INTEGRACIÓN END-TO-END A TRAVÉS DEL MODELO DEL PRODUCTO 4.0

I. Sarachaga
Departamento de Ingeniería de Sistemas y Automática, Escuela de Ingeniería de Bilbao, UPV/EHU
isabel.sarachaga@ehu.eus

A. Burgos, N. Iriondo, M.L. Alvarez, M. Marcos
Departamento de Ingeniería de Sistemas y Automática, Escuela de Ingeniería de Bilbao, UPV/EHU
arantzazu.burgos@ehu.eus, nagore.iriondo@ehu.eus, marialuz.alvarez@ehu.eus, marga.marcos@ehu.eus

Resumen

La digitalización es una necesidad acuciante en el entorno industria para todas aquellas empresas que persiguen avanzar hacia Industry 4.0. En este trabajo se presenta una propuesta para la digitalización de la primera fase del ciclo de vida del producto correspondiente al diseño y fabricación en el entorno de la empresa, que permite la integración con proveedores y clientes. Este enfoque adopta los estándares propuestos en el modelo RAMI 4.0 y el modelo de componente I4.0. La solución tecnológica empleada en el prototipo utiliza servicios web con API REST y bases de datos distribuidas XML. Este prototipo se ha empleado en el caso de estudio para mostrar la interoperabilidad de las aplicaciones a través del ciclo de vida del producto. Concretamente, se muestra el intercambio de información entre el planificador y el modelo del producto.

Palabras clave: Integración End-to-End, Ciclo de vida del producto, Servicios Web, Industry 4.0.

1 INTRODUCCION

En el entorno industrial, la digitalización no es una elección; se trata de una necesidad acuciante para todas aquellas empresas que pretenden estar en sintonía con Industry 4.0. Dada su importancia, se incluye en una de las cinco dimensiones del índice de Economía y Sociedad Digital (DESI - Digital Economy and Society Index) que emplea la Comisión Europea para analizar la evolución de los estados miembros en materia de competitividad digital. Su informe de 2019 [2] recoge que grandes empresas en las que se emplean soluciones ERP (Enterprise Resource Planning) tiene mayor incidencia en el indicador asociado con intercambio electrónico de información, mientras que las pymes son más activas en el indicador referente a las redes sociales al apostar por el comercio electrónico. También destaca los servicios en la nube y la gestión de las relaciones con los clientes como posibles oportunidades tecnológicas.

Pero Industry 4.0 no se reduce al intercambio electrónico de información; persigue la digitalización del ciclo de vida del producto completo [1], el cual expresa cómo se diseña, fabrica, entrega, usa, mantiene y finalmente se recicla o elimina el producto.

En este sentido, los modelos a tener presentes son la arquitectura de referencia tridimiensinal RAMI 4.0 (Reference Architectural Model Industrie 4.0) y el modelo de componente I4.0 [12].

Por un lado, RAMI 4.0 propone: 1) la norma IEC 62890 como modelo de datos consistente para todo el ciclo de vida del producto distinguiendo entre tipo e instancia, 2) las normas IEC 62264 e IEC 61512 como jerarquía funcional para todos los componentes de la Industria 4.0, y 3) un modelo de capas que permite integrar diferentes tecnologías para representar los componentes desde distintos puntos de vista.

Por otro lado, el modelo de componente I4.0 ayuda a crear componentes individuales según la norma IEC 62832. Cada componente dispone de un *asset* o elemento con valor en la empresa, y de una capa de administración específica con las funciones a ejecutar por o sobre el asset, así como otra información que precise para estar activo y comunicarse con la empresa digital.

Por tanto, los productos inteligentes constan de componentes físicos, componentes inteligentes y componentes que permiten la conectividad, siendo estos dos últimos los que permiten añadir nuevas funcionalidades al producto de forma escalonada desde una monitorización inicial hasta conseguir la autonomía final, pasando por el control y la optimización [22].

En este trabajo se presenta la digitalización de la primera fase del ciclo de vida del producto correspondiente al diseño y fabricación en el entorno

de la empresa, teniendo en cuenta no sólo los estándares propuestos en el modelo RAMI 4.0 y el modelo de componente I4.0, sino también la colaboración entre distintas disciplinas, el desarrollo distribuido globalmente y la integración de proveedores y clientes. Como solución tecnológica para acometer esto retos, se propone utilizar servicios web empleando el estilo arquitectónico REST y bases de datos distribuidas XML.

La estructura del artículo es la siguiente: en el apartado 2 se realiza la revisión de trabajos relacionados tanto con el impacto de la digitalización en el ciclo de vida del producto para establecer los retos actuales, como con las soluciones tecnológicas que se pueden emplear en su implementación. En el apartado 3 se perfilan los requisitos de los nuevos sistemas de gestión del ciclo de vida del producto y se propone un nuevo enfoque en el marco del modelo RAMI 4.0. El apartado 4 presenta el prototipo que se ha empleado en la prueba de concepto descrito en el apartado 5. Finalmente, las conclusiones se exponen en el apartado 6.

2 TRABAJOS RELACIONADOS

El análisis del impacto de la digitalización en el ciclo de vida del producto ha sido objeto de numerosas revisiones bibliográficas, de congreso monográficos y de encuestas a empresas.

En [23] se analiza en detalle el impacto de la digitalización en las tres fases del ciclo de vida del producto: una primera fase de diseño y fabricación en el entorno de la empresa; una segunda fase de operación/servicio y soporte (reparaciones y mantenimiento) en el entorno del cliente; y una tercera fase correspondiente al fin de la vida útil en la que la reutilización y el reciclaje deben minimizar los impactos medio-ambientales. En este sentido, en [8] se proponen distintas eco-estrategias a incorporar en las distintas fases del ciclo de vida del producto.

En [21] se analizan 71 modelos de ciclo de vida de productos de mayor o menor complejidad según la finalidad que tengan en cada trabajo. El objetivo consiste en identificar los elementos y estructuras comunes en estos modelos para posteriormente modelarlos con el lenguaje de modelado Lifecycle Modelling Language (LML) [11] en varios casos de estudio. La decisión de emplear dicho lenguaje limitó el número de herramientas software a usar para dar soporte al modelo de ciclo de vida y además permitió poner de manifiesto las deficiencias que presenta dicho lenguaje[22].

En [15] se identifican las principales deficiencias de las soluciones IT actuales para la gestión del ciclo de vida del producto (Product Lifecycle Management - PLM): 1) las soluciones propietarias en los modelos de datos subyacentes y los enlaces de información; 2) la dificultad de integrar desarrollo de terceros; 3) y la gestión manual de los datos por parte de los usuarios. Con objeto de solventar estas deficiencias, estos autores proponen un marco holístico conceptual para PLM y un modelo escalonado para soluciones PLM inteligentes en los que se precisa el uso de estándares industriales, los interfaces abiertos y los metamodelos de datos.

El análisis de los trabajos relacionados permite indicar que los desafíos incluyen no sólo la gestión del ciclo de vida del producto y la colaboración entre distintas disciplinas, sino también el desarrollo distribuido globalmente y la integración de proveedores y clientes. En este sentido, los estándares y guías son claves para hacer uso de buenas prácticas y desarrollar sistemas que se puedan integrar.

En cuanto a las soluciones tecnológicas que permiten acometer estos desafíos, destacan CORBA (Common Object Request Broker Architecture), RMI (Java Remote Method Invocation), DCOM (Distributed Component Object Model) y los más recientes Servicios Web (SW) [5]. Las tres primeras tecnologías causan problemas de seguridad y compatibilidad al estar ligadas a un cierto lenguaje o plataforma de desarrollo, son incompatibles entre sí, transportan los datos mediante protocolos propietarios y en formato binario, y son complejas de implementar [6]. Sin embargo, las tecnologías basadas en SW permiten la intercomunicación dinámica de procesos a nivel de aplicaciones utilizando la web como canal de comunicación común y estándares abiertos, un ejemplo claro se muestra en [18]. Así, los SW permiten la integración de diferentes sistemas de información que pueden basarse en diferentes plataformas, lenguajes de programación y/o tecnologías.

El trabajo presentado en [17] destaca SOAP (Simple Object Access Protocol) y REST (Representational State Transfer) como las soluciones más extendidas para la implementación de los SW. Estas dos soluciones se analizan en los siguientes sub-apartados.

2.1 SOAP (Simple Object Access Protocol)

SOAP [19] es un protocolo para el intercambio de información estructurada en un entorno descentralizado y distribuido que utiliza XML y diferentes protocolos para la comunicación (HTTP, SMTP o FTP). SOAP consta de tres partes:

- Un **sobre** (envelope) que define qué hay en el mensaje y cómo procesarlo. Se trata de un documento XML que contiene cero o varias cabeceras y un cuerpo. Las cabeceras transportan

información de control, por ejemplo, atributos de calidad de servicio, mientras que el cuerpo contiene la identificación del mensaje y sus parámetros. El sobre es el elemento raíz del documento XML y tanto las cabeceras como el cuerpo son elementos hijo del sobre.
- Un **conjunto de reglas de codificación** para expresar instancias de tipos de datos. SOAP define un esquema para los tipos de datos independiente del lenguaje de programación.
- Una **convención** para representar llamadas a procedimientos y respuestas.

En los servicios SOAP el proveedor de servicios publica una descripción del servicio o una interfaz en un registro de servicios, que permite al consumidor encontrar el servicio adecuado.

Hay tres tipos principales de nodos SOAP:

- Emisor SOAP: genera y envía un mensaje SOAP.
- Receptor SOAP: recibe, procesa el mensaje SOAP, y puede generar una respuesta (mensaje o error) SOAP como resultado.
- Intermediario SOAP: es un emisor y receptor SOAP. Recibe y procesa los encabezados de los mensajes SOAP dirigidos a él y reenvía los mensajes SOAP hacia otro receptor SOAP.

2.2 REST (Representational State Transfer)

REST es un estilo arquitectónico basado en los principios que hacen escalable la World Wide Web (WWW) [4]. Para transmitir datos e informar al servidor de las operaciones a realizar sobre los mismos utiliza el protocolo HTTP [10]. Estos datos pueden estar en cualquier formato (XML, JSON, o cualquier tipo MIME). Por tanto, el acceso a los servicios utiliza estándares conocidos sin agregar nuevos protocolos a la pila de comunicaciones.

En los mensajes REST se destacan tres elementos fundamentales: recursos, verbos y representación. El cliente REST puede navegar a través de los recursos, siguiendo enlaces de recurso a recurso. Cada recurso se identifica con una URI (Uniform Resource Identifier) única en la plataforma web.

Los verbos HTTP (POST, GET, PUT, DELETE, UPDATE, etc.) indican qué acción se debe realizar sobre el recurso: GET para recuperar, POST para crear, PUT para modificar, DELETE para eliminar, … un recurso.

REST soporta todos los formatos sin ninguna restricción para la representación de los datos. En función de la capacidad del cliente y del servidor para trabajar con los formatos, puede utilizar JSON, XML o cualquier otro formato MIME.

2.3 REST versus SOAP

En [5], [20], [17], [13] y [9] se realizan comparaciones entre REST y SOAP. La Tabla 1 es un compendio de estas las comparativas.

Desde el punto de vista de la implementación, todos los autores están de acuerdo que los sistemas basados en REST son más sencillos que los sistemas basados en SOAP, y en consecuencia presentan una curva de aprendizaje más pequeña. Además, siendo REST una arquitectura que sigue el estilo de la WWW, se prevé que en los próximos años dominará el espacio tecnológico. (Muehlena, Nickersona y Swensonb 2005) pone de ejemplo a la plataforma Amazon.com, que proporciona ambos servicios, en la que el 85% de los desarrolladores prefieren usar la interfaz REST.

Otra diferencia destacable es que el cliente y el servidor REST están débilmente acoplados. Esta característica hace a los sistemas escalables, al permitir modificar los servicios sin realizar ningún cambio en el cliente. Sin embargo, existe un fuerte acoplamiento entre el cliente y el servidor SOAP, provocando que cualquier cambio en el servicio implique cambios en el cliente.

La Figura 1 muestra las diferencias que presentan ambas soluciones cuando usan el protocolo HTTP. REST lo utiliza no sólo como herramienta de comunicación en la web, sino que además identifica la acción a realizar sobre el recurso.

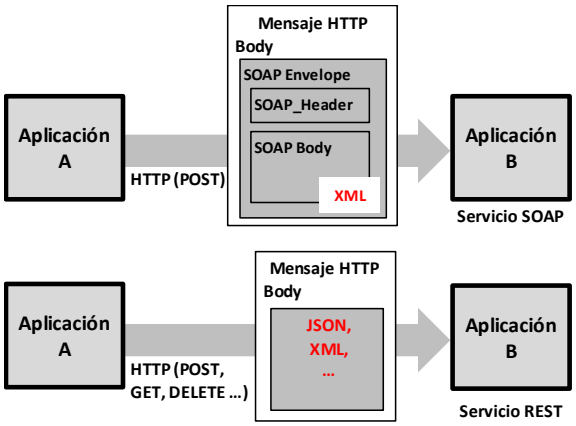

Figura 1: HTTP en SOAP y REST

También destacar que mientras los datos en SOAP se limitan a documentos XML, en REST se pueden utilizar una gran variedad de formatos.

La visión de futuro, la facilidad de aprendizaje, el permitir diferentes formatos para la información y la escalabilidad ha determinado la elección de la tecnología REST.

Tabla 1: SOAP vs. REST

SOAP	REST
Protocolo ligero de acceso a objetos (SOAP)	Estilo arquitectónico
Bien conocida, tecnología tradicional	Nueva en comparación con SOAP
La interacción cliente-servidor está estrechamente acoplada	La interacción cliente-servidor está acoplada débilmente
Un cambio en el servicio a menudo implica un cambio de código complicado en el cliente	El cambio de un servicio no requiere ningún cambio en el código del cliente
Usa HTTP, SMTP, FTP,…	Utiliza HTTP
Con el protocolo HTTP utiliza el método POST y requiere que se cree una solicitud XML compleja en el cuerpo del mensaje	Utiliza los métodos de HTTP (POST, GET, UPDATE, etc.) y las URI como identificadores de recursos
Siempre devuelven datos XML	Flexibilidad en el tipo de datos que devuelve
Más difícil de desarrollar, requiere herramientas	Mucho más sencillo de implementar
Utiliza nuevos protocolos para definir mecanismos de seguridad	Utiliza los mecanismos de seguridad integrados en HTTP
Más adecuado para transporte de gran cantidad de datos	Las solicitudes REST no son adecuadas para una gran cantidad de datos (especialmente GET)

3 INTEGRACIÓN END TO END

La integración end-to-end (E2E) persigue la interoperabilidad real de todas las herramientas de ingeniería utilizadas en las diferentes fases del ciclo de vida del producto, garantizando la integridad de los datos y todos aquellos aspectos relacionados con la seguridad.

A lo largo del ciclo de vida de un producto numerosas aplicaciones interactúan directamente con el modelo del producto y de manera indirecta entre ellas, añadiendo, modificando y compartiendo información del producto. Por un lado, estas aplicaciones tienen fines muy diferentes y se pueden ejecutar en plataformas heterogéneas y en localizaciones muy dispares. Por otro lado, la naturaleza de la información asociada con el producto puede ser muy variada, desde una imagen, un documento CAD/CAM/CAE, un fichero de texto, los modelos asociados a la norma IEC 61512 (modelo de recetas, modelo físico, modelo de procedimiento, modelo entidad equipo, modelo de estados), ...

Por tanto, los requisitos que se persiguen para acometer la integración E2E en este trabajo son los siguientes:

- Proporcionar soporte a las funciones de empresa en base a servicios, precisando servicios internos para garantizar la integración E2E y servicios externos que integren a los clientes y proveedores.
- Gestionar la gran cantidad de datos digitales que se generan a lo largo del ciclo de vida del producto.
- Disponer de modelos de datos para dar soporte a la integración e interoperabilidad de aplicaciones multidisciplinares distribuidas.
- Permitir añadir nuevas aplicaciones y funcionalidades sin afectar a las aplicaciones que utilizan los servicios.
- Integrar las aplicaciones discretas heredadas.

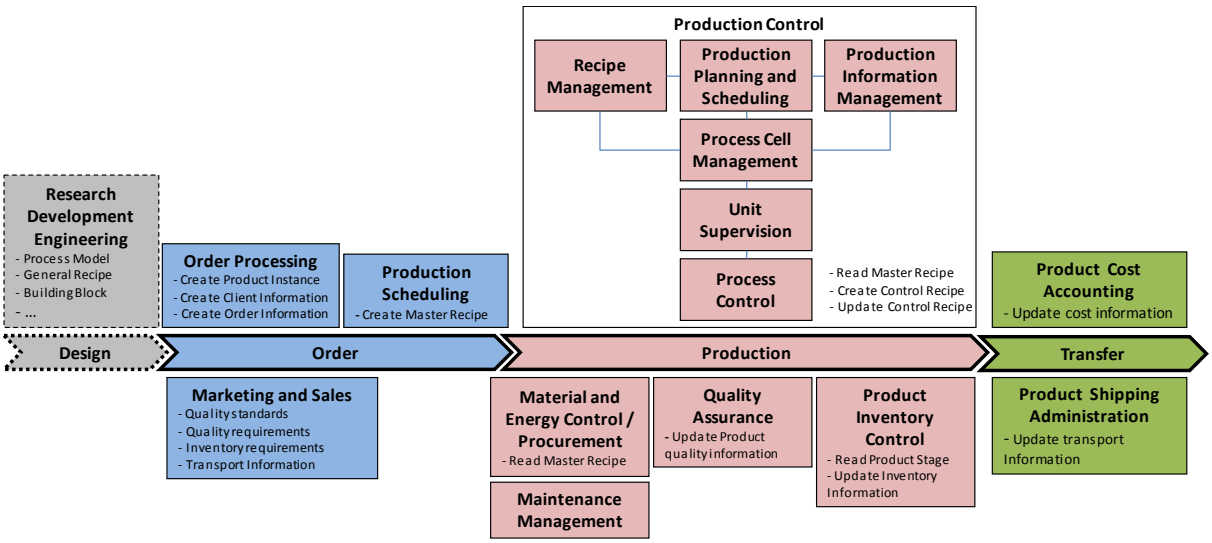

Figura 2: Información del producto en el ciclo de vida

Además, los estándares y guías son claves para hacer uso de buenas prácticas y desarrollar sistemas que se puedan integrar. Por ello se han considerado las normas propuestas por el modelo RAMI 4.0. Concretamente, la norma IEC 62264 establece las funciones empresariales y determina la información que se intercambian entre las mismas, facilitando la integración de las funciones empresariales y los sistemas de control de producción, mientras que la norma IEC 61512 proporciona los modelos de referencia para definir los requisitos de control de las plantas de fabricación.

Con objeto de identificar las funciones de la capa de administración del modelo del producto que van a ser solicitadas por cada una de las funciones de empresa, se ha identificado qué información del producto lee, añade o actualiza cada una de las funciones identificadas en el modelo funcional propuesto por la norma IEC 62264. Además, en el caso de la función correspondiente al Control de Producción se ha considerado el Modelo de Actividad de Control de la norma IEC 61512 para establecer las actividades de control y sus relaciones. La figura 1 recoge una sinopsis de este procedimiento.

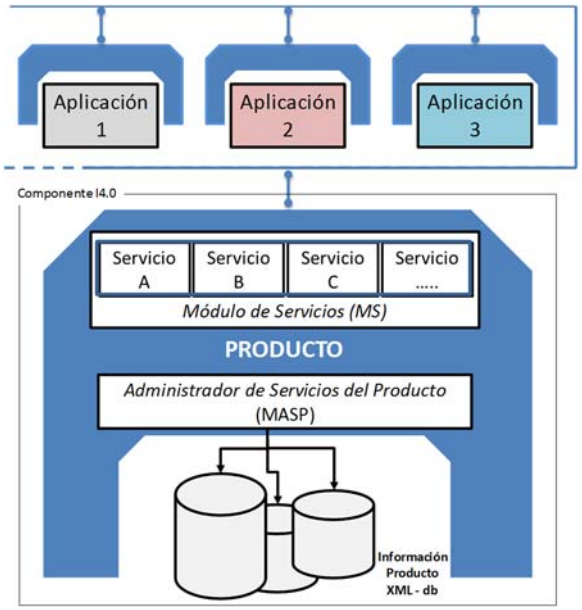

Figura 3: Prototipo del componente Producto 4.0

4 PROTOTIPO

El prototipo se ha desarrollado siguiendo los consejos y buenas prácticas obtenidas de [16] y [7].

De manera simplificada, el producto se ha definido como un documento XML con un conjunto de metadatos y referencias a todos aquellos modelos, documentos y ficheros relacionados con el mismo a lo largo de su ciclo de vida. Se ha seleccionado la base de datos eXist (eXistdb 2018) para organizar y almacenar la información del producto de forma distribuida. Esta base de datos, nativa XML, permite manipular directamente estos documentos y su API REST proporciona todas las utilidades requeridas para la gestión de los repositorios.

En la Figura 3 se muestra de manera esquemática el prototipo del producto. El producto consta de dos módulos: *Módulo Administrador de Servicios del Producto* (MASP) y *Módulo de Servicios* (MS).

El MASP del producto es el intermediario entre los servicios y la base de datos. Este módulo hace uso de la API REST que proporciona eXist db.

El MS o *API REST del Producto* ofrece los servicios CURD (Create – Update – Read - Delete) a las aplicaciones para acceder a la información del producto, como se muestra en la Figura 4.

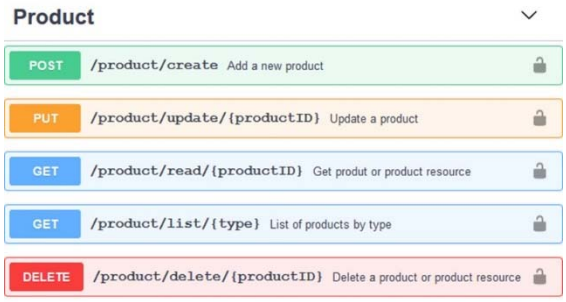

Figura 4: Servicios básicos del producto

5 CASO DE ESTUDIO

Para mostrar la interoperabilidad de las aplicaciones a través del ciclo de vida del producto, se presenta un caso de estudio que aborda el intercambio de información entre el planificador y el modelo del producto.

Como se aprecia en la Figura 2, cuando el cliente realiza un pedido, se crea una instancia del producto en la base datos con la información del cliente y del pedido. Asimismo, se asocia la información de diseño incluyendo la receta general necesaria para la fabricación del producto.

Una vez confirmado el pedido, el planificador debe optimizar el uso de los recursos de producción para conseguir reducir tiempos y costes, para lo cual necesita información sobre el proceso y las operaciones de fabricación del producto. Utilizando la información de la *Receta General* y la disponibilidad de los recursos de producción, el planificador crea la *Receta Maestra*, añadiendo los equipos y recursos que se van a utilizar para realizar las operaciones de fabricación. La información del producto se actualiza incluyendo esta nueva receta.

La interacción o intercambio de información entre el producto y el planificador se realiza utilizando la *API REST del producto*. Para ello, se realizan las siguientes transacciones:

1. El planificador solicita al producto la *Receta General*: petición GET donde se pasa como parámetro el recurso solicitado:

GET /product/read/*productID*?resource=*GeneralRecipe*

El producto devuelve la *Receta General* en formato XML dentro del cuerpo del mensaje de respuesta.

2. El planificador genera o selecciona la *Receta Maestra* y se actualiza la información del producto.

Si la receta no está en la base de datos, utiliza una petición PUT pasando como parámetros el recurso y el documento asociado en el cuerpo del mensaje. La petición devuelve la URI del nuevo recurso creado.

PUT /product/read/*productID*?resource=*GeneralRecipe*
Body= MasterRecipe.xml

En caso de que el recurso ya exista, se actualizará el producto utilizando la misma petición. En este caso solamente se envía la URI asociada con el recurso en el cuerpo del mensaje.

PUT /product/read/*productID*?resource=*GeneralRecipe*
Body= URI MasterRecipe.xml

La figura 5 muestra las transacciones HTTP entre el planificador y el producto.

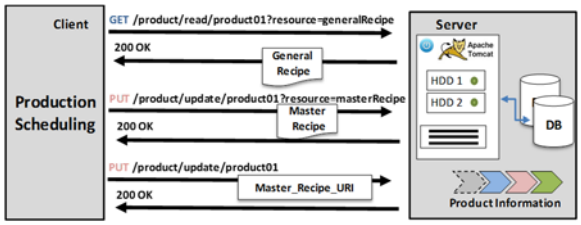

Figura 5: Transacciones HTTP entre el planificador y el producto

6 CONCLUSIONES

En este trabajo se presenta la digitalización de la primera fase del ciclo de vida del producto correspondiente al diseño y fabricación en el entorno de la empresa dentro del enfoque de Industry 4.0.

El prototipo empleado en el caso de estudio ha permitido demostrar la viabilidad de colaboración entre distintas disciplinas, el desarrollo distribuido globalmente, y la integración de proveedores y clientes.

El trabajo futuro contempla la ampliación del modelo del producto y del prototipo para abarcar la fase de operación/servicio/soporte del producto en el entorno del cliente, así como la fase final de la vida útil del producto.

Agradecimientos

Este trabajo está financiado por MCIU/AEI/FEDER, UE (proyecto DPI2015-68602-R).

English summary

END-TO-END INTEGRATION THROUGH THE PRODUCT 4.0 MODEL

Abstract

Digitization is a clear need in the industrial environment for all companies seeking to move towards Industry 4.0. This paper presents a proposal for the digitization of the first phase of the product lifecycle corresponding to the design and manufacture in the company environment, which allows integration with suppliers and customers. This approach adopts the standards proposed in the RAMI 4.0 model and the I4.0 component model. The technological solution used in the prototype uses web services with API REST and distributed XML databases. This prototype has been used in a case study to show the interoperability of applications throughout the product lifecycle. Specifically, it shows the exchange of information between the scheduler and the product model.

Keywords: End-to-End integration, Product lifecycle, Web Services, Industry 4.0.

Referencias

[1] Armengaud, E., et al., (2017) "Industry 4.0 as Digitalization over the Entire Product Lifecycle: Opportunities in the Automotive Domain", Systems, Software and Services Process Improvement. Ostrava, Czech Republic, Springer, pp. 334-351.

[2] European Commission, (2019) "Digital Economy and Society Index Report 2019 - Integration of Digital Technology".

[3] eXistdb, (2018) "eXist-db - The Open Source Native XML Database", http://exist-db.org/exist/apps/homepage/index.html (accessed 6 20, 2019).

[4] Fielding, R.T., (2000) "Architectural styles and the design of network-based software architectures", Doctoral Dissertation, University of California, Irvine.

[5] Halili, F., Ramadani, E., (2018) "Web Services: A Comparison of Soap and Rest Services", Modern Applied Science, pp. 175-183.

[6] Herrera-Morales, et al., (2009) "La Interoperabilidad de Aplicaciones y los Servicios Web", Tópicos Selectos de Tecnologías de Información con Aplicaciones Prácticas, Universidad de Colima, pp. 85-87.

[7] Infosys, (2018) "Best practices for building Restfull Web services", https://www.infosys.com/digital/insights/Documents/restful-web-services.pdf.

[8] Iuga, A., Popa, V., Popa, L., (2017) "Industrial Product Life Cycle Stages and Lifecycle Eco-design", 5th International Conference on Advanced Manufacturing Engineering and Technologies, Newtech, Springer, 365-374.

[9] Kumari, V., (2015) "Web Services Protocol: SOAP vs REST", International Journal of Advanced Research in Computer Engineering & Technology, pp. 2467-2469.

[10] Leach, P.J., Berners-Lee, T., Mogul, J.C., Masinter, L., Fielding, R.T., Gettys, J., (1999) "Hypertext Transfer Protocol - HTTP/1.1", https://tools.ietf.org/html/rfc2616.

[11] Lifecyclemodeling.org, (2017) "Lifecycle Modeling Language", http://www.lifecyclemodeling.org/ (accessed 6 20, 2019).

[12] Marcon, P., et al., (2019) "New Approaches to Implementing the SmartJacket into Industry 4.0", Sensors 19, no. 7, pp. 1592-1612.

[13] Muehlena, M., Nickersona, J.V., Swensonb, K.D., (2005) "Developing web services choreography standards - the case of REST vs. SOAP", Decision Support Systems, pp. 9-29
.

[14] Serrano, N., Hernantes, J., Gallardo, G, (2014) "Service-Oriented Architecture and Legacy Systems", IEEE Software, pp. 15-19.

[15] Stark, R., et al., (2014) "Intelligent Information Technologies to Enable Next Generation PLM", 11th IFIP International Conference on Product Lifecycle Management (PLM), Yokohama (Japan), Springer, pp. 485-495.

[16] Tarkowska, A., et al., (2018) "Eleven quick tips to build a usable REST API for life sciences", PLOS, Computational Biology, pp. 1-8.

[17] Tihomirovs, J., Grabis, J., (2016) "Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics", Information Technology and Management Science, pp. 92-97.

[18] Valero-Gómez, A., et al., (2013) "Arquitectura de integración basada en Servicios Web para sistemas heterogéneos y distribuidos: aplicación a robots móviles interactivos", Revista Iberoamericana de Automática e Informática industrial, pp. 85–95.

[19] W3C, (2007) "SOAP Version 1.2 Part 0: Primer, https://www.w3.org/TR/2007/REC-soap12-part0-20070427/#L1244.

[20] Wagh, K.S., Thool, R., (2012) "A Comparative study of SOAP vs REST web services provisioning techniques for mobile host", Journal of Information Engineering and Applications 2, no. 5, pp. 12-16.

[21] Wellsandt, et al., (2016) "A survey of product lifecycle models: towards complex products and service offers", International Journal of Product Lifecycle Management 9, no. 4, pp. 353-390.

[22] Wellsandt, S., et al., (2019) "Life Cycle Management for Product-Service Systems", edited by Cattaneo, L., Terzi, S., Models, Methods and Tools for Product Service Design, Springer, pp. 29-43.

[23] Xin, Y., Ojanen, V., (2017) "The Impact of Digitalization on Product Lifecycle Management: How to Deal with it?", IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 1098-1102.