# Formel use case

Indhold

1. Titel
   1. Skal have et navn
   2. Skal have id
2. Navnet skriver den primære aktørs overordnede handling på bydeform
3. Scope skal være angivet
4. Use Case niveau (Brugermål evt, eller bruges den af andre)
5. Den primære aktør er angivet
6. Interessenter er angivet
7. Interessenter for hver interessent er angivet
8. Forudsætning er angivet
9. Succes guarantee er angivet
10. Hovedscenariet er beskrevet i nummerede trin i nutidsform
11. Variationer fra hovedscenariet er angivet med alternative trin og trigger og henvisning til variations trinnet
12. Hvert trin angivet med en aktør, herunder fx. systemet og en handling
13. Ikke-funktionelle krav er angivet
14. Eksplicitte krav til teknologi er angivet
15. Hyppighed er angivet
16. Domænebetegnelser matcher de, i andre use cases

UML

1. Navnet skal passe til use-case diagrammet
2. Use casens trin matcher et evt. aktivitetsdiagram

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

# 

# 

# Projektplan (i kravfasen)

1. Projektplanen angiver en startdato for projektet.
2. Projektplanen angiver en forventet slutdato for projektet.
3. Projektplanen angiver hver fase.
4. Projektplanen angiver forventet start- og slutdato for hver fase.
5. Projektplanen angiver konkrete og nedbrudte aktiviteter i kravfasen.
6. Projektplanen angiver overordnede aktiviteter for alle faser undtagen kravfasen.
7. Alle aktiviteter er estimerede.
8. Alle aktiviteter er angivet med forventet starttidspunkt.
9. Alle aktiviteter tildelt en eller flere personer som skal udføre aktiviteten.
10. Aktiviteter der kræver særlige ressourcer er eksplicit tildelt disse.
11. Ingen personer eller resurser er dobbeltbookede til samtidige aktiviteter.
12. Projektplanen visualiserer projektet som beskrevet for udviklingsprocessen.
13. Projektplanen angiver den kritiske sti.

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

# 

# 

# Usability goals

1. Usability goals beskriver effectiveness
2. Usability goals beskriver efficiency
3. Usability goals beskriver safety
4. Usability goals beskriver utility/funktioner
5. Usability goals beskriver learnability
6. Usability goals beskriver memorability
7. Krav/beskrivelser er målbare
8. Kravene skal kunne tilbageføres til beskrivelsen i visionsdokumentet
9. Hver krav skal kunne tilbageføres til mindst én use case

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

# 

# Risikoanalyse

1. Risikoanalyse angiver identificerede risici helt konkret og afgørligt.
2. Konsekvensen er beskrevet for hver risiko
3. Sandsynlighed er beskrevet for hver risiko
4. Risikotallet er beregnet for hver risiko
5. Imødegåelsesstrategi er beskrevet for risikoerne med de højeste risikotal
6. Imødegåelsesstrategi er beskrevet for risikoerne med størst konsekvens
7. Imødegåelsesstrategi er beskrevet for risikoerne med meget høj sandsynlighed (80+ %)
8. Revideret sandsynlighed er beskrevet for hver risiko med en imødegåelsesstrategi
9. Revideret konsekvens er beskrevet for hver risiko med en imødegåelsesstrategi
10. Hver imødegåelsesstrategi er konstrueret ud fra RMMM-model
11. Imødegåelsesstrategien er operationelt beskrevet og realistisk (under egen kontrol)
12. Risikoanalysen giver overblik over prioriteringen af samtlige risikoer

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

# 

# 

# Use case-diagram

1. Der er angivet én primær aktør til hver use case
2. Use case-diagram viser interaktion mellem den primære aktør og use cases.
3. Gestaltlove understøtter overblik over systemets funktioner
4. Betegnelser for aktører og use cases skal matche eksisterende use case-beskrivelser
5. Evt. understøttende aktører er angivet
6. Systemgrænse er klart markeret
7. Der er anvendt korrekt UML-notation ([Larman] afs. 6.17)
   1. Aktør
   2. Use case
   3. Kommunikation
   4. Relationer mellem use cases
   5. Systemafgrænsning
8. Primære aktører er angivet til venstre for systemafgrænsningen
9. Understøttende aktører er angivet til højre for systemafgrænsningen
10. Hver use-case har et id
11. Use case-betegnelser angiver den primære aktørs overordnede handling på bydeform
12. Alle termer er en del af problemdomænets særegne terminologi (kundens sprog)

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.



# 

# Visionsdokument

NB: Vi vil gennemgå visionsdokument i uge 6.

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

# 

# 

# Supplerende kravspecifikation (FURPS+)

NB: Vi vil gernnemgå supplerende kravspecifikation i uge 6.

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

# 

# 

# Domænemodel

UML

1. Der er angivet associationer mellem relaterede koncepter med label og evt. læseretning
2. Associationer er angivet uden retning
3. Multiplicitet er angivet i begge retninger for hver association
4. Hver multiplicitet stemmer overens med domænereglerne
5. Attributter er angivet med lower camel-case
6. Navne på koncepter er angivet med upper camel-case
7. Navne på koncepter er angivet i ental
8. Korrekt UML-notation for klassediagram er anvendt
   1. Klasser
   2. Associationer
   3. Multipliciteter
   4. Labels
9. Labels er angivet i nutid
10. Alle associationer, herunder labels og multiplicitet, udtrykker relationen mellem to koncepter på ethvert tidspunkt
11. Læseretnings pile

Indhold

1. Titel og ID
2. Domæne model angiver konceptuelle klasser, dvs. at de beskriver koncepter fra den virkelige verden.
3. Relevante attributter er angivet for hvert koncept
4. Alle attributter kan repræsenteres med simple typer
5. Hver attribut skal beskrive dets koncept
6. De konceptuelle klasser indeholder ikke metoder/operationer
7. Gestaltlove understøtter overblik over sammenhænge mellem koncepterne

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

# 

# 

# Systemsekvensdiagram

1. Hver systemoperation (stimulus) går fra den primære aktør til systemet
2. Systemoperationerne angiver nødvendige parametre
3. Systemoperationer anvender syntaks for synkrone beskeder
   1. Fuldt optrukket linje med lukket, udfyldt pilehoved ved modtageren
4. Hver systemoperations navn beskriver den overordnede handling som operationen udfører
   1. Eksempel på gode metodenavne: <https://youtu.be/AN4NCnc4eZg?t=2m23s>
5. Hver systemoperations navn er skrevet med lower camel-case
6. Al respons fra systemet er vist som returkald
   1. Stiplet linje med åbent pilehoved ved modtageren
7. Systemet er repræsenteret som en “black box”
8. Benyttede variable, fx i guard-udtryk, skal være defineret før de bruges
   1. Tidligere i diagrammet (fx i parametre)
   2. Eller i kendte objekter (fx fra domænemodellen)
9. Systemsekvensdiagrammet afbilleder ét use case-scenario
10. Den primære aktør har en passende rollebetegnelse der matcher med use casen
11. Den primære aktør er vist til venstre for systemet
12. Aktører og system er vist som et objekter
    1. Kolon foran angivelse “type”
13. Evt. understøttende aktører (eksterne systemer) er vist til højre for systemet
14. Diagrammet viser hvad og hvornår der bliver kommunikeret med evt. understøttende aktører
15. Rækkefølgen af systemoperationer afspejler scenariets sekvens i use casen, herunder forekomster loops

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

# 

# 

# Operationskontrakt

1. Operationskontrakten er navngivet efter navnet på systemoperationen uden parametre
2. Operationskontrakten er tildelt et id
3. Den tilknyttede systemoperation er beskrevet med parametre
4. Operationskontrakten skal krydsreferere med mindst en use case.
5. Preconditions beskriver den nødvendige tilstand for at systemoperationen kan udføres
6. Preconditions er angivet i nutid
7. Preconditions er angivet i termer af domænemodellen
   1. Eksistensen af domæneobjekter+
   2. Tilstanden af attributter i domæneobjekter
   3. Associationer mellem domæneobjekter
8. Postconditions beskriver den konsistente tilstand efter udførsel af systemoperationen
9. Postconditions er angivet i førdatid
   1. Eksempel på førdatid: En instans af hold *er oprettet*
10. Postconditions er angivet i termer af domænemodellen
    1. Oprettelse og nedlæggelse af domæneobjekter
    2. Ændring af attributter i domæneobjekter
    3. Oprettelse og nedlæggelse af associationer mellem domæneobjekter
11. Alle referencer til domæneobjekter sker til veldefinerede referencer

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

# 

# 

# Test case

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

* Testsuiten er navngivet så det stemmer overens med operations kontrakten
* Testsuiten har et ID
* Testsuiten skal indeholde brugte systemoperationer
* Hver Test case har et ID
* Input indeholder en metode med evt. parametre
* Metoden i input stemmer overens med metoden i operations kontrakten
* Output indeholder hvad metoden returnerer
* Forudsætninger er angivet
* Evt. forkortelser er tydeligt angivet
* Evt. fejlmeddelelser stemmer overens med domænemodellen

# 

# 

# Sekvensdiagram

1. Diagrammet viser kommunikationen mellem forskellige ifm. ét operationskald inden for ét lag
2. Der er angivet et navn og id for diagrammet
3. Hver lifeline skal afspejle det pågældende objekts levetid
   1. Lifeline starter på det tidspunkt hvor objektet bliver skabt
   2. Lifeline slutter på det tidspunkt hvor objektet bliver destrueret
4. Diagrammet viser et korrekt kontrolflow (bolde)
5. Korrekt UML-notation
   1. Skabelse af nye objekter (stiplet streg med lukket, udfyldt pilehoved ved det nye objekt)
   2. Returkald (stiplet streg med åbent pilehoved ved det modtagende objekt)
   3. Synkrone operationskald (fuldt optrukket streg med lukket, udfyldt pilehoved ved det modtagende objekt)
   4. Asynkrone operationskald (fuldt optrukket streg med åbent pilehoved ved det modtagende objekt)
   5. Execution specification dækker kun lifeline på tidspunkter hvor det pågældende objekt har kontrollen eller venter på at få kontrollen retur
   6. Evt. action box er placeret på lifeline
   7. Angivelse af objekter repræsenterende en klasse er noteret med metaclass keyword
   8. Objekter er angivet med en kasse indeholdende objektets evt. reference (kan være blank) og objektets type (reference : type)
   9. Lifelines er angivet med en stiplet, lodret streg
   10. Kontrol-frames har et boolsk guard-udtryk
   11. Found beskeder er angivet med udfyldt cirkel modsat pilehovedet
   12. Lost beskeder er angivet med udfyldt cirkel ved pilehovedet
6. Korrekt angivelse af besked eller værdi
   1. Operationskald ved videregivelse af kontrol
   2. Værdi med evt. typeangivelse ved tilbagegivelse af kontrol
7. Operationskald foretages kun til objekter som er kendte af det pågældende objekt
   1. Kendt som attribut
   2. Kendt som parameter
   3. Har selv skabt objekt
8. Forstyrrende lange “tekster” er angivet med note-notation
9. Objekters referencenavne er skrevet i lower camel-case
10. Objekters typer er skrevet i upper camel-case
11. Operationernes betegnelser skal matche med en evt. operationskontrakt samt øvrige designartefakter
12. Evt. ref-frames har en matchende sd-frame (i et andet diagram)
13. Operationer angiver nødvendige parametre
14. Operationer og parametre er skrevet i lower camel-case
15. Loop-frames angiver (evt. ved action box) progression i kørselsbetingelsen frem mod terminering

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

**Pile:**

1. Skabelse af nye objekter (stiplet streg med lukket, udfyldt pilehoved ved det nye objekt)
2. Returkald (stiplet streg med åbent pilehoved ved det modtagende objekt)
3. Synkrone operationskald (fuldt optrukket streg med lukket, udfyldt pilehoved ved det modtagende objekt)
4. Asynkrone operationskald (fuldt optrukket streg med åbent pilehoved ved det modtagende objekt)
5. Found beskeder er angivet med udfyldt cirkel modsat pilehovedet
6. Lost beskeder er angivet med udfyldt cirkel ved pilehovedet

**Objekter:**

1. Hver lifeline skal afspejle det pågældende objekts levetid

a. Lifeline starter på det tidspunkt hvor objektet bliver skabt

b. Lifeline slutter på det tidspunkt hvor objektet bliver destrueret

1. Evt. action box er placeret på lifeline
2. Angivelse af objekter repræsenterende en klasse er noteret med metaclass keyword
3. Objekter er angivet med en kasse indeholdende objektets evt. reference (kan være blank) og objektets type (reference : type)
4. Lifelines er angivet med en stiplet, lodret streg
5. Objekters referencenavne er skrevet i lower camel-case
6. Objekters typer er skrevet i upper camel-case

**Operation:**

1. Execution specification dækker kun lifeline på tidspunkter hvor det pågældende objekt har kontrollen eller venter på at få kontrollen retur

a. Execution specification kan opstå ved operationskald

b. Execution specification slutter når en operations returnere

1. Korrekt angivelse af besked eller værdi

a. Operationskald ved videregivelse af kontrol

b. Værdi med evt. typeangivelse ved tilbagegivelse af kontrol

1. Operationskald foretages kun til objekter som er kendte af det pågældende objekt

a. Kendt som attribut

b. Kendt som parameter

c. Har selv skabt objekt

1. Operationernes betegnelser skal matche med en evt. operationskontrakt samt øvrige designartefakter
2. Operationer angiver nødvendige parametre
3. Operationer og parametre er skrevet i lower camel-case

**Frames:**

1. Kontrol-frames har et boolsk guard-udtryk
2. Evt. ref-frames har en matchende sd-frame (i et andet diagram)
3. Loop-frames angiver (evt. ved action box) progression i kørselsbetingelsen frem mod terminering

**Konsistens:**

1. Operationernes betegnelser skal matche med en evt. operationskontrakt samt øvrige designartefakter
2. Operationernes metoder og navne skal kunne findes i et tilhørende klassediagram, som har udgang i den samme operationskontrakt som sekvensdiagrammet.
3. sekvensdiagrammet har en metode, som stemmer overens med en systems operation fra en system sekvensdiagrammet fra samme use case (med de nødvendige parametre, hvis de findes)

**Navngivning:**

1. Forstyrrende lange “tekster” er angivet med note-notation
2. Der er angivet et navn og id for diagrammet

**Generelt:**

1. Diagrammet viser kommunikationen mellem forskellige objekter ifm. ét operationskald inden for ét lag
2. Diagrammet viser et korrekt kontrolflow (bolde)
3. Hvis found-massage, er en system operation, så udfører den forskellige kriterier over flere use cases

# 

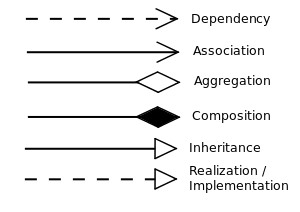
# 

# Klassediagram

**Generelt:**

1. Navne på klasser og metoder matcher med andre designartefakter
2. Klassediagrammet har et beskrivende navn og et id
3. Abstrakte klassenavne er angivet med kursiv
4. Klassens navn er angivet med upper camel-case
5. Constructors er angivet med keyword og understregning
6. Gestaltlove understøtter overblikket over diagrammet

**Klasser:**

1. Klassenavne matcher (i høj grad) med domænemodellens koncepter (low representational gap)
2. Klasser har attributter og metoder angivet i compartments
3. Attributter er angivet med lower camel-case og med type
4. Evt. constraints for attributter er angivet med krøllede parenteser (krøllede parenteser : { })
5. Metoder er angivet med lower camel-case
6. Abstrakte metoder er angivet med kursiv
7. Har man flere klassediagrammer - hvor klasserne har andre/flere metoder eller attributter som bruger i de diagrammer, som er irrelevante for nuværende klassediagram, så tilføjer man ikke disse attributter/metoder - man skriver blot ‘...’ som betegnelse for, at der er mere til klassen, men det benyttes ikke her.
8. Attributter og metoder er markeret med visibility
   1. public (+, kan udelades for metoder)
   2. private (-, kan udelades for attributter)
   3. protected (#)
   4. default (~)
9. Evt. parametre til metoder er separeret med komma
10. Returtype er angivet for alle metoder undtagen metoder med returtypen void
11. Statiske metoder er understreget
12. Interfaces er angivet med keyword

**Relationer / pile:**

1. Relationer viser sammenhængen mellem klasser
   1. dependency (stiplet streg med åbent pilehoved ved klassen der er kendt af den anden)
   2. association (fuldt optrukket streg (evt.) med åbent pilehoved ved klassen der er kendt af den anden)
   3. aggregation (fuldt optrukket streg med uudfyldt diamant ved klassen der kender den anden)
   4. composition (fuldt optrukket streg med udfyldt diamant ved klassen der kender den anden)
   5. nedarving (fuldt optrukket streg med lukket uudfyldt pilehoved ved superklassen)
   6. implementation (stiplet streg med lukket uudfyldt pilehoved ved interfacet)
2. Multiplicitet er angivet for hver relation
3. Der kan evt. angives labels til associationerne

### 

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

* Objecter (titel og liste af metoder)
* Pile/Streger (korrekt type)
* Der skal være angivet korrekt notation. AAD
* Der skal være angivet labels
* Klassediagram angiver alle klasser og interface i systemet.
* Der er angivet associationer mellem klasser.
* Metoderne angiver nødvendige parametre.

public

# 

# 

# Datamodel

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

# Aktivitetsdiagram

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

1. Der er brugt korrekt UML Notation
   1. Aktivitetsdiagrammet startes med en Initial Node
   2. Aktivitetsdiagrammet sluttes med en Final Node
   3. Decisions er angivet med diamantformede symboler
   4. Merges er angivet med diamantformede symboler
   5. Actions er angivet med afrundede rektangulære kasser
   6. flows er angivet med en fuldt optrukket pil med åbent pilehovede
   7. Forks er angivet med en sort tynd rektangel
   8. Joins er angivet med en sort tynd rektangel
2. Aktivitetsdiagrammet matcher en Use Case

# Kode

### Forslag til yderligere review-punkter

Skriv dine forslag herunder. Forslag må først tilføjes til den officielle liste efter fælles accept.

**Package**

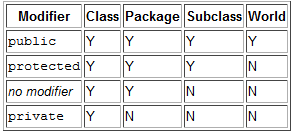
1. Navngivning står med lower camelcase

**Klasser**

1. Navngivning står med uppercamelcase
2. Er acces levels (private, public, protected) brugt så det passer med klassens brug
3. Løser klassen opgaver som ikke hører til dens primære hovedfunktion
4. Ingen:
   1. unødvendige imports
   2. ubrugte metoder
   3. udkommenteret linjer kode som ikke bliver brugt
5. Klassens størrelse er overskuelig (hold den gerne under 150 linjer). Hvis der er for mange metoder, bør klassen deles op.
   1. Den kan evt. deles op i en subklasse
6. Har klassen high cohesion (Høj samhørighed)

**Metoder**

1. Navngivning står med lower camelcase
2. Navngivning på metoderne er tydelig og visende for hvad metodens funktionalitet er
3. Fornuftig brug af service metoder
4. Lad ikke metoder være længere end 10 linjer (gerne hold den på 8)
5. Er acces levels (private, public, protected, default) brugt så det passer med metodens brug



**Variable navne**

1. Navngivning står med lower camelcase
2. Er acces levels (private, public, protected, default) brugt så det passer med variablernes brug

**Generelt**

1. Fornuftig brug af comments.
   1. Ved kode kompliceret kode.
   2. Ingen unødvendige kommentare.
2. Fornuftig brug af linjeskift (fx efter statements og for at skabe overblik i koden)
3. Er der et godt høj-niveau design
   1. arkitektoniske faktorer der især vedrører kobling og samhørighed, herunder (tegn på) anvendelse af designmønstre
4. Er der et godt lav-niveau design
   1. Designmønstre, læsevenlighed osv.

**Test**

1. Er størstedelen af koden dækket af test
   1. prioriter avancerede metoder, hvis alt ikk er dækket
2. I brug af test, bruges mock-klasser fornuftigt