TDDM4IoTS: Una metodología de desarrollo basada en pruebas para sistemas basados en la Internet de los objetos (IoT)

Suscríbete a DeepL Pro para poder editar este documento.  
Entra en www.DeepL.com/pro para más información.

Gleiston Guerrero-Ulloa  
 Facultad de Ciencias de la Ingeniería   
Universidad Técnica Estatal de Quevedo

Quevedo, Ecuador

gguerrero@uteq.edu.ecMiguel J. Hornos Departamento de Ingeniería del Software

Universidad de Granada

Granada, España [mhornos@ugr.es](mailto:email@email.com)

Carlos Rodríguez-Domínguez  
 Departamento de Ingeniería del Software   
Universidad de GranadaGranada  
 España   
carlosrodriguez@ugr.es

*Resumen-Este* trabajo presenta una metodología de desarrollo de sistemas (IoTS) basados en Internet de los objetos que recoge ideas de varios de los paradigmas de desarrollo de software más destacados en la actualidad, como la Ingeniería Basada en Modelos (MDE) y el Desarrollo Basado en Pruebas (TDD), además de incorporar los principios que rigen las metodologías ágiles de desarrollo de software, como SCRUM y XP. La metodología aquí presentada, llamada Test-Driven Development Methodology for IoTS (TDDM4IoTS), ha sido propuesta tras una revisión exhaustiva de diferentes metodologías de desarrollo de software, lo que nos lleva a la conclusión de que ninguna de ellas está especialmente orientada al desarrollo de IoTS. La metodología consiste principalmente en once fases, cuyo orden de aplicación puede ser establecido por el equipo que desarrollará el proyecto en cuestión. En este trabajo, sugerimos un orden a seguir, así como las herramientas de software existentes que podrían ser utilizadas como soporte para la obtención de los correspondientes entregables en cada fase.

Palabras clave - Metodología de desarrollo de software, desarrollo basado en pruebas, ingeniería basada en modelos, metodologías ágiles, Internet de los objetos (IoT), sistemas basados en IoT.

# Introducción

En la Ingeniería del Software, las metodologías han sido siempre el segundo problema abordado por los investigadores, justo después de la propuesta de nuevos lenguajes y paradigmas de programación. Así, la programación estructurada surgió primero[[1]](#footnote-1)[[2]](#footnote-2), y luego se propusieron metodologías apropiadas para el Análisis y Diseño Estructurado (SAD). Asimismo, la programación orientada a objetos se propuso por primera vez en 1972[1], mientras que las propuestas sobre Análisis y Diseño Orientado a Objetos (OOAD) y una metodología para el desarrollo de Software Orientado a Objetos se publicaron respectivamente en 1978[2] y 1982[3]. Con el surgimiento de la era de Internet y la World Wide Web (WWW), los desarrolladores se enfrentaron a la necesidad de adaptar las metodologías existentes para desarrollar sistemas basados en la Web, lo que llevó a la propuesta de otras más apropiadas. La primera metodología de desarrollo Web documentada fue presentada por Schwabe y Rossi en 2002[4]. Por lo tanto, podemos concluir que, en Ingeniería de Software, ha sido tradicionalmente necesario revisar las metodologías de desarrollo tras la aparición de nuevos paradigmas tecnológicos.

Hoy en día, la Internet de los objetos (IoT) es uno de los paradigmas tecnológicos más destacados. Este término, acuñado por Kevin Ashton[5], nace con el objetivo de "digitalizar los objetos físicos" para que puedan interactuar entre sí y con las personas que los rodean y mejorar su estilo de vida y su productividad[6]. La IoT es el resultado de la confluencia/colaboración de varias áreas de investigación, como la comunicación y la cooperación, la localización y la identificación, las redes de sensores y actuadores, el tratamiento integrado de la información distribuida, la inteligencia artificial y las interfaces de usuario adaptables, por nombrar sólo algunos de los campos convergentes más importantes.

Los primeros sistemas basados en la IoT (IOTS) se han desarrollado utilizando metodologías ad-hoc, específicas para cada equipo de desarrollo, derivadas de la correspondiente adaptación de metodologías pensadas para el desarrollo de sistemas de información más tradicionales (SI).

Sin embargo, el desarrollo de IoTS difiere del desarrollo de los sistemas informáticos tradicionales en varios aspectos clave. Por ejemplo, el desarrollo de IoTS implica necesariamente el despliegue y la configuración de componentes de hardware (sensores, actuadores, controladores...) para interactuar con el entorno físico y digital, lo que no suele ser el caso en los sistemas de información tradicionales. Cada uno de los dispositivos de hardware desplegados en el entorno (como sensores, actuadores y ordenadores monoplaca) requiere una programación y configuración específica, así como la implementación de mecanismos de difusión de la información (Publish/Subscribe o Request/Response, entre los más comunes) para distribuir eficientemente los datos y crear complejos flujos de datos entre ellos.

Por otro lado, se deben implementar aplicaciones de software para interactuar con las personas en ambos tipos de sistemas. Las aplicaciones cliente de usuario final pueden estar basadas en la web[7] o en el móvil[8], dependiendo de las necesidades de los propios usuarios[9]. En la literatura, la gran mayoría de las metodologías de desarrollo de IoTS se centran exclusivamente en la implementación de estas aplicaciones, sin cubrir la configuración/implementación de los dispositivos de hardware que tienen que ser desplegados en loTS.

Universidad Técnica Estatal de Quevedo

Los principales objetivos de este documento son: (1) Presentar una revisión exhaustiva de las metodologías de desarrollo de IoTS existentes, basadas en metodologías TDD, MDE y/o ágiles; (2) Verificar que no existe una metodología específicamente diseñada para el desarrollo de loTS; y (3) Proponer una nueva metodología de desarrollo de loTS que, además del software encargado de la lógica de negocio y de la interacción usuario-sistema, aborde la configuración y despliegue del hardware (sensores, actuadores, procesadores, etc.)....) y la programación de ordenadores monoplaca (Arduino, Raspberry,...), para que puedan realizar un preprocesamiento adecuado de los datos capturados por los sensores.

El resto de este documento está estructurado de la siguiente manera: La Sección 2 presenta el estado del arte de las metodologías de desarrollo de IoTS basadas en TDD, MDE, y/o metodologías de desarrollo ágil. En la sección 3 se propone una nueva metodología para el desarrollo de los IoTS que intenta superar la ausencia de una metodología específica para el desarrollo de los IoTS. Por último, en la sección 4 se esbozan nuestras conclusiones y nuestro trabajo futuro.

# Estado del arte

Se buscaron artículos publicados sobre las metodologías de desarrollo de IoTS en la plataforma Web of Science[[3]](#footnote-3), en su colección principal, sin limitar el año de publicación. Se seleccionaron Artículos, libros, capítulos de libros, y en inglés[[4]](#footnote-4), ya que es el idioma adoptado internacionalmente para las publicaciones científicas. Como se muestra en la columna central de la TABLE I. , los términos de búsqueda que se utilizó integran una combinación inicial de palabras clave de los temas de interés.

1. Tabla 1. Palabras clave iniciales y resultados de búsqueda

| Sentence | Sentence Query | Results |
| --- | --- | --- |
| #1 | TS=(IoT OR "Internet of Things") | 15.658 |
| #2 | TS=(TDD OR "Test-Driven Development" ) | 1.572 |
| #3 | TS=(MDE OR "Model-Driven Engineering" ) | 2.280 |
| #4 | TS=(MDA OR "Model-Driven Architecture") | 56.259 |
| #5 | TS=(MDD OR "Model-Driven Development" OR "Model-Driven Design") | 12.303 |
| #6 | TS=(Agile OR SCRUM OR XP OR "Extreme Programming" OR "Agile Inception" OR "Design Sprint" OR Kanban) | 14.462 |
| #7 | TS=(Development OR Deploy OR Implement\* OR Design OR construct\*) | 7.392.445 |

Posteriormente se procedió a combinar todos los resultados, obteniendo los resultados finales mostrados en la TABLE III.

1. Resultados finales y seleccionados de la búsqueda

| Sentence | Combination | Results/Selected |
| --- | --- | --- |
| 1 | #1 AND #2 AND #7 | 6/0 |
| #2 | #1 AND #3 AND #7 | 5/5 |
| #3 | #1 AND #4 AND #7 | 6/3\* |
| #4 | #1 AND #5 AND #7 | 14/10\* |
| #5 | #1 AND #6 AND #7 | 43/3 |

\* 2 trabajos coincidentes

Se obtuvo 73 documentos que fueron analizados, y se obtuvo 21 documentos que exponían con algún grado de claridad una metodología o framework de desarrollo de IoTS.

## Fundamentos de las metodologías revisadas

Ninguno de los documentos analizados relacionados con la DDT presentaba una metodología de desarrollo para el IoTS, a diferencia de los relacionados con el MDE y las metodologías de desarrollo ágil. En la TABLE III. se muestran las referencias en las que se han encontrado las diferentes metodologías, así como los enfoques en los que se basan, además del tipo de IoTS y el dominio para el que se han desarrollado o al que se han aplicado.

1. Metodologías para el desarrollo de IoTS

| Referencia | Enfoques | General IoTS | Dominio |
| --- | --- | --- | --- |
| [10] | MDE | ✓ | Alumbrado público inteligente |
| [11] | MDD, SOA♣ | 🗶 | IIoT♠, Automóviles |
| [12] | MDD, MDA♥ | 🗶 | Aplicaciones móviles |
| [13] | Diseño basado en componentes, BIP♦, Diseño incremental | 🗶 | Sistemas de redes de área personal inalámbricas |
| [14] | MDD | ✓ | Domótica, IIoT♠ |
| [15] | MDE | 🗶 | Vigilancia de la salud |
| [16] | SOA\*, Principios de desarrollo ágil | ✓ | Sistemas de gestión ambiental y de riesgos para la IIoT |
| [17] | Marco SCRUM, Metamodelos, SOA♣ | ✓ | Casas inteligentes |
| [18] | MDE, SOA♣ | ✓ | General |
| [19] | MDA♥ | ✓ | Red de Sensores Inalámbricos |
| [20] | Cascada, principios ágiles | ✓ | No se aplica |
| [21] | División por funciones o responsabilidades | ✓ | Edificios inteligentes |

♣ Arquitectura orientada a servicios; Arquitectura basada en modelos; ♥  
♠IoT industrial; Prioridad a la interacción con el comportamiento;♦  
✓ Metodología para IoTS en general; 🗶Metodología para IoTS específicos

## Análisis de las metodologías existentes

El estudio de los requisitos del sistema es el primer paso en el desarrollo de un sistema. Por lo tanto, debería ser la primera fase de la metodología aplicada a su desarrollo. La **¡Error! No se encuentra el origen de la referencia.** muestra una comparación, centrada en el análisis de los requisitos, de las metodologías existentes, detallando todas las herramientas y modelos utilizados para el desarrollo de IoTS.

Analizando el estado del arte de las metodologías de desarrollo para IoTS, nos dimos cuenta de que algunos trabajos[10],[11] no mencionan los requisitos del sistema. Por consiguiente, estas metodologías no consideran la fase de análisis de requisitos. El resto de las metodologías revisadas coinciden en la importancia del análisis de necesidades para el desarrollo de un IoTS. La metodología presentada en [12] presenta el análisis de requerimientos en mayor profundidad, y presenta algunas herramientas que los desarrolladores pueden utilizar para recolectar y analizar los requerimientos. Mientras que las metodologías de [13]–[15] asumen que los requisitos están disponibles antes de que comience el desarrollo, las de [16],[17] consideran que los requisitos raramente están disponibles al principio del desarrollo de un IoTS. Por la experiencia de algunos desarrolladores en

1. Tabla 2. Comparación de metodologías

| Referencia | Análisis de requerimientos | Usar UML | Diagramas UML | | | | | | BPMN[[5]](#footnote-5) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Casos de uso | Actividades | Clases | Estados | Secuencias | Despliegue |
| [10] | 🗶 | ✓ | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| [11] | 🗶 | ✓ | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| [12] | ~ | ✓ | ✓ | ✓ | ✓ | 🗶 | 🗶 | 🗶 | ✓ |
| [13] | ~ | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| [14] | ~ | ✓ | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| [15] | ~ | ✓ | 🗶 | ✓ | ✓ | ✓ | 🗶 | 🗶 | 🗶 |
| [16] | ✓ | ✓ | ✓ | ✓ | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| [17] | ~ | ✓ | 🗶 | ✓ | ✓ | ✓ | ✓ | 🗶 | 🗶 |
| [18] | ✓ | ✓ | ✓ | ✓ | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| [19] | ✓ | ✓ | 🗶 | ✓ | ✓ | 🗶 | 🗶 | ✓ | 🗶 |
| [20] | ~ | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |
| [21] | ~ | ✓ | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 | 🗶 |

✓ Usado o considerado; ~ Mencionado; 🗶No especificado

La naturaleza de IoTS hace que sea importante considerar cuidadosamente todos los estados y transiciones del sistema, ya que el sistema tendrá que reaccionar a los eventos que ocurran en el entorno que controla. Este aspecto sólo es considerado por dos de las metodologías estudiadas; las otras no especifican nada al respecto. Además, los diagramas de despliegue, que son los únicos que pueden mostrar enlaces entre los componentes del sistema (basados en software y hardware), sólo se utilizan en una metodología.

Un enfoque que puede reducir y aliviar el trabajo de los desarrolladores es especificar el contexto del entorno que debe controlar el sistema IoTS: redes disponibles, calidad de servicio (QoS), niveles de privacidad, el entorno físico en el que se desplegará el sistema, así como las preferencias que pueda tener el usuario, tales como cambios estéticos y parámetros de accesibilidad. Esta especificación también puede ayudar a tomar decisiones sobre las tecnologías y herramientas específicas a utilizar. Estos aspectos se consideran importantes en el presente trabajo, aunque no están incluidos en ninguna de las metodologías revisadas.

Otro aspecto ampliamente olvidado del desarrollo de IoTS en la bibliografía revisada es el almacenamiento de información. Aunque la mayoría de los autores destacan la importancia del almacenamiento de información, asumen que está listo para ser utilizado al principio del desarrollo, a través de componentes de software existentes que no tienen ninguna importancia en el proceso de desarrollo. Sin embargo, consideramos que el tipo de base de datos (relacional, NoSQL,...) a utilizar debe ser cuidadosamente elegido durante el proceso de desarrollo, además de diseñar adecuadamente su estructura y seleccionar los recursos dedicados a su gestión (motor de base de datos, servidor local o en la nube, etc.).

Valoración de entregables

Mantenimiento

Implementación de la capa tecnológica

Refinamiento del software

Refinamiento de modelos

Generación de software

Generación de pruebas

Generación y montaje de modelos

Análisis detallado de las necesidades

Diseño de la capa tecnológica

Análisis preliminar

1. Figura 1. Fases del ciclo de vida de TDDM4IoTS

En consecuencia, hemos llegado a la conclusión de que las metodologías existentes tienen importantes deficiencias en cuanto a su aplicabilidad a los IoTS. Por ello, en el siguiente apartado, presentamos una nueva propuesta metodológica para abordar mejor los requisitos específicos de los IoTS.

# Una visión general de TDDM4IoTS

Presentamos una nueva metodología específicamente diseñada para el desarrollo de IoTS que integra ideas de las más destacadas metodologías de desarrollo de software, que intenta mitigar las debilidades encontradas en las metodologías revisadas. De hecho, se basa en las fases de la metodología TDD[22], aplicando los fundamentos del MDE y los principios de las metodologías ágiles. Por eso lo hemos denominado metodología de desarrollo basada en pruebas para sistemas basados en la IoT (TDDM4IoTS). Además, se hace hincapié en el uso de herramientas que, según los expertos, aseguran que el software cumple con los requisitos que el cliente ha proporcionado.

Proponemos TDDM4IoTS como una metodología independiente de las herramientas o marcos de automatización específicos, de modo que los desarrolladores tengan la libertad de elegir la(s) herramienta(s) adecuada(s) a utilizar, en función de sus necesidades y preferencias. No obstante, estamos trabajando en el desarrollo de una herramienta automatizada para soportar TDDM4IoTS en todas sus fases. Para la especificación de los requisitos, proponemos utilizar casos de uso en lugar de describirlos en lenguaje natural, que es el más habitual y el que los hace más propensos a errores y llenos de ambigüedades[16]. Uno de los objetivos de nuestra herramienta automatizada será reducir las ambigüedades a este nivel. Los casos de uso, junto con el modelo de clase conceptual, nos permitirán generar automáticamente tanto las pruebas como las partes del software que deben pasar esas pruebas. Los desarrolladores se centrarán en especificar y analizar los requisitos del sistema, así como en completar y adaptar el software generado automáticamente.

Las fases del ciclo de vida de TDDM4IoTS, mostradas en la Figura 1, han sido planteadas tratando de considerar el desarrollo de todos los tipos de IoTS. Por ello, el orden y la frecuencia de aplicación, así como la asignación de recursos para cada fase, dependerán tanto de la naturaleza del proyecto como de los conocimientos, habilidades, experiencia y tamaño de los miembros del proyecto. No obstante, el orden de aplicación sugerido en la Figura 1 (mostrado por los números) sería válido para el desarrollo de un buen número de IoTS.

Estas fases se repetirán de forma iterativa (para cada entrega). Sin embargo, en el desarrollo de algunos entregables, puede no ser necesario aplicar algunas de estas fases. Por ejemplo, podría no ser necesario realizar el análisis preliminar en una segunda iteración, o llevar a cabo el refinamiento del modelo en el desarrollo de un entregable dado. El equipo de desarrollo debe estimar los esfuerzos y la duración del proceso para obtener cada entregable o componente en el que normalmente se divide un proyecto. La prioridad (orden de desarrollo) de cada entregable se negociará con el cliente, a diferencia del marco de desarrollo de SCRUM, donde el equipo de desarrollo asigna prioridades de desarrollo a los entregables (Sprints)[23].

TDDM4IoTS requiere que la responsabilidad del facilitador del proyecto se asigne al miembro con mayor experiencia en la gestión de proyectos y que posea las características de un líder. Los desarrolladores que siguen TDDM4IoTS no están sujetos a imposiciones de tareas sino a negociaciones. El facilitador no es responsable de todo el proyecto, es sólo el gerente de la negociación entre los equipos de desarrollo. La responsabilidad del proyecto recae en cada uno de sus miembros. En consecuencia, TDDM4IoTS adopta un enfoque de gestión horizontal con responsabilidad compartida, dejando que los miembros del equipo se autoorganicen. La cantidad de desarrolladores dependerá del tamaño del proyecto, y considerando que sugerimos equipos"ágiles", no deben superar los diez[23],[24]. Los miembros del proyecto tienen diferentes responsabilidades, dependiendo de sus funciones, como se muestra en la Tabla 1. Cada equipo de desarrollo está formado por un máximo de tres desarrolladores, y debe estar equilibrado en términos de conocimientos y experiencia[24]. Uno de sus miembros será (no formalmente) designado como consejero, si es necesario.

En SCRUM, los participantes pueden desempeñar tres papeles principales, que se muestran en la **¡Error! No se encuentra el origen de la referencia.**, junto con sus responsabilidades. En los siguientes apartados se describe cómo aplicar los fundamentos en los que se basa TDDM4IoTS y sus fases, así como las actividades a realizar y las posibles herramientas a utilizar en cada una de ellas.

## Fundamentos de TDDM4IoTS

A continuación se describen los principales fundamentos en los que se basa TDDM4IoTS.

#### Valores y principios para el desarrollo ágil de software

El desarrollo ágil de software se rige por cuatro valores, a saber[25]-[26]: (i) Individuos e interacciones antes de procesos y herramientas, (ii) Software trabajando antes de documentación extensa, (iii) Colaboración con el cliente antes de la negociación contractual, y (iv) Respuesta al cambio antes de seguir un plan. Las metodologías ágiles han revolucionado el proceso de desarrollo de software, poniendo de manifiesto que los equipos de desarrollo que han completado sus proyectos a tiempo y con éxito no han respetado las metodologías rígidas y pesadas[27]. Para cumplir con estos valores, se detallan doce principios[25]-[26]. Estos son los principios que una metodología debe cumplir para ser calificada como una metodología ágil.

1. Tabla 1. Funciones y responsabilidades en TDDM4IoTS

| Rolea | **Descripción** | **Responsabilidades** |
| --- | --- | --- |
| Facilitador del proyecto | Experto con amplia experiencia en gestión de proyectos y en el desarrollo ágil de IoTS. Resolvedora de conflictos, formadora, facilitadora y con las características innatas de una líder[28],[29]. | Apoyar al equipo de desarrollo en el logro de sus objetivos.  Contribuir con su experiencia en la orientación para la planificación del desarrollo de los entregables.  Negociar con el cliente los aspectos del desarrollo de IoTS (orden de entrega, tiempo, recursos,...)b. |
| Consejero | Miembro del equipo de desarrollo que se convierte en "líder" (sin designación formal) por su desempeño. | Instruir a su compañero de equipo sobre los temas de su dominio. |
| Cliente / Usuario final | Persona con facilidad de palabra y conocimiento de toda la funcionalidad de IoTS que encarga al equipo de desarrollo el desarrollo de la misma. | Contribuir con los requisitos de los IoTS. Aprobar la funcionalidad de los entregables terminados y de los IoTS finales. |
| Equipo de desarrollo | Grupo multidisciplinar de expertos con conocimientos en los diferentes ámbitos del proyecto (comunicaciones, redes de sensores, desarrollo de software,...), responsable del desarrollo de los IoTS. Facilitadores de conocimiento y experiencia. | Negociar con el cliente los aspectos del desarrollo de IoTS (orden de entrega, tiempo, recursos,...)b.  Crear entregables que satisfagan plenamente los requisitos del cliente.  Mantenga a los clientes satisfechos. |

1. Dependiendo del proyecto, es posible que se necesiten expertos en otros campos para que presten sus servicios durante un tiempo específico.
2. Esta responsabilidad es compartida por dos roles.
3. Tabla 3. Roles y responsabilidades en SCRUM[25]-[27]

| Papel | Descripción | Responsabilidades |
| --- | --- | --- |
| Propietario del producto | Líder del proyecto, que es la única autoridad que puede especificar las funcionalidades del sistema. | Mantener y comunicar la visión del producto.  Equilibrar las necesidades y prioridades de los grupos de interés del lado del cliente. |
| Maestro SCRUM | Experto en desarrollo ágil, facilitador, coach y agente de cambio. | Eliminar los impedimentos para el progreso y la productividad del equipo.  Transferir al equipo sus conocimientos y experiencia ágiles. |
| Equipo de desarrollo | Grupo multifuncional, auto-organizado, emprendedor y adaptable. | Determinar el trabajo a realizar, crear entregables y decidir cuál es el siguiente entregable a desarrollar. |

A continuación se presentan los principios que rigen las metodologías de desarrollo ágil[25] aplicadas a TDDM4IoTS.

1. Lo más importante es satisfacer al cliente con entregas constantes y oportunas de software valioso.
2. Los cambios son bienvenidos, independientemente de que el desarrollo se retrase o no. Los procesos ágiles aprovechan el cambio como una ventaja competitiva para el cliente.
3. Las entregas de software en ejecución deben realizarse con frecuencia (cuanto más corto sea el intervalo de tiempo, mejor, máximo cada 2 meses).
4. El cliente debe colaborar con los desarrolladores en todo momento.
5. Los equipos de desarrolladores motivados tienen más éxito. Apóyelos en lo que necesitan y confíe en ellos para que hagan su trabajo.
6. Una conversación cara a cara es más efectiva para transmitir información hacia y desde el equipo de desarrollo.
7. El software de ejecución es la medida más importante del progreso.
8. Los procesos ágiles promueven el desarrollo sostenible. Los que participan en el desarrollo deben crear las condiciones para mantener un ritmo constante indefinidamente.
9. La búsqueda constante de la excelencia técnica y el buen diseño mejora la agilidad.
10. La simplicidad, es decir, la maximización del trabajo no realizado, es importante.
11. Las mejores arquitecturas, requerimientos y diseños surgen de equipos auto-organizados.
12. Los equipos reflexionan sobre cómo aumentar su efectividad en intervalos de tiempoestablecidos, y luego refinan y ajustan su comportamiento para lograrlo.

TDDM4IoTS cumple con estos doce principios, teniendo en cuenta que, como ya se ha dicho, un IoTS comprende el despliegue de una serie de dispositivos de hardware, además del software correspondiente. Por lo tanto, un entregable será un elemento terminado del sistema, obtenido como resultado de una iteración, que tiene cierta importancia para el cliente.

#### La DDT como metodología ágil.

TDD[30]-[32] es una de las metodologías utilizadas para el desarrollo de SI tradicionales que no se ha aplicado al desarrollo de IoTS. Como este tipo de metodología garantiza que el software del sistema desarrollado satisface los requisitos que el usuario ha proporcionado, lo hemos incorporado a nuestra propuesta. Hoy en día, muchos entornos de desarrollo integrado (IDE) soportan TDD, como IntelliJ IDEA, .Net o Eclipse, por nombrar sólo algunos[33].

La DDT, que se considera una metodología ágil, se aplica para asegurar la calidad de los resultados. Tiene tres fases en su ciclo de desarrollo: (i) Redacción de las pruebas que debe superar el software a desarrollar, aunque se espera que inicialmente fallen al ejecutarse; (ii) redacción del código requerido para que el sistema pase estas pruebas; y (iii) Refactorización del código para eliminar redundancias y mejorar su mantenibilidad[31]. Al escribir primero las pruebas y luego el código que pasa las pruebas, usted se asegura de que el software haga exactamente lo que el cliente quiere (es decir, las pruebas especifican formalmente los casos de uso). Esto intenta eliminar el desperdicio de recursos que implicaría la creación de software innecesario que los desarrolladores suponen que el cliente necesita[22].

Las metodologías de desarrollo de software ágil que se han considerado en esta investigación son XP (eXtreme Programming)[34] y SCRUM[23]. Nos interesa su forma de abordar los proyectos de desarrollo de software, y especialmente en estos aspectos: entrega de entregables funcionales en poco tiempo, aceptación de cambios en lugar de seguir un plan congelado, dar mayor importancia a los entregables que a una documentación exhaustiva, y la calidad del software (probado y aprobado por el cliente) es una de las características más importantes, entre otros principios de metodologías ágiles que deben ser cumplidas. XP es una de las metodologías que ha promovido la DDT[33]. Para hacer esto, primero se escriben las pruebas que el software debe pasar, luego se escribe el código de software para pasar estas pruebas y luego se refina el código. Por otro lado, SCRUM entrega software probado, es decir, software que ha pasado las pruebas, sin especificar si las pruebas se escriben antes o después del código de software. Sin embargo, separa muy bien las pruebas del desarrollo, ya que otro equipo (probador) prueba el software.

#### MDE.

Es un paradigma de Ingeniería de Software enfocado al desarrollo de software a partir de representaciones abstractas o modelos de un dominio en particular. Los modelos abstractos ayudan a entender la estructura y el comportamiento esperado del sistema sin necesidad de una implementación operativa. La heterogeneidad de las tecnologías y estándares de IoTS hacen de MDE una base importante para TDDM4IoTS. Un objetivo es reutilizar estos modelos (con o sin adaptaciones) en el desarrollo de otros sistemas, para lo cual estos modelos podrían convertirse en código ejecutable[35].

## Fases de TDDM4IoTS

Las fases de TDDM4IoTS están diseñadas para que las herramientas utilizadas por los desarrolladores en cada una de ellas sean de su elección. Lo que se hace es sugerir las herramientas que los desarrolladores pueden utilizar para cumplir los objetivos de cada fase. El primer desafío a superar es la comunicación entre los miembros del proyecto y entre los miembros de cada equipo de desarrollo. La mejor manera de lograr una comunicación efectiva es cara a cara, como se indica en el ágil manifiesto[26], considerando reuniones periódicas fijadas por los equipos ya sea al inicio del proyecto o al inicio del desarrollo del correspondiente entregable. La determinación de la frecuencia de estas reuniones es responsabilidad de todos los miembros del proyecto. Sin embargo, se recomiendan de tres a cinco reuniones por semana. Se deben buscar mecanismos para que los equipos de desarrollo se sientan motivados y cómodos con su trabajo, con la organización, con el resto de los miembros del proyecto y con su propio equipo, a fin de lograr un mejor desempeño[36].

#### Análisis preliminar.

El objetivo de esta fase es obtener un estudio de viabilidad del sistema (completo) en sus aspectos tecnológicos, económicos y operativos, así como un análisis del contexto en el que se desplegará el sistema, basado en los requisitos y objetivos expresados por el cliente. Las actividades que se llevarán a cabo podrán incluir:

* **Análisis de requerimientos.** Este análisis determina dos tipos de requisitos: (i) *funcionales*, que son las especificaciones del cliente (lista de entregables), determinando su prioridad de implementación, y (ii) *no funcionales*, también llamados *atributos de calidad*, tales como privacidad, escalabilidad, intrusismo, estética ambiental (despliegue, apariencia, etc.).
* **Análisis de tecnología.** Esto determina la tecnología que se va a utilizar y que cumple con los requisitos del sistema: (i) Recursos de hardware ya disponibles; (ii) Hardware existente, teniendo en cuenta sus características y costes; (iii) Herramientas (software) para la configuración del hardware; (iv) Herramientas de desarrollo de software; y (v) Desarrollo de nuevo hardware, si es necesario y factible.
* **Análisis del medio ambiente.** El cliente puede especificar las características del entorno en el que se desplegará el sistema. Por ejemplo, puntos de suministro de energía disponibles y/o viables, redes de comunicación de datos - acceso a Internet, métodos de interacción preferidos por el cliente, etc.
* **Análisis de viabilidad.** Entre los tipos de factibilidad a analizar se encuentran: i) *Viabilidad técnica*, para la que se responderían las preguntas: a) ¿Existen las tecnologías necesarias para desarrollar el proyecto? b) ¿Se dispone de personal capacitado para desarrollar el sistema? c) ¿Se puede desarrollar el sistema? (ii) *Viabilidad económica*, donde estaría la pregunta a responder: ¿Existe un presupuesto adecuado para desarrollar el proyecto? iii) La *viabilidad operacional*, que es importante para que el sistema siga funcionando después de su aplicación, por lo que es preciso responder a estas preguntas: a) ¿Será posible instalar el sistema una vez terminado? b) ¿Podrá funcionar el sistema con los recursos disponibles? c) ¿Existen las garantías necesarias para que el sistema siga funcionando una vez instalado? d) ¿Tendrán los botes un mantenimiento debidamente programado?

Dado que el sistema IoTS puede implicar el uso de múltiples tecnologías, la disponibilidad de cada una de ellas podría alterar el orden o las prioridades de los resultados. Por lo tanto, es necesario que esta primera fase se lleve a cabo de forma global al inicio del proyecto, y se revise al inicio del desarrollo de cada entregable, para considerar los cambios relacionados con la tecnología (nuevos dispositivos, nuevas herramientas, etc.) y los requisitos que puedan surgir durante el desarrollo del proyecto.

No existen herramientas específicas para alcanzar el objetivo de esta fase en particular. Sin embargo, para la planificación del proyecto, se sugiere utilizar una herramienta de software libre, como OpenProj, GanttProject, dotProject, y para aquellos que prefieren software propietario, MS-Project, entre muchos otros.

El resultado de esta fase es fundamental para las primeras negociaciones con el cliente, en cuanto a entregas, tiempos y presupuesto. Todo esto de manera preliminar, en ningún momento puede ser definitivo, ya que puede ser negociado entre ambas partes al inicio del desarrollo de cada entregable.

#### Diseño de la capa tecnológica.

El objetivo de esta fase es obtener el primer diseño del sistema global que sirva de guía para los equipos de desarrollo. Esto es muy importante, dado que la IoT implica tecnologías emergentes y heterogéneas, y hasta ahora es difícil encontrar un profesional que pueda dominar todas las tecnologías implicadas en el desarrollo de un IOTS[21].

Para el diseño del sistema, se pueden utilizar herramientas de diseño de circuitos que puedan representar de la forma más clara posible los elementos que se han determinado para el proyecto. Por ejemplo, si se utilizan placas Arduino, se pueden utilizar herramientas en línea como Circuito.io o Fritzing. El equipo de desarrollo puede complementar los diseños obtenidos. Si es necesario, el diseño resultante puede ser actualizado al final de cada entrega. Este será uno de los documentos de mayor interés para todos los equipos de desarrollo del proyecto. Por lo tanto, siempre debe ser accesible para todos ellos.

En esta fase se espera definir la arquitectura con la que se implementará el sistema. Por consiguiente, el resultado de esta fase servirá de guía para todo el proceso de desarrollo.

#### Análisis detallado de los requisitos.

El objetivo de esta fase, que se ejecutará para cada entregable delsistema, es obtener los requisitos detallados del entregable a desarrollar. Además, se le pedirá al cliente que describa las pruebas junto con los desarrolladores, para reducir las ambigüedades. Para la especificación de los requisitos, se recomienda utilizar herramientas comprensibles para todos los miembros del proyecto, considerando al cliente como uno de ellos. Una de las notaciones a utilizar podría ser UML, con sus diferentes herramientas, como casos de uso y casos de uso semiestructurados, utilizando plantillas preestablecidas, buscando eliminar ambigüedades, además de otras herramientas, como diagramas de estado y de despliegue, que ayuden a entender los requerimientos proporcionados por el cliente.

#### Generación y adaptación de modelos.

El objetivo de aplicar MDE en TDDM4IoTS es reutilizar modelos y mejorar así la productividad del equipo de desarrollo. Por lo tanto, en esta fase se generarán nuevos modelos o se adaptarán los modelos existentes. El uso de modelos abstractos o al menos minimiza los aspectos de heterogeneidad de las tecnologías, ayudando a una buena comunicación entre los equipos de desarrollo y los clientes (usuarios finales). Además, dependiendo de las herramientas de software utilizadas para el modelado, el software puede ser generado automáticamente a través de modelos, desde modelos abstractos hasta modelos específicos[10],[18]. Uno de los modelos es el diagrama de clases, que se utiliza para generar la base de datos.

Los lenguajes de modelado sugeridos son: UML, BPMN[12], la adaptación de uno de ellos[14], su combinación o la creación de uno nuevo, buscando siempre cubrir las particularidades de los IoTS en cuestión y ser fácilmente entendidos por los involucrados en el proyecto. Entre las herramientas automatizadas para el modelado de sistemas se encuentran StarUML, ArgoUML, MagicDraw y Visual Studio.net, por ejemplo. Para aquellos que opten por BPMN, pueden seleccionar herramientas como Lucidchart o VisualParadigm, entre muchas otras.

#### Generación de pruebas.

TDDM4IoTS sigue el paradigma TDD, por lo que debe generar las pruebas que el software debe superar para asegurar la calidad del sistema. Las pruebas se pueden agrupar en dos grupos: (1) Las pruebas escritas por los desarrolladores, dentro de las cuales hay: a) pruebas unitarias, que son las más exhaustivas, para examinar el funcionamiento completo de una función, es decir, se comprueba si la función produce los resultados que debería producir e incluso si admite las excepciones que puedan surgir, y b) pruebas de integración, que también incluyen pruebas de sistemas. Y (2) pruebas documentadas por el cliente, que son básicamente pruebas de aceptación, incluyendo pruebas funcionales[22].

Las herramientas automatizadas para esta fase dependerán del IDE que se haya seleccionado para el desarrollo, aunque hasta ahora ninguna herramienta genera automáticamente pruebas basadas en los requisitos.

#### Generación de software.

Esta fase se basa en modelos y pruebas. El desarrollador debe escribir/generar el código de las pruebas a superar[31],[37]. Los nuevos modelos generados y/o los modelos existentes adaptados forman parte de la documentación del sistema. Por lo tanto, a diferencia de las metodologías SCRUM y XP, que sugieren que la documentación se escriba al final del desarrollo de cada entregable (análisis y diseño de la solución)[27], TDDM4IoTS propone hacerlo antes de la generación de código, ya que existen herramientas que, desde los modelos, ayudan en esta tarea. Además, al realizar encuestas a los desarrolladores de software que trabajan o han trabajado con la metodología SCRUM y/o XP, la mayoría está de acuerdo en que la solución que se implementará más tarde debe al menos esbozarse al principio del ciclo de vida de la metodología. Esto corrobora lo expuesto en TDDM4IoTS en cuanto al análisis y diseño de la solución, que debe llevarse a cabo a medida que avanza el desarrollo de los entregables.

Una vez generado el software, se comprueba que pasa las pruebas correspondientes, y luego finaliza esta fase. El resultado de esta fase es el software probado y funcionando, aunque casi siempre tendrá que ser refinado más tarde.

Para generar el código se pueden utilizar algunas de las herramientas mencionadas en la fase de generación y refinamiento del modelo. De hecho, existen varias herramientas, tanto de software libre como de software propietario, que permiten incluso simular el comportamiento del sistema.

#### Refinamiento del modelo.

Los modelos UML facilitan el refinamiento del modelo. Tenga en cuenta que la solución encontrada es una solución funcional, pero no necesariamente una solución óptima. Por lo tanto, se debe considerar hacer los ajustes y refinamientos necesarios, tales como mejorar la robustez, escalabilidad o reutilización de los entregables que conforman los IoTS a desarrollar[38]. El resultado de esta fase será el modelo definitivo, en base al cual se generará el código del sistema.

Las herramientas sugeridas para realizar las tareas de esta fase son las mismas que en la fase de generación y adaptación del modelo.

#### Refinamiento de software.

El trabajo de los desarrolladores en esta fase será garantizar la calidad del software, eliminando redundancias y haciendo que el software sea fácil de mantener. Las herramientas que apoyarán esta actividad serán las proporcionadas por el IDE elegido. Debe asegurarse de que el software final cumple las especificaciones de un código limpio[32].

#### Despliegue de hardware y capas de almacenamiento.

Una vez que el software ha sido probado (simulado), se implementa e implementa en los dispositivos y recursos a utilizar en el sistema, confirmando el cumplimiento de los requisitos finales negociados entre el cliente y el equipo de desarrollo. En este punto, y para el primer entregable, ya se habrá configurado el sistema de almacenamiento de información, así como instalado las aplicaciones que servirán para la interacción usuario-sistema. Para las entregas posteriores, se realizarán los cambios necesarios en esta infraestructura ensamblada para añadir las nuevas entregas. Dado que los resultados posteriores dependen de la tecnología ya instalada, debe garantizarse el funcionamiento del (nuevo) sistema integrado antes de continuar el proceso de desarrollo.

Las herramientas que se utilizarán en esta fase dependerán de la tecnología utilizada (ordenadores de placa única, sensores integrados,...).

#### Valoración del entregable.

Una vez finalizado el desarrollo del entregable (tuvo que pasar las pruebas necesarias para garantizar su funcionamiento), las pruebas de integración, las pruebas del sistema y, por supuesto, las pruebas funcionales deben volver a realizarse en esta fase.

## La comunicación como instrumento esencial

La comunicación es el primer desafío que debe superarse en el desarrollo de un proyecto, ya que la comunicación efectiva es un elemento clave para la finalización exitosa de un proyecto[39].

La toma de decisiones en equipo es fundamental para el éxito de un proyecto. Con TDDM4IoTS, buscamos involucrar a los miembros del proyecto en la toma de decisiones. De hecho, deben decidir desde la planificación de las reuniones periódicas hasta las herramientas de desarrollo a utilizar, e incluso los recursos y el tiempo asignados a cada uno de los productos.

#### Sugerencias para la comunicación. Debido a la importancia que tiene la comunicación para el éxito de un proyecto, es conveniente que los miembros del proyecto conozcan los medios formales/informales (correo electrónico, aplicación de mensajería móvil,...) que se utilizarán para comunicarse entre sí o informar sobre algo. Además, es aconsejable utilizar un pizarrón (físico o virtual) para el desarrollo diario del proyecto. SCRUM recomienda usar pizarras físicas para mantener a todos informados sobre los procesos involucrados en la metodología (es parte de ella).

#### Personal motivado y dispuesto. Para conocer el grado de motivación de los miembros del proyecto, se sugieren dos herramientas que pueden utilizarse para medirlo. (1) Un buzón de sugerencias, que es una herramienta de uso frecuente para medir la satisfacción del cliente; sin embargo, puede ser utilizado para medir el grado de satisfacción y/o motivación que siente el equipo. Y (2) una pizarra disponible para que los miembros del equipo dibujen en notas de papel un emoticono con el que calificar su día de trabajo. El líder del proyecto debe canalizar la solución de cada sugerencia o calificación negativa tan pronto como sea posible. La ausencia de calificación también debe considerarse negativa. Además, hay reuniones periódicas en las que el líder debe estar atento a la conducta de los miembros de su equipo. Dentro de equipos pequeños, es factible tomar decisiones rápidamente y animar a quien sea a mejorar la motivación y el rendimiento de todo el equipo.

#### 3.3.3 Reuniones de trabajo periódicas.

Los miembros del proyecto establecen la periodicidad y el calendario de las reuniones, excepto en situaciones particulares. Las decisiones se tomarán democráticamente. Aunque la periodicidad de las reuniones debe establecerse al principio del proyecto, se recomienda utilizar el pizarrón para especificar también la hora preferida para la próxima reunión.

#### 3.3.4 Información del proyecto.

Los documentos del proyecto deben ser accesibles a todos los miembros, sin excepción (en un servidor de archivos), y más específicamente, el diseño de las redes de sensores, los diagramas de despliegue, la arquitectura del sistema (global), así como el progreso de ambos entregables y el proyecto deben ser visibles para todos (se pueden utilizar los pizarrones de la metodología SCRUM).

# Conclusiones y trabajos futuros

Tras una revisión exhaustiva de las metodologías utilizadas para el desarrollo de IoTS, se ha descubierto que no existe una metodología estándar para este ámbito de aplicación. Este hecho ha llevado a los desarrolladores de IoTS a utilizar metodologías diseñadas para desarrollar otros tipos de sistemas informáticos y a hacer ajustes ad-hoc para satisfacer las necesidades particulares de cada proyecto. Además, el uso de estas metodologías no específicas para el desarrollo de los sistemas IoTS ha pasado por alto la necesidad de especificar aspectos importantes en estos sistemas, como el análisis de los requisitos y las particularidades del hardware que se va a desplegar.

En consecuencia, se ha propuesto una nueva metodología, denominada TDDM4IoTS, para abordar específicamente el desarrollo de los IoTS, que incluye los fundamentos más importantes de TDD, MDE y desarrollo ágil, y trata de resolver cada uno de los aspectos que se han detectado como debilidades en las metodologías revisadas que se han utilizado para el mismo fin.

Uno de nuestros próximos trabajos será validar TDDM4IoTS, para conocer su efectividad y aceptación. Además, se desarrollará una herramienta para apoyar la generación automatizada de los entregables correspondientes a cada fase de TDDM4IoTS.

##### RECONOCIMIENTOS

A la Universidad Técnica Estatal de Quevedo, por la financiación de esta investigación, a la Universidad de Granada, por aportar los recursos tecnológicos, y a los desarrolladores que cumplimentaron una encuesta sobre el desarrollo de IoTS y sus experiencias profesionales.

##### Referencias

1] O.-J. Dahl y C. A. R. Hoare, Capítulo III: Estructuras jerárquicas de programas. Academic Press Ltd., 1972.

[2] D. H. H. Ingalls, "The Smalltalltalk-76 programming system design and implementation", en Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages - POPL '78, 1978, pp. 9-16.

[3] A. Pashtan, "Sistemas operativos orientados a objetos: Una metodología de diseño emergente", en Proceedings of the ACM '82 conference on - ACM 82, 1982, pp. 126-131.

[4] D. Schwabe y G. Rossi, "The object-oriented hypermedia design model," Commun. ACM, vol. 38, no. 8, pp. 45-46, agosto de 2002.

[5] K. Ashton, "That `Internet of Things' thing," RFID J., vol. 22, no. 7, pp. 97-114, 2009.

[6] P. P. Ray, "A survey on Internet of Things architects", J. King Saud Univ. - Comput. Inf. Sci., vol. 30, no. 3, pp. 291-319, julio de 2018.

[7] M. Leotta y otros, "An acceptance testing approach for Internet of Things systems", IET Softw., vol. 12, no. 5, pp. 430-436, octubre de 2018.

[8] J. I. Benedetto, L. A. González, P. Sanabria, A. Neyem y J. Navón, "Towards a practical framework for code offloading in the Internet of Things", Futur. Generar. Comput. Syst., vol. 92, págs. 424-437, marzo de 2019.

[9] J. W. Cervantes-Solis, C. Baber, A. Khattab y R. Mitch, "Rule and theme discovery in human interactions with an'Internet of Things,'" en Proceedings of the 2015 British HCI Conference on - British HCI `15, 2015, pp. 222-227.

[10] F. Ciccozzi y R. Spalazzese, "MDE4IoT: Apoyar el Internet de los objetos con ingeniería basada en modelos", Stud. Comput. Intell. vol. 678, págs. 67-76, 2017.

[11] H. Khaleel y otros, "Heterogeneous applications, tools, and methodologies in the car manufacturing industry through an IoT approach," IEEE Syst. J., vol. 11, no. 3, pp. 1412-1423, Sep. 2017.

[12] H. Cai, Y. Gu, A. V. Vasilakos, B. Xu y J. Zhou, "Model-driven development patterns for mobile services in Cloud of Things", IEEE Trans. Cloud Comput., tomo 6, no. 3, pp. 771-784, julio de 2018.

[13] A. Lekidis, E. Stachtiari, P. Katsaros, M. Bozga y C. K. Georgiadis, "Model-based design of IoT systems with the BIP component framework," Softw. - Pract. Exp., vol. 48, no. 6, pp. 1167-1194, Jun. 2018.

[14] M. Brambilla, E. Umuhoza y R. Acerbis, "Model-driven development of user interfaces for IoT systems via domain-specific components and patterns," J. Internet Serv. Appl., vol. 8, no. 1, p. 14, diciembre de 2017.

[15] A. Harbouche, N. Djedi, M. Erradi, J. Ben-Othman y A. Kobbane, "Model driven flexible design of a wireless body sensor network for health monitoring," Comput. Networks, vol. 129, págs. 548-571, diciembre de 2017.

[16] T. Usländer y T. Batz, "Agile service engineering in the Industrial Internet of Things", Futur. Internet, vol. 10, no. 10, p. 100, octubre de 2018.

[17] P. Pico-Valencia, J. A. Holgado-Terriza, y P. Paderewski, "A systematic method for building internet of agents applications based on the linked open data approach," Futur. Generar. Comput. Syst. vol. 94, pp. 250-271, mayo de 2019.

[18] C. M. Sosa-Reyna, E. Tello-Leal y D. Lara-Alabazares, "Methodology for the model-driven development of service oriented IoT applications," J. Syst. Archit. vol. 90, págs. 15-22, octubre de 2018.

[19] C. M. de Farias y otros, "COMFIT: A development environment for the Internet of Things", Futur. Generar. Comput. Syst., vol. 75, pp. 128-144, octubre de 2017.

20] G. Fortino y otros, "Towards multi-layer interoperability of heterogeneous IoT platforms: The INTER-IoT approach", Internet of Things, no. 9783319612997, pp. 199-232, 2018.

[21] P. Patel y D. Cassou, "Enabling high-level application development for the Internet of Things", en Journal of Systems and Software, 2015, vol. 103, pp. 62-84.

[22] R. C. Martin, "Clean coder blog", TDD, 2017. .

[23] L. Rising y N. S. Janoff, "Scrum software development process for small teams", IEEE Softw. vol. 17, no. 4, pp. 26-32, 2000.

[24] A. Holzinger, M. Errath, G. Searle, B. Thurnher y W. Slany, "From extreme programming and usability engineering to extreme usability in software engineering education (XP+UE→XU)" (De la programación extrema y la ingeniería de usabilidad a la usabilidad extrema en la enseñanza de la ingeniería de software), en Proceedings - International Computer Software and Applications Conference (Actas - Conferencia internacional sobre software y aplicaciones informáticas), 2005, vol. 2, pp. 169-172.

[25] K. Beck y otros, "Manifiesto para el desarrollo ágil de software", Manifiesto para el desarrollo ágil de software, 2001. .

[26] M. Fowler y J. Highsmith, "The Agile Manifesto", 2001.

[27] O. Hazzan e Y. Dubinsky, "The Agile Manifesto", en SpringerBriefs in Computer Science, no. 9783319101569, 2014, pp. 9-14.

[28] R. T. Mowday, "Características del líder, confianza en sí mismo y métodos de influencia ascendente en situaciones de decisión organizacional", Acad. Manag. J., vol. 22, no. 4, págs. 709-725, diciembre de 1979.

[29] H. Koo y C. Park, "Foundation of leadership in Asia: Leader characteristics and leadership styles review and research agenda", Asia Pacific Journal of Management, vol. 35, no. 3. Springer US, pp. 697-718, Sep-2018.

[30] A. Tort, A. Olivé y M. R. Sancho, "An approach to test-driven development of conceptual schemas", Data Knowl. Eng., vol. 70, no. 12, pp. 1088-1111, 2011.

[31] D. Janzen y H. Saiedian, "Test-driven development: Concepts, taxonomy, and future direction," Computer (Long. Beach. Calif), vol. 38, no. 9, pp. 43-50, Sep. 2005.

[32] R. C. Martin, código limpio. A handbook of agile software craftsmanship, 1ª edición. Boston: Pearson Education, Inc, 2011.

[33] L. Madeyski y M. Kawalerowicz, "Continuous test-driven development: A preliminary empirical evaluation using agile experimentation in industrial settings", en Studies in Computational Intelligence, vol. 733, Kosiuczenko P.; y Madeyski L., Eds. Springer, Cham, 2018, págs. 105-118.

[34] K. Braithwaite y T. Joyce, "XP expanded: Distributed extreme programming", Springer, Berlín, Heidelberg, 2005, pp. 180-188.

[35] C. M. Sosa-Reyna, E. Tello-Leal y D. Lara-Alabazares, "An approach based on model-driven development for IoT applications", en Proceedings - 2018 IEEE International Congress on Internet of Things, ICIOT 2018 - Part of the 2018 IEEE World Congress on Services, 2018, pp. 134-139.

[36] R. H. Rasch y H. L. Tosi, "Factores que afectan al rendimiento de los desarrolladores de software: Un enfoque integrado", MIS Q., vol. 16, no. 3, pág. 395, septiembre de 1992.

[37] C. Blé Jurado, J. M. Beas, J. Gutiérrez Plaza, F. Reyes Perdomo, and G. Mena, Diseño agil con TDD, 1st ed., vol. 1. iExpertos.com, 2010.

[38] Z. Chen, Z. Liu, A. P. Ravn, V. Stolz y N. Zhan, "Refinement and verification in component-based model-driven design," Sci. Program, vol. 74, no. 4, pp. 168-196, febrero de 2009.

[39] J. Pollack y P. Matous, "Testing the impact of targeted team building on project team communication using social network analysis," Int. J. Proj. Manag., vol. 37, no. 3, págs. 473-484, abril de 2019.

Pi

1. Documento más antiguo publicado en 1964. [↑](#footnote-ref-1)
2. Documento más antiguo publicado en 1974. [↑](#footnote-ref-2)
3. <http://wos.fecyt.es> [↑](#footnote-ref-3)
4. LANGUAGE: (English) ANDDOCUMENT TYPES: (Article OR Book OR Book Chapter OR Book Review OR Editorial Material OR Review) *Indexes=SCI-EXPANDED, SSCI, A&HCI, CPCI-S, CPCI-SSH, BKCI-S, BKCI-SSH, ESCI, CCR-EXPANDED, IC Timespan=All years* [↑](#footnote-ref-4)
5. Modelo de procesos de negocio y notación [↑](#footnote-ref-5)