Universitat Politècnica de Catalunya

Visión por computador

Bachelor Degree in Computer Science

Práctica SID

Simulación de la actividad de un río

Autor:
Joaquim Marset
Marta Barroso

Profesor:
Ulises Cortés
Luís Oliva

Q2 Curso 2017-2018

Índice

1. Introducción

Esta práctica ha consistido en la implementación de un sistema multiagente que simula el comportamiento de un río y de las industrias y depuradoras que interactúan con él. Previamente a su desarrollo hemos diseñado la ontología (*Protege*) y el sistema (*Prometheus*). Posteriormente se ha llevado a cabo la implementación de los agentes y de sus protocolos de interacción.

Respecto a los agentes, hemos definido los siguientes: Río, EDAR, Industria y Lluvia. A lo largo de la práctica describiremos sus atributos y qué conversaciones establecen.

Además para agilizar la comprensión de los mensajes, como contenido hemos usado acciones y conceptos. Para ello hemos implementado un vocabulario y una ontología que nos permite definir los *schemas* de cada acción y concepto.

2. Objetivos

Tal y como se ha comentado inicialmente, el objetivo principal es simular la actividad de un río y otros actores que desarrollan su actividad en él. Pero antes hemos tenido que:

- Conocer y aplicar metodologías de desarrollo de sistemas multiagente.
- Conocer los protocolos de interacción entre agentes.
- Analizar las necesidades de comunicación de cada agente y saber desarrollar sus protocolos de comunicación.
- Analizar el correcto funcionamiento del sistema.

3. Diseño del sistema

A continuación se muestran una simplicación el diseño del sistema.

Industria			EDAR		
Escenario	Percepción	Evento	Escenario	Percepción	Evento
Usar agua	cíclico	coger agua	Depurar agua	recibimiento agua sucia (solicitud de vertido aceptada)	aumentar volumen tanque / depurar agua
				lluvia	aumentar volumen tanque
	coge agua	aumentar volumen del tanque		tanque lleno al 75%	vertir agua en río
	coge agua	ensuciar agua	Gestionar peticiones ajenas de vertido	solicitud de vertido	gestionar solicitud
Gestionar agua	tanque lleno	realizar petición de vertido a EDAR		solicitud gestionada	responder petición a l
Hacer acción	solicitud de vertido aceptada	vertir agua a la EDAR	Gestionar peticiones propias de vertido	tanque con capacidad libre	realizar petición de recibir vertido
	tanque con capacidad suficiente	realizar vertido en tanque	Gestionar peticiones ajenas de vertido/ Gestionar peticiones propias de vertido	solicitud de vertido aceptada	informar coste depuración a l
	solicitud de vertido rechazada y tanque lleno y lluvia	realizar vertido en río (ilegal)			
Gestionar agua	solicitud de vertido aceptada	pagar cuota depuración			
	cíclico	aumentar presupuesto			
	solicitud de vertido	responder petición a EDAR			

Figura 1: Tabla esquemática de los agentes Industria y EDAR

Agua		
Escenario	Percepción	Evento
Coger agua	Industria coge agua	ensuciarse
Depurar agua	recibimiento agua sucia a EDAR	limpiarse
Hacer acción/depurar agua	realizar vertido en tanque/recibimiento agua sucia/lluvia	mezclarse
Hacer acción/ depura agua	vertir agua en río	avanzar
Usar agua	extraer agua	enviar

Figura 2: Tabla esquemática de las operaciones a realizar de una masa de agua

4. Descripción de la ontología

Para la construcción de la ontología nos hemos basado en la ontología WaWo. Dado que ésta contenida mucha información que considerábamos irrelevante para esta práctica hemos creado nuestra propia ontología.

Tal y como se puede ver en la siguiente imagen hemos definido también los conceptos y acciones. Tendremos como único concepto la masa de agua y como clases: Industria, Río, EDAR (depuradora) y Lluvia. Aunque no aparece de manera explícita en el enunciado, hemos considerado crear una clase Lluvia para que las industrias, la EDAR y el río sepan cuando llueve y para de llover.

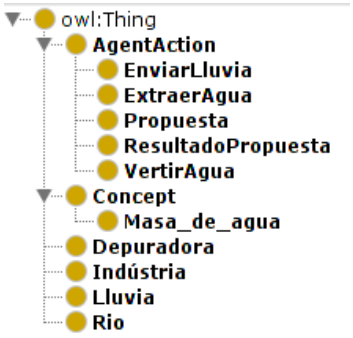

Figura 3: Clases

A continuación se pueden ver las propiedades de todos los objetos de cada clase. Las industrias y la depuradora dispondrán de una masa de agua a la que llamaremos tanque donde podrán ir guardando el agua. Además la EDAR tendrá un conjunto de identificadores de industrias con quienes podrá comunicarse. Por último, el río estará formado por un conjunto de tramos.

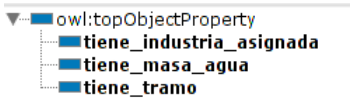

Figura 4: Object properties

Por último, mostramos todos los atributos privados que pueden tener los objetos:

- Masa de agua: Se caracteriza por un volumen y un conjunto de contaminantes. También tendrá un coste asociado.
- Industria: Tiene un identificador, un presupuesto, un tanque (masa de agua) con una cierta capacidad.
- EDAR: Tiene un identificador, un tanque y una capacidad.
- Río: Está formado por un conjunto de tramos y un índice que indica el tramo actual donde las industrias extraerán agua y la depuradora vertirá.
- Lluvia: Está formada por una masa de agua que llamaremos aguaLluvia.

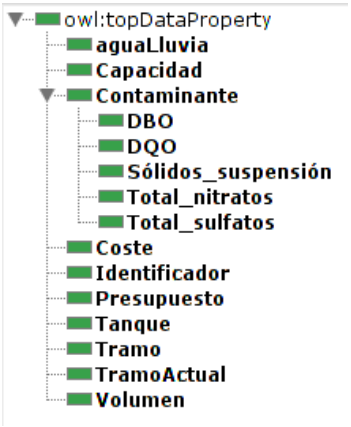

Figura 5: Data properties

5. Descripción del sistema

El sistema está distribuido en los siguientes directorios:

- jade: Se encuentran todas las librerías necesarias para compilar y ejecutar el código.
- src/Agentes: Están definidas las 4 clases de los agentes: Río, Industria, EDAR y Lluvia.
- src/Ontologia: Se encuentras las clases de los conceptos, acciones y el vocabulario y la ontología para la codificación y decodificación de mensajes.

6. Uso del programa

Para ejecutar el sistema es necesario ejecutar el script *main.sh* ubicado en el directorio de la práctica. Este fichero compila todas las clases java y define los parámetros de entrada de los agentes. El usuario será el encargado de decidir el número de industrias así como la capacidad de cada una de ellas. Para ello, existen dos opciones: definir la capacidad de cada industria manualmente o definir la misma capacidad para todas las industrias. Después el script pasará como argumentos a los agentes Lluvia y EDAR la cantidad de industrias y los AIDS que necesita cada agente.

7. Conceptos

A lo largo de la práctica usamos un único concepto: MasaDeAgua. Toda masa de agua consta de un volumen (litros) y un grupo de contaminantes:

- DBO (demanda biológica de oxígeno): Cantidad de oxígeno que los microorganismos, hongos y plancton, consumen durante la degradación de las sustancias orgánicas contenidas en la muestra.

- DQO (demanda químico de oxígeno): Cantidad de oxígeno necesario para oxidar la materia orgánica por medios químicos y convertirla en dióxido de carbono y agua.
- Sólidos en suspensión: Fracción del total de sólidos en el agua que pueden ser separados por filtración.
- Nitratos: Sal química derivada del nitrógeno presente en el agua de suministro público debido a la contaminación de las aguas naturales por compuestos nitrogenados.
- Sulfatos: Sales solubles que se presentan en las aguas naturales en un amplio intervalo de concentraciones. Pueden actuar como laxantes cuando se ingieren en cantidades elevadas.

Distinguiremos tres tipos: agua potable/limpia (el agua se ha limpiado), agua contaminada/sucia/-residual y agua muy contaminada. Consideraremos que el agua estará muy contaminada cuando más de la mitad de los contaminantes superen una constante.

Para pasar de un tipo a otro hemos implementado dos métodos:

- ensuciar(): Se le suma una constante a cada contaminante.
- limpiar(): Por cada contaminante hemos establecido un rango en el que lo consideramos nocivo. Para considerar que el agua esté limpia, todos los contaminantes deben tener un valor que pertenezca al rango asociado.

Contaminante	Valor mínimo	Valor máximo
DBO	0.75	1.5
DQO	1	5
Sólidos en suspensión	25	50
Nitratos	10	50
Sulfatos	50	400

Tabla 1: Valores mínimos y máximos de los contaminantes para agua potable

Respecto a dos masas de agua, éstas se pueden mezclar. La masa de agua final tendrá como volumen la suma de ambas masas. Cada contaminante será la suma de la proporción (mg) del contaminante en cada masa.

8. Codificación de los mensajes

Con el objetivo de simplificar la codificación y decodificación de los mensajes hemos implementado una un vocabulario y una ontología. La ontología específica de la aplicación describe los elementos que pueden usarse como contenido de los mensajes del agente. Se compone de dos partes, un vocabulario (*Vocabulary*) que describe la terminología de los conceptos utilizados por los agentes en su espacio de comunicación y la nomenclatura (*OntologyMgr()*) de las relaciones entre estos conceptos, que describen su semántica y estructura. Para ello es necesario que todos los agentes tengan como atributos el lenguaje de códec que implementa JADE y la ontología definida previamente:

$$Codec codec = new SLCodec()$$

$$Ontology ontology = OntologyMgr.getInstance()$$

9. Agentes

9.1. Río

El agente Río representa un medio que contiene un conjunto de tramos donde cada tramo es una masa de agua. Cumple con las funciones de avanzar los tramos de manera cíclica y de recibir agua. Las industrias extraerán agua de él y las EDAR vertirán. Siempre aceptará agua pero no aceptará infinitos vertidos ya que se puede quedar sin agua.

Durante la ejecución del sistema se comunicará con las industrias para enviarles las masas de agua que puedan necesitar.

9.1.1. Atributos

- *List<MasaDeAgua>tramos*: Conjunto de tramos que forman el río. Todos los tramos se inician con la misma masa de agua.
- *int tramoActual*: Índice que representa el tramo donde las EDAR vierten y las industrias cogen agua. Cada 20 segundos el índice avanza al tramo siguiente.
- *float volumenTramos*: Indica el volumen de cada tramo del río. Corresponderá al triple de la capacidad máxima de todas las industrias.

9.1.2. Protocolos de interacción

- FluirRio *TickerBehaviour*: Simula el transcurro de las aguas del río. Cada 20 segundos el índice avanza al siguiente tramo.
- GestionarMensajesRecibidosBehaviour *SimpleAchieveREResponder*: Reacciona a los siguientes mensajes:
  - Mensaje informativo de Lluvia: El río recibe la masa de agua enviada por la lluvia y la mezcla con la masa de cada tramo.
  - Mensaje informativo de EDAR: El río recibe la masa de agua a vertir de la EDAR y la mezcla con el agua del tramo actual.
  - Solicitud de extracción de agua de Industria: El río extrae la masa de agua del volumen especificado por la industria siempre y cuando el río contenga suficiente agua para satisfacer la solicitud. En caso afirmativo enviará la masa de agua a la industria y actualizará la masa de agua del tramo actual.

9.2. EDAR

El agente EDAR representa una depuradora que recibe el agua usada por las industrias, la depura y la vierte al río. Disponen de un tanque donde van acumulando el agua a depurar. Por cada solicitud de agua que aceptan cobran un coste de depuración. Puede tomar la decisión de no aceptar la solicitud de vertido en caso que el agua esté muy contaminada o bien no tenga espacio suficiente en el tanque (la masa de agua debe caber por completo, no se puede fraccionar). En caso de tener libre más de un 30 % de la capacidad del tanque pedirá vertidos a las industrias.

A lo largo de la ejecución se comunicará con el río para informar de vertidos de agua limpia y con las industrias ya sea para gestionar vertidos solicitados por ellas o por la misma EDAR.

9.2.1. Atributos

- *float capacidad*: Cantidad de litros que puede almacenar en el tanque.
- *MasaDeAgua aguaTanque*: Masa de agua almacenada en el tanque. Inicialmente la masa de agua será nula. A medida que se vayan aceptando y haciendo vertidos, el agua se irá mezclando y limpiando antes de vertirse al río.
- *List<AID> industrias*: Identificadores de industrias que pueden solicitar vertidos. Así mismo, la EDAR también podrá solicitarlos.
- *AID rioAID*: Identificador del agente Río.
- *AID lluviaAID*: Identificador del agente Lluvia.

9.2.2. Protocolos de interacción

- ContractNetInitiatorBehaviour *ContractNetInitiator*: Se encarga de recibir todas las peticiones de vertido de las industrias. Después calcula el coste de depuración asociado y en orden descendiente va procesando todas las peticiones. Tanto si son aceptadas como rechazadas la EDAR informará a la industria en cuestión. El objetivo de la EDAR es conseguir el máximo beneficio posible por ese motivo trata con las solicitudes de mayor coste primero hasta que el tanque se llena completamente.
- RepetirProcesoDepuracionBehaviour *TickerBehaviour*: En caso que haya un volumen igual o superior a un 30 % la EDAR solicitará vertidos a todas las industrias.
- GestionarPeticionVertidoBehaviour *SimpleAchieveREResponder*: Reacciona a los mensajes de:
  - Solicitud de vertido de una industria: Responderá con un mensaje informativo a la solicitud de vertido de manera afirmativa siempre y cuando el agua no esté muy contaminada y tenga capacidad suficiente para almacenarla. De lo contrario rechazará la petición.
  - Mensaje informativo de la Lluvia: Recibe el agua de la lluvia y guardará toda la que quepa en el tanque. El agua sobrante suponemos que desaparece y la que cabe se mezcla con el agua que ya había anteriormente.
- RepetirVertirAguaRioBehaviour *TickerBehaviour*: Siempr e y cuando el volumen del tanque exceda el 75 % de su capacidad la EDAR enviará toda el agua del tanque al río. El mensaje es de tipo informativo, el río no responderá.

9.3. Industria

El agente Industria es quien inicia el sistema extrayendo agua. Dispone de un tanque donde puede ir guardando el agua usada y de un presupuesto para hacer frente a los costes de depuración. Podrá realizar extracciones siempre que el tanque no esté lleno. También podrá solicitar vertidos a la depuradora. Para ello la industria debe tener agua y presupuesto suficiente para pagar los costes de limpieza de aguas residuales. Estos criterios son los mismos que se deben cumplir cuando la EDAR es quien hace las peticiones.

9.3.1. Atributos

- *float capacidad*: Cantidad de agua que cabe en el tanque.
- *float presupuesto*: Cantidad de dinero para pagar los costes de depuración.
- *MasaDeAgua aguaTanque*: Masa de agua que hay en el tanque en ese instante. Inicialmente es nula. El agua recogida del río es contaminada antes de mezclarse con el agua del tanque.
- *AID rioAID*: Identificador del Río.
- *AID edarAID*: Identificador de la EDAR.
- *lluiaAID*: Identificador de la lluvia.
- *boolean estaLloviendo*: Indica 1 (llueve) o 0 (no llueve). Su valor se modifica cada vez que se recibe un mensaje informativo del agente Lluvia.

9.3.2. Protocolos de interacción

- ExtraerAguaBehaviour *SimpleAchieveREInitiator*: Se encarga de ensuciar y mezclar el agua enviada por el río con el agua del tanque.
- RepetirProcesosIndustrialesaBehaviour *TickerBehaviour*: Se comunica con el agente Rio para llevar a cabo la extracción de agua. Se intentará llenar el tanque lo máximo posible.
- RepetirVaciarTanqueBehaviour *SimpleAchieveREInitiator*: Recoge la respuesta de la EDAR. Si ésta ha aceptado su solicitud, la industria pagará y vaciará su tanque. De lo contrario permanecerá igual.
- RepetirVaciarTanqueBehaviour *TickerBehaviour*: Solicitará vertido a la EDAR solo si el tanque no está vacío y tiene dinero suficiente para pagar. En ese caso la industria enviará todo el agua del tanque a la EDAR. La industria podría decidir hacer un vertido ilegal si los flags *estaLloviendo* y *hacerByPass* están a 1. A continuación, el río recibirá la masa de agua contaminada.
- ResponderContractNetBehaviour *ContractNetResponder*: El comportamiento es muy similar al anterior pero en este caso es la EDAR quien realiza la petición. Cuando la industria la reciba deberá comprobar que tiene agua y presupuesto suficiente. Si las condiciones se cumplen enviará toda la masa de agua del tanque. En el caso que la depuradora acepte su solicitud, la industria actualizará su presupuesto y el tanque.

- RepetirIncrementarPresupuestoBehaviour *TickerBehaviour*: Incrementa el presupuesto en 20000 euros cada 20 segundos. Inicialmente el presupuesto es 100000.
- ComprobarSituacionLluvia *CyclicBehaviour*: Comprueba constantemente si el agente Lluvia ha enviado algún mensaje notificando si hay lluvia o no. En caso afirmativo, el flag *estaLloviendo* se pondrá a 1, 0 en caso contrario.

9.4. Lluvia

El agente Lluvia es el encargado de enviar una masa de agua al agente Río y a la EDAR. También debe avisar a todas las industrias para que éstas sepan que pueden hacer bypass.

9.4.1. Atributos

- *List <AID> industrias*: Lista de identificadores de industrias a las que la lluvia informará cuando se active el comportamiento.
- *AID edarAID*: Identificador de la EDAR.
- *private AID rioAID*: Identificador del río
- *MasaDeAgua aguaLluvia*: Masa de agua que se enviará cuando llueva. Tendrá unos valores iniciales de contaminación que se mantendrán constantes salvo el volumen que variará según el agua se envíe a la EDAR o al río.

9.4.2. Protocolos de interacción

- LloverBehaviour *TickerBehaviour*: Se encarga de informar a todas las industrias de cuándo llueve y deja de llover. Para ello va cambiando el valor del *status* de 1 (llueve) a 0 (no llueve). También envía una masa de agua a la EDAR con un volumen comprendido entre 1000 y 2000 y al río con un volumen entre 5000 y 10000.

10. Acciones

- EnviarLluvia: Lluvia envía a EDAR la masa de agua que llenará su tanque.
- ExtraerAgua: Industria envía al Río el volumen que tiene disponible en el tanque.
- Propuesta: Industria envía la masa de agua y el presupuesto que está dispuesta vertir a la EDAR.
- ResultadPropuesta: La EDAR envía el volumen y el coste de la masa de agua que aceptará, a la industria.
- VertirAgua: EDAR envía masa de agua a vertir al Río.
- SolicitarVertido: La industria solicita vertir una masa de agua sucia a la EDAR.

11. Conclusión

A lo largo del desarrollo han tenido lugar varios problemas. Por un lado la captación de la respuesta por el agente correcto. Para solucionarlo solo fue necesario modificar el *templateMaching* del comportamiento. Y por otro, la gestión de los mensajes del *ContractNet*. En un primer momento decidimos que la EDAR aceptaría toda la masa de agua de las peticiones. Después decidimos aceptar solo un % de esa masa, tal y como marca el enunciado.

También acordamos que tanto la depuradora como las industrias sabrían el coste de depuración asociado a una masa de agua. Esto es útil para descontar el presupuesto de la industria y saber cuando puede hacer un bypass. Y en el caso de las EDAR, para que pueda comunicarles el coste del vertido.

A modo de ampliación se podría cambiar la manera en la que la EDAR escoge los "mejores"vertidos. Nosotros usamos la relación volumen de la /presupvertirmasa a uesto. Escogemos primero las peticiones que den como resultado un cuociente pequeño, hasta que el tanque de la EDAR se llena. Otra manera posible, basándose en que el objetivo de la depuradora sería obtener el máximo beneficio, sería aplicar el algoritmo de la mochila (*knapsack problem*). Los pesos serían los volúmenes de las masas y los valores el coste de depuración asociado. De esta manera, se intentará aceptar el máximo número de propuestas tal que el valor obtenido es máximo.