**Agente**

Para determinar si un elemento de software o hardware es un agente o no, nos basamos en criterios de agencia:

1. Un agente trabaja para beneficio o representación de alguien. El agente realiza una tarea para un usuario o un sistema.
2. Un agente es autónomo.

Proactividad: el agente decide qué hacer a continuación. Propone sus propios objetivos.

Persistencia: el agente tiene su propio hilo de ejecución

1. Un agente se comunica.

Capacidad social: el agente se comunica a través de mensajes con:

* Su usuario ... Proporcionando recomendaciones y recibiendo retroalimentación.
* Un sistema… Proporcionando datos obtenidos de un repositorio.
* Otros agentes ... Colaboración en un sistema multiagente.

1. Un agente percibe su entorno y lo cambia.

Reactividad: el agente reacciona según lo que percibe o se le dice / pregunta.

Deliberación: el agente planea y actúa de acuerdo a lo que cree sobre su entorno.

**Tipos de agentes**

Agentes de software

Objetos que coinciden con los criterios de agencia.

Agentes de hardware

Robots.

Agentes de usuario y agentes de recomendación

Modelado y personalización de usuarios.

Agentes inteligentes

Agentes deliberativos y agentes de aprendizaje.

Sistemas de agentes múltiples

Agentes colaborativos.

**AOP: AgentOriented Programming**

Guías/ Reglas para el diseño y desarrollo de agentes inteligentes deliberativos.

Componentes del agente:

* Mensajes (información o solicitudes).
* Creencias (representación de su mundo dinámico).
* Compromisos (objetivos, metas).
* Capacidades (acciones a realizar en el mundo real).

Algoritmo básico:

**while**(!hasFinished)

**{**

**Reads** messages (from its user or other agents).

**Perceives** the situation.

**Revises** beliefs (representación del mundo basada en los mensajes de recepción y la situación).

**Establishes** commitments (based on beliefs)

**Selects** a commitment.

**Executes** capabilities to cumplir con el compromiso

**}//end while**

**Sistemas multiagente**

* **Inteligencia Artificial Distribuida:** El estudio, construcción y aplicación de sistemas multiagente.
* **Sistema multiagente:** Un sistema en el que varios agentes inteligentes interactúan para perseguir un conjunto de objetivos o realizar un conjunto de tareas.

Cúando y cómo deben interactuar los agentes inteligentes; y cúales deben interactuar para lograr?

**Dos enfoques:**

* **Top down**

Reglas de nivel de grupo que restringen adecuadamente la interacción a nivel de agentes individuales.

* **Buttom up**

Capacidades específicas del agente individual que resultan en una interacción adecuada a nivel de grupo.

**Challenging issues:**

1. **¿Cómo habilitar a los agentes para descomponer sus objetivos y tareas y para sintetizar resultados y soluciones parciales?**
2. **¿Cómo habilitar a los agentes para comunicarse? ¿Qué lenguajes de comunicación y protocolos utilizar?**
3. **¿Cómo habilitar a los agentes para representar y razonar sobre las acciones, planes y conocimientos de otros agentes con el fin de interactuar adecuadamente con ellos?**
4. **¿Cómo habilitar a los agentes para representar y razonar sobre el estado de su proceso de interacción? ¿Cómo permitirles averiguar si han logrado progresos y mejorar su coordinación?**

**Código Agente**

UDLAgent.java

Sortknowledgebase es usa: Cuando no es requerido aprender, proveemos la misma prioridad a todas las reglas para evitar repetición de acciones o situaciones.

Cuando es requerido aprender, el agente establece prioridad a sus reglas de acuerdo al resultado obtenido.

Todos los agentes deben tener la capacidad de aprender; así como avisar en Blackboard cuando se cumplan los milestones.

El agente carga su knowledge base de las reglas que son independientes del mismo. Leemos desde un archivo .txt para asignar el número de regla, las condiciones y los compromisos y crear la regla. Si es regla prioritaria se indica, y se agrega al knowledge base. El knowledge base es un vector de reglas.

**Proceso de deliberación del agente:** Corresponde a la generación de posibles compromisos basados en el knowledge base y creencias. En caso de tener compromisos pendientes, ordenamos el knowledge base y obtenemos todas las reglas de nuestro knowledge base. Para cada regla, si la condición de la creencia coincide con la condición de la regla, agregamos el compromiso de dicha regla.

El agente **selecciona un compromiso** de los candidatos por 2 opciones:

1. Considera sólo candidatos que producen situaciones que aún no han sucedido. Después selecciona la mejor situación de acuerdo a la función heurística (definido en la clase Situation).

2. Si todas las posibles situaciones han pasado, selecciona aleatoriamente un compromiso candidato.

Un agente de Puzzle extiende de Agent, además de que ordena su base, checa si hay compromisos prioritarios, incluye compromisos indicados en una regla, checa los milestones, ejecuta las reglas de estrategia para determinar cuando no considerar compromisos que no convienen.

El agente interpreta un compromiso produciendo la situación que ocurrirá si el compromiso es aplicado, devolviendo una situación.

El agente aprende y construye una nueva regla cuando un milestone es completado: Checamos si existe algún predicado previo, la condición del milestone y la secuencia de movimientos para crear la regla. Esta nueva regla la agregamos al knowledge base.

Una regla está formada por número, condiciones y compromisos.