Descripción de la ontología de los casos de uso en CAMPERO

J.P. Bandera
Grupo AVISPA, UMA

DSR: Cómo lo usamos

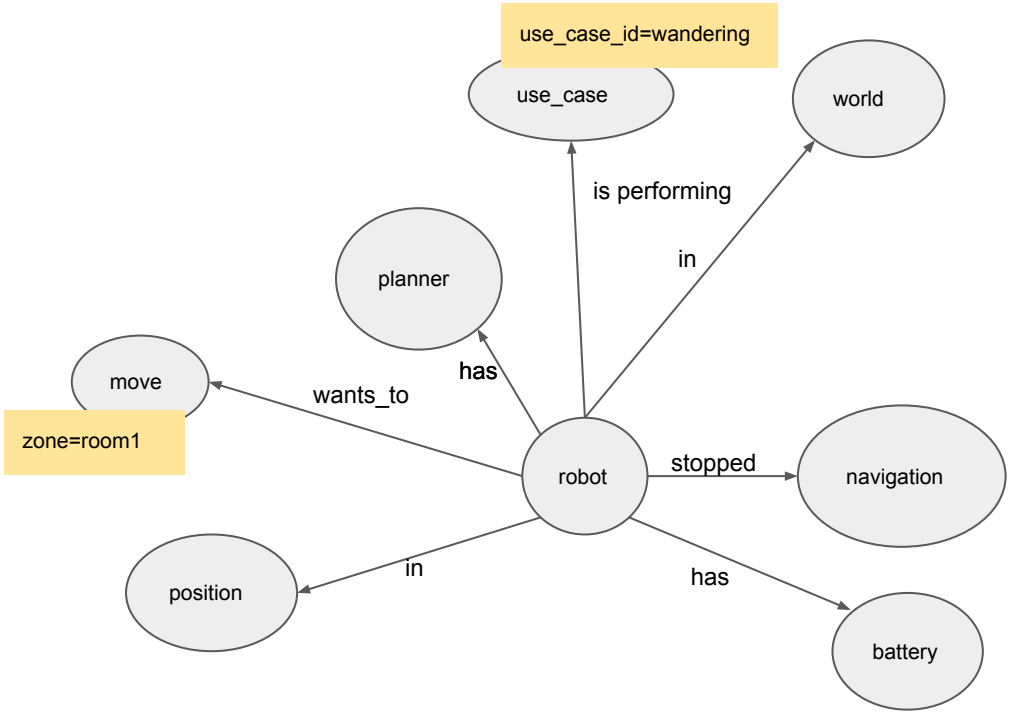
- No voy a explicaros lo que es el DSR… ya lo sabéis.
- En UMA estamos utilizando el DSR como núcleo de la arquitectura cognitiva.
  - No hay comunicaciones entre componentes.
  - Los componentes se enganchan al DSR a través de un agente.
  - Cualquier información que se trasiega, se trasiega a través del DSR.

  - *El DSR es el conocimiento que el robot tiene acerca de su **contexto instantáneo***.
    - Ese conocimiento incluye perfiles de usuario, agendas de actividades, etc… Cuando es pertinente, estos datos son volcados por un agente (bbddAgent) a los nodos correspondientes del DSR.
      - Ejemplo: al reconocer a una persona, en el nodo que la representa se cargan atributos que indican su perfil de usuario, sus preferencias de comunicación, sus opciones de menú y su agenda de actividades para el día.

DSR: Un ejemplo de uso (1/4)

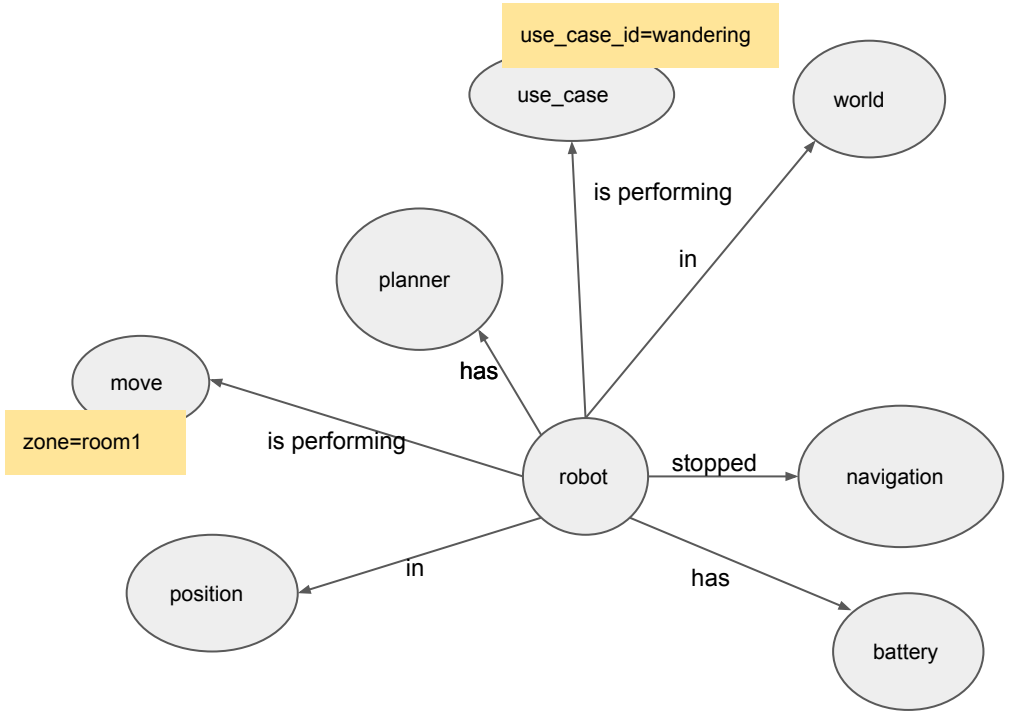
En este ejemplo, el robot está ejecutando el caso de uso “wandering” (vaya, vagabundeando por la sala “room1”).

Concretamente, alguien (un planificador) quiere que se mueva a una posición de esa sala (robot->wants_to->move)



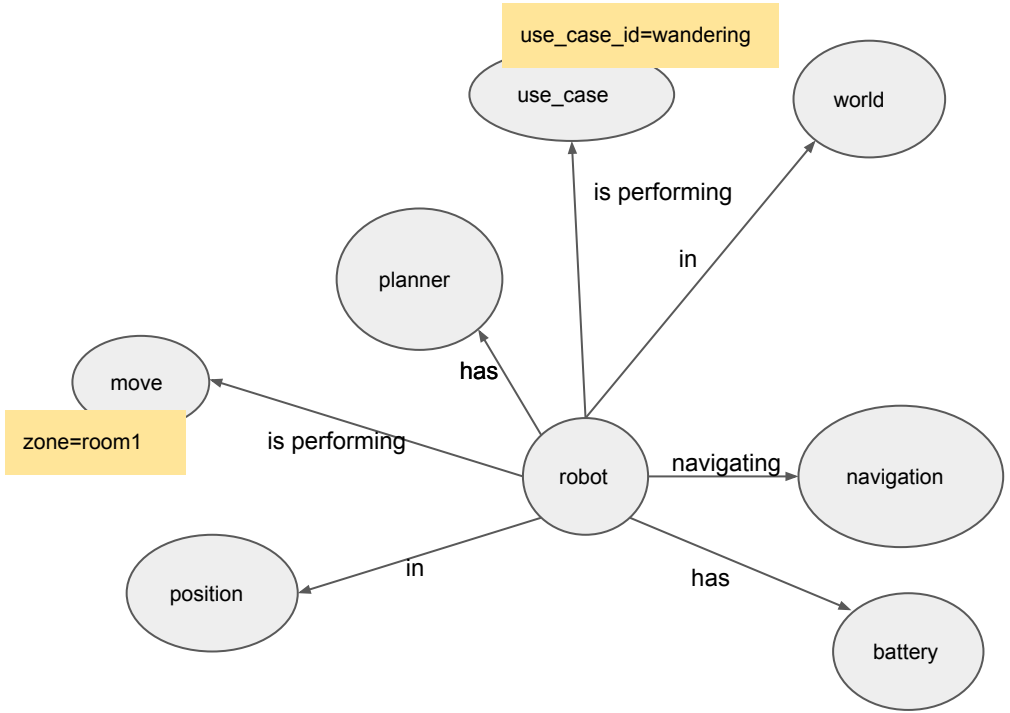
DSR: Un ejemplo de uso (2/4)

El componente que se encarga de la navegación, al ver que el robot se quiere mover, cambia el estado de la acción “move” de “wants_to” a “is performing”



DSR: Un ejemplo de uso (3/4)

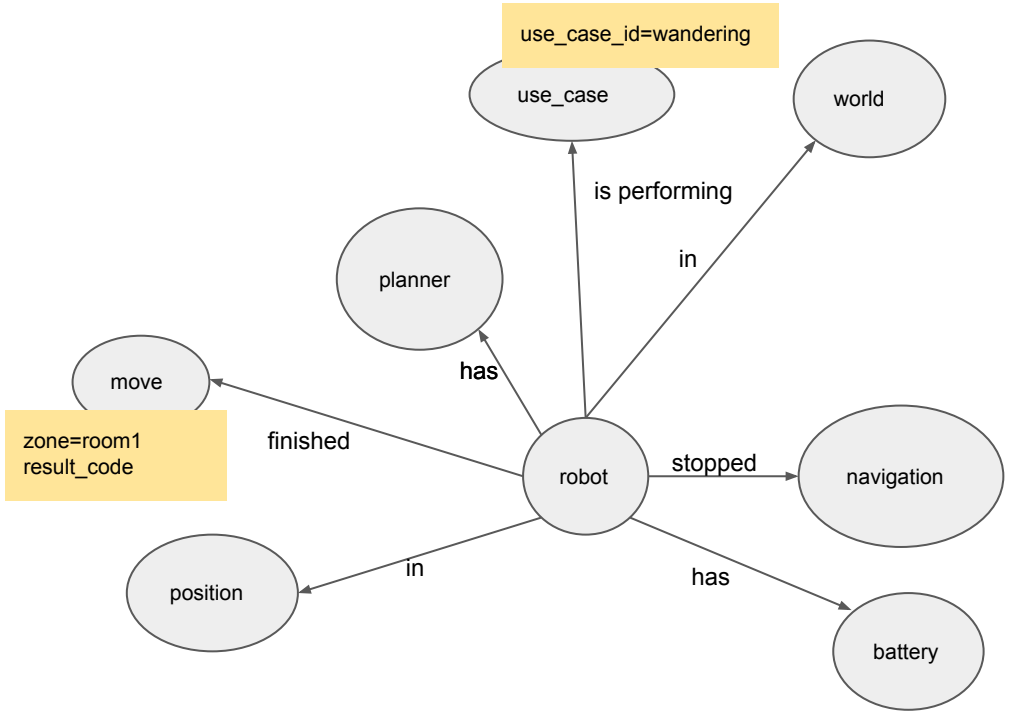
Ahora el robot se está moviendo.



DSR: Un ejemplo de uso (4/4)

Cuando concluye el movimiento, el agente de navegación actualiza el DSR en consecuencia.

Fijaos que aparece (robot->finished->move) y un atributo “result_code” en la acción que indica cómo ha ido



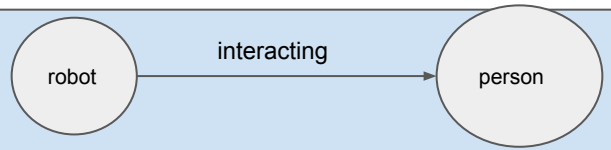
ONTOLOGÍA DEL DSR

- Vaya por delante que igual no estoy usando la palabra “ontología” bien.
- En el DSR, hemos definido una serie de reglas y normalizado la información que aparece, para que la funcionalidad del robot sea más fácil de manejar y, sobre todo, más escalable.
- Así, tenemos:
  - Tipos de nodos.
  - Tipos de enlaces.
  - Secuencias de ejecución de acciones y casos de uso.
  - Perfiles de usuario.
  - Agendas (de personas o robots).
- Vamos a irlos explicando…

ONTOLOGÍA DEL DSR: TIPOS DE NODOS

- **Nodo *robot*:**
  - Del nodo robot cuelgan una serie de nodos (navigation, battery, planner, etc) que bien podrían ser atributos del nodo. Se siguen manteniendo porque no hacen daño, pero cualquier día les damos pasaporte y metemos toda esa info como atributos del nodo robot.
- **Nodo *person*:**
  - Representa a una persona detectada por el robot. Contiene su posición. Si se reconoce a la persona, en este nodo aparecen como atributos el ID de la persona, su agenda, su perfil de usuario, sus preferencias de interacción con el robot, y su selección de menú. De todo esto hablamos luego.
  - El nodo *person* se enlaza al nodo *robot* de alguna de estas maneras:
    - *(person -> is_with -> robot)*: Personas detectadas cerca del robot.
    - *(robot -> interacting -> person)*: Persona que interactúa con el robot. En nuestros casos de uso sólo puede haber una.

Un ejemplo del aspecto que tiene en el DSR lo que pongo de *(nodo->enlace->nodo)*, en este caso *(robot->interacting->person)*. Para el resto, extrapoláis



ONTOLOGÍA DEL DSR: TIPOS DE NODOS

Nodos de acción:

- Indican una acción del robot. Cuelgan del nodo “robot” a través de unos enlaces determinados.
- Tipos de nodos de acción (*<nodo_acción>*): Ahora mismo tenemos:
  - **move**, **say**, **play**, **show, dock, get_random_goal**
  - Algunos de estos nodos tienen atributos. Así:
    - move: pose, goal y zones.
    - say, show: el texto que se quiere decir / mostrar
    - play: el sonido que se quiere reproducir
    - get_random_goal: la zona donde se va a obtener un destino aleatorio para la navegación
- Enlaces entre el nodo robot y el nodo acción (*robot-><tipo_enlace>-><nodo_acción>*):
  - *<tipo_enlace>*:
    - **wants_to:** Indica que alguien (normalmente un agente planificador) quiere que el robot realice una acción. Por ejemplo, *(robot->wants_to->play)* es que alguien quiere que el robot reproduzca un sonido.
    - **is_perfoming:** Indica que el robot está ejecutando esa acción.
    - **finished:** Indica que el robot ha terminado de ejecutar esa acción. Cuando aparece este enlace, en el *<nodo_acción>* aparece también un atributo, *result_code*, que indica el resultado de dicha acción.
    - **abort:** Indica que una acción que estaba en ejecución (en *is_performing*) no se va a poder concluir. También aparece un *result_code* en el *<nodo_acción>*.
    - **cancel:** Indica que una acción que estaba planificada (en *wants_to*) finalmente no se va a ejecutar. También aparece un *result_code* en el *<nodo_acción>*.

ONTOLOGÍA DEL DSR: TIPOS DE NODOS

- **Nodo “use_case”:**
  - Contiene el caso de uso que está ejecutando el robot. Enlaza con el nodo robot igual que los nodos de acción, pero usando sólo los enlaces

    wants_to, is_performing, finished

  - Al igual que con los nodos de acción, al aparecer (robot->finished->use_case), en el nodo *use_case* también aparece un atributo *result_code* con el resultado del caso de uso.

ONTOLOGÍA DEL DSR: TIPOS DE ENLACES

- Lo dicho:
- Para las personas, tenemos:
  - is_with
  - interacting
- Para acciones tenemos:
  - wants_to
  - is_performing
  - finished
  - abort
  - cancel

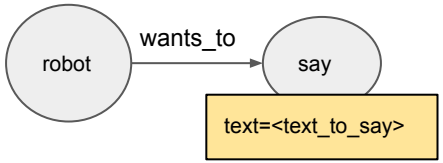
Estos tres también para el nodo use_case

- Y luego otros tipos de enlaces asociados a esos nodos que cuelgan del robot.

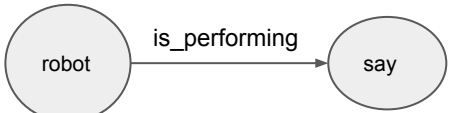
ONTOLOGÍA DEL DSR: SECUENCIAS DE EJECUCIÓN DE ACCIONES Y CASOS DE USO

Para los nodos de acción y de caso de uso, el flujo normal (si no se aborta ni cancela nada) es:

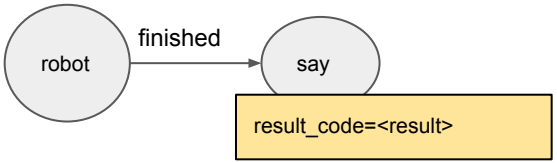
- *(robot -> wants_to -> <X>)*
- *(robot -> is_performing- > <X>)*
- *(robot -> finished -> <X>)* (check *result_code* in node <*X*>)



(a)



(b)



(c)

ONTOLOGÍA DEL DSR: PERFILES DE USUARIO

- Hay un agente (bbddAgent) que trasiega datos entre el DSR y una base de datos mongodb donde se almacena información persistente (recordemos: DSR sólo es contexto inmediato).
- Cuando se reconoce una persona, el bbddAgent carga en el nodo person una serie de atributos.
  - **identifier:** Una *string* con el identificador (nombre, normalmente), de la persona
  - **safe_distance:** Un *float* que indica la distancia mínima a la que el robot se puede acercar a la persona. Esta es la distancia de interacción.
  - **comm_parameters:** Un JSON serializado que contiene las preferencias de comunicación de la persona.
  - **skills_parameters:** Un JSON serializado que contiene el perfil de usuario (estructura **profile**).
  - **menu:** Un JSON serializado que contiene las opciones de menú de esa persona para la siguiente cena y el siguiente almuerzo. Si no está, entonces es que aún no se saben.
  - **activities:** Un JSON serializado que contiene la agenda de la persona.
  - **reminder:** Un *string* que contiene un recordatorio que se ha hecho a la persona. Se utiliza para evitar múltiples recordatorios (si está ese atributo, es que ya se ha hecho el recordatorio).

ONTOLOGÍA DEL DSR: PERFILES DE USUARIO

Lo que hay dentro de los JSON serializados del nodo *person*:

```
struct CommParameters{
    bool enable;
    float volume;
    bool subtitles;
    int text_size;
    bool only_images;
};
```

```
// indica si esa persona permite que el robot interactúe con ella o no
// volumen de voz deseado para el robot
// indica si el robot debe o no usar subtítulos al interactuar
// tamaño de letra deseado
// indica si el interfaz debe mostrar texto, o sólo imágenes
```

ONTOLOGÍA DEL DSR: PERFILES DE USUARIO

Lo que hay dentro de los JSON serializados del nodo *person*:

```
// Person profile
struct Profile{
    int avisual;
    int ccogni;
    int cmov;
    int disco;
    int humor;
    int inro;
    int naudicion;
    int ncansancio;
};
```

```
// agudeza visual
// capacidad cognitiva
// capacidad de movimiento
// disposición a cooperar
// humor
// nivel deseado de interacción con el robot
// nivel de audición
// nivel de cansancio
```

ONTOLOGÍA DEL DSR: PERFILES DE USUARIO

Lo que hay dentro de los JSON serializados del nodo *person*:

```
// Menu Choices
struct MenuChoices{
    std::string primerPlato;
    std::string segundoPlato;
    std::string postre;
};
```

ONTOLOGÍA DEL DSR: AGENDAS

La agenda es un atributo de la persona… pero también es un atributo que aparece en el nodo robot, indicando si el robot tiene agendadas actividades (por ejemplo, ir a animar el cotarro en terapia musical).

```
// Activities
struct Activity{
   std::string nombre;
   std::string hora;
   std::string lugar;
};
```

```
// nombre de la actividad
// hora a la que se realiza la actividad
// lugar en que se realiza la actividad
```