Jason Case Study: Airport Security Robots

Jomi F. Hübner and Rafael H. Bordini

Scenario

- The year is 2070 ad
- London Heathrow uses 3 types of robots:
  - CPH903: cute, polite, handy robots
  - MDS79: multi-device security robots
    - x-ray, metal detectors, and computed tomography for detecting explosive devices
    - ion trap mobility spectrometry (ITMS) for detecting traces of explosives
    - equipment for detecting radioactive materials (gamma ray and neutrons) used in dirty bombs
  - Bomb-disarming robots
- Check-in and security is carried out at the gates
- Normally, there are two replicas of robot model MDS79 at each departure gate

Scenario (Cont.)

- When unattended luggage is reported, all robots in the vicinity are informed of its location through a wireless LAN
- The robots then start a process of negotiation to decide which robots will rellocate to handle the reported unattended luggage

  This is based one proximity, expected delays in the gates (e.g., for lack of MDS79 robots for helping with check in)
- Normally, one MDS79 and one CPH903 robot can cooperate to handle unattended luggage as follows...

Scenario (Cont.)

1. The MDS79 replica moves to the location of the unattended luggage
2. The CPH903 replica moves to that same location
3. MDS79 runs all of the security checks normally available to it:
   a. x-ray, metal detector and computed tomography to scan for explosive devices
   b. the ITMS equipment to check for traces of explosive substances
   c. and finally the radiation detection system to check for dirty bombs

Scenario (Cont.)

4. If nothing suspicious is detected by MDS79, then:
   a. MDS79 informs CPH903 that it is safe to pick up the luggage
   b. CPH903 picks up the luggage and takes it either to its owner or to the lost luggage centre

   Otherwise, MDS79 calls the team of robotic bomb-disarmers

- **This case study is available at *Jason*'s download page, accessbile from `http://jason.sf.net`**

Setting up the MAS

```
MAS heathrow {

    environment:
        HeathrowEnv

    agents:
        mds agentClass mds.MDSAgent
            #5;

        cph agentArchClass cph.CPHAgArch
            agentClass cph.CPHAgent
            #10;

        bd  #3;
}
```

AgentSpeak Code for MDS Robots

```
free.   // I'm free, initially
mds(5). // There are 5 MDS robots (including me)

+unattended_luggage(Terminal,Gate,RN) : true
   <- !negotiate(RN).

+!negotiate(RN) : free
   <- .myName(I);  // Jason internal action
      +bids(RN,1); // number of bids I'm aware of
      mds.calculateMyBid(RN,MyBid); // user int. action
      +winner(RN,I,MyBid);
      .broadcast(tell, bid(RN,MyBid)).

+!negotiate(RN) : not free
   <- .broadcast(tell, bid(RN,0)).
```

AgentSpeak Code for MDS Robots (Cont.)

```
@pb1[atomic]  // for a bid better than mine
+bid(RN,B)[source(Sender)] : winner(RN,I,MyBid)
                           & .myName(I) & MyBid < B
   <- -winner(RN,I,MyBid);
      +winner(RN,Sender,B);
      .print("just lost to another MDS").

// for other bids when I'm still the winner
// and negotiation hasn't finished yet
@pb2[atomic]
+bid(RN,B) : .myName(I) & winner(RN,I,MyBid)
           & bids(RN,N) & mds(M) & N < M
   <- !incBidCounter(RN);
      !check_negot_finished(RN).
```

AgentSpeak Code for MDS Robots (Cont.)

```
@pb3 // just to remember who won anyway
+bid(RN,B)[source(Sender)] : winner(RN,W,WB) & B > WB
   <- -winner(RN,W,WB);
      +winner(RN,Sender,B).

@pb4 // ignore loosing bids, as I'm not the winner
+bid(RN,B) : true <- true.

+!check_negot_finished(RN) : .myName(I)
   & winner(RN,I,MyBid) & bids(RN,N) & mds(M) & N >= M
      <- -free; // I'm in charge of this RN
         !check_luggage(RN);
         !finish_case(RN).
+!check_negot_finished(RN) : true <- true.
```

AgentSpeak Code for MDS Robots (Cont.)

```
+!incBidCounter(RN) : true
   <- ?bids(RN,N);
      .plus(N,1,M);
      -bids(RN,N);
      +bids(RN,M).

+!check_luggage(RN) : true // mybid was the highest
   <- ?unattended_luggage(T,G,RN);
      !go(T,G);              // not included here
      !do_all_checks(RN).  // not included here
```

AgentSpeak Code for MDS Robots (Cont.)

```
// tell bomb-disarmer bd1 about the bomb
+!finish_case(RN) : bomb(RN,Type)
   <- .send(bd1, tell, bomb(RN,Type)).

// it wasn't a bomb afterall, just tidy up
+!finish_case(RN) : not bomb(RN,Type)
   <- +free;
      -bids(RN,X).

// fake plans (for the time being)
+!go(T,G) : true <- true.
+!do_all_checks(RN) : true <- +bomb(RN,bioBomb).
```

AgentSpeak Code for CPH Robots

```
// carry a Bomb to a safe place
+!carryToSafePlace(RN,Place) : true
   <- ?unattended_luggage(Trmnl,Gate,RN);
      !go(Trmnl,Gate);
      pick_up(Bomb);
      !go(Place);
      drop(Bomb).

// fake plans (for the time being)
+!go(T,G) : true <- true.
+!go(P)   : true <- true.
```

AgentSpeak Code for BD Robots

```
skill(pasticBomb).
skill(bioBomb).
~skill(nuclearBomb).

safe_area(field1).

// received a message about a new bomb.
+bomb(RN, BombType) : skill(BombType)
   <- ?unattended_luggage(Terminal,Gate,RN);
      !go(Terminal, Gate);
      !check_if_safe_to_disarm(RN,BombType);
      !attempt_disarm(RN,BombType).

+bomb(RN, BombType) : ~skill(BombType)
   <- .broadcast(tell,security_alert).
```

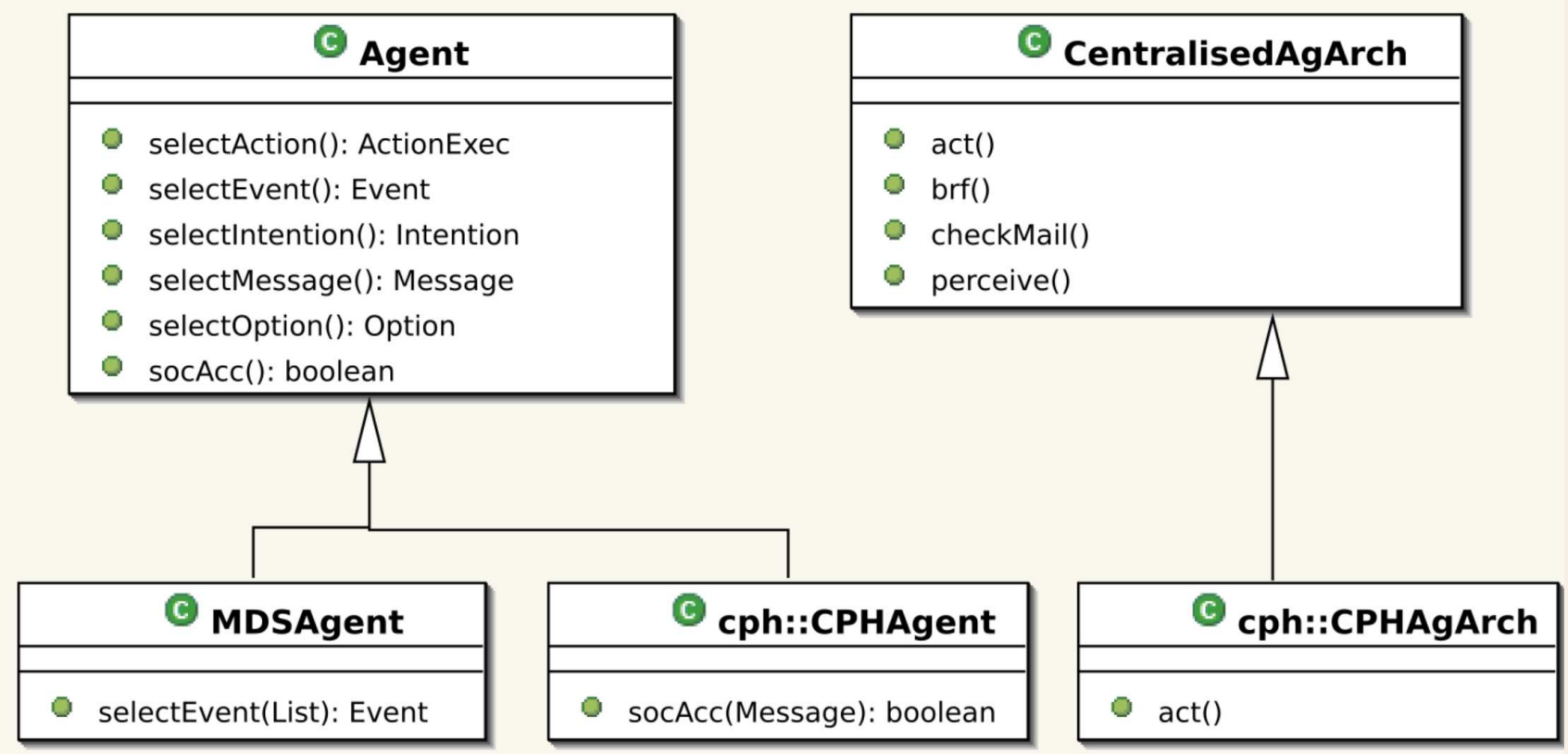
AgentSpeak Code for BD Robots (Cont.)

```
+bomb(RN, BombType) : not skill(BombType)
                    & not ~skill(BombType)
   <- .send(security_manager, tell,
            unkown_bomb_type(RN,BombType)).

+!attempt_disarm(RN,BombType) : safe_to_disarm(RN)
   <- !disarm(BombType).

+!attempt_disarm(RN,BombType) : not safe_to_disarm(RN)
   <- !move_to_safe_area(RN).
```

AgentSpeak Code for BD Robots (Cont.)

```
+!move_to_safe_area(RN) : true
   <- ?safe_area(Place);
      !discoverFreeCPH(FreeCPH);
      .send(FreeCPH, achieve,
            carryToSafePlace(RN,Place)).

// temporary plans, not implemented
+!go(T,G) : true <- true.
+!go(P)   : true <- true.
+!discoverFreeCPH(cph1) : true <- true.
+!check_if_safe_to_disarm(RN,BombType) : true
   <- true. // +safe_to_disarm(RN).
+!disarm(plasticBomb) : true <- true.
+!disarm(bioBomb) : true <- true.
+!disarm(dirtyBomb) : true <- true.
```

Agent and Architecture Class Customisation



MDS – calculateMyBid.java

```
package mds;

import jason.asSemantics.*;
import jason.asSyntax.*;

public class calculateMyBid {

    public static boolean execute(TransitionSystem ts,
                Unifier un, String[] args) throws Exception {
        String id = ts.getAgArch().getName().substring(3);
        int bid = Integer.parseInt(id) * 10;
        // args[0] is the unattended luggage Report Number
        un.unifies(Term.parse(args[1]), Term.parse(""+bid));
        return true;
    }
}
```

MDS – Customised Agent Class (mds/MDSAgent.java)

```
package mds;

import jason.asSemantics.*;
import jason.asSyntax.*;
import java.util.*;

/** example of agent function overriding */
public class MDSAgent extends Agent {

   /** unattendedLuggage event has priority */
   public Event selectEvent(List evList) {
      Iterator i = evList.iterator();
      while (i.hasNext()) {
         Event e = (Event)i.next();
         if (e.getTrigger().getFunctor().equals(
                              "unattended_luggage")) {
            i.remove();
            return e;
         }
```

MDS – Customised Agent Class (Cont.)

```
        // the unattended Luggage event could generate a
        // sub-goal that generates other events
        if (e.getIntention() != null) {
          Iterator j = e.getIntention().iterator();
          while (j.hasNext()) {
            IntendedMeans im = (IntendedMeans)j.next();
            Trigger it = (Trigger)im.getPlan().getTriggerEvent();
            if (it.getFunctor().equals("unattended_luggage")) {
              i.remove();
              return e;
            }
          }
        }
      }
      return super.selectEvent(evList);
    }
  }
```

CPH – Customised Agent Class (cph/CPHAgent.java)

```
package cph;

import jason.asSemantics.Agent;

public class CPHAgent extends Agent {
   /** only accepts "achieve" messages
    *  from mds robots
    */
    public boolean acceptAchieve(String sender,
                                 String content) {
        if (sender.startsWith("mds")) {
            return true;
        } else {
            return false;
        }
    }
}
```

CPH – Customised AgArch Class (cph/CPHAgArch.java)

```
package cph;

import jason.architecture.CentralisedAgArch;
import jason.asSemantics.Message;
import jason.asSyntax.Term;

import java.util.Iterator;
import java.util.List;
```

CPH – Customised AgArch Class (Cont.)

```
public class CPHAgArch extends CentralisedAgArch {

  /** overridden to ignore bid messages */
  public void checkMail() {
  List mbox = (List)fEnv.getAgMbox(getName());
  synchronized (mbox) {
    Iterator i = mbox.iterator();
    while (i.hasNext()) {
      Message im = (Message)i.next();
        if (! im.getPropCont().startsWith("bid")) {
          fTS.getC().getMB().add(im);
          if (fTS.getSettings().verbose()>=1) {
            System.out.println("Agent "+getName()+
                        " received message: " + im);
          }
        }
      i.remove();
    }
  }}
```

CPH – Customised AgArch Class (Cont.)

```
public void act() {
 // get the action to be performed
 if (fTS.getC().getAction() != null) {
 Term action = fTS.getC().getAction().getActionTerm();
  if (! action.getFunctor().equals("disarm")) {
   if (fEnv.getUserEnvironment().executeAction(
                                    getName(), action)) {
    fTS.getC().getAction().setResult(true);
   }
   else {
     fTS.getC().getAction().setResult(false);
   }
   fTS.getC().getFeedbackActions().add(fTS.getC().getAction());
  }
 }
}
}
```