Dette dokumentet er basert på EMMA´S IN1030 BIBEL

Og er hovedsaklig en redesign av av den orginale filen til Emma.

Innholdsfortegnelse:

MODUL A **4**
Samspill – mennesker & teknologier 4
Rike bilder 4
Elementer i metoden 5
Notasjon 5
Eksempel på et rikt bilde 6
Hva er en interessent? 6
Hvem er en aktør? 7
Undersøkelser av bruk 7
Hva er brukerundersøkelse? 8
Hvorfor brukerundersøkelser? 8
Datainnsamling 10
Teoretisk planlegging 10
Praktisk planlegging 10
Pilotintervju 11
Samtykkeerklæring 11
Sekvens av handlinger 12
Situasjonsbestemte handlinger 12
Sekvenstabell 12
Oppmerksomhet og distraksjon 14
Etikk og lover 15
Hva er etikk? 15
Personopplysningsloven/ GDPR 15
Hva er en personopplysning? 16
Arbeidsmiljøloven 16
Åndsverkloven 17
Likestillings- og diskrimineringsloven 17
Universell utforming 17
Tilgjengelighet og Inkludering 18
WCAG 18
De fire prinsippene 18
Hva er etikk? 19
Etisk refleksivitet 19
Personvern 20
Hva er personvern? 20
Dataportabilitet 20
Særlige personopplysninger 21
Personvernerklæring: 22
Vilkår for bruk 22
Tilgjengelighetserklæring 22

MODUL B **23**
Hva er systemutvikling? 23

Eksempler på roller 24
Hvilke typiske aktiviteter inngår i en systemutviklingsprosess? 25
Usikkerheter i et prosjekt 25
Vedlikehold 25
Hva er prosessmodellen? 25
Prosessmodeller versus den reelle prosessen? 26
Ukesoppgaver 27
Plandreven utvikling 29
Smidig utvikling 30
Inkrement og iterasjon i systemutvikling 30
Scrum 31
Fordeler ved Scrum for ansatte 32
Large scale scrum 34
Kanban 35
Forelder ved Kanban 35
Kundeinvolvering 36
Prosjektplanlegging 37
Smidige praksiser og teamarbeid 37
Teamarbeid 37
Smidig tilnærming 37
De tre mest brukte smidige teknikkene (Scrum) 38
12 prinsippene 39
Krav 39
Hva er krav? 40
Brukerkrav 40
Systemkrav 40
Funksjonelle krav 41
Ikke-Funksjonelle krav 41
Kravhåndtering 43
Kravspesifikasjon 44
Brukerhistorie 45
UML - modellering 45
Hvorfor modellerer vi? 45
Use case (brukstilfelle) 46
Notasjon: 46
Use case – identifisering 46
Forklaring 46
To relasjoner i et use case 47
Include relasjonen 47
Extend-relasjonen 47
Tekstlig beskrivelse 48
Use case vs. User stories 49
Sekvensdiagrammer 50
Hvorfor er det nyttig å benytte sekvensdiagram? 50

Modellering AV SEKVENSDIAGRAMMER 51
Aktivitetsdiagrammer 51
Hva er et aktivitetsdiagram? 52
Hvorfor er det nyttig å lage aktivitetsdiagram? 52
Klassediagram 53
Design pattern 54
Kohesjon 55
Kobling 55
Informasjonssikkerhet 56
Konfidensialitet 56
Integritet 56
Tilgjengelighet 56
Nyttige begreper innen informasjonssikkerhet 58
Risiko 58
Risikonivå 58
Trussel 58
Sårbarhet 58
Sikkerhetstiltak 59
Trusselmodellering 59
Redundante system 59
Hva er sikkerhet? 59
Hva er informasjonssikkerhet? 60
Autentisering 60
Autorisering 61
Økende behov for informasjonssikkerhet 61
Innebygd personvern og informasjonssikkerhet 62
Sikkerhetstiltak 63
Risiko 64
Risikoanalyse 64
Risikohåndtering 64
Risikomatrise 64
DevOps 65
Prinsipper 66
Fordeler 66
Kodeadministrasjon 66
Automatisering 67
Måling 68
IT-kontrakter 69
Innhold 69
Ulike kategorier 69
Utfordringer 69
Smidig 70
Plandrevet 70

MODUL A

Samspill – mennesker & teknologier

Læringsmål: Samspill og interessenter: du skal kunne drøfte samspillet mellom teknologi og individer, organisasjon og samfunn.

Rike bilder

Nøkkelord knyttet til rike bilder:

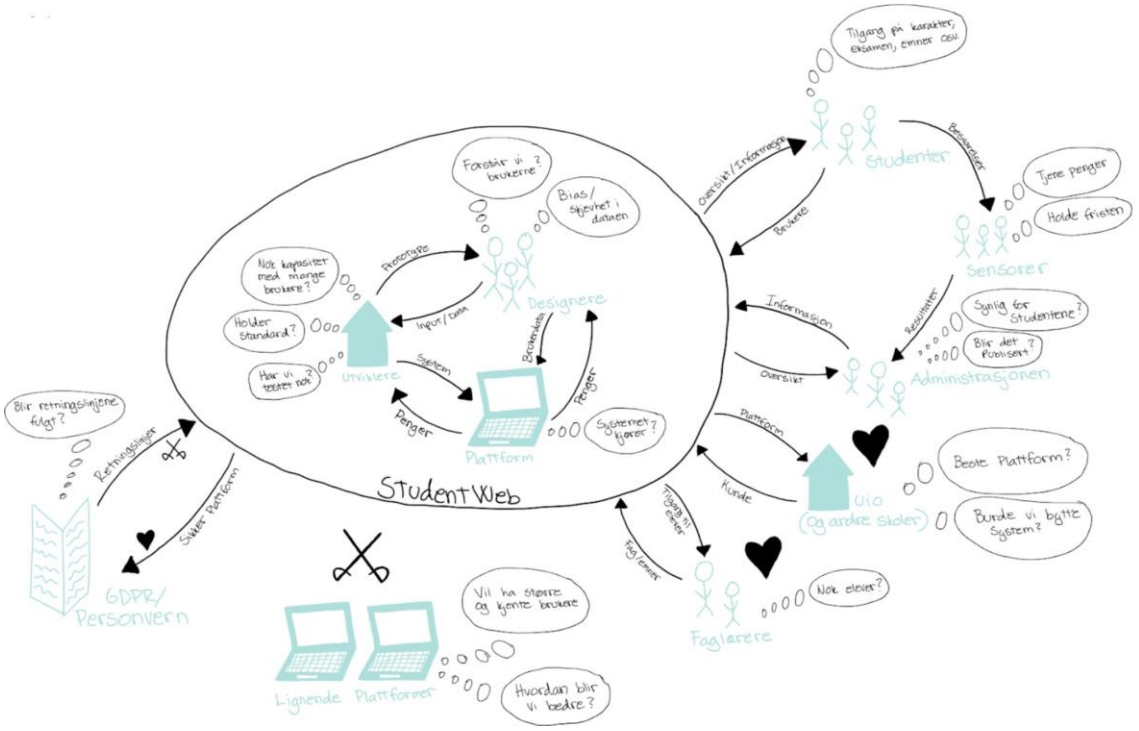
Rike bilder brukes for å få oversikt over situasjonen og alle som er involverte eller har interesse i situasjonen.

- Må velge hva som skal med, man forenkler en virkelighet som er kompleks. Det er derfor viktig å tenke over avgrensningene i bildet (det man ikke tar med) er med på å avgrense problemområdet.
- Et verktøy for å klargjøre tanker og skaffe deg oversikt
- Er en teknikk for beskrivelse og analyse av (problematiske) situasjoner
- Illustrerer avhengigheter

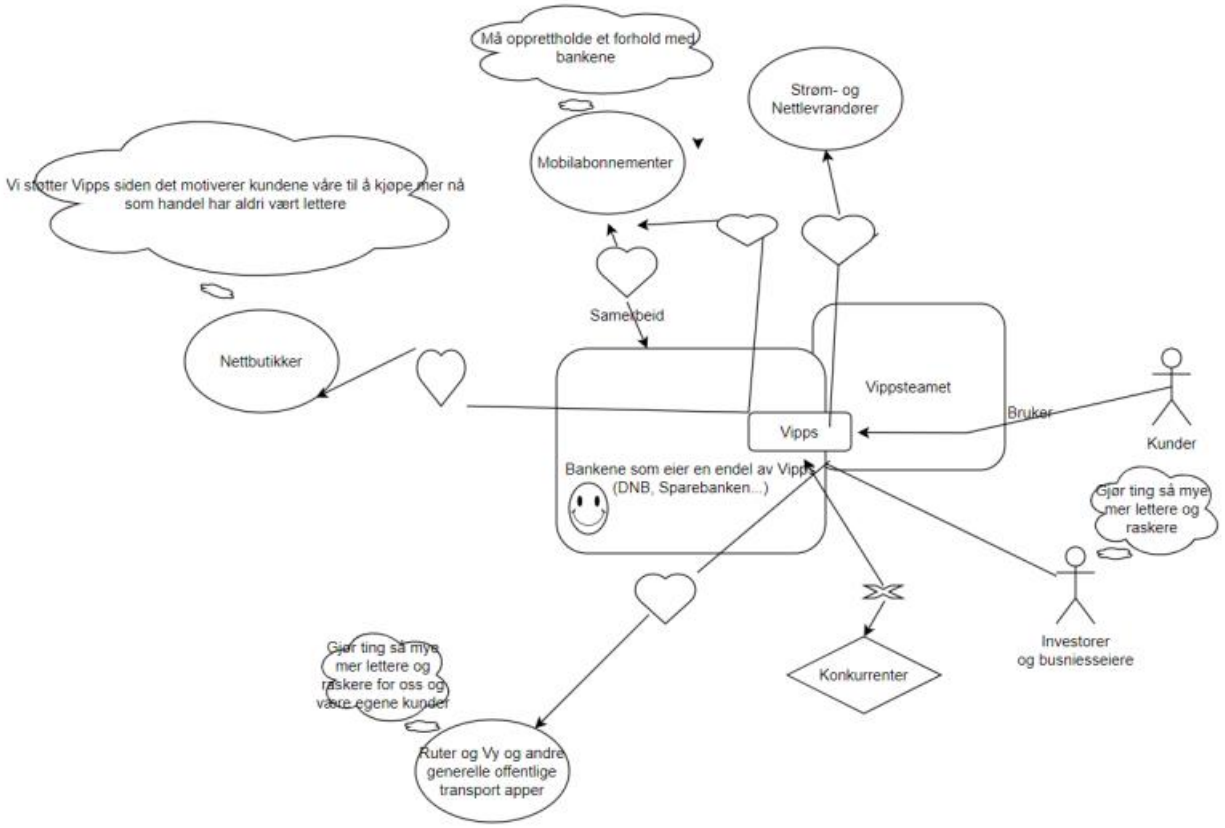
Notasjon

- **Kryssende sverd:** konflikt
- **Piler:** samspill eller avhengighet
- **Hjerter:** godt samarbeid
- **Tanke/snakkebobler:** concerns

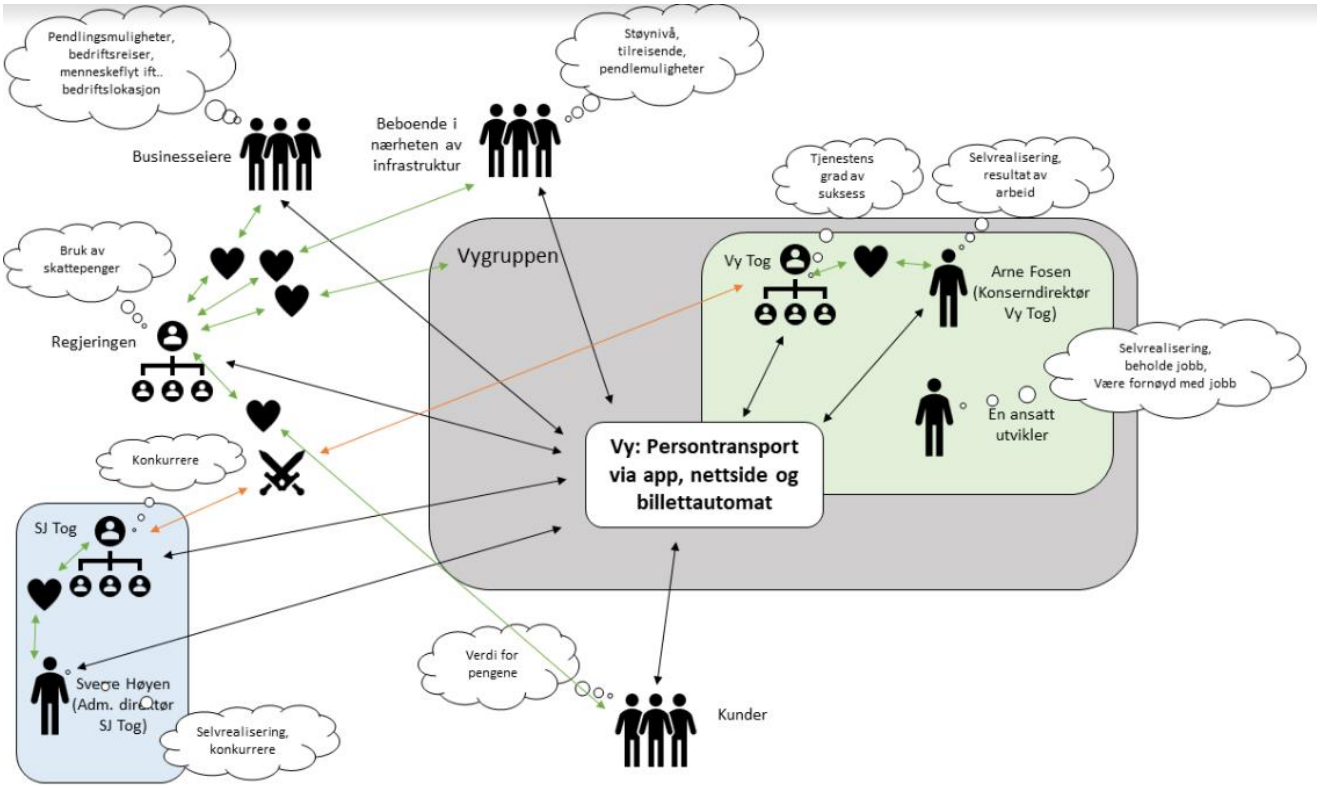
Eksempel på et rikt bilde



Min shitty variant av et rikt bilde:



Fra Tobias Rade Evensen 2021V



Hva er en interessent

Det kan være:

- En person, gruppe, organisasjon eller institusjon
    - konkurrenter, utviklere, ansatte, produsent osv.
- For en interessent har utforming av en løsning eller utfallet av en hendelse betydning

Ofte er vi ute etter tre egenskaper hos interessentene:

- Interesse (Concern)
- Relasjon mellom systemet og interessent (Evt. konflikter)
- Relasjon mellom interessentene (Evt. konflikter mellom interessentene)

Hvem er en aktør?

Det dekker alle som har noe å si/interesse i systemet og som derfor vil legge føringer på systemet. Aktør blir da betegnelsen for å alle grupper eller personer som anvender systemet

- Det kan også være øvrige systemer som blir anvendt av systemet
- To typer aktører:

- Sekundæraktører: Er nødvendig for å realisere målene til primæraktør
- Primæraktører: har egne mål med systemet

En aktør for et system representerer en rolle som menneske eller et annet system som har et mål med systemet. Vi har to typer aktører: primære og sekundære. Primære aktører har et eget mål. Og sekundære aktører trengs for å oppfylle de primære aktørers mål.

!!Det er ofte flere interessenter enn aktører, og en aktør er som også en i. interessent

Undersøkelser av bruk

Læringsmål: Du skal kunne gjennomføre enkle brukerundersøkelser

Ting vi ofte lurer på som vi kan finne ut av ved hjelp av brukerundersøkelser:

- Er det brukeren eller systemet det er feil med?

- Er det noen feil med brukeren?
- Er brukere dumme? Handler det om samspillet?

Hva er brukerundersøkelse?

- Det er: involvering av brukergruppe eller representanter for brukergruppe for å samle inn data om bruken av et produkt/system. Kan være observasjon eller intervju.

Hvorfor brukerundersøkelser?

Vi ønsker å forstå noe som vi lurer på, og gjennom det bringe frem ny kunnskap som skal være til nytte for individer, organisasjoner og kanskje hele samfunnet.

- **Utvikle** nye produkter, tjenester, omgivelser, interaksjonsmekanismer
- **Forberede** noe eksisterende
- **Evaluere** noe, hvordan fungerer det i praksis
- **Forstå** noe, utforske, undersøke, etablere kunnskap
- **Analysere**, kartlegge, teste
- Vi ønsker å forstå noe som vi lurer på, og gjennom det bringe frem ny kunnskap som skal være til nytte for individer, organisasjoner og kanskje hele samfunnet

- Brukere må/bør/skal/vil være med!
- Universell utforming – tilgjengelighet – inkludering
- Alle skal være med
- Kostnader/ressurser ved utvikling
  - Historien viser seg at det kan være «dyrt» å ikke ha med brukere

“What people say, what they do, and what they say they do are entirely different things” - Vi må derfor “forske” for å finne ut hva som faktisk er sannheten

Alle datainnsamlinger har samme mål:

fra

rådata

til

informasjon

til

forståelse/konklusjon

Utføring av intervju:

- Planlegge: sette oppgaver, mål, pilotere, rekruttere, samtykke
- Gjennomføre: samle inn data
- Beskrive og presentere dataene
- Analysere dataene
- Bruke resultatene

Brukervennlige løsninger

- Lett å lære
- Effektivt
- Lett å huske
- Relativt feilfritt og feiltolerant
- Behagelig å bruke

Brukerundersøkelser kan for eksempel gjøres gjennom å operasjonalisere noe av det overfor – og undersøke spesifikke brukskvaliteter (må definere noen metrikker). Gjennom å være mer spesifikk enn å kun bruke ordet brukervennlig, så kan vi gå inn og finne mer ut av enkelte deler av brukskvaliteten, som for eksempel hvordan det er å lære for en gitt bruker, for en gitt oppgave, i en gitt kontekst.

Viktige spørsmål å stille seg

- Hvem er brukerne mine? Hva prøver brukerne mine å oppnå?
    - er de unge/gamle
    - er de eksperter eller ikke
        - hvilke ord/begreper kan du bruke. En ekspert vil kunne forstå fagbegreper
- Hvordan kan mitt system best mulig støtte brukeren i sine oppgaver?
- Hvilke opplevelser synes brukerne mine er gode og givende?
- Hvordan bør mitt system oppføre seg?
- Hvilken form bør mitt system ha?
- Hvordan skal brukerne interagere med produktet?
- Hvordan kan mitt systems funksjoner best mulig organiseres?
- Hvordan skal mitt system fremstå før førstegangsbrukere?
- Hvordan kan mitt system virke forståelig tiltrekkende og kontrollerbar?

Datainnsamling

Det krever mye planlegging i forkant. Vi har med mennesker å gjøre og vi kan ikke styre dem - det vil vi heller ikke - målet er en objektiv datainnsamling. Det er flere ting som kan spille inn og påvirke:

- De har en dårlig dag
    - syk/stressa/sliten osv.
- Omgivelsene
    - komme forstyrrelser
    - høye lyder, dårlig vær osv.
- Teknisk svikt
    - lyden blir ikke tatt opp
    - internettforbindelse (veldig relevant i korona tider)

Teoretisk planlegging

- Hvilken type data ønsker vi å samle inn?
- Hvilke spørsmål skal vi stille i et intervju?
- Hva forventer vi i observasjonen av bruk?
- Trenger vi samtykke? Personvern, sensitiv data
- Kanskje ha noen oppfølgingsspørsmål på lager

Praktisk planlegging

- Hvordan skal vi samle inn data?
    - Lab eller naturlige
    - Intervju, observasjon, spørreundersøkelse (survey)
- Hvilke redskap/verktøy trenger vi for å gjøre en god datainnsamling?
    - Lydopptaker
    - Videokamera
    - Kamera
    - Iphone
    - Eller annen prototype
- Hva kan forstyrre datainnsamlingen?
    - Omgivelsene
    - Bakgrunnen til personen

- Hvordan skal samtykkeskjema utformes?
    - På papir
    - Skal den leses opp også har man på video at de har godkjent
    - Elektronisk dokument som de kan signere

Pilotintervju

Det er en testundersøkelse som skal avdekke feil og mangler ved metodebruk før man begynner med hovedstudie. Man kan se på det som en form for generalprøve før selve hovedundersøkelsen. Her kan det da være smart å gjennomføre pilot på en som ligner den du skal gjennomføre hovedstudien på.

- teste teknikken
- teste spørsmålene sine
    - spør jeg etter det jeg egentlig lurer på?

Samtykkeerklæring

En kontrakt mellom deltaker og systemutviklere over hvilke rettigheter deltaker har i forhold til data som blir samlet inn. Bør og spesifisere hva data skal brukes til, hvordan data skal bli samlet inn, tilgjengeligheten til data, hvordan og hvor lenge data skal lagres.

- Fine ting å ha med:
    - Mulighet til å se over materialet som blir brukt
    - Kontaktinformasjon
    - Krav om anonymitet
    - Frivillig – kan trekke seg når som helst
    - Hvor lenge blir dataen lagret
    - Hva skal dataen brukes til

Kreves en aktiv handling à muntlig/skriftlig «ja»/signatur

Måling, indikator, tolkning – kvantitativ og kvalitativ

- Måling/indikator (enklere å teste hvis man har spesifikke metrikker å teste - kvantitativ data)
    - Tid
    - Antall klikk
    - Antall forsøk

- Indikator/tolkning (kvalitativ data)
    - Kjempebra!
    - Helt forferdelig
    - Funket fint

Sekvens av handlinger

Situasjonsbestemte handlinger

Kopimaskin eksempelet med utdannende mennesker som skulle prøve å bruke en kopieringsmaskin (*kan ikke finne videoen, hvis jeg finner den skal jeg legge en link*)

- En av målene hennes var å vise at vi handler ut fra situasjonen, og at vi tolker handlinger og det vi skal gjøre ut fra omstendighetene i en konkret situasjon.
- Konteksten spiller en veldig viktig rolle til brukeren
- I brukssituasjonen så justeres og tolkes de planene av brukeren
- Kopimaskinene endrer seg ikke, men hvordan brukere tolker den vil endre seg i situasjonen

Sekvenstabell

Tenk på det som en tidslinje til hvordan ting skjer under undersøkelsen. Vi ønsker å studere interaksjonen mellom bruker og teknologi og vi kan ta i bruk sekvenstabellen for å gjøre dette:

- Hva skjer når?
- Hvem aktiverer hvem?
- Når tar det lang tid?
- Hvorfor?
- Kan interaksjonen forbedres?

Hensikt: vi ønsker å se hva som skjer i samhandling mellom maskin og menneske og identifisere hvor det oppstår konflikter/utydeligheter.

Kolonne 1 – Bruker: handling ikke synlig for maskin

I denne kolonnen fører man inn det brukeren tenker høyt, gjør og sier, kroppsspråket – alt menneskelige som ikke er direkte interaksjon med maskinen. Det kan for eksempel være «tar seg til hodet», «himler ned øynene», «sukker høyt» eller «sier: jeg forstår ikke hvordan jeg skal gjøre dette», «hvordan kommer jeg meg videre herfra» osv.

Kolonne 2 – Bruker: handling synlig for maskinen

I denne kolonnen skal man føre inn brukerens direkte interaksjon med systemet. Hva er det brukeren trykker på, skriver evt. sier, som systemet vil registrere. Dette kan for eksempel være at man trykker på et ikon, skriver inn et søkeord og trykker søk, markerer tekst osv.

Kolonne 3 – Maskin: effekt synlig for bruker

I denne kolonnen fører man inn effekten av interaksjonen som bruker har med systemet og som bruker oppfatter. Det vil si at dersom bruker har skrevet inn et søkeord og trykket søk, vil man her føre inn hvilke søkeresultater bruker blir presentert for. Eventuelt hvis bruker trykker «lagre» i et dokument og maskinen responderer med «dokumentet lagret»

Kolonne 4 – Maskin: effekt ikke synlig for bruker (design rasjonale)

I denne kolonnen føres inn det som skjer inne i maskinen, dvs. alle algoritmer, bits, bytes og kontakt med internett – alt som skjer som bruker ikke ser. I eksempelet vårt vil bruker se hvilket ord som blir søkt på, søkeresultater, men ikke selve prosessen bak. Dette kan være vanskelig å føre inn dersom man ikke har slik kunnskap, men i en systemutviklingsprosess vil det være andre teammedlemmer som har denne kunnskapen.

NB:

Det er viktig, og et poeng med analysen, at sekvensen av handlinger, interaksjoner og effekter er synlige i tabellen. Med dette mener vi at rekkefølgen i hver enkelt punkt reflekterer det som skjedde under den faktiske observasjonen. Det vil si at dersom maskinen eller tjenesten bruker lang tid på å respondere, skal dette visualiseres i tabellen. Dette gjøres ved at det som føres i hver kolonne ikke er på samme «rad». Man kan også visualisere tidslinjen på en annen måte, gjennom nummering

Tid

Bruker		Maskinen	
Handling ikke synlig for maskinen	**Handling synlig for maskinen**	**Effekt synlig for brukeren**	**Design rasjonale**
"Hva gjør jeg nå?"			
"Jeg skriver brukernavn og passord og trykker her"	Innloggingsinput gis og trykker på "logg inn"		
		Siden laster	Henter brukerinformasjon gitt som input og sammenligner med registrerte brukere

Min shitty variant av en sekvenstabell over bruken av vippsappen: 🤢 🤮
(M er meg og P er Pappa):

Brukeren		Vipps	
Handling ikke synlig for maskinen	**Handling synlig for maskinen**	**Effekt synlig for bruker**	**Design-rasjonale**
M: Du kan begynne med å åpne appen			
	Brukeren åpner appen	Appen ber brukeren om den personlioge koden tilhørende den registrerte brukren	«Sikkerhetstiltak»
P: Hva var koden? Kan du skrive den jeg husker			

ikke det. M: Det er vel fødselsdat oen din. P: Å, ja!			
	Brukere n skriver koden sin	Appen åpner til hovedsiden	Gir tilgang
M: Du kan prøve å sende penger til meg, for å se hvordan du ville ha gjort det. P: Vel jeg må først trykke på send.			
	Trykker på «Send»	Appen viser en søkebar der appen ber å skrive numeret elle navnet på den vedkommene man ønsker å sende til.	Søkebar
	‘	Appen viser en treff	
	Taster inn numme ret mitt	Appen ber om å skrive den personlige koden om igjen for sikkerhetsgrun ner	«Sikkerhetstilt ak»

	Trykker		Hovedfunksjo

	på kontoen og skriver summe n man ønsker å sende		n
	Trykker på «Send»	Pengene er sendt	
M: Det var det og nå er jeg 1 kr rikere!			

Fra Tobias Rade Evensen 2021V

Brukeren		**Maskinen**	
Handling ikke synlig for maskinen	**Handling synlig for maskinen**	**Effekt synlig for bruker**	**Design-rasjonale**
T: Finn en reise fra Oslo til Håneskrysset i Kristiansand som om du skal kjøpe billett for den, og stopp når du kommer dit at du må skrive inn betalingsinformasjon H: Da går jeg inn på finn reise, for det gir mening			
	Trykk på «Finn reise»	Fanen for «Finn reise» synliggjøres	
		Vindu som spør «vil du fortsatt ha denne fanen som din oppstartsfane?» dukker opp med alternativ ja i grønn og nei i grå.	Dette er første gangen brukeren åpner denne fanen på egenhånd siden oppdateringen, kanskje brukeren vil ta stilling til dette.
H leser høyt. H før ferdig lest: Jeg skjønner ikke helt, men ja.			
	Trykk på «ja»	Vindu forsvinner.	
	Kjapt trykk på «fra»		
		Boksen til «fra» og «til» fyller skjermen.	Fokus
H: Det er fordi jeg ikke gidder, ikke sant. Jeg er her for business. (←om ja-valget) Og jeg skulle finne fram til …			
	Skriver «Oslo»		
		Steder som har med Oslo dukker opp	Søkefunksjon
H: Oslo S, da			

	Trykker på «Oslo S»	Oslo S blir valgt og vist i fra-feltet. søkefeltet tømmes og fokuset til tastaturet flyttes ned til «til»-feltet.	Søkefunksjon, hjelp med utfylling
	Skriver «Håneskrysset»	Håneskrysset dukker opp som øverste alternativ sammen med andre alternativ.	Søkefunksjon
	Trykker på «Håneskrysset	Boksen til «fra» og «til» trekkes sammen igjen. Under dukker det opp en rad med knappene «Reis nå» og «flere alternativer» og en liste med reiser. Reisene den viser går alle fra Oslo bussterminal, og i «fra-boksen» står det fortsatt Oslo S. Oppe i høyre hjørne dukker det opp en grønn firkant hvor det står «1 voksen».	Nå har «fra» og «til» blitt fylt inn, brukeren er klar til å velge reise og hvordan.
H: Det er veldig enkelt, jeg reiser fra og til. Ikke reis nå…			
	Trykker på «Reis nå»	Viser valgalternativene for reisetidspunkt	«Reis nå» er det vanlig å kunne trykke på og endre i andre apper og tidligere utgaver av denne appen.»
T: Finn en reise du liker, og bestill den som om du skal kjøpe den, helt til du må taste inn betalingsinformasjon. H: Først og fremst er jeg ikke voksen, det så jeg nå…			
	Trykk på «1 voksen»	Starter veilederen for å legge inn seg selv i «passasjerer»-funksjonen.	Dette er første gang brukeren åpner «passasjerer»-funksjonen
H leser. H: Ja			
	Trykker på «Den er grei»	Spør om fødselsdatoen til profilen hennes er riktig.	Neste steg i veilederen. Lurt å dobbeltsjekke og vise at fødselsdato blir brukt for å regne ut pasasjertype.
H: Ja, det stemmer det			
	Trykker på «Bekreft»	Spør «Reiser du med noen av disse rabattene?», viser alternativer for rabatter, viser en stor knapp det står «lagre» på.	Neste steg i veilederen.
H leser			
	Trykker på «Studentrabatt», trykker på «lagre»	Sier ifra om at «informasjonen er lagret», viser en «lukk»-knapp	Lukk-knappen tar deg ut av veilederen og tilbake til der du velger reise.
	Trykker på den første reisen	Viser mer info om reisen med kart øverst og detaljer nedover.	
H: Her står det litt informasjon om covid, jeg må bare lese det her.			

	Trykker på ett av infokortene som har en pil ved siden av seg.	Kortet utvider seg nedover og viser mer informasjon og covid-19	Pil ned viser at det er mer info som dukker opp om du trykker, og gråfargen mot det hvite viser at det er en tekstboks
H: Ja, det gir mening. Greit. H ser over info om reisen. H: Det var litt irriterende at den var i veien for den der nede (←om en snakkeboble som informerte om funksjonalitet i en knapp for å endre antall reisende, men som sto i veien for å lese det som sto nederst på informasjonskortet om reisen)			
	Trykker på «Fortsett»	Viser et kort med tittel Enkeltbillett, og to alternativer: refunderbar og ikke refunderbar.	Få et valg om hvilken av de to billettypene brukeren vi ha.
H: Wæ			
H: på grunn av corona	Velger «refunderbar»	Viser en side med tittel «Velg avdeling	
H leser. H: ja?			
	Trykker «lagre og fortsett»	Viser en side med tittel «Velg sete» og mange seter farget enten grått eller grønt. Oppe hjørnet er det et lite kryss som på alle sidene, og nederst er det en stor grå knapp det står bekreft på	Grått = opptatt/ikke mulig å velge, grønt = ledig/mulig å velge. Jeg har ingen anelse hvorfor det ikke er noe uttrykkelig alternativ for å ikke velge sete, alternativene til brukeren har vært veldig tydelige i alle andre sider, og eneste måten å ikke velge sete virker det som er å trykke på krysset oppe i høyre hjørne og bekrefte billettvalget på nytt.
H: Skal vi se. «Vennligst velg sete» (leser). Selvfølgelig velger jeg det setet.			
	Velger et av de grønne fremste setene i 2. etasje	Viser ekstra pris for å reservere sete	Å reservere sete koster ekstra
H: Hvorfor står det +129 kroner, var det noe jeg gikk glipp av tidligere?			
(for å se om de har andre priser?)	Trykker på andre seter	Viser andre priser	Ulike seter har ulik pris
H: Åååå, de er sleipe! Det er dyrt å sitte foran der! Men det kom de ikke med noen informasjon om. Er det billigere å sitte nede, da? (sjekker) nei. Wow, det var litt tungvint. Og de har ikke en gang noen oversikt over prisklassene. Men da har ikke jeg lyst til å reise			

dyrt, så jeg bestiller et sånt sete			
	Velger et billig sete	Viser ny pris	——‖——
	Trykker Bekreft	Viser side med oppsummering	
H: Men det er sikkert enda billigere å bare ikke kunne velge sete, og det så jeg ista, men det var kanskje ikke min skyld. Greit, betal			
	Trykker betal		

Oppmerksomhet og distraksjon

- **Multitasking**: nyttig eller problematisk? Eks. klasserom, stolpekrasj.
    - Evner vi egentlig å multitaske?
- **"Alone together**": Man er mentalt utilgjengelig selv om man er sammen.
    - Eks. samtale stopper opp fordi noen "må bare svare på en melding", "snappe mat"
    - Hva gjør dette med samhandling og relasjoner til dem rundt oss?
        - Sherry Turkle snakker om:
            - Blir vi mer redd for øyekontakt
            - Blir vi mer redd for intimitet
- **Tilgjengelighetskultur:** ny teknologi gjør oss mer tilgjengelige → man kan regulere egen tilgjengelighet, men man er samtidig under press fra samfunnet rundt.
    - **Hva skjer dersom man velger å "stå utenfor"**? **Hvilke alternativ har man?**
    - Sosialt press?
    - Men, digitale teknologier tilrettelegger også for kommunikasjon og er dermed også et nyttig verktøy for mange grupper.
        - Funksjonshemning – bidra mer i samfunnet
        - Eksempelet om den ene i vennegjengen som eventuelt logger av
- **Sosial responsivitet**: det er normalt å si hei til noen som hilser på deg.

- Det er normalt å svare på et direkte spørsmål etc.
- Hvordan kan dette overføres/sammenlignes med normene som styrer kommunikasjon via digitale media?
    - Eks. velger å ikke ta telefonen, venter et par timer med å svare på meldingene osv.

Etikk og lover

Hva er etikk?

Læringsmål: Du skal kjenne til sentrale lover og forskrifter for utvikling av digitale systemer, og kan drøfte etiske problemstillinger.

Etikk er en tradisjonell gren av filosofien som undersøker hva som er rett og hva som er galt, og som diskuterer normer og prinsipper for riktig handling. Dersom du sier at noe er uetisk eller umoralsk, mener du at det er i strid, med visse moralnormer. De tre hovedperspektivene eller teoriene er:

1. regeletikk/pliktetikk–fokuser på handlingene.
2. konsekvensetikk–fokuser på konsekvensene eller resultatene av handling
3. holdningsetikk–fokuser på relasjonen mellom den som handler og den som handlingen retter seg mot.

4 viktige lover for informatikere
1. Personopplysningsloven/GDPR
2. Arbeidsmiljøloven
3. Likestillings- og diskrimineringsloven/WCAG2.0
4. Åndsverkloven

Personopplysningsloven/ GDPR

- Beskytte spesifikke aspekter ved informasjon som kan relateres til fysiske personer
- Forhindre urettmessig innsamling og oppbevaring av personinformasjon
- Forhindre urettmessig bruk av innsamlet personinformasjon
- Sørge for at personinformasjon er korrekt
- Sørge for åpenhet og innsyn
- Retten til å bli glemt at en kunde skal kunne si «jeg vil ikke være kunde mer»

- Sørge for adekvat informasjonssikkerhet rundt personinformasjon
- Definere klar ansvarsfordeling
- Personopplysningsloven regulerer behandling av personopplysninger
- Det er kun tillatt å behandle personopplysninger dersom vilkår for å behandle personopplysninger er oppfylt. Det viktigste vilkåret i studentarbeid er samtykke
- Fomålsbegrensning: personopplysninger skal kun behandles for spesifikke uttrykkelige, angitte og legitime formål. Opplysninger som er samlet inn for et formål kan ikke brukes til andre formål

For informatikere er det særlig viktig å kjenne til hvordan personopplysningsloven regulerer vilkårene for innhenting av personopplysninger, og vilkårene for at personopplysninger lovlig kan behandles i et datasystem.

GDPR

I 2018: ny forordning i EU som kalles «general data protection regulation» GDPR tatt inn i personopplysningsloven

- Det legges større ansvar på at virksomhetene selv ivaretar personvernet når de behandler personopplysninger
- Kravene til utvikling og vedlikehold av internkontroll skjerpes og Forordning har større fokus på informasjonssikkerhet og avvikshåndtering
- Nytt i forordningen er kravet om at informasjonssystemer skal designes med innebygd personvern

Hva er en personopplysning?

- Identitet
- Kontaktinfo
- Finans
- Aktivitet
- Pseudonymisert opplysninger
- Kommunikasjon
- Særlige kategorier

Arbeidsmiljøloven

- Regulerer hvordan arbeidslivet skal foregå, og hvilke rettigheter og plikter arbeidstakere og arbeidsgivere har

- §4.2: krav til tilrettelegging, medvirkning og utvikling som er viktig for informatikere å kjenne til. Her står det at arbeidstakeren (og deres tillitsvalgte) skal informeres om endringer i systemer de bruker i arbeidet, og de skal medvirke ved utforming av det
- Det legges stor vekt på et godt psyko-sosialt arbeidsmiljø

Åndsverkloven

- Regulerer rettighetene omkring åndsverk: det er selve verket man har opphavsrett til, og som gjør at man har både ideelle og økonomiske rettigheter
- Et datamaskinprogram er et åndsverk. Vær midlertidig klar over at åndsverkloven har en særregel som sier at hovedregelen er at opphavsrett til datamaskinprogram som er skapt av en arbeidstaker under utførelsen av oppgaver som omfattes av arbeidsforhold automatisk overføres til arbeidsgiveren.
- Hvis man skal bruke andre sine koder, er det viktig å kilde til dette slik at det ikke blir plagiering

Likestillings- og diskrimineringsloven

- § 18. her står det i lovteksten at «Løsninger for IKT som underbygger virksomhetens alminnelige funksjoner, og som er hoved-løsninger rettet mot eller stilt til rådighet for allmennheten, skal være universelt utformet fra det tidspunktet som er fastsatt i §41. Med IKT menes teknologi og systemer av teknologi som, brukes til å uttrykke, skape, omdanne, utveksle, lagre, mangfoldiggjøre og publisere informasjon, eller som på en annen måte gjør informasjon anvendbar
- Forskrift om universell utforming av IKT-løsninger stiller krav om at nettsider må oppfylle 35 av 61 suksesskriterier i standarden: retningslinjer for tilgjengelig webinnhold WCAG2.0
- Likestillinstillings – og diskrimineringsloven må følge retningslinjene til WCAG. (web content acsesssebility guidelines)

Universell utforming

Det er et begrep innen samfunnsplanlegging, design, arkitektur, tjeneste- og produktutvikling. Handler om å utforme samfunnet slik at så mange som mulig kan delta aktivt uavhengig av funksjonsevne

Hvilke kategorier har vi for funksjonsnedsettelser?

- Perseptuell/sensorisk = hørsel, syn og førlighet
- Motorisk/bevegelse = sitter i rullestol
- Kognitiv = svikt (slag/alderdom)
- Situasjonell = barnevogn eller sykkel

Funksjonshemning er ikke noe som er, det er noe vi gjør. Funksjonshemning kan skapes, forverres eller minimeres. Det er ikke mulig å forstå funksjonshemning bare ved å fokusere på «funksjonshemmedes» behov og forutsetninger - Man må se på den sosiale rammen

Tilgjengelighet og Inkludering

Tilgjengelighet: Handler om å gjøre all informasjon tilgjengelig for alle.

- Dette innebærer at informasjon skal kunne høres, sees, leses og forstås ved tegnspråk. Det optimale her er at all informasjon er tilgjengelig for alle.
- Dersom noe er tilgjengelig for noen er det ikke nødvendigvis tilgjengelig for alle.

Inkludering: Dette begrepet handler i denne konteksten om å inkludere alle mennesker når man utvikler et system.

- Man må tenkte på alle deler av samfunnet og folk. Ofte kan dette gjøres ved å ha med mennesker som er annerledes enn en selv i design og utviklingsprosessen, slik at man får alles perspektiv.
- Dersom noe er inkluderende for alle er det også tilgjengelig for alle.

WCAG

Web content accessibility guidelines

- WCAG er en standard for universell utforming
- Retningslinjer som skal gjøre web – innhold mer tilgjengelig for brukere med ulike typer begrensninger

De fire prinsippene

1. Mulig å oppfatte

Informasjon og brukergrensesnittkomponenter må presenteres for brukere på måter som de kan oppfatte. For eksempel: at en lyd-fil har en transkribert tekst-fil med samme innhold tilgjengelig. Et eksempel på dette er YouTubes teksting av videoer.

2. Mulig å betjene
Det må være mulig å betjene brukergrensesnittkomponenter og navigeringsfunksjoner. Eksempel: mulig å navigere seg med piltaster.

3. Forståelig
Det må være mulig å forstå informasjon og betjening av brukergrensnitt. Et eksempel på dette er Wikipedia og Snl sine muligheter for å undersøke ord man ikke forstår i teksten.

4. Robust
Innholdet må være robust nok til at det skal tolkes på en pålitelig måte av brukeragenter, inkludert kompenserende teknologi. Det betyr å gjøre softwaren så fleksibel og anvendelses vennlig som mulig, ved å gjøre den syntaktiske analysen nøye og ha tydelige navn, roller og verdier for alt – så det er enkelt å bruke softwaren med andre nye assisterende hjelpemidler. For eksempel et hjelpemiddel for å lese en webside høyt. Da er det viktig at hjelpemiddelet forstår softwaren.

Hva er etikk?

Etikk handler blant annet om hvilke moralske prinsipper som påvirker hvordan mennesker tar avgjørelser og velger å leve livene sine. Etikk handler også om hva som er bra for individer og hva som er bra for samfunnet. Balansegangen mellom innovasjon/teknologisk utvikling og hvordan dette påvirker menneskeheten, er et etisk dilemma som stadig debatteres – hvilke valg ligger til grunn og hvilke konsekvenser har de?
- Teknologien selv er et verktøy og kan derfor ikke pålegges moralske eller etiske kvaliteter – ligger ansvaret da på oss som informatikere? Dette har dere diskutert i oblig rundt selvkjørte biler

Etisk refleksivitet

- Evne til å se betydningen av din egen rolle i utviklingsprosessen
- de verdier og forståelser du bringer med deg i utviklingsprosessen

    - hvordan du samhandler med andre i prosessen
    - de teoretiske og metodiske perspektivene du bruker
- Evne til å diskutere og kritisere din egen posisjon fra et mer distansert perspektiv
- Refleksivitet handler om å se seg selv utenfra - sin sosiale kontekst.

Personvern

Hva er personvern?

Retten til privatliv og retten til å bestemme over egne personopplysninger

Dataportabilitet

Styrke din kontroll over egne personopplysninger. Retten til dataportabilitet innebærer:

- Få utlevert dine egne personopplysninger
- Du kan flytte dine persondata
- Virksomheter kan ikke ta penger/gebyr for dette

Personopplysningsloven: 6 sentrale definisjoner

Personopplysning → identifiserer en person
Behandling → operasjon på/med data
Register → lagring av data
Behandlingsansvarlig → bestemmer formål (org. el enkeltperson)
Databehandler → behandler data på vegne av b.ansvarlig (org. el enkeltperson)
Samtykke → **FRIVILLIG; SPESIFIKT; INFORMERT**

Særlige personopplysninger

- rasemessig eller etnisk opprinnelse,
- politisk oppfatning,
- religion,
- Filosofisk overbevisning eller
- fagforeningsmedlemskap, samt behandling av
- genetiske opplysninger og biometriske opplysninger med det formål å entydig identifisere en fysisk person,
- helseopplysninger eller
- opplysninger om en fysisk persons seksuelle forhold eller seksuelle orientering,

Disse er det FORBUDT å BEHANDLE!!
Med mindre::
Punkt 1 (forbud mot å behandle) får ikke anvendelse dersom:

- Det foreligger samtykke
- Behandlingen er nødvendig for at den behandlingsansvarlige eller den registrerte skal kunne oppfylle sine forpliktelser og utøve sine særlige rettig heter på området arbeidsrett, trygderett og sosialrett
- Behandlingen er nødvendig for å verne den registrertes eller en annen fysisk persons vitale interesser **dersom den registrerte fysisk eller juridisk ikke er i stand til å gi samtykke.**
- **Behandlingen utføres av en stiftelse, sammenslutning eller et annet ideelt organ hvis mål er av politisk, religiøs eller fagforeningsmessig art, som ledd i organets berettigede aktiviteter**
- Behandlingen gjelder personopplysninger som **det er åpenbart at den registrerte har offentliggjort.** (person A er prest)
- Behandlingen er nødvendig for å fastsette, gjøre gjeldende eller forsvare rettskrav eller når domstolene handler innenfor rammen av sin domsmyndighet.
- Behandlingen er nødvendig **av hensyn til viktige allmenne interesser,** (person B

skal jobbe et sted som krever politiattest)
- Behandlingen er nødvendig i forbindelse med forebyggende **medisin eller arbeidsmedisin** (person C har diabetes)
- Behandlingen er nødvendig av **allmenne folkehelsehensyn**,
- Behandlingen er nødvendig for **arkivformål eller statistiske formål i allmennhetens interesse**

Personvernerklæring:

Hvordan nettstedet behandler personopplysninger. Husk krav fra GDPR.

Inneholder blant annet:

1. Kontaktinformasjon
2. Hva slags informasjon samler vi inn?
3. Formålet med behandlingen av personvernopplysningene?
4. Samtykke
5. Hvor lenge lagres opplysningene?
6. Hvem kan informasjonen deles med?
7. Hvor oppbevares opplysningene?
8. Dine rettigheter i forhold til innsyn, sletting og eventuelt flytting

11. Klage

Vilkår for bruk

Eksplisere retningslinjer for hvordan bruker kan bruke tjenesten. Må ikke være med, men bør. Eks: dersom noen oppfører seg ufint på nettsiden kan de ikke kastes ut før det ufine bryter vilkårene.

- Gjøre det tydelig, forklare, utrede

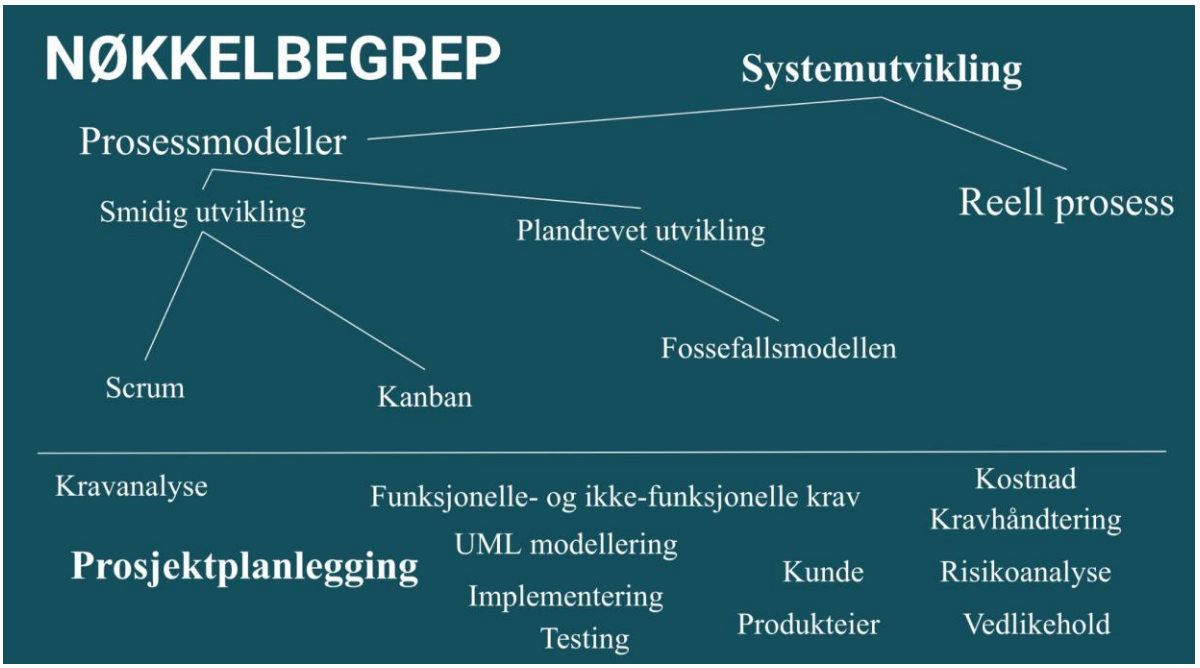
Tilgjengelighetserklæring

Redegjør for hvordan nettstedet oppfyller kravene (35 av WCAG) til universell utforming.

MODUL B

Hva er systemutvikling?

- Læren om utvikling og forvaltning av programvaresystemer av høy kvalitet innen gitte tids-og kostnadsrammer.
- Viktige kvalitetsegenskaper er funksjonell egenhet, effektivitet, pålitelighet, brukskvalitet, kapabilitet, vedlikeholdbarhet, sikkerhet og overførbarhet
- Software engineering inkluderer også å utvikle og evaluere arbeidsmåter, metoder og verktøy som støtter slike aktiviteter



Systemutviklingsprosess

Er de aktivitetene som utføres for å utvikle et IT-system. Aktivitetene varierer, men vil alltid ha elementer av:

- Spesifisering av kravene → hva skal systemet gjøre
- Design av systemet → datamodel
- Implementering av koden → programmering
- Validering av at systemet gjør det kunden ønsker

- Endringer av systemet i forhold til nye og endrede krav hos kunden

Systemutviklingsprosessen vil påvirke kvaliteten både på prosjektet selv og systemet som utvikles. Måten man jobber på påvirker også arbeidsmiljøet som i igjen påvirker prosjekt- og produktkvaliteten

Hvilke hovedaktiviteter inngår i en systemutviklingsprosess?

Planlegging	hva som skal lages og innenfor hvilke rammer/krav.
Kravinnsamling	
Kravanalyse	
Design	design og programmering.
Programmering	
Testing	validerer at systemet er det kunden vil ha.
Konfigurasjonsstyring	systemet modifiseres etter kunden og markedets krav/behov.
Versjonshåndtering	

Hvilke faser inngår i en systemutviklingsprosess?

System-spesifisering:	hva som skal lages og innenfor hvilke rammer/krav.
Design og implementering:	design og programmering.
System-validering:	validerer at systemet er det kunden vil ha.
System-evolusjon:	systemet modifiseres etter kunden og markedets krav/behov.

Eksempler på roller

- Utvikler
- Vedlikeholder

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

Hvilke typiske aktiviteter inngår i en systemutviklingsprosess?

Planlegging, kravinnsamling, kravanalyse, design, programmering/koding, testing, konfigurasjonsstyring og versjonshåndtering.

Konfigurasjonsstyring omhandler styringen av innhold, endringer og status på delt informasjon i et prosjekt.

Versjonshåndtering er programvare som kan holde orden på de forskjellige versjonene av en eller flere datafiler. Når en fil oppdateres eller forandres, slettes ikke den gamle versjonen, men blir lagret i en database som inneholder tidligere versjoner av filene. Gamle versjoner kan hentes frem, og det er som oftest mulig å vise forskjeller mellom de forskjellige versjonene og lage utviklingsgrener fra disse versjonene.

Usikkerheter i et prosjekt

- Tid
- Kostnader
- Levert funksjon - Kvalitet

Vedlikehold

Vedlikehold av IT-systemer betyr ikke å bevare originalversjonen mest mulig slik som for bil og hus.

- Vedlikehold er alle endringer utført i systemet etter at det er satt i drift
- Utgjør 50-90 av kostnadene i levetiden til et system

Hva er prosessmodellen?

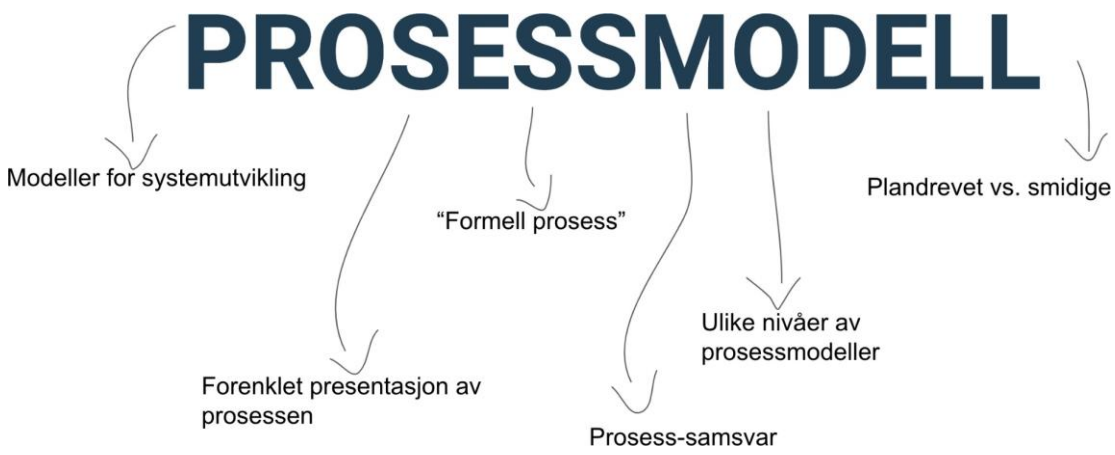
En modell er en abstraksjon, en forenkling, som skal gi deg et slags «veikart» som kan brukes som utgangspunkt for å planlegge ulike aspekter ved et prosjekt.

- Modeller for systemutviklingsprosesser
- Ulike nivåer av prosessmodeller

- Forenklet presentasjon av prosessen
- OBS: prosessmodeller er ulike modeller for å få gjort alt dette effektivt og oversiktlig.
- man må lage en prosessmodell før man starter en prosess.
- Denne modellen bør inkludere hvilke metoder og teknikker man skal bruke, hvordan man skal jobbe sammen, hvilke aktiviteter og hva de skal føre til (hvor skal man), Roller til de som er med .

Prosessmodeller versus den reelle prosessen?

En modell er en abstraksjon, en forenkling, som skal gi deg et slags «veikart» som kan brukes som utgangspunkt for å planlegge ulike aspekter ved et prosjekt.



Prosess-samsvar:

- Hvordan man faktisk jobber og hvordan dette samsvarer med modellen
- I hvilken grad følger man modellen?
- Dette er viktig fordi:
  - Hvis man mislykkes i et prosjekt og man ikke vet om man fulgte modellen, så vet men heller ikke om man må endre på modellen
  - Men vet ikke om det er modellen som var årsaken til at prosjektet gikk dårlig, og da kan det hende man havner i akkurat samme fella igjen
  - Følger man modellen og det gikk dårlig – så burde man kanskje endre på modellen?

Formell prosess

Beskrivelse med ulike typer diagrammer

- En skisse

- Prosessbeskrivelse

Ulike nivåer

- Generelle prosessmodeller
    - Scrum
    - Fossefall
    - Kanban
- Firmaspesifikke prosessmodeller
    - Ulike firma har ulike prosessmodeller
- Prosjekt/gruppe spesifikke prosessmodeller
    - Er det store selskap
    - Er det ulik fra grupper og typer prosjekter
    - Olje vs helse
- Systemutvklingsprosessen
    - Prosess-samsvar
        - Hvordan man faktisk jobber og hvordan dette samsvarer med modellen
        - I hvilken grad følger man modellen
        - Dette er viktig fordi:
            - Hvis man mislykkes i et prosjekt og man ikke vet om man fulgte modellen, så vet men heller ikke om man må endre på modellen
            - Man vet ikke om det er modellen som var årsaken til at prosjektet gikk dårlig
            - Og da kan det hende man havner i akkurat samme fella igjen
            - Følger man modellen og det gikk dårlig – så burde man kanskje endre på modellen

Hvilke fundamentale aktiviteter utføres i systemutvikling utover programmering?

Programmering er viktig, men

- Programanalyse og planlegging
- Kravarbeid (kravinnsamling, kravanalyse, kravspesifikasjon)
- Utforming og design
- Testing, validering og dokumentering

- Konfigurasjonsstyring og versjonshåndtering
- Innføring, vedlikehold, videreutvikling

Hvilke aspekter ved systemutvikling tilsier at det er en ingeniørdisiplin?

Fokus på systematiske metoder.

- Planlegging og forutsigbarhet
- Oppdeling og strukturering av problemer i mindre komplekse bestanddeler - Modularitet og gjenbruk
- Abstraksjon og modellering
- Dokumentert prosess og systematisk kvalitetssikring

Hva er en systemutviklingsprosess?
Systemutviklingsprosessen er de aktivitetene som inngår i å utvikle et IT-system. Hvordan aktivitetene organiseres, hvor mye man gjør av hva og utdeling av roller. (En prosessmodell er et rammeverk for hvordan man skal komme fra A til Å i utviklingen av et system).

Hvorfor er det viktig å ha en god systemutviklingsprosess?
Systemutviklingsprosessen påvirker resultatet på ulike nivåer:

- Prosjektstyring
- Arbeidsmiljø
    - Mangel på motivasjon
- Type og mengde kommunikasjon utviklere har med kunder og med hverandre
    - Misforståelse
- Estimering av tidsbruk
    - Svarer ikke på kravene
- Hvor godt man tar høyder for endringer

Det er ofte stor prisvariasjon i anbud på IT-prosjekter sammenlignet med andre bransjer som bygg- og anleggsbransjen. Hva kan denne prisvariasjon skyldes?

- Utvikling av IT-systemer → ny disiplin
    - Manglende statistikk for estimering av ressursbruk
    - Estimering kan være svært vanskelig
- Utvikling → hva skal lages?

    - Kunden må forstå hva de vil ha/utviklere må forstå hva de skal lage
- Kvalitet av ulike oppfatninger av hva begrepet innebærer
    - Avhengig av kontekst og personlige oppfatninger
- Ulike utviklingsprosesser / måter å jobbe på
    - Stor variasjon i ressurser og kostnader (eks: testing)

Hvorfor er det hensiktsmessig å definere prosessmodeller på ulike nivåer (generelle, bedriftsnivå, avdelings/prosjektnivå)?

Den generelle prosessmodellen er en abstrakt og forenklet formell representasjon av en prosess. Man kan for eksempel tilpasse aktiviteter, roller, ansvarsforhold og frekvens på rapporter. Å definere ulike nivåer er hensiktsmessig fordi:

- Man er i større stand til å tilpasse seg
- Man skaper bedre arbeidsflyt ved å ta hensyn til den faktiske arbeidspraksisen
- Type prosessmodell bør påvirkes av type prosjekt, kulturen i bedriften, og kompetanse
- Lokal praksis må tas i betraktning
    - Bedrifter er ulike: kompetanser, kulturen og prosjekter varierer
- Graden av generaliser barhet påvirkes

Plandreven utvikling

Hva er fossefallsmodellen?

Fossefallsmodellen er en plandrevet prosessmodell

- Består av separate faser
- Vanskelig å tilpasse endringene i krav underveis
- Utviklingen styres av forhåndsdefinerte planer
- Består av 5 veldefinerte faser:
    - Kravspesifisering
    - System- og softwaredesign
    - Implementering og enhetstesting og Integrasjon- og systemtesting
    - installasjon og vedlikehold

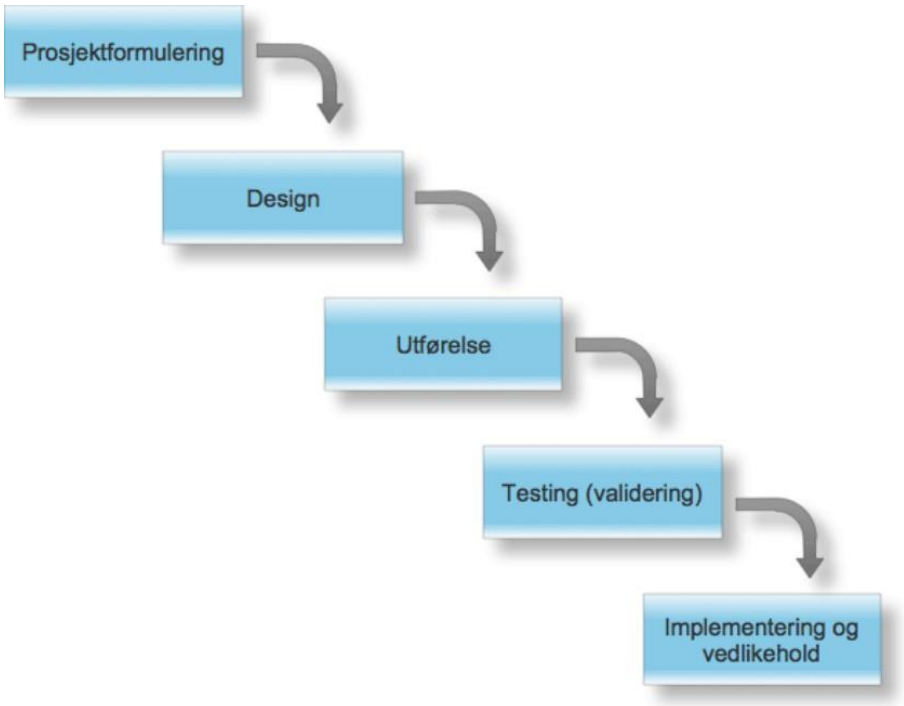
Om en fase er ferdig, kan man ikke gå tilbake til den. Derav fossefallsmodellen

Plandrevne prosessmodeller er smart å bruke i prosesser hvor man ikke har mulighet å gå tilbake i prosessen som for eksempel i et byggeprosjekt. Man har ikke muligheten til å gå tilbake hvis man allerede har begynt å borre hull til tunellen. Dette krever nøye planlegging i

starten slik at man unngår slike tilfeller. Det blir katastrofalt hvis man havner i en slik situasjon. Her har man da ett møte med kunden på starten og planlegger på vegne av dem videre.

Foreslå et utviklingsprosjekt der det kan være gunstig å benytte fossefallmodellen.
Det kan (men ikke alltid) være gunstig å benytte fossefallsmodellen hvis:

- Systemet som skal utvikles er stort (krever koordinering og god oversikt)
- Det er store geografiske avstander mellom utviklere
    - Fordi det krever mindre kommunikasjon i plandreven utvikling
- Systemet som skal utvikles er velkjent (stabile/velkjent krav) - Systemet er et kritisk sanntidssystem: sikkerhet er viktig



Smidig utvikling

Inkrement og iterasjon i systemutvikling

Et inkrement er et tillegg i funksjonaliteten – et aspekt ved systemet. En iterasjon er en syklus i utviklingen – er aspekt ved prosessen.

- Et nytt inkrement utvikles gjennom en ny iterasjon

- En ny iterasjon kan også forbedre kvaliteten på samme funksjonalitet, dvs. Man lager ikke noe nytt inkrement, men bare forbedrer det eksisterende systemet.

Hva er forskjellen på smidig utvikling og plandrevet utvikling?
Hovedforskjellen på smidig utvikling og plandrevet utvikling er dens evne til å håndtere kravendringer underveis. Smidig utvikling er en utviklingsmetode der man prioriterer å håndtere endringer fremfor slavisk å følge en plan. Smidig vektlegger inkrementell utvikling og korte iterasjoner.

I plandrevne prosesser leveres systemet først når det er helt ferdig, og kravhåndtering forgår hovedsakelig i første fase av prosessen.

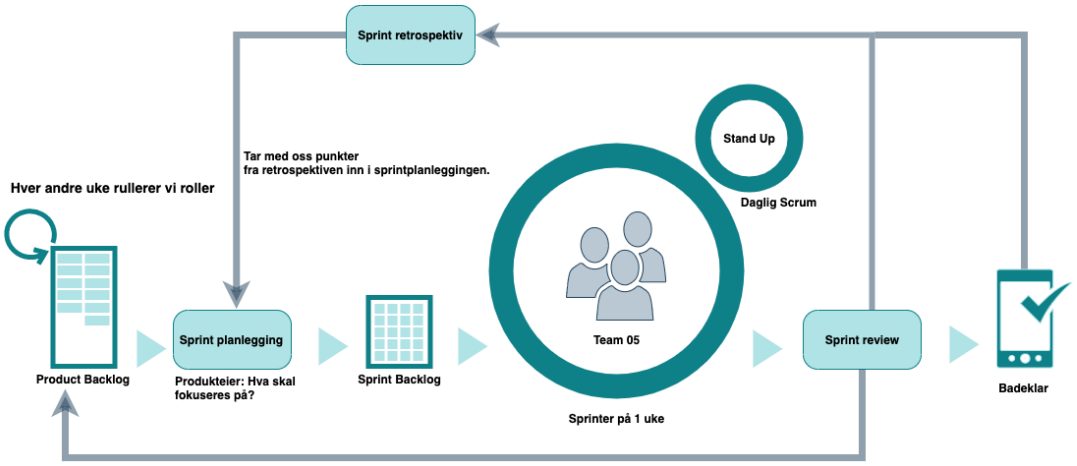
Scrum

Scrum er den smidige utviklingsmetodikken som benyttes mest i dag. Baserer seg på tidsbokser også kalt sprinter, som ofte har en varighet på 2-4 uker. Har definert oppstarts- og avslutningstidspunkt. Denne prosessmodellen består av tre faser:

1. Planleggingsfasen
   Overordnede mål etableres/programvarearkitektur designet
2. Gjennomføringsfasen
   Serie med iterasjoner (sprinter) hvor hver sprint leverer et inkrement av systemet
3. Avslutningsfasen
   Dokumentasjon og manualer ferdigstilles. Nødvendig dokumentasjon som hjelpefunksjoner og brukermanualer fullføres, og man oppsummerer hva man har lært i prosjektet.

Det er vanlig med «daily stand-ups» der alle på teamet kort besvarer følgende spørsmål:

- Hva har jeg gjort siden i går?
- Hva skal jeg gjøre i dag?
- Hvilke eventuelle hindringer har jeg?



Fordeler ved Scrum for ansatte

- 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 forløpende og kan gi tilbakemelding på hvordan deler av systemet fungerer

- Tillit mellom kunder og utviklere kan etableres tidlig

- Kryss-funksjonelle team sikrer framtid og reduserer risiko.

Hva er sprint, og hvilke ulike faser består den av?

En sprint er en iterasjon i gjennomføringsfasen som vanligvis er på 2-4 uker og består av følgende faser:

Assess: Etablere nye, eller endre eksisterende oppgaver (krav, brukerhistorier) i backloggen

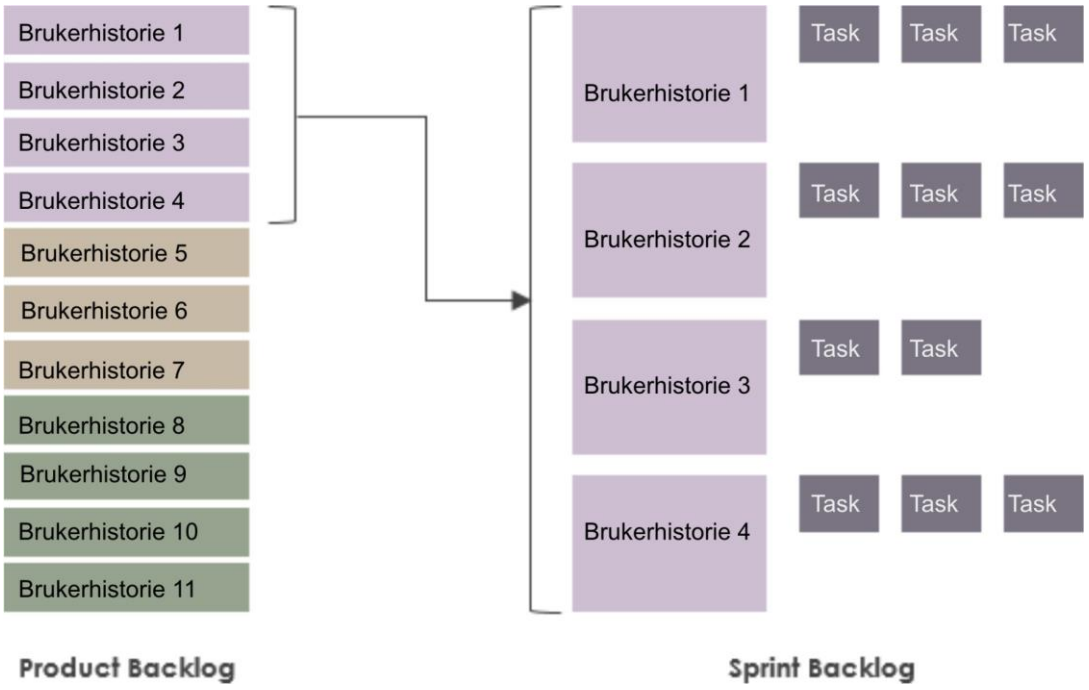
Select: Velge oppgaver til sprint. Her velger man de mest prioriterte oppgavene i product backlog, ut fra hvor mye teamet har av kapasitet per sprint

Implement: Her skal Scrum Master beskytte teamet fra forstyrrelser og teamet skal implementere oppgavene som har blitt valgt for sprinten.

Evaluation: Evaluere hvordan sprinten gikk. Ble vi ferdige med alle målene vi satte oss? Var det dette kunden ville ha?

Hva er en backlog?

En prioritert liste med arbeidsoppgaver som produkteieren (product owner) har ansvar for å vedlikeholde. Består ofte av brukerhistorier. Brukes til å estimere hva som kan gjennomføres i neste sprint.



Hvilke roller er vanlig i scrum?

Scrum utføres av et team som består av følgende roller:

Scrum Master: ansvarlig for å holde daily-standups og å beskytte utviklingsteamet mot forstyrrelser fra blant annet kunde. I implementeringsfasen er det derfor Scrum Master som hovedsakelig skal ta seg av all kundekontakt.

Produkteier: Eier av visjon og representant for interessentene. Ansvarlig for product backlog og for hvilke oppgaver som skal prioriteres. Produkteier skal aldri være Scrum Master.

Scrum team: Et tverrfaglig team som skal utvikle inkrementet i hver sprint. Alle utover de to øvrige rollene er likestilte, og enhver har ansvar for å bidra med sin kompetanse under sprint.

Large scale scrum

- Bruk av Scrum i store prosjekter med mange team
- Bruk av Scrum i store prosjekter er krevende – innebærer store endringer for å bli ordentlig smidige

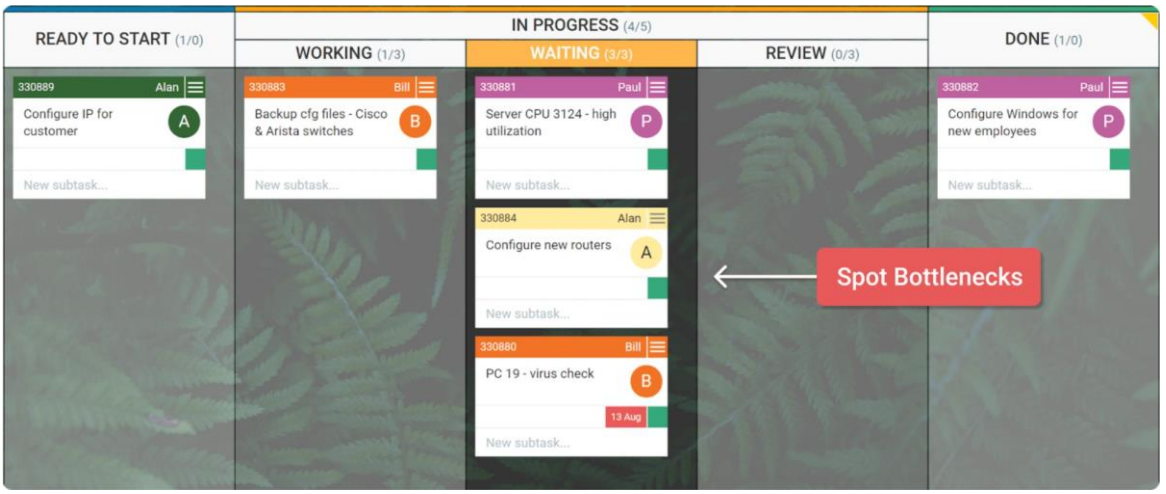
Kanban

Kanban er en smidig utviklingsprosess som ble utviklet av Toyota for å finne et system som forbedrer og opprettholder et høyt nivå av produksjon. Man gjør en oppgave (Work in progress) til den er ferdig uten å nødvendigvis sette en tid for dette, slik at man sikrer god utførelse. Når en oppgave er utført, velger man en ny som er viktigst der og da. Oppstår det et kritisk problem underveis, stopper man produksjonen og fokuserer all energi på å løse problemer før man fortsetter normal produksjon. Dette skal hindre at problemet hoper seg opp og at de blir løst.

- Ved å ikke gjøre flere oppgaver til enhver tid enn teamet tåler, skal det sikre prosjektet godt fly.
- Fokus på å få oppgaver raskt utført = antall brukerhistorier implementert per tidsenhet
- Begrense antall arbeidspakker som det jobbes med i parallell for å hindre flaskehalser.
- Mindre fokus på estimering av tid og kostbare.

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 i veldefinerte sprinter ofte ikke gi mye mening.
- Gunstig der det er svært vanskelig å estimere oppgavene



Hva er forskjellen på Scrum og Kanban?

Kanban baserer seg på tidsflyt (jobber med en oppgave til den er ferdig). Kanban har mindre fokus på estimering enn Scrum, og fokuserer mer på gjennomstrømming av arbeidspakker (WIP = work in progress). Scrum baserer seg på tidsbokser (velger oppgaver til en sprint som varer et visst antall uker) med klart definerte faser og roller. Tidsbokser (Scrum) - Velg noen prioriterte oppgaver og jobb med dem i faste tidsintervaller med definerte oppstart og avslutningsaktiviteter (tidsbokser). Flyt av oppgaver (Kanban) - Definerer et sett med oppgaver eller «features» som skal lages og leveres så snart som mulig man er ferdig. Oppgaver skal flyte uten avbrudd gjennom de nødvendige aktivitetene til de er ferdig (oppgaveboksing).

Kundeinvolvering

Fordeler og ulemper med kundeinvolvering

Fordeler:

- Raske tilbakemeldinger
- Involverer en person med god domeneforståelse
- Sørger for at systemet opprettholder brukerens behov

Ulemper:

- Krever mye tid og ressurser av kunde
- Krever at kunde er tilgjengelig

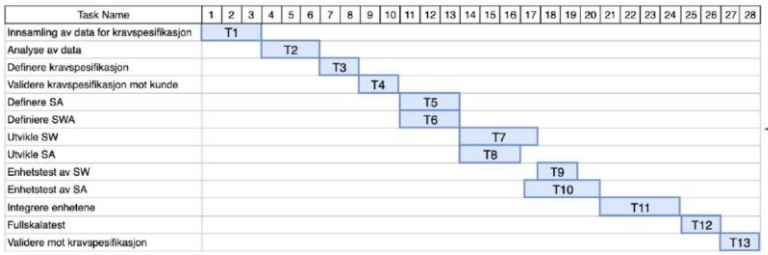
Prosjektplanlegging

Oversikt over alle oppgaver, hvor lang tid de tar, og avhengigheter til hverandre → lager en visualisering i form av Bar chart/søylediagram

Task	Name	Effort	Duration	Dependencies
T1	Innsamling av data for kravspes.	10	15	T0
T2	Analyse av data	10	15	T1
T3	Definere kravspesifikasjon	8	10	T2
T4	Validere kravspes. mot kunde	10	10	T3
T5	Definere systemarkitektur	10	15	T4
T6	Definere software	10	15	T4
T7	Utvikle software	20	25	T5
T8	Utvikle sys.ark	10	20	T6
T9	Enhetstest av SW	8	10	T7
T10	Enhetstest av SA	10	15	T8
T11	Integrere enhetene	10	20	T9, T10
T12	Fullskalatest	8	10	T11
T13	Validere mot kravspes.	8	10	T12

Prosjektplanlegging

Hvilke aktiviteter avhenger av hverandre?
Hvor lang tid tar de?



← Bar chart/ Søylediagram

Smidige praksiser og teamarbeid

Teamarbeid

- Et team som fungerer et samlet og har en god teamfølelse
- Teamets mål er viktigere enn egne mål
- Kommunikasjon er en nøkkelfaktor for å lykkes
- Fleksibilitet i teamsammensetning er ofte begrenset av hvem som er tilgjengelig

Hva er spesielt med software team?
- Hyppige endringer gjør det vanskelig å planlegge: Komplekse sosiale og tekniske systemet
- Får etablert teorier om systemutvikling

Smidig tilnærming

Smidige metoder i programvareutvikling er en iterativ tilnærming der programvaren blir utviklet og levert til kundene som tillegg (increments)

- Til forskjell for plandrevet tilnærming, er funksjonaliteten til tilleggene (increments) ikke planlagt på forhånd men avgjøres under utviklingen
- Hva som tas med i en iterasjon avhenger av utviklingen i prosjektet og kundens prioriteringer
- Kundens prioriteringer og krav endrer seg. Derfor kan det være fornuftig å ha en fleksibel plan som kan ta høyde for disse endringene

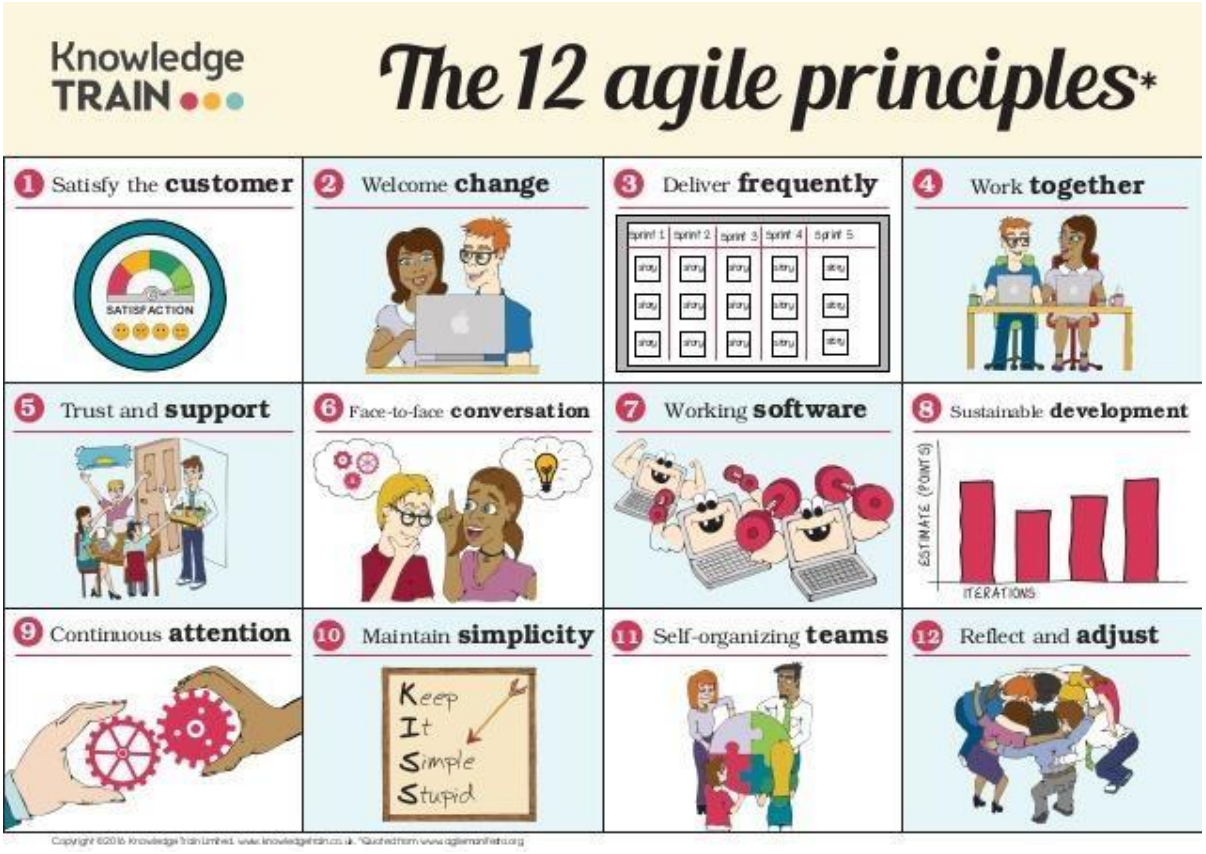
De tre mest brukte smidige teknikkene (Scrum)

Aktiviteter/sermonier

- Daily standup
    - Et kort, daglig møte. Gjennomføres til samme tid, på samme sted hver dag. Alle skal (helst) stå oppreist
- Sprint/iteration planning
    - Et møte hvor hovedmålet er å planlegge en sprint, se på backlogg og sprint-backloggen
- Retrospectives (retrospektive møter)
    - Et møte hvor fokus er på hvordan kan bruke lærdom fra denne sprinten i neste sprint.
- Sprint review
    - Inkrement eller ferdig produkt leveres

12 prinsippene

1. Tilfredsstille kunden ved å levere programvare tidlig og kontinuerlig
2. Ønske kravendringer velkommen, til og med sent i prosessen.
3. Levere fungerende programvare hyppig
4. Daglig samarbeid mellom produktets eier/forretningsfolk og utviklere gjennom hele prosjektet
5. Bygg prosjektet på motiverte individer. Gi de godt arbeidsmiljø, støtt og stol på dem.
6. Ansikt til ansikt er mest effektivt.
7. Fungerende programvare bestemmer progresjon.
8. Bærekraftig utvikling i form av kontinuerlig arbeidsflyt.
9. Fokus på (excellence and) godt design.
10. Enkelhet - kunsten å maximere ikke-gjort arbeid - er essensielt.
11. Selvorganiserte team lager den beste arkitekturen, krav og design.
12. Regelmessig refleksjon over effektivitet og handlinger - for å bli bedre.



Krav

Hva er krav?

Vi har ulike typer krav: funksjonelle krav og ikke-funksjonelle krav (produktkrav, organisatoriske krav og eksterne krav)

Hvorfor?

Vi utvikler IT-systemer for å løse et problem/identifisere og utnytte muligheter. Kravene forteller oss noe om hva som skal lages. Kostbart å rette feil i kravene etter systemleveranse. Utilstrekkelig kravhåndtering à viktigste årsak for problemer i systemutviklingsprosjekter. Kravendringer vil alltid forekomme.

Er også viktig for:

- Kontrakt oppdragsgiver → leverandør
- Planlegging og oppfølging
- Arkitektur, design og test
- Å støtte videreutvikling og vedlikehold

Kravene bør være:

- Forståelige: alle interessenter/stakeholders må kunne forstå kravspesifikasjonen
- Testbare: vi må kunne avgjøre om det ferdige systemet gjør det det skal
- Sporbare: vi må vite hvilken del av koden som skal endres når det kommer nye krav

Brukerkrav

høynivå beskrivelse av systemets tjenester eller begrensninger de må operere under

- Krav uttrykt i naturlig språk eller diagrammer som viser ønskede tjenester (funksjoner) til systemet og føring som gjelder (kvalitetsegenskaper)
- Skal forstås greit av kunden

Systemkrav

Detaljert, formell beskrivelse av programvarens funksjoner, tjenester eller operasjonelle begrensninger

- Strukturert, detaljert beskrivelse av systemet funksjoner og føringer som gjelder (kvalitetsegenskaper)
- Definere hva som skal implementeres

- Utgangspunkt for kontrakt mellom kunde (oppdragsgiver) og utviklerorganisasjon

Funksjonelle krav

Funksjonelle krav beskriver hva systemet skal gjøre (men kan også beskrive hva systemet ikke skal gjøre):

- Hvilke tjenester/funksjoner skal systemet tilbyr?
- Hvordan skal det reagere på ulike typer input?
- Avhenger av hvilket system som skal utvikles, systemet brukere og de som er ansvarlig for å beskrive kravene
- Variere fra generelle krav til hva systemet skal gjøre, til mer spesifikke krav som reflekterer arbeidsmetoder eller en organisasjons allerede eksisterende system
  Funksjonelle krav skrives gjerne på formen «systemet skal... /systemet bør (nice to have)...»

Eksempler på funksjonelle krav fra Ruters billettapp:

- Systemet skal kunne vise en oversikt over en brukers betalte billetter

- Systemet skal gi en beskjed når det er under 24 timer til en billett løper ut

- Systemet skal tilby funksjonalitet for valg av billettype

- Systemet bør tilby hurtigkjøp av tidligere valgte billetter

- Systemet skal fjerne et valgt bankkort hvis brukere taster feil pin 3 ganger

Ikke-Funksjonelle krav

Ikke-funksjonelle krav definerer hvordan systemet skal innfri de funksjonelle kravene
- Sier noe om hvilke kvalitetsattributter systemet skal ha
- Sier noe om egenskaperàhvordan skal systemet oppføre seg?
- Må være målbar
- Kan også beskrive om kvalitetsønsker – det vil si krav til systemet som ikke handler om funksjonaliter

Eksempler på ikke-funksjonelle krav til Ruters billettapp:

- En ny kunde skal kunne betale for en billett på under tre minutter - Systemet skal kunne håndtere 10.000 brukere samtidig

- Systemet skal utvikles ved hjelp av smidige utviklingsmetoder

Ikke-funksjonelle krav kan være knyttet til systemegenskaper som pålitelighet, effektivitet og brukskvalitet. Man deler også de ikke-funksjonelle kravene inn i:
Produktkrav

Beskriver brukskvalitet/brukervennlighet, ytelse og effektivitet samt l lagringsplass, pålitelighet og lagring av data

- Brukskvalitet/brukervennlighet
- Varierer for ulike brukergrupper
- Universell utforming
- Ytelse
- Kapasitet
- Antall samtidige brukere
- Responstid

Organisasjonskrav

Omhandler gjerne kostnader og ressurser, leveransetidspunkt, prosess- og utviklingsmodeller, programmeringsspråk, verktøy og komponenter samt generelle standarder og regler

Eksterne krav

Andre krav knyttet til for eksempel personvern, sikkerhet eller etiske problemstillinger

Kravhåndtering

Hensikten med å utvikle eller forbedre et IT-system: Å løse nye utfordringer og utnytte potensialer

- Kravhåndtering er prosessen å identifisere, analysere og spesifisere kravene til det nye eller forbedre systemet.

Kravhåndteringsprosessen

- **Forstudie/målanalyse**
  - Kost/nytte-analyser, risikoanalyser, gevinstrealisering
  - Analyserer nå situasjonen, ønsket situasjon og mulig tiltak for å oppnå ønsket situasjon
- **Kravinnsamling og kravanalyse**
  - Hva ønsker interessentene seg? Hva har de behov for? - Prioritering av kravene
- **Kravspesifisering**
  - Utgangspunktet for anbud og kontrakt (mellom kunde og leverandør)
  - Utgangspunkt for design, implementasjon og testing
  - Utgangspunkt for estimater (tid og kostnader)
- **Validering av kravspesifikasjonen**
  - Uttrykker kravspesifikasjonen det kunden og interessentene faktisk ønsker seg?
- **Håndtering av kravendringer**
  - Brukere oppdager nye behov etter at systemet tas i bruk
  - Formell prosess for vurdering og eventuell gjennomføring av foreslåtte endringer
    - Hvilken endring foreslås? Hvem foreslår endringen?
    - Vurdering av foreslått ending: konsekvensanalyse
    - Beslutning om endringen skal implementeres
    - Hvem skal følge opp endringene?

- **Kravspesifikasjon som grunnlag for testing**
    - Kravspesifikasjon sier noe om systemets funksjonalitet og oppførsel - Kan brukes som grunnlag for testinstanser
    - I smidige systemutvikling er det færre detaljer i krevspesefikasjonen

Kravspesifikasjon

Basis for anbud

- Ulike tilbydere vil kunne tilby ulike måter å løse kundens behov på
- Basis for kontrakt
- Basis for design og implementasjon av systemet

Hva er kravspesifikasjon?

Et dokument som spesifiserer kravene til et system:

- Spesifisere system- og brukerkrav
- Definerer hva som skal lages – ikke hvordan oppgaven skal løses
- Er ofte en del av kontrakten for systemutviklingsprosjektet
- Informasjonen i dokumentet vil avhenge av type system og utviklingsprosjekt - Finnes ulike standarder for å skrive kravspesifikasjon
- Viktig at kravene som beskrives her ikke er tvetydige da det kan føre til at kunde og utviklere tolker dem forskjelligàkonflikt

Hvorfor er det nødvendig å lage en kravspesifikasjon?

For å lage et system som møter brukerens krav og behov. En kravspesifikasjon er også:

- Basis for anbud
    - Her vil det være rom for fortolkninger
    - Ulike tilbydere kan ha ulike måter å løse kundens behov på
- Basis for kontrakt/design og implementasjon av systemet

En god kravspesifikasjon …

- Skaper felles forståelse av systemet

- Skaper enighet om hva som skal leveres
- Er grunnlag for kontrakt som viser hva leverandøren og kunde blir enige om
- Forhindre eventuelle konflikter som skal oppstå på bakgrunn av uklare forventninger

Brukerhistorie

En brukerhistorie er en kort beskrivelse av en bruker i en brukerkontekst, men hensikt å klargjøre kravene til et system

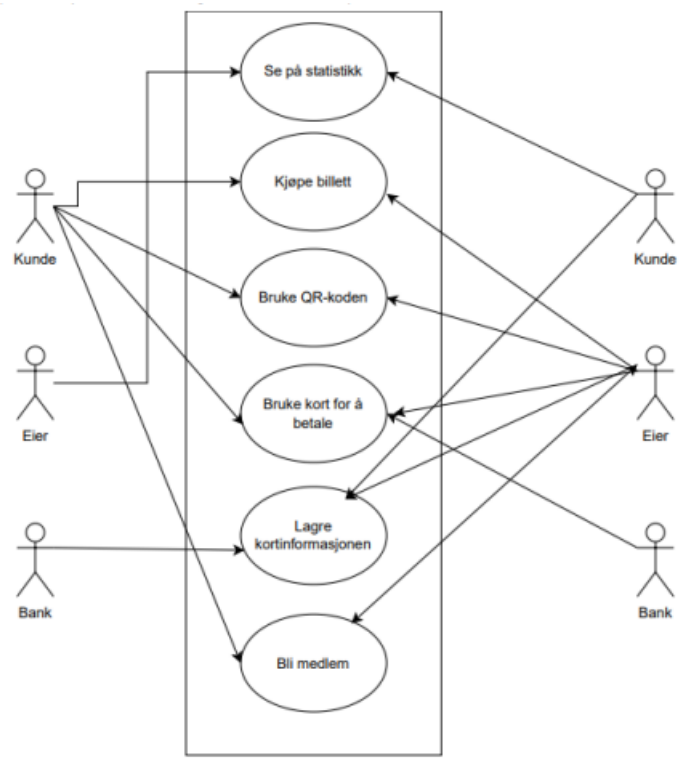
- Brukerhistorier beskriver hva brukeren ønsker å få ut av systemet
- Består av ulike elementer: brukerens rolle, ønsket funksjon og nytteverdi av funksjonen
- Som [rolle] ønsker jeg [funksjon] for å oppnå [nytteverdi]

Nevn noen fordeler ved å bruke denne teknikken til å beskrive krav
Enkelhet og kommuniser kontekst der systemet skal tas i bruk, og hva brukeren faktisk har behov for
- Man forstår raskt hvorfor det er nødvendig å implementere funksjoner
- Det er lettere å se hvem kravet er tiltenkt
- Man trenger ikke teknisk kompetanse for å forstå kravetàskjuler kompleksitet - Kravene uttrykkes på en kort og konsis måte

UML – modellering

https://www.youtube.com/watch?v=zid-MVo7M-E

Min shity version av en UML Use Case Diagram for et billettsystem:



Hvorfor modellerer vi?

I systemutvikling er modellering prosessen hvor man utvikler abstrakte modeller av et system. Det er viktig å forstå at en systemmodell ikke er en komplett representasjon av systemet, men at den viser kun ett perspektiv. Derfor trenger man flere typer modeller som synliggjør ulike aspekter ved systemet: ulike modeller representerer ulike måter å se systemet på. UML (Unified Modeling Language) er en standard for objektorientert modellering

Hvordan brukes abstrakte modeller i systemutvikling?

Som utgangspunkt for fokusert diskusjon om et eksisterende eller foreslått system

Use case (brukstilfelle)

Handler om å identifisere aktørens målàet use case. Interaksjonsperspektiv: en beskrivelse av hvordan systemet oppnår et mål av verdi for en aktør

Notasjon:

- Figur: oval
- Merkelapp: navnet på use caseàverbfrase
- I et use case diagram er aktøren strekfigurene. Disse må ha et rollenavn

Use case – identifisering

- Hvilke mål ønsker aktøren å oppnå med bruk av systemet?
- Hvilke resultater vil aktøren oppnå med bruk av systemet?
- Hva er de viktigste oppgavene som aktøren ønsker at systemet skal kunne utføre?
- Vil aktøren skape, lagre, endre, lese eller slette data i systemet?
- Vil aktøren ha behov for å informere systemet om eksterne endringer?
- Har aktøren behov for å bli informert om hendelser i systemet?

Forklaring

- Viser systemets funksjonalitet og samspillet mellom systemet og omgivelsene (brukere, andre systemer, komponenter)
- Består av:
    - Aktører → tegnet som strekfigurer
    - Use caseàtegnet som ovaler med merkelapper o Heltrukne streker mellom aktører og use case
    - Stiplet linje for include/extend mellom use case

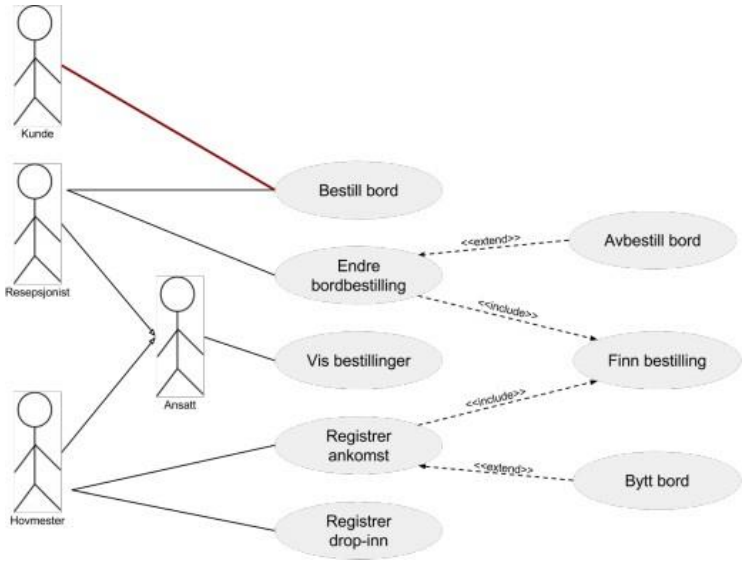
To relasjoner i et use case

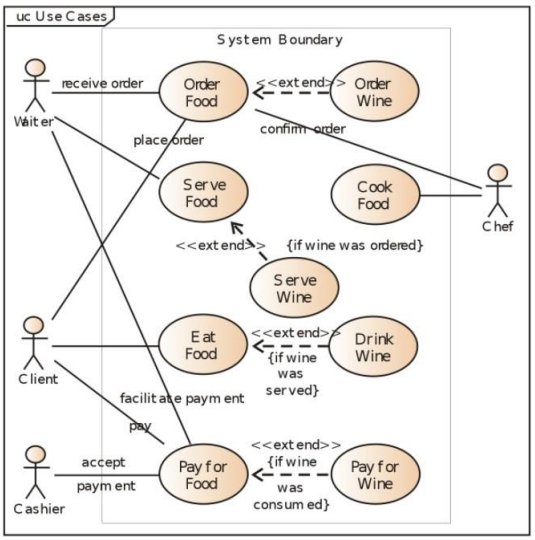
Include relasjonen

- Et use case kan være en del av ett eller flere andre use case
- indikerer at et (sub) use case inneholder nødvendig funksjonalitet for gjennomførelsen av et annet basiscase

Extend-relasjonen

- Et use case som beskriver tilleggs oppførsel som utføres under gitte omstendigheter
- Utvider oppførselen/funksjonaliteten til et basiscase, som utføres under spesielle omstendigheter





Tekstlig beskrivelse

Tekstlig beskrivelse av use case «reservere bil»

Navn: reservere bil

Aktør: kundebehandler

Prebetingelser: ingen (betingelser som må være på plass før et use case kan utføres)

Potbestingelser: leiekontrakt for spesifisert bil og kunde med gitte utkeierdatoer er opprettet (en endring i systemet som skal være på plass etter at et use case er utført)

Hovedflyt:

1. kundebehandleren velger tidsintervall(hentedato og returdato)
2. systemet returnerer en liste over tilgjengelige biler innenfor de spesifiserte datoene
3. kundebehandleren velger en av bilene
4. systemet ber om kunder og finner kunden i systemet
5. systemet bekrefter at bilen er reservert for den gitte perioden

Alternativ flyt

2.1 det finnes ingen tilgjengelige bilder i valgt tidsintervall

2.2 systemet opplyser om at det ikke er tilgjengelig biler innenfor oppgitt tidsintervall

2.3 kundebehandler oppgir et nytt tidsintervall eller avslutter bruksmønster

4.1 kunden finnes ikke

4.2 systemet oppretter ny kunde og returnerer til steg 3

Hva er en tekstlig beskrivelse av et use case?

En tekstlig beskrivelse av en use case tar for seg interaksjonen mellom systemet og bruker. Nummerert liste som beskriver interaksjonen steg for steg. En tekstlig beskrivelse skal inneholde følgende:

- Navn på use case → hvilken funksjonalitet dreier det seg om?
- Aktør → hvem/hva er det som interagerer med systemet?
- Prebetingelser → hva skal til for å starte use caset?
- Postbetingelser → hva skal til for å avslutte use caset
- Hovedflyt → hvilke steg inngår i gjennomførelsen av use caset?
- Alternativflyt → hvilke steg inntreffer ved avvik i hovedflyten?

Use case vs. User stories

Likheter, begge viser:

- Hvem som skal bruke systemet
- Hva de skal gjøre med det
- Hvorfor de skal gjøre det

Forskjeller:

- Omfang (use case er mer detaljert), kompletthet, livslengde, hensikt
- User stories er godt egnet for å finne krav og bruke disse i smidig utvikling i samarbeid med produkteier/kunde

- Use case er mer detaljert, har flere bruksområder videre i prosjektet og er mer egnet som dokumentasjon

Men det er en flytende overgang mellom dem
Hva er typisk med testing – jo man bruker use case (kan fort komme på eksamen)

Sekvensdiagrammer

- Flyt i et use case kan modelleres med sekvensdiagrammer
- For hvert use case lages typisk sekvensdiagrammer for hovedflyt og for hyppig forekommende alternativ flyt
- Stegene (sekvensene) i et use case vises som meldinger som sendes mellom objektene ved kall på objektenes metoder

Et sekvensdiagram er et interaksjonsdiagram

- Modellen viser en interaksjonssekvens mellom objektene som finner sted under et bestemt use case
    - Hvilke objekter som inngår i et bruksmønster
    - Interaksjonen mellom objektene/deres rekkefølge o Data/informasjon som sendes mellom objektene

Hvorfor er det nyttig å benytte sekvensdiagram?

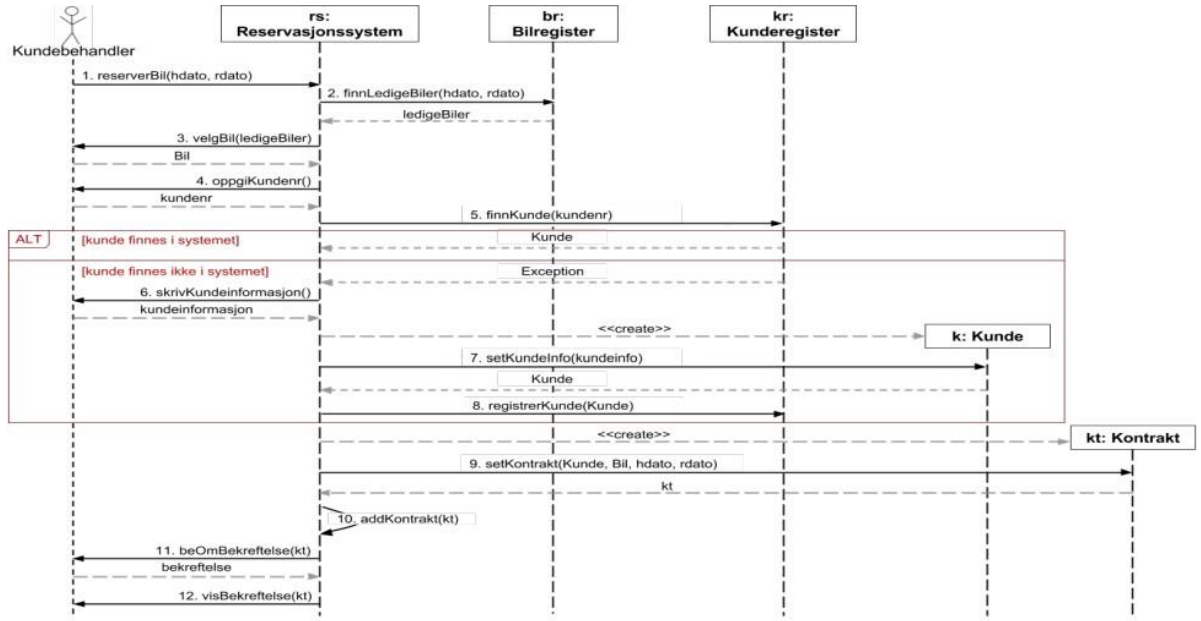
Viktig å kunne vise hva som skjer/burde skje ved kjøretid altså: viser den dynamiske oppførselen til et program. Oversikt over nødvendige klasser og objekter for å gjennomføre et bruksmønster kan gjøre det enklere å implementere et system

- svært kodenært diagram
- mulig å generere kode automatisk fra et sekvensdiagram
- viser hvordan objektene kommuniserer
- oversikt over data som sendes mellom objektene og rekkefølgen

Modellering AV SEKVENSDIAGRAMMER

Desto mer detaljert den tekstlige beskrivelsen er, desto enklere blir det å modellere det tilhørende sekvensdiagrammet. Tekstlig beskrivelse viser interaksjon mellom bruker og systemet

- Gir oss informasjon om hvem(bruker) og hva (objekter/metodekall/data)
- Følg rekkefølgen som gis fra beskrivelsen
- **Tips!** Lag en tekstlig beskrivelse (for bruksmønsteret) om denne ikke er gitt
    - Beskriv hendelsesforløpet i detalj → vis interaksjonen
- Fra beskrivelsenàkartlegg aktører og objekter
    - Følg oppsett gitt av rekkefølgen i beskrivelse
    - Modeller flyten (piler frem og tilbake) med dette som utgangspunkt



Aktivitetsdiagrammer

- Et aktivitetsdiagram kan grafisk representere hendelsesflyten i et use case
- Stegene i use case vises som aktiviteter (rektangel)
- Beslutninger undervises som (diamant)
- Aktivitetsdiagrammer og sekvensdiagrammer brukes noe overlappende, men sekvensdiagrammer er typisk mer kodenært mens aktivitetsdiagrammer er mer forretnings nært

Hva er et aktivitetsdiagram?

Aktivitetsdiagram = flytskjema(flowchart)
- Grafisk representasjon av arbeidsflyt
- Viser aktiviteter og tilhørende handlinger
- Viser overordnet kontrollflyt
- Beskriver hvordan, ulike utfall av en aktivitet påvirker flyten og Viser aktiviteter som kan utføres parallelt

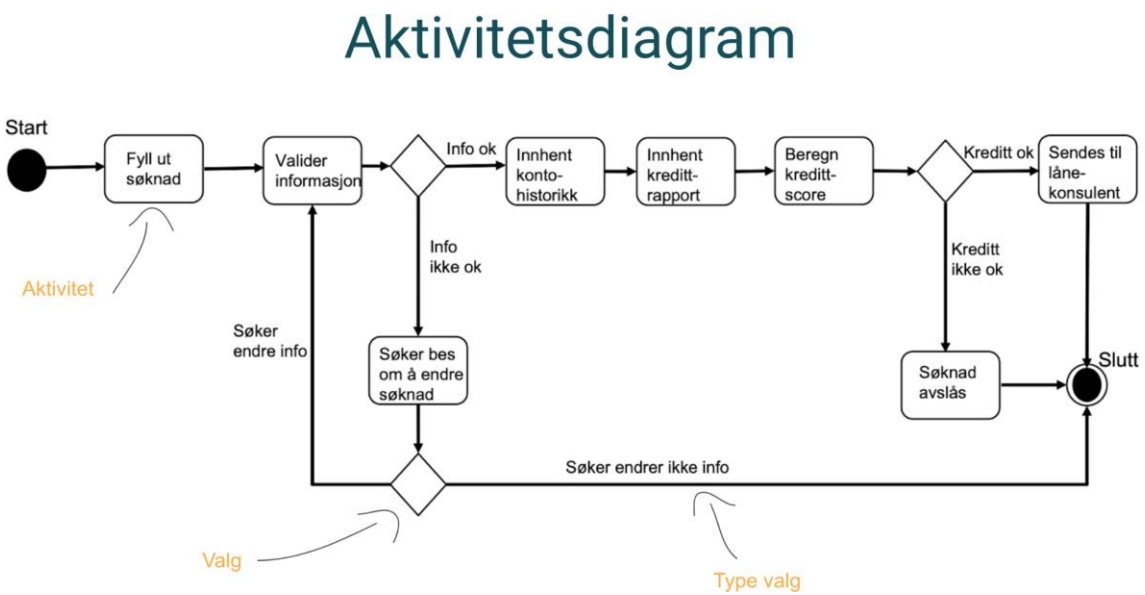
Hvorfor er det nyttig å lage aktivitetsdiagram?

- Definerer flyten for en gitt aktivitet som kan gjennomføres i systemet

- Finner prosesser som kan kjøres parallelt og er uavhengig av hverandre - Finner deadlocks i systemet

- Økt forståelse av arbeidsrutiner

- Aktivitetsdiagram kan modellere arbeidsflyt og organisasjonsflyt

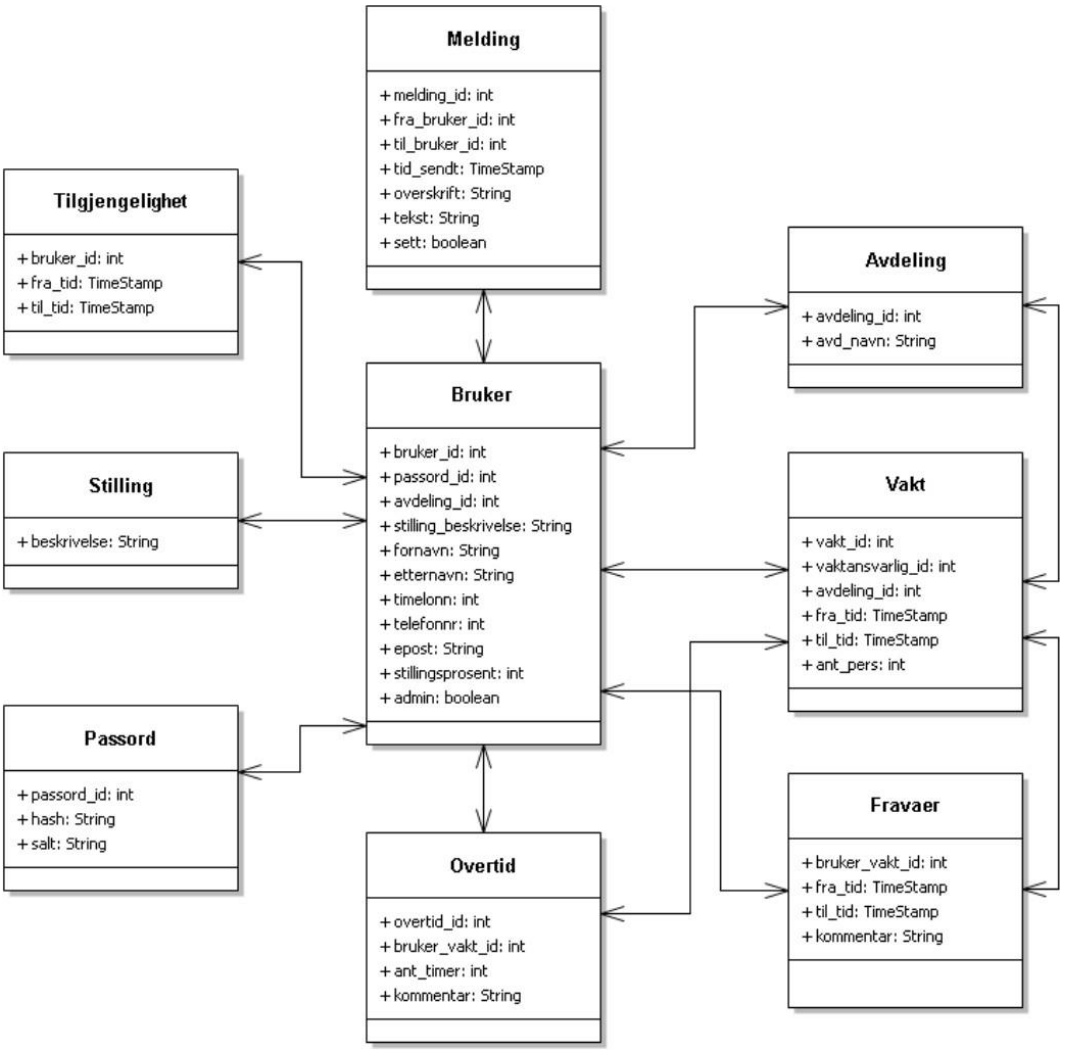


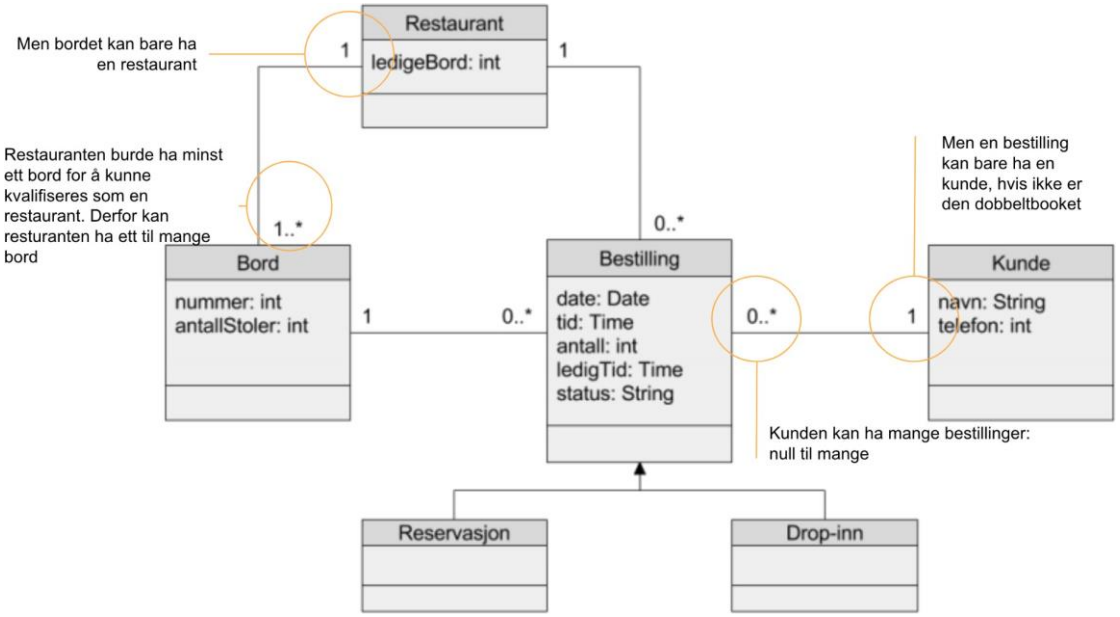
Klassediagram

Hva er en domenemodell, og hvorfor er det nyttig å lage en domenemodell for et gitt system?

En demomodell er en representasjon av de ulike objektene (i domenet) et system består av. Modelleres på samme måte som et klassediagram, men uten metoder. Det er nyttig for å:

- Kartlegge hvilke objekter som man bør ta hensyn til
- Å kommunisere at man har forstått domenet
- Se på relasjoner mellom objektene



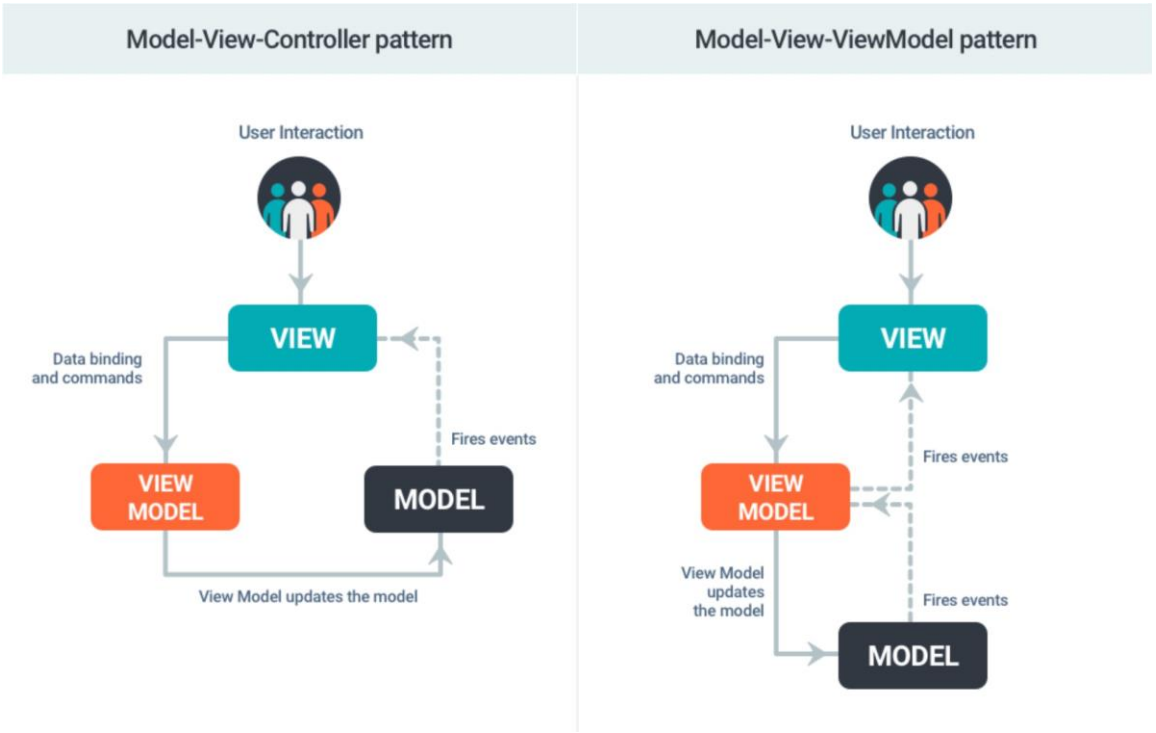


Design pattern

Modeller brukes (blant annet) for å skape felles forståelse for arkitekturen, slik at utviklere kan jobbe på hver sin enhet etter samme mønster - for så å “merge” enhetene sammen til et stort system.

- Retningslinjer for hvordan ansvar skal fordeles i ulike situasjoner
- Legger til rette for at systemet kan være oversiktlig og mer brukervennlig
- Hvis andre skal jobbe med den eller hvis det går litt tid
- Gjør arkitekturen til systemet forutsigbart og lesbart for flere

To ulike typer:



Høy kohesjon: Del opp i lag som har egne ansvarsområder
Lav kobling: fragments i view laget kommuniserte bare med view laget og ikke med hverandre (kan endre på den ene uten å endre på alle)

Kohesjon

- Når vi lager klassediagram og domenemodeller kan dette hjelpe oss å finne grad av kohesjon og grad av kobling.
- Høy kohesjon
  - Et objekt skal bare ha ansvar for relaterte ting
  - Det blir vanskelig å finne feil hvis det skulle oppstå
  - Blir fort rotete
- Lav kohesjon
  - Da har objektet ansvar for mange oppgaver

Kobling

- Lav kobling
  - Et objekt skal samarbeide med et begrenset antall andre objekter
  - Et objekt som relaterer seg med mange andre objekter

    - Oppstår det en feil i et objekt så må man teste og sjekke alle andre objektene også om det har forplantet seg
    - Det blir vanskeligere å fjerne og legge til nye objekter ved videreutvikling
        - Fordi det ofte kan kreve endringer gjort i andre objekter også
- Sterk/høy kobling
    - Et objekt er koblet/knyttet til mange andre objekter
    - Gjør endringer vanskelig

Informasjonssikkerhet

Informasjonssikkerhet betyr å beskytte informasjonsressurser mot skade - definisjon av informasjonssikkerhet

- beskyttelse av konfidensialitet, integritet og tilgjengelighet
- i tillegg kan andre egenskaper f.eks. autentisert, sporbarhet, avviseliget og pålitelighet omfattes.
- Informasjonssikkerhet er en kontinuerlig prosess for å oppdage og hindre trusler og å fjerne sårbarheter
- Informasjonssikkerhet er tradisjonelt definert som opprettholdes av KIT
    - **K**onfidensialitet **I**ntegritet **T**ilgjengelighet

Konfidensialitet

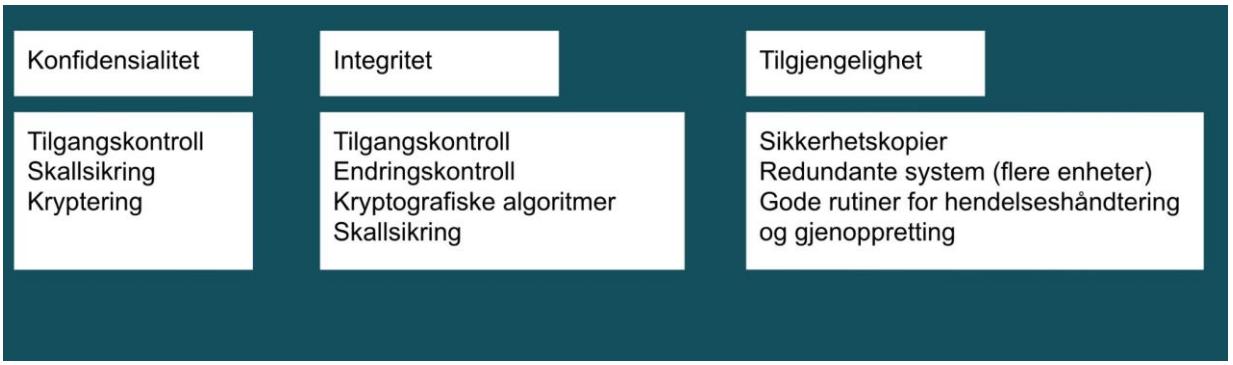
Egenskaper av at informasjon ikke blir gjort tilgjengelig eller vist til uautoriserte individer, entiteter eller prosesser

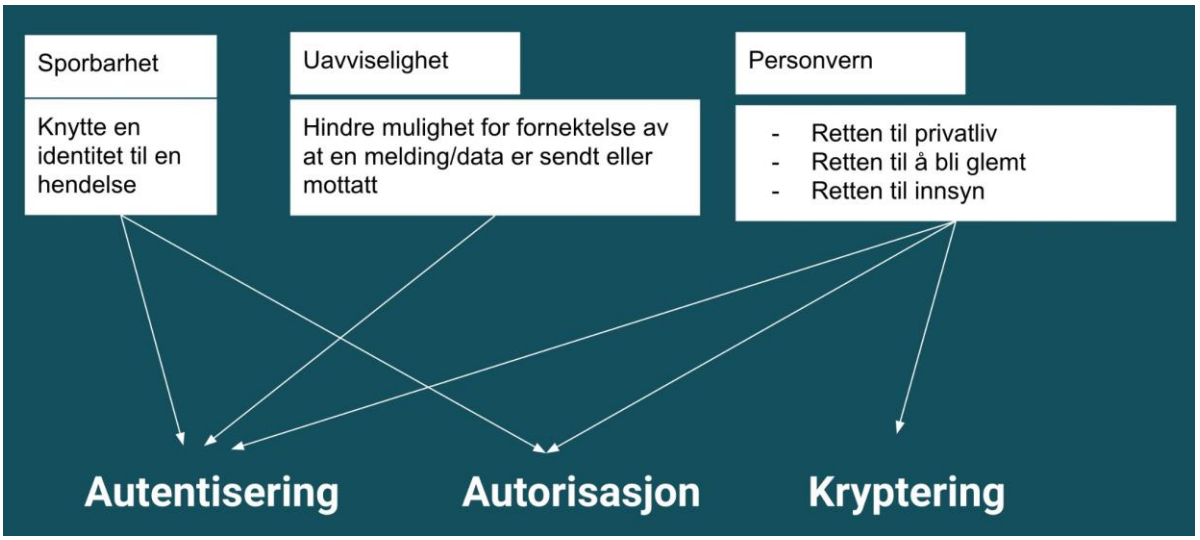
Integritet

Egenskapen av at data ikke har blitt endret eller slettet på en autorisert måte. Egenskapen av å opprettholde korrekthet og kompletthet av dataressurser

Tilgjengelighet

Egenskapen av at data og tjenester er tilgjengelig og anvendbare ved forespørsel fra en autorisert entitet.


Konfidensialitet
Tilgangskontroll
Skallsikring
Kryptering
Integritet
Tilgangskontroll
Endringskontroll
Kryptografiske algoritmer
Skallsikring
Tilgjengelighet
Sikkerhetskopier
Redundante system (flere enheter)
Gode rutiner for hendelseshåndtering og gjenoppretting


Sporbarhet
Knytte en identitet til en hendelse
Uavviselighet
Hindre mulighet for fornektelse av at en melding/data er sendt eller mottatt
Personvern
- Retten til privatliv
- Retten til å bli glemt
- Retten til innsyn
Autentisering
Autorisasjon
Kryptering

Nyttige begreper innen informasjonssikkerhet

Risiko

En kombinasjon av en trussel, en sårbarhet overfor trusselen, og hendelsens konsekvens representerer en spesifikk risiko.

Risikonivå

Defineres kvalitativ, relativ eller kvantitativt

Trussel

En konkret trussel mot en informasjonsressurs.

Sårbarhet

Svakheter som kan utnyttes for å angripe systemer og informasjonsressurser. Det finnes ingen sårbarheter uten trusler

- en sårbarhet er en mangel på eller svakhet i beskyttelse mot trusler - fjerning av sikkerhetssårbarhet er å stoppe mulige trusler
- fjerningen gjøres med sikkerhetstiltak

Sikkerhetstiltak

Et tiltak som etableres for å redusere eller modifisere risiko

Trussel modellering

Er å identifisere, analysere og beskrive relevante trussel scenarioer. Avdekker sårbarheter ved å se hvordan et trussel scenario er mulig å gjennomføre → hvordan det ikke stoppes

- Trussel modellering er viktig under systemutvikling
    - Brukes til å identifisere, fjerne og unngå sårbarheter ved utvikling av programvare og systemer.

Redundante system

- dupliseringen av kritiske komponenter og funksjoner av et system med den hensikt å øke påliteligheten til systemet
- Redundans bygges ofte inn i systemer som krever høy pålitelighet. I datasystemer kan to eller flere datamaskiner jobbe parallelt med samme oppgaver og speile hverandre, slik at dersom en av dem skulle gå ned så kan den andre ta over.
- Sikkerhetskopi av data er en annen viktig form for redundans som brukes for å beskytte seg for informasjonstap. En variant av dette er mellomlagring av data til nye filer under redigering, for å sikre de siste endringene.

Hva er sikkerhet?

Sikkerhet er beskyttelse av verdier mot skade Ulike typer:

- Fysisk sikkerhet

- Samfunnssikkerhet

- Nasjonal sikkerhet

- Hms

- Miljøsikkerhet

- Informasjonssikkerhet - Personvern

Hva er informasjonssikkerhet?

Informasjonssikkerhet betyr å beskytte informasjonsressurser mot skade

Hvilke informasjonsressurser skal beskyttes?

- Data, programvare, konfigurering, utstyr og infrastruktur
- Dekker både tilsiktet og utilsiktet skade.
- Trussel agenter kan være mennesker eller naturlige hendelser
    - Mennesker kan gjøre skade både tilsiktet og utilsiktet
- Definisjon av informasjonssikkerhet:
    - Beskyttelse av konfidensialitet, integritet og tilgjengelighet
    - I tillegg kan andre egenskaper f.eks. autentisitet, sporbarhet, uavviselighet og pålitelighet omfattes

Autentisering

- **Er du deg?**
- Autentisering: er bruker brukeren?
- Entitets autentisering
    - Brukerautentisering
        - § Passord
    - Systemautentisering
        - § Kryptografiske protokoller
        - § Transport layer security

- Data autentisering
    - Hvem har sendt meg dette dokumentet
    - MAC

    - Digital signatur

Autorisering

- Tilgang til ressurser

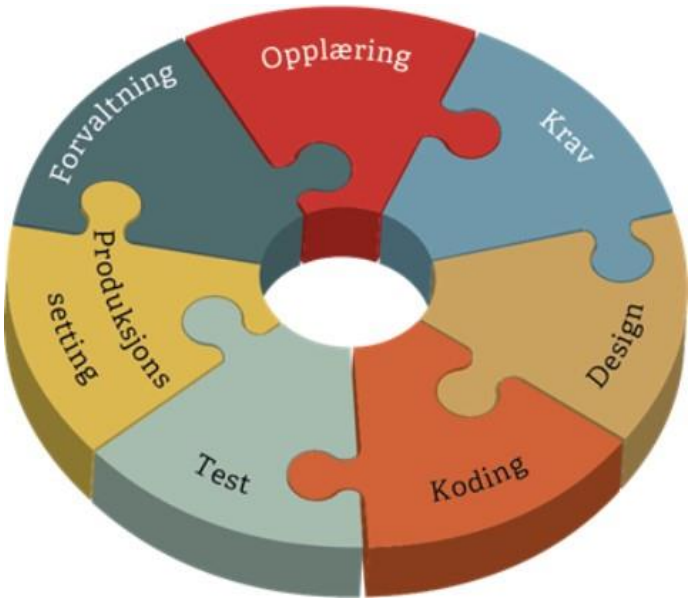
  Autorisering: utdeling av rettigheter (fra bruker til moderator til administrator)
- Logger inn
- Konfigureringsfasen
    - Registrerer ny bruker
    - Hva blir man autorisert til?
        - § Hva skal man ha tilgang til
- Driftsfasen
    - Logger inn
    - Sjekker om vi har tilgang til det vi skal ha tilgang til

Økende behov for informasjonssikkerhet

Vi har ikke mulighetene til å løse alle sikkerhetsproblemer i verden Dette er fordi:

- Digitaliseringen skaper nye sårbarheter og intensiver for angrep
- Flere og flere tjenester er eksponert online
- Økende antall cybertrusler
- Informasjonssikkerhet ignoreres ofte når utviklere har tidspress
- Kriminalitet følger nye tjenester og forretningsmodeller
- Mer effektive angrepsverktøy utvikles

Innebygd personvern og informasjonssikkerhet



- Veileder fra datatilsynet
- Opplæring
    - De som bygger it systemer må ha kompetanse
    - Basiskunnskap og forståelse for personvern
    - De har etablerte retningslinjer for personvern og informasjonssikkerhet
- Krav
    - Når man lager kravspecen
    - Må definere personopplysninger som skal lagres
    - Skal man bruke sktyleverandører
- Design
    - Minimere innhenting
    - Separer behandlinger osv.
    - Trussel modellering
        - § Fjerne sårbarheter osv.
- Koding
    - Vurdere sårbarheter i biblioteker
    - Vurdere sårbarheter i verktøy
- Test
    - Verifisere at trussel scenariene i designfasen er håndtert
        - § Og kravspecen
    - Og at de nye trussel scenariene som eventuelt kom under kodingen

- Produksjonssetting
    - Systemet skal ut i drift
    - Det vil skje noe
        - § Hendelseshåndtering
        - § Hvordan skal vi håndtere hendelser
        - § Har vi ressurser
        - § Har vi en plan for å fjerne sårbarheten
        - § Hvis det skjer hat vi en plan for varsling
            - Plikt til å varsle datatilsynet
            - Gi beskjed til pressen à økt tillit
- Forvaltning
    - Når man har en plan for alt – da gjelder det å sette det ut i drift – forvalte systemet
    - Skjer det noe
        - § Holde seg til planen

Sikkerhetstiltak

Det er ofte nødvendig å benytte en kombinasjon av tiltak fra elle tre faser for å opprettholde dekkende beskyttelse

- Preventive tiltak:
    - Forhindre og avskrekke angrepsforsøk
    - F.eks. kryptering av flere filer for konfidensialitet
- Detektiv tiltak:
    - Varsle angrep som forsøkes eller som allerede har hendt inntrengingsdeteksjon
- Korrigerende tiltak
    - Gjenopprette skade på dataressurser etter angrep og hente ut backup av programmer og data ved tap/kompromittering av ressurser.

Risiko

Risikoanalyse

- vurderer sannsynligheten og mulig konsekvens for hver risiko

- sannsynlighet kan være svært lav, lav, moderat, høy eller svært høy

- konsekvensen kan være katastrofal, alvorlige, mindre alvorlig eller ubetydelig

Risikohåndtering

Tre hovedtyper for risiko:

- Prosjektrisiko
- Produktrisiko
- Forretningsrisiko

Risikomatrise

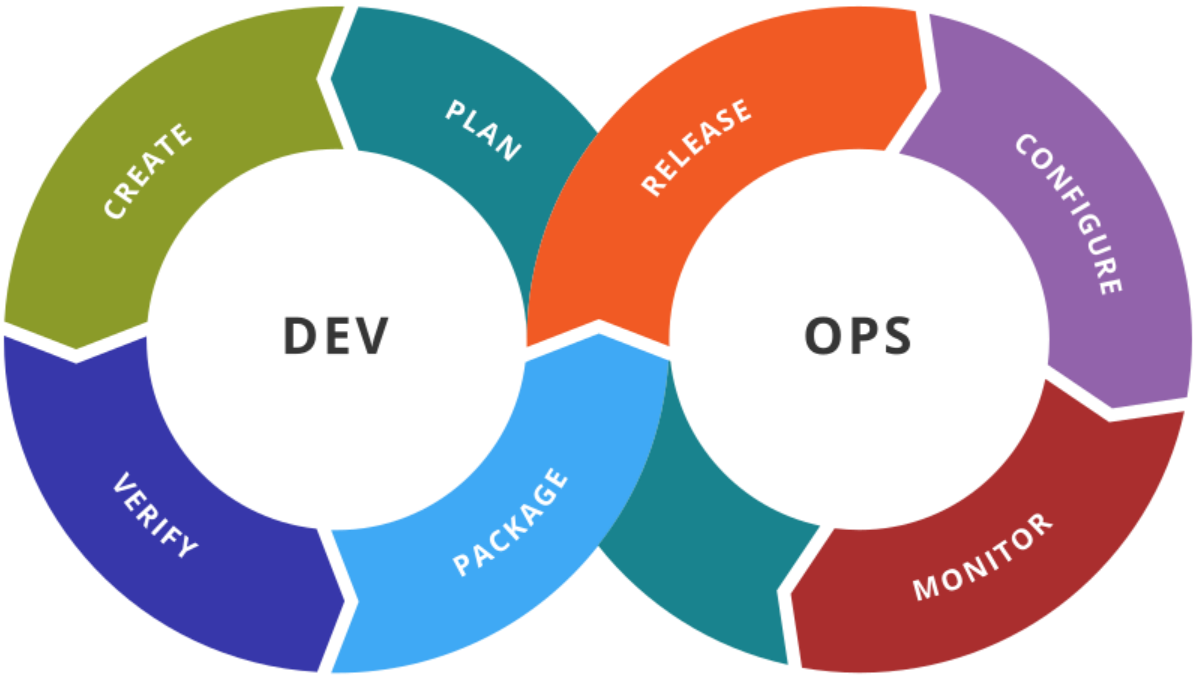
Lag en slik tabell:

Risiko	Sannsynlighet	Konsekvens	Tiltak
Ansatte blir syke	Høy	Alvorlig	Ha tilgang til vikarbyrå
Intern konflikt	Svært høy	Katastrofal	Teambuilding HMS-oppfølging

DevOps

- DevOps (development + operations) er en alternativ modell som integrerer alt ovenfor i **et eneste team**
- Integrer utvikling, utgivelse og support til ett team som har ansvar for alle de tre aktivitetene

- Ingen fast definisjon av DevOps
  - Fordi det blir implementert forskjellig i ulike organisasjoner/bedrifter
  - Varier etter deres kultur og hvilken type Software de utvikler
- Spesielt nyttig for skybaserte tjenester
- Men det er tre sånne fundamentale prinsipper som er basis for DevOps



Prinsipper

DevOps prinsipp	Forklaring
Alle ansvarlige for alt	Alle i teamet har delt ansvar for utvikling, utgivelse og vedlikehold/support av programvaren.
Alt som kan bli automatisert burde bli det	All testing-, utgivelse- og supportsaktiviteter bør bli automatisert når det er mulig. Legg opp til minst mulig manuelt arbeid med utgivelsen av programvaren.
Mål først, endre etterpå	DevOps burde bli drevet av målingsprogram hvor du samler data om systemet og operasjonene. Avgjørelser som angår endring i DevOps prosessen og verktøy, bør tas med denne dataen som grunnlag.

Fordeler

- **Raskere utgivelse/leveranse (Deploy)**
  - Fordi: kommunikasjons delay mellom teamene er blitt dramatisk redusert
- **Redusert risiko**
  - Fordi: inkrementet i hver leveranse er mindre, og dermed enklere å lokalisere kilden til problemet
- **Raskere reparasjon**
  - DevOps teamene jobber sammen for å få programvaren oppe og går
  - Man trenger heller ikke å finne ansvarlig team og vente på at de fikser det dersom et problem oppstår
- **Produktive team**
  - Teamene blir gladere og mer produktive fordi de jobber sammen og ikke på separate områder

Kodeadministrasjon

- **Hva er kodeadministrasjon?**

  - DevOps er avhengig av kildekodestyring – administrasjonssystem for nettopp dette
  - Hvem her har hørt om GitHub?

  - Kodeadministrasjon er et sett med programvare-støttede praksiser for å administrere en inkrementelt voksende kodebase
    - § Trengs for at endringer gjort av forskjellige utviklere ikke skaper problemer, men merges godt sammen :)

  - Kodeadministrasjonsverktøy gjør det lett å skape et kjørende produkt fra kildekode/base og kjøre automatiske tester
    - § Slik at eventuelle problemer blir oppdaget før man har sendt det ut til globale repository – globale repository

- Dette kode administrasjons systemet gir et sett med funksjoner som

støtter fire generelle områder

- **Kodeoverføring**
  - Utviklere tar kode inn i deres personlige fillager for å jobbe med det, så returnerer de det til det delte kodeadministrasjonssystemet
- **Versjonlager og henting**
  - Filer kan bli lagret i forskjellige versjoner, og spesifikke versjoner kan bli hentet
- **Merging/Branching**
  - Parallell utviklingsgrener kan bli laget for å bli jobbet med samtidig. Endringer gjort av utviklere på forskjellige branches kan bli merget
- **Versjoninformasjon**
  - Informasjon om de forskjellige versjonene holdt i systemet kan bli lagret og hentet
  - Avhengig for å kunne holde kontroll over alle endringer som blir gjort

Automatisering

- Alt som kan automatiseres skal automatiseres – fundamentalt prinsipp for DevOps
  - På den kan du redusere kostnadene i tid og penger på integrasjon, levering og utgivelse
- DevOps automasjons- og målingsverktøy interagerer med kodeadministrasjonssystemet
- Finnes mange flere automatiseringsverktøy, som hjelper med dette:

· **Kontinuerlig integrasjon**
  - hver gang en utvikler foretar en endring i prosjektets hovedgren, blir en kjørbar versjon av systemet bygget og testet
  - hver gang en endring blir pushet til den delte repositoryen så blir det integrert
  - samler alle delene og integrer dem til et samlet system som funker
  - passe på å ikke brak the build
    - § pusher kode til repository som vil ødelegge når det blir integrert
  - dette gjør at det er enklere å løse eventuelle feil
    - § Hvis man gjør en liten endring og testen feiler så kan man nesten sikkert si at kilden er i den nye koden man pushet
  - I tillegg så blir det fungerende systemet tilgjengelig for alle i teamet hele tiden – ved at man oppdaterer jevnlig

· **Kontinuerlig levering**
  - en simulering av produktets driftsmiljø opprettes og den kjørbare programvareversjonen blir testet
    - § slik kan du oppdage eventuelle bugs som ikke vistes i test miljøet
    - § kan også sikre at det nye endret systemet er klar for utgivelse

til brukerne
  - overordnet strategi, korte sykler med produksjon som sørger for at programvaren alltid er funksjonell og klar for å slippes. Mål: bygge, teste og slippe programvare hurtigere og oftere.
  - Selv om man kontinuerlig tester så betyr ikke dette at man pusher programvaren ut til brukerne med en gang
- **Kontinuerlig utgivelse:**
  - En ny versjon av systemet blir gjort tilgjengelig til brukere hver gang det er gjort endringer på master branchen av programvaren
  - automatisert testing av ny funksjonalitet fører til automatisert slipp av ny funksjonalitet.
- **Infrastruktur som kode:**
  - støtte alle de ulike fysiske og virtuelle serverne.
  - Programvaren som skal installeres, for eksempel kompilatorer og biblioteker, er inkludert i infrastrukturmodellen.

Måling

- Så når man har tatt i bruk DevOps så ønsker man å kontinuerlig forbedre seg for å oppnå raskere utgivelse og bedre kvalitet på programvaren. Og da trenger man å måle litt forskjellig for å se hvordan ståa er...
- Prosessmåling og tjeneste måling er de to mest relevante innen for DevOps
- Vanskelig å finne svar på det man lurer på
  - Hva er enkelt?
  - Hva er god kvalitet?
  - Må derfor finne metrikker som man enklere kan teste for å finne svar på det overordnede man lurer på.
- Automatisering
  - Burde gjøre dette automatisk
  - Systemet samler inn data om seg selv
  - Benytte seg av en form for overvåkningssystem
- Og jevnlig så presenterer man dette ofte i form av grafer
  - Kan enklere se når noe gikk bra eller når noe gikk dårlig
  - Rundt den og den datoen så økte det kraftig med klager fra kunder
    - § Ble en ny funksjon lanser?
    - § Kanskje var det en funksjon som ble tatt bort eller erstattet?

IT-kontrakter

"En kontrakt er en avtale som mellom partene etablerer en bindende forpliktelse til å gjøre eller til å unnlate å gjøre noe."

Innhold

- Partnere
  - Kunde
  - leverandør
- Leveranse
- Fremtidsplan
- Spesifisert resultat
- Definert tjenestenivå eller kvalitet
  - Hvilke tjenester skal man ha og hvilket nivå det skal leveres på
- Pris
- Hvilken kompetanse skal leverandøren ha?

Ulike kategorier

- One time off kontrakter
  - Du leverer èn gang og så er du ferdig
- Rammeavtaler
  - Avtaler over tid
- Løpende tjenestekontrakter
  - Tjenester over tid (skybaserte tjenester, vedlikeholdsavtaler)
- Samarbeidsavtaler
  - Oppstarten på et prosjekt, samarbeid om en tilbudsprosess
  - F.eks. hvem skal gjør hvilken jobb?
- Garantier
  - Garanti av et morselskap: hvis datterselskapet ikke klarer å levere så må morselskap steppe inn å fullføre leveransen eller eventuelt betale regningen

Utfordringer

- Kunde: vet ikke alltid hva de vil ha eller hvilke ressurser som må til for å skape it-tjenesten.
- Leverandør: forsøker å forklare og tegne opp estimater, men klarer ikke alltid å forklare godt.
- Kompleksitet
  - Høy grad av kompleksitet
- Utvikling av noe nytt
- Sosiotekniske systemer: Skal bruke av mennesker og kan innebære endringer i arbeidsprosesser og organisering
- Abstrakte og usynlige systemer: iallfall for brukeren

- Mangel på modenhet i IT-bransjen
    - IT bransjen er ung
    - Og det kommer stadig nye metoder å gjøre ting på

Smidig

Smidig → Fleksibilitet i kontrakt

*"I **gjennomføringen** bestemmer du som kunde hva du skal ha"* - Ståle L. Hagen

- Reagerer på endring vs. følge en plan
- Samarbeid med kunde → Når IT-tjenester utvikles sammen med kunden. Kravendringer og fleksibilitet i kontrakten er viktig.
- Korte iterasjoner
- Lite dokumentasjon

Plandrevet

Plandrevet → Forutsigbarhet i kontrakt

- Spesifisering av det vi skal ha før man går til leverandørmarkedet - man skal få det man spesifiserte at man ville ha.
- Når man klarer å spesifisere hvilken IT-tjeneste man skal ha før man går til leverandørmarkedet er det mulig å skrive en forutsigbar kontrakt.
- Spesifisert resultat
    - Plan
    - Datoer → **når skal ting være ferdig**
- Fast pris
- Ikke veldig fleksibelt

Viktig å forstå at det er utfordrende å skrive kontrakt, og at standardene hjelper med å skrive en god kontrakt.