UKE 8 - Håndtering av krav

IN1030 - Gruppe 2

Hva skal vi i dag?

- Krav
- Kravspesifikasjon
- Validering av kravspesifikasjonen
- Kravendringer
- Ukesoppgaver

Frist for å melde inn grupper i morgen!!

(Oblig 4 blir lagt ut på onsdag)

Gruppearbeid - gode linker:

For samarbeid: Zoom, Teams, MetroRetro, Google Drive.

For tegning av UML: https://www.draw.io

Menti

Læringsmål - forklart:

kan du anvende metoder og teknikker for **kravhåndtering:**

- innhente, analysere og spesifisere krav.
- skille mellom funksjonelle og ikke-funksjonelle krav.
- skrive kravspesifikasjon.
- utføre modellering av UML-diagram basert på krav (neste uke).

VIDEO:

Planleggingsmøte med umulig utgangspunkt og hvor krav endres hele tiden

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

Kravendringer vil alltid forekomme

Kravene bør være...

- **Forståelig**
  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

Krav

Hva er krav?

Vi finner krav ved å analysere konteksten til et system. Og evt gjennom kontrakt med kunde.
Krav bestemmer hvilke funksjoner et system skal ha.

Brukerkrav: påstander om planlagte tjenester og/eller begrensninger til systemet, basert på kunden/brukers behov. Fra brede beskrivelser av systemets egenskaper til detaljerte funksjonsbeskrivelser.

Systemkrav: detaljert, formell beskrivelse av programvarens funksjoner, tjenester eller operasjonelle begrensninger. Implementasjonen av disse skrives i detalj i et systemkrav dokument.

Spørsmål?

Funksjonelle krav

Hva systemet skal gjøre.

Funksjonelt brukerkrav: Systemet skal tilby bruker å laste opp bilder.

Funksjonelt systemkrav: Systemet bør tillate brukere å laste opp både bilder i jpg og png

Skrives ofte slik: “Systemet skal…../systemet bør (nice to have)....”

Ruter som eksempel

Systemet skal kunne vise en oversikt over en brukes betalte billetter
Systemet bør tilby hurtigkjøp av tidligere valgte billetter
Systemet skal gi en beskjed når det er under 24 timer til en billett løper ut
Systemet skal fjerne bankkort hvis brukeren taster feil pin tre ganger
Systemet skal tilby funksjonalitet for valg av billettype

Ikke-funksjonelle krav

Krav som ikke direkte beskriver funksjonene som leveres til sluttbruker, men karakteristikker med systemet som helhet; *Hvordan skal det fungere?*

Sier noe om kvalitetsattributter systemet skal ha
Sier noe om egenskaper - hvordan systemet skal oppføre seg
Bør være målbare (mulighet for å teste dem)

Sier noe om:

- **Implementasjon**
- **Reliabilitet**
- **Responstid**
- **Sikkerhet**

Mer kritiske → Møter man ikke disse kan det føre til at hele systemet ikke fungerer.

Ikke-funksjonelle krav

Produktkrav: **Brukskvalitet/brukervennlighet, ytelse og effektivitet samt lagringsplass, pålitelighet og lagring av data.**
Eks: Systemet skal kun bruke 5 MB lagringsplass på bilder.

Organisatoriske krav: **kostnader & ressurser, leveransetidspunkt & prosess; utviklingsmodeller, programmeringsspråk, verktøy og komponenter samt generelle standarder og regler.**
Eks: Systemets ulike funksjoner skal testes hver andre måned.
Systemets kodehåndtering skal gjøres i GitHub.

Eksterne krav: **lovverk, begrensninger & etiske problemstillinger.**
Eks: Systemet skal tilfredsstille 39 av 61 WCAG-retningslinjer.

Ruter som eksempel

En ny kunde skal kunne betale for en billett på under tre minutter
Systemet skal kunne håndtere 10.000 brukere samtidig
Systemet skal utvikles i programmeringsspråk Java

Spørsmål?

Kravhåndteringsprosessen

Steg 1: Forstudie/målanalyse
- Kost/nytte-analyser | Risikoanalyser | Gevinstrealisering

Steg 2: Kravinnsamling og kravanalyse
- Hva ønsker interessentene seg? Hva har de behov for?
- Prioritering av kravene

Steg 3: Kravspesifisering
- Utgangspunkt for anbud og kontrakt (mellom kunde og leverandør)
- Utgangspunkt for design, implementasjon og testing
- Utgangspunkt for estimater (tid og kostnad)

Steg 4: Validering av kravspec

Steg 5: Håndtering av kravendringer

Forstudie:

Hvem er aktør?
Hvilke funksjoner ønsker aktør?
Politiske, økonomiske eller alternative aktørers krav?
Ressursbruk: tid, kostnader, arbeidskraft.
Klassifisering, prioritering, dokumentasjon

Kravhåndteringsprosessen

Steg 1: Forstudie/målanalyse
- Kost/nytte-analyser | Risikoanalyser | Gevinstrealisering

Steg 2: Kravinnsamling og kravanalyse
- Hva ønsker interessentene seg? Hva har de behov for?
- Prioritering av kravene

Steg 3: Kravspesifisering
- Utgangspunkt for anbud og kontrakt (mellom kunde og leverandør)
- Utgangspunkt for design, implementasjon og testing
- Utgangspunkt for estimater (tid og kostnad)

Steg 4: Validering av kravspec

Steg 5: Håndtering av kravendringer

Kravspesifikasjon:

Skaper felles forståelse av systemet.
Skaper enighet om hva som skal leveres.
Er grunnlag for kontrakt som viser hva leverandør og kunde blir enige om.
Forhindrer eventuelle konflikter som kan oppstå på bakgrunn av uklare forventninger.

Kravhåndteringsprosessen

Steg 1: Forstudie/målanalyse
- Kost/nytte-analyser | Risikoanalyser | Gevinstrealisering

Steg 2: Kravinnsamling og kravanalyse
- Hva ønsker interessentene seg? Hva har de behov for?
- Prioritering av kravene

Steg 3: Kravspesifisering
- Utgangspunkt for anbud og kontrakt (mellom kunde og leverandør)
- Utgangspunkt for design, implementasjon og testing
- Utgangspunkt for estimater (tid og kostnad)

Steg 4: Validering av kravspec

Steg 5: Håndtering av kravendringer

Validering av kravspec:

Møter vi aktørers faktiske behov?

Finne eventuelle problem/mangler med kravspesifikasjonen.

Kravhåndteringsprosessen

Steg 1: Forstudie/målanalyse
- Kost/nytte-analyser | Risikoanalyser | Gevinstrealisering

Steg 2: Kravinnsamling og kravanalyse
- Hva ønsker interessentene seg? Hva har de behov for?
- Prioritering av kravene

Steg 3: Kravspesifisering
- Utgangspunkt for anbud og kontrakt (mellom kunde og leverandør)
- Utgangspunkt for design, implementasjon og testing
- Utgangspunkt for estimater (tid og kostnad)

Steg 4: Validering av kravspec

Steg 5: Håndtering av kravendringer

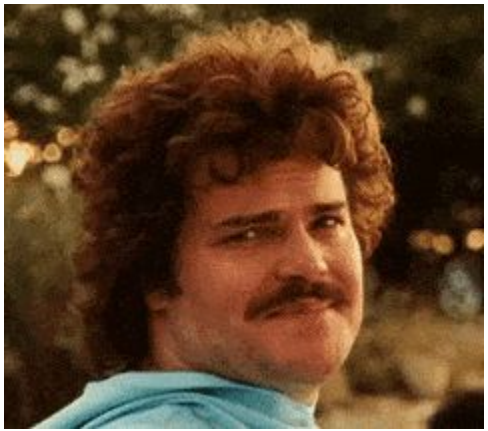
Kravendringer:

Når forståelsen av et problem endrer seg.
Plan for kravhåndtering
Håndtering av kravendringer

Spørsmål?

Ukesoppgaver

Talk but write



UKESOPPGAVER

Fasit

Spørsmål 1

Spørsmål: Et system utvikles i henhold til en presis og detaljert kravspesifikasjon skrevet av kunden. Det er først under testing at kunden begynner å ane at det var mangler ved kravspesifikasjonen. Systemet tas i bruk og brukerne oppdager at systemet er ubrukelig fordi det ikke løser deres problem.

Diskuter hva som kunne vært gjort for å hindre eller redusere dette problemet

Svar: Dette kan forhindres ved at **brukeren** får teste systemet **underveis** i systemutviklingsprosessen, får være med å **validere** kravspesifikasjonen; både før og underveis i prosjektet og at man justerer systemutviklingsprosessen etter potensielle endrede krav. Skal man hindre det **helt** må kunden ta **stor** del av systemutviklingsprosessen, eller man må ha en **tydelig** kommunikasjon og validering av hva som skal lages.

- Levere **inkrementelt**
- La bruker prøve tidligere versjoner, gjerne i naturlige omgivelser
  - Utviklere får bekreftet/avkreftet hvorvidt de er på rett vei
- Involvere kunden i større grad
- Forstå hvilke problemer kunden egentlig trenger å få løst
  - Vet kunden hva de vil ha?
  - Har utviklerne faktisk laget dette?

Spørsmål 2A

Spørsmål: Hva er en kravspesifikasjon?

Svar del 1: Et **dokument** som spesifiserer kravene til et system:

- **Spesifiserer** system- og brukerkrav
- Deler mellom funksjonelle krav og ikke-funksjonelle krav
- 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 utvikler tolker dem forskjellig → **konflikt**

Spørsmål 2A

Spørsmål: Hva er en kravspesifikasjon?

Svar del 2:
Vi deler i to underkategorier av funksjonelle og ikke-funksjonelle krav:

- **Brukerkrav:**
  - Krav uttrykt i **naturlig** språk eller diagrammer som viser ønskede tjenester (funksjoner) til systemet og føringer som gjelder (kvalitetsegenskaper)
  - Skal forstås greit av kunden
  - Må **ikke** ha en **teknisk bakgrunn** for å forstå dem
- **Systemkrav:**
  - **Strukturert**, detaljert beskrivelse av systemets funksjoner og føringer som gjelder (kvalitetsegenskaper)
  - Definerer **hva** som skal **implementeres**
  - Utgangspunkt for kontrakt mellom kunde (oppdragsgiver) og utviklerorganisasjon

Spørsmål 2B

Spørsmål: Hvorfor er det nødvendig å lage en kravspesifikasjon?

Svar: For å lage et system som **møter** brukernes **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 leverandør og kunde blir enige om
- **Forhindrer** eventuelle **konflikter** som kan oppstå på bakgrunn av uklare forventninger

Spørsmål 3

Spørsmål: Gi en definisjon av begrepet “interessent”.

List opp noen interessenter for en app som finner restauranter i nærheten av der du befinner deg.

Svar del 1: En interessent er en **person/gruppe/organisasjon** som deltar i, eller som har interesse av systemet:

- Blir **påvirket** av eller **påvirker** systemets utvikling og bruk
  - Både direkte og indirekte
- Interessenter både påvirker og påvirkes av **kravspesifikasjonen**
- Eksempler er oppdragsgivere, brukergrupper, ledere, utviklere, vedlikeholdere, systemeiere og forvaltere, pluss andre som fagforeninger, lovgivere osv.
- Konkurrenter er også interessenter

Spørsmål 3

Spørsmål: Gi en definisjon av begrepet “interessent”.

List opp noen interessenter for en app som finner restauranter i nærheten av der du befinner deg.

Svar del 2: Man snakker gjerne om fire hovedkategorier:

- **Kunde:** kjøper/bestiller produkt
- **Bruker:** sluttbruker av systemet
- **Leverandør:** de som utvikler systemet
- **Andre:** øvrige

Spørsmål 3

Spørsmål: Gi en definisjon av begrepet “interessent”.

List opp noen interessenter for en app som finner restauranter i nærheten av der du befinner deg.

Svar del 3: List opp noen interessenter

- Restauranteier: ønsker mange kunder
- Ansatte: ønsker en trygg arbeidsplass og tips
- Utviklere: ønsker gode tekniske løsninger og gode referanser
- Kunder: ønsker å finne frem til en restaurant (ønsker brukervennlighet)
- Lovverk: ønsker ivaretakelse av f.eks. personvern
- Konkurrenter: ønsker å være en attraktiv app og trekke flest mulige brukere
    - Google maps (restaurant funksjonen)
    - Yelp
    - TheFork

Spørsmål 4A

Spørsmål: Hva er funksjonelle og ikke-funksjonelle krav?

Svar del 1: Funksjonelle krav beskriver hva systemet skal gjøre (men kan også beskrive hva systemet ikke skal gjøre):

- **Hvilke** tjenester/funksjoner skal systemet tilby?
- **Hvordan** skal det reagere på ulike typer input?
- Avhenger av hvilket system som skal utvikles, systemets brukere og de som er ansvarlige for å beskrive kravene
- Varierer fra generelle krav til hva systemet skal gjøre, til mer spesifikke krav som reflekterer arbeidsmetoder eller en organisasjons allerede eksisterende system
- Skrives gjerne på formen “Systemet skal../Systemet bør (nice to have)...”.

Spørsmål 4A

Spørsmål: Hva er funksjonelle og ikke-funksjonelle krav?

Svar del 2: Ikke-funksjonelle krav definerer hvordan systemet skal innfri de funksjonelle kravene og hvordan systemutviklingsprosessen skal utføres. Sier noe om hvilke **kvalitetsattributter** systemet skal ha, noe om **egenskaper** → hvordan skal systemet oppføre seg?. Må være **målbare**, og kan beskrives som *kvalitetsønsker:* krav til systemet som ikke handler om funksjonalitet, men kvalitet og systemegenskaper som **pålitelighet**, **effektivitet** og **brukskvalitet.**
Man deler de ikke-funksjonelle kravene inn i:

- **Produktkrav:** beskriver brukskvalitet/brukervennlighet, ytelse og effektivitet samt lagringsplass, pålitelighet og lagring av data
- **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

Spørsmål 4B

Spørsmål: Skriv fem funksjonelle krav til appen beskrevet under oppgave 3 (Resturant appen)

Svar:

1. Systemet skal inneholde funksjonalitet for å legge til en restaurant
2. Systemet skal inneholde funksjonalitet for å legge til en vurdering av en restaurant
3. Systemet skal inneholde funksjonalitet for å vise restauranter i nærheten av brukeren
4. Systemet skal inneholde funksjonalitet for å legge til en meny for en restaurant
5. Systemet skal inneholde funksjonalitet for å legge til kategori for en restaurant

Spørsmål 4C

Spørsmål: Skriv fem ikke-funksjonelle krav til den samme appen (Restaurant appen)

Svar:

1. Systemet må være raskt - det skal ikke ta mer enn 2 sekunder å laste inn en side
2. Systemet må være brukervennlig - en ny kunde skal finne en restaurant på under ett minutt
3. Systemet må være plattformuavhengig - det skal fungere på alle mobile plattformer (android, ios osv)
4. Systemet skal kunne håndtere mange brukere samtidig - 50 000 samtidige brukere.
5. All systemdokumentasjon skal være forståelig - engelsk skal brukes som et felles språk for dokumentasjon

Spørsmål 4D

Spørsmål: Beskriv hvordan du kan evaluere de ikke-funksjonelle kravene.

Svar del 1: Forutsetninger for å evaluere ikke-funksjonelle krav:

- Kravene må være **målbare** → unngå at utviklere tolker kravene på egen måte
- Definer krav som er direkte målbare
- Eksempel på krav: Et system skal kunne håndtere **200.000** brukere samtidig

Derfor:

- Når det er generelle krav, spesifiser konkrete **metrikker** i evalueringen
- Eksempel på krav: Systemet skal være raskt
- **Metrikk:** En ny side skal laste inn på under **tre sekunder** (tid)

Spørsmål 4D

Spørsmål: Beskriv hvordan du kan evaluere de ikke-funksjonelle kravene.

Svar del 2:

1. Systemet må være raskt - det skal ikke ta mer enn 2 sekunder å laste inn en side
   Test: flere tester ved å måle tid, også under ulike dekningsforhold
2. Systemet må være brukervennlig - en ny bruker skal finne en restaurant på under ett minutt
   Test: brukertest der man undersøker ulike forhold
3. Systemet må være plattformuavhengig- det skal fungere på alle mobile plattformer (android, ios, osv.)
   Test: test systemet på ulike mobile plattformer
4. Systemet skal kunne håndtere mange brukere samtidig - 50 000 brukere samtidig
   Test: stresstest - hvor mye tåler systemet? Når nås bristepunktet?
5. All systemdokumentasjon skal være forståelig - engelsk skal brukes som er felles språk på dokumentasjonen.
   Test: ja/nei-spørsmål, sjekkes direkte.

Spørsmål 5

Spørsmål: Hva vil det si å validere et system og hvorfor er dette viktig?

Svar: Når man validerer et system så sjekker man hvorvidt systemet **faktisk** **møter** brukernes behov.

- Er systemet man har laget systemet man faktisk trenger? Dette er viktig å **avklare** hele veien så man ikke sløser med ressurser
- Det er viktig at dette blir gjort **kontinuerlig** gjennom systemutviklingsprosessen
- Skiller seg fra verifisering: **verifisering** sjekker om man har bygget x riktig, **validering** sjekker om det er riktig å bygge x.

Man kan validere mot en kravspesifikasjon (samt validere en kravspesifikasjon):

- Beskriver kravspesifikasjonen systemet kunden ønsker?

Viktig fordi det koster svært mye å endre et system når det er ferdig

- Et system kan være godt laget, men **ubrukelig** for kunden om det ikke løser kundens behov

Spørsmål 6A

Spørsmål: I smidig utvikling benytter man gjerne brukerhistorier. Hva er en brukerhistorie?

Svar:
En brukerhistorie er en **kort** beskrivelse av en **bruker** i en **brukskontekst**, med hensikt om å klargjøre kravene til et system.

Brukerhistorier beskriver **hva** brukeren ønsker å få ut av systemet og består av ulike elementer: brukerens **rolle**, ønsket **funksjon** og **nytteverdi** av funksjonen

Mal: Som [ROLLE] ønsker jeg [FUNKSJON] for å oppnå [NYTTEVERDI]

Eksempel: Som en bruker ønsker jeg å vurdere en restaurant for at jeg og andre kan få nytte av disse tilbakemeldingene

Spørsmål 6B

Spørsmål: Nevn noen fordeler ved å bruke denne teknikken til å beskrive krav.

Svar: Det er en **enkel** måte å **kommunisere** konteksten systemet skal tas bruk i, 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 kravet → **skjuler** kompleksitet
- Kravene uttrykkes på en kort og konsis måte

Spørsmål 6C

Spørsmål: Drøft utfordringer ved å benytte brukerhistorier beskrevet på lapper på en tavle i store, smidige utviklingsprosjekter.

Svar: Det kan fort bli **uoversiktlig** og dermed vanskelig å prioritere de viktigste oppgavene først

- Det kan lett bli **kaos** med mange lapper
- Lappene kan **forsvinne**
- Man er avhengig av å være tilstede for å se hva som står der
- Kan miste den **helhetlige** forståelsen av det som skal lages
- Kan tolkes på flere måter
- Skjuler kompleksitet → Kan føre til at funksjonen kan være vanskeligere å implementere enn antatt
- Skjuler underliggende krav → Hva er den faktiske betydningen av brukerhistorien?

NB! Det finnes elektroniske løsninger for dette

Spørsmål 6D

Spørsmål: Skriv noen brukerhistorier for appen beskrevet under oppgave 3.

Svar:

- «Som **kunde** ønsker jeg å kunne **se** hva andre har kommentert på restauranter, slik at det blir **lettere** å ta et godt valg.»
- «Som **kunde** ønsker jeg å kunne **sortere** restauranter etter kategori, slik at jeg kan **finne** en restaurant som tilbyr mat jeg liker/tåler.»
- «Som **restauranteier** ønsker jeg at appen skal kunne **vise** menyen til restauranten, slik at kunden får et godt **inntrykk** av hva vi tilbyr.»

Spørsmål 7A

Spørsmål: I systemutviklingsprosjekter med tett kundemedvirkning er det en fare for at kunden blir påvirket av utviklingsteamet og adopterer deres perspektiv. Da kan brukernes behov bli tillagt for liten vekt.

Foreslå tre måter å redusere dette problemet på.

Svar: Unngå at kunden blir påvirket av utviklerne og blir en del av teamet:

- Involvere kunden **kun** der det er nødvendig
- Ha **ulike** representanter for kunden som deltar
- Inkluder brukerne mer i utviklingsprosessen
- Hold tekniske diskusjoner til et **minimum**
- **Prototyp** regelmessig og evaluer – slik vil du se om utviklingen følger brukerkrav og behov.

Spørsmål 7B

Spørsmål: Diskuter fordeler og ulemper med kundeinvolvering.

Svar:
FORDELER:

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

ULEMPER:

- Krever **mye** tid og ressurser av kunde
- Krever at kunde er **tilgjengelig**
- Kan resultere i at man mister fokus fra hva man faktisk skal utvikle

Spørsmål 8

Spørsmål: Forklar hvorfor det er nødvendig med to kravaktiviteter i prosessen for gjenbruksbasert systemutvikling (angitt i figur 2.3 i pensumboken).

Svar:
Gjenbruksbasert systemutvikling (def): Eksisterende programvare brukes på nytt i nye systemer.
Eksempler:

1. Komponentbasert utvikling: Benytter seg av komponenter fra ulike pakker
2. Tjenesteorientert utvikling: Benytter seg av tjenester som finnes på nett/i skyene

TO KRAVAKTIVITETER:
1. Standard kravinnsamling: bestemmer hva som skal lages og hvordan det skal gjøres.
2. Etter å ha undersøkt hva som finnes på markedet modifiserer man kravspesifikasjonen med utgangspunkt i den eksisterende programvare. For ønsket funksjonalitet vil ofte variere fra det som allerede finnes.

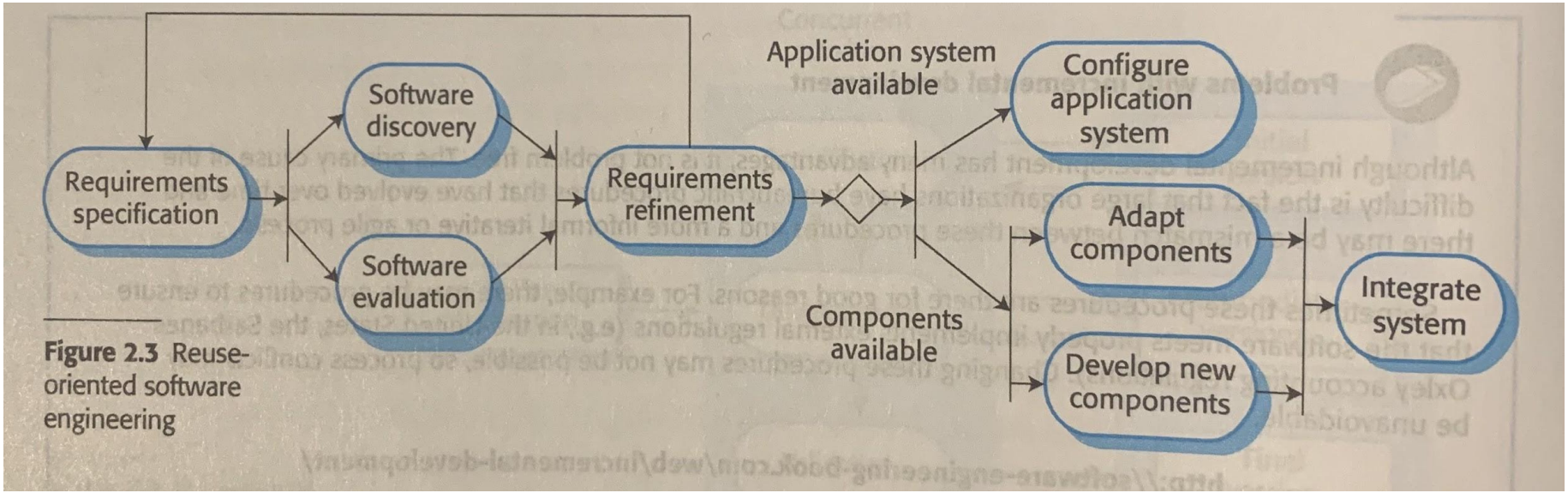


Figure 2.3 Reuse-oriented software engineering

emmatv@uio.no