UNIVERSIDAD DE CUENCA

Facultad de Ingeniería

Carrera de Ingeniería de Sistemas

Creación de una infraestructura auto adaptativa de monitoreo mediante un middleware de configuración de componentes sensibles al contexto en ambientes de vida asistida

Trabajo de titulación previo a la obtención del título de Ingeniero en Sistemas

Autores:

José Alfredo Moyano Dután
CI: 0105751473
Correo electrónico: moyano.jose.a@gmail.com

Ariana Isabel Roman Alvarado
CI: 1105375222
Correo electrónico: ariana.roman0903@gmail.com

Directora:

Ing. Irene Priscila Cedillo Orellana, PhD
CI: 0102815842

Cuenca, Ecuador

01-abril-2021

Resumen:

Hoy en día la tecnología forma parte de nuestra vida cotidiana, ya que ayuda a solventar diferentes necesidades y brinda soluciones que facilitan el desarrollo de actividades, ya sean empresariales, educativas, de salud, ambientales, etc. Además de brindar autosuficiencia a las personas frente a las tareas cotidianas, no sólo a usuarios comunes, sino también, a personas con habilidades y destrezas especiales, como es el caso de adultos mayores o personas con discapacidad; esto, con la finalidad de mejorar su calidad de vida y otorgarles mayor independencia, ya sea en centros de cuidado o entornos residenciales familiares.

Varias investigaciones acerca de tecnologías de asistencia se han llevado a cabo, basadas en un nuevo paradigma de la tecnología de la información denominado "inteligencia ambiental"; este paradigma ayuda a potenciar las capacidades de las personas de avanzada edad, personas con diversidad funcional o personas con enfermedades crónicas, a través de entornos digitales que son sensibles, adaptables y responden a las necesidades humanas.

Del sentido, del apoyo y soporte a los adultos mayores, nace el concepto de Ambientes de Vida Asistida (*AAL - Ambient Assisted Living*), que se pueden describir como sistemas inteligentes de asistencia para una vida mejor y más segura. Por tanto, un sistema inteligente debe estar compuesto de diferentes “objetos” (electrodomésticos, sensores, actuadores, etc.) de la vida cotidiana interconectados entre sí para crear servicios a partir de la detección de algún evento físico sin la necesidad de la intervención humana directa y conseguir facilitar tareas en el hogar.

Hoy en día, a la prestación de servicios avanzados mediante la interconexión de objetos es lo que se denomina el internet de las cosas (*IoT – Internet of things*). Unir los términos de AAL e IoT da como resultado entornos modernos de ejecución, los cuales se han vuelto cada vez más descentralizados, heterogéneos, inciertos y cambiantes. Lo que ha llevado a la comunidad de ingeniería de software a proponer formas innovadoras para construir, ejecutar y administrar sistemas y servicios que permiten que el software y sus modelos se puedan reconfigurar a sí mismo sin la necesidad de ser reconstruidos.

Además, se necesita abordar nuevas teorías, mecanismos y métodos de modelado de software adecuados para construir entornos auto-adaptativos dotados del IoT, es por ello que los modelos en tiempo de ejecución son una herramienta que permite abordar la complejidad de las adaptaciones dinámicas, a fin de mantener un modelo abstracto de un sistema en tiempo de ejecución. Por tanto, es necesario conceptualizar a los modelos en tiempo de ejecución como representaciones simples que construyen a propósitos específicos, de tal manera que, por medio de una relación causal, si un sistema cambia su modelo asociado también cambia o viceversa.

En consecuencia, el desafío en nuestro trabajo de titulación es conseguir adaptar y auto configurar entidades digitales (plataformas cloud, gateway, dispositivos IoT) y entidades físicas (usuarios del sistema, entornos), de forma automática, en una arquitectura para IoT desplegada sin necesidad de detener el sistema en ejecución. En este aspecto, para conseguir vencer el desafío, nuestra visión será enfocada mediante los modelos en tiempo de ejecución (*M@rt - Models at runtime*) con el objetivo de monitorear y recopilar datos, así como también para reducir el tiempo que tomaría reiniciar el sistema y perder información sensible del contexto.

En el presente trabajo de titulación, se plantea el diseño de una infraestructura enfocada al monitoreo, que permita generar datos para el soporte posterior a la auto adaptación de los diferentes componentes y/o dispositivos de IoT mediante un middleware de monitoreo en tiempo de ejecución.

Además, a manera de implementación de la solución propuesta y como un entregable adicional, se ha desarrollado un lenguaje de dominio especifico (*DSL - Domain Specific Language*) denominado Monitor-IoT, el mismo que, brinda al diseñador y desarrollador de soluciones aplicado a un estudio de caso, una herramienta para el desarrollo de arquitecturas para internet de las cosas y permite analizarlas desde el punto de vista de las percepciones del usuario.

Palabras Clave: Middleware. Internet de las Cosas. Modelos en Tiempo de Ejecución. Computación en la Nube.

Abstract:

Today technology is part of our daily life, since it helps to solve different needs and provides solutions that facilitate the development of activities, be they business, educational, health, environmental, etc. In addition to providing self-sufficiency to people facing daily tasks, not only common users, but also people with special abilities and skills, such as the elderly or people with disabilities; this to improve their quality of life and grant them greater independence, whether in care centers or family residential settings.

Several investigations about assistive technologies have been carried out, based on a new paradigm of information technology called "ambient intelligence"; This paradigm helps to enhance the capacities of the elderly, people with functional diversity or people with chronic diseases, through digital environments that are sensitive, adaptable and respond to human needs.

From the sense, support and support to the elderly, the concept of Assisted Living Environments (*AAL - Ambient Assisted Living*) was born, which can be described as intelligent assistance systems for a better and safer life. Therefore, an intelligent system must be composed of different “objects” (household appliances, sensors, actuators, etc.) from everyday life interconnected to each other to create services based on the detection of a physical event without the need for human intervention. direct and to facilitate tasks at home.

Nowadays, the provision of advanced services through the interconnection of objects is what is called the internet of things (*IoT - Internet of things*). Bringing together the terms of AAL and IoT results in modern execution environments, which have become increasingly decentralized, heterogeneous, uncertain, and changeable. This has led the software engineering community to propose innovative ways to build, run and manage systems and services that allow the software and its models to reconfigure itself without the need to be rebuilt.

In addition, it is necessary to address new theories, mechanisms, and appropriate software modeling methods to build self-adaptive environments endowed with the IoT, which is why runtime models are a tool that allows addressing the complexity of dynamic adaptations, to maintain an abstract model of a system at run time. Therefore, it is necessary to conceptualize the run-time models as simple representations that are built for specific purposes, in such a way that through a causal relationship, if a system changes its associated model also changes or vice versa.

Consequently, the challenge in our degree work is to adapt and self-configure digital entities (cloud platforms, gateway, IoT devices) and physical entities (system users, environments), automatically, in an architecture for IoT deployed without need to stop the running system. In this aspect, to overcome the challenge, our vision will be focused through the models at runtime (*M@rt - Models at runtime*) to monitor and

collect data, as well as to reduce the time it would take to restart the system and lose context sensitive information.

In the present degree work, the design of an infrastructure focused on monitoring is proposed, which allows generating data for the subsequent support of the self-adaptation of the different components and / or IoT devices through a monitoring middleware at runtime.

In addition, as an implementation of the proposed solution and as an additional deliverable, a DSL (*Domain Specific Language - Lenguaje de dominio especifico*) called Monitor-IoT has been developed, which provides the designer and developer of solutions applied to a case study with a tool for the development of architectures for the internet of things and allows analyzing them from the point of view of user perceptions.

Keywords: Middleware. Internet of Things. Models at Runtime. Cloud Computing.

Índice del Trabajo

Índice de Figuras

Índice de Tablas

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Ariana Isabel Roman Alvarado en calidad de autor/a y titular de los derechos morales y patrimoniales del trabajo de titulación "Creación de una infraestructura auto adaptativa de monitoreo mediante un middleware de configuración de componentes sensibles al contexto en ambientes de vida asistida", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 01 de abril del 2021

Ariana Isabel Roman Alvarado
C.I: 1105375222

Cláusula de Propiedad Intelectual

Ariana Isabel Roman Alvarado, autora del trabajo de titulación "Creación de una infraestructura auto adaptativa de monitoreo mediante un middleware de configuración de componentes sensibles al contexto en ambientes de vida asistida", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autora.

Cuenca, 01 de abril del 2021

Ariana Isabel Roman Alvarado

C.I: 1105375222

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Moyano Dután José Alfredo en calidad de autor/a y titular de los derechos morales y patrimoniales del trabajo de titulación "Creación de una infraestructura auto adaptativa de monitoreo mediante un middleware de configuración de componentes sensibles al contexto en ambiente de vida asistida", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 01 de abril de 2021

José Alfredo Moyano Dután

C.I: 0105751473

Cláusula de Propiedad Intelectual

Moyano Dután José Alfredo, autor/a del trabajo de titulación "Creación de una infraestructura auto adaptativa de monitoreo mediante un middleware de configuración de componentes sensibles al contexto en ambiente de vida asistida", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor/a.

Cuenca, 01 de abril de 2021

José Alfredo Moyano Dután

C.I: 010575173

Capítulo 1

1. Introducción

Este capítulo presenta la justificación del tema del trabajo de titulación denominado "Creación de una infraestructura auto adaptativa de monitoreo mediante un middleware de configuración de componentes sensibles al contexto en ambientes de vida asistida"; y se estructura de la siguiente manera. La sección 1.1 muestra la justificación por la cual se lleva a cabo el tema planteado como trabajo de titulación. La sección 1.2 presenta la problemática del tema propuesto. La sección 1.3 describe la solución que se busca conseguir para hacer frente a la problemática descrita. La sección 1.4 expone los objetivos planteados de la investigación, tanto el objetivo general, como los específicos. La sección 1.5 describe la metodología de investigación utilizada para el desarrollo del tema. Finalmente, la sección 1.6 detalla la estructura propuesta en el desglose de este trabajo.

1.1. Motivación y Contexto

Actualmente la tecnología forma parte de nuestra vida cotidiana, ya que esta ayuda a solventar diferentes necesidades y brinda soluciones que facilitan el desarrollo de actividades cotidianas, ya sean empresariales, educativas, de salud, ambientales, etc.

En este sentido, la tecnología brinda autosuficiencia a las personas frente a las tareas cotidianas, no sólo a usuarios comunes, sino también, a personas con habilidades y destrezas especiales, como es el caso de adultos mayores o personas con discapacidad; esto con la finalidad de mejorar su calidad de vida y otorgarles mayor independencia, ya sea en centros de cuidado o entornos residenciales familiares.

En los últimos años, se han realizado varias investigaciones acerca de tecnologías de asistencia, basadas en un nuevo paradigma de la tecnología de la información denominado "inteligencia ambiental"; este paradigma ayuda a potenciar las capacidades de las personas, a través de entornos digitales que son sensibles, adaptables y responden a las necesidades humanas (Rashidi & Mihailidis, 2012).

Según Augusto et al. (2010) se puede definir a la Inteligencia Ambiental como: *"Un entorno digital que apoya de manera proactiva pero sensata a las personas en su vida cotidiana"*. Este nuevo paradigma gira en torno a personas de avanzada edad, personas con diversidad funcional, personas con enfermedades crónicas, donde mejorar su calidad de vida mediante el desarrollo de tecnologías y servicios innovadores es la motivación principal (Mainetti et al., 2016).

En este sentido, del apoyo y soporte a los adultos mayores nace el concepto de ambientes de vida asistida (*AAL - Ambient Assisted Living*). Como ejemplos de sistemas que son parte de soluciones para AAL se encontró: sistemas para seguimiento a distancia de signos vitales (Oliveira et al., 2013), sistemas para la

detección de situaciones de desamparo en el hogar (Botia et al., 2012), servicios que incluyen asistencia en la rutina diaria, servicios de tratamiento de emergencia (Schneider & Becker, 2008), entre otros. Por lo tanto, AAL podría traducirse mejor como *"sistemas inteligentes de asistencia para una vida mejor y más segura"* (Mainetti et al., 2016).

Estos sistemas, ambientes o entornos de vida contienen diferentes "objetos" (electrodomésticos, sensores, actuadores, etc.) de la vida cotidiana interconectados entre sí; por los cuales, las personas interactúan, es decir, los "objetos" se comunican entre ellos para crear servicios que faciliten tareas en el hogar a partir de la detección de algún evento físico sin la necesidad de la intervención humana directa (Aazam & Huh, 2016).

La unión internacional de telecomunicaciones (*ITU - International Telecommunication Union*), define al internet de las cosas (*IoT – Internet of things*) como una infraestructura mundial para la sociedad de la información, que propicia la prestación de servicios avanzados mediante la interconexión de objetos (SECTOR & ITU, 2012). Entre sus servicios incluye monitoreo ambiental, monitoreo de atención médica, monitoreo industrial, monitoreo de tráfico, etc.; los cuales son aplicados en diferentes entornos (Aazam & Huh, 2016).

Sin embargo, los entornos modernos de ejecución de software se han vuelto cada vez más descentralizados, heterogéneos, inciertos y cambiantes. Lo que ha llevado a la comunidad de ingeniería de software a proponer formas innovadoras para construir, ejecutar y administrar sistemas y servicios que permitan que el software y sus modelos se puedan reconfigurar a sí mismo sin la necesidad de ser reconstruidos (Thiry & Schmidt, 2017).

En consecuencia, una de las ramas de la Informática que brinda importancia a los modelos, es la ingeniería dirigida por modelos (*MDE - Model-Driven Engineering*), una metodología de ingeniería de software que se enfoca en crear y explotar modelos de dominio, considerando los modelos como un componente central en el proceso de desarrollo de software, y de igual forma se utiliza para la documentación y comunicación de componentes de software (Szvetits & Zdun, 2016; Zhu et al., 2018).

Además, se ha centrado en el uso de modelos en las fases de diseño, implementación y despliegue durante el ciclo de vida de desarrollo de software. Sin embargo, a medida que los sistemas se vuelven más adaptables, reconfigurables y auto-gestionables, se necesitan modelos en tiempo de ejecución para abordar la complejidad de las adaptaciones dinámicas, a fin de mantener un modelo abstracto del sistema en ejecución.

En este sentido los modelos en tiempo de ejecución de forma progresiva se han convertido en un tema de investigación y aplicación en diversos contextos ya sea servicios cloud (Botta et al., 2016), mashups, hogares inteligentes (Weyns et al., 2018), administración de aplicaciones ubicuas, etc., con el objetivo de permitir que un

sistema ejecutándose pueda ser manipulado en tiempo de ejecución para un propósito en particular (Bencomo et al., 2013).

De acuerdo con lo mencionado por Bencomo et al. (2013) y Szvetits & Zdun (2016) se afirma que: determinar un modelo en tiempo de ejecución (*Model at runtime* - M@rt), conduce a identificar un objetivo y que requiere de técnicas y arquitecturas para respaldar la realización de dicho objetivo.

Por lo cual, los modelos en tiempo en ejecución se han utilizado ampliamente en diferentes sistemas para admitir la manipulación de datos, la reparación automática y la adaptación dinámica (Zhu et al., 2018).

En la literatura revisada, se han encontrado diferentes propuestas que utilizan modelos en tiempo real auto adaptativos en el dominio de AAL, lo que nos servirá como un preámbulo para la implementación de una infraestructura de monitoreo que ayude a la auto configuración de diferentes dispositivos IoT y a la recopilación y control de información de un entorno de vida asistida.

1.2. Planteamiento del Problema

El envejecimiento de la población genera un mayor número de ciudadanos con enfermedades crónicas. Estas personas requieren un control y supervisión constantes, de donde los sistemas tradicionales de salud pública no pueden proporcionar de manera satisfactoria (Da Xu et al., 2014; Sun et al., 2009). Los avances en el internet de las cosas (*IoT - Internet of things*) y el progreso tecnológico relacionado con los dispositivos móviles (p. ejem., teléfonos inteligentes, tabletas, etc.) y dispositivos portátiles (p. ejem., gafas de Google, relojes inteligentes, sensores en la ropa y el cuerpo, etc.) permiten crear sistemas que puedan proporcionar un entorno adecuado para incrementar el tiempo de vida de las personas de avanzada edad dentro de un AAL (Da Xu et al., 2014; Sun et al., 2009).

En este sentido, un AAL debe cumplir varios requisitos desafiantes, uno de estos es la capacidad de prestar sus servicios a un nivel de funcionalidad y calidad lo suficientemente alto como para permitir una vida independiente; en donde, se requiere una comprensión sólida de las necesidades de los usuarios y su entorno, y además la disponibilidad de los recursos IoT necesarios. Otro de los requisitos es la capacidad de adaptar y extender el comportamiento del sistema, ya que la asistencia personal de vida difiere entre los diferentes individuos (envejecimiento diferencial) y los cambios durante la vida de una persona (Schneider & Becker, 2008).

Con el fin de alcanzar estos requisitos, la investigación del estado del arte reveló una cantidad considerable de enfoques y soluciones adaptables, en donde los modelos en tiempo de ejecución han sido ya utilizados para la monitorización de varios tipos de características de sistemas pertenecientes a diversos dominios; así como también, para la monitorización de características no funcionales (Ardagna & Zhang, 2010) e implementaciones de middleware con diversos objetivos (Costa et al., 2006). Además, la investigación y planteamiento de soluciones continúan en constante crecimiento;

es por ello que en relación al tema de titulación propuesto se encontró un estudio que indica el funcionamiento que puede tener una implementación entre modelos en tiempo de ejecución y AAL; desafortunadamente, solo existe un nivel conceptual muy generalizado y no proporciona un aporte significativo.

En torno al internet de las cosas la investigación describe como una de las características más desafiantes asociadas con la amplia aplicación de IoT, a la heterogeneidad de los dispositivos (Davis, 1985; Ma, 2011); para lo cual Da Xu et al. (2014), describe acerca de la investigación actual el planteamiento de estándares técnicos de IoT para definir la especificación para el intercambio de información, el procesamiento y las comunicaciones entre las cosas, de tal forma que proporcionan interoperabilidad, compatibilidad, confiabilidad y operaciones efectivas a escala global.

Además el entorno dinámico de un sistema IoT, da lugar a otra brecha que permita solventar la capacidad de despliegue dinámico para lograr la adaptabilidad de los servicios de IoT en los entornos (Ma, 2011). Por lo tanto, se necesita abordar nuevas teorías, mecanismos y métodos de modelado de software adecuados para el entorno IoT. En consecuencia, el desafío de resolver la autoadaptación de la prestación del servicio IoT se plantea como un problema científico clave.

Por tal razón, el desafío en nuestro trabajo de titulación es *conseguir adaptar y auto configurar entidades digitales (cloud, gateway, dispositivos IoT) y entidades físicas (usuarios del sistema, entornos), de forma automática, en una arquitectura para IoT desplegada sin necesidad de detener el sistema en ejecución*. En este aspecto, para conseguir vencer el desafío, nuestra visión será enfocada mediante los modelos en tiempo de ejecución o M@rt con el objetivo de monitorear y recopilar datos, así como también para reducir el tiempo que tomaría reiniciar el sistema y perder información sensible del contexto.

1.3. Solución propuesta

En el presente trabajo de titulación, la solución propuesta plantea el diseño de una infraestructura enfocada al monitoreo, que facilite la auto adaptación de los diferentes componentes y/o dispositivos IoT, mediante un middleware de configuración en tiempo de ejecución.

Para el cumplimiento de la solución propuesta, se analizará la literatura existente sobre las diferentes infraestructuras de modelos en tiempo de ejecución en diversos ámbitos. Además, la infraestructura propuesta deberá estar constituida por dos características importantes sobre los componentes y/o dispositivos IoT: (i) su monitoreo y (ii) su autoconfiguración mediante el uso de un middleware.

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar una infraestructura auto adaptativa de monitoreo mediante un middleware de configuración de componentes sensibles al contexto en ambientes de vida asistida.

1.4.2. Objetivos Específicos

- Presentar un estudio del estado del arte a fin de conocer los avances existentes y que puedan ser de utilidad en la solución del problema.
- Diseñar el meta-modelo en tiempo de ejecución y una infraestructura auto adaptativa para el monitoreo de dispositivos IoT.
- Implementar una instancia de la solución propuesta, a fin de comprobar la factibilidad de la misma desde el punto de vista técnico.
- Evaluar empíricamente la infraestructura adaptativa de monitoreo propuesto mediante experimentos controlados, con la finalidad de proporcionar evidencias sobre la facilidad de uso y utilidad de los métodos y tecnologías propuestos.

1.5. Metodología de la investigación

La metodología seguida para la realización del presente trabajo sigue el modelo de transferencia tecnológica descrito por Gorschek et al. (2006). Este modelo incluye ocho actividades aplicadas en un proceso iterativo que busca las soluciones adecuadas y su correspondiente evaluación empírica que ayuda a dirigir esfuerzos hacia una solución realista. Las ocho actividades de la metodología aplicada son:

1. *Análisis del problema*: requiere comprender el problema que se busca resolver en el tema de investigación, para lo cual se debe observar el dominio que permitirá identificar y entender las necesidades de los usuarios interesados.

El dominio de nuestro trabajo de titulación son los modelos en tiempo de ejecución (M@rt - Models at runtime) desplegados con el internet de las cosas (*IoT - Internet of Things*) en entornos de ambientes de vida asistidos (*AAL - Ambient Assisted Living*).

Tomando en cuenta que en un ambiente de vida asistida el/los sistemas deben monitorear datos en tiempo real de forma constante y sin interrumpir su flujo cuando se requiera actualizar configuraciones; nuestra investigación se enmarca en conseguir configurar y/o actualizar información y/o dispositivos sin necesidad de detener el sistema.

2. *Formulación del problema*: una vez identificado el problema, éste se debe formular de manera clara y concisa a fin de incluir factores de contexto, planteamiento de objetivos y preguntas de investigación, con la finalidad de justificar el estudio realizado.

Para realizar actualizaciones o nuevas configuraciones de dispositivos del internet de las cosas en un sistema de ambiente de vida asistida, se plantea la creación de una infraestructura auto adaptativa de monitoreo que permita modelar y modificar todo el sistema en tiempo de ejecución.

3. *Revisión del estado del arte*: comprende un estudio de la literatura con el objetivo de conocer el estado actual de la investigación, donde se incluye soluciones y arquitecturas disponibles a fin de abordar y solventar brechas en el tema planteado.

Esta actividad se desarrolla aplicando las fases descritas por Kitchenham & Charters (2007), como son: i) planificación de la revisión, ii) ejecución de la revisión y iii) reporte o difusión de resultados.

A fin de conocer cómo se lleva la investigación en los estudios primarios donde se haga uso de los modelos en tiempo de ejecución en el dominio de internet de las cosas.

4. *Solución candidata*: se diseña una solución al problema identificado, mediante un metamodelo, un lenguaje de dominio especifico (*DSL - Domain Specific Language*) y la implementación de una infraestructura para monitoreo.

Para la implementación de la infraestructura, en primera instancia se realiza un metamodelo que permita definir el proceso de monitoreo de componentes en arquitecturas IoT. A partir del metamodelo se desarrolla un DSL para representar este tipo de arquitecturas de forma gráfica mediante la selección y configuración de componentes; como resultado se obtiene un DSL de la arquitectura IoT.

El DSL resultante es el archivo de entrada que sirve para configurar la infraestructura de monitoreo a la cual se ha denominado MIoT2InMon, nombre compuesto por las iniciales de los temas más relevantes de la investigación.

La infraestructura está formada por dos componentes principales el *Sincronizer* y el *Controller.* El *Sincronizer* procesa la representación gráfica de la arquitectura y envía información a los *Controllers* dentro de la arquitectura. El *Controller* comunica los elementos de la arquitectura para el monitoreo y la reconfiguración del entorno en tiempo de ejecución.

La solución presentada permite representar e implementar arquitecturas de monitoreo para IoT a nivel de DSL, sin preocuparse por la codificación o detener el sistema en ejecución para configuraciones posteriores.

5. *Entrenamiento*: consiste en desarrollar una etapa de tipo incremental, donde en las fases de entrenamiento se brinda el conocimiento base de la solución propuesta a los profesionales del área, de tal manera que tengan una idea clara de la infraestructura desarrollada.

Esta actividad consiste en una presentación de los temas más relevantes que engloba el DSL, así como, una descripción de las diferentes entidades y comunicaciones a través de un escenario de IoT. El escenario sirve como entrenamiento para entender

las clases que se plantean, cómo se deben comunicar y la forma en que se deben representar en el editor gráfico.

6. *Validación inicial*: se realiza un estudio de caso de entrenamiento con alumnos de la carrera de Ingeniería de Sistemas y Telemática.

La validación se lleva a cabo por un grupo de alumnos de los últimos ciclos de Ingeniería de Sistemas de la Universidad de Cuenca y estudiantes de la carrera de Telemática de la Universidad del Azuay, a fin de validar la facilidad de uso, utilidad e intención de uso futuro del DSL (*Monitor-IoT*) para desarrollar arquitecturas de monitoreo para IoT.

La validación inicial consiste en presentar el funcionamiento del DSL *(Monitor-IoT*) mediante un ejercicio de entrenamiento donde se plantea una arquitectura para IoT de un invernadero.

Los estudiantes en este ejercicio observan la forma en que se debe representar y configurar cada componente de la arquitectura para poder replicar el proceso en un ejercicio práctico.

El ejercicio práctico consiste en replicar una arquitectura para IoT de un ambiente de vida asistido y diseñar la solución utilizando la herramienta de Obeo Designer como el editor gráfico de DSL propuesto.

Como resultado se obtiene un DSL del escenario de un ambiente de vida asistida, el mismo que sirve como punto de partida para desplegar la infraestructura de monitoreo *MIoT2InMon*, que requiere información de la configuración de los dispositivos, servicios, API´S y el flujo de comunicación, de esta forma se constata el correcto propósito del presente trabajo de titulación.

7. *Validación realista*: se atribuye a los ambientes reales en el entorno industrial por medio de estudios de caso y/o experimentos controlados, desarrollando prototipos para la aplicación de la solución propuesta.

En el presente trabajo de titulación, esta actividad se plantea como parte de trabajos futuros.

8. *Liberación de la solución*: se realiza una valoración de los resultados obtenidos y se preparan las herramientas necesarias para su despliegue y uso posterior.

La solución final antes de la validación realista es: el metamodelo, el DSL y el código fuente de la infraestructura *MIoT2InMon* desarrollada.

Sin embargo, este paso al ser posterior a la validación realista, se deja planteado como trabajo futuro.

La Figura 1.1 presenta cada una de las actividades y el flujo iterativo de los procesos de la metodología seguida en el desarrollo del presente trabajo de titulación, dando cumplimiento a las seis primeras actividades y planteando como trabajo futuro para desarrollar las dos últimas actividades de validación realista y liberación de la solución.

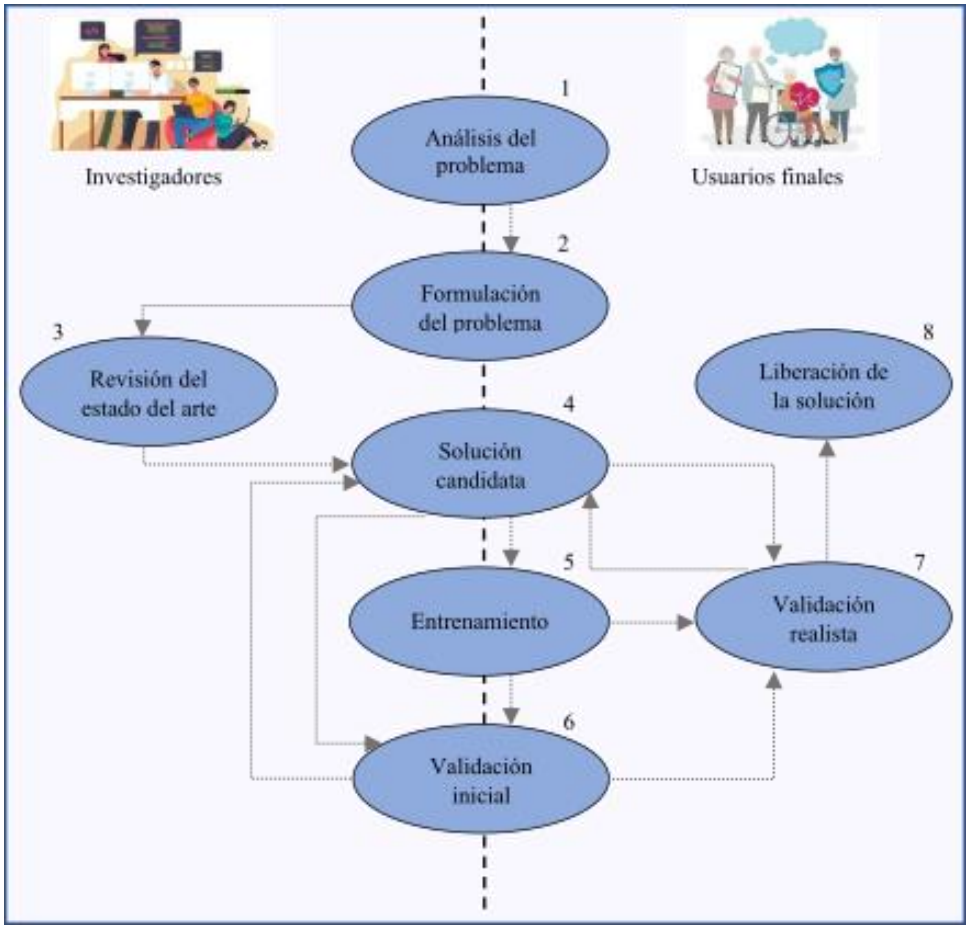


Figura 1.1. Metodología de la investigación de Gorschek et al. (2006). Fuente: Elaboración propia.

1.6. Estructura del Trabajo

En este ítem se describe la estructura del presente trabajo de titulación con una breve sinopsis de cada capítulo.

- Capítulo 1: Introducción.

Presenta la justificación, planteamiento del problema y la solución que se plantea, el objetivo general con los objetivos específicos de este trabajo de titulación, las hipótesis, la metodología considerada para llevar a cabo la investigación. Estos puntos dan cumplimiento a la primera y segunda actividad de la metodología adoptada relacionadas con el análisis y formulación del problema.

- Capítulo 2: Marco tecnológico.

Describe los términos, tecnologías y conceptos principales que forman parte de la propuesta para brindar una mejor comprensión del presente trabajo de titulación. Este capítulo se cumple en el marco de la actividad tres de la metodología seleccionada que corresponde a la revisión del estado del arte.

- Capítulo 3: Estado del arte.

El estado del arte se realiza basado en las fases propuestas por Kitchenham & Charters (2007), mediante un estudio sistemático de la literatura cuyo objetivo es la búsqueda, identificación y análisis de estudios primarios publicados en revistas y conferencias digitales relacionados al dominio en el que se enfoca el presente trabajo de titulación. El desarrollo de este capítulo, al igual que el capítulo anterior, forma parte de la actividad número tres de la metodología adoptada.

- ➢ Capítulo 4: Diseño e Implementación de la Infraestructura de Monitoreo.

Presenta el meta-modelo y el DSL que constituyen la infraestructura de monitoreo que se propone como solución candidata, dichas herramientas brindan la descripción y funcionamiento de cada uno de sus componentes. Este capítulo forma parte de las actividades cuatro y cinco de la metodología adoptada, donde se propone la solución candidata y el entrenamiento de la solución.

- ➢ Capítulo 5: Evaluación de un estudio de caso.

Este capítulo presenta la evaluación del DSL (Monitor-IoT) presentado en el capítulo 4 mediante un estudio de caso en la que intervienen alumnos de la carrera de Ingeniería de Sistemas y carrera de Telemática. Este capítulo se realiza conforme a la actividad seis de la metodología adoptada.

- ➢ Capítulo 6: Conclusiones y trabajos futuros.

Se presentan las conclusiones y reflexiones del trabajo de titulación, se recogen los principales problemas presentados y la manera en la que se dio solución a cada uno de ellos, se muestran las direcciones de investigación futura.

- ➢ Apéndices.

Presenta la documentación generada durante el desenvolvimiento del presente trabajo de titulación y se adjunta resultados del estudio de la literatura, diagramas, modelos, entre otros.

La Figura 1.2 indica el diagrama de la estructura del trabajo de titulación en el orden de cada capítulo juntamente con la actividad de la metodología adoptada.

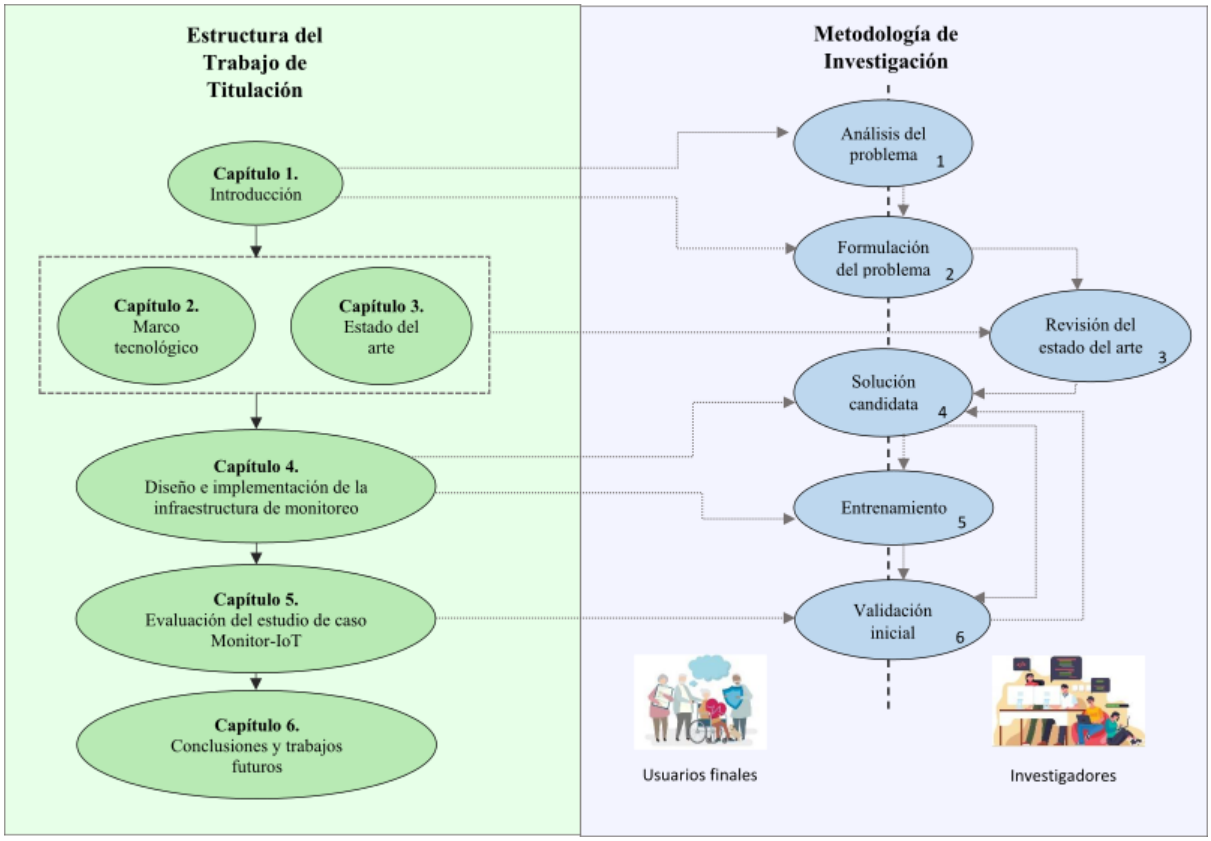


Figura 1.2. Estructura del trabajo de titulación. Fuente: Elaboración propia.

Capítulo 2

2. Marco tecnológico

En este capítulo se presentan los conceptos de los términos esenciales para llevar a cabo este trabajo de titulación. Estos conceptos brindarán un mejor entendimiento de los temas en los capítulos subsiguientes, de tal manera que se pueda conocer los conceptos y tecnologías involucradas.

La estructura de este capítulo se distribuye de la siguiente manera. En la sección 2.1 se define el término AAL o Ambiente de Vida Asistida y sus objetivos. La sección 2.2 presenta la definición de Internet de las Cosas, su arquitectura, características, dispositivos y protocolos. La sección 2.3 describe la definición de modelos en tiempo de ejecución, clasificación y arquitectura. La sección 2.4 presenta el significado de la computación en la nube (*Cloud Computing*). Finalmente, la sección 2.5 detalla el concepto de computación en la niebla (*Fog Computing*).

2.1. Ambiente de vida asistida

El informe general europeo de Ambiente de Vida Asistida (*AAL - Ambient Assisted Living*) define a los sistemas AAL como una respuesta a la pregunta de cómo permitir a las personas con necesidades especiales (por ejemplo, personas mayores o discapacitadas), vivir vidas independientes más largas en entornos residenciales familiares (Schneider & Becker, 2008; Sun et al., 2009).

En este sentido, se precisa una fuerte relación de un AAL con el paradigma de "inteligencia ambiental" el cual por medio del Internet de las cosas (*IoT - Internet of Things*) brinda una mayor seguridad y ayuda a potenciar las capacidades de las personas a través de entornos digitales que son sensibles, adaptables y responden a las necesidades humanas. Por lo tanto, AAL podría definirse mejor como "sistemas inteligentes de asistencia para una vida mejor y más segura" (Rashidi & Mihailidis, 2012; Schneider & Becker, 2008).

Estos sistemas inteligentes se pueden lograr por medio de IoT mediante el desarrollo de aplicaciones, para apoyar a los usuarios, que permitan el monitoreo de enfermedades crónicas (salud), suministro a pedido de alimentos frescos, de medicamentos (salud), sistemas de alarma (seguridad), servicios de recordatorio (tranquilidad) y la comunicación de persona a persona es decir con familiares (contacto social). Todo esto con la finalidad de controlar, prevenir, curar y mejorar el bienestar y las condiciones de salud en adultos mayores y personas con necesidades especiales (Dohr et al., 2010; Rashidi & Mihailidis, 2012).

Finalmente, Serral Asensio et al. (2014) precisa que los sistemas AAL generalmente están estructurados en tres niveles: hardware (detección, redes inalámbricas), middleware (captura de datos, seguridad de datos, integración de TI) y servicios

(procesamiento de bioseñal, orientado a la aplicación procesos, servicios comunitarios) y se caracteriza por estar conectado, adaptarse al contexto y anticiparse a posibles sucesos.

2.1.1. Objetivo y servicios

El objetivo principal de AAL es lograr beneficios para personas mayores o personas con necesidades especiales fomentando su autonomía, de tal forma que aumente la seguridad en su estilo de vida y en el entorno de su hogar.

Para cumplir el objetivo de los AAL se deben cumplir diversos requisitos, entre estos (i) la capacidad de prestar servicios a un nivel de calidad lo suficientemente alto como para mejorar la posibilidad de una vida independiente, considerando la situación actual de los usuarios y su entorno y la disponibilidad de los recursos necesarios, y (ii) la capacidad de adaptar y extender el comportamiento del sistema, ya que las demandas de asistencia para la vida difieren sustancialmente entre los diferentes individuos (envejecimiento diferencial) y los cambios durante la vida de una persona (Schneider & Becker, 2008).

Entre los servicios de asistencia que facilitan la vida diaria en el dominio ALL existe un alta gama que se pueden clasificar como lo describe (Schneider & Becker, 2008), mediante una estructura formada por seis subdominios estereotípicos con sus respectivas responsabilidades en cada dominio. En la Figura 2.1 se presentan los parámetros de clasificación utilizados, (i) Fila: ubicación donde se presta el servicio de vida asistida, y (ii) Columna: tipos de asistencia.

	Servicios de tratamiento de emergencia	**Servicios de mejora de la autonomía**	**Servicios de comodidad**
Asistencia Interior	Predicción Detección Prevención	Bebiendo Comiendo Cocinando Vistiéndose Medicándose	Servicios logísticos Servicios encontrando cosas Servicios de infoentretenimiento
Asistencia al aire libre	Predicción Detección Prevención	Asistencia de compras Asistencia de viaje Asistencia bancaria	Servicios de transporte Servicios de orientación

Figura 2.1. Esquema de clasificación para los servicios AAL. Fuente: (Schneider & Becker, 2008).

2.1.2. Requisitos de interfaz

Además de los servicios el considerar la combinación de requisitos no funcionales es un aspecto desafiante en los entornos AAL, debido al gran impacto que produce en

las personas mayores el aprender y acostumbrarse a la asistencia ofrecida, de tal modo que AAL tenga gran aceptación general del sistema y sea utilizado por los usuarios este debe cumplir los siguientes requisitos relacionados con la interfaz (Kleinberger et al., 2007):

2.1.2.1. Adaptabilidad

Es deseable que los sistemas en un AAL se adapten al contexto y mejor aún si esta adaptación se la realiza en tiempo de ejecución, de manera que presten servicios de la mejor forma posible acorde a la situación, contemplando requisitos de calidad, por ejemplo, reducen la complejidad de la interfaz en caso de emergencias para reducir la carga cognitiva del usuario (Kleinberger et al., 2007).

2.1.2.2. Interacción natural y anticipada humano-computador

AAL debe proporcionar interfaces accesibles para diferentes grupos de usuarios objetivo: personas que reciben asistencia, que tienen limitaciones y discapacidades, también personal médico y de atención, familiares y operadores de mantenimiento. En este sentido, con el fin de mejorar la usabilidad y accesibilidad los paradigmas de interacción multimodal, combinan varios modos, por ejemplo, gesto, sonido, entre otros (Kleinberger et al., 2007).

2.1.2.3. Heterogeneidad

Los sistemas de vida asistida deben integrar varios subsistemas proporcionados por diferentes fabricantes. Sin embargo, el usuario debe enfrentarse a una interfaz homogénea que integre los servicios de los diferentes subsistemas de manera transparente (Kleinberger et al., 2007).

2.2. Internet de las Cosas

Los avances y conceptos innovadores en tecnología de la información dan origen al fenómeno tecnológico denominado Internet de las cosas (*IoT - Internet of Things*), este fenómeno se asocia con diferentes conceptos que tienen un fuerte impacto para su continuo desarrollo como son: comunicación/conectividad ubicua, computación generalizada e inteligencia ambiental (Dohr et al., 2010).

La comunicación ubicua es la capacidad de los objetos para comunicarse (en cualquier lugar y en cualquier momento); la computación generalizada es la mejora de los objetos con poder de procesamiento (el entorno que nos rodea se convierte en la computadora) y la inteligencia ambiental es la capacidad de los objetos para registrar cambios en el entorno físico y, por lo tanto, interactuar activamente en un proceso.

En general, los objetos que cumplen con los requisitos antes mencionados, se denominan “objetos inteligentes”. Por tanto, se puede definir el IoT como la capacidad de los objetos inteligentes para comunicarse entre sí y formar parte de una red IoT (Dohr et al., 2010).

Da Xu et al. (2014) conceptualiza a IoT como una infraestructura de red global dinámica, con capacidades de autoconfiguración, basadas en protocolos de comunicación estándar e interoperables, en donde las “Cosas” físicas y virtuales tienen identidades, atributos físicos y usan interfaces inteligentes y se integran perfectamente en la red de información.

2.2.1. Dispositivos

El internet de las cosas consiste en interconectar diferentes tipos de dispositivos al Internet, los mismos que permiten la transmisión y captura de información. Entre los dispositivos más destacados se encuentran sensores y actuadores.

2.2.1.1. Sensores

Rayes & Salam (2017) definen un sensor como un dispositivo comúnmente electrónico encargado de detectar eventos o cambios en su entorno físico, sean estos eventos de temperatura, sonido, calor, presión, flujo, magnetismo, movimiento y parámetros químicos y bioquímicos, y proporciona la salida correspondiente.

Los sensores se pueden relacionar con los cinco sentidos humanos, porque forman el extremo principal de los dispositivos IoT, es decir las "Cosas". Los sensores son muy importantes en diferentes dominios IoT en donde son utilizados (p. ejem., ciudades inteligentes, redes inteligentes, cuidado de la salud, agricultura, seguridad y monitoreo ambiental y estacionamiento inteligente), porque son los encargados de unir los objetos físicos del mundo con la Internet (Rayes & Salam, 2017).

2.2.1.2. Actuadores

Los actuadores se definen por Rayes & Salam (2017) como un tipo de motor responsable de controlar o recoger medidas en un sistema, tomando una fuente de datos o energía como presión de fluido hidráulico, otras fuentes de energía, y además, convierten los datos/energía en movimiento para controlar un sistema.

Anteriormente se describieron los sensores como los responsables de detectar cambios en su entorno, recopilar datos relevantes y poner dichos datos a disposición de los sistemas de monitoreo. Por el contrario, los actuadores utilizan los datos recopilados y analizados por sensores para controlar los sistemas IoT, por ejemplo, cerrar el flujo de gas cuando la presión medida está por debajo de cierto umbral (Rayes & Salam, 2017).

2.2.1.3. Gateway IoT

El objetivo de un gateway IoT (puerta de enlace IoT) es conectar varias redes de dominio de detección con redes de comunicación pública o Internet, resolver la heterogeneidad entre estas diversas redes y fortalecer la gestión de la puerta de enlace de IoT y de los nodos terminales (Kang et al., 2017).

Una gateway IoT admite una variedad de protocolos de comunicación y tipos de datos entre varios sensores, que pueden realizar la conversión y unificación del formato de datos que se comunicará entre variedades de sensores (Kang et al., 2017).

2.2.2. Arquitectura de una aplicación

Da Xu et al. (2014) define desde la perspectiva de las funcionalidades un diseño arquitectónico de una aplicación IoT formada por cuatro capas: interfaz, servicio, red y detección, cómo se presenta en la Tabla 2.1, junto con la descripción de cada una.

CAPAS	DESCRIPCIÓN
Capa de Detección	Esta capa está integrada con el hardware existente (RFID, sensores, actuadores, etc.) para detectar / controlar el mundo físico y adquirir datos
Capa de Red	Esta capa proporciona soporte de red básica y transferencia de datos a través de una red inalámbrica o cableada.
Capa de Servicio	Esta capa crea y gestiona servicios. Proporciona servicios para satisfacer las necesidades del usuario.
Capa de Interfaz	Esta capa proporciona métodos de interacción a los usuarios y otras aplicaciones.

Tabla 2.1. Arquitectura de cuatro capas para IoT. Fuente: (Da Xu et al., 2014).

2.2.3. Protocolos

Las diferentes tecnologías de comunicación utilizadas para IoT se comunican a través de la red utilizando un conjunto diverso de protocolos y estándares. Un protocolo es un estándar utilizado para definir un método de intercambio de datos a través de una red informática, como una red de área local, Internet, Intranet, etc.

Para transferir datos entre dispositivos IoT, con un rango de alcance medio, se utilizan los protocolos de comunicación, mientras que para transmitir información máquina a máquina (*M2M - Machine to machine*) se utilizan los protocolos de aplicación.

2.2.3.1. Protocolos de Comunicación

Los protocolos de comunicación más utilizados para IoT son: Wifi, Bluetooth, IEEE 802.15.4, Z-wave y LTE-Advanced.

WiFi, utiliza ondas de radio para intercambiar datos entre cosas dentro del rango de 100m y permite que los dispositivos inteligentes se comuniquen e intercambien información (Al-Fuqaha et al., 2015).

Bluetooth, presenta una tecnología de comunicación que se utiliza para intercambiar datos entre dispositivos en distancias cortas utilizando radio de onda corta para minimizar el consumo de energía. Recientemente, el grupo de interés especial de Bluetooth (SIG) produjo Bluetooth 4.1 que proporciona Bluetooth de baja energía, así como conectividad IP y de alta velocidad para soportar IoT (Al-Fuqaha et al., 2015).

IEEE 802.15.4, Al-Fuqaha et al. (2015) lo describe como un estándar el cual especifica una capa física y un control de acceso medio para redes inalámbricas de baja potencia dirigidas a comunicaciones confiables y escalables.

LTE (*Long-Term Evolution - evolución a largo plazo*), es originalmente una comunicación inalámbrica estándar para la transferencia de datos a alta velocidad entre teléfonos móviles basada en tecnologías de red GSM/UMTS. LTE-A (*LTE Advanced*) es una versión mejorada de LTE que incluye extensión de ancho de banda que admite hasta 100 MHz, multiplexación espacial de enlace descendente y ascendente, cobertura extendida, mayor rendimiento y latencias más bajas (Al-Fuqaha et al., 2015).

2.2.3.2. Protocolos de aplicación

Los protocolos de aplicación más utilizados son: MQTT, CoAP, HTTP.

MQTT (*Message Queue Telemetry Transport - Transporte de telemetría de la cola de mensajes*) fue lanzado por IBM y apunta a comunicaciones ligeras M2M. Es un protocolo de publicación/suscripción asincrónica, que se ejecuta sobre la pila TCP. Los protocolos de publicación/suscripción cumplen mejor los requisitos de IoT que la solicitud/respuesta ya que los clientes no tienen que solicitar actualizaciones, por lo tanto, el ancho de banda de la red y la necesidad de usar recursos computacionales disminuye (Karagiannis et al., 2015).

En MQTT hay un agente(servidor) que contiene temas. Cada cliente puede ser un editor que envía información al agente sobre un tema específico y/o un suscriptor que recibe mensajes automáticos cada vez que hay una nueva actualización en un tema que está suscrito (Karagiannis et al., 2015).

CoAP (*Constrained Application Protocol - Protocolo de aplicación restringido*) es un protocolo de capa de aplicación de solicitud/respuesta síncrona que fue diseñado por el grupo de trabajo de ingeniería de Internet (IETF) para apuntar a dispositivos de recurso restringido. Fue diseñado usando un subconjunto de los métodos HTTP haciéndolo interoperable para utilizar los métodos GET, POST, PUT y DELETE, de tal forma que permita proporcionar interacciones orientadas a los recursos en una arquitectura cliente-servidor (Karagiannis et al., 2015).

A diferencia de MQTT, CoAP usa un identificador universal de recursos (*URI - Universal Resource Identifier*) en lugar de temas, por lo cual, el editor publica datos en el URI y el suscriptor se suscribe a un recurso particular indicado por el URI, por tanto cuando un editor publica nuevos datos en el URI todos los suscriptores son notificados sobre el nuevo valor según lo indicado por el URI (Naik, 2017).

HTTP (*HiperText Transfer Protocol - protocolo de transferencia de hipertexto*) es un protocolo de mensajería web, desarrollado originalmente por Tim Berners-Lee. Más tarde, fue desarrollado por IETF (*Internet Engineering Task Force - grupo de trabajo de ingeniería de internet*) y W3C (*World Wide Web Consortium - consorcio mundial de la red*) conjuntamente y publicado por primera vez como un protocolo estándar en 1997.

Admite arquitectura web RESTful (*Representational state transfer - transferencia de estado representacional*) de solicitud/respuesta y de forma análoga a CoAP, HTTP utiliza el URI, de modo que el servidor envía datos a través del URI y el cliente recibe datos a través de un URI particular.

HTTP es un protocolo basado en texto y no define el tamaño de las cargas útiles de encabezado y mensajes, sino que depende del servidor web o de la tecnología de programación. Utiliza TCP como protocolo de transporte predeterminado y TLS/SSL por seguridad, por lo tanto, la comunicación entre el cliente y el servidor está orientada a la conexión (Naik, 2017).

2.2.4. Aplicaciones en la industria

En IoT es importante mencionar las aplicaciones desarrolladas en la Industria mediante el uso de diferentes dispositivos inteligentes, las cuales son:

2.2.4.1. IoT en la industria de servicios de salud

Los servicios de salud basados en IoT permiten que los objetos en los sistemas de salud (p. ejem., personas, equipos, medicamentos, etc.) puedan ser rastreados y monitoreados de forma constante gracias a su conectividad global, de tal forma, que toda la información relacionada con la atención médica (logística, diagnóstico, terapia, recuperación, medicación, gestión, finanzas e incluso la actividad diaria) se pueda recopilar, administrar y compartir de manera eficiente (Da Xu et al., 2014).

Por ejemplo, la frecuencia cardíaca de un paciente puede ser recolectada por sensores y luego enviada al consultorio del médico mediante el uso de dispositivos informáticos personales (p. ejem., computadora portátil, teléfono móvil, tableta) y acceso a Internet móvil (p. ejem., WiFi, 3G, LTE), todo eso, gracias a la amplia difusión del servicio de Internet móvil que ha permitido acelerar el desarrollo de los servicios de atención médica a domicilio con IoT, enfocándose siempre en la seguridad y privacidad de la información (Da Xu et al., 2014).

2.2.4.2. IoT para ciudades inteligentes

El término “ciudades inteligentes” se utiliza para denotar el ecosistema ciber físico emergente mediante el despliegue de infraestructura de comunicación avanzada y servicios novedosos en escenarios de toda la ciudad.

Las tecnologías de IoT pueden encontrar varias aplicaciones diversas en escenarios de ciudades inteligentes, por ejemplo, proporcionar sistemas avanzados de control de tráfico que permita monitorear el tráfico de automóviles en grandes ciudades o

carreteras y desplegar servicios que ofrezcan consejos de enrutamiento de tráfico para evitar la congestión, en esta perspectiva, se entenderá que los automóviles representan “objetos inteligentes”.

Además, sistemas de dispositivos de estacionamiento inteligente, basados en RFID (*Radio Frequency Identification - Identificación por radiofrecuencia*) y tecnologías de sensores, para identificar los espacios de estacionamiento disponibles y brindar a los conductores asesoramiento de estacionamiento automatizado, a fin de mejorar la movilidad en el área urbana (Miorandi et al., 2012).

Además, los sensores pueden monitorear el flujo del tráfico vehicular en las carreteras y recuperar información agregada, como la velocidad promedio y la cantidad de automóviles. (Atzori et al., 2010; Miorandi et al., 2012).

Otra de las aplicaciones de IoT es el hecho de poder emplear los dispositivos IoT con el objetivo de detectar el nivel de contaminación del aire; a fin de otorgar información a las entidades de salud para el análisis y planteamientos estratégicos para reducir la contaminación en las ciudades (Miorandi et al., 2012).

2.2.4.3. IoT en la lucha contra incendios

IoT se ha utilizado en el campo de seguridad contra incendios para detectar posibles incendios y proporcionar una alerta temprana ante posibles desastres por incendios. Al aprovechar las etiquetas RFID, los lectores RFID móviles, las cámaras de video inteligentes, las redes de sensores y las redes de comunicación inalámbricas; la autoridad competente de supervisión de incendios u organizaciones relacionadas podrían realizar un diagnóstico automático para realizar un monitoreo ambiental, una alerta temprana de incendios y un rescate de emergencia en tiempo real según sea necesario (Da Xu et al., 2014).

2.3. Modelos en Tiempo de Ejecución

Según Cetina et al. (2009) un modelo en tiempo de ejecución en un enfoque de Ingeniería Orientada a Modelos (*MDE - Model Driven Engineering*) es una abstracción o representación simplificada de un sistema que se construye con propósitos específicos, a los cuales se les asigna tareas en tiempo de ejecución, de tal manera que por medio de una relación causal, si el sistema cambia el modelo cambia o viceversa (Giese et al., 2014).

Una definición alternativa es descrita por Bencomo et al. (2013) donde define un modelo en tiempo de ejecución como una abstracción de un sistema ejecutándose que puede ser manipulado en tiempo de ejecución para un propósito en particular.

De igual forma Alférez & Pelechano (2012) describen los modelos en tiempo de ejecución como auto-representaciones causalmente conectadas del sistema asociado que enfatizan la estructura, el comportamiento o los objetivos del sistema desde una perspectiva del espacio del problema. En respuesta a los cambios en el contexto, el propio sistema puede consultar estos modelos para determinar las modificaciones necesarias en la arquitectura subyacente.

En este contexto Lehmann et al. (2010) establece como propiedades típicas de un modelo en tiempo de ejecución las siguientes:

- Una parte prescriptiva del modelo, donde se precisa cómo debería ser.
- Una parte descriptiva del modelo, que detalle cómo el modelo está
- Las modificaciones válidas en el modelo de las partes descriptivas, ejecutables en tiempo de ejecución.
- Las modificaciones válidas de las partes descriptivas, ejecutables en tiempo de ejecución.
- Las modificaciones válidas de las partes prescriptivas, ejecutables en tiempo de ejecución.
- La conexión causal entre el modelo y el sistema representado en forma de un flujo de información.

2.3.1. Objetivos

Los objetivos perseguidos por un sistema, el cual utiliza modelos en tiempo de ejecución según la clasificación propuesta por (Szvetits & Zdun, 2016) se dividen en siete clases (Figura 2.2): adaptación, abstracción, independencia con la plataforma, consistencia y conformidad, chequeo y cumplimiento de políticas, manejo de errores y por último monitorización, simulación y predicción.

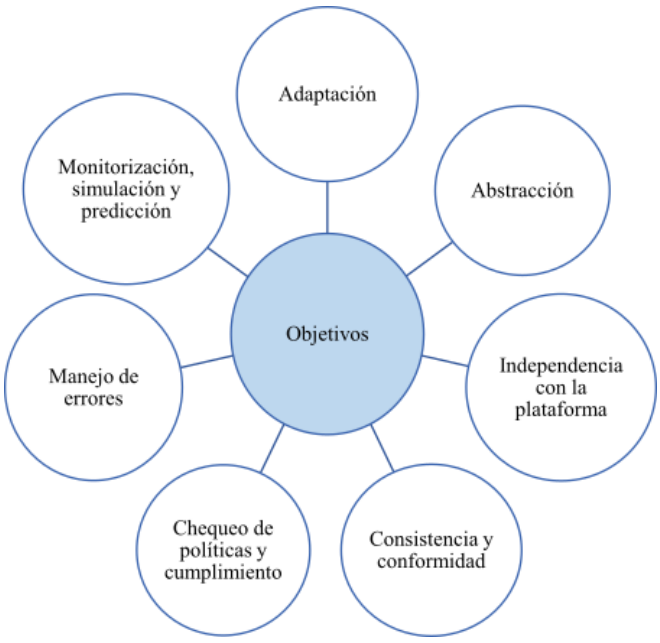


Figura 2.2. Objetivos de los modelos en tiempo de ejecución (Szvetits & Zdun, 2016). Fuente: Elaboración propia.

2.3.1.1. Adaptación

La adaptación, que consiste en cambiar el sistema de acuerdo a los cambios de ambientes y requisitos.

En este campo, Szvetits & Zdun (2016) describen dos soluciones. Por un lado, se tiene un proceso de desarrollo de software para soportar evoluciones consistentes

con la ayuda de modelos de contexto, modelos de características y un ciclo de control. Por otro lado, está AMOEBA-RT, que es un verificador de modelos para sistemas adaptativos que recopila información del estado de ejecución a través de técnicas orientadas a aspectos y verifica violaciones a especificaciones formales. AMOEBA-RT utiliza un enfoque basado en autómatas para determinar si la información de tiempo de ejecución satisface ciertas propiedades especificadas en la lógica temporal.

Szvetits & Zdun (2016) presenta cinco escenarios principales: la adaptación de la interfaz de usuario, los cambios de requisitos, los cambios contextuales, el cumplimiento de QoS (Calidad de Servicio) y el análisis de impacto de cambio.

La adaptación de la interfaz de usuario aborda la interacción dinámica de la interfaz de usuario, el diseño y la gestión de múltiples perspectivas del sistema. Una solución para la adaptación de la interfaz de usuario, presentada por Garzon & Cebulla (2010) utiliza una combinación de tres modelos para admitir la adaptación automática por medio del aprendizaje de las interacciones del usuario: un modelo de sistema, un modelo de interacción y un modelo de personalización.

Los requisitos y los cambios contextuales abordan las modificaciones necesarias para satisfacer la evolución de los requisitos y los cambios en los entornos operativos. Una solución para este escenario, es el lenguaje de requisitos RELAX, creado para sistemas auto adaptativos que admite la expresión explícita de incertidumbre ambiental en los requisitos (Whittle et al., 2009). RELAX se basa en una lógica temporal de ramificación difusa y proporciona operadores modales, temporales y ordinales para expresar la incertidumbre.

En el escenario de cumplimiento de QoS, se aborda la satisfacción de las propiedades no funcionales. Una solución utilizada para cumplir los requerimientos no funcionales de eficiencia y efectividad es MADAM. MADAM automatiza la toma de decisiones de adaptación en sistemas basados en componentes reflexivos para maximizar la utilidad de una aplicación, por ejemplo, a través del consumo mínimo de recursos, el consumo máximo de recursos o la máxima eficiencia (Alférez & Pelechano, 2012).

De igual forma, Cedillo et al. (2015) proponen el diseño de un middleware de monitoreo basada en el uso de modelos en tiempo de ejecución independiente de la plataforma para servicios en la nube, que respalda el monitoreo del cumplimiento de acuerdos de nivel de servicio (*SLA - Service Level Agreements*) y proporciona un informe que contiene violaciones de SLA que pueden ayudar a las partes interesadas a tomar decisiones sobre cómo mejorar la calidad de los servicios en la nube.

El middleware permite el cambio dinámico de los requisitos de calidad y/o la selección dinámica de diferentes operaciones métricas (fórmulas de cálculo) con las que medir la calidad de los servicios. Y para demostrar la viabilidad del enfoque, realizan la creación de instancias del middleware propuesto que se puede utilizar para monitorear los servicios cuando se implementa en la plataforma Microsoft Azure.

Para complementar, P Cedillo (2016) en su tesis doctoral presenta el método de monitorización de servicios cloud (Cloud MoS@RT) refinado a partir del estudio

anterior (Cedillo et al., 2015). "*El método está basado en la técnica del bucle autonómico de control de los modelos en tiempo de ejecución, cuyo principio es medir los parámetros del sistema, analizarlos, planear acciones correctivas si es necesario y ejecutar esas acciones para mejorar el sistema*".

Cloud MoS@RT proporciona un alto grado de flexibilidad en la especificación de requisitos de monitorización de servicios cloud y permite cambiar dinámicamente los requisitos de monitorización sin la necesidad de intercambiar y/o re-implementar la infraestructura de monitorización, todo esto debido a que cualquier modificación en los requisitos de monitorización se incluyen en el modelo en tiempo de ejecución (P Cedillo, 2016).

Finalmente, el escenario de análisis de impacto de cambio, aborda la detección de partes del sistema afectadas en caso de requisitos o cambios ambientales, así como las consecuencias que surgen de las transiciones a estados futuros.

2.3.2. Abstracción

La segunda, la abstracción, que interactúa con el sistema usando modelos que son cercanos al espacio del problema.

Levantar el nivel de abstracción se define como uno de los objetivos más destacados de los modelos de software sin importar si será utilizado en tiempo de diseño o en tiempo de ejecución. Con la llegada del desarrollo dirigido por modelos, los modelos independientes de la plataforma han ganado una atención considerable (Szvetits & Zdun, 2016).

2.3.3. Independencia con la plataforma

Como tercer punto, la Independencia con la plataforma, que provee vistas independientes de la plataforma sobre el sistema bajo observación.

2.3.4. Consistencia y conformidad

La cuarta es la consistencia y conformidad, que evita contradicciones entre diferentes partes del software y/o artefactos de desarrollo. Asegura la conformidad hacia otros modelos o restricciones de integridad en general.

La consistencia asegura que no existan contradicciones entre las diferentes partes de un sistema de software y los artefactos de desarrollo relacionados con el sistema. La conformidad asegura que un sistema de software reúna un estándar específico. Los requisitos de consistencia y conformidad vienen desde diferentes clases de recursos, tales como artefactos de diseño, combinaciones de características válidas, otros modelos como un modelo de sincronización o políticas de integridad (P Cedillo, 2016).

Los modelos en tiempo de ejecución pueden ayudar a chequear o asegurar los requisitos de consistencia o conformidad, puesto que ponen a disposición la información del modelo en tiempo de ejecución, sea esta información de las reglas de consistencia y conformidad o los artefactos del sistema a ser chequeados.

2.3.5. Verificación y cumplimiento de políticas

La verificación y cumplimiento de políticas incorporan los modelos usados en tiempo de ejecución para hacer frente a políticas, tales como restricciones en tiempo real, control de acceso y regulaciones de seguridad u otras reglas de cumplimiento.

Algunos tipos de modelos en tiempo de ejecución ayudan a satisfacer tales políticas: Modelos de seguridad describen propiedades de seguridad de los sistemas, modelos de políticas de control de acceso basados en roles y modelos en tiempo real que soportan el modelado de restricciones de tiempo a ser preservados en tiempo de ejecución (P Cedillo, 2016).

2.3.6. Manejo de errores

La sexta es el manejo de errores, los modelos en tiempo de ejecución como: modelos de máquina de estados y modelos de flujo de trabajo comúnmente son utilizados por su desempeño en la localización de errores, trazabilidad y autocorrección en ambientes cambiantes. Estos modelos facilitan la revisión contra trazas de ejecución dadas, obtenidas por eventos emitidos o registros de ejecución.

2.3.7. Monitorización, simulación y predicción

Finalmente, la monitorización, simulación y predicción, que ayuda a monitorizar el sistema a través de modelos y trazando el comportamiento de la aplicación, simula cambios a nivel de modelo y analiza sus consecuencias, y predice propiedades del sistema tales como rendimiento por análisis a nivel de modelo.

Los modelos de tiempo de ejecución cumplen el objetivo de la monitorización, donde simulan evoluciones futuras mediante el análisis de impactos en el nivel del modelo y la predicción de propiedades del sistema como el rendimiento y la confiabilidad en caso de reconfiguración del sistema (Szvetits & Zdun, 2016).

Estos modelos efectúan los tres objetivos a un nivel de abstracción cercano al espacio del problema, más específicamente, estos están estrechamente relacionados con la adaptación, ya que la mayoría de las técnicas de adaptación requieren el monitoreo del sistema en ejecución. Además, la simulación y la predicción se utilizan como entrada para las adaptaciones (Szvetits & Zdun, 2016).

2.3.8. Técnicas

Szvetits & Zdun (2016) clasifica las técnicas utilizadas en combinación con modelos de tiempo de ejecución como se presenta en la Figura 2.3: (i) lazo de control autónomo para analizar el sistema en ejecución y planificar acciones correctivas; (ii) introspección enfocado en extraer datos de un sistema en ejecución; (iii) conformidad del modelo que permita garantizar la conformidad y la coherencia con los modelos; (iv) comparación de modelos a fin de verificar diferencias entre dos modelos; (v) transformación de modelos que ayuda en el cambio de representación de modelos; y (vi) ejecución del modelo para ejecutar modelos con semántica operativa.

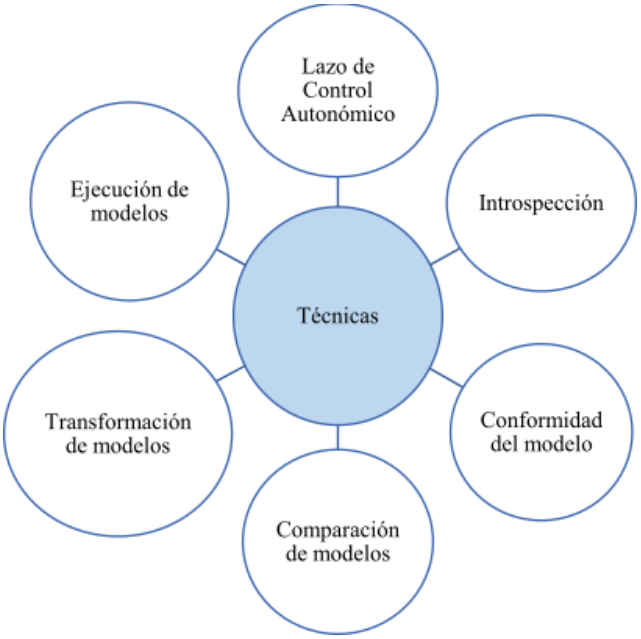


Figura 2.3. Técnicas de los modelos en tiempo de ejecución (Szvetits & Zdun, 2016). Fuente: Elaboración propia.

2.3.8.1. Bucle de control automático

Su principal función se basa en el análisis, ejecución del sistema y planificación de acciones correctivas.

El bucle de control automático es considerado un concepto clave para adaptar sistemas en tiempo de ejecución. La idea de utilizar esta técnica consiste en medir parámetros del sistema, analizarlos, planificar acciones correctivas si es necesario y ejecutarlas. La Figura 2.4 ilustra el funcionamiento del bucle de control automático.

Figura 2.4. Bucle de control automático (P Cedillo, 2016). Fuente: Elaboración propia.

Este procedimiento se conoce como ciclo Monitoreo-Análisis-Planeación-Ejecución (*MAPE/MAPE-K*) con un componente de conocimiento compartido. El componente de planificación del bucle debe identificar configuraciones alternativas (es decir, una disposición de las partes del sistema) que satisfacen los requisitos y restricciones contextuales actuales. De forma alternativa Vogel & Giese (2011) describen esta técnica como útil para diseñar un bucle de retroalimentación que controla la

autoadaptación mediante el monitoreo y análisis del sistema en ejecución y su entorno, y la planificación y ejecución de cambios en el sistema en ejecución.

De igual forma (Garlan et al., 2004) introduce un enfoque análogo donde su marco utiliza modelos de arquitectura abstracta para monitorear un sistema, verificar violaciones de restricciones y activar la adaptación a nivel global y de módulo si en algún caso esto fuera necesario.

2.3.8.2. Introspección

La introspección se encarga de la extracción de datos del sistema en tiempo de ejecución para efectuar técnicas basadas en modelos y bucles de control al comportamiento analizado (Szvetits & Zdun, 2016). Una subdivisión de los mecanismos de introspección se plantea en tres grupos diferentes:

- Comprobación del registro de eventos

Lleva a cabo la extracción de datos escaneando las entradas del registro de eventos, las cuales deben seguir una estructura definida, deben procesarse externamente y por tanto, estar relacionadas con representaciones de modelos en tiempo de ejecución, las mismas que serán emitidas por el sistema en observación.

En concreto, esta técnica permite verificar la consistencia entre el sistema en ejecución y el modelo, para lo cual efectúa tareas de identificación y abstracción de las partes relevantes del registro de eventos (Szvetits & Zdun, 2016).

- Instrumentación

En la instrumentación, la funcionalidad de monitorización está insertada en el sistema bajo observación. La inserción del código de monitorización requiere que el código fuente esté disponible y es a menudo realizado usando programación orientada a aspectos. La extracción de los datos está relacionada a las representaciones en tiempo de ejecución de los modelos para habilitar el razonamiento a un alto nivel de abstracción (P Cedillo, 2016).

- API’s de administración

Las API’s de administración incluyen una interfaz en el sistema de destino que permite la extracción de información del estado en tiempo de ejecución. Una interfaz de ejemplo son las API de reflexión de lenguajes de programación modernos que permiten realizar consultas y modificar la estructura de un programa de computadora en tiempo de ejecución (Szvetits & Zdun, 2016).

2.3.8.3. Conformidad del modelo

Esta técnica aborda la conformidad y consistencia de los datos y procesos de acuerdo con los modelos. La misma que está estrechamente relacionada con los objetivos de los modelos en tiempo de ejecución descritos en la sección 2.3.7, de consistencia y conformidad, verificación, cumplimiento de políticas y manejo de errores, dado que la información recopila en tiempo de ejecución se verifica en relación a los modelos a fin de reconocer violaciones de restricciones y comportamiento del sistema.

La técnica de introspección en combinación con representaciones en tiempo de ejecución es requerida por la conformidad del modelo para el procesamiento en tiempo de ejecución de los modelos (Szvetits & Zdun, 2016).

2.3.8.4. Comparación de modelos

Se enfoca en la comparación de dos modelos, a fin de proporcionar información sobre sus diferencias y operaciones necesarias para transformar un modelo en otro modelo. Está técnica se utiliza principalmente para calcular las operaciones necesarias para las transiciones entre estados del sistema en tiempo de ejecución.

Comparación del marco de modelado de Eclipse (*EMF Compare - Eclipse Modeling Framework Compare*) es una de las herramientas más utilizadas para proporcionar la funcionalidad de comparación de modelos, además existen alternativas como el marco de modelado de Kevoree (*KMF - Kevoree Modeling Framework*) que ayuda a mejorar el rendimiento, reducir la huella de memoria y eliminar las dependencias de la biblioteca (Szvetits & Zdun, 2016).

2.3.8.5. Transformación del modelo

Está técnica se utiliza para cambiar la representación de un modelo de origen en un modelo de destino, con el objetivo de reducir los errores a través de la automatización, al tiempo que se garantiza la coherencia entre el modelo de origen y el de destino.

Entre los tipos de transformación de modelo se encuentra: transformación de modelo a texto (*M2T - Model to Text*) es la transformación de un modelo en una representación textual y, transformación de modelo a modelo (*M2M - Model to Model*) donde su salida es otro modelo. Para llevar a cabo este tipo de transformaciones se utilizan herramientas y lenguajes establecidos como son: consulta-vista-transformación (*QVT - Query-View-Transformation*) y gramática de triple grafo (*TGG - Triple Graph Grammar*) (Szvetits & Zdun, 2016).

2.3.8.6. Ejecución del modelo

La ejecución del modelo se define por Szvetits & Zdun (2016) como una técnica separada, debido a que los cambios interpretados en un modelo en tiempo de ejecución implican directamente cambios en el comportamiento del sistema en ejecución; esta propiedad explícitamente da lugar a la definición de una conexión causal. En este sentido, esta técnica colabora a la ejecución directa de modelos que presentan semántica operativa (Gjerlufsen et al., 2009).

2.3.9. Arquitecturas

Para el procesamiento de modelos en tiempo de ejecución es fundamental una arquitectura formada por capacidades introspectivas para extraer información relevante de tiempo de ejecución y relacionar los datos recopilados con los modelos. Los modelos en tiempo de ejecución operan en un nivel más alto de abstracción orientados al espacio problemático (Blair et al., 2009).

El estudio realizado por Szvetits & Zdun (2016) define cinco principales arquitecturas aplicadas en los modelos en tiempo de ejecución. La Figura 2.5 especifica las arquitecturas, las cuales son: Arquitectura monolítica, Arquitectura de flujo de datos local, Arquitectura Middleware "consciente del modelo", Arquitectura Middleware de comunicación y Arquitecturas de repositorio.

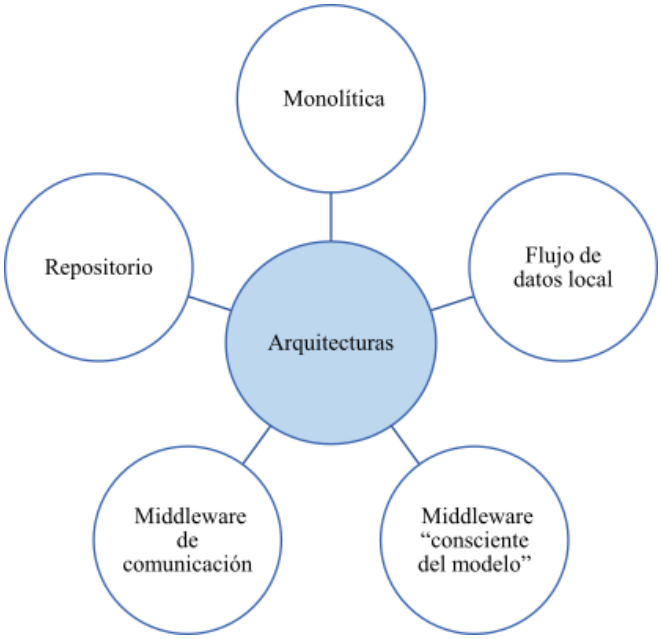


Figura 2.5. Arquitecturas de los modelos en tiempo de ejecución (Szvetits & Zdun, 2016). Fuente: Elaboración propia.

2.3.9.1. Arquitectura monolítica

En una arquitectura monolítica, toda la funcionalidad del sistema se captura en una sola unidad sin separación en múltiples componentes del sistema (Szvetits & Zdun, 2016). En los modelos en tiempo de ejecución, se dice que emplea una arquitectura monolítica si el modelo en tiempo de ejecución es accedido localmente y/o manipulado por el propio sistema en ejecución o está directamente integrado en un componente de software.

La Figura 2.6 muestra tres ejemplos de acceso al modelo en tiempo de ejecución que usan una arquitectura monolítica: (a) un sistema que consume un archivo del modelo local (por ejemplo, en formato XMI), (b) un sistema que tiene acceso a un modelo en memoria (por ejemplo, a través del código de modelo) y (c) un componente que tiene información del modelo incrustado con una interfaz de comunicación proporcionada.

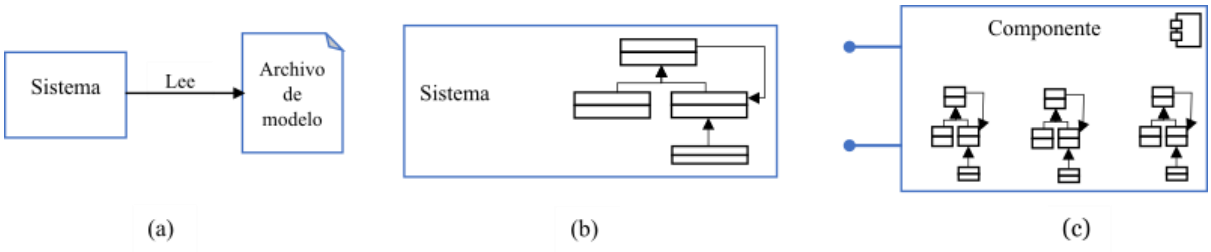


Figura 2.6. Ejemplos de acceso al modelo al usar una arquitectura monolítica. (a) Acceda a un archivo de modelo. (b) Acceso a un modelo en memoria. (c) Componente con modelo incrustado (Szvetits & Zdun, 2016). Fuente: Elaboración propia.

2.3.9.2. Arquitectura de flujo de datos local

Las arquitecturas locales de flujo de datos se utilizan para descomponer una tarea local en diferentes subtareas con menor complejidad, compensando así las desventajas de las arquitecturas monolíticas introducidas por la falta de acoplamiento.

El flujo de datos entre subtareas locales puede implementarse a través de mecanismos de comunicación entre procesos y en proceso, como sockets y subprocesos. La Figura 2.7 muestra dos ejemplos, de pipes y filtros y comunicaciones de socket, que ayudan a describir una posible representación de la arquitectura de flujo de datos.

En la Figura 2.7(a), la arquitectura de pipes y filtros se utiliza para la posterior transformación de la información recopilada del sistema en datos a visualizar en un modelo, lo que permite el monitoreo y la simulación a través de un componente reutilizable.

La Figura 2.7(b) muestra la comunicación entre un sistema y una parte del controlador compatible con el modelo a través de sockets. Dado que las comunicaciones de socket son bidireccionales, esta arquitectura califica para enfoques dinámicos con capacidades de adaptación, como lo señala la flecha de doble punta.

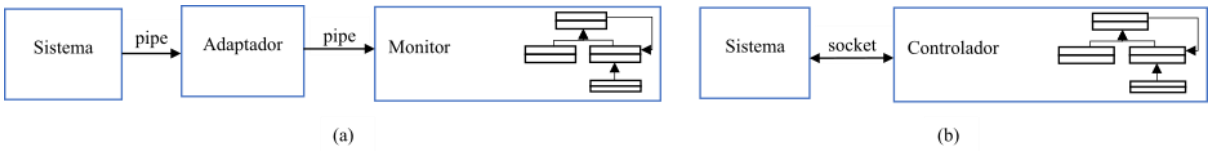


Figura 2.7. Comunicación entre procesos para separar el modelo del sistema bajo observación. (a) Comunicación de pipes y filtros. (b) Comunicación de socket (Szvetits & Zdun, 2016). Fuente: Elaboración propia.

2.3.9.3. Arquitectura Middleware "consciente del modelo"

Las arquitecturas de middleware permiten la comunicación distribuida y proporcionan funcionalidad adicional, como mejorar el control, la supervisión y el registro (Szvetits & Zdun, 2016). Una arquitectura de middleware compatible con el modelo, consiste en una entidad central responsable de procesar la información del modelo y proporcionar servicios de acceso al modelo dentro de un sistema distribuido. Dicha arquitectura ayuda a mantener los mecanismos de monitoreo y adaptación en un lugar central, lo que aumenta la reutilización al tiempo que compensa las limitaciones de los límites de máquinas locales existentes en arquitecturas de flujo de datos monolíticas y locales.

La Figura 2.8 muestra un ejemplo de middleware compatible con el modelo que monitorea y controla un sistema mientras recolecta información de otras fuentes de

datos (por ejemplo, sensores o datos de entrada manual que impulsen acciones de reconfiguración).

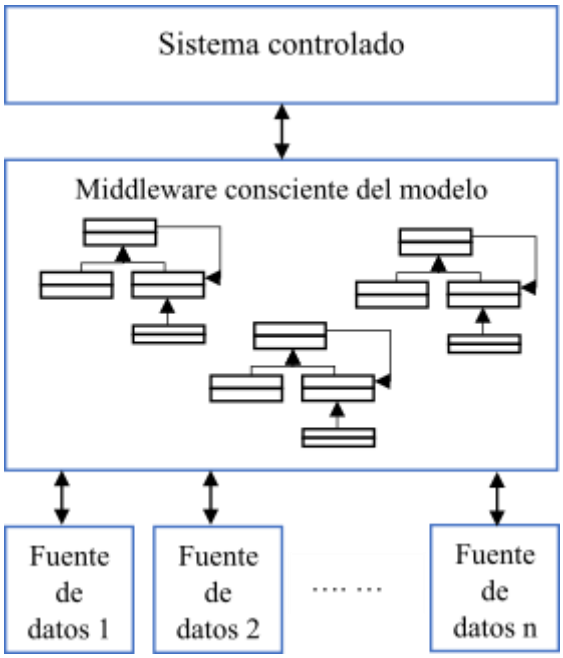


Figura 2.8. Middleware consciente del modelo para monitorear y controlar un sistema de acuerdo con los datos de entrada (Szvetits & Zdun, 2016). Fuente: Elaboración propia.

Existe una variedad de enfoques de middleware entre los cuales se destaca: middleware consciente del contexto (*CA3M - Context-aware middleware with contextual metamodel*) utiliza modelos de conciencia de contexto presentes en tiempo de ejecución que se actualizan para cumplir con los nuevos requisitos de la aplicación. Y el middleware MADAM que maneja un marco de planificación que automatiza la toma de decisiones de adaptación en sistemas adaptativos basados en componentes reflexivos.

2.3.9.4. Arquitectura Middleware de comunicación

El objetivo de un middleware de comunicación es abstraer la complejidad de la comunicación de red entre componentes distribuidos. La mayoría de los enfoques basados en middleware utilizan servicios como método de comunicación principal entre sus componentes.

Un desafío clave de la comunicación distribuida es la interoperabilidad entre los diferentes participantes de la red. En este contexto, para manejar eficientemente la interoperabilidad, Grace et al. (2005) proponen un middleware reflexivo llamado ReMMoC que permite que los clientes móviles se desarrollen independientemente de los mecanismos de descubrimiento e interacción de los servicios web. ReMMoC utiliza un marco vinculante para la interoperación con diferentes tipos de interacción y un marco de descubrimiento de servicios que hacen uso de una gama de protocolos para averiguar servicios.

2.3.9.5. Arquitectura de repositorio

Las arquitecturas basadas en repositorios emplean un almacenamiento central de modelos o datos relacionados con modelos. La ventaja de este tipo de arquitectura radica en el acceso controlado de las instancias del modelo, así como en el almacenamiento centralizado de la información de evolución del modelo.

Los repositorios se usan para establecer una arquitectura distribuida que facilite el acceso y la gestión de los datos del modelo al proveer interfaces estandarizadas para la manipulación del modelo. Además, cada repositorio puede almacenar múltiples versiones de modelos de manera que las versiones antiguas se utilicen hasta que sus instancias se eliminen o migren a otras nuevas. Por lo tanto, la ventaja del repositorio radica en conectar en tiempo de ejecución las nuevas versiones del modelo para que las instancias recién creadas puedan usarlas instantáneamente (Szvetits & Zdun, 2016).

La Figura 2.9 presenta una arquitectura de repositorio que permite el acceso simultáneo al modelo y el control de versiones. Las actualizaciones del repositorio pueden aplicarse mediante acciones manuales u otros sistemas que tienen acceso de escritura al repositorio.

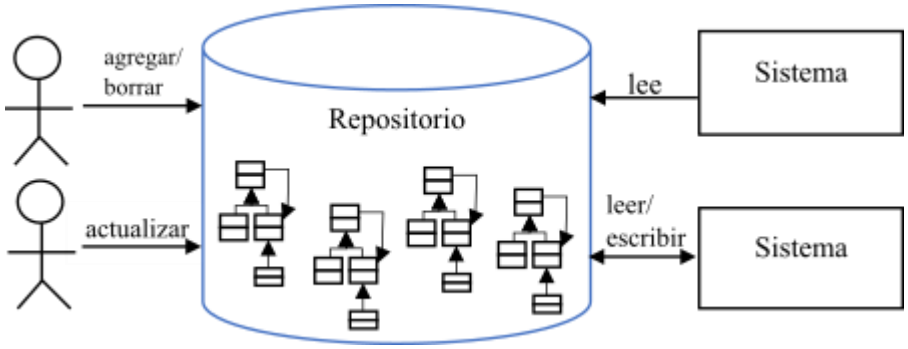


Figura 2.9. Arquitectura de repositorio que permite el acceso simultáneo al modelo y el control de versiones (Szvetits & Zdun, 2016). Fuente: Elaboración propia.

2.4. Computación en la nube

El Instituto Nacional de Estándares y Tecnologías (NIST) y Mell & Grance (2011) definen a la computación en la nube como,

> *"un modelo para permitir el acceso de red ubicuo, conveniente y bajo demanda a un grupo compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que se pueden aprovisionar y liberar rápidamente con un mínimo esfuerzo de gestión o interacción del proveedor de servicios".*

Mahmood (2011) brinda una definición alternativa donde describe a la computación en la nube como un estilo de computación donde las capacidades habilitadas por TI masivamente escalables se entregan "como un servicio" a clientes externos que usan

tecnologías de Internet, donde el objetivo más amplio de computación en la nube es hacer que la supercomputación esté disponible para los usuarios.

Aunque la idea principal detrás de la computación en la nube no era nueva, el término comenzó a ganar popularidad después de que el CEO de Google, Eric Shmidt, la usó en 2006 y en los últimos años la aparición de la computación en la nube ha impactado enormemente en la industria de TI.

La disponibilidad de almacenamiento virtualmente ilimitado y las capacidades de procesamiento a bajo costo permitieron la realización de un nuevo modelo de computación, en el cual los recursos virtualizados se pueden arrendar bajo demanda, proporcionándole como servicios generales.

Las grandes empresas como Amazon, Google, Facebook, entre otras, adoptaron ampliamente este paradigma, con el fin de ofrecer servicios a través de Internet, a fin de obtener beneficios tanto económicos como técnicos (Botta et al., 2016).

2.4.1. Arquitectura

La Figura 2.10 muestra la arquitectura de un ambiente de computación en la nube dividida en cuatro capas: (i) la capa de hardware, (ii) la capa de infraestructura, (iii) la capa de plataforma y (iv) la capa de aplicación. A estas capas se las puede denominar "*modelos de servicio*". Cada capa se puede ver como un servicio para la capa anterior y como un consumidor para la capa siguiente (Zhang et al., 2010).

La capa de hardware es la capa responsable de gestionar los recursos físicos de la nube, incluidos servidores físicos, enrutadores, conmutadores, sistemas de alimentación y refrigeración. La capa de infraestructura es un componente esencial de la computación en la nube, debido a que permite la asignación dinámica de recursos, solo disponibles a través de las tecnologías de virtualización. La capa de plataforma, está construida sobre la capa de infraestructura, y consiste en sistemas operativos y marcos de aplicaciones.

Finalmente, la capa de aplicación, consiste en las aplicaciones de nube reales. A diferencia de las aplicaciones tradicionales, las aplicaciones en la nube pueden aprovechar la función de escalado automático para lograr un mejor rendimiento, disponibilidad y un menor costo operativo.

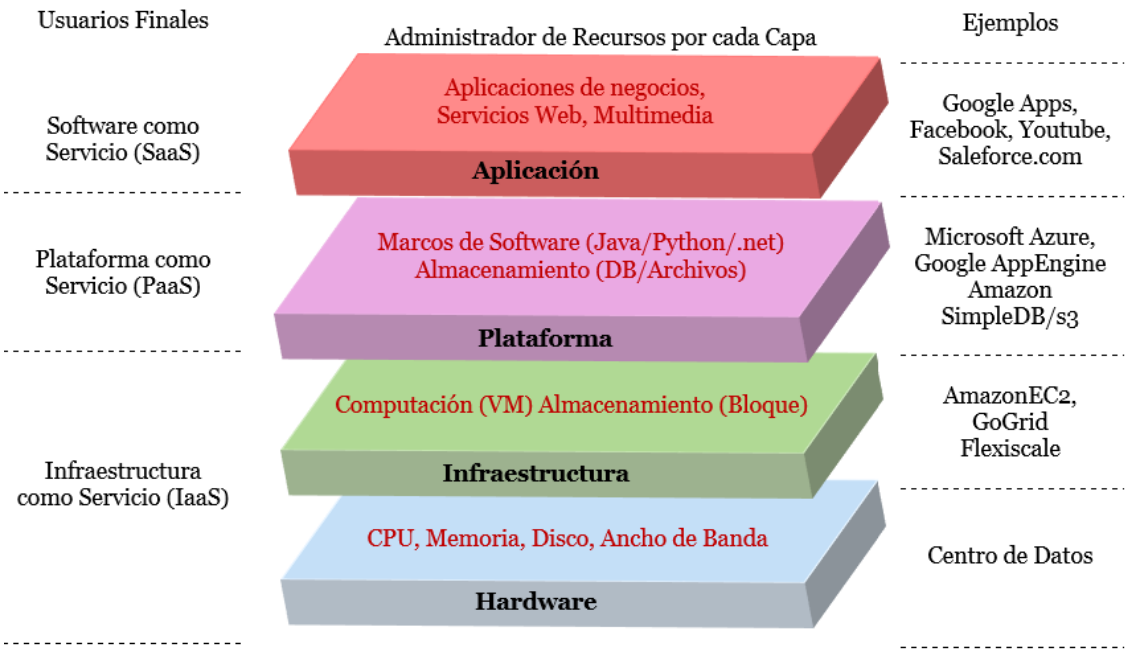


Figura 2.10. Arquitectura de Computación en la Nube (Jing & Jian-Jun, 2010). *Fuente: Elaboración propia.*

2.4.2. Características

Según la NIST Mell & Grance (2011) se describen:

Autoservicio bajo demanda (*On-demand self-service*): Un consumidor puede abastecer capacidades informáticas de forma automática con cada proveedor de servicios, sin necesidad de la interacción humana.

Amplio acceso a la red (*Broad network access*): Las capacidades están disponibles a través de la red y se accede a ellas a través de mecanismos estándar que promueven el uso de plataformas heterogéneas de clientes delgados o gruesos (por ejemplo, teléfonos móviles, tabletas, computadoras portátiles y estaciones de trabajo).

Agrupación de recursos (*Resource pooling*): Los recursos informáticos del proveedor se agrupan para servir a múltiples consumidores, con diferentes recursos físicos y virtuales asignados y reasignados dinámicamente de acuerdo con la demanda del consumidor (por ejemplo, almacenamiento, procesamiento, memoria y ancho de banda de red).

Rápida elasticidad (*Rapid elasticity*). Las capacidades se pueden aprovisionar y liberar elásticamente, en algunos casos automáticamente, para escalar rápidamente hacia afuera y hacia adentro de acuerdo con la demanda.

Servicio medido (*Measured service*): Los sistemas se controlan y optimizan automáticamente al uso de los recursos al aprovechar una capacidad de medición apropiada para el tipo de servicio (por ejemplo, almacenamiento, procesamiento, ancho de banda y cuentas de usuario activas).

2.4.3. Modelo de servicio

Los modelos de servicio de computación en la nube son: software como servicio (*SaaS - Software as a Service*), plataforma como servicio (*PaaS - Platform as a Service*) e infraestructura como servicio (*IaaS - Infrastructure as a Service*).

Sin embargo, en los últimos años han sugerido otras categorías como servicios, por ejemplo: almacenamiento como servicio, base de datos como servicio, seguridad como servicio, comunicación como servicio, gestión/gobierno como servicio, integración como servicio, pruebas como servicio y proceso de negocio como servicio (Mahmood, 2011).

A continuación, se describen los modelos de servicio más utilizados actualmente, definidos por la NIST de la siguiente manera:

Software como servicio (SaaS): proporcionar al consumidor la capacidad de usar las aplicaciones del proveedor que se ejecutan en una infraestructura en la nube. Se puede acceder a las aplicaciones desde varios dispositivos cliente. El consumidor no gestiona ni controla la infraestructura de la nube subyacente, incluida la red, los servidores, los sistemas operativos, el almacenamiento o incluso las capacidades de aplicaciones individuales, con la posible excepción de los ajustes de configuración de la aplicación específicos del usuario (Mell & Grance, 2011).

Entre los servicios ofertados existen Gmail, Google Doc, Finance, Collaboration, Communication, Business, CRM, ERP, HR; ejemplo de plataformas que ofrecen los servicios descritos son: Zoho, Salesforce, Google apps (Youssef, 2012).

Plataforma como servicio (PaaS)*:* proporciona al consumidor la capacidad de implementar en la infraestructura de la nube aplicaciones creadas o adquiridas por el consumidor a través de lenguajes de programación, bibliotecas, servicios y herramientas compatibles con el proveedor. El consumidor no administra ni controla la infraestructura de nube subyacente, incluida la red, servidores, sistemas operativos o almacenamiento, pero tiene control sobre las aplicaciones implementadas y posiblemente las configuraciones para el entorno de alojamiento de aplicaciones (Mell & Grance, 2011).

Los servicios ofertados son Web 2 application runtime, Java 2 runtime, Developer tools, Middleware; ejemplo de plataformas que ofrecen los servicios descritos son: Windows Azure, Aptana, Google apps engine, etc. (Youssef, 2012).

Infraestructura como servicio (IaaS)*:* proporciona al consumidor la capacidad de aprovisionar procesamiento, almacenamiento, redes y otros recursos informáticos fundamentales donde se puede implementar y ejecutar software arbitrario, que puede incluir sistemas operativos y aplicaciones. El consumidor no administra ni controla la infraestructura de nube subyacente, pero tiene control sobre los sistemas operativos, el almacenamiento y las aplicaciones implementadas; y posiblemente un control limitado de componentes de red seleccionados (por ejemplo, firewalls de host) (Mell & Grance, 2011).

Los servicios ofertados son Servers, Storage, Processing power, Networking, Bandwidth; ejemplo de plataformas que ofrecen los servicios descritos son: Amazon web services, Dropbox, Akamai, etc. (Youssef, 2012).

2.4.4. Modelo de despliegue

Los modelos de servicio dentro de la computación en la nube pueden ser: nube pública, nube privada, nube comunitaria y nube híbrida. La Figura 2.11 a continuación muestra diferentes modelos de despliegue en la nube.

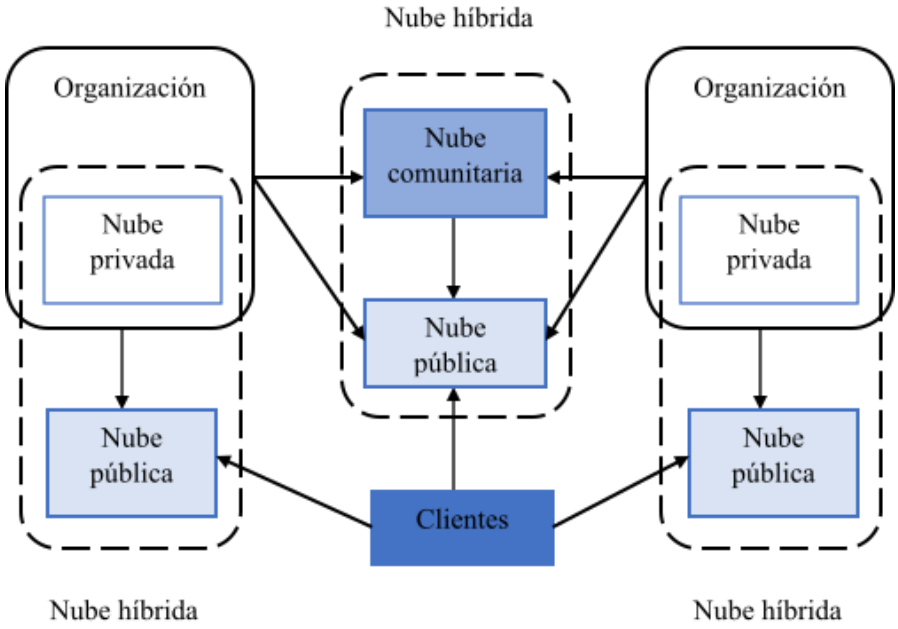


Figura 2.11. Modelos de implementación de computación en la nube. Fuente: Elaboración propia.

Nube pública: el público en general puede hacer uso abierto de los recursos provisionados en la infraestructura de la nube.

Nube privada*:* una organización con múltiples consumidores (por ejemplo, unidades de negocios), puede hacer uso exclusivo de los recursos provisionados en la infraestructura de la nube.

Nube comunitaria*:* una comunidad específica de consumidores de organizaciones que tienen preocupaciones compartidas (por ejemplo, misión, requisitos de seguridad, políticas y consideraciones de cumplimiento), puede hacer uso exclusivo de los recursos provisionados en la infraestructura de la nube.

Nube Híbrida: La infraestructura de la nube es una composición de dos o más infraestructuras de nube distintas (privadas, comunitarias o públicas) que son entidades únicas, pero están unidas por tecnología estandarizada o patentada que permite la portabilidad de datos y aplicaciones (por ejemplo, la explosión de la nube para el equilibrio de carga entre nubes).

2.4.5. Aplicaciones

Algunas de las aplicaciones donde la computación en la nube desempeña un papel importante son:

2.4.5.1. E-learning

El e-learning es una nueva tendencia en educación que trata de hacer el mejor uso de la tecnología de la información (TI). La computación en la nube es un entorno atractivo para estudiantes, profesores e investigadores porque puede proporcionar a las universidades y centros de investigación una infraestructura computacional potente y rentable.

Los estudiantes pueden conectarse a los servicios educativos del campus a través de sus dispositivos móviles personales desde cualquier lugar. Los miembros de la facultad pueden tener acceso eficiente y flexible a su material del curso en sus aulas. Los investigadores pueden encontrar artículos, modelos y ejecutar sus experimentos en la nube más rápido que nunca (Youssef, 2012).

2.4.5.2. CloudIoT

Se denomina CloudIoT a la fusión de las tecnologías en la nube e internet de las cosas, a pesar de ser diferentes entre sí, a menudo sus características son complementarias. CloudIoT dio lugar a un nuevo conjunto de servicios y aplicaciones inteligentes, que pueden tener un fuerte impacto en la vida cotidiana, a continuación, se describe un conjunto de aplicaciones que se hacen posibles o se mejoran significativamente gracias al paradigma CloudIoT (Botta et al., 2016).

Cuidado de la salud: La adopción del paradigma CloudIoT en el campo de la salud puede brindar varias oportunidades para la TI médica simplificando los procesos de atención médica, además los expertos creen que puede mejorar significativamente los servicios de salud y contribuir a su innovación continua y sistemática.

Un ambiente de vida asistida (AAL), en particular, tiene como objetivo facilitar la vida cotidiana de las personas con discapacidad y afecciones médicas crónicas; mediante la aplicación de CloudIoT en este campo, es posible proporcionar muchos servicios innovadores, como: recopilar datos vitales de los pacientes a través de una red de sensores conectados a dispositivos médicos, entregar los datos a la nube de un centro médico para su almacenamiento y procesamiento, mediante el manejo adecuado de la información proporcionada por sensores o que garantiza el acceso ubicuo o el intercambio de datos médicos como Registros electrónicos de atención médica (EHR).

Las aplicaciones de atención médica generalizadas generan una gran cantidad de datos de sensores que deben gestionarse adecuadamente para su posterior análisis y procesamiento. La adopción de una nube representa una solución prometedora para la gestión eficiente de los datos del sensor de salud y permite abstraer detalles técnicos, eliminando la necesidad de experiencia o control sobre la infraestructura

tecnológica. Además, conduce a una fácil automatización del proceso de recopilación y entrega de datos a un costo reducido (Botta et al., 2016).

Hogar inteligente. Las redes domésticas se han identificado como el entorno donde los usuarios actúan principalmente: CloudIoT tiene una gran aplicación en entornos domésticos, donde la adopción conjunta de dispositivos integrados heterogéneos y la nube permite la automatización de actividades internas comunes.

Varias aplicaciones de hogares inteligentes propuestas en la literatura involucran redes de sensores (inalámbricas) y realizan la conexión de dispositivos inteligentes a Internet para monitorear de manera remota su comportamiento (p ejm., para monitorear el uso de energía de los dispositivos para mejorar los hábitos de uso de energía) o controlarlos de forma remota para gestionar la iluminación, la calefacción y el aire acondicionado (Botta et al., 2016).

Video vigilancia. CloudIoT en el contexto de la videovigilancia inteligente conduce a almacenar, administrar y procesar de manera fácil y eficiente los contenidos de video originados por sensores de video (cámaras IP) y a extraer automáticamente el conocimiento de las escenas.

Se ha convertido en una herramienta de gran importancia para varias aplicaciones relacionadas con la seguridad donde las soluciones propuestas pueden entregar transmisiones de video a múltiples dispositivos de usuario a través de Internet, mediante la distribución de las tareas de procesamiento ejecutadas mediante el uso de los recursos del servidor físico de manera equilibrada en la carga y tolerante a fallas (Botta et al., 2016).

Transporte y movilidad inteligente. Como tecnología emergente, se espera que IoT ofrezca soluciones prometedoras para transformar los sistemas de transporte y los servicios de automóviles (es decir, Sistemas de Transporte Inteligente, ITS).

La integración de la nube con tecnologías IoT representa una oportunidad prometedora que permite desarrollar e implementar una nueva generación de minería de datos vehicular para brindar muchos beneficios comerciales, como aumentar la seguridad vial, reducir la congestión vial, administrar el tráfico y el estacionamiento, realizar análisis de garantía y recomendar el mantenimiento o reparación de automóviles (Botta et al., 2016).

Energía inteligente y red inteligente. IoT y la nube pueden fusionarse efectivamente para proporcionar una gestión inteligente de la distribución y el consumo de energía en entornos heterogéneos locales y de área amplia.

Los nodos IoT que generalmente participan en este tipo de procesos tienen capacidades de detección, procesamiento y conexión en red, pero recursos limitados. Por lo tanto, las tareas informáticas se pueden exigir adecuadamente a la nube, donde se pueden tomar decisiones más complejas y completas.

La computación en la nube hace posible analizar y procesar grandes cantidades de datos e información provenientes de diferentes fuentes distribuidas a lo largo de redes

de área amplia, con el fin de implementar un control inteligente para los objetos (Botta et al., 2016).

Monitoreo ambiental. El uso combinado de la nube e IoT puede contribuir al despliegue de un sistema de información de alta velocidad entre la entidad encargada de monitorear entornos de área amplia y los sensores/actuadores implementados adecuadamente en el área.

Algunas aplicaciones pueden estar relacionadas con el monitoreo continuo y a largo plazo del nivel del agua (para lagos, arroyos, aguas residuales), la concentración de gas en el aire (en laboratorios, depósitos), la humedad del suelo y otras características, la inclinación de estructuras estáticas (puentes, presas), cambios de posición (deslizamientos de tierra), condiciones de iluminación (para detectar intrusiones en lugares oscuros), radiación infrarroja para incendios o detección de animales.

Otras aplicaciones potenciales de este tipo son: transmisión de información agrícola y detección inteligente, control de cultivo inteligente, seguimiento de seguridad alimentaria, riego de precisión, identificación de bosques y seguimiento de árboles (Botta et al., 2016).

2.5. Fog Computing

Fog Computing o Computación en niebla es un término propuesto por Cisco que describe un modelo en el que los datos pueden ser analizados y procesados por aplicaciones que se ejecutan en dispositivos dentro de la red en lugar de una nube centralizada. Se encuentra la mayoría de veces, pero no siempre en el borde de la red de forma que organiza y administra inteligentemente los recursos de cómputo y almacenamiento, por esta razón también se define como un paradigma de computación distribuida que extiende los servicios proporcionados por la nube al borde de la red (Bonomi et al., 2012; Yannuzzi et al., 2014).

Fog Computing no fue desarrollado para competir con Cloud, sino para trabajar en conjunto en diversos casos de usos y aplicaciones donde Cloud Computing resulta muy costoso. Cloud y Fog Computing son una combinación ideal para comunicaciones, orquestaciones y asignación de recursos de cómputo, y almacenamiento que solventan los requisitos de IoT (Bonomi et al., 2012; Gupta et al., 2017; Yannuzzi et al., 2014).

2.5.1. Características

Fog Computing posee nueve características, entre las más destacadas. La primera, la reducción de latencia de propagación, especialmente para aplicaciones en contextos críticos que requieren procesamiento de datos en tiempo real. Algunos de los ejemplos son la robótica en la nube, el control de aeronaves de vuelo por cable o los frenos antibloqueo en un vehículo.

La segunda, la capacidad de filtrar y procesar una gran cantidad de datos entrantes en dispositivos de borde, refiérase a dispositivos de borde aquellos dispositivos que

proporcionan un punto de entrada a la red (p. ejem., routers, conmutadores de enrutamiento, etc.), dando lugar a que la arquitectura de procesamiento de datos sea distribuida y, por ende, escalable (Gupta et al., 2017).

La tercera está constituida de redes de sensores a gran escala para monitorear el entorno. La cuarta es el apoyo de la distribución geográfica, es decir, los servicios y aplicaciones a los que se dirige la Niebla demandan implementaciones ampliamente distribuidas (More, 2015; Mukherjee et al., 2017).

La quinta, la facilidad de administración y programación de los servicios de computación, redes y almacenamiento entre los centros de datos y los dispositivos finales. Finalmente, el sexto es la capacidad de procesamiento de alto número de nodos (Dastjerdi & Buyya, 2016; Mukherjee et al., 2017).

La octava, la escalabilidad de procesar los datos entrantes más cercanos de la fuente, por lo que reduce la carga de ese procesamiento en la nube, a fin de solventar los problemas de escalabilidad.

Finalmente, el manejo de los datos de IoT de forma local utilizando clientes o dispositivos de borde que permita llevar a cabo una cantidad sustancial de almacenamiento, comunicación, control, configuración y administración (Dastjerdi & Buyya, 2016).

2.5.2. Arquitectura

La Arquitectura global de Fog Computing se constituye por seis capas como son la capa física y de virtualización, la capa de monitoreo, la capa de preprocesamiento, la capa de almacenamiento temporal, la capa de seguridad y la capa de transporte, las cuales se describen en la Figura 2.12.

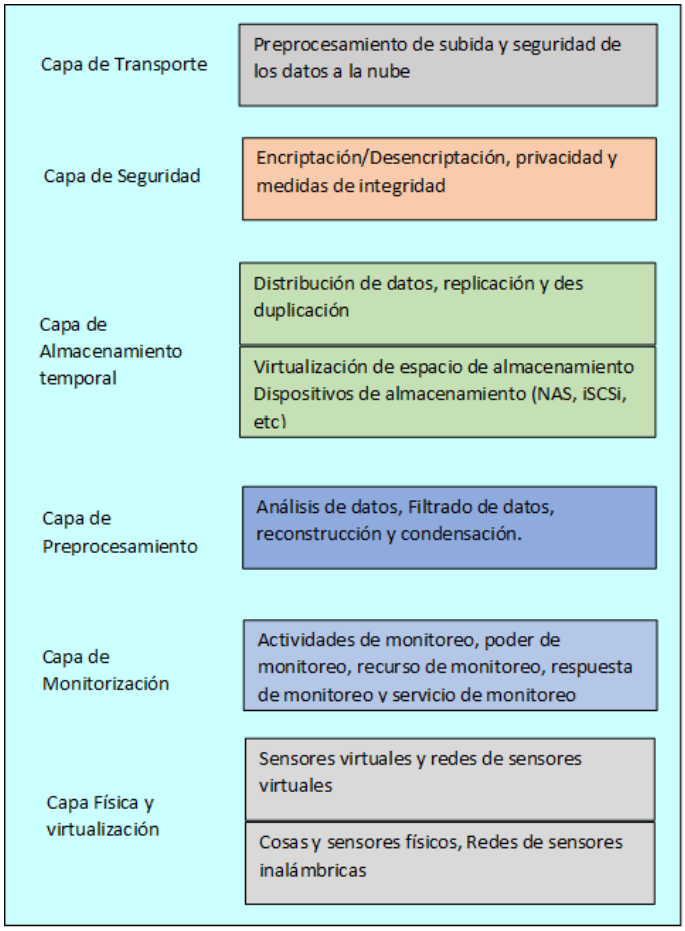


Figura 2.12. Arquitectura Fog Computing (Aazam & Huh, 2016). Fuente: Elaboración propia.

La Capa física y de virtualización, está compuesta por nodos físicos, WSN (*Wireless Sensor Networks, sensores de redes inalámbricas*), nodos virtuales y redes de sensores virtuales (VSN) se administran y mantienen de acuerdo con los requisitos del usuario (Aazam & Huh, 2016; Dastjerdi et al., 2016).

La Capa de monitoreo, se encarga de supervisar las actividades de los nodos y redes subyacentes, observando qué nodo realiza qué tarea, la hora en que realiza y qué se requiere de él.

La capa de pre-procesamiento, es responsable de realizar tareas relacionadas con la gestión de datos, es decir, analiza los datos recopilados, realiza el filtrado y recorte de datos y, genera datos más significativos y necesarios, que son almacenados en Fog temporalmente y cuando se cargan a la nube son eliminados.

La capa de Seguridad, en algunos contextos como la atención médica ubicua y los servicios de salud inteligentes, los datos de los pacientes que se generan son privados por lo cual estos datos deben ubicarse en lugares de almacenamiento seguro, esto se logra con la capa de seguridad.

Finalmente, la capa de transporte prepara los datos para enviar y cargar en la nube, a fin de que la nube cree servicios más útiles.

2.5.3. Aplicaciones

Se presentan cuatro aplicaciones en las cuales ha sido utilizada la capa de Fog Computing. La primera aplicación, los vehículos conectados, que muestran una variedad de conexiones e interacciones ya sea automóviles a automóviles, automóviles a puntos de acceso (Wi-Fi, 3G, LTE, luces inteligentes de tránsito) y puntos de acceso a puntos de acceso.

Fog tiene una serie de atributos que la convierten en la plataforma ideal para soporte de tráfico y análisis, ya sea geo-distribución (en ciudades y carreteras), movilidad o reconocimiento de ubicación en tiempo real del vehículo conectado (Bonomi et al., 2012).

La segunda aplicación, es el cuidado de la salud, Dastjerdi et al. (2016) describen un sistema de análisis distribuido asistido para monitorear la caída de pacientes, quienes han sufrido un accidente cerebrovascular, mediante el desarrollo de varios algoritmos basados en mediciones de aceleración y métodos de análisis de series de tiempo, para ello diseñaron un sistema de detección de caídas en tiempo real que divide la tarea entre los dispositivos de borde y la nube.

La tercera aplicación, relacionada con la realidad aumentada, está formada por aplicaciones donde la latencia es un factor importante a considerar, porque incluso pequeños retrasos en la respuesta de una aplicación pueden dañar la experiencia del usuario, es por ello que la computación en niebla tiene el potencial de convertirse en una pieza fundamental en estas aplicaciones.

Según Dastjerdi et al. (2016), una implementación de este contexto se construyó mediante un juego de interacción de computadora cerebral aumentada y los datos que se transmitirán.

Finalmente, existen las redes inalámbricas de sensores y actuadores, donde los nodos de sensores inalámbricos funcionan a una potencia extremadamente baja para alargar el tiempo de vida útil de la batería o para la recolección de energía.

La mayoría de sensores cuentan con un gran número de ancho de banda bajo, poca energía, baja potencia de procesamiento, poca memoria, que funcionan como un recopilador unidireccional. Este tipo de sensores son útiles en una variedad de escenarios para recopilar datos ambientales (p. ejem., humedad, temperatura, cantidad de lluvia, intensidad de luz, etc.) (Bonomi et al., 2012).

Capítulo 3

3. Estado del Arte

En el presente capítulo se describe el estudio de literatura realizado en torno al dominio en el que se enfoca el presente trabajo de titulación. Este estudio tiene como objetivos la búsqueda, identificación y análisis de los diferentes estudios primarios publicados en revistas digitales.

El capítulo se desglosa de la siguiente manera: la sección 3.1 presenta una introducción a la metodología empleada para la revisión sistemática. La sección 3.2 expone la etapa de planificación en donde se identifican las actividades para realizar la revisión sistemática. La sección 3.3 explica la etapa de ejecución de las actividades que se plantearon en el punto 3.2. Finalmente, en la sección 3.4 se describe la etapa de reporte de los resultados obtenidos en la revisión sistemática.

3.1. Metodología

En la realización del estudio de la literatura se han seguido los pasos de la metodología planteada por Kitchenham & Charters (2007), ésta consiste en plantear las preguntas de investigación con respecto al tema relacionado con el trabajo de titulación planteado, siguiendo una serie de actividades que permita conocer el estado actual de la investigación en el tema.

Las actividades están agrupadas en tres fases: i) la de planificación; ii) la de ejecución y iii) la de reporte de resultados. Al finalizar esta metodología, se obtiene como resultado una investigación basada en estudios primarios, garantizando que se puede contar con evidencia que provenga de fuentes confiables.

3.2. Etapa de planificación

La primera fase es denominada etapa de planificación, la misma que está compuesta por seis actividades que deben ser cumplidas dentro de una revisión sistemática de literatura. En este apartado se establecen las preguntas y sub-preguntas de investigación, se define la estrategia de búsqueda de estudios primarios, cadena de búsqueda, criterios de extracción de datos y finalmente se realiza una evaluación de la calidad de los estudios primarios.

3.2.1. Preguntas de investigación

El objetivo de investigación de la revisión sistemática nos lleva a plantear la siguiente pregunta de investigación: *"¿Para qué y cómo se utilizan los modelos en tiempo de ejecución para ambientes de internet de las cosas?".* respondida en el artículo

"Models@runtime and Internet of Things: A Systematic Literature Review" de Erazo-Garzón et al. (2020), publicado en el ICI2ST 2021: *II International Conference on Information Systems and Software Technologies* dentro del contexto de este trabajo de titulación.

Además, con el fin de dar respuesta a la pregunta de investigación general, se definen las siguientes sub-preguntas:

RQ1: ¿En qué sub-dominios se emplean los modelos en tiempo de ejecución dentro del dominio de internet de las cosas?

RQ2: ¿Para qué propósitos se han implementado los modelos en tiempo de ejecución en el dominio de internet de las cosas?

RQ3: ¿Cómo se aborda la investigación en los estudios relacionados con la utilización de modelos en tiempo de ejecución en el dominio de internet de las cosas?

3.2.2. Estrategia de búsqueda

Para responder a las preguntas de investigación se plantea una estrategia de búsqueda que nos permita la obtención de información de fuentes primarias, las librerías digitales son: ACM Digital Library, IEEE Xplorer Digital Library, Springer Link y Science Direct. Las librerías indicadas, se tomaron en cuenta porque son de gran relevancia en áreas tecnológicas donde muchos artículos son publicados. De estas librerías se recopilan artículos científicos de las conferencias más importantes y que se encuentren indexados en revistas de excelencia, de tal forma que la información se clasifique de las fuentes primarias más apropiadas.

3.2.3. Cadena de búsqueda

Para la búsqueda automática en las bibliotecas digitales seleccionadas, se ha determinado un grupo de palabras clave, que están relacionadas entre sí mediante conectores lógicos, logrando formar la siguiente cadena de búsqueda.

CADENA	SUBCADENA	CONECTOR
model	*model*	AND
runtime	Runtime	OR
run.time	run.time	AND
Internet of Things	Internet of Things	OR
IoT	IoT	OR

Ambient Intelligence	Ambient Intelligence	OR
AmI	AmI	OR
ubiquitous computing	ubiquitous computing	OR
cyber-physical system	cyber-physical system*	OR
cyber system	cyber system*	OR
industry 4.0	industry 4.0	

Tabla 3.1. Palabras de la cadena de búsqueda.

La Tabla 3.1, muestra las palabras usadas para generar la cadena de búsqueda, (“*model*”) AND (“runtime” OR “run.time”) AND (“Internet of Things” OR “IoT” OR “Ambient Intelligence” OR “AmI” OR “ubiquitous computing” OR “cyber-physical system*” OR “cyber system*” OR “industry 4.0”).

3.2.4. Período de búsqueda

Luego de generar la cadena de búsqueda se determina el periodo de búsqueda, en donde se limitó a considerar información de los estudios primarios a partir del 2006, año en que el término modelos en tiempo de ejecución fue acuñado por primera vez (Bencomo et al., 2019), la fecha de corte de la revisión sistemática es diciembre 31 de 2019. El término IoT fue adoptado posteriormente en el año 2009.

3.2.5. Criterios de extracción de datos

Cada estudio primario obtenido de la búsqueda automática o manual será evaluado por un conjunto de criterios de extracción de datos donde por cada pregunta de investigación se describe un o más criterios y las posibles respuestas esperadas (ver Tabla 3.2).

COD.	CRITERIO	POSIBLES RESPUESTAS
RQ1: ¿En qué sub-dominios se emplean los modelos en tiempo de ejecución dentro del dominio de internet de las cosas?		

EC1	Dominio de IoT	Transporte, Educación, Gobierno, Salud, Finanzas y banca, Seguridad pública y emergencia, Monitoreo y control ambiental (agua, energía, alcantarillado, etc.), Agricultura, Logística y venta al detalle (Control de la cadena de suministro), Control industrial, Entretenimiento y deporte, Turismo, Ambientes inteligentes (Domótica), Control inteligente de animales, Otros.
RQ2: ¿Para qué propósitos se han implementado los modelos en tiempo de ejecución en el dominio de internet de las cosas?		
EC2	Propósitos de los modelos en tiempo de ejecución	Monitoreo, Aseguramiento de calidad (requerimientos no funcionales), Interoperabilidad, Auto-adaptación (self-adaptation), Auto-optimización (self-optimization), Auto-organización (self-organization), Simulación / Predicción, Auto-recuperación/Tolerancia a fallos/consistencia, Otro.
EC3	Técnicas utilizadas en combinación con los modelos en tiempo de ejecución	Transformación de modelos, Bucle de control automático (MAPE-K), Conformidad de modelos, Aprendizaje automático / razonamiento, Modelado de variabilidad, Ingeniería de requerimientos, Flujos de trabajo, Comparación de modelos, Líneas de producto software, Otro.
RQ3: ¿Cómo se aborda la investigación en los estudios relacionados a la implementación de infraestructuras que usan modelos en tiempo de ejecución en el dominio de internet de las cosas?		
EC4	Tipo de estudio	Nuevo, Extensión
EC5	Tipo de validación	Encuesta, Experimento, Prototipo, Prueba de concepto, Caso de estudio, Ninguno

EC6	Alcance del enfoque	Industria, Academia

Tabla 3.2. Criterios de extracción de datos.

3.3. Etapa de ejecución

La segunda fase planteada por Kitchenham & Charters (2007) consiste en el proceso de selección y revisión de los estudios primarios publicados en las revistas digitales y conferencias identificadas en la etapa previa; de esta manera se puede proceder con la extracción de información que responda a las preguntas de investigación planteadas.

Las actividades en esta fase consisten en la selección de los estudios primarios y la evaluación o aseguramiento de la calidad de cada estudio para determinar si deben ser incluidos o excluidos del mapeo sistemático.

3.3.1. Selección de estudios primarios

Una vez realizadas las búsquedas tanto automática como manual se procede a efectuar la evaluación de los metadatos (título, resumen, palabra clave) de cada estudio primario encontrado. Para esta evaluación se han considerado los estudios que cumplan con al menos uno de los siguientes criterios de inclusión.

A. **Criterios de Inclusión**

- ➢ Artículos publicados en cualquiera de las fuentes de información especificada para las búsquedas manuales y automáticas.
- ➢ Estudios donde se refleje el uso de modelos en tiempo de ejecución (modelos causalmente conectados con el sistema) dentro del dominio de IoT.
- ➢ Artículos en cuya investigación se presenta información que permita dar respuesta a cualquiera de las sub-preguntas de investigación.

En el discernimiento de la información se consideraron estudios que cumplan con al menos uno de los siguientes criterios de exclusión.

B. **Criterios de Exclusión**

- ➢ Editoriales, prólogos, opiniones, discusiones, entrevistas, noticias o posters.
- ➢ Estudios duplicados en diferentes fuentes.
- ➢ Artículos cortos con menos de cuatro páginas.
- ➢ Artículos escritos en idiomas distintos al inglés.

La selección de estudios primarios aplicando los criterios de inclusión y exclusión se realizó en 3 etapas. La primera etapa comprende la recopilación de los estudios primarios de las revistas definidas, la segunda etapa se llevó a cabo mediante la

revisión de los títulos, resúmenes y palabras clave de los estudios primarios obtenidos en la primera etapa, y en la tercera etapa se llevó a cabo la revisión del texto completo de cada estudio primario y se incluyeron los más relevantes del tema, en total se obtuvieron 50 estudios, de los cuales, 40 fueron estudios de búsqueda automática y 10 de búsqueda manual.

La Tabla 3.3 presenta un resumen de la cantidad de estudios primarios obtenidos en las librerías digitales especificadas en el apartado de estrategia de búsqueda. Los estudios seleccionados para responder las preguntas de investigación se encuentran en el Apéndice A.

BIBLIOTECA DIGITAL	ESTUDIOS	PORCENTAJE
SpringerLink	27	55 %
ACM	6	12 %
IEEE XPLORE	13	25 %
ScienceDirect	3	6 %
Otras revistas	1	2 %
TOTAL	50	100 %

Tabla 3.3. Número de artículos incluidos.

3.3.2. Aseguramiento de la calidad

Con la finalidad de valorar la calidad de cada estudio se realiza una evaluación considerando el número de citas donde fue mencionado dicho artículo, para lo cual se emplea una escala de Likert de tres puntos, donde (+1) el puntaje para aquellos estudios que tiene más de tres citas, (0) si el estudio tiene entre una hasta tres citas, y (-1) si el estudio no tiene citas.

La Tabla 3.4 detalla los resultados de evaluar la calidad acorde a los puntajes definidos.

PUNTAJE (Descripción)	CANTIDAD	PORCENTAJE
+1 (Más de tres citas)	34	69 %
0 (De una a tres citas)	12	23 %
-1	4	8 %

(No tiene citas)		
TOTAL	50	100 %

Tabla 3.4. Evaluación de la calidad de los estudios primarios.

3.4. Etapa de reporte

Finalmente, en la última fase denominada etapa de reporte se presentan los resultados obtenidos, mediante la generación de tablas en las que se resumen los valores generados para cada criterio de extracción. Estas tablas facilitan realizar comparaciones entre los criterios de extracción planteados con el fin de descubrir el panorama actual del contexto estudiado para dar respuesta a las preguntas de investigación planteadas, y conocer el estado actual del uso de modelos en tiempo de ejecución en el dominio de IoT.

3.4.1. Porcentajes individuales

Culminada la etapa de recopilación y evaluación de la información, en este apartado se presenta los resultados obtenidos de los artículos en cada uno de los criterios de extracción planteados, para esto se ha desarrollado tablas y gráficos estadísticos que permitan la visualización del total de estudios y su porcentaje.

A continuación, se presentan los criterios de extracción más relevantes para tabular y los resultados totales por cada criterio se detallan en el Apéndice B.

El contexto donde se ha desarrollado e implementado sistemas IoT basados en modelos en tiempo de ejecución se presenta en la Tabla 3.5.

COD.	CRITERIO	RESPUESTAS	Nº ESTUDIOS	%
RQ1: ¿En qué sub-dominios se emplean los modelos en tiempo de ejecución dentro del dominio de internet de las cosas?				
EC1	Sub-dominio de IOT	Ambientes inteligentes (Domótica)	21	42 %
		Salud	8	16 %
		Seguridad pública y emergencia	7	14 %
		Monitoreo y control ambiental (agua, energía,	5	10 %

		alcantarillado, etc.)		
		Transporte	4	8 %
		Logística y venta al detalle (Control de la cadena de suministro)	4	8 %
		Control industrial	3	6 %
		Gobierno	2	4 %
		Otros	2	4 %
		Educación	1	2 %
		Finanzas y banca	1	2 %
		Agricultura	1	2 %
		Entretenimiento y deporte	1	2 %
		Control inteligente de animales	1	2 %

Tabla 3.5. Porcentajes individuales para el criterio de extracción EC1 de la sub-pregunta RQ1.

En la Figura 3.1 muestra el porcentaje de estudios primarios según el subdominio de aplicación, lo que demuestra que existe un gran interés en el subdominio de ambientes inteligentes (42%), debido a la necesidad cada vez mayor de interconectar aparatos de uso diario al internet para brindar mejores servicios a los usuarios en sus ambientes de vida natural (Jonckers et al., 2018).

Muy por detrás, se encuentran los subdominios: salud (16%), seguridad pública y emergencia (14%), monitoreo y control ambiental (10%), transporte (8%), logística (8%) y control industrial (6%). Finalmente, entre los subdominios que han utilizado muy poco para aplicar los modelos en tiempo de ejecución están: educación (2%), finanzas (2%), agricultura (2%), entretenimiento y deporte (2%) y control inteligente de animales (2%).

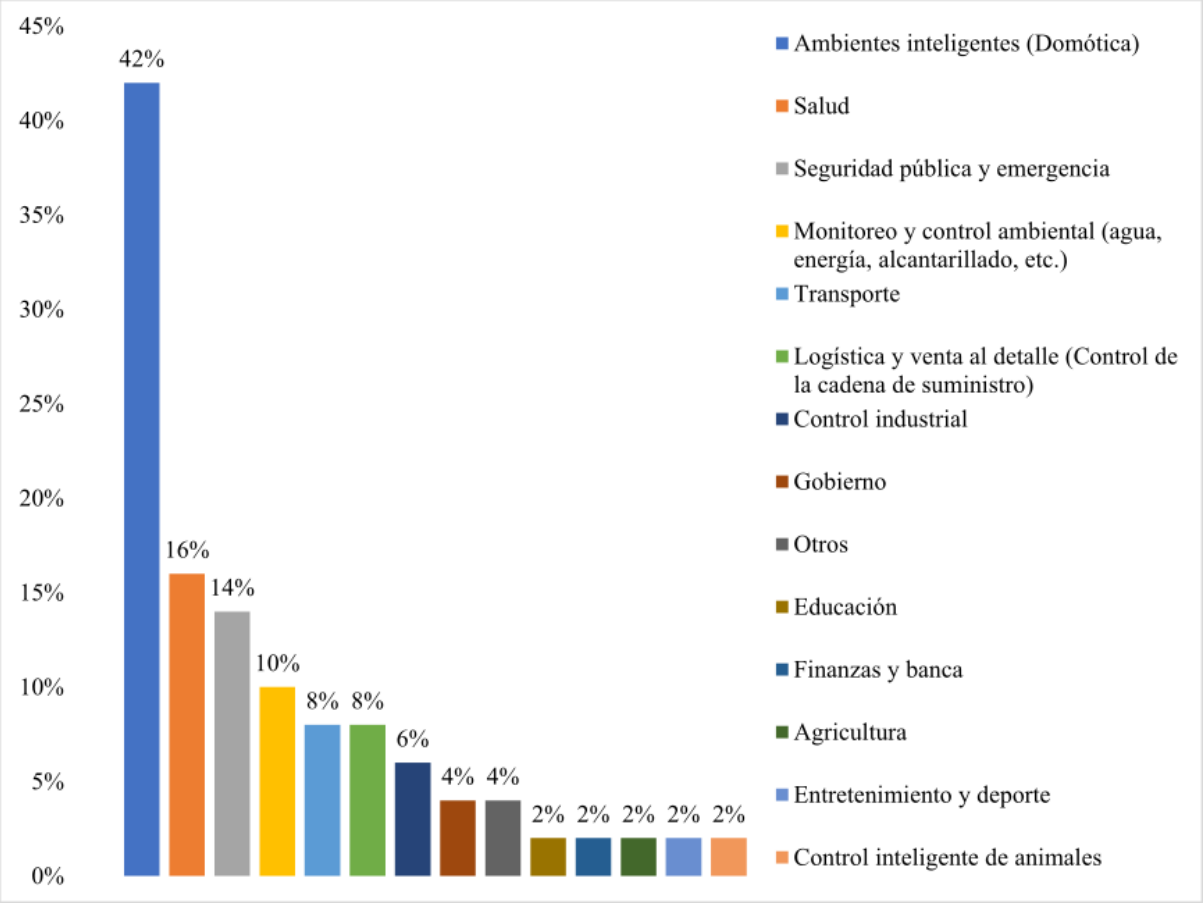


Figura 3.1. EC1 - Porcentaje de contextos donde se ha desarrollado e implementado sistemas IoT basados en modelos en tiempo de ejecución. Fuente: Elaboración propia.

Para dar respuesta al enfoque en que se ha desarrollado y aplicado técnicas en combinación con los modelos en tiempo de ejecución en el dominio de IoT se presenta la Tabla 3.6.

COD.	CRITERIO	RESPUESTAS	Nº ESTUDIOS	%
RQ2: ¿Para qué propósitos se han implementado los modelos en tiempo de ejecución en el dominio de internet de las cosas?				
EC2	Propósitos de los modelos en tiempo de ejecución	Monitoreo	31	62 %
		Aseguramiento de calidad (requerimientos no funcionales)	7	14 %
		Interoperabilidad	7	14 %

		Auto-adaptación (self-adaptation)	37	74 %
		Auto-optimización (self-optimization)	2	4 %
		Auto-organización (self-organization)	3	6 %
		Auto-recuperación / Tolerancia a fallos / consistencia	13	26 %
		Simulación / Predicción	11	22 %
		Otro	1	2 %

Tabla 3.6. Porcentajes individuales para el criterio de extracción EC2 de la sub-pregunta RQ2.

La Figura 3.2 muestra que los propósitos principales de los modelos en tiempo de ejecución en el dominio IoT son: autoadaptación (74%) y monitoreo (62%), en efecto porque los sistemas IoT operan en escenarios altamente cambiantes e inciertos, siendo complejos para obtener todos sus requisitos en las etapas de desarrollo de software, por lo tanto, los modelos en tiempo de ejecución se pueden aplicar de manera efectiva, con un alto nivel de abstracción, para monitorear el comportamiento de estos sistemas y su entorno, y si es necesario adaptar sus funcionalidades a las necesidades del usuario (Lee et al., 2018).

En un nivel intermedio, los modelos en tiempo de ejecución se centran en aumentar la tolerancia a fallas (26%), así como en simular o predecir las propiedades y el comportamiento del sistema IoT observado y, en última instancia, determinar los posibles impactos en su funcionamiento (22%).

Finalmente, los propósitos que requieren mayor preocupación de la comunidad científica son: interoperabilidad (14%), garantía de calidad (14%), auto-organización (6%) y auto-optimización (4%).

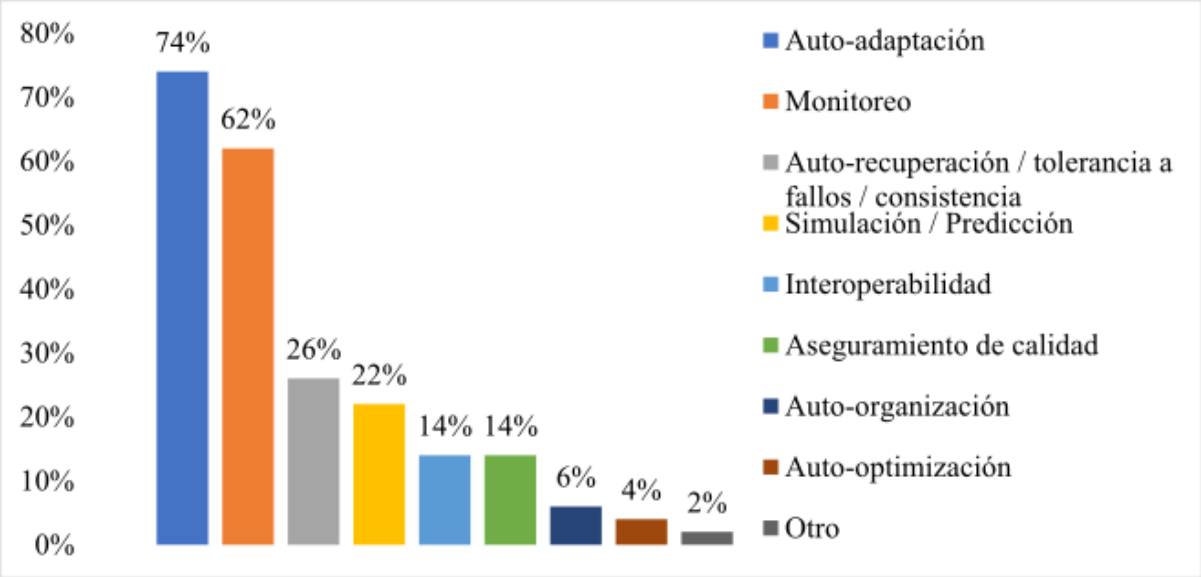


Figura 3.2. EC2 - Porcentaje de estudios primarios por propósito de aplicación. Fuente: Elaboración propia.

A continuación, se presentan en la Tabla 3.7 los resultados obtenidos para dar respuesta a la sub-pregunta 3 de investigación acerca de la forma en que se aborda la investigación en el desarrollo e implementación de los modelos en tiempo de ejecución en el dominio de IoT, y permite conocer el tipo de validación que con mayor frecuencia efectúan los estudios.

COD.	CRITERIO	RESPUESTAS	Nº ESTUDIOS	%
RQ3: ¿Cómo se aborda la investigación en los estudios relacionados a la implementación de infraestructuras que usan modelos en tiempo de ejecución en el dominio de internet de las cosas?				
EC4	Tipo de validación	Experimento	8	16 %
		Prototipo	8	16 %
		Prueba de concepto	6	12 %
		Caso de estudio	25	50 %
		Ninguno	7	14 %
		Encuesta	0	0 %

Tabla 3.7. Porcentajes individuales para el criterio de extracción EC4 de la sub-pregunta RQ3.

La Figura 3.3 presenta como tipo de validación empleada para evaluar las soluciones basadas en modelos en tiempo de ejecución en el dominio de IoT al caso de estudio

(50%). En un porcentaje menor, los estudios utilizan experimentos (16%), prototipos (16%), o pruebas de concepto (12%) y no se encontraron estudios que utilicen la encuesta como un tipo de validación. Finalmente, se puede ver que los estudios en un 14% no llevan a cabo ningún tipo de validación.

Como conclusión se puede decir que la mejor forma, pero no la única, como tipo de validación de los modelos en tiempo de ejecución es plantearse un caso de estudio en entornos de ciudades inteligentes (Cetina et al., 2009; Incki & Ari, 2018), plantación inteligente (Zhu et al., 2018), ambientes de vida asistida, espacios inteligentes (Freitas et al., 2014), centro comercial inteligente (Ayala et al., 2016), entre los más importantes.

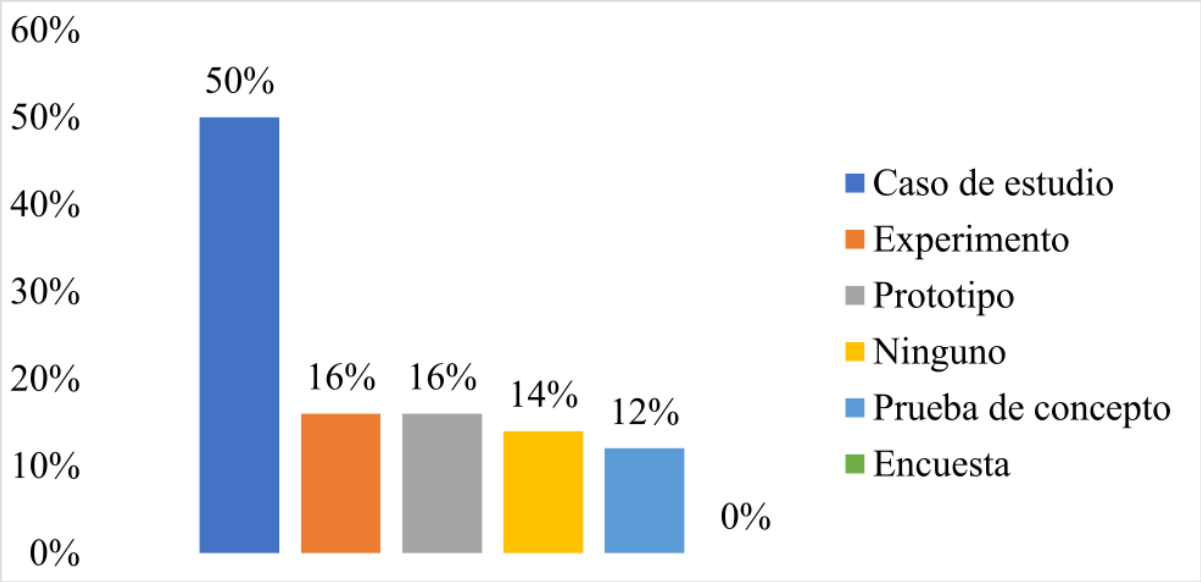


Figura 3.3. EC4 - Porcentaje de estudios primarios por tipo de validación. Fuente: Elaboración propia.

3.4.2. Comparación de criterios

A fin de obtener resultados más profundos se realizó diagramas de burbujas entre los criterios de extracción de tal manera que se puedan establecer comparaciones y contrastar diferentes criterios en el tema planteado.

La Figura 3.4 presenta un diagrama de burbujas entre los criterios de extracción EC1 y EC2, para contrastar cuáles son los propósitos principales de aplicar los modelos en tiempo de ejecución por subdominio IoT, así como para conocer las intersecciones donde existen brechas abiertas de investigación.

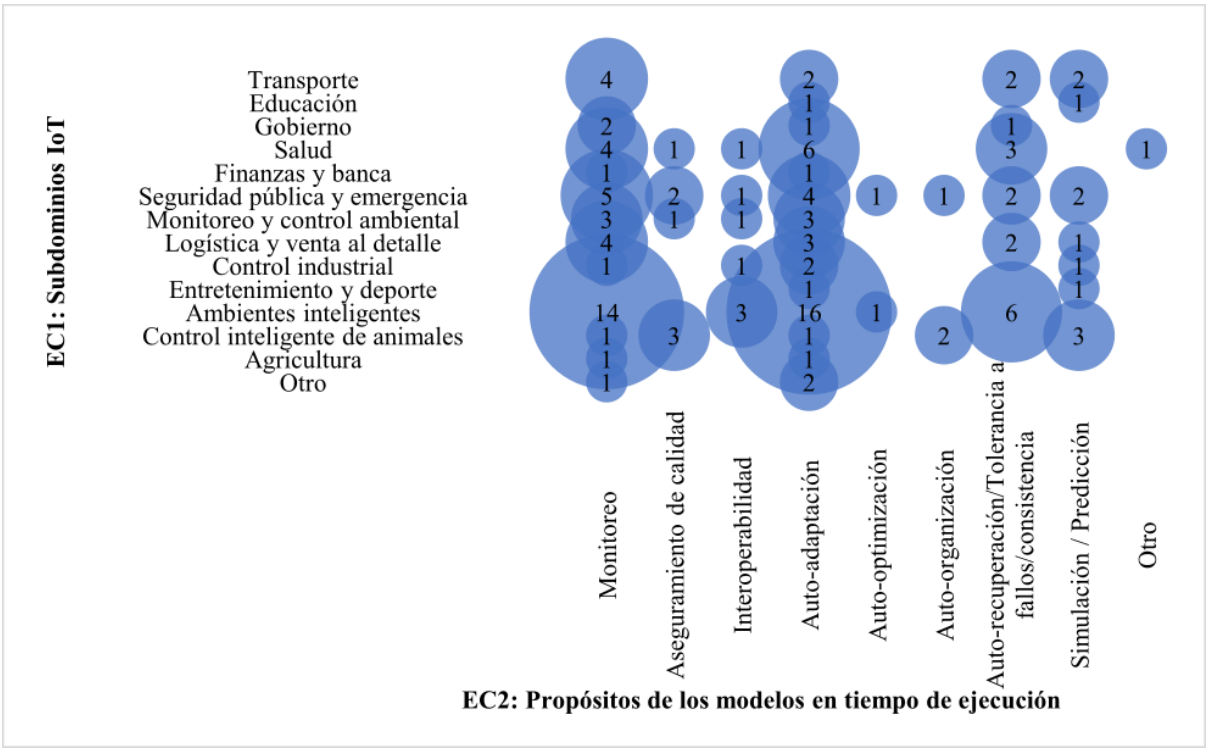


Figura 3.4. Comparación entre el criterio EC1: Subdominio de IoT contra el criterio EC2: Propósitos de los modelos en tiempo de ejecución.

Adicionalmente, se puede observar que los esfuerzos se han centrado en el uso de modelos en tiempo de ejecución para monitorear y adaptar los sistemas de IoT en entornos inteligentes, seguidos de salud, seguridad pública y emergencias, logística y transporte. También existe una presencia significativa de trabajo para aumentar la tolerancia a fallas de los sistemas IoT en el subdominio de entornos inteligentes. Por el contrario, hay muy pocos trabajos dirigidos a la interoperabilidad, garantía de calidad, auto-organización y auto-optimización.

Por otro lado, la Figura 3.5 presenta una comparación entre los criterios EC2: Propósitos de los modelos en tiempo de ejecución y EC5: Tipo de validación en el eje de las abscisas entre EC3: Técnicas utilizadas en combinación con los modelos en tiempo de ejecución en el eje de las ordenadas.

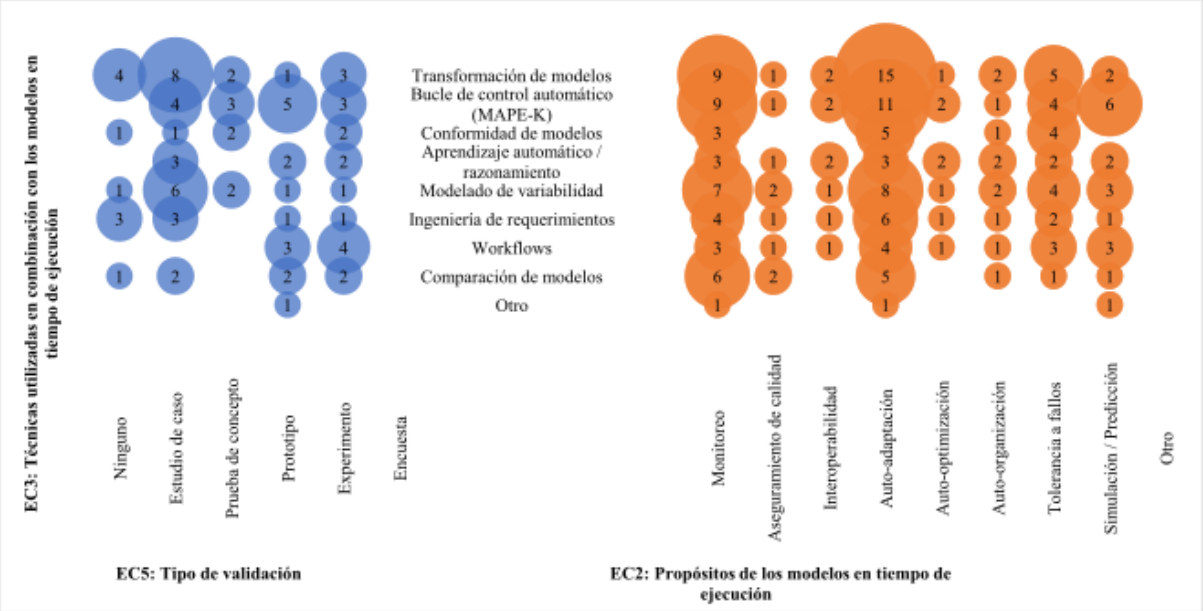


Figura 3.5. Comparación de EC2: Propósitos de los modelos en tiempo de ejecución y EC5: Tipo de validación entre EC3: Técnicas utilizadas en combinación con los modelos en tiempo de ejecución. Fuente: Elaboración propia.

En la figura se puede observar que en la mayoría de los estudios donde se utiliza técnica de Transformación de modelos y modelado de variabilidad se realiza la validación de estudio de caso. Además, en las técnicas de transformación de modelos y bucle de control MAPE-K se aplicó en la mayoría de los estudios, auto-adaptación como propósito de los modelos en tiempo de ejecución. Por último, los menos aplicados en los tipos de validación son la prueba de concepto en estudios donde se aplicó workflows como técnicas utilizada con los modelos en tiempo de ejecución.

3.4.3. Discusión

En este apartado se presenta un resumen de los resultados obtenidos una vez realizada la revisión de los estudios primarios con el objetivo de dar respuesta a la pregunta y sub-preguntas de investigación planteadas en la sección 3.2.1.

En primera instancia se describe el número total de estudios seleccionados en el mapeo sistemático en base al periodo de tiempo de búsqueda determinado desde el año 2006 hasta 2019.

La Figura 3.6 presenta el total de estudios por cada año, donde se obtiene como resultado que, en los años 2014, 2017 y 2018 se realizaron estudios de gran importancia en el contexto de esta investigación, de tal forma que en los últimos años el aporte e interés de la comunidad científica respecto a los modelos en tiempo de ejecución en el dominio de IoT continúa en constante crecimiento.

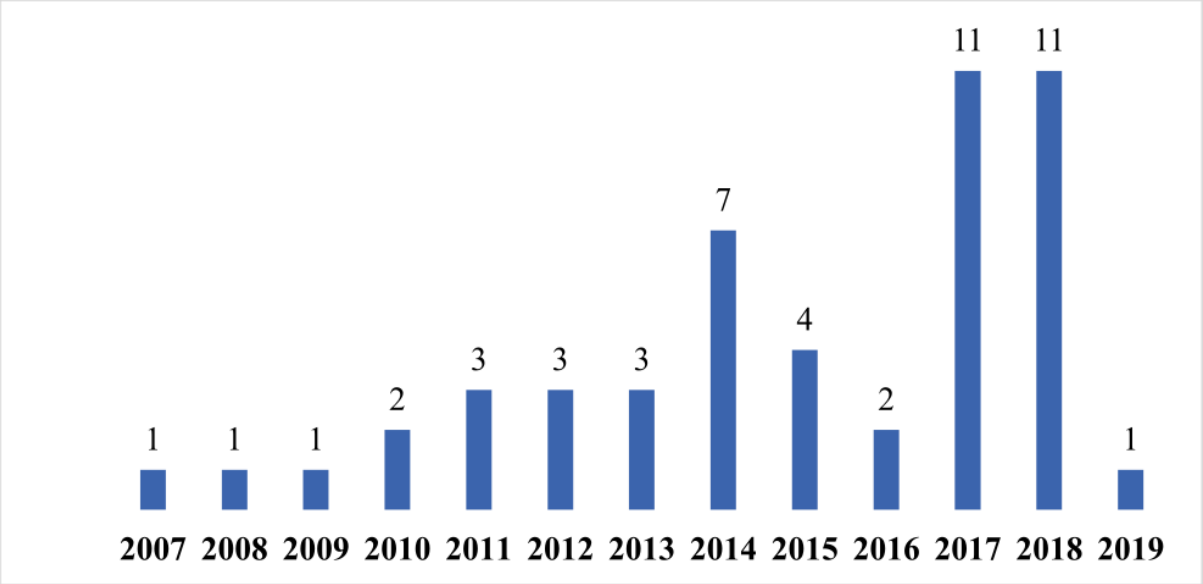


Figura 3.6. Número de estudios seleccionados por año de publicación para el mapeo sistemático. Fuente: Elaboración propia.

Una vez presentado los resultados de los estudios obtenidos por año, se procede a dar respuesta a las sub-preguntas y pregunta de investigación planteadas para el presente trabajo de titulación.

RQ1: ¿En qué sub-dominios se emplean los modelos en tiempo de ejecución dentro del dominio de internet de las cosas?

El contexto en el cual más se destaca el desarrollo e implementación de sistemas IoT basados en modelos en tiempo de ejecución es en ambientes inteligentes sean estos ciudades inteligentes (Cetina et al., 2009; Incki & Ari, 2018), plantación inteligente (Zhu et al., 2018), ambientes de vida asistida, espacios inteligentes (Freitas et al., 2014), centro comercial inteligente (Ayala et al., 2016), entre los más importantes, y el subdominio de salud, donde los modelos en tiempo de ejecución cumplen con el propósito de monitoreo y auto-adaptación de los sistemas.

Estos resultados demuestran la gran importancia del IoT en diferentes sectores, pero sobresale en ambientes relacionados con la automatización de entornos para beneficiar a los usuarios con el propósito de facilitar actividades y mejorar la calidad de vida de las personas.

RQ2: ¿Para qué propósitos se han implementado los modelos en tiempo de ejecución en el dominio de internet de las cosas?

Los modelos en tiempo de ejecución se han desarrollado mediante el uso de modelos y lenguajes específicos de dominio por medio de servicios, componentes, o capas que han permitido describir de forma más detallada la arquitectura o procesos de un sistema en el dominio de IoT. Todo esto se ha implementado usando las técnicas más relevantes de los modelos en tiempo de ejecución como la transformación de modelos

y el bucle de control automático (MAPE-K) que facilita el monitoreo, análisis, plan, ejecución y conocimiento base del sistema.

Además, dichas técnicas se aplican bajo con el propósito de auto-adaptación de los modelos arquitectónicos vinculados y actualizados en tiempo de ejecución, es decir, sus modificaciones se propagan automáticamente al software en ejecución, sin generar una degradación severa del rendimiento.

RQ3: ¿Cómo se aborda la investigación en los estudios relacionados con el desarrollo e implementación de modelos en tiempo de ejecución en el dominio de internet de las cosas?

La investigación de los estudios seleccionados muestra que la mayoría son una extensión de estudios previos, en algunos casos incorporando los modelos en tiempo de ejecución en métodos, marcos o herramientas existentes.

Generalmente las validaciones se enfocan en un caso de estudio en particular, ya sea en un hogar o en el desarrollo de actividades concretas, y por el contrario la encuesta como validación no es considerada en los estudios evaluados. Adicionalmente la investigación de los modelos en tiempo de ejecución en el dominio de IoT se realizaron bajo un enfoque académico a lo que se puede concluir que el ámbito académico está dedicado más a la investigación de tecnologías.

En base a las respuestas de cada sub-pregunta de investigación, se brinda respuesta a la pregunta de investigación general en el tema estudiado.

RQG: ¿Para qué y cómo se utilizan los modelos en tiempo de ejecución para ambientes IoT?

Los modelos en tiempo de ejecución han sido utilizados principalmente para el monitoreo y auto-adaptación, aplicados a los ambientes inteligentes y salud como subdominios de IoT; donde los estudios revelan como objetivo principal al usuario, para el cual se busca beneficiar, facilitar actividades y mejorar su calidad de vida. Para lo cual se han utilizado técnicas como transformación de modelos y el bucle de control automático (MAPE-K) que facilita el monitoreo, análisis, plan, ejecución y conocimiento base del sistema.

En resumen, por medio de los modelos en tiempo de ejecución se consigue adaptar la estructura y/o comportamiento de los sistemas en respuesta a los cambios en el contexto de ejecución y las diferentes necesidades del usuario, a través de la reconfiguración dinámica de modelos mientras el sistema se encuentra en ejecución.

Finalmente, los estudios demuestran que la importancia dada a estas tecnologías viene dada desde un enfoque académico, en el cual las investigaciones se validan por medio de casos de estudio, ya sea en un hogar o en el desarrollo de actividades concretas que corroboren la veracidad de la teoría descrita.

3.4.4. Amenazas a la validez

En el desarrollo del presente trabajo de titulación existen amenazas que pueden afectar en la validez del estudio de literatura, y para mitigar estos problemas se ha resuelto de la siguiente manera.

Entre algunas amenazas a la validez externa se encuentra el riesgo de sesgo en la selección de estudios ya que se observa un creciente número de trabajos de literatura lo cual no garantiza la captura completa de toda la información referente a las áreas del presente trabajo de titulación. Esta amenaza fue mitigada teniendo en cuenta todas las revistas principales en las áreas de estudio y se integró la búsqueda manual, la búsqueda automatizada y la búsqueda en forma de bola de nieve para obtener el conjunto final de estudios relevantes.

Respecto a las amenazas internas a la validez, estas vienen dadas por errores sistemáticos en el diseño y realización de la revisión de literatura. Para reducir esta amenaza se establece una metodología a seguir, y en este trabajo se aplicó la metodología de Kitchenham & Charters (2007). Además, la lectura y clasificación de los artículos se dividió en dos grupos entre los autores y se corroboró la información con el intercambio posterior de tal forma que los autores tengan conocimiento de todos los artículos seleccionados, al concluir la clasificación los resultados finales se derivan de la integración de sus resultados individuales.

Capítulo 4

4. Diseño e Implementación de la Infraestructura de Monitoreo

En este capítulo se presenta MIoT2InMon, una infraestructura de monitoreo de componentes sensibles al contexto mediante modelos en tiempo de ejecución. MIoT2InMon, permite monitorizar características y valores recopilados por componentes desplegados en un contexto construido con IoT. La infraestructura proporciona una serie de ventajas, tales como escalabilidad, flexibilidad, modificabilidad al momento de adicionar o reemplazar nuevos componentes e interoperabilidad entre entidades descritas en la infraestructura a través de distintos servicios de acceso de datos.
El capítulo se organiza de la siguiente manera: la sección 4.1 presenta una breve introducción de la infraestructura MIoT2InMon. La sección 4.2 describe los recursos utilizados en la construcción de la infraestructura. Finalmente, la sección 4.3 presenta la arquitectura de MIoT2InMon y describe los componentes principales.

4.1. Introducción

La infraestructura de monitoreo propuesta se ha denominado *MIoT2InMon,* nombre que hace referencia a las dos áreas principales de dominio y el propósito de crear una infraestructura de monitoreo que engloba el trabajo de titulación. El nombre se constituye considerando los siguientes acrónimos: "*M*", tomado de Models at runtime; "*IoT*", tomado de Internet of Things; "*2*", que alude a *to*; "*In*", tomado de Infraestructura y "*Mon*", tomado de Monitoreo.
MIoT2InMon significa el trabajo conjunto entre los modelos en tiempo de ejecución y el internet de las cosas para formar una infraestructura de monitoreo que sea el soporte de nuevas configuraciones en un sistema en ejecución sin necesidad de detenerlo.

La infraestructura de monitoreo MIoT2InMon está diseñada para brindar soluciones que minimicen la intervención humana en la construcción o soporte de sistemas implementados bajo el dominio de IoT mediante el uso de modelos en tiempo de ejecución. La infraestructura tiene como finalidad, facilitar la auto-adaptación de un sistema en ejecución sin necesidad de interrumpir su funcionamiento.

El modelo en tiempo de ejecución descrito para MIoT2InMon comprende la técnica del bucle autonómico de control, en la fase de monitoreo cuyo principio es medir los parámetros del sistema y almacenarlos de forma adecuada en el/los repositorios correspondientes, de tal forma que en un futuro se pueda efectuar las fases

subsecuentes del bucle MAPE-K de los modelos en tiempo de ejecución. En la Figura 4.1 se presenta el bucle MAPE-K donde se resalta la fase desarrollada y se muestra sus fases subsecuentes.

Figura 4.1. Bucle de control automático (P Cedillo, 2016). Fuente: Elaboración propia.

MIoT2InMon se construye a partir del meta-modelo Monitor-IoT propuesto por Erazo-Garzón et al., (2020) en el artículo *Monitor-IoT: A Domain Specific Language for designing monitoring architectures for IoT Systems*; cabe mencionar que Monitor-IoT se desarrolló alineado a la ISO/IEC - CD 30141 Arquitectura de referencia de internet de las cosas. A partir del meta-modelo, se desarrolla un DSL mediante la herramienta de SIRIUS que permite describir un estudio de caso enfocado a ambientes de vida asistida; como resultado se obtiene la representación de entidades y flujos de comunicación entre entidades, que da lugar a la implementación de la infraestructura.

MIoT2InMon se ha desarrollado en el lenguaje de programación JavaScript mediante la codificación de las clases descritas en el DSL y con ello se consigue desplegar la infraestructura en un entorno de vida asistida sean estos centros geriátricos, hogares, etc. para brindar asistencia a diferentes usuarios como adultos mayores y personas con discapacidad; sin embargo, se deja abierta la posibilidad de que MIoT2InMon sea aplicada en cualquier contexto relacionado con IoT.

En este aspecto, MIoT2InMon se utiliza en un estudio de caso relacionado con un ambiente de vida asistida, a fin de llevar a cabo la monitorización constante de componentes del IoT desplegados en el entorno. Como resultado se debe obtener la recopilación de información precisa y exacta cuando se realice modificaciones o nuevas configuraciones sin detener el sistema en ejecución.

4.2. Recursos utilizados en la construcción de la infraestructura

4.2.1. Meta-modelo Monitor-IoT

La construcción de la infraestructura de monitoreo se desarrolla en base al meta-modelo *Monitor-IoT* propuesto por Erazo-Garzón et al. (2020), como parte de su tesis doctoral denominada *"Metodología basada en modelos en tiempo de ejecución para la construcción de sistemas auto-conscientes para Internet de las cosas".*

Para el diseño de Monitor-IoT las meta-clases han sido alineadas a la norma ISO/IEC - CD 30141 Arquitectura de referencia del internet de las cosas (ISO/IEC, 2016), documento orientado a ingenieros y gerentes técnicos que van a desarrollar o diseñar aplicaciones de IoT. Además, facilita la especificación de las meta-clases de una forma ordenada y estandarizada utilizando un vocabulario común, diseños reutilizables y aplicando las mejores prácticas de la industria. La Figura 4.2 presenta las clases principales del meta-modelo Monitor-IoT.

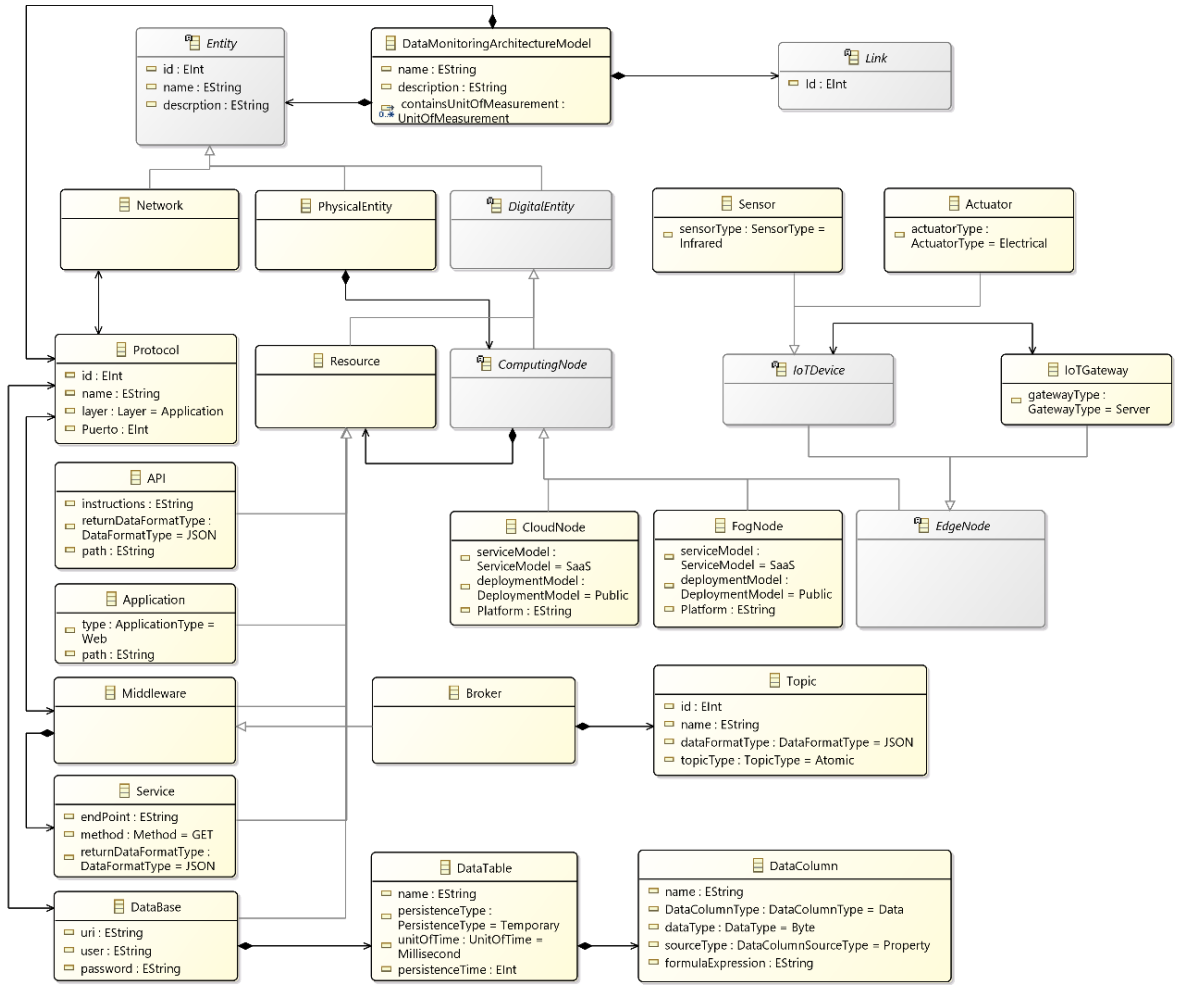


Figura 4.2. Meta-modelo Monitor-IoT (Erazo-Garzón et al., 2020).

El meta-modelo Monitor-IoT incluye como meta-clases principales:

DataMonitoringArchitectureModel, es la meta-clase inicial que engloba todo el modelo de arquitectura que representa un entorno de forma general. La meta-clase *IoTSystem* representa el sistema compuesto por diferentes componentes IoT (por ejemplo, sistema de emergencia). La meta-clase *Entity* es la representación general de los objetos del mundo real (*Physical Entity*), elementos computacionales (*DigitalEntity*) o redes de comunicación (Network), formada por diversas propiedades que permiten identificarse en un *IoTSystem.*

Una *Entity* puede estar formada por diversas características del tipo *GenericProperty* y *SpecificProperty. Las GenericProperty* representan propiedades generales que otras *Entity* también poseen, por ejemplo: la marca de un sensor, nombre, tipo, etc. En cambio, las *SpecificProperty* son propiedades propias de una *Entity* y que pueden dar valor a las *GenericProperty*, por ejemplo: marca: Philips, nombre: sensorDHL11, etc.

La meta-clase *Physical Entity* representa objetos del mundo real; por el contrario, la meta-clase *DigitalEntity* es un elemento computacional o de datos que incluyen aplicaciones, servicios y almacenes de datos, y la meta-clase *Network* permite la comunicación de datos entre entidades a través de protocolos. Un *Protocol* describe el puerto de comunicación entre las entidades y la capa utilizada dentro del modelo OSI.

Entre las meta-clases que se pueden extender de una *Physical Entity* son: *HumanUser* representa una persona que interactúa en el *IoTSytem* y *NonHumanUser* representa un objeto que realiza tareas mediante el consumo de servicios, como por ejemplo un robot.

Las meta-clases que se pueden extender de una *DigitalEntity* son*: ComputingNode* y *Resource*. Un *ComputingNode* es un tipo de tecnología que posee capacidad de procesamiento y el uso de servicios, entre los que se destaca *CloudNode, FogNode* y *EdgeNode.*

La meta-clase *CloudNode*, representa la computación en la nube destinada al procesamiento y almacenamiento de información. *FogNode*, permite procesar información antes de enviarla a la nube. *EdgeNode* permite extender *IoTDevices* y *IoTGateway*; un *IoTDevice* es un dispositivo IoT que recopila información mediante un sensor o efectúa una acción a través de un actuador. *IoTGateway* es un dispositivo capaz de receptar y/o emitir información a los *IoTDevices*.

La meta-clase *Resource* describe una herramienta que brinda funcionalidades como almacenamiento, servicios o configuraciones en un *ComputingNode*. Entre las meta-clases que derivan de *Resource* son: *DataBase*, *Middleware, Service, Application, API*.

La meta-clase *DataBase* representa una base de datos donde se realiza el almacenamiento de la información recopilada desde los *IoTDevice*. Esta meta-clase está compuesta por *DataTable* que a su vez se compone de un *DataColumn* y cada *DataColumn* almacena el/los valores recopilados desde un *IoTDevice.*

La meta-clase *DataFlow* describe la ruta de proceso que deben seguir las *Entity* para la comunicación y envío de información de un punto a otro. Adicionalmente un *DataFlow* puede contener *DataMappingRule* que son sentencias o consultas que se ejecutan de forma directa en los *DataBase,* a fin de simplificar el uso de servicios intermedios.

La comunicación entre meta-clases de un *DataFlow* puede ser de tipo asíncrono, recibe y envía datos de forma directa según un tópico de suscripción mediante un agente; y síncrono que envía datos a través de servicios, una comunicación directa entre los dispositivos y la base de datos.

La ruta de proceso puede ser de cuatro tipos: *DataCollectionFlow, DataAggregationFlow, ActuatingFlow, ReportGenerationFlow*. Para la infraestructura propuesta se hace uso de *DataCollectionFlow* y *DataAggregationFlow,* sin embargo, los tipos restantes se implementan en trabajos futuros.

El tipo *DataCollectionFlow* declara una ruta de proceso que sólo recopila datos y los envía, mientras que el tipo *DataAggregationFlow* especifica la ruta de proceso para agregar datos en el/los *DataBase.*

La meta-clase *Link*, permite definir las *Entity* conectadas por donde se lleva a cabo un *DataFlow*.

La meta-clase *Middleware* sirve como un orquestador de tipo síncrono, que mantiene comunicación mediante servicios.

Por el contrario, un *Broker* es un intermediario de tipo asíncrono que recibe y envía datos de forma directa a las *Entity* según el tópico al que se suscribieron. La meta-clase *Topic* es un nombre que ayuda a identificar los mensajes que se envían bajo una temática dentro de la arquitectura.

La meta-clase *Service* define los parámetros necesarios para propagar la comunicación entre aplicaciones cliente y servidor, sobre todo especificando el tipo de método del servicio sea GET, POST, DELETE, OPTIONS y PUT.

La meta-clase *Application* es un componente de software que ofrece una colección de funciones con las que un usuario puede realizar una tarea.

Finalmente, la meta-clase API es el método que permite obtener y enviar información de los *IoTDevice*; es común utilizarla en la comunicación con *Application*.

Cabe señalar que la descripción detallada de las meta-clases de *Monitor-IoT* son presentadas en el artículo "*Monitor-IoT: A Domain Specific Language for designing monitoring architectures for IoT Systems*" (Erazo-Garzón et al., 2020).

4.2.2. DSL-Monitor-IoT

Un lenguaje de dominio específico (*DSL - Domain Specific Language*) es considerado como un lenguaje de programación con un mayor nivel de abstracción optimizado para una clase específica de problemas (JetBrains, 2000).

Uno de los usos más frecuentes de un DSL radica en el desarrollo de software dirigido por modelos, ya que permiten más expresión y facilidad de uso en comparación con un lenguaje de propósito general. Además, un DSL puede aumentar la calidad del producto creado: menos errores, mejor conformidad arquitectónica y mayor capacidad de mantenimiento (JetBrains, 2000).

La aplicabilidad del meta-modelo Monitor-IoT se desarrolla a través de un DSL creado en *Obeo Designer Community Edition* (JetBrains, 2000), un paquete gratuito de Eclipse que incluye solo tecnologías de código abierto: Sirius, Ecore Tools y EMF Compare.

En este trabajo se ha seleccionado Sirius (Foundation, 2004), un proyecto de código abierto de la fundación de Eclipse creado por Obeo y Thales que permite crear fácilmente un banco de trabajo de modelado personalizado, ya sea basado en lenguajes específicos de dominio (DSL) o metamodelos estándar (p. ejm., UML, SysML, Togaf, BPMN.), a través de las tecnologías de modelado de Eclipse (Foundation, 2004; JetBrains, 2000). Los bancos de trabajo de modelado creados con Obeo Designer están compuestos por un conjunto de editores (diagramas, tablas y árboles) determinados por un modelo que define la estructura completa del banco de trabajo de modelado, su comportamiento y todas las herramientas de edición y navegación, de tal forma que para los usuarios sea fácil crear, editar y visualizar modelos en el framework de modelado de Eclipse (*Eclipse Modeling Framework - EMF*)(Foundation, 2004).

EMF es útil para la creación de modelos y metamodelos, con facilidades para la generación automática de código y para la construcción de herramientas basadas en modelos (Foundation, 2004).

Por lo tanto, mediante Obeo se desarrolla un DSL como representación gráfica de las meta-clases de Monitor-IoT, en la Figura 4.3 se presenta los componentes creados a partir del meta-modelo a fin de brindar mayor facilidad para crear, editar y visualizar las meta-clases de forma gráfica.

En el siguiente capítulo se utiliza el DSL Monitor-IoT para instanciar un AAL a modo de integración de la solución en un dominio específico.

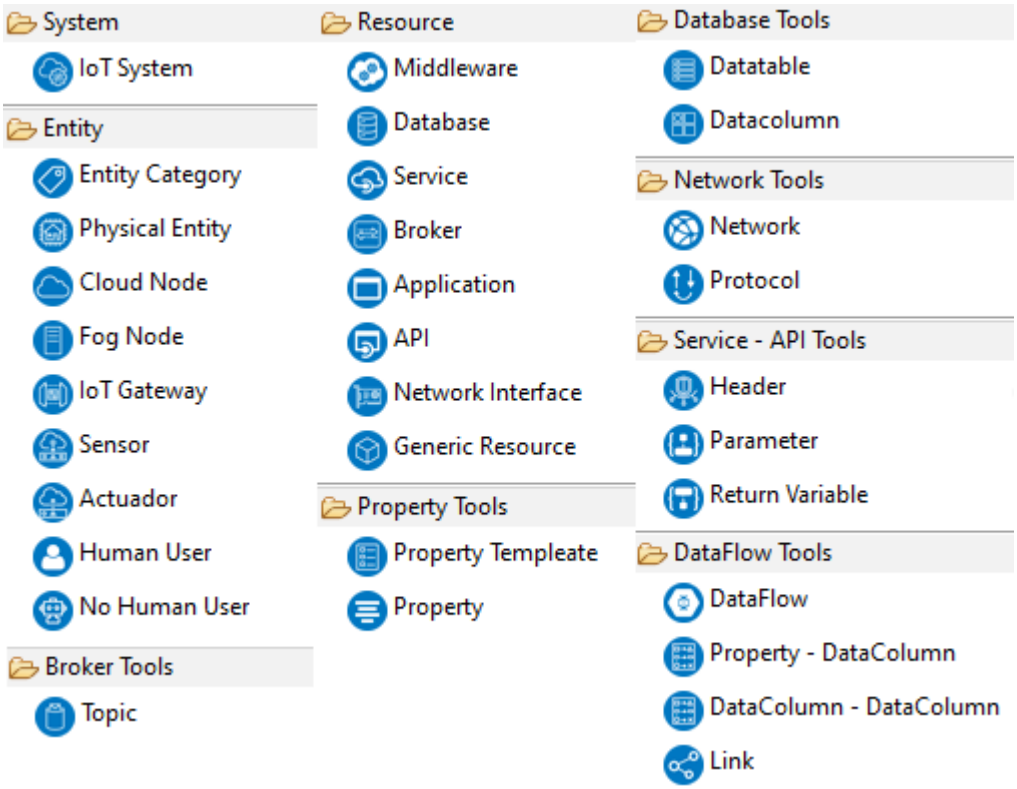


Figura 4.3. Componentes del DSL.

El DSL agrupa los componentes de la instancia de las meta-clases de Monitor-IoT por secciones entre las cuales: System, Entity, Resource, Property Tools, Database Tools, Network y Service - API Tools.

La Tabla 4.1 presenta la representación gráfica de cada uno de los componentes del DSL (Monitor-IoT) que en el siguiente capítulo se utilizan para representar un DSL de una arquitectura para IoT de un entorno de vida asistida.

La representación del DSL es una parte importante en la infraestructura MIoT2InMon, puesto que el archivo resultante sirve como entrada para las respectivas configuraciones de monitoreo de MIoT2InMon.

ELEMENTO	REPRESENTACIÓN GRÁFICA	DESCRIPCIÓN
System		
IoT System		Representa el entorno IoT que se desea monitorear.
Entity		

Entity Category		Categoriza las entidades tanto físicas como digitales.
Physical Entity		Representa objetos del mundo real.
Cloud Node		Representa el almacenamiento en la nube.
Fog Node		Es un nodo intermedio para almacenar y procesar datos antes de enviar a la nube.
IoT Gateway		Nodo de computación capaz de controlar sensores y actuadores receptando y enviando información desde o hacia estos dispositivos.
Sensor		Dispositivo que está capacitado para detectar acciones o estímulos externos.
Actuador		Dispositivo que recibe una orden y activa un elemento final.
Resource		
Middleware		Orquestador que mantiene comunicación mediante servicios.
DataBase		Lugar donde se realiza el almacenamiento de la información.
Service		Permite intercambiar datos entre aplicaciones cliente y servidor.
Broker		Intermediario que recibe y envía datos de forma directa según un tópico al que esté suscrito.
Application		Componente de software que ofrece una colección de funciones.
API		Método que permite obtener y enviar información desde o hacia los dispositivos IoT.
Network Interface		Permite la comunicación entre los diferentes componentes.
Property Tools		

Property Template		Propiedad genérica que comparten un conjunto de entidades del mismo tipo.
Property		Propiedad específica de una entidad.
Database Tools		
DataTable		Es una tabla de la DB que contiene Data Column.
DataColumn		Almacena el/los valores recopilados.
Network Tools		
Network		Describe el tipo de red sea de proximidad, LAN o Internet.
Protocol		Permite que dos o más entidades se comuniquen entre ellas para transmitir información.
Service - API Tools		
Header		Información adicional que se envía en un Servicio.
Parameter		Valores que recibe un Servicio/API.
Return Variable		Valores que retorna un Servicio/API.
Broker Tools		
Topic		Identificador que ayuda a ubicar los mensajes que se envían.
Dataflow Tools		
Data Flow		Proceso para la comunicación entre los dispositivos y la base de datos.
Link		Permite definir las Entity conectadas por donde se lleva a cabo un DataFlow.

Tabla 4.1. Componentes del DSL (Erazo-Garzón et al., 2020).

Adicionalmente, la descripción detallada de la estructura, funcionalidad y evaluación empírica del DSL se presenta en el artículo "*Monitor-IoT: A Domain Specific Language for designing monitoring architectures for IoT Systems*" (Erazo-Garzón et al., 2020), próximo a publicarse, el cual fue desarrollado dentro del contexto de este trabajo de titulación y la tesis doctoral de Erazo-Garzón (2021).

4.3. Arquitectura de la implementación de MIoT2InMon

La arquitectura de la implementación de la Infraestructura de monitoreo MIoT2InMon se presenta en la Figura 4.4 con sus componentes principales, desde el DSL como punto de inicio hasta la configuración del entorno.

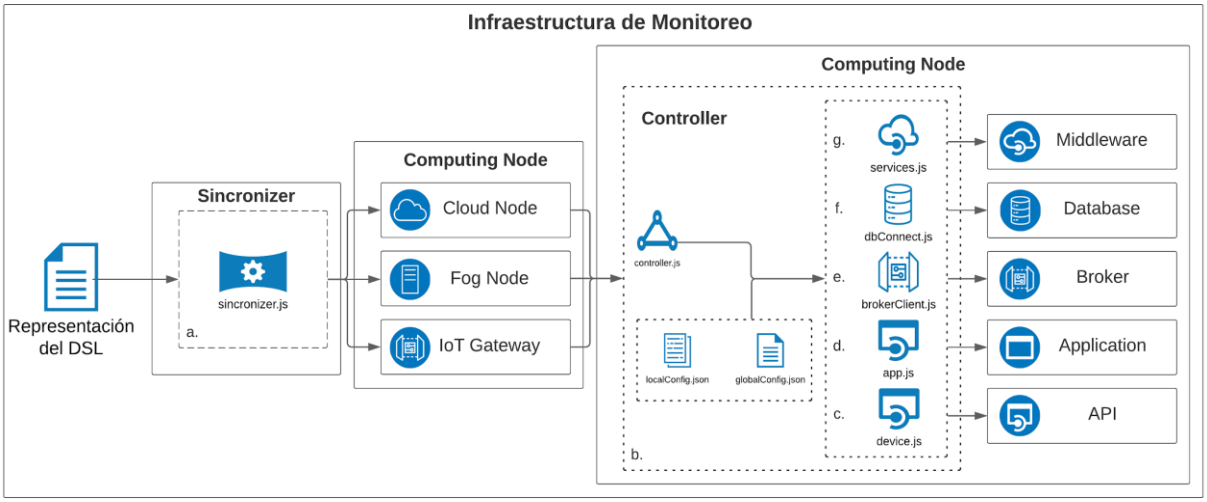


Figura 4.4. Infraestructura de Monitoreo. Elaboración propia.

La infraestructura está formada por dos elementos indispensables que representan el middleware de configuración de los componentes; el primer elemento *Sincronizer,* Figura 4.4.a recibe como entrada la representación del DSL y el segundo elemento *Controller,* Figura 4.4.b configura y ejecuta los diferentes elementos existentes en la arquitectura IoT.

Para el desarrollo de la infraestructura de monitoreo MIoT2InMon se ha utilizado Node.js (Node.js, 2021), framework de JavaScript. Se eligió Node.js, por ser un entorno de código abierto multiplataforma que ejecuta código JavaScript un lenguaje de programación simple e interpretado.

4.3.1. Componentes de la infraestructura de monitoreo

La infraestructura se despliega a partir del ingreso de la representación del DSL. El *DSL* permite definir el entorno que se va a monitorear de forma gráfica, además describe cómo se va a almacenar información, como va a ser el envío y la recepción

de datos y flujos de comunicación entre los dispositivos. Como resultado se obtiene un archivo que se procesa en la infraestructura a través del *Sincronizer*.

El *Sincronizer* es el encargado de procesar la representación del DSL, dicha representación es en formato XML. Para un mejor manejo de la información en JavaScript, se transforma de formato XML a JSON y se almacena en un archivo (*modelTransform.json*). Siguiente, se procesa la nueva transformación para conocer todos los diferentes nodos de computación *(Cloud Node, Fog Node e Iot Gateway)* existentes en la arquitectura. A continuación, se busca la dirección host de cada nodo de computación para enviar la configuración local y global al *Controller*.

El *Controller* se ejecuta en cada nodo de computación y se encarga de manejar, configurar e iniciar todas las entidades descritas en la representación del DSL. Para la recepción de la información enviada por el *Sincronizer*, existe un servicio de configuración que guarda el resultado en dos diferentes archivos (*controller.js*). El primer archivo (*localConfig.json*) almacena la información que le pertenece al dispositivo. El segundo archivo (*globalConfig.json*) guarda la información completa del sistema. Para configurar los recursos existentes en cada nodo de computación, hay dos procesos, la configuración y la comunicación.

Para la configuración se tiene cuatro pasos:

1) El nodo debe contener al menos una API (Figura 4.4.c) y una Aplicación (Figura 4.4.d), lo que indica que se comunica con dispositivos IoT. Para la configuración se accede a la información local y se extraen las instrucciones que debe ejecutar el dispositivo para enviar/recibir información desde el nodo (*device.js).* Adicional, se establece la configuración de las variables y tipo de datos que va a recibir/enviar la aplicación (*app.js*).

2) El nodo debe contener al menos un Broker (Figura 4.4.e) para inicializar el cliente/servidor y obtener la lista de los diferentes topics que van a suscribir/publicar información (*brokerClient.js*).

3) El nodo debe contener al menos una Base de Datos (Figura 4.4.f) para proceder a obtener el los Datatables y los Datacolumns, y generar el URI para inicializar la comunicación (*dbConnect.js*).

4) El nodo debe tener al menos un middleware (Figura 4.4.g) para configurar y establecer los diferentes servicios que va a brindar el nodo. En cada servicio se establece su método, su endpoint y en caso de existir, sus parámetros (*services.js*).

Una vez realizada la configuración e inicialización de todos los recursos de la infraestructura, se procede a comunicarles, para esto se hace uso de los Data Flows, su tipo, su tipo de comunicación y su regla de mapeo. Para la comunicación existen dos procesos:

1) Si *Dataflow* es de tipo Agregación, debe tener una regla de mapeo *DataColumnToDataColumn.* Al ser un caso de agregación, es un proceso que se ejecuta de manera automática y el tiempo se establecerá en el propio *Dataflow*. La comunicación será *DataColumn* origen > *Service* > *DataColumn* destino

2) Si *Dataflow* es de tipo Colección, debe tener una regla de mapeo *PropertyToDataColumn.*

    a) Si el *Dataflow* tiene un tipo de comunicación síncrona, la comunicación será *API* > *Application* > *Service* > *Database*

    b) Si el *Dataflow* tiene un tipo de comunicación asíncrona, la comunicación será *API* > *Application* > *Broker* > *Service* > *Database*

Con todos los archivos creados y las configuraciones realizadas se tiene como resultado el sistema en ejecución que representa el entorno que se va a monitorear. Si en lo posterior se requiere nuevas configuraciones se las debe realizar a nivel del DSL y todo el sistema se reconfigura de forma automática, de tal manera que el flujo de monitoreo de datos no se vea interrumpido y evita en todo sentido el detener el sistema y perder información sensible al contexto.

En resumen, la configuración de la MIoT2InMon consta de tres pasos. Como primer paso, se requiere la arquitectura representada en un DSL; segundo paso, desplegar un sincronizador para leer el DSL y crear los controladores por cada nodo de la arquitectura; y tercer paso, cada uno de los controladores se encarga de crear y desplegar los servicios, API´s y flujos de comunicación.

Finalmente, se obtiene la infraestructura de monitoreo de componentes IoT MIoT2InMon, configurada y compuesta por cada uno de los archivos creados en los pasos anteriores de forma automática a nivel de un DSL. Además, MIoT2InMon puede ser reconfigurada en tiempo de ejecución a nivel de DSL sin necesidad de detener el flujo de funcionamiento del sistema.

4.3.2. Implementación de la infraestructura de monitoreo

La implementación de monitoreo MIoT2InMon se desarrolla en el framework NodeJS y requiere como entrada un DSL que represente la arquitectura de monitoreo de IoT para ser desplegada.

La arquitectura planteada se muestra en la Figura 4.5, la cual forma parte de la evaluación realizada en el estudio de caso del presente trabajo de titulación y su documentación se detalla en el Apéndice C.

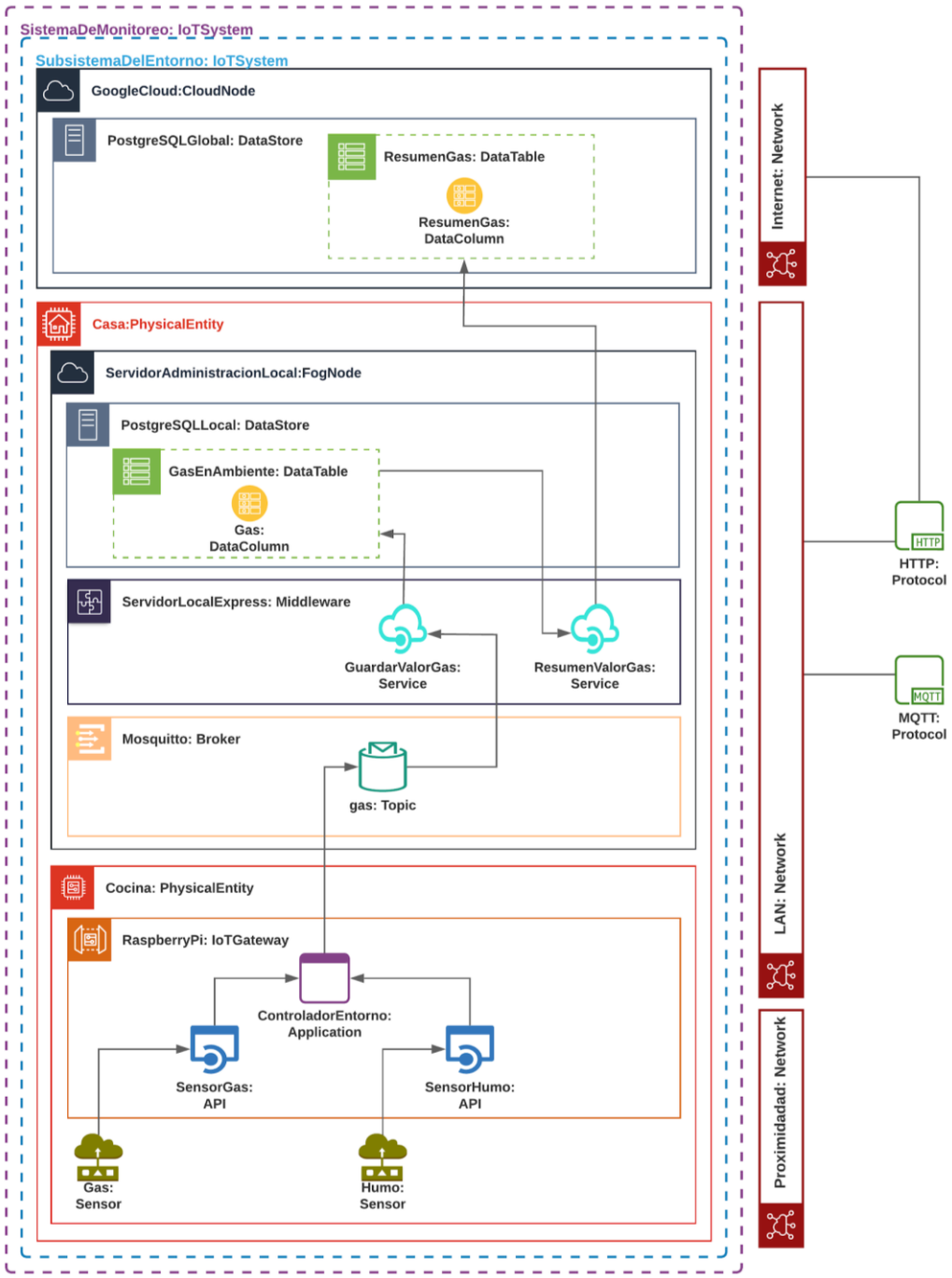


Figura 4.5. Arquitectura de internet de las cosas de un ambiente de vida asistida.

La Figura 4.5 se desarrolla mediante el DSL Monitor-IoT y describe los parámetros para el almacenamiento de información, la forma en que será el envío y recepción de datos y los flujos de comunicación entre los dispositivos.

Como resultado de la arquitectura del internet de las cosas presentada en la Figura 4.5 se obtiene el DSL descrito en la Figura 4.6, donde se representan los componentes planteados del DSL Monitor.

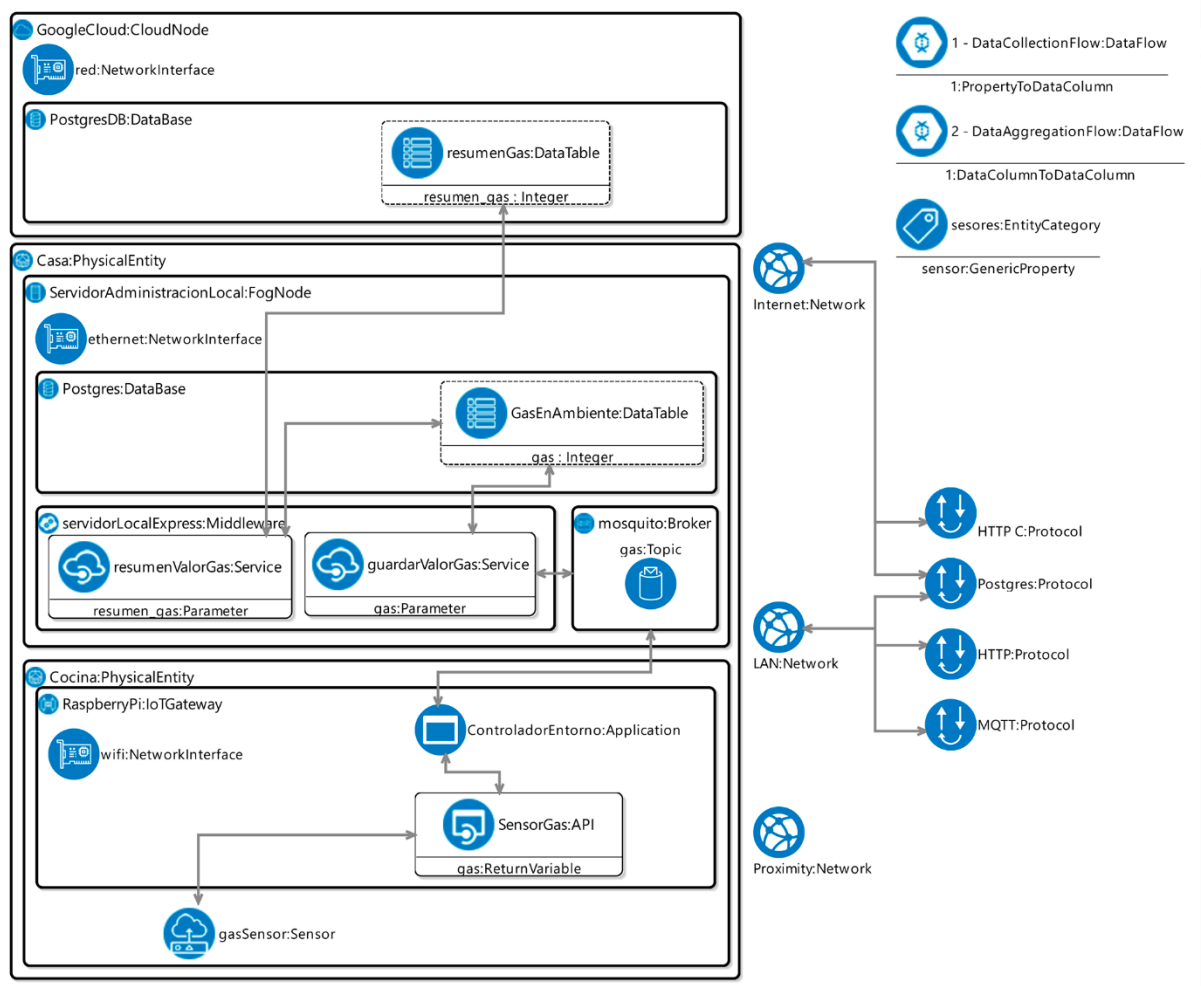


Figura 4.6. DSL de la arquitectura de internet de las cosas para un ambiente de vida asistida.

La infraestructura se despliega a partir de la representación del DSL y se ejecuta el middleware de configuración a través de sus dos elementos principales *Sincronizer* y *Controller*.

El *Sincronizer* se encarga de procesar la representación del DSL cuyo formato es de tipo XML, la Figura 4.7 muestra la definición del DSL de cada componente de la arquitectura con sus atributos respectivos; para un mejor manejo de la información en la infraestructura se transforma de formato XML a JSON y se almacena en un archivo denominado *modelTransform.json* (Ver Figura 4.8).

```
<containsEntity xsi:type="MonitorIoT:PhysicalEntity" id="1" name="Casa">
  <containsComputingNode xsi:type="MonitorIoT:CloudNode" id="1" name="Admin Node">
    <containsResource xsi:type="MonitorIoT:DataBase" id="1" name="new DataBase 1" uri="postgres://tiiienwz:9g4OW6s6JmwajeRpxhFDfIoYA5o0p2ia@salt.d
      <containsDataTable name="temperature_cn" hasLinkServiceToDatable="//@containsLink.0">
        <composedOfDataColumn name="temperature_cn" hasRuleAsDestination="//@containsDataFlow.1/@containsDataMappingRule.0" dataType="Integer"/>
      </containsDataTable>
    </containsResource>
    <containsResource xsi:type="MonitorIoT:Middleware" id="1" name="middleware 1" usesProtocol="//@containsProtocol.3">
      <containsService id="1" name="saveData" endPoint="save_t_h" method="POST" hasLinkServiceToDataTable="//@containsLink.0 //@containsLink.7">
        <containsParameter name="temperature_cn" dataType="Integer"/>
      </containsService>
    </containsResource>
    <containsResource xsi:type="MonitorIoT:NetworkInterface" id="3" name="red" networkAddress="localhost"/>
  </containsComputingNode>
  <subPhysicalEntity id="1" name="Casa">
    <containsComputingNode xsi:type="MonitorIoT:IoTGateway" id="4" name="IoTGateway 4" controls="//@containsEntity.1">
      <containsResource xsi:type="MonitorIoT:API" id="1" name="API" instructions="var temperature = Math.round(Math.random() * (100 - 30)) + 30; r
        <containsParameter name="time" dataType="Date"/>
        <containsReturnVariable name="temperature_fn" dataType="Integer" returns="//@containsEntityCategory.0/@has.0"/>
      </containsResource>
      <containsResource xsi:type="MonitorIoT:Application" id="2" name="App" hasLinkAppToService="//@containsLink.8" hasLinkAppToAPI="//@containsLi
      <containsResource xsi:type="MonitorIoT:NetworkInterface" id="3" name="wifi" networkAddress="localhost"/>
    </containsComputingNode>
```

Figura 4.7. Estructura del archivo del DSL denominado model en formato XML. Fuente: Elaboración propia.

```
"containsComputingNode": [
    {
        "$": {
            "xsi:type": "MonitorIoT:FogNode",
            "id": "3",
            "name": "ServidorAdministracionLocal"
        },
        "containsResource": [
            {
                "$": {
                    "xsi:type": "MonitorIoT:Middleware",
                    "id": "1",
                    "name": "servidorLocalExpress",
                    "usesProtocol": "//@containsProtocol.0"
                },
                "containsService": [
                    {
                        "$": {
                            "id": "1",
                            "name": "resumenValorGas",
                            "endPoint": "save_resumen_gas",
                            "method": "POST",
                            "hasLinkServiceToDataTable": "//@containsLink.5 //@containsLink.6"
                        },
                        "containsParameter": [
                            {
                                "$": {
                                    "name": "resumen_gas",
                                    "dataType": "Integer"
                                }
                            }
```

Figura 4.8. Estructura del archivo del DSL denominado modelTransform en formato JSON. Fuente: Elaboración propia.

A continuación, se procesa el nuevo archivo para conocer todos los diferentes nodos de computación (*Cloud Node, Fog Node e Iot Gateway*) existentes en la arquitectura. Se busca la dirección host de cada nodo de computación para enviar la configuración local y global al *Controller*.

El *Controller* se ejecuta en cada nodo de computación y se encarga de manejar, configurar e iniciar todas las entidades descritas en la representación del DSL. Para la recepción de la información enviada por el *Sincronizer*, existe un servicio de configuración (*controller.js*) que guarda el resultado en dos diferentes archivos. El primer archivo *localConfig.json* almacena la información que le pertenece al dispositivo como se presenta en la Figura 4.9.


```
"$": {
    "xsi:type": "MonitorIoT:CloudNode",
    "id": "1",
    "name": "GoogleCloud"
},
"containsResource": [
    {
        "$": {
            "xsi:type": "MonitorIoT:DataBase",
            "id": "1",
            "name": "PostgresDB",
            "uri": "postgres://tiiienwz:9g4OW6s6JmwajeRpxhFDfIoYA5o0p2ia@salt.db.elephantsql.com:5432/tiiienwz"
        },
        "containsDataTable": [
            {
                "$": {
                    "name": "resumen_gas",
                    "hasLinkServiceToDatable": "//@containsLink.5"
                },
                "composedOfDataColumn": [
                    {
                        "$": {
                            "name": "resumen_gas",
                            "hasRuleAsDestination": "//@containsDataFlow.1/@containsDataMappingRule.0",
                            "dataType": "Integer"
                        }
                    }
                ]
            }
        ]
    },
```


Figura 4.9. Estructura del archivo localConfig. Fuente: Elaboración propia.

El segundo archivo *globalConfig.json* guarda la información completa del sistema como se describe en la Figura 4.10.

```
{
    "$": {
        "xsi:type": "MonitorIoT:Network",
        "id": "3",
        "name": "Proximity"
    }
},
{
    "$": {
        "xsi:type": "MonitorIoT:PhysicalEntity",
        "id": "1",
        "name": "Casa"
    },
    "containsComputingNode": [
        {
            "$": {
                "xsi:type": "MonitorIoT:FogNode",
                "id": "3",
                "name": "ServidorAdministracionLocal"
            },
            "containsResource": [
                {
                    "$": {
                        "xsi:type": "MonitorIoT:Middleware",
                        "id": "1",
                        "name": "servidorLocalExpress",
                        "usesProtocol": "//@containsProtocol.0"
                    },
                    "containsService": [
                        {
                            "$": {
                                "id": "1",
```


Figura 4.10. Estructura del archivo globalConfig. Fuente: Elaboración propia.

Para configurar los recursos existentes en cada nodo de computación, hay dos procesos, la configuración y la comunicación. La configuración permite corroborar que el componente contenga al menos una aplicación, al menos un broker, al menos un middleware y al menos una base de datos; estos recursos se definen como necesarios para llevar a cabo el proceso de monitorización de los dispositivos IoT.

Una vez realizada la configuración e inicialización de todos los recursos de la infraestructura, se procede a realizar la comunicación entre ellos, para lo cual se hace uso de los Data Flows, tipo, tipo de comunicación y su regla de mapeo.

Con todos los archivos creados y las configuraciones realizadas se tiene como resultado el sistema en ejecución que representa el entorno que se va a monitorear, los datos recopilados del proceso de monitorización se describen en la Figura 4.11.

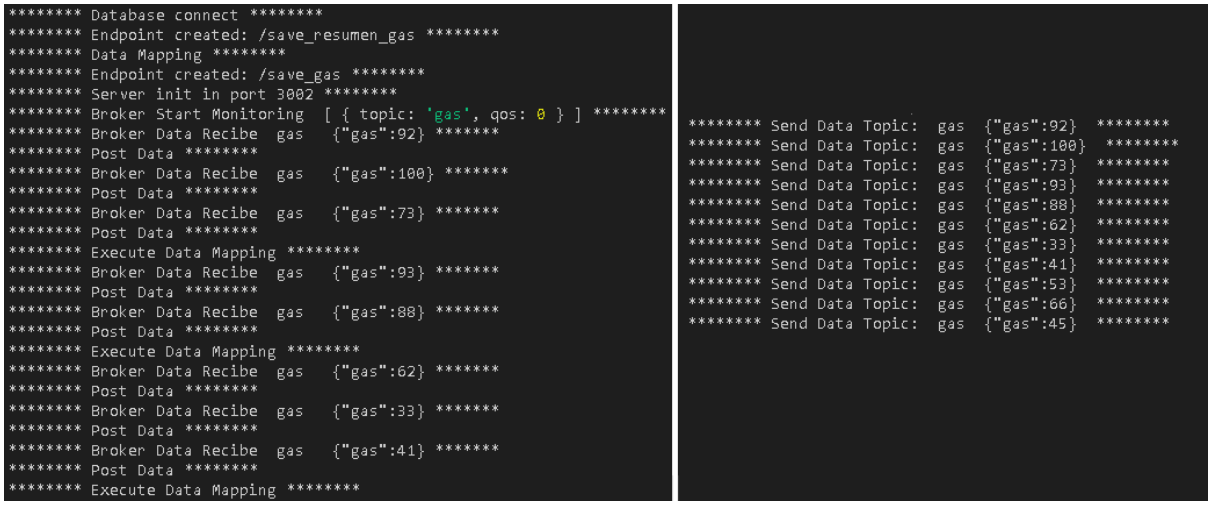


Figura 4.11. Estructura de los datos recopilados en el proceso de monitorización.

Finalmente, si en lo posterior se requiere nuevas configuraciones o modificaciones se las debe realizar a nivel del DSL y la infraestructura de monitoreo MIoT2InMon se encarga de que todo el sistema se reconfigure de forma automática, de tal manera que el flujo de monitoreo de datos no se vea interrumpido y evita en todo sentido el detener el sistema y perder información sensible al contexto.

Capítulo 5

5. Evaluación del estudio de caso Monitor-IoT

En el presente capítulo se describe la evaluación del estudio de caso del DSL (Monitor-IoT) a través de una arquitectura aplicada al dominio de AAL. La evaluación de un estudio de caso tiene como objetivo proporcionar resultados de una investigación a partir de proyectos del mundo real.

Para realizar la evaluación de un estudio de caso se han seguido los pasos de la metodología planteada por Runeson & Höst (2009), la cual consta de cinco actividades esenciales: i) Diseñar y planificar el estudio de caso; ii) Preparar la recolección de datos; iii) Recolección de datos; iv) Analizar e interpretar los datos recopilados y v) Informar sobre los resultados obtenidos.

La metodología aplicada se distribuye de la siguiente manera: la sección 5.1 describe las preguntas de investigación, hipótesis y los pasos aplicados para la evaluación del estudio de caso, tareas englobadas en la actividad de diseño y la planificación de la metodología. La sección 5.2 detalla las actividades previas a la recolección de datos, y describe los modelos de evaluación de tecnologías aplicados TAM y MEM. La sección 5.3 presenta el proceso de evaluación que permite la recolección de datos en el estudio de caso. La sección 5.4 evalúa los resultados obtenidos en la sección 5.3 mediante pruebas, estadística descriptiva y boxplots. En la sección 5.5 se reportan los resultados de la evaluación del estudio de caso. La sección 5.6 presenta la discusión de los resultados obtenidos. En la sección 5.7 describe las amenazas a la validez y finalmente, en la sección 5.8 se presentan las conclusiones de la evaluación del estudio de caso.

5.1. Diseñar y planificar el estudio de caso

En esta sección, se presenta el estudio de caso que se lleva a cabo para validar empíricamente la utilidad percibida por los usuarios sobre el DSL (Monitor-IoT).

Un estudio de caso es descrito por Besnard et al. (2018); Robson (2002); Yin (2003), cómo un método empírico o estrategia de investigación para la recopilación de información de fenómenos contemporáneos en su contexto, a través de la selección limitada de entidades (personas, grupos, organizaciones) para extraer fuentes de evidencia.

Otra definición propuesta por Wohlin (2007) afirma que un estudio de caso es una investigación empírica, similar a un experimento, en la cual la asignación de

tratamiento a sujetos no puede estar basada en la aleatoriedad, pero emerge desde las características de los sujetos u objetos en sí mismo.

I. Diseño

El estudio de caso de este trabajo de titulación fue diseñado de acuerdo al proceso experimental propuesto por Wohlin (2007), y en este caso será usada para predecir la probabilidad de aceptación de la actividad del diseño de sistemas basado en las percepciones de los usuarios.

De acuerdo al paradigma *Goal-Question Metric* (GQM) propuesto por Basili et al. (1994) la meta de este experimento ha sido definida de la siguiente manera:

- *Evaluar*: la fase de representación de una arquitectura para IoT a través del DSL (Monitor-IoT).
- *Con el propósito de*: evaluar el DSL propuesto con respecto a su eficacia percibida.
- *Desde el punto de vista del*: investigador.
- *En el contexto de*: un grupo de estudiantes de último año de Ingeniería en Sistemas de la Universidad de Cuenca y del Azuay; quienes pueden ser considerados como parte del experimento por estar próximos a culminar la carrera, ya que además bajo ciertos criterios estudiantes desarrolladores de último año están en la capacidad de evaluar la validez de un software al nivel de profesionales desarrolladores (B. A. Kitchenham et al., 2002).

De tal manera que se requiere plantear preguntas de investigación para comprobar si a través de un DSL el proceso de plantear arquitecturas para IoT resulta más fácil, útil y se pretende usarlo en un futuro; en este trabajo de titulación se plantean las siguientes preguntas:

RQ1: ¿Monitor-IoT es percibido como fácil de usar y útil? De ser así, ¿las percepciones de los usuarios son el resultado de su rendimiento cuando utilizan el DSL para configurar arquitecturas de monitoreo para el internet de las cosas?

RQ2: ¿Existe una intención de uso de Monitor-IoT en el futuro? De ser así, ¿tales intenciones de uso es el resultado de las percepciones de los participantes?

Estas preguntas de investigación pueden ser evaluadas a través de la prueba de varias hipótesis. En particular, la primera pregunta de investigación puede ser estudiada mediante las siguientes hipótesis:

$H1_0$: El DSL se percibe como difícil de usar, $H1_0 = \neg H1_1$.

$H2_0$: El DSL no se percibe como útil, $H2_0 = \neg H2_1$

$H4_0$: La facilidad de uso percibida no puede verse determinada por la eficiencia, $H4_0=\neg H4_1$.

$H5_0$: La percepción de la utilidad no está determinada por la efectividad. $H5_0=\neg H5_1$.

Por otra parte, la segunda pregunta de investigación puede ser estudiada a través de la formulación de las siguientes hipótesis:

$H3_0$: No existe intención de utilizar este DSL en el futuro $H3_0=\neg H3_1$.

$H6_0$: La utilidad percibida no es determinada por la facilidad de uso percibida $H6_0=\neg H6_1$.

$H7_0$: La intención de uso no es determinada por la facilidad de uso percibida $H7_0=\neg H7_1$.

$H8_0$: La intención de uso no está determinada por la utilidad percibida. $H8_0=\neg H8_1$.

II. Planificación

El proceso de realización de un estudio de caso se considera una investigación de tipo flexible, donde una buena planificación de un estudio de caso es fundamental para su éxito (Runeson & Höst, 2009).

En la planificación del estudio de caso, el ***primer paso*** es seleccionar el contexto. Para esta evaluación el contexto está determinado por los estudiantes que son los participantes que evalúan el estudio de caso como es el DSL Monitor-IoT que será la herramienta que usaran y se ha descrito cada uno de sus componentes a más detalle en el capítulo 4.

Por tanto, la investigación se caracteriza como un estudio de caso único y holístico aplicando la clasificación de (Yin, 2003), y la unidad de análisis se presenta en la Figura 5.1.

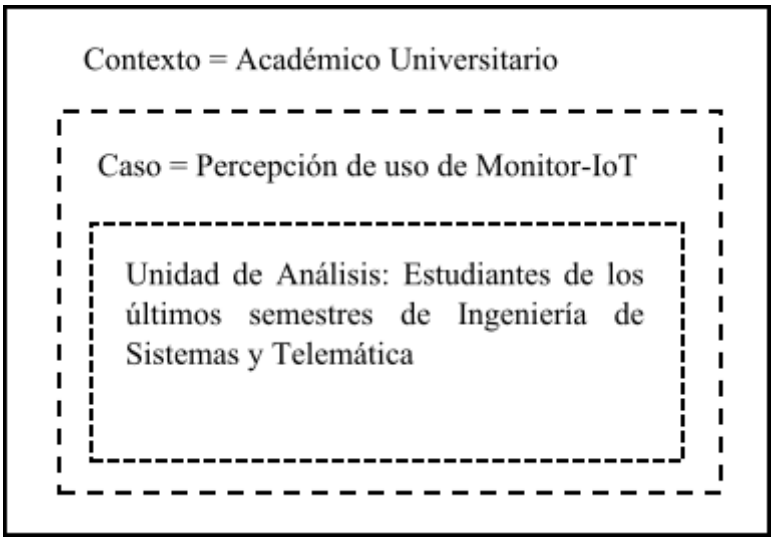


Figura 5.1. Estudio de caso único y holístico

El objetivo de Monitor-IoT es brindar la facilidad de instanciar arquitecturas de monitoreo para IoT mediante la actividad de seleccionar, agregar y configurar componentes usando un editor gráfico. El resultado de esta actividad es un DSL que representa la arquitectura y que sirve como entrada para desplegar la Infraestructura de monitoreo MIoT2InMon descrito en el capítulo 4.

En el uso de Monitor-IoT, los participantes, actúan asumiendo el rol de arquitectos de software, el cual incluye las siguientes tareas: (i) entender el dominio de la arquitectura para IoT que se debe desarrollar en el DSL (ver Anexo C.2), (ii) identificar cada componente de la arquitectura, (iii) seleccionar los componentes relacionados con la arquitectura en el editor gráfico del DSL, y (iv) generar el nuevo DSL.

Finalmente, 17 participantes fueron seleccionados, todos del último año de la carrera de la carrera de Ingeniería de Sistemas de la Universidad de Cuenca y de la carrera de Telemática de la Universidad del Azuay. Los estudiantes tienen un buen conocimiento en temas relacionados con arquitectura de software.

El ***segundo paso*** de la planificación es definir las tareas del estudio de caso. La principal tarea consiste en la resolución de un ejercicio, donde se plantea una arquitectura para IoT, y el participante debe crear un DSL de la arquitectura.

La tarea principal del ejercicio consiste en seleccionar correctamente cada componente para construir el DSL, para ello se definieron 7 categorías de los componentes entre estos sistemas, entidades físicas, nodos de computación, recursos, redes, enlaces, y propiedades. El participante debe elegir el componente correspondiente del editor gráfico en relación a la arquitectura y debe colocar el nombre y sus enlaces de comunicación.

El ***tercer paso*** de la planificación consiste en presentar las variables dependientes de interés basadas en la percepción, de acuerdo al MEM, las cuales fueron usadas para evaluar el DSL propuesta en la práctica. La Tabla 5.1 muestra las variables las mismas que son medidas usando un cuestionario con una escala de Likert con un conjunto de 9 preguntas cerradas (3 para facilidad de uso percibida, 3 para utilidad percibida y 3 para intención de uso futura).

El valor agregado para cada variable subjetiva fue calculado como la media aritmética de las respuestas a las preguntas asociadas con cada variable subjetiva dependiente.

VARIABLE	DESCRIPCIÓN
Facilidad de Uso Percibida (PEOU)	El grado en el cual los participantes creen que al aprender y usar el DSL propuesto estarán

	libres de esfuerzo.
Utilidad Percibida (PU)	El grado en el cual los participantes creen que usando el DSL propuesto se incrementará su rendimiento.
Intención de Uso (ITU)	El grado en el cual los participantes piensan usar el DSL en el futuro.

Tabla 5.1. Variables dependientes basadas en la percepción.

En la Tabla 5.2 se muestran las variables basadas en el rendimiento de interés y la función de medición, usadas para determinar los valores de las constantes necesarias para las evaluaciones del rendimiento.

VARIABLE	DESCRIPCIÓN
Efectividad	$Efectividad = \frac{\sum_{i=1}^{m} calificacion\_tarea_i}{n}$
Eficiencia	$Eficiencia = \sum_{i=1}^{m} tiempo\_ejecutando\_tarea_i$

Tabla 5.2. Variables dependientes basadas en el rendimiento.

El ***cuarto paso*** de la planificación es definir el material del estudio de caso, el cual está compuesto de documentación con los conceptos, descripción de la herramienta, componentes y la representación de la arquitectura para IoT y el cuestionario para medir la percepción del usuario una vez que se ha realizado el experimento.

La documentación en su totalidad se incluye en el Apéndice C de este trabajo de titulación, de igual forma los recursos se presentan en la página web *Monitor-IoT-Arc*, y se detalla a continuación:

1. *Presentación*: describe conceptos, herramienta, componentes del DSL, descripción e ilustración de un caso de entrenamiento donde se presenta la arquitectura para IoT aplicada a un invernadero y un caso práctico aplicado a un ambiente de vida asistida.
2. *Editor Gráfico*: provee la herramienta Obeo Designer y las carpetas de configuración, además de un video descriptivo del procedimiento para configurar el DSL.
3. *Evaluación*: se brinda un documento con la descripción e ilustración de una arquitectura para IoT aplicado a un ambiente de vida asistida. Se solicitó a los participantes que escriban la hora exacta de inicio y fin para desarrollar el DSL,

así como también adjuntar el DSL final. Para este paso cada estudiante reenvió el archivo con la información solicitada mediante correo electrónico.

4. *Formulario*: consta de preguntas cerradas para analizar las variables subjetivas y algunas preguntas abiertas para permitir a los participantes expresar su opinión sobre la arquitectura.

Todos los recursos, la evaluación y recopilación de información se realiza de forma virtual a través de la plataforma Zoom, Google forms y correo electrónico. En general todos los documentos y la información se crean en el idioma español, ya que este es el idioma nativo de los participantes.

5.2. Preparar la recolección de datos

La recolección de datos del estudio de caso está basada en el Método de Evaluación de Modelos (*MEM - Method Evaluation Model*)(Moody, 2001) y el modelo de evaluación de tecnologías (TAM)(Davis, 1985).

Por un lado, TAM propuesto por Davis (1985), es un modelo teórico ampliamente usado desde la perspectiva del uso en general de un sistema. Utiliza dos adaptadores potenciales, la facilidad de uso percibida y la utilidad percibida de la tecnología, estas actitudes influencian las intenciones y de ahí el comportamiento. En este modelo, el uso es modelado como una función directa de la intención (ver la Figura 5.2).

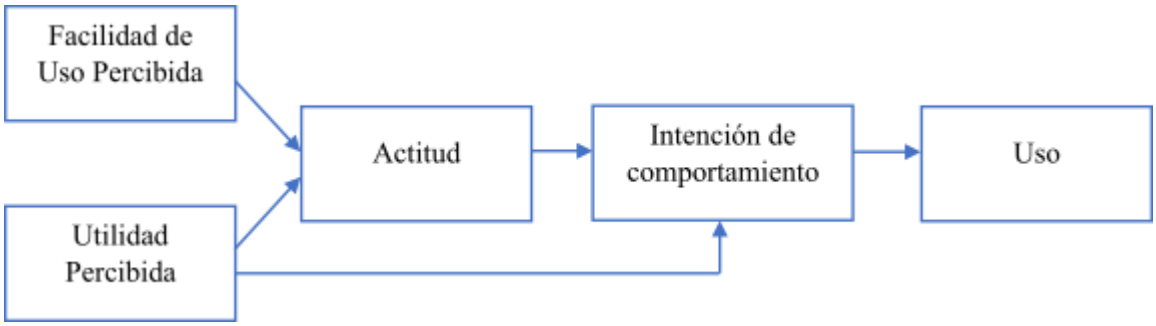


Figura 5.2. Modelo de aceptación de Tecnología - TAM simplificado (Davis, 1985). Fuente: Elaboración propia.

El significado de cada constructor de TAM es la facilidad de uso percibida, la utilidad percibida, la actitud, la intención de comportamiento y el uso.

1) La *facilidad de uso percibida (PEOU - Perceived Ease Of Use)* es el grado al cual los usuarios esperan que el sistema objetivo sea libre de esfuerzo.

2) La *utilidad percibida (PU - Perceived Usefulness)* indica la probabilidad del usuario de que utilizando una aplicación específica podría incrementar su rendimiento laboral en un contexto organizacional.

3) La *actitud (A)* es el deseo del usuario para usar el sistema. Tanto PU como PEOU predicen la actitud hacia usar el sistema.

4) La *intención de comportamiento (IB - Intention of Behavior)* indica la medida de la resistencia a ejecutar un comportamiento específico. Tanto A y PU influencian al individuo de IB a usar el sistema.

5) Finalmente, el *uso,* indica si el usuario estaría dispuesto a usar el sistema actual.

Por otro lado, está el MEM, extiende el modelo de aceptación de tecnología (*TAM - Technology Acceptance Model*) (Davis, 1985), el cual ha sido validado empíricamente en numerosos estudios que demuestran su utilidad para analizar la facilidad de uso percibida (PEOU - *Perceived Ease Of Use*), utilidad percibida (PU - *Perceived Usefulness*) e intención de uso (ITU - *Intention to Use*) de los participantes aplicando un método para predecir su posible aceptación.

Para utilizar el MEM en la evaluación del uso del DSL propuesto (Monitor-IoT) es necesario operacionalizar este modelo teórico para su uso en este tipo específico de DSL. Esta operacionalización consiste en definir los constructores del modelo en función de las variables relevantes para Monitor-IoT y redefinir los ítems del cuestionario que permitirán medir las variables de percepción.

Los constructos del MEM son: la eficacia actual, la eficacia percibida y el uso actual. La Figura 5.3 ilustra las diferentes partes de este método, además visualiza los constructores del modelo y las relaciones causales a lo largo de los constructores del mismo.

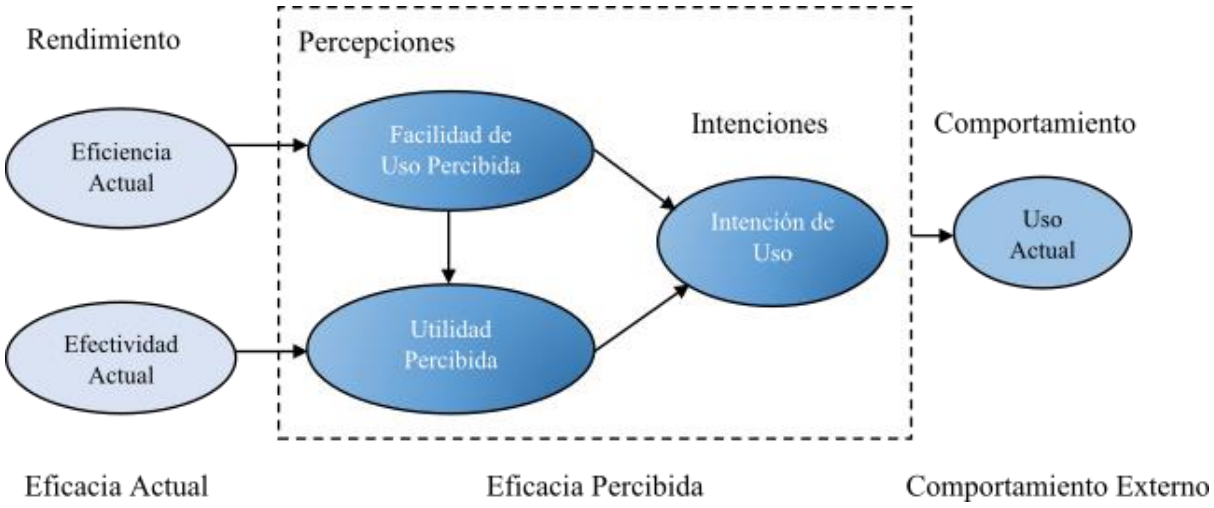


Figura 5.3. Método de Evaluación del Modelo – MEM (Moody, 2001). Fuente: Elaboración propia.

1) *La eficacia actual,* contiene dos variables basadas en el rendimiento del usuario, la eficiencia actual y la efectividad actual.

La *eficiencia actual* es el esfuerzo requerido para aplicar un método, mientras que la *efectividad actual* es el grado en el cual un método consigue sus objetivos.

A manera de ejemplo, P Cedillo (2016) define un método de monitorización de servicios en Cloud Computing como "una colección de reglas y procedimientos designados para asistir a la gente al momento de ejecutar una acción particular" orientada a ambientes en la nube, mediante modelos en tiempo de ejecución.

2) La *eficacia percibida* presenta variables basadas en la percepción, la facilidad de uso percibida, la utilidad percibida y la intención de uso.

   La *facilidad de uso percibida* es el grado en el cual una persona cree que usando un método en particular puede estar libre de esfuerzo.

   La *utilidad percibida* es el grado en el cual una persona cree que usando un método particular podría mejorar su rendimiento en el trabajo.

   La *intención de uso* es el modo en el que una persona intenta usar un método particular. Las relaciones causales sugeridas que perciben la facilidad de uso y la utilidad percibida directamente afectan la intención de usar una nueva tecnología.

3) El *uso actual* representa una variable basada en el comportamiento, definida como la intención de utilizar una nueva tecnología en la práctica. De acuerdo con la relación causal hipotética, el uso actual debe estar determinado por la intención del uso.

I. Adaptando el MEM para evaluar el estudio de caso Monitor-IoT

Para adaptar el MEM mediante el uso del DSL (Monitor-IoT) se requiere una secuencia de actividades, empezando por definir los objetivos específicos de Monitor-IoT. Seguido por, describir los constructores generales del MEM, los cuales pueden ser instanciados en variables dependientes concretas basadas en estos objetivos.

La evaluación de la eficacia en Monitor-IoT involucra la medición del esfuerzo requerido para crear un DSL dado una arquitectura para IoT de un escenario específico usando el DSL Monitor-IoT la arquitectura dado un escenario específico y la calidad de los resultados en la creación del DSL empleando la arquitectura.

El esfuerzo requerido para entender y/o aplicar Monitor-IoT (eficiencia actual) en la creación de un DSL pueden ser medidos mediante el tiempo o el esfuerzo cognitivo.

La calidad del resultado del DSL (efectividad actual) puede ser medida evaluando los resultados de la creación de un DSL haciendo uso de los componentes del editor gráfico.

Dichas variables de rendimiento se emplean para medir la eficiencia y la efectividad de los usuarios utilizando Monitor-IoT para crear un DSL dada una arquitectura para IoT de un escenario definido y serán evaluadas de la siguiente manera:

Efectividad actual

Esta variable se evalúa según el grado de completitud o representación de componentes, es decir si todos los componentes se configuraron (seleccionado y relacionado) correctamente se asigna un puntaje.

- 1, cuando los componentes han sido configurados correctamente.
- 0.5, cuando los componentes han sido configurados parcialmente.
- 0, cuando los componentes han sido configurados incorrectamente.

La efectividad total se evaluará con la ecuación 5.1, donde n representa la cantidad de componentes dentro del ejercicio.

$$Efectividad = \frac{\sum_{i=1}^{m} calificacion\_tarea_i}{n}$$

Ecuación 5.1

Eficiencia actual

Se calcula sumando los tiempos empleados en la realización de la tarea ejecutada por los participantes, para ello se usa la ecuación 5.2.

$$Eficiencia = \sum_{i=1}^{m} tiempo\_ejecutando\_tarea_i$$

Ecuación 5.2

Con el objetivo de medir las variables basadas en percepción, se ha adaptado un instrumento de medición utilizado en el MEM. La Figura 5.3 muestra como el MEM ha sido operacionalizado para evaluar el uso del DSL cuando se plantean arquitecturas para IoT.

Para medir cada uno de los tres constructores (facilidad de uso percibida, utilidad percibida e intención de uso), se han definido conjuntos de preguntas basados en los ítems mostrados en la Tabla 5.1. La Figura 5.4 muestra el modelo teórico propuesto para evaluar el uso del DSL.

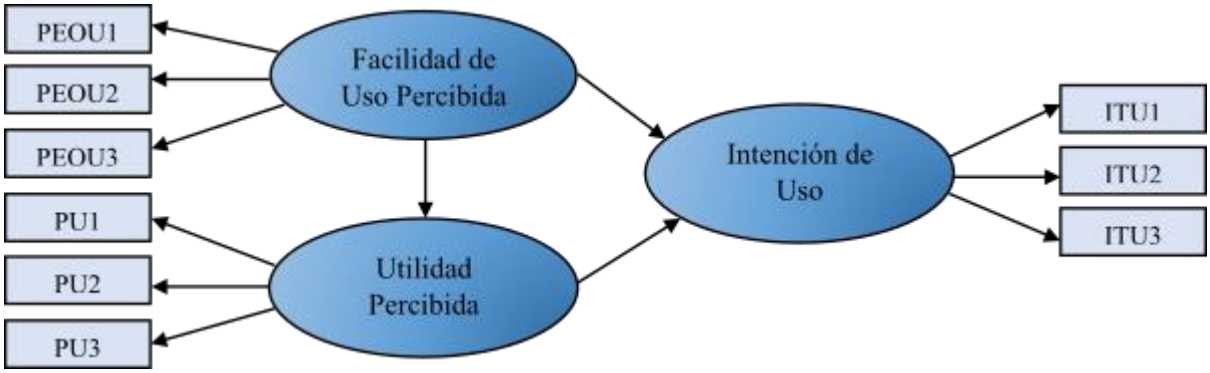


Figura 5.4. Distribución de preguntas del cuestionario aplicado al estudio de caso. Fuente: (P Cedillo, 2016).

Básicamente, se han usado medidas basadas en el rendimiento como factores que influencian las variables basadas en la percepción, cada una de estas influencias serán probadas mediante la evaluación de diversas hipótesis, como se observa en la Figura 5.5.

De acuerdo a nuestra adaptación de MEM, la probabilidad de que el DSL (Monitor-IoT) sea aceptado en la práctica puede ser predicho probando las siguientes hipótesis:

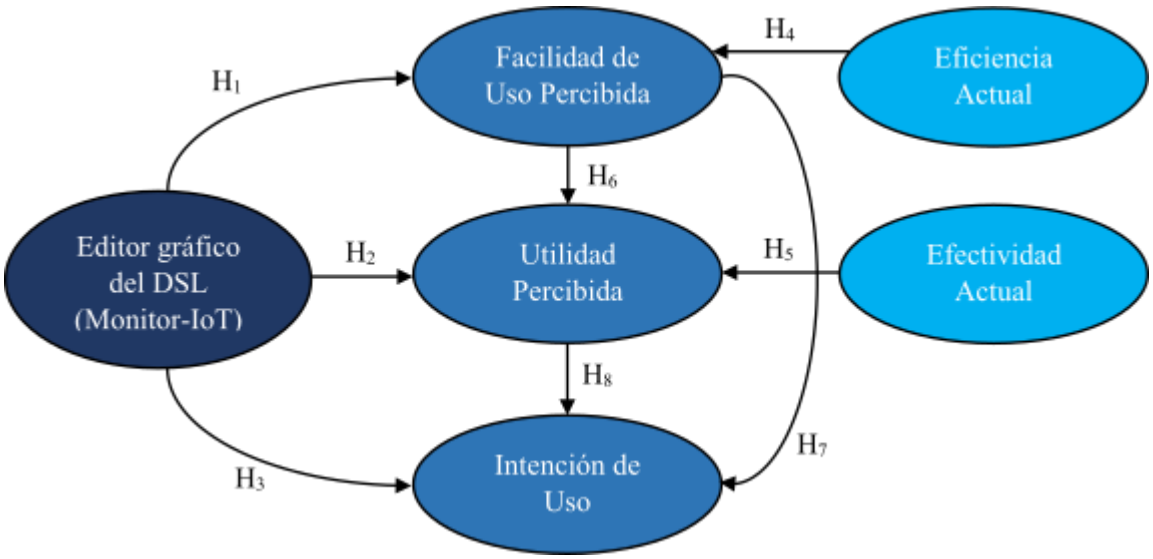


Figura 5.5. Modelo teórico para la evaluación del estudio de caso. Fuente: (P Cedillo, 2016).

$H1_0$: El DSL (Monitor-IoT) se percibe como difícil de usar, $H1_0=\neg H1_1$.

$H2_0$: El DSL (Monitor-IoT) no se percibe como útil, $H2_0=\neg H2_1$

$H3_0$: No existe intención de utilizar el DSL (Monitor-IoT) en el futuro $H3_0=\neg H3_1$.

A continuación, se presentan las hipótesis que muestran una relación directa entre el uso del DSL (Monitor-IoT) y el rendimiento, percepciones e intenciones de los

usuarios. El modelo de evaluación además propone un número de hipótesis que indican una relación causal entre las variables dependientes:

$H4_0$: La facilidad de uso percibida no puede verse determinada por la eficiencia, $H4_0=\neg H4_1$. La razón de esta hipótesis es que la eficiencia representa una medida basada en el rendimiento de la eficiencia actual; mientras que, la facilidad de uso percibida representa una medida basada en la percepción. De acuerdo a MEM la eficiencia mide el esfuerzo requerido en el uso del DSL, el cual debería determinar las percepciones del esfuerzo requerido.

$H5_0$: La percepción de la utilidad no está determinada por la efectividad. $H5_0=\neg H5_1$. La razón de esta hipótesis es que la efectividad representa una medida basada en el rendimiento; mientras que, la utilidad percibida representa una medida basada en la percepción de la efectividad. De acuerdo al MEM las percepciones de efectividad deberían estar determinadas por la efectividad actual.

$H6_0$: La utilidad percibida no es determinada por la facilidad de uso percibida $H6_0=\neg H6_1$. Esta hipótesis es tomada desde el TAM, en el cual la facilidad de uso percibida se encuentra que no tiene una influencia directa sobre la utilidad percibida.

$H7_0$: La intención de uso no es determinada por la facilidad de uso percibida $H7_0=\neg H7_1$. Esta hipótesis es tomada desde el TAM, en el cual se encuentra que la facilidad de uso percibida tiene influencia sobre la intención de uso.

$H8_0$: La intención de uso no está determinada por la utilidad percibida. $H8_0=\neg H8_1$. Esta hipótesis es tomada del TAM, en la cual se encontró que la utilidad percibida tiene una influencia directa sobre la intención de uso.

La Tabla 5.3 muestra las preguntas definidas para medir las variables basadas en la percepción, las cuales fueron combinadas en un cuestionario con 9 preguntas formuladas utilizando una escala de 5 puntos de Likert, con el formato de preguntas opuestas.

La facilidad de uso percibida (PEOU) se mide utilizando tres preguntas del cuestionario, la utilidad percibida (PU) se mide con tres preguntas y para la intención de uso futuro (ITU) se utilizó tres preguntas. El cuestionario se puede encontrar en el Anexo C.2.

PREGUNTA	DECLARACIÓN POSITIVA (5 PUNTOS)
PEOU1	El editor gráfico del DSL para diseñar arquitecturas de monitoreo para IoT es intuitivo y fácil de entender.
PEOU2	La notación gráfica (representación de los componentes) utilizada por el DSL es intuitiva y fácil de entender.
PEOU3	El editor gráfico del DSL para diseñar arquitecturas de monitoreo

	para IoT es fácil de usar.
PU1	El uso del DSL podría reducir el tiempo y esfuerzo requerido en el desarrollo, despliegue y mantenimiento de sistemas de IoT.
PU2	Considera que el DSL sería de utilidad para apoyar las actividades de desarrollo, despliegue y mantenimiento de sistemas de IoT.
PU3	El uso del DSL Monitor-IoT podría mejorar su rendimiento en el desarrollo, despliegue y mantenimiento de sistemas de IoT.
ITU1	En caso de necesitar diseñar arquitecturas de monitoreo para IoT en el futuro, consideraría el DSL Monitor-IoT.
ITU2	En caso de necesitar diseñar arquitecturas de monitoreo para IoT en el futuro, utilizará el DSL Monitor-IoT.
ITU3	Recomendaría el uso de este DSL para diseñar arquitecturas de monitoreo para IoT.

Tabla 5.3. Cuestionario para medir las variables de percepción.

5.3. Recolección los datos

La modalidad del estudio de caso se realizó de forma virtual en la plataforma Zoom; como primer paso, cada participante debe descargar la herramienta *Obeo Designer* donde se desarrolló el editor gráfico del DSL (Monitor-IoT).

Para dar lugar a la sesión del estudio de caso se realiza un entrenamiento de 30 minutos a fin de presentar los conceptos relacionados al DSL (Monitor-IoT), sus componentes, la forma de instanciar elementos y cómo enlazarlos entre sí, todo esto a través de un ejemplo demostrativo.

El entrenamiento incluyó un gráfico representativo de una arquitectura para IoT de un invernadero y se replicó dicha arquitectura usando el DSL (Monitor-IoT). El estudio de caso se realizó con un grupo de 17 estudiantes de Ingeniería de Sistemas, 4 de ellos de la Universidad de Cuenca y 13 alumnos de la Universidad del Azuay.

Los participantes encargados de realizar el estudio de caso clarificaron cualquier duda o pregunta a lo largo de la sesión y subsiguiente se ejecutó la tarea del estudio de caso que consta de ejercicio práctico compuesto por una arquitectura de IoT en ambientes de vida que debe ser replicada en el DSL (Monitor-IoT). La arquitectura de la tarea y el resultado esperado se presenta en el *Apéndice C.*

En resumen, para la recolección de datos en el estudio de caso se aplica una de las categorías de métodos propuesto por Lethbridge et al. (2005); el método es de

segundo grado del tipo indirecto porque la información se recopila sin interactuar de forma presencial con el grupo de estudiantes a través del documento de especificación y la encuesta en la modalidad en línea, y adicionalmente, porque los estudiantes hacen uso de una herramienta para este estudio de caso utilizan Obeo Designer con el paquete de Sirius para llevar a cabo el proceso de representación y configuración de la arquitectura para IoT en el contexto de AAL.

5.4. Analizar e interpretar los datos recopilados

Para el análisis de los resultados, se usaron pruebas, estadística descriptiva y boxplots para analizar los datos recopilados. Los datos fueron analizados de acuerdo a las hipótesis establecidas por medio de los criterios de aceptación o rechazo de hipótesis, para lo cual se utiliza los niveles de significancia sugeridos por Moody (2001), los cuales se muestra en la Tabla 5.4 y los resultados fueron obtenidos usando la herramienta SPSS con un $\alpha = 0.05$.

VALOR DE SIGNIFICANCIA	RANGO
No significativo	$p > 0.1$
Baja significancia	$p < 0.1$
Media significancia	$p < 0.05$
Alta significancia	$p < 0.01$
Muy alta significancia	$p < 0.001$

Tabla 5.4. Niveles de significancia sugeridos por Moody (2001).

Análisis de las percepciones de usuario

La Figura 5.6 muestra los diagramas de caja para cada variable de percepción en donde se observa que la media de cada variable es mayor que 3, que representa al valor neutro de la escala de Likert

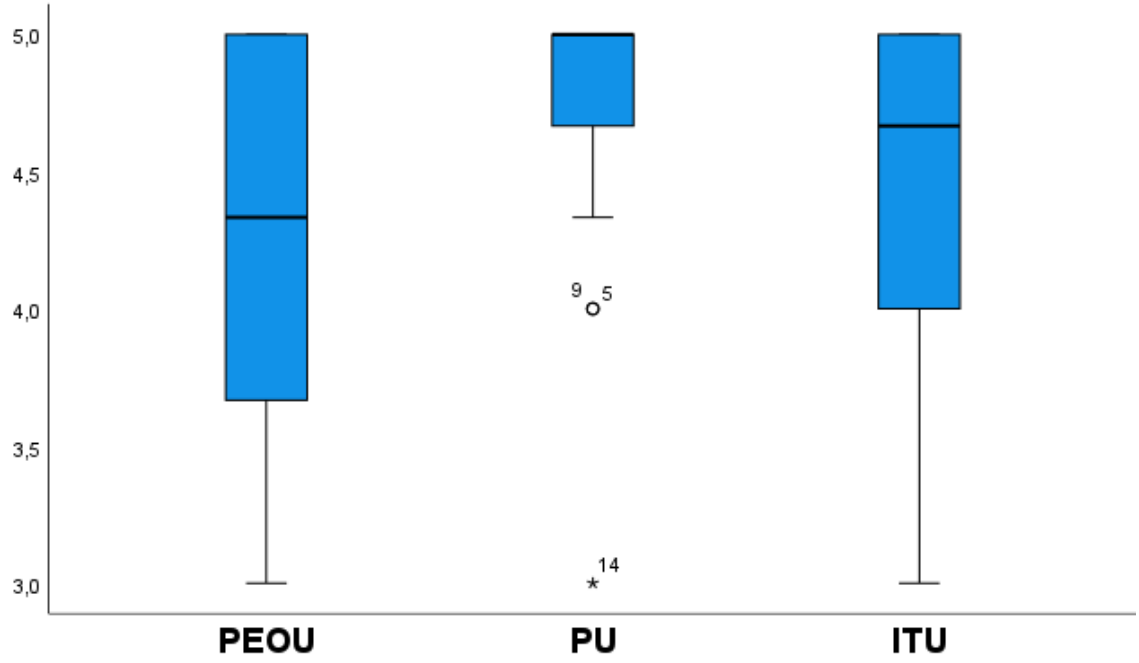


Figura 5.6. Diagrama de cajas para las variables PEOU (Facilidad de Uso Percibida), PU (Utilidad Percibida) e ITU (Intención de Uso) del estudio de caso. Fuente: Elaboración propia.

Los diagramas de caja muestran algunos puntos de datos anómalos (participantes con id=5, 9 y 14), los cuales corresponden a participantes que se conectaron minutos después de explicar el alcance que brinda el DSL (Monitor-IoT), sin embargo, no son descartados porque culminaron el DSL de forma satisfactoria.

Luego, se ha aplicado la prueba de Shapiro-Wilk porque la muestra utilizada es menor que 50 participantes; en esta evaluación participaron 17 estudiantes. Con esta prueba se puede verificar si los datos se encuentran distribuidos normalmente para determinar el test que podría usarse para chequear las hipótesis H1, H2 y H3.

La Tabla 5.5 muestra los resultados de la prueba Shapiro-Wilk para las variables estudiadas. Se han aplicado pruebas para verificar las hipótesis comparando si la media de las respuestas a las preguntas relacionadas con una variable dada, fueron significativamente más altas que el valor neutro de la escala Likert.

VAR	MIN	MAX	MEDIA	STD. DEV	1-T. p-value	SHAPIRO - WILK test p-value
PEOU	3	5	4.27	0.65	0.000 < 0.05	0.088 > 0.05
PU	3	5	4.67	0.55	0.000 < 0.05	0.000 < 0.05
ITU	3	5	4.43	0.64	0.000 < 0.05	0.009 < 0.05

Tabla 5.5. Prueba de Shapiro-Wilk para las variables subjetivas.

De los resultados presentados en las Tabla 5.5, se puede observar en la columna Shapiro-Wilk, el valor de la variable PEOU tiene una distribución normal ($p>0.05$) por tanto se ha probado la hipótesis aplicando una prueba paramétrica como es el t-test one-tailed, y para las variables PU e ITU, que no se tiene una distribución normal ($p<0.05$) se ha aplicado una prueba no paramétrica como es el test Wilcoxon one-tailed one-sample con un valor de prueba igual a 3, ya que este valor corresponde al valor neutral de la escala Likert del cuestionario.

Los resultados de los test aplicados se presentan en la columna 1-T de la Tabla 5.5, donde la significancia para las tres variables PEOU, PU, ITU es $p<0.05$, donde 0.05 es el nivel de significancia referencial para estas pruebas, por lo tanto, esto nos permite rechazar las hipótesis nulas $H1_0$, $H2_0$ y $H3_0$ lo que significa que el DSL (Monitor-IoT) es percibido como fácil de usar, útil y que los participantes consideran usarlo en el futuro cuando requieran diseñar arquitecturas para IoT.

Análisis del Rendimiento del Usuario

Se realizó la medición de la efectividad y la eficiencia de los participantes cuando utilizan el DSL(Monitor-IoT) en la práctica. La Tabla 5.6 muestra los valores de estadística descriptiva para las variables de efectividad y eficiencia.
Los participantes tuvieron un promedio del 95 % de efectividad al replicar el ejercicio del estudio de caso, lo que indica que la mayoría de los participantes fueron capaces de instanciar y configurar los componentes planteados correctamente. La eficiencia se calculó como el esfuerzo requerido (en minutos) para aplicar el método de Moody (2001).

Los participantes tuvieron una eficiencia que fue de 24 a 65 minutos. Es necesario considerar que la eficiencia puede variar dependiendo de varios factores, como, por ejemplo, la experiencia de los sujetos y el uso de este tipo de herramientas. A pesar de esas limitaciones, el propósito de este experimento fue probar si las percepciones de los usuarios fueron resultado de su rendimiento. Estos resultados además proporcionan información para entender cómo los participantes han usado el editor gráfico.

VARIABLE	MIN	MAX	MEDIA	STD. DEV
Efectividad	0.86	1	0.95	0.05
Eficiencia	24	65	43.94	10.70

Tabla 5.6. Estadística descriptiva para variables basadas en el Rendimiento del Usuario.

Análisis de Relaciones Causales

En esta sección se realiza la validación de la parte estructural de MEM en términos de las relaciones causales que hay entre sus constructores, exceptuando el Uso Actual. Para ello, se usa un análisis de regresión para evaluar la operacionalización

de MEM, ya que las hipótesis son relaciones causales entre variables continuas. Para efectuar este análisis, se usan los niveles de significancia dados por Moody (2001) que fueron presentados en la Tabla 5.4.

➢ Eficiencia vs Facilidad de Uso Percibida

La hipótesis H4 ha sido probada para verificar si las percepciones de la Facilidad de Uso Percibida (PEOU) son determinadas por la eficiencia al usar el editor gráfico. Para ejecutar este análisis se ha construido un modelo de regresión simple en el cual la eficiencia fue usada como la variable independiente (predictor) y PEOU como la variable dependiente (predicho). La ecuación de regresión resultante del análisis es la siguiente:

$$PEOU = 4{,}348 + (-0{,}002) * Eficiencia$$

REG. ELEMENTO	COEF (b)	STD. E	STD. COEF	t	SIG (p)	R	R^2
Constante	4.348	0.706		-6.162	0.000		
Eficiencia	-0.002	0.016	-0.028	0.107	0.916 > 0.1	0.028	0.001

Tabla 5.7. Regresión Simple entre la Eficiencia Actual y la Facilidad de Uso Percibida.

El modelo de regresión no es significativo con p>0.1. El R^2 muestra que la variable eficiencia permite explicar solamente el 0.1% de la varianza en PEOU, indicando que la eficiencia actual de los participantes no influencia sus percepciones de facilidad de uso.

Estos resultados no nos permiten rechazar la hipótesis nula $H4_0$ y aceptar su hipótesis alternativa, esto confirma que la facilidad de uso percibida (PEOU) no está determinada por la Eficiencia.

➢ Efectividad vs Utilidad Percibida

La hipótesis H5 se ha definido con el objetivo de comprobar si las percepciones de la Utilidad Percibida (PU) están determinadas por la Efectividad de los participantes. De igual forma, se ha realizado la construcción de un modelo de regresión simple en donde la Efectividad es la variable independiente y PU la variable dependiente. La ecuación de regresión resultante del análisis es la siguiente:

$$PU = 6{,}468 + (-1{,}889) * Efectividad$$

REG. ELEMENTO	COEF (b)	STD. E	STD. COEF	t	SIG (p)	R	R^2
Constante	6.468	2.384		2.713	0.016		

Efectividad	-1.889	2.495	-0.192	-0.757	0.461 > 0.1	0.192	0.037

Tabla 5.8. Regresión Simple entre la Eficiencia Actual y la Facilidad de Uso Percibida.

El modelo de regresión no es significativo, con p>0.1. El R^2 muestra que la variable Efectividad permite explicar solamente el 3,7 % de la varianza de PU, lo cual indica que muy pocas de las percepciones con respecto a PU están determinadas por la efectividad de los participantes.

Estos resultados no nos permiten rechazar la hipótesis nula $H5_0$ y aceptar su hipótesis alternativa, lo que significa que la utilidad percibida (PU) no está determinada por la Efectividad de los participantes.

➢ Facilidad de Uso Percibida vs Utilidad Percibida

La hipótesis H6 se ha definido con el objetivo de comprobar si las percepciones de la Utilidad Percibida (PU) están determinadas por la Facilidad de Uso Percibida (PEOU). De igual forma, se ha realizado la construcción de un modelo de regresión simple en donde PEOU es la variable independiente y PU la variable dependiente. La ecuación de regresión resultante del análisis es la siguiente:

$$PU = 2{,}263 + 0{,}562 * PEOU$$

REG. ELEMENTO	COEF (b)	STD. E	STD. COEF	t	SIG (p)	R	R^2
Constante	2.263	0.716		0.625	0.006		
PEOU	0.562	0.166	0.659	3.395	0.004 < 0.01	0.659	0.434

Tabla 5.9. Regresión Simple entre la Facilidad de Uso Percibida y la Utilidad Percibida.

El modelo de regresión es altamente significativo, con p<0.01. El R^2 muestra que la variable PEOU permite explicar el 43.4 % de la varianza de PU, lo cual indica que ciertas percepciones con respecto a PU están determinadas por PEOU.

Estos resultados nos permiten rechazar la hipótesis nula $H6_0$ y aceptar su hipótesis alternativa, lo que significa que la utilidad percibida (PU) sí está determinada por la facilidad de uso percibida (PEOU) de los participantes.

➢ Intención de Uso vs Facilidad de Uso Percibida

La hipótesis H7 se ha definido con el objetivo de comprobar si las percepciones de la Intención de Uso (ITU) están determinadas por la Facilidad de Uso Percibida (PEOU). Para el análisis, se ha realizado la construcción de un modelo de regresión simple en donde PEOU es la variable independiente e ITU la variable dependiente. La ecuación de regresión resultante del análisis es la siguiente:

$$ITU = 1{,}529 + 0{,}620 * PEOU$$

REG. ELEMENTO	COEF (b)	STD. E	STD. COEF	t	SIG (p)	R	R^2
Constante	1.529	0.919		1.665	0.117		
Efectividad	0.620	0.205	0.615	3.019	0.009 < 0.01	0.615	0.378

Tabla 5.10. Regresión Simple entre la Facilidad de Uso Percibida y la Intención de Uso.

El modelo de regresión es altamente significativo, con p<0.01. El R^2 muestra que la variable PEOU permite explicar el 37,8 % de la varianza de ITU, lo cual indica que existe un alto grado de que ITU esté determinada por las percepciones de facilidad de uso (PEOU).

Estos resultados nos permiten rechazar la hipótesis nula $H7_0$ y aceptar su hipótesis alternativa, lo que significa que se ha encontrado que la intención de uso (ITU) sí está determinada por la facilidad de uso percibida (PEOU) de los participantes.

➢ Intención de Uso vs Utilidad Percibida

La hipótesis H8 se ha definido con el objetivo de comprobar si las percepciones de la Intención de Uso (ITU) están determinadas por la Utilidad Percibida (PU). Para el análisis, se ha realizado la construcción de un modelo de regresión simple en donde PU es la variable independiente e ITU la variable dependiente. La ecuación de regresión resultante del análisis es la siguiente:

REG. ELEMENTO	COEF (b)	STD. E	STD. COEF	t	SIG (p)	R	R^2
Constante	2.359	0.790		2.985	0.009		
Efectividad	0.521	0.177	0.606	2.948	0.01 < 0.05	0.606	0.367

Tabla 5.11. Regresión Simple entre la Intención de Uso y la Utilidad Percibida.

El modelo de regresión presenta una significancia media, con p<0.05. El R^2 muestra que la variable PU permite explicar el 36,7 % de la varianza de ITU, lo cual indica que existe un alto grado de que ITU esté determinada por las percepciones de utilidad (PU).

Estos resultados nos permiten rechazar la hipótesis nula $H8_0$ y aceptar su hipótesis alternativa, lo que significa que se ha encontrado que la intención de uso (ITU) sí está determinada por la utilidad percibida (PU) de los participantes.

5.5. Resultados de la evaluación

En esta sección se presenta un resumen de los resultados obtenidos tras la realización del experimento, con el objetivo de encontrar posibles semejanzas y

diferencias. La Tabla 5.12 muestra cada una de las variables con los valores obtenidos, su media y desviación estándar.

VARIABLE	ESTUDIANTES DE SISTEMAS Y TELEMÁTICA	
	MEDIA	STD. DEV
Efectividad	0.95	0.56
Eficiencia	43.94	10.70
Facilidad de uso percibida (PEOU)	4.27	0.65
Utilidad percibida (PU)	4.67	0.55
Intención de uso (ITU)	4.43	0.64

Tabla 5.12.Tabla de resumen de estadísticos descriptivos.

Los resultados globales nos han permitido concluir que el DSL (Monitor-IoT) ha mejorado el rendimiento de los participantes en casi la totalidad de las estadísticas analizadas.

Eficiencia

La Figura 5.7 presenta el diagrama de caja para la variable *Eficiencia* del estudio de caso. De este diagrama se puede concluir que, bajo las condiciones del experimento, el diseño de un DSL de arquitecturas para IoT usando el DSL (Monitor-IoT) va desde los 24 hasta los 65 minutos aproximadamente.

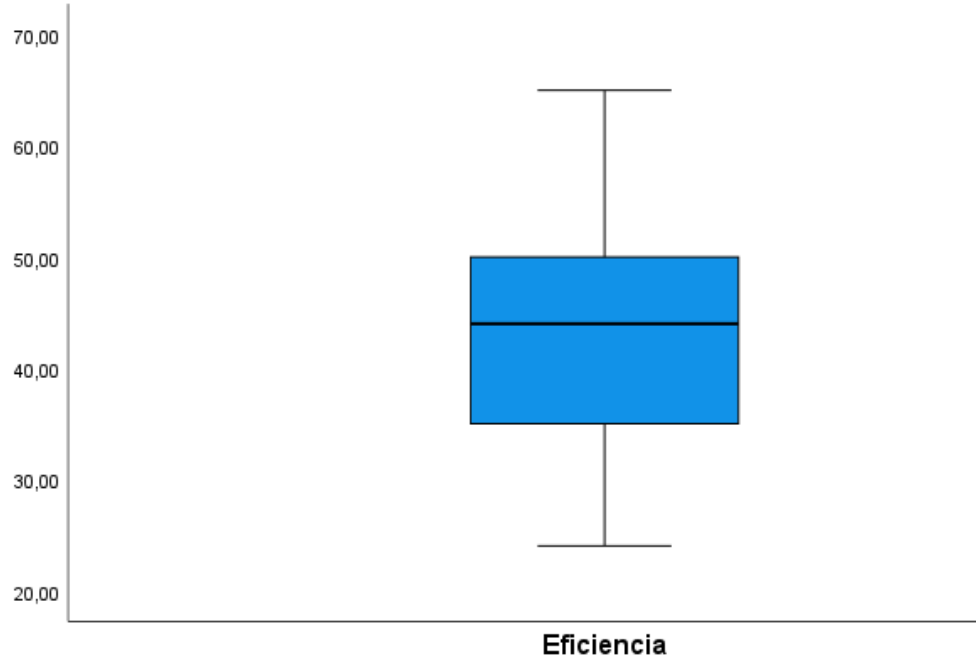


Figura 5.7. Diagrama de cajas para la variable Eficiencia en el estudio de caso realizado. Fuente: Elaboración propia.

Efectividad

La Figura 5.8 presenta el diagrama de caja para la variable *Efectividad* del estudio de caso. De este diagrama se puede concluir que los participantes han sido capaces de responder efectivamente a la tarea propuesta, replicando la mayoría de los componentes de manera correcta. En general, los participantes tuvieron alrededor de un 95 % de efectividad al realizar el ejercicio del estudio de caso.

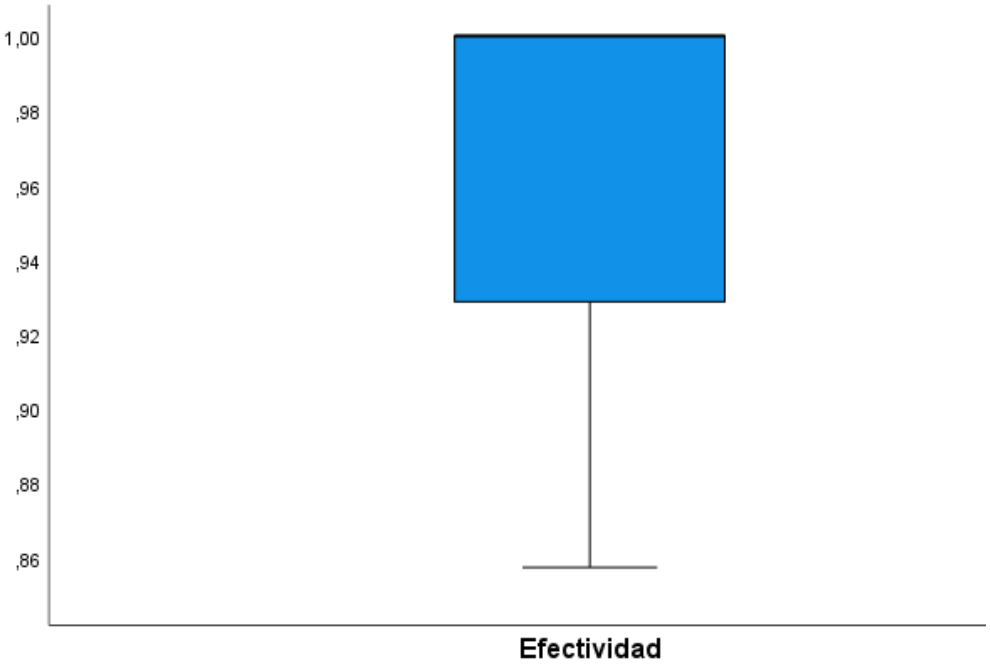


Figura 5.8. Diagrama de cajas para la variable Efectividad en el estudio de caso realizado. Fuente: Elaboración propia.

Facilidad de uso percibida (PEOU)

La Tabla 5.13 presenta un resumen de la media obtenida para la facilidad de uso percibida en el estudio de caso. El DSL (Monitor-IoT), en general, fue percibido como fácil de usar en relación con el valor de prueba v=3.

VARIABLE	ESTUDIANTES DE SISTEMAS Y TELEMÁTICA
Facilidad de Uso Percibida (PEOU)	4.27

Tabla 5.13. Tabla de resumen de la media de la Facilidad de Uso Percibida (PEOU).

Utilidad percibida (PU)

La Tabla 5.14 presenta un resumen de la media obtenida para para la utilidad percibida en el estudio de caso. El DSL (Monitor-IoT), en general, fue percibido como útil en relación con el valor de prueba v=3.

VARIABLE	ESTUDIANTES DE SISTEMAS Y TELEMÁTICA
Utilidad Percibida (PU)	4.67

Tabla 5.14. Tabla de resumen de la media de la Utilidad Percibida (PU).

Intención de uso (ITU)

La Tabla 5.15 presenta un resumen de la media obtenida para la intención de uso percibida en el estudio de caso. En general, los usuarios tienen la intención de utilizar el DSL (Monitor-IoT), cuando necesiten diseñar e implementar arquitecturas para IoT en AAL a un nivel alto de DSL en relación con el valor de prueba v=3, esto se confirma por las respuestas favorables recibidas en el cuestionario de percepción post-experimento.

VARIABLE	ESTUDIANTES DE SISTEMAS Y TELEMÁTICA
Intención de Uso (ITU)	4.43

Tabla 5.15. Tabla resumen de la media de la Intención de Uso (ITU).

Resumen de aceptación de las hipótesis

La Tabla 5.16 presenta un resumen de los resultados obtenidos luego de realizar el ejercicio del estudio de caso, donde se indica el número de participantes y las hipótesis que fueron aceptadas y rechazadas.

PARTICIPANTES	NÚMERO DE PARTICIPANTES	HIPÓTESIS ACEPTADAS	HIPÓTESIS NULAS RECHAZADAS
Estudiantes de los últimos años de Ingeniería de Sistemas y Telemática	17	$H1_1$, $H2_1$, $H3_1$, $H4_0$, $H5_0$, $H6_1$, $H7_1$, $H8_1$	$H1_0$, $H2_0$, $H3_0$, $H4_1$, $H5_1$, $H6_0$, $H7_0$, $H8_0$

Tabla 5.16. Tabla resumen de aceptación y rechazo de hipótesis del estudio de caso.

Considerando los resultados del estudio de caso, se puede concluir que, en todos los casos, el DSL (Monitor-IoT) es percibido como fácil de usar, útil y se tiene la intención de usarla en el futuro en caso de tener la necesidad de diseñar arquitectura de monitoreo para IoT.

Respecto a la efectividad, en el estudio de caso, se consiguió una media de alrededor del 95 %, lo cual significa que el diseño de un DSL mediante la selección y configuración de componentes en el editor gráfico del DSL (Monitor-IoT) se lleva a cabo de una manera correcta.

Por otro lado, la eficiencia de los participantes, en el estudio de caso, es aceptable con una media de 44 minutos para realizar el ejercicio. Sin embargo, los resultados de la efectividad y la eficiencia podrían ser aún más favorables a medida que aumente la experiencia y entendimiento de los usuarios en la utilización de los diferentes componentes del editor gráfico del DSL.

5.6. Discusión

En esta sección, se presentan las conclusiones globales obtenidas de cada pregunta de investigación:

RQ1: *¿Monitor-IoT es percibido como fácil de usar y útil? De ser así, ¿las percepciones de los usuarios son el resultado de su rendimiento cuando utilizan el editor gráfico del DSL para configurar arquitecturas de monitoreo para el internet de las cosas?*

Para responder a la primera parte de esta pregunta de investigación, es necesario referirse a las hipótesis H1 y H2. En el estudio de caso se rechazaron las hipótesis nulas H1 y H2 aceptando sus alternativas, esto significa que, para la mayoría de los participantes, el DSL (Monitor-IoT) fue percibido como útil y fácil de usar. Esto se ve reforzado por la efectividad alcanzada por los participantes al desarrollar el ejercicio del estudio de caso, la cual tuvo una media del 95.

Respecto a si las percepciones del usuario son el resultado de su rendimiento al utilizar el editor gráfico del DSL, se puede observar que en el estudio de caso la hipótesis nula H4 fue aceptada y se rechazó su alternativa, esto significa que la facilidad de uso percibida (PEOU) no es el resultado de la eficiencia de una gran cantidad de participantes, es decir, el tiempo que los participantes emplearon en resolver el ejercicio del estudio de caso no se relaciona con que el editor gráfico sea visto como fácil de usar.

Respecto a la comprobación de si la utilidad percibida es resultado de la efectividad de los participantes, en el estudio de caso se aceptó la hipótesis nula H5; esto significa que, de manera general, la utilidad percibida (PU) no está influenciada por la efectividad de los participantes al desarrollar el ejercicio.

Por último, en el estudio de caso, se rechazó la hipótesis nula H6 y se aceptó su alternativa, esto significa que, de forma general, la utilidad percibida (PU) sí está determinada por la facilidad de uso percibida (PEOU), es decir, los participantes consideran que, el percibir el editor gráfico como fácil de usar implica que también es útil.

RQ2: *¿Existe una intención de uso de Monitor-IoT en el futuro? De ser así, ¿tales intenciones de uso es el resultado de las percepciones de los participantes?*

Respecto a la intención de uso futura del DSL propuesto, se puede observar que se rechazó la hipótesis nula H3 y se aceptó su alternativa, esto significa que los participantes si tienen la intención de usar el DSL (Monitor-IoT) en el futuro cuando exista la necesidad de diseñar arquitecturas de monitoreo para IoT dentro de AAL.

Respecto a la incidencia de la facilidad de uso (PEOU) sobre la intención de uso (ITU), se observa que la hipótesis nula H7 fue rechazada y se aceptó su alternativa, esto significa que la intención de uso (ITU) sí está determinada por la facilidad de uso percibida (PEOU), es decir, los participantes consideran que el hecho de tener un DSL fácil de usar implica que esta pueda ser considerada para su uso en el futuro.

Finalmente, se puede observar que la hipótesis nula H8 fue rechazada y se aceptó su alternativa, esto significa que la intención de uso (ITU) también está determinada por la utilidad percibida (PU), es decir, los participantes consideran que, si el DSL resulta ser útil, éstos tendrían la intención de usarla en el futuro cuando sea necesario.

Los resultados globales del análisis de regresión son resumidos en la Figura 5.9, estos resultados constituyen una primera aproximación empírica de la evaluación del DSL propuesto. Como trabajo futuro, se presenta la necesidad de investigar sobre la influencia de otras variables basadas en el rendimiento y la percepción para predecir la aceptación del editor gráfico del DSL (Monitor-IoT) en la práctica.

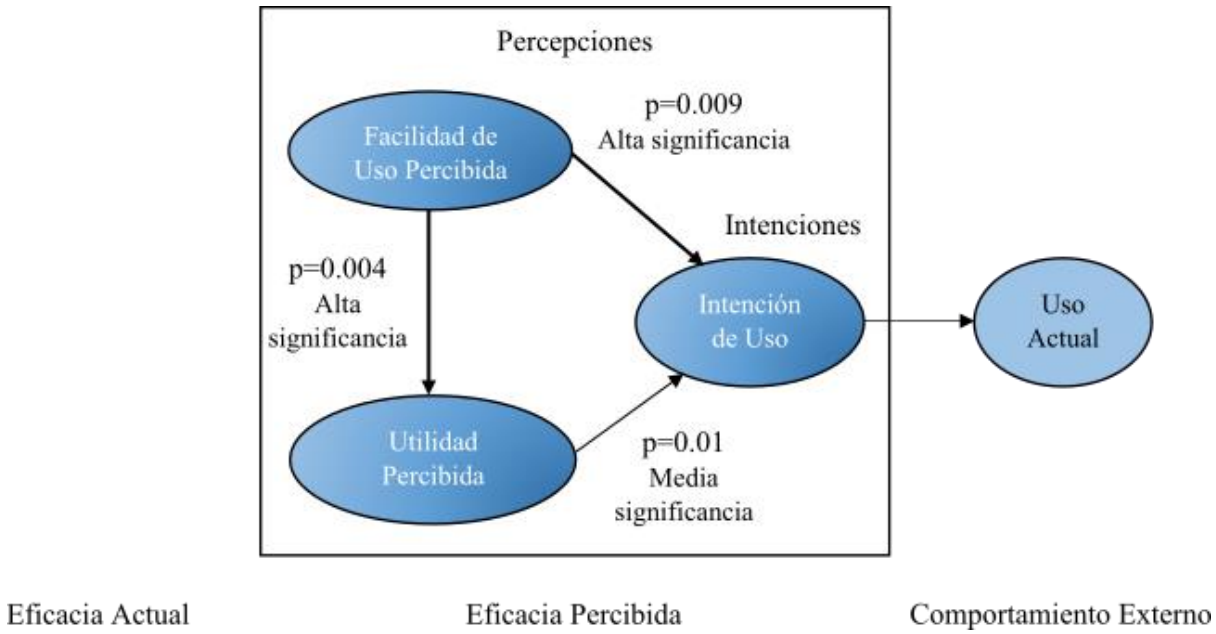


Figura 5.9. Conclusiones de la aplicación de MEM al DSL (Monitor-IoT) propuesto. Fuente: Elaboración propia.

5.7. Amenazas a la validez

A continuación, se explican los principales problemas que pueden poner en peligro la validez del estudio de caso, al considerar los cuatro tipos de amenazas que se proponen por Cook et al. (1979).

Validez interna

Las principales amenazas a la validez interna fueron: la experiencia de los participantes, los sesgos del autor, los sesgos relacionados a la forma de estructurar un DSL basado en el editor gráfico del DSL (Monitor-IoT) propuesto y la modalidad utilizada para llevar a cabo el experimento. Este tipo de amenaza a la validez es relevante en los estudios que tratan de establecer relaciones causales.

Para reducir la amenaza relacionada con la experiencia de los participantes, se planteó y se desarrolló un ejemplo de entrenamiento representativo, el cual muestra cada paso del proceso de creación de un DSL y provee a los usuarios un entendimiento de la forma en la que se estructura y se replica una arquitectura de monitoreo para IoT mediante el DSL (Monitor-IoT) propuesto.

Los sesgos del autor y los sesgos producidos por la entendibilidad del material fueron reducidos mediante una revisión previa de dicho material llevada a cabo por un investigador experto en el dominio de Ingeniería de Software para así reducir posibles errores o malentendidos relacionados con el experimento, y un experimento piloto aplicado a estudiantes de prueba con la finalidad de recibir retroalimentación que ayude a cubrir agujeros en la manera en la que se desarrollará el estudio de caso.

Por otro lado, la modalidad en la cual se realizó el estudio de caso fue de forma virtual debido a la situación que vive el país por el COVID-19, por tanto, depende mucho del proveedor de internet que utilicen los evaluadores como los participantes y eso implica que en algunos casos se escuchen de manera entrecortada indicaciones o explicaciones efectuadas, o que los evaluadores no puedan responder de forma rápida inquietudes de los participantes.

Para solventar esta amenaza se brindó todos los recursos en una página web con anterioridad donde se adjuntó un video de instalación de la herramienta y una presentación con el desglose de la información necesaria para realizar el ejercicio del estudio de caso.

Validez externa

Esta se refiere a la habilidad para generalizar los resultados en diferentes contextos. Según P Cedillo (2016), la principal amenaza a la validez externa, es la representatividad de los resultados que puede verse afectada por el diseño de la evaluación, el contexto de participantes seleccionados, el tamaño y complejidad de la tarea del estudio de caso y la difícil situación que atraviesa el país respecto a la pandemia del Covid-19.

El diseño de la evaluación puede tener un impacto en la generalización de los resultados debido a la complejidad de la arquitectura para IoT en AAL, por sus características y componentes, por tanto, se redujo este problema proponiendo un escenario bastante simple y descriptivo de forma que los participantes tengan un panorama bastante claro de lo que se desea construir.

Además, en la fase de entrenamiento se desarrolló un ejemplo en donde los evaluadores se centraron en la réplica de cada componente de una arquitectura para IoT en un invernadero, explicando a detalle la relación entre componentes de la arquitectura y componentes del editor gráfico del DSL, así como también y respondiendo dudas y cuestionamientos.

Con respecto a la experiencia de los participantes, el estudio de caso fue conducido con alumnos de Ingeniería de Sistemas y Telemática quienes han tomado cursos de Ingeniería de software y tienen un buen conocimiento de modelos y arquitecturas de software.

El tamaño y complejidad de la tarea podrían también afectar la validez externa. Para ello, se propone una sola tarea del estudio de caso con un nivel suficiente de complejidad.

Las enfermedades y virus que se producen en el mundo es difícil controlar, y hoy en día el mundo atraviesa por la pandemia del Covid-19 por lo cual toda la evaluación fue de forma virtual a través de la plataforma Zoom.

Validez del constructo

Las variables subjetivas están basadas en el Método de Evaluación del Modelo (MEM), el cual, como se ha indicado, es un método muy bien conocido y validado empíricamente para la evaluación de tecnologías de la información.

La principal amenaza es la confiabilidad del cuestionario realizado por los participantes para lo cual se realizó una prueba de confiabilidad del alfa de Cronbach para cada conjunto de preguntas relacionadas a cada variable subjetiva.

El umbral mínimo aceptado en el campo tecnológico es = 0.70, en esta validación empírica, para la medición de la facilidad de uso percibida (PEOU) se obtuvo un =0.749, para la medición de la utilidad percibida (PU) se obtuvo un =0.761 y para la medición de la intención de uso (ITU) se obtuvo un =0.789.

Validez de la conclusión

En este caso, las amenazas que afectan la validez de la conclusión se refieren a las conclusiones estadísticas, ejemplo de estas son la elección de los métodos estadísticos, y la elección del tamaño de la muestra, entre otros (P Cedillo, 2016).

Uno de los principales problemas de validez es el tamaño de la muestra en el estudio de caso, donde se toma una muestra de 17 participantes, pudiendo resultar en un problema de validez de la conclusión, dado que puede afectar el tema de causalidad entre las diferentes variables; sin embargo, los resultados fueron alentadores ya que los participantes lograron realizar exitosamente el ejercicio propuesto.

Por otro lado, para mitigar la validez de las conclusiones estadísticas, se aplicó la misma función de medición a cada variable dependiente.

5.8. Conclusiones

En este capítulo se ha presentado la planificación y ejecución de un estudio de caso, con la finalidad de evaluar la eficacia percibida por un grupo de estudiantes del área de Ingeniería de Sistemas y Telemática durante la selección y configuración de componentes en el editor gráfico del DSL (Monitor-IoT) propuesto para el diseño de arquitecturas de monitoreo para IoT.

El método que se usó para validar este enfoque fue el Método de Evaluación del Modelo (MEM), el cual considera dos aspectos importantes que son complementarios: rendimiento actual y probabilidad de aceptación en la práctica.

Para la implementación de MEM se definieron variables basadas en el rendimiento (eficiencia y efectividad) como factores de influencia para las variables basadas en la percepción (facilidad de uso percibida, utilidad percibida e intención de uso).

Los resultados obtenidos revelan que: (i) la mayoría de los participantes han encontrado el DSL (Monitor-IoT) como útil; (ii) la mayoría de los participantes están de acuerdo en que, de ser necesario, sí usarían el editor gráfico del DSL propuesto en el futuro; (iii) el rendimiento de los participantes en el estudio de caso determinó sus percepciones positivas; (iv) las percepciones determinan la intención de usar el editor gráfico del DSL propuesto.

Aunque los resultados son prometedores en cuanto a la utilidad del DSL (Monitor-IoT) propuesto, estos son preliminares y deben ser considerados con cautela ya que se basan en la evaluación de un conjunto reducido de atributos y, además, ha sido aplicado a usuarios sin experiencia previa en el desarrollo de sistemas o aplicaciones para los dominios de IoT y AAL.

Por lo tanto, es necesario realizar el experimento de forma presencial y que estén enfocados en otros aspectos del DSL (Monitor-IoT), como por ejemplo el despliegue de toda una infraestructura de monitoreo utilizando el DSL como entrada y verificar el rendimiento de cada uno de los componentes involucrando a sujetos expertos, de preferencia profesionales de la industria.

Capítulo 6

6. Conclusiones y Trabajos Futuros

En este capítulo se presentan las conclusiones de este trabajo de titulación desde el punto de vista de los objetivos de investigación, su nivel de cumplimiento y los principales hallazgos obtenidos. Además, se presentan las contribuciones de esta investigación y las posibles líneas y oportunidades de investigación futura.

La sección 6.1 presenta las conclusiones desde el punto de vista del objetivo general, los objetivos específicos de este trabajo y la hipótesis de investigación. La sección 6.2 describe las limitaciones de la infraestructura desarrollada y finalmente, la sección 6.3 presenta los trabajos futuros.

6.1. Conclusiones

A continuación, se presentan las conclusiones obtenidas según los objetivos planteados en este trabajo de titulación.

Objetivo general

El objetivo general de este trabajo de titulación es *"Desarrollar una infraestructura auto adaptativa de monitoreo mediante un middleware de configuración de componentes sensibles al contexto en ambientes de vida asistida"*.

Este objetivo ha sido cumplido en su totalidad ya que, a lo largo de este trabajo de titulación, se ha planteado un editor gráfico del DSL (Monitor-IoT) que permite crear una arquitectura de monitoreo para IoT donde se definen todos los componentes de hardware y configuraciones de software.

El editor se desarrolló en base a la ISO/IEC - CD 30141 Arquitectura de referencia de Internet de las cosas (ISO/IEC CD 30141, 2016) que nos brinda un estándar para diseñar aplicaciones de IoT.

Monitor-IoT es una parte importante en la infraestructura porque permite definir la arquitectura para IoT a fin de obtener un DSL que es el archivo de entrada para desplegar la infraestructura de monitoreo MIoT2InMon.

MIoT2InMon está desarrollada con NodeJS, su código fuente se encuentra en github y la estructura del repositorio se presenta en el Apéndice D. Esta infraestructura está formada por dos partes principales Sincronizer y Controller que representan el middleware de configuración para desplegar, API´s, servicios, bases de datos, puertos y toda la arquitectura descrita en el DSL.

En este sentido, la infraestructura de monitoreo MIoT2InMon presenta varios beneficios, a continuación, se presentan algunos de ellos:

- MIoT2InMon está pensada de tal manera que permita su implementación para cualquier contexto relacionado con IoT, independientemente del dominio que se esté modelando.
- MIoT2InMon genera toda la programación necesaria de la arquitectura IoT que se esté modelando y que haya sido configurada a nivel del DSL usando el editor gráfico Monitor-IoT.
- MIoT2InMon tiene una relación directa a nivel del DSL; es decir, cuando se realiza alguna modificación o nueva configuración del DSL, en tiempo de ejecución sus cambios se ven reflejados en la infraestructura sin necesidad de detener el sistema.
- Dado que el diseño de la arquitectura es independiente de su implementación, esto permite que el diseño sea flexible y se pueda incorporar a cualquier otro entorno o aplicación.

Objetivos específicos

Para lograr el objetivo general, se plantearon algunos objetivos específicos los cuales son analizados a continuación.

- **Objetivo específico 1**: *Presentar un estudio del estado del arte a fin de conocer los avances existentes y que puedan ser de utilidad en la solución del problema.*

Este objetivo específico fue efectuado en su totalidad, para ello se desarrolló el capítulo 3 donde se describe el mapeo sistemático de la literatura enfocado en encontrar “para qué” y “cómo" se están utilizando los modelos en tiempo de ejecución para ambientes de internet de las cosas.

En otras palabras, conocer los dominios de mayor enfoque de internet de las cosas (IoT), así como también, técnicas y propósitos para los cuales se emplean los modelos en tiempo de ejecución y la manera en la que se ha estado llevando a cabo la investigación de estos temas.

Como resultados, se encontró que los modelos en tiempo de ejecución han sido utilizados principalmente para el monitoreo y auto-adaptación de sistemas aplicados en ambientes inteligentes y salud, como subdominios de internet de las cosas (IoT), por tal razón, el objetivo principal es el usuario para el cual se busca beneficiar, facilitar actividades y mejorar su calidad de vida.

Además, los resultados nos ayudaron a conocer cómo se está trabajando con los modelos en tiempo de ejecución con el internet de las cosas (IoT), información que

sirvió para elegir la forma en que se pueden utilizar en nuestra infraestructura, así como también tecnologías que agilicen procesos de configuración o despliegue de componentes.

Finalmente, los estudios demuestran que la importancia de estas tecnologías viene dada desde un enfoque académico.

- **Objetivo específico 2**: *Diseñar los metamodelos en tiempo de ejecución y una infraestructura auto adaptativa para el monitoreo de dispositivos IoT.*

En el cumplimiento de este objetivo se trabajó de forma conjunta con un grupo de investigación y como parte de una tesis doctoral, por tal motivo con el presente trabajo de titulación se colaboró en parte del diseño de un metamodelo en tiempo de ejecución, el diseño de un DSL y el desarrollo de una infraestructura auto adaptativa para el monitoreo de dispositivos de internet de las cosas (IoT).

El diseño del metamodelo permite definir el proceso de monitoreo de componentes en arquitecturas de internet de las cosa (IoT); para representar el metamodelo se desarrolló el DSL Monitor-IoT que permite generar una estructura gráfica de la arquitectura, su resultado es la entrada para la infraestructura MIoT2InMon, la cual posee un middleware compuesto por dos elementos fundamentales de configuración; el *Sincronizer*, recibe la representación del DSL y el *Controller*, configura y ejecuta los diferentes elementos existentes en la arquitectura de internet de las cosas (IoT).

En conclusión, para implementar arquitecturas de monitoreo para internet de las cosas (IoT), se las representa a través del DSL (Monitor-IoT) que sirve de entrada para desplegar la infraestructura de monitoreo junto con el código fuente de todas las configuraciones de los componentes, además de permitir la variación de atributos o nuevos componentes en tiempo de ejecución sin necesidad de interrumpir el funcionamiento del sistema.

- **Objetivo específico 3**: *Implementar una instancia de la solución propuesta, a fin de comprobar la factibilidad de la misma desde el punto de vista técnico*.

La instancia de la solución se ha conseguido en la sección 4.4 del capítulo 4 de este trabajo. En dicho capítulo se ilustra tanto la representación de una arquitectura para internet de las cosas (IoT) de un ambiente de vida asistido (AAL) mediante el DSL Monitor-IoT, propuesta para la evaluación del estudio de caso, así como la ejecución de la infraestructura de monitoreo a partir del DSL resultante de la arquitectura propuesta en este trabajo de titulación.

La arquitectura está representada por componentes que responden a necesidades de un adulto mayor en un escenario específico, permitiendo notificarle si existe alguna fuga de gas en el entorno de tal forma que se evite algún accidente y que la vida del usuario no corra ningún riesgo, a fin de brindarle una mejor calidad de vida.

El DSL Monitor fue implementado mediante el paquete Sirius del proyecto Eclipse y la infraestructura de monitoreo MIoT2InMon se implementó en Node.js el entorno en tiempo de ejecución basado en el lenguaje de programación JavaScript, además del middleware de configuración lo compone dos elementos principales Sincronizer y Controller que son los intermediarios para la comunicación y autoconfiguración del DSL y MIoT2InMon en tiempo de ejecución.

De esta manera se consigue configurar, desplegar y monitorear arquitecturas para internet de las cosas a partir de la instanciación de un DSL, donde se genera la descripción y ejecución de los elementos, logrando así reducir tiempos en la implementación de cada componente y en caso de realizar cambios en el desarrollo de la arquitectura por medio del DSL estos se ven reflejados de manera inmediata en el despliegue en ejecución sin necesidad de detener el sistema.

- **Objetivo específico 4:** *Evaluar empíricamente la infraestructura adaptativa de monitoreo propuesto mediante experimentos controlados, con la finalidad de proporcionar evidencias sobre la facilidad de uso y utilidad de los métodos y tecnologías propuestos*.

Para cumplir este objetivo, se ha realizado un estudio de caso con personas que tienen conocimientos sobre arquitecturas de software y se evaluó como instancia la representación de una arquitectura para internet de las cosas (IoT) en un ambiente de vida asistida (AAL) a través del DSL Monitor-IoT. Cabe mencionar que se considera importante la evaluación en torno al DSL porque es fundamental una buena representación de la arquitectura de IoT para que la infraestructura se cree, configure y despliegue de forma correcta.

La evaluación se realizó aplicando el Modelo de evaluación de métodos (MEM) que considera dos aspectos fundamentales como el rendimiento actual y la probabilidad de aceptación en la práctica. Para aplicar el método se definieron variables basadas en el rendimiento (eficiencia y efectividad) como factores de influencia para las variables basadas en la percepción (facilidad de uso percibida, utilidad percibida e intención de uso).

Los resultados obtenidos revelan que: (i) la mayoría de los participantes han encontrado al DSL Monitor-IoT como fácil de usar y útil. (ii) la mayoría de los participantes están de acuerdo en que, de ser necesario, sí usarían el DSL propuesta en el futuro; y (iii) las percepciones de facilidad de uso y utilidad del DSL sí determinan la intención de usarla en el futuro.

6.2. Limitaciones

En los diferentes capítulos presentes en este trabajo de titulación se presentaron algunas limitaciones descritas a continuación.

En el estudio de literatura realizado, se rechazaron artículos científicos escritos en un idioma diferente al inglés y que no se encuentren publicados en las diferentes revistas de indexación. Esto puede generar una limitación, debido a que estudios relevantes se pudieron escribir en otro idioma y podían servirnos en alguna parte de nuestro estudio.

Las limitaciones en cuanto a la evaluación empírica se dan principalmente por el momento crítico que atraviesa el país debido a la pandemia del Covid-19, por lo cual todo este proceso se realizó de forma virtual.

Otra de las limitaciones se enmarca en los participantes, si bien tienen la formación necesaria en tecnologías de la información, no nos asegura su conocimiento en modelos en tiempo de ejecución es por ello que la evaluación se centró en el modelado de un escenario usando DSL (Monitor-IoT) que en la infraestructura de monitoreo MIoT2InMon.

Finalmente, se recomienda replicar el experimento realizado considerando más participantes, atributos, métricas de calidad, y de forma presencial, para de esta manera determinar posibles limitaciones de la solución.

De igual forma, establecer evaluaciones para dominios más complejos y diferentes, debido a que las evaluaciones realizadas están enfocadas en un dominio específico.

6.3. Trabajos Futuros

A continuación, se plantean actividades que se podrían realizar para mejorar la infraestructura propuesta en cuanto a diseño, implementación y la validación empírica de MIoT2InMon.

Para la implementación de la infraestructura de monitoreo MIoT2InMon, a futuro se utilizarán herramientas y materiales necesarios para el despliegue y uso en ambientes reales.

Además, de complementar el metamodelo Monitor-IoT con las fases restantes del Bucle MAPE-K de los modelos en tiempo de ejecución, puesto que en este trabajo se abarca solo la etapa de monitoreo (M).

En la validación y evaluación, se deben generar diversos casos de estudio para otros dominios y verificar el funcionamiento correcto de la infraestructura de monitoreo. Desde otra perspectiva, se podrían hacer evaluaciones empíricas o evaluar la usabilidad/experiencia de usuario comparando la infraestructura de monitoreo propuesta con arquitecturas similares.

Anexos

Apéndice A

Estudios Primarios

TÍTULO	AÑO	Nº PÁGS	Nº CITAS
SpringerLink			
Runtime model-based approach to IoT application development	2015	14	20
Using Models at Runtime to Adapt Self-managed Agents for the IoT	2016	19	7
Leveraging design and runtime architecture models to support self-awareness	2017	18	0
Applying Architecture-Based Adaptation to Automate the Management of Internet-of-Things	2018	19	14
Addressing the evolution of automated user behaviour patterns by runtime model interpretation	2013	34	8
Dynamic Evolution of Context-Aware Systems with Models at Runtime	2012	17	23
The role of models@run.time in supporting on-the-fly interoperability	2013	24	43
Mechanisms for Leveraging Models at Runtime in Self-adaptive Software	2014	28	39
Distributed Graph Queries for Runtime Monitoring of Cyber-Physical Systems	2018	18	7
Deterministic High-Level Executable Models Allowing Efficient Runtime Verification	2018	26	1

Architecting Dynamic Reconfiguration in Dependable Systems	2007	25	26
Autonomic Computing Driven by Feature Models and Architecture in FamiWare	2011	16	33
Modeling and Specifying Parametric Adaptation Mechanism for Self-Adaptive Systems	2014	15	11
Requirements and Assessment of Languages and Frameworks for Adaptation Models	2011	16	15
Contextualised Security Operation Deployment Through MDS@run.time Architecture	2015	12	3
Engineering Trust-Awareness and Self-adaptability in Services and Systems	2014	30	1
MCaaS: Model Checking in the Cloud for Assurances of Adaptive Systems	2017	17	3
Embedded UML Model Execution to Bridge the Gap Between Design and Runtime	2018	10	3
Runtime Adaptation of Architectural Models: An Approach for Adapting User Interfaces	2012	15	11
Supporting Runtime System Evolution to Adapt to User Behaviour	2010	15	25
Modeling foundations for executable model-based testing of self-healing cyber-physical systems	2018	31	2
Coordinated Actors for Reliable Self-adaptive Systems	2016	19	9
Emerging Techniques for the Engineering of Self-Adaptive High-Integrity Software	2013	14	21
Runtime Adaptability of Ambient Intelligence Systems Based on	2017	24	0

Component-Oriented Approach			
Distributed graph queries over models@run.time for runtime monitoring of cyber-physical systems	2019	24	1
Towards application driven security dashboards in future middleware	2012	9	6
Living with Uncertainty in the Age of Runtime Models	2014	54	33
ACM			
Towards a Model@runtime Middleware for Cyber Physical Systems	2014	6	5
Automating routine tasks in AmI systems by using models at runtime	2010	10	4
An Architecture for a Smart Spaces Virtual Machine	2014	6	4
Adaptive Exchange of Distributed Partial Models@run.time for Highly Dynamic Systems	2015	7	12
ThingsJS: towards a flexible and self-adaptable middleware for dynamic and heterogeneous IoT environments	2017	6	11
Expect the unexpected: Towards a middleware for policy adaptation in IoT platforms	2018	4	1
IEEE Explore			
Runtime knowledge graph-based approach to smart home application development	2018	8	0
Autonomic computing through reuse of variability models at runtime: The case of smart homes	2009	7	178

Self-Healing for Distributed Workflows in the Internet of Things	2017	8	3
ActivFORMS: A Runtime Environment for Architecture-Based Adaptation with Guarantees	2017	4	4
Self-Adaptive Framework with Game Theoretic Decision Making for Internet of Things	2018	6	1
Models@run.time for Creating In-Cloud Dynamic Cyber-Physical Ecosystems	2017	7	6
Runtime Management and Quantitative Evaluation of Changing System Goals	2017	8	10
Model-Driven Engineering of Decentralized Control in Cyber-Physical Systems	2017	6	6
Adaptive Service-Oriented Architectures for Cyber Physical Systems	2017	6	5
A context realization framework for ubiquitous applications with runtime support	2011	10	17
Stream my models: Reactive peer-to-peer distributed models@run.time	2018	10	20
A Model-Driven Methodology for the Design of Autonomic and Cognitive IoT-Based Systems: Application to Healthcare	2017	11	33
Towards self-adaptation for cyber-physical systems using a distributed MAPE-K schema over XMPP	2017	5	0
Science Direct			
RINGA: Design and verification of finite state machine for self-adaptive software at runtime	2018	23	13
Model-Based Runtime Monitoring of	2018	8	2

Smart City Systems			
Democratization of runtime verification for internet of things	2018	11	1
Otras revistas			
Runtime models for self-adaptation in the ambient assisted living domain (2008).	2008	10	18

Tabla A.1. Estudios primarios seleccionados para la revisión sistemática.

Apéndice B

Resultados porcentuales de los criterios de extracción

COD.	CRITERIO	RESPUESTAS	Nº ESTUDIOS	%
RQ1: ¿Bajo qué contexto se han desarrollado e implementado los sistemas IOT basados en modelos en tiempo de ejecución?				
EC1	Sub-dominio de IOT	Transporte	4	6 %
		Educación	1	1 %
		Gobierno	2	3 %
		Salud	8	11 %
		Finanzas y banca	1	1 %
		Seguridad pública y emergencia	5	7 %
		Monitoreo y control ambiental (agua, energía, alcantarillado, etc.)	6	8 %
		Agricultura	1	1 %
		Logística y venta al detalle (Control de la cadena de suministro)	3	4 %
		Control industrial	1	1 %
		Entretenimiento y deporte	2	3 %
		Ambientes inteligentes (Domótica)	19	27 %
		Control inteligente de animales	1	1 %

		Otros	17	24 %

Tabla B.1. Porcentajes individuales para los criterios de extracción de la pregunta RQ1.

COD.	CRITERIO	RESPUESTAS	Nº ESTUDIOS	%
RQ2: ¿Para qué propósitos se han implementado los modelos en tiempo de ejecución en el dominio de internet de las cosas?				
EC2	Propósitos de los modelos en tiempo de ejecución	Monitoreo	31	62 %
		Aseguramiento de calidad (requerimientos no funcionales)	7	14 %
		Interoperabilidad	7	14 %
		Auto-adaptación (self-adaptation)	37	74 %
		Auto-optimización (self-optimization)	2	4 %
		Auto-organización (self-organization)	3	6 %
		Auto-recuperación / Tolerancia a fallos / consistencia	13	26 %
		Simulación / Predicción	11	22 %
		Otro	1	2 %
EC3	Técnicas utilizadas en combinación con los modelos en tiempo de ejecución	Transformación de modelos	17	23 %
		Bucle de control automático (MAPE-K)	12	16 %
		Conformidad de modelos	5	7 %
		Aprendizaje automático / razonamiento	6	8 %
		Modelado de variabilidad,	11	15 %
		Ingeniería de	7	9 %

		requerimientos		
		Flujos de trabajo	6	8 %
		Comparación de modelos	7	9 %
		Líneas de producto software	2	3 %
		Otro	2	3 %

Tabla B.2. Porcentajes individuales para los criterios de extracción de la pregunta RQ2.

COD.	CRITERIO	RESPUESTAS	Nº ESTUDIOS	%
RQ3: ¿Cómo se aborda la investigación en los estudios relacionados a la implementación de infraestructuras que usan modelos en tiempo de ejecución en el dominio de internet de las cosas?				
EC4	Tipo de estudio	Nuevo	19	38 %
		Extensión	31	62 %
EC5	Tipo de validación	Experimento	8	16 %
		Prototipo	8	16 %
		Prueba de concepto	6	12 %
		Caso de estudio	25	50 %
		Ninguno	6	14 %
		Encuesta	0	0 %
EC6	Alcance del enfoque	Industria	2	4 %
		Academia	48	96 %

Tabla B.3. Porcentajes individuales para los criterios de extracción de la pregunta RQ3.

Apéndice C

Estudio de caso

C.1. Página Web

C.2. Documento de evaluación

DSL de una arquitectura para el internet de las cosas en un ambiente de Vida Asistida.

El presente ejercicio tiene como objetivo crear una arquitectura para el internet de las cosas (Internet of things - IoT) en un ambiente de vida asistida, a través de la instancia y configuración mediante un editor gráfico del DSL (Domain Specific Language - Lenguaje de dominio específico) con la herramienta de Obeo Designer y su paquete Sirius.

El proceso consiste en brindar al estudiante las especificaciones y el gráfico de la arquitectura que debe ser instanciada y configurada mediante Obeo Designer, y se requiere que cada estudiante registre la hora de inicio y fin de la tarea especificada al inicio del documento, y se concluye completando una encuesta en línea de la actividad realizada.

El resultado al culminar el ejercicio es un DSL que representa una arquitectura para el internet de las cosas en un ambiente de vida asistida, él mismo que sirve como entrada para la infraestructura de monitoreo *MIoT2InMon* que se desarrolló en el presente trabajo de titulación, con el fin de tener un sistema que se configure a nivel del DSL en tiempo de ejecución.

Descripción del Ambiente de vida asistida

La arquitectura de monitoreo para IoT se plantea respecto al escenario de vida de un adulto mayor independiente, él vive en una casa, la cual consta de diversas habitaciones entre ellas, cocina, dormitorio, sala.
A dicha persona le apasiona cocinar y a diario él prepara su comida, sin embargo, por su avanzada edad en ocasiones olvida apagar la estufa, por lo que puede provocar un accidente si no se detecta a tiempo la fuga de gas o en la presencia de humo en el entorno. Para esta persona, se considera necesario que se le pueda notificar si existe alguna fuga de gas o si existe humo en el ambiente para poder evitar algún accidente y que la vida del usuario no corra ningún riesgo, a fin de brindarle una mejor calidad de vida.

Componentes del DSL (Monitor-IoT)

 ***IoT System*:** representa el entorno IoT que se desea monitorear.

 ***Entity Category:*** categoriza las entidades tanto físicas como digitales.

 ***Physical Entity:*** representa objetos del mundo real.

 ***Cloud Node:*** representa el almacenamiento en la nube.

 ***Fog Node:*** es un nodo intermedio para almacenar y procesar datos antes de enviar a la nube.

 ***IoT Gateway:*** nodo de computación capaz de controlar sensores y actuadores receptando y enviando información desde o hacia estos dispositivos.

 ***Sensor:*** dispositivo que está capacitado para detectar acciones o estímulos externos.

 ***Actuador:*** dispositivo que recibe una orden y activa un elemento final.

 ***Middleware:*** orquestador que mantiene comunicación mediante servicios.

 ***Database:*** lugar donde se realiza el almacenamiento de la información.

 ***Service:*** permite intercambiar datos entre aplicaciones cliente y servidor.

 ***Broker:*** intermediario que recibe y envía datos de forma directa según un tópico al que esté suscrito.

 ***Topic:*** identificador que ayuda a ubicar los mensajes que se envían.

 ***Application***: componente de software que ofrece una colección de funciones.

 ***API:*** método que permite obtener y enviar información desde o hacia los dispositivos IoT.

 ***Network Interface:*** permite la comunicación entre los diferentes componentes.

 ***Generic Property:*** propiedad genérica que comparten un conjunto de entidades del mismo tipo.

 ***Specific Property:*** propiedad específica de una entidad.

 ***Data Table:*** es una tabla de la DB que contiene Data Column.

 ***Data Column:*** almacena el/los valores recopilados.

 ***Network:*** describe el tipo de red sea de Proximidad, LAN o Internet.

 ***Protocol:*** permite que dos o más entidades se comuniquen entre ellas para transmitir información.

 ***Header:*** información adicional que se envía en un Service.

 ***Parameter:*** valores que recibe un Service/API.

 ***Return Variable:*** valores que retorna un Service/API.

 ***Data Flow:*** comunicación directa entre los dispositivos y la base de datos.

 ***Property - DataColumn:*** Regla de mapeo de una propiedad a una columna de la base de datos.

 ***DataColumn - DataColumn:*** Regla de mapeo de una columna de una base de datos a otra base de datos.

Link: permite definir las Entity conectadas por donde se lleva a cabo un DataFlow.

Arquitectura para el internet de las cosas en un ambiente de vida asistida.

Nombres: **Apellidos:**
Profesión:
Hora Inicio: **Hora Fin:**

Arquitectura

En base a la descripción del ambiente de vida asistido planteado, en la figura 1 presenta la arquitectura para el internet de las cosas en un ambiente de vida asistida.

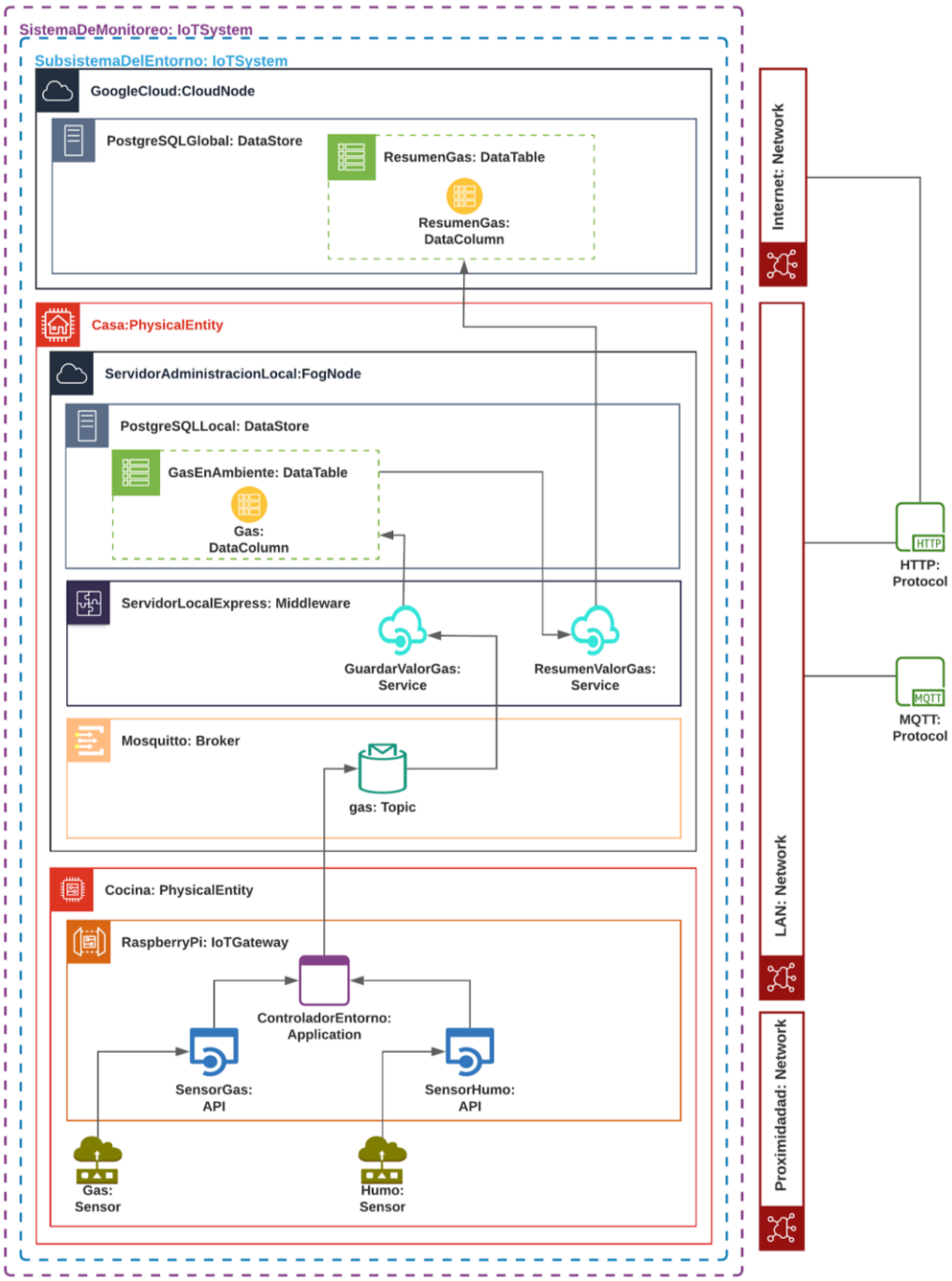


Figura 1. Arquitectura para IoT en un AAL.

Componentes

Los componentes que forman parte de la arquitectura para IoT y se deben instanciar y configurar en Obeo Designer son los siguientes:

SistemaDeMonitoreo, es el nombre general que recibe el sistema para monitorear datos en un ambiente de vida asistida.

SubsistemaDelEntorno, es el nombre que recibe el sistema para monitorear datos acerca del espacio habitacional de un adulto mayor.

Internet, es la interfaz de red utilizada para enviar o receptar información desde la casa del adulto mayor hacia la nube utilizando el protocolo *HTTP*.

LAN, es la interfaz de red utilizada para intercambiar información a través del protocolo *MQTT* dentro de la casa del adulto mayor.

Proximidad, es la interfaz de red utilizada para enviar datos desde el sensor al *IoTGateway* correspondiente.

GoogleCloud, es la plataforma de almacenamiento y procesamiento en la nube que aloja *PostgreSQL*, la base de datos donde se almacena toda la información final recopilada de los dispositivos IoT.

ResumenGas, es una tabla de la base de datos localizada en la nube que almacena un resumen de los datos del sensor de gas, a fin de no almacenar toda la información en la nube que a largo plazo genera grandes costos. Esta tabla contiene una columna denominada *ResumenGas* donde se almacenan los valores recopilados por el sensor de gas.

Casa, es el espacio habitacional del adulto mayor donde se despliegan los dispositivos.

ServidorAdministraciónLocal, es un nodo Fog intermedio que maneja/procesa los datos previamente a ser publicados en la nube. Este nodo aloja *PostgreSQLLocal*, una base de datos compuesta por tablas y columnas que describen los datos recopilados de cada sensor.

ServidorExpressLocal, es un middleware desplegado en el *ServidorAdministraciónLocal,* contiene servicios para enviar información en una base de datos local y directo a la nube.

Mosquitto, es un broker desplegado en el *ServidorAdministraciónLocal,* que utiliza un modelo de publicación/suscripción bajo un tópico de comunicación entre el ControladorEntorno y el servicio GuardarValorGas.

Cocina, es la habitación monitoreada donde se localiza el sensor de gas y humo.

RaspberryPi, es un tipo de *IoTGateway* que maneja el flujo de comunicación de la aplicación *ControladorEntorno* con el servicio desplegado en el middleware *ServidorExpressLocal* hacia la base de datos *PostgreSQLLocal*. Y despliega las API´s *SensorHumo* para la recepción de datos del sensor *Humo* y *SensorGas* para la recepción de datos del sensor *Gas* y se encarga del envío de datos hacia el *ControladorEntorno.*

ControladorEntorno, es una aplicación que se ejecuta de forma privada y no visible al usuario final, sin embargo, en lo posterior puede ser utilizado de forma pública como interfaz donde el usuario final pueda revisar la recopilación de datos en tiempo de

ejecución. Esta aplicación se encarga de receptar y enviar información a los nodos superiores o visceversa.

Gas, es el sensor encargado de recopilar los valores de presencia de gas en el aire de forma constante y enviarlos hacia la *API* correspondiente para almacenarlos en la base de datos de los nodos superiores.

Humo, es el sensor encargado de recopilar los valores de humo en el aire de forma constante y enviarlos hacia la *API* correspondiente para almacenarlos en la base de datos de los nodos superiores.

DSL de la arquitectura

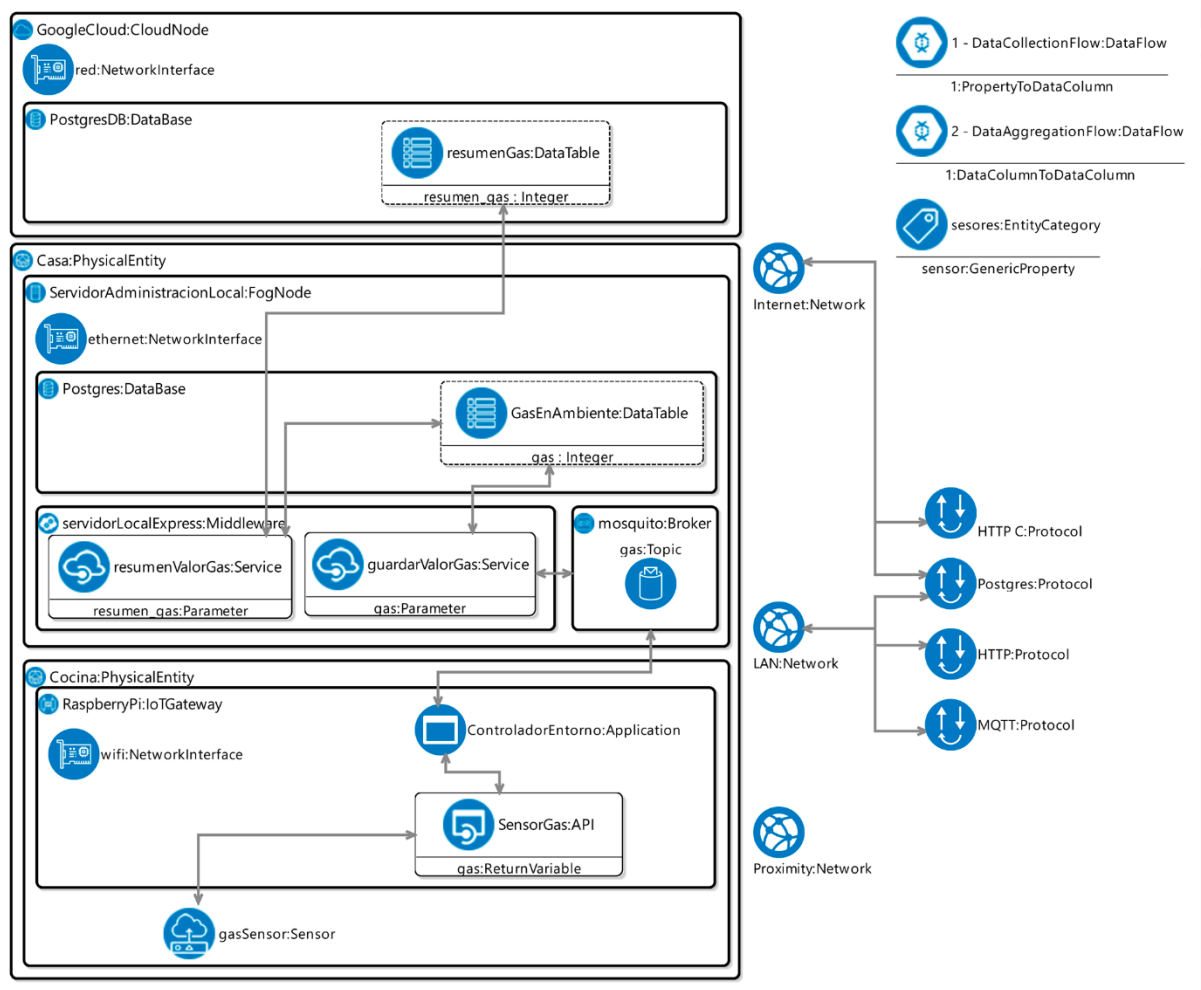
Formulario

Para finalizar el ejercicio, es necesario completar la siguiente encuesta en línea a través del enlace adjunto.

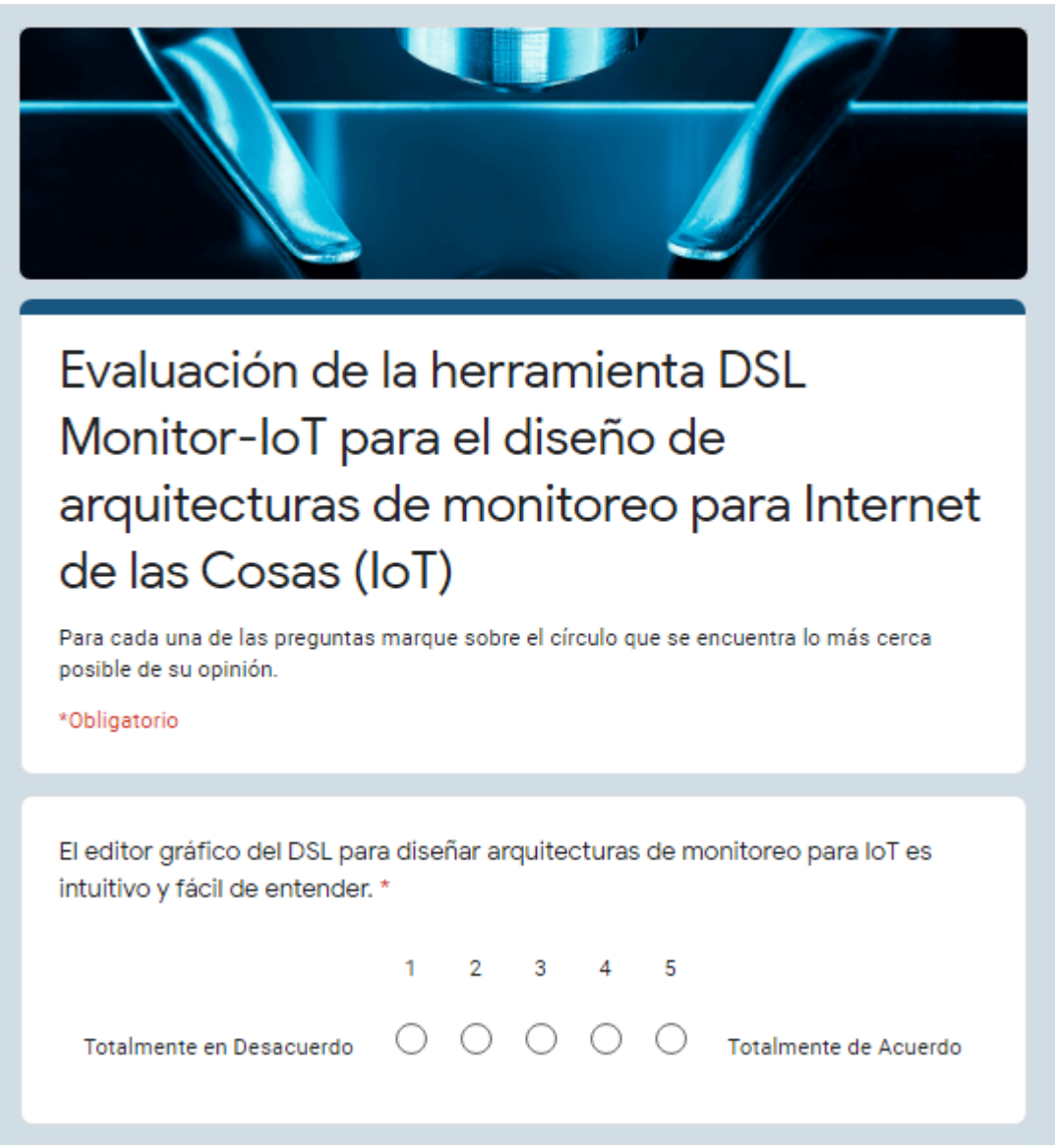
Enlace: https://forms.gle/f2HfQHSY6qJ9hUtNA

¡¡GRACIAS POR SU COLABORACIÓN!!

C.3. Solución de la arquitectura en el DSL Monitor-IoT



C.4. Formulario



Evaluación de la herramienta DSL Monitor-IoT para el diseño de arquitecturas de monitoreo para Internet de las Cosas (IoT)

Para cada una de las preguntas marque sobre el círculo que se encuentra lo más cerca posible de su opinión.

*Obligatorio

El editor gráfico del DSL para diseñar arquitecturas de monitoreo para IoT es intuitivo y fácil de entender. *

	1	2	3	4	5	
Totalmente en Desacuerdo	○	○	○	○	○	Totalmente de Acuerdo

La notación gráfica (representación de los componentes) utilizada por el DSL es intuitiva y fácil de entender. *

	1	2	3	4	5	
Totalmente en Desacuerdo	○	○	○	○	○	Totalmente de Acuerdo

El editor gráfico del DSL para diseñar arquitecturas de monitoreo para IoT es fácil de usar. *

	1	2	3	4	5	
Totalmente en Desacuerdo	○	○	○	○	○	Totalmente de Acuerdo

El uso del DSL podría reducir el tiempo y esfuerzo requerido en el desarrollo, despliegue y mantenimiento de sistemas de IoT. *

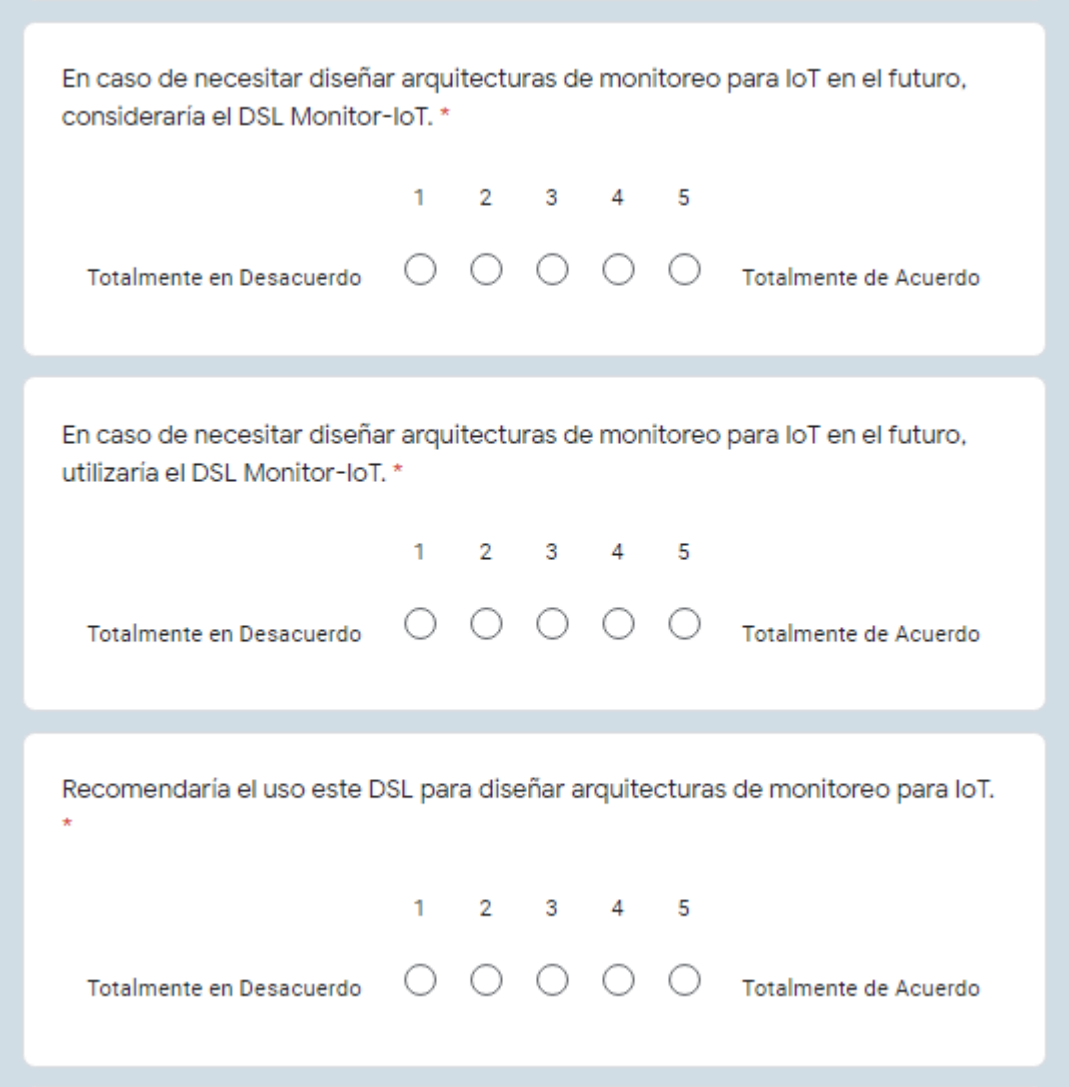
	1	2	3	4	5	
Totalmente de Desacuerdo	○	○	○	○	○	Totalmente en Acuerdo

Considera que el DSL sería de utilidad para apoyar las actividades de desarrollo, despliegue y mantenimiento de sistemas de IoT. *

	1	2	3	4	5	
Totalmente en Desacuerdo	○	○	○	○	○	Totalmente de Acuerdo

El uso del DSL Monitor-IoT podría mejorar su rendimiento en el desarrollo, despliegue y mantenimiento de sistemas de IoT.

	1	2	3	4	5	
Totalmente de Desacuerdo	○	○	○	○	○	Totalmente de Acuerdo



C.4. Resultados

Métricas de Calificación

C1: Componentes del tipo Sistemas
C2: Componentes del tipo Entidades Físicas
C3: Componentes del tipo Nodos Computación
C4: Componentes del tipo Recursos
C5: Componentes del tipo Redes
C6: Componentes del tipo Enlaces
C7: Componentes del tipo Propiedades

ANÁLISIS DE EFECTIVIDAD									
ID	**NOMBRE**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**TOTAL**
1	Alexander Peñafiel	1	1	1	1	0	1	1	0,86
2	Andrea Andrade	1	1	1	1	1	0,5	1	0,93

3	Brian Mora	1	1	1	1	1	1	1	1,00
4	Bryan Alba	1	1	1	1	1	0,5	0,5	0,86
5	Christian Alvarado	1	1	1	1	1	1	1	1,00
6	Diego Bonifaz	1	1	1	1	1	0,5	1	0,93
7	Edison Pesantez	1	1	1	1	1	0,5	0,5	0,86
8	Evelin Reinoso	1	1	1	1	1	1	1	1,00
9	Jaime Panatta	1	1	1	1	1	1	1	1,00
10	Jhon Calle	1	1	1	1	1	1	0,5	0,93
11	Juan Avila	1	1	1	1	1	1	1	1,00
12	Luis Guerrero	1	1	1	1	1	1	0,5	0,93
13	Nicolas Alvarez	1	1	1	1	1	1	1	1,00
14	Pablo Loja	0,5	1	1	1	1	1	1	0,93
15	Sebastian Cavache	1	1	1	1	1	1	1	1,00
16	Sebastian Pinos	1	1	1	1	1	1	1	1,00
17	Water Larriva	1	1	1	1	1	1	1	1,00

Tabla C.1. Resultados efectividad de estudiantes.

ANÁLISIS DE EFICIENCIA		
ID	**NOMBRE**	**TOTAL**
1	Alexander Peñafiel	44
2	Andrea Andrade	50
3	Brian Mora	34
4	Bryan Alba	53
5	Christian Alvarado	24
6	Diego Bonifaz	60
7	Edison Pesantez	45
8	Evelin Reinoso	65
9	Jaime Panatta	55
10	Jhon Calle	34
11	Juan Avila	35
12	Luis Guerrero	40
13	Nicolas Alvarez	35
14	Pablo Loja	47

15	Sebastian Cavache	49
16	Sebastian Pinos	35
17	Water Larriva	42

Tabla C.2. Resultados eficiencia de estudiantes.

C.5. Evaluación

Apéndice D

Código fuente

Enlace: *https://github.com/Kramerx/runtimeIoT*

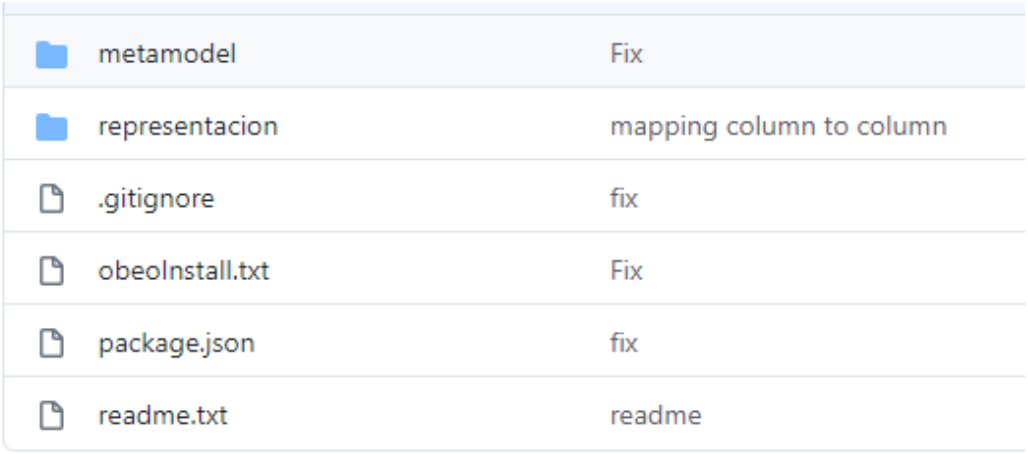


Figura D.1. Carpetas del repositorio Monitor-IoT.

metamodel: contiene las carpetas necesarias para desplegar el DSL (Monitor-IoT).

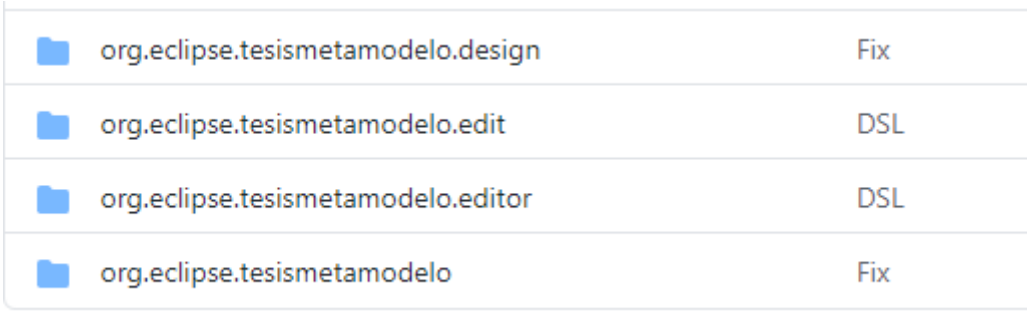


Figura D.2. Subcarpetas de la carpeta metamodel.

representación: contiene toda la programación de la infraestructura en base al DSL.

Figura D.3. Subcarpetas de la carpeta representación.

package.json: describe todas las configuraciones del proyecto.

24 lines (24 sloc) | 1.64 KB

```
{
  "name": "mqttproject",
  "version": "1.0.0",
  "description": "1. Instalar nodejs (https://nodejs.org/es/download/)\r
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "express": "^4.17.1",
    "morgan": "^1.9.1",
    "mqtt": "^3.0.0",
    "pg": "^7.18.1",
    "request": "^2.88.2",
    "xml-js": "^1.6.11",
    "xml2js": "^0.4.23"
  },
  "devDependencies": {
    "nodemon": "^2.0.2"
  }
}
```

Figura D.4. Configuraciones generales del proyecto.

Bibliografía

Aazam, M., & Huh, E.-N. (2016). Fog computing: {The} cloud-iot{\textbackslash}/ioe middleware paradigm. *IEEE Potentials*, *35*(3), 40–44.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: {A} survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, *17*(4), 2347–2376.

Alférez, G. H., & Pelechano, V. (2012). *Dynamic evolution of context-aware systems with models at runtime*. 70–86.

Ardagna, D., & Zhang, L. (2010). *Run-time Models for Self-managing Systems and Applications*. Springer Science & Business Media.

Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: {A} survey. *Computer Networks*, *54*(15), 2787–2805.

Augusto, J. C., Nakashima, H., & Aghajan, H. (2010). Ambient intelligence and smart environments: {A} state of the art. In *Handbook of ambient intelligence and smart environments* (pp. 3–31). Springer.

Ayala, I., Horcas, J. M., Amor, M., & Fuentes, L. (2016). *Using models at runtime to adapt self-managed agents for the {IoT}*. 155–173.

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). Encyclopedia of Software Engineering, chapter The goal question metric approach. *Wiley&Sons Inc*.

Bencomo, N., Bennaceur, A., Grace, P., Blair, G., & Issarny, V. (2013). The role of models@ run. time in supporting on-the-fly interoperability. *Computing*, *95*(3), 167–190.

Bencomo, N., Götz, S., & Song, H. (2019). Models@ run. time: a guided tour of the state of the art and research challenges. *Software & Systems Modeling*, *18*(5), 3049–3082.

Besnard, V., Brun, M., Jouault, F., Teodorov, C., & Dhaussy, P. (2018). *Embedded {UML} {Model} {Execution} to {Bridge} the {Gap} {Between} {Design} and {Runtime}*. 519–528.

Blair, G., Bencomo, N., & France, R. B. (2009). Models@ run. time. *Computer*, *42*(10), 22–27.

Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). *Fog computing and its role in the internet of things*. 13–16.

Botia, J. A., Villa, A., & Palma, J. (2012). Ambient Assisted Living system for in-home monitoring of healthy independent elders. *Expert Systems with Applications*, *39*(9), 8136–8148.

Botta, A., De Donato, W., Persico, V., & Pescapé, A. (2016). Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems*, *56*,

684–700.

Cedillo, P. (2016). *Monitorización de calidad de servicios cloud mediante modelos en tiempo de ejecución*.

Cedillo, Priscila, Jimenez-Gomez, J., Abrahao, S., & Insfran, E. (2015). *Towards a monitoring middleware for cloud services*. 451–458.

Cetina, C., Giner, P., Fons, J., & Pelechano, V. (2009). Autonomic computing through reuse of variability models at runtime: {The} case of smart homes. *Computer*, *42*(10), 37–43.

Cook, T. D., Campbell, D. T., & Day, A. (1979). *Quasi-experimentation: {Design} & analysis issues for field settings* (Vol. 351). Houghton Mifflin Boston.

Costa, F. M., Provensi, L. L., & Vaz, F. F. (2006). Using runtime models to unify and structure the handling of meta-information in reflective middleware. *International Conference on Model Driven Engineering Languages and Systems*, 232–241.

Da Xu, L., He, W., & Li, S. (2014). Internet of things in industries: {A} survey. *IEEE Transactions on Industrial Informatics*, *10*(4), 2233–2243.

Dastjerdi, A. V., & Buyya, R. (2016). Fog computing: {Helping} the {Internet} of {Things} realize its potential. *Computer*, *49*(8), 112–116.

Dastjerdi, A. V., Gupta, H., Calheiros, R. N., Ghosh, S. K., & Buyya, R. (2016). Fog computing: {Principles}, architectures, and applications. In *Internet of {Things}* (pp. 61–75). Elsevier.

Davis, F. D. (1985). *A technology acceptance model for empirically testing new end-user information systems: {Theory} and results*.

Dohr, A., Modre-Opsrian, R., Drobics, M., Hayn, D., & Schreier, G. (2010). *The internet of things for ambient assisted living*. 804–809.

Foundation, T. E. (2004). Sirius - {Overview}. In *What is Sirius?* https://www.eclipse.org/sirius/overview.html

Freitas, L. A., Costa, F. M., Rocha, R. C. A., & Allen, A. (2014). *An architecture for a smart spaces virtual machine*. 1–6.

Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., & Steenkiste, P. (2004). Rainbow: {Architecture}-based self-adaptation with reusable infrastructure. *Computer*, *37*(10), 46–54.

Garzon, S. R., & Cebulla, M. (2010). *Model-based personalization within an adaptable human-machine interface environment that is capable of learning from user interactions*. 191–198.

Erazo-Garzon, L., Roman, A., Moyano, J., Cedillo, P. (2020). Models@runtime and Internet of Things: A Systematic Literature Review. *II International Conference on Information Systems and Software Technologies.*

Erazo-Garzon, L., Cedillo, P., Moyano, J., Roman, A. (2021). *Monitor-IoT: A Domain Specific Language for designing monitoring architectures for IoT Systems.* Article in preparation.

Giese, H., Bencomo, N., Pasquale, L., Ramirez, A. J., Inverardi, P., Wätzoldt, S., & Clarke, S. (2014). Living with uncertainty in the age of runtime models. In *Models@ run. time* (pp. 47–100). Springer.

Gjerlufsen, T., Ingstrup, M., & Olsen, J. W. (2009). Mirrors of meaning: {Supporting} inspectable runtime models. *Computer*, *42*(10), 61–68.

Gorschek, T., Garre, P., Larsson, S., & Wohlin, C. (2006). A model for technology transfer in practice. *IEEE Software*, *23*(6), 88–95.

Grace, P., Blair, G. S., & Samuel, S. (2005). A reflective framework for discovery and interaction in heterogeneous mobile environments. *ACM SIGMOBILE Mobile Computing and Communications Review*, *9*(1), 2–14.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., & Buyya, R. (2017). {iFogSim}: {A} toolkit for modeling and simulation of resource management techniques in the {Internet} of {Things}, {Edge} and {Fog} computing environments. *Software: Practice and Experience*, *47*(9), 1275–1296.

Incki, K., & Ari, I. (2018). Model-based runtime monitoring of smart city systems. *Procedia Computer Science*, *134*, 75–82.

ISO/IEC. (2016). *Information technology – {Internet} of {Things} {Reference} {Architecture} ({IoT} {RA)}*. https://www.w3.org/WoT/IG/wiki/images/9/9a/10N0536_CD_text_of_ISO_IEC_30141.pdf

JetBrains. (2000). What are Domain-Specific Languages (DSL) | MPS by JetBrains. In *JetBrains*. https://www.jetbrains.com/mps/concepts/domain-specific-languages/

Jing, X., & Jian-Jun, Z. (2010). *A brief survey on the security model of cloud computing*. 475–478.

Jonckers, D., Lagaisse, B., & Joosen, W. (2018). *Expect the unexpected: {Towards} a middleware for policy adaptation in {IoT} platforms*. 7–10.

Kang, B., Kim, D., & Choo, H. (2017). Internet of everything: {A} large-scale autonomic {IoT} gateway. *IEEE Transactions on Multi-Scale Computing Systems*, *3*(3), 206–214.

Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F., & Alonso-Zarate, J. (2015). A survey on application layer protocols for the internet of things. *Transaction on IoT and Cloud Computing*, *3*(1), 11–17.

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. El, & Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, *28*(8), 721–734. https://doi.org/10.1109/TSE.2002.1027796

Kitchenham, B., & Charters, S. (2007). *Guidelines for performing systematic literature reviews in software engineering*.

Kleinberger, T., Becker, M., Ras, E., Holzinger, A., & Müller, P. (2007). *Ambient intelligence in assisted living: enable elderly people to handle future interfaces*. 103–112.

Lee, E., Kim, Y.-G., Seo, Y.-D., & Baik, D.-K. (2018). *Self-adaptive framework with game theoretic decision making for {Internet} of things*. 2092–2097.

Lehmann, G., Blumendorf, M., Trollmann, F., & Albayrak, S. (2010). *Meta-modeling runtime models*. 209–223.

Lethbridge, T. C., Sim, S. E., & Singer, J. (2005). Studying software engineers: {Data} collection techniques for software field studies. *Empirical Software Engineering*, *10*(3), 311–341.

Ma, H.-D. (2011). Internet of things: {Objectives} and scientific challenges. *Journal of Computer Science and Technology*, *26*(6), 919–924.

Mahmood, Z. (2011). *Cloud computing: {Characteristics} and deployment approaches*. 121–126.

Mainetti, L., Manco, L., Patrono, L., Secco, A., Sergi, I., & Vergallo, R. (2016). *An ambient assisted living system for elderly assistance applications*. 1–6.

Mell, P., & Grance, T. (2011). *The {NIST} definition of cloud computing*.

Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012). Internet of things: {Vision}, applications and research challenges. *Ad Hoc Networks*, *10*(7), 1497–1516.

Moody, D. L. (2001). *A {Practical} {Method} for {Representing} {Large} {Entity} {Relationship} {Models}*.

More, P. (2015). Review of implementing fog computing. *International Journal of Research in Engineering and Technology*, *4*(06), 335–338.

Mukherjee, M., Matam, R., Shu, L., Maglaras, L., Ferrag, M. A., Choudhury, N., & Kumar, V. (2017). Security and privacy in fog computing: {Challenges}. *IEEE Access*, *5*, 19293–19304.

Naik, N. (2017). *Choice of effective messaging protocols for {IoT} systems: {MQTT}, {CoAP}, {AMQP} and {HTTP}*. 1–7.

Node.js. (2021). Node.js. In *Node.js*. https://nodejs.org/es/

Oliveira, T. J. M., Costa, Â., Neves, J., & Novais, P. (2013). A comprehensive clinical guideline model and a reasoning mechanism for AAL systems. *International Journal of Artificial Intelligence*, *11*(A13), 57–73. http://files/149/Oliveira et al. - 2013 - A comprehensive clinical guideline model and a rea.pdf

Rashidi, P., & Mihailidis, A. (2012). A survey on ambient-assisted living tools for older

adults. *IEEE Journal of Biomedical and Health Informatics*, *17*(3), 579–590.

Rayes, A., & Salam, S. (2017). The things in iot: {Sensors} and actuators. In *Internet of {Things} {From} {Hype} to {Reality}* (pp. 57–77). Springer.

Robson, C. (2002). *Real world research: {A} resource for social scientists and practitioner-researchers* (Vol. 2). Blackwell Oxford.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131.

Schneider, D., & Becker, M. (2008). Runtime models for self-adaptation in the ambient assisted living domain. *Technical Report COMP COMP-005-2008 Lancaster University*, 47.

SECTOR, S., & ITU, O. F. (2012). Series y: {Global} information infrastructure, internet protocol aspects and next-generation networks next generation networks–frameworks and functional architecture models. *International Telecommunication Union, Geneva, Switzerland, Recommendation ITU-T Y*, *2060*.

Serral Asensio, E., Valderas Aranda, P. J., & Pelechano Ferragud, V. (2014). Supporting ambient assisting living by using executable context-adaptive task models. *International Journal On Advances in Software*, *7*(1&2), 77–87.

Sun, H., De Florio, V., Gui, N., & Blondia, C. (2009). *Promises and challenges of ambient assisted living systems*. 1201–1207.

Szvetits, M., & Zdun, U. (2016). Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime. *Software & Systems Modeling*, *15*(1), 31–69.

Thiry, M., & Schmidt, R. A. (2017). *Self-adaptive {Systems} {Driven} by {Runtime} {Models}.* 248–253.

Vogel, T., & Giese, H. (2011). *Requirements and assessment of languages and frameworks for adaptation models*. 167–182.

Weyns, D., Iftikhar, M. U., Hughes, D., & Matthys, N. (2018). *Applying architecture-based adaptation to automate the management of internet-of-things*. 49–67.

Whittle, J., Sawyer, P., Bencomo, N., Cheng, B. H. C., & Bruel, J.-M. (2009). *Relax: {Incorporating} uncertainty into the specification of self-adaptive systems*. 79–88.

Wohlin, C. (2007). *Introduction to aggregation of case studies why aggregation*.

Yannuzzi, M., Milito, R., Serral-Gracià, R., Montero, D., & Nemirovsky, M. (2014). *Key ingredients in an {IoT} recipe: {Fog} {Computing}, {Cloud} computing, and more {Fog} {Computing}*. 325–329.

Yin, R. K. (2003). Applied social research methods series. *Case Study Research: Design and Methods*, *5*(1).

Youssef, A. E. (2012). Exploring cloud computing services and applications. *Journal of Emerging Trends in Computing and Information Sciences*, *3*(6), 838–847.

Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, *1*(1), 7–18.

Zhu, M., Ye, X., Xiang, T., Ma, Y., & Chen, X. (2018). *Runtime knowledge graph based approach to smart home application development*. 110–117.