Sistemas Multi-Agentes

Medios Ambientes

Dr. Alejandro Guerra-Hernández

Universidad Veracruzana
Instituto de Investigaciones en Inteligencia Artificial
Campus Sur, Calle Paseo Lote II, Sección Segunda No 112,
Nuevo Xalapa, Xalapa, Ver., México 91097

aguerra@uv.mx
https://www.uv.mx/personal/aguerra/

Maestría en Inteligencia Artificial 2023

Ambientes en Jason

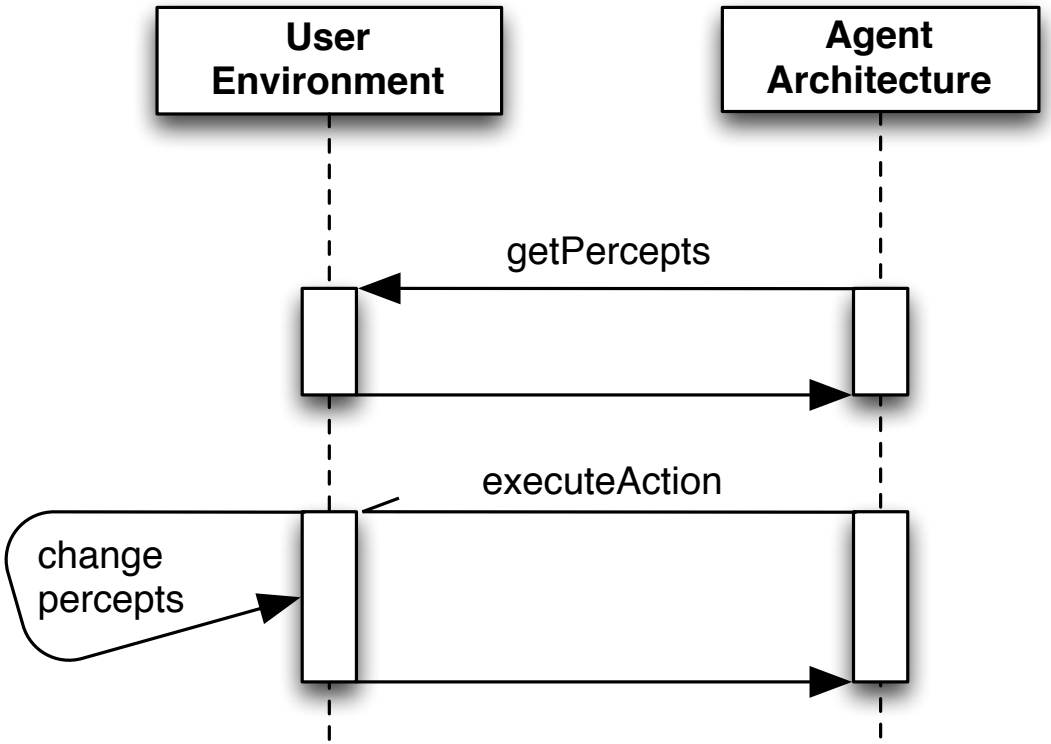
- Jason permite definir explícitamente un ambiente para un SMA.
- Esto no es obligatorio, pero presenta ventajas:
  - Abstracción del medio ambiente real –Java vs AgentSpeak(L).
  - Validación del SMA por simulación.
- Dos aproximaciones:

  Exógena: El ambiente es ajeno al SMA, p. ej., la clase `environment` de Jason (Bordini, Hübner y Wooldridge [2]).

  Endógena: El ambiente es parte integral del SMA, p. ej., CArtAgO (Ricci, Piunti y Viroli [3]).

Universidad Veracruzana

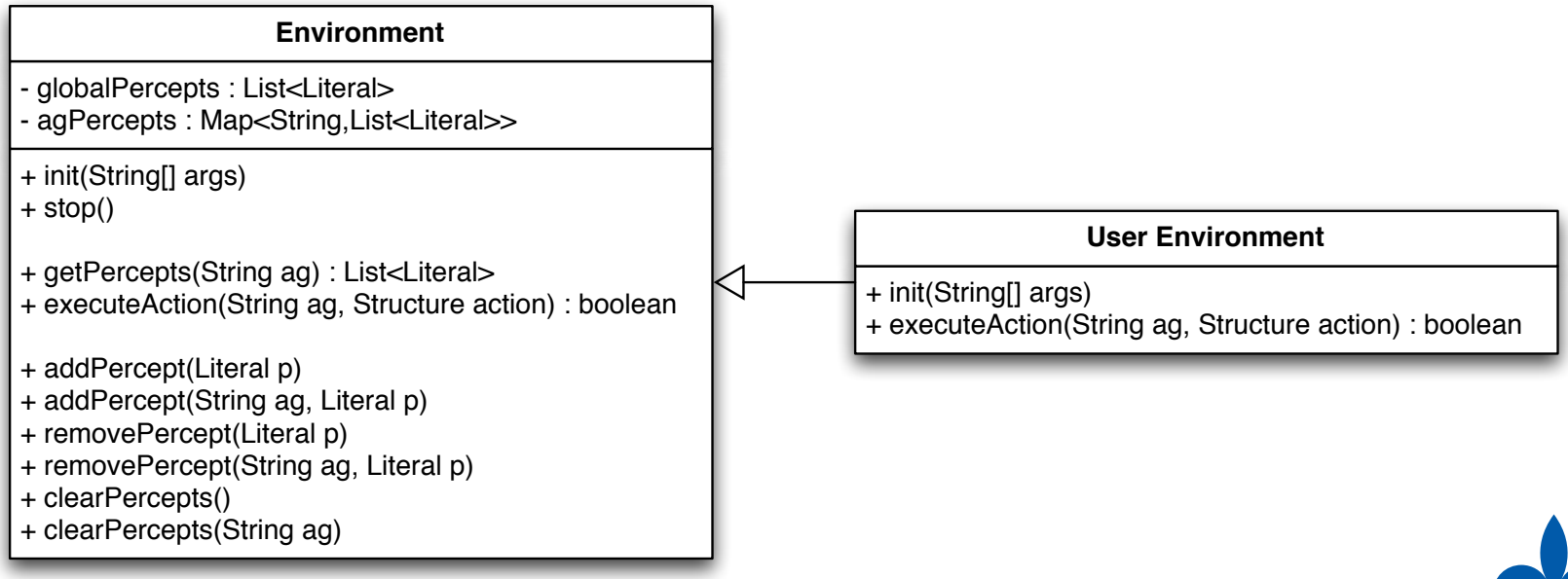
Ambiente Jason en UML



Universidad Veracruzana

La clase Environment

- Un ambiente se define extendiendo la clase `Environment` y sobrecargando los métodos `init` y `executeAction`.



Métodos de la clase `Environment`

Método	Descripción
`addPercept(L)`	Agrega la literal L a la lista de todos los agentes perciben L.
`addPercept(A,L)`	Agrega la literal L a la lista de percepciones del agente A, solo A percibe L.
`removePercept(L)`	Elimina la literal L de la lista de percepciones globales.
`removePercept(A,L)`	Elimina la literal L de la lista de percepciones del agente A.
`clearPercepts()`	Borra todas las percepciones la lista global.
`clearPercepts(A)`	Borra todas las percepciones de la lista del agente A.

Universidad Veracruzana

Estructura general del programa ambiente I

```
1  import jason.asSyntax.*;
2  import jason.environment.*;
3
4  public class <EnvironmentName> extends Environment {
5    // Otros los miembros de la clase
6
7    @override
8    public void init(String[] args) {
9      // Estableciendo la creencia inicial p(a):
10     addPercept(Literal.parseLiteral("p(a)"));
11     // Si no usamos el Supuesto del Mundo Cerrado:
12     addPercept(Literal.parseLiteral("~q(b)"));
13     // Solo el agente "ag1" percibe p(a):
14     addPercept("ag1", Literal.parseLiteral("p(a)"));
15   }
16
17   @override
18   public void stop() {
19     // Lo que se deba hacer cuando el sistema se detenga.
20   }
21
```

Universidad Veracruzana

Estructura general del programa ambiente II

```
22      @override
23      public boolean executeAction(String ag, Term act) {
24        // Acciones
25      }
26  }
```

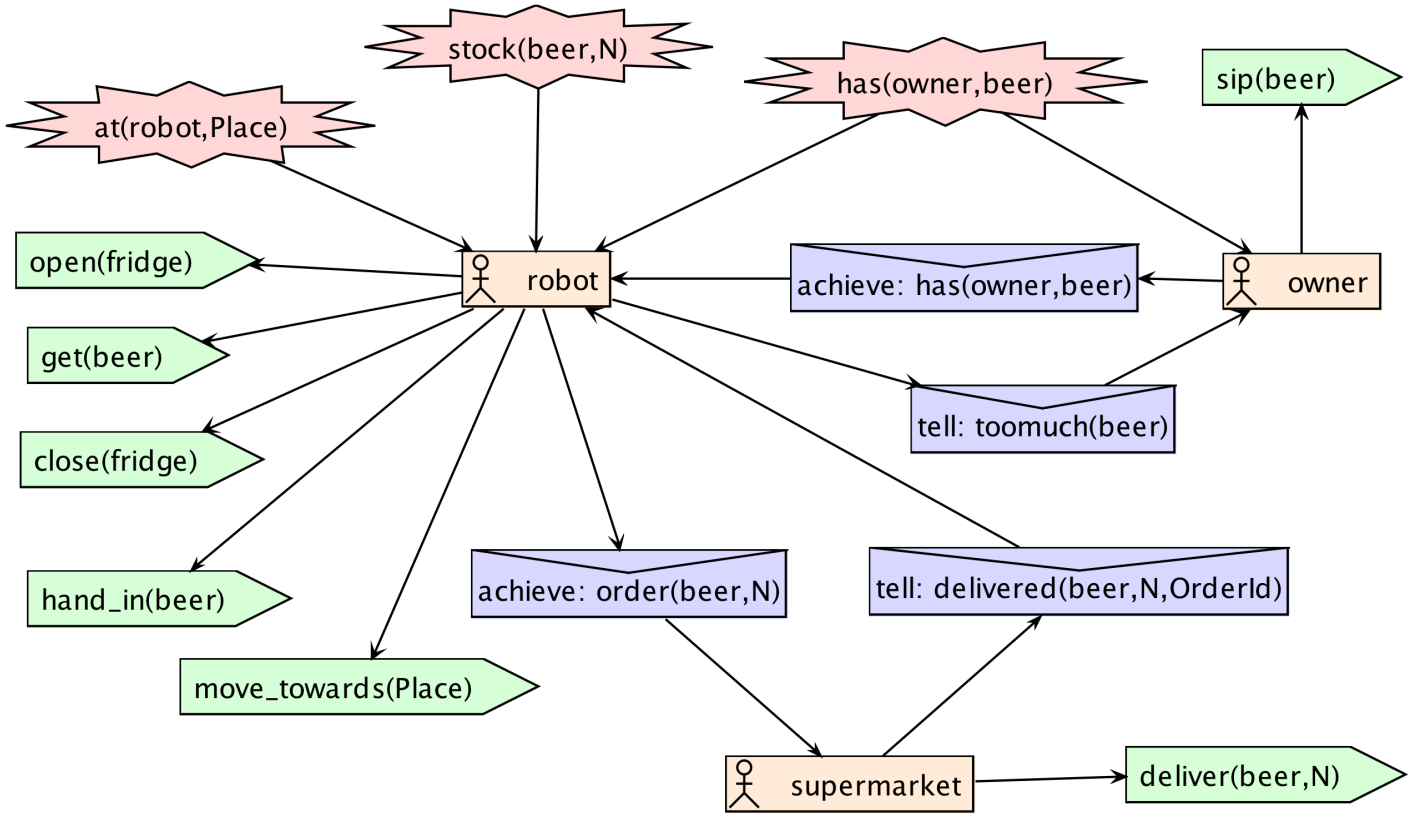
Universidad Veracruzana

Consideraciones

- Unicamente es posible agregar como percepciones objetos de la clase `Literal`.
- Se pueden usar literales negadas fuertemente ($\sim \alpha$).
- `executeAction` suspende su intención asociada.
- Las acciones externas son booleanas, si la acción regresa `false`, su plan asociado falla.
- Si se desea que el agente recuerde las creencias que ya no son percibidas, agregue una nota mental.
- Cuide la consistencia de tipos entre los nombres de las percepciones y acciones Jason/Java.

Universidad Veracruzana

Un robot chelero



Universidad Veracruzana

Diseño Modelo-Vista-Controlador

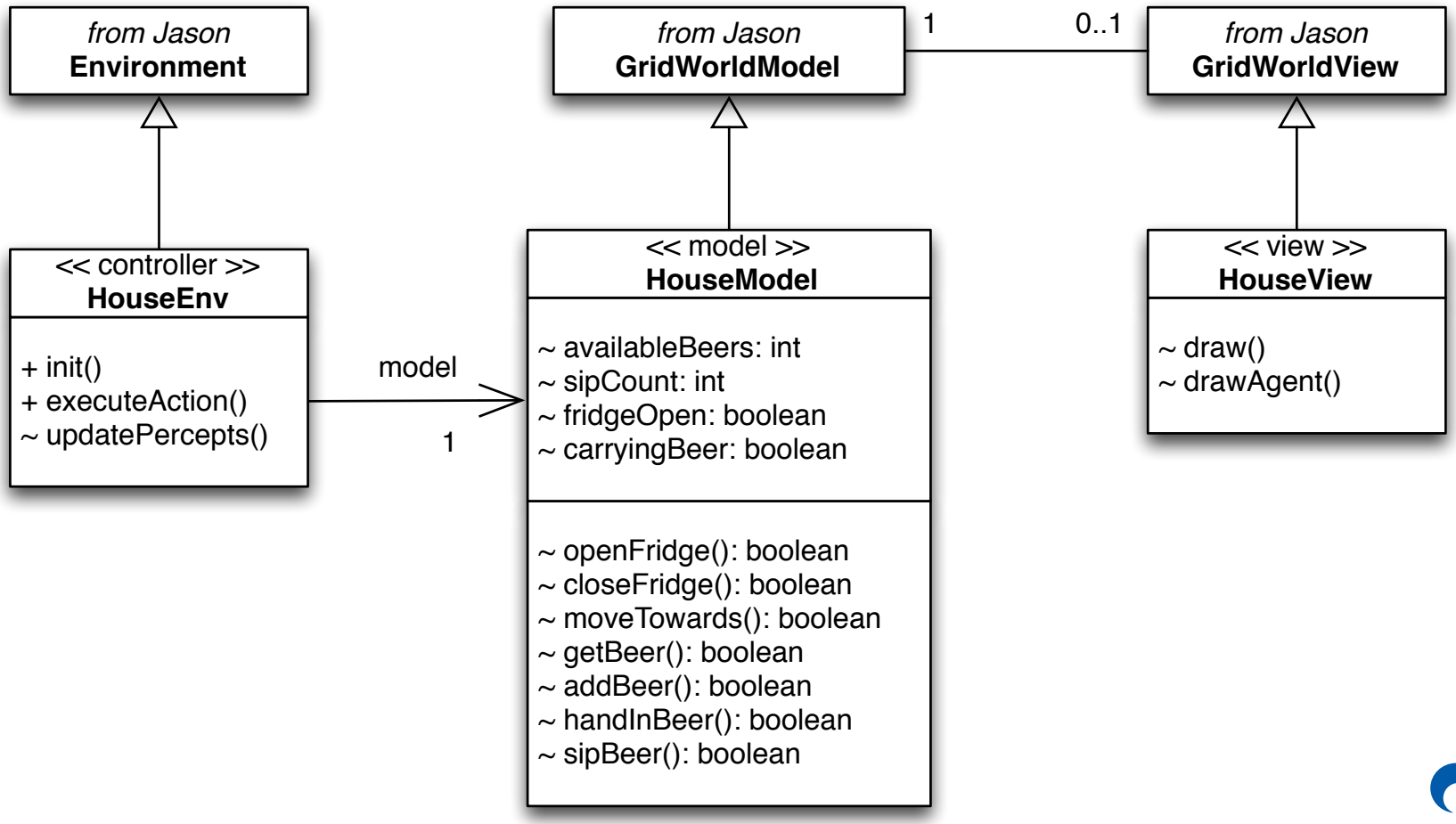
Modelo. Guarda la información acerca del estado del ambiente y su dinámica. Por ej., la posición de un agente robot y su nueva posición cuando éste se mueve.

Vista. Implementa el despliegue adecuado en pantalla del ambiente, usando la información del modelo.

Controlador. Este elemento interactua con los agentes e invoca cambios en el modelo y por ende en la vista. P. ej., al ejecutar la acción `move` la posición del agente cambia.

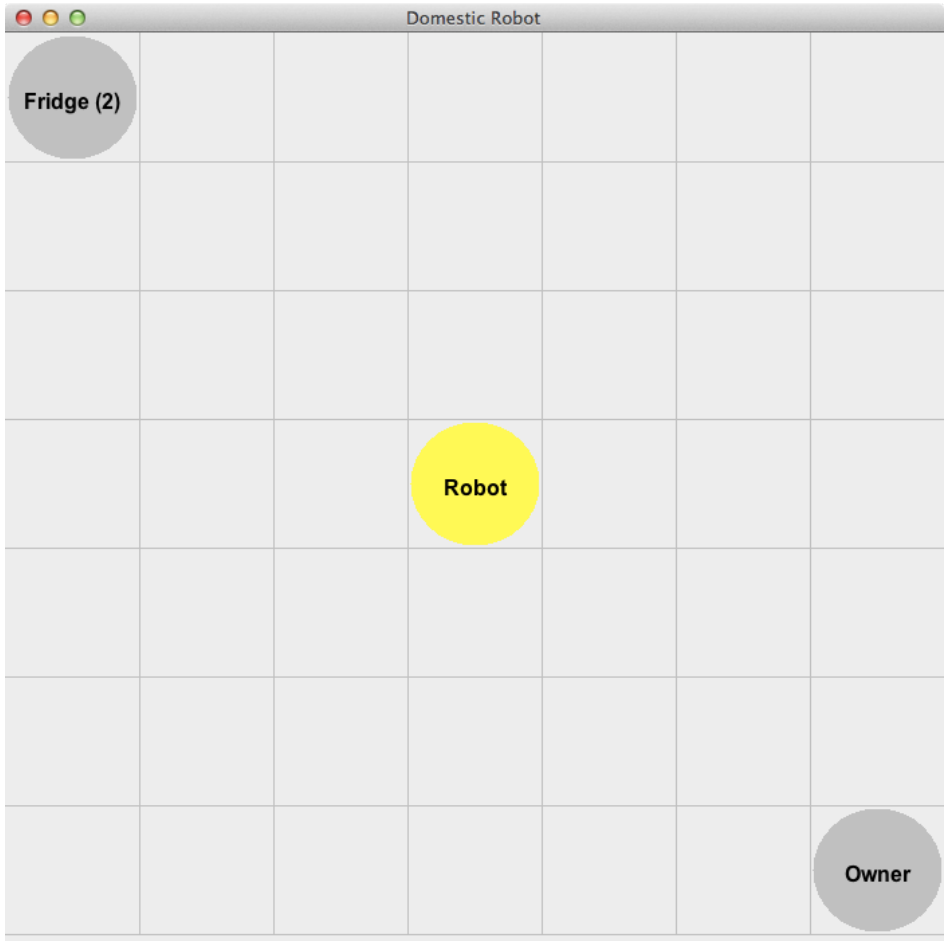
Universidad Veracruzana

Diagrama de clases para el robot chelero



Universidad Veracruzana

La vista del robot chelero (GUI)



Universidad Veracruzana

El SMA del robot chelero

```
1  /* Jason Project
2
3     See Prometheus specification in doc folder
4
5  */
6
7  MAS domestic_robot {
8      infrastructure: Centralised
9      environment: HouseEnv(gui) // use "nogui" as parameter to not show
       ↪  the GUI
10     agents: robot;
11             owner;
12             supermarket;
13     aslSourcePath:
14       "src/asl";
15 }
```

Universidad Veracruzana

El controlador: Defínción de literales I

```
1  import jason.asSyntax.*;
2  import jason.environment.Environment;
3  import jason.environment.grid.Location;
4  import java.util.logging.Logger;
5
6  public class HouseEnv extends Environment {
7
8    // common literals
9    public static final Literal of = Literal.parseLiteral("open(fridge)");
10   public static final Literal clf =
     ↪  Literal.parseLiteral("close(fridge)");
11   public static final Literal gb = Literal.parseLiteral("get(beer)");
12   public static final Literal hb = Literal.parseLiteral("hand_in(beer)");
13   public static final Literal sb = Literal.parseLiteral("sip(beer)");
14   public static final Literal hob =
     ↪  Literal.parseLiteral("has(owner,beer)");
15
16   public static final Literal af =
     ↪  Literal.parseLiteral("at(robot,fridge)");
17   public static final Literal ao =
     ↪  Literal.parseLiteral("at(robot,owner)");
```

Universidad Veracruzana

El controlador: Defición de literales II

```
18
19      static Logger logger = Logger.getLogger(HouseEnv.class.getName());
20
21      HouseModel model; // the model of the grid
```

Universidad Veracruzana

El controlador: Método inicial

```
23  @Override
24  public void init(String[] args) {
25    model = new HouseModel();
26
27    if (args.length == 1 && args[0].equals("gui")) {
28      HouseView view = new HouseView(model);
29      model.setView(view);
30    }
31
32    updatePercepts();
33  }
```

Universidad Veracruzana

El controlador: Actualización de percepciones I

```
36    void updatePercepts() {
37      // clear the percepts of the agents
38      clearPercepts("robot");
39      clearPercepts("owner");
40
41      // get the robot location
42      Location lRobot = model.getAgPos(0);
43
44      // add agent location to its percepts
45      if (lRobot.equals(model.lFridge)) {
46        addPercept("robot", af);
47      }
48      if (lRobot.equals(model.lOwner)) {
49        addPercept("robot", ao);
50      }
51
52      // add beer "status" the percepts
53      if (model.fridgeOpen) {
54        addPercept("robot", Literal.parseLiteral("stock(beer,"
55                                                 + model.availableBeers
56                                                 + ")"));
```

El controlador: Actualización de percepciones II

```
57        }
58        if (model.sipCount > 0) {
59          addPercept("robot", hob);
60          addPercept("owner", hob);
61        }
62      }
```

Universidad Veracruzana

El controlador: Ejecución de las acciones I

```
65  public boolean executeAction(String ag, Structure action) {
66    System.out.println("[" + ag + "] doing: " + action);
67    boolean result = false;
68    if (action.equals(of)) { // of = open(fridge)
69      result = model.openFridge();
70
71    } else if (action.equals(clf)) { // clf = close(fridge)
72      result = model.closeFridge();
73
74    } else if (action.getFunctor().equals("move_towards")) {
75      String l = action.getTerm(0).toString();
76      Location dest = null;
77      if (l.equals("fridge")) {
78        dest = model.lFridge;
79      } else if (l.equals("owner")) {
80        dest = model.lOwner;
81      }
82
83      try {
84        result = model.moveTowards(dest);
85      } catch (Exception e) {
```

Universidad Veracruzana

El controlador: Ejecución de las acciones II

```
86          e.printStackTrace();
87        }
88
89      } else if (action.equals(gb)) {
90        result = model.getBeer();
91
92      } else if (action.equals(hb)) {
93        result = model.handInBeer();
94
95      } else if (action.equals(sb)) {
96        result = model.sipBeer();
97
98      } else if (action.getFunctor().equals("deliver")) {
99        // wait 4 seconds to finish "deliver"
100       try {
101         Thread.sleep(4000);
102         result = model.addBeer((int) ((NumberTerm)
            ↪  action.getTerm(1)).solve());
103       } catch (Exception e) {
104         logger.info("Failed to execute action deliver!" + e);
105       }
```

El controlador: Ejecución de las acciones III

```
106
107     } else {
108       logger.info("Failed to execute action " + action);
109     }
110
111     if (result) {
112       updatePercepts();
113       try {
114         Thread.sleep(100);
115       } catch (Exception e) {
116       }
117     }
118     return result;
119   }
```

Universidad Veracruzana

El modelo: propiedades

```
1  import jason.environment.grid.GridWorldModel;
2  import jason.environment.grid.Location;
3
4  /** class that implements the Model of Domestic Robot application */
5  public class HouseModel extends GridWorldModel {
6
7    // constants for the grid objects (a binary mask 1 for agent,2 for
     ↪  obstacle)
8    public static final int FRIDGE = 16;
9    public static final int OWNER = 32;
10
11   // the grid size
12   public static final int GSize = 7;
13
14   boolean fridgeOpen = false; // whether the fridge is open
15   boolean carryingBeer = false; // whether the robot is carrying beer
16   int sipCount = 0; // how many sip the owner did
17   int availableBeers = 2; // how many beers are available
18
19   Location lFridge = new Location(0, 0);
20   Location lOwner = new Location(GSize - 1, GSize - 1);
```

Universidad Veracruzana

El modelo: acciones I

```
44  boolean closeFridge() {
45    if (fridgeOpen) {
46      fridgeOpen = false;
47      return true;
48    } else {
49      return false;
50    }
51  }
52
53  boolean moveTowards(Location dest) {
54    Location r1 = getAgPos(0);
55    if (r1.x < dest.x)
56      r1.x++;
57    else if (r1.x > dest.x)
58      r1.x--;
59    if (r1.y < dest.y)
60      r1.y++;
61    else if (r1.y > dest.y)
62      r1.y--;
63    setAgPos(0, r1); // move the robot in the grid
64
```

Universidad Veracruzana

El modelo: acciones II

```
65      // repaint the fridge and owner locations
66      if (view != null) {
67        view.update(lFridge.x, lFridge.y);
68        view.update(lOwner.x, lOwner.y);
69      }
70      return true;
71    }
```

Universidad Veracruzana

La vista: y su modelo

```
1  import jason.environment.grid.*;
2
3  import java.awt.Color;
4  import java.awt.Font;
5  import java.awt.Graphics;
6
7  /** class that implements the View of Domestic Robot application */
8  public class HouseView extends GridWorldView {
9
10   private static final long serialVersionUID = 1L;
11
12   HouseModel hmodel;
13
14   public HouseView(HouseModel model) {
15     super(model, "Domestic Robot", 700);
16     hmodel = model;
17     defaultFont = new Font("Arial", Font.BOLD, 16); // change default
       ↪  font
18     setVisible(true);
19     repaint();
20   }
```

Universidad Veracruzana

La vista: dibujando I

```
22    /** draw application objects */
23    @Override
24    public void draw(Graphics g, int x, int y, int object) {
25      Location lRobot = hmodel.getAgPos(0);
26      super.drawAgent(g, x, y, Color.lightGray, -1);
27      repaint();
28      switch (object) {
29      case HouseModel.FRIDGE:
30        if (lRobot.equals(hmodel.lFridge)) {
31          super.drawAgent(g, x, y, Color.yellow, -1);
32        }
33        g.setColor(Color.black);
34        drawString(g, x, y, defaultFont, "Fridge (" + hmodel.availableBeers
          ↪  + ")");
35        break;
36      case HouseModel.OWNER:
37        if (lRobot.equals(hmodel.lOwner)) {
38          super.drawAgent(g, x, y, Color.yellow, -1);
39        }
40        String o = "Owner";
41        if (hmodel.sipCount > 0) {
```

Universidad Veracruzana

La vista: dibujando II

```
42            o += " (" + hmodel.sipCount + ")";
43          }
44          g.setColor(Color.black);
45          drawString(g, x, y, defaultFont, o);
46          break;
47        }
48      }
49
50      @Override
51      public void drawAgent(Graphics g, int x, int y, Color c, int id) {
52        Location lRobot = hmodel.getAgPos(0);
53        if (!lRobot.equals(hmodel.lOwner) && !lRobot.equals(hmodel.lFridge))
          ↪  {
54          c = Color.yellow;
55          if (hmodel.carryingBeer)
56            c = Color.orange;
57          super.drawAgent(g, x, y, c, -1);
58          g.setColor(Color.black);
59          super.drawString(g, x, y, defaultFont, "Robot");
60        }
```

Universidad Veracruzana

La vista: dibujando III

```
61        }
62    }
```

Agentes: el dueño I

```
3  !get(beer).   // initial goal: get a beer
4  !check_bored. // initial goal: verify whether I am getting bored
5
6  +!get(beer) : true
7     <- .send(robot, achieve, has(owner,beer)).
8
9  +has(owner,beer) : true
10    <- !drink(beer).
11 -has(owner,beer) : true
12    <- !get(beer).
13
14 // while I have beer,sip
15 +!drink(beer) : has(owner,beer)
16    <- sip(beer);
17       !drink(beer).
18 +!drink(beer) : not has(owner,beer)
19    <- true.
20
21 +!check_bored : true
22    <- .random(X); .wait(X*5000+2000);  // i get bored at random times
23       .send(robot, askOne, time(_), R); // when bored,I ask the robot
       ↪  about the time
```

Agentes: el dueño II

```
24          .print(R);
25          !!check_bored.
26
27    +msg(M)[source(Ag)] : true
28       <- .print("Message from ",Ag,": ",M);
29          -msg(M).
```

Universidad Veracruzana

Agentes: el supermercado

```
1  last_order_id(1). // initial belief
2
3  // plan to achieve the goal "order" for agent Ag
4  +!order(Product,Qtd)[source(Ag)] : true
5    <- ?last_order_id(N);
6       OrderId = N + 1;
7       -+last_order_id(OrderId);
8       deliver(Product,Qtd);
9       .send(Ag, tell, delivered(Product,Qtd,OrderId)).
10
```

Universidad Veracruzana

Agentes: el robot (creencias)

```
1  /* Initial beliefs and rules */
2
3  // initially,I believe that there is some beer in the fridge
4  available(beer,fridge).
5
6  // my owner should not consume more than 10 beers a day :-)
7  limit(beer,10).
8
9  too_much(B) :-
10     .date(YY,MM,DD) &
11     .count(consumed(YY,MM,DD,_,_,_,B),QtdB) &
12     limit(B,Limit) &
13     QtdB > Limit.
```

Universidad Veracruzana

Agentes: el robot (planes) I

```
18  +!has(owner,beer)
19     :  available(beer,fridge) & not too_much(beer)
20     <- !at(robot,fridge);
21        open(fridge);
22        get(beer);
23        close(fridge);
24        !at(robot,owner);
25        hand_in(beer);
26        ?has(owner,beer);
27        // remember that another beer has been consumed
28        .date(YY,MM,DD); .time(HH,NN,SS);
29        +consumed(YY,MM,DD,HH,NN,SS,beer).
30
31  +!has(owner,beer)
32     :  not available(beer,fridge)
33     <- .send(supermarket, achieve, order(beer,5));
34        !at(robot,fridge). // go to fridge and wait there.
35
36  +!has(owner,beer)
37     :  too_much(beer) & limit(beer,L)
38     <- .concat("The Department of Health does not allow me to give you
       ↪  more than ", L,
```

Agentes: el robot (planes) II

```
39                  " beers a day! I am very sorry about that!",M);
40          .send(owner,tell,msg(M)).
41
42
43  -!has(_,_)
44     :  true
45     <- .current_intention(I);
46        .print("Failed to achieve goal '!has(_,_)'. Current intention is:
          ↪  ",I).
47
48  +!at(robot,P) : at(robot,P) <- true.
49  +!at(robot,P) : not at(robot,P)
50    <- move_towards(P);
51       !at(robot,P).
52
53  // when the supermarket makes a delivery,try the 'has' goal again
54  +delivered(beer,_Qtd,_OrderId)[source(supermarket)]
55    :  true
56    <- +available(beer,fridge);
57       !has(owner,beer).
58
```

Universidad Veracruzana

Agentes: el robot (planes) III

```
59  // when the fridge is opened,the beer stock is perceived
60  // and thus the available belief is updated
61  +stock(beer,0)
62     :  available(beer,fridge)
63     <- -available(beer,fridge).
64  +stock(beer,N)
65     :  N > 0 & not available(beer,fridge)
66     <- -+available(beer,fridge).
67
68  +?time(T) : true
69    <-  time.check(T).
70
```

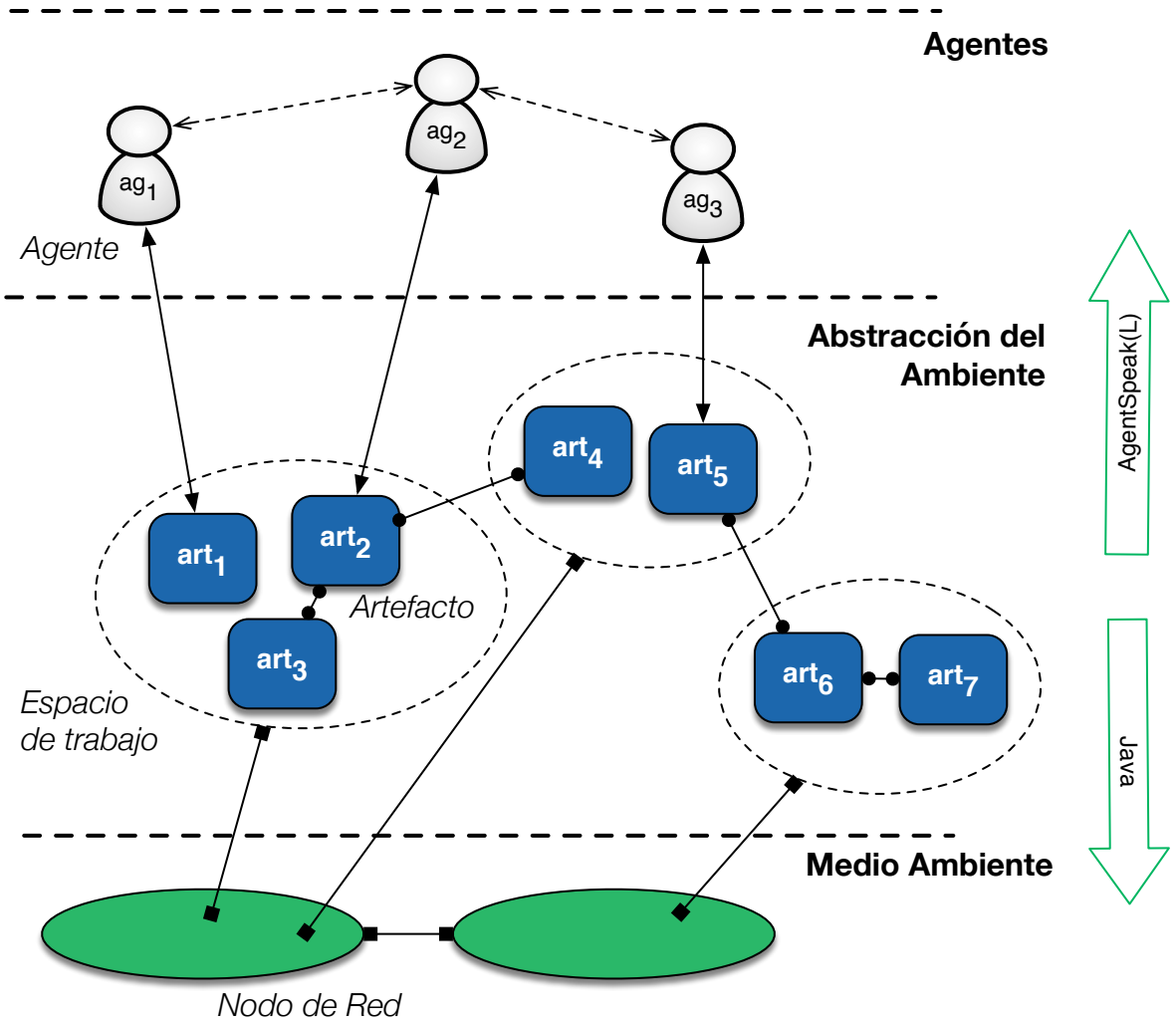
Universidad Veracruzana

Modelo y Metáfora

- La aproximación endógena a los medios ambientes se basa en el meta-modelo de Agentes & Artefactos, tal y como lo implementa CArtAgO.
- La metáfora a seguir es bastante intuitiva:
  - El ambiente se compone de uno o más espacios de trabajo, por ej., salón, oficina, casa, etc.
  - Cada ambiente de trabajo incluye un conjunto de artefactos que el agente puede utilizar. Por ej., proyector, computadora, calculadora, weka, etc.
  - El agente puede trabajar en los diferentes espacios de trabajo, focalizando en diferentes artefactos.

Universidad Veracruzana

La capa de Agentes y Artefactos: JaCa



Universidad Veracruzana

Funciones del ambiente

- Esta concepción de medio ambiente provee funciones de:

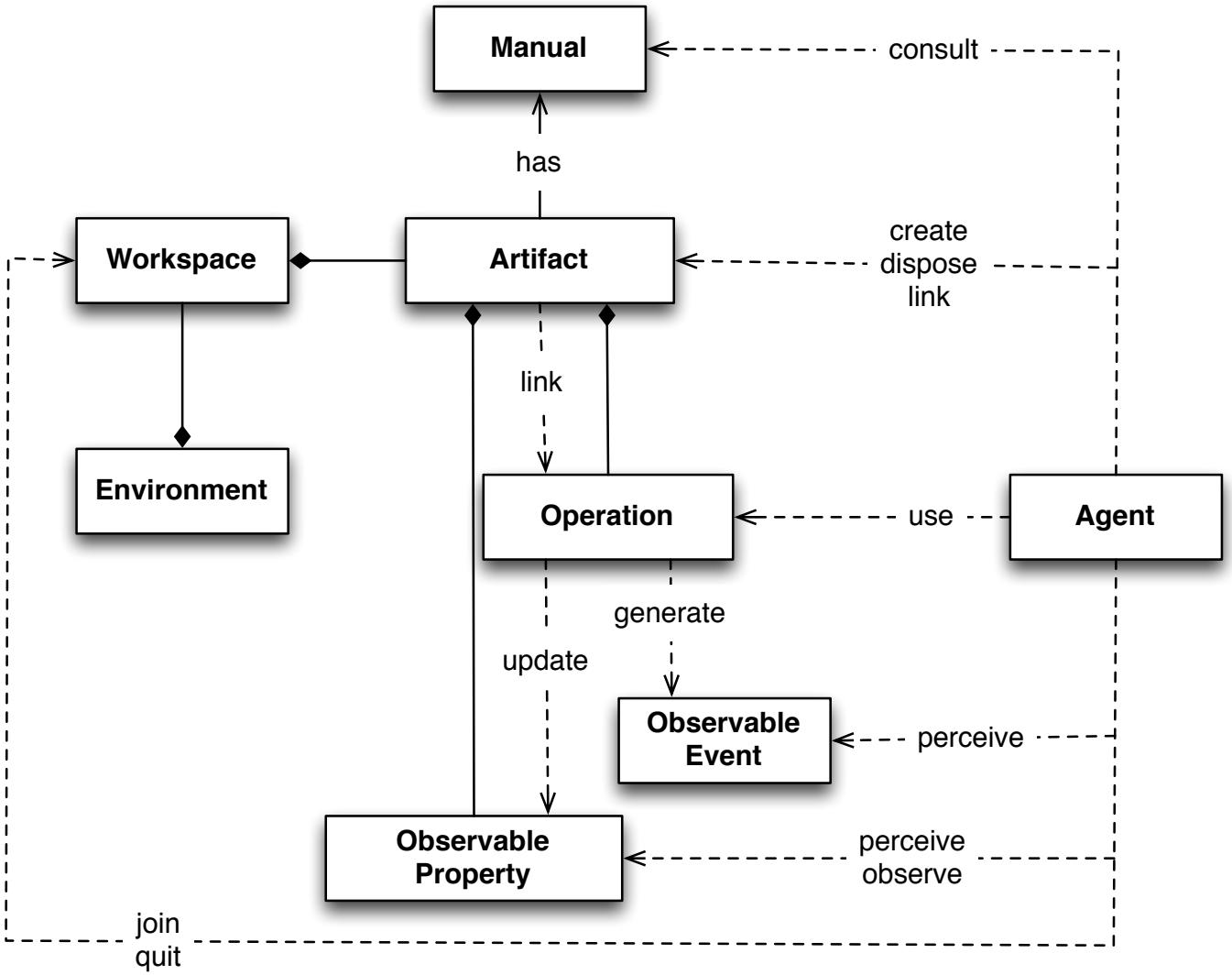
Despliegue. Acceso a los recursos externos de software y hardware con los que el SMA puede interactuar: sensores, actuadores, impresoras, redes, bases de datos, servicios web, paquetes, etc.

Abstracción. Una interfaz entre la representación a nivel agente y los detalles de bajo nivel presentes en el contexto de despliegue, de forma que el programador no necesite acceso a estos últimos.

Mediación. Un mecanismo para regular el acceso a recursos compartidos y para mediar la interacción entre agentes.

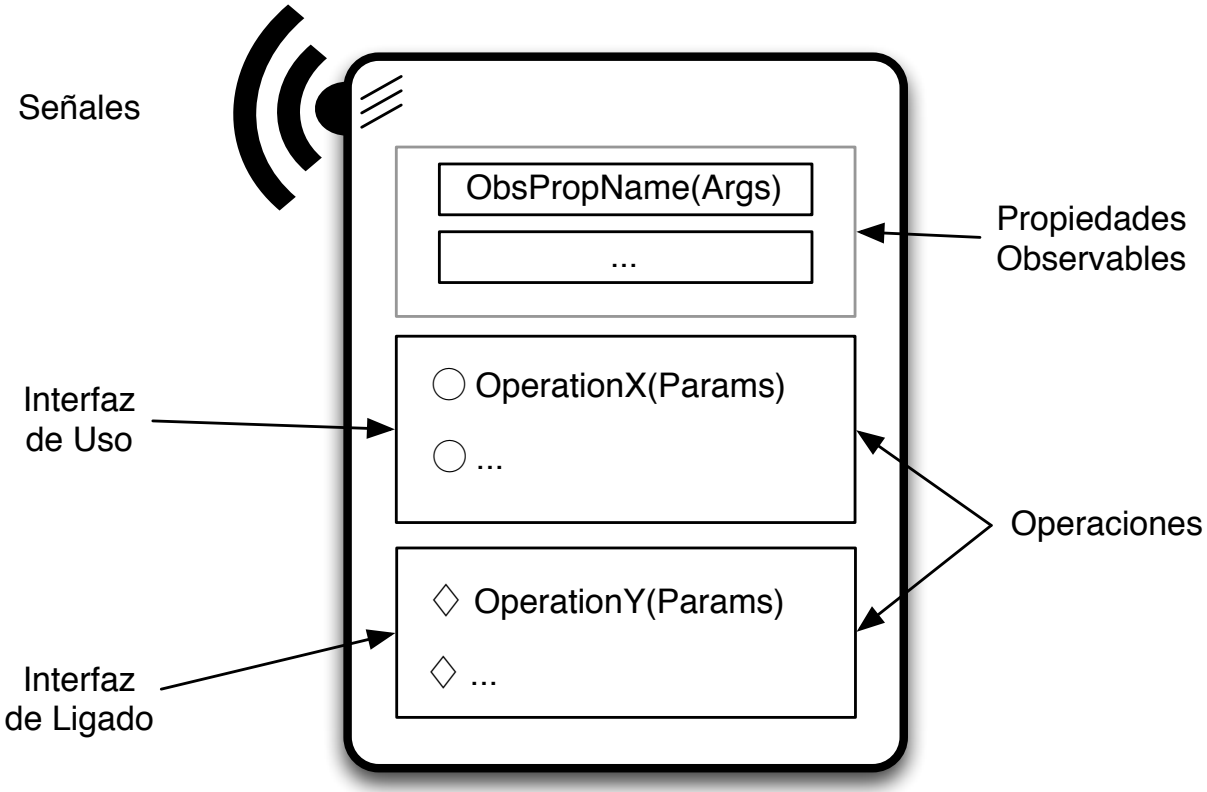
Universidad Veracruzana

Meta modelo de Agentes & Artefactos



Universidad Veracruzana

Artefacto



Universidad Veracruzana

Jacamo

- Para usar el meta-modelo de agentes y artefactos con Jason es necesario instalar Jacamo [1].
- La última versión se puede descargar de:

  `https://github.com/jacamo-lang/jacamo/releases`
- El archivo `jacamo-bin-1.2.zip` provee una instalación similar a la de Jason.
- Agregar a su configuración del *shell* lo siguiente:

```
1 export JACAMO_HOME=/Users/aguerra/Documents/code/jacamo
2 export PATH=$JACAMO_HOME/bin:$PATH
```

- Cuando reinicien su terminal tendrán disponible el comando `jacamo` (análogo a `jason`).

Universidad Veracruzana

Creación, destrucción y búsqueda de artefactos

- `makeArtifact(Nombre, Tipo, Params, Id)` crea un nuevo artefacto en el espacio de trabajo. La variable `Id` es de salida.
- Ej. `makeArtifact("c0","tools.Contador",[0],Id).`
- `disposeArtifact(Id)` remueve al artefacto con identificador `Id`.
- `lookupArtifact(Nombre, Id)` busca un artefacto por nombre, regresando su identificador `Id`.
- Ej. `lookupArtifact("c0", Id)`

Universidad Veracruzana

Uso de un artefacto

- `focus(Id,Filtro)` focaliza la atención del agente en el artefacto `Id`. Opcionalmente, las propiedades observadas pueden ser sujetas a un `Filtro`.
- `stopFocus(Id)` es el dual de la acción anterior. Provee el mecanismo para dejar de observar al artefacto con identificador `Id`.

Universidad Veracruzana

Ligado de artefactos

- `linkArtifacts(Id1, ID2, Puerto)`. El artefacto `ID1` liga a `ID2`. El parámetro `Puerto` es necesario cuando se liga el mismo artefacto a múltiples artefactos. En ese caso, el artefacto `ID1` debe proveer varios puertos etiquetados y adjuntar cada artefacto ligado a un puerto en particular.
- `unlinkArtifacts(Id1, ID2)` es el dual de la operación anterior.

Universidad Veracruzana

Caso 1: Hola mundo

- El SMA es similar a uno de Jason, `jacamo00.jcm`:

```
15 mas jacamo00 {
16
17   agent bob
18
19 }
```

- Y el agente, también! Salvo la librería y la acción externa.

```
7  !start.
8
9  /* Plans */
10
11 +!start
12     <- println("hello world.").
13
14 { include("$jacamo/templates/common-cartago.asl") }
```

Universidad Veracruzana

Caso 2: Contador

- Este SMA ya tiene más elementos propios de Jacamo: Espacios de trabajo y artefactos.

```
15  mas jacamo01 {
16
17    agent observer {
18        join: w
19    }
20
21    agent user {
22        focus: w.c1
23    }
24
25    workspace w {
26        artifact c1: tools.Counter(0)
27    }
28  }
```

Universidad Veracruzana

El artefacto contador

```
1  // CArtAgO artifact code for project jacamo01
2
3  package tools;
4
5  import cartago.*;
6
7  public class Counter extends Artifact {
8    void init(int initialValue) {
9      defineObsProperty("count", initialValue);
10   }
11
12   @OPERATION
13   void inc() {
14     ObsProperty prop = getObsProperty("count");
15     prop.updateValue(prop.intValue()+1);
16     signal("tick");
17   }
18 }
```

Universidad Veracruzana

El agente usuario

```
1  // Agent user in project jacamo01
2
3  /* Initial beliefs and rules */
4
5  /* Initial goals */
6
7  !start.
8
9  /* Plans */
10
11 +!start
12     <- println("Hello,world! Using my counter artifact...");
13        inc;
14        inc;
15        inc.
16
17 { include("$jacamo/templates/common-cartago.asl") }
```

Universidad Veracruzana

El agente observador

```
1  // Agent observador in jacamo01.jcm
2
3  /* Initial goals */
4
5  !observe.
6
7  /* Plans */
8
9  +!observe
10     <- focusWhenAvailable(c1).
11
12 +count(V)[artifact_name(ArtifactName)]
13     <- .print("New observed value in artifact ",ArtifactName, ": ",V).
14
15 +tick
16     <- .print("New tick perceived!").
17
18 { include("$jacamo/templates/common-cartago.asl") }
```

Universidad Veracruzana

La ejecución

- La salida en consola de este ejemplo suele ser:

```
[Cartago] Workspace w created.
[Cartago] artifact c1: tools.Counter(0) at w created.
[observer] join workspace /main/w: done
[user] join workspace /main/w: done
[user] focusing on artifact c1 (at workspace /main/w) using namespace
↪  default
[user] focus on c1: done
Hello,world! Using my counter artifact...
[observer] New observed value in artifact c1: 0
[observer] New tick perceived!
[observer] New observed value in artifact c1: 1
[observer] New tick perceived!
[observer] New observed value in artifact c1: 2
[observer] New tick perceived!
[observer] New observed value in artifact c1: 3
```

Universidad Veracruzana

Caso 3: Fallo en las acciones

- Veamos otro sistema usuario/observador.

```
15  mas jacamo02 {
16
17    agent usuario
18    agent observador
19
20  }
```

- Pero con un contador acotado.

Universidad Veracruzana

El artefacto contador acotado

```
1  package tools;
2
3  import cartago.*;
4
5  public class ContadorAcotado extends Artifact {
6    private int max;
7
8    void init(int max){
9      defineObsProperty("valor",0);
10     this.max = max;
11   }
12
13   @OPERATION void inc(){
14     ObsProperty prop = getObsProperty("valor");
15     if (prop.intValue() < max) {
16       prop.updateValue(prop.intValue()+1);
17       signal("tic");
18     } else {
19       failed("La operación inc falló","inc_fallo","max_alcanzado",max);
20     }
21   }
22 }
```

El agente usuario

```
1  // Agent usuario in project cartagoErrorAccion.ma2j
2
3  /* Initial goals */
4  !creaUsaCont.
5
6  /* Plans */
7  +!creaUsaCont : true
8    <-  !crea(C);
9        !usa(C).
10
11 +!crea(C)
12   <-  makeArtifact("contador00","tools.ContadorAcotado",[50],C).
13
14 +!usa(C)
15   <-  for (.range(I,1,100)) {
16         inc[artifact_id(C)];
17       }.
18
19 -!usa(C)[error_msg(Msg),env_failure_reason(inc_fallo("max_alcanzado",Val))]
20   <-  .print(Msg);
21       .print("Ultimo valor fue ",Val).
22
```

Universidad Veracruzana

El agente observador

```
1  // Agent observador in project cartagoErrorAccion.mas2j
2
3  /* Initial goals */
4  !observa.
5
6  /* Plans */
7  +!observa
8    <-  .wait(10);
9        lookupArtifact("contador00",Id);
10       focus(Id).
11
12 +valor(V)[artifact_name(ArtName)]
13   <-  .print("Un nuevo valor observado para ", ArtName, ": ",V).
14
15 +tic[artifact_name(ArtName)]
16   <-  .print("Tic percibido,proveniente de ", ArtName).
```

Universidad Veracruzana

La ejecución

```
...
[observador] Tic percibido
[observador] Un nuevo valor observado para contador00: 12
[usuario] La operación inc falló1
...
[observador] Un nuevo valor observado para contador00: 19
[usuario] Ultimo valor fue 50
[observador] Tic percibido
[observador] Un nuevo valor observado para contador00: 20
...
```

Universidad Veracruzana

Caso 4: Operaciones con entrada y salida

- Un sistema con un solo usuario

```
15  mas jacamo03 {
16
17      agent usuario
18
19  }
```

- De una calculadora.

Universidad Veracruzana

El artefacto calculadora

```
1  package tools;
2
3  import cartago.*;
4
5  public class Calculadora extends Artifact {
6
7    @OPERATION
8    void suma(double a, double b, OpFeedbackParam<Double> suma) {
9      suma.set(a+b);
10   }
11
12   @OPERATION
13   void resta(double a, double b, OpFeedbackParam<Double> resta) {
14     resta.set(a-b);
15   }
16 }
```

Universidad Veracruzana

El agente usuario

```
1  // Agent usuario in project cartagoSalidaOps.mas2j
2
3  /* Initial goals */
4  !usaCalculadora.
5
6  /* Plans */
7  +!usaCalculadora <-
8    makeArtifact("miCalculadora","tools.Calculadora",[],_);
9    suma(4,5,Suma);
10   .print("La suma de 4 y 5 es ", Suma);
11   resta(4.0,5.0,Resta);
12   .print("La resta de 4 y 5 es ", Resta).
```

Universidad Veracruzana

La ejecución

```
1 [usuario] La suma de 4 y 5 es 9
2 [usuario] La resta de 4 y 5 es -1
```

Universidad Veracruzana

Caso 5: Operaciones con guardia

- Consideremos ahora un SMA con un proveedor de datos y un consumidor de ellos:

```
15  mas jacamo04 {
16
17      agent productor
18      agent consumidor
19
20  }
```

- Usando un buffer acotado para sincronizarlos.

Universidad Veracruzana

El artefacto buffer acotado

```
1  package tools;
2
3  import cartago.*;
4  import java.util.*;
5
6  public class BuferAcotado extends Artifact {
7
8    private LinkedList<Object> elems;
9    private int nmax;
10
11   void init(int nmax) {
12     elems = new LinkedList<Object>();
13     defineObsProperty("nElems",0);
14     this.nmax = nmax;
15   }
16
17   @OPERATION(guard="buferNoLleno")
18   void poner(Object obj) {
19     elems.add(obj);
20     getObsProperty("nElems").updateValue(elems.size());
21   }
```

Universidad Veracruzana

El artefacto buffer acotado, cont...

```
23      @OPERATION(guard="elemDisponible")
24      void obtener(OpFeedbackParam<Object> res) {
25        Object elem = elems.removeFirst();
26        res.set(elem);
27        getObsProperty("nElems").updateValue(elems.size());
28      }
29
30      @GUARD
31      boolean elemDisponible(OpFeedbackParam<Object> res){
32        return elems.size() > 0;
33      }
34
35      @GUARD
36      boolean buferNoLleno(Object obj){
37        return elems.size() < nmax;
38      }
39    }
```

Universidad Veracruzana

El agente productor I

```
1  // Agent productor in project cartago04.mas2j
2
3  /* Initial bels */
4  elemAProducir(0).
5
6  /* Initial goals */
7  !producir.
8
9  /* Plans */
10 +!producir <-
11   !crear(Bufer);
12   !producirElems.
13
14 +!producirElems <-
15   ?proxElemAProducir(E);
16   poner(E);
17   !!producirElems.
18
19 +?proxElemAProducir(N) <-
20   -elemAProducir(N);
21   +elemAProducir(N+1).
```

Universidad Veracruzana

El agente productor II

```
22
23  +!crear(B) <-
24    makeArtifact("miBufer","tools.BuferAcotado",[1],B).
25
26  -!crear(B) <-
27    lookupArtifact("miBufer",B).
```

Universidad Veracruzana

El agente consumidor I

```
1  // Agent consumidor in project cartago04.mas2j
2
3
4  /* Initial goals */
5  !consumir.
6
7  /* Plans */
8  +!consumir <-
9    ?buferListo;
10   !consumirElems.
11
12 +!consumirElems <-
13   obtener(Elem);
14   !imprimirElem(Elem);
15   !!consumirElems.
16
17 +!imprimirElem(E) <-
18   .my_name(Me);
19   .print(Me,": ",E).
20
21 +?buferListo <-
```

Universidad Veracruzana

El agente consumidor II

```
22      lookupArtifact("miBufer",_).
23
24    -?buferListo <-
25      .wait(50);
26      ?buferListo.
```

Universidad Veracruzana

La ejecución

```
1  [consumidor] consumidor: 0
2  [consumidor] consumidor: 1
3  [consumidor] consumidor: 2
4  [consumidor] consumidor: 3
5  [consumidor] consumidor: 4
6  [consumidor] consumidor: 5
7  ...
```

Universidad Veracruzana

Caso 6: Operaciones estructuradas

- Ahora veremos un SMA de dos agentes que usan esperas en su artefacto:

```
15  mas jacamo05 {
16
17    agent usuario1
18    agent usuario2
19
20  }
```

Universidad Veracruzana

Un artefacto complejo I

```
1  package tools;
2
3  import cartago.*;
4
5  public class ArtefactoComplejo extends Artifact {
6    int ContadorInterno;
7
8    void init() {
9      ContadorInterno = 0;
10   }
11
12   @OPERATION void operacionCompleja(int nVeces){
13     trabaja();
14     signal("paso1Completado",ContadorInterno);
15     await("miCondicion",nVeces);
16     signal("paso2Completado",ContadorInterno);
17   }
18
19   @GUARD boolean miCondicion(int nVeces) {
20     return ContadorInterno >= nVeces;
21   }
```

Universidad Veracruzana

Un artefacto complejo II

```
22
23      @OPERATION void actualiza(int delta) {
24        ContadorInterno += delta;
25      }
26
27      private void trabaja(){}
28    }
```

Universidad Veracruzana

El agente usuario1

```
1  /* Initial goals */
2  !prueba.
3
4  /* Plans */
5
6  @prueba
7  +!prueba
8    <-   .print("Creando el artefacto...");
9         makeArtifact("a0","tools.ArtefactoComplejo",[],Id);
10     focus(Id);
11     .print("Ejecutando la acción compleja... ");
12     operacionCompleja(5);
13     .print("Acción completada.").
14
15 +paso1Completado(C)
16   <-   .print("Primer paso completado. Contador = ",C).
17
18 +paso2Completado(C)
19   <-   .print("Segundo paso completado. Contador = ",C).
```

Universidad Veracruzana

El agente usuario2 I

```
1  /* Initial goals */
2  !prueba.
3
4  /* Plans */
5
6  +!prueba
7    <-  !descubreArtefacto("a0");
8        !usaArtefacto(10).
9
10 +!usaArtefacto(N) : N>0
11   <-  actualiza(1);
12       .print("Artefacto actualizado.");
13       !usaArtefacto(N-1).
14
15 +!usaArtefacto(0)
16   <-   .print("Uso del artefacto completado").
17
18 +!descrubreArtefacto(NombreArt)
19   <-   lookupArtifact(NombreArt,_).
20
21 -!descubreArtefacto(NombreArt)
```

Universidad Veracruzana

El agente usuario2 II

```
22      <-    .wait(10);
23          !!descubreArtefacto(NombreArt).
24
25
```

Universidad Veracruzana

La ejecución

```
1  [usuario1] Creando el artefacto...
2  [usuario1] Ejecutando la acción compleja...
3  [usuario1] Primer paso completado. Contador = 0
4  [usuario2] Artefacto actualizado.
5  [usuario2] Artefacto actualizado.
6  [usuario1] Segundo paso completado. Contador = 6
7  [usuario1] Acción completada.
8  [usuario2] Artefacto actualizado.
9  [usuario2] Artefacto actualizado.
10 [usuario2] Artefacto actualizado.
11 [usuario2] Artefacto actualizado.
12 [usuario2] Artefacto actualizado.
13 [usuario2] Artefacto actualizado.
14 [usuario2] Artefacto actualizado.
15 [usuario2] Artefacto actualizado.
16 [usuario2] Uso del artefacto completado
```

Universidad Veracruzana

Caso 7: Una interfaz gráfica

- El agente usa una intefaz gráfica implementada como un artefacto:

```
15  mas jacamo07 {
16
17      agent usuario
18  }
```

Universidad Veracruzana

Un artefacto interfaz gráfica I

```
1  package tools;
2
3  import cartago.*;
4  import cartago.tools.*;
5  import javax.swing.*;
6  import java.awt.event.*;
7
8  public class Gui extends GUIArtifact {
9
10   private MiMarco marco;
11
12   public void setup() {
13     marco = new MiMarco();
14     linkActionEventToOp(marco.botonOk,"ok");
15     linkKeyStrokeToOp(marco.texto,"ENTER","textoModificado");
16     linkWindowClosingEventToOp(marco,"cerrado");
17
18     defineObsProperty("valor",obtenerValor());
19     marco.setVisible(true);
20   }
21
```

Universidad Veracruzana

Un artefacto interfaz gráfica II

```
22    @INTERNAL_OPERATION void ok(ActionEvent ev) {
23      signal("ok");
24    }
25
26    @INTERNAL_OPERATION void cerrado(WindowEvent ev) {
27      signal("cerrado");
28    }
29
30    @INTERNAL_OPERATION void textoModificado(ActionEvent ev){
31      getObsProperty("valor").updateValue(obtenerValor());
32    }
33
34    @OPERATION void asignarValor(int valor){
35      marco.asignarTexto(""+valor);
36      getObsProperty("valor").updateValue(obtenerValor());
37    }
38
39    private int obtenerValor(){
40      return Integer.parseInt(marco.obtenerTexto());
41    }
```

Universidad Veracruzana

La clase MiMarco I

```
43    class MiMarco extends JFrame {
44
45    private static final long serialVersionUID = 1L;
46
47    private JButton botonOk;
48      private JTextField texto;
49
50      public MiMarco(){
51        setTitle("Artefacto como GUI");
52        setSize(200,100);
53        JPanel panel = new JPanel();
54        setContentPane(panel);
55        botonOk = new JButton("incrementa");
56        botonOk.setSize(80,50);
57        texto = new JTextField(10);
58        texto.setText("0");
59        texto.setEditable(true);
60        panel.add(texto);
61        panel.add(botonOk);
62      }
63
```

Universidad Veracruzana

La clase MiMarco II

```
64      public String obtenerTexto() {
65        return texto.getText();
66      }
67
68      public void asignarTexto(String s) {
```

Universidad Veracruzana

El agente gui I

```
1  // Agent gui in project cartago07.mas2j
2
3  /* Initial goals */
4  !prueba.
5
6  /* Plans */
7
8  +!prueba2 <- // Test with the REPL agent
9    makeArtifact("gui","tools.Gui",[],Id);
10   focus(Id);
11   asignarValor(1000).
12
13 +!prueba <-
14   makeArtifact("gui","tools.Gui",[],Id);
15   focus(Id).
16
17 +valor(V) <-
18   .print("Valor actualizado: ",V).
19
20 +ok : valor(V) <-
21   asignarValor(V+1).
```

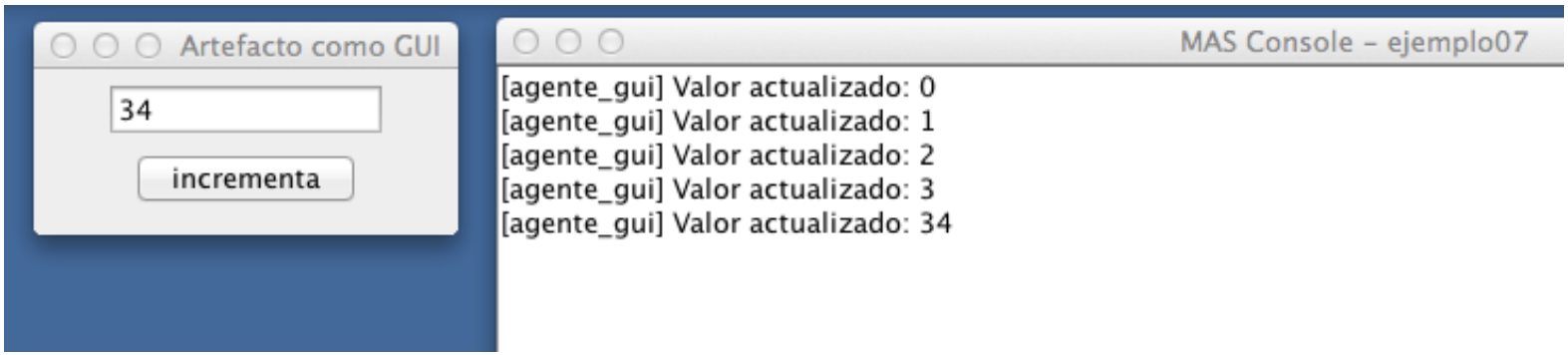
Universidad Veracruzana

El agente gui II

```
22
23  +cerrado <-
24    .my_name(Yo);
25      .print("Adios");
26      .kill_agent(Yo).
```

Universidad Veracruzana

La ejecución



Universidad Veracruzana

Caso 8: Operaciones de ligado

- Un agente que liga dos artefactos para computar su meta:

```
15  mas jacamo08 {
16
17      agent agente
18
19  }
```

Universidad Veracruzana

El artefacto ligable

```
1  package tools;
2  import cartago.*;
3
4  public class ArtefactoLigable extends Artifact {
5
6    int contador;
7
8    void init(){
9      contador = 0;
10   }
11
12   @LINK void inc(){
13     log("inc invocada.");
14     contador++;
15   }
16
17   @LINK void obtenerValor(OpFeedbackParam<Integer> v){
18     log("obtenterValor invocada.");
19     v.set(contador);
20   }
21 }
```

Universidad Veracruzana

El artefacto ligador I

```
1  package tools;
2  import cartago.*;
3
4  @ARTIFACT_INFO(
5    outports = { @OUTPORT(name="salida1") }
6  )
7
8  public class ArtefactoLigador extends Artifact{
9
10   @OPERATION void test1() {
11     log("Ejecutando test1.");
12     try{
13       execLinkedOp("salida1","inc");
14     } catch (Exception ex) {
15       ex.printStackTrace();
16     }
17   }
18
19   @OPERATION void test2(OpFeedbackParam<Integer> v) {
20     log("Ejecutado test2.");
21     try{
```

El artefacto ligador II

```
22          execLinkedOp("salida1","obtenerValor",v);
23          log("Valor regresado: " + v.get());
24        } catch (Exception ex){
25          ex.printStackTrace();
26        }
27      }
```

Universidad Veracruzana

El artefacto ligador, cont...

```
29  @OPERATION void test3() {
30    log("Ejecutando test3.");
31    try{
32      ArtifactId id = makeArtifact("nuevoLigado",
33        "tools.ArtefactoLigable",ArtifactConfig.DEFAULT_CONFIG);
34      execLinkedOp(id,"inc");
35    } catch (Exception ex) {
36      ex.printStackTrace();
37    }
38  }
```

Universidad Veracruzana

El agente

```
1  // Agent agente in project cartagoLigado.mas2j
2
3  /* Initial beliefs and rules */
4
5  /* Initial goals */
6
7  !start.
8
9  /* Plans */
10
11 +!start <-
12   makeArtifact("miArtefacto","tools.ArtefactoLigador",[],Id1);
13   makeArtifact("contador","tools.ArtefactoLigable",[],Id2);
14   linkArtifacts(Id1,"salida1",Id2);
15   println("Artefactos ligados: procede prueba.");
16   test1;
17   test2(V);
18   println("El valor regresado es: ",V);
19   test3.
20
```

Universidad Veracruzana

La ejecución

```
1  [agente] Artefactos ligados: procede prueba.
2  [miArtefacto] Ejecutando test1.
3  [contador] inc invocada.
4  [miArtefacto] Ejecutado test2.
5  [contador] obtenterValor invocada.
6  [miArtefacto] Valor regresado: 1
7  [agente] El valor regresado es: 1
8  [miArtefacto] Ejecutando test3.
9  [nuevoLigado] inc invocada.
```

Universidad Veracruzana

Caso 9: Espacios de Trabajo

- Los agentes se pueden mover a diferentes espacios de trabajo:

```
15  mas jacamo09 {
16
17      agent viajero
18
19  }
```

Universidad Veracruzana

El agente I

```
1  // Agent agente in project cartagoWS.mas2j
2
3  /* Initial goals */
4
5  !start.
6
7  /* Plans */
8
9  +!start <-
10     ?joinedWsp(V1,V2,V3);
11     .print("V1: ", V1, " V2: ", V2, " V3: ", V3);
12     .print("Creando nuevos espacios de trabajo...");
13     createWorkspace("ws1");
14     createWorkspace("ws2");
15     joinWorkspace("ws1",WS1Id);
16     ?joinedWsp(WS1Id,_,WS1Name);
17     .print("Ahora en el espacio de trabajo ",WS1Name);
18     makeArtifact("miContador","tools.Contador",[],ArtId);
19     focus(ArtId);
20     joinWorkspace("/main/ws2",WS2Id);
21     ?joinedWsp(WS2Id,_,WS2Name);
```

Universidad Veracruzana

El agente II

```
22      .print("Ahora en el espacio ",WS2Name);
23      .print("Usando un artefacto de otro espacio de trabajo...");
24      inc[artifact_id(ArtId)];
25      joinWorkspace("/main/ws1",WS1Id2);
26      .print("Hola,de nuevo en ",WS1Id2);
27      .print("Usando artefacto en el espacio actual...");
28      inc;
29    ?joinedWsp(WS1Id2,_,WS1Name2);
30      .print("Saliendo de ", WS1Name2, "...");
31      quitWorkspace(WS1Id2).
32      //?joinedWsp(,WS3Id,WS3Name);
33      //println("De regreso en ",WS3Name);
34      //joinWorkspace(ws1,);
35      //?joinedWsp(ws1,,);
36      //println("... y finalmente en el espacio ",ws1," otra vez.").
37
38    +valor(V)[artifact_name(ArtName), workspace(Wsp,_)]
39      <- .print("Valor actualizado en el artefacto ", ArtName, " en ", Wsp,
      ↪  ": ",V).
40
41    +tick <- .print("Ouch").
```

Universidad Veracruzana

El agente III

```
42
43  { include("$jacamo/templates/common-cartago.asl") }
```

Universidad Veracruzana

La corrida del agente

```
1  Runtime Services (RTS) is running at 127.0.0.1:56088
2  Agent mind inspector is running at http://127.0.0.1:3272
3  CArtAgO Http Server running on http://127.0.0.1:3273
4  [viajero] V1: cobj_0 V2: main V3: /main
5  [viajero] Creando nuevos espacios de trabajo...
6  [viajero] Ahora en el espacio de trabajo /main/ws1
7  [viajero] Valor actualizado en el artefacto miContador en /main/ws1: 0
8  [viajero] Ahora en el espacio /main/ws2
9  [viajero] Usando un artefacto de otro espacio de trabajo...
10 [viajero] Ouch
11 [viajero] Valor actualizado en el artefacto miContador en /main/ws1: 1
12 [viajero] Hola,de nuevo en cobj_2
13 [viajero] Usando artefacto en el espacio actual...
14 [viajero] Ouch
15 [viajero] Valor actualizado en el artefacto miContador en /main/ws1: 2
16 [viajero] Saliendo de /main/ws1...
```

Universidad Veracruzana

Referencias

[1] O Boissier et al. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems using JaCaMo*. Intelligent Robotics and Autonomous Agents. Cambridge, MA, USA: MIT Press, 2020.

[2] RH Bordini, JF Hübner y M Wooldridge. *Programming Multi-Agent Systems in Agent-Speak using Jason*. John Wiley & Sons Ltd, 2007.

[3] A Ricci, M Piunti y M Viroli. "Environment programming in multi-agent systems: an artifact-based perspective". En: *Autonomous Agents and Multi-Agent Systems* 23.2 (2011), págs. 158-192.

Universidad Veracruzana