

# Eksamen INF5120 06.06.2005

Et løsningsforslag

# Oppgave 1

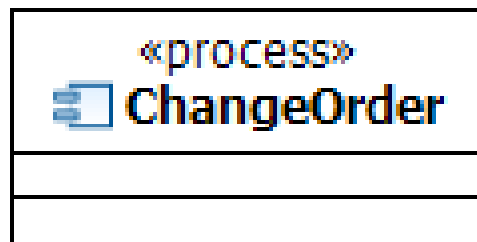
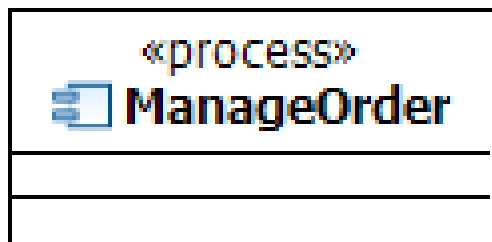
## a) Business Model

Oppgaven spør om en business model for samhandlingen mellom Buyer og Seller, og det er da viktig å ikke modellere alt det andre!!!

Man har normalt litt dårlig tid på eksamen (!) og regn ikke med å få tid til å lage fullstendige modeller. Det blir dermed viktig å velge ut de modellene man mener er mest hensiktsmessige for å vise det oppgaven ber om.

# Oppgave 1 a) forts.

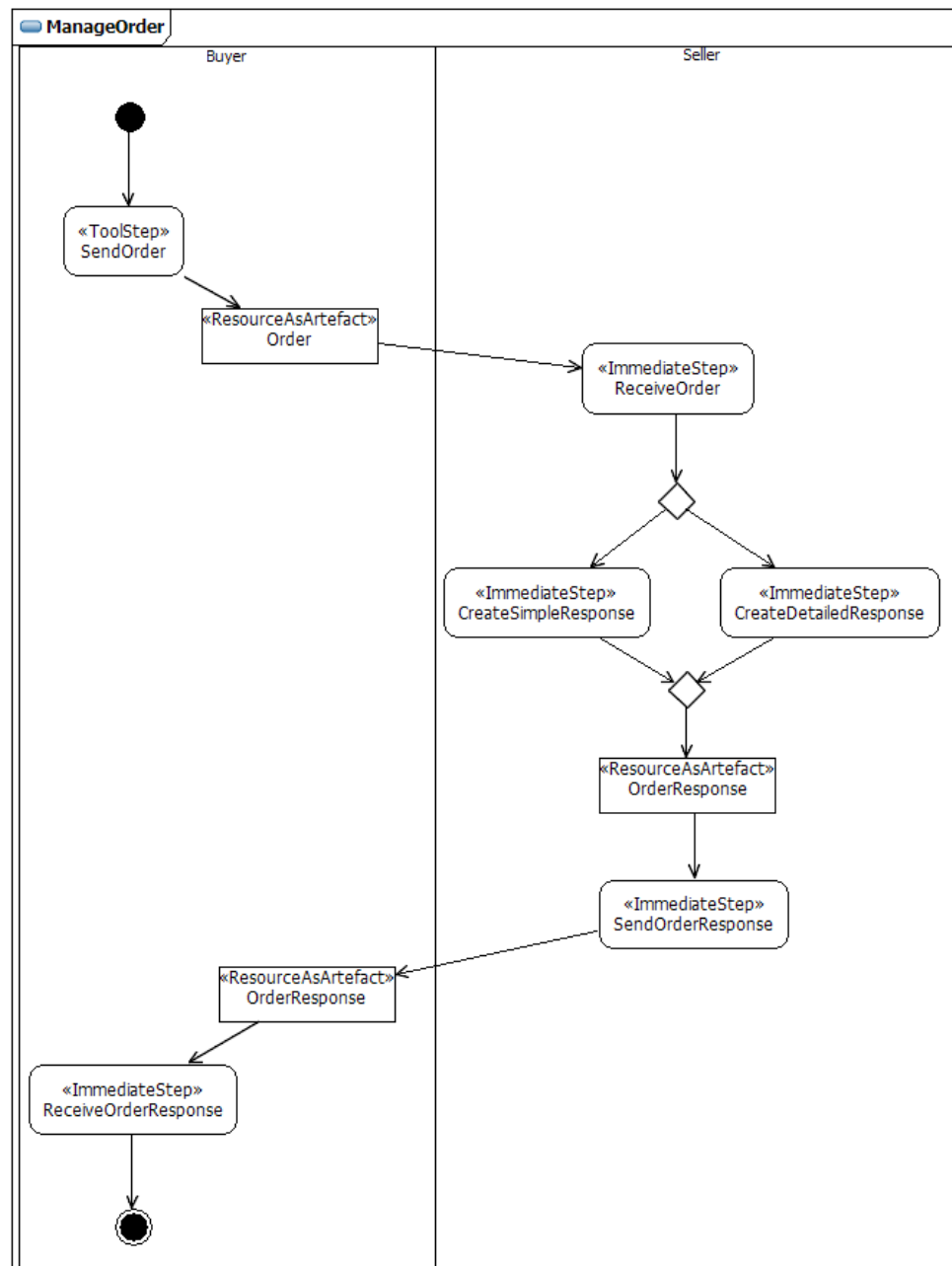
Siden det er samhandlingen mellom Buyer og Seller som er interessant definerer jeg business processes som støtter opp om denne samhandlingen, og modellerer disse på et høyere detaljnivå ved å bruke aktivitetsdiagram. Har man god tid er det naturlig å også definere noen mål man kan knytte disse prosessene til.



# Oppgave 1 a) forts.

Hvordan en ordre skal behandles er tekstlig beskrevet i oppgaveteksten, og blir derfor relativt enkelt å modellere.

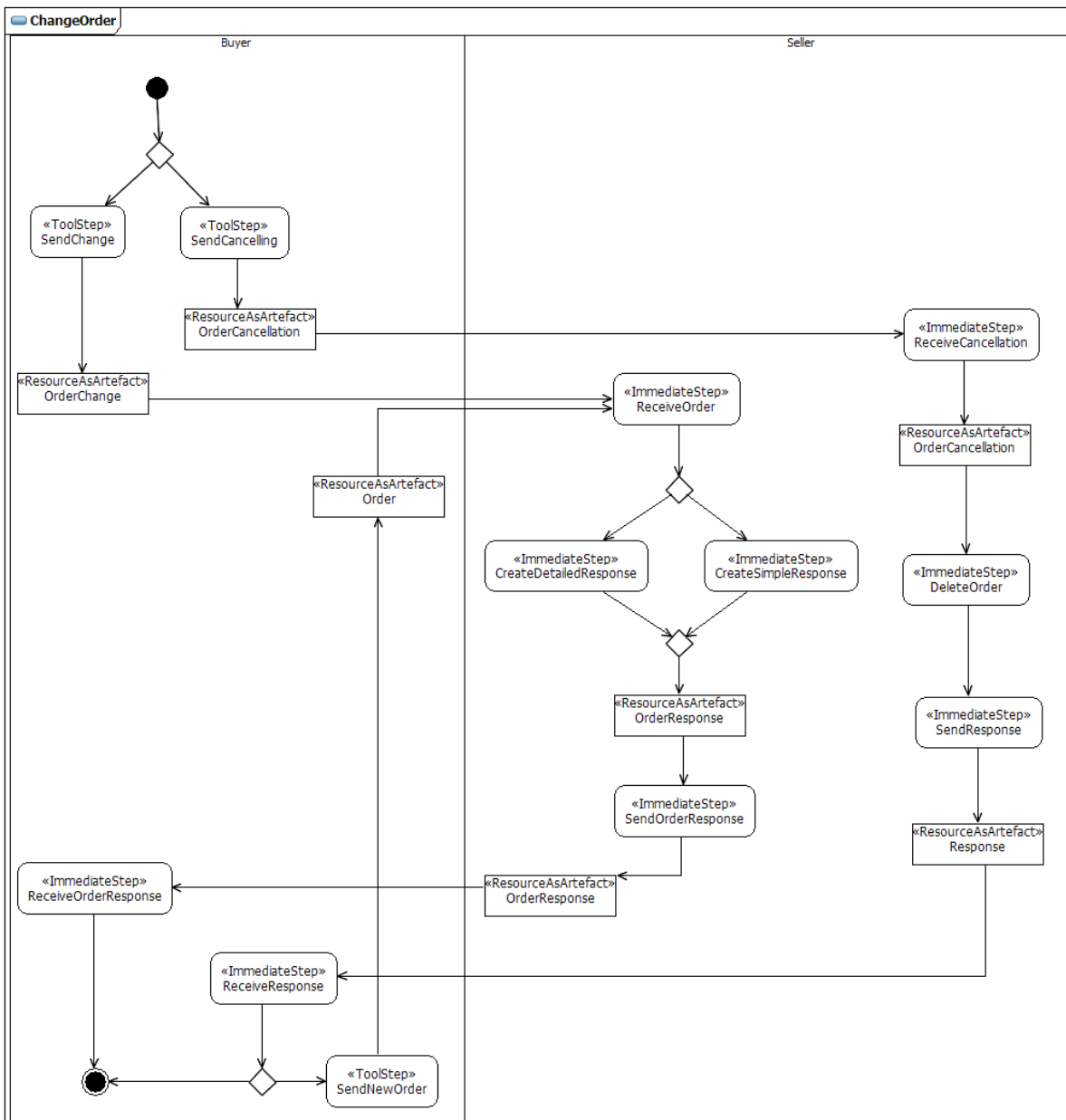
Buyer sender en ordre til Seller som mottar denne, genererer en tilbakemelding som Seller sender til Buyer.



# Oppgave 1 a) forts.

Proessen å endre en ordre er også beskrevet i oppgaveteksten, og prosessen ser kanskje mer kompleks ut enn den i virkeligheten er.

Denne prosessen har to ulike løp; Buyer kan enten slette en ordre for deretter å sende inn en ny til Seller, eller Buyer kan sende inn en endring på en eksisterende ordre til Seller.



# Oppgave 1

## b) Architecture Model

Oppgaven ber om en arkitekturmodell for den delen av systemet som har med Seller å gjøre. Også her er det viktig å kun fokusere på Seller, og ikke på de andre aktørene, Buyer og Receiver.

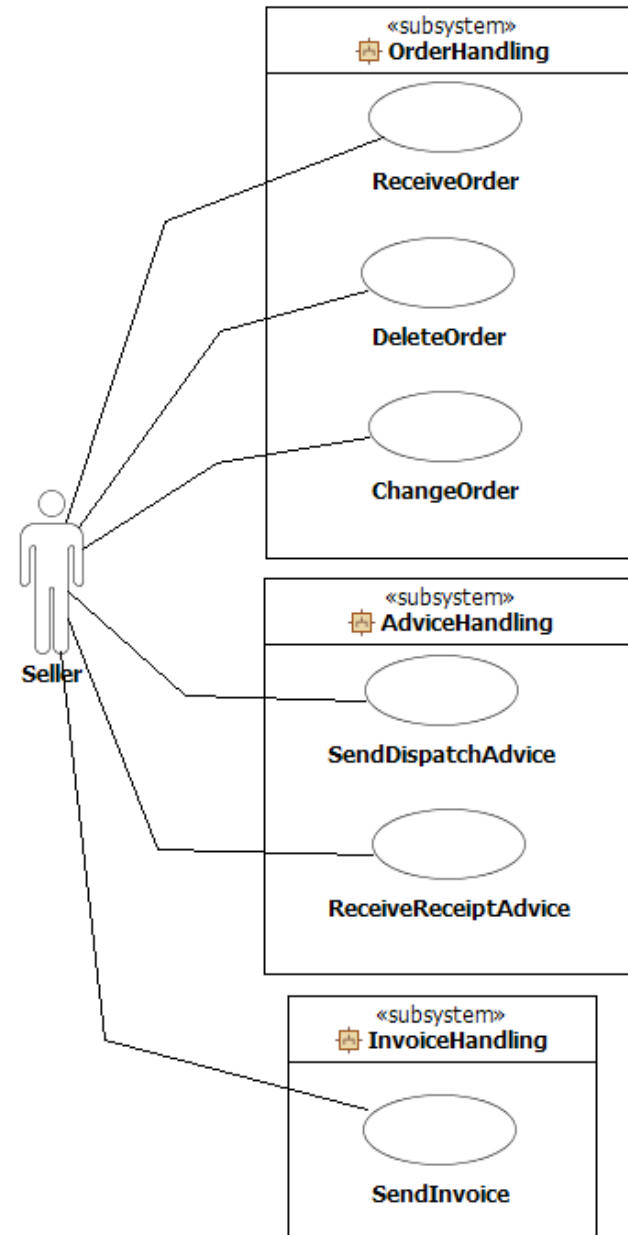
Man vil heller ikke her få tid til å modellere alle modellene for å komme frem til en arkitekturmodell, og man må derfor velge fornuftige kravmodeller og arkitekturmodeller. En slik oppgave gis typisk for å sjekke om man forstår sammenhengen mellom de ulike modelltypene i COMET, og hvordan de ulike modellene fungerer som input til hverandre.

# Oppgave 1 b) forts.

Siden det i denne oppgaven bes om en arkitekturmodell for Seller sitt system er det viktig å fokusere KUN på denne aktøren. For å få frem dette fokuset velger jeg å modellere et Use Case Diagram med Seller som eneste aktør, og deretter gruppere diagrammet med hensyn på fornuftige subsystem, detaljere viktige Use Cases med sekvensdiagram, og lage en Component Structure Model og en Component Interface Model og om man får tid, en Component Information Model.

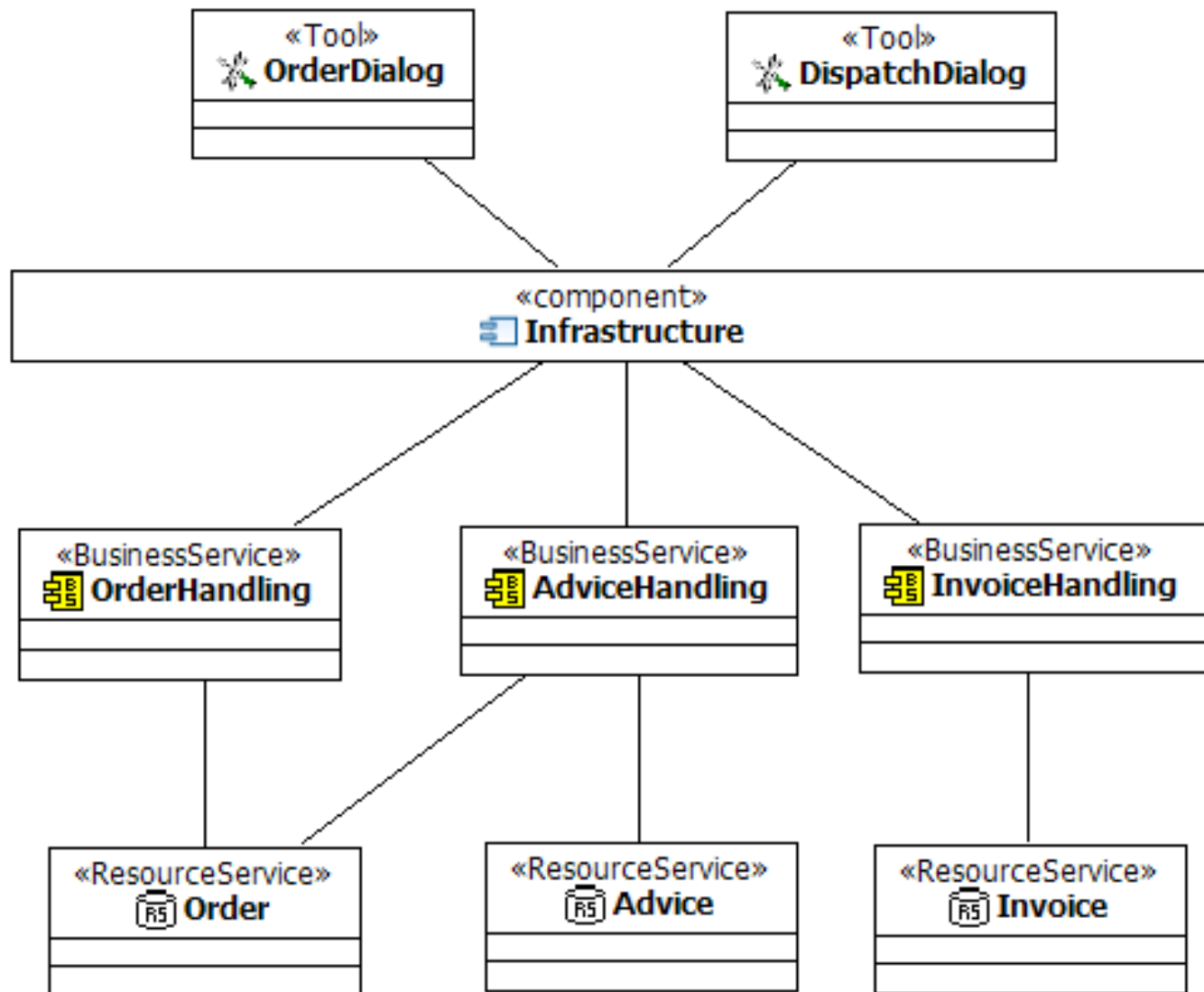
# Oppgave 1 b) forts.

Her er Seller eneste aktør, fordi det er denne aktøren vi skal lage arkitekturmodellen for. Seller bruker i alt seks Use Cases som er gruppert etter ansvarsområder i systemet; Order, Advice og Invoice.



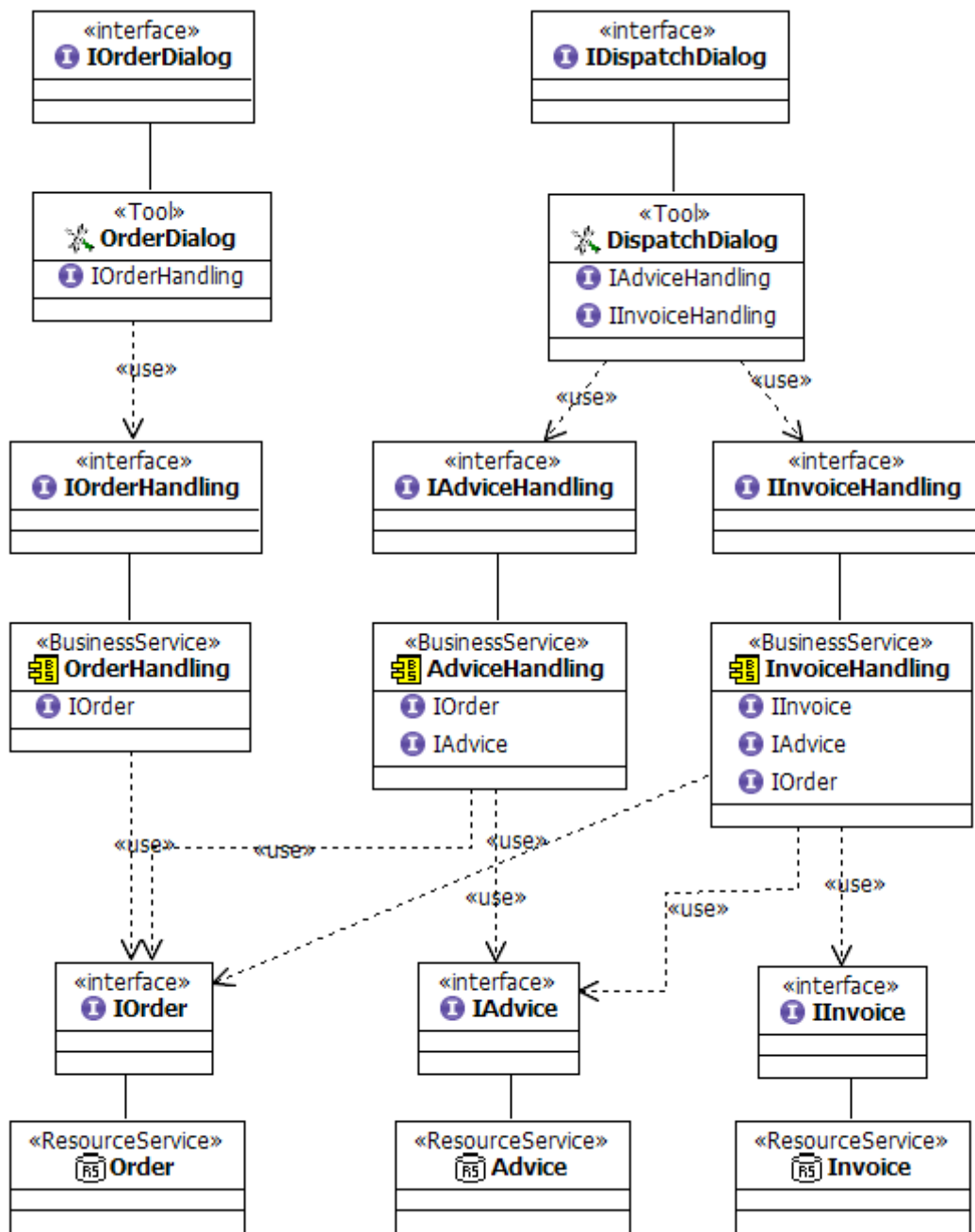
# Oppgave 1 b) forts.

Dette er en strukturmodell av arkitekturen med utgangspunkt i det grupperte Use Case diagrammet. I tillegg introduserer jeg to verktøy/brukergrensesnitt, en infrastruktur og tre datalagere. Disse komponentene med grensesnitt beskrives ytterligere i figuren på neste side, Component Interface Modellen.



# Oppgave 1 b) forts.

Denne modellen beskriver mye av det samme som Component Structure modellen, men i en Component Interface Model viser man i tillegg grensesnittene i arkitekturen, og hvilke komponenter som benytter seg av hvilke grensesnitt. Denne modellen vil derfor fungere som input til sekvensdiagrammene.



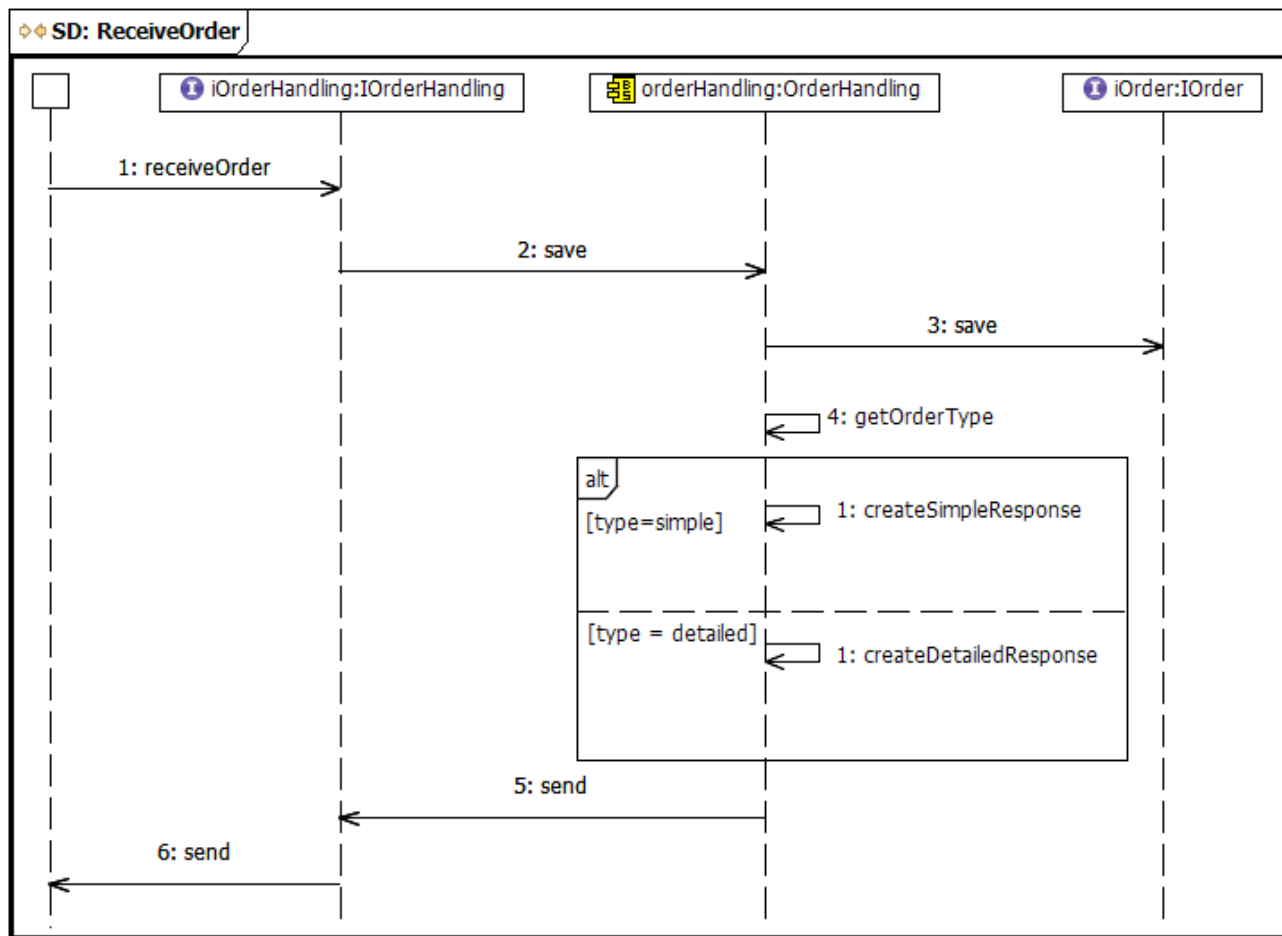
# Oppgave 1 b) forts.

Når jeg nå skal modellere sekvensdiagram er det lurt å velge de sekvensdiagrammene som man anser å være de viktigste i systemet, og hvor man viser at man bruker de andre modellene man har laget.

Jeg velger å modellere sekvensdiagram for ReceiveOrder, DeleteOrder og ChangeOrder, da jeg anser ordrehåndtering som Sellers viktigste oppgave.

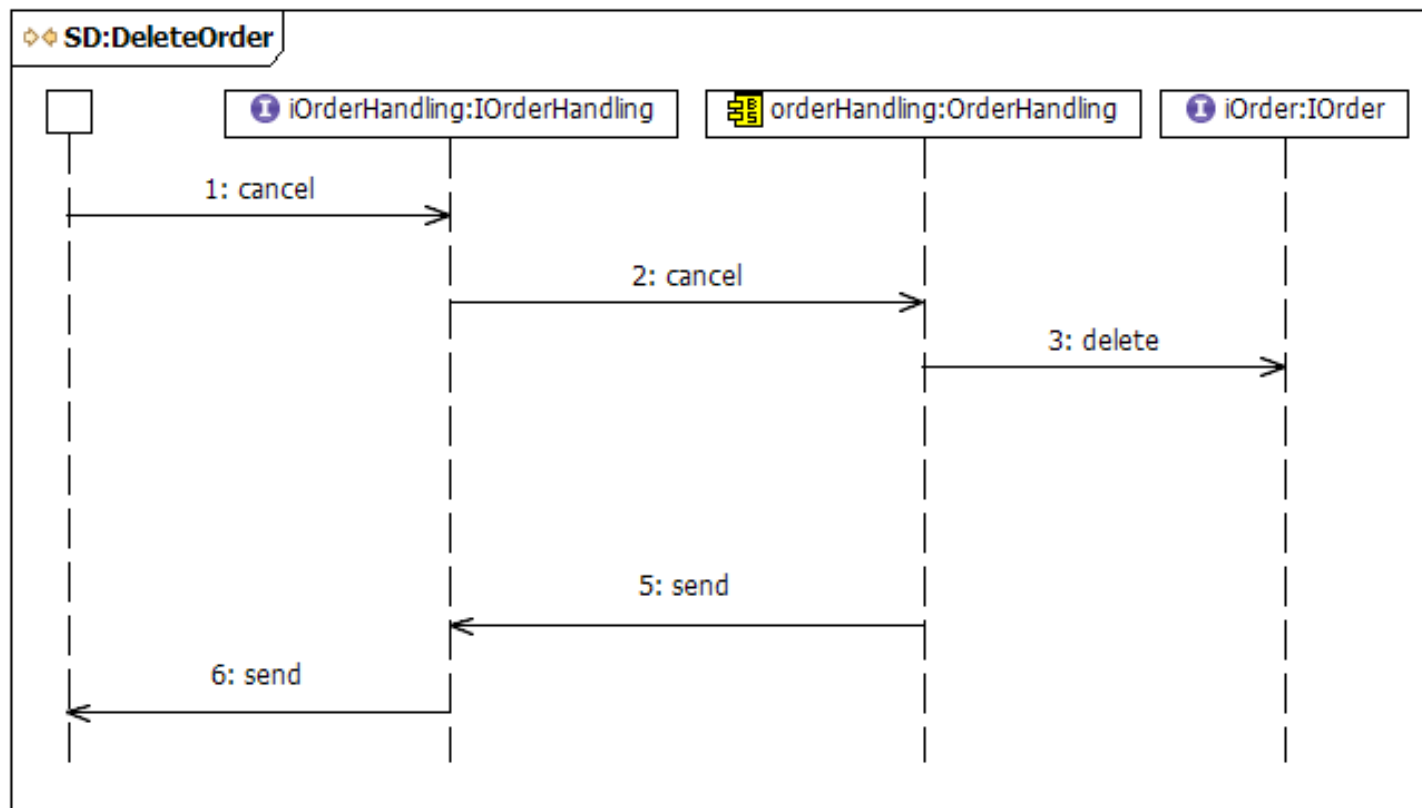
# Oppgave 1 b) forts.

Dette sekvensdiagrammet beskriver interaksjonen i systemet når systemet mottar en ny ordre.



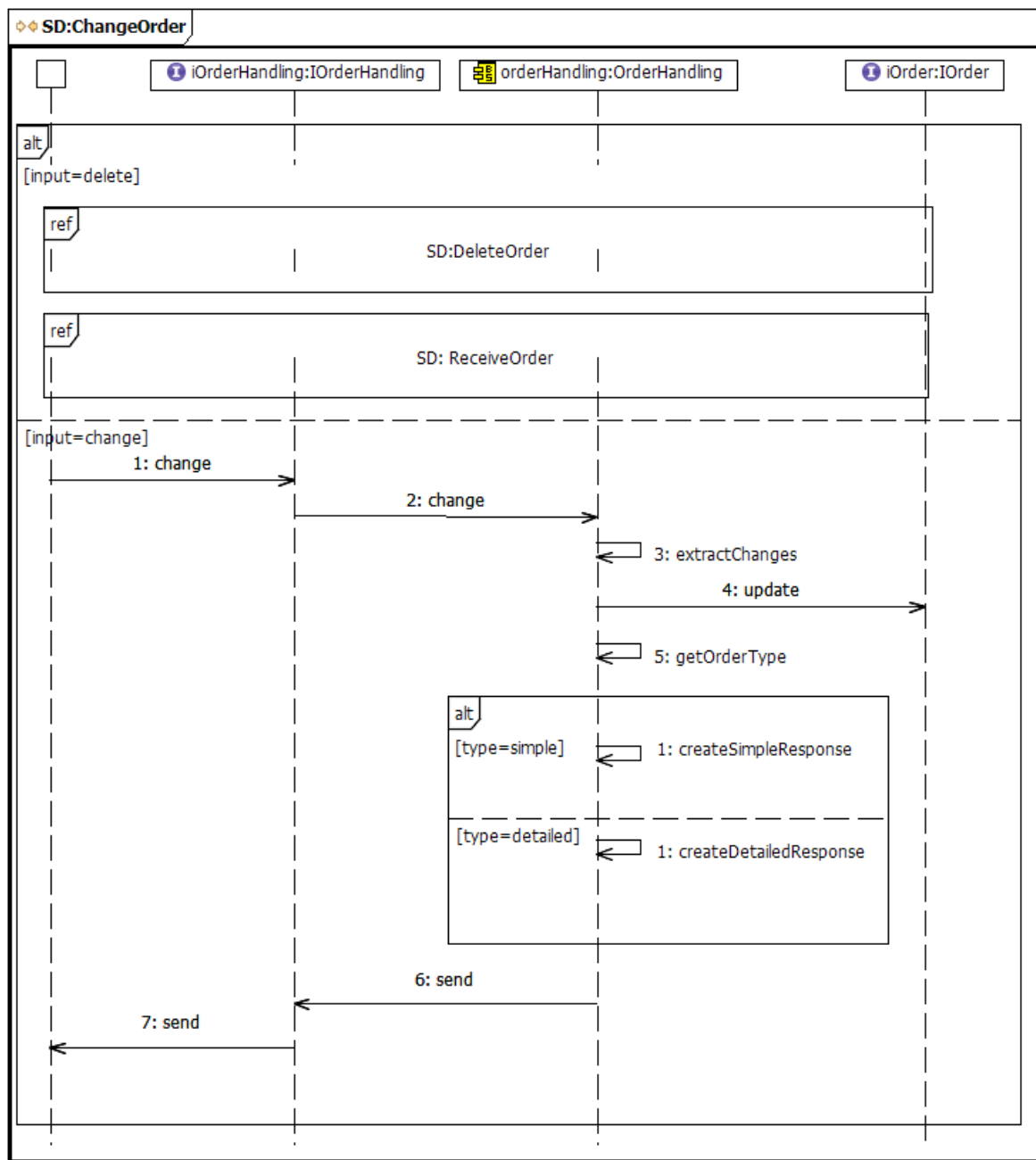
# Oppgave 1 b) forts.

Dette sekvensdiagrammet beskriver interaksjonen i systemet når systemet sletter en ordre.



# Oppgave 1 b) forts.

Dette sekvensdiagrammet beskriver interaksjonen i systemet når systemet skal endre en ordre. Dette skjer enten ved at Buyer sletter en ordre (referanse til SD:DeleteOrder) og deretter sender inn en ny ordre (referanse til SD:ReceiveOrder), eller Buyer sender inn en endring i en eksisterende ordre.



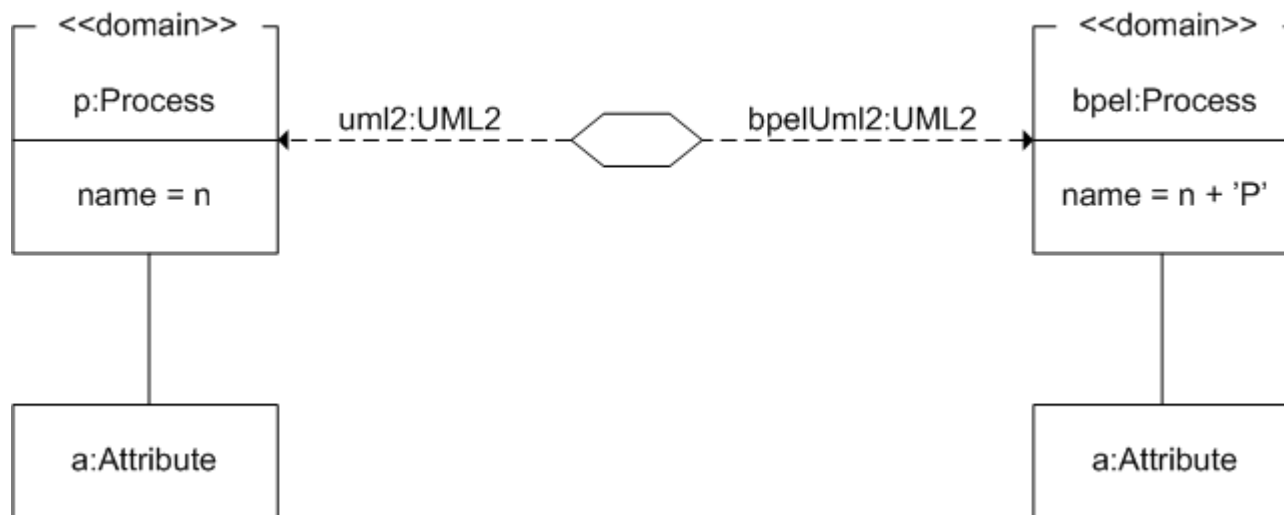
# Oppgave 2

- a) I denne oppgaven skal man beskrive og illustrere hvordan man kan definere transformasjonsregler med ATL fra en plattform uavhengig modell (PIM) til plattform spesifikke modeller (PSM) for dokumenter (XML), grensesnitt (WSDL) og prosess (BPEL).

Dette betyr IKKE at man skal skrive alle transformasjonsreglene i ATL (takk og pris...). Man velger typisk å kun beskrive en mapping mellom to klasser (én source og én target). Oppgaveteksten sier imidlertid også at man kan illustrere fremgangsmåten med en fin figur.

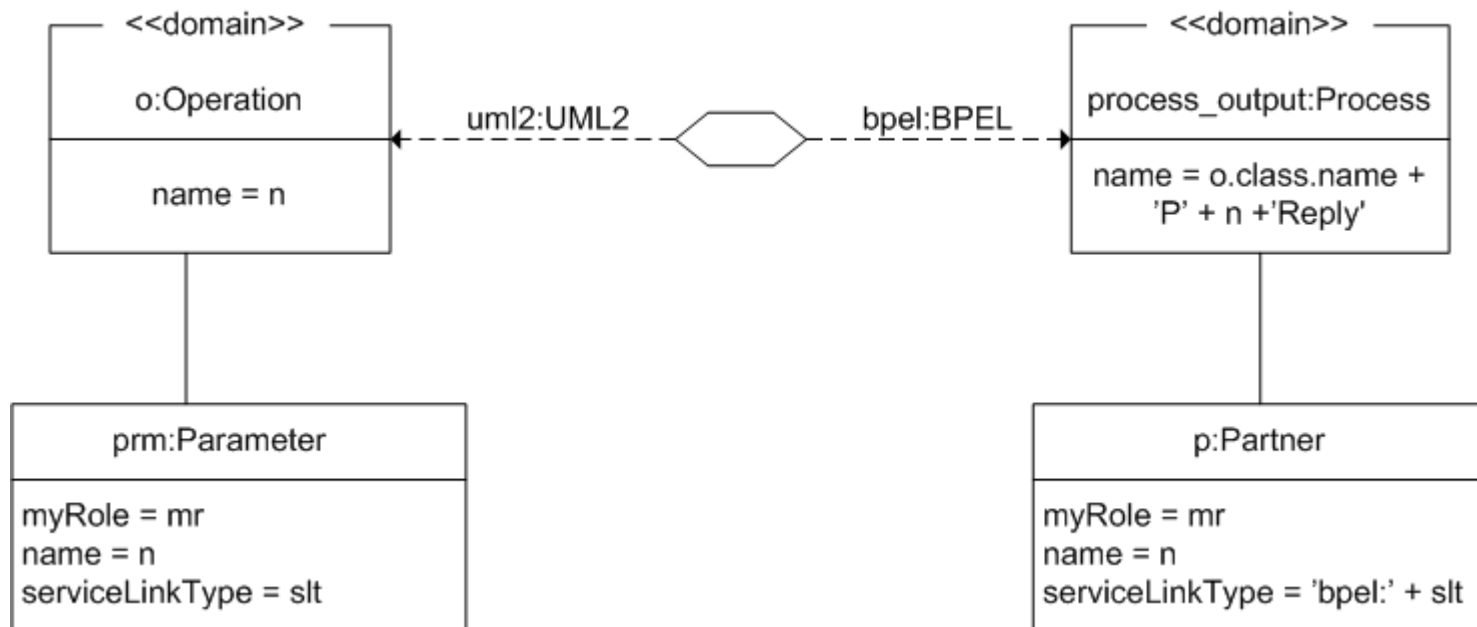
# Oppgave 2 a) forts.

Figuren under viser en mulig mapping mellom en UML PIM modell til en UML PSM modell. I dette tilfellet er PSM modellen en modell av BPEL.



# Oppgave 2 b)

Figuren under viser en mulig mapping mellom en UML PSM modell til BPEL-kode



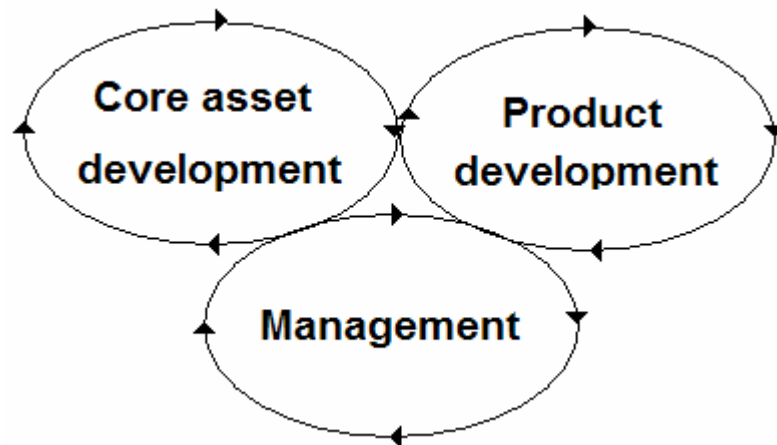
# Oppgave 3

En produktlinje kan beskrives som en mengde produkter som markedsmessig har fellestrekk og som til en viss grad likner på hverandre, men som ikke nødvendigvis har tilsvarende tekniske fellestrekk.

En produktfamilie har i tillegg til markedsmessige likheter også en felles arkitektur, som i større grad muliggjør gjenbruk på tvers av produktene.

# Oppgave 3 forts.

For å utvikle produkter i en produktlinje og/eller en produktfamilie må man i hovedsak ta hensyn til tre aspekter:



hvor core assets i figuren henviser til fellestrekkene for produktene

# Oppgave 3 forts.

Dersom man ser for seg flere varianter av et slikt ordre/faktureringsystem åpner man for å lage en produktfamilie (og kanskje en produktlinje) av systemer. For at dette skal kunne la seg gjøre er det viktig å modellere med et relativt høyt abstraksjonsnivå, slik at komponentene i en modell enkelt kan gjenbrukes for å bygge nye produkter.

Når man skal lage slike variasjoner over produkter blir det viktig å identifisere fellestrekkene disse produktene skal ha, i tillegg til å identifisere forskjellene og hva produktene skal kunne variere over.