Enfoque para Generar Aplicaciones Orientadas a Servicios para IoT mediante el Desarrollo Dirigido por Modelos

Claudia M. Sosa-Reyna[1], Edgar Tello-Leal[2], David Lara-Alabazares[1]

[1] Departamento de posgrado en Eléctrica y Electrónica, UAM Reynosa-Rodhe, Universidad Autónoma de Tamaulipas, Reynosa, Tamaulipas, México
[2] Facultad de Ingeniería y Ciencias, Universidad Autónoma de Tamaulipas, Victoria, Tamaulipas, México
clauqueen1@gmail.com, etello@uat.edu.mx, dlara@docentes.uat.edu.mx

Resumen El Internet de las Cosas (*Internet of Things*, IoT) es una parte integral de la Internet del futuro. IoT se puede entender como una infraestructura de red global dinámica con capacidad de auto configuración, basada en protocolos de comunicación estándar e interoperables, donde "cosas" físicas y virtuales tienen identidad, atributos físicos y personalidades virtuales. El principal problema en las soluciones tecnológicas para IoT se presenta en un nivel de aplicación, es decir, un conjunto de componentes de software que se requiere desarrollar para gestionar las cosas, objetos o dispositivos a interconectar. Estos requerimientos propician la utilización de los principios del Desarrollo Dirigido por Modelos (*Model-Driven Development*, MDD) para la construcción de aplicaciones de software, permitiendo generar modelos en diferentes niveles de abstracción, con la posibilidad de generar artefactos de implementación de software (código) en diferentes plataformas. En este artículo, se propone una metodología basada en MDD con diferentes niveles de abstracción, punto de vista, y granularidad, con el objetivo de guiar el desarrollo de aplicaciones de software para IoT. La metodología es soportada mediante métodos de transformación de modelos posibilitando la generación del código de las aplicaciones de software para IoT. Adicionalmente, se presenta una arquitectura orientada a servicios para el despliegue de aplicaciones de software, conformada de cuatro capas que permiten identificar los componentes requeridos para la implementación de sistemas de IoT.

Keywords: Internet de las Cosas, MDD, SOA, Aplicaciones de software, Métodos de transformación

1. Introducción

Recientemente, varios estudios de la academia, industria, y gobierno, han tratado de conectar todas las cosas u objetos en el mundo de la Internet, con el fin de proporcionar un sistema integrado para mejorar su rendimiento en la transmisión de información, a lo que se le ha llamado Internet de las Cosas

(del inglés *Internet of Things*, IoT) [13,5]. Cuando se hace referencia a IoT, normalmente se relaciona con una cosa u objeto físico, pero un componente fundamental en IoT es el software que se ejecuta en las *cosas*. Este software puede estar presente en varias formas: embebido, *middleware*, aplicaciones, lógica en la composición de servicios, y herramientas de gestión [1,11].

Los avances alcanzados en materia de automatización y ambientes inteligentes han llegado a niveles importantes, cada vez son más las aportaciones en este sentido a través de los conceptos de IoT: ciudades inteligentes, automatización de casas, *e-health*, manejo inteligente de uso de agua, por mencionar algunos. Con los beneficios que han generado estos nuevos paradigmas, se ha presentado también la complejidad en su desarrollo; es por ello que se han buscado alternativas que faciliten los procesos de generación e implementación de aplicaciones innovadoras. Entonces, el contar con modelos que conceptualicen el dominio de un problema específico, y que permitan identificar, clasificar, y abstraer los elementos que lo conforman, representa la posibilidad de alcanzar una implementación automática eficiente.

En este sentido, el Desarrollo Dirigido por Modelos (del inglés *Model Driven Development*, MDD) está basado en modelos que, en un principio, minimizan los aspectos tecnológicos de manera que sea más eficiente la comunicación entre los usuarios, los analistas, y los desarrolladores de un sistema, permitiendo seleccionar la plataforma tecnológica hasta el final del proceso. La utilización de MDD representa una mayor productividad y eficiencia, en cuanto a que se genera automáticamente el código como producto de una serie de transformaciones de modelos, favoreciendo a la consistencia como producto de la automatización. Los modelos de alto nivel de abstracción se transforman en modelos de nivel más bajo, en donde la relación entre ambos da como resultado una dependencia que guarda el proceso que se ha seguido hasta llegar a una solución tecnológica, contribuyendo a comprender las consecuencias de los cambios en cualquier momento del proceso de desarrollo [12]. Al trabajar en base a modelos, es posible lograr una fácil adecuación a los cambios, tanto tecnológicos como de los requerimientos de los usuarios de negocio que puedan surgir en el proceso de desarrollo, convirtiendo a los modelos en unidades re-utilizables y perdurables.

Como parte del proceso de modelado, en donde los modelos son unidades productivas de las cuales emanan las implementaciones automáticas, encontramos como puntos medulares a la abstracción, representada por lenguajes de modelado de alto nivel; la automatización, que permite transformar los modelos en programas computacionales; y los estándares, o herramientas complementarias de desarrollo; con la finalidad de obtener los artefactos o modelos formales que puedan ser comprendidos por una computadora [7]. Tomando en cuenta la ubicuidad y particularidad presente en los ambientes inteligentes, MDD permite el manejo de tecnologías heterogéneas mediante modelos automáticos de transformación y generación de código para plataformas específicas. En MDD la transformación de modelos puede ser vertical, en donde afina modelos abstractos en modelos más específicos. También, puede ser en forma horizontal, elaborando mapeos entre modelos del mismo nivel de abstracción y de esta manera identificar la

mejor solución [12]. En este sentido, en el enfoque de la Arquitectura Orientada a Servicios (del inglés, *Service-Oriented Architecture*, SOA), un sistema complejo es tratado como un conjunto de objetos bien definidos o subsistemas [10]. Estos objetos o subsistemas pueden ser reutilizados, manteniendo su forma individual. Entonces, los componentes de software y hardware en una arquitectura IoT, implementados con SOA, pueden ser reutilizados y actualizados de manera eficiente. Por lo tanto, cuando SOA es aplicado en IoT, el diseño generado puede proporcionar extensibilidad, escalabilidad, modularidad, e interoperabilidad entre cosas heterogéneas, así como las funcionalidades y capacidades que se encapsularon en un conjunto de servicios.

En este artículo se propone un metodología basada en el MDD que permita guiar el proceso de desarrollo de aplicaciones de software orientadas a servicios, que posibiliten satisfacer los requerimientos de negocio del dominio de IoT. La metodología se conforma de un conjunto de métodos basados en MDD con diferentes niveles de abstracción, punto de vista y granularidad. Además, se presenta una arquitectura para sistemas de IoT compuesta de cuatro capas, y que se basa en el enfoque de SOA. Lo anterior, permite garantizar la interoperabilidad entre dispositivos heterogéneos en múltiples vías, estableciendo un puente entre el mundo físico y el mundo virtual de IoT.

2. Arquitectura basada en SOA para sistemas de IoT

El principal requerimiento de IoT es que las cosas u objetos en la red deben estar interconectadas. La arquitectura de un sistema de IoT debe garantizar las operaciones de las cosas, permitiendo establecer un puente entre las cosas (parte física) y el mundo virtual de IoT. La arquitectura basada en SOA para el desarrollo de sistemas de IoT propuesta se conforma de 4 capas, tal como se muestra en la Figura 1 y se describen a continuación.

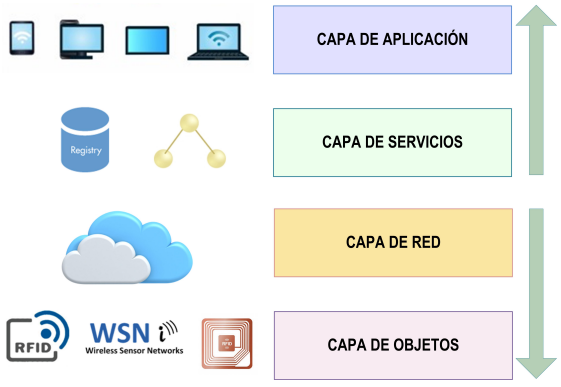

Figura 1. Arquitectura basada en SOA para IoT

Capa de Objetos. Esta integrada con los objetos de hardware disponibles en la red que detectan el estado de las cosas. En la capa de objetos, los sistemas inteligentes mediante etiquetas (*tags*) o sensores, son capaces de detectar automáticamente el medio ambiente y el intercambio de datos entre los dispositivos. Los objetos de esta capa deben tener una identidad digital (identificador único universal, UUID), lo que permite raestrear al objeto en el dominio digital, posibilitando cumplir con la expectativa de IoT de ser una red física inter-conectada en todo el mundo, en el que las cosas están conectadas a la perfección y se pueden controlar de forma remota [10].

Capa de Red. Consiste de la infraestructura que soporta las conexiones por cable, inalámbricas o móviles entre las cosas, permitiendo detectar su entorno, lo que habilita compartir datos entre las cosas conectadas, posibilitando la gestión de eventos y el procesamiento inteligente de IoT. En el enfoque SOA, los servicios serán consumidos por las cosas que han sido habilitadas en la capa de red. La capa de red es crucial en cualquier enfoque de IoT, considerando funcionalidades de QoS, gestión eficiente de energía en la red y en las cosas, procesamiento de señales y datos, seguridad y privacidad, entre otras.

Capa de Servicio. En esta capa se crean y gestionan los servicios requeridos por los usuarios o aplicaciones de software. La capa de servicio se basa en la tecnología de *middleware*, la cual es fundamental para consumir servicios y la ejecución de aplicaciones de IoT, donde las plataformas de hardware y software pueden ser reutilizables. Es una de las capas de operación crítica de la arquitectura, que funciona en modo bidireccional. Esta capa opera como interfaz entre la capa de objetos (en la parte inferior de la arquitectura), y la capa de aplicación (en la parte superior de la arquitectura). Es responsable de funciones como la gestión de dispositivos, gestión de información, filtrado de datos, agregación de datos, análisis semántico, y descubrimiento de información [2]. La *capa de servicios* se conforma de: descubrimiento de servicios, composición de servicios, APIs, y gestión de confiabilidad (ver Figura 2), entre otros. El *descubrimiento de servicios* permite encontrar los objetos que pueden proporcionar el servicio requerido y la información necesaria de forma eficaz, mediante el UUID en el registro de servicios o repositorio de servicios. La *composición de servicios* permite la interacción entre las cosas conectadas mediante la combinación de los servicios disponibles para realizar una tarea específica, es decir, cuando los servicios están creados y almacenados en el repositorio de servicios, se pueden combinar en servicios de mayor nivel de complejidad a partir de la lógica de negocio.

Capa de Aplicación. Es la capa responsable de la entrega de las aplicaciones a los diferentes usuarios de IoT. La intención de la arquitectura es soportar aplicaciones verticales. El desarrollo de aplicaciones en IoT se ha centrado en las áreas de salud, agricultura, transporte, ciudades inteligentes, automatización de casas, sistemas complejos para la toma de decisiones, gestión de uso de agua, etc [4].

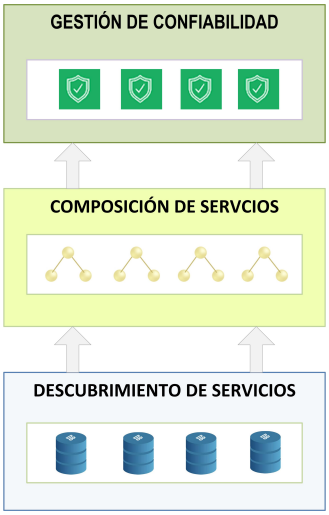

Figura 2. Esquema de la capa de servicios de SOA

3. Metodología MDD para Aplicaciones de IoT

La metodología propuesta apunta al uso de modelos conceptuales en diferentes puntos de vista, niveles de abstracción, y de granularidad. Los artefactos de salida de las fases de esta metodología se representan por medio de modelos (de procesos y/o sistemas), generados mediante la aplicación de los principios del MDD. El resultado final son los artefactos de implementación de software, es decir, el código de las aplicaciones o sistemas de software para IoT. En la Figura 3 se presentan las fases que componen esta metodología: 1) análisis de requerimientos de negocio, 2) definición de la lógica de negocio, 3) diseño de la solución integrada de servicios, y 4) generación de la solución tecnológica.

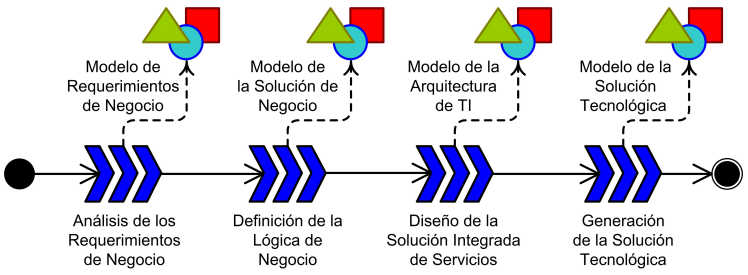

Figura 3. Metodología basada en MDD para desplegar aplicaciones de IoT

Fase 1. Análisis de los requerimientos de negocio. Esta fase consiste en analizar el dominio del problema e identificar los requerimientos de negocio. Esto se realiza considerando los requerimientos funcionales y no funcionales del sistema. Para definir el *modelo de requerimientos de negocio* se utiliza el lenguaje UML, capturando el flujo del proceso de software mediante diagramas de casos de uso y diagramas de actividades, generando un modelo definido en un nivel PIM de MDD.

Fase 2. Definición de la lógica de negocio. Esta fase se enfoca en el diseño de los procesos de negocio requeridos para soportar la lógica de negocio y los requerimientos de negocio. Entonces, se utiliza como entrada a la fase, el modelo de requerimientos de negocio generado previamente, y se complementa mediante la definición de la lógica del proceso de negocio, utilizando el lenguaje de modelado y notación de procesos de negocio (*Business Process Model and Notation*, BPMN), lo cual permite generar un modelo de la solución de negocio definido en un nivel PIM del MDD, describiendo el comportamiento e interacciones del proceso de negocio desde un punto de vista global.

Fase 3. Diseño de la solución integrada de servicios. Esta fase tiene por objetivo la definición de un modelo de la arquitectura de TI, en un nivel independiente de la plataforma para separar la solución de la lógica de negocio de los aspectos técnicos de implementación (TI), lo que permite que este tipo de implementación pueda generarse en diferentes plataformas destino. El modelo de arquitectura de TI se deriva del modelo de la solución de negocio generado en la fase anterior, el modelo generado se mantiene sin cambios en cualquier plataforma. En este caso, el modelo de la arquitectura de TI se genera siguiendo el enfoque orientado a servicios SOA.

Fase 4. Generación de la solución tecnológica. En esta fase se utilizan conceptos específicos de la plataforma de implementación con el fin de convertir esta solución en código ejecutable de una aplicación de software en particular. Esta fase se realiza mediante la ejecución de dos etapas: 1) diseño de la solución de la plataforma específica de TI, y 2) generación de las especificaciones o código del sistema de software. La primera etapa consiste en la definición de un modelo de especificaciones basado en un estándar o tecnología específica (por ejemplo, TinyOS 2.0 o WSN Operating Systems), utilizando como entrada a la fase, el modelo de la arquitectura de TI generado previamente. El modelo de la solución tecnológica contiene la información requerida para la plataforma específica (mensajes concretos en el formato de enviar o recibir para las cosas u objetos, protocolos de transporte usados, UUID del sensor, emisor o receptor, etc.). La segunda etapa consiste de una transformación del modelo PSM de MDD a texto, que representa el esqueleto de código o el código ejecutable de una aplicación, normalmente en especificaciones basadas en XML.

3.1. Métodos de Transformación de Modelos

Para reducir los costos y el tiempo de desarrollo, la metodología es soportada por métodos basados en el enfoque MDD, permitiendo transformaciones automáticas y semi-automáticas de modelos para generar los modelos de salida de cada fase. Una transformación de un modelo consiste de un conjunto de reglas de transformación, que definen cómo un modelo de entrada es mapeado a uno o más modelos o código ejecutable de salida. Para soportar las transformaciones de modelos necesarias para la metodología, se propone la aplicación de diferentes métodos de transformación de modelos, tal como se muestra en la Figura 4. Por un lado, un *modelo independiente de la plataforma* (del inglés *Platform Independent Model*, PIM) es una vista del sistema independiente de la plataforma, es decir, un modelo con alto nivel de abstracción independiente de cualquier tecnología o lenguaje de implementación que exhibe un grado suficiente de independencia de la plataforma a fin de permitir su mapeo a una o más plataformas. Por otro lado, un *modelo específico de la plataforma* (del inglés *Platform Specific Model*, PSM) presenta una vista del sistema desde la perspectiva de una plataforma tecnológica específica, es decir, un modelo de solución asociado a una plataforma que incluye los detalles del PIM y que describe cómo realizar la implementación en dicha plataforma.

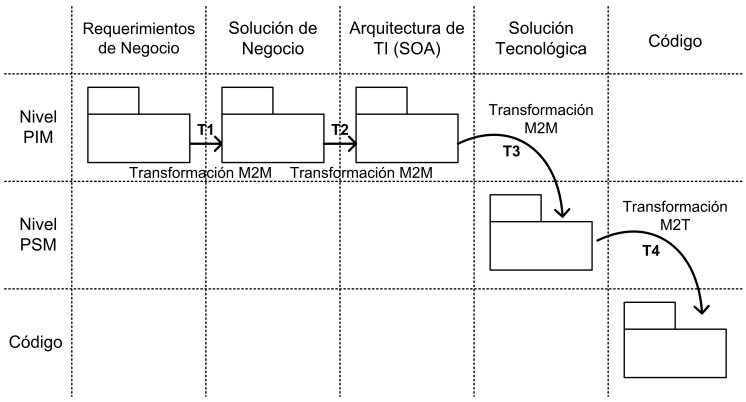

Figura 4. Transformaciones de modelos en diferentes niveles de abstración

La Figura 4 muestra los métodos MDD que se proponen para generar las soluciones tecnológicas en ambientes de IoT, mediante aplicaciones de software orientadas a servicios. Se detallan los niveles de abstracción de modelos de acuerdo a MDD, que van desde un modelo en nivel PIM hasta el código. También, se detallan los artefactos (de modelos y de código) que se generan en cada fase. Para cada uno de ellos se indica la transformación que se realiza, el modelo y el nivel de origen (de entrada), y el modelo o código que resulta (de salida o destino), así como el nivel al que corresponde el modelo generado. Mediante la

transformación **T1** se genera un modelo conceptual de la solución de negocio a partir de un modelo de requerimientos de negocio, complementado con la lógica de negocio y con el proceso de negocio diseñado, a través de una transformación horizontal PIM-a-PIM. La transformación modelo-a-modelo **T2** apunta a generar un modelo de la arquitectura de TI, definido en un nivel PIM mediante una transformación horizontal. Esta transformación se deriva del modelo de la solución de negocio, utilizando los conceptos de la arquitectura orientada a servicios, manteniendo una independencia de la plataforma de implementación. En la Figura 5 se muestra un ejemplo de una regla de transformación (**T2**), utilizando como entrada el metamodelo del lenguaje BPMN y como destino el metamodelo de la arquitectura SOA (Regla 1). Los métodos de transformación y las reglas propuestas se definieron utilizando el lenguaje de transformación de modelos *Eclipse Atlas Transformation Language* (ATL)[3].

La solución tecnológica (ver Figura 4) es generada mediante dos métodos dirigidos por modelos. El primer método aplica una transformación modelo-a-modelo **T3**, generando un modelo de salida basado en la plataforma de implementación específica que se seleccione, utilizando un modelo de la arquitectura de TI como entrada. El modelo generado es definido en un nivel específico de la plataforma PSM, utilizando conceptos de la plataforma de implementación de IoT. En la Figura 5 se muestra un ejemplo de una regla de transformación (Regla 2) del metamodelo de SOA (nivel PIM) a un metamodelo en un nivel PSM. En el ejemplo se generan los datos que se enviarán a un sensor de una aplicación de software de un simulador de vehículos inteligentes. El segundo método se lleva a cabo mediante la transformación directa modelo-a-texto **T4**, que consiste en la generación de un documento con el código fuente que representa la estructura y comportamiento del objeto ante un evento.

```
rule participant2scope {
      from
            participant: MMbpmn2!Participant (
                  not
participant.processRef.oclIsUndefined()
            )
      to
            scope: MMUML4SOA!Receive (
                  lnk <- participant.name,
                  rvc <- participant.document
            )
}
```

Regla 1

```
rule scope2vehicle {
      from
            participant: MMUML4SOA!Recieve (
                  not
recieve.processRef.oclIsUndefined()
            )
      to
            vehicle: MMSmartVehicle!sensor (
                  sensorData <- receive.lnk,
                  command <- receive.rvc
            )
}
```

Regla 2

Figura 5. Ejemplo de reglas de transformación del método **T2** (Regla 1) y **T3** (Regla 2)

[3] www.eclipse.org/atl/

4. Trabajos Relacionados

En [9] se propone el uso del enfoque de MDD para crear métodos para la generación de código flexible. Se muestra cómo, usando una transformación modelo a modelo, se deriva un modelo abstracto. Se presentan las bases para la generación de código para construir codificadores y decodificadores eficientes, con lo que se pueden llevar a cabo traducciones automáticas entre diferentes esquemas de codificación que pueden ser elegidos libremente en base a los requerimientos específicos de dominio. Mediante esquemas de MDD se genera un modelo común de representación de formatos de datos derivado de diferentes fuentes. En base a este esquema de generación de código, se implementa un sistema de enlace, el cual proporciona una interfaz de comunicación con servicios de redes de sensores inalámbricos. La propuesta presentada en [6] es un esquema de desarrollo de aplicaciones para sensores FRASAD, basada en MDD. En la arquitectura propuesta se utiliza un modelo basado en reglas y DLS para describir las aplicaciones. Se utilizan una interfase gráfica para el usuario, componentes de generación de código, y herramientas de soporte para auxiliar a los desarrolladores en el diseño, implementación, y prueba de las aplicaciones de IoT.

Por otro lado, en [3] se describe una herramienta basada en MDD para IoT, la cual se fundamenta en el servicio de descubrimiento semántico, permitiendo una selección y localización dinámica de recursos o dispositivos mediante una interfaz gráfica. Se enfoca en las características de la plataforma para el manejo de IoT, describiendo cómo el enfoque de *middleware* puede simplificar el uso de un manejador de IoT en un ambiente real de manufactura. Una herramienta de desarrollo llamada IoTLink basada en el enfoque de MDD se presenta en [8]. IoTLink permite a los desarrolladores inexpertos elaborar aplicaciones combinadas mediante el lenguaje específico de dominio (DSL), el cual puede ser configurado para crear aplicaciones de IoT. A través de componentes visuales, IoTLink encapsula la complejidad de la comunicación con los dispositivos y servicios de internet, y los abstrae como objetos virtuales que son accesibles por medio de diferentes tecnologías de comunicación, resolviendo la interoperabilidad entre componentes IoT heterogéneos.

5. Conclusiones

En este trabajo de investigación se presentó una metodología para el desarrollo de aplicaciones de software para IoT. La metodología se basa en los principios del desarrollo dirigido por modelos (MDD), por lo cual se definieron un conjunto de métodos de transformación de modelos, los cuales se especificaron con diferentes puntos de vista, niveles de abstracción y granularidad. La metodología propuesta permite guiar el proceso de desarrollo de aplicaciones de software orientadas a servicios, a partir de modelos conceptuales hasta llegar al código de una aplicación específica y en una plataforma tecnológica seleccionada. El enfoque propuesto tiene como objetivo disminuir los tiempos y costos del desarrollo de software mediante la implementación de transformaciones de modelos automáticas y semi-automáticas.

Además, se propuso una arquitectura para soportar las aplicaciones y/o sistemas de software para IoT. La arquitectura describe de forma genérica las diferentes capas requeridas para el despliegue de aplicaciones de software en IoT, utilizando los conceptos de la arquitectura orientada a servicios (SOA).

Referencias

1. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Comput. Netw. 54(15), 2787–2805 (2010)
2. Bandyopadhyay, D., Sen, J.: Internet of things: Applications and challenges in technology and standardization. Wireless Personal Communications 58(1), 49–69 (2011)
3. Conzon, D., Brizzi, P., Kasinathan, P., Pastrone, C., Pramudianto, F., Cultrona, P.: Industrial application development exploiting iot vision and model driven programming. In: Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on. pp. 168–175 (2015)
4. Fang, S., Xu, L.D., Zhu, Y., Ahati, J., Pei, H., Yan, J., Liu, Z.: An integrated system for regional environmental monitoring and management based on internet of things. IEEE Transactions on Industrial Informatics 10(2), 1596–1605 (2014)
5. Miorandi, D., Sicari, S., Pellegrini, F.D., Chlamtac, I.: Internet of things: Vision, applications and research challenges. Ad Hoc Networks 10(7), 1497 – 1516 (2012)
6. Nguyen, X.T., Tran, H.T., Baraki, H., Geihs, K.: Frasad: A framework for model-driven iot application development. In: Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on. pp. 387–392 (2015)
7. Pons, C., Giandini, R., Perez, G.: Desarrollo de software dirigido por modelos. Editorial Universidad Nacional La Plata - McGraw Hill, Argentina, 1a. edn. (2010)
8. Pramudianto, F., Kamienski, C.A., Souto, E., Borelli, F., Gomes, L.L., Sadok, D., Jarke, M.: Iot link: An internet of things prototyping toolkit. In: Ubiquitous Intelligence and Computing, 2014 IEEE 11th Intl Conf on and IEEE 11th Intl Conf on and Autonomic and Trusted Computing, and IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UTC-ATC-ScalCom). pp. 1–9 (2014)
9. Riedel, T., Yordanov, D., Fantana, N., Scholz, M., Decker, C.: A model driven internet of things. In: 2010 Seventh International Conference on Networked Sensing Systems (INSS). pp. 265–268 (2010)
10. Shancang, L., Da, X.L., Shanshan, Z.: The internet of things: a survey. Information Systems Frontiers 17(2), 243–259 (2015), http://dx.doi.org/10.1007/s10796-014-9492-7
11. Simon, S., A., F.B., Thomas, V.: Designing an application store for the internet of things: Requirements and challenges. In: De Ruyter, B., Kameas, A., Chatzimisios, P., Mavrommati, I. (eds.) Ambient Intelligence: 12th European Conference, AmI 2015, Athens, Greece, November 11-13, 2015, Proceedings, pp. 313–327. Springer International Publishing (2015)
12. Tello-Leal, E., Ríos-Alvarado, A.B., López-Arévalo, I., Chiotti, O., Villarreal, P.D.: Metodología basada en el desarrollo dirigido por modelos para la ejecución de procesos colaborativos mediante agentes de software. Ediciones UAT - Plaza y Valdes, México, 1a. edn. (2016)
13. Tsai, C.W., Lai, C.F., Vasilakos, A.V.: Future internet of things: open issues and challenges. Wireless Networks 20(8), 2201–2217 (2014)