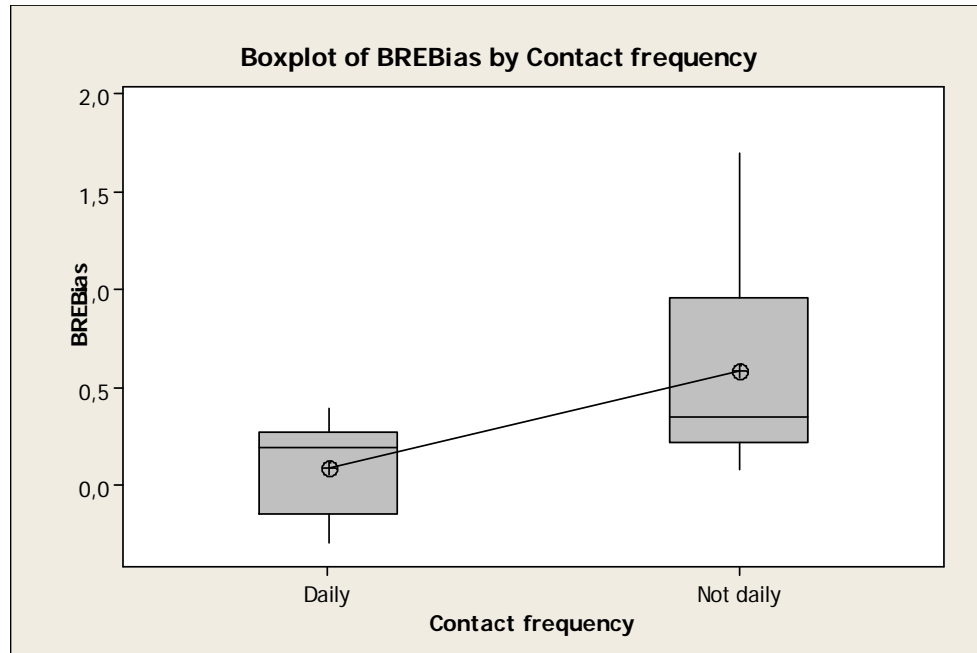
Kommunikasjon

- Studier har vist at opptil 70% av total tid i IT-prosjekter benyttes til kommunikasjon¹.
- Hyppig kommunikasjon kan benyttes til å prioritere krav, fokusere på feilretting, inkludere nye krav eller avklare løsningsmuligheter.
- Delvis motivert av Cockburn², ønsket vi å undersøke effekten av kommunikasjonsfrekvens mellom kunde og leverandør.

¹ Teasley, S.D., Covi, L.A., Krishnan, M.S. and Olson, J.S. (2002). Rapid Software Development through Team Collocation. IEEE Transactions on Software Engineering, 28 (7), 671-683.

² Cockburn, "The End of Software Engineering and the Start of Economic-Cooperative Gaming," *ComSIS*, 2004.

Kommunikasjonsfrekvens

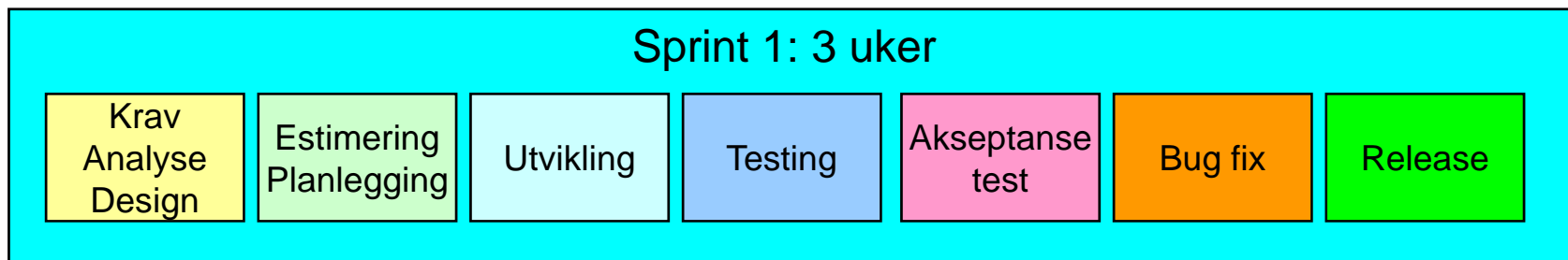


Overskridelser

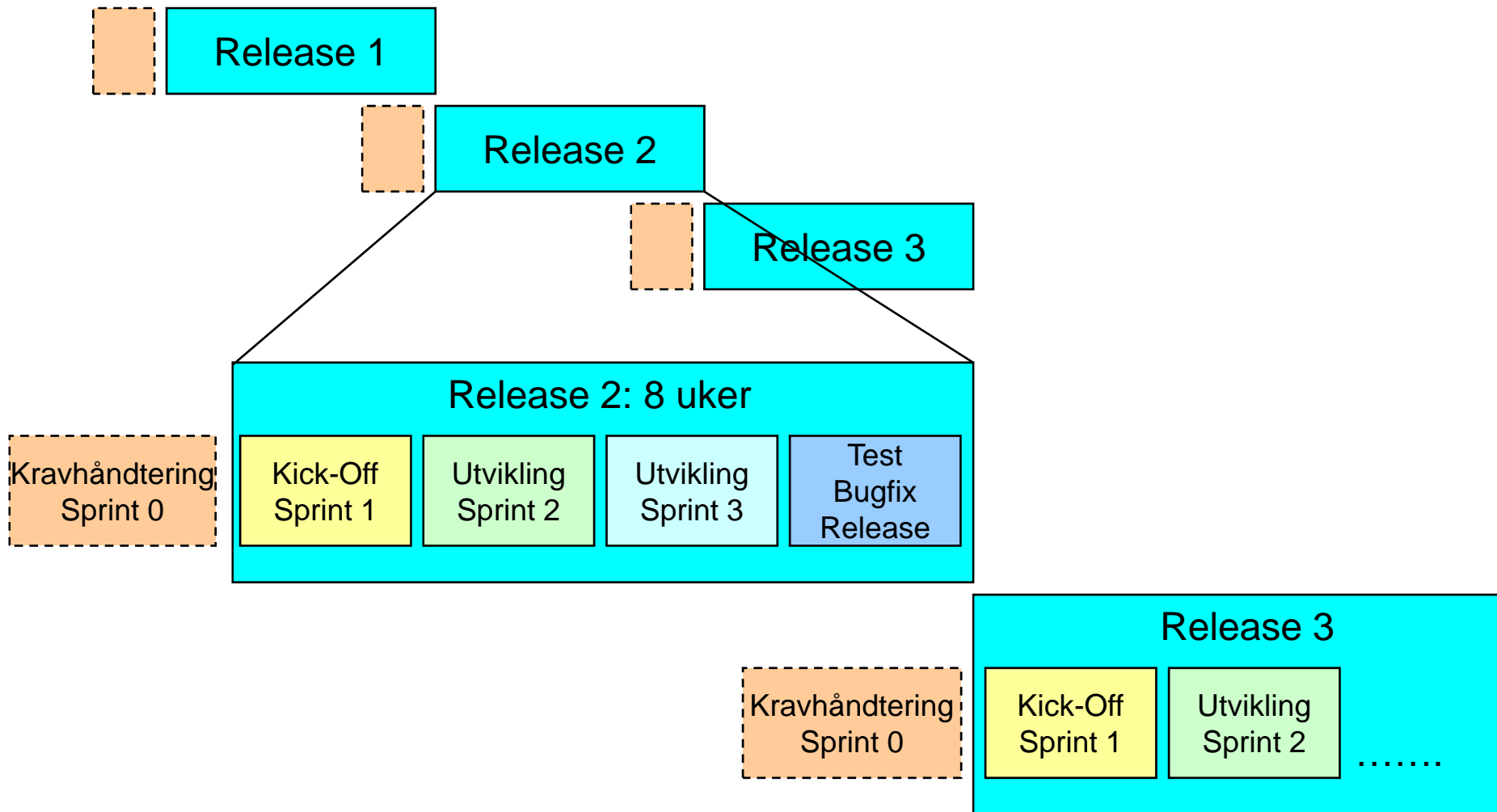
Daglig: 9%

Ikke daglig: 58%

Eksempel 1: Sprint



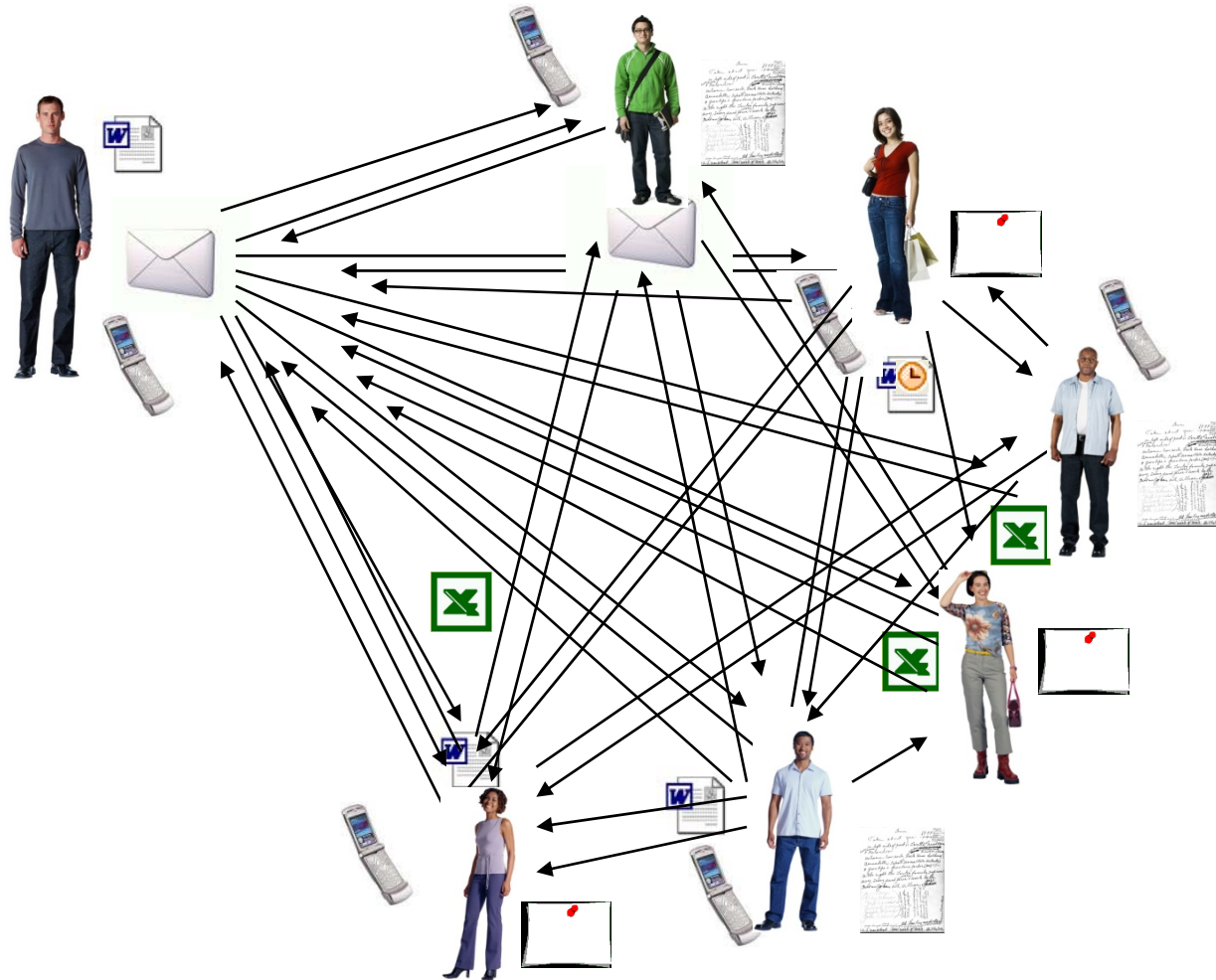
Eksempel 2: Release



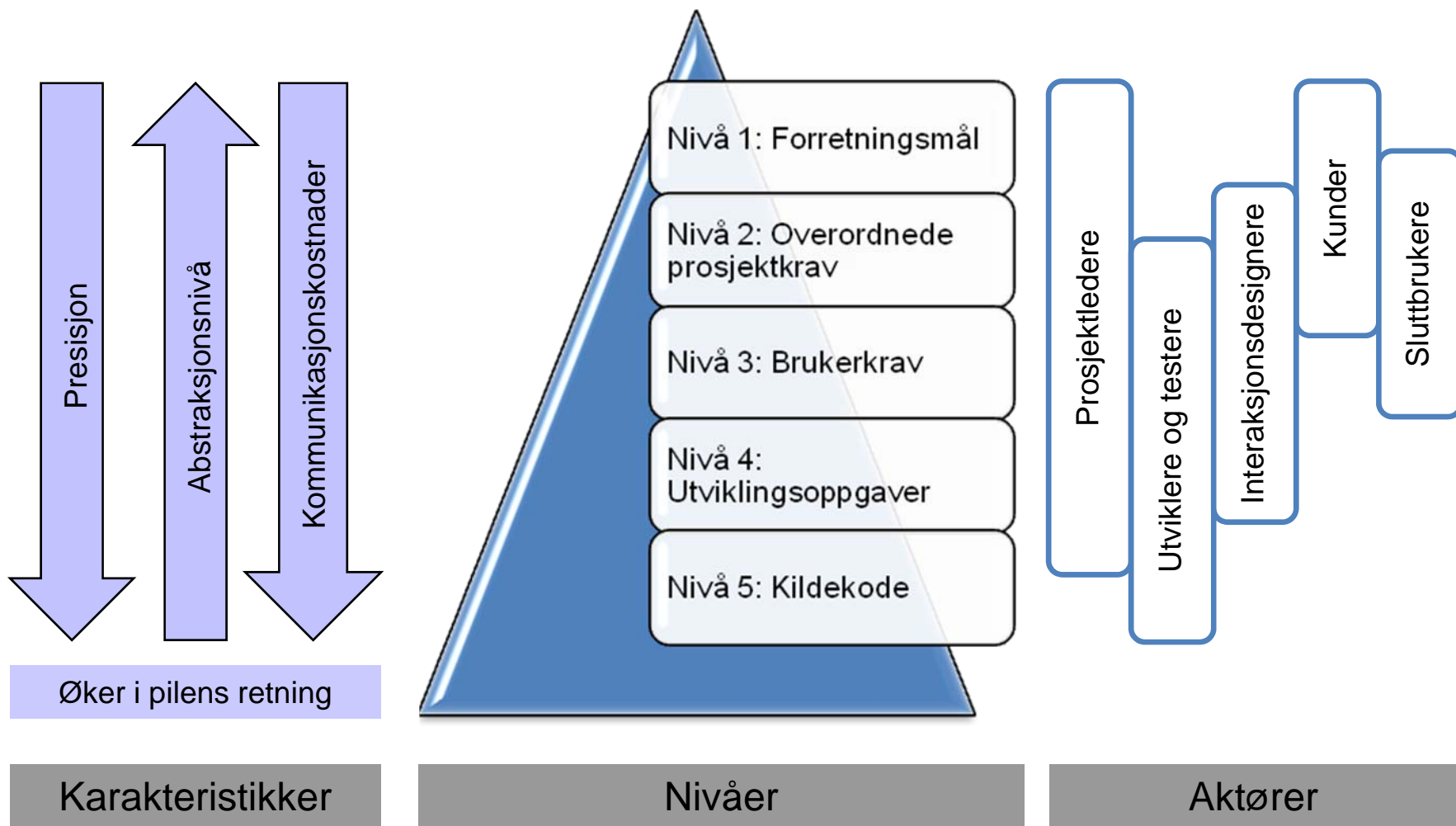


Effektive praksiser for kravhåndtering og forretningsprioritering

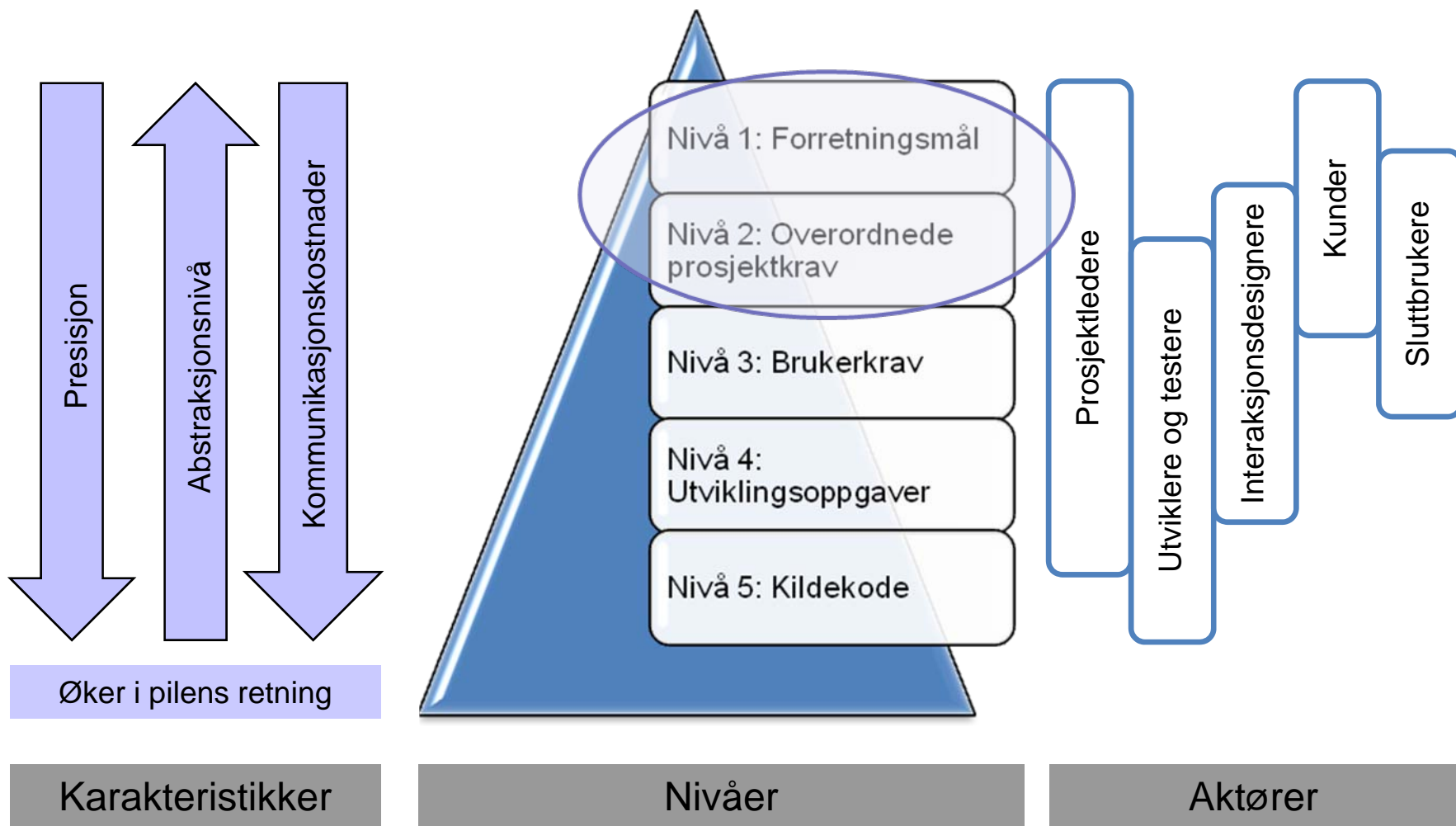
Frustrerende samarbeid!



Kravpyramiden ©



Kravpyramiden © - Nivå 1 og 2



Forretningsmål

- Avhengig av (eksempelvis):
 - Type prosjekt.
 - Kunde.
 - Prioriteter.
- Eksempler:
 - "Portere eksisterende løsning til ny teknologi".
 - "Effektivisere saksbehandlingstid med 10%".
 - "De 10 viktigste funksjonene skal fungere minimum 20% raskere".
- Hvordan bør de settes opp?

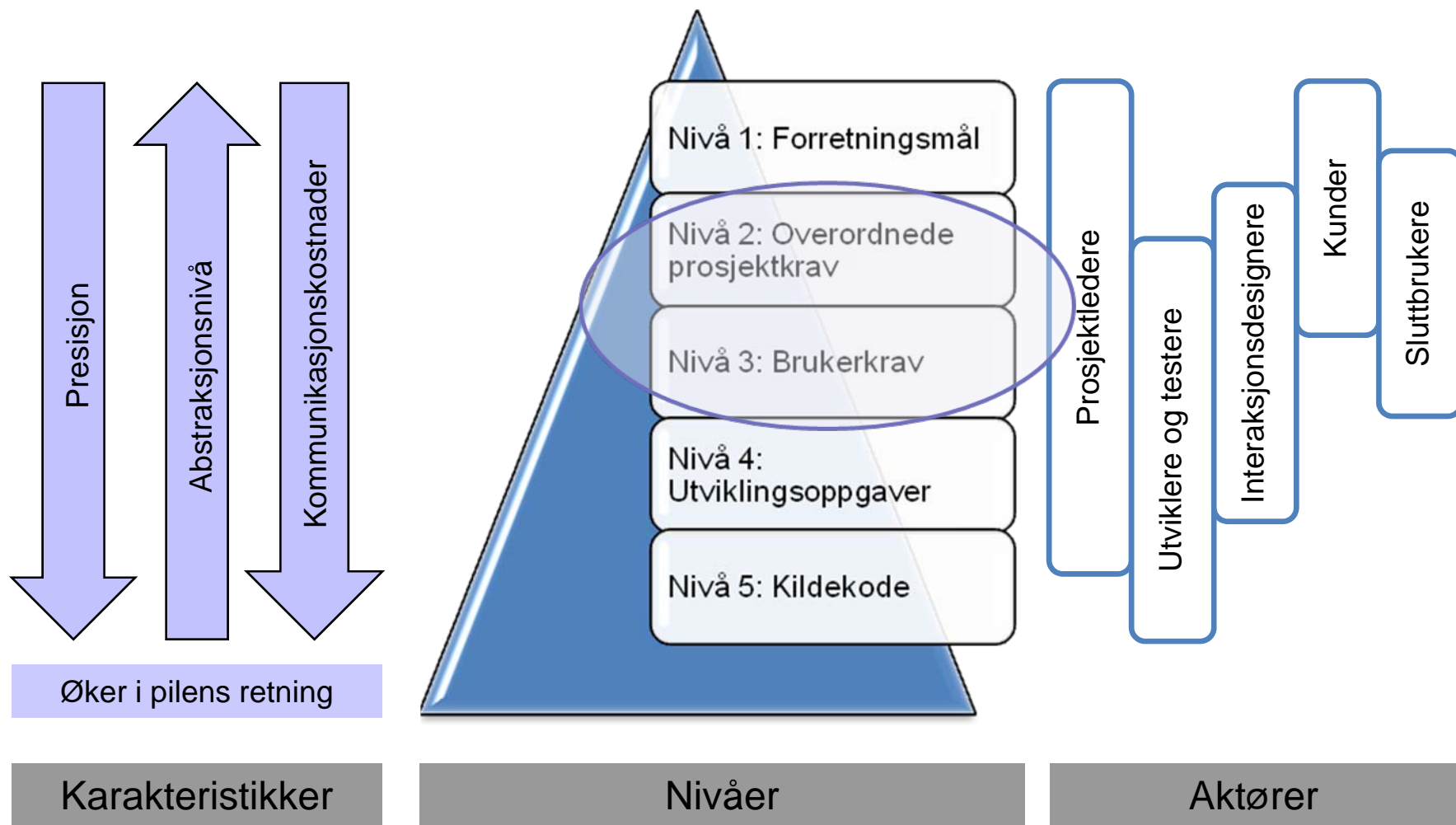
Behov

- Hva er behovene man søker å løse?
- Tenk behov (arbeidsprosess) fremfor "features"
 - Features SKAL kunne linkes til et definert behov.
- Skille mellom typer av behov:
 - Kunde/bestillerbehov.
 - Sluttbrukerbehov.
 - Leverandørforslag.
 - Trender, arkitektur, vedlikeholdbarhet etc. som spiller inn.
 - Foreslå behov for kunden.

Innovasjon

- I nivå 1 og 2 bør det være rom for innovative konsepter.
- Dette må leverandøren utfordre kunden på.
- Utfordrende spørsmål:
 - Hvordan gjøres dette i dag (uten denne funksjonaliteten)?
 - Hvem trenger denne funksjonaliteten?
 - Hvor ofte benyttes denne funksjonaliteten?
 - Finnes det eksisterende løsninger som løser dette på en god måte?
 - Finnes det andre, og kanskje mer innovative, løsninger?

Kravpyramiden © - Nivå 2 og 3



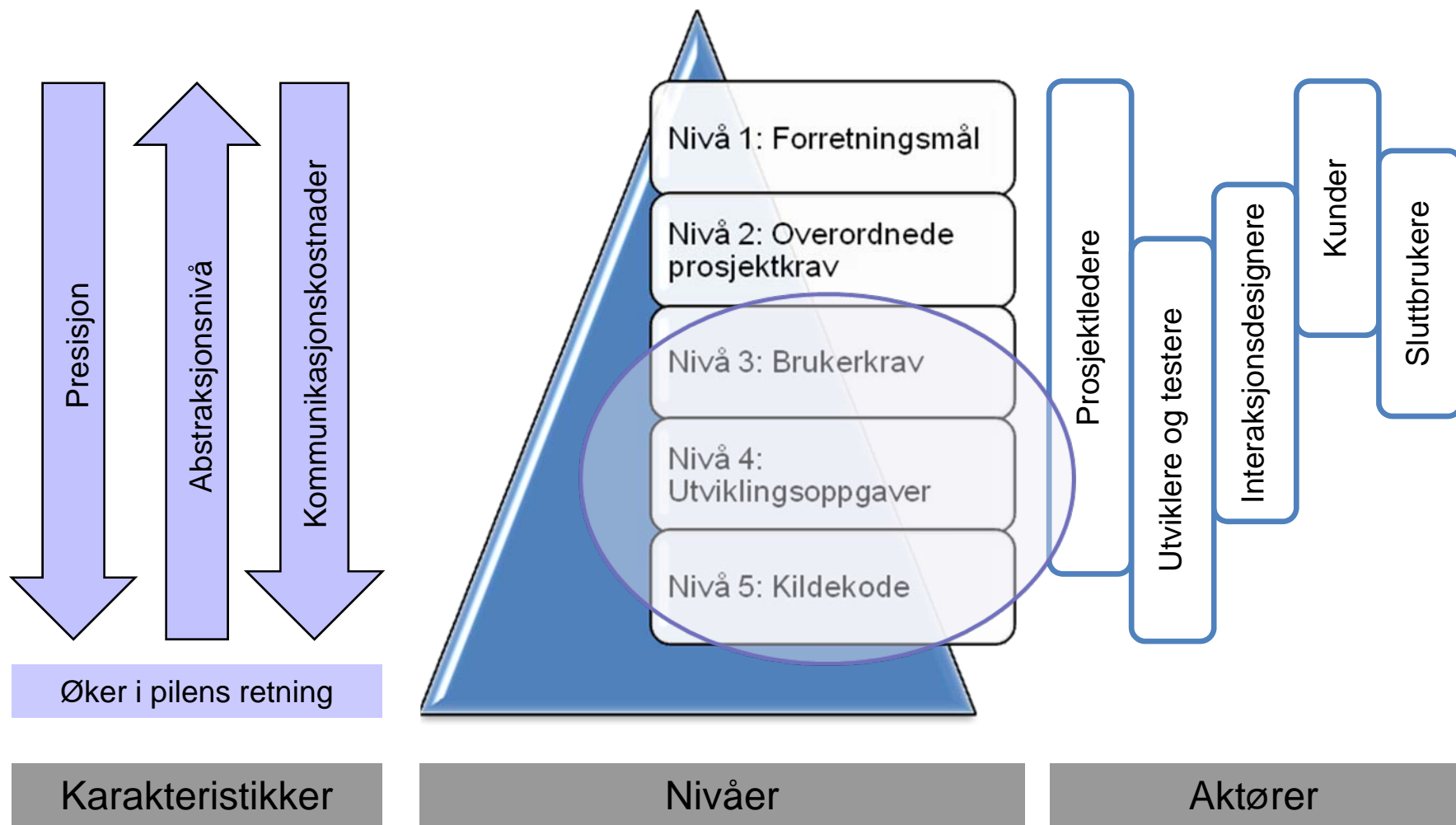
Kravhåndteringsprosess

- Gjelder for ideer (krav) som har blitt prioritert inn i et prosjekt eller delprosjekt.
 - Denne delen involverer også iterasjonsplanlegging ("Product Backlog").
 - Skille tydeligere mellom ønske og krav (ønske gjelder nivå 1-2 i modellen ovenfor).
- Utdype spesifiseringen fra idéprosessen med nye og mer detaljerte parametere.
 - Inkluderer å vurdere muligheten for å legge flere krav sammen i en "modul" etc.
- Jo mer kundesiden bidrar med, desto bedre blir funksjonaliteten og desto færre feil blir det (pga misforståelser etc).

Kravhåndteringsprosess forts.

- En brukerhistorie kan suppleres med mer detaljerte parametere:
 - Andre aktører (tilgangsstyring).
 - Input (trigger).
 - "Happy day" scenario.
 - Use frequency (relevant for prioritering og testing).
 - GUI (dersom ikke tilgang til skisser, beskriv kort hvordan det skal løses GUI-messig eller lag noe enkelt i powerpoint etc).
 - Test case(s) og akseptansekrav.
 - Output (hvordan avsluttes denne funksjonen? Hva skjer i systemet når funksjonen har kjørt?).
 - Spesielle avvik fra "Happy day" scenario.

Kravpyramiden © - Nivå 3 til 5



Oppgavehåndteringsprosess

- Løsningsansvarlig og utviklere deler opp "user stories" (krav) i håndterbare oppgaver.
- IKKE benytt høy/middels/lav prioritet. Benytt heller prinsippet om "rekkefølge" (uten avhengigheter).
 - For eksempel kan dere lage kategorier a'la first, second, third, fourth, fifth.

Oppgavehåndteringsprosess forts.

- Identifiser mulige løsninger.
 - Alle krav har minst to løsninger?
- Utvikleren identifiserer uklarheter/
vanskeligheter.
 - Dette gir innspill til estimeringen (som må gjøres av ansvarlig utvikler og minst en annen utvikler).
 - Hvis vanskelig: vurder om denne skal løses i par, dvs. analysere, skrive tester og teste i par (kode individuelt).

Diskusjon

- Hvilken størrelse er "passe" for oppgaver?
 - En uke, tre dager, en dag, fire timer?



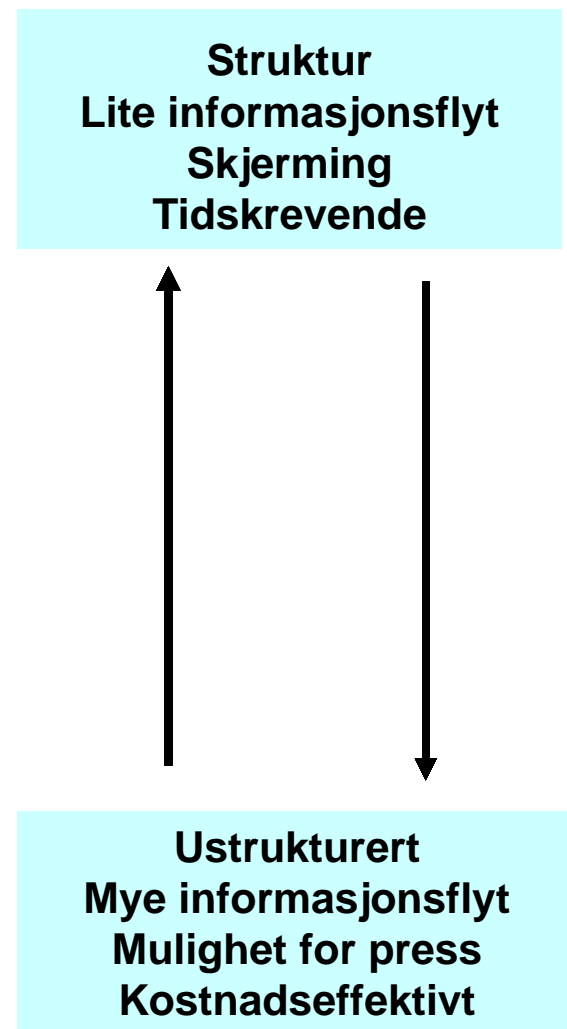
Effektive praksiser for å samarbeide om estimering og planlegging

Estimering bør være en integrert del av kravprosessen

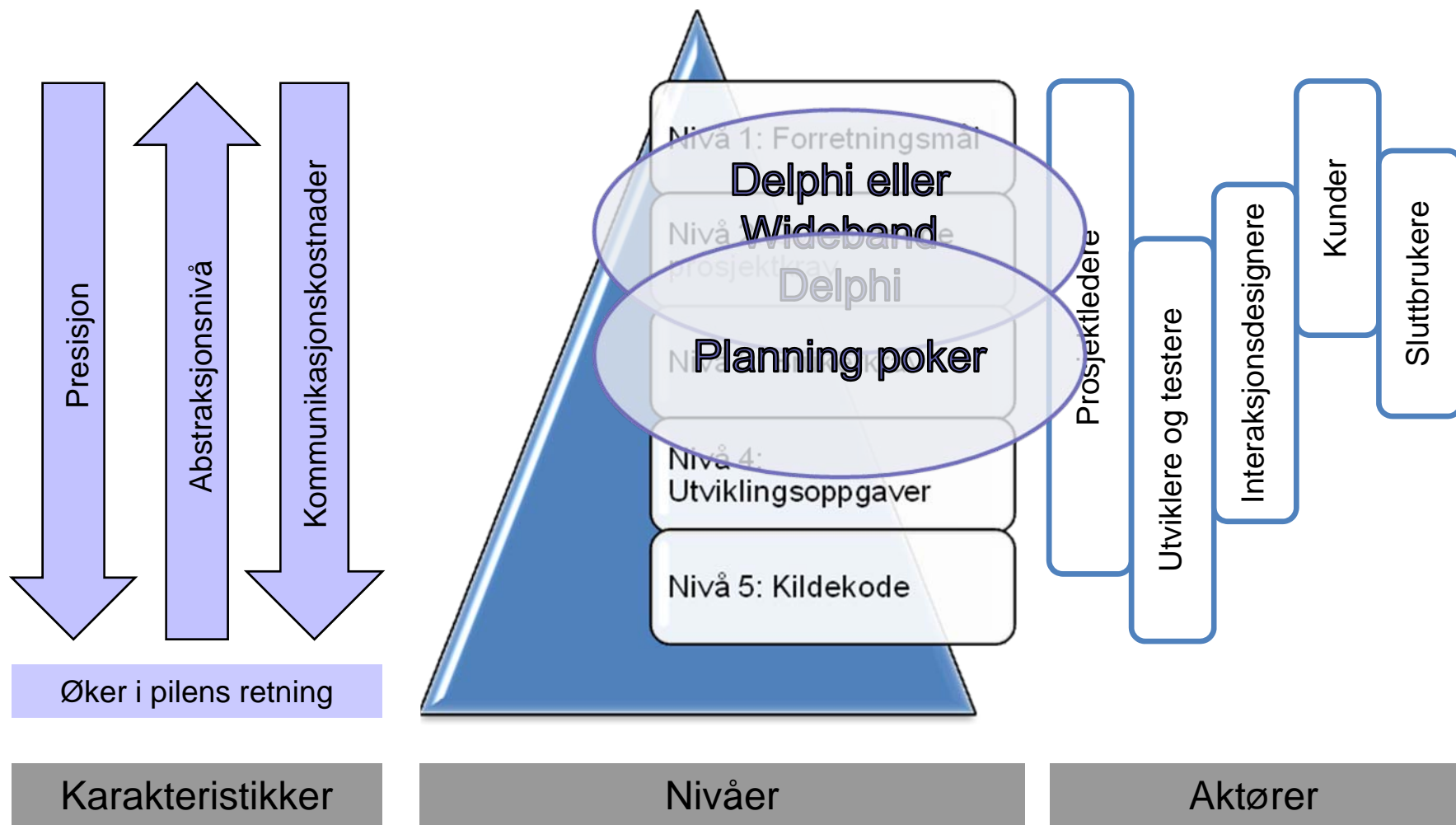
- Estimering og prioritering av krav gir nye innfallsvinkler og konkretiserer utfordringer.
- Flere roller bør involveres i estimeringen, både fra leverandør og kundeside.
- Estimeringen bør skje på forskjellige nivåer (ref. pyramiden) og trianguleres.

Metoder for kombinasjon av estimater

- Decision markets
- Delphi
- Wideband-Delphi
- Planning Poker
- Semi-strukturerte grupper
- Statistiske grupper



Kravpyramiden © - estimering



Eksempel fra in-house utvikling

1. Kundesiden prioriterer viktighet og rekkefølge på krav.
2. Utviklingsteamet distribuerer tilgjengelige timer for utvikling på kravene.
 - I kundens prioriterte rekkefølge.
 - Mer fokus på det man vet man rekker å gjøre! → Ikke bindende estimer → Tidlig avgrensning.
3. Felles Wideband Delphi på overordede krav man tror man rekker.
 - Skaffe mer sikkerhet i estimatene.
 - Diskutere krav.
 - Prioritere på nytt!
4. Utviklingsteam gjør oppdeling med Planning poker på brukerhistorier.
 - Splittes senere i oppgaver.

Noen fordeler ved kombinasjon av estimer

- Kombinerer kunnskap fra flere kilder.
- Ekstreme utslag kan bli moderert.
- Synkronisert oppfatning av hva estimatene innebærer angående aktiviteter og antakelser.
- Mer eierskap til estimatene.
- Moderator kan sørge for at irrelevant informasjon ikke ødelegger for realismen.
- Moderator kan være “djevelens advokat”.

Mulige ulemper ved gruppeestimering

- Kan være utfordring med passive deltakere.
- Avhengig av valgt metode: politisk press (groupthink).
- Krever gode eksperter og moderatorer.
- Tids- og kostnadskrevende (?).

Planlegging av iterasjoner

- Timeboxing: Lærer hva tidsintervallet betyr for deg.
 - Eksempel: faste to ukers iterasjoner.
 - Eller: variere lengde for å "bli ferdig"?
- Hvordan avsluttes en iterasjon?
 - Legge det man ikke fikk ferdigstilt tilbake i Product Backlog?
 - Overføre det man ikke rakk til neste iterasjon?
 - Utfordring: Det er ikke tatt med i selve planleggingen!
 - Man kjøper seg uansett en "straff" når krav ikke blir ferdigstilt. Spørsmålet er hvordan denne "straffen" skal behandles!



Avslutning

Studie om prosjektprosesser

- Et studie fant signifikante forskjeller i gjennomsnittlige overskridelser av estimert ressursbruk (arbeidstid) ¹.
 - Prosjekter med sekvensielle prosesser: 55%.
 - Prosjekter med fleksible (inkl. smidige) prosesser: 24%.
- Fleksible prosjekter fikk mer positiv omtale på:
 - Gode krav.
 - Bra samarbeid/kommunikasjon med kunde.
- Dette er ofte (de mest) kritiske utfordringer i prosjekter!

¹ Moløkken-Østvold and Jørgensen, "A Comparison of Software Project Overruns", IEEE Transactions on Software Engineering, 2004.

Mulige årsaker?

- Økt interaksjon med kunden øker muligheten for å oppdage feil eller mangler på et tidlig tidspunkt.
 - Kunden involveres gjennom hele prosjektet og ikke bare i en startfase.
- Utvikling av kjernefunksjonalitet først, deretter videre på prioriteringslisten.
- Prosjektene utvikles i mer håndterbare (mindre) enheter.

Debatt om prosesser

- Er smidige prosesser svaret?
- Hva må man ha av kunnskap om prosesselementer?
- Hvordan kan man velge de rette praksisene i sin prosess?

Effektiv kravhåndtering!

- Tenk korte iterasjoner og hyppige leveranser.
 - Enklere å håndtere.
 - Enklere å lære.
- Bli enige om en standard for beskrivelse av krav på ulike nivåer (ut i fra hvem som faktisk skal involveres).
- Tydelige roller.
 - Hvem har overblikket?
 - Hvem prioriterer og tar avgjørelser?
 - Hvem kommuniserer med utviklingssiden?
 - Hvem skriver brukerhistorier?

Spørsmål?

- hans@symphonical.com
- kjetil@conceptos.no
- www.symphonical.com
- www.conceptos.no