Tallinna Tehnikaülikool
Informaatikainstituut

Raviasutust külastava patsiendi assistent

Projekt aines „Agentorienteeritud modelleerimine ja multiagentsüsteemid“

Koostaja:
Villu Jürimaa
124495 IABMM
villuj@hot.ee

Juhendaja:
Kuldar Taveter

Tallinn 2013

Sisukord

Sissejuhatus

Käesolev töö on iseseisev ainetöö õppeaines "Agentorienteeritud modelleerimine ja multiagentsüsteemid". Töö eesmärgiks on koostada patsienti raviasutuses viibimise ajal abistava interaktiivse assistendi funktsionaalsuse kirjeldus ning mudelid, lähtudes agentidel baseeruva süsteemi põhimõtetest.

Raviasutuse all võib mõelda lisaks tavapärastele haiglatele ka kõiksugu teisi ravi- ning terviseteenuseid pakkuvaid asutusi (sanatooriumid, taastusravihaiglad jne).

Eesmärgimudelid

Süsteemi põhieesmärgiks on pakkuda patsiendile mugavat ning informeeritud raviasutuses viibimist. See tähendab:

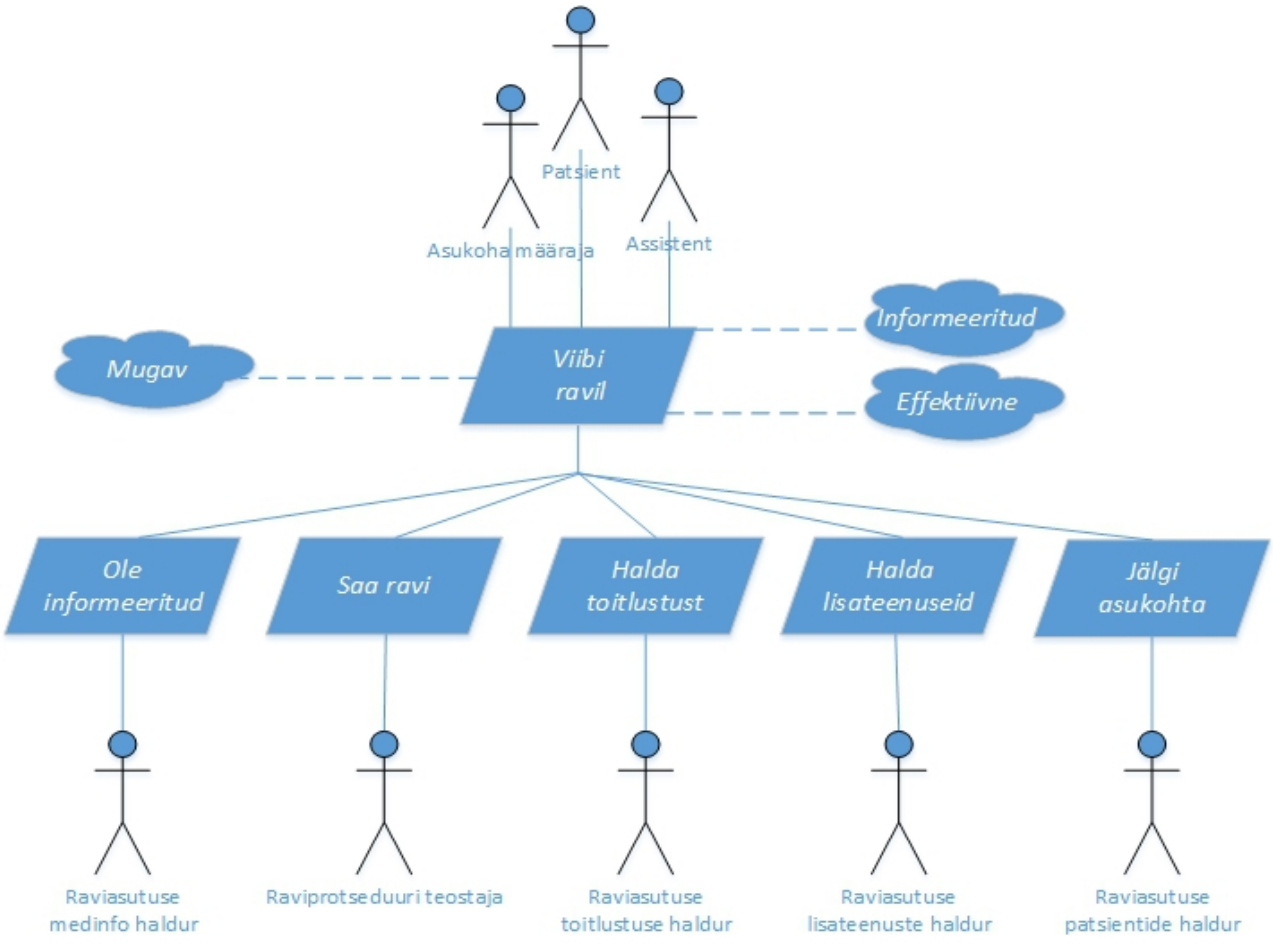
- patsiendi teadlikkust oma ravitoimingutest;
- võimalust olla teadlik oma päevakavast ning võimalusel selle kujundamisel osaleda;
- võimalust tellida mugavalt lisateenuseid (s.h. toitlustust).

Dokumendi lõpuosas leiavad detailsemat kajastamist loetelu kaks viimast põhieesmärki, kuna on loogiliselt keerukamad.

Edaspidsele süsteemi arendusvõimalustele mõeldes on antud töös ka patsiendi asukoha määramine, kuid seda infot otsustuste juures ei kasutata.

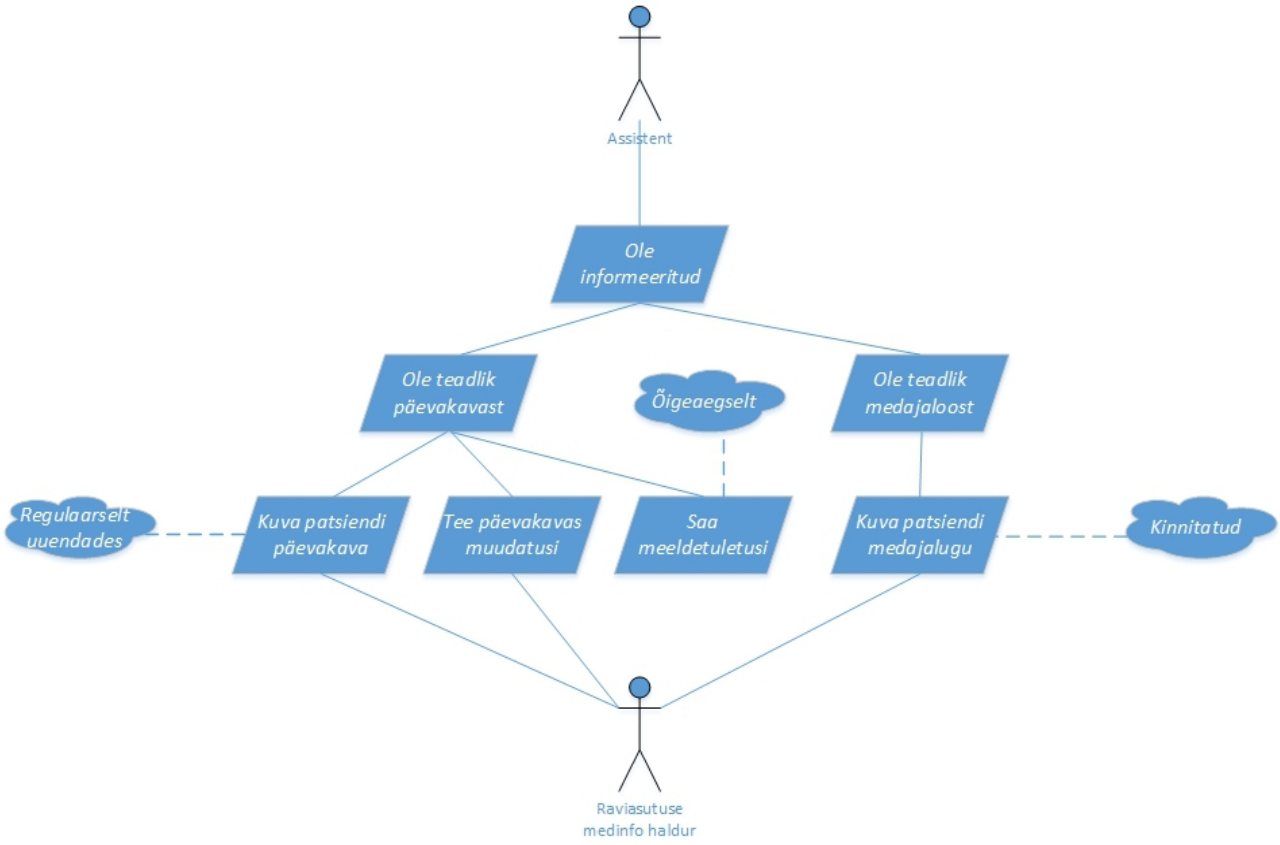
Süsteemil praeguse skoobi juures on kasutusel seitse põhirolli. Kolm neist on otseselt seotud patsiendiga ning toetavad ja võimaldavad tema tegevusi. Ülejäänud neli tegelevad raviasutuse poolel erinevate patsiendiga seotud toimingutega.

Järgmine mudel esitab süsteemi põhieesmärgid ning nendega seotud rollid.



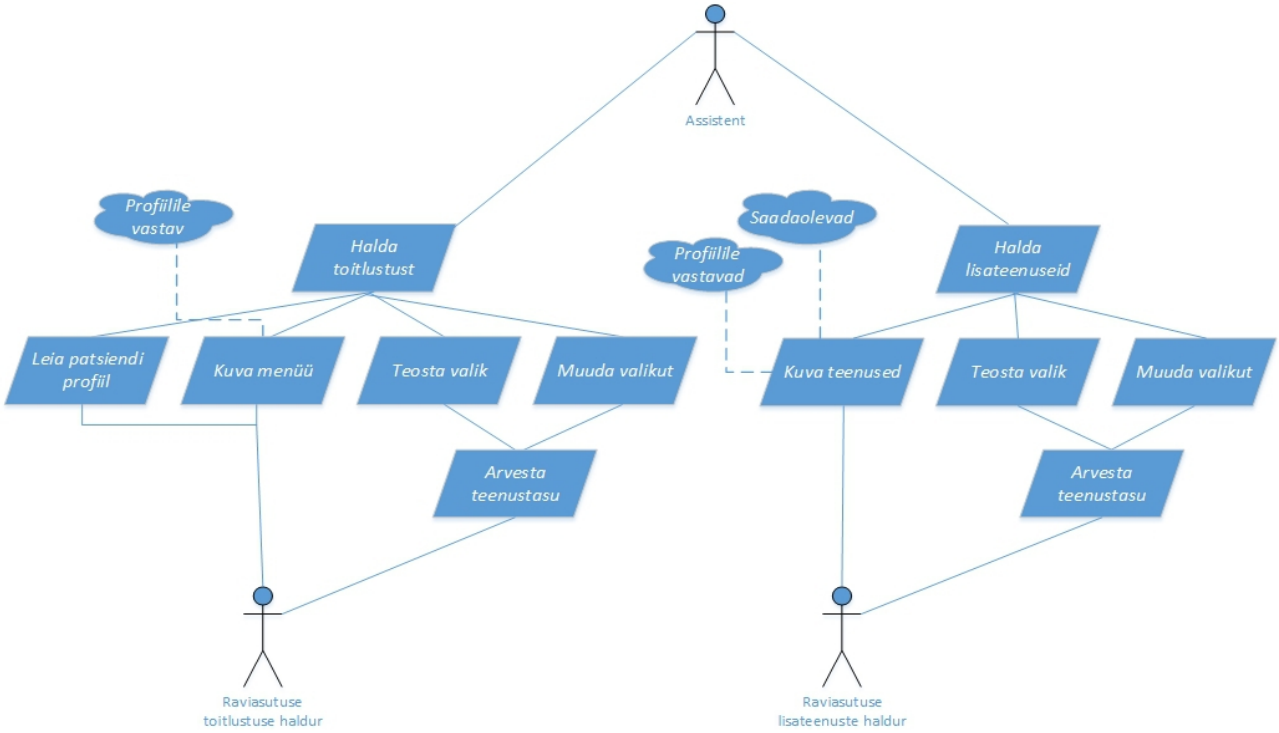
Patsiendi informeerituse eesmärgimudel

Järgnev mudel kujutab patsiendi informeerituse eesmärgiga seotud alameesmärke ning rolle.



Lisateenuste haldamise eesmärgimudel

Järgnev mudel kujutab alameesmärke ning rolle, mis on seotud põhieesmärgiga pakkuda patsiendile võimalus hallata oma lisateenuseid.



Rollimudel

Järgnevalt on kirjeldatud süsteemiga seotud rollid ning nende ülesanded.

Rolli nimi	Patsient
Kirjeldus	Ravi saaja
Ülesanded	Käia raviprotseduuridel Informeerida assistenti soovitud päevakavast Informeerida assistenti soovitud lisateenustest
Lisatingimused	

Rolli nimi	Patsiendi assistent
Kirjeldus	Patsiendi assistent, mis vahendab patsiendi ning medasutuse teiste agentide vahelist suhtlust
Ülesanded	Pärida medasutuselt patsiendi päevakava Informeerida patsienti päevakavast Edastada medasutusele soovitud päevakava Edastada patsiendile päevakava meeldetuletusi Pärida rav. toitl. haldurilt võimalik menüü Informeerida patsienti võimalikust menüüst Edastada rav. toitl. haldurile patsiendi soovitud menüü Pärida rav. lis. haldurilt võimalikud lisateenused Informeerida patsienti võimalikest lisateenustest Edastada rav. lis. haldurile patsiendi soovitud lisateenused
Lisatingimused	Päevakavas muudatuste tegemine peab olema patsiendile mugav Meeldetuletused peavad toimuma õigeaegselt

Rolli nimi	Asukoha määraja
Kirjeldus	Patsiendi asukoha määraja
Ülesanded	Määrata patsiendi asukoht Edastada assistendile info patsiendi asukoha kohta Edastada patsientide haldurile info patsiendi asukoha kohta
Lisatingimused	Info edastamine peab toimuma regulaarselt

Rolli nimi	Raviprotseduuri teostaja
Kirjeldus	Patsiendi raviprotseduuride teostaja
Ülesanded	Teostada raviprotseduure
Lisatingimused	

Rolli nimi	Raviasutuse toitlustuse haldur
Kirjeldus	Raviasutuse toitlustust reguleeriv süsteem
Ülesanded	Informeerida patsiendi assistenti võimalikust menüüst Võtta vastu patsiendi soovitud menüü Tekitada tasuliste teenuste puhul patsiendile arve
Lisatingimused	Võimalik menüü peab vastama patsiendi profiilile

Rolli nimi	Raviasutuse lisateenuste haldur
Kirjeldus	Raviasutuse lisateenuseid reguleeriv süsteem
Ülesanded	Informeerida patsiendi assistenti patsiendi profiilile vastavast võimalikust menüüst Võtta vastu patsiendi soovitud menüü Tekitada tasuliste teenuste puhul patsiendile arve
Lisatingimused	Võimalikud lisateenused peavad vastama patsiendi profiilile

Rolli nimi	Raviasutuse medinfo haldur
Kirjeldus	Raviasutuse meditsiinilist infot haldav süsteem
Ülesanded	Edastada patsiendi assistendile patsiendi päevakava Võtta patsiendi assistendilt vastu soovitud päevakava muudatused Edastada patsiendi assistendile patsiendi meditsiiniline ajalugu
Lisatingimused	Patsiendi päevakava edastades peab andma ette võimalike muudatuste kohad. Patsiendi soovitud päevakava vastu võttes tuleb see valideerida ning tagastada vajadusel keeldumisteade arusaadava põhjendusega. Patsiendi medajaloo esitamisel ei tohi edastada kinnitamata andmeid.

Rolli nimi	Raviasutuse patsientide haldur
Kirjeldus	Raviasutuse patsientide infot haldav süsteem
Ülesanded	Määrata patsiendi profiil Tagastada küsijatele patsiendi profiil Võtta vastu info patsiendi asukoha kohta
Lisatingimused	

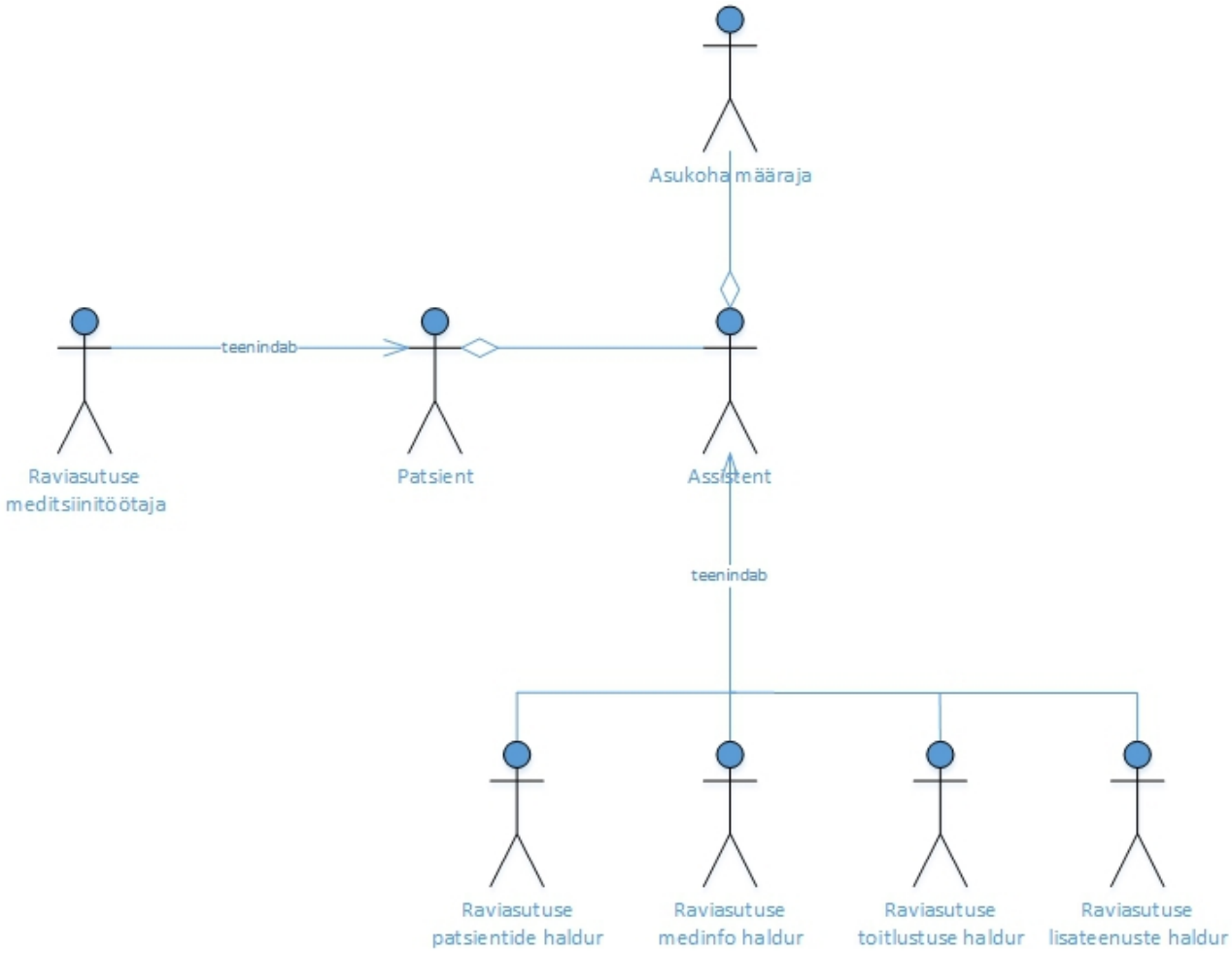
Organisatsioonimudel

Järgnevas mudelis on näidatud töö skoobis olevate rollide vahelised seosed.

Patsienti teenindab Raviasutuse meditsiinitöötaja. Patsient suhtleb ülejäänud süsteemidega läbi Assistendi.

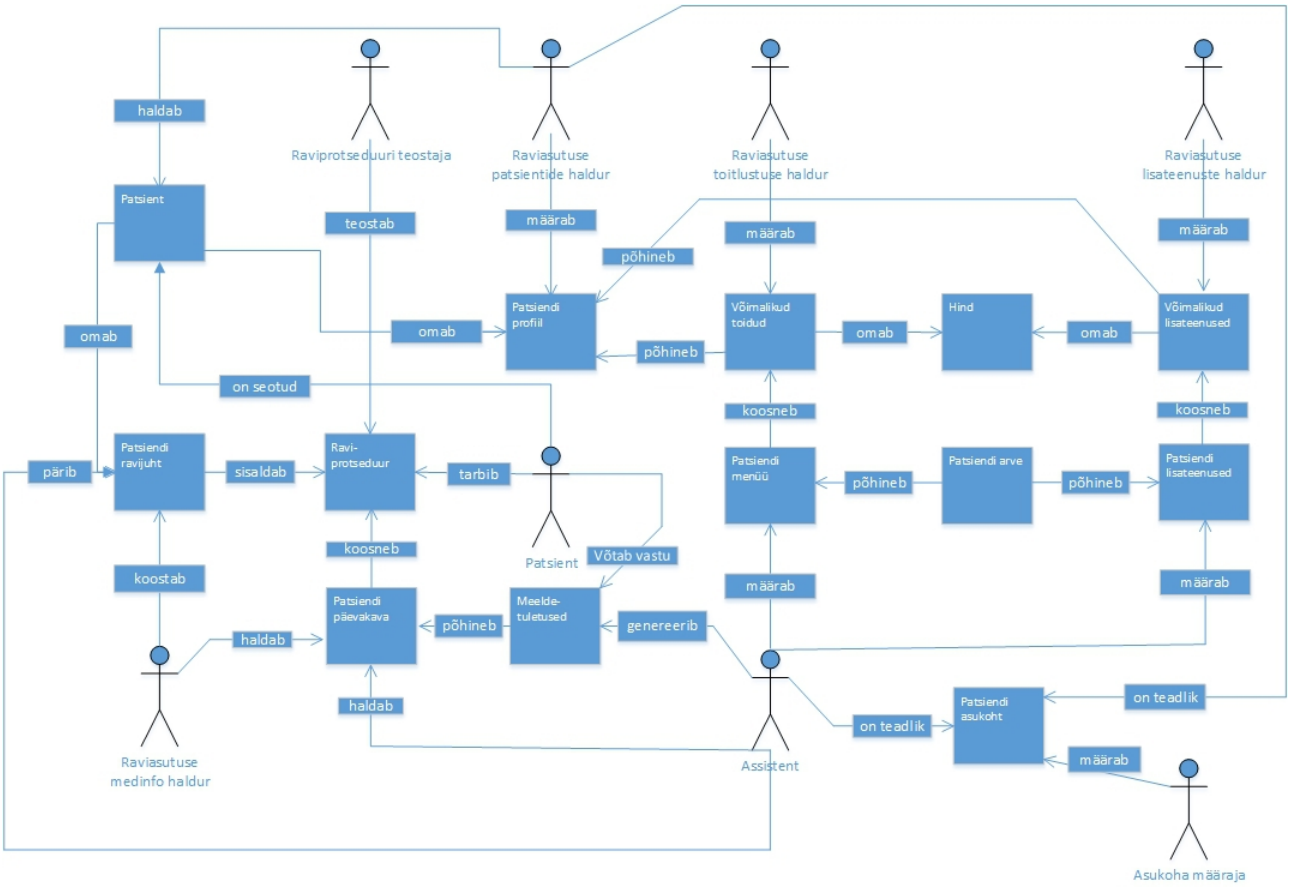
Assistendi juurde kuulub patsiendi asukoha määraja, millelt assistent saab patsiendi asukoha info ning edastab selle Raviasutuse patsientide haldurile.

Raviasutuse neli rolli teenindavad assistenti oma spetsiifilise funktsionaalsusega.



Domeenimudel

Järgnev mudel näitab kasutajarollide teadlikkust erinevast süsteemis paiknevast ning liikuvast informatsioonist.



Agentide mudel

Järgnevalt on toodud süsteemi realiseerimisel loodavad reaalsed agendid, nende seotus rollidega ning teadlikkus süsteemis paikevast informatsioonist.

Agendi nimi	Patsiendi assistent
Kirjeldus	Digitaalne assistent, mis abistab patsiendi ning raviasutuse tegevusi patsiendi ravil viimisel ajal
Roll(id)	Assistent, Asukoha määraja
Teadlikkus	Patsiendi asukoht, Patsiendi päevakava, Patsiendi haigusjuht, Patsiendi menüü, Patsiendi lisateenused, Meeldetuletused

Agendi nimi	Patsient
Kirjeldus	Füüsiline isik, kes raviasutuses viibib
Roll(id)	Patsient
Teadlikkus	Patsiendi profiil, Meeldetuletused, Raviprotseduur

Agendi nimi	Meditsiiniinfosüsteem
Kirjeldus	Raviasutuse meditsiinilist infot haldav süsteem
Roll(id)	Raviasutuse medinfo haldur, Raviasutuse patsientide haldur
Teadlikkus	Patsiendi haigusjuht, Patsiendi päevakava, Patsiendi asukoht

Agendi nimi	Lisateenuste süsteem
Kirjeldus	Raviasutuse poolt pakutavaid lisateenuseid haldav süsteem
Roll(id)	Raviasutuse toitlustuse haldur, Raviasutuse lisateenuste haldur
Teadlikkus	Võimalikud toidud, Võimalikud lisateenused, Patsiendi profiil, Patsiendi menüü, Patsiendi lisateenused, Patsiendi arve

Agendi nimi	Meditsiinitöötaja
Kirjeldus	Raviasutuse poolt pakutavaid raviteenuseid teostav isik
Roll(id)	Raviprotseduuri teostaja
Teadlikkus	Raviprotseduur, Patsiendi haigusjuht, Patsiendi päevakava

Tutvusmudel

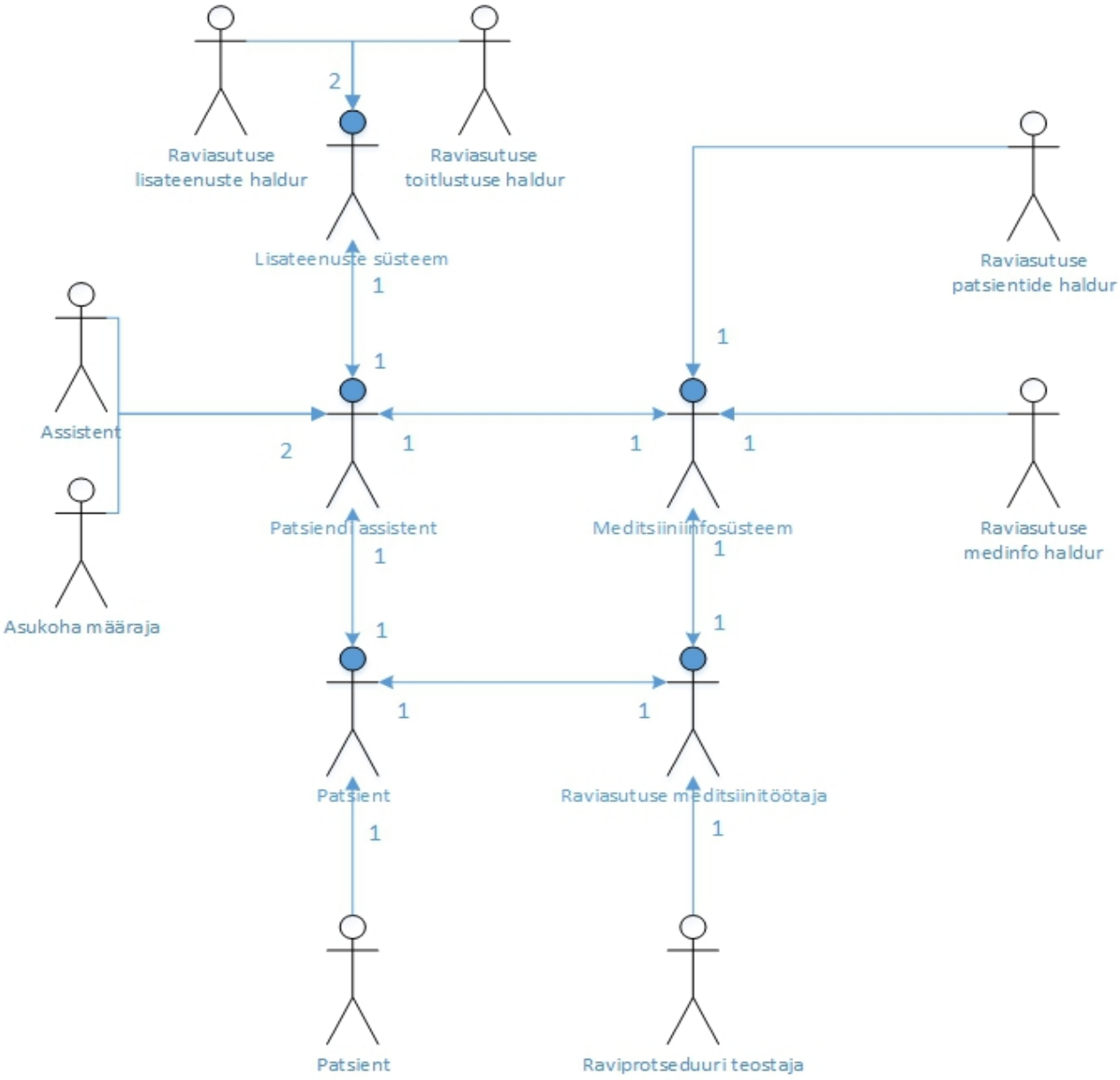
Järgnevalt on kujutatud süsteemis paiknevad agendid ning nendega seotud rollid (valged pead tähistavad rolle, sinised pead agente).

Lisateenuste süsteemi agent täidab nii toitlustuse kui lisateenuste halduri rolle (kuna nende rollide ülesanded on võrdlemisi sarnased).

Patsiendi assistendi agent täidab nii assistendi kui asukoha määraja rollide ülesandeid.

Meditsiiniinfosüsteem on raviasutuse patsientide meditsiinilist infot haldav agent, mis antud töö skoobis täidab patsientide halduri ning medinfo halduri rollide ülesandeid (reaalseid ülesandeid on rohkem).

Patsiendi ning Raviteenuste teostaja agendid on reaalsed füüsilised isikud ning täidavad kumbki oma rolli ülesandeid. Antud töö skoobis on meditsiinitöötaja rolli ülesandeks pakkuda patsiendi rollile raviteenuste teostamise teenust.

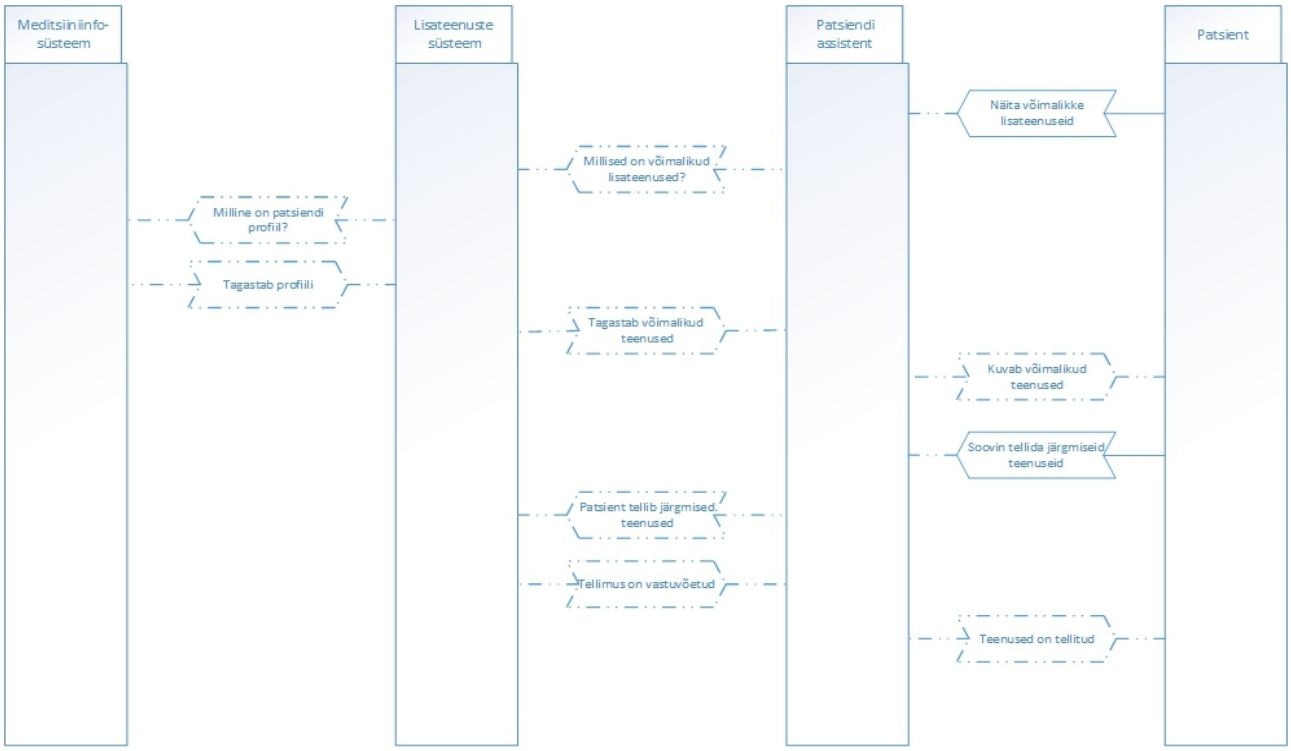


Suhtlusmudelid

Järgenvalt on kujutatud süsteemi põhifunktsionaalsuse juures toimuv suhtlus süsteemi agentide vahel.

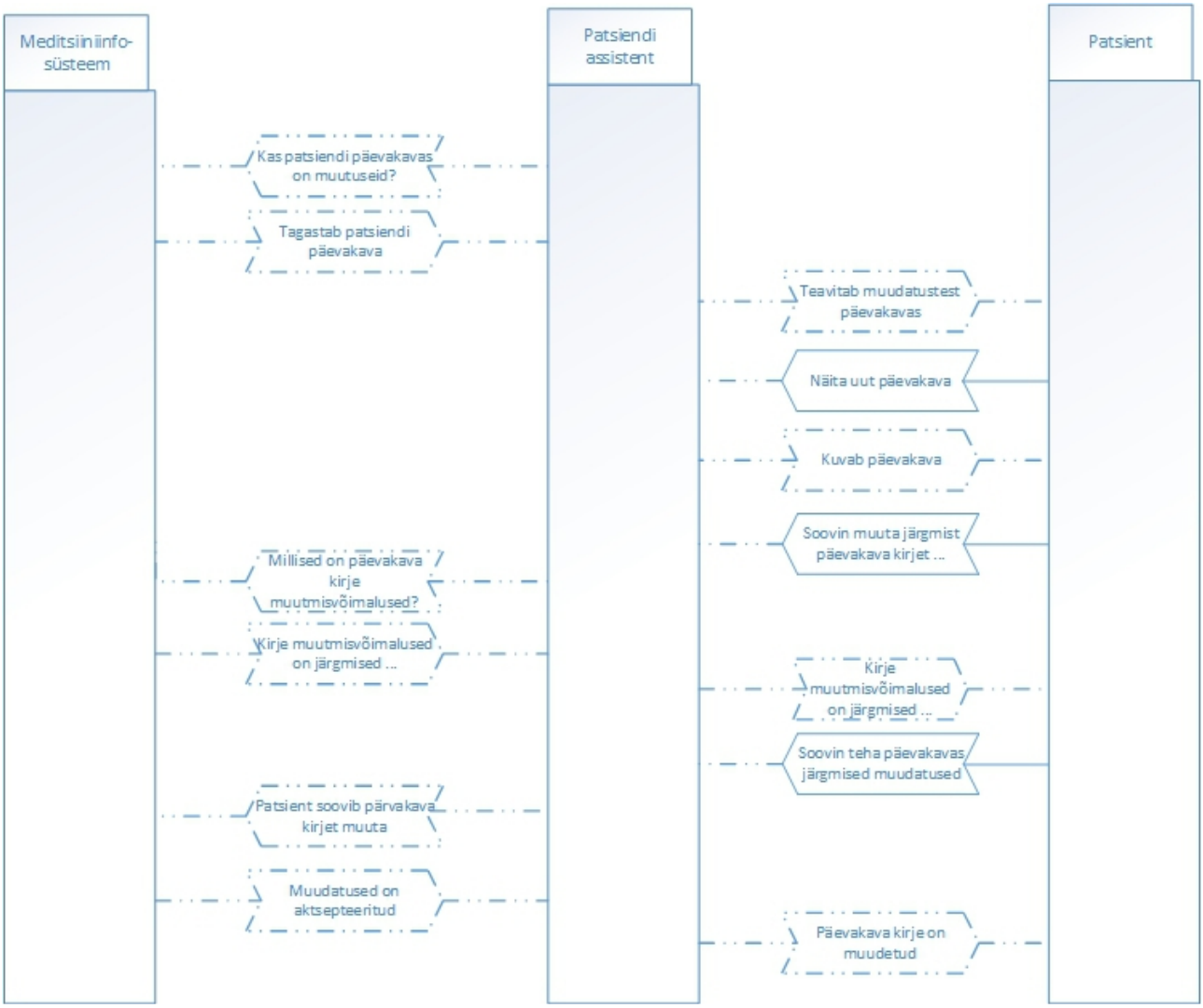
Lisateenuste haldamise suhtlusmudel

Lisateenuste haldamisel pärib Patsient Patsiendi assistendilt võimalikke lisateenuseid. Patsiendi assistent edastab päringu Lisateenuste süsteemile. Lisateenuste süsteem küsib Meditsiiniinfosüsteemilt patsiendi profiili ning tagastab Patsiendi assistendile profiilile vastavad võimalikud teenused. Juhul, kui patsient soovib mõnda lisateenust tellida, edastab ta Patsiendi assistendi kaudu vastava päringu lisateenuste süsteemile, mis omakorda vastab kas õnnestumise või toimingu ebaõnnestumise teatega (näiteks on teenus juba mõne teise patsiendi poolt kasutuses, mis ei olnud nii veel lisateenuste nimekirja pärimise hetkel).



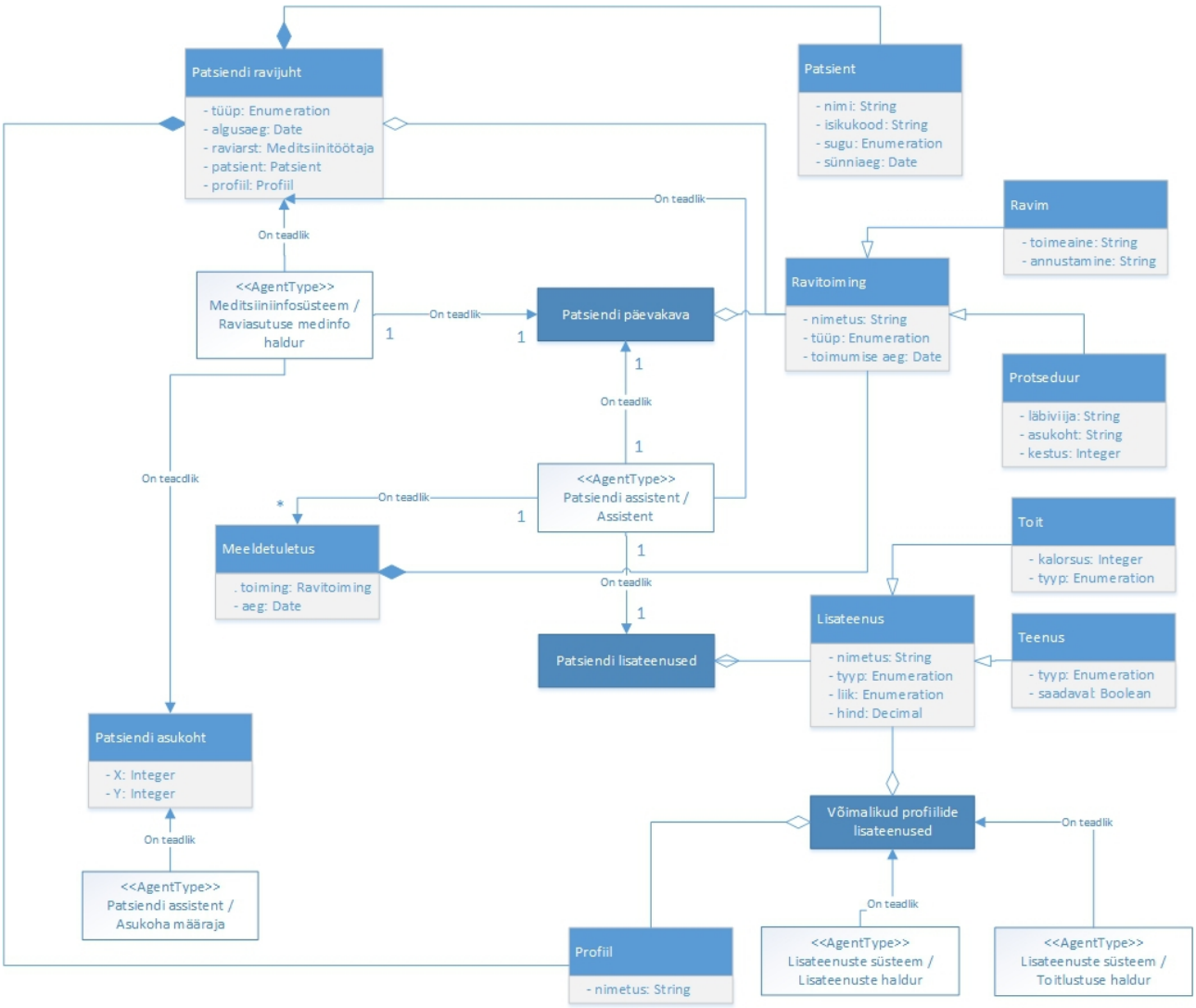
Päevakava haldamise suhtlusmudel

Päevakava haldamisel pärib Patsiendi assistent regulaarselt Meditsiininfosüsteemilt patsiendi päevakava. Muudatuste korral teavitab neist Patsienti. Juhul, kui Patsient soovib mõnda päevakava kirjet muuta, pärib Patsiendi assistent Meditsiiniinfosüsteemilt antud kirje muutmisvõimalused ning tagastab need Patsiendile. Patsient edastab soovi korral Patsiendi assistendile soovitud muudatuse info. Patsiendi assistent kooskõlastab muudatuse Meditsiiniinfosüsteemiga ning tagastab Patsiendile tulemuse kohta teate.



Teadmusmudel

Järgnevalt on esitatud antud töö skoobis olevate agentide teadlikkus süsteemis olevast informatsioonist.



Stsenaariumid

Järgnevalt on esitatud süsteemi töös eksisteerivad olulisemad stsenaariumid, alustades üldisest põhistsenaariumist.

STSENAARIUM 1					
Eesmärk	Ravil viibimine				
Algataja	Patsient				
Triger	Patsient saabub raviasutusse				
KIRJELDUS					
Tingimus	Samm	Tegevus	Agendi tüübid ja rollid	Ressursid	Kvaliteedieesmärgid
Samaaegselt		Jälgi ravijuhtumit	Patsient/Patsient, Patsiendi assistent/Assistent, Meditsiiniinfosüsteem/Raviasutuse medinfo haldur	Patsiendi ravijuht, Ravitoiming	
		Halda päevakava (Stsenaarium 2)	Patsient/Patsient, Patsiendi assistent/Assistent, Meditsiiniinfosüsteem/Raviasutuse medinfo haldur	Patsiendi päevakava, Ravitoiming, Meeldetuletus	Mugavalt
		Halda lisateenuseid (Stsenaarium 3)	Patsient/Patsient, Patsiendi assistent/Assistent, Lisateenuste süsteem/Raviasutuse toitlustuse haldur, Lisateenuste süsteem/Raviasutuse lisateenuste haldur	Võimalikud lisateenused, Lisateenus, Patsiendi lisateenused, Profiil	Mugavalt
		Saa ravi	Patsient/Patsient, Meditsiinitöötaja/Raviprotseduuri teostaja	Patsiendi päevakava, Ravitoiming	
		Jälgi asukohta	Patsiendi assistent/Asukoha määraja, Meditsiiniinfosüsteem/Raviasutuse	Patsiendi asukoht	Täpse

			patsientide haldur		
STSENAARIUM 2					
Eesmärk	Halda päevakava				
Algataja	Patsiendi assistent				
Triger	Tekib vajadus pärida päevakava				
KIRJELDUS					
Tingimus	Samm	Tegevus	Agendi tüübid ja rollid	Ressursid	Kvaliteedieesmärgid
Tsükliliselt	1	Päri patsiendi päevakava	Patsiendi assistent/Assistent, Meditsiiniinfosüsteem/Raviasutuse medinfo haldur		
Järjestikkuselt	2	Leia ravitoimingute põhjal patsiendi päevakava	Meditsiiniinfosüsteem/Raviasutuse medinfo haldur	Patsiendi ravijuht, Ravitoiming, Patsiendi päevakava	Kiire
	4	Edasta pärijale patsiendi päevakava info	Meditsiiniinfosüsteem/Raviasutuse medinfo haldur, Patsiendi assistent/Assistent,		
	5	Kontrolli, kas saadud päevakavas oli muudatusi (võrreldes salvestatuga)	Patsiendi assistent/Assistent	Patsiendi päevakava	
	6	Salvesta uus päevakava	Patsiendi assistent/Assistent	Patsiendi päevkava	
	7	Teavita patsienti päevakava muudatustest	Patsiendi assistent/Assistent, Patsient/Patsient		
	8	Võta patsiendilt vastu päevakava kirje muutmissoov	Patsiendi assistent/Assistent, Patsient/Patsient		
	9	Edasta patsiendilt saadud päevakava kirje muutmissoov	Patsiendi assistent/Assistent, Meditsiiniinfosüsteem/Raviasutuse medinfo haldur		

	10	Leia võimalikud päevakava muutmisvõimalused	Meditsiiniinfosüsteem/Raviasutuse medinfo haldur	Patsiendi ravijuht, Ravitoiming, Patsiendi päevakava	Korrektne, Kiire
	11	Teavita kirje võimalikest muudatustest / muutmisvõimaluste puudumisest	Meditsiiniinfosüsteem/Raviasutuse medinfo haldur, Patsiendi assistent/Assistent		
	12	Teavita kirje võimalikest muudatustest / muutmisvõimaluste puudumisest	Patsiendi assistent/Assistent, Patsient/Patsient		
	13	Edasta soovitud muudatus	Patsient/Patsient, Patsiendi assistent/Assistent		
	14	Edasta soovitud muudatus	Patsiendi assistent/Assistent, Meditsiiniinfosüsteem/Raviasutuse medinfo haldur		
	15	Teavita muudatuste aktsepteerimisest (salvesta muudatused) / mitteaktsepteerimisest	Meditsiiniinfosüsteem/Raviasutuse medinfo haldur, Patsiendi assistent/Assistent	Patsiendi päevkava	
	16	Teavita patsienti muudatuste aktsepteerimisest / mitteaktsepteerimisest	Patsiendi assistent/Assistent, Patsient/Patsient		
	17	Muudatuste aktsepteerimisel salvesta päevakava	Patsiendi assistent/Assistent	Patsiendi päevkava	
	18	Genereeri meeldetuletused	Patsiendi assistent/Assistent	Meeldetuletus	Korrektne
Tsükliliselt	19	Edasta patsiendile meeldetuletused	Patsiendi assistent/Assistent, Patsient/Patsient		Õigeaegne

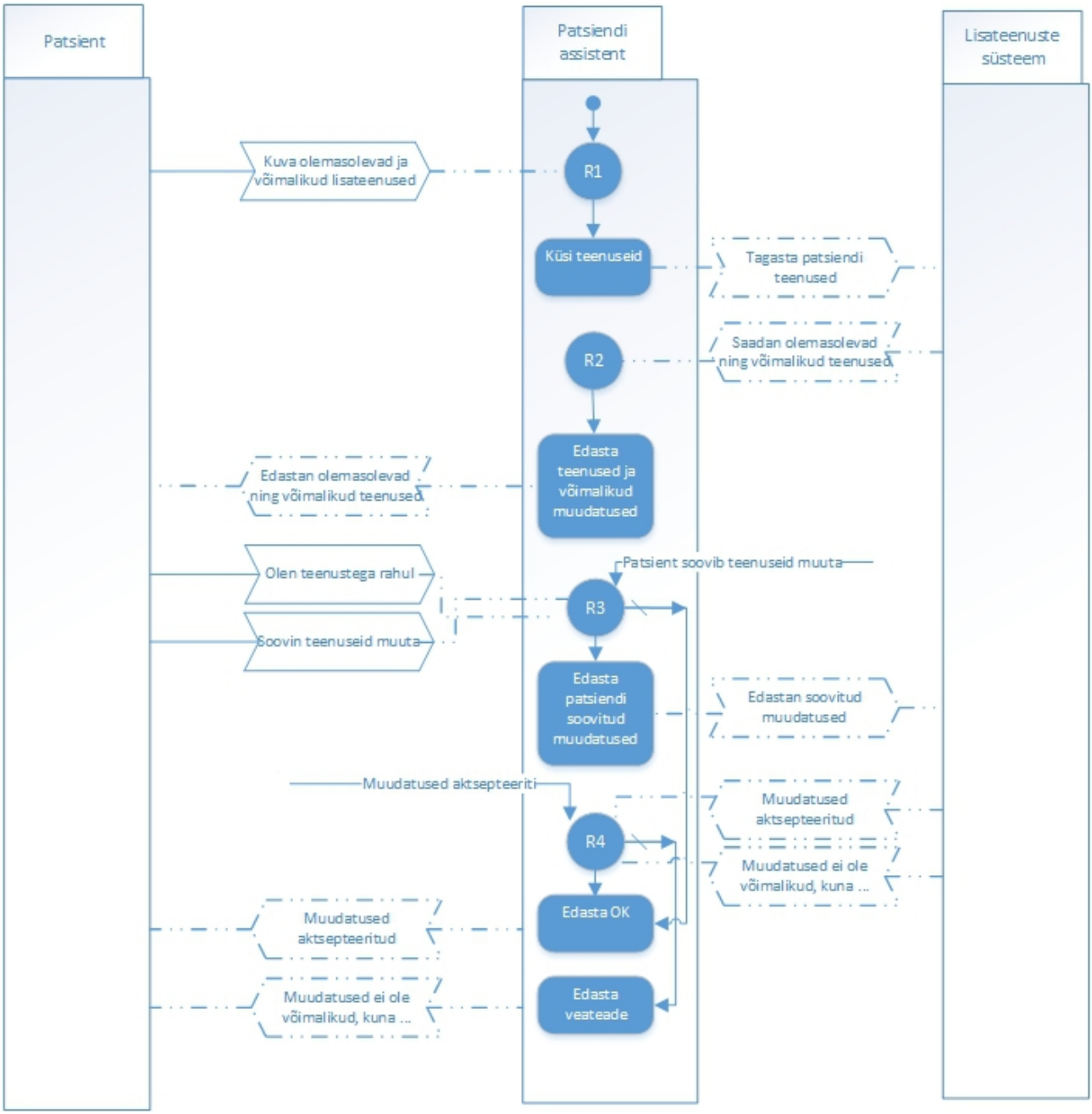
STSENAARIUM 3					
Eesmärk	Halda lisateenuseid				
Algataja	Patsient				
Triger	Patsient soovib tellida või muuta lisateenuseid				
KIRJELDUS					
Tingimus	Samm	Tegevus	Agendi tüübid ja rollid	Ressursid	Kvaliteedieesmärgid
Järjestikkuselt	1	Päri olemasolevaid ning võimalikke lisateenuseid	Patsient/Patsient, Patsiendi assistent/Assistent		
	3	Päri patsiendi olemasolevaid ning võimalikke lisateenuseid	Patsiendi assistent/Assistent, Lisateenuste süsteem/Raviasutuse lisateenuste haldur või Lisateenuste süsteem/Raviasutuse toitlustuse haldur		
	4	Leia patsiendi profiil	Lisateenuste süsteem, Meditsiiniinfosüsteem/Raviasutuse medinfo haldur	Profiil	Kiire
	5	Leia profiilile vastavad ning saadaolevad võimalikud lisateenused	Lisateenuste süsteem	Võimalikud lisateenused, Profiilide lisateenused	Kiire
	6	Tagasta olemasolevad ning võimalikud lisateenused	Lisateenuste süsteem, Patsiendi assistent/Assistent		
	7	Edasta patsiendile olemasolevad ning võimalikud lisateenused	Patsiendi assistent/Assistent, Patsient/Patsient		
	8	Edasta lisateenuste muutmissoovid	Patsient/Patsient, Patsiendi assistent/Assistent		Mugav
	9	Edasta lisateenuste muutmissoovid	Patsiendi assistent/Assistent, Lisateenuste süsteem		

	10	Kontrolli teenuste saadavust	Lisateenuste süsteem	Võimalikud lisateenused	Korrektne
	11	Fikseeri teenused ning redigeeri patsiendi arvet	Lisateenuste süsteem	Patsiendi lisateenused	
	12	Edasta teade toimingu õnnestumisest / ebaõnnestumisest	Lisateenuste süsteem/Patsiendi assistent		
	13	Edasta teade toimingu õnnestumisest / ebaõnnestumisest	Patsiendi assistent/Assistent, Patsient/Patsient		

Agentide käitumismudelid

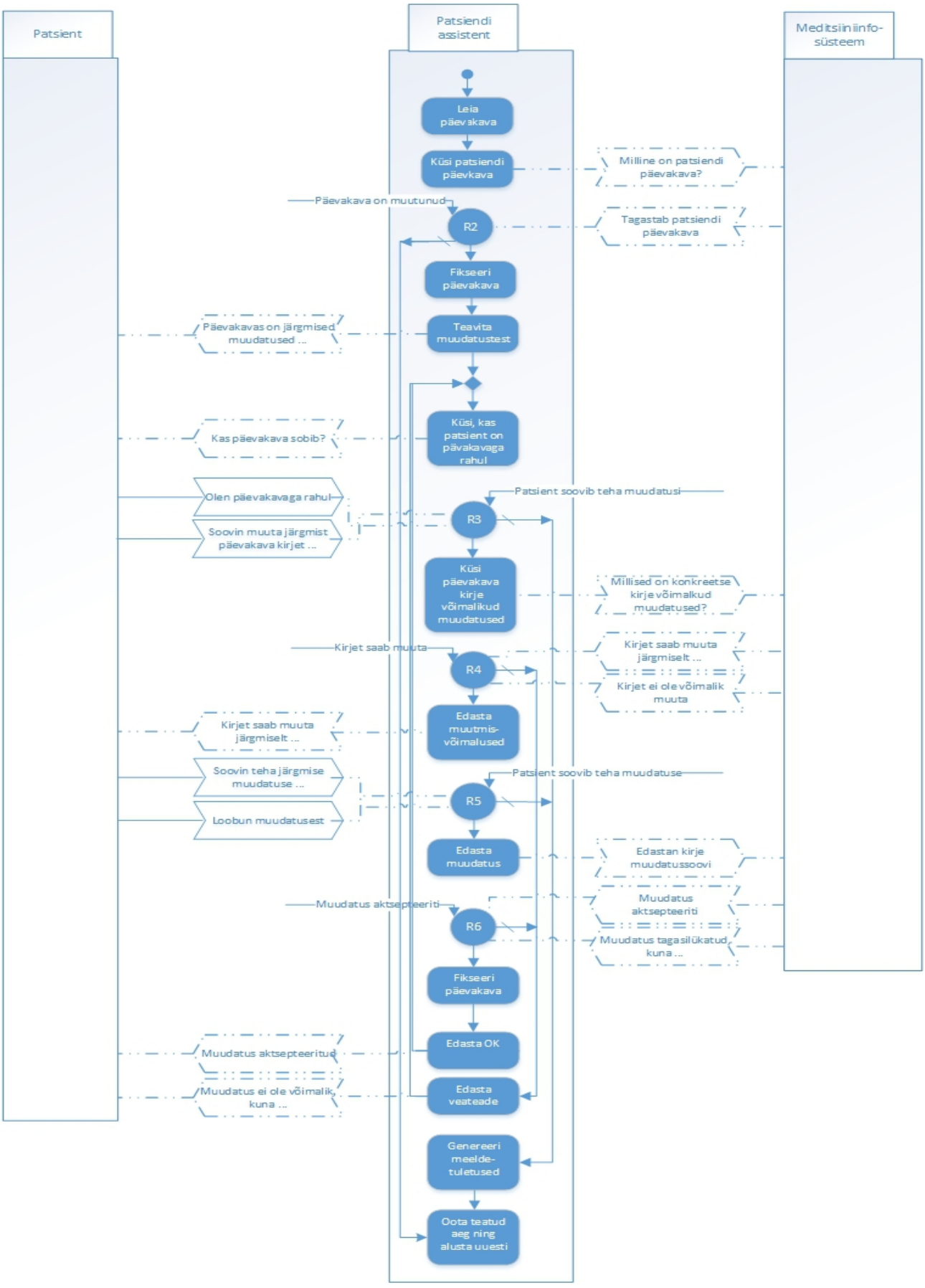
Lisateenuste haldamise käitumismudel

Järgnevalt on esitatud Patsiendi assistendi käitumine patsiendi lisateenuste haldamisel.



Patsiendi päevakava haldamise käitumismudel

Järgnevalt on esitatud Patsiendi assistendi käitumine patsiendi päevakava haldamisel.



Realisatsioon

Järgnevalt on esitatud realisatsiooni käigus produtseeritud Java kood. Agentide realiseerimise ning suhtluse juures on kasutatud JADE vahendit ning andmete edastamisel osaliselt JSON andmestruktuure.

PatientAssistantAgent.java

```
package medsys;

import org.json.*;

import jade.core.AID;
import jade.core.Agent;
import jade.core.behaviours.CyclicBehaviour;
import jade.core.behaviours.OneShotBehaviour;
import jade.domain.DFService;
import jade.domain.FIPAException;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;

public class PatientAssistantAgent extends Agent {

	private PatientAssistantGui gui;
	private AID medicalInformationSystemAgent;
	private AID servicesSystemAgent;

	@Override
	protected void setup() {

		CommonUtil.print("Patient Assistant agent started: " + getAID().getName());

		gui = new PatientAssistantGui(this);
		gui.showGui();

		// initialize medical
		medicalInformationSystemAgent = new AID("medical", false);

		// initialize services
		servicesSystemAgent = new AID("services", false);

		// Register the assistant service in the yellow pages
		DFAgentDescription dfd = new DFAgentDescription();
		dfd.setName(getAID());
		ServiceDescription sd = new ServiceDescription();
		sd.setType("patient-assistant");
		sd.setName(this.getLocalName());
		dfd.addServices(sd);
		try {
			DFService.register(this, dfd);
		}
		catch (FIPAException fe) {
			fe.printStackTrace();
		}

		// listen for service responses
		addBehaviour(new ServicesResponseServer());

		// listen for service status change responses
		addBehaviour(new ServiceStatusChangeResponseServer());
	}

	/**
    This is invoked by the GUI when the user wants to manage ordered services
	 */
	public void handleServices() {
		addBehaviour(new OneShotBehaviour() {
			public void action() {
				CommonUtil.print("Assistant sending services request: " +
getAID().getLocalName());

				// send profile request message
				ACLMessage newMsg = new ACLMessage(MedsysMessageType.SERVICES_REQUEST);
				newMsg.setConversationId(myAgent.getLocalName());
				newMsg.addReceiver(servicesSystemAgent);
				newMsg.setContent(myAgent.getLocalName());
				myAgent.send(newMsg);
			}
		} );
	}

	/**
    This is invoked by the GUI when the user wants to manage ordered services
```

```
	 */
	public void changePatientServiceStatus(final Integer serviceId, final String newStatus) {
		addBehaviour(new OneShotBehaviour() {
			public void action() {
				CommonUtil.print("Assistant changing service status ("+ getAID().getLocalName()
+ ")... " + serviceId + ": " + newStatus);

				// send profile request message
				ACLMessage newMsg = new
ACLMessage(MedsysMessageType.PATIENT_SERVICE_CHANGE_REQUEST);
				newMsg.setConversationId(myAgent.getLocalName());
				newMsg.addReceiver(servicesSystemAgent);
				newMsg.setContent(serviceId + ":" + newStatus);
				myAgent.send(newMsg);
			}
		} );
	}

	private class ServicesResponseServer extends CyclicBehaviour {

		@Override
		public void action() {
			MessageTemplate mt =
MessageTemplate.MatchPerformative(MedsysMessageType.SERVICES_RESPONSE);
			ACLMessage msg = myAgent.receive(mt);
			if (msg != null) {
				CommonUtil.print("Recieved services: " + msg.getContent());
				try {
					JSONArray arrService = new JSONArray(msg.getContent());
					gui.showPatientServices(arrService);
				} catch (JSONException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
			}
			else {
				block();
			}
		}
	}

	private class ServiceStatusChangeResponseServer extends CyclicBehaviour {

		@Override
		public void action() {
			MessageTemplate mt =
MessageTemplate.MatchPerformative(MedsysMessageType.PATIENT_SERVICE_CHANGE_RESPONSE);
			ACLMessage msg = myAgent.receive(mt);
			if (msg != null) {
				CommonUtil.print("Recieved service status change response: " +
msg.getContent());
				if(msg.getContent().startsWith("ok")) {
					gui.showInfoMessage("Service status changed");
				}
				else {
					gui.showErrorMessage("There was an error: " + msg.getContent());
				}
			}
			else {
				block();
			}
		}
	}

	@Override
	protected void takeDown() {
		// Close the GUI
		gui.dispose();
		// show message
		CommonUtil.print("Patient Assistant agent killed: " + getAID().getName());
	}
}
```

MedicalInformationSystemAgent.java

```
package medsys;

import jade.core.AID;
import jade.core.Agent;
import jade.core.behaviours.CyclicBehaviour;
import jade.core.behaviours.TickerBehaviour;
import jade.domain.DFService;
import jade.domain.FIPAException;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import java.util.HashMap;
```

```
import java.util.Map;

public class MedicalInformationSystemAgent extends Agent {

	private MedicalInformationSystemGui gui;

	// List of registered patient assistants
	Map<String, PatientAssistant> patientAssistants = new HashMap<String, PatientAssistant>();

	@Override
	protected void setup() {
		CommonUtil.print("Medical IS agent started: " + getAID().getName());

		gui = new MedicalInformationSystemGui(this);
		gui.showGui();

		// Add a TickerBehaviour that schedules a request every 3 seconds
		addBehaviour(new TickerBehaviour(this, 3000) {
			protected void onTick() {
				// Update the list of seller agents
				DFAgentDescription template = new DFAgentDescription();
				ServiceDescription sd = new ServiceDescription();
				sd.setType("patient-assistant");
				template.addServices(sd);
				try {
					DFAgentDescription[] result = DFService.search(myAgent, template);
					for (int i = 0; i < result.length; ++i) {
						AID agent = result[i].getName();
						PatientAssistant assistant = new PatientAssistant(agent);
						if(false ==
patientAssistants.containsKey(agent.getLocalName())) {
							patientAssistants.put(agent.getLocalName(),
assistant);
							CommonUtil.print("Medical IS found new patient
assistant agent: " + agent.getName());
						}
					}
				}
				catch (FIPAException fe) {
					fe.printStackTrace();
				}

				// update GUI agents list
				updateGuiAgentInfo();
			}
		} );

		// add profile change listener
		addBehaviour(new PatientProfileChangeServer());

		// add patient profile request listener
		addBehaviour(new PatientProfileRequestsServer());
	}

	private void updateGuiAgentInfo() {
		StringBuffer info = new StringBuffer();
		for(String agentName : patientAssistants.keySet()) {
			PatientAssistant agent = patientAssistants.get(agentName);
			info.append(agentName + ", profile: " + agent.getProfile() + "\n");
		}
		gui.setInfoText(info.toString());
	}

	@Override
	protected void takeDown() {
		CommonUtil.print("Medical IS agent killed: " + getAID().getName());
	}

	private class PatientProfileChangeServer extends CyclicBehaviour {
		public void action() {
			MessageTemplate mt =
MessageTemplate.and(MessageTemplate.MatchPerformative(ACLMessage.INFORM),
					MessageTemplate.MatchConversationId("profile-change"));
			ACLMessage msg = myAgent.receive(mt);
			if (msg != null) {
				// CFP Message received. Process it
				String content = msg.getContent();
				ACLMessage reply = msg.createReply();
				String[] arrContent = content.split(":");
				if (arrContent.length == 2
						&& patientAssistants.containsKey(arrContent[0].trim())) {
					// assistant existed

	patientAssistants.get(arrContent[0].trim()).setProfile(arrContent[1].trim());
					reply.setPerformative(ACLMessage.AGREE);
					reply.setContent("ok");
					CommonUtil.print("Patient profile changed ... " + arrContent[0] + ": "
+ arrContent[1]);
				}
				else {
```

```
                                        // there was an error
                                        reply.setPerformative(ACLMessage.REFUSE);
                                        reply.setContent("not-available");
                                }
                                myAgent.send(reply);
                        }
                        else {
                                block();
                        }
                }
        }

        private class PatientProfileRequestsServer extends CyclicBehaviour {
                public void action() {
                        MessageTemplate mt =
MessageTemplate.MatchPerformative(MedsysMessageType.PROFILE_REQUEST);
                        ACLMessage msg = myAgent.receive(mt);
                        if (msg != null) {
                                String assistantName = msg.getContent();

                                CommonUtil.print("MedSys recieved profile request: " + assistantName);

                                ACLMessage reply = msg.createReply();
                                if (patientAssistants.containsKey(assistantName)) {
                                        // assistant existed
                                        reply.setContent(patientAssistants.get(assistantName).getProfile());
                                        reply.setPerformative(MedsysMessageType.PROFILE_RESPONSE);
                                }
                                else {
                                        // there was an error
                                        reply.setPerformative(ACLMessage.REFUSE);
                                        reply.setContent("assistant not available");
                                }
                                CommonUtil.print("MedSys sent profile response: " + reply.getContent());

                                myAgent.send(reply);
                        }
                        else {
                                block();
                        }
                }
        }

        private class PatientAssistant {
                String profile = PatientProfile.REGULAR;
                AID agent;

                public PatientAssistant(AID agent) {
                        this.agent = agent;
                }

                public String getProfile() {
                        return profile;
                }

                public AID getAgent() {
                        return agent;
                }

                public void setProfile(String profile) {
                        this.profile = profile;
                }


        }
}
```

ServicesSystemAgent.java

```
package medsys;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import jade.core.AID;
import jade.core.Agent;
import jade.core.behaviours.CyclicBehaviour;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;

public class ServicesSystemAgent extends Agent {
```

```
	// All services
	Map<Integer, ServiceModel> services = new HashMap<Integer, ServiceModel>();

	// Service qunatities
	Map<Integer, Integer> serviceQuantities = new HashMap<Integer, Integer>();

	// Available services by profile
	Map<String,List<ServiceModel>> profileServices = new HashMap<String, List<ServiceModel>>();

	// Services ordered by patient
	Map<String,List<ServiceModel>> patientServices = new HashMap<String, List<ServiceModel>>();

	// Patient assistants
	Map<String, AID> patientAssistantAgents = new HashMap<String, AID>();

	AID medicalInformationSystemAgent;

	@Override
	protected void setup() {
		CommonUtil.print("ServiceSys agent started: " + getAID().getName());

		// initialize medical
		medicalInformationSystemAgent = new AID("medical", false);

		// Init the list of available services by profile
		Object[] args = getArguments();
		if (args != null && args.length > 0) {
			for(int i = 0; i < args.length; i++) {

				String arg = (String)args[i];
				String[] arrArg = arg.split("-");
				if(arrArg.length == 3) {
					int serviceId = i + 1;
					ServiceModel service = new ServiceModel(serviceId, arrArg[0], new
BigDecimal(arrArg[1]));
					services.put(serviceId, service);
					// TODO quantity
					serviceQuantities.put(serviceId, 1);

					String[] arrProfile = arrArg[2].split("_");
					for(int j = 0; j < arrProfile.length; j++) {
						String profile = arrProfile[j];
						// set profile list, if not set
						if(false == profileServices.containsKey(profile)) {
							profileServices.put(profile, new
ArrayList<ServiceModel>());
						}
						profileServices.get(profile).add(service);
					}
				}
			}
		}
		CommonUtil.print("Available services by profile: " + this.profileServices.toString());

		// listen for service requests
		addBehaviour(new ServicesRequestsServer());

		// listen for profile responses
		addBehaviour(new ProfileResponsesServer());

		// listen for patient service orders
		addBehaviour(new PatientServiceHandlerServer());
	}

	@Override
	protected void takeDown() {
		CommonUtil.print("Services System agent killed: " + getAID().getName());
	}

	private class ServicesRequestsServer extends CyclicBehaviour {

		@Override
		public void action() {
			MessageTemplate mt =
MessageTemplate.MatchPerformative(MedsysMessageType.SERVICES_REQUEST);
			ACLMessage msg = myAgent.receive(mt);
			if (msg != null) {
				CommonUtil.print("ServiceSys recieved services request, sending profile
request");

				// set assistant agent to map, so we know who to reply to
				if(false == patientAssistantAgents.containsKey(msg.getConversationId())) {
					patientAssistantAgents.put(msg.getConversationId(), msg.getSender());
				}
				// send profile request message
				ACLMessage newMsg = new ACLMessage(MedsysMessageType.PROFILE_REQUEST);
				newMsg.setConversationId(msg.getConversationId());
				newMsg.addReceiver(medicalInformationSystemAgent);
				newMsg.setContent(msg.getContent());
```

```
                                myAgent.send(newMsg);
                        }
                        else {
                                block();
                        }
                }
        }

        private class ProfileResponsesServer extends CyclicBehaviour {

                @Override
                public void action() {
                        MessageTemplate mt =
MessageTemplate.MatchPerformative(MedsysMessageType.PROFILE_RESPONSE);
                        ACLMessage msg = myAgent.receive(mt);
                        if (msg != null) {
                                String patientId = msg.getConversationId();
                                String profile = msg.getContent();
                                CommonUtil.print("ServiceSys recieved profile response: " + profile);
                                // get profile response
                                ACLMessage newMsg = new ACLMessage(MedsysMessageType.SERVICES_RESPONSE);
                                newMsg.setConversationId(patientId);
                                newMsg.addReceiver(patientAssistantAgents.get(patientId));
                                JSONArray arrService = new JSONArray();
                                if(profileServices.containsKey(profile)) {

                                        for(ServiceModel service : profileServices.get(profile)) {
                                                // search patient services to see, if current service is
ordered
                                                String status = isServiceAvailable(service.getId()) ?
ServiceStatus.AVAILABLE : ServiceStatus.NOT_AVAILABLE;
                                                if(patientServices.containsKey(patientId)) {
                                                        for(ServiceModel patientService :
patientServices.get(patientId)) {

        if(patientService.getId().equals(service.getId())) {
                                                                        status = ServiceStatus.ORDERED;
                                                                        break;
                                                                }
                                                        }
                                                }

                                                JSONObject objService = new JSONObject();
                                                try {
                                                        objService.put("id", service.getId());
                                                        objService.put("name", service.getName());
                                                        objService.put("price", service.getPrice());
                                                        objService.put("status", status);
                                                } catch (JSONException e) {
                                                        // TODO Auto-generated catch block
                                                        e.printStackTrace();
                                                }
                                                arrService.put(objService);
                                        }
                                }
                                newMsg.setContent(arrService.toString());
                                myAgent.send(newMsg);
                        }
                        else {
                                block();
                        }
                }
        }

        private class PatientServiceHandlerServer extends CyclicBehaviour {

                @Override
                public void action() {
                        MessageTemplate mt =
MessageTemplate.MatchPerformative(MedsysMessageType.PATIENT_SERVICE_CHANGE_REQUEST);
                        ACLMessage msg = myAgent.receive(mt);
                        if (msg != null) {
                                String patientId = msg.getConversationId();
                                ACLMessage newMsg = msg.createReply();
                                newMsg.setPerformative(MedsysMessageType.PATIENT_SERVICE_CHANGE_RESPONSE);
                                String[] arrContent = msg.getContent().split(":");
                                Integer serviceId = Integer.valueOf(arrContent[0]);
                                String status = arrContent[1];
                                if(status.equals(ServiceStatus.CANCELLED)) {
                                        serviceQuantities.put(serviceId, serviceQuantities.get(serviceId) + 1);
                                        for(ServiceModel patService : patientServices.get(patientId)) {
                                                if(patService.getId().equals(serviceId)) {
                                                        patientServices.get(patientId).remove(patService);
                                                        break;
                                                }
                                        }
                                        newMsg.setContent("ok");
                                }
                                else if(status.equals(ServiceStatus.ORDERED)) {
                                        if(isServiceAvailable(serviceId)) {
```

```
                                                serviceQuantities.put(serviceId,
serviceQuantities.get(serviceId) - 1);
                                                if(false == patientServices.containsKey(patientId)) {
                                                        patientServices.put(patientId, new
ArrayList<ServiceModel>());
                                                }
                                                patientServices.get(patientId).add(services.get(serviceId));
                                                newMsg.setContent("ok");
                                        }
                                        else {
                                                newMsg.setContent("Selected item not availabble!");
                                        }
                                }
                                myAgent.send(newMsg);
                        }
                        else {
                                block();
                        }
                }
        }

        private boolean isServiceAvailable(Integer serviceId) {
                return serviceQuantities.get(serviceId).compareTo(0) > 0;
        }
}
```

MedsysMessageType.java

```
package medsys;

/**
 * Possible message types
 */
public final class MedsysMessageType {
        public static final int SERVICES_REQUEST = 1;
        public static final int SERVICES_RESPONSE = 2;
        public static final int PROFILE_REQUEST = 3;
        public static final int PROFILE_RESPONSE = 4;
        public static final int PROFILE_CHANGE = 5;
        public static final int PATIENT_SERVICE_CHANGE_REQUEST = 6;
        public static final int PATIENT_SERVICE_CHANGE_RESPONSE = 7;
}
```

ServiceModel.java

```
package medsys;

import java.math.BigDecimal;

public class ServiceModel {
        private Integer id;
        private String name;
        private BigDecimal price;

        public ServiceModel (Integer id, String name, BigDecimal price) {
                this.id = id;
                this.name = name;
                this.price = price;
        }

        public String getName() {
                return name;
        }

        public BigDecimal getPrice() {
                return price;
        }


        public Integer getId() {
                return id;
        }

        @Override
        public String toString() {
                return "{" + name + ": " + price.toString() + "}";
        }
}
```

ServiceStatus.java

```
package medsys;

/**
 * Possible statuses of a service
 */
```

```
public abstract class ServiceStatus {
        public static final String AVAILABLE = "available";
        public static final String ORDERED = "ordered";
        public static final String NOT_AVAILABLE = "not_available";
        public static final String CANCELLED = "cancelled";
}
```

CommonUtil.java

```
package medsys;

public class CommonUtil {
        public static void print(String msg) {
                System.out.println(msg);
        }
}
```

PatientAssistantGui.java

```
package medsys;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

public class PatientAssistantGui extends JFrame {
        PatientAssistantAgent agent;
        JTextArea infoArea;

        JPanel mainPanel;
        JPanel servicesPanel;

        PatientAssistantGui(PatientAssistantAgent a) {
                super(a.getLocalName());
                agent = a;

                mainPanel = new JPanel(new GridLayout(3, 1));
                getContentPane().add(mainPanel);

                JLabel label = new JLabel("Patient assistant: " + a.getLocalName());
                label.setFont(new Font("Arial", Font.CENTER_BASELINE, 20));
                mainPanel.add(label, BorderLayout.CENTER);

                JButton buttonServices = new JButton("Halda lisateenuseid");
                buttonServices.addActionListener(new ActionListener() {

                        @Override
                        public void actionPerformed(ActionEvent e) {
                                agent.handleServices();
                        }
                });

                mainPanel.add(buttonServices);


                // Make the agent terminate when the user closes
                // the GUI using the button on the upper right corner
                addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {
                                agent.doDelete();
                        }
                });

                Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
                int centerX = (int) screenSize.getWidth() / 2;
                int centerY = (int) screenSize.getHeight() / 2;
                setLocation(centerX - getWidth() / 2, centerY - getHeight() / 2);
        }
```

```
        public void showPatientServices(JSONArray arrService) {
                if(servicesPanel != null) {
                        mainPanel.remove(servicesPanel);
                }

                servicesPanel = new JPanel();

                servicesPanel.setLayout(new GridLayout(arrService.length(), 4));
                for (int i = 0; i < arrService.length(); i++) {

                        try {
                                final JSONObject objJson = arrService.getJSONObject(i);
                                servicesPanel.add(new JLabel(objJson.getString("name")));
                                servicesPanel.add(new
JLabel(Double.valueOf(objJson.getDouble("price")).toString()));
                                String statusName = objJson.getString("status");
                                servicesPanel.add(new JLabel(statusName));
                                if (objJson.getString("status").equals(ServiceStatus.AVAILABLE)) {
                                        JButton button = new JButton("Order");
                                        button.addActionListener(new ActionListener() {
                                                @Override
                                                public void actionPerformed(ActionEvent e) {
                                                        try {

        agent.changePatientServiceStatus(objJson.getInt("id"), ServiceStatus.ORDERED);
                                                        } catch (JSONException e1) {
                                                                // TODO Auto-generated catch block
                                                                e1.printStackTrace();
                                                        }
                                                }
                                        });
                                        servicesPanel.add(button);
                                } else if (objJson.getString("status").equals(
                                                ServiceStatus.ORDERED)) {
                                        JButton button = new JButton("Cancel order");
                                        button.addActionListener(new ActionListener() {
                                                @Override
                                                public void actionPerformed(ActionEvent e) {
                                                        try {

        agent.changePatientServiceStatus(objJson.getInt("id"), ServiceStatus.CANCELLED);
                                                        } catch (JSONException e1) {
                                                                // TODO Auto-generated catch block
                                                                e1.printStackTrace();
                                                        }
                                                }
                                        });
                                        servicesPanel.add(button);
                                } else {
                                        servicesPanel.add(new JLabel(""));
                                }
                        } catch (JSONException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
                mainPanel.add(servicesPanel);
                servicesPanel.setVisible(true);
                showGui();
        }

        public void showInfoMessage(String msg) {
                JOptionPane.showMessageDialog(PatientAssistantGui.this, msg, "Information",
JOptionPane.INFORMATION_MESSAGE);
        }

        public void showErrorMessage(String msg) {
                JOptionPane.showMessageDialog(PatientAssistantGui.this, msg, "Information",
JOptionPane.ERROR_MESSAGE);
        }

        public void showGui() {
                pack();
                super.setVisible(true);
                super.setResizable(true);

        }
}
```

MedicalInformationSystemGui.java

```
package medsys;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.event.WindowAdapter;
```

```
import java.awt.event.WindowEvent;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

public class MedicalInformationSystemGui extends JFrame {
	MedicalInformationSystemAgent agent;
	JTextArea infoArea;

	MedicalInformationSystemGui(MedicalInformationSystemAgent a) {
		super(a.getLocalName());
		agent = a;

		JPanel p = new JPanel();
		p.setLayout(new GridLayout(2, 2));

		// information area
		infoArea = new JTextArea();
		infoArea.setColumns(20);
		infoArea.setRows(5);
		infoArea.setEditable(false);

		JScrollPane scrollingArea = new JScrollPane(infoArea);
		getContentPane().add(scrollingArea, BorderLayout.CENTER);

		// Make the agent terminate when the user closes
		// the GUI using the button on the upper right corner
		addWindowListener(new    WindowAdapter() {
			public void windowClosing(WindowEvent e) {
				agent.doDelete();
			}
		} );

		setResizable(false);
	}

	public void showGui() {
		pack();
		Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
		int centerX = (int)screenSize.getWidth() / 2;
		int centerY = (int)screenSize.getHeight() / 2;
		setLocation(centerX - getWidth() / 2, centerY - getHeight() / 2);
		super.setVisible(true);
	}

	public void setInfoText(String txt) {
		infoArea.setText(txt);
	}
}
```