Norges teknisk-naturvitenskapelige universitet
Institutt for telematikk

EKSAMENSOPPGAVE I TTM4115 – SYSTEMERING AV DISTRIBUERTE SANNTIDSSYSTEMER
EXAM TTM4115 ENGINEERING DISTRIBUTED REAL-TIME SYSTEMS
SOLUTION PROPOSAL
(IN THE ENGLISH TEXT)

Contact person/Faglig kontakt under eksamen:	Rolv Bræk
Phone/Tlf.:	415 44 605
Exam date/Eksamensdato:	28. mai 2011
Time/Eksamenstid:	09:00-13:00
Credits/Studiepoeng:	7,5 SP
Remedies/Tillatte hjelpemidler:	A: All written and handwritten examination support materials are permitted. All calculators are permitted A: Alle trykte og håndskrevne hjelpemidler tillatt. Alle kalkulatorer tillatt
Languages/Språkform:	
Antall sider bokmål:	1
Number of pages in English:	1
Tal på sider nynorsk:	1
Attachment/Antall sider vedlegg:	7
Results/Sensurdato[1]:	**21. juni 2011**

[1] Merk! Studentene må primært gjøre seg kjent med sensur ved å oppsøke sensuroppslagene.

Bokmål (Eksamen utgjør 75% av sluttkarakteren.)

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

Oppgave 1. (25%) SDL struktur

1. Definer *Flexibus* systemet i Figur 2 som en SDL blokktype med indre blokker av type *Central*, *Bus* og *Mobile Phone.* Disse blokktypene skal ha en SDL gate for hver tilknyttet kanal. Ta med blokk type referanser for alle de indre blokk typene. Definer signalene som fremgår av Figur 3, 4 og 5, og angi hvilke av signalene som formidles på de forskjellige kanalene.
2. Definer blokktypene *Central* og *Mobile Phone* når vi antar at de indre enhetene er SDL prosesser. Ta med signaler og nødvendige prosess type referanser.
3. I hvilke tilfeller kan man generelt sett utelate å adressere et signal med *TO PId* og likvel være sikker på at signalet kommer frem til en bestemt prosess? Er det eksempler på dette i Flexibus?
4. I semesteroppgaven benyttet dere *RAMSES* og *ActorFrame*. Forklar kort hensikten med disse og deres hovedfunksjoner.

Oppgave 2. (30%) SDL oppførsel

Vi antar i det følgende at datatypene fra Figur 1 er blitt predefinert i et bibliotek. Dette gjelder også typen *UserList* med operasjonene *Add(UId, PId)* for å legge inn en ny bruker og *Get(UId)->Pid* for å finne *PId* til en *UserAgent* tilsvarende en *UId*.

1. Definer oppførselen til SDL prosessen *UserMgr.* Ta med erklæring og bruk av variabler.
2. Definer *LogOn* og *LogOff* prosedyrene i *UserClient,* se Figur 5. Ta med erklæring og bruk av variabler.
3. Definer oppførselen til tilstanden *Booking* i Figur 5 slik at den er kompatibel med tilsvarende oppførsel for *UserAgent* gitt i Figur 4. Kansellering trigges av at bruker trykker *CancelBtn* i *GUI*.
4. Sjekk *UserAgent* i Figur 4 for input konsistens og forklar hva som må gjøres for å rette opp eventuelle inkonsistenser.

Oppgave 3. (20%) Prosessalgebra, CCS

I denne oppgaven skal vi modellere betalingen på bussen med prosessalgebra (CCS). Følgende tre prosesser, som også er beskrevet på grafisk form i Figur 6, inngår:

PaymentUnit: PU = pay; card'; enter; card'; leave; paid'; PU
TicketStation: TS = card; check'; (first; enter'; TS + second; leave'; TS)
BusClient: BC = check; (first'; BC + second'; BC)

1. Ekspander oppførselen *BS = BC||TS* (gjerne på grafisk form).
2. Ekspander oppførselen *PS = BS||PU* (gjerne på grafisk form).
3. Er det noen problemer i *PS*? Hvilke?
4. Anta at oppførselen *BS* erstattes av spesifikasjonen (også beskrevet i Figur 6):
   BS1 = card; (leave'; BS1 + enter'; BS1)

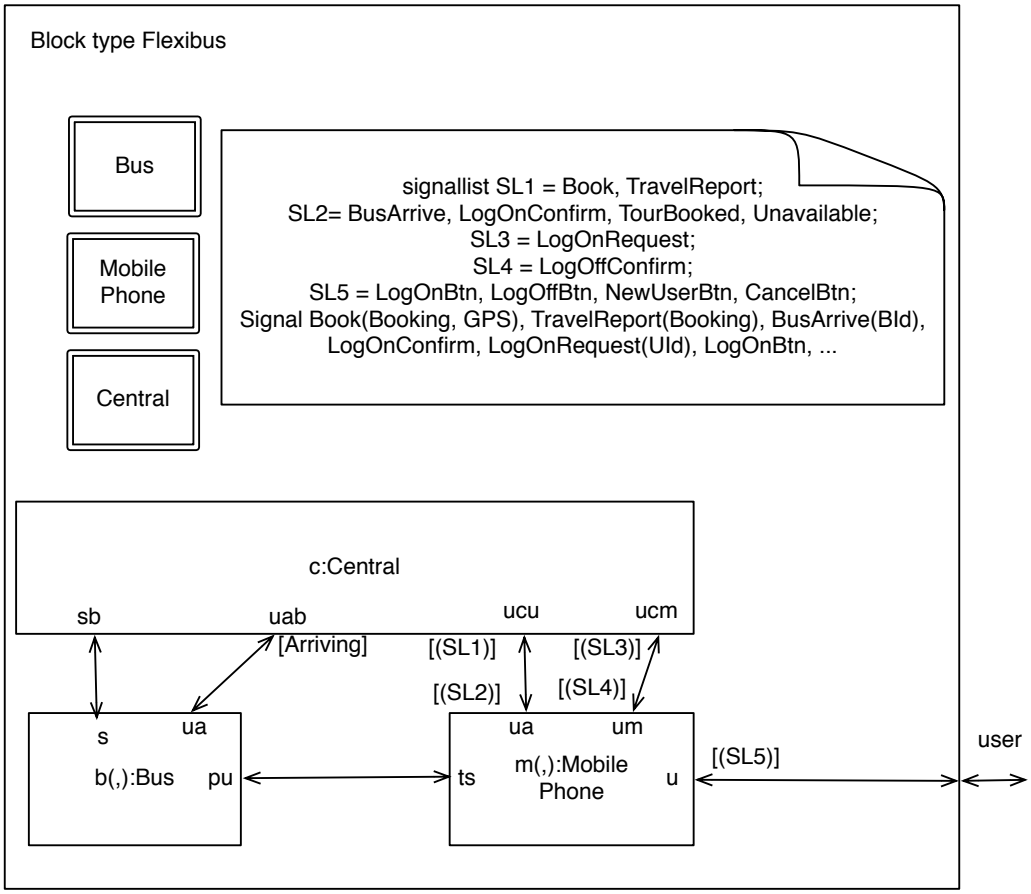
   Ekspander *BS1||PU* (gjerne på grafisk form).
   Er *BS1* ekvivalent med *BS*? Gi et begrunnet svar.

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

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

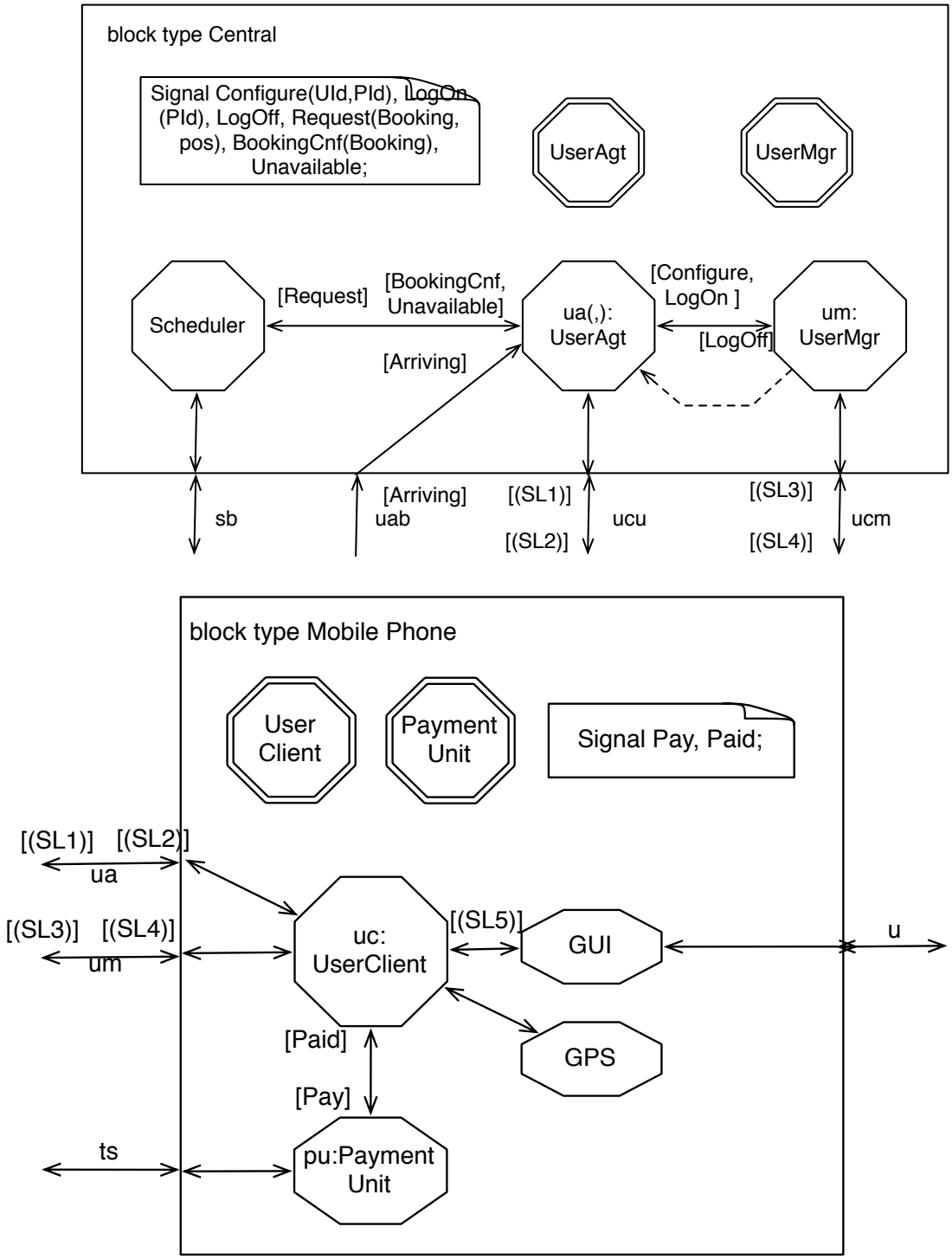
Question 1. (25%) SDL structure

1. Define the *Flexibus* system in Figure 2 as an SDL block type having inner blocks of type *Central*, *Bus* and *Mobile Phone.* These block types shall have an SDL gate for each connected channel. Include block type references for all inner block types. Define all signals that are used in Figures 3, 4 and 5, and specify which of these signals are passed on each channel.



Right structure: 30
Typedefined blocks: 40
Type references: 10
Signaldefs: 20
Missing signalparameters: -20
Gates missing: -20
Channels missing: -10
Wrong signal on channel:-10

2. Define the block types *Central* and *Mobile Phone* assuming that all inner entities are processes. Include signals and necessary process type references.



Reading GPS is like reading a local variable, so a process is not quite right, but good enough. Scoring is roughly as for Q1.1.
Create flow (dashed): +10 bonus *Missing signals on channels: - 10*
Missing gates: -20

3. Under which circumstances can one omit to address signals using *TO PId* and still be certain the signals will arrive at a unique destination? Are there any examples of this in Flexibus?

There are three options:

- *TO Name: can be used within a block where there is only one instance with that name.*
- *VIA: when the path leads to a unique process that can receive the signal*

- *No address: when there is only one process instance in the whole system that can receive the signal*

This is used for the Request signal to the Scheduler.

Via: +30 *General statement about uniqueness with example: +75*
Name: + 30
No name: +30
Uniqueness: +10

4. In the term assignment you used *RAMSES* and *ActorFrame*. Explain shortly their purposes and main functionality.

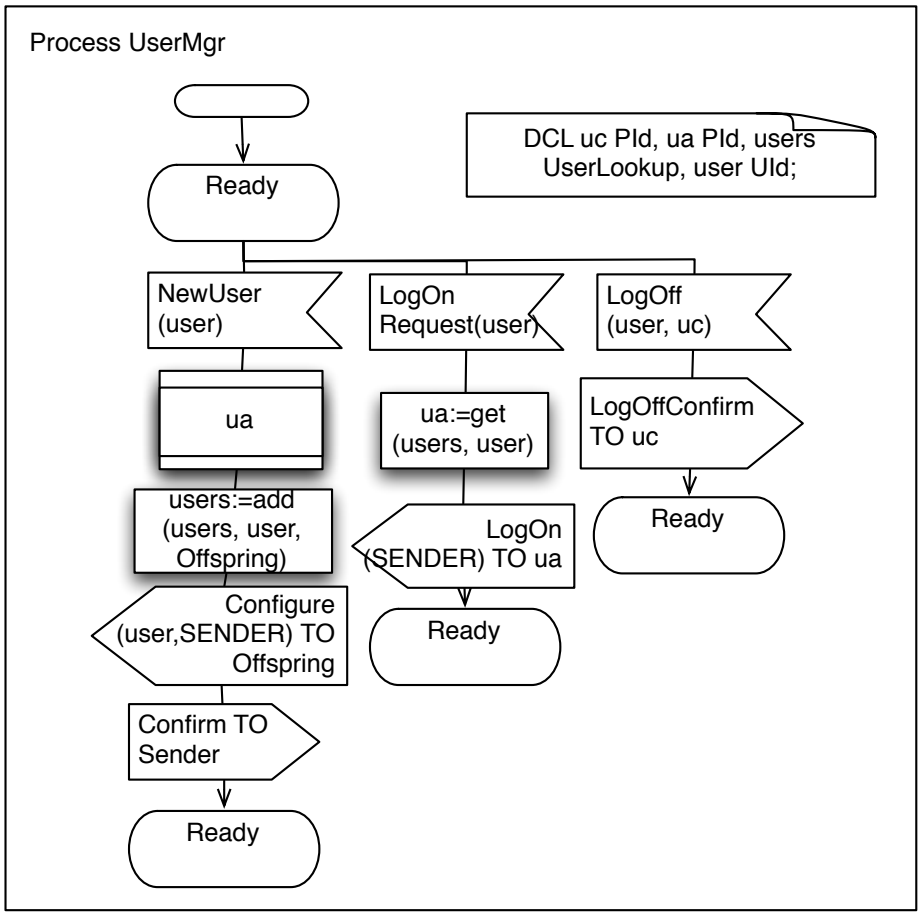
   RAMSES is an Eclipse tool for editing of UML systems and state machines, analysing and code generation to Java for ActorFrame
   ActorFrame is a runtime support system that provides message routing and state machine support in Java. Has also support for actors and role request/session initiation.

   Ramses OK:+50
   ActorFrame OK: +50

Question 2. (30%) SDL behaviour

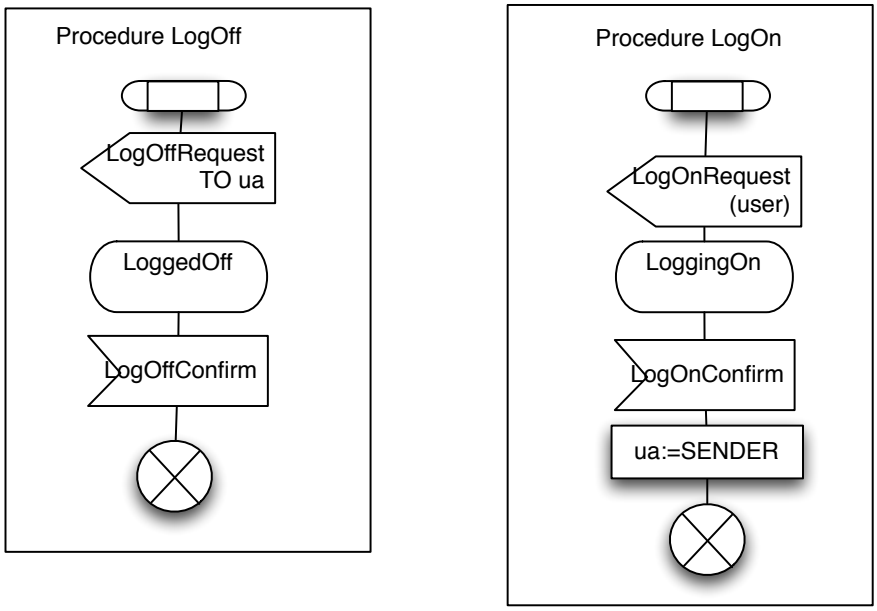
We assume in the following that the data types from Figure 1 have been predefined in a library. So is the data type *UserList* having the operations *Add(UId, PId)* for adding new users and *Get(UId)->Pid* for finding the *PId* of the *UserAgent* corresponding to a *UId*.

1. Define the behaviour of the SDL process *UserMgr.* Include declarations and use of variables.



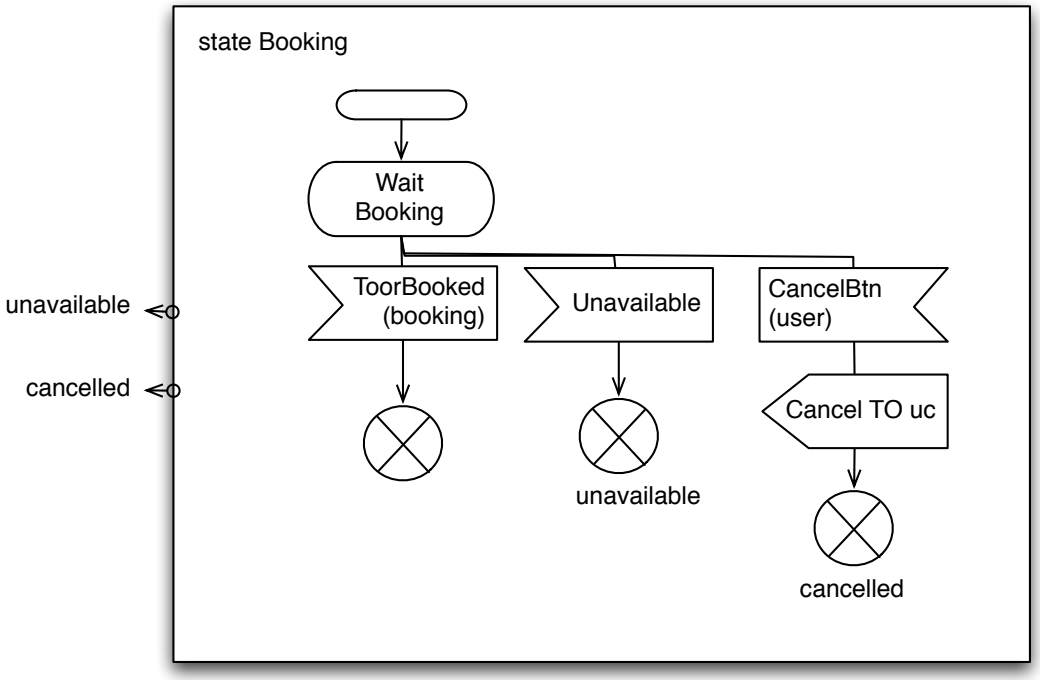
Main flow OK: + 70
Create and use Offspring: +10
DCL OK: +10
Correct use of variables and parameters: + 20
Included external behaviour: -20
Stateful behaviour: -20
Missing DCL: -10
Minor errors in papmeters/variables: -10

2. Define the *LogOn* and *LogOff* procedures used in the *UserClient* in Figure 5. Include declaration and use of variables.



Main Flow OK: + 70 *Not procedures: -30*
ua assignment: +20 *Included external signals:-30*
Parameters: +10

3. Define the behaviour of composite state *Booking* in Figure 5 so that it is compatible with the corresponding behaviour of the *UserAgent* given in Figure 4. Cancellation is triggered when the customer presses the *CancelBtn* in the *GUI.*



Composite state OK: +60 *Included external behaviour: -20*
Flow OK: +30 *Too many exit labels: -10*
Parameters: +10 *Forgot Cancel signal: -10*
Missing exit labels on frame: -10

- Check the *UserAgent* in Figure 4 for input consistency and explain what has to be done to correct the inconsistencies, if any.

There is a mixed initiative in state Wait booking leading to inconsistency in transitions:

- *WaitBook --> WaitBus on Booking Cnf and WaitBook--> Ready on Unavailable. To resolve these one must add a transition triggered by Cancel to WaitBook (Will cause secondary inconsistency) and Ready.*
- *WaitBook --> Ready on Cancel. To resolve one must add a transition triggered by BookingCnf and Unavailalbel to Ready.*

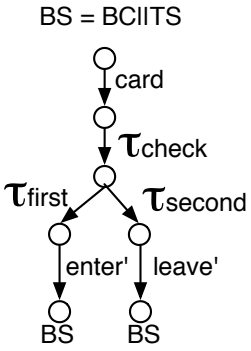
Mixed initiative mentioned without anything else: +20
Mixed initiative with some cases: +10
One case correct: +50
Each additional: +20
Dealing with secondary inconsistencies: +20 (up to 100)

Question 3. (20%) Process algebra, CCS

In this question we model the payment in the bus using process algebra (CCS). The following three processes, also defined in graphic form in Figure 6, take part:

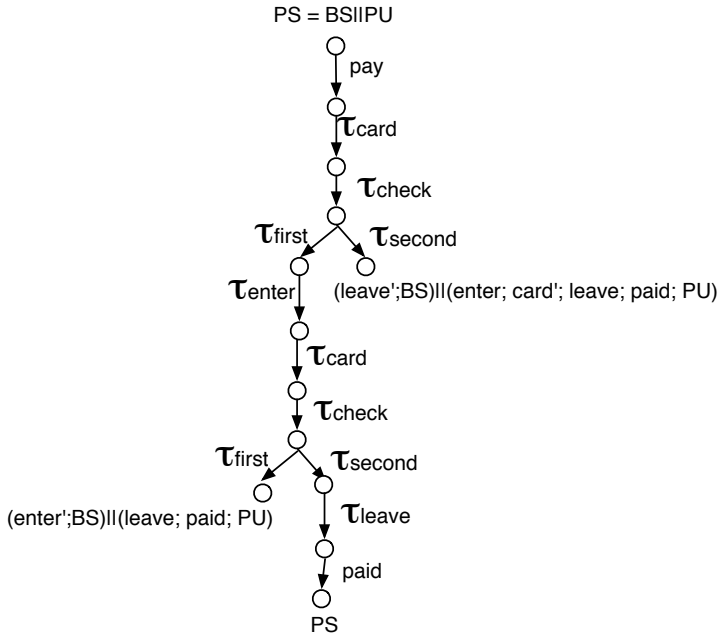
PaymentUnit: PU = pay; card'; enter; card'; leave; paid'; PU
TicketStation: TS = card; check'; (first; enter'; TS + second; leave'; TS)
BusClient: BC = check; (first'; BC + second'; BC)

1. Expand the behaviour *BS = BC||TS* (use the graphic form, if you like).



First transition OK: +30 *Notation errors:-10*
Some correct Tau: 40 *Taking transitions not enabled: -50*
The rest: + 30

2. Expand the behaviour *PS = BS||PU* (use the graphic form, if you like).



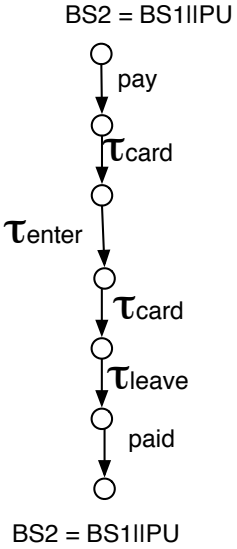
First transition OK: +20 *Missing 1 deadlock if rest is OK: -10*
Main sequence: +40
Deadlocks: +40

3. Are there any problems in PS? What problems may occur?

   There are two deadlocks indicated by the unresolved expressions.

4. Assume that the behaviour *BS* is replaced by the specification (also described in Figure 6):
   BS1 = card; (leave'; BS1 + enter'; BS1)

   Expand the behaviour *BS1||PU* (use the graphic form, if you like).



   Is *BS1* equivalent with *BS*? Justify your answer.
   When composing with PU there is deadlock with BS and not with BS1. Therefore they are not equivalent seen from PU, and hence not observation equivalent.

   Erroneous claims about trace equivalence to be ignored.

   Correct expansion: +50

Correct about equivalence: +50 ignoring the deadlocks found in 3. here: -30
Observation equivalence is not correct but give +20 if this is the only argument and the argument is that tau can be ignored. (It cannot under choice, but the students have not worked much with that.)

Nynorsk (Eksamen utgjer 75% av sluttkarakteren.)

Oppgåvene referer seg til systemet som er skildra i vedlegg. Studer vedlegget fyrst.

Oppgåve 1. (25%) SDL struktur

1. Definer *Flexibus* systemet i Figur 2 som ein SDL blokktype med indre blokker av type *Central*, *Bus* og *Mobile Phone.* Desse blokktypane skal ha ein SDL gate for kvar tilknytta kanal. Ta med blokktype referansar for kvar av dei indre blokktypane. Definer signala som er bukte i Figur 3, 4 og 5, og angje kva for nokre av signala som formidlast på dei einskilde kanalane.
2. Definer blokktypane *Central* og *Mobile Phone* når vi antek at dei indre delane er SDL prosessar. Ta med signal og naudsynte prosesstype referansar.
3. I kva tilfeller kan ein generelt sett utelate å adressere eit signal med *TO PId* og samstundes vera trygg på at signalet kjem frem til ein bestemt prosess? Er det døme på dette i Flexibus?
4. I semesteroppgåva nytta de *RAMSES* og *ActorFrame*. Forklar kort føremålet med desse og deira hovudfunksjonar.

Oppgåve 2. (30%) SDL oppførsel

Vi antek i det fylgjande at datatypane frå Figur 1 er blitt predefinerte i eit bibliotek. Dette gjeld også typen *UserList* med operasjonane *Add(UId, PId)* for å leggja inn ein ny brukar og *Get(UId)->Pid* for å finne *PId* til ein *UserAgent* tilsvarande ein *UId*.

1. Definer oppførselen til SDL prosessen *UserMgr.* Ta med erklæring og bruk av variablar.
2. Definer *LogOn* og *LogOff* prosedyrane i *UserClient,* sjå Figur 5. Ta med erklæring og bruk av variablar.
3. Definer oppførselen til tilstanden *Booking* i Figur 5 slik at den er kompatibel med tilsvarande oppførsel for *UserAgent* gitt i Figur 4. Kansellering trigges av at brukar trykker *CancelBtn* i *GUI*.
4. Sjekk *UserAgent* i Figur 4 for input konsistens og forklar kva som må gjerast for å rette opp eventuelle inkonsistensar.

Oppgåve 3. (20%) Prosessalgebra CCS

I denne oppgåva skal vi modellere betaling i bussen med prosessalgebra (CCS). Fylgjande tre prosessar, som også er synt på grafisk form i Figur 6, inngår:

PaymentUnit: PU = pay; card'; enter; card'; leave; paid'; PU
TicketStation: TS = card; check'; (first; enter'; TS + second; leave'; TS)
BusClient: BC = check; (first'; BC + second'; BC)

1. Ekspander oppførselen *BS = BC||TS* (gjerne på grafisk form).
2. Ekspander oppførselen *PS = BS||PU* (gjerne på grafisk form).
3. Er det nokre problem i *PS*? Kva for problem?
4. Anta at oppførselen *BS* erstattast av spesifikasjonen (også gjeve i Figur 6):
   BS1 = card; (leave'; BS1 + enter'; BS1)

   Ekspander *BS1||PU* (gjerne på grafisk form).
   Er *BS1* ekvivalent med *BS*? Gje eit grunngjeve svar.

Vedlegg/ Appendix

Bokmål

Flexibus

Vi skal her studere et system for fleksibel busstransport i en by. Bussene kjører ikke faste ruter, men får tildelt ruter dynamisk etter behov. Bussene stopper bare på angitte holdeplasser som ligger tett i byen. Kundene bruker mobiltelefoner med GPS posisjonering til bestilling og betaling. Telefonene har nærfeltskommunikasjon, *NFC*, til registrering og betaling ombord på bussene. Figur 1 definerer noen datatyper (passive objekter) i *Flexibus*, mens Figur 2 viser den funksjonell strukturen til systemet i litt uformell UML notasjon.

Central enheten har en *UserAgent* til å betjene hver kunde og en *UserMgr* til å opprette nye *UserAgenter* og til å delta ved pålogging og avlogging, som vist i Figur 3. En *UserAgent* vil eksistere permanent, også mens kunden ikke er pålogget.

I *MobilePhone* er det en *UserClient* med tilknyttet *GUI*, *PaymentUnit* og *GPS*. *UserClient* logger seg på med sitt brukernavn, *UId*, som vist i Figur3. *UserMgr* finner den tilhørende *UserAgent* og initierer en sesjon mellom *UserClient* og *UserAgent*. Oppførselen til *UserAgent* er gitt i Figur 4 og oppførselen til *UserClient* er gitt i Figur 5. Bemerk at prosessgrafen i Figur 5 inneholder en composite state *Booking* (se SDL 2000) som skal defineres i en av oppgavene. Signaler fra *GUI* har navn som ender på ”*Btn*” (for ”button”).

Ved bestilling sender *UserClient* sin *GPS* posisjon og en utfylt *booking* med navnet på ønsket ende-holdeplass, *Stn*, til *UserAgent* som sender bestillingen videre til *Scheduler* med Signalet *Request.* Denne holder rede på alle aktive busser, hvor de er og hvilken *Schedule*, de har fått tildelt. *Scheduler* kvitterer med å tildele en bestillingsbekreftelse, *BookingCnf,* med en utfylt *booking* verdi som angir stoppestedet bussen vil hente passasjeren på, bussens identitet, *BId*, og antatt hentetidspunkt. Bussen varsler *UserAgent* som varsler *UserClient* når den ankommer hentestedet. *UserClient* vil da aktivisere *PaymentUnit* i mobiltelefonen. Kundene legger telefonen inntil en *TicketStation* i bussen når de går om bord, og gjør det samme når de forlater bussen etter endt reise. *UserClient* får så kvittering i form av en *booking* verdi med status *ended* direkte fra *PaymentUnit* og sender denne videre til *UserAgent* for registrering.

English

Flexibus

We shall here study a system for flexible bus transport in a City. Buses do not run fixed schedules, but are assigned schedules dynamically on customer demand. Buses stop only on designated bus stops that are densely distributed throughout the city. Customers use mobile phones with GPS positioning for booking and payment. The phones have near field communication, *NFC*, for registration and payment on the bus. Figure 1 defines some data types used by the system while Figure 2 defines the functional structure using slightly informal UML notation.

The *Central* has a *UserAgent* to serve each customer, and a *UserMgr* to create new *UserAgents* and assist during logon and logoff as shown in Figure 3. *UserAgents* will exist permanently even when the customer is not logged on.

The *MobilePhone* has a *UserClient* and a connected *GUI*, *PaymentUnit* and *GPS*. A *UserClient* logs on with a username, *UId*, as shown in Figure 3. The *UserMgr* finds the corresponding *UserAgent* and initiates a session between the *UserClient* and *UserAgent*. The *UserAgent* behaviour is defined in Figure 4 and the *UserClient* behaviour is defined in Figure 5. Note that the process graph in Figure 5 contains a composite state *Booking* (see SDL 2000) that you shall define in one of the questions. Signals from the *GUI* have names ending with ”*Btn*”.

When booking, the *UserClient* sends its *GPS* position and a filled in *booking* with the desired destination bus stop name, *Stn*, to the *UserAgent* that sends the booking on to the *Scheduler* in the signal *Request.* The *Scheduler* manages all the buses and their schedule. The *Scheduler* responds by sending *BookingCnf,* with a filled in *booking* containing the pick-up station, the bus identity, *BId*, and the expected pick-up time. When approaching the pick-up station the bus will inform the *UserAgent* who notifies the *UserClient*. The *UserClient* will then activate the *PaymentUnit* in the *Mobile Phone*. The customer touches a *TicketStation* in the bus when entering, and again when leaving. The *UserClient* then receives a receipt in the form of a *booking* value with status *ended* directly from the *PaymentUnit* and sends it on to the *UserAgent* for registration.

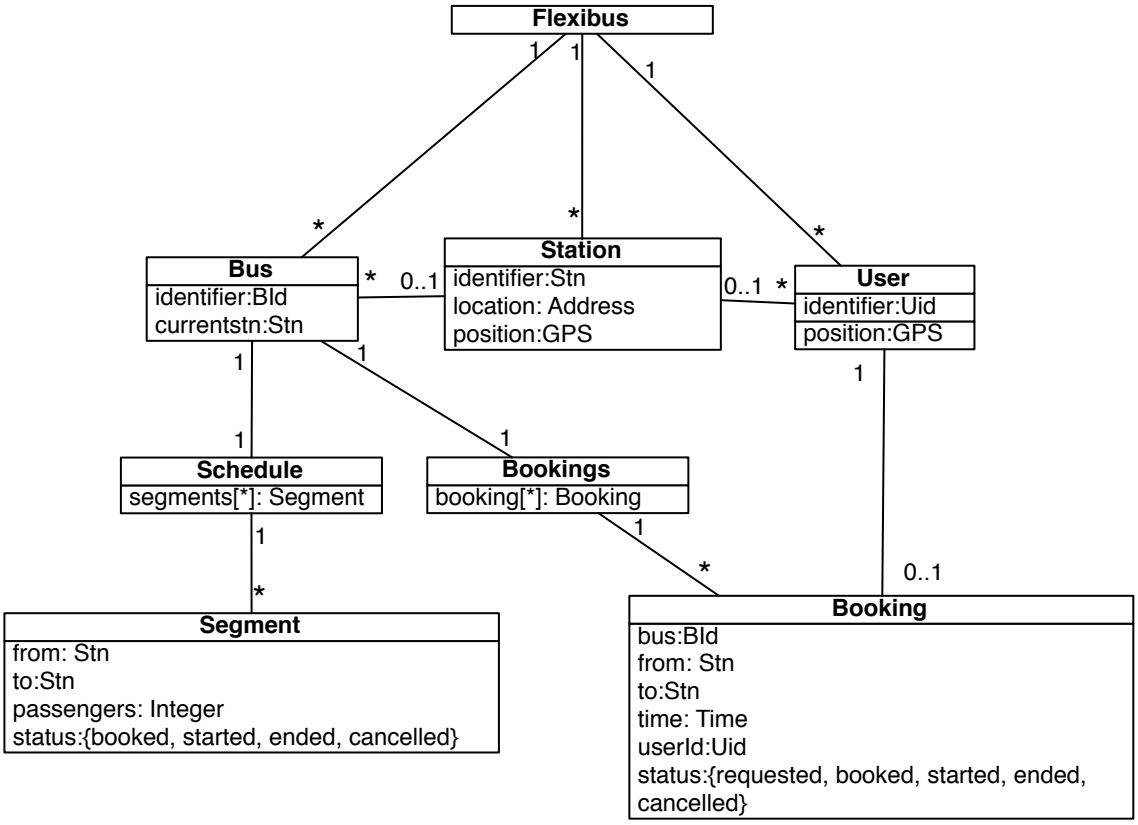


Figure 1 Data types in the Flexibus system

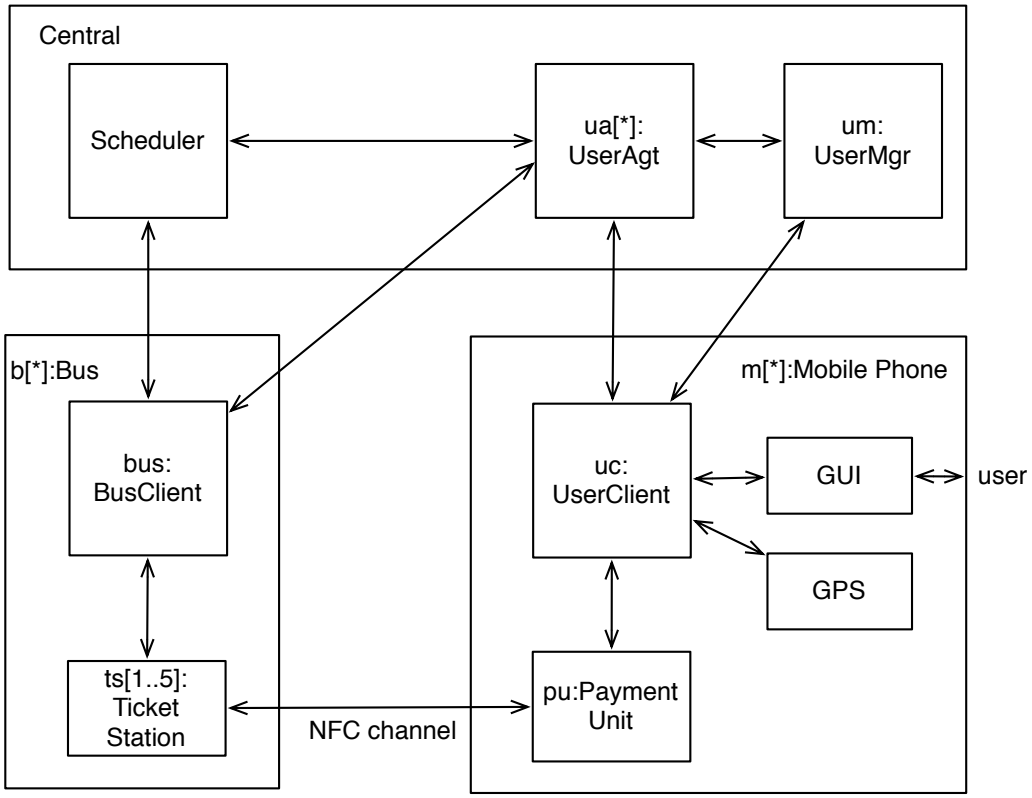


Figure 2 The Flexibus System

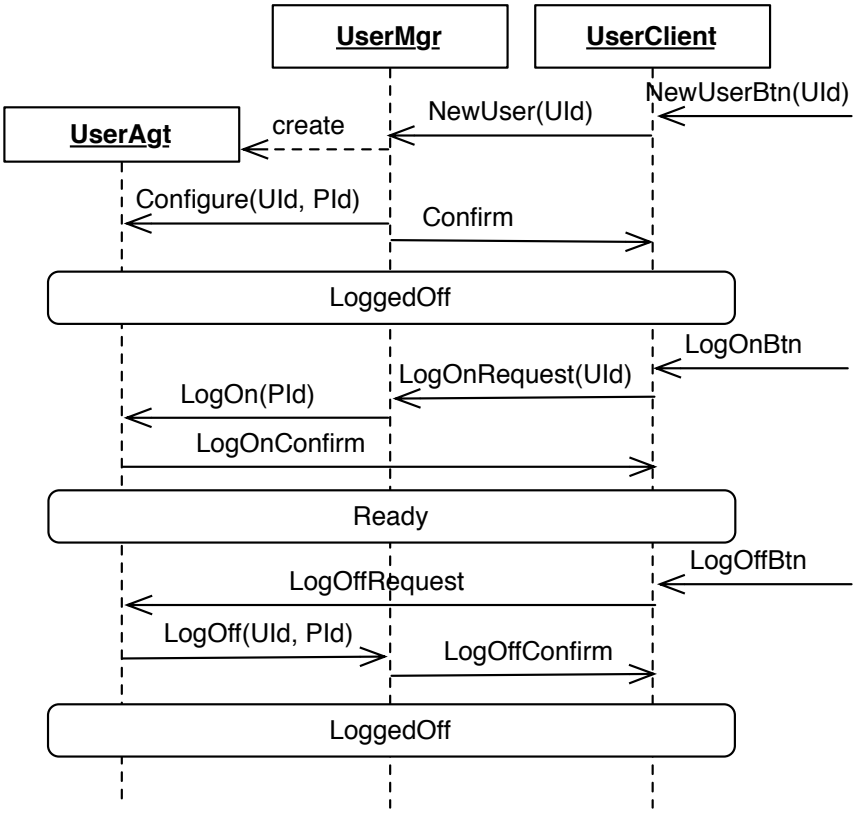


Figure 3 Creating a new UserAgent, Logging On and Logging Off

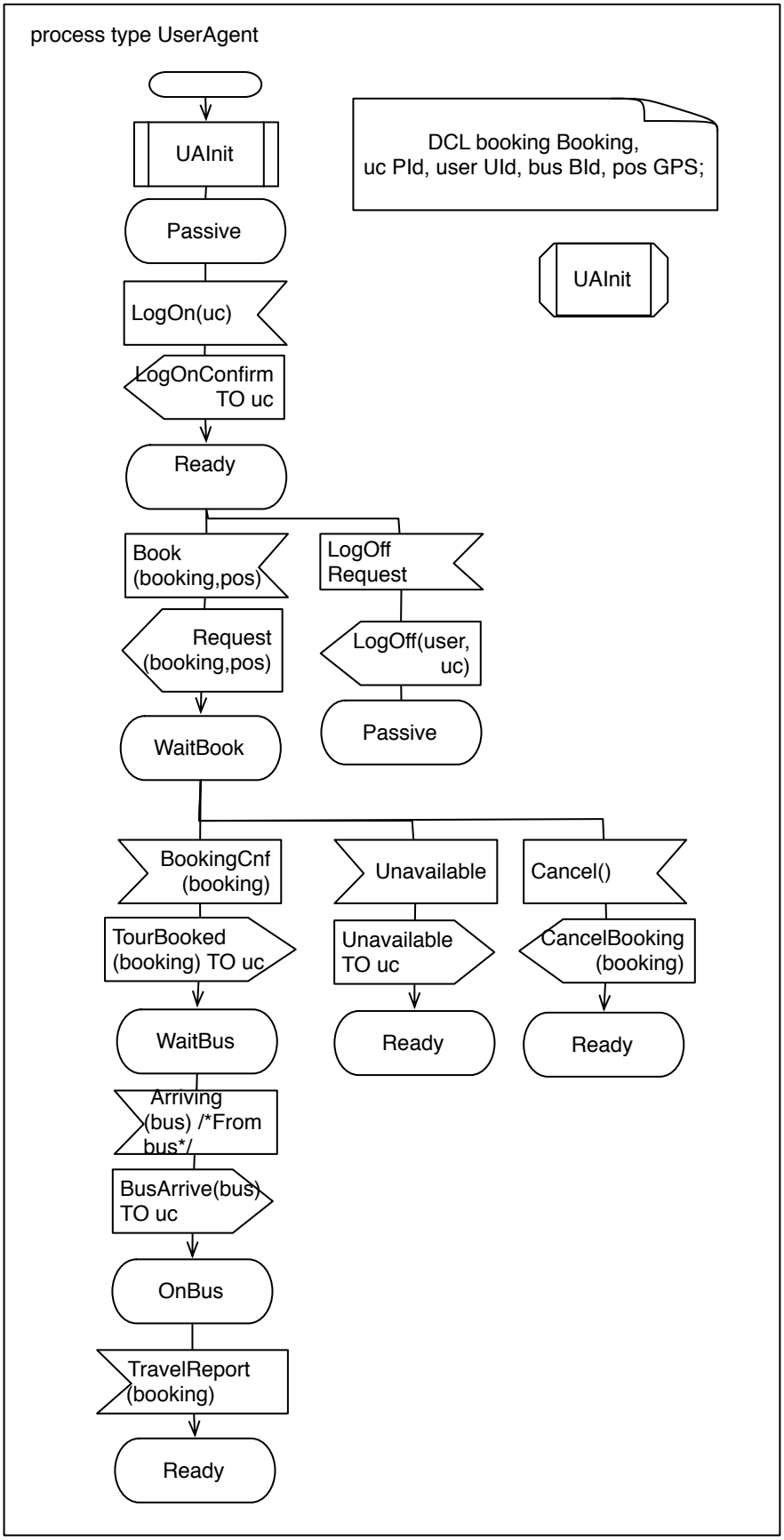


Figure 4 UserAgent behaviour

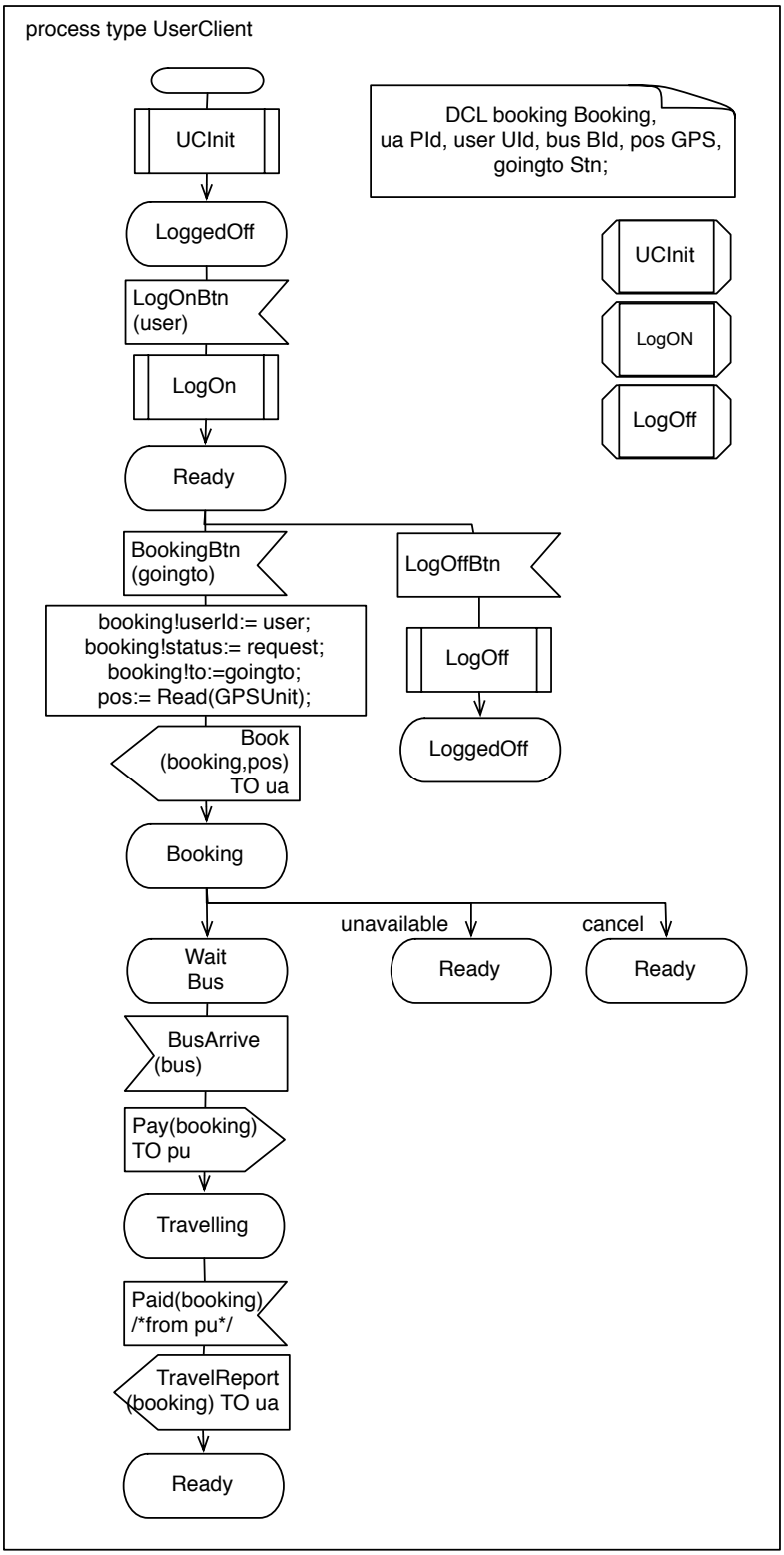


Figure 5 UserClient behaviour

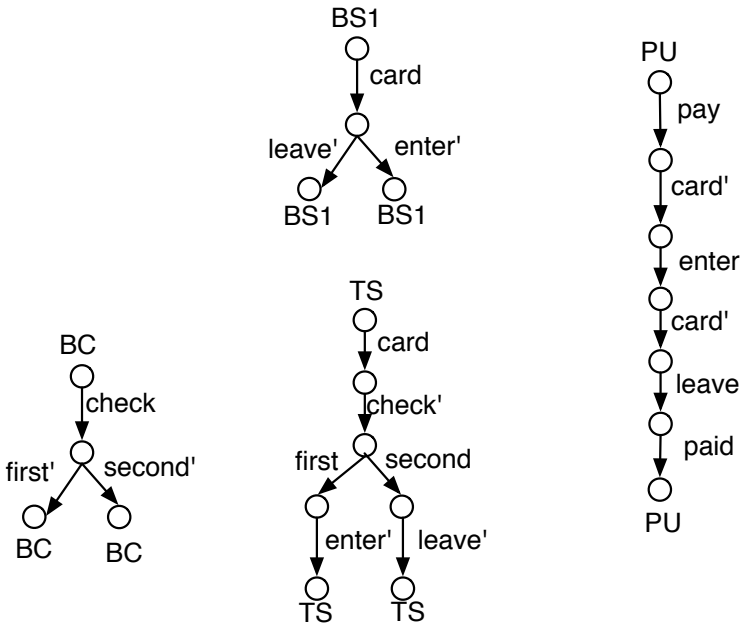


Figure 6 CCS processes in the payment system