DESARROLLO DE UN ENTORNO EIOT SOBRE DEVS: SISTEMA DE ALERTA Y GESTIÓN DE BLOOMS DE CIANOBACTERIAS

Segundo Esteban San Román
Fac. CC. Físicas, UCM, sesteban@ucm.es

Jesús Chacón Sombría
Fac. CC. Físicas, jeschaco@ucm.es

José Luis Risco Martín
Fac. de Informática, jlrisco@ucm.es

Gonzalo Carazo Barbero
Fac. de Informática, UCM, gocarazo@ucm.es

Eva Besada Portas
Fac. CC. Físicas, UCM , evabesada@ucm.es

https://doi.org/10.17979/spudc.9788497498418.0017

Resumen

Se utiliza Discrete Event System Specification (DEVS) como herramienta para modelar, simular, desarrollar y desplegar un proyecto acorde al paradigma Internet de la Cosas Medioambiental (EIoT). El software/hardware se diseña y desarrolla con una arquitectura típica de IoT, por capas: Cloud, Fog, Edge y Things. Se presenta como caso de aplicación un sistema para monitorizar y controlar Unmaned Surface Vehicles (USVs) que toman medidas en masas de agua dulce con el objetivo de monitorizar y predecir Blooms de cianobacterias. Modelar un proyecto EIoT utilizando DEVS permite un diseño software progresivo, que a su vez facilita mezclar elementos simulados y reales. Este diseño dirigido por modelos permite refinar los modelos mediante simulaciones, para facilitar el despliegue final del sistema real.

Palabras clave: Modelado, Simulación, Despliegue de Software, Discrete Event System Specification (DEVS), Diseño basado en modelos, Internet de las Cosas (IoT) Medioambiental (EIoT), Cianobacterias, Calidad de Aguas.

1 INTRODUCCIÓN

Debido al calentamiento global nos enfrentamos a una proliferación de cianobacterias en aguas continentales. Estas cianobacterias pueden ser muy perjudiciales para la salud del ser humano, sobre todo cuando se producen crecimientos masivos denominados *Blooms* [5]. Estos *Blooms* de cianobacterias tienen un comportamiento dinámico, se mueven horizontalmente, desplazadas por corrientes y vientos, y verticalmente, por cambios de densidad provocados por su metabolismo. Es necesario monitorizar el crecimiento y el desplazamiento de estos *Blooms* para identificar modelos dinámicos de su comportamiento.

Con el objetivo de modelar y predecir el comportamiento de estos *Blooms* se ha planteado el proyecto "Hacia un sistema Integral para la Alerta y GEStión e BLOOMs de cianobacterias en aguas continentales" (IA-GES-BLOOM-CM) [3]. Es un proyecto multidisciplinar, que incluye a Biólogos, Físicos, Matemáticos, Ingenieros Informáticos e Ingenieros Electrónicos, que deben aunar esfuerzos para conseguir desarrollar el sistema. Los Biólogos, como buenos conocedores del sistema, transmiten su conocimiento histórico a los ingenieros, que tienen que desarrollar el hardware y el software para poder monitorizar el sistema biológico. Utilizando los datos, los científicos deben modelar el sistema biológico con suficiente precisión como para poder realizar predicciones a corto plazo. Estas predicciones serán de gran utilidad para los gestores del agua.

Los diferentes subsistemas del proyecto (USVs y Sensores, Visión, Modelado y Software EIoT) se van desarrollando en paralelo. El entorno EIoT debe integrar la información que suministra los otros subsistemas, siguiendo el paradigma de IoT aplicado al medio ambiente [6]. Estos subsistemas todavía están en desarrollo y por ello se adopta un diseño basado en modelos, que inicialmente permita su simulación y posteriormente el despliegue de los subsistemas reales de una forma progresiva. Además, los subsistemas se comunicarán a través de eventos, por lo que un formalismo como DEVS [7] es idóneo para su diseño e implementación. La figura 1 muestra el entorno EIoT que proponemos desarrollar, siguiendo una división por capas.

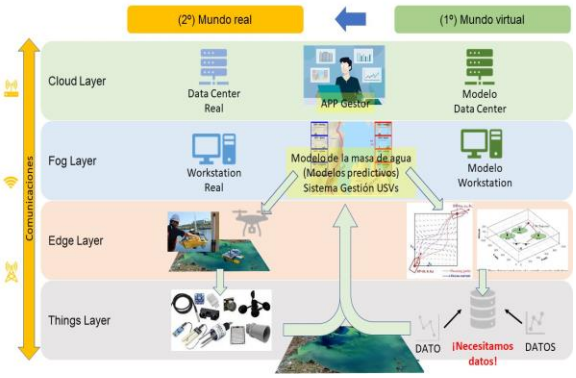

Figura 1: Capas de EIoT Simuladas y Reales.

La zona derecha muestra la simulación de las capas sobre un mundo virtual. La zona izquierda muestra la implementación de las capas ya en el mundo real. Los diferentes subsistemas se implementarán primero de forma simulada y posteriormente se desplegarán de forma progresiva sobre los sistemas reales.

La capa inferior, formada por *Things* o sensores, tomarán medidas del entorno, de la masa de agua y del *Bloom*. Estas medidas serán procesadas por la capa *Edge* para el guiado de los USVs o por la capa Fog para planificar la navegación de los USVs. La capa *Fog* también procesará y almacenará los datos, mostrando en tiempo real el estado del sistema al controlador de la masa de agua. Además, toda la información es subida a la capa *Cloud*, donde modelos basados en IA podrán predecir y generar nuevas zonas de interés para la capa *Fog*. Además, la capa *Cloud* deberá generar informes web para que el Gestor de Aguas pueda tomar decisiones sobre la captación, tratamiento, cortes o prohibiciones.

En el artículo primero se presenta una arquitectura DEVS de las capas EIoT que se utilizarán para

https://doi.org/10.17979/spudc.9788497498418.0017

gestionar la información del proyecto. Tras explicar cada una de las capas, el artículo se centra en la capa Edge-Things, que es la que más elementos de control contiene. Después se presentan algunas simulaciones desarrolladas sobre esta capa. Finalmente se presentan los siguientes pasos que se darán en el desarrollo del proyecto.

2 Arquitectura por capas EIoT

La figura 2 muestra la arquitectura DEVS que se ha construido para modelar el entorno EIoT.

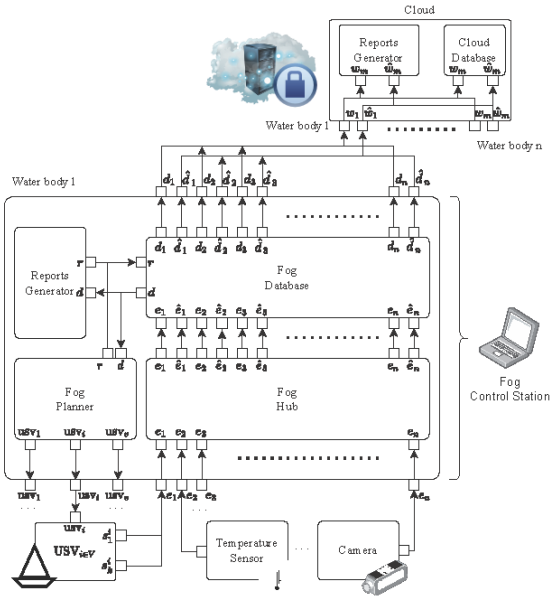

Figura 2: Arquitectura DEVS de las capas de EIoT.

La comunicación entre elementos se realiza a través de eventos, que comparten un interfaz. El código 1 nuestra una implementación Python.

Código 1: Interfaz de los eventos.

```
class Event:

  '''A message to model events'''
  id: str
  source: str
  target: str = field(default=None)
  timestamp: datetime =
              field(default_factory=datetime.now)
  payload: dict = field(default_factory=dict)
```

En Python los diccionarios son una herramienta muy potente. Ofrecen mucha flexibilidad para que el campo *payload* permita enviar información de diferentes tipos. Esto permite unificar el tipo de puerto y evento, siendo tan solo necesario definir si el puerto es de entrada o de salida.

2.1 Capa Cloud

La capa *Cloud* debe encargarse de gestionar múltiples masas de agua. Esta capa marca la jerarquía del sistema, pues contendrá instancias de las capas *Fog* enlazadas. Implementa una base de datos que puede almacenar el histórico de datos procesados por cada capa *Fog*. De momento tan solo almacena los ficheros de telemetría generados durante todas las simulaciones. Gracias a este histórico y a algoritmos avanzados de modelado podrá emitir informes de predicción. Estos informes pueden ser para el Gestor de Aguas, que maneja la captación del agua, o para el Supervisor de la masa de agua, que maneja el centro de control de la capa *Fog* asociada a esa masa de agua.

La capa *Cloud* finalmente deberá ejecutarse en un servidor IoT. Gracias a la implementación modular este despliegue será automático.

2.2 Capa Fog

Cada masa de agua será controlada por una capa *Fog*, que a su vez controlará múltiples *USVs* y *Things*. Esta capa, al igual que la anterior, dispone de una base de datos para guardar la información actual, pero en este caso su tamaño es más reducido, pues no es necesario disponer de los históricos.

Toma medidas a través del Hub, que la conecta con la capa Edge, y realiza el preprocesado de los datos. Actualmente en esta capa tan solo se está realizando una detección de *outliers*.

Con las medidas actuales o recientes se generan informes en tiempo real para el Supervisor de la masa de agua. Utilizando estos informes y las predicciones facilitadas por la capa Cloud, tanto el planificador automático como el Supervisor pueden generar un plan de toma de muestras para la capa Edge.

2.3 Capa Edge

En esta capa tendremos, tanto sensores básicos (*Things)*, como sensores más complejos como son los *USVs*. Estos últimos tendrán capacidad de cómputo, de movimiento y manejarán múltiples sensores.

2.3.1 Things

Inicialmente los diferentes sensores deben ser simulados, para posteriormente ser sustituidos por sensores reales de forma gradual, cuando estén disponibles. Para ello se ha definido un interfaz de entrada/salida. Este diseño nos ha permitido trabajar con varias versiones del sensor: *FileSensor, LogicSensor, LocalEEMSSensor* y *RemoteSensor*, como se muestra en la figura 3.

https://doi.org/10.17979/spudc.9788497498418.0017

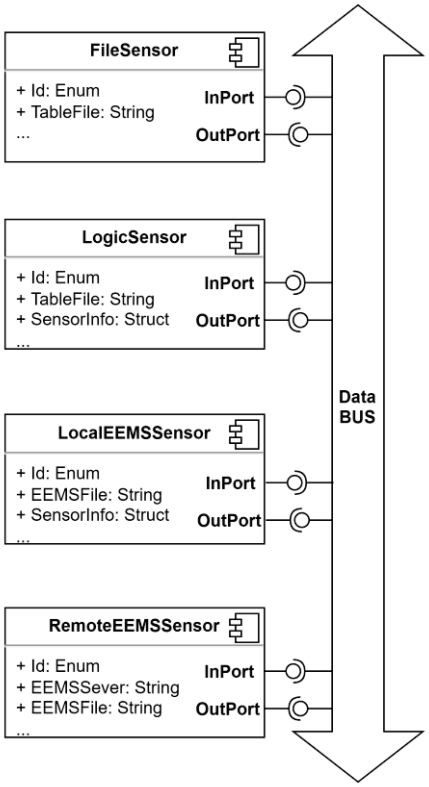

Figura 3: Diferentes versiones de Sensores simulados.

En la versión *FileSensor* es el mismo sensor quien dispara un evento de medida sobre el Bus cuando la simulación alcanza el tiempo pre-programado en el fichero *TableFile*. El evento contiene la medida, una marca de tiempo, que habitualmente está en formato *DateTime,* y el identificador del sensor. Este evento será propagado por el BUS al resto del entorno *EIoT*, siendo capturado por otras unidades que procesarán la información y disparan otros eventos internos o externos.

En la versión *LogicSensor* el sensor recibe una petición de medida a través del puerto de entrada. La tabla 1 muestra un ejemplo del contenido del mendaje de petición de medidas de Nitrógeno y Oxígeno.

Tabla 1: Solicitud de medidas.

DateTime	Lat	Lon	Depth	Sensor
2008-09-12 00:29:00	47,5	-122,242	0.5	WQ_N
2008-09-12 00:30:00	47,50417	-122,242	0.5	WQ_O
2008-09-12 00:39:00	47,50833	-122,242	1.0	WQ_N
2008-09-12 00:40:00	47,5125	-122,242	1.0	WQ_O

El Sensor ya tiene una lógica de comportamiento básica. Cuando recibe la primera petición sincroniza su tiempo local y devuelven un mensaje con las características del sensor, ver tabla 2.

Tabla 2: Primer mensaje de respuesta de un Sensor.

Id	Source	DateTime	Descr.	Delay(s)	max	min	...
WQ_N	SimSenN	2008-09-12 00:10:00	Sonda de Nitrogeno	41	0,5	0	...
WQ_O	SimSenO	2008-09-12 00:10:00	Sonda de Oxigeno	32	10	0	...

Esta información puede ser utilizada posteriormente por los elementos de proceso del entorno EIoT para manejar la información que envíe el sensor, p.e. para detección de *Outliers*. Después, ya envía la medida con el retardo y precisión indicadas en *SensorInfo*.

Un nivel más avanzado de simulación consiste en utilizar una simulación previa del entorno y de la masa de agua. Para ello se pueden utilizar herramientas externas aceptadas por el sector [1], como EFDC+, EEMS, etc. Estos simuladores de sistemas biológicos generan grandes cantidades de datos, que son guardados en ficheros con formato HDF5 (*Hierarchical Data Format*). Gracias a este formato, a pesar de su gran tamaño, estos ficheros pueden ser manejados por un computador personal.

La figura 4 muestra la simulación del lago Washington realizada con el simulador EEMS. Esto ha sido implementado en la versión *LocalEEMSSensor*. Como actualmente no podemos simular cianobacterias, en su lugar se simula el crecimiento de algas.

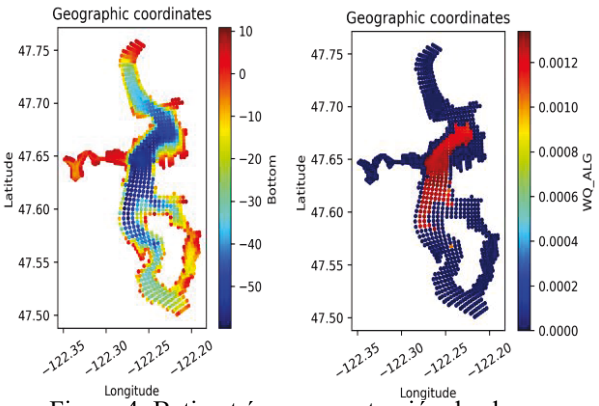

Figura 4: Batimetría y concentración de algas.

El sensor implementa un retardo, proporcionado en la propiedad *Delay*, equivalente al tiempo que tarda en realizar la medida ese sensor. A continuación, se busca en el fichero *EEMSFile* la señal solicitada, para el tiempo y lugar indicados, se le añade ruido, sesgo y se cuantifica acorde a las características técnicas del sensor. Finalmente se genera el evento con la medida y su información espacial y temporal como se muestra en la tabla 3. Se puede observar que cada sensor está introduciendo una latencia diferente.

https://doi.org/10.17979/spudc.9788497498418.0017

Tabla 3: Información de los mensajes del sensor.

DateTime	Lat	Lon	Depth	WQ_O	WQ_N
2008-09-12 00:29:41	47,5	-122,242	0.5	8,61	
2008-09-12 00:30:32	47,50417	-122,242	0.5		0,40
2008-09-12 00:39:41	47,50833	-122,242	1.0	8,58	
2008-09-12 00:40:32	47,5125	-122,242	1.0		0,36

Debido al gran tamaño de los ficheros que simulan las masas de agua, del orden de 100 GB, se ha generado la versión *RemoteEEMSSensor* que corre sobre un servidor con varios TB de disco duro. Este servidor alberga simulaciones de múltiples masas de agua. En este caso se ha montado un servicio con FLASK [2], con el que se le pueden realizar solicitudes en formato JSON, que está disponible en Python [4]. Esto añade una pequeña espera de milisegundo, por las latencias de internet, pero introduce realismo tanto a nivel de simulación como a nivel de despliegue del sistema real. Para ello, tan solo será necesario que el sensor real esté conectado en red y responda a las peticiones con el mismo formato que utilizamos en el simulador.

2.3.2 USVs

Los USVs se han modelado a nivel de unidades hardware, véase figura 5. En este modelo se contempla la unidad de potencia, unidad de comunicación, actuadores, sensores y un procesador. Los sistemas están interconectados por tres tipos de buses: potencia, datos y comunicaciones. El módulo de comunicaciones se conecta a través de un canal inalámbrico con la capa *Fog*. El formalismo DEVS permite emitir eventos con destino o en modo Broadcast, por lo que ambos modos se utilizan en el simulador.

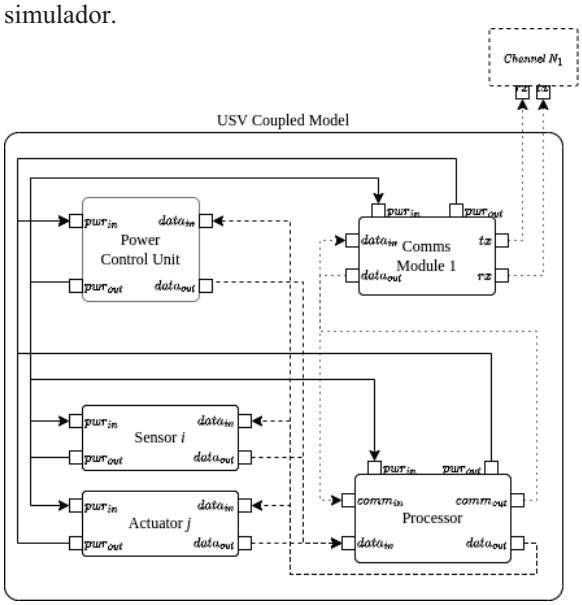

Figura 5: Arquitectura DEVS de los USVs.

Cada USV simulará su consumo energético en función de las unidades activas y su estado. Pudiendo en ciertos casos disparar eventos de Power Warning, que abortarán la misión para volver a puerto.

Cada USV pude estar especializado, albergar diferentes tipos de sensores, actuadores, algoritmos a bordo del procesador e incluso sistemas de comunicación.

En la figura 6 se muestra el informe de una simulación de barrido de una zona que han realizado dos barcos en modo cooperación. En este caso los barcos disponen de algoritmos de guiado y navegación básicos para barrer una zona planificada. Durante la simulación se realizan medias con los sensores descritos anteriormente.

Esta simulación incluye la navegación, control y dinámica de los barcos, el consumo energético, la toma de muestras utilizando la versión *RemoteEEMSSensor* y las comunicaciones con la capa *Fog*. Para integrar los elementos dinámicos se ha utilizado un integrador de Runge-Kutta.

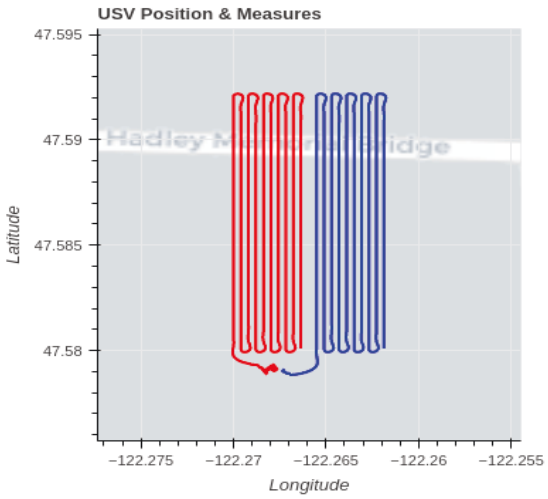

Figura 6: Simulación de barrido de una zona utilizando 2 USVs (Cruzan por debajo de un puente).

3 Conclusiones y próximos pasos

Actualmente ya se dispone de un entorno EIoT para realizar simulaciones de monitorización de Blooms de cianobacterias. Esta infraestructura ha sido diseñada bajo el formalismo DEVS, lo que nos permite trabajar con eventos y modelos. Gracias a este formalismo el diseño y verificación se puede hacer de forma simulada, para finalmente realizar el despliegue del sistema real de forma gradual.

El entorno contempla las típicas capas del IoT: Cloud, Fog y Edge. Gran parte del contenido de estas capas está implementado de forma simulada. Esto

incluye la gestión de ficheros de entrada y de salida, la representación gráfica para los informes y diferentes unidades típicas del caso al que se aplica.

La simulación de sensores está muy avanzada. Actualmente se simulan sensores de: temperatura, radiación solar, oxígeno, nitrógeno, fósforo, batimetrías, temperatura del agua, etc. Además, muchos de estos sensores están basados en ficheros generados con simuladores de aguas aceptados en el sector, que han sido montados como servicios en red, por lo que son fácilmente sustituibles por sensores reales.

La simulación de los barcos también está muy avanzada, pues se están simulado al nivel de hardware. Se simulan unidades hardware, sus consumos de potencias, el intercambio de datos por buses y los sistemas de comunicaciones. Actualmente disponen de un procesador que permite ejecutar algoritmos de guiado y navegación, y solicitar medidas de los sensores.

Actualmente se está trabajando en una consola de comandos, que permita lanzar simulaciones parciales de diferentes unidades y subsistemas. En un futuro cercano se introducirán sensores reales dentro de la simulación, para posteriormente ir sustituyendo las diferentes unidades simuladas por elementos reales.

Agradecimientos

Este trabajo ha sido financiado a través de los proyectos IA-GES-BLOOM-CM (Ref: Y2020/TCS6420) y AMPBAS RETOS-INVESTIGACIÓN (Ref: RTI2018-098962-B-C21).

English summary

DEVELOPMENT OF AN EIOT FRAMEWORK USING DEVS: CYANOBACTERIAL BLOOM ALERT AND MANAGEMENT SYSTEM.

Abstract

The Discrete Event System Specification (DEVS) is used to model, simulate, develop and deploy a project according to the Environmental Internet of Things (EIoT) paradigm. The software and hardware are designed and developed following a typical IoT architecture, with layers: Cloud, Fog, Edge and Things. As an application case, a system to monitor and control unmanned surface vehicles (USVs) that perform measurements in freshwater bodies to monitor and predict cyanobacteria blooms is presented. Modeling an EIoT project using DEVS enables a progressive design of the software/hardware, which in turn facilitates the coexistence of simulated and real elements. This model-based design allows the models to be refined through simulations, facilitating the final deployment of the real system.

Keywords: Simulation, Software deployment, Discrete event system specification (DEVS), Model-based design, Environmental Internet of Things (EIoT), Cyanobacteria Blooms, Water Quality.

Referencias

[1] Environmental Fluid Dynamics Code (EFDC+), https://www.eemodelingsystem.com/ee-modeling-system/efdc-plus/overview

[2] Flask Python package, https://docs.python.org/3/library/json.html

[3] ISCAR, (2021-2024) AI-GES-BLOOM-CM "Hacia un sistema Integral para la Alerta y GEStión de BLOOMs de cianobacterias en aguas continentales", Comunidad de Madrid (Y2020/TCS-6420), http://www.dacya.ucm.es/area-isa/investigacion/proyectos/competitivos.html

[4] JSON Python package, https://docs.python.org/3/library/json.html

[5] Meriluoto, J., Spoof, L., Codd, G.A., (2016) Handbook of cyanobacterial monitoring and cyanotoxin análisis, John Wiley & Sons Ltd, DOI:10.1002/9781119068761.

[6] Recomendación UIT-T Y.2060, (2012) Descripción general de Internet de los objetos, https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.2060-201206-I!!PDF-S&type=items

[7] Zeigler, B. P., A. Muzy, and E. Kofman, (2018), Theory of modeling and simulation: discrete event & iterative system computational foundations. Academic press, DOI:10.1016/C2016-0-03987-6.

https://doi.org/10.17979/spudc.9788497498418.0017