

Teamarbeid i software engineering

IN2000 - vår 2026

Yngve Lindsjørn

I dag skal vi snakke om

- Smidig utvikling
- Smidige team
- Stor-skala smidig
- Utfordringer i teamarbeid
- Tips til teamarbeidet

Læringsmål for forelesningen

Etter å ha fullført IN2000:

- har du kunnskap om de viktigste systemutviklingsmetodene, inkludert deres styrker og svakheter
- har du kunnskap om sentrale prosesser og aktører i prosjekt- og teamarbeid som anvender smidige prinsipper
- har du kunnskap om følgende aktiviteter innenfor systemutvikling: kravinnsamling og -analyse, design, programmering, testing, samt vedlikehold og videreutvikling
- kan du anvende profesjonelle systemutviklingsmetoder, -teknikker og -verktøy
- har du kompetanse for å arbeide i team og evne til å reflektere over eget og teamets arbeid i systemutviklingsprosjekter
- kjenner du til metoder og prinsipper for innebygget sikkerhet og universell utforming

SMIDIG UTVIKLING

Snakk med de
ved siden av deg:



Hva er smidig utvikling?



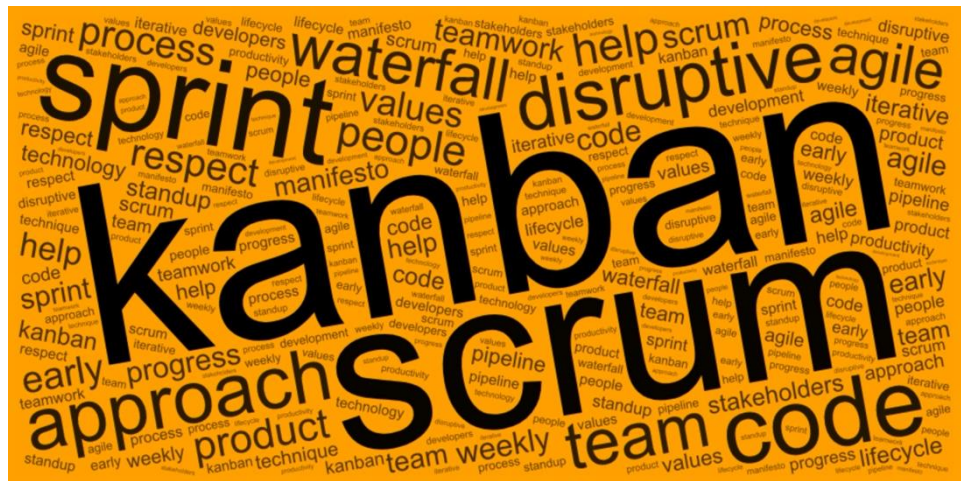
Har dere jobbet med
smidige metoder tidligere?
Når?



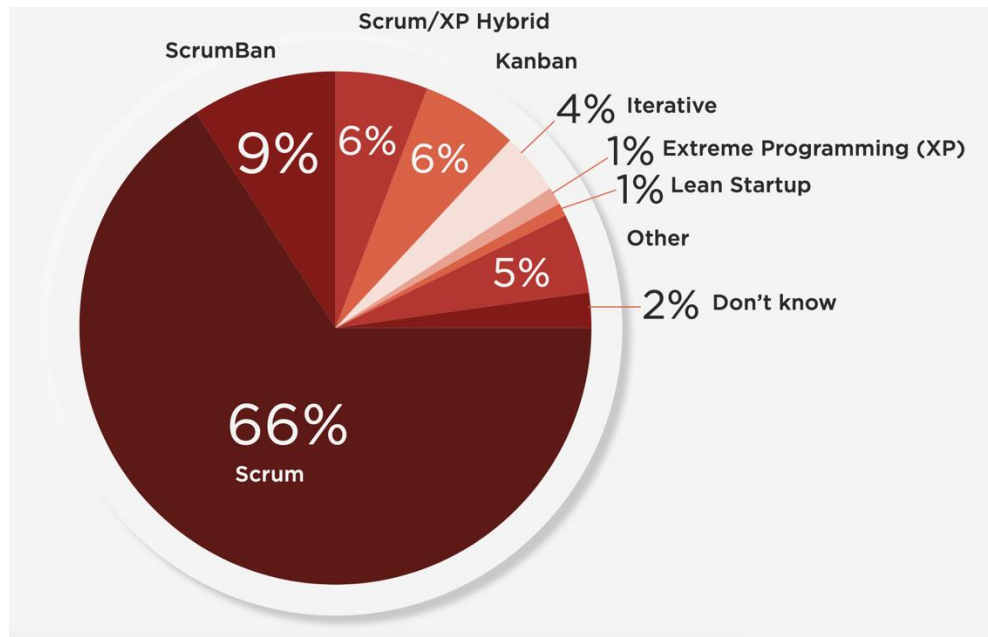
Hvordan var det?

Det er mange tilnærminger til smidig, alle med noen underliggende ideer, verdier og prinsipper.

- Smidige metoder
- Smidige prosesser
- Smidig praksis
- Smidige prinsipper
- Smidige team
- Smidige organisasjoner
- Smidige transformasjoner
- Stor-skala smidig
- **SMIDIG TANKESETT**



Smidige metoder



Version One, The 15th state of agile report, 2021

2024:



63%

According to this year's survey, a majority of team-level Agile user follow the Scrum methodology. 1 surprising: since our first survey i Scrum has been the most popula methodology.

Scrumprosessen



Produkteier



Scrum Master

Sprint (2-4 uker)

Daglige stand-up møter



Autonomt utviklingsteam



Visjon



Produkt-oppgaveliste



Sprint-oppgaveliste

Planleggings-møte

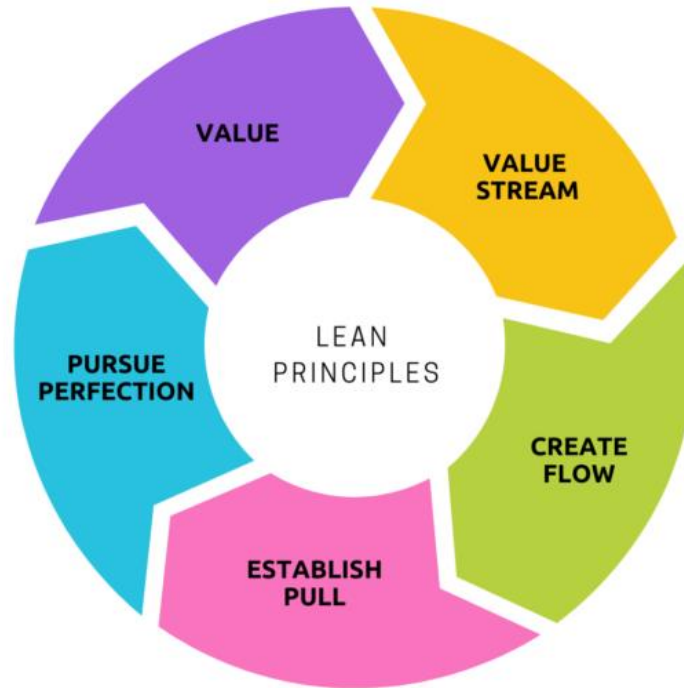
Leveranse



Demo og retrospektiv-møte

LITT OM LEAN

Mål: Maksimere kundeverdi og minimere ressursbruk



Just-In-Time-prinsippet (JIT) og pull-prinsippet

- Handler om å lage produkter eller komponenter først når det er behov for dem.
- Ikke ta endelige avgjørelser for tidlig.
- Fokuser kun på nødvendige detaljer.
- Basert på «reell etterspørsel»



Just-In-Time-prinsippet (JIT)

- Ikke lag noe før det er etterspurt.
- Ikke ta endelige avgjørelser for tidlig.
 - Mange detaljer kan utsettes etter at fundamentale aspekter er implementert.
 - Fokuser kun på detaljer som er nødvendig for å implementere oppgavene som er i prosess.
- Unngå waste:
 - Kan vise seg at enkelte oppgaver ikke er nødvendige å implementere i det hele tatt.
 - Ved å utsette oppgaver blir de ofte bedre implementert enn om de blir tatt tidlig.
 - Reduser flaskehalser – ting som stopper farten.

Flyt-prinsippet

- Sikre at arbeidsprosessene flyter jevnt uten avbrudd eller flaskehalser.
- Dette innebærer ofte å omstrukturere prosessene for å skape en kontinuerlig flyt av oppgaver gjennom utviklingsprosessen.



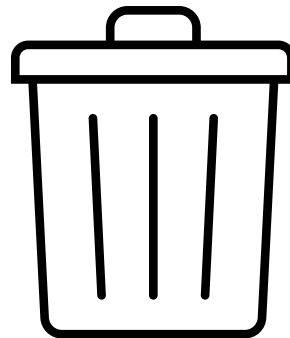
Visualisering



Waste

Noen kilder til “waste”

- Unødvendig funksjonalitet
- Komplexitet
- For stort skille mellom de som tar avgjørelser og de som gjør jobben
- Teknisk gjeld



Waste

Alt som krever ressurser

- tid
- arbeidsinnsats
- utstyr og lager
- penger

...Og som ikke gir verdi for kunden

Noen kilder til “waste”

- Unødvendig funksjonalitet
- Kompleksitet
- For stort skille mellom de som tar avgjørelser og de som gjør jobben
- Teknisk gjeld

Grunner til waste: Teknisk gjeld!

- En metafor som beskriver “snarveier” for å oppnå kortsiktige fordeler
 - Suboptimal kode som implementeres raskt
- Men som i det lange løp vil føre til økte kostnader på vedlikehold og videre utvikling



Mer om teknisk gjeld

- = "Alt som gjør det vanskelig å endre/modifisere kode"
- Siden kode alltid må vedlikeholdes/endres, må vi «betale tilbake» gjelden.
- Skriv "clean code", konsis, enkel med få avhengigheter
- Ny funksjonalitet leder til større kodebase og økt kompelsitet; refaktorering reduserer gjelden.

Refaktoring reduserer teknisk gjeld

Å omstrukturere koden for å forbedre interne kvalitet uten å endre kodens funksjonalitet

Hvordan det bør gjøres:

- Refaktoring handler kun om å forbedre arkitekturs kvaliteten
- Refaktoring bør "gjøre designet enklere"
- Ingen duplisering av kode
- Så få klasser og metoder som mulig
- Bør gjøres i små, inkrementelle steg
- Teknisk gjeld akkumuleres over tid
- Ofte utsetter man å håndtere helt til en «big refactoring» må til for å løse opp i problemer
- Kostnaden for å håndtere teknisk gjeld, inkludert refaktoring, i store organisasjoner er estimert til utgjøre i snitt 25 % av hele utviklingstiden (Martini et al., 2018)
- Refaktoring blir i økende grad automatisert (Baquais & Alshayeb, 2020)

SMIDIGE TEAM

Hvorfor er **team** så viktig i smidig?

De fleste programmer og produkter utvikles av **grupper av mennesker** som trenger å **interagere for å levere et produkt**

Smidig prinsipp #11:

*“De beste arkitekturer, krav og design vokser frem fra **selvstyrte team**”*

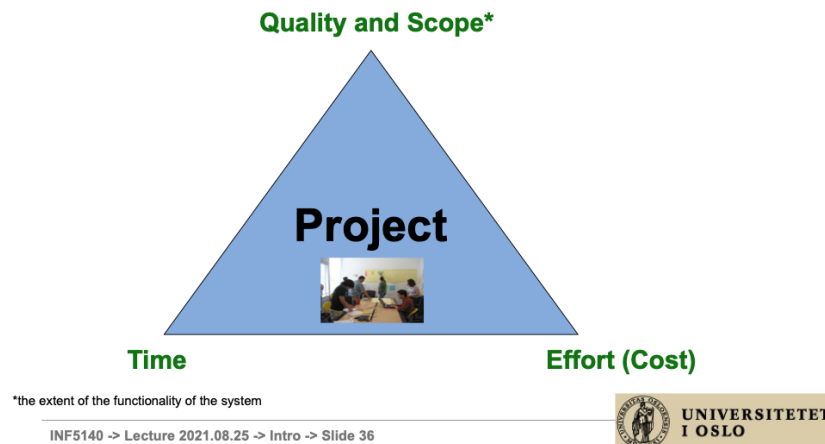


Kvaliteten på teamarbeidet påvirker:

1. Kvaliteten på **produktet** som utvikles (funksjonalitet, robusthet, reliabilitet og ytelse)
2. Kvaliteten på utviklings**prosjektet** (tid og kostnad)
3. Team-medlemmers **tilfredshet og læringsutbytte**

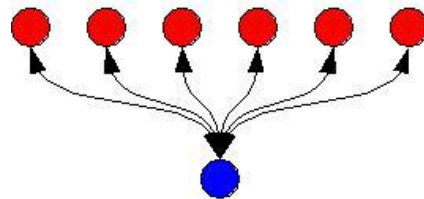
Smidige praksiser kan påvirke disse kvalitetsaspektene!

The Magic Triangle



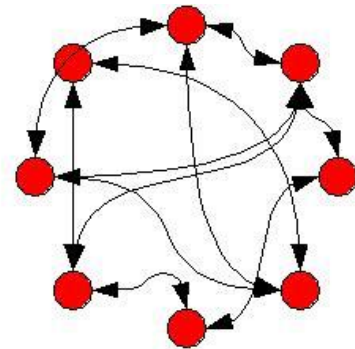
Gruppe eller team?

- Gruppe = en samling av personer som jobber individuelt, med en leder som styrer gruppen og koordinerer individuelle bidrag. Ofte lav felles forpliktelse.
- Team = en samling av individer som **deler et felles oppdrag** med ulike målsetninger. Team-medlemmene er forpliktet til hverandre og til teamets målsetninger. Teamet styrer seg selv.



GROUP

people working towards a goal whose work is coordinated by someone else (e.g. a manager) for them



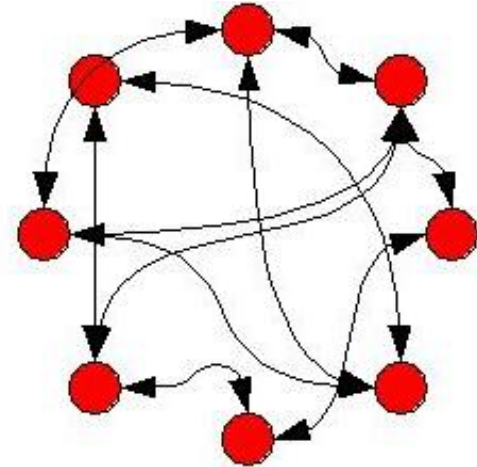
TEAM

people working towards a common goal who coordinate their work amongst themselves

Å jobbe sammen i (smidige) team

- Arbeidet er koblet gjennom avhengigheter mellom team-medlemmer
- Medlemmer holder hverandre gjensidig ansvarlige
- Selvstyrte team*
- Teamets mål er viktigere enn individuelle mål

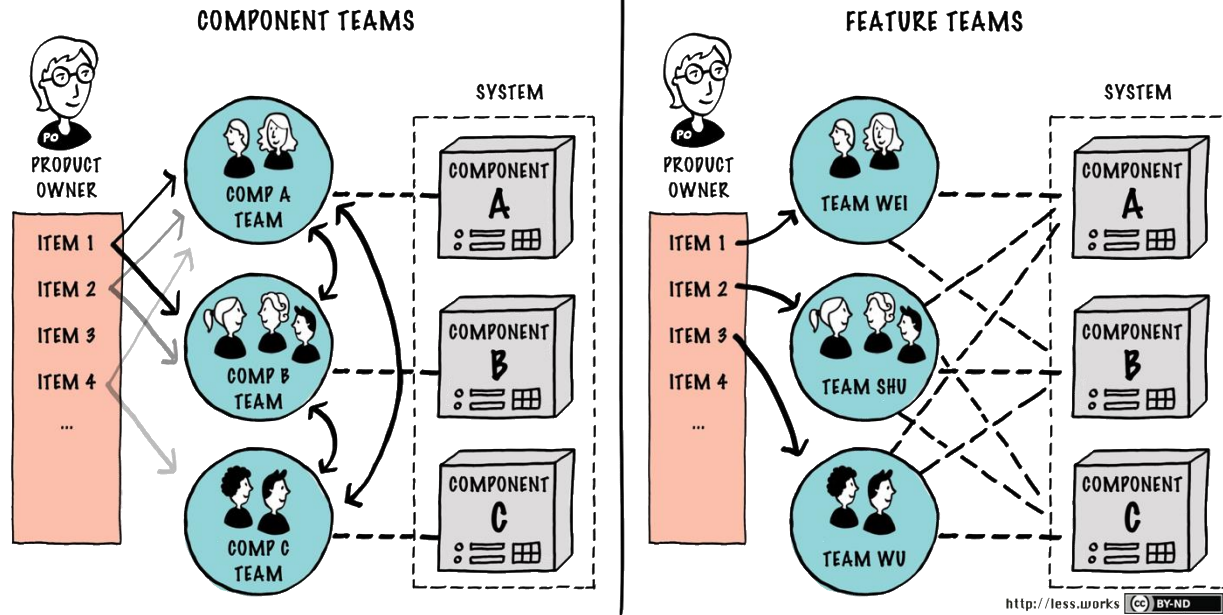
*Teamet bestemmer ikke nødvendigvis *hva* det skal lage eller jobbe med, men de bestemmer *hvordan* de skal jobbe.



TEAM

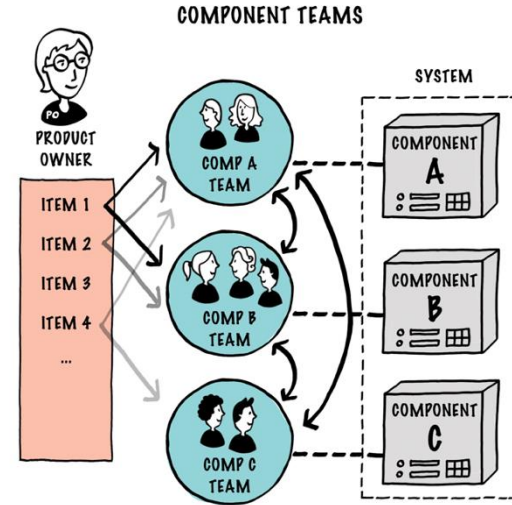
people working towards a
common goal who coordinate
their work amongst themselves

To typer utviklingsteam



Komponent-team (spesialiserte team)

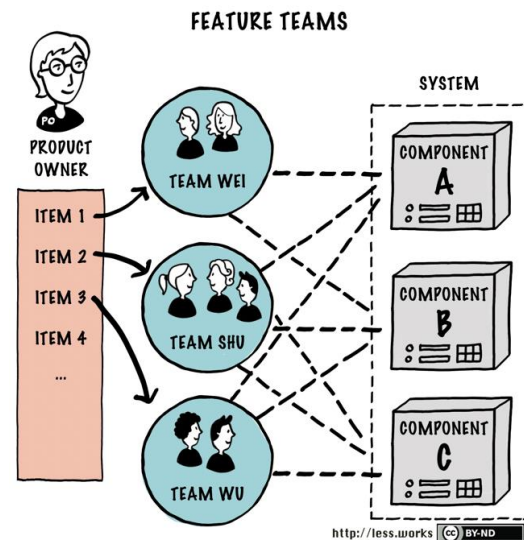
- Fokuser på én komponent, én del av det totale produktet, eller én del av utviklingsprosessen
- Teammedlemmene er alle spesialiserte på denne delen av systemet
- For eksempel:
 - Backend team
 - Front-end team
 - Test team
- Forstår sin egen komponent godt, men kan mangle overordnet systemforståelse



Kryss-funksjonelle (feature teams)

Det ideelle smidige teamet

- Bygger funksjoner/deler av et overordnet produkt
- Teammedlemmer er ikke låst til ett spesialistområde (dvs. de er fullstack)
- Ideelt sett kan enhver utvikler jobbe med hvilken som helst oppgave
- Bør jobbe sammen over lengre tid (fortrinnsvis samlokalisert)
- Typisk 7 ± 2 medlemmer



Diskuter med
naboen



Hvilke smidige prinsipper
har med teamarbeid og
samarbeid å gjøre?

De smidige prinsippene

1. Vår høyeste prioritet er å tilfredsstille kunden gjennom tidlige og kontinuerlige leveranser av programvare som har verdi.
2. Ønsk endringer i krav velkommen, selv sent i utviklingen. Smidige prosesser bruker endringer til å skape konkurransefortrinn for kunden.
3. Lever fungerende programvare hyppig, med et par ukers til et par måneders mellomrom. Jo oftere, desto bedre.
4. Forretningssiden og utviklerne må arbeide sammen daglig gjennom hele prosjektet.
5. Bygg prosjektet rundt motiverte personer. Gi dem miljøet og støtten de trenger, og stol på at de får jobben gjort.
6. Den mest effektive måten å formidle informasjon inn til og innad i et utviklingsteam, er å snakke ansikt til ansikt.
7. Fungerende programvare er det primære målet på fremdrift.
Smidige metoder fremmer bærekraftig programvareutvikling. Sponsorene, utviklerne og brukerne bør kunne holde et jevnt tempo hele tiden.
7. Kontinuerlig fokus på fremragende teknisk kvalitet og godt design fremmer smidighet.
8. Enkelhet – kunsten å maksimere mengden arbeid som ikke blir gjort – er essensielt.
9. De beste arkitekturer, krav og design vokser frem fra selvstyrte team.
10. Med jevne mellomrom reflekterer teamet over hvordan det kan bli mer effektivt og justerer adferden sin deretter.



STOR-SKALA SMIDIG



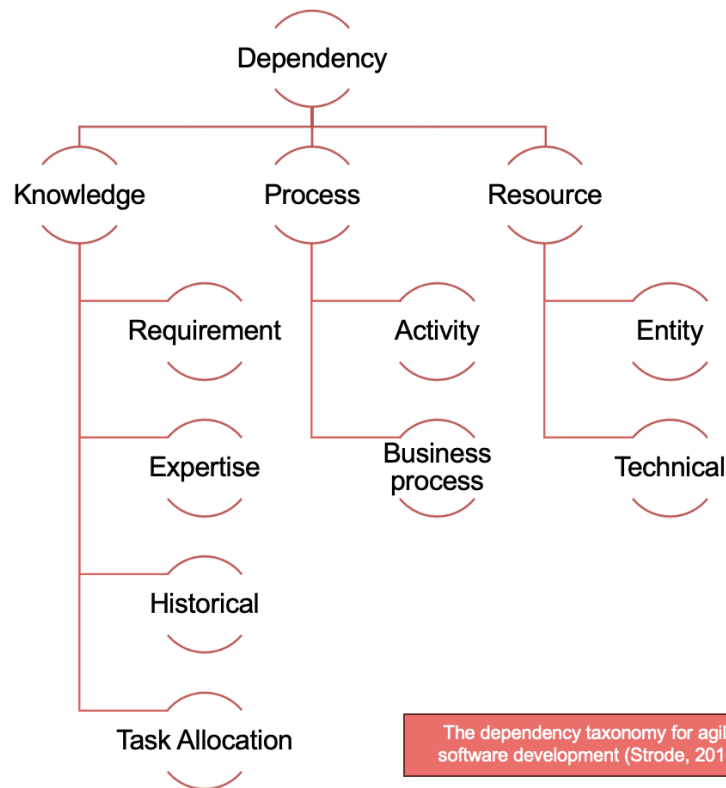
Hva er STOR-SKALA SMIDIG?

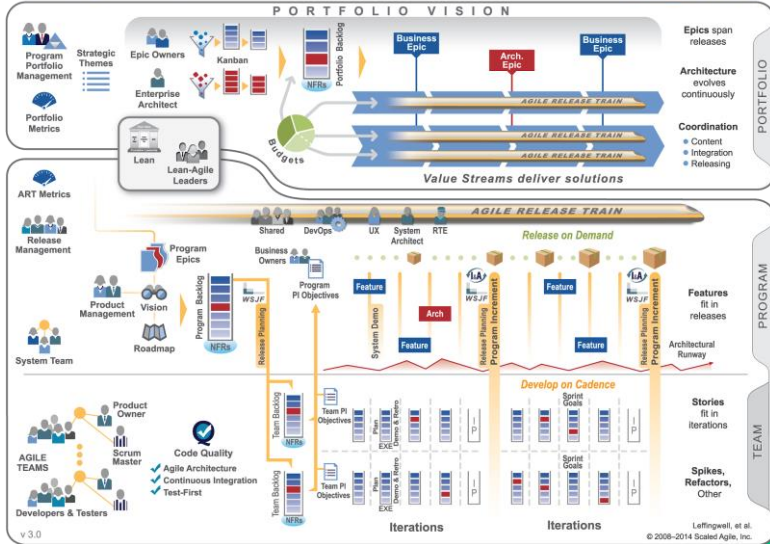
- «*En programvareutviklings-organisasjon med 50 eller flere personer eller minst seks team.*» (Dikert et al., 2016)
- *Stor-skala = 2-9 team, **Veldig** stor-skala = mer enn 10 team og som krever inter-team koordineringsforum* (Dingsøy et al., 2014)
- «*En kompleks integrasjon med ulike interne og eksterne informasjonssystemer og flere interessenter med ulike prioriteringer*» (Rolland et al., 2016)
- «*Adopsjon av smidige metoder i settinger med minst to team, eller adopsjon av smidige meotder på organisasjonsnivå på tvers av flere mulit-team konstellasjoner*» (Uludag et al., 2020)

Stor-skala utvikling krever
koordinering!

Hvorfor er koordinering viktig?

- Håndtere avhengigheter!
 - Blokkerer fart og leveranseevne både innad og mellom team
- Man kan være avhengig av:
 - Informasjon fra noen andre (kunnskapsavhengigheter)
 - At noe skjer før noe annet (prosessavhengigheter)
 - Bestemte verktøy eller tilganger (ressursavhengigheter)
 - At gitt kode eller arkitektur er på plass (teknisk avhengighet, en form for ressursavhengighet)

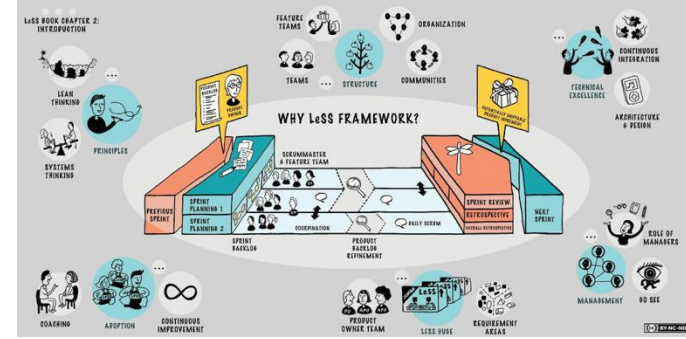




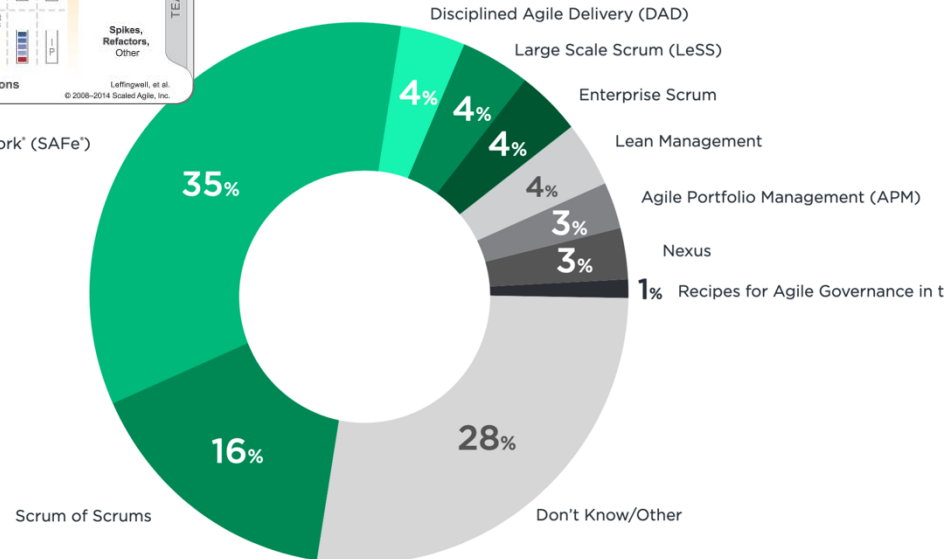
SAFe

Scaled Agile Framework® (SAFe®)

Stor-skala rammeverk



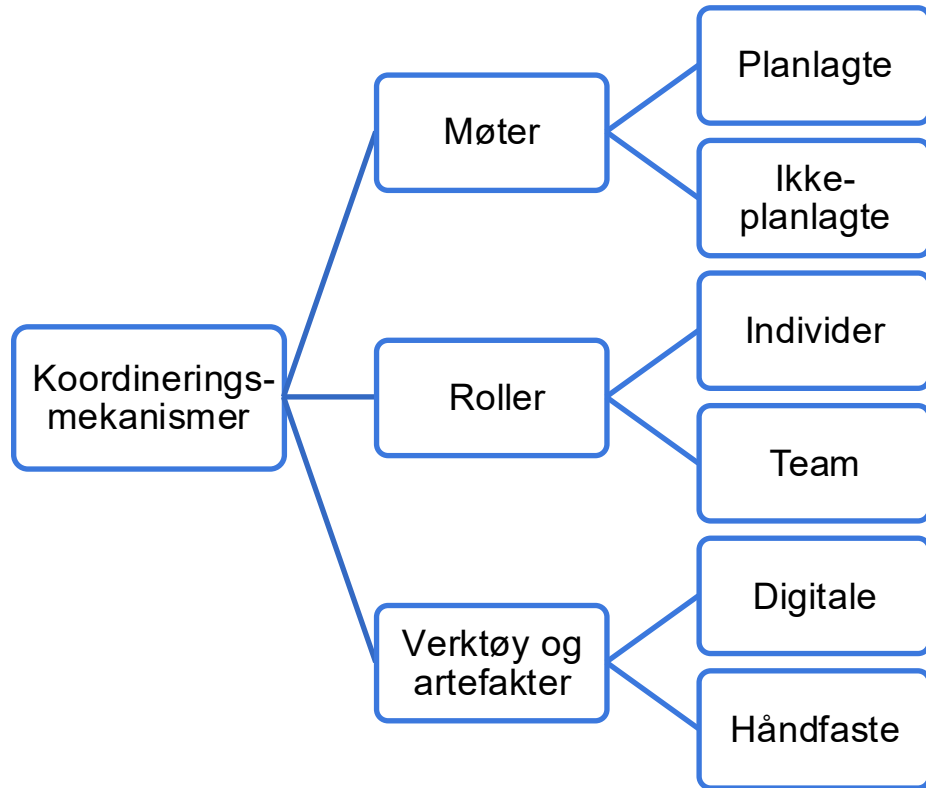
LeSS



Koordineringsmekanismer

Møter, roller, verktøy eller andre prosesser og gjenstander som brukes for å håndtere **avhengigheter** i en utviklingsprosess

Avhengigheter finnes både innad i team og mellom team





En **produsent** (rolle)
går gjennom
prioriteringer på en
multi-team
oppgavetavle

Møt DreamTeam

- 5 IFI-studenter: To fra PROSA, en fra Design, en fra Robotikk og en fra DigØk
- Skal jobbe sammen i 10 uker for å utvikle en mobilapp, og skrive og presentere en rapport samt gjennomføre teamaktiviteter
- To på teamet kjenner hverandre før, resten av dem har ikke møtt hverandre
- De får instruksjoner og veiledning underveis, og forventes å jobbe som et selvorganiserende, smidig team.
- Ingen av studentene har jobbet på en slik måte før!



SKUESPILL!

Oppsummering av skuespill



- Poeng: Fange opp situasjonen og gjøre det beste ut av det
- Fasilitator prøver å løse problemet med forsentkommingen, i fellesskap, saklig og respektfullt.
- Kjefter eller bebreider ikke, men skjuler heller ikke at dette har skjedd før
- Fasilitator prøver å få hjelp av andre til å komme i tide
 - uttrykk for tvil om at personen vil klare det alene,
 - håndsrekning (hjelp og sparring).
- På neste møte innrømmer personen at det er en underliggende grunn til forsentkommingen, og dette blir tatt imot på en respektfull og løsningsorientert måte av fasilitator og resten av teamet.

UTFORDRINGER I TEAMARBEID



Diskuter med
naboen og svar
på menti:



Hva slags problemer kan
man støte på når man
jobber i team?

Noen utfordringer i teamarbeid

- At ikke alle bidrar (social loafing)
- Varierende grad av motivasjon
- Varierende grad av kunnskap
- Ulik personlig stil og preferanse for arbeidsmetodikk
- Ikke alle kommer like godt overens!

Teammedlemmer som ikke bidrar

- Hva skal vi gjøre når Tom aldri stiller opp på møtene? Tom er sjelden med i undervisningen og det virker som det er helt andre ting enn prosjektarbeidet som blir prioritert.
 - Skal vi ha med navnet på prosjektrapporten?
 - Start med å undersøke **HVORFOR**
 - **Be om hjelp!**



Manglende innsats har generelt to årsaker:

- Personlige årsaker:
 - Vanskeligheter i privatlivet eller sykdom
 - En feilaktig forestilling om at kurset kan gjennomføres med minimal innsats
 - Manglende motivasjon
- Teamdynamikken. Kan bli kjørt ut på et sidespor. Deltar i starten, men opplever at det blir vanskelig å bidra, pga. teamets måte å jobbe på.

«Jeg deltok i starten, men hver gang vi produserte noe, ble alt det jeg hadde gjort skrevet om. Jeg følte meg dum og verdiløs og lot etter hvert være å møte opp»

Ulike preferanser i arbeidsmåter kan skape gnisninger

“Jeg vil helst bare jobbe etter kl 16 for da er jeg mest opplagt”

“Jeg jobber best alene!”

“Jeg venter med å gjøre det vi har avtalt til etter trening”

Ulike preferanser → kompromisser alle kan leve med



- Bruk tid I begynnelsen av teamarbeidet til å etablere rutiner som alle ønsker å jobbe etter!
- Skriv dem ned, gjerne i *teamkontrakten* (stay tuned)
- Revider rutinene underveis, gjerne under et *retrospektiv*

Planlegging av arbeidet er viktig

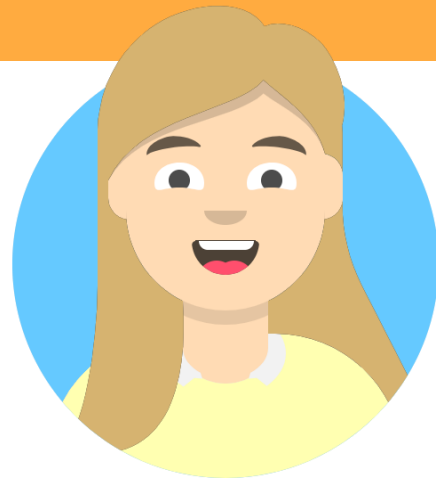
- **Eksempel – skrive en rapport sammen**
 - Hva skal kapitlet handle om, og hvordan strukturere underkapitler.
 - Viktig at teamet diskuterer hvordan en oppgave skal løses sammen.
 - Vanskelig å skrive oppgave sammen. Noen vil at det skal være bedre, og noen er flinkere til å skrive enn andre.
 - Må ofte gi slipp på det perfekte.
 - Fordeling av oppgaver er en av de største utfordringene.
 - Diskuter opplegget.
 - hva er målet etc. før fordeling.

TIPS TIL EFFEKTIVT TEAMARBEID



Fasilitering av arbeidsprosessene

- Gir en form for struktur
 - Hvis ikke - lett for at de som dominerer tar styringen
- Fasilitator (f.eks Scrum Master)
 - Styrer formen – ikke innholdet
 - Struktur, rekkefølge
 - Ikke prosjektleder, ingen ansvar for hvem som skal utføre hvilke oppgaver
 - Fasilitator har kun ansvar for å sikre at samtalen og arbeidet i teamet organiseres eller formes best mulig.
 - Hva teammedlemmene presterer innenfor avtalte rammer, er *deres* ansvar.



Gode praksiser

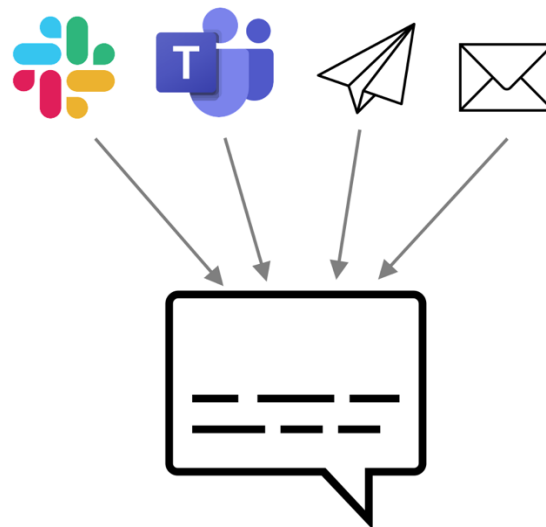
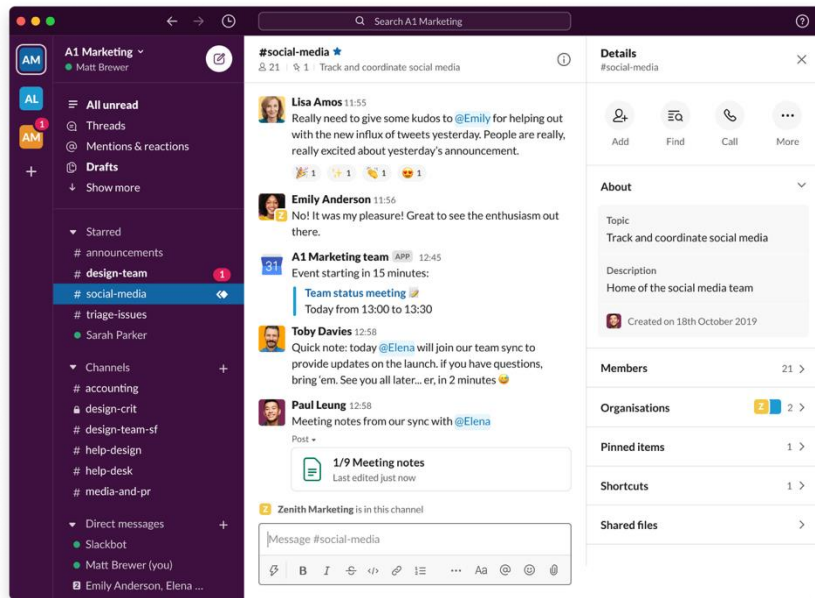
- Blokkere tid i kalenderen om morgenen til fokusert arbeid
- Kortere møter
- Parprogrammering



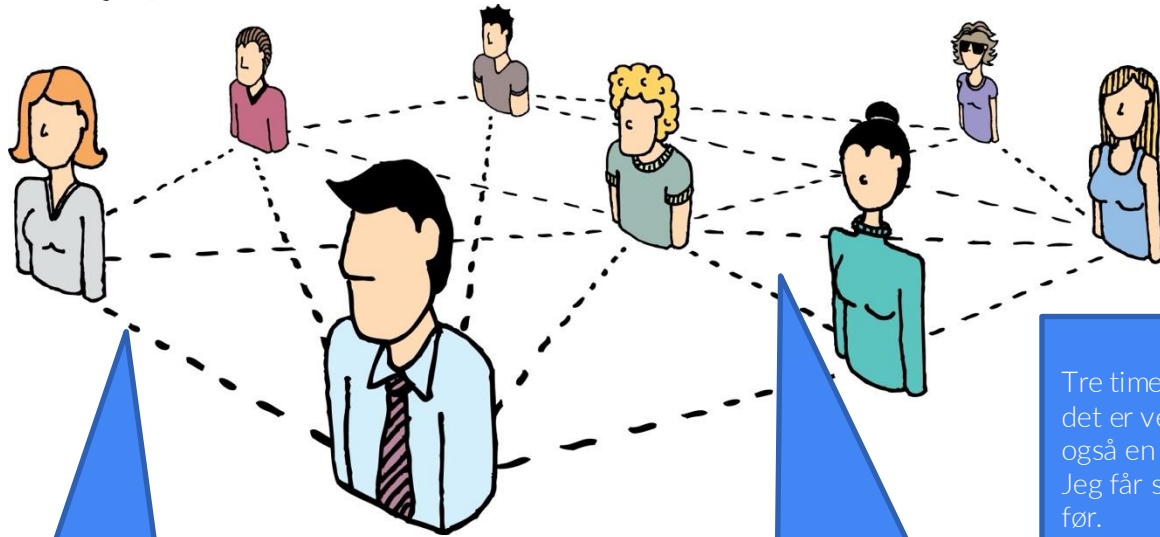
Tips til effektive møter

- **Lage dagsorden** (forberedelse)
- **Start gjerne 5, 10 eller 15 minutter over hel**, så folk får tid til å lande, få en kopp kaffe, sagt hei etc. Snakker som regel mest med dem man kjenner best. Fasilitator oppmerksom på at for eksempel en sitter alene.
- **Start til avtalt tid**. Viktig å sende et signal om at det ikke er OK å komme for sent. Alle kan komme for sent av og til, men ikke hver gang! Ikke gjør stort nummer av det, men la det likevel få konsekvenser!
- **Oppfølging av avtaler fra forrige møte**. Kan være følsomt, da mange kanskje ikke har fulgt opp. Fasilitator spiller en viktig rolle her. Søker løsning er viktig, igjen smidig. Viktig ikke å henge ut den som ikke har gjort det den skal, men likevel det blir lagt merke til!
- **Fasilitering av skriving**. Styre prosessen for å få form og struktur på oppgaveskrivingen, ikke om det skal brukes 2 eller 6 timer på problemstilling. Må sørge for felles forståelse, få folk til å gi feedback til hverandre...
- **Fordeling av oppgaver**. Fasilitator skal *ikke bestemme hvem som skal gjøre hva*, men kan sikre at beslutninger tas og at noen utfører dem slik at teamet får gjort et de skal.
- **Konklusjon og avslutning av møtet**. Kan være vanskelig for en selvstyrt gruppe å avslutte, ofte noen som har mer å si. Fasilitator må ta en vurdering. Slutt i god tid (noen tripper for å komme av gårde, noen diskuterer videre) slik at man har tid til å gjøre nye avtaler. Viktig at fasilitator også ser om alle punktene nås på dagsorden for eks en halvtime før slutt for å se hvilke punkter som kan utsettes til neste gang.
- **Evaluering av møtene – bruk retrospektiv**. Kan også ha retrospektiv av selve møtene. Hva fungerer, hva kan gjøres bedre. Gjerne tre ting (en på hva går bra, to på hva kan gjøres bedre). a) fasiliteringen, b) alle i teamet, c) et bestemt teammedlem.

Etabler kanaler for skriftlig kommunikasjon



Får kjapt kontakt med alle



Så det er jo de som trenger tett kontakt med diverse leverandører, de synes jo det er veldig fint å ha dem i Slack og slippe den mail-tråden i alle mulige retninger.

Jeg koordinerer og kommuniserer med andre hovedsakelig på Slack og så har jeg en sånn stand-up-call tre ganger i uken. Etter stand-up har jeg ofte direkte dialog med en på Slack eller en Slack video call.

Tre timer tok det å få svar på denne meldingen her og det er vel det lengste jeg har måttet vente, men det er også en «big shot guy», viktig fyr som har masse å gjøre. Jeg får som oftest svar innen en halvtime, gjerne mye før.

Bruk parprogrammering som et verktøy for samarbeid!



- Best fysisk, men fungerer også bra digitalt
- Anbefales å bytte roller og par underveis!
- Mange foretrekker å jobbe med de samme personene fra gang til gang
- De fleste foretrekker å jobbe sammen med noen som er på omtrent samme nivå som dem selv

Etablér en product-backlog

- Samler alle slags oppgaver
 - Kunde-spesifikke (Få en oversikt over luftkvaliteten i byer)
 - Forbedringsmål (Restrukturere kode)
 - Skrive innledning i rapporten
- “Items” i produktkøen kan uttrykkes som
 - User stories
 - Use Cases
 - Andre måter å spesifisere krav på som passer

	Nye estimater i Sprint					
Prioritet	Brukerhistorie	Estimat (Initielt)	1	2	3	4
1	Som en bruker ønsker jeg å få en oversikt over luftkvalitet i Oslo en bestemt dag	8				
....						
	Restrukturer koden for å ta i bruk pattern MVVM (Model View View Model)	13				

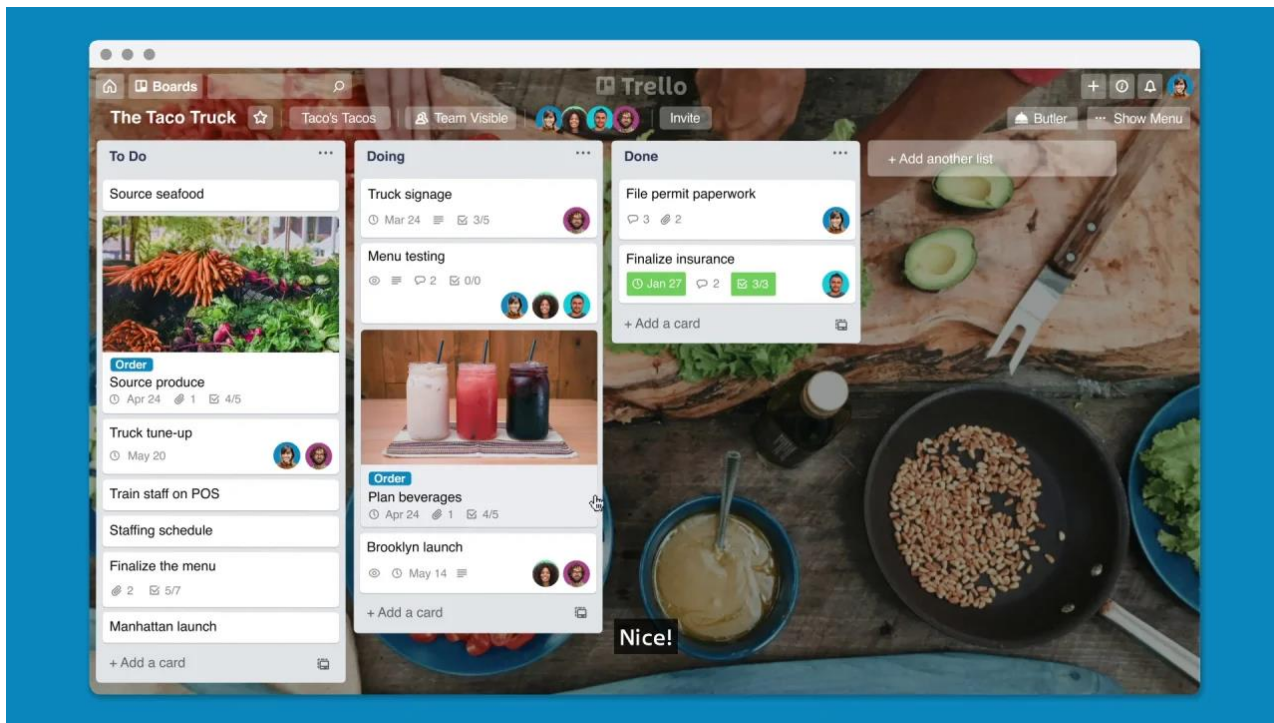
Sprint-planlegging gjøres ofte i egne møter

En måte å lage sprint backlog



		Nye estimater etter dag					
Brukerhistorie	Oppgave i Sprint	Estimat (Initielt)	Team medlem	1	2	3	4
Som en bruker ønsker jeg å få en oversikt over luftkvalitet i Oslo en bestemt dag	Finne riktig API og tolke data	2	Syver				
	Lage datastruktur - klasser	5	Anne				
	Vise kart med fargekode	8					
	Gi tekstlig beskrivelse i kartet	5					
Restrukturer koden for å ta i bruk pattern «Model View Presenter» eller MVVM							

Bruk arbeidsflytverktøy



Teamkontrakt

- Skal være del av oblig 3 og kan for eksempel inneholde noe om:
 - Tilstedeværelse
 - Tidsbruk
 - Forventninger til den enkeltes bidrag
 - Hva skjer ved avvik eller uenigheter
- Bruk denne aktivt som del av konflikthåndtering



Konflikthåndtering i gruppearbeidet

Tre steg:

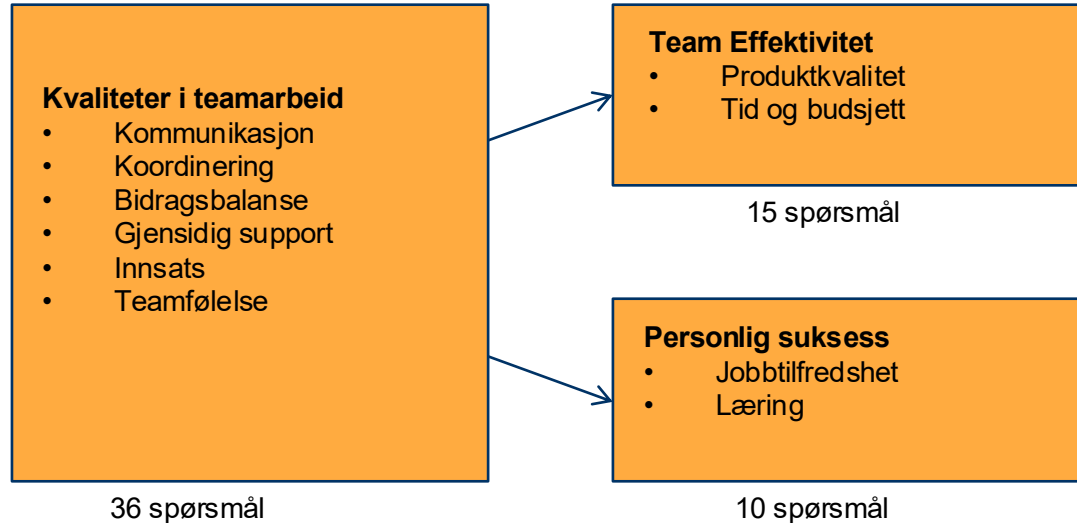
1. Bruk teamkontrakten
2. Snakk med veileder
3. Involver fagledelsen



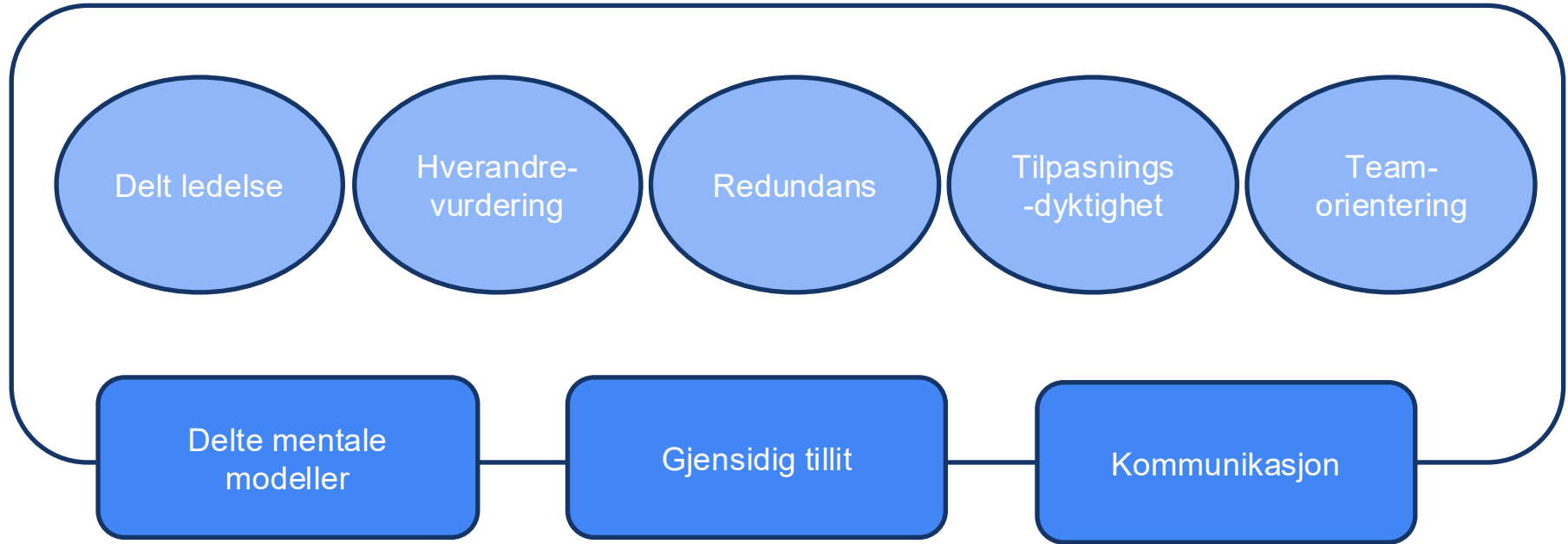
Hva sier forskning om teameffektivitet?

- Finnes mange (!) modeller som tar for seg teamarbeid
- Mye innen gruppepsykologi, men vi har også egne modeller innen software engineering
- Ser på spørsmål som:
 - Hvordan få alle teammedlemmene motivert og få ut potensiale?
 - Kommunikasjon, koordinering av oppgaver, god teamfølelse, tillit til hverandre, teamledelse,.....

Teameffektivitets-modell i smidige team (TWQ)



Agile teamwork effectiveness model - ATEM



Koordineringsmekanismer med atferdsmarkører i ATEM-modellen

Koordineringsmekanismer for teamarbeid	Atferdsmarkører
Delte mentale modeller	• Forutse og forutsi hverandres behov. • Felles forståelse av arbeidsprosess. • Felles forståelse av oppgaver. • Felles forståelse av individuelle ferdigheter.
Kommunikasjon	• Teamet følger opp fremdrift av oppgaver. • Visualisere prosjektinformasjon. • Tilrettelegge for uformell kommunikasjon.n.
Gjensidig tillit	• Informasjonsdeling. • Vilje til å innrømme feil og ta imot tilbakemeldinger. • Støttende miljø.

Teamarbeidskomponenter med atferdsmarkører, ATEM-modell

Teamarbeids-komponent	Atferdsmarkører
Delt ledelse	• Det smidige teamet legger til rette for teamproblemløsning. • Det smidige teamet synkroniserer og kombinerer bidrag fra individuelle teammedlemmer ved å bruke smidige praksiser kombinert med automatiserte verktøy. • Det smidige teamet søker og evaluerer informasjon som påvirker teamfunksjonen. • Smidige verdier og metoder bestemmer rollene til teammedlemmene. • Smidige verdier og metoder bestemmer hyppigheten og typen av forberedende møter og tilbakemeldinger.
Team-orientering	• Ta hensyn til alternative løsninger levert av andre teammedlemmer og vurdere disse innspillene for å finne ut hva som er mest riktig. • Økt oppgaveinvolvering, informasjonsdeling, strategisering og deltakende målsetting. • Teamet "holder sammen og forblir samlet".
Redundans	• Erkjennelse av at det er et problem med fordeling av arbeidsbelastning i teamet deres. • Flytting av arbeidsansvar til underutnyttede teammedlemmer. • Gjennomføring av hele oppgaven eller deler av oppgaver av andre teammedlemmer.
Tilpasnings-evne	• Identifiser tegn på at en endring har skjedd, tildel endringen mening og lag en plan for å håndtere endringene. • Identifiser muligheter for forbedring og innovasjon for vanlige eller rutinemessige praksiser. • Vær oppmerksom på endringer i teamets interne og eksterne miljø.
Peer feedback	• Identifiser feil og bortfall i andre teammedlemmers handlinger. • Regelmessige tilbakemeldinger angående teammedlemshandlinger for å lette selvkorrigerings.

Informasjon om rapport og aktiviteter

- Dere skal utføre teamaktiviteter rapportere på at disse er gjennomført – poenggivende!
 - Møter med veileder
 - Parprogrammering
 - Brukertestning
 - Retrospektiv
- Dere får poeng både for rapporten og aktiviteter!



Informasjon om rapporten

- Start å skrive tidlig (ikke vent til dere er ferdige med å utvikle appen)
- Struktur
 - Presentasjon av teamet og case
 - Kort sammendrag
 - Brukerdokumentasjon (noen skjermbilder ...)
 - Kravspesifikasjon (backlog- user stories,...)
 - Modellering (use case diagram, klassediagram, sekvensdiagram...) (legges i prosjektet i GIT)
 - Produktdokumentasjon (mer teknisk; arkitektur, teknologier....)
 - Testdokumentasjon (enhetstesting, integrasjonstesting,...)
 - Prosessdokumentasjon
 - Endringer, sprinter (sprint backlog – fra user stories til tasks)
 - Refleksjon (hvordan har teamet fungert, hva kunne vært gjort annerledes.....)
 - Referanser, kilder
 - Eventuelle vedlegg (f.eks. intervjuguide, spørreskjema, eksempel på kildekode...)