Agentes Inteligentes (AIN)

Presentación de pyGOMAS

Agentes Inteligentes (AIN)

- Profesorado:
  - Carlos Carrascosa (carrasco@dsic.upv.es)
  - Vicente Julián (vinglada@dsic.upv.es)

Prácticas

					P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
M					9-3	16-3	23-3	20-4	27-4	4-5	11-5	18-5	25-5	1-6

Introduction

SPADE

Smart Python Agent Dev Env

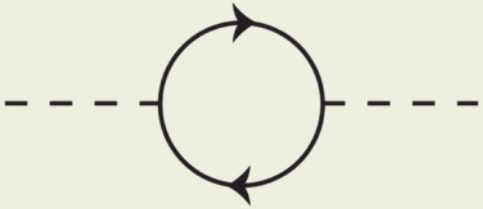



based on:

Python >= 3.6

JABBER / XMPP

AsyncIO



Agente Híbrido SPADE

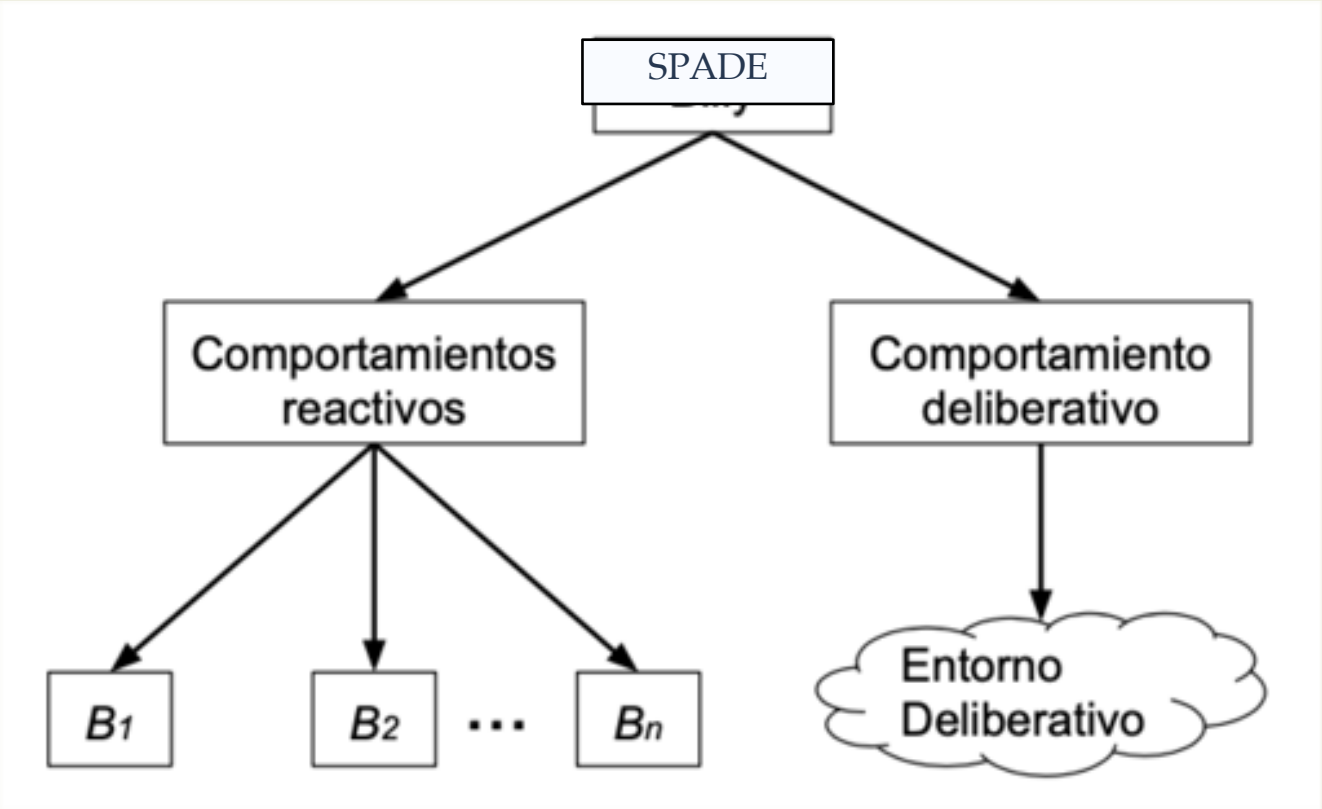
Instalación

https://spade-mas.readthedocs.io

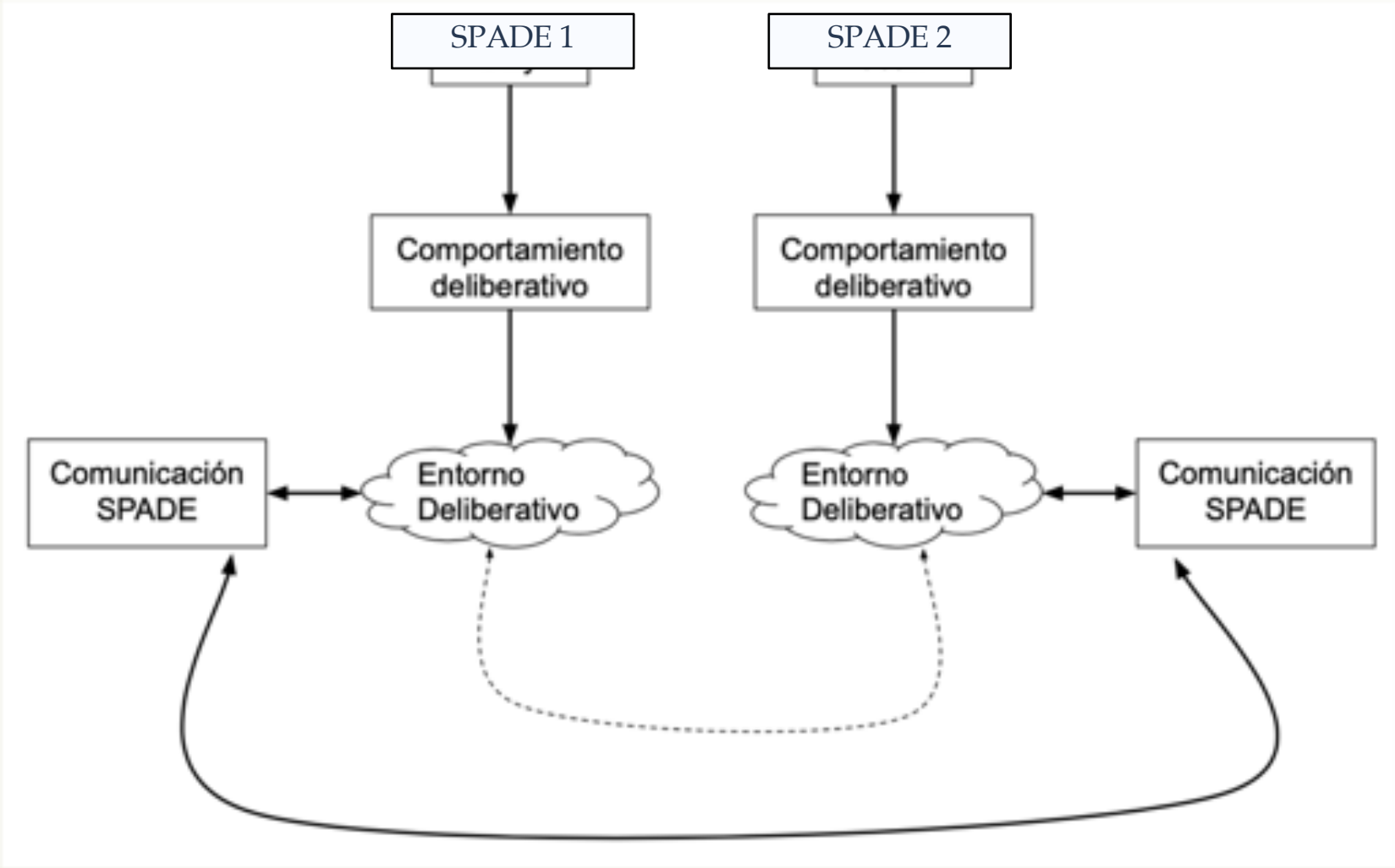
- GitHub:
  - Descargar código de: https://github.com/javipalanca/spade_bdi
  - `python setup.py install`
- PyPi:
  - Anaconda: Crear un entorno con Python 3.7
  - `$ pip install spade_bdi`

```
(Optional) Installing XMPP Server: Prosody IM:

        - Install Homebrew:

                $ /usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install
/master/install)"

                $ brew tap prosody/prosody

                $ brew install prosody

        - Configure Prosody IM: Activate in-band
register (in prosody is: allow_registration=true in
prosody.cfg.lua)

Launch Prosody IM
```

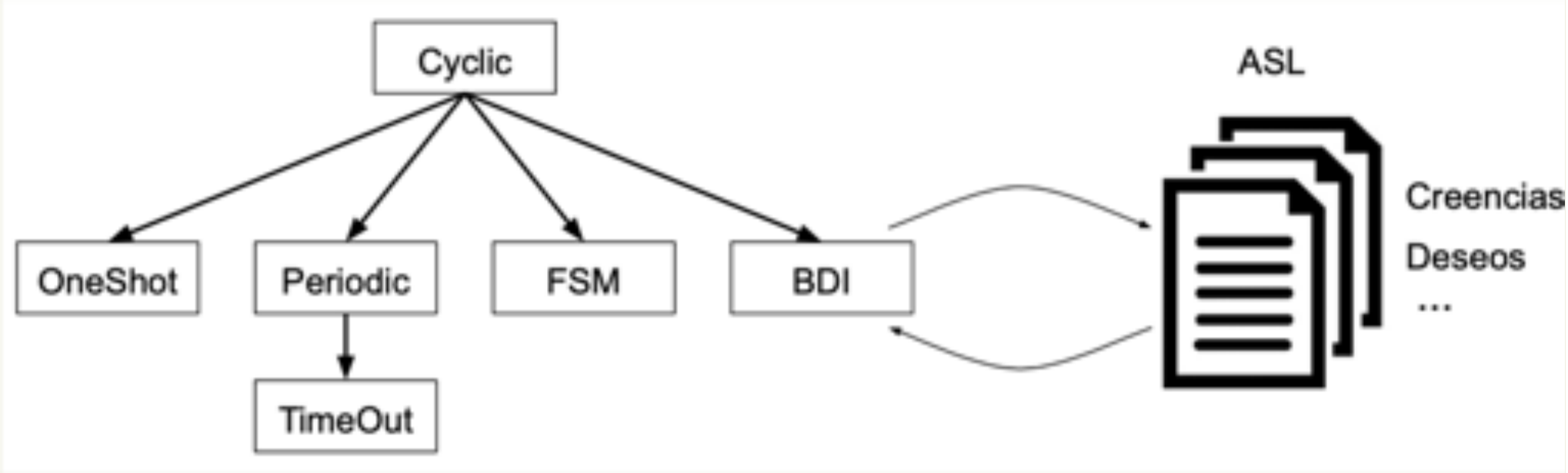
Agente Híbrido SPADE



Agente Híbrido SPADE

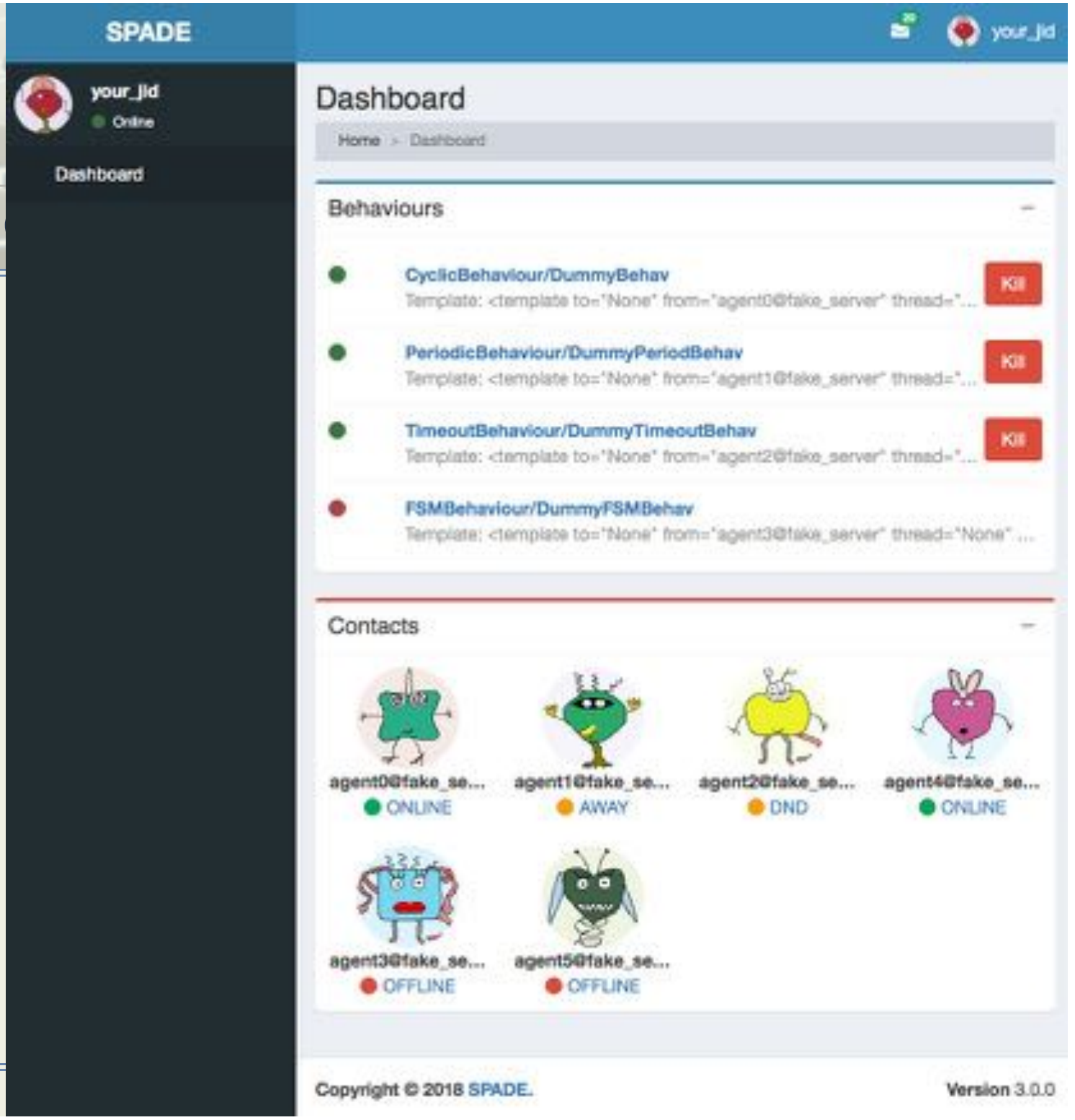


Agente Híbrido SPADE

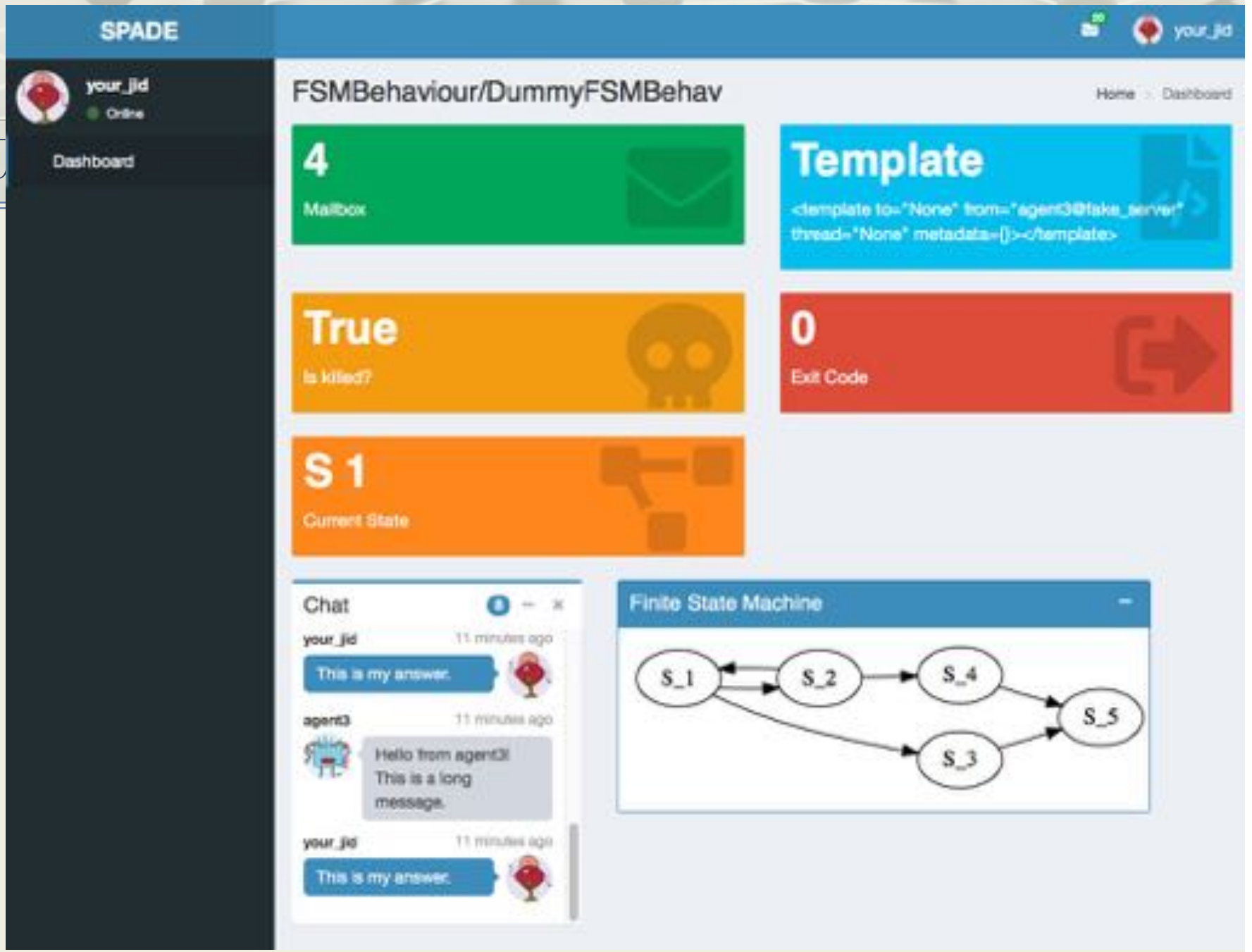


Introdu

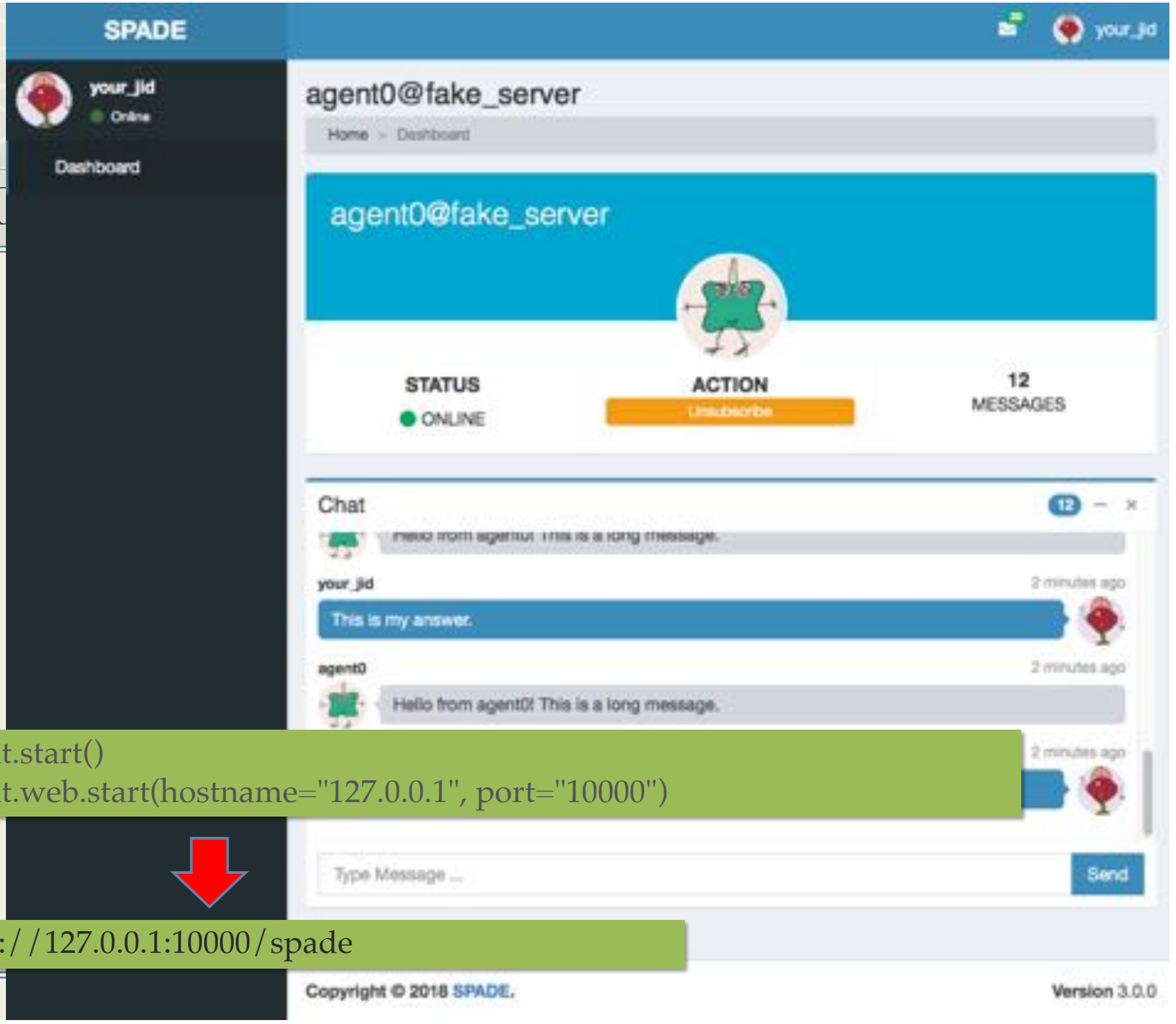
User Interface



User Interface



User Interface




```
agent.start()
agent.web.start(hostname="127.0.0.1", port="10000")
```

http://127.0.0.1:10000/spade

Agente Híbrido SPADE

- Inserción, eliminación y modificación de creencias desde dentro del comportamiento deliberativo (`BDIBehaviour`)
  - `get_belief`: consultar una creencia que tenga el agente.
  - `get_beliefs`: consultar todas las creencias.
  - `get_belief_value`: consultar el valor que toma una creencia.
  - `set_belief`: establecer o modificar una creencia.
  - `remove_belief`: eliminar una creencia que tenga el agente.
- Comunicación
  - `.send`: con el contenido a enviar, destinatario y receptor. En los metadatos se especifica la ilocución y la performativa ("BDI") con que se enviará el mensaje por el Dispatcher.
  - `Performativas: tell, untell y achieve`

Agente Híbrido SPADE

Agente Hello

```python
import argparse

from spade import quit_spade

from spade_bdi.bdi import BDIAgent

parser = argparse.ArgumentParser(description='spade bdi hello example')
parser.add_argument('--login', type=str, default="basicagent", help='your UPV login.')
parser.add_argument('--server', type=str, default="localhost", help='XMPP server address.')
parser.add_argument('--password', type=str, default="bdipassword", help='XMPP password for the agents.')
args = parser.parse_args()

a = BDIAgent("HelloAgent_{}@{}".format(args.login,args.server), args.password, "hello.asl")

a.start()

import time
time.sleep(1)

a.stop().result()

quit_spade()
```

```
!start.

+!start <-
        .print("Hello World!").
```

$ python hello.py --login <user_login> --server gtirouter.dsic.upv.es

Agente Híbrido SPADE

Agente Básico

```python
import argparse

from spade import quit_spade

from spade_bdi.bdi import BDIAgent

parser = argparse.ArgumentParser(description='spade bdi basic example')
parser.add_argument('--login', type=str, default="basicagent", help='your UPV login.')
parser.add_argument('--server', type=str, default="localhost", help='XMPP server address.')
parser.add_argument('--password', type=str, default="bdipassword", help='XMPP password for the agents.')
args = parser.parse_args()

a = BDIAgent("BasicAgent_{}@{}".format(args.login,args.server), args.password, "basic.asl")

a.start()

import time
time.sleep(1)

a.bdi.set_belief("car", "azul", "big")
a.bdi.print_beliefs()
print("GETTING FIRST CAR BELIEF")
print(a.bdi.get_belief("car"))
a.bdi.print_beliefs()
a.bdi.remove_belief("car", 'azul', "big")
a.bdi.print_beliefs()
print(a.bdi.get_beliefs())
a.bdi.set_belief("car", 'amarillo')

time.sleep(1)

a.stop().result()

quit_spade()
```

```
!start.

+!start <-
    +car(rojo);
    +truck(azul).

+car(Color)
 <- .print("El carro es ",Color).
```

$ python basic.py --login <user_login> --server gtirouter.dsic.upv.es

Agente Híbrido SPADE

Agente Actions

```
arguments = parser.parse_args()


class MyCustomBDIAgent(BDIAgent):
    def add_custom_actions(self, actions):
        @actions.add_function(".my_function", (int,))
        def _my_function(x):
            return x * x

        @actions.add(".my_action", 1)
        def _my_action(agent, term, intention):
            arg = agentspeak.grounded(term.args[0], intention.scope)
            print(arg)
            yield


a = MyCustomBDIAgent("{}@{}".format(arguments.login, arguments.server), arguments.password, "actions.asl")

a.start()
```

```
!start.

+!start <-
    .my_function(4, R);
    .my_action(R).
```

$ python actions.py --login <user_login> --server gtirouter.dsic.upv.es

Agente Híbrido SPADE

Agente Launcher

```
import argparse

from spade import quit_spade

from spade_bdi.bdi import BDIAgent

parser = argparse.ArgumentParser(description='spade bdi launcher example')
parser.add_argument('--login', type=str, default="basicagent", help='your UPV login.')
parser.add_argument('--server', type=str, default="localhost", help='XMPP server address.')
parser.add_argument('--password', type=str, default="bdipassword", help='XMPP password for the agents.')
parser.add_argument('--asl', type=str, default="default.asl", help='asl file with JASON code.')
parser.add_argument('--time', type=int, default=1, help='duration time (in seconds) of the execution.')
args = parser.parse_args()

a = BDIAgent("Agent_{}@{}".format(args.login,args.server), args.password, args.asl)

a.start()

import time
time.sleep(args.time)

a.stop().result()

quit_spade()
```

ASL

$ python asl_launcher.py --login <user_login> --server gtirouter.dsic.upv.es
--asl <nombre_fichero.asl> --time <máximo_segs_ejecución>

Agente Híbrido SPADE

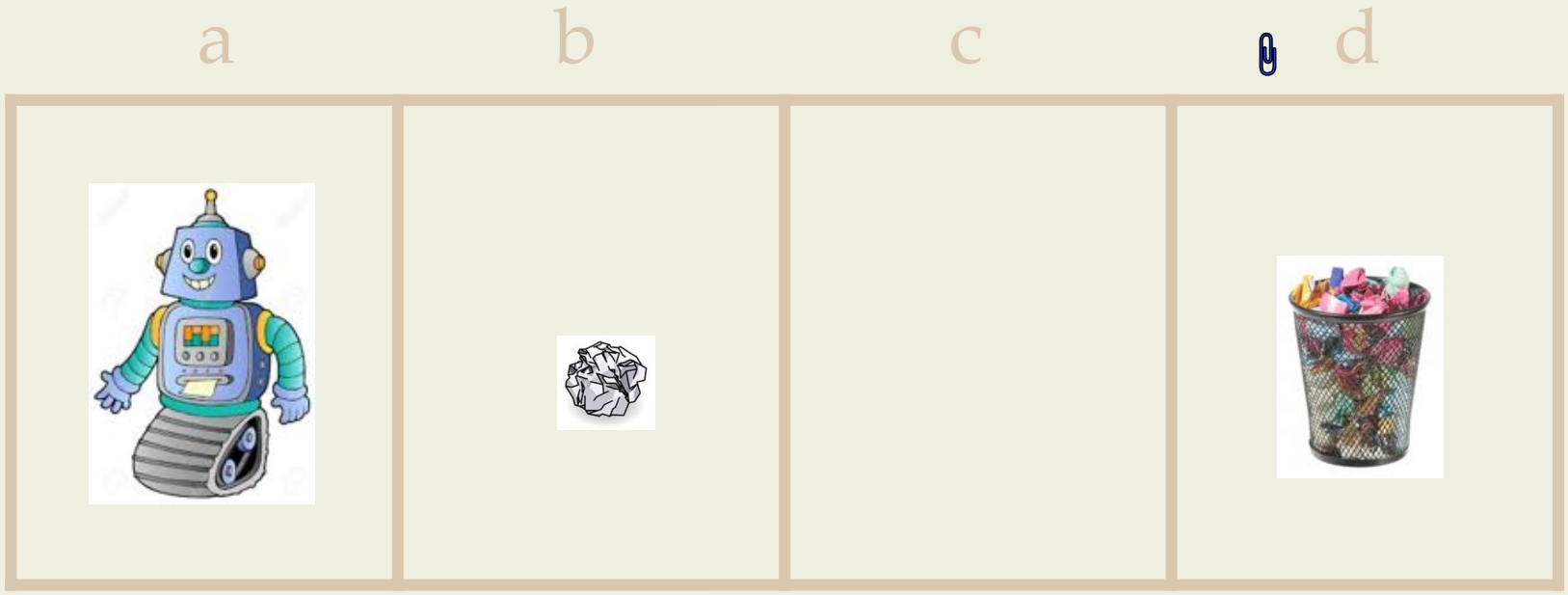
Agente Launcher 2

- Usando el agente *asl_launcher* comentado, ejecutar el ejemplo del factorial.

```
/* Creencias iniciales */
fact(0,1).

/* Planes */
+fact(X,Y) : X < 5
  <-   +fact(X+1,  (X+1) * Y ).

+fact(X,Y) : X = 5
  <-   .print("fact 5 == ", Y ).
```

Agente Híbrido SPADE

Agente robot limpiador



- Usando el agente *asl_launcher* comentado, ejecutar el ejemplo del robot limpiador con la situación inicial del dibujo.

Agente Híbrido SPADE

Agente robot limpiador

- Robot que trata de limpiar basura
  - Movimientos en una sola dimensión
  - La basura hay que cogerla y llevarla a la papelera
- Las creencias iniciales podrían ser:

```
adyacente( a, b).
adyacente( b, c).
adyacente( c, d).
localizado( robot, a).
localizado(papel_usado, b).
localizado(papelera, d).
```

Agente Híbrido SPADE

Agente robot limpiador

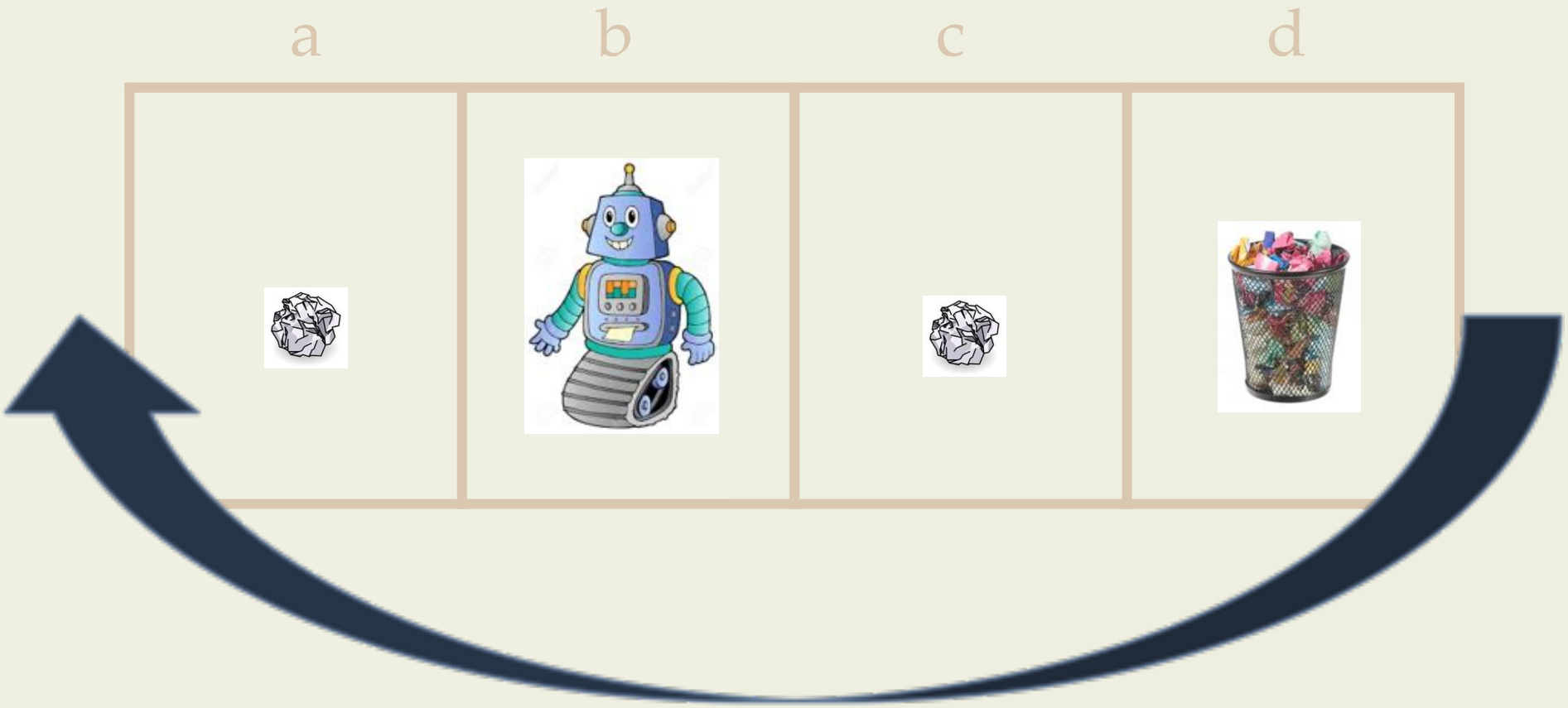
Objetivo inicial: !localizado(robot, d).

Planes:

```
+localizado(robot, X) : localizado(papel_usado, X)
                    <- .print("papel cogido en ", X);
                        -localizado(papel_usado, X); // simula el coger papel
                        +llevando(papel_usado).  //simula el llevar papel

+localizado(robot, X) : localizado(papelera, X) & llevando(papel_usado)
                    <- -llevando(papel_usado);
                        .print("todo el papel tirado a la papelera en ", X).  //simula tirar a papelera

+!localizado(robot, X) : localizado(robot, X) <- .print("Ha llegado a su destino").

+!localizado(robot, X) : localizado(robot, Y) & (not (X=Y)) & adyacente(Y,Z)
                    <- .print("mover de ", Y, " a ", Z);
                    -localizado(robot, Y);
                    +localizado(robot, Z);  // simula el mover de Y a Z
                     !localizado(robot, X).
```

Agente Híbrido SPADE

Agente robot limpiador 2



Entorno circular, con esta nueva situación inicial…