SEP TNM

# INSTITUTO TECNOLÓGICO DE CULIACÁN


## METODOLOGÍA DE DESARROLLO DE APLICACIONES DEL INTERNET DE LAS COSAS (MeDAIC)

TESIS

PRESENTADA ANTE EL DEPARTAMENTO ACADÉMICO DE ESTUDIOS DE POSGRADO DEL INSTITUTO TECNOLÓGICO DE CULIACÁN EN CUMPLIMIENTO PARCIAL DE LOS REQUISITOS PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

POR:

MIGUEL ANGEL MONTOYA DEL CAMPO
LICENCIADO EN INFORMATICA

DIRECTOR DE TESIS:

DR. RICARDO RAFAEL QUINTERO MEZA

CULIACÁN, SINALOA 26 DE MARZO DE 2019

**"2019, Año del Caudillo del Sur, Emiliano Zapata"**

Culiacán, Sin., 06 de Junio del 2019

**DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN**
OFICIO: DEPI:307/VI/2019

ASUNTO: **Autorización Impresión**

**LIC. MIGUEL ANGEL MONTOYA DEL CAMPO**
**ESTUDIANTE DE LA MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**
**PRESENTE.**

Por medio de la presente y en virtud de que ha completado los requisitos para el examen de grado de la **Maestría en Ciencias de la Computación**, se concede autorización para la impresión de la tesis titulada: **"METODOLOGÍA DE DESARROLLO DE APLICACIONES DEL INTERNET DE LAS COSAS" (MeDAIC),** bajo la dirección del(a) **Dr. Ricardo Rafael Quintero Meza**

Sin otro particular reciba un cordial saludo.

**ATENTAMENTE**
*"CON LA TÉCNICA AL PROGRESO"*

**M.C. MARÍA ARACELY MARTÍNEZ AMAYA**
**JEFE(A) DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN**

SEP TecNM
Instituto Tecnológico de Culiacán
División de Estudios de Posgrado e Investigación

C.c.p. archivo

MAMA/lucy *

# "METODOLOGÍA DE DESARROLLO DE APLICACIONES DEL INTERNET DE LAS COSAS" (MEDAIC)

Tesis presentada por:

**LIC. MIGUEL ANGEL MONTOYA DEL CAMPO**

Aprobada en contenido y estilo por:

Dr. Ricardo Rafael Quintero Meza
**Director de Tesis**

Dra. María Lucía Barrón Estrada
Secretario

M.C. Gloria Ekaterine Peralta Peñuñuri
Vocal -1

Dr. Ramón Zatarain Cabada
Vocal -2

M.C. María Aracely Martínez Amaya
Jefe(a) de la División de Estudios de Posgrado e Investigación

# Agradecimientos

Agradezco a mis padres que siempre me han apoyado durante toda mi vida, ustedes que fueron mis primeros maestros y que me enseñaron a recorrer este mundo lleno de obstáculos enfrentando cada situación difícil con valentía y honor, ustedes que me impulsaron a salir adelante brindándome su apoyo incondicional.

A mis compañeros y amigos de maestría: Sandra Lucia Ramírez Ávila, Aldo Uriarte Portillo, Jesús Armando Guerra García y Giovanni Manjarrez Montelongo con quienes tuve la suerte de compartir la experiencia de estudiar una maestría y quienes lograron convertirse en algo más que simples compañeros de clase, sino que lograron convertirse en mi familia por elección.

A mis maestros: Dra. Lucia Barrón Estrada, Dr. Ramón Zatarain Cabada, Dr. Héctor Rodríguez Rangel, MC. Gloria Ekaterine Peñuñuri Peralta, y en especialmente a mi director de tesis a quien le tengo una especial admiración y respeto Dr. Ricardo Rafael Quintero Meza. Gracias porque sin ustedes este logro no hubiera podido haberse vuelto realidad ya que, gracias a su paciencia, sus enseñanzas y su ejemplo lograron hacer de mí una persona más consciente y capaz de enfrentar los nuevos retos que me arroje la vida.

Al Ingeniero Francisco Javier Mosqueda Alarcón quien gracias a su apoyo hizo posible que pudiera cumplir con esta meta de vida y que espero seguir contando con su respaldo para futuros proyectos que serán igual de relevantes en mi vida.

Por último, al Consejo Nacional de Ciencia y Tecnología (CONACyT), por financiar mis estudios de posgrado.

# Palabras clave

- Internet de las cosas
- Autonomia
- Modernidad
- Telecomunicaciones
- Metodologías
- Desarrollo de software
- Ingenieria de software
- Proyectos
- Desarrollo ágil
- Modelado
- Estructura de datos
- Algoritmos
- Diagramas de flujo
- Organización de proyectos
- Seguridad
- Trabajo en equipo

# Declaración de autenticidad

Por la presente declaro que, salvo cuando se haga referencia específica al trabajo de otras personas, el contenido de esta tesis es original y no se ha presentado total o parcialmente para su consideración para cualquier otro título o grado en esta o cualquier otra Universidad. Esta tesis es resultado de mi propio trabajo y no incluye nada que sea resultado de algún trabajo realizado en colaboración, salvo que se indique específicamente en el texto.

Miguel Angel Montoya Del Campo.

Culiacán, Sinaloa, México, 2019

# Resumen

El Internet De Las Cosas (IoT por sus siglas en inglés) ha cobrado gran auge e importancia en los diferentes sectores de nuestra sociedad afectando directamente al sector económico, productivo y social. Esto ha provocado un incremento en el desarrollo de Software especializado orientado al IoT exponiendo con ello algunas deficiencias muy marcadas en el ámbito de la administración y gestión de proyectos de este tipo.

El presente trabajo de investigación se enfoca en una de las mencionadas deficiencias como lo es la selección de la metodología más apropiada para el desarrollo de un proyecto orientado al IoT que contemple las etapas críticas, los artefactos necesarios en cada una de estas etapas, así como la distribución de responsabilidades de cada uno de los involucrados en dicho proyecto. A continuación, se hablará de la Metodología de Desarrollo de Aplicaciones del Internet de las Cosas (MeDAIC por sus siglas) la cual se presenta como un marco de desarrollo de software que consta de ocho niveles en los cuales se describen las acciones, responsabilidades y compromisos que se deben asumir durante el desarrollo de dicho proyecto IoT.

Los procesos que conforman la metodología MeDAIC son: Definición de propósito y requisitos de alto nivel, Refinamiento de los requisitos de alto nivel, Especificación del modelo de información de entidades físicas y virtuales, Definición del formato para el intercambio de datos, Visión general de la arquitectura, Construcción, Pruebas y validación, Desarrollo del diagrama de despliegue e Implantación del sistema.

Todos estos elementos se integran en la documentación del proyecto y se enriquece con diagramas de línea de tiempo, artefactos, tipos de usuario, casos de uso y una serie de artefactos que intervienen en cada uno de los procesos de implementación de MeDAIC con la finalidad de darle orden y sentido a cada uno de los niveles que conforman la metodología.

Los resultados obtenidos en los experimentos fueron muy reveladores y satisfactorios ya que MeDAIC fue implementado en 3 proyectos productivos de la iniciativa privada, se establecieron una serie de métricas las cuales fueron evaluadas imparcialmente buscando siempre que los resultados fueran reales y evitando el sesgo.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# 1. Introducción

El desarrollo de aplicaciones y sistemas de software ha prevalecido desde el año 1940 a la fecha. A lo largo de la historia fueron surgiendo nuevas técnicas, estilos y patrones de diseño en la programación que fueron enriqueciendo esta disciplina dando como resultado que en la actualidad exista una gran variedad de formas de organizar, documentar e implementar el desarrollo de proyectos de software. Además, se han desarrollado herramientas que apoyan al programador en la organización de sus proyectos brindando la posibilidad de trabajar colaborativamente a través del uso de repositorios locales o bien mediante la implementación de repositorios remotos dándole versatilidad y dinamismo a los desarrollos y permitiendo incorporar a diversos tipos de participantes a los proyectos como lo son: programadores, diseñadores gráficos, administradores de proyectos, expertos de dominio entre otros.

La ciencia y la tecnología avanzan rápidamente y el desarrollo de sistemas software no es la excepción. Así lo demuestran las demandas que presentan las nuevas Tecnologías de la Información y Comunicación (TIC), el desarrollo de aplicaciones empresariales o bien el desarrollo de aplicaciones orientadas al IoT. Dichos avances plantean nuevos retos en el área de la Ingeniería de Software en materia de organización y construcción de proyectos ya que en la actualidad se siguen utilizando técnicas y metodologías de desarrollo de software que fueron creadas ya hace varias décadas.

Actualmente el desarrollo de software orientado al IoT ha cobrado gran notoriedad derivado de los avances tecnológicos aunados a la creciente necesidad de ser humano por controlar su entorno mediante el uso de las nuevas tecnologías por ello es importante incrementar los esfuerzos por actualizar y mejorar las técnicas de gestión y ejecución de proyectos.

En el presente trabajo de tesis se presenta una propuesta de marco de desarrollo denominado MeDAIC con la cual se pretende cambiar el paradigma de la programación de sistemas orientados al IoT ya que en dicho marco de trabajo se contemplan las particularidades que intervienen en proyectos de esta naturaleza.

## 1.1. Descripción del problema

En el mundo del desarrollo de software se suelen utilizar marcos de trabajo (*framework*) que proporcionan una serie de pasos bien definidos para la correcta construcción de los sistemas de software. Dichos marcos de trabajo empezaron a surgir en la década de 1960 donde se utilizaban en desarrollos de software a gran escala. El objetivo que se perseguía era el de desarrollar los sistemas de información en una muy deliberada y estructurada metodología reiterando cada una de las etapas del ciclo de vida en cada uno de los proyectos de desarrollo de software que se presentara.

Las metodologías de desarrollo de software presentan un conjunto de técnicas tradicionales y modernas de modelado de sistemas que permiten desarrollar software de calidad incluyendo heurística de construcción y criterios de comparación de modelos de sistemas. Para tal fin se describen, fundamentalmente, herramientas de análisis y diseño orientado a objetos por ejemplo el Lenguaje de Modelado Unificado (UML por sus siglas en inglés) sus diagramas, especificación y criterios.

Actualmente la iniciativa privada requiere de una gran cantidad de desarrollos de software orientados al IoT y se busca que éstos se intercomuniquen entre sí o bien puedan ser administrados a distancia mediante el uso de las nuevas tecnologías a través de internet, con esto se puede inferir claramente que los requisitos funcionales y de calidad presentan nuevos retos ya que al momento de diseñar dichos sistemas se deben considerar una serie de variantes que en el pasado no figuraban, como es el caso de los sensores, actuadores, relevadores o bien maquinaria especializada capaz de ser controlada mediante sistemas remotos a través de dispositivos móviles.

Los sistemas IoT incorporan estos elementos dentro de su funcionalidad ya que son capaces de controlar distintos tipos de terminales físicas mediante el uso de dispositivos inteligentes aprovechando la conectividad que ofrece el Internet. Si bien es cierto que cada vez se presenta más interacción humana a través de los dispositivos móviles y existe un gran auge en el desarrollo de aplicaciones para este tipo de dispositivos, también es cierto que los programadores utilizan las mismas técnicas de desarrollo de software con las que se formaron hace ya varias décadas, es decir, que los métodos de desarrollo siguen siendo artesanales y en ocasiones obsoletos al momento de enfrentar un requerimiento IoT no

quedando otra alternativa por parte del programador más que la de tratar de resolver la problemática mediante el uso de sus conocimientos refinados a través de la experiencia apegándose a los paradigmas de la programación con los que se formó.

En materia de metodologías de desarrollo de software existe una amplia variedad de paradigmas que apoyan al desarrollador en la administración y estructuración de sus proyectos. Hay paradigmas tradicionales, paradigmas orientados a objetos y paradigmas ágiles. Sin embargo, en ninguno de estos paradigmas se contempla abiertamente el concepto de desarrollo de aplicaciones IoT provocando que los desarrolladores tomen alguna de estas propuestas y la adecuen a sus proyectos dando como resultado que un sistema IoT sea comparable con cualquier otro sistema no IoT siendo esta la causa de que los desarrolladores programen los sistemas y aplicaciones orientados al IoT de manera tradicional y así brindarle el tratamiento adecuado a los circuitos electrónicos, relevadores y actuadores que intervienen en un proyecto orientado al IoT representándolos solo como simples objetos.

Por lo anterior, en este documento de tesis se presenta una metodología de desarrollo de software que es inclusiva con las nuevas tecnologías para que desde la etapa de planeación de proyectos sean visibles las tecnologías del IoT que intervendrán en el sistema, considerando sus características particulares, sus tiempos de generación de información así como la forma en que dichos dispositivos interactúan con su entorno.

## 1.2. Justificación

En la actualidad los desarrollos de software orientados al IoT carecen de una metodología de desarrollo propia que contemple las características particulares de un proyecto de esta naturaleza en donde intervienen de forma directa una serie de circuitos eléctricos y electrónicos los cuales fungen una labor medular durante el tiempo de vida del software, es por ello que en el presente trabajo de tesis se propone una alternativa de solución que contemple aquellos artefactos y elementos antes mencionados desde el momento de la planeación hasta la instalación y puesta a punto del sistema que se desarrolle con estas características.

### 1.2.1. Pertinencia

Normalmente los sistemas de desarrollo de software orientados al IoT son programados utilizando metodologías tradicionales que surgieron en la época donde no existían los teléfonos inteligentes, tablet ni cualquier otro tipo de tecnología que hoy en día se considera como moderna. Se ha identificado que existe un área de oportunidad al proponer una metodología de desarrollo que cumpla con los requerimientos necesarios que exige un sistema orientado al IoT ya que actualmente no existe en el mercado alternativa alguna que ofrezca lo que se plantea.

### 1.2.2. Relevancia

Como ya se ha mencionado en párrafos anteriores existe una problemática en el rubro del desarrollo de software y esta es que no existe una metodología Ad Hoc que cumpla con los requerimientos específicos de un proyecto orientado al IoT es por ello que en el presente trabajo de tesis se propone a MeDAIC como una alternativa de solución a esta problemática ya que esta metodología contempla aquellos elementos que son relevantes en un proyecto de este tipo

## 1.3. Objetivo general

Definir una metodología de desarrollo de aplicaciones del internet de las cosas que coadyuve en el proceso de planeación y administración de procesos que componen a un sistema orientado al IoT.

## 1.4. Objetivos específicos

1) Desarrollar una metodología capaz de permitir la administración de proyectos orientados al IoT definiendo sus etapas y los artefactos adecuados en cada una de ellas.
2) Identificar los requisitos funcionales y de calidad más comunes entre los proyectos de IoT para encontrar un común denominador y con ello definir reglas generales.
3) Sentar las bases necesarias para la fácil identificación de los requisitos funcionales y de calidad que requieran los proyectos orientados al IoT.

## 1.5. Organización de la tesis

En este apartado se presenta la organización del presente trabajo, el cual consta de seis capítulos en donde se explican claramente cada uno de los temas que se desarrollaron.

**Capítulo 1:** Inicia con una introducción que expone antecedentes históricos relacionados con la problemática que se ataca en el presente documento. Se realiza un planteamiento del problema, una justificación, se enlistan los objetivos generales, así como los objetivos particulares.

**Capítulo 2:** El marco teórico es una recopilación de temas relacionados directamente con la ingeniería de software particularmente presenta las metodologías de desarrollo de software existentes y sus componentes, las categorías en las que se dividen, así como los elementos que las conforman.

**Capítulo 3:** En el estado del arte encontrarán los esfuerzos que otros autores han realizado en el tema de la creación de metodologías de desarrollo de software orientadas al IoT y cuales han sido sus aportaciones más relevantes.

**Capítulo 4:** Se desarrolla la metodología MeDAIC como propuesta de solución para el desarrollo de aplicaciones IoT. Se explican de manera detallada los ocho pasos que conforman a la metodología. Se proponen las herramientas tecnológicas que se pueden emplear en cada uno de los pasos.

**Capítulo 5:** En esta sección se describe los casos de prueba a los que se le implementó la metodología MeDAIC, así como los resultados obtenidos, esto con la finalidad de soportar la solución propuesta en el capítulo 4 con datos estadísticos obtenidos en el campo de pruebas.

**Capítulo 6:** Se plantean las conclusiones a las que se llegó después de realizar las pruebas de la metodología MeDAIC.

# Capítulo 2

# 2. Marco teórico

La ingeniería de software es el área de la informática que se apoya en el uso de metodologías y técnicas para desarrollar y mantener software de calidad. En cada proyecto se utiliza una metodología de desarrollo de software capaz de brindar orden y estructura al proceso de construcción del sistema. Existen diversas metodologías que permiten enfrentar el problema desde distintos enfoques. A continuación, se mencionan algunas particularidades que conforman a las metodologías más populares en el mundo de la ingeniería de software.

Sin embargo, antes de iniciar con el tema de las metodologías es conveniente que se analice a detalle y de forma breve ¿Qué es un método?, ¿qué es una metodología?

El ingeniero Oscar Tinoco Gómez (Tinoco Gómez O. R., 2010), menciona en su articulo que:

> "Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología esta formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo".

Actualmente existe una gran variedad de metodologías de desarrollo de software sin embargo, dichas metodologías están basadas en ciertos modelos generalistas que fueron creados hace muchos años, incluso se remontan a los principios de nuestra era tecnológica. Regularmente cada metodología de desarrollo de software tiene un modelo de desarrollo bien marcado, a continuación, se describirán los modelos tradicionales más populares y su funcionalidad.

## 2.1. Modelos para el desarrollo de software

Los modelos de desarrollo de software han existido desde el origen de la programación y estos surgieron originalmente para poner orden en el caos del desarrollo de software

(Pressman R. S., 1988). La historia ha demostrado que estos modelos de desarrollo han proporcionado a los desarrolladores una estructura de trabajo útil durante las etapas de construcción de sus proyectos.

La función de los modelos en un proyecto de construcción de software consiste en describir el conjunto de elementos que intervienen y los procesos que realizan: actividades estructurales, tareas especificas, artefactos resultantes, aseguramiento de la calidad, así como los mecanismos de control de cambios.

A continuación, se mencionarán los tres modelos que se consideran como representativos de los modelos de procesos típicos.

1) El modelo en cascada
2) El modelo de desarrollo evolutivo (Modelos en Espiral)
3) El modelo de desarrollo basado en componentes

### 2.1.1. El modelo en cascada

Es un proceso que opera de forma secuencial y se caracteriza por ser fácil de desarrollar ya que los pasos que lo conforman son vistos hacia abajo (como en una cascada de agua) a través de las fases de análisis, diseño, implementación, pruebas y mantenimiento, la primera descripción formal del modelo de cascada fue publicada por Wiston Royce W en 1970, aunque Royce no utilizó el término "Cascada".

Según Royce , el modelo de cascada se derivó de procesos de sistemas más generales. Sus principales etapas se transforman en actividades fundamentales del desarrollo:

1) **Análisis:** Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirven de manera específica al sistema.
2) **Diseño:** Para realizar el diseño de un sistema lo más conveniente es trasformar los requerimientos en pequeños sistemas los cuales en conjunto conformen el sistema principal. Para el caso del diseño de software se identifican los elementos abstractos que son fundamentales para el software, así como las relaciones entre estos.

3) **Implementación:** Se construyen un conjunto de programas orientados a resolver las tareas específicas solicitadas en los requerimientos, posteriormente se ejecutan pruebas de unidad para verificar que cada programa efectivamente cumple con la tarea para la que fue concebido.

4) **Prueba:** Se procede con las pruebas verificando que el conjunto de los componentes trabaja de manera correcta, para ello se deben de platear distintos escenarios de acción por parte de los usuarios poniendo especial atención a los detalles buscando identificar posibles anomalías que hayan sido dejadas de lado en cuyo caso deberán ser atendidas en una nueva iteración después de terminar el ciclo.

5) **Mantenimiento:** En esta fase el sistema se instala y se pone en funcionamiento práctico el mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren en nuevos requerimientos.

La Figura 2.1 muestra el modelo de cascada, uno de los más utilizados, base fundamental para otros modelos. Los conceptos desarrollados por Royce prácticamente son los mismos que se utilizan en otros modelos. El modelo en cascada es sin lugar a duda el modelo ideal para la mayoría de los proyectos de desarrollo de software.

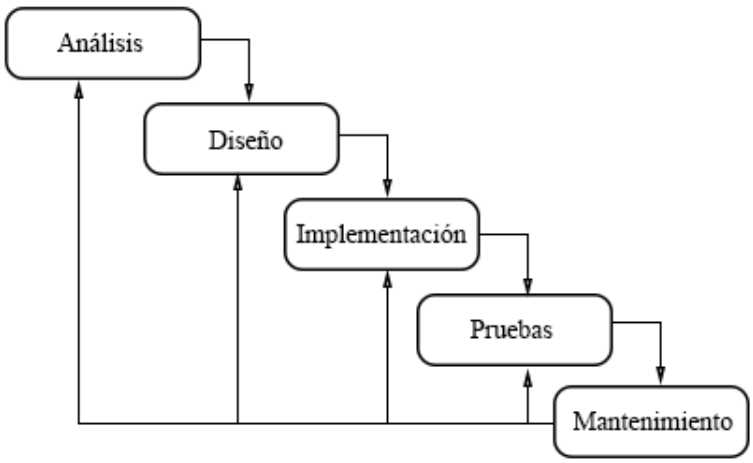


*Figura 2.1 Modelo en cascada.*

Una de las ventajas más grandes del modelo en cascada es que cada una de las fases que lo conforman se encuentra perfectamente documentada por lo que si el proyecto se suspende por cualquier circunstancia en alguna de las fases, cualquier desarrollador que desee continuar con el proyecto lo podría hacer solo con leer la documentación.

### 2.1.2. El modelo de desarrollo evolutivo (Modelo en Espiral)

Las actividades de este modelo se conforman en una espiral que esta es una línea curva genera por un punto que se va alejando progresivamente del centro a la vez que gira alrededor de él en la que cada bucle o iteración representa un conjunto de actividades. Cuando se habla del termino Iteración se hace referencia al acto de repetir un proceso con la intención de alcanzar una mata deseada, objetivo o resultado.

El modelo en espiral tiene en cuenta fuertemente el riesgo que aparece a la hora de desarrollar software, para ello se comienza analizando las posibles alternativas de desarrollo, se opta por la del riesgo mas asumible y se hace un ciclo de la espiral y si el cliente desea seguir haciendo mejoras en el software se vuelve a evaluar las distintas nuevas alternativas y riesgos y se realiza otra vuelta de la espiral así hasta que llegue el momento en el que el producto de software sea aceptado y no necesite seguir mejorándose con otro nuevo ciclo.

El modelo en espiral puede adaptarse y aplicarse a lo largo de la vida de software de computadoras. Básicamente consiste en una serie de ciclos que se repiten en forma de espiral comenzando desde el centro, se suele interpretar como que dentro de cada ciclo de la espiral se sigue un modelo en cascada, pero no necesariamente debe ser así ya que el espiral puede verse como un modelo evolutivo que conjuga la naturaleza iterativa del modelo de prototipos con los aspectos controlados y sistemáticos del modelo de cascada con el agregado de gestión de riesgo (Cervantes Ojeda, 2012). La Figura 2.2 muestra el diagrama de modelo en espiral y los componentes que la conforman.

El modelo en espiral toma parte del modelo en cascada y lo complementa con prototipos desechables. Al avanzar a través de las distintas etapas se puede observar que los modelos que van apareciendo forman parte de los modelos anteriores, pero se realizan mejoras con cada iteración.


*Figura 2.2 Modelo en espiral*

En conclusión, se puede ver que el modelo en espiral es claramente distinto a los demás ya que existen fases claramente definidas las cuales deben respetar el orden que se indicó desde el principio.

### 2.1.3. El modelo basado en componentes

El paradigma de desarrollo de software basado en componentes es un modelo de desarrollo que busca crear partes que sean reutilizables para cualquier proyecto de software que cumpla determinadas condiciones. Según Sametinger (Sametinger, 1997) definía en su libro lo siguiente:

> "la reutilización de software es el proceso de crear sistemas de software a partir de software existente, en lugar de desarrollarlo desde el comienzo".

El producto de este paradigma recibe el nombre de componente; componente es un software que cumple una serie de condiciones especificas dentro de unas condiciones determinadas y funciona dentro de una o varias interfaces sin que el cliente tenga acceso al código fuente.

Las características que conforman al modelo basado en componentes son:

1) Reutilización de software. Consiste en la creación de componentes que hayan sido planeados con ese fin desde su concepción con la finalidad de poder ser utilizados en otros proyectos reduciendo costos operativos y tiempos de construcción.

2) Simplifica las pruebas. Ya que el desarrollador no tiene acceso al código fuente del componente no es necesario que realice pruebas sobre el mismo, tradicionalmente se sabe que al realizar pruebas de funcionalidad de los softwares lleva un tiempo y un costo considerable por lo que evitar esta actividad tiene un gran beneficio.
3) Simplifica el mantenimiento del sistema. Ya que los componentes son elementos complementarios del software que se esta desarrollando, la responsabilidad de mantener el componente recae sobre el fabricante o proveedor de este.

En este tipo de modelo es común formularse la siguiente pregunta ¿Como elegir un componente? Los componentes se obtienen de formas distintas:

1) Pueden ser comprados a un tercero.
2) Pueden ser creados dentro de la misma empresa o durante el mismo proyecto.
3) Puede reutilizarse un componente que haya sido creado con anterioridad.

El modelo de desarrollo de software basado en componentes

Figura 2.3 creado por (Boehm, 1988) tiene la ventaja de reducir la cantidad de software que se debe desarrollar y por ende reduce los costos y los riesgos. También permite una entrega más rápida del software. Sin embargo, los compromisos a los requerimientos son inevitables y esto da lugar a un sistema que no cumpla con las necesidades reales de los usuarios. (Pressman R. , 2006), detecto que:

"El software de computadoras moderno se caracteriza por el cambio continuo, los tiempos de entrega son muy reducidos y una necesidad intensa de satisfacer al cliente/usuario. En muchos casos, el tiempo de llegada al mercado es el requisito de gestión más importante. Si se pierde una ventana del mercado, el mismo proyecto de software puede perder su significado".

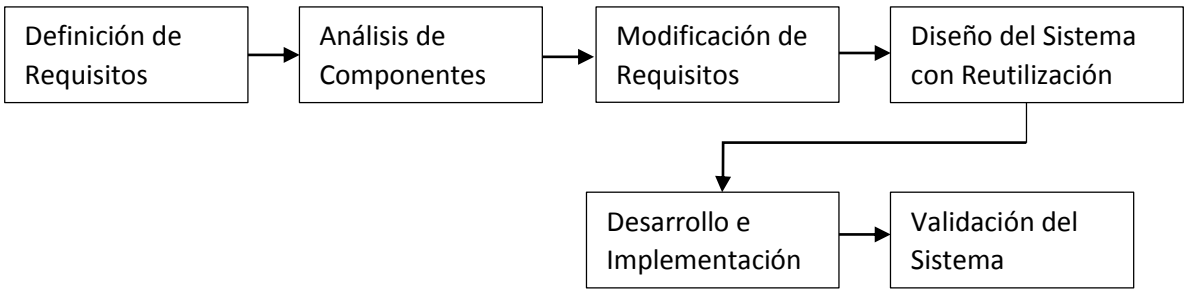

*Figura 2.3 Modelo basado en componentes*

En conclusión, los modelos de desarrollo de software proveen la mecánica necesaria para elaborar la documentación de cada proceso que interviene durante la construcción del sistema de tal manera que, si el proceso de construcción del software se detienen por cualquier situación, sería posible reanudar el trabajo en cualquier momento incluso con un equipo de programadores distinto quienes solo tendrían que leer la documentación para posteriormente continuar con el proyecto.

### 2.1.4. Metodologías de Desarrollo de Software

Dijkstra, con un influyente artículo "Go to statement considered harmful" (Dijkstra, 1968), sienta las bases para la creación de las metodologías, como las conocemos ahora.

Las metodologías de desarrollo de software han evolucionado de manera significativa en las últimas décadas permitiendo así el éxito o el fracaso de muchos de los sistemas desarrollados para distintas áreas. A continuación, se mencionarán algunas metodologías de desarrollo de software consideradas como las más populares.

Cuando se habla de metodologías se refiere a un conjunto de procesos, procedimientos, técnicas y/o herramientas que proporcionan una guía para el cumplimiento de metas u objetivos, en el campo del desarrollo de software existe una gran variedad de metodologías enfocadas a brindar a el equipo de trabajo (Análisis, diseño, desarrollo, pruebas, etc.) lineamientos para la construcción de un sistema de calidad.

Estas metodologías se conforman por un conjunto de faces que van a guiar al equipo de trabajo en el cumplimiento de objetivos, a aplicar una serie de técnicas según la fase en la que se encuentre. De esta manera se puede planificar, controlar y verificar todas las tareas que se realizan con el objetivo de validar que cada una de las fases cumple con los objetivos que suman al producto terminado.

Existen dos tipos de metodologías de desarrollo de software las metodologías tradicionales y las metodologías agiles.

#### 2.1.4.1. Metodologías tradicionales

Este tipo de metodología se enfoca principalmente en el detalle de la planificación y la estructuración del proyecto como tal; esto se consigue mediante un alto esfuerzo a nivel de requerimientos, diseño, modelado y de procesos claramente definidos, así como un alto grado de documentación.

Hablando del desarrollo de software las metodologías tradicionales atacan básicamente todo el ciclo de vida del desarrollo que son:

1) Planteamiento del problema
2) Análisis
3) Diseño
4) Implementación
5) Pruebas
6) Implementación

Cada fase se ataca de forma individual y progresiva de tal manera que no se puede iniciar una fase sin que se haya terminado la fase anterior, esto hace a estas metodologías restrictivas y poco adaptables a los cambios

Como ejemplo de este tipo de metodologías se pueden mencionar a RUP (Rational Unified Process) y a MSF (Microsoft Solutions Framework) las cuales son insignia de los procesos tradicionales ya que atacan todo el ciclo de vida del software con una serie de faces previamente definidas y muy estructuradas que los miembros del equipo deben acatar según el contexto o enfoque del proyecto.

Por ejemplo, la metodología RUP cuenta con faces muy definidas que son:

1) Inicio
2) Elaboración
3) Construcción
4) Transición

Aunado a las fases se cuenta con las tareas que se atacan durante el ciclo de vida del proyecto:

1) Modelado de negocios

2) Requerimientos
3) Análisis y diseño
4) Implementación
5) Pruebas
6) Entradas
7) Configuración de cambios y administración
8) Administración de proyecto
9) Ambiente

En la metodología RUP normalmente existe un gran esfuerzo en la etapa inicial del proyecto durante el modelado de negocio y en la definición de requerimientos, pero a medida que se avanza en la construcción del software cada una de las faces va disminuyendo su complejidad y esfuerzo requerido lo que permite al equipo de trabajo tener un control sobre las etapas de trabajo.

Una gran ventaja de trabajar con este tipo de metodologías es que se cuenta con una estructura previamente definida con los procesos claramente explicados lo que lo hace fácil de seguir, sin embargo, también se cuenta con algunas desventajas al ser esta una metodología tan rigurosa en su implementación. Una de estas desventajas es que por lo regular hay que esperar a terminar una etapa para continuar con la siguiente ya que cada etapa genera una serie de entregables que son entradas para el inicio de la siguiente etapa. Si algo esta mal en una etapa anterior y no se identifica, afecta las etapas posteriores. Otro inconveniente que se puede identificar es que al tener etapas tan rigurosas y cerradas probablemente haya muchos inconvenientes a nivel de control de cambios que por lo regular son comunes en proyectos grandes. Es común que el cliente cambie de opinión con respecto a alguna funcionalidad y si la etapa esta muy avanzada estos cambios son muy difíciles de vincular.

En general se pueden enlistar muchos inconvenientes en la metodología RUP sin embargo el mas notable es la larga espera por parte del cliente para poder acceder al producto terminado y es que cabe mencionar que en la metodología RUP el cliente no se contempla como parte del equipo de trabajo, simplemente es la persona que realiza la solicitud del proyecto mismo que se dará cuenta de los resultados hasta el final del proceso de construcción del sistema provocando que en algunas ocasiones lo que se le entrega al cliente no era lo que se esperaba

ya que muchas veces se encontraron problemas en etapas anteriores que al no tener un correcto control de cambios se vieron afectadas en etapas posteriores.

La

Figura *2.4* hace referencia a las distintas etapas por las que pasa al momento de planear y ejecutar un software utilizando la metodología RUP como base.

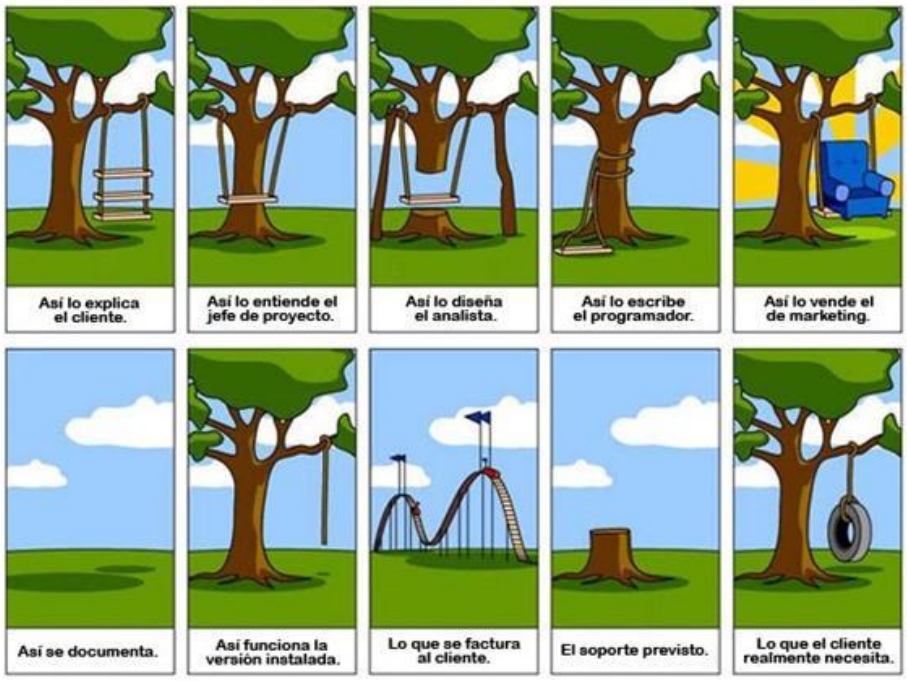


*Figura 2.4 La crisis del software*

*Esta figura fue tomada del obtenida de articulo La Crisis del Software (informática, 2011).*

A pesar de que la estructura de la metodología RUP esta bien definida resulta complejo seguirla ya que las etas que conforman a la metodología son muy rígidas y poco tolerante a cambios debido a tantas reglas y tanto nivel de detalles entre otros aspectos.

### 2.1.4.2. Metodologías Ágiles

Teniendo en cuenta las diferentes problemáticas, complejidades y lo robustas que pueden llegar a ser las metodologías tradicionales en el año 2001 se reunieron un conjunto de personas en Utah – EE. UU. En la reunión participaron 17 expertos de la industria del software incluyendo alguno de los creadores o impulsores de metodologías de software. A partir de ese momento nació el termino Ágil (Canós, 2012).

Las metodologías Ágiles se enfocan en procesos incrementales con entregas funcionales de un producto, buscando aumentar la confianza de los clientes al vincularlos en el proceso mediante cooperación entre estos y el equipo de trabajo. Estas metodologías se caracterizan por reducir la gran cantidad de documentación que es común en las metodologías tradicionales.

Las metodologías ágiles se ajustan a las necesidades del cliente, se adaptan a las circunstancias y se enfoca en dar solución a los problemas de una forma más ágil y efectiva. Como resultado de esta reunión nació a lo que se le llama Manifiesto Ágil el cual consta de un conjunto de reglas, normas y directrices que debe de cumplir una metodología para considerarse ágil (Canós, 2012).

El manifiesto ágil describe 4 aspectos clave.

1) Individuos e interacciones sobre Procesos y herramientas: tiene que ver directamente con el equipo el como se integran y como trabajan en conjunto para el cumplimiento de objetivos sin descuidar los procesos, la metodología y las herramientas de trabajo.
2) Software funcionando sobre Documentación exhaustiva: El énfasis esta en buscan que el sistema funciones de forma correcta sin necesidad de documentación rigurosa tal y como se hace en las metodologías tradicionales.
3) Colaboración con el cliente sobre La negociación contractual: Se prioriza la comunicación con el cliente para lograr un buen acompañamiento y establecer una correcta relación a pesar de no estar estipulado en el contrato. De esta manera se busca que el cliente sienta confianza al sentirse involucrado en todo el proceso.
4) Respuesta al cambio sobre Seguimiento de un plan: Se le dará prioridad a los requerimientos que se consideren generarán un impacto positivo en la construcción del proyecto sin dejar de lado los demás requerimientos secundarios.

Basadas en el manifiesto ágil nacieron algunas metodologías que siguen esos principios tales como:

- eXtreme Programming XP
- Iconix
- Test Driven Development (TDD)

- SCRUM
- Kanbas

Dentro de las principales ventajas que se pueden observar en las metodologías ágiles esta su adaptación. Si bien las metodologías tradicionales están enfocadas a proyectos muy grandes y complejos las metodologías ágiles también pueden atacar ese tipo de proyectos con la ventaja de que los tiempos de espera entre el desarrollo y la entrega de un producto funcional son mucho mas cortas ya que se busca realizar entregas iterativas de módulos funcionales e incrementales para que el cliente pueda ir conociendo paso a paso como crece su proyecto y como se construye su necesidad.

En SCRUM al trabajar por etapas o módulos (Sprint) se ataca una parte del proyecto facilitando su gestión de forma independiente aplicando el principio de "divide y vencerás".

#### 2.1.4.2.1. Scrum

En el articulo de (Canós, 2012) se define la metodología Scrum de la siguiente forma:

> "Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos".

Su enfoque inicial es para atacar proyectos de software, sin embargo, puede ser adaptable a cualquier contexto o cualquier área del conocimiento ya que es ligero y fácil de entender, aunque, si no se interpretan correctamente sus reglas puede ser complicado de implementar.

En el ciclo de vida de Scrum existen tres roles principales mediante los cuales se delegan las responsabilidades de cada uno de los involucrados en el proyecto, es también se le conoce como el Scrum Team y consta de lo siguiente:

Product owner (Dueño del proyecto): Es la persona que esta del lado del cliente y que representa al cliente, se puede comparar con el ingeniero de requisitos ya se encarga de recabar toda la información y conocer de primera mano todas las necesidades del cliente para luego trasmitir dicha información al Scrum Master y al equipo de trabajo (Development Team) para que ellos se encarguen de construir esa necesidad.

1) SCRUM Master: Es el líder del proyecto y su función es modelar el proyecto y guiar al equipo de trabajo para que estos puedan entender cuales son las necesidades del cliente y cual es la necesidad que el Product Owner les ha manifestado.
2) Equipo de trabajo: Es el equipo de desarrollo que construye la necesidad que el Product Owner ha manifestado. Independientemente de sus roles, el equipo de trabajo va a ser conocido como los desarrolladores ya que todos los involucrados tienen esta distinción.

El ciclo de vida de Scrum es muy simple de entender tal y como se muestra en la Figura 2.5 donde se muestran claramente los roles que intervienen donde se puede destacar lo siguiente:

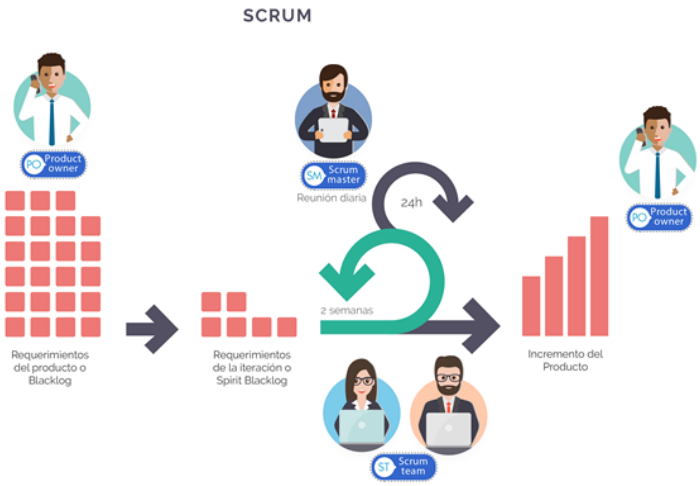


*Figura 2.5 Ciclo de vida de Scrum*

Esta imagen tomada de la documentación de Scrum *(Scrum, 2019)*

Primero el Product Owner va a definir un artefacto el cual es la lista completa de los requerimientos del cliente, a este artefacto se le conoce como "Product BackLog". En dicho artefacto se plasman todas necesidades, ideas y requisitos que harán cumplimiento a la solicitud del cliente. Posteriormente se trasmites dichas necesidades al equipo de desarrollo y al Scrum master. Lo anterior se realizar mediante una reunión con el equipo de trabajo denominada Sprint Planning Meeting, en donde se planea como se le dará solución a una primera parte del producto final.

Como resultado del Sprint Planning Meeting se obtiene una lista de funcionalidades llamada Sprint BackLog, estas son tomadas del Product BackLog las cuales consiste de un conjunto de requisitos que se deben construir en un tiempo que va desde una a cuatro semanas dependiendo de la complejidad de la actividad definida en el BackLog, a ese tiempo se le denomina Sprint el cual es considerado el corazón del proceso ya que el Sprint corresponde al proceso de desarrollo o construcción de las necesidades del cliente  pero divididas en un módulo funcional o en un producto incremental.

En el Sprint interviene el Scrum Master y el equipo de trabajo estos últimos son los encargados de desarrollar y construir esa necesidad que define al Sprint mientras que el Scrum Master se encarga de ayudar facilitando los recursos para que el equipo de trabajo pueda operar.

Una de las actividades mas representativas del Scrum son las reuniones diarias denominadas Daily Scrum. Esas reuniones tienen como objetivo hacer seguimiento diariamente a todos los procesos que estén activos dentro del Sprint y se resuelven cuatro cuestionamientos básicos ¿Qué se hizo ayer?, ¿Qué se esta haciendo hoy?, ¿Qué se va a hacer mañana? y ¿Qué problema se encontró? El objetivo es que las reuniones sean cortas no mayores a 15 minutos y que diariamente se pueda tener un contexto global de cual es el estado actual de Sprint para facilitar el seguimiento del proyecto y la toma de decisiones.

Scrum utiliza un tablero tipo kanban tal y como se muestra en la Figura 2.6 tomada del sitio http://managementplaza.es/blog/scrumban/. El tablero es una herramienta que facilita el proceso de entendimiento del Sprint que se esta trabajando garantizando el principio de trasparencia para que todos los miembros del equipo puedan aportar a la solución.

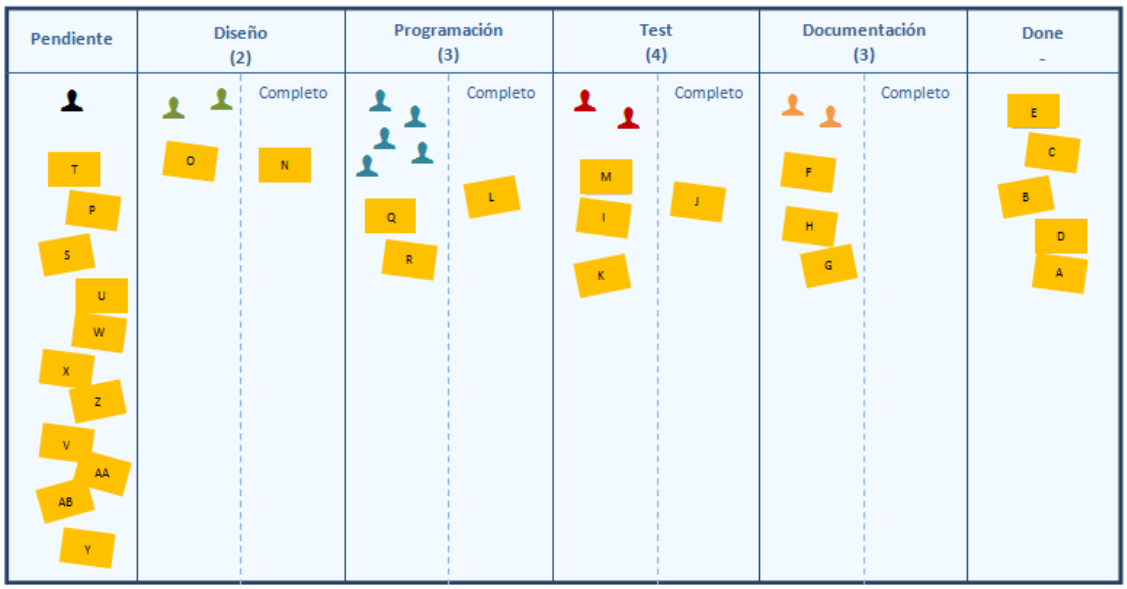


*Figura 2.6 Tablero kanban Scrum*

Al final del Sprint se realiza una nueva reunión denominada Sprint Review en donde están involucrados el Product Owner, el Scrum Master y el Development Team para verificar el cumplimiento de las metas o los objetivos del Sprint en cuestión para garantizar la entrega del producto al cliente final.

Después de haber entregado el producto final se realiza una nueva reunión denominada Sprint Restrospective y en esta se busca analizar cuáles son los resultados del Sprint anterior en donde se identifican las áreas de oportunidad en donde se puede mejorar y aplicar los cambios necesarios en el próximo Sprint. De esta manera se da inicio a un nuevo Sprint tomando otra de las funcionalidades del Product BackLog para definir nuevamente el Sprint BackLog e iniciar nuevamente el proceso hasta tener un nuevo producto funcional.

El cliente irá recibiendo paulatinamente estos productos entregables de esta manera se irá enterando de los avances del proyecto hasta que terminen el desarrollo de todos los Sprint y con ello se tenga el producto terminado.

En conclusión, se puede decir que el éxito del Sprint depende básicamente de todo el equipo de desarrollo ya que se busca que todas las personas puedan tener asignaciones de las cuales sean responsables y cuando estas responsabilidades sean cumplidas pueda apoyar a otras personas en sus asignaciones para poder dar cumplimiento a los objetivos del Sprint y al final poder dar cumplimiento a los tiempos definidos.

#### 2.1.4.2.2. Metodología XP Programación Extrema

Es una de las metodologías de desarrollo de software más exitosa utilizada para proyectos de corto plazo, con recursos limitados y cuyo tiempo de entrega es muy corto (Campos, 2015). La metodología consiste en una programación rápida y extrema cuya particularidad es tener como miembro del equipo al usuario final ya que es uno de los requisitos para llegar al éxito del proyecto.

En este tipo de metodología se pone énfasis en la adaptabilidad, es decir, trabajar de forma tal que el proyecto pueda ser adaptado a los requisitos sobre la marcha ya que es común que los requisitos cambien de forma abrupta.

Esta metodología se basa en un conjunto de valores y buenas practicas que al ser utilizados en conjunto dan solidez al proyecto mientras este se mantiene flexible durante los cambios que el cliente solicite. Se suelen asociar los siguientes valores:

1) Simplicidad: El desarrollo del sistema debe mantenerse simple tanto para los desarrolladores como para los usuarios.
2) Comunicación: Se requiere que exista una comunicación clara y precisa entre todos los involucrados en el desarrollo del sistema, que se utilice un lenguaje eficaz al momento de comunicar los requisitos para evitar confusiones al momento del desarrollo.
3) Realimentación: Al ser una metodología que satisface los requerimientos del cliente sobre la marcha es importante que cada versión realimente a la siguiente para evitar inconvenientes.
4) Coraje: Las personas involucradas en el desarrollo de un proyecto utilizando la metodología extrema debe poder realizar los cambios que el cliente le solicite sobre la marcha sin temor a equivocarse e informar a los demás miembros del equipo de los cambios solicitados.
5) Respeto: El programador debe cumplir con el diseño pactado, debe respetar los esfuerzos realizados por los demás desarrolladores involucrados en el proyecto con la finalidad de entregar un producto completo con el compromiso de todos.

La programación extrema también se basa en una serie de buenas prácticas que se deben de seguir las cuales se enlistan a continuación:

- Planificación
- Entregas pequeñas
- Metáfora
- Diseño simple
- Pruebas
- Refactorización
- Programación en parejas
- Propiedad colectiva del código
- Integración continua
- 40 horas por semana
- Cliente in-situ
- Estándares de programación

En la Figura 2.7 se ilustra un gráfico representativo de las características antes mencionadas, es un diseño cíclico que busca interactuar con los diferentes procesos involucrados en la programación extrema con la finalidad de conseguir una versión funcional que pueda ser lanzada, este proceso se sigue una y otra vez hasta culminar con todos los requerimientos del cliente.

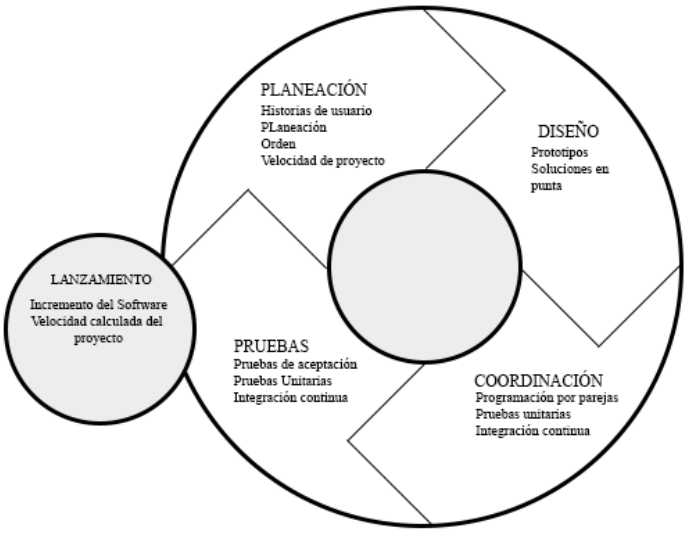


*Figura 2.7 Metodología Programación Extrema (XP)*

Dentro de las ventajas que se pueden mencionar en esta metodología resaltan las siguientes:

- Adaptabilidad

- Control de errores
- Versionamiento constante
- Programación organizada
- Mayor interacción con el cliente

A pesar de que la programación extrema es muy popular cabe mencionar que existe una serie de desventajas que deben ser consideradas antes de optar por elegir esta metodología como base de un desarrollo de software, las desventajas son:

- Puede llegar a ser demasiado complejo (Dependiendo del proyecto)
- El cliente goza de mucha libertad
- Imposibilidad de previsión global

En conclusión, se puede mencionar que las metodologías de desarrollo ágil han revolucionado la forma de gestionar y administrar proyectos, siendo esta metodología la que actualmente lidera en el mundo de la ingeniería de software. SCRUM se ha destacado por su versatilidad y dinamismo al momento al momento de desarrollar las tareas en intervalos de tiempo relativamente cortos. Definitivamente SCRUM cuenta con cualidades muy atractivas que fueron consideradas al momento de conformar la metodología MeDAIC.

#### 2.1.4.2.3. Desarrollo Adaptativo de Software (DAS)

Es una metodología que fue introducida en el año de 1998 por Jim Highsmith como una técnica para desarrollar software y otros sistemas complejos. Esta metodología se propuso como una alternativa de desarrollo de software donde no se tuviera suficiente claridad en los requerimientos en cuyo caso los miembros del equipo buscan realizar aportaciones con la finalidad de obtener el desarrollo.

Cadavid menciona en su articulo sobre metodologías ágiles (Cadavid, 2013) que:

> "El ciclo de vida de DAS está orientado al cambio y se compone de las fases: especulación, colaboración y aprendizaje. Estas fases se caracterizan por estar enfocadas en la misión, estar basadas en características, ser iterativas, tener marcos de tiempo especificados, ser orientadas por los riesgos y ser tolerantes a los cambios."

A continuación, se dará una descripción de las fases que componen la metodología DAS:

1) Especular: Se realizar un levantamiento de aquellos requisitos que se crean necesarios para conformar el proyecto, esto se hace de forma especulativa ya que no existe una certeza de los requerimientos por parte del cliente.
2) Contribución: Se explotan las fortalezas de cada uno de los miembros del equipo de desarrollo ya para poder desarrollar un sistema se requerirá de programadores, diseñadores, expertos en bases de datos, expertos en comunicaciones, etc.
3) Aprendizaje: En esta etapa los desarrolladores analizan los avances del proyecto y buscan alternativas para realizar cambios o mejoras dentro del proyecto sin afectar la funcionalidad de este.

En la Figura 2.8 se ilustra un diagrama representativo de la metodología DAS en donde se puede notar cada uno de los puntos antes mencionados.

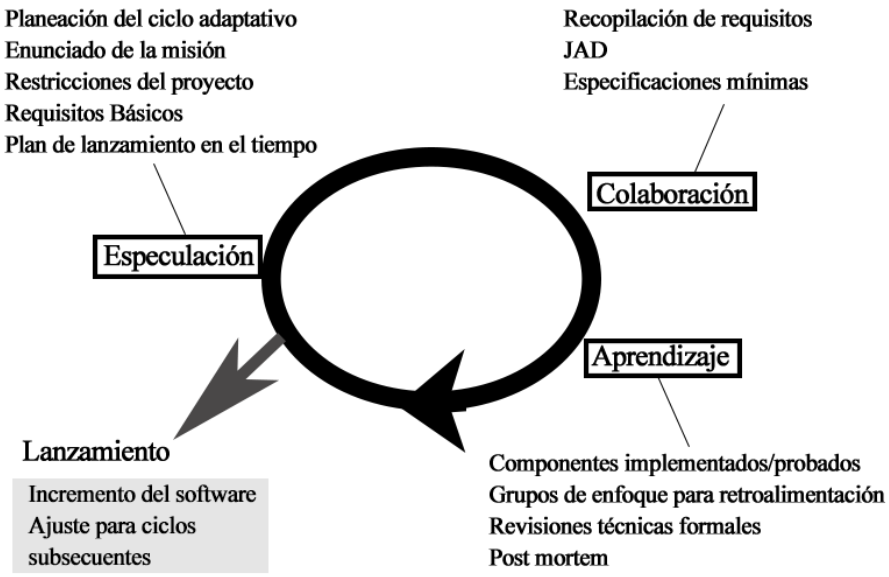


*Figura 2.8 Desarrollo Adaptativo de Software (DAS)*

## 2.2. Conclusión.

Como ya se ha comentado en el presente capítulo, existen diversas metodologías de desarrollo de software que fungen como marco de trabajo durante el desarrollo de sistemas computacionales, dichos marcos de trabajo proveen al desarrollador una manera de ordenar gestionar, administrar y mantener los proyectos que desarrolle.

Queda claro que existe una metodología apropiada para cada tipo de proyecto, sin embargo, cabe mencionar que los proyectos orientados al IoT no cuentan con una metodología completamente afín que proporcione los medios necesarios para desarrollar un sistema donde se contemplen las características propias de un desarrollo IoT.

Uno de los objetivos particulares que se plantean en el presente trabajo de investigación es la creación de una metodología que considere las características propias de un proyecto orientado al IoT, pero no es intención del autor de este documento iniciar desde cero, sino que, se pretende tomar lo mejor de cada metodología y armonizar estos elementos en un solo proyecto funcional que le resulte familiar al desarrollador sobre todo que resulte útil en la creación de proyectos orientados al IoT.

# Capítulo 3

# 3. Estado del arte

El IoT es un término propuesto por Kevin Ashton en el Auto-ID center del MIT en 1999 en donde se realizaban investigaciones en el campo de la identificación por radiofrecuencia en red y tecnologías de sensores. En la actualidad se están desarrollando una gran cantidad de sistemas que entran en la clasificación del IoT ya que la industria busca sistematizar sus procesos y mantener el control de su maquinaria a través del uso de dispositivos móviles.

El problema con el desarrollo de aplicaciones orientados al IoT es que no se tiene una metodología especialmente diseñada para este fin por lo que se tienen que tomar metodologías ya existentes y probadas para adecuarlas a este tipo de proyectos, si bien es cierto que existen múltiples intentos por desarrollar un método homogéneo para el desarrollo de aplicaciones de IoT siendo esta disciplina un campo de acción multifacético ya que es tan grande la cantidad de dispositivos electrónicos así como de sistemas operativos los que pueden interactuar con la aplicación final que se desarrolla que es difícil determinar una metodología que abarque todo el espectro de acción que tiene el IoT.

A continuación, se mencionarán los temas más relevantes en el campo de la IoT y los esfuerzos que se están realizando en el campo de la ingeniería de software para el desarrollo

de herramientas y metodologías que coadyuven en el desarrollo de proyectos orientados al IoT.

## 3.1. Internet de Las Cosas

La historia ha mostrado que los avances tecnológicos se mueven continuamente, que cada corto periodo de tiempo surgen nuevas tecnologías y con ello nuevas tendencias y maneras de realizar las actividades cotidianas. Hoy en día, una de estas tendencias que ha llegado para quedarse y que rápidamente está tomando más peso en la sociedad es la IoT, que se refiere a una visión donde los objetos se vuelven parte de Internet, donde cada objeto es identificado de manera única, es accesible a la red, su posición/estado son conocidos y donde servicios e inteligencia son agregados para expandir esta Internet, fusionando el mundo digital y físico impactando así nuestros ambientes profesionales, personales.

Con el surgimiento de esta nueva visión del Internet se abren las puertas para el desarrollo de una gama de aplicaciones que hasta hace algunos años era impensable. Para desarrollar estas aplicaciones es necesario el entendimiento de los principios fundamentales de la IoT como conexión en cualquier lugar, en cualquier momento y de cualquier cosa, tal y como se muestra en la Figura 3.1, para que el ingeniero de software tenga conciencia de la necesidad de incluir entre sus requisitos un uso extensivo de sensores y actuadores, una serie de tecnologías como las web, computación en la nube, protocolos de comunicación e incluso inteligencia artificial dependiendo del nivel de complejidad de la aplicación.

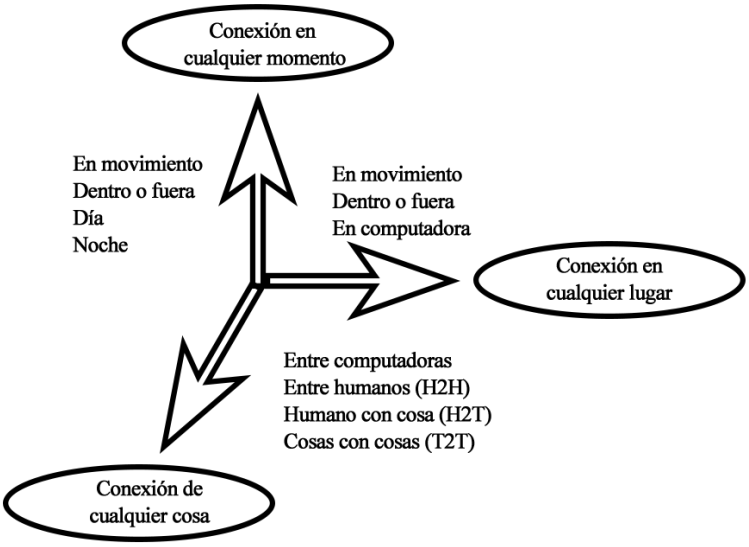


*Figura 3.1 Ubicuidad de la IoT*

El IoT domina nuestro entorno ya que la producción masiva de dispositivos electrónicos, así como el gusto de las personas por controlar todos sus aparatos mediante el uso de una tableta de manera remota convierte a esta disciplina en un potencial de innovación.

Pero si bien es cierto el IoT es una actividad innovadora en nuestra época, también es cierto que la forma de gestionar, administrar y desarrollar los proyectos sigue siendo de forma tradicional es por ello por lo que se requiere de una metodología capaz de incorporar los componentes del IoT desde la etapa de planeación hasta la etapa de implementación. A continuación, se hará mención de algunas propuestas de solución orientado a problemas específicos.

### 3.1.1. Metodología para el diseño de aplicaciones IoT

La presente metodología ha sido presentada por Bagha y Madisetti en el año 2014 y ha sido creada con base en modelo de referencia arquitectónico IoT-A (Bahga, 2014) y consiste de una serie de diez pasos que guían al diseñador por un proceso altamente documentado.

Los pasos definidos en esta metodología de diseño son:

1) Propósito y especiación de requerimientos.
2) Especificación del proceso.
3) Especificación del modelo de dominio.
4) Especificación del modelo de información.

5) Especificación de servicios.
6) Especificación del nivel IoT.
7) Especificación de la vista funcional.
8) Especificación de la vista operacional.
9) Integración de dispositivo y componente.
10) Desarrollo de la aplicación.

Propósito y especificación de requerimientos. En esta etapa de la metodología se define cual es el propósito del sistema, su comportamiento y los requerimientos son capturados.

Especificación del proceso. Como segundo paso de la metodología se crea un conjunto de casos de uso derivado del conjunto de requisitos que fueron recolectados con anterioridad; además se realiza un diagrama de proceso que cumpla con los requisitos especificados.

Especificación del modelo de dominio. En esta etapa se realiza mediante un modelo independiente de plataforma, una abstracción de los conceptos, entidades y objetos que interactúan en el sistema y se establecen las relaciones que existen entre estos diferentes elementos. Dada la naturaleza de la IoT, la metodología enfatiza cinco tipos de elementos de dominio a incluir en el modelo: entidades físicas, entidades virtuales (representación digital de una entidad física), dispositivos, recursos (componentes software o recursos de red) y servicios.

Especificación del modelo de información. En esta etapa se realiza la especificación de todos los atributos y relaciones que tienen las entidades virtuales. El modelo generado en esta etapa nuevamente es independiente de plataforma y no expresa detalles de almacenamiento de datos.

Especiación de servicios. Esta etapa de la metodología consiste en realizar la especificación de servicios a través del modelo de información y la especificación de procesos. Mediante la especificación de procesos y el modelo de información se pueden identificar los diferentes estados y atributos. Un servicio es derivado de cada atributo o estado identificado y para cada uno de ellos se especifican sus entradas y salidas de datos. Los servicios servirán para modificar el valor de los atributos o estados o bien para simplemente obtener su información.

Especificación del nivel IoT. En esta etapa se define el nivel de la aplicación de la IoT. De acuerdo con Bagha y Madisetti existen seis niveles de aplicaciones de la IoT. Cada nivel es determinado por las características con las que cuenta la aplicación.

Estas características van desde el tipo de conexión que tiene la aplicación (si son aplicaciones locales o existe acceso por Internet), si existe almacenamiento de datos en la nube o si existe minado de los datos que son recolectados a través de la aplicación (Bahga, 2014).

- Nivel 1. Tiene un solo nodo con sensores y/o actuadores, almacena información, realiza análisis y la aplicación está programada en el mismo nodo. La cantidad de datos almacenados no es grande y no se requiere computo intensivo para analizar los datos.
- Nivel 2. Tiene un solo nodo con sensores y/o actuadores y analiza información localmente. En este nivel la información es almacenada en la nube y la aplicación típicamente es basada en ella.
- Nivel 3. Tiene un solo nodo con sensores y/o actuadores. La aplicación es basada en la nube y los datos son almacenados y analizados en ella.
- Nivel 4. Tiene múltiples nodos con sensores y/o actuadores que realizan análisis local. Los datos son almacenados en la nube y la aplicación es basada en ella.
- Nivel 5. Tiene múltiples nodos esclavos y un nodo maestro que se encarga de coordinarlos. Los nodos esclavos tienen sensores y/o actuadores y envían la información que recolectan al nodo coordinador para que este los envíe a la nube. La aplicación, análisis y almacenamiento de datos están en la nube.
- Nivel 6. Tiene múltiples nodos con sensores y/o actuadores. Cada nodo envía datos a la nube. La aplicación, análisis y almacenamiento de datos están en la nube.

Especiación de la vista funcional. En esta etapa todas las funciones del sistema son agregadas en diferentes grupos funcionales. Los grupos funcionales que define las metodologías son: Dispositivo, Comunicación, Servicios, Administración, Seguridad y Aplicación. Cada grupo funcional provee información sobre las entidades que agrupa. Como resultado de esta etapa se obtiene una vista de la relación que existe entre todos los elementos del sistema y los grupos funcionales.

Especificación de la vista operacional. En esta etapa, aspectos operacionales y de despliegue del sistema son definidos. Por ejemplo, se definen las opciones de comunicación, las opciones de servicio de hosting, opciones de dispositivo, almacenamiento y tecnológicas a utilizar.

Integración de dispositivo y componente. En este paso se realiza un esquema de como deberán ser conectados los dispositivos y componentes.

Desarrollo de la aplicación. Como último paso de esta metodología de diseño se procede a realizar el desarrollo del sistema.

Esta metodología presenta un enfoque tradicional para la elaboración de aplicaciones IoT. A pesar de tener un enfoque consistente para el diseño de este tipo aplicaciones, no considera la posibilidad de que los requisitos sean cambiados en ningún momento, por lo que no se toman en cuenta pasos en los que los requisitos sean revisados con los stakeholders del proyecto para poder tomar medidas ante algún posible cambio.

### 3.1.2. CA Mobile App Services

Para ayudar a las empresas a ofrecer una buena (y cada vez más importante) experiencia al desarrollador, CA Technologies ha creado una nueva categoría de tecnologías de desarrollo móviles. CA Mobile App Services acelera el desarrollo de aplicaciones de IoT y móviles con un kit de herramientas de desarrollador que incluye varios SDK y API. El kit de herramientas proporciona al desarrollador de la empresa las ventajas que se enumeran a continuación y que reducen la codificación repetitiva y la complejidad.

### 3.1.3. VENTAJAS de CA Mobile App Services

- Las interfaces abiertas liberan a los desarrolladores y a las empresas de la obligación de ceñirse a un solo distribuidor.
- La infraestructura de seguridad subyacente reduce el riesgo en aplicaciones móviles empresariales.
- La funcionalidad de publicación/suscripción permite desarrollar aplicaciones reactivas para utilizar la propagación de eventos y datos prácticamente en tiempo real.
- Los grupos ad hoc mejoran la compartición en aplicaciones de colaboración.

- Las interfaces aptas para el IoT y el protocolo de transporte MQTT permiten una integración sin precedentes con una gran cantidad de dispositivos.

Los elementos de desarrollo móvil repetibles, aunque esenciales, como la gestión de usuarios, el almacenamiento o la integración entre el dispositivo y el back-end, se presentan en forma de funciones de un SDK invocables y sencillas para los desarrolladores. Estos profesionales pueden recurrir a tales llamadas de SDK para realizar tareas clave. El marco subyacente funciona con la puerta de enlace móvil líder de CA Technologies y las tecnologías de gestión de API ejecutan dichas llamadas y completan las tareas. Ahora, los desarrolladores se pueden centrar más en la creación de una completa experiencia de usuario sin preocuparse por las funciones de back-end.

### 3.2. Conclusión

Como se planteó en este capítulo, existen diversos esfuerzos para modernizar una vez más la forma de gestionar y administrar los proyectos, pero en esta ocasión dando un énfasis en la creación de proyectos orientados al IoT, en donde se pueda considerar el uso de los circuitos, actuadores y relevadores como componentes vivos del sistema y que estos tengan sus roles bien definidos desde el momento de la concepción del proyecto.

Para el desarrollo de la metodología MeDAIC se tomaron en cuenta las fortalezas de las metodologías mas populares con la finalidad crear un marco de trabajo que fuera versátil y moderno, que contemple los mejores mecanismos de gestión de proyectos que existen en el mercado, pero enfocado a la creación de proyectos IoT.

# Capítulo 4

# 4. Metodología MeDAIC

El presente trabajo de tesis brinda una aportación complementaria al trabajo realizado por el M.C. Ramon Alberto Isijara Medina quien desarrolló la primera parte de este proyecto (Isijara, 2015). A continuación, se explicarán los pormenores que conforman a la metodología y sus implicaciones en el uso de esta en proyectos productivos.

Como ya se ha mencionado con anterioridad en este trabajo de tesis las metodologías que conocemos en la actualidad son el resultado de años de refinamiento de los procesos que fueron adecuándose a las nuevas tendencias tecnológicas, sin embargo, esas metodologías hoy en día no cumplen con los requerimientos que la modernidad demanda ya que la tendencia hacia el desarrollo de aplicaciones del IoT se atiende mediante metodologías tradicionales y no así con una metodología ad hoc a sus necesidades.

Debido a la necesidad de establecer un nuevo mecanismo con el cual se pueda afrontar la problemática antes mencionada es que en el presente trabajo se tesis se propone una nueva Metodología de Desarrollo de Aplicaciones del Internet de las Cosas (MeDAIC por sus siglas). MeDAIC se compone de ocho niveles en cada uno de los cuales se especifica la forma de actuar por parte de los involucrados, así como los artefactos que deben resultar al finalizar cada operación. Los niveles son:

1) Definición del propósito y requisitos de alto nivel.
2) Refinamiento de los requisitos de alto nivel.
3) Especificación del modelo de información de entidades físicas y virtuales.
4) Definición del formato y estructura para el intercambio de datos.
5) Visión general de la arquitectura.
6) Construcción, pruebas y validación.
7) Desarrollo del diagrama de despliegue.
8) Implantación del sistema.

En la Figura 4.1 se ilustran las etapas y la relación progresiva que existe entre ellas hasta completar la iteración. Mediante la imagen se puede deducir la narrativa del ciclo de vida de un proyecto orientado al IoT.

Las conexiones que existen entre cada paso son enlaces ascendentes, es decir, no se puede avanzar al siguiente paso sin haber terminado el anterior, solamente el paso 6 presenta una bifurcación ya sea en retroceso al paso 2 (para refinar procesos que no hayan sido considerados o para realizar mejoras a los requisitos) o bien avanzar al paso 7 el cual realiza las tareas de desarrollar el diagrama de despliegue.


*Figura 4.1 Metodología MeDAIC*

A continuación, se describirá cada una de las etapas que conforman la metodología MeDAIC, se explicarán los procesos específicos que se deben cumplir en cada nivel, así como también se describen los artefactos entregables que deben surgir como producto de cada nivel.

## 4.1. Definición de Propósito y Requisitos de Alto Nivel

En todo tipo de proyecto el primer paso es definir los objetivos y el alcance que se desean lograr con dicho proyecto. Para ello se requiere que se hayan capturado los requisitos de los interesados oportunos, es decir, que se haya entendido exactamente los que se quiere conseguir con el proyecto.

### 4.1.1. Identificación del Propósito y Alcance del Proyecto

Es importante identificar el propósito y alcance del proyecto, no dejar lugar a dudas de lo que se espera obtener de cada una de las etapas que conforman a la metodología. A continuación, se enumeran las necesidades que se deben cumplir en esta primera etapa:

1) Definición: Se debe elaborar una definición del producto, servicio resultado que se espera obtener al ejecutar el proyecto.
2) Alcance: Se elabora un documento denominado acta constitutiva del proyecto en la cual se definen los compromisos que asumirá el equipo de desarrollo y los stakeholders.
3) Entregables: Con el objetivo de conseguir una descripción detallada de los procesos se definen los artefactos entregables, así como los trabajos necesarios para lograrlo.
4) Trabajos necesarios: El Project Manager, junto con el Equipo de Dirección del Proyecto y los actores interesados, confecciona una descripción pormenorizada de las tareas necesarias para realizar con éxito el Proyecto. El director del proyecto, tras hacer un análisis del producto que se pretende obtener con la ejecución del Proyecto, genera alternativas para conseguirlo.

### 1.1.1. Modelo Estratégico i* (SDM)

Una vez definidos los puntos anteriores se procede a identificar los requisitos de alto nivel que intervendrán en la planeación del proyecto por medio de un modelo estratégico de dependencias i* (SDM por sus siglas en ingles).

SDM es un modelo que se forma a partir de un conjunto de nodos y ligas, donde cada nodo representa a un actor y cada liga entre dos actores indica que un actor depende de otro en algo para que el primero logre cumplir con una meta. Cabe resaltar que el SDM no muestra los detalles que conlleva la red de dependencias para el logro de los objetivos generales del sistema.

Dada la definición anterior, se puede resumir que el SDM es un modelo que muestra la intencionalidad de los actores del sistema y sus dependencias estratégicas para el logro de objetivos generales. A continuación, se muestran algunos elementos de modelado del marco de trabajo i* para la construcción de un SDM.

Los actores presentados en la Figura 4.2 se definen de acuerdo con (istarwiki.org, 2011) de la siguiente manera:

- Actores: Son entidades que realizan acciones para lograr una meta. Se utiliza el termino actor para hacer referencia a cualquier unidad a la cual se le puedan atribuir dependencias intencionales. Los agentes, roles y posiciones son actores especializados.
- Agente: Es un actor con una concreta manifestación física, tal como un ser humano.
- Posición: Es una abstracción intermedia que puede ser usada entre un rol y un agente.
- Rol: Es una caracterización abstracta del comportamiento de un actor en un contexto especializado.

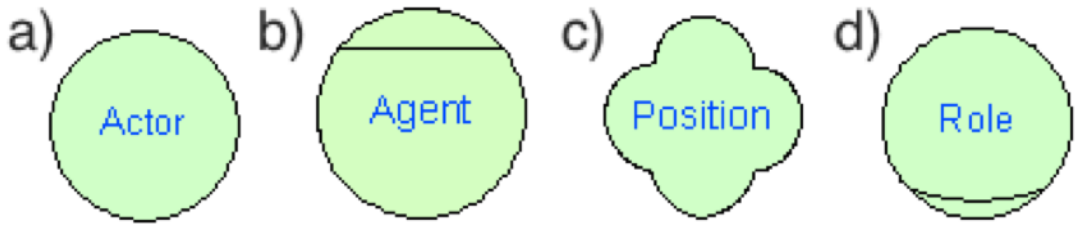


*Figura 4.2 Actores i**

En la **¡Error! No se encuentra el origen de la referencia.** se muestran los Elementos que onforman a la metodología i*. consta de cuatro formas geométricas que representan: tarea, meta, recurso y meta suave. Estos elementos fueron tomados de (istarwiki.org, 2011).

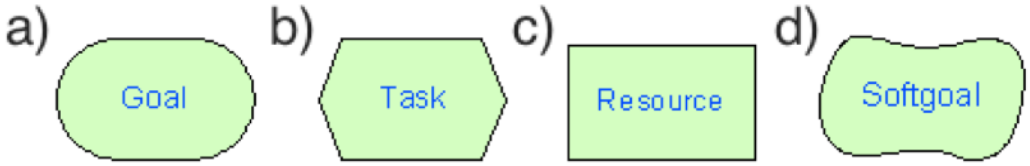


*Figura 4.3 Elementos i**

a) **Meta**. Representa la intención de un actor. La especificación de como la meta es satisfecha no es descrita por ella misma.

b) **Tarea**. El actor quiere cumplir alguna tarea específica, realizada de un modo particular. La especificación de la tarea puede ser descrita descomponiendo la tarea en subelementos.

c) **Recurso**. Provee al actor de una entidad física o de información.

d) **Meta suave**. Son similares a las metas, con la excepción de que el criterio para la satisfacción no es claro y tiende a ser juzgado por el punto de vista del actor.

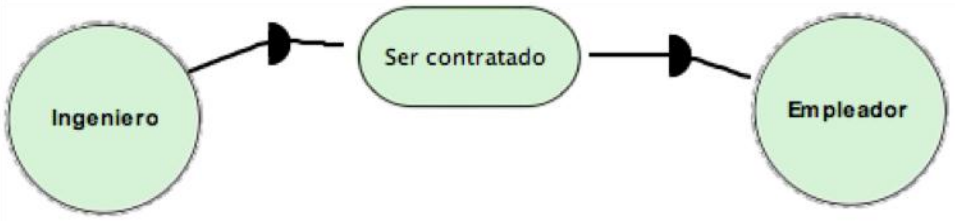


*Figura 4.4 Dependencias estratégicas en i**

Las dependencias estratégicas son modeladas mediante una liga que contiene una especie de "D". Las ligas dependencia son leídas en el sentido de la "D", esto quiere decir que, en el caso de la figura mostrada anteriormente el actor ingeniero, depende del actor empleador para ser contratado. Una vez que se definió el propósito y alcance del proyecto se debe evaluar si se han cumplido los objetivos; para ello se utiliza un sistema denominado SMART (Doran, 1981):

- S (specific): El alcance del proyecto tiene que estar claramente definido. No puede haber lugar a confusión. No importa el tiempo que le cueste al director del proyecto definirlo, ni las dificultades, no hay excusa posible.
- M (measurable): Todo en el proyecto debe ser medible. Obviamente el alcance del proyecto también debe serlo. El éxito o el fracaso del Proyecto debe ser totalmente objetivo.
- A (achievable): Tenemos un sistema, claro y definido, que facilite su consecución.
- R (realista): Tiene que estar ajustado a la realidad. Tenemos un tiempo y un plazo concreto, por lo tanto, ellos nos determinarán un alcance realista de Proyecto.
- T (time-related): tiene que tener una duración determinada.

El artefacto entregable que se produce en esta etapa es la lista de requisitos de alto nivel que conformaran al proyecto sin necesidad de entrar en detalle del comportamiento de dichos componentes en la interacción del sistema.

## 4.2. Refinamiento de los Requisitos de Alto Nivel

Como elemento de entrada se espera la lista de requisitos de alto nivel que conforman el proyecto para posteriormente realizar una reducción en el grado de abstracción del modelo estratégico de dependencias, para poder realizar el procedimiento se elabora un modelo estratégico relacional i*. Siguiendo la metodología SRM se realiza la documentación de la estructura lógica interna de cada actor dentro del límite el cual se representa con una línea circular, vea Figura 4.5

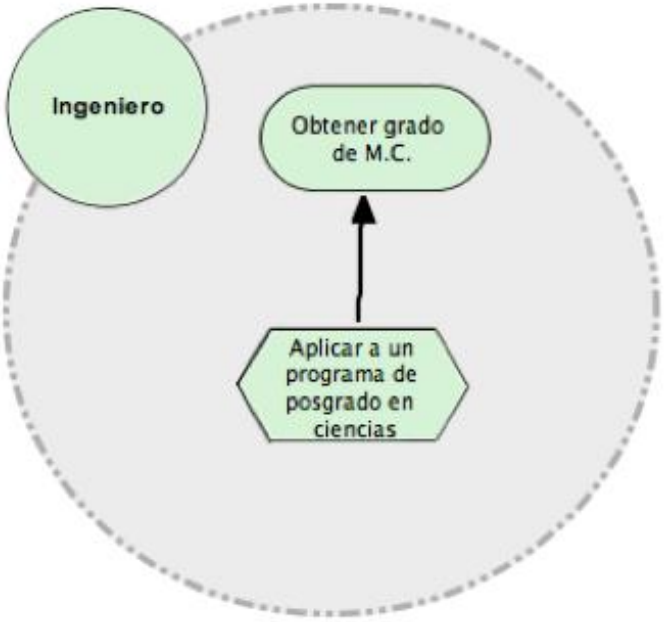


*Figura 4.5 Límites de un actor en el modelo i**

Una vez que se definió el propósito y requisitos de alto nivel y que los interesados inmediatos están de acuerdo con los estipulado en el acta de constitución del proyecto se procede a realizar el refinamiento de los datos.

La estructura lógica interna de los actores se define en términos de metas, tareas, recursos y metas suaves (Atributos de calidad), los cuales se representan por medio de diferentes tipos de ligas como se representa en la Figura 4.6

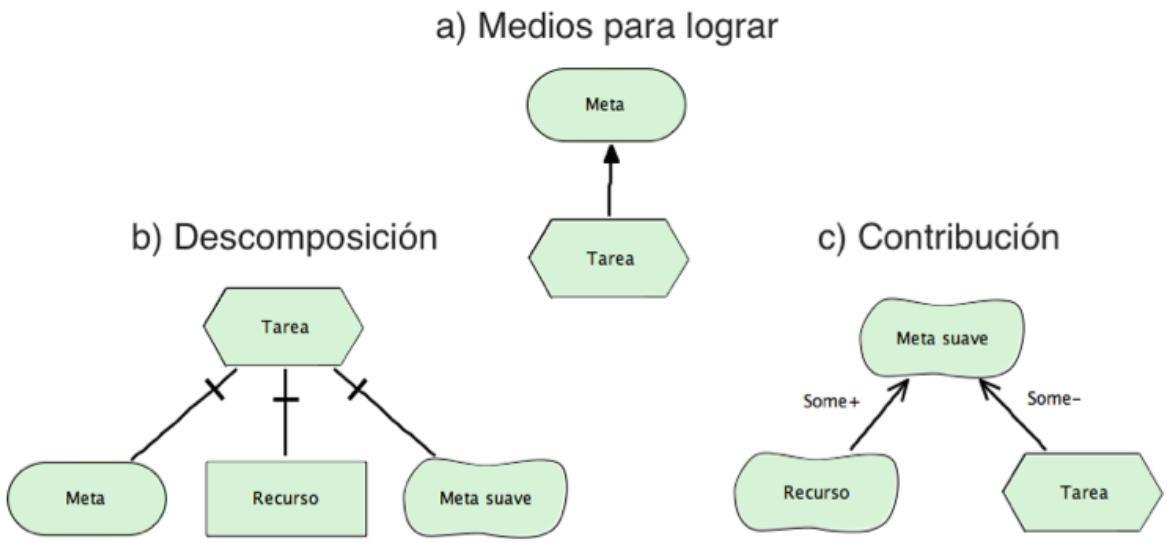


*Figura 4.6 Relaciones del Framework i**

a) Medios para lograr: Representa una tarea necesaria para el logro de la meta.
b) Descomposición: Representa la descomposición de tareas generales en diversos elementos individuales.
c) Contribución: Indica la existencia de una contribución positiva o negativa para el logro de los atributos de calidad del sistema

Con este tipo de relaciones se pueden representar las distintas etapas de un sistema, así como las relaciones que tienen con otros elementos y la interacción que es necesaria que suceda entre los elementos para lograr los resultados deseados, el modelo estratégico de dependencias ofrece una visión general del sistema que se esta elaborando y se pueden identificar las áreas de oportunidad para cuestiones de mantenimiento o para una futura expansión. Una vez definidos los tipos de ligas se puede establecer que un modelo estratégico de dependencias que puede representarse como se muestra en la Figura 4.7

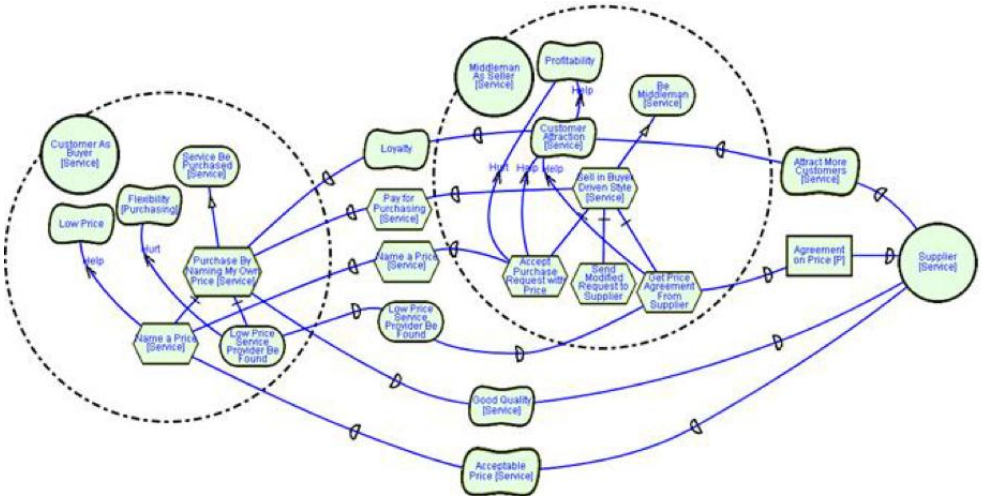


*Figura 4.7 Modelo estratégico de dependencias*

En esta etapa de la metodología, además de la realización del SRM, se procede a la elaboración de diagramas de secuencia de sistema de las tareas que en este modelo impliquen un proceso de desarrollo. La elaboración de diagramas de secuencia de sistema en esta etapa tiene la finalidad de establecer el flujo de ejecución de los procesos del sistema en cuestión.

## 4.3. Especificación del Modelo de Información de Entidades Físicas y Virtuales

El IoT es un concepto que hace referencia a la interconexión digital de objetos cotidianos con Internet (Conner, 2010). Dichos objetos podrían ser sensores, relevadores o actuadores los cuales podrían estar presentes en cualquier tipo de dispositivo de uso cotidiano como lo son refrigeradores, televisiones, equipos de sonido o bien, podría tratarse de maquinaria de precisión o maquinaria especializada de la industria.

Los sistemas de información orientados al IoT deben interactuar directamente con los dispositivos mencionados anteriormente, por ende, es importante contar con una metodología que permita establecer dicha comunicación a través de un lenguaje claro y ordenado.

El tercer paso de MeDAIC consiste en la definición de un modelo de información de las entidades físicas de las cuales existe interés por compartir algún tipo de información y su relación con otras entidades virtuales. Para elaborar el modelo se emplea el uso del Lenguaje Unificado de Modelado (UML) (Perdita Stevens, 2002).

Otro aspecto fundamental del nivel 3 en el esquema propuesto por MeDAIC es la estrecha relación que existe entre los stakeholders y los desarrolladores del sistema ya que son estos últimos quienes deben comprender a la perfección los objetivos que persiguen los skateholders al incorporar los circuitos electrónicos propios del IoT en el sistema que se está desarrollando.

La **¡Error! No se encuentra el origen de la referencia.** ilustran el mecanismo de comunicación que existe entre los elementos involucrados en una operación simple donde intervienen una serie de objetos que interactúan entre sí mediante el envío y recepción de mensajes.

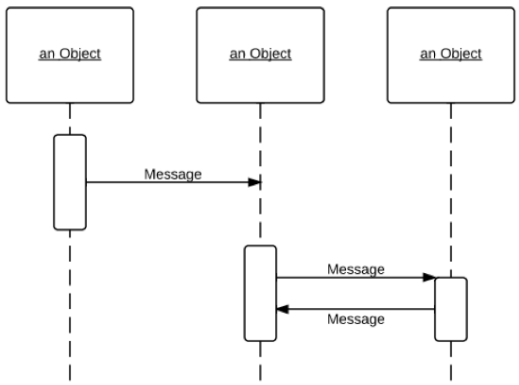


*Figura 4.8 Diagrama de Secuencia UML*

## 4.4. Definición del Formato Para Intercambio de Datos

Los sistemas de información comúnmente generan datos que se almacenan y procesan posteriormente de alguna manera ya sea mediante el empleo de algún motor de bases de datos SQL, tecnologías NO SQL, archivos binarios, archivos de texto o bien la utilización de repositorios en la nube entre otras opciones. Sea cual sea la tecnología seleccionada es importante tener claramente segmentada la sección que realizará esta tarea dentro de la funcionalidad del sistema.

La forma recomendada para incorporar la funcionalidad de almacenamiento de datos en el sistema será mediante la implementación del “servicios” mediante el uso de un gestor de bases de datos distribuidas Figura 4.9, es decir, que debe existir un objeto “Base de Datos”

que se encargue de recibir las peticiones del sistema y dicho objeto sea capaz de resolver las solicitudes sin importar que tipo de sistema de almacenamiento de datos fue elegido.

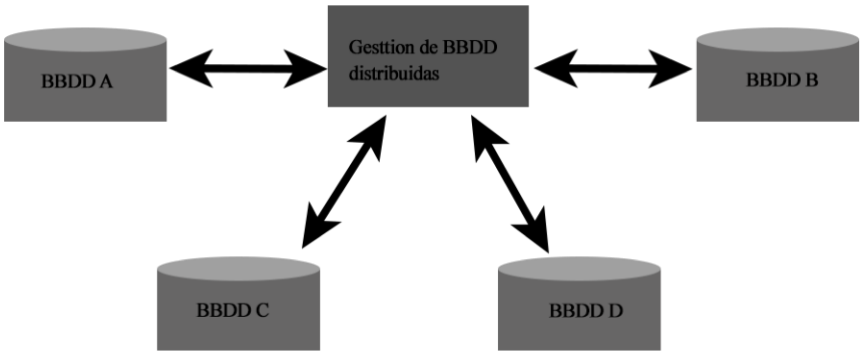


*Figura 4.9 Esquema de la base de datos distribuida*

Como elemento de salida se espera en esquema de base de datos distribuidas el cual permita al sistema ser versátil y no limitarse a un solo esquema de bases de datos. No perdamos de vista que los dispositivos móviles (Tablet, teléfonos inteligentes y otros dispositivos) con los que será manipulado el sistema cuentan con diversas y variadas opciones de almacenamiento de datos tanto síncronos como asíncronos lo que les da a las aplicaciones orientadas al IoT mayor margen de maniobrabilidad en el tema de almacenamiento e intercambio de datos.

## 4.5. Visión General de la Arquitectura

El siguiente paso en la metodología MeDAIC consiste en la creación de un plano arquitectónico del proyecto mediante el cual se pueda explicar a los distintos interesados el cómo está conformado el proyecto, cuales son los componentes más relevantes, cuales son los comportamientos que tendrán los distintos módulos del sistema y quienes son los actores que participan en cada una de las etapas.

Antes de crear la visión general de la arquitectura es necesario comprender con claridad qué es y cómo se conforma la arquitectura de software ya que esto servirá de guía a la hora de realizar los planteamientos funcionales de cada una de las etapas del proyecto.

Cómo valores de entrada se reciben los artefactos entregables que se fueron diseñando a lo largo de la metodología con la finalidad de crear el plano con distintas perspectivas que den paso a la justificación de los elementos que conforman el proyecto.

## 4.6. Construcción, Pruebas y Validación

Es la etapa de construcción del proyecto, se identifican los módulos que pasarán a producción y se procede a su construcción a través del equipo de programadores expertos los cuales darán forma y funcionalidad a cada uno de los procesos que conforman el sistema. La línea de producción se compone principalmente por analistas, equipo de programadores, equipo de testing, líder proyecto y un productor general.

- Analistas: Son los encargados de realizar el análisis de los requerimientos funcionales y de calidad del sistema que se está desarrollando, así como de plantear los escenarios deseables que deberán ser desarrollados por los programadores.
- Equipo de programadores: Son expertos que dominan los lenguajes de programación necesarios para el desarrollo del sistema.
- Equipo de pruebas: También conocidos como departamento de Aseguramiento de la calidad o QA (Quality Assurance) por sus siglas en ingles. Son los encargados de realizar las pruebas de funcionalidad y de cumplimiento de requerimientos a los módulos desarrollados por los programadores. Cabe mencionar que una prueba de QA no la puede realizar la misma persona que desarrollo el módulo que se está analizando ya que no se puede ser juez y parte en un mismo proceso.
- Líder de proyecto: Es el responsable de dar seguimiento a los avances que realizan los equipos de programadores y el equipo de testeo, así como de lograr que se cumplan los tiempos de desarrollo pactados.
- Productor general: Es el responsable del proyecto y quien representa al equipo ante los clientes e interesados que no pertenezcan al equipo de desarrollo.

Una vez terminado el proceso de construcción se realiza una junta de retroalimentación en la cual se le presenta a todos los interesados los avances y se valida que lo realizado hasta ese momento cumpla con los requerimientos solicitados por el cliente. En el caso de que existan inconsistencias en el desarrollo o bien que el cliente decida realizar modificaciones en la funcionalidad entonces el proceso regresaría a la etapa número 2 del modelo MeDAIC para realizar las modificaciones necesarias y cumplir con el ciclo de la metodología hasta llegar a este mismo punto, este ejercicio se realiza de manera cíclica hasta culminar con todos los módulos que conforman el sistema.

Una vez superado el procedimiento desarrollo de construcción, pruebas y validación y que estos no presentaran problema alguno que no amerite continuar con el ciclo de mejoras planteado en el párrafo anterior se continuará con la siguiente etapa de la metodología MeDAIC.

## 4.7. Desarrollo del Diagrama de Despliegue

Como penúltimo paso se desarrolla el diagrama de despliegue el cual modela la arquitectura en tiempo de ejecución de un sistema. Esto con la finalidad de mostrar la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en ese nodo.

Los diagramas de desplioegue están formados por varias formas UML que se muestra en la Figura 4.10. Las cajas tridimensionales, conocidas como nodos, representan los elementos básicos de software o hardware, o nodos, en el sistema. Las líneas de nodo a nodo indican relaciones y las formas más pequeñas contenidas en los cuadros representan los artefactos de software que se implementan.

Los diagramas de despliegue tienen varias aplicaciones como:

- Mostrar qué elementos de software se implementan mediante qué elementos de hardware.
- Ilustrar el procesamiento en tiempo de ejecución para el hardware.
- Proporcione una vista de la topología del sistema de hardware.

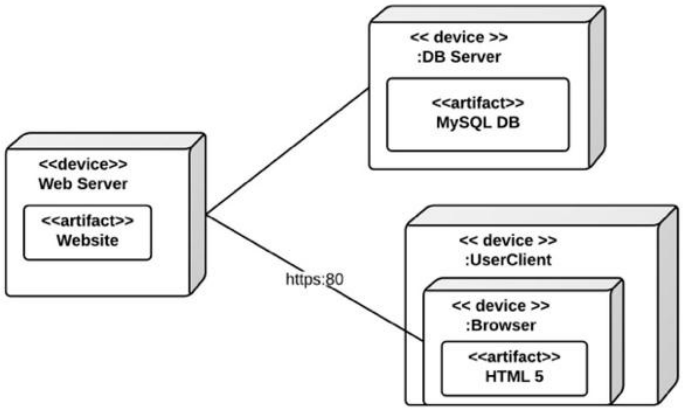


*Figura 4.10 Diagrama de despliegue*

Con el diagrama de despliegue se busca comprender la relación que tienen los elementos físicos (equipos de cómputo, dispositivos inteligentes, relevadores y actuadores entre otros) con los procesos lógicos que interactúan con ellos.

## 4.8. Implementación del Sistema

Como último paso de la metodología MeDAIC se encuentra la etapa de implementación y es cuando se realiza la instalación del sistema resultante en su lugar de operación. El responsable de esta tarea realizará y validará una lista de elementos mínimos necesarios con los que el cliente deberá contar en su establecimiento para realizar una correcta implementación. La lista de elementos mínimos necesarios surgirá a partir del diagrama de despliegue.

## 4.9. Conclusión

Como puede ver, cada uno de los niveles que conforman la metodología MeDAIC establece una serie de metas específicas las cuales buscan construir una serie de artefactos a medida que se va avanzando en la resolución del proyecto, también se define claramente qué tipo de conocimientos debe tener el participante que se involucra en cada una de las etapas, con ello se busca usar de manera eficiente los recursos humanos y materiales con lo que cuenta la organización.

La metodología MeDAIC busca ser una alternativa seria para los desarrolladores que decidan incursionar en el mundo de desarrollo de software orientado al IoT buscando en todo momento que el presente trabajo de investigación les aporte los medios necesarios para la gestión, administración y desarrollo de su proyecto IoT.

# Capítulo 5

# 5. Pruebas

Todo proceso de innovación tecnológica debe tener un proceso de construcción de prototipos y un proceso de pruebas en donde mediante el uso de métricas claramente definidas se pueda realizar un monitoreo del comportamiento, medir el rendimiento del prototipo y determinar su viabilidad para su posterior implementación.

En el presente capítulo se presentarán las pruebas realizadas a la metodología MeDAIC las cuales constan de pruebas de laboratorio y pruebas de campo, estas últimas se desarrollaron en la empresa LUXELARE y Spiral Media Labs durante una estancia que se realizó con esta finalidad.

## 5.1. Características de las pruebas

Para comprobar la eficiencia de MeDAIC fue necesario establecer una serie de métricas cuyos resultados pudieran ser comparados con otras metodologías ya existentes con la finalidad de identificar las fortalezas, debilidades y aquellas áreas de oportunidad en las que se pueda mejorar el proceso de planeación y administración de sistemas orientados al IoT.

Para cumplir con este objetivo la metodología MeDAIC fue implementada en dos proyectos de desarrollo de software, sin embargo, en el presente trabajo solo se hablará de la estructura de uno de estos proyectos al cual se le monitoreo su comportamiento y evolución de los sistemas tomando en cuenta las métricas definidas con el objetivo de llegar a una conclusión ya sea que le metodología funciono satisfactoriamente o bien que la metodología no cumplió con las expectativas mínimas deseada.

A continuación, se mencionarán las métricas definidas para calificar la funcionalidad de la metodología, cabe mencionar que dichas métricas fueron seleccionadas después de analizar detalladamente las métricas utilizadas tradicionalmente en otras metodologías de desarrollo de Software con la finalidad de que este ejercicio fuera lo más apegado a la realidad.

## 5.2. Métricas establecidas

Las métricas son cualquier conjunto de medidas destinadas a conocer el tamaño o alcance de un software o sistema de información. Las métricas utilizadas en el presente proyecto son las siguientes:

- Cantidad de equipamiento dedicado a la transformación.
- Tiempo invertido por los directores en actividades de trasformación.
- Tiempo promedio de conceptualización o definición de los nuevos proyectos IoT.
- Número de proyectos autorizados para implementación.
- Tiempo promedio de desarrollo del proyecto.
- Tiempo perdido al iterar la metodología en el proceso de refinamiento.
- Porcentaje de éxito (Número de proyectos exitosos frente a los fallidos).
- Nivel de experiencia entre los desarrolladores
- Tiempo perdido al pasar de un nivel a otro (Niveles de la metodología MeDAIC)
- Tiempo perdido durante el refinamiento del sistema (ir del paso 6 al paso 2 de la metodología MeDAIC)

### 5.2.1. Caso de prueba 1: My Smart Hotel

El Tecnológico Nacional de México lleva acabo el Encuentro Nacional Estudiantil de Innovación Tecnológica (ENEIT) con el objetivo de desarrollar proyectos disruptivos o incrementales que fortalezcan las competencias creativas, emprendedoras e innovadoras de los participantes a través de la transferencia tecnológica y comercialización, dando respuesta a las necesidades de los sectores estratégicos del país.

Se desarrolló una aplicación orientada al IoT denominado “My Smart Hotel” con el objetivo de participar en el encuentro nacional en la modalidad “Reto Empresarial”, el proyecto debía cumplir con las siguientes características:

1) Aplicación orientada a la industria hotelera y el turismo
2) La aplicación debe desarrollarse para sistemas IOS y Android
3) Debe ser innovador
4) Debe contar con documentación que lo soporte

A continuación, se exponen de manera resumida el caso de prueba.

#### 5.2.1.1. Paso 1: Definición de Propósitos y Requisitos de Alto Nivel

**¿Porque se construye el proyecto?**

El proyecto busca resolver una problemática planteada por la industria hotelera y la Secretaría de Turismo en donde se requiere la elaboración de un sistema que incentive a huéspedes del hotel a consumir productos y servicios de la localidad en la que se encuentre.

El sistema también deberá permitir a los huéspedes del hotel acceder a servicios proporcionados por el propio hotel como lo es servicio a la habitación, control de algunos componentes de la habitación mediante una aplicación instalada en su celular, poder realizar quejas y sugerencias entre otros servicios.

**¿Qué beneficios espera obtener?**

Incrementar la experiencia de los huéspedes brindándoles la capacidad de controlar algunos elementos funcionales de su habitación, así como el poner a su disposición una oferta turística variada a través de una aplicación instalada en su teléfono inteligente.

Mediante la aplicación móvil el cliente podrá realizar las siguientes actividades:

- Abrir la puerta de la habitación
- Controlar la iluminación de la habitación
- Solicitar servicio a la habitación
- Controlar la temperatura del aire acondicionado
- Consultar la oferta de restaurantes aledaños al hotel
- Consultar la oferta de museos aledaños al hotel
- Consultar la oferta de actividades culturales de la ciudad
- Consultar indicaciones de ruta para llegar a los destinos antes mencionados
- Solicitar un Uber

El cliente principal del proyecto (el hotel) podrá monitorear el comportamiento de los clientes en sus habitaciones mediante la interpretación de los datos que proporcionen los sensores y actuadores que hacen posible que el huésped interactúe con la habitación. El administrador del sistema tendrá la capacidad de activar o desactivar los actuadores y relevadores de las

habitaciones de manera remota, así como también podrá recibir alertas cuando se detecten anomalías en los datos.

Los requisitos funcionales y de calidad se enlistan en el Acta de constitución del proyecto misma que servirá como soporte para marcar los compromisos y establecer los límites del proyecto acordados con los interesados.

#### 5.2.1.2. Paso 2: Refinamiento de los requisitos de alto nivel

Se identifican aquellos requisitos funcionales y de calidad candidatos a ser utilizados durante el proceso de construcción del sistema. El desarrollador deberá filtrar la lista de requisitos usando su criterio y experiencia, determinando cuales requisitos de la lista plantada en el paso anterior cumplen tareas similares unificando procesos o bien determinando cuales requisitos son innecesarios en el proceso de construcción.

Una vez que se cuenta con la lista de requisitos funcionales y de calidad refinados se procede con la creación del modelo estratégico de dependencias i* el cual sirve como guía visual al momento de determinar el funcionamiento y la relación que tienen los requisitos entre sí. A continuación, se presenta un ejemplo en el proceso de creación del modelo estratégico de dependencias i* para la tarea de check-in y check-out dentro de la funcionalidad del sistema My Smart Hotel.

**Modelo estratégico de dependencias i***

El sistema deberá ser capaz de realizar la acción de check-in y check-out a los huéspedes del hotel y mediante esta acción el usuario podrá acceder a los beneficios que la aplicación móvil le proporciona al interior del hotel. La Figura 5.1ilustra el procedimiento mediante un diagrama SDM

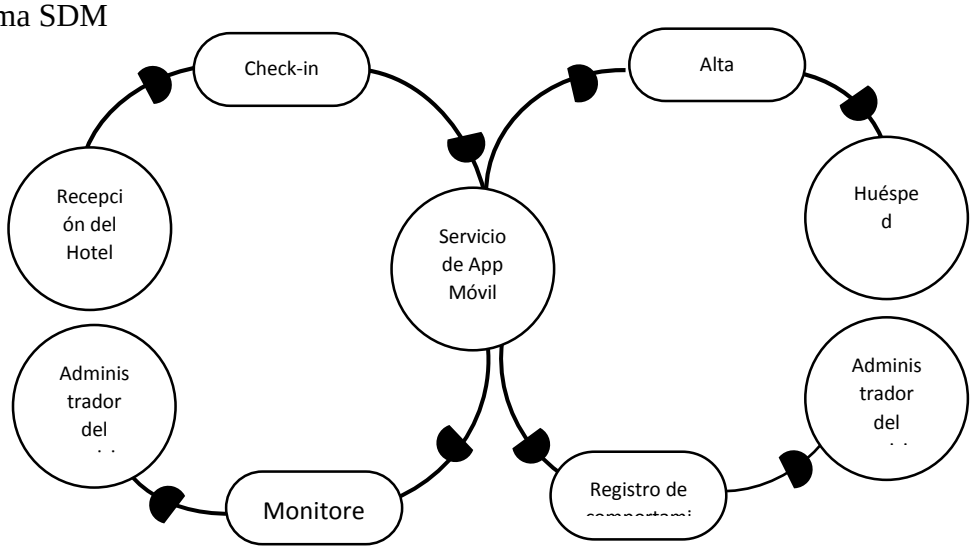


*Figura 5.1 SDM de la tarea check-in del sistema*

Como ya se ha mencionado anteriormente la funcionalidad del diagrama SDM radica en especificar las relaciones y dependencias que existen entre los actores y las relaciones que estos tienen con la aplicación, el flujo de la información está indicado por los marcadores distintivos de la metodología i*.

El mismo procedimiento se aplica para todos los requisitos de alto nivel que se tengan identificados, se realiza un modelo estratégico de dependencia por cada escenario planteado.

#### 5.2.1.3. Paso 3: Modelo de información de entidades físicas y virtuales

En este paso del proceso se genera un modelo estratégico racional a partir del modelo estratégico de dependencias generado en la etapa anterior. El SDM se muestra con un grado más elevado de detalle, las metas que tiene cada actor involucrado en el sistema y con qué recursos cuenta, así como también se indican las restricciones o tareas que deben de cumplir para lograr el objetivo planteados.

En la Figura 5.2 se presenta un diagrama SRM donde se describe la forma en que interactúa un usuario con cualquier componente IoT instalado en el hotel, como primera instancia se debe realizar el proceso de autenticación el cual es habilitado al momento de hacer check-in en el hotel.

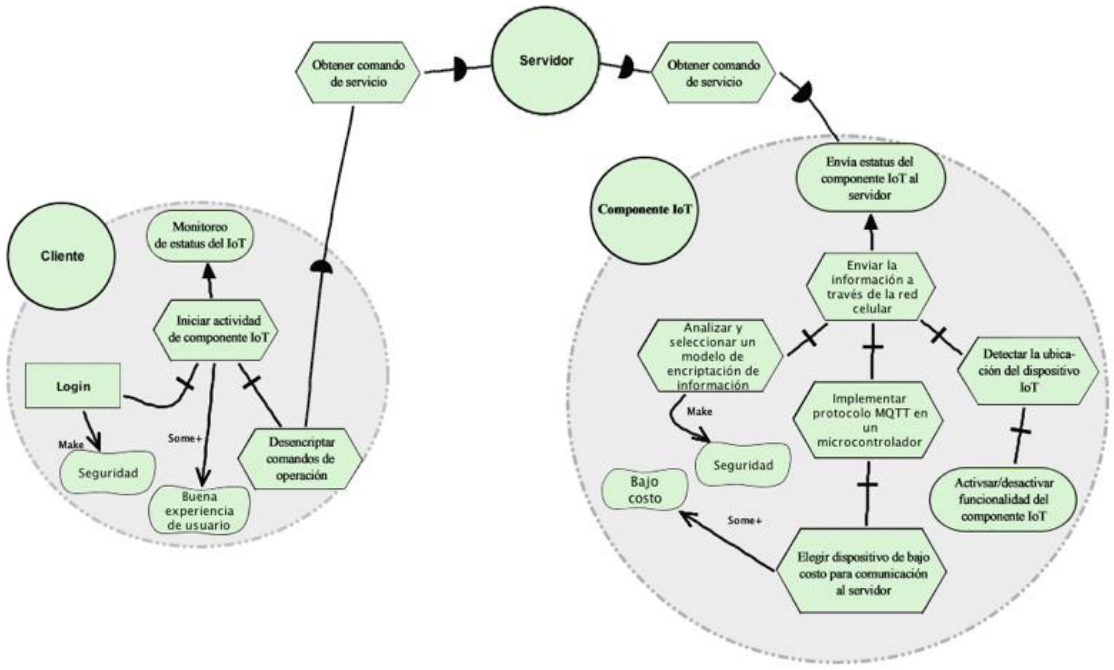


*Figura 5.2 Sistema de monitoreo de componentes IoT*

Como se puede observar el modelo relacional SRM a diferencia del modelo de dependencias i* es más detallado ya que se representa mediante un circulo al Actor del que se está hablando y de igual manera se representa mediante una circunferencia con línea punteada el área de afectación que esta representa.

En el modelo se puede observar que la tarea “Iniciar actividad de componente IoT” es el medio para que el actor Cliente logre cumplir con su meta de controlar los circuitos IoT a los que tiene acceso. La tarea es descompuesta para su realización con el recurso “Login” y la tarea “Descifrado de operación” la cual tiene una dependencia con el servidor para poder ser realizada (necesita de ésta para poder obtener los datos para su descifrado); además de esto, se muestra como la tarea “Iniciar actividad de componente IoT”.

El actor Servidor se muestra en el modelo como un agente intermediario entre los actores Cliente y Componente IoT (Los actores mostrados anteriormente en la Figura 5.2 son representados como Cliente). Del Servidor depende que el cliente reciba la información sobre el estatus y actividad del componente IoT, así como el servidor del envío de la información por parte la unidad vial para transmitir la información a todos los clientes suscritos.

El actor “Componente IoT” obtiene el estatus del componente y lo envía al servidor. Para lograrlo necesita cumplir con la tarea “Enviar la información a través de la red celular”. A su vez para lograr este paso es necesario completar tres sub tareas que son:

- Analizar y seleccionar modelos de encriptación
- Detectar la ubicación del dispositivo IoT
- Implementar el protocolo MQTT en un microcontrolador.

Para implementar el protocolo MQTT en un microcontrolador es necesario que primero sea seleccionado un microcontrolador de bajo costo.

**Diagrama de secuencia del sistema**

A continuación, se elabora el diagrama de secuencia de sistema. Los diagramas que fueron elaborados con base a los procesos de comunicación y desarrollo que expresa el modelo estratégico racional puede ser observado en la Figura 5.3.

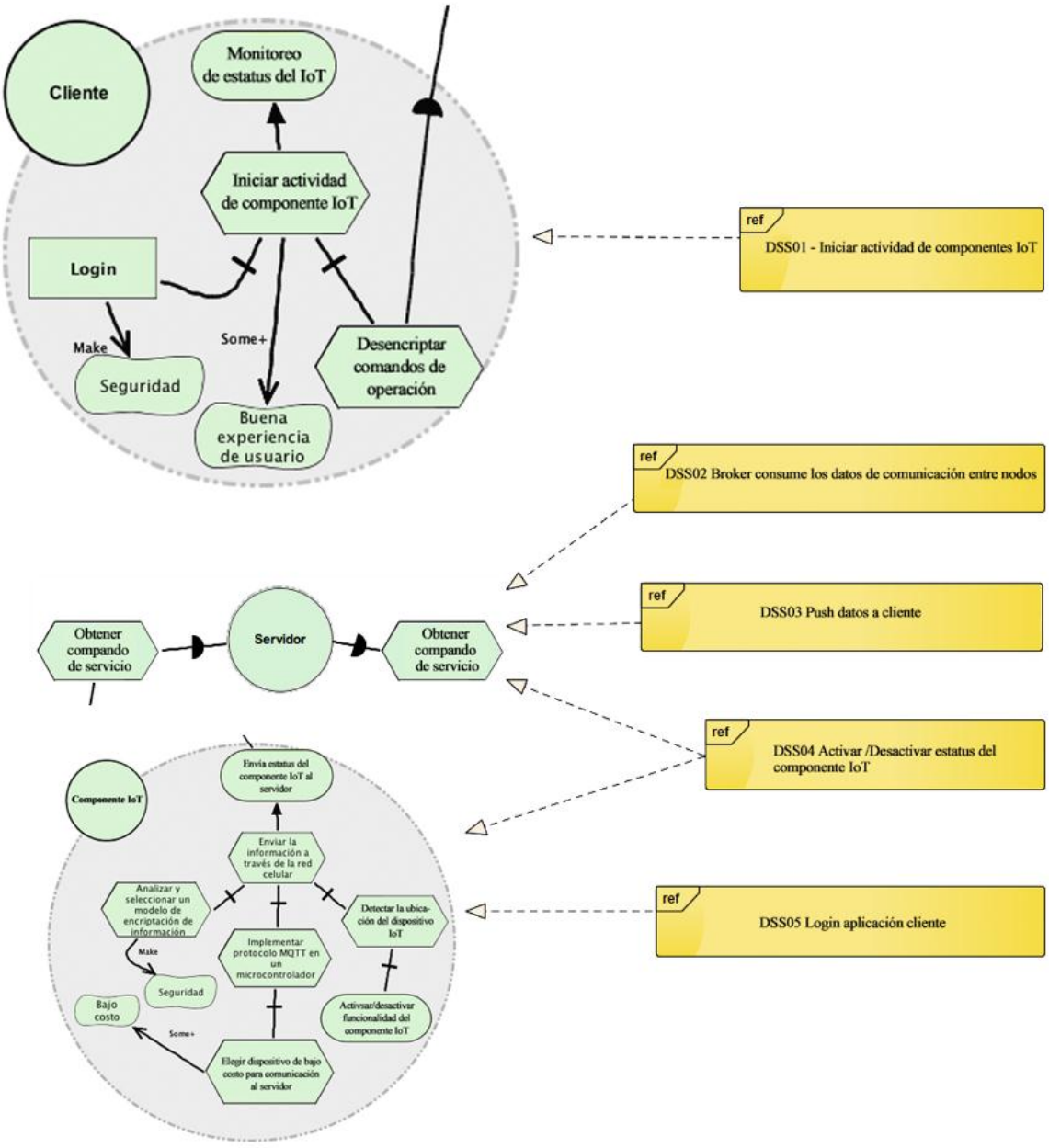


*Figura 5.3 Trazabilidad de diagramas de secuencia con modelo estratégico racional.*

Si el sistema que se va a desarrollar es demasiado grande, es una buena práctica el segmentar el sistema en pequeños elementos funcionales para posteriormente unirlos en un solo sistema más grande.

Una vez definida la trazabilidad se procede a realizar el diagrama de secuencia el cual indicará a los desarrolladores la forma en que se espera se comporte esta sección del sistema. Para realizar esta tarea se recomienda usar UML.

En la Figura 5.4 se muestra la actividad para obtener el estatus de los componentes IoT que pueden ser accedidos por parte del usuario (en este caso representado como “Publicador”). Se toman lecturas cada 2 segundos y posteriormente son enviadas al servidor (denominado como Broker)

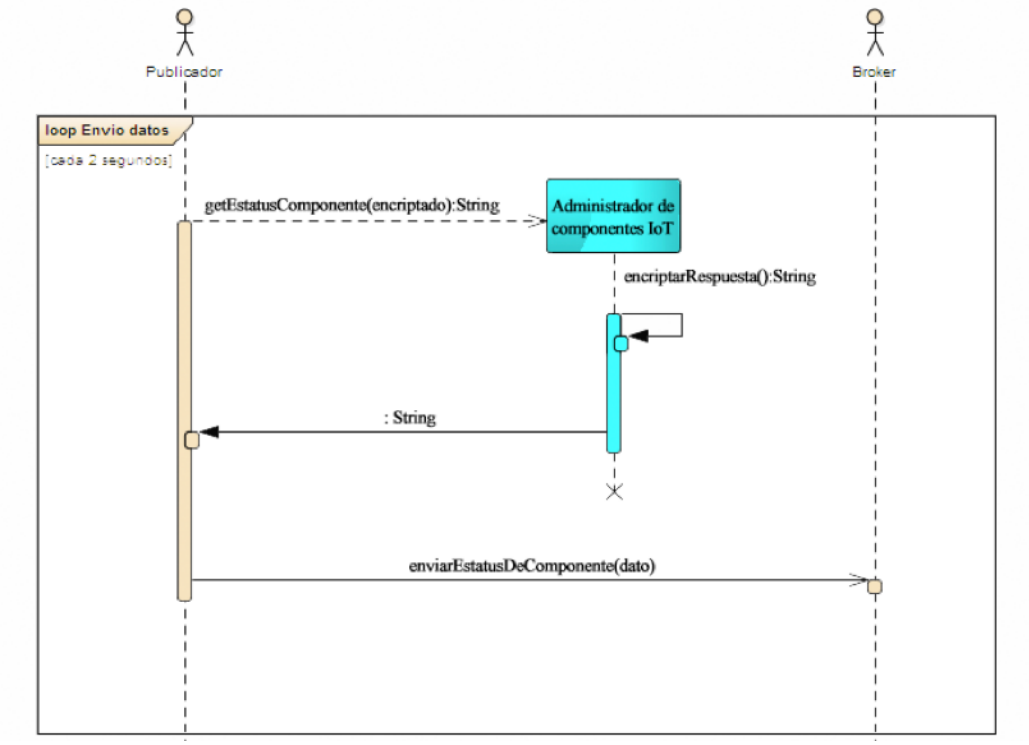


*Figura 5.4 DSS01 Iniciar componentes IoT.*

#### 5.2.1.4. Paso 4: Definición del formato para intercambio de datos

En el paso numero 4 del modelo MeDAIC se define el formato para el intercambio de datos entre las entidades físicas y virtuales, es decir, la comunicación que existe entre el sistema de información orientado al IoT y los circuitos, actuadores y relevadores conectados al entorno de desarrollo. Para lograr dicha comunicación se propone el uso de JSON por ser un formato ligero pero potente y que es soportado por múltiples plataformas ya que JSON es un subconjunto de notaciones literales de objetos de JavaScript muy popular y muy ligero en su implementación.

JSON es un formato de texto sencillo para el intercambio de datos ya que se trata de un subconjunto de la notacion literal de objetos de JavaScript. es por ello que recomiena su uso en el paso de mensajes entre los componentes del sistema orientado al IoT. A continuacion se ilustra un ejemplo de la estructura JSON obtenida de (Crockford, 2006).

```
{"habitación": {
  "id": "file",
  "value": "File",
  "componentesIoT": {
    "menuitem": [
      {"value": "lampara", "onclick": "getStatus()"},
      {"value": "clima", "onclick": "getStatus()"},
      {"value": "seguroPuerta", "onclick": "getStatus()"}
    ]}
}}
```

#### 5.2.1.5. Paso 5: Visión general de la arquitectura

La arquitectura publicador-suscriptor mostrada en la Figura 5.5 fue elegida para el desarrollo del proyecto por la naturaleza de la IoT.

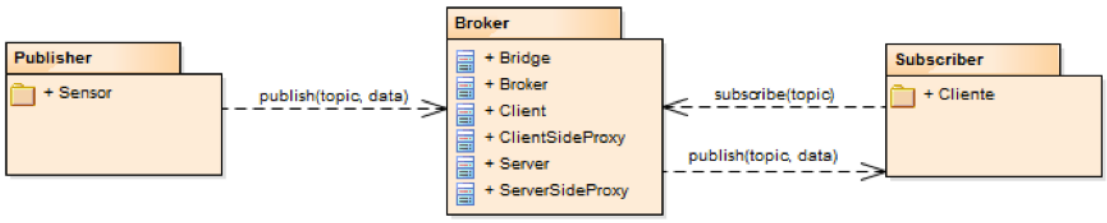


*Figura 5.5 Arquitectura publicador subscriptor.*

El uso de la arquitectura publicador-suscriptor es recomendada debido a la facilidad que esta arquitectura brinda para que un gran número de clientes puedan conectarse al servicio que brinda el sistema "My Smart Hotel". Permite que múltiples equipos de desarrollo creen sus propios clientes con la única restricción de respetar el protocolo de comunicación establecido por el publicador.

#### 5.2.1.6. Paso 6: Construcción, pruebas y validación

En este paso se procede a la construcción del Software; esto incluye el desarrollo de prototipos hardware y definición del alcance del proyecto mediante la interacción con los stakeholders con la finalidad de cubrir todas las necesidades básicas con las que debe cumplir el proyecto.

Si durante el proceso de desarrollo del software se detectan anomalías u omisiones con respecto a componentes que debieron ser agregados al proyecto entonces se regresa a al paso numero 2 para corregir las anomalías y continuar con una nueva iteración.

Al no existir anomalías durante el proceso de construcción del proyecto se procede al paso siguiente en la lista de la metodología MeDAIC.

#### 5.2.1.7. Paso 7: Desarrollo de diagrama de despliegue

Como penúltimo paso se tiene el desarrollo de un diagrama de despliegue. La elaboración del diagrama de despliegue tiene la finalidad de capturar las relaciones entre un particular elemento conceptual o físico que conforman el sistema modelado y los artefactos software que contienen (OMG, 2013). Sirve como guía para llevar a cabo la implantación del sistema en cuestión.

A través del diagrama de despliegue se indican los requerimientos tecnológicos con los que debe contar el cliente en su establecimiento para propiciar el correcto funcionamiento del sistema. Es importante resaltar que esta es una actividad en la que el desarrollador y los stakeholders deben colaborar muy de cerca ya que es una de las etapas más cruciales durante el desarrollo del proyecto. En la Figura 5.6 se puede observar la distribución topológica del sistema My Smart Hotel. El diagrama de despliegue muestra al nodo de propagación con su ambiente de ejecución en un microcontrolador, muestra como Mosquitto (bróker de eclipse de código abierto) se ejecuta en el servidor Unix de Eclipse y provee su servicio a través del puerto 1883, también muestra el ambiente de ejecución un servidor Unix que se encarga de comunicarse con Mosquitto utilizando el protocolo MQTT. Posteriormente utilizando un web socket se envía la información a los diversos clientes que puedan existir.

Los clientes tienen un entorno de ejecución web y móvil el cual es habilitado por la recepción del hotel al momento de que el cliente realiza el check-in, a continuación, se le asigna una habitación y con ello el control total de los componentes IoT de la habitación asignada. El huésped podrá controlar su entorno en la habitación mediante el uso sus dispositivos inteligentes ya sea un Smartphone, una Tablet o una computadora.

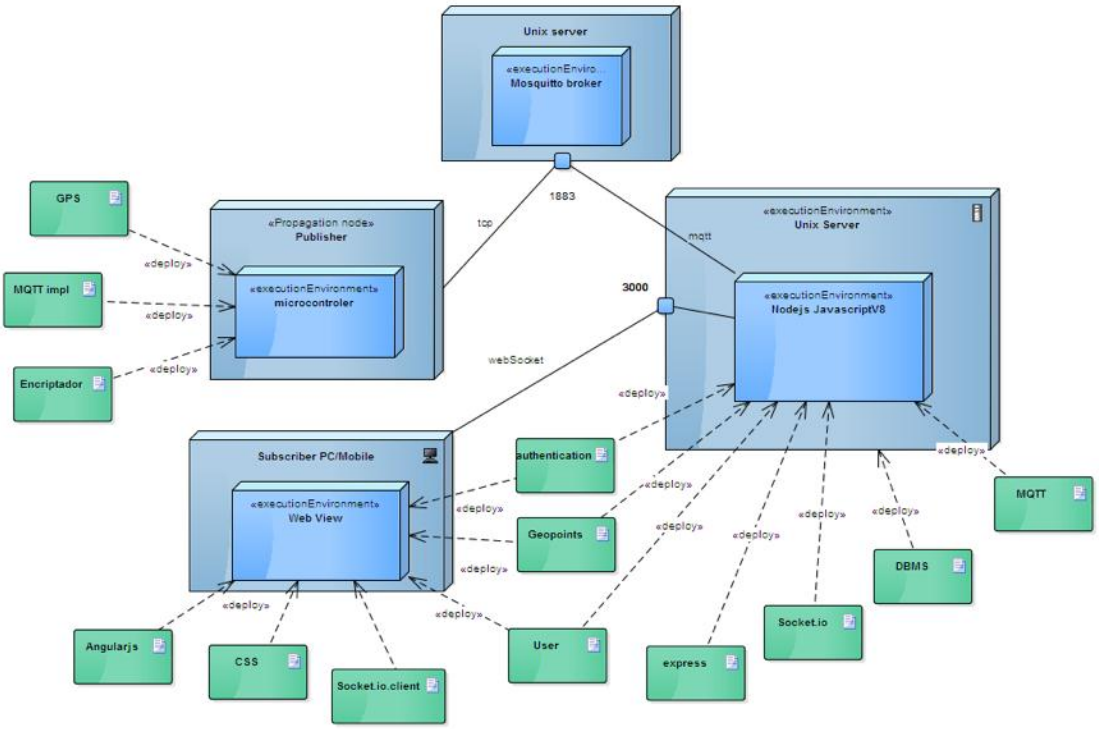


*Figura 5.6 Diagrama de despliegue del sistema My Smart Horel.*

### 5.2.1.8. Paso 8: Implementación del sistema

Como último paso se procede con la instalación del sistema usando como base el diagrama de despliegue. El sistema My Smart Hotel consta de 2 partes, la primera es un entorno web mediante el cual los administradores puedan monitorear el comportamiento de los sensores en cada una de las habitaciones y la segunda parte del sistema es una aplicación móvil disponible para IOS y Android. En la Figura 5.7 se ilustra el logotipo de la aplicación móvil que se desarrollo como caso de prueba.


*Figura 5.7 Logotipo My Smart Hotel*

URL Google Play: https://play.google.com/store/apps/details?id=xyz.mrarc.smarthotel

Cabe mencionar que la aplicación para los dispositivos móviles es completamente gratuita, aunque solo funciona si el usuario realiza un check-in en algún hotel ofrezca este servicio.

**Aplicación móvil**

My Smart Hotel está diseñado para resolver dos problemáticas propuestas por la Secretaria de Turismo Federal y la Industria Hotelera del país en donde se busca incentivar el turismo ofreciendo una experiencia más completa para los huéspedes de los hoteles. La aplicación provee de control total a los usuarios para controlar ya sea la iluminación de la habitación, regular la temperatura del aire acondicionado, solicitar servicio a la habitación entre otros beneficios.

Otro beneficio de la aplicación My Smart Hotel es que le permite al usuario buscar lugares de interés a 10km a la redonda de su ubicación actual ya que el sistema toma como referencia la posición GPS para realizar búsquedas de dichos sitios como lo son: restaurantes, bares, cines, centros comerciales, parques o incluso hasta eventos especiales. En la Figura 5.8 se ilustran las distintas pantallas que conforman la aplicación móvil My Smart Hotel.


*Figura 5.8 Pantallas de la aplicaciòn My Smaer Hotel*

El sistema My Smart Hotel es un proyecto totalmente funcional con el que se representó dignamente al Instituto Tecnológico de Culiacán en el concurso nacional ENEIT 2017 que se llevó a cabo en Tlalnepantla de Baz, Estado de México siendo acreedor del 1er lugar en la categoría "Proyecto Empresarial" y que llamo la atención de la Secretaría de Turismo donde se solicitó madurar el proyecto para ser implementado en los "Pueblos Mágicos". Actualmente el proyecto ya cuenta con un registro de propiedad intelectual.

### 5.2.1.9. Métricas para evaluación del método

En la Tabla 1 se presentan las métricas y resultados obtenidos durante el desarrollo de los proyectos involucrados. Esto incluye los proyectos desarrollados en el laboratorio así como los que se realizaros durante la estancia en la empresa LUXELARE.

*Tabla 1 Metricas y resultados de los proyectos*

| | | | |
|---|---|---|---|
| 1. | Cantidad de equipamiento dedicado a la transformación. | 3 computadoras | • MacBook Pro (13-inch, Early 2011) 2.8 GHz Intel Core i7, 8 GB 1333 MHz DDR3, Intel HD Graphics 3000 512 MB<br>• Hp Elitebook 840 G5 Core I7 8550u 8gb Ram 256 Gb Ssd<br>• Toshbia Intel Ci3 Exp 16gb 15.6 Led Touch |
| 2. | Tiempo invertido por los directores en actividades de trasformación. | 2 horas diarias | El gerente de la empresa y jefe de producción reservaba 2 horas diarias (8:00am a 9:00am y de 5:00pm a 6:00pm) para la revisión de avances y planteamiento de nuevas metas. |
| 3. | Tiempo promedio de conceptualización o definición de los nuevos proyectos IoT. | 96 horas | |
| 4. | Número de proyectos autorizados para implementación. | 3 proyectos | My Smart Hotel<br>Sistema de monitoreo de parcelas<br>Sistema pronosticador de precios |
| 5. | Tiempo promedio de desarrollo del proyecto. | 820 horas (4 meses) | |
| 6. | Tiempo perdido al iterar la metodología en el proceso de refinamiento. | 20% | Se comparó el tiempo promedio de desarrollo de una tarea completa hecha con Scrum y se comparo contra el tiempo promedio de desarrollo de una tarea con MeDAIC y se encontró que exite un retardo promedio de entre 15 y 20 % |
| 7. | Numero de desarrolladores involucrados | 4 personas | 1 director de proyecto<br>3 desarrolladores |
| 8. | Nivel de experiencia entre los desarrolladores | | 3 desarrolladores senior<br>1 desarrollador semi senior<br>4 desarrolladores junior<br>1 ingeniero en aeronáutica |

### 5.2.2. Tabla comparativa entre metodologías

A continuación, en la Tabla 2 se presenta una comparativa entre MeDAIC y las principales metodologías de desarrollo de software tradicionales. Los rubros comparativos son los habituales en enstas practicas y se enfatiza mucho en la calidad del producto terminal.

*Tabla 2 Tabla comparativa de metodologías*

| | MSF | RUP | XP | MeDAIC |
|---|---|---|---|---|
| Desarrollo de Software Iterativo | ✓ | ✓ | ✓ | ✓ |
| La calidad como un objetivo | ✓ | ✓ | | ✓ |
| Verificación continua de calidad | ✓ | ✓ | ✓ | ✓ |
| Requerimientos del cliente | ✓ | ✓ | ✓ | ✓ |
| Arquitectura conducida | ✓ | ✓ | ✓ | ✓ |
| Enfocado en equipo | ✓ | ✓ | ✓ | ✓ |
| Programación en par | | | ✓ | |
| Adaptación con restricciones | ✓ | ✓ | | |
| Administración de cambios y configuraciones | | ✓ | | ✓ |
| Administración del riesgo | ✓ | | | ✓ |
| Experiencia de los desarrolladores | ✓ | | | ✓ |

## 5.3. Conclusión

Es preciso mencionar que al principio fue complicado seguir las indicaciones que marca la metodología MeDAIC ya que siempre está presente el fantasma de la experiencia que nos dicta viejos hábitos al momento de elaborar la documentación derivado de la vasta experiencia en el uso de marcos de trabajos tradicionales, sin embargo, una vez que los desarrolladores comprendieron la mecánica todo fue sobre ruedas y se pudo cumplir con el objetivo planteado que fue el desarrollo de una aplicación móvil así como la implementación de la metodología MeDAIC en su totalidad.

# Capítulo 6

# 6. Conclusiones y trabajo futuro

Es innegable que las metodologías de desarrollo de software les permiten a los programadores contar con un marco de trabajo que les facilite la administración de sus proyectos, vemos a estas herramientas administrativas como una gran ventaja al implementarlo en procesos que se deben controlar de manera minuciosa y con ello poder determinar los porcentajes de avance o retraso que se ciernen sobre el trabajo que se realiza. Tomando en cuenta esta reflexión se puede decir que el establecer una Metodología de Desarrollo de Aplicaciones del Internet de las Cosas no solo era necesario por la creciente demanda que existe en el mercado en cuanto a desarrollos de aplicaciones IoT se refiere, sino que también es pertinente ya que es un campo sumamente explotado en la parte programática, pero poco explorado en la parte organizacional dejando la administración de procesos a metodologías ya obsoletas.

## 6.1. Conclusiones

Por su naturaleza el proyecto MeDAIC representa un aporte en el área de la administración de proyectos IoT y busca apoyar a los programadores y diseñadores de software en el proceso de planeación de sus proyectos, cada una de las 8 etapas que componen al proyecto está especialmente desarrollada para cumplir con los requerimientos generales que se presentan habitualmente en los proyectos de ingeniería de Software.

Al implementar la metodología MeDAIC en proyectos de la iniciativa privada se logró detectar que esta se adapta perfectamente a proyectos orientados al IoT ya que los parámetros que se definieron para su medición y seguimiento fueron arrojando resultados satisfactorios en cada una de las etapas, además cabe mencionar que los equipo involucrados en la implementación de la metodología se mostraron receptivos y satisfechos con la estructura propuesta por MeDAIC.

Se logró dar notoriedad a los elementos IoT que intervienen en los proyectos desde el momento de la planeación y visibilidad durante todo el proceso de desarrollo e implementación del proyecto dando como resultado que los dispositivos IoT pudieran ser atendido con un mayor énfasis.

## 6.2. Trabajos futuros

Como trabajos futuros se pretende dar seguimiento a los proyectos que ya están implementando la metodología MeDAIC en búsqueda de perfeccionar las etapas que conforman a dicha metodología ya que partimos de la primicia de que "Todo es perfectible".

Se buscará establecer un número mayor de parámetros de medición que arrojen mayor certeza en los resultados de seguimiento y que estos puedan ser comparables con otros proyectos similares.

Se propondrá la metodología MeDAIC en más proyectos funcionales en conjunto con la iniciativa privada con la finalidad de recibir retroalimentación por parte de los usuarios y poder evolucionar la metodología a través de los comentarios y observaciones que pudiesen surgir.

# Bibliografía


Cadavid, A. N. (2013). Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*.

Campos, S. G. (2015). Programación Extrema: Prácticas, Aceptación y Controversia. *CULCyT*.

Canós, J. H. (2012). Metodologías ágiles en el desarrollo de software.

Canós, J. H. (2012). Metodologías ágiles en el desarrollo de software. *roa.ult*.

Cervantes Ojeda, J. &. (2012). Taxonomía de los modelos y metodologías de desarrollo de software más utilizados. *Universidades*.

Conner, M. (2010). *Sensors empower the "Internet of Things".*

Crockford, D. (2006). The application/json media type for javascript object notation (json). *ietf*.

Crockford, D. (2008). *JavaScript: The Good Parts, vol. 44.*

Bahga, A. &. (2014). *Internet of Things: A hands-on approach.* Universities Press.

Boehm, B. W. (1988). A spiral model of software development and enhancement.

Deming, W. E. (1982). *Quality, productivity, and competitive position (Vol. 183).*

Dijkstra, E. W. (1968). Letters to the editor: go to statement considered harmful. *Communications of the ACM*.

Doran, G. T. (1981). There's a S.M.A.R.T. way to write management's goals and objectives.

informática, H. d. (2011). *Historia de la informática*. Obtenido de Museo de la informática: https://histinf.blogs.upv.es/2011/01/04/la-crisis-del-software/

Isijara, R. A. (2015). Diseño y aplicacion de una metodología para el desarrollo de aplicaciones de internet. *ItCuliacan*.

istarwiki.org. (2011). i* guide.

Juran, J. M. (1988). *Juran on planning for quality.*

Madisetti, A. B. (2014). *Internet of Things: A Hands-On Approach.*

Méndez Nava, E. M. (2006). *Modelo de evaluación de metodologías para el desarrollo de software.* Caracas, Venezuela.

OMG. (2013). *OMG Uni ed Modeling Language: Version 2.5*. Obtenido de OMG Uni ed Modeling Language: Version 2.5: http: //www.omg.org/spec/UML/2.5/Beta2/. 23, 32

Perdita Stevens, R. P. (2002). *Utilización de UML en Ingeniería del Software con Objetos y Componentes.*

Pérez, O. A. (2011). Cuatro enfoques metodológicos para el desarrollo de Software RUP–MSF–XP-SCRUM. *NVENTUM*.

Pressman, R. (2006). *Ingeniería de software. Un enfoque práctico.*

Pressman, R. S. (1988). *Ingeniería del software.* McGraw Hill.

Pressman, R. S. (1988). *Ingeniería del software.* Mc Graw Hill.

Royce, W. W. (1987). Managing the development of large software systems: concepts and techniques. In Proceedings of the 9th international conference on Software Engineering.

Scrum. (2019). *Conectar*. Obtenido de Conectar: https://blog.conectart.com/la-metodologia-scrum-scrum-methodology/

Sametinger, J. (1997). *Software engineering with reusable components.* Springer Verlag.

Tinoco Gómez, O. &. (2010). Criterios de selección de metodologías de desarrollo de software. *Industrial Data*.

Tinoco Gómez, O. R. (2010). Criterios de selección de metodologías de desarrollo de software. *Industrial Data*, 5.

Zuser, W. H. (2005). *Software quality development and assurance in RUP, MSF and XP.*