Norges teknisk-naturvitenskapelige universitet
Institutt for telematikk

EKSAMENSOPPGAVE I TTM4115 – SYSTEMERING AV DISTRIBUERTE SANNTIDSSYSTEMER

Faglig kontakt under eksamen:	Rolv Bræk
Tlf.:	415 44 605
Eksamensdato:	28. mai 2008
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:	1
Antall sider nynorsk:	0
Antall sider engelsk:	1
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 (Eksamen utgjør 75% av sluttkarakteren.)

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

Oppgave 1. (25%) System design

1. Hvordan stemmer systemstrukturen beskrevet i Figure 1 og Figure 2 overens med reglene i SDL metoden? Hvilke regler er brukt.
2. Oppførselen til kollaborasjonen C*onsultation* er definert med MSC i Figure 4. Definer oppførselen til rollen *docpat* med SDL slik at den direkte tilsvarer oppførselen til *docpat* beskrevet i Figure 4.
3. Kontroller at rollen for *docpat* som du laget under punkt 2 er input konsistent. Gjør de endringene som eventuelt er nødvendige for at den skal bli input konsistent.
4. Hva er sammenhengen mellom roller i kollaborasjoner og klasser (*Classes*) i UML 2; hva betyr det at en rolle, som *docpat* er bundet til en klasse som *DoctorAgent*?

Oppgave 2. (25%) Oppførsel og endringer

1. Gjør kort rede for virkemåten til *RoleRequest* protokollen i *ActorFrame*.
2. I *Telemedicine* systemet skal vi bruke en variant av *RoleRequest*, kalt *AgentRequest*, se Figure 3 a) og Figure 5. Denne tar hensyn til at agentene er persistente og at forespørsler stilles i kø når det ikke er ledige agenter. Beskriv noen typiske forløp av oppførselen til *AgentRequest* med MSC.
3. Definer en SDL prosess for *agtAllo* rollen i *AgentRequest*. Anta at det finnes en datatype *Queue*, med operasjonene *insert*, *extract* og *length*, som kan benyttes.
4. Vi skal legge til funksjonalitet for at pasientene i stedet for å bruke pasientterminaler kan ringe inn via vanlige telefoner og snakke med en resepsjonist som gjør registreringen. Det skal være mulighet for at flere resepsjonister kan dele lasten. Foreslå en struktur og forklar hvordan registreringen blir gjort.

Oppgave 3. (25%) Diverse

Figure 6 viser to systemer *VM1* og *VM2* beskrevet med prosessalgebra, CCS.

1. Foreta ekspansjon av utrykket for *VM2 = IF* || *CM* || *TM*
2. Sammenlign uttrykkene for *VM1* og *VM 2*. Er de observasjonsekvivalente eller ikke? Begrunn.
3. Finn uttrykket for en omgivelse *E* for *VM1* som er slik at alle deler av *VM1* blir benyttet uten at det oppstår vranglås. Ekspander *E* || *VM1*.
4. Forklar hensikten med Tagger (Tags) i ASN.1, og gi et par eksempel på bruken.

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

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

Question 1. (25%) System design

1. How does the system structure described in Figure 1 and Figure 2 comply with the rules of the SDL method? Which rules apply?
2. The behavior of the C*onsultation* collaboration is defined in Figure 4 using MSC. Define the behavior of the *docpat* role using SDL such that it directly corresponds to the *docpat* behavior given in Figure 4.
3. Check that the *docpat* role you just designed (answer to point 2) is input consistent. Make the corrections necessary to make the role behavior input consistent.
4. What is the relationship between collaboration roles and Classes in UML2; what does it mean that a role such as *docpat* is bound to a Class such as *DoctorAgent*?

Question 2. (25%) Behavior and changes

1. Explain briefly how the *RoleRequest* protocol in *ActorFrame* works.
2. In the *Telemedicine* system we use a variant of *RoleRequest*, called *AgentRequest,* see Figure 3 a) and Figure 5. It takes into account that the agents are persistent and that requests shall be queued when no agents are available. Describe some typical cases of *AgentRequest* behavior using MSC.
3. Define an SDL process corresponding to the *agtAllo* role of *AgentRequest*. You may assume a datatype *Queue*, with the operations *insert*, *extract* and *length* that may be used.
4. We shall now add functionality that enables patients without patient terminals to call in using ordinary telephones and talk to a (human) receptionist that takes care of the registration. It shall be possible to distribute the load among several receptionists. Propose a structural addition to the system and explain how registration is done.

Question 3. (25%) Miscellaneous

Figure 6 describes two systems *VM1* and *VM2* using processalgebra, CCS.

1. Expand the expression for *VM 2= IF || CM || TM*
2. Compare the expressions for *VM1* and *VM2*. Are they observation equivalent or not? Justify.
3. Find the expression for an environment *E* of *VM1* such that all parts of *VM1* are explored without any deadlock. Expand the expression *E || VM1*.
4. Explain the purpose of Tags in ASN.1, and give a couple of examples of their use.

5.

Vedlegg/ Appendix

Bokmål

Systemet som inngår i oppgavene 1 og 2: Telemedicine
I oppgavene 1 og 2 ser vi på funksjonaliteten til et system for telemedisin, vist med UML i Figure 1.

Pasienter kommuniserer med systemet via pasientterminaler, representert som *patient[n]* i Figure 1. Pasientterminalene har utstyr for å utføre enkle tester på pasienten. Doktorer kommuniserer via legeterminaler representert som *doctor[m]* i Figure 1. I systemet er det en agent, *Patients*, som har en indre agent for hver pasient, *PatientAgent*, samt oppførsel for å administrere pasientene. Videre er det en agent, *Doctors*, som har en indre agent for hver doktor, *DoctorAgent*, samt oppførsel for å administrere doktorene. Figure 2 viser hvordan disse agentene alternativt kan representeres som SDL200 blokktypene *Patients* og *Doctors*. Vi antar at pasientagenter og doktoragenter er persistente objekter som eksisterer så lenge en pasient/doktor er registrert i systemet. De skapes altså ikke dynamisk for hver pålogging eller sesjon.

Vi tenker oss at pasientene lider av sykdommer som krever hyppige konsultasjoner med en lege. Når en pasient ønsker kontakt med en lege sender dens *PatientAgent* en forespørsel til *Doctors* i henhold til en kollaborasjon kalt *AgentRequest.* Dersom ingen doktor er ledig settes forespørselen i kø inntil en doktor blir ledig. Når en doktor er ledig for nye pasienter, vil dens *DoctorAgent* signalisere dette til sin *Doctors* agent i henhold til *AgentRequest* kollaborasjonen.

Kollaborasjonene som brukes til pålogging, *Logon*, og til å opprette sesjoner mellom pasienter og doktorer, *AgentRequest*, er vist i Figure 3 a). Kollaborasjonene som foregår under en sesjon mellom lege og pasient er vist i Figure 3 b). Oppførselen til kollaborasjonene C*onsultation* og *DoctorInterface* er gitt i Figure 4.

English

In questions 1 and 2 we consider the functionality of a telemedicine system described using UML in Figure 1.

Patients communicate with the system using patient terminals represented by *patient[n]* in Figure 1. The patient terminals have equipment to carry out simple tests on the patient. Doctors communicate with the system using doctor terminals represented by *doctor[m]* in Figure 1. The system has an agent, *Patients*, that has an inner agent for each patient, *PatientAgent*, as well as behavior to manage the patients. Another agent, *Doctors*, has an inner agent for each doctor, *DoctorAgent* as well as behavior to manage the doctors. Figure 2 depicts these agents alternatively as SDL 2000 block types *Patients* and *Doctors*. We assume that patient agents and doctor agents are persistent objects that exist as long as a patient and doctor are registered in the system. They are not created dynamically for each Logon and session.

We assume that a patient is being treated for an illness that requires frequent consultations with a doctor. When a patient wants to contact a doctor its *patientAgent* will issue a request to *Doctors* according to a collaboration called *AgentRequest*. If there are no free doctors, the request is queued until a doctor becomes available. When a doctor is free to accept a new patient its *DoctorAgent* will signal this to the *Doctors* agent according to the *AgentRequest* collaboration.

The *Logon* collaborations as well as the *AgentRequest* collaboration is shown in Figure 3 a). The collaborations taking place during a session are shown in Figure 3 b). The behavior of collaborations *Consultation* and *DoctorInterface* is given in Figure 4.

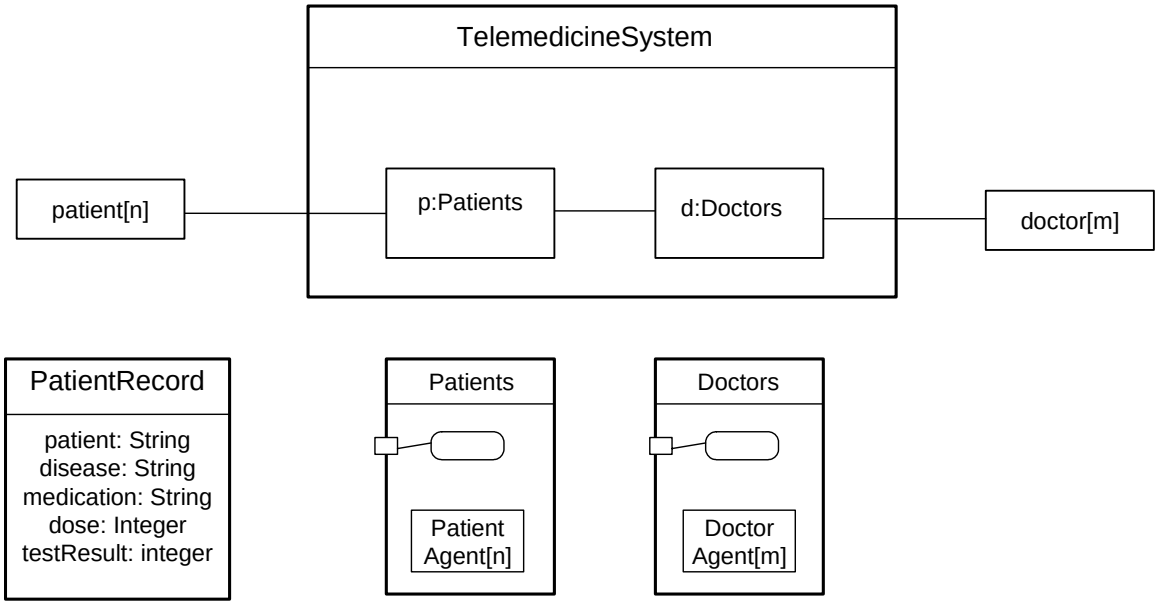


Figure 1 The Telemedicine system

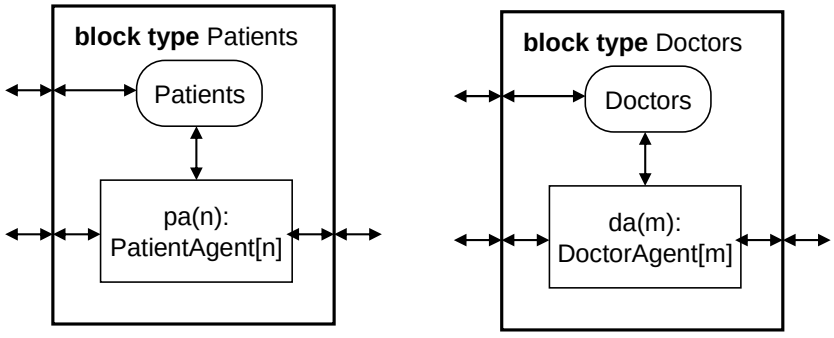
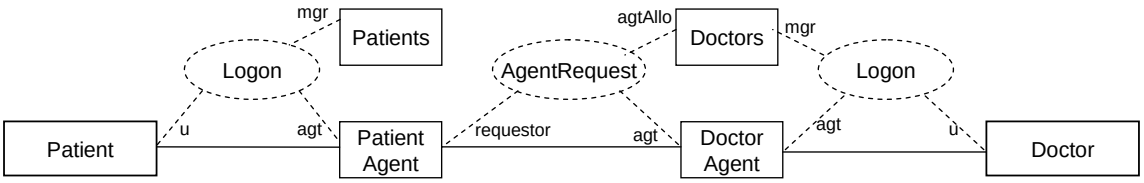
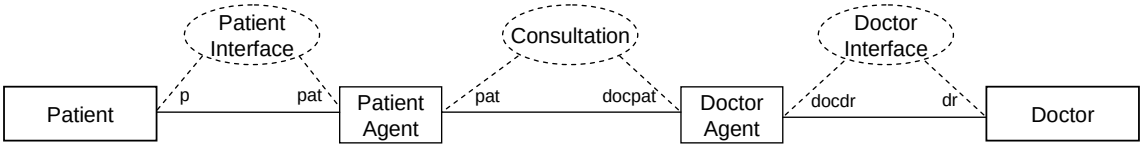


Figure 2 Alternative description of Patients and Doctors as SDL 2000 block types



a) Collaborations for logging on and for setting up sessions between patients and doctors



b) Collaborations in an established session between a patient and a doctor

Figure 3 Collaborations

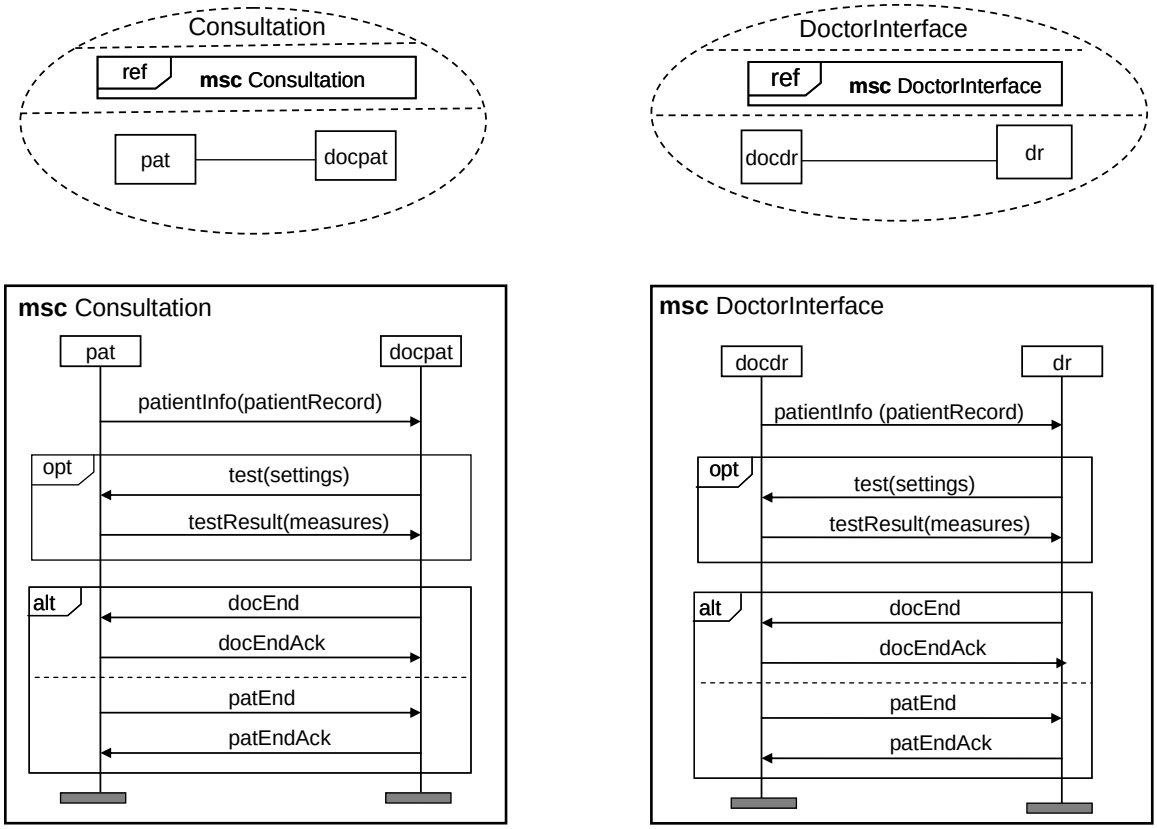


Figure 4 The *Consultation* and the D*octorInterface* collaboration

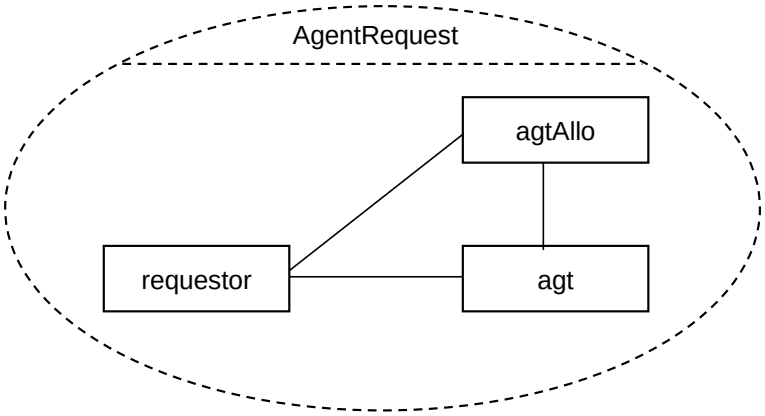
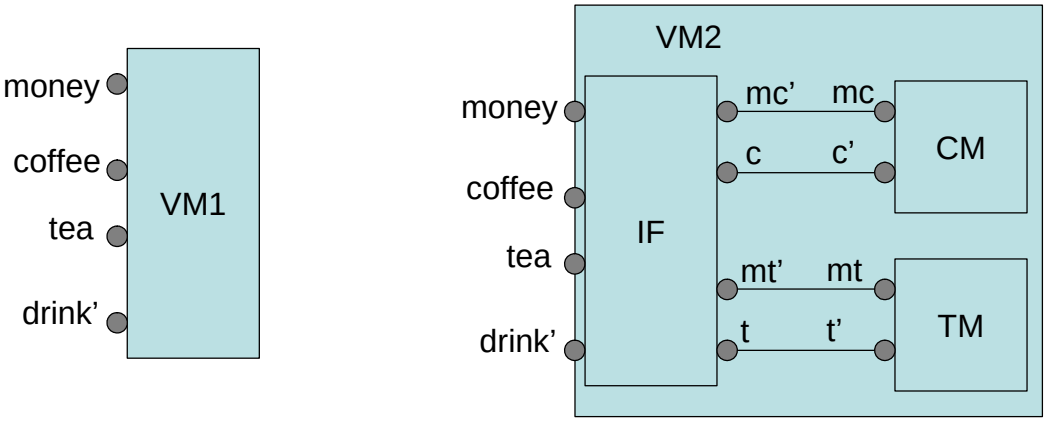


Figure 5 The *agentRequest* collaboration structure



VM1 = money;
((coffee; τ; drink'; VM1)
+ (tea; τ; drink'; VM1))

VM2 = IF || CM || TM

IF = money;
((coffee; mc'; c; drink'; IF)
+ (tea; mt'; t; drink'; IF))
CM = mc; c'; CM
TM = mt; t'; TM

Figure 6 Vending Machines VM1 and VM2