Norges teknisk-naturvitenskapelige universitet
Institutt for telematikk

EKSAMENSOPPGAVE I TTM4115 – SYSTEMERING AV DISTRIBUERTE SANNTIDSSYSTEMER LØSNINGSFORSLAG

Faglig kontakt under eksamen:	Rolv Bræk
Tlf.:	415 44 605
Eksamensdato:	04. juni 2007
Eksamenstid:	09:00-13:00
Studiepoeng:	7,5 SP
Tillatte hjelpemidler:	A: Alle trykte og håndskrevne hjelpemidler tillatt. Alle kalkulatorer tillatt
Språkform:	
Antall sider bokmål:	2
Antall sider nynorsk:	2
Antall sider engelsk:	2
Antall sider vedlegg:	4
Sensurdato[1]**:**	**26. juni 2007**

[1] Merk! Studentene må primært gjøre seg kjent med sensur ved å oppsøke sensuroppslagene. Evt. telefoner om sensur må rettes til sensurtelefonene. Eksamenskontoret vil ikke kunne svare på slike telefoner.

Bokmål

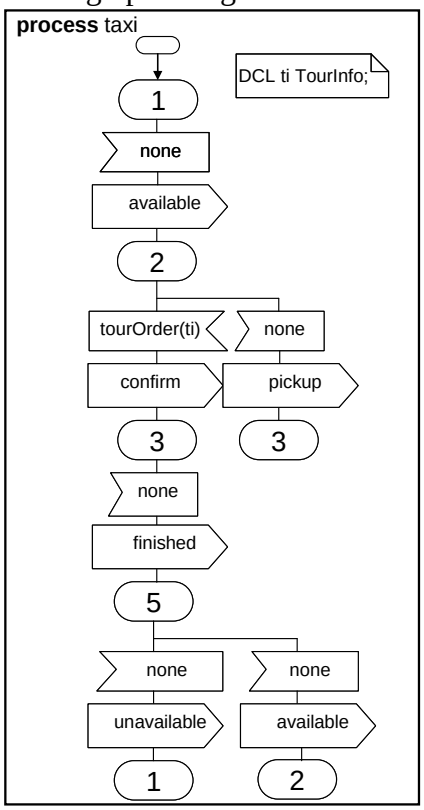
Oppgavene referer seg til systemet som er beskrevet i vedlegg. Studer vedlegget først.

(Eksamen utgjør 75% av sluttkarakteren. Poeng på hvert punkt 0-10)

Oppgave 1. (25%) taxiInterface

Figur 5 viser en ufullstendig beskrivelse av *taxiInterface* hvor bare den ene rollen (*td*) er beskrevet med en SDL prosessgraf.

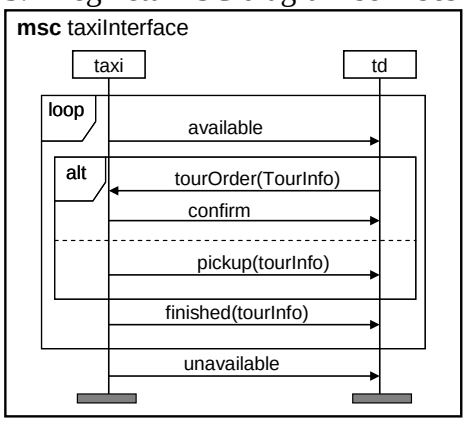
1. Forklar hva signalet *none* formelt betyr i SDL og hva det benyttes til å representere i en rolleoppførsel.
   "None" representerer en spontan trigging av en transisjon som foregår uavhengig av bestemte inputsignal. I en rolleoppførsel representerer none et signal på et skjult grensesnitt. Vi har bare berørt none i forbindelse med roller, så det er nok å svare at det representerer en input på et skjult grensesnitt. -1 for ikke å nevne SDL betydningen spontan
2. Tegn prosessgrafen for *taxi* rollen slik at den er kompatibel med den gitte *td* rollen.



Her har vi bare speilet rollen uten å ta hensyn til kollisjoner. Det er ok. *Vi krever at none skal brukes for å trigge transisjoner, selv om det ikke er sagt i oppgaven.-2. Uten none er det ikke korrekt SDL. Vi ba om rolleoppførsel, så det er ikke helt riktig å ta med interaksjon med drosjesjåføren -1, -2. Vi krever også skille mellom ledig og opptatt tilstand, dvs 1 og 2. Trekk -2 for dette hvis ikke kompatibel med td, -1 ellers.*

Grov sdl feil -2, liten feil -1 Glemt DCL ti -1. Ingen kjøretid/ventetilstander -1. Manglende logikk for å skiller available, unavailable: -1

3. Tegn ett MSC diagram som beskriver alle interaksjonene i *taxiInterface*.



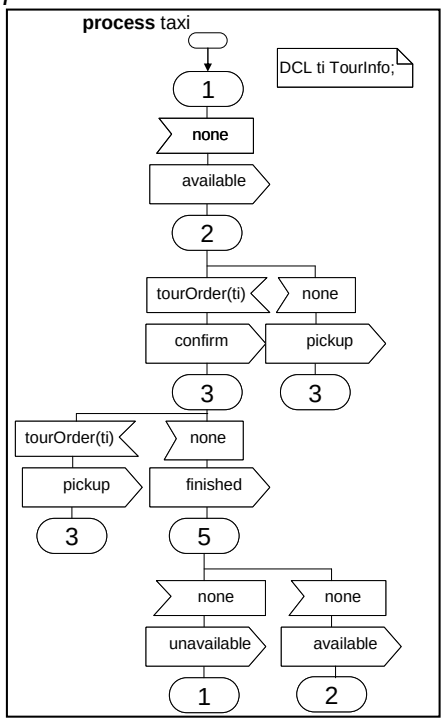
Krever her at man skiller available og unavailable tilstandene som i oppgaven(-1 hvis allerede trukket under pkt 1.3, -2 ellers. Feil sekvens available -2. Mangler loop -1. Brukt mer enn ett diagram -2

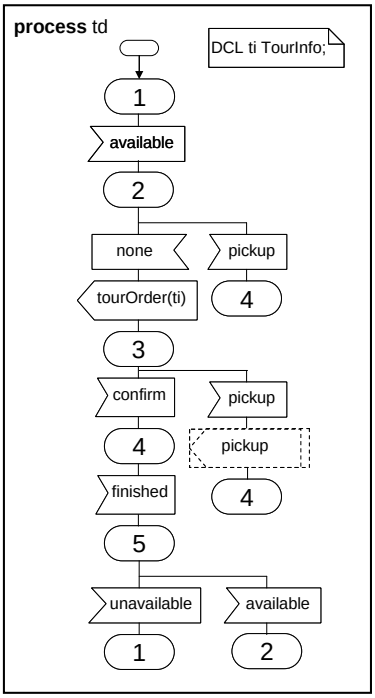
4. Undersøk rollen *td* for input konsistens.

td inneholder et mixed initiative mellom tourOrder og pickup i tilstand 2. Signalet Pickup må også kunne mottas i tilstand 3. Ressonering rundt det skjulte grensesnittet er ok men ikke nødvendig. Feil påstand om det: -1

5. Gjør diagrammene for *td* og *taxi* input konsistente og fullstendig kompatible. Forklar hvilket prinsipp du har fulgt for konfliktløsing.

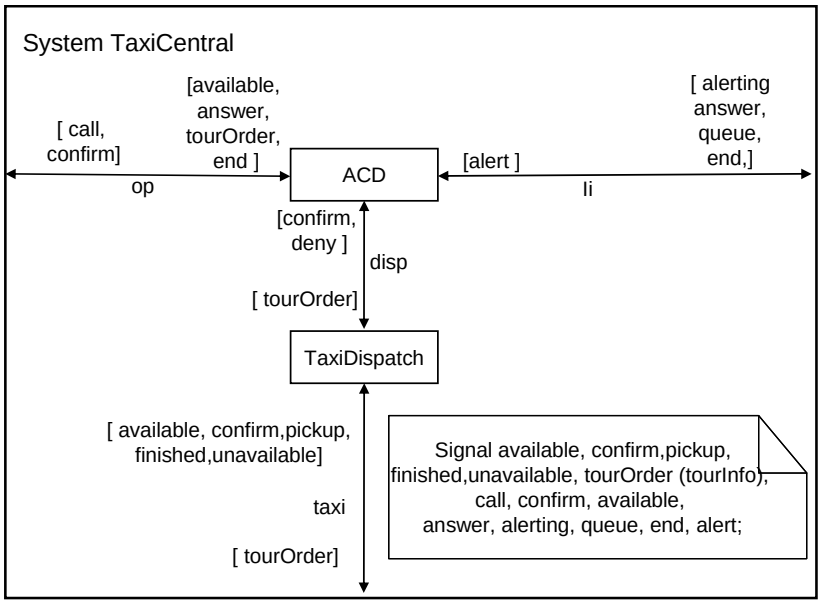
taxi inneholder tilsvarende mixed initiativ i tilstand 2. Løsningsprinsippet her må være at Pickup får prioritet fordi det da allerede er en kunde i drosjen. Ved kollisjon bør dette signaliseres til acd, angitt med stiplet output i td nedenfor. Siden dette snittet er skjult er det ikke veldig galt ikke å nevne dette (-1). På det andre snittet mangler det mottak av signalet som er representert ved none i tilstand 4, men dette er utenfor oppgaven. Ikke funnet inkonsistens i taxi -3. Inkompatible prosesser -2. Ikke angitt prinsipp -2





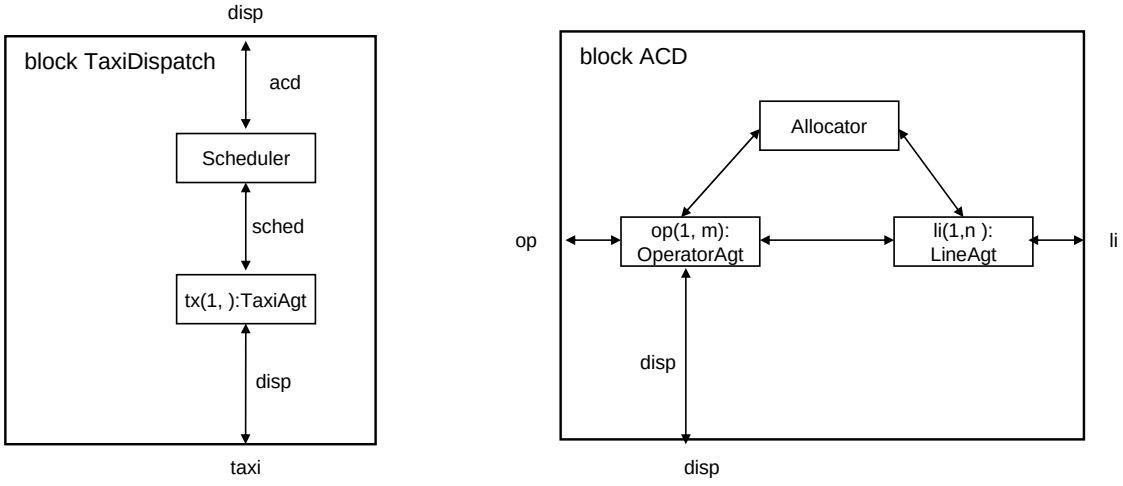
Oppgave 2. (25%) System design

1. Figur 1 viser strukturen til *TaxiCentral* systemet i UML2 notasjon. Definer den tilsvarende strukturen i SDL notasjon. Ta med alle signaler og signaldefinisjoner som fremgår av Figurene 2-5.



System type, block og block type er ok her. Tegnet omgivelser -1. Mangler signaldef -2. Mangler signaler på kanaler -2. Noen stud har tydeligvis prøvd å bruke svaret på en tidligere eksamen der løsningen var å lage en package med stereotyper for sdl begrepene <<process>> etc uten å lage struktur. Her spurte vi etter sdl struktur tilsvarende uml strukturen i oppgaven, så da blir det helt feil. -5 uten struktur, -3 med.

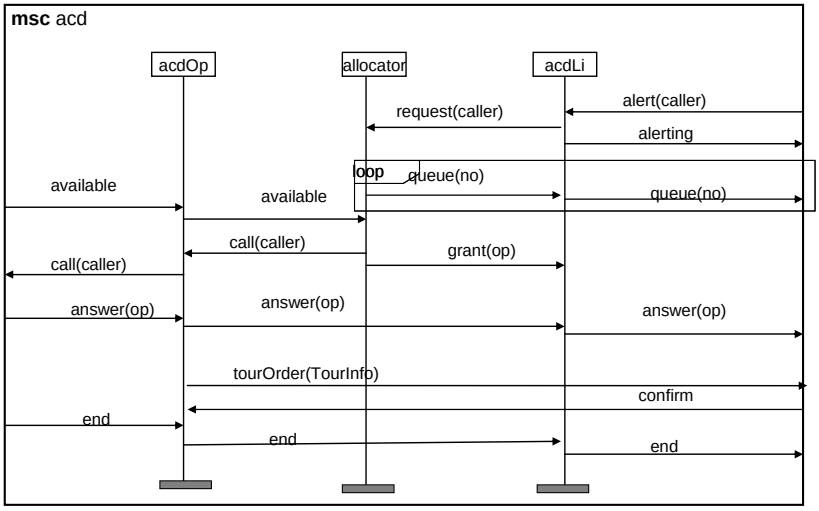
2. Foreslå en SDL struktur for delsystem *TaxiDispatch* som viser alle prosessene det består av. Forklar oppgavene til prosessene og begrunn valgene dine.
   Se nedenfor. Scheduler holder oversikt over opptatt/ledige taxi og hvor de er. Velger ut nærmeste ledige taxi og sender tourOrder til denne. Det er en taxi agent pr taxi ettersom det er en tilstandsfull sesjon mellom system og taxi, som beskrevet i TaxiInterface. Taxiagenten håndterer denne. Signaler ikke nødvendige her. Kun en taxi agent -3.. SDL feil -1 til -3. Uklare roller -1. Dårlige roller -3.



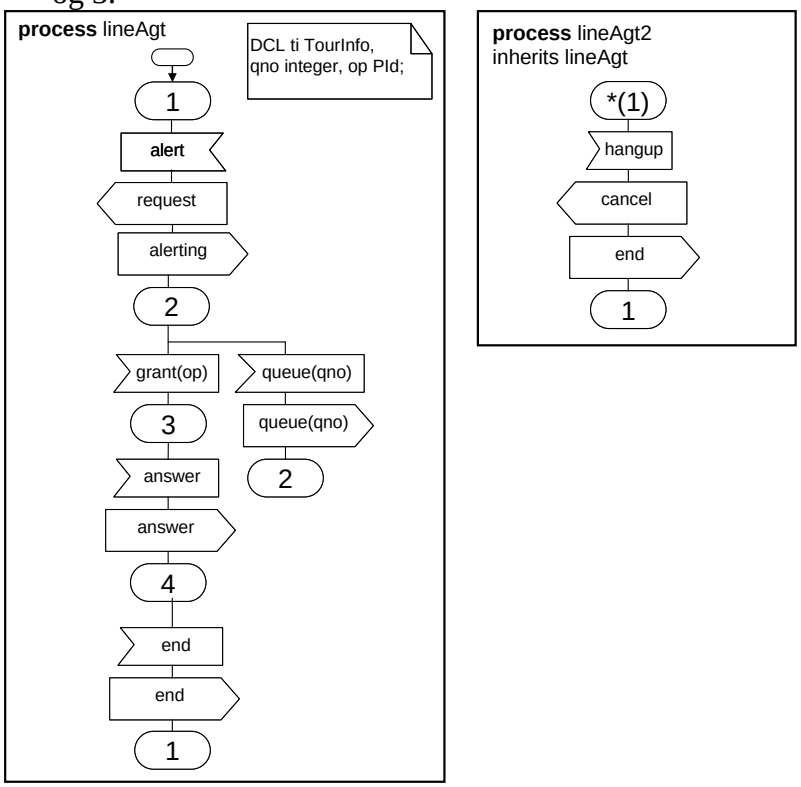
3. Foreslå en SDL struktur for delsystem *ACD* som viser alle prosessene det består av. Forklar oppgavene til prosessene og begrunn valgene dine.
   Se over. op og li speiler omgivelsene, mens allokatoren tar seg av å administrere

operatørene som ressurs. Manglende prosessett på op og li -3. Uklare roller -1. Dårlige roller -3.

4. Lag MSC diagram som viser interne interaksjoner i *ACD* for et anrop der kunden må stå i kø før operatør tildeles.
   Det var ønskelig med angivelse av eksterne signal også som nednfor, men oppgaven forlangte det ikke. Bør vise et helt forløp. Manglende forløpsdeler -1 til -2. Ulogiske forløp -1 til -2. MSC feil -1 til -3. Mangler en instans -2.



5. Definer oppførselen til den SDL prosessen i *ACD* som spiller rollen *acdLi*, se Figur 2 og 3.



Dette diagrammet bør bruke de interne signalene som er vist i eget MSC. -2 for ikke å gjøre det. Mangelfull oppførsel -1 til -3.

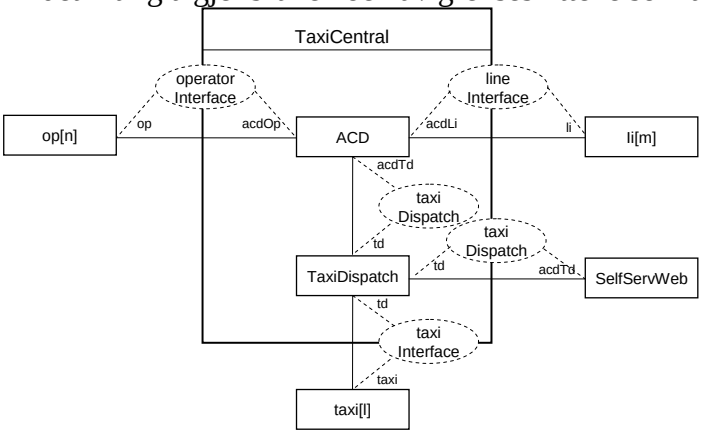
Oppgave 3. (25%) Endringer

1. I *lineInterface* (se Figur 3) er det ikke tatt hensyn til at kunden kan avslutte en samtale når som helst. Anta at dette skal gjøres ved at *li* rollen sender *hangUp* til *acdLi*, som

kvitterer med å sende signalet *end*. Er det mulig å benytte arving i SDL til å innføre denne oppførselen i en modifisert utgave av rollen *acdLi*, kalt *acdLi2*? Vis hvordan dette gjøres med utgangspunkt i besvarelsen på oppgave 2.5. (Dersom du ikke har svart på 2.5 så forklar en prinsippløsning.)
Ja, det er mulig som vist i figuren over. Ikke brukt arv -3. Feil funksjonalitet -2. Ikke vist diagram -3 hvis meget godt forklart, ellers -5.

2. Anta at vi endrer *li* rollen tilsvarende til *li2*. Vil den modifiserte *li2* rollen være kompatibel med den opprinnelige *acdLi* rollen? Begrunn!
   Nei den vi ikke være kompatibel fordi den sender hangup som den opprinnelige li ikke tar imot. ikke svar på kompatiblet -5.
   Er det mulig å benytte arving i SDL for å utføre denne modifikasjonen? Begrunn.
   Dersom vi antar at hangup bare sendes når vi mottar hangup fra linjen, er det mulig å gjøre dette ved arv. Vi legger til oppførsel trigget av hangup. Bemerk at dette bryter med ideen om at subtyper alltid kan erstatte supertyper. Dette holder altså ikke alltid i SDL.
3. Gjør om MSC diagrammet for *lineInterface* (Figur 3) til et sett med diagrammer uten *in-line* expressions (*alt, loop,* etc.) som til sammen beskriver den samme oppførselen.
   Det som må gjøres er å innføre conditions (labels) som binder diagrammene sammen. Feil i ett diagram -2. Manglende diagram -2 Feil bruk av label -2
4. Definer datastrukturen *TourInfo* vist i Figur 3 med ASN.1. Bruk de ASN.1 datatypene du synes passer best.
   TourInfo::= SEQUENCE{customer IA5String,
   taxi STRING, pickup LOCATION, start LOCATION, drop LOCATION, start GeneralizedTime, end GeneralizedTime, distance INTEGER}
   LOCATION ::= SEQUENCE {street IA5String, number INTEGER}
   Glemt location -2 Syntaxfeil -1 Det er greit med OCTETSTRING og UniversalTime.

5. Anta at vi skal legge til funksjonalitet for selvbetjening av kunder via et web grensesnitt som realiseres med en webserver. Anta at webserveren ligger utenfor systemet og oppfører seg som en SDL prosess sett fra systemet. Hvordan kan selvbetjeningsfunksjonaliteten legges til systemet?
   Foreslå en modifisering av Figur 1.
   UML figuren under viser hvordan dette kan gjøres.
   Er det mulig å benytte arving for å gjøre denne modifikasjonen?
   Arving i strukturen er enkelt, vi legger bare til det nye grensesnittet i strukturen. TaxiDispatch delsystemet må også få et nytt grensesnitt, så den tilhørende blokktypen må være virtuell. Oppførselen behøver neppe endres mye, hvis all rutingen er basert på TO PId.
   Er det mulig å gjenbruke noen av grensesnittene som allerede er definert?



SelfService knyttes til systemet ved gjenbruk av taxiDispatch grensesnittet
Tilknytning til ACD -2, ufullstendig om arv -1 til -2. Feil om grensesnitt -2

English

The questions refer to the system described in the appendix (Vedlegg). Study the appendix first.

(The exam counts 75% towards the final grade.)

Question 1. (25%) taxiInterface

Figure 5 gives an incomplete description of the *taxiInterface* where only one of the roles (*td*) is defined by an SDL process sgraph.

1. Explain what the signal *none* formally stands for in SDL and what it is used to represent in a role behavior.
2. Draw a process graph for the *taxi* role so that it is compatible with the given *td* role.
3. Draw a single MSC diagram describing all interactions in *taxiInterface*.
4. Check the *td* role for input consistency.
5. Make the diagrams for *td* and *taxi* input consistent and completely compatible. Explain what principle you have used for conflict resolution.

Question 2. (25%) System design

1. Figure 1 shows the *TaxiCentral* system structure using UML2 notation. Define the corresponding structure using SDL notation. Include all signals and signal definitions that follow from Figures 2-5.
2. Propose an SDL structure for the *TaxiDispatch* sub system showing all processes it contains. Explain the task of each process and justify your choices.
3. Propose an SDL structure for the *ACD* sub system showing all processes it contains. Explain the task of each process and justify your choices.
4. Make an MSC diagram showing the internal interactions in *ACD* for a call where the customer is queued before an operator is assigned.
5. Define the behavior of the SDL process in *ACD* that plays the *acdLi* role, se Figur 2 and 3.

Question 3. (25%) Modifications

1. In *lineInterface* (see Figure 3) one has not treated the case that the customer may end a call at any time. Assume now that the *li* role may send *hangUp* at any time after a call is initiated and that the *acdLi* role will respond by sending the *end* signal. Is it possible to use inheritance in SDL to define a modified *acdLi* role, called *acdLi2,* having this behavior? Demonstrate this by using your answer to question 2.5. (In the case you have no answer on this point, describe a principal solution)
2. Assume that the *li* role is modified accordingly to *li2*. Will the modified *li2* role be compatible with the original *acdLi* role? Justify!
   Is it possible to use inheritance in SDL to perform this modification? Justify!
3. Replace the MSC diagram for *lineInterface* (Figure 3) by a set of diagrams without *in-line* expressions (*alt, loop*, etc.) that together describe the same behavior.
4. Define the datastrukture *TourInfo* shown in Figure 3 using ASN.1. Use the ASN.1 datatypes that in your opinion fit best.
5. Assume we shall add self-service functionality to the system using a web interface realized by a web server. Assume that the web server is outside the system and

behaves like an SDL process seen from the system. How can the self-service functionality be added to the system? Propose a modification of Figure 1. Is it possible to do this modification using inheritance? Can any of the interfaces already defined be re-used?

Nynorsk

Ver merksam på at dersom det er skilnad i meininga mellom nynorskteksten og bokmålteksten er det bokmålsteksten som gjeld. Ved problem, ta kontakt med faglærar.

Oppgåvene er knytt til systemet som er beskrive i vedlegg. Studer vedlegget fyrst.

(Eksamen utgjer 75% av sluttkarakteren.)

Oppgave 1. (25%) taxiInterface

Figur 5 gjev ein ufullstendig skildring av *taxiInterface* der berre den eine rollen (*td*) er beskriven med ein SDL prosessgraf.

1. Forklar kva signalet *none* formelt tyder i SDL og kva det nyttes til å representere i ein rolleoppførsel.
2. Teikn prosessgrafen for *taxi* rollen slik at den er kompatibel med den gjevne *td* rollen.
3. Teikn eitt MSC diagram som beskriv alle interaksjonane i *taxiInterface*.
4. Gransk rollen *td* for input konsistens.
5. Gjer diagramma for *td* og *taxi* input konsistente og fullstendeg kompatible. Forklar prinsippet du har fulgt for konfliktløysing.

Oppgave 2. (25%) System design

1. Figur 1 syner strukturen til *TaxiCentral* systemet i UML2 notasjon. Definer den tilsvarande strukturen i SDL notasjon. Tek med alle signal og signaldefinisjonar som går fram av Figurane 2-5.
2. Foreslå ein SDL struktur for delsystem *TaxiDispatch* som syner alle prosessane det inneheld. Forklar oppgåvane til prosessane og grunngjev valga dine.
3. Foreslå ein SDL struktur for delsystem *ACD* som syner alle prosessane det inneheld. Redegjer for oppgåvane til prosessane og grunngjev valga dine.
4. Lag MSC diagram som syner interne interaksjonar i *ACD* for eit anrop der kunden må stå i kø før operatør tildelelast.
5. Definer oppførselen til den SDL prosessen i *ACD* som spelar rollen *acdLi*, se Figur 2 og 3.

Oppgave 3. (25%) Endringar

1. I *lineInterface* (sjå Figur 3) er det ikkje tatt omsyn til at kunden kan avslutta ein samtale kvar tid som helst. Anta at dette skal gjerast ved at *li* rollen sender *hangUp* til *acdLi*, som kvitterar med å senda signalet *end*. Er det mogleg å nytte arving i SDL til å innføre denne oppførselen i ein modifisert utgåve av rollen *acdLi*, kalla *acdLi2*? Syn korleis dette gjerast med utgangspunkt i besvarelsen på oppgåve 2.5. (Dersom du ikkje har svart på 2.5 så redegjer for ein prinsippløsning.)
2. Anta at me endrar *li* rollen tilsvarende til *li2*. Vil den modifiserte *li2* vere kompatibel med den opprinnelege *acdLi* rollen? Grunngjev!
   Er det mogleg å nytte arving i SDL for å utføre denne modifikasjonen? Grunngjev.
3. Gjer om MSC diagrammet for *lineInterface* (Figur 3) til eit sett med diagram utan *in-line* expressions (*alt*, *loop*, etc.) som til saman beskriv den same oppførselen.

4. Definer datastrukturen *TourInfo* synt i Figur 3 med ASN.1. Bruk dei ASN.1 datatypane du tykkjer passar best.
5. Anta at me skal leggje til funksjonalitet for sjølbetening av kunder via eit web grensesnitt som realiserast med ein webserver. Anta at webserveren ligg utanfor systemet og ter seg som ein SDL prosess sett frå systemet. Korleis kan sjølbeteningsfunksjonaliteten legges til systemet? Foreslå ein modifisering av Figur 1. Er det mogleg å nytte arving for å gjera denne modifikasjonen? Er det mogleg å nytte om att nokre av grensesnitta som allerede er definert?

Vedlegg

Bokmål

Systemet som inngår i oppgavene: en drosjesentral.
I disse oppgavene ser vi på funksjonaliteten til en drosjesentral. Drosjesentralen består av delsystemene *ACD* og *TaxiDispatch*, som vist i Figur 1. Oppførselen på de forskjellige grensesnittene er definert med UML 2.0 kollaborasjoner som vist i Figur 2-5. Bemerk at Figur 2 angir bindingen av kollaborasjonsroller til delsystemer og omgivelser.

Delsystemet *ACD* er en køordner sentral (”automatic call distributor”). Den har et antall innkommende telefonlinjer (*li*), og et antall operatører (*op*) som tar imot bestillinger. Kunder bestiller drosje ved å ringe nummeret til drosjesentralen. Anropene rutes av telefonnettet (som er utenfor oppgaven) frem til en ledig innkommende linje (*li*). Dersom ingen operatører er ledige stilles anropene i kø inntil det blir en ledig operatør. Mens kundene (*li*) står i kø får de løpende oppdatert informasjon om sin køplass. Operatørene sender bestilling på drosjer med angivelse av henteadresse ved å sende en melding (*tourOrder*) til *TaxiDispatch* delsystemet.

TaxiDispatch holder oversikt over ledige drosjer (*taxi*) og tildeler drosjer så rettferdig som mulig. (Systemet tar hensyn til hvor drosjene og kundene befinner seg, men dette er utenfor oppgavene.) Når en drosje er ledig for oppdrag melder den seg ledig (signalet *available* i Figur 5). Den kan da enten motta en bestilling fra drosjesentralen (signalet *tourOrder* i Figur 5) eller plukke opp en kunde på gaten. Plukkes en kunde opp fra gaten må drosjen melde dette til *TaxiDispatch* delsystemet (signalet *pickup* i Figur 5) slik at drosjen markeres som opptatt. Når turen er over kan drosjen melde seg ledig for en ny tur, eller melde seg utilgjengelig.

English

The system used in the questions: a taxi central
We consider the functionality of a taxi central consisting of the subsystems *ACD* and *TaxiDispatch* as shown in Figure 1. The interface behaviors are defined using UML 2.0 collaborations as shown in Figures 2-5. Note that Figure 2 specifies which collaboration roles are bound to which subsystems and environment entities.

The ACD subsystem is an automatic call distributor having a number of incoming lines (*li*) and a number of operators (*op*) that answer calls and receive taxi bookings. Customers can book taxis by placing calls to the taxi central. The calls are routed through the telephone network (which is not considered here) to a free incoming line (*li*). In case no operators are free the call will be placed in queue until an operator becomes available. While waiting in queue the customer will be updated about the current queue position. The operators issue taxi orders by sending a message (*tourOrder*) with the pickup location to the TaxiDispatch subsystem.

The *TaxiDispatch* subsystem keeps an overview of available taxis (*taxi*) and assigns taxis to customers as fairly as possible (taking the location of taxis and customers into account, but this is outside the questions). A taxi marks itself as available for new trips by sending the signal *available*, see Figure 5. It may then either receive a booking (signal *tourOrder* in Figure 5) or pick up a customer from the street and inform the *TaxiDispatch* subsystem about this (signal *pickup* in Figure 5) so that the taxi is marked as taken. When a trip is ended the taxi may either mark itself as *available* or *unavailable*.

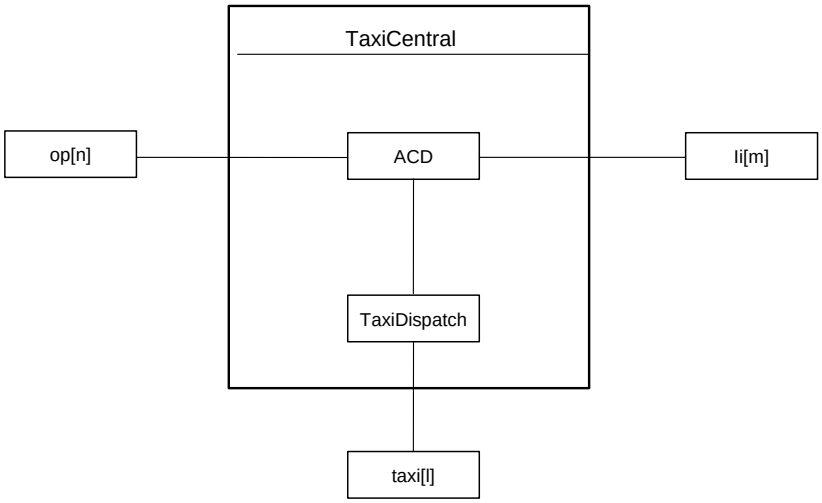


Figure 1 UML2 diagram showing the TaxiCentral in context

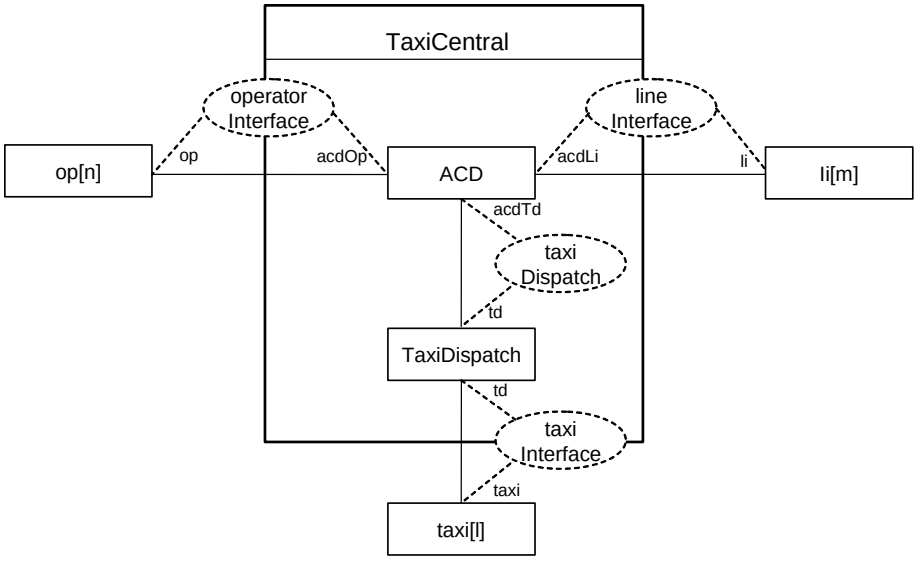


Figure 2 The TaxiCentral with interface collaborations

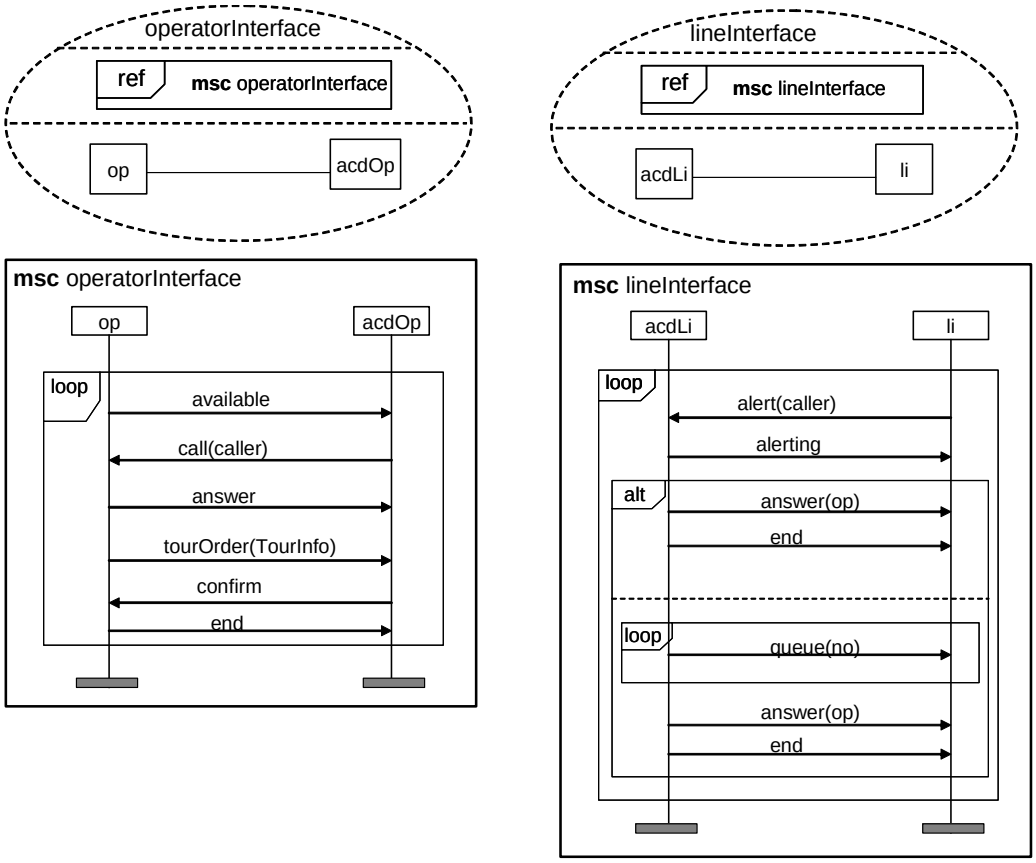


Figure 3 The operatorInterface and lineInterface

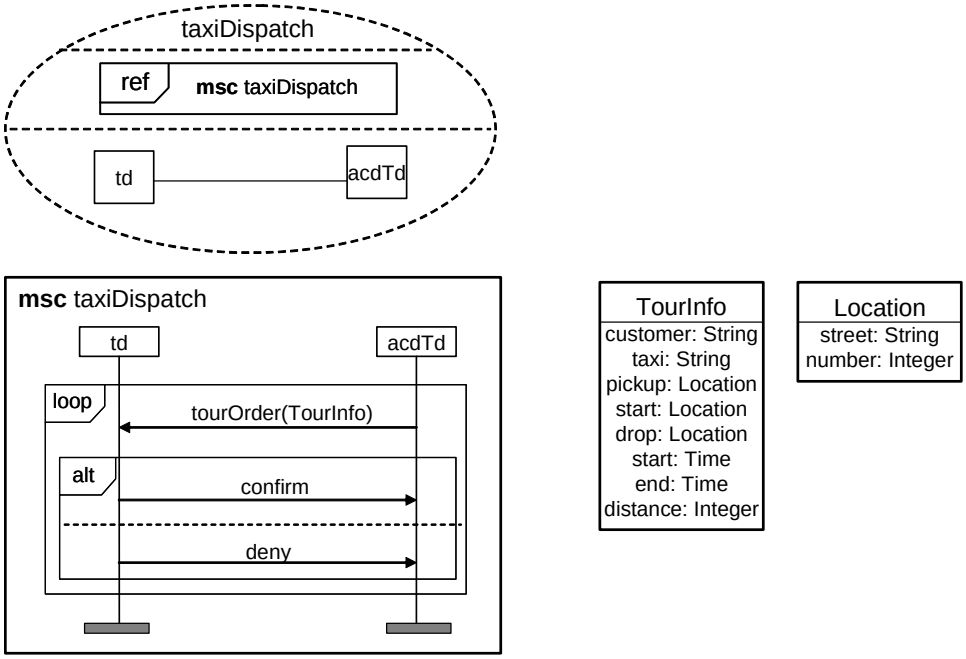


Figure 4 The taxiDispatch interface

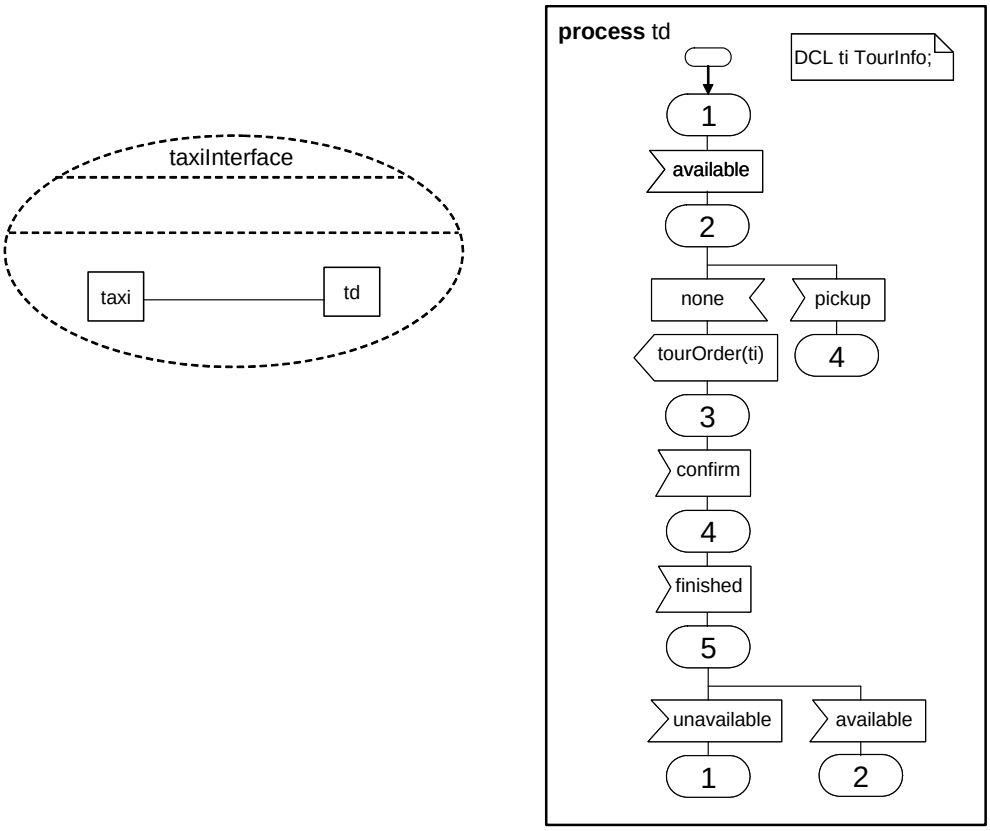


Figure 5 The taxiInterface with the td role behavior defined using SDL