Una Aproximación Basada en Modelos para la Definición de Arquitecturas Asíncronas

Abel Gómez[1], Iker Fernandez de Larrea[2], Markel Iglesias-Urkia[2], Beatriz Lopez-Davalillo[2], Aitor Urbieta[2], and Jordi Cabot[1,3]

[1] Universitat Oberta de Catalunya (UOC), Spain
agomezlla@uoc.edu
[2] Ikerlan Technology Research Centre, Spain
{ifernandezdelarrea|miglesias|blopezdavalillo|aurbieta}@ikerlan.es
[3] ICREA, Spain
jordi.cabot@icrea.cat

Resumen En la nueva era del *Internet de las cosas* (IoT), nuestros objetos cotidianos se han convertido en los llamados *sistemas ciberfísicos* (CPS). El uso y despliegue de los CPS ha calado especialmente en la industria, dando lugar a la llamada *Industria 4.0* o *IoT Industrial (IIoT)*. Típicamente, las arquitecturas IIoT son distribuidas y asíncronas, estando la comunicación guiada por eventos como por ejemplo la publicación (y correspondiente suscripción) a mensajes. No obstante, las mejoras en escalabilidad y tolerancia al cambio de estas arquitecturas tienen sus desventajas, y es fácil que el conocimiento sobre los mensajes y su categorización (*topics*) se diluya entre los elementos de la arquitectura, dando lugar a problemas de interoperabilidad entre los agentes implicados. En este artículo, presentamos nuestra propuesta para automatizar el diseño e implementación de estas arquitecturas mediante técnicas basadas en modelos. Para ello nos apoyamos en *AsyncAPI*, una propuesta para la especificación de API dirigidas por mensajes.

Keywords: Publicación-Suscripción · Sistemas ciberfísicos · Arquitecturas asíncronas · AsyncAPI

1. Introducción

La aparición del *Internet de las cosas* [2] (*Internet of Things*, IoT), ha cambiado drásticamente la concepción de los objetos físicos en la sociedad y en la industria. En la nueva era del IoT, todo objeto se convierte en un sistema *ciberfísico* (*Cyber-Physical Systems*, CPS) [4] complejo, donde tanto las características físicas como el software que las gestiona se encuentran muy interrelacionadas entre sí. Así, en la actualidad, numerosos objetos cotidianos son de hecho CPS, que utilizan cada vez más sensores e interfaces (API) para interactuar e intercambiar datos a través de la nube [9].

La filosofía del IoT ha sido especialmente abrazada desde la industria, dando lugar al *IoT Industrial* (*Industrial IoT*, IIoT), o también llamada *Industria 4.0* [5]. Es aquí donde los CPS cobran especial relevancia, fundamentalmente en tareas de control y monitorización [6]. Con el objeto de alcanzar altos grados de escalabilidad, los CPS se despliegan típicamente en arquitecturas asíncronas dirigidas por eventos que mejoran el comportamiento global y la confiabilidad de los sistemas. Uno de los paradigmas más populares en

la actualidad es el llamado *publicador-suscriptor* [1] —que sigue, por ejemplo, el protocolo *Message Queuing Telemetry Transport*, MQTT— donde los mensajes que se envían desde y hacia los CPS no se dirigen a un determinado destinatario, sino que son publicados y consumidos proactivamente por los agentes implicados según determinados criterios o categorías. No obstante, aunque estas arquitecturas distribuidas son especialmente escalables y tolerantes al cambio, no están exentas de problemas: puesto que la comunicación se realiza entre iguales, debe existir un acuerdo entre todas las partes en cuáles son las categorías de mensajes esperados, así como su formato y estructura interna.

Ikerlan, como centro tecnológico, coordina en la actualidad distintos proyectos en los que desarrolla soluciones para sistemas de monitorización, control y supervisión de dispositivos IoT remotos en plantas de fabricación, almacenes o edificios inteligentes. Dichas soluciones deben dar soporte a entornos donde un gran número de dispositivos envían datos (e.g., información de sensores, datos procesados en lotes, etc). Siguiendo la tendencia actual, las soluciones desarrolladas se basan en arquitecturas asíncronas mediante *publicación-suscripción*. En este artículo presentamos una propuesta basada en modelos para diseñar dichas arquitecturas de forma eficiente, proporcionando soluciones que permitan atajar los problemas de cohesión típicos de dichas arquitecturas.

2. Arquitecturas Asíncronas

En los entornos IIoT, las necesidades de monitorización y control, así como los requisitos de seguridad y confiabilidad, posibilitan, e incluso hacen deseable, la reutilización de una arquitectura de referencia genérica. Dicha arquitectura se ejemplifica en la Figura 1, que muestra un caso típico en el que un conjunto de CPS —formado cada uno por un *dispositivo* y una *pasarela IoT*— publican datos de monitorización y estado. Dichos datos, cuyo volumen puede ser muy elevado, son consumidos por una aplicación en la nube que los filtra y procesa. Los datos procesados son mostrados a los usuarios del sistema a través de un *Frontend*. Estos usuarios pueden, a su vez, enviar mensajes de control para reconfigurar los CPS a través del *Frontend*. Como se puede observar, el elemento central de la arquitectura es el *broker de mensajería*, que se encarga de gestionar las publicaciones, suscripciones, y el flujo de mensajes entre los elementos de la red. En esta arquitectura, siguiendo el paradigma de publicación-suscripción, los mensajes no se envían directamente a los destinatarios que los consumirán, sino que se publican bajo una determinada categoría que recibe el nombre de *topic*. Solo los dispositivos suscritos a un determinado *topic* recibirán los mensajes publicados bajo este.

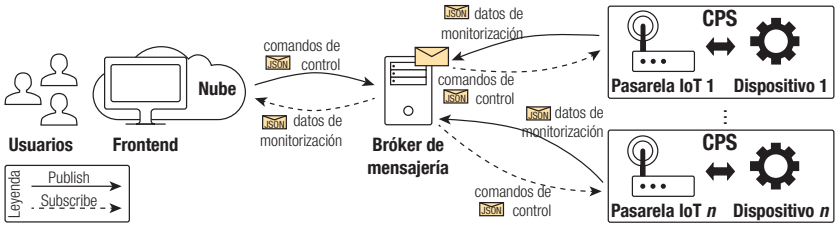

Figura 1. Arquitectura de referencia de un sistema IoT con comunicación asíncrona

Uno de los mayores problemas a los que se enfrentan este tipo de arquitecturas es la consistencia en la comunicación [3]: el formato de los mensajes intercambiados y los *topics* bajo los que se publican y suscriben han de mantenerse consistentes a lo largo de todo el ciclo de vida del sistema. De no cumplirse, podría producirse un mal funcionamiento del sistema: si cualquiera de los agentes introduce (mínimos) cambios en la definición de las comunicaciones, y dichos cambios no son propagados a todos los agentes implicados, se producen inevitablemente problemas de interoperabilidad. Así, la comunicación puede dejar de ser efectiva por dos razones: *(i)* que exista una **divergencia en los *topics*** bajo los que se publican los mensajes, dando lugar a que los agentes no reciban mensajes de interés; o *(ii)* que exista una **divergencia en el formato** de los mensajes de un determinado *topic*, y que por tanto estos no puedan ser comprendidos por los suscriptores que los reciban.

3. Estandarización y Automatización de Arquitecturas Asíncronas

Los problemas de interoperabilidad previos no son exclusivos de las arquitecturas asíncronas. De hecho, las arquitecturas síncronas también los manifiestan. No obstante, en este campo, la industria ya ha propuesto soluciones estandarizadas para soportar el desarrollo de mensajería síncrona. Un ejemplo es *OpenAPI* [8] y su completo ecosistema.

Para arquitecturas asíncronas, e inspirándose en *OpenAPI*, ha aparecido recientemente la propuesta *AsyncAPI* [7]. En *AsyncAPI*, las especificaciones de una API se pueden definir en YAML o JSON, y permiten especificar, por ejemplo, los brókers de mensajería, los *topics* de interés, o los diferentes formatos de los mensajes asociados a cada uno de los *topics*, entre otros aspectos. *AsyncAPI* se encuentra, sin embargo, en etapas tempranas de desarrollo, y por tanto, su utillaje se encuentra infradesarrollado, limitándose, fundamentalmente, a la generación de documentación para ser consumida por humanos.

Pese a sus actuales limitaciones, no obstante, en este artículo apostamos por el empleo de *AsyncAPI* para la definición de las API asíncronas, complementando su uso con una aproximación basada en modelos que permite cubrir las carencias del actual ecosistema. De este modo, la API especificada para una arquitectura es el único *single source of truth* que se comparte entre todos los agentes implicados.

La Figura 2 muestra nuestro actual flujo de trabajo que permite la definición de arquitecturas asíncronas mediante *AsyncAPI*. Este flujo de trabajo está parcialmente soportado por nuestro *AsyncAPI Toolkit*[4]. Partiendo de la **especificación de *AsyncAPI*** disponible en [7], hemos desarrollado una **gramática** en Xtext[5] que valida definiciones de APIs asíncronas conformes a la especificación. Estas definiciones pueden crearse de forma asistida en el editor que se deriva automáticamente de la gramática Xtext. Igualmente, a partir de dicha gramática hemos obtenido el **metamodelo** equivalente para *AsyncAPI*, de tal forma que una **especificación de una API** asíncrona (en formato JSON) se puede representar de forma automática y directa como un **modelo** conforme al metamodelo de *AsyncAPI*. Por último, dada la disponibilidad tanto de un metamodelo para *AsyncAPI*, como de la especificación de una API como un modelo conforme, podemos explotar todo el conjunto de herramientas disponible en el ecosistema de *Eclipse*

[4] `https://github.com/SOM-Research/asyncapi-toolkit`
[5] `https://www.eclipse.org/Xtext`

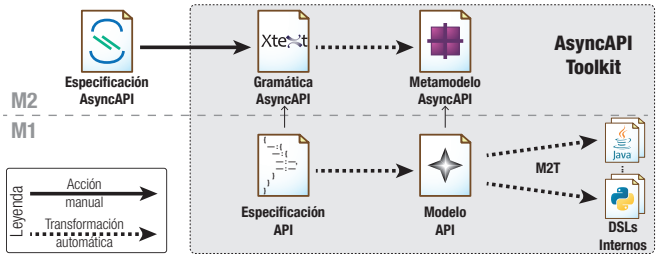

Figura 2. Propuesta para la definición de arquitecturas asíncronas basada en modelos

Modeling Framework (EMF)[6] para generar el código que será desplegado en cada uno de los agentes participantes. Es en esta dirección donde se enfocan actualmente nuestros principales esfuerzos, y más específicamente, en la generación de una serie de lenguajes específicos de dominio (*Domain Specific Languages*, DSL) internos que asistan a los desarrolladores en la creación de mensajes bien formados.

Acknowledgment This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No. 737494. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation program and from Sweden, France, Spain, Italy, Finland & Czech Republic.

Referencias

1. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys Tutorials **17**(4), 2347–2376 (Fourthquarter 2015)
2. Evans, D.: The internet of things: How the next evolution of the internet is changing everything. CISCO white paper **1**(2011), 1–11 (2011)
3. Iglesias., A., Iglesias-Urkia., M., López-Davalillo., B., Charramendieta., S., Urbieta., A.: TRILATERAL: Software Product Line based Multidomain IoT Artifact Generation for Industrial CPS. In: Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD. pp. 64–73. SciTePress (2019)
4. Jazdi, N.: Cyber physical systems in the context of industry 4.0. In: 2014 IEEE International Conference on Automation, Quality and Testing, Robotics. pp. 1–4 (May 2014)
5. Kagermann, H., Helbig, J., Hellinger, A., Wahlster, W.: Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group. Forschungsunion (2013)
6. Leitão, P., Colombo, A.W., Karnouskos, S.: Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. Computers in Industry **81**, 11–25 (2016)
7. Méndez, F., et al.: AsyncAPI, URL: `https://www.asyncapi.com/`. Acceso: abril 2019
8. OpenAPI Initiative: OpenAPI Specification, URL: `https://github.com/OAI/OpenAPI-Specification`. Acceso: abril 2019
9. Schwab, K.: The Fourth Industrial Revolution. Crown Publishing Group (2017)

[6] `https://www.eclipse.org/emf`