Workshop 7 in AOM & MAS

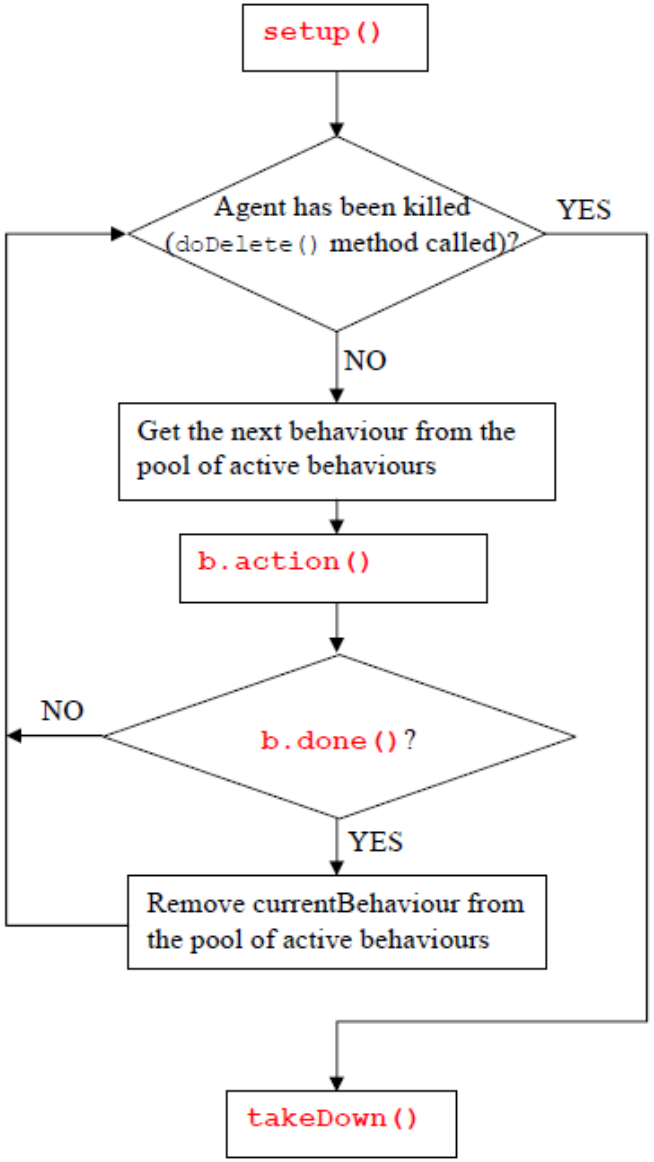
Msury Mahunnah,

Tallinn University of Technology

Concurrent tasks of a JADE agent

- An agent must be able to carry out several concurrent tasks in response to different external events
- Every JADE agent is composed of a single execution thread
- Concurrent tasks are modelled and can be implemented as instances of jade.core.behaviours.Behaviour

Agent thread in JADE



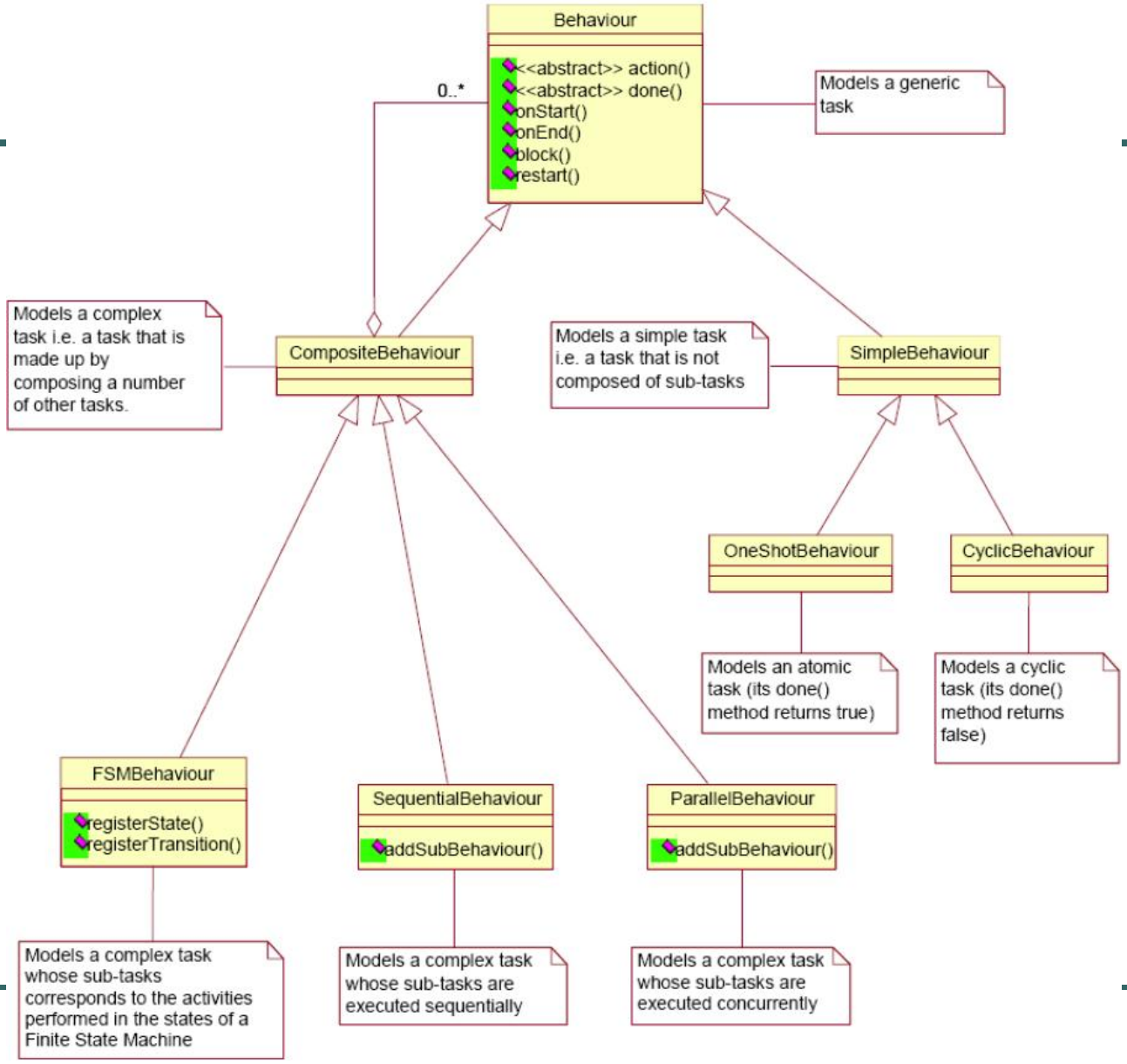
- Initializations
- Addition of initial behaviours

Highlighted in red the methods that programmers have to implement

- Agent "life" (execution of behaviours)

- Clean-up operations

Hierarchy of behaviours



Defining JADE agent

```
import jade.core.*;

public class PhysicianAgent extends Agent {

  //Put agent initialization here
  protected void setup() {
          // Adding behavours
          addBehaviour(new MessageHandler(this));
     ...
   }

  //If needed, put agent clean-up operations here
 protected void takeDown(){
          System.out.println("Physician-agent "+getAID().getName()+" terminated");
          ...
   }
}
```

OneShotBehaviour

```
public class MyOneShotBehaviour extends OneShotBehaviour {
  public void action() {
    // perform operation X
  }
}
```

Operation X is performed only once.

CyclicBehaviour

```
public class MyCyclicBehaviour extends CyclicBehaviour {
  public void action() {
    // perform operation Y
  }
}
```

Operation Y is performed repetitively forever

Generic behaviour

```
public class MyThreeStepBehaviour extends Behaviour {
  private int step = 0;
  public void action() {
    switch (step) {
    case 0:
      // perform operation X
      step++;
      break;
    case 1:
      // perform operation Y
      step++;
      break;
    case 2:
      // perform operation Z
      step++;
      break;
    }
  }

  public boolean done() {
    return step == 3;
  }
}
```

Operations X, Y and Z are performed one after the other

Scheduling operations in JADE

- Jade provides two ready-made classes for easily implementation of behaviours that execute certain operations at given points in time.
    - WakerBehaviour
    - TickerBehaviour

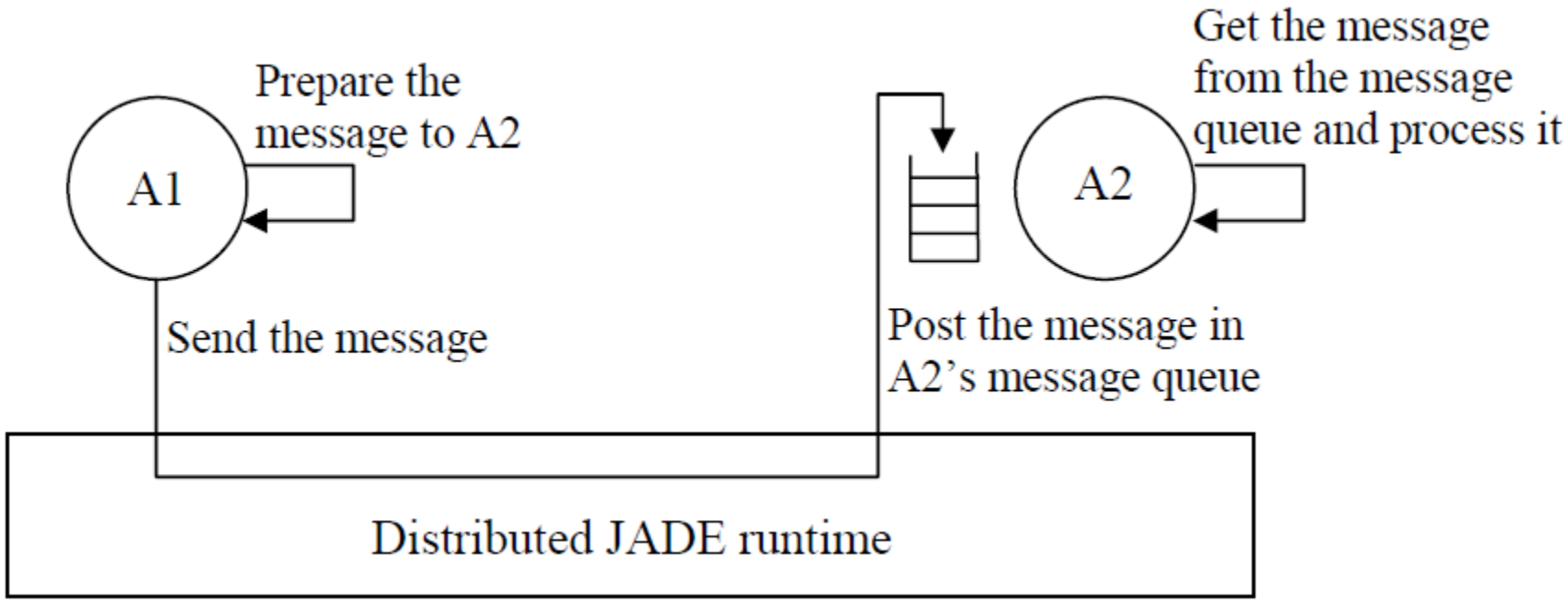
WakerBehaviour

```
public class MyAgent extends Agent {
  protected void setup() {
    System.out.println("Adding waker behaviour");
    addBehaviour(new WakerBehaviour(this, 10000) {
      protected void handleElapsedTimeout() {
        // perform operation X
      }
    } );
  }
}
```

TickerBehaviour

```
public class MyAgent extends Agent {
  protected void setup() {
    addBehaviour(new TickerBehaviour(this, 10000) {
      protected void onTick() {
        // perform operation Y
      }
    } );
  }
}
```

Agent Communication



The ACL Language

- Specifies messages exchanged by JADE agents according to FIPA.
- The format comprises a number of fields, including:
  - *Sender* of the message
  - List of *receivers*
  - Communicative intention (*performative*)
  - *Content*
  - Content *language*
  - *Ontology*

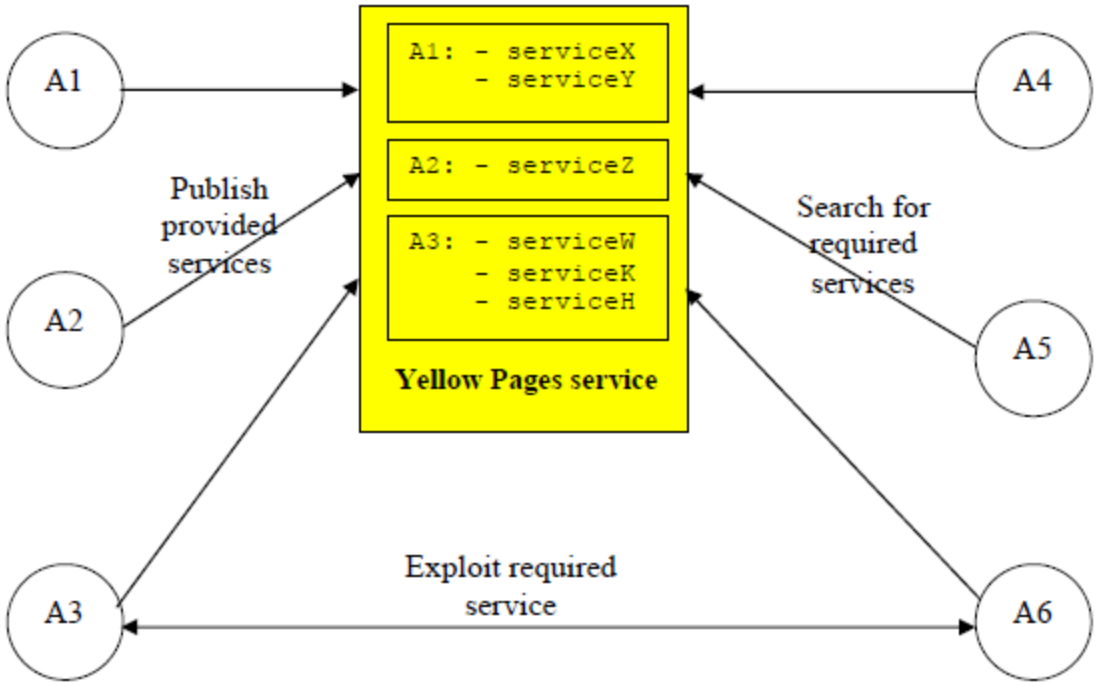
Sending messages

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID("Physician", AID.ISLOCALNAME));
msg.setLanguage("English");
msg.setOntology("Health-delivery-ontology");
msg.setContent("My Glucose level is 130mg/dl");
send(msg);
```

Receiving messages with blocking method

```
public void action() {
  ACLMessage msg = myAgent.receive();
  if (msg != null) {
    // Message received. Process it
    ...
  }
  else {
    block();
  }
}
```

The DF Agent


A1
A2
A3
A4
A5
A6
A1: - serviceX
- serviceY
A2: - serviceZ
A3: - serviceW
- serviceK
- serviceH
Yellow Pages service
Publish provided services
Search for required services
Exploit required service

Publishing a service

```
protected void setup() {
  // Printout a welcome message
    System.out.println("Hello! Physician-agent "+getAID().getLocalName()+" is ready.");

    Object[] args = getArguments();
    if (args != null && args.length > 0){
        language = (String) args [0];
        System.out.println("Language spoken by "+getAID().getLocalName()+" is: "+language);

        DFAgentDescription dfd = new DFAgentDescription();
        dfd.setName(getAID());
        ServiceDescription sd = new ServiceDescription();
        sd.setType("Measurements-verification");
        sd.setName("JADE-personalised-healthcare");
        dfd.addServices(sd);
        try {
        DFService.register(this, dfd);
        }
        catch (FIPAException fe) {
        fe.printStackTrace();
        }
        addBehaviour(new PatientRequestServer());
    }
    else {
        System.out.println("No language has been assigned");
        doDelete();
    }
    }
```

De-register published services

```
protected void takeDown(){
  // deregister from the yellow pages
    try {
    DFService.deregister(this);
    }
    catch (FIPAException fe) {
    fe.printStackTrace();
    }

    System.out.println("Physician-agent "+getAID().getLocalName()+" terminated");
}
```

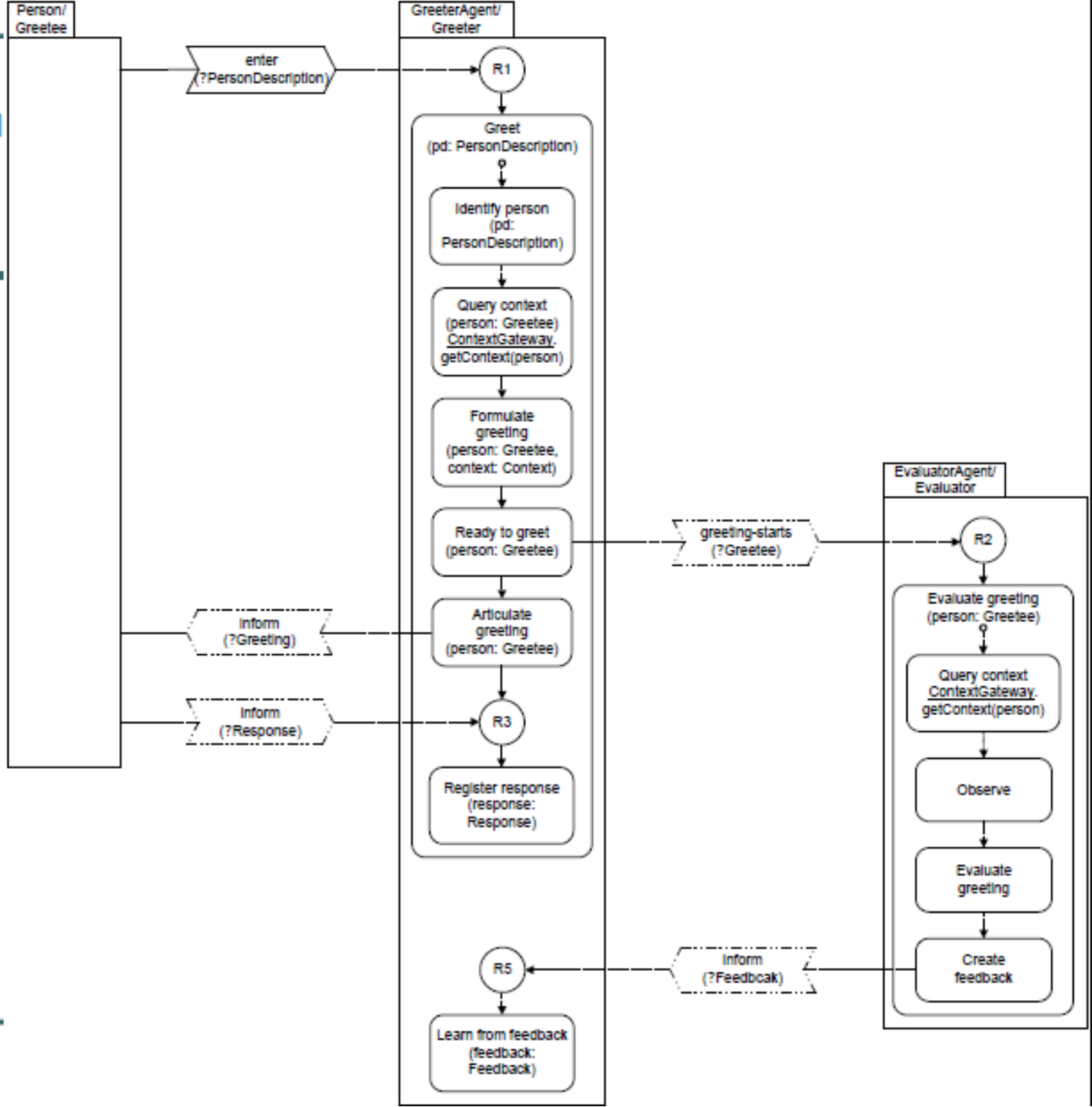
Searching for services

```
//add a WakerBehaviour that sends the measurement to the Physician Agent
  addBehaviour(new WakerBehaviour(this,5000){
  protected void handleElapsedTimeout() {

      // Update the list of physician agents
        DFAgentDescription template = new DFAgentDescription();
        ServiceDescription sd = new ServiceDescription();
        sd.setType("Measurements-verification");
        template.addServices(sd);

        try {
        DFAgentDescription[] result = DFService.search(myAgent, template);
        physicianAgents = new AID[result.length];
        for (int i = 0; i < result.length; ++i) {
        physicianAgents[i] = result[i].getName();
        }
        }
        catch (FIPAException fe) {
        fe.printStackTrace();
        }

        myAgent.addBehaviour(new RequestService());

    }
});
```

Greeting Role Model

- Greetee:
  - *To be greeted by greeter*
  - Responsibilities:
    - To be noticed by greeter; To perceive greeting
  - Constraints: None
- Greeter:
  - *To greet another agent coming within environment*
  - Responsibilities:
    - To notice greetee; To formulate greeting; To articulate greeting
  - Constraints: Articulation within 10 seconds of noticing; Formulation must be appropriate to greetee + environment
- Evaluator:
  - *To evaluate the greeting*
  - Responsibilities:
    - To observe greeting; To evaluate greeting; To publish report
  - Constraints: timeliness

Combined behaviour and interaction model for greeting



Exercises

- Create two JADE agents that greet each other. Follow the Greeting Goal Model and Greeting Role Model.
- Continue with the design for your miniproject either manually or using a suitable tool.