

Sistemas Multiagente

Tema 3: Herramientas y Plataformas de agentes

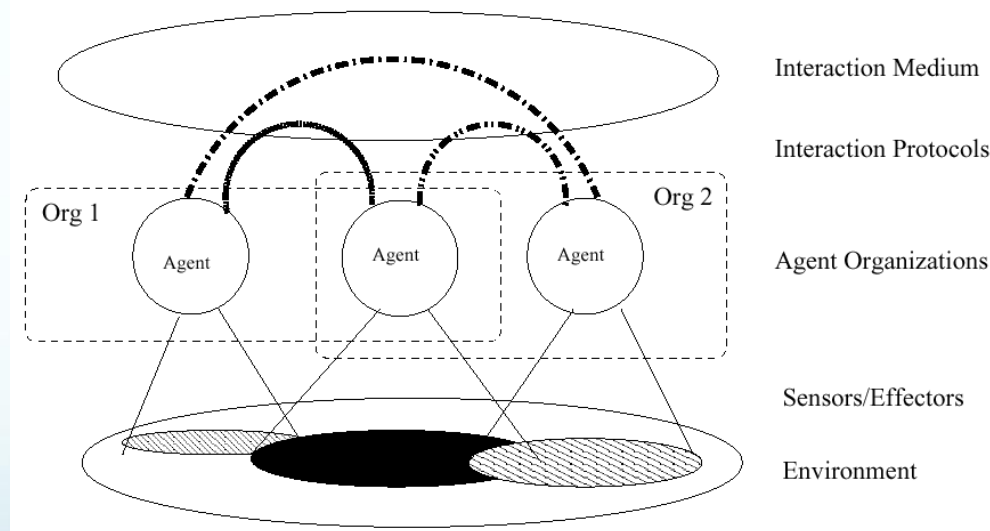
Authors: Vicent Botti, Vicente Julián, Javier Palanca

Conceptos. Introducción

- Problema a resolver:
 - Modelar sistemas reales complejos y con características claramente distribuidas
- Visión de un sistema como una organización computacional consistente de varios “roles” interactuando.
- Diseño (Métodos) → Implementación (Lenguajes) → Implantación (Middleware)

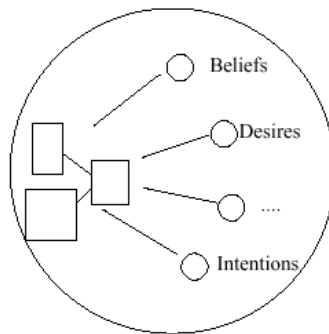
Conceptos. Introducción

- Identificar:
 - los diferentes subsistemas que forman parte del sistema global
 - las posibles interacciones y dependencias entre ellos



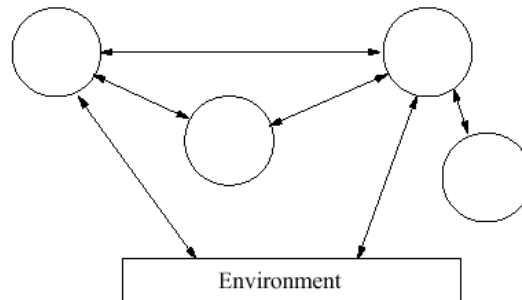
Conceptos. Introducción

- A tener en cuenta:
 - punto de vista interno → un agente
 - punto de vista externo → varios agentes



The Intra-Agent Viewpoint

(agent's own purpose, internal structure, technology)



The Inter-Agent Viewpoint

(interaction with the environment and with other agents)

La necesidad de utilizar una metodología

Desarrollar SMA es complejo

Hay soluciones tecnológicas:

- Arquitecturas: subsunción, BDI, capas, ...
- Entornos o Lenguajes de desarrollo: AgentBuilder, JADEL...
- Plataformas de ejecución: JADE, Jack, Jason... SPADE

Pero hace falta responder a varias cuestiones:

- Qué problema tengo que resolver
- Desarrollar muchos agentes sencillos o pocos agentes de elevada complejidad
- Explicar a otros cómo es el SMA
- Trabajar más de una persona: Coordinar el trabajo
- Valorar la calidad del desarrollo
- Qué proceso seguir
- ...

Plataformas y Middleware

¿Cómo ejecutamos un sistema multiagente?

Plataformas de Agentes

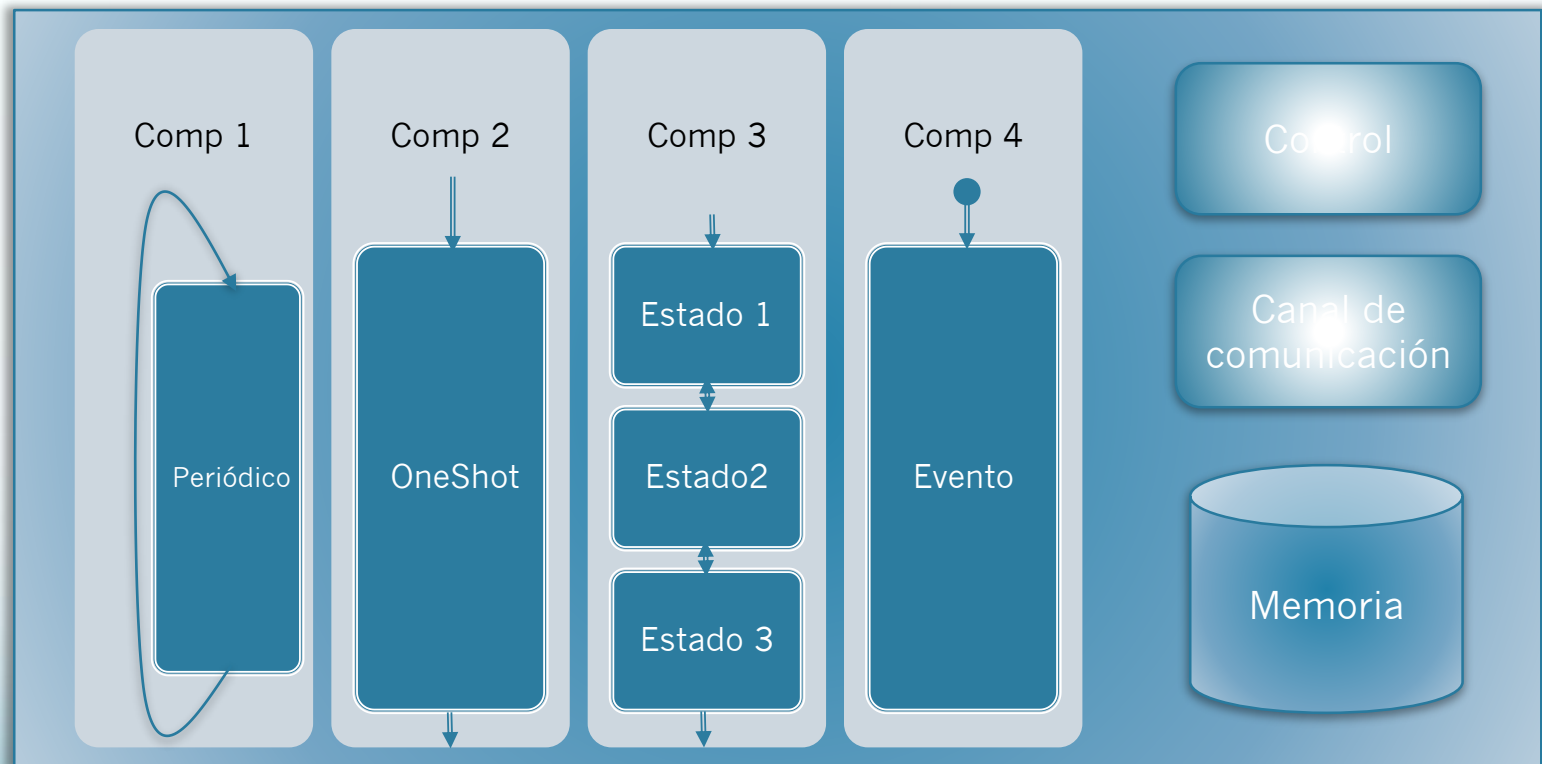
- ¿Qué nos debería ofrecer una plataforma de agentes?
- ¿Qué plataformas hay existentes?

Plataformas

- Funcionalidades:
 - Proporciona un **alto nivel de abstracción** para el desarrollo de determinadas aplicaciones
 - Conjunto de **herramientas** que facilitan el desarrollo, fundamentalmente la **comunicación**
 - **API** adaptada al dominio de las aplicaciones
 - Facilitan la **interoperabilidad**

Plataformas

- Modelo Computacional: Basado en comportamientos



Plataformas existentes

- Muchos *intentos* de realizar la plataforma definitiva
- Focalización únicamente en unos pocos aspectos
 - Mundo de los agentes demasiado amplio para hacer una plataforma “*definitiva*”
 - Cada plataforma enfocada a los temas de investigación de la institución que la desarrolla
- Pocos estándares ampliamente aceptados
 - FIPA (<http://www.fipa.org>)
- Posibilidad de crear sistemas “multiagente” utilizando tecnologías no únicamente de agentes

(algunas) Plataformas

- **Jade** (BELLIFEMINE, F., Caire, G., Poggi, A. & Rimassa, G. (2003). JADE: A white Paper. *EXP in search of innovation*, 3(3), 6–19.)
- **JadeX** (BRAUBACH, L. & Pokahr, A. (2013). The Jadex Project: Simulation. *Multiagent Systems and Applications*, 45, 107–128)
- **JACK** (<https://aosgrp.com.au/jack/>) **CoJACK** (<https://aosgrp.com.au/cojack/>) y **C-BDI** (<https://aosgrp.com.au/c-bdi/>)
- **Jason** (BORDINI, R., Hübner, J. & Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley Series in Agent Technology)
- **PADE** (<https://github.com/grei-ufc/pade>)
- **Spade** (<https://pypi.org/project/SPADE/>)



Para simulación: **NetLogo**, **Repast**, **AnyLogic**, **GAMA**, **MATSim**, **MESA** ...

(algunas) Plataformas existentes

Overview of Software Agent Platforms Available in 2023

Review

Overview of Software Agent Platforms Available in 2023

Zofia Wrona ¹, Wojciech Buchwald ¹, Maria Ganzha ¹, Marcin Paprzycki ^{2,*}, Florin Leon ³, Noman Noor ¹
and Constantin-Valentin Pal ³

¹ Faculty of Mathematics and Information Science, Warsaw University of Technology, 00-662 Warsaw, Poland; zofia.wrona.stud@pw.edu.pl (Z.W.); wojciech.buchwald.stud@pw.edu.pl (W.B.); Maria.Ganzha@pw.edu.pl (M.G.); noman.noor.stud@pw.edu.pl (N.N.)

² Systems Research Institute, Polish Academy of Sciences, 01-447 Warsaw, Poland

³ Faculty of Automatic Control and Computer Engineering, “Gheorghe Asachi” Technical University of Iași, 700050 Iași, Romania; florin.leon@academic.tuiasi.ro (F.L.); constantin-valentin.pal@student.tuiasi.ro (C.-V.P.)

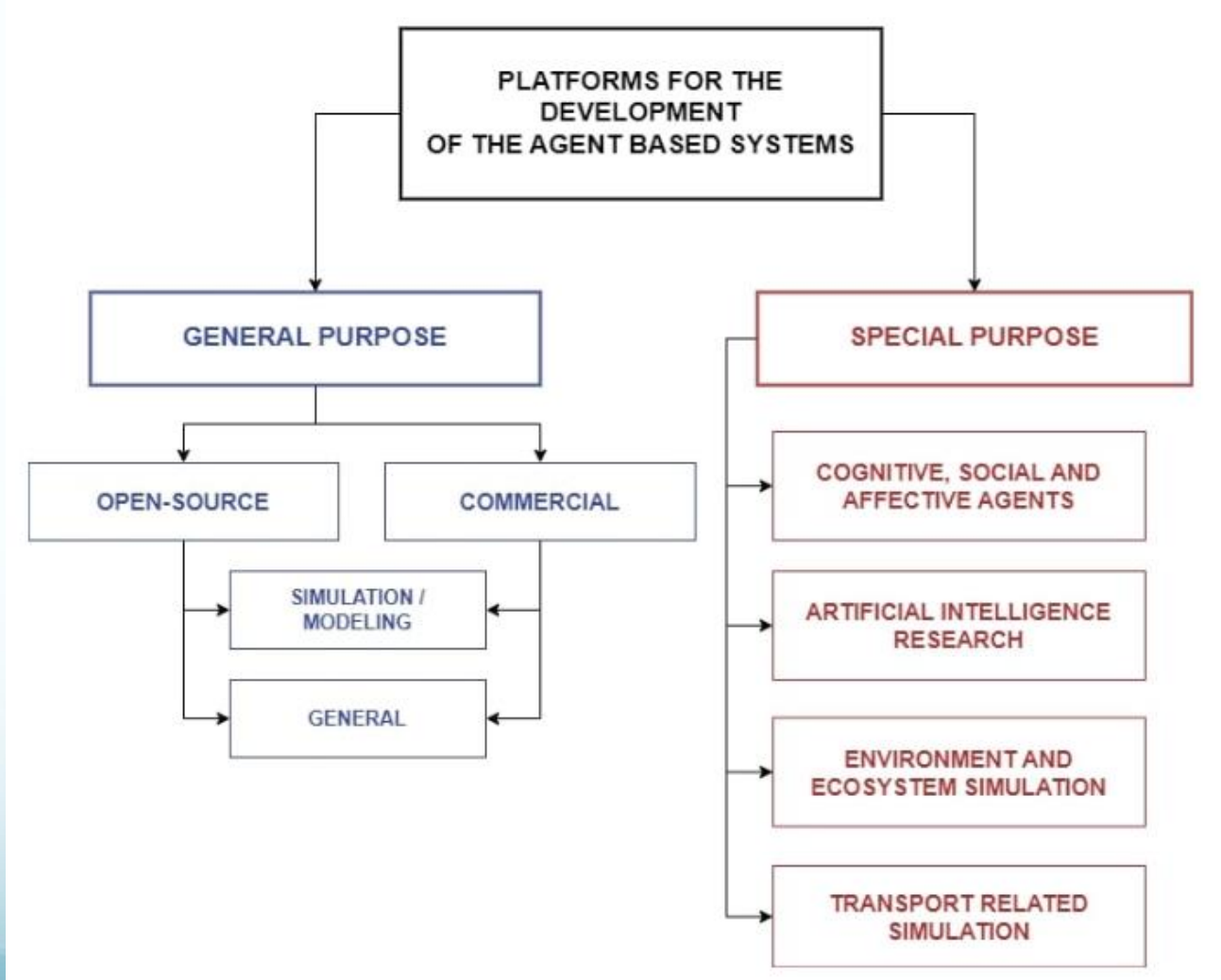
* Correspondence: marcin.paprzycki@ibspan.waw.pl

Abstract: Agent-based computing remains an active field of research with the goal of building (semi-)autonomous software for dynamic ecosystems. Today, this task should be realized using dedicated, specialized frameworks. Over almost 40 years, multiple agent platforms have been developed. While many of them have been “abandoned”, others remain active, and new ones are constantly being released. This contribution presents a historical perspective on the domain and an up-to-date review of the existing agent platforms. It aims to serve as a reference point for anyone interested in developing agent systems. Therefore, the main characteristics of the included agent platforms are summarized, and selected links to projects where they have been used are provided. Furthermore, the described platforms are divided into general-purpose platforms and those targeting specific application domains. The focus of the contribution is on platforms that can be judged as being under active development. Information about “historical platforms” and platforms with an unclear status is included in a dedicated website accompanying this work.

Keywords: agent systems; agent-based systems; multi-agent systems; agent platforms; modeling; simulation; swarm intelligence; artificial intelligence

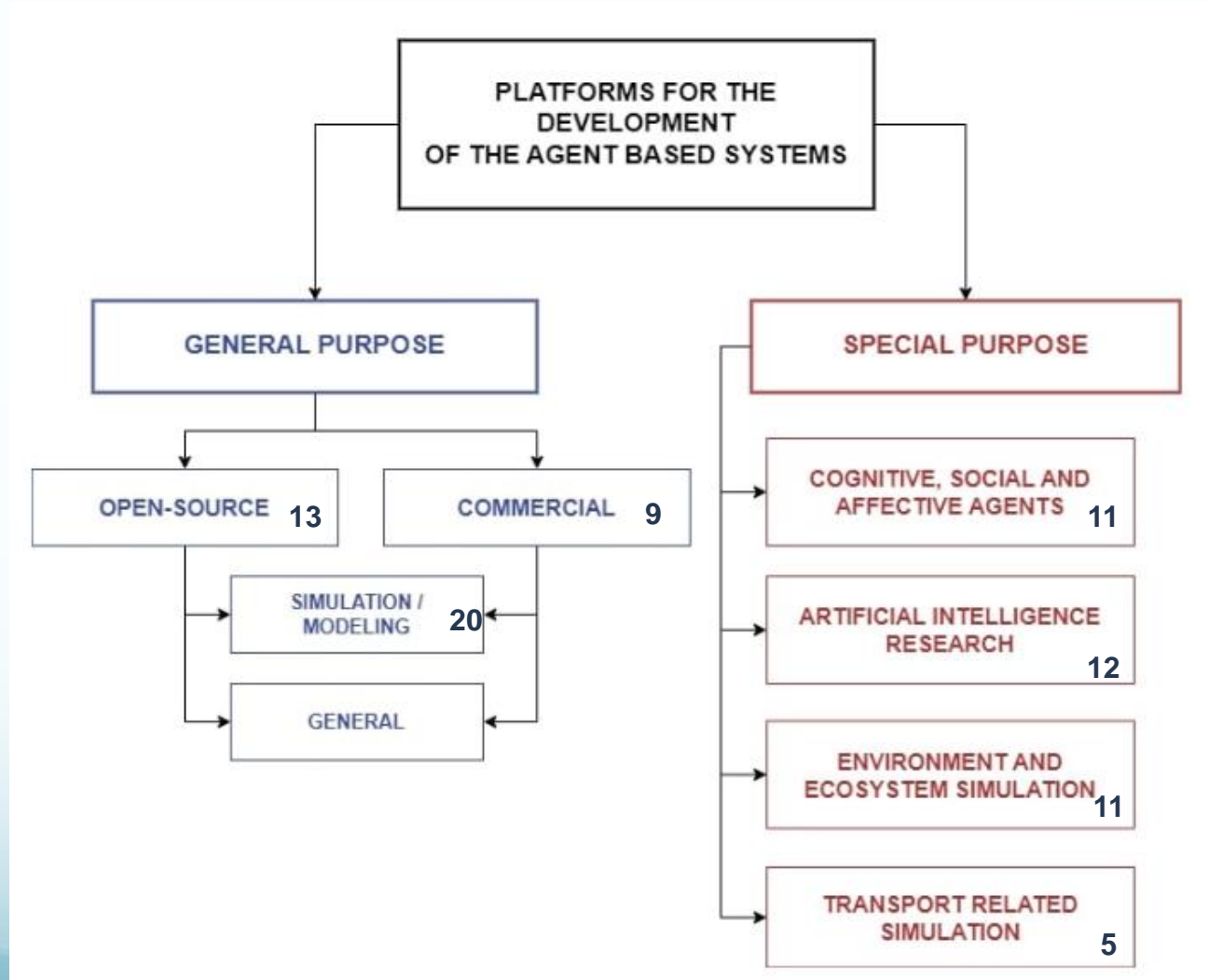
(algunas) Plataformas existentes

Wrona, Z., Buchwald, W., Ganzha, M., Paprzycki, M., Leon, F., Noor, N., & Pal, C. V. (2023). Overview of Software Agent Platforms Available in 2023. *Information*, 14(6), 348.



(algunas) Plataformas existentes

Wrona, Z., Buchwald, W., Ganzha, M., Paprzycki, M., Leon, F., Noor, N., & Pal, C. V. (2023). Overview of Software Agent Platforms Available in 2023. *Information*, 14(6), 348.



¿Cuál era la más usada?

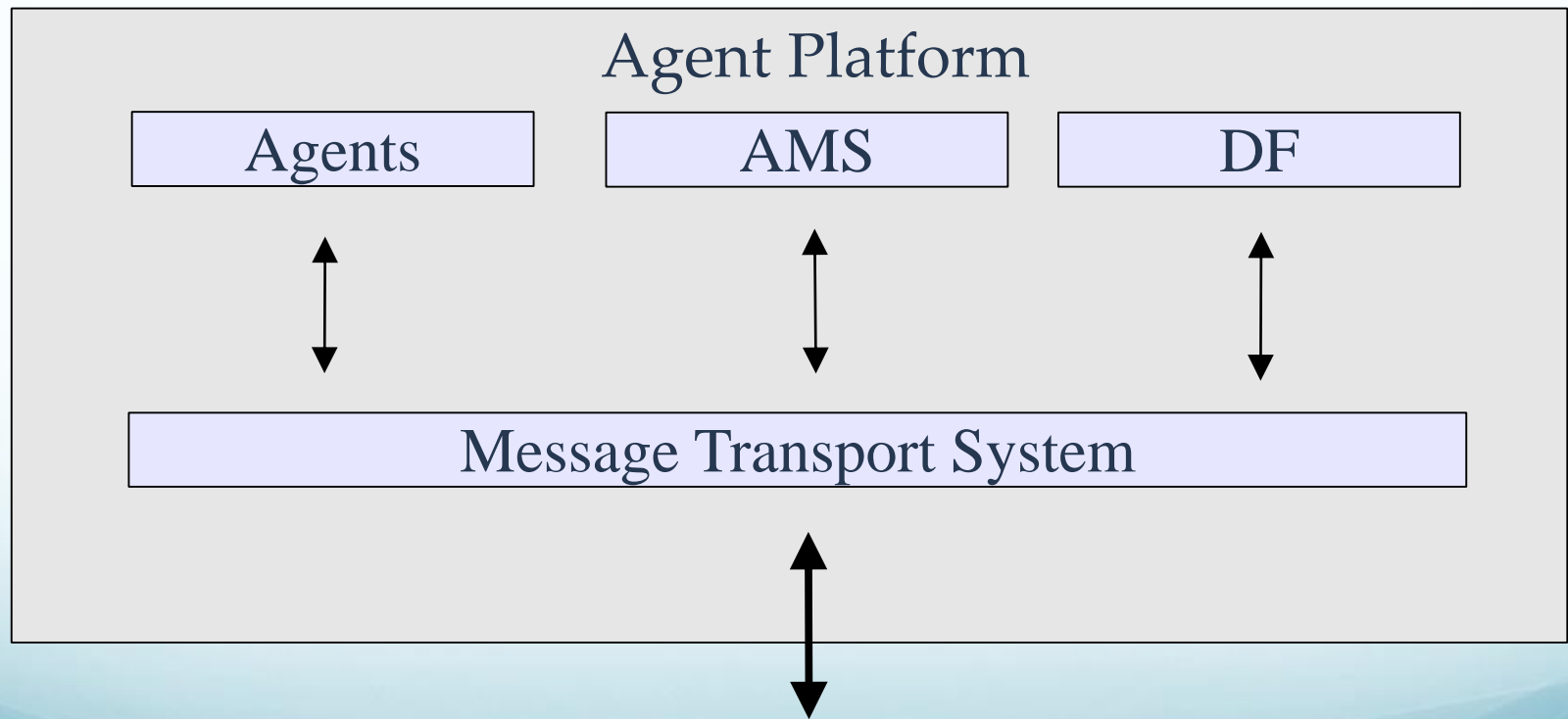


- JADE (Java Agent Development Framework) : Marco de trabajo para el desarrollo de SMA y aplicaciones acorde con el estándar FIPA:
 - Plataforma para agentes (FIPA-compliant).
 - Paquete para el desarrollo de agentes en Java.
- <http://jade.tilab.com> (versión 4.6.0 del 19/12/2022)

JADE: Principales características

- Plataforma distribuida de Agentes.
- Interfaz gráfica.
- Herramientas de depuración.
- Movilidad de los agentes intra-plataforma.
- Soporte para la ejecución de múltiples actividades de los agentes (Behaviours).
- Plataforma FIPA-Compliant.
 - AMS, DF y ACC
- Librería de protocolos de interacción FIPA.
- Registro y desregistro automático de agentes con el AMS.
- Permite usar Ontologías definidas y lenguajes de Contenido.

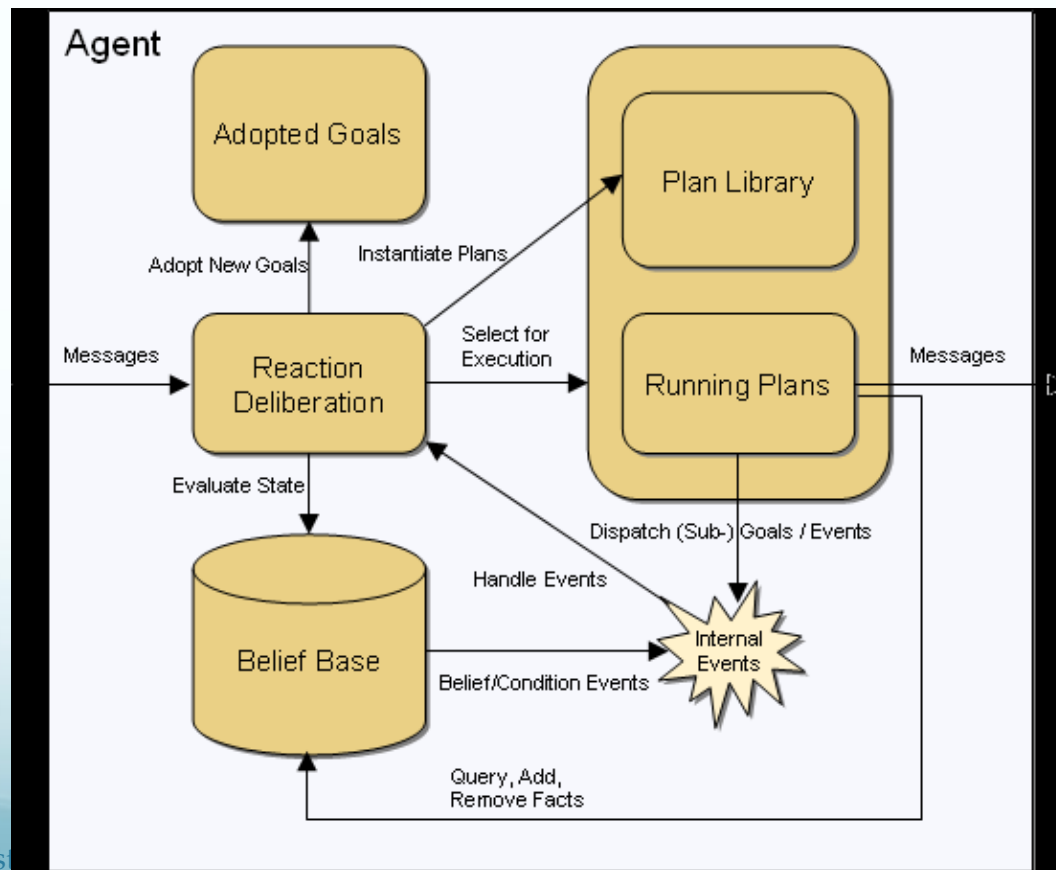
La plataforma JADE



Plataformas existentes: JADEX

- JADE + Diseño BDI de los agentes:

- <https://www.activecomponents.org/>
- <https://github.com/actoron/jadex>



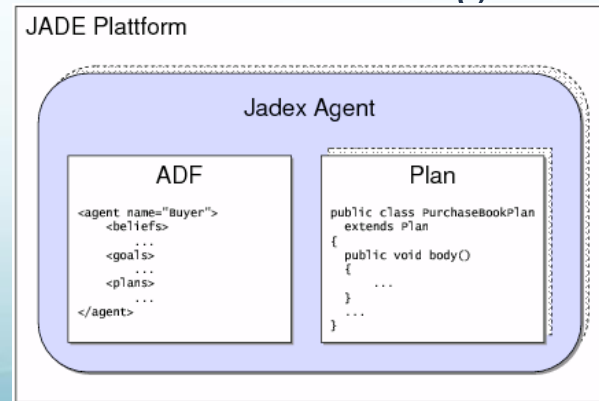
JADEx

Extensión de Jade

Ultima versión: 15/6/2021 Jadex 4.0.241 Release

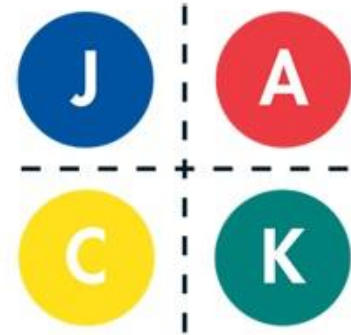
Los agentes Jadex:

- Se definen con dos elementos: una descripción en xml y clases java que representan sus planes.
- La implementación en java de cada plan accede y modifica las creencias, satisface/abandona objetivos y adopta nuevos objetivos.
- En el fichero xml incluimos información sobre: creencias, deseos (goals), planes, mensajes y estados iniciales del agente.



Plataformas existentes: Jack

- Jack está desarrollado por la empresa **AOS Group**
- No es de libre distribución
- Desarrollada en Java
- Basada en BDI
- <https://aosgrp.com.au/jack/>
- version 5.6
- Recientemente hay nuevas versiones
 - CoJACK: para modelar el comportamiento humano. Se utiliza en sistemas de simulación para apoyar a actores virtuales.
 - C-BDI: apropiada para sistemas embebidos. Centrado en sistemas aeroespaciales y de defensa autónomos y semiautónomos en misión crítica. Diseñado para su uso en dominios en tiempo real
 - Jack Teams



Plataformas existentes: Jason

- Libre distribución (GNU LGPL)
- Desarrollada en Java
- Basada en BDI: Uso de AgentSpeak
- <https://jason-lang.github.io>
- Última versión 3.2.0 (15/04/2023)



- Principales características
 - Jason es un intérprete de una versión extendida de AgentSpeak.
 - Implementa la semántica operacional de ese lenguaje.
 - Provee una plataforma para el desarrollo de sistemas multi-agente

Plataformas existentes: Janus (SARL)

- Plataforma de código abierto totalmente implementada en Java
- Maneja de forma nativa el concepto de agentes recursivos y holones.
- Soporta el Lenguaje de Programación Orientado a Agentes SARL.
- Ofrece: concurrencia, distribución, interacción, descentralización, reactividad, autonomía y reconfiguración dinámica
- SARL facilita la ejecución de agentes y la comunicación directa.
- Se proporciona una extensión para soportar los entornos simulados.
- <http://www.sarl.io> Última versión V3 0.13.0 (19/09/2023)



Plataformas existentes:

Entornos de simulación

- Las veremos en el último tema
- Enfocadas a sistemas a gran escala con agentes ligeros
- Algunos ejemplos:
 - Netlogo - <http://ccl.northwestern.edu/netlogo/>
 - Anylogic - <https://www.anylogic.com>
 - MATSim - <https://www.matsim.org>
 - Gama - <https://gama-platform.github.io>
 - Repast - <https://repast.github.io>
 - MESA - <https://mesa.readthedocs.io/en/stable/>

Otras opciones ... SPADE

- Plataforma desarrollada en Python
- Y que vamos a ver en detalle ...

SPADE

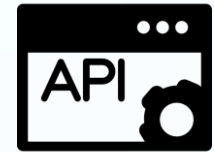
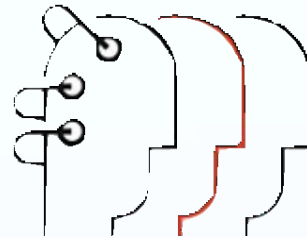
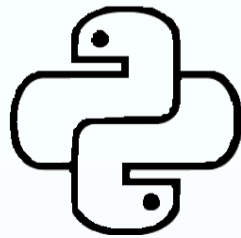
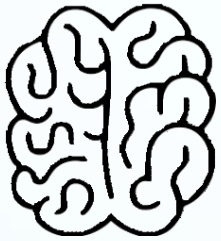
SMART PYTHON AGENT
DEVELOPMENT ENVIRONMENT

3.X



SPADE

Smart Python Agent Dev Env



Basado en:

Python ≥ 3.8

JABBER / XMPP

AsyncIO





¿Qué es Jabber / XMPP?

- ★ Jabber es un protocolo abierto basado en XML para mensajería instantánea y notificación de presencia.
- ★ Proyecto iniciado en 1998 por Jeremie Miller.
 - ★ Primera release en 2000.
- ★ Estandarizado por la IETF y el W3C para la mensajería instantánea a través de Internet.
- ★ Ahora se llama XMPP por **eXtensible Messaging and Presence Protocol**



¿Cómo es XMPP?

Abierto, público y gratuito

Estándar

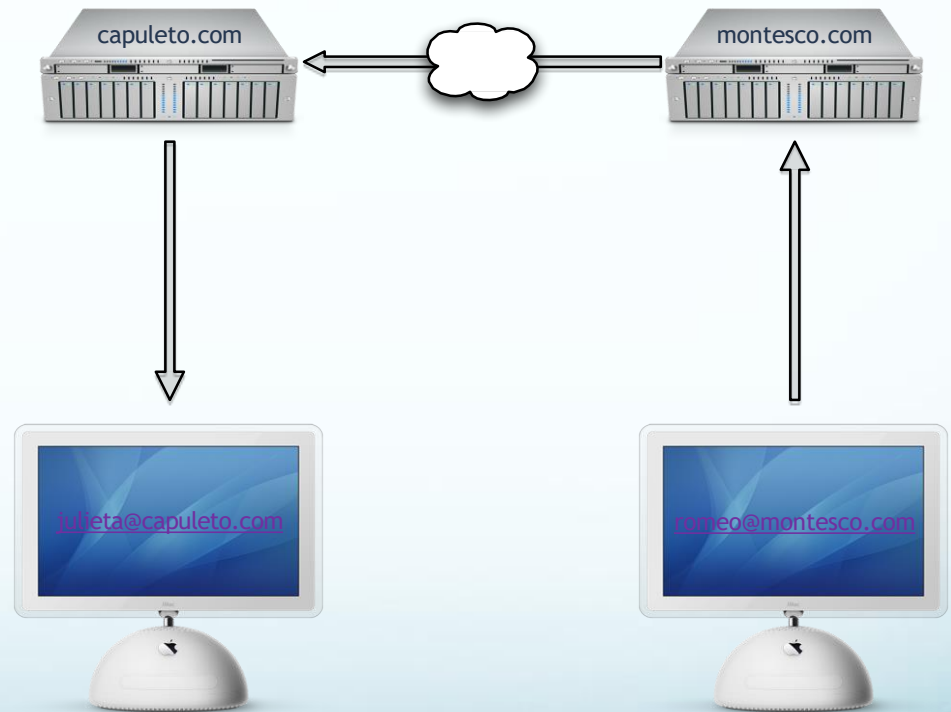
Testado

Descentralizado

Seguro

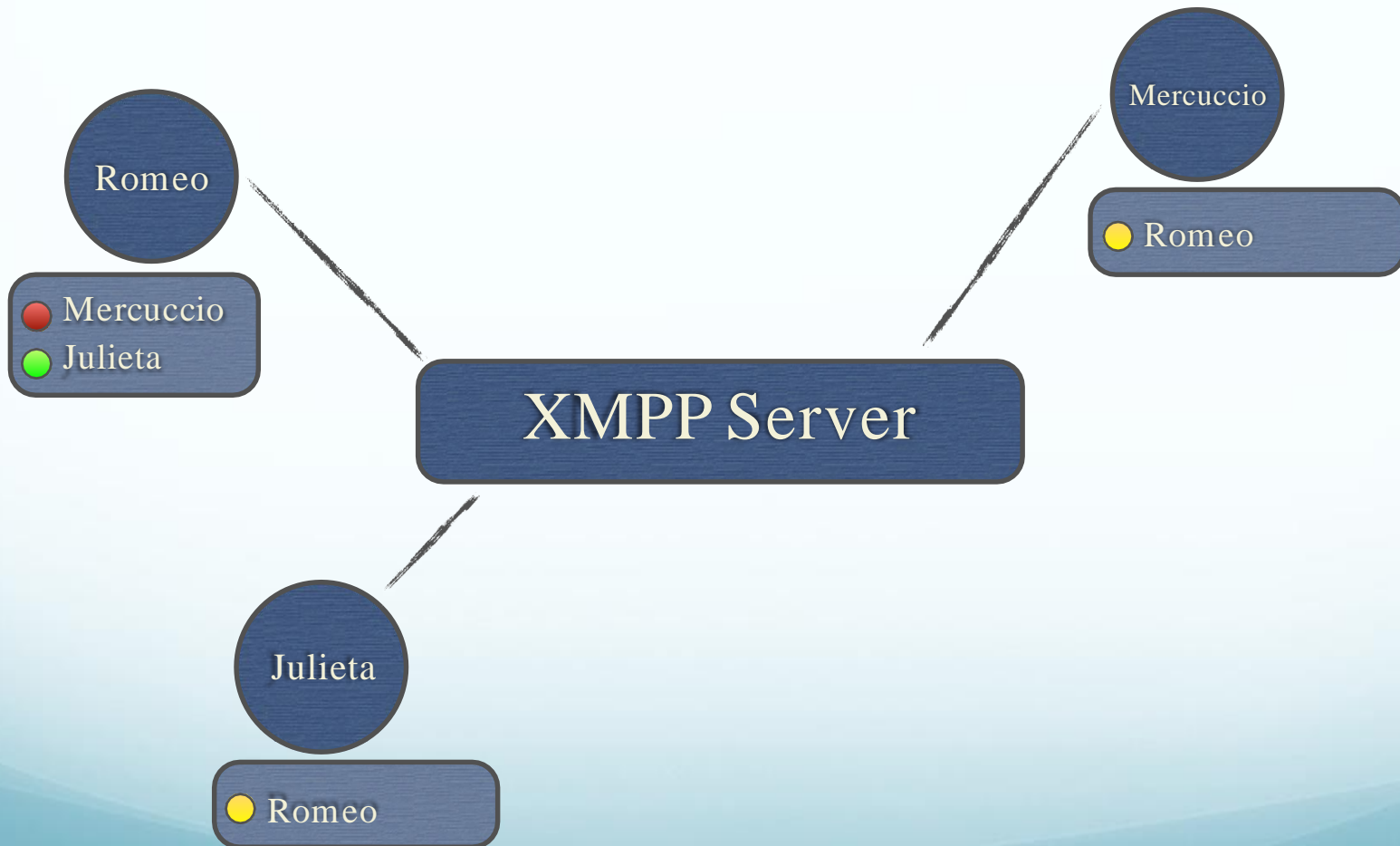
Extensible

Flexible



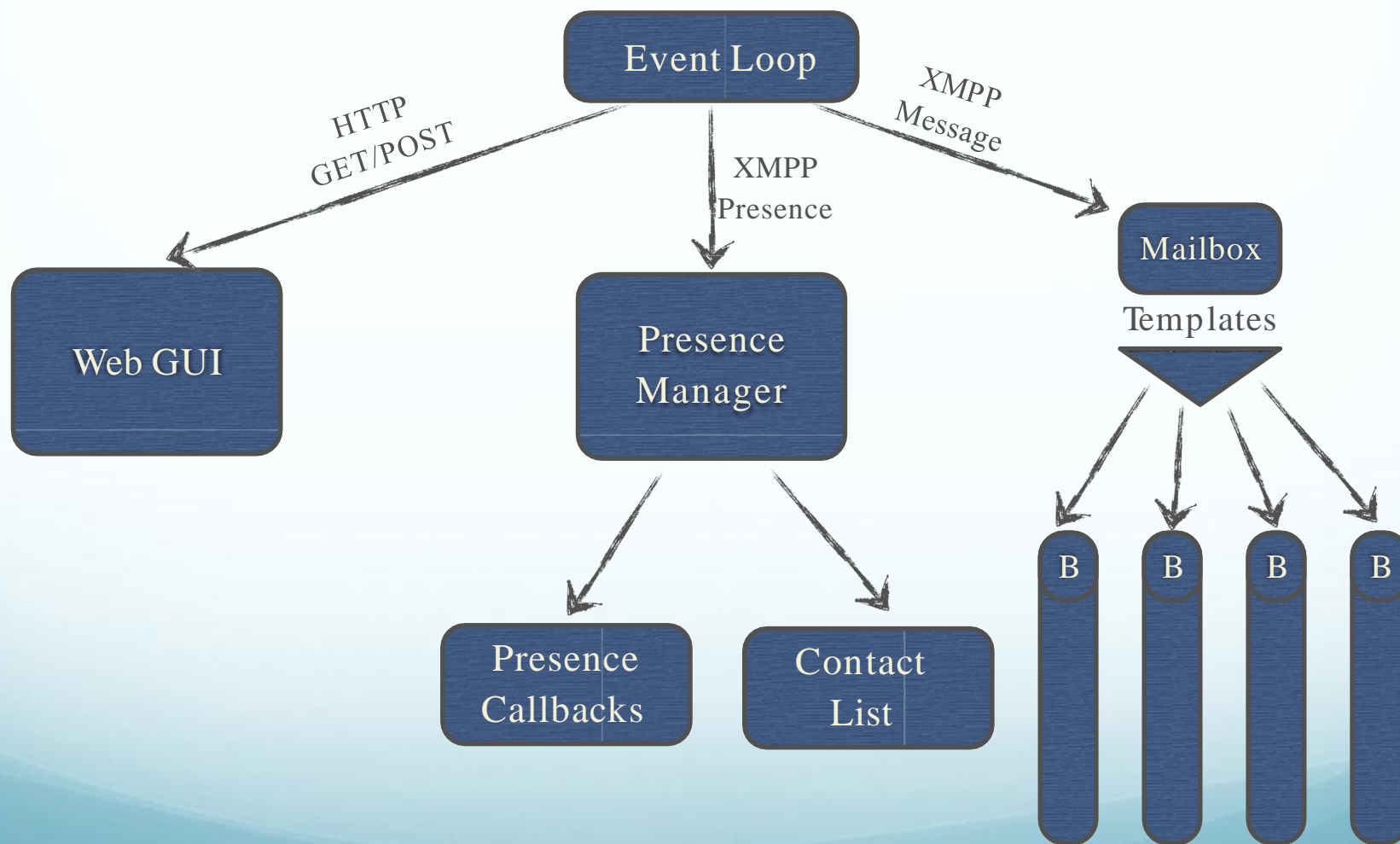


Notificación de Presencia





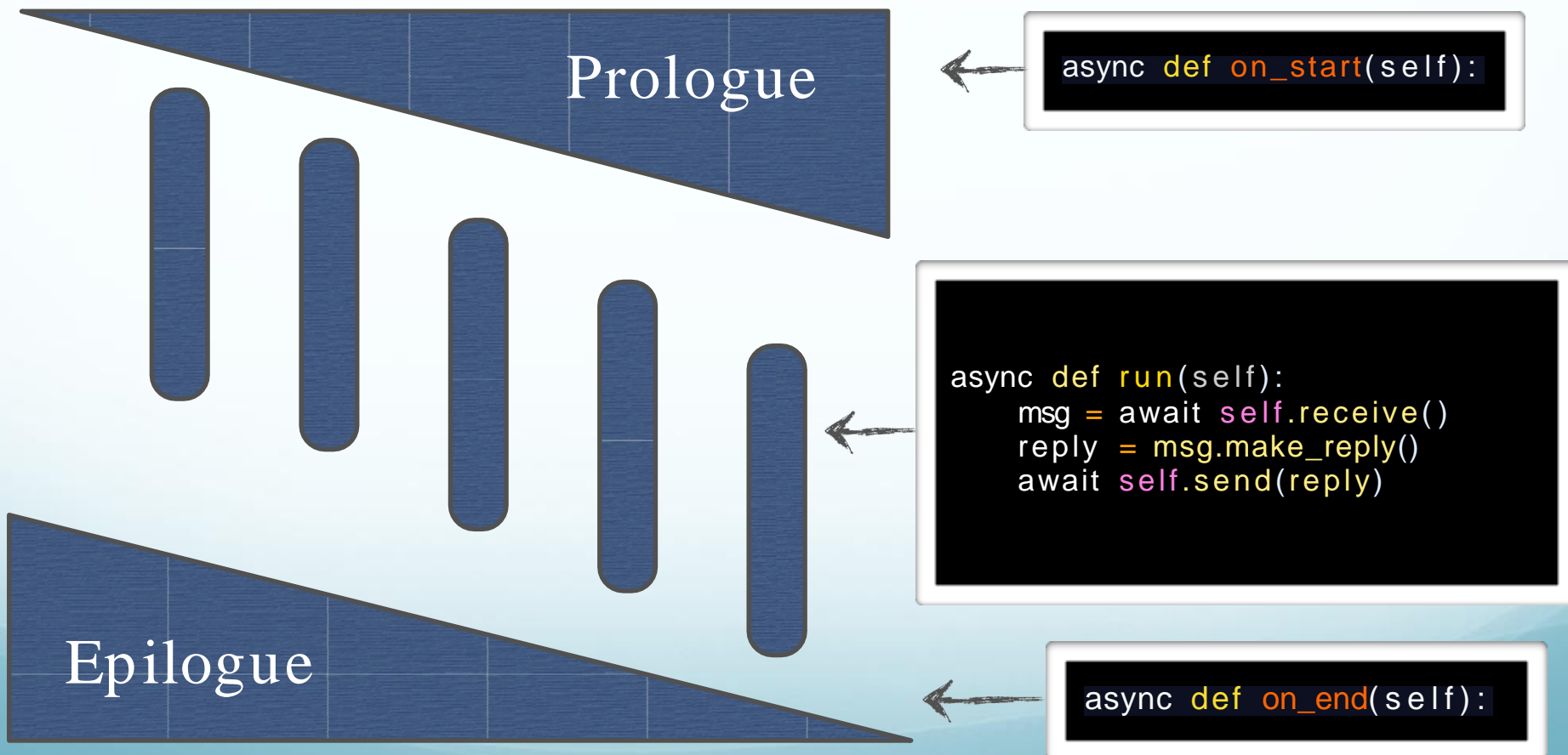
Arquitectura de Agente





Ciclo de vida

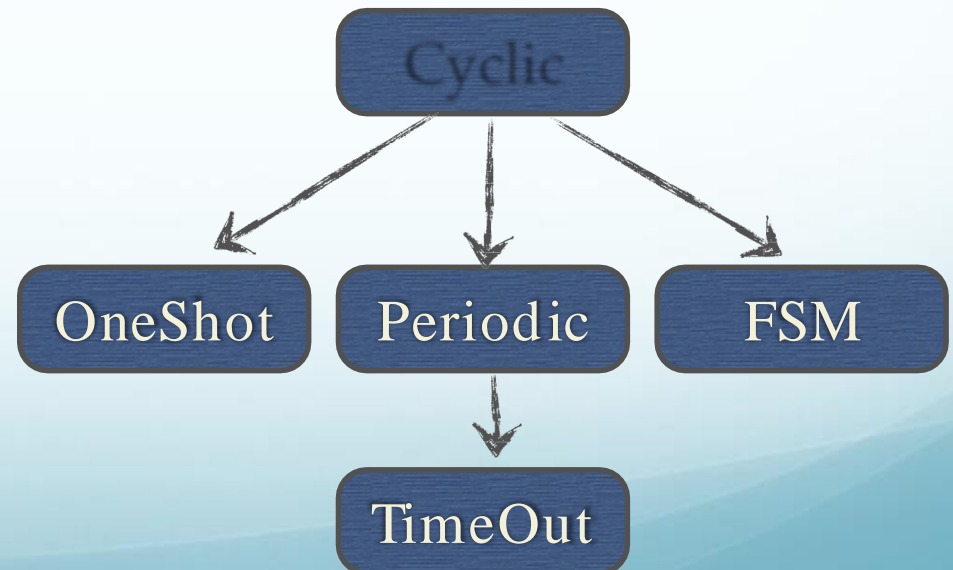
Basado en Async Behaviours





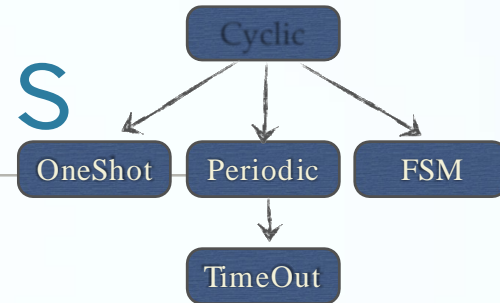
Comportamientos

- **Comportamientos asíncronos**
- **Cada comportamiento tiene su propio buzón**
- **Plantillas de mensajes**
- **Protocolos de interacción con Templates**





Comportamientos




- **CyclicBehaviour**: comportamiento que se ejecuta en un ciclo infinito. Es útil para tareas que deben realizarse continuamente, como escuchar mensajes entrantes.
- **OneShotBehaviour**: comportamiento que se ejecuta solo una vez. Es ideal para tareas que solo necesitan realizarse una vez.
- **PeriodicBehaviour**: similar a CyclicBehaviour, pero se ejecuta a intervalos regulares. Es útil para tareas periódicas, como el monitoreo o la actualización de estados.
- **FSMBehaviour (Finite State Machine Behaviour)**: comportamiento que permite definir una máquina de estados finitos. Cada estado es un comportamiento en sí mismo, y el agente puede transitar entre estados según las condiciones definidas.
- **TimeoutBehaviour**: comportamiento que se ejecuta hasta que se alcanza un tiempo límite. Es útil para tareas que deben completarse en un período específico.

Interfaz de usuario



SPADE



your_jid

Online

Dashboard

Dashboard

Behaviours

CyclicBehaviour/DummyBehav
Template: <template to="None" from="agent0@fake_server" thread="..."

Kill

PeriodicBehaviour/DummyPeriodBehav
Template: <template to="None" from="agent1@fake_server" thread="..."


Kill

TimeoutBehaviour/DummyTimeoutBehav
Template: <template to="None" from="agent2@fake_server" thread="..."


Kill

FSMBehaviour/DummyFSMBehav
Template: <template to="None" from="agent3@fake_server" thread="None" ...


Contacts




agent0@fake_se...
ONLINE




agent1@fake_se...
AWAY




agent2@fake_se...
DND



agent4@fake_se...
ONLINE



agent3@fake_se...
OFFLINE



agent5@fake_se...
OFFLINE


Copyright © 2018 SPADE.

Version 3.0.0

Interfaz de usuario



SPADE

**your_jid**
Online


Dashboard

FSMBehaviour/DummyFSMBehav

Home > Dashboard


4

Mailbox




True

Is killed?




S 1

Current State




Template

```
<template to="None" from="agent3@fake_server" thread="None" metadata={}></template>
```



0

Exit Code




Chat

8 — x

your_jid


11 minutes ago

This is my answer.



agent3


11 minutes ago

 Hello from agent3!
This is a long message.

your_jid

11 minutes ago

This is my answer.



Finite State Machine




```
graph LR; S1((S_1)) --> S2((S_2)); S2 --> S1; S2 --> S4((S_4)); S2 --> S3((S_3)); S4 --> S5((S_5)); S3 --> S5
```

Interfaz de usuario



SPADE




your_jid
● Online

Dashboard

agent0@fake_server

Home > Dashboard

agent0@fake_server




STATUS
● ONLINE

ACTION
Unsubscribe

12
MESSAGES

Chat

12 — ×




Hello from agent0! This is a long message.

your_jid


2 minutes ago

This is my answer.



agent0

2 minutes ago




Hello from agent0! This is a long message.

your_jid

2 minutes ago

This is my answer.



Type Message ...

Send

Copyright © 2018 SPADE.

Version 3.0.0

Instalación

<https://spade-mas.readthedocs.io>

- ⦿ Se recomienda **Virtualenvs** (conda, Pipenv, ...)
- ⦿ Usar **Python >= 3.8**
- ⦿ Necesario instalar un servidor **XMPP** o usar uno público

Prosody en Linux

Openfire en Windows

- ⦿ Activar **in-band register** en el servidor XMPP

En Prosody es: **allow_registration=true** en el fichero prosody.cfg.lua

```
$ pip install spade # if using conda
$ pipenv install spade # if using pipenv
```

Instalación

En Linux:

```
# Create virtual environment with python = 3.9
```

```
# Conda or pipenv
```

```
$ conda create -n spade python=3.9
```

```
$ conda activate spade
```

```
# Install spade
```

```
(spade)$ pip install spade
```

Instalación

En Linux (Polilabs):

```
$ source ruta_anaconda
```

```
$ conda init
```

Abrir otro terminal

```
# Create virtual environment with python = 3.9
```

```
# Conda or pipenv
```

```
$ conda create -n spade python=3.9
```

```
$ conda activate spade
```

```
# Install spade
```

```
(spade)$ pip install spade
```

Instalación

Como servidor XMPP usaremos (solo en la red de la UPV) :

gtirouter.dsic.upv.es

CUIDADO!!! Es un servidor compartido

Cada agente debe tener un nombre diferente

Si se quiere instalar un servidor propio:

- **Prosody en Linux**
- **Openfire en Windows**

Mi primer agente

<https://spade-mas.readthedocs.io>

```
from spade import agent

class DummyAgent(agent.Agent):
    async def setup(self):
        print("Hello World! I'm agent {}".format(str(self.jid)))

dummy = DummyAgent("your_jid@your_xmpp_server", "your_password")
await dummy.start()

await dummy.stop()
```

```
$ python dummyagent.py
Hello World! I'm agent your_jid@your_xmpp_server
$
```


Mi primer comportamiento

```
import time
from spade.agent import Agent
from spade.behaviour import CyclicBehaviour

class DummyAgent(Agent):
    async def setup(self):
        print("Agent starting . . .")
        b = self.MyBehav()
        self.add_behaviour(b)

    class MyBehav(CyclicBehaviour):
        async def on_start(self):
            print("Starting behaviour . . .")
            self.counter = 0

        async def run(self):
            print("Counter: {}".format(self.counter))
            self.counter += 1
            await asyncio.sleep(1)

async def main():
    dummy = DummyAgent("your_jid@your_xmpp_server", "your_password")
    await dummy.start()
    print("DummyAgent started. Check its console to see the output.")

    print("Wait until user interrupts with ctrl+C")
    await wait_until_finished(dummy)

if __name__ == "__main__":
    spade.run(main())
```

```
$ python dummyagent.py
Agent starting...
Starting behaviour...
Counter: 0
Counter: 1
Counter: 2
Counter: 3
Counter: 4
Counter: 5
Counter: 6
Counter: 7
```

Sender & Receiver

```
from spade.agent import Agent
from spade.behaviour import OneShotBehaviour
from spade.message import Message

class SenderAgent(Agent):
    class InformBehav(OneShotBehaviour):
        async def run(self):
            print("InformBehav running")
            msg = Message(to="receiver@xmpp_server")
            msg.set_metadata("performative", "inform")
            msg.body = "Hello World"

            await self.send(msg)
            print("Message sent!")

            # stop agent from behaviour
            await self.agent.stop()

    async def setup(self):
        print("SenderAgent started")
        self.b = self.InformBehav()
        self.add_behaviour(self.b)
```

```
from spade.agent import Agent
from spade.behaviour import OneShotBehaviour
from spade.template import Template

class ReceiverAgent(Agent):
    class RecvBehav(OneShotBehaviour):
        async def run(self):
            print("RecvBehav running")
            msg = await self.receive(timeout=10)
            if msg:
                print("Message received {}".format(msg.body))
            else:
                print("No message after 10 seconds")

            await self.agent.stop()

    async def setup(self):
        print("ReceiverAgent started")
        b = self.RecvBehav()
        template = Template()
        template.set_metadata("performative", "inform")
        self.add_behaviour(b, template)
```

```
async def main():
    receiveragent = ReceiverAgent("receiver@your_xmpp_server", "receiver_password")
    await receiveragent.start(auto_register=True)
    print("Receiver started")

    senderagent = SenderAgent("sender@your_xmpp_server", "sender_password")
    await senderagent.start(auto_register=True)
    print("Sender started")














    await spade.wait_until_finished(receiveragent)
    print("Agents finished")

if __name__ == "__main__":
    spade.run(main())
```



Más ejemplos de SPADE

- En <https://github.com/javipalanca/spade/tree/master/examples>

..	
 agent_inside_agent.py	Fixed documentation. Added Github CI.
 counter_behav.py	- Fixed bug comparing JIDs in trace _agent_in_msg function.
 dummyagent.py	Fixed documentation. Added Github CI.
 dummybehavior.py	Fixed documentation. Added Github CI.
 fsm_example.py	Fixed documentation. Added Github CI.
 join.py	- Fixed bug comparing JIDs in trace _agent_in_msg function.
 kill_behavior.py	Fixed documentation. Added Github CI.
 performance.py	Merge conflict.
 periodic_example.py	Fixed documentation. Added Github CI.
 presence_subscribe_friends.py	blackstyled code.
 send_and_rcv.py	Merge conflict.
 timeout_example.py	Fixed documentation. Added Github CI.
 web_interface.py	Merge conflict.



SPADE

- **Free Software** project (LGPL)
- Abierto a colaboraciones (**PULL REQUEST!**)
- Contacto:
- **web:** <http://github.com/javipalanca/spade>
- **doc:** <https://spade-mas.readthedocs.io/>

spade

Summary

PyPI link
<https://pypi.org/project/spade>

Total downloads
137,863

Total downloads - 30 days
2,719

Total downloads - 7 days
763

Badges

downloads 138k

[[Downloads]



downloads/month 3k

[[Downloads]



downloads/week 763

[[Downloads]



Conclusiones

- La tecnología de middleware (plataforma) se ha convertido en algo necesario para desarrollar sistemas multiagente
- El mundo de los sistemas multiagente es muy amplio y no existen plataformas “universales”, foco en distintos aspectos:
 - Diseño interno del agente
 - Comunicaciones
 - Normas
 - Organizaciones
 - Negociación
 - Argumentación
 - Confianza y reputación
 - ...

Conclusiones

- Elección de plataforma
 - ¿Existe alguna que se adapte? El dominio importa
 - IOT
 - Cyber Physical Systems
 - Industria 4.0
 - Nivel de abstracción muy alto
 - Cada plataforma implementa los sistemas multiagente de una manera particular
 - Utilización de muchas tecnologías
 - Puesta en funcionamiento