



# Prácticas AIN

---

## pyGOMAS

### Práctica 2: Comunicación y Coordinación

---



# Índice

---

- ❖ Registro de Servicios
  - ❖ Comunicación y Coordinación
  - ❖ Trabajo a realizar
-

# Recordatorio

---

- ✧ Hay definidos tres tipos de roles en los **agentes externos**:
    - ✧ *Soldier*: soldado de tipo general
      - ✧ ALLIED: va a por la bandera y vuelve a la base
      - ✧ AXIS: patrulla alrededor de la bandera
    - ✧ *Medic*: acude a curar
    - ✧ *FieldOps*: acude a dar munición
  - ✧ Un agente asume un único rol durante toda la partida
  - ✧ Cada rol tiene unas características y ofrece unos determinados **servicios**
-

# Registro de servicios (I)

---

- ✧ Un rol debe registrar un **servicio** para que el resto de roles puedan solicitarlo:

*.register\_service("servicio\_a")*

Envia un mensaje al agente de servicio para registrar un servicio denominado "servicio\_a" que estará disponible para su equipo.

- ✧ Ej:

*.register\_service("general");*

registra el servicio "general" para su equipo.

---

# Registro de servicios (II)

---

Registros que se hacen por defecto en todos los agentes:

## ✧ALLIED

*.register\_service( "allied");*

Soldado: *.register\_service( "backup");*

Médico: *.register\_service( "medic");*

Fieldops: *.register\_service( "fieldops");*

## ✧AXIS

*.register\_service( "axis");*

Soldado: *.register\_service( "backup");*

Médico: *.register\_service( "medic");*

Fieldops: *.register\_service( "fieldops");*

---

# Registro de servicios (IV)

---

❖ ¿Cómo saber que servicios hay disponibles desde un agente?

*.get\_medics*: Solicita al agente de servicios la lista de los médicos de su equipo.  
Con la respuesta se crea una creencia: *myMedics(Medics\_list)*

*.get\_fieldops*: Solicita al agente de servicios la lista de los operadores de campo de su equipo. Con la respuesta se crea una creencia: *myFieldops(Fieldops\_list)*

---



# Registro de servicios (IV)

---

❖ ¿Cómo saber que servicios hay disponibles desde un agente?

*.get\_backups*: Solicita al Agente de Servicios los soldados de su equipo.

*.get\_service("servicio\_a")*: Solicita al Agente de Servicios otro servicio (distinto de los tres anteriores) a los agentes tropa de su equipo que lo ofrezcan.

La respuesta llega en forma de nueva creencia *servicio\_a(L)*

+*servicio\_a(L)*

<-

*.print("Los agentes de mi equipo con el servicio\_a son: ", L).*

---

# Registro de servicios (V)

---

- ❖ NOTA: Todas las acciones `.get` siempre excluyen al propio agente que hace la solicitud de la lista que devuelven.

- ❖ Ejemplo de uso

Desde un plan se ejecuta: *`.get_medics`*;

Existe otro plan de la forma:

*`+myMedics(M)`*

*`<- .println("Mis médicos disponibles son: ", M);`*

*`.length(M, X);`*

*`if (X==0){ .println("No tengo médicos"); }`*

Nota: si el agente que ejecuta este código fuese médico no aparecería en la lista M

---



# Registro de servicios (V)

---

- ❖ Ejemplo de uso de un servicio nuevo

Un agente A ejecuta: *.register\_service("coronel");*

Otro agente B ejecuta: *.get\_service("coronel");*

B además dispone del siguiente plan:

*+coronel(A)*

*<-*

*.print("Mi coronel es:", A);*

*-coronel(\_).*

---

# Registro de servicios (V)

---

- ❖ Ejemplo de uso de un servicio nuevo

Si el coronel ha muerto la lista estará vacía.

Una alternativa es que B ejecute lo siguiente:

```
.get_service("coronel");
```

```
.wait(2000); // un tiempo prudencial
```

```
if (coronel(A)) { .print("Mi coronel es:", A); -coronel(_); } .
```

---



# Coordinación (I)

---

- ❖ pyGOMAS dispone de mecanismos que permiten la coordinación entre agentes:
    - ❖ Sin comunicación (implícita):
      - ❖ Sensorización del entorno (ya visto en la práctica 1)
    - ❖ Con comunicación (explícita):
      - ❖ Mediante paso de mensajes
-

# Coordinación (II)

---

- ✧ Con comunicación
  - ✧ Envío de mensajes mediante la acción interna

*.send(Rec, Perf, Cont)*

Donde:

Rec → receptor del mensaje (puede ser una lista)

Perf → performativa (tell, untell, achieve, ...)

Cont → contenido

---

# Coordinación (III)

---

Ej: A1 quiere enviar un mensaje a los soldados médicos de su equipo diciendo que vayan a su posición (para ayudar, para coordinarse, para reagruparse, ...)

...

*?position(Pos);*

*?myMedics(M); // se supone que antes he ejecutado .get\_medics*

*.send(M, tell, ir\_a(Pos));*

---

# Coordinación (IV)

---

Ej: los médicos del equipo deberían disponer de un plan de la forma:

```
+ir_a(Pos)[source(A)]
```

```
<-
```

```
.println("Recibido un mensaje de tipo ir_a de ", A, "para ir a: ", Pos).
```

---



# Coordinación (V)

---

Ej: Si queremos que los soldados hagan algo más sofisticado

*+ir\_a(Pos)[source(A)]*

*<-*

*.println("Recibido mensaje ir\_a de: ", A, " para ir a: ", Pos);*

*+ayudando;*

*.goto(Pos).*

Mejoras:

- Comprobar si A tiene autoridad sobre el soldado
  - Hacer caso sólo si tengo salud, armamento o las dos cosas
  - Revisar antes otras tareas pendientes
-

# Coordinación (VI)

---

- ❖ Estrategias vistas (o por ver) en clase:
    - ❖ Organización jerárquica: El jefe manda !!!
    - ❖ Contract Net: Delegación de tareas
    - ❖ Social Choice: Votamos !!!
    - ❖ Subastas: quien me ofrece algo mejor !!!
    - ❖ ...
-

# Trabajo a Realizar (próximas sesiones)

---

- ❖ **Objetivos:**

- ❖ Diseñar e implementar **un equipo de 10 agentes** con la distribución de tipos que deseéis (médicos, soldados y fieldops) para jugar a **capturar la bandera** en un mapa cualquiera como **atacante o como defensor**.
  - ❖ Es **necesario** realizar trabajo en los siguientes aspectos:
    - 1.Coordinación vía paso de mensajes** entre agentes del mismo equipo.
    - 2.Servicios nuevos:** se debe incluir algún servicio nuevo por parte de un agente y el uso del mismo por parte de otros agentes.
    - 3.Comportamientos internos de los agentes:** Se deben realizar mejoras de **comportamientos** existentes (por ej. tratar de evitar el fuego amigo).
    - 4.Acción Interna:** se debe incluir al menos una nueva **acción interna** en Python.
-

# Trabajo a Realizar (próximas sesiones)

---

- ❖ ¿Qué os damos?
    - ❖ En Poliformat disponéis de una carpeta "práctica 2" con:
      - ❖ Un fichero json de ejemplo de configuración de la partida con 20 soldados
        - ❖ 10 allied y 10 axis (8 soldados, 1 médico y 1 fieldop)
      - ❖ Tres ficheros asl con la implementación **básica** de un soldado, un médico y un fieldop.
  - ❖ *Nota: la configuración de vuestro equipo es libre*
-

# Trabajo a Realizar (próximas sesiones)

---

## Posibles estrategias

### ✧ ALLIED

- ✧ Elegir un capitán que coordine el ataque del resto
- ✧ Dividir el equipo en dos y atacar por oleadas
- ✧ Coordinar la retirada cuando se tiene la bandera

### ✧ AXIS

- ✧ Elegir un capitán que coordine la defensa
  - ✧ Coordinar a los agentes para patrullar con distintos radios
  - ✧ Añadir algún agente vigía
  - ✧ Identificar que la bandera ha sido capturada y buscarla
-

# Trabajo a Realizar (Normas)

---

- ❖ **Reglas Básicas:**

- ❖ No se puede consultar / solicitar información del sistema sobre el bando contrario que no sea suministrada por el entorno.
  - ❖ No puede existir comunicación entre agentes que no sea usando la acción interna *.send* y de acuerdo a la especificación proporcionada.
  - ❖ La práctica puede hacerse en grupo de dos alumnos.
-



# Trabajo a Realizar (Entrega)

---

- ❖ Entrega:

- ❖ **Ficheros \*.asl y \*.py** desarrollados, así como el fichero json preparado para lanzar a los agentes del equipo.
    - ❖ **IMPORTANTE:** los nombres de vuestros agentes deben incorporar vuestro login para diferenciarlos del resto
  - ❖ El código, comentado y documentado debe seguir unas mínimas normas de estilo: tabulado y comentado.
  - ❖ Comprimir todo el directorio en un fichero <nombre\_equipo>.zip
  - ❖ **Pequeña memoria**, indicando las principales ideas de mejora aplicadas al equipo, así como unas breves conclusiones sobre los resultados obtenidos.
-

# Trabajo a Realizar (Entrega)

---

- ✧ Plazos

- ✧ 24 de mayo (tarea en Poliformat)
  - ✧ Sesiones del 3, 10 y 17 de mayo serán para trabajar en la práctica.
-