DISEÑO BASADO EN MODELOS DE APLICACIONES FOG COMO WORKFLOW DE MICROSERVICIOS

E. Hurtado*, A. López*, I. Sarachaga*, A. Armentia*, E. Estévez**, M. Marcos*
*Dpto. Ingeniería de Sistemas y Automática, UPV/EHU, España
{ekaitz.hurtado, alejandro.lopez, isabel.sarachaga, aintzane.armentia, marga.marcos}@ehu.eus
**Dpto. Ingeniería Electrónica y Automática EPS de Jaén, España
e-mail: eestevez@ujaen.es

Resumen

La irrupción de la Industria 4.0 ha dado lugar a integraciones con tecnologías innovadoras, las cuales permiten el desarrollo de aplicaciones que utilizan datos recogidos desde la planta para optimizar los procesos industriales. En un principio, estas aplicaciones se desplegaron en la nube. Sin embargo, por problemas de latencia y seguridad, la niebla ha surgido como un nuevo paradigma, con capacidades similares a la nube, pero más cerca de los activos de planta. El despliegue de aplicaciones en la niebla ha sido un tema de discusión creciente. No obstante, a pesar de que diferentes autores conciben las aplicaciones según la misma lógica (como un flujo dirigido de componentes), no se ha presentado una solución global. Así, los autores proponen un enfoque genérico basado en modelos para la definición de aplicaciones de computación en la niebla.

Palabras clave: Industria 4.0, Fog Computing, aplicación, meta-modelo, workflow, Node-RED.

1 INTRODUCCIÓN

La Industria 4.0 ha surgido en los últimos años como respuesta a las exigencias del mercado en términos de calidad y optimización. Uno de los pilares de la Industria 4.0 es generar valor añadido en el proceso de fabricación a partir de los datos generados por los activos de la planta (el conjunto de recursos de la fábrica a nivel de planta [6]).

Así, con el avance de la Industria 4.0, varias tecnologías se han integrado a este paradigma (e.g., sistemas multiagente (MAS), inteligencia artificial (IA), Internet de las cosas (IoT), computación en la nube, etc.) [15], habilitando el desarrollo de aplicaciones innovadoras relacionadas con la analítica inteligente, la planificación dinámica o el mantenimiento preventivo [7], [12], [15]. Se trata, por lo tanto, de aplicaciones que pueden tener cabida en ámbitos de aplicación dispares, por lo que pueden presentar requisitos dependientes del dominio. Por ejemplo, en las aplicaciones del ámbito de la sanidad, se deben garantizar unas latencias predefinidas, así como un acceso seguro [1], mientras que en aplicaciones industriales, otros requerimientos toman más importancia, como la viabilidad y la rentabilidad [2].

Desde un punto de vista estructural, la jerarquía de los sistemas de fabricación también se ha visto transformada en un modelo de interconectividad global que se vertebra a través de tres niveles: 1) la planta o edge, en la que los activos físicos (robots, máquinas, transportes) ofrecen los servicios de fabricación y transporte necesarios en el proceso productivo; 2) la niebla o fog, que ofrece recursos de computación y almacenamiento ubicados en un entorno cercano al origen de los datos; y 3) la nube o cloud, que ofrece unas funcionalidades equiparables a las de la niebla, con la diferencia de que tanto los recursos de procesamiento como de almacenamiento se encuentran en servidores externos, alejados del origen de los datos y sólo accesibles a través de Internet, aumentando la capacidad de cómputo y almacenamiento [1], [13], [15].

Debido a sus limitaciones en cuanto a capacidad de procesamiento y almacenamiento, los activos de planta no cuentan, por norma general, con recursos computacionales necesarios para soportar estas aplicaciones, por lo que, en primera instancia, estas aplicaciones se desplegaron en la nube [2]. Sin embargo, el despliegue de aplicaciones en la nube conlleva nuevos problemas, como la aparición de latencias y la vulnerabilidad en el acceso a los datos de las empresas [13]. Por ese motivo, dichas aplicaciones se han desplegado específicamente en uno de los niveles comentados anteriormente: el Fog.

El software disponible para las aplicaciones Fog se puede clasificar en dos categorías: framework y plataforma de software. La primera se refiere al conjunto de metodologías y herramientas para el desarrollo de este tipo de aplicaciones, mientras que la segunda se refiere al entorno que proporciona las funcionalidades y mecanismos para la orquestación de las aplicaciones [11]. Estas plataformas deben cumplir

https://doi.org/10.17979/spudc.9788497498043.701

una serie de características: flexibilidad, reconfigurabilidad, adaptabilidad, seguridad, escalabilidad, interoperabilidad y portabilidad, entre otras [1], [2], [15]. Por ello, diferentes instituciones han promovido la estandarización del paradigma Fog Computing, con enfoques como OpenFog [8] o FORA [10].

Por otro lado, las aplicaciones Fog suelen expresarse como un flujo dirigido en el que cada componente tiene entradas y/o salidas (workflow) [4], [5], [7], [9], [14]. Cada componente debe cumplir con una función específica, distribuyendo así la funcionalidad de la aplicación [7]. Este enfoque ofrece facilidad y flexibilidad en el desarrollo de las aplicaciones. Esto se debe a que el hardware, los protocolos y la funcionalidad se abstraen como nodos en el flujo, por lo que el diseño de la lógica de la aplicación se simplifica considerablemente [2]. De hecho, varios trabajos han aplicado técnicas de modelado para la definición de aplicaciones Fog siguiendo este enfoque orientado al flujo de datos. Sin embargo, ninguno de ellos ofrece herramientas para abordar la concepción, diseño y despliegue de este tipo de aplicaciones [3], [4], [9]. Es más, a pesar de los intentos de estandarización, por lo que los autores saben, el diseño y la composición de las aplicaciones suele resolverse con el desarrollo de soluciones ad-hoc para cada situación, no siendo aplicables a diferentes dominios de aplicación.

Por lo tanto, en este artículo se propone un proceso genérico para la definición y desarrollo de aplicaciones Fog pertenecientes a diferentes dominios. La propuesta está basada en el concepto de workflow y en el uso de técnicas de meta-modelado. Así, se permite definir y desarrollar la aplicación como una composición de servicios y de la que pueden generarse los componentes que la forman.

El resto del artículo está organizado de la siguiente manera: la sección 2 contiene el modelado de componentes y aplicaciones Fog propuesto. La sección 3 presenta una herramienta de diseño para este tipo de aplicaciones. La sección 4 describe el caso de estudio desarrollado en el ámbito de la Industria 4.0. Finalmente, la sección 5 resume las principales aportaciones del trabajo y presenta las líneas de trabajo futuro.

2 MODELADO DE APLICACIONES FOG

Como se ha comentado, las aplicaciones Fog suelen expresarse como un conjunto de componentes organizados en un flujo dirigido. Esta sección presenta la propuesta de los autores para el modelado de aplicaciones Fog que consta de dos meta-modelos: 1) uno para la definición de componentes Fog, los cuales, después de ser desarrollados, se almacenan en una biblioteca de componentes reutilizables y, posiblemente, de terceros; y 2) otro para la definición de las propias aplicaciones Fog. Estos meta-modelos tienen en cuenta las propiedades de los sistemas Fog Computing para satisfacer los requisitos de las aplicaciones.

2.1 META-MODELO PARA COMPONENTES FOG

La Figura 1 representa el meta-modelo para la definición de los componentes Fog. Los componentes deben ser de un tipo concreto (TComponent), siendo el número y variedad de tipos dependientes del dominio. Es decir, se crearán nuevos tipos de componentes, o nuevas funcionalidades de componentes existentes, cuando alguna aplicación necesite un nuevo servicio que no está desarrollado.

Para la comunicación entre los elementos Fog es esencial un mecanismo de identificación estandarizado. La nomenclatura es muy importante en Fog Computing para asegurar varios conceptos: la autenticación de la identidad, el control y la gestión de objetos, o la comunicación de datos, entre otros. Por ese motivo, se han incluido los conceptos de identificador (id) y de nombre. Por otro lado, en el atributo de la descripción se pueden añadir más detalles sobre el componente.

Los componentes ofrecen un conjunto de microservicios o funciones que pueden ejecutar por sí mismos. Estas funciones se caracterizan por el nombre y los parámetros de entrada y salida asignados. Asimismo, los parámetros se definen por su nombre y el tipo de dato (puede ser un dato simple o una estructura de datos).

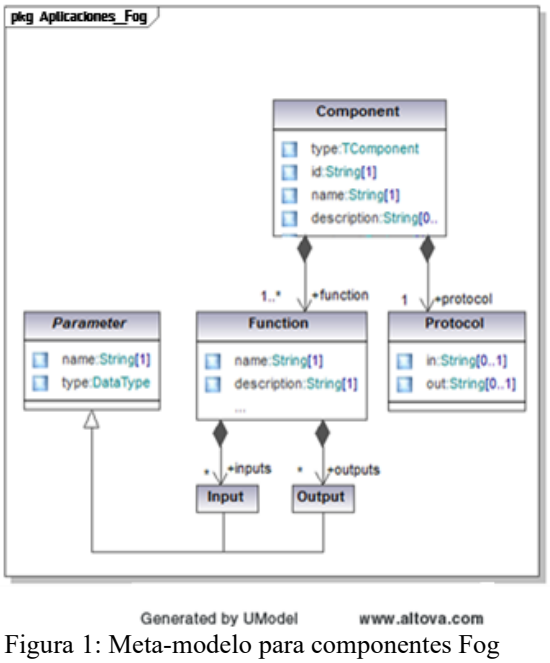

Figura 1: Meta-modelo para componentes Fog

https://doi.org/10.17979/spudc.9788497498043.701

Los componentes interaccionan entre sí, por lo que deben de tener la capacidad de enviar o recibir datos, o ambas cosas, para poder realizar sus funciones. Para garantizar el intercambio de datos, hay que especificar los protocolos de comunicación de entrada y/o salida soportados por el componente. Los protocolos de entrada y salida no tienen que ser necesariamente los mismos, pero si un componente envía datos a otro, sus protocolos (en este caso el de salida del remitente y entrada del receptor) sí deben ser idénticos.

2.2 META-MODELO PARA APLICACIONES FOG

Siguiendo el concepto de workflow antes mencionado, la Figura 2 recoge el meta-modelo para la definición de aplicaciones para Fog Computing como un conjunto de dos o más componentes (ComponentInstance) concatenados creando un flujo dirigido. De este modo, las aplicaciones sirven a un propósito específico ejecutando secuencialmente las funciones realizadas por sus componentes (Function). Por lo tanto, cuando un componente es instanciado en una aplicación Fog, se debe seleccionar el microservicio que se desee que ejecute, entre los proporcionados por dicho componente.

El direccionamiento de mensajes a través de los componentes es relevante para la aplicación ya que cada uno de ellos puede tener diferentes interfaces. La capacidad de comunicación de un componente depende de sus puertos de entrada y salida (Ports), los cuales definen las estructuras de datos que se espera enviar y recibir a través de ellos (Data). Así, el inicio

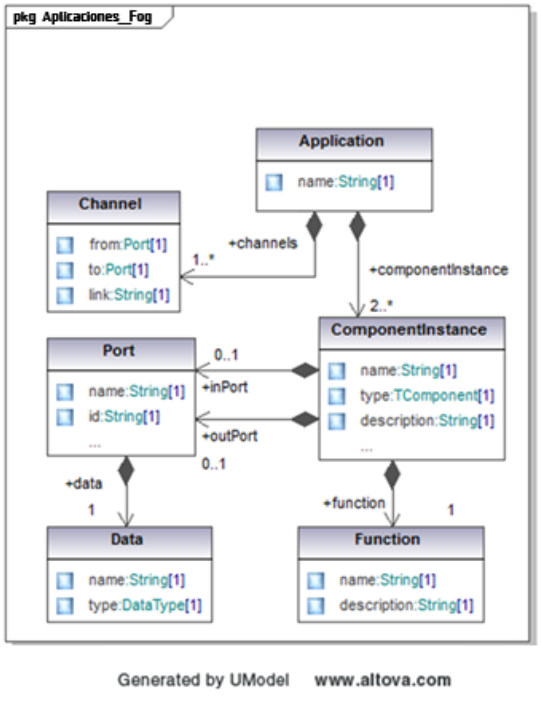

Figura 2: Meta-modelo para aplicaciones Fog

y el final del flujo quedará determinado por los componentes que sólo tienen un puerto de entrada o un puerto de salida, respectivamente.

La conexión entre el puerto de salida de un componente y el puerto de entrada del siguiente se especifica como canal de comunicación (Channel). Se debe definir un canal con el protocolo adecuado para cada conexión entre todas las instancias de componentes, y sus interfaces deben ser compatibles. De no ser así, las instancias de los componentes serán incapaces de comunicarse entre sí.

3 HERRAMIENTA DE DISEÑO DE APLICACIONES FOG

Dado que se recomienda el uso de herramientas de modelado para facilitar el diseño y la generación de aplicaciones Fog [5], esta sección ilustra la aplicabilidad de la herramienta Node-RED para el modelado de dichas aplicaciones. Esta herramienta permite diseñar aplicaciones de forma gráfica concatenando nodos para formar un flujo [4], por lo que encaja con la concepción de los autores sobre aplicación Fog.

Para el diseño de estas aplicaciones la herramienta de modelado debe incluir una librería que disponga de todos los tipos de componentes necesarios para cualquier aplicación que se desee crear. En este sentido, Node-RED proporciona plantillas predefinidas para algunos componentes [16], aunque también permite desarrollar nodos personalizados [5]. De este modo, es posible incluir componentes conformes al meta-modelo de componente propuesto en la sección anterior y así garantizar que todos ellos tengan la estructura correcta. Por lo tanto, Node-RED permite crear una librería de componentes Fog basados en modelos (librería de Fog Computing).

Un ingeniero software deberá ser el encargado de desarrollar y clasificar todos los tipos de componentes Fog existentes en la librería de Fog Computing. La Figura 3 presenta la vista de este ingeniero software en la herramienta Node-RED. A la hora de añadir nuevos componentes a la librería de Fog Computing se deberá seguir una serie de pautas: 1) Analizar qué tipo de componente se quiere añadir y comprobar si ya está disponible en la librería; 2) Especificar qué microservicios podrá ejecutar el nuevo componente; 3) Desarrollar el código fuente del componente que permita ejecutar todas sus funciones; 4) Crear la imagen base del componente y almacenarla en la librería.

El encargado de desarrollar las aplicaciones Fog tendrá otra vista de Node-RED, representada en la Figura 5, en la que creará el flujo añadiendo los nodos deseados. Al instanciar cada componente deberá

https://doi.org/10.17979/spudc.9788497498043.701

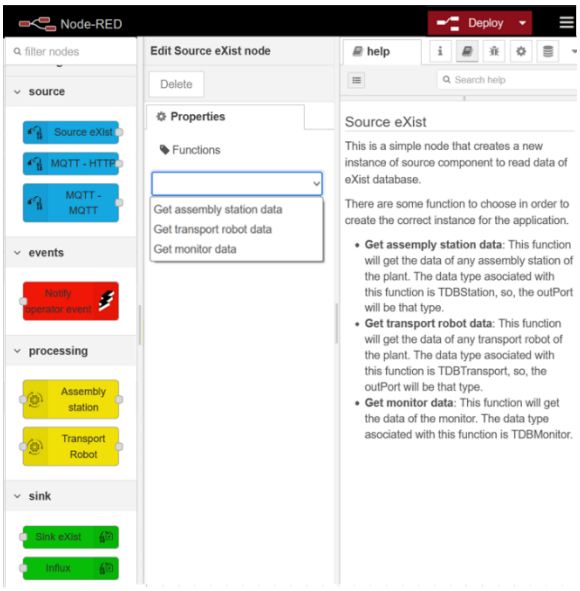

Figura 3: Vista de Node-RED para desarrollar la librería de Fog Computing

seleccionar una función entre las disponibles. A modo de ayuda, Node-RED permite añadir una descripción de cada componente y de sus funciones. En esta descripción se detallarán las estructuras de datos con las que trabajará cada función. El ingeniero software será el que añada estas descripciones al desarrollar los tipos de componentes.

4 CASO DE ESTUDIO

Una de las bases de la Industria 4.0 es tener la capacidad de obtener información útil a partir de los datos recogidos de la planta. Para ello, es necesario asegurar la comunicación entre la planta y la niebla, y almacenar los datos obtenidos para evitar pérdidas. También es necesario ser capaz de extraer información útil de los datos para tomar decisiones basadas en el estado actual del sistema. Por último, se requiere la capacidad de retroalimentar el proceso de fabricación con las acciones inferidas del análisis de la información.

Esta sección describe un caso práctico en el que se satisfacen dichas necesidades mediante aplicaciones Fog definidas conforme a los modelos propuestos. De este modo, se proponen dos aplicaciones: en primer lugar, una aplicación de adquisición capturará los datos de los recursos de fabricación de la planta y los almacenará en una base de datos basada en XML. En segundo lugar, una aplicación de procesamiento leerá los datos y los procesará para obtener la eficacia global de las estaciones. En este caso, calculará el Overall Equipment Effectiveness (OEE), el cual es un indicador de rendimiento. La misma aplicación gestionará la retroalimentación de la planta, activando un evento para avisar al operario en caso de que el OEE baje de un umbral predefinido, o si mantiene una tendencia negativa.

El sistema de fabricación utilizado en este caso de estudio, representado en la Figura 4, está estructurado en dos niveles: la planta y el Fog. Los recursos a nivel de planta consisten en dos estaciones de montaje interconectadas mediante robots de transporte. Estas estaciones generan datos que alimentan la entrada de las aplicaciones Fog, que capturan la información a través de MQTT. Además de estos recursos de fabricación, el sistema proporciona varios recursos a nivel Fog, que pueden ser utilizados y compartidos por las aplicaciones Fog activas (una base de datos XML, un bróker MQTT para la comunicación Planta-Fog y una base de datos de series temporales, entre otros).

4.1 DESARROLLO DE LA LIBRERÍA DE FOG COMPUTING

Para la generación de la librería de Fog computing se sigue la pauta indicada en el apartado 3. En primer lugar, se identifican los componentes necesarios y su funcionalidad. Por un lado, para la aplicación de adquisición se necesitan dos tipos de componentes: un cliente MQTT para obtener los datos de planta a través del protocolo MQTT sub/pub (source MQTT-HTTP) y un componente para leer/escribir en la base de datos

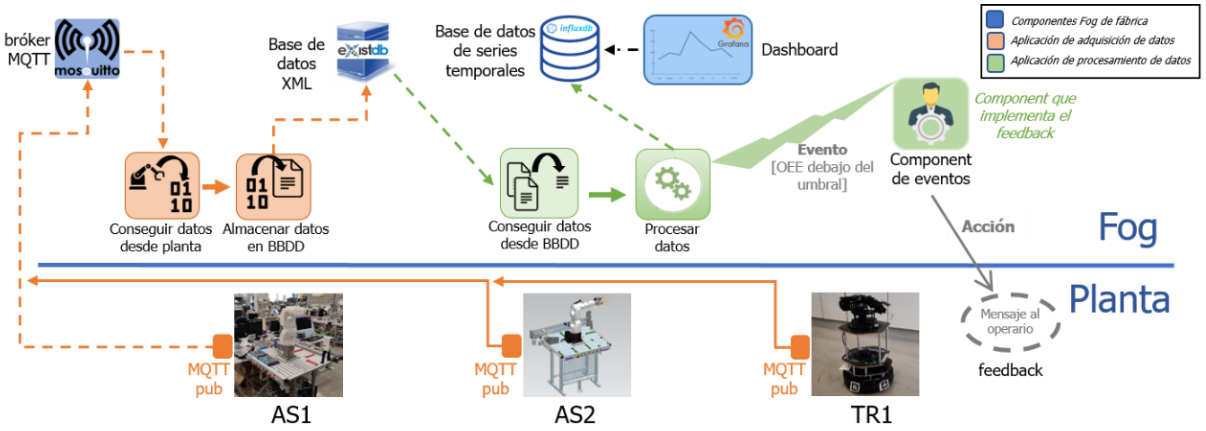

Figura 4: Perspectiva general de las aplicaciones de adquisición y procesamiento

https://doi.org/10.17979/spudc.9788497498043.701

XML (sink eXist). Por otro lado, la aplicación de procesamiento hará uso de un componente que lee de la base de datos (source eXist), otro para procesar el OEE (Processing assembly station) y, si es necesario, generar un evento al componente de eventos (Notify operator), que es el encargado de realimentar al operador de la planta.

A continuación, se desarrolla el código fuente que permite al componente realizar sus funciones. Los componentes desarrollados conforme al meta-modelo propuesto estarán encapsulados en contenedores Docker. Así, cada componente está aislado y es independiente de los demás, por lo que es posible actualizar y/o cargar sólo funciones/contenedores específicos de forma totalmente independiente.

Finalmente, se asocia cada componente a una imagen de contenedor que se utiliza para lanzar las instancias de componentes. Las imágenes de contenedores se definen mediante un archivo *dockerfile* usando una imagen base, que puede ser de terceros o construida desde cero [7]. Una vez creadas las imágenes base de los componentes se almacenarán en la librería de Fog Computing. Dichas imágenes contienen la funcionalidad de todos los microservicios que puede realizar el componente.

4.2 DESARROLLO DE LAS APLICACIONES FOG

El usuario de la herramienta de modelado, diseñará las aplicaciones como un conjunto de instancias de componentes de la librería de Fog Computing. Para cada una de las instancias deberá seleccionar una de sus funcionalidades. También debe definir el flujo de la aplicación mediante las conexiones entre las instancias. Por cada componente de aplicación se crea una nueva imagen de componente, a partir de su correspondiente imagen base del repositorio, mediante un archivo *dockerfile*.

La Figura 5 ilustra el uso de la herramienta Node-RED para la composición de las dos aplicaciones mencionadas anteriormente. Para las aplicaciones Fog de adquisición y procesamiento de datos del caso de estudio, la Figura 6 representa la secuencia que siguen los datos (workflow). Cada envío de datos se realiza con una estructura de datos especifica. Esta estructura será la misma en el puerto de salida del remitente y el puerto de entrada del receptor.

Los componentes de las aplicaciones Fog interactuarán con los componentes fog de fábrica, destacados con el color azul. Por ejemplo, el componente source eXist conseguirá los datos desde la base de datos eXist o el componente Assembly station Influx almacenará los cálculos de los OEE en la base de datos de series temporales Influx.

La aplicación de adquisición, destacada en naranja, va a interactuar de esta manera:

- Cuando un activo de planta haya finalizado alguna operación enviará los datos recogidos (tiempo inicio, tiempo final, etc…) vía MQTT al bróker situado en el Fog.
- El componente source MQTT-HTTP, conectado al bróker, recibirá estos datos, para enviárselos al elemento Sink eXist vía HTTP.
- Sink eXist tendrá la capacidad de gestionar la base de datos eXist, la cual está basada en XML, y en el caso de esta aplicación, procesará y almacenará los datos generados por los activos de planta.

Por otro lado, la aplicación de procesamiento, destacada en verde, será la encargada de analizar esos datos generados en el nivel de planta:

- El primer componente, source eXist, conseguirá los datos de la BBDD y se los enviará al segundo componente en una estructura de datos especifica (TDBStation).
- El segundo componente será el encargado de procesar esos datos, calculando el OEE. Cuando tenga el resultado, le enviará los datos al Sink Influx, para que puedan ser almacenados en la base de datos de series temporales. Después, analizará los resultados conseguidos.
- Si el OEE calculado está por debajo de un umbral definido para esa estación, es decir, si la eficiencia de la estación ha caído por debajo de valores asumibles, se pedirá al último elemento (Notify operator) que le

Figura 5: Vista de Node-RED para composición de la aplicación de procesamiento.

https://doi.org/10.17979/spudc.9788497498043.701

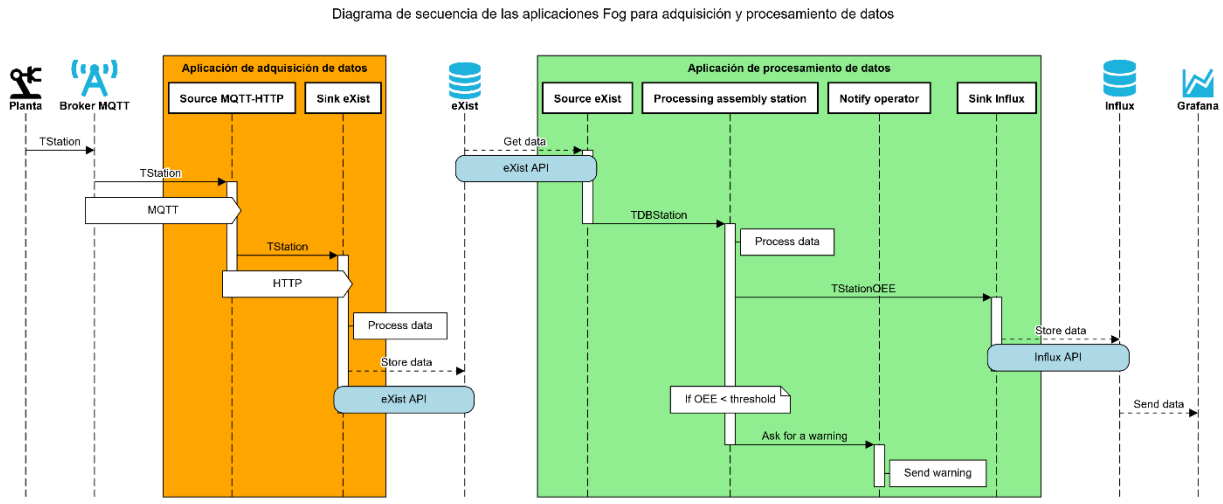

Figura 6: Diagrama de secuencia de las funcionalidades de las aplicaciones de adquisición y procesamiento

mande una advertencia al operario, avisándole del estado actual del sistema, para que pueda llevar a cabo acciones de corrección u optimización.

Además, se ha desarrollado un nuevo elemento, ajeno a esta aplicación, para visualizar gráficamente el rendimiento actual de la fábrica. Así, el elemento Grafana recogerá los cálculos de los OEE y los mostrará en una gráfica, junto con el umbral predefinido, para así poder ver en cualquier momento el estado actual de cualquier activo de planta.

La Figura 7 representa la definición de los archivos *dockerfile* de los componentes source, sink y processing de las aplicaciones desarrolladas. La secuencia de los datos representada en la Figura 6 se puede apreciar en los puertos de entrada y salida definidos en los *dockerfile*. Como se ha comentado anteriormente, estos puertos deben ser coherentes en cuanto al workflow. Por ejemplo, en la Figura 7 se puede ver que el puerto de salida del componente Source MQTT-HTTP coincide con el puerto de entrada del Sink eXist (TStation). Las estructuras de datos corresponden con las presentadas en el flujo de la Figura 6.

```
FROM source_mqtt_http_base_image
ENV function=assemblyStationData
ENV outPort=TStation
CMD python3 source.py
```

```
FROM source_exist_base_image
ENV function=getAssemblyStationData
ENV outPort=TDBStation
CMD python3 source_eXist.py
```

```
FROM sink_exist_base_image
ENV function=storeAssemblyStationData
ENV inPort=TStation
CMD java -jar sin_exist.jar
```

```
FROM processing_assembly_base_image
ENV function=processingOEE
ENV inPort=TDBStation
ENV outPort=TStationOEE
CMD java -jar processing.jar
```

Figura 7: Dockefiles de los componentes source HTTP-MQTT, sink eXist y processing assembly Station.

5 CONCLUSIONES Y TRABAJO FUTURO

En la actualidad, el diseño y composición de aplicaciones Fog se realiza principalmente de forma ad-hoc. En este artículo se hace uso de técnicas de meta-modelado para proponer un enfoque genérico con objeto de definir y desarrollar aplicaciones Fog como un flujo de componentes que implementan microservicios. En concreto se proponen dos meta-modelos: uno para los componentes Fog y otro para la definición de las aplicaciones en base a dichos componentes.

El meta-modelo para componentes Fog permite recoger las características principales de cada componente, siendo posible crear una librería con todos los tipos de componentes desarrollados y su imagen Docker asociada. El meta-modelo de las aplicaciones permite definir la interconexión de los componentes. Así, se consigue separar la definición de los componentes de la definición de la lógica de aplicación al mismo tiempo que se favorece la reutilización de las imágenes a la hora de desarrollar las aplicaciones.

Además, se ha probado que Node-RED es una herramienta útil para el diseño tanto de los componentes como de las aplicaciones.

No obstante, la aproximación no puede asegurar la correcta composición de las aplicaciones, ni proporciona el concepto de aplicación en tiempo de ejecución. Por ello, los próximos pasos se centran en el desarrollo de una arquitectura Fog que resuelva estas limitaciones y sea capaz de gestionar el ciclo de vida de las aplicaciones. Además, se debería garantizar la reactividad de las aplicaciones Fog para realimentar el proceso de fabricación.

https://doi.org/10.17979/spudc.9788497498043.701

Agradecimientos

Este trabajo está financiado por MCIU/AEI/FEDER, UE (proyecto RTI2018-096116-B-I00) y GV/EJ (proyecto IT1324-19).

English summary

MODEL-BASED DESIGN OF FOG APPLICATIONS AS MICROSERVICE WORKFLOWS

Abstract

The emergence of Industry 4.0 has led to integrations with innovative technologies, enabling the development of applications that use plant-level data to optimize industrial processes. Initially, these applications were deployed in the cloud. However, due to latency and security problems, the fog has arisen as a new paradigm, with similar capabilities to the cloud, but closer to the plant assets. Application deployment in the fog has been a topic of growing discussion. Still, although different authors conceive applications based on the same logic (as a workflow of components), no global solution has been presented. Thus, the authors propose a generic model-based approach for the definition of fog computing applications.

Keywords: Industry 4.0, Fog Computing, application, meta-model, workflow, Node-RED

Referencias

[1] Akrivopoulos, O., Chatzigiannakis, I., Tselios, C., Antoniou, A., (2017) "On the Deployment of Healthcare Applications over Fog Computing Infrastructure", *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*.

[2] Bellavista, P., Zanni, A., (2017) "Feasibility of Fog Computing Deployment based on Docker Containerization over RaspberryPi", *Proceedings of the 18th International Conference on Distributed Computing and Networking*.

[3] Dintén, R., López, P., Zorrilla, M., (2021) "Arquitectura de referencia para el diseño y desarrollo de aplicaciones para la Industria 4.0", *Revista Iberoamericana de Automática e Informática industrial.*

[4] Giang, N. K., Blackstock, M., Lea, R., Leung, V. (2015), "Developing IoT applications in the Fog: A Distributed Dataflow approach", *2015 5th International Conference on the Internet of Things (IOT)*.

[5] Giang, N. K., Lea, R., Leung, V., (2020) "Developing applications in large scale, dynamic fog computing: A case study", *Software: Practice and Experience*.

[6] Glossary, disponible en https://www.plattform-i40.de/PI40/Navigation/EN/Industrie40/Glossary/glossary.html.

[7] Hurtado, E., López, A., Armentia, A., Sarachaga, I., Casquero, O., Marcos, M., (2021) "On the development of Fog-Edge feedback applications", aceptado para presentación en *IEEE CASE 2021*.

[8] *IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing*.

[9] Kum, S. W., Moon, J., Lim, T.-B., (2017) "Design of Fog Computing based IoT Application Architecture", *2017 IEEE 7th International Conference on Consumer Electronics*.

[10] Pop, P., Zarrin, B., Barzegaran, M., Schulte, S., Punnekkat, S., Ruh, J., Steiner, W., (2021) "The FORA Fog Computing Platform for Industrial IoT", *Information Systems*.

[11] Puliafito, C., Mingozzi, E., Longo, F., Puliafito, A., Rana, O., (2019) "Fog Computing for the Internet of Things: A Survey", *ACM Transactions on Internet Technology*.

[12] Seitz, A., Henze, D., Miehle, D., Bruegge, B., Nickles, J., Sauer, M., (2018) "Fog Computing as Enabler for Blockchain-Based IIoT App Marketplaces - A Case Study", *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*.

[13] Stojmenovic, I., Wen, S., (2014) "The Fog Computing Paradigm: Scenarios and Security Issues", *2014 Federated Conference on Computer Science and Information Systems*.

[14] Taherizadeh, S., Stankovski, V., (2017) "Incremental Learning from Multi-level Monitoring Data and Its Application to Component Based Software Engineering", *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*.

https://doi.org/10.17979/spudc.9788497498043.701

[15] Wang, S., Wan, J., Li, D., Zhang, C., (2016) "Implementing Smart Factory of Industrie 4.0: An Outlook", *International Journal of Distributed Sensor Networks*.

[16] Xhafa, F., Kilic, B., Krause, P., (2020) "Evaluation of IoT stream processing at edge computing layer for semantic data enrichment", *Future Generation Computer Systems*.

© 2021 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution CC BY-NC-SA 4.0 license (https://creativecommons.org/licenses/by-ncsa/4.0/deed.es).

https://doi.org/10.17979/spudc.9788497498043.701