DOI: 10.24275/uama.5813.7540

UNIVERSIDAD AUTÓNOMA METROPOLITANA

Casa abierta al tiempo

UNIDAD AZCAPOTZALCO, CASA ABIERTA AL TIEMPO
DIVISIÓN DE CIENCIAS Y ARTES PARA EL DISEÑO

Método de selección de técnicas de levantamiento de requerimientos para el desarrollo de Software con un enfoque de Experiencia de Usuario

Darío Emmanuel Vázquez Ceballos

Tesis para optar por el grado de Doctor en Diseño
Posgrado en Diseño y Visualización de la Información:

Directora de tesis: Dra. Marcela Esperanza Buitrón de la Torre
Codirector de tesis: Dr. Edwing Antonio Almeida Calderón
Miembros del jurado:
Dra. Selene Marisol Martínez Ramírez
Dra. Bibiana Obregón Quintana
Dr. Roman Anselmo Mora Gutiérrez

México CDMX 22 de marzo de 2021

CyAD
Ciencias y Artes para el Diseño

Programa de investigación: Doctorado en Diseño y Visualización de la Información.

LGAC: Sistemas de Visualización en Información Científica.

Tesis: Método de selección de técnicas de levantamiento de requerimientos para el desarrollo de Software con un enfoque de Experiencia de Usuario .

M. en C. Dario Emmanuel Vázquez Ceballos.

Directora de tesis: Dra. Marcela Esperanza Buitrón de la Torre
Codirector de tesis: Dr. Edwing Antonio Almeida Calderón

Miembros del jurado:
Dra. Selene Marisol Martínez Ramírez
Dra. Bibiana Obregón Quintana
Dr. Roman Anselmo Mora Gutierrez

22 de marzo de 2021

AGRADECIMIENTOS

Groucho Marx decía que encontraba a la televisión muy educativa porque cada vez que alguien la encendía, él se iba a otra habitación a leer un libro. Utilizando un esquema similar, quiero agradecer a Word de Microsoft el haberme forzado a utilizar LaTeX.

A todos los que la presente vieren y entendieren.
Inicio de las Leyes Orgánicas. Juan Carlos I.

A mis padres: Azucena y Gerardo.
A mis hermanas: Talia, Dennys y Miriam.
Gracias por todo el cariño y apoyo en esta etapa académica de mi vida.

A la UAM
Gracias por brindarme las facilidades que me permitieron realizar esta tesis.

A la **Dra. Marcela Esperanza Buitrón de la Torre** y al **Dr. Edwing Antonio Almeida Calderón.**
Les agradezco su tiempo y esfuerzo en guiarme en la elaboración de esta tesis.
Gracias por el apoyo.

A mis profesores.
Gracias por todas sus enseñanzas y consejos.
Al Dr. Roman Anselmo Mora Gutiérrez. Le agradezco su apoyo y observaciones.

A mis amigos del posgrado
Gracias por su compañía en cada momento que compartimos durante el posgrado.

RESUMEN DE TESIS

La presente tesis aborda una de las principales problemáticas en la fase inicial del desarrollo de software conocida como ingeniería de requerimientos. Se hace un análisis de las dificultades en las actividades que se pueden realizar en está etapa: como la selección inadecuadamente de ténicas de levantamiento de requerimientos, no contar con una estrategia ni orden y ambigüedades en el entendimiento de las necesidades de los clientes. De estas observaciones se identificó la tarea de socializar el entendimiento de los requerimientos del software al equipo de desarrollo. En este proceso es un factor determiante en la comunicación y estrategia del equipo de desarrollo que impacta en el éxito del software.

En este trabajo se propone una solución empleando artefactos esquemáticos a problemáticas en el levantamiento de requerimientos de software, considerando aspectos relacionados al diseño centrado en el usuario y la experiencia de usuario.

El proceso se apega a lo propuesto por Scrum en el seguimiento de los requerimientos mediante una tabla que lleva registro del avance por cada ciclo de desarrollo (sprint). Como variante propuesta se incorpora el artefacto Kanban etiquetando los estados por columnas.

Al problema del entendimiento de los requerimientos de software y su socialización adecuada al equipo se incorporaron propuestas de variantes de los artefactos: persona, descripción de escenarios, historias de usuario, mapa de experiencia del cliente y mapa de empatía. Estás variantes se han incorporado a la propuesta por el éxito conseguido en diferentes proyectos de software, en los que se puedo verificar el éxito al socializar adecuadamente los requerimientos de software entre los miembros del equipo, así como en la reducción de ambigüedades de los requerimientos recabados.

La evaluación del proceso propuesto se realizó mediante la técnica Delphi con expertos. En los resultados de las rondas de preguntas y sus análisis, los expertos coincidieron que emplear los artefactos incorporados en el trabajo ayudan en el desarrollo de software y permiten una mejor comunicación entre los involucrados en la implementación del producto de software. El cómite de expertos coincidió que emplear el proceso reduciría dificultades en las etapas iniciales del desarrollo y el uso de los artefactos propuestos permiten una mejor comunicación entre los miembros del equipo.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

1

INTRODUCCIÓN

En el presente trabajo de investigación se aborda el proceso de desarrollo de software considerando aspectos de la experiencia de usuario en sus etapas del proceso. En el proceso documental se identificaron situaciones y dificultades que se presentan en la puesta en marcha de un proyecto y durante las actividades del desarrollo de software, algunas de ellas son: la selección adecuada de la metodología, método o framework para su desarrollo; administración y seguimiento del proyecto; análisis y planeación del proceso; ingeniería de requerimientos; desarrollo y pruebas del sistema; implantación, mantenimiento y cierre. Estas etapas junto a sus procesos se pueden encontrar en la literatura bajo la búsqueda de ciclo de vida y desarrollo del software y como administración de proyectos de TIC.

En la literatura se pueden encontrar trabajos que abordan las coincidencias y diferencias entre metodologías tradicionales de desarrollo de software con respecto a los procesos ágiles. En los cuáles se describe el proceso desde la perspectiva de la ideología base del proceso de desarrollo, el paradigma y características involucradas como el enfoque que describe el ciclo de vida del desarrollo o el enfoque que considera a las personas involucradas en el software a implementar. En las perspectivas se encuentra la coincidencia por naturaleza misma del software, que en las diferentes etapas se presentan entradas, se realizan un conjunto de operaciones o actividades y devuelven salidas que a su vez pueden ser entradas de otras etapas. Otro aspecto común abordado en la literatura es el manejo documental en los procesos poniendo enfasís en la reuniones y decisiones tomadas en las tareas de cambios, mantenimiento y control de riesgos en el proyecto.

En los trabajos que abordan la administración de proyectos de TIC y metodologías de desarrollo de software, se sugiere como tarea importante para asegurar el proceso y proyecto de software exitoso la selección de herramientas y técnicas que permitan la comunicación efectiva durante las etapas del procesos de desarrollo del software entre los equipos encargados del área de back-end (considerada la responsable de desarrollar la lógica del negocio), los encargados del área del front-end (considerada

en proporcionar la interfaz gráfica de usuario con la parte lógica del software) y administradores o líderes del proyecto (persona o grupo de personas encargadas de la dirección y avance del proyecto de software).

Se propone una variente del framework Scrum[1] con Kanban[2], añadiendo en las etapas del procesos técnicas, herramientas y buenas prácticas que permitan mejorar la comunicación y entendimiendo de las ideas y objetivos planteados para la implementación del software que de solución a un problema o necesidad de la organicación.

El software es un sistema computacional que permite cumplir con objetivos y solucionar problemas que requieren las personas.

El diseño de la experiencia del usuario (UX) para el software, involucra más elementos que simplemente el diseño de la Interfaz de Usuario (User Interface - UI) y el diseño industrial de los dispositivos o hardware. Son los elementos más visibles de un producto, pero el diseño de productos involucra más elementos para cumplir con la experiencia del usuario.

La experiencia de usuario, es un conjunto de factores relacionados a las interacciones que tiene un usuario al utilizar un producto o servicio. Durante la interacción, estos productos o servicios producen en el usuario una percepción positiva o negativa en los usuarios (Moule, 2012). Este aspecto se refiere a lo que experimenta un usuario (emociones, sensaciones y percepciones) antes, durante y después de interactuar con un producto o servicio. Las percepciones generadas al interactuar con el producto o servicio son el reflejo de las percepciones que tiene hacia la marca.

El diseño de la interacción define las formas en las que el usuario puede operar la interfaz de usuario (UI). Por ejemplo si la interfaz permite la selección de elementos o el ingreso de información se puede realizar mediante el teclado, mouse, touch o combinación de todas. También se refiere a la navegación y flujo de secciones asociadas a las respuestas del sistema.

Como estrategia de metodología de investigación se inicio el proceso recabando fuentes bibliográficas en diferentes motores de búsqueda como: Google Scholar; recursos del consorcio CONRICyT (de las siglas Consorcio Nacional de Recursos de Información Cientifica Y Tecnológica) de CONACYT (de las siglas Consejo Nacional de Ciencia y Tecnología). En el cual se realizaron búsquedas avanzadas sobre los te-

1 Scrum (ver sección 2.2.3.2): Es un proceso de desarrollo dep royectos en el que se aplican un conjunto de buenas prácticas de trabajo colaborativo.

2 Kanban (ver sección 3.4.5): del japonés letrero, se refiere a un sistema de información que controla de modo armónico los procesos de la fabricación de un producto.

mas a abordar en la investigación. Las principales fuentes edioriales empledas fueron: EBSCO, Elsevier, IEEE, Wiley y ACM.

La bibliografía empleada para la investigación esta constituida principalmente de artículos de investigación, capítulos de libros, artículos de congresos, libros especializados en los procesos de desarrollo de software y gestión de proyectos de TIC.

1.1 PLANTEAMIENTO DEL PROBLEMA

La industria de software se enfrentan al reto de implementar tecnologías que ofrezcan servicios que mejoren el quehacer de los usuarios, con una mayor eficiencia en tiempos de operación, recursos y errores en los procesos asociados al desarrollo de software. Los analistas deben dar seguimiento a las problemáticas cotidianas actual que se presentan a los clientes, para determinar la mejor forma de desarrollar un software de calidad que solucione dichas necesidades.

En los procesos de desarrollo de software se ha identificado que se ha descuidado la parte en el que se involucra a los usuarios finales, que son precisamente quienes harán uso de las aplicaciones. Durante el diseño e implementación de los sistemas, se deben considerar las percepciones que el usuario involucrado tiene al interacturar con los sistemas dirigidos. La percepción que tienen los usuarios positiva o negativa sobre el sistema computacional, impacta a todo el sistema incluyendo personal involucrado y a las empresas.

Los usuarios finales tienen una perpectiva desde el dispositivo de escritorio o móvil, los tipos de interacción, la interfaz gráfica de usuario (la visualización y acceso a la información que despliega el sistema). Los dipositivios móviles y sistemas de escritorio son herramientas utilizados en el sistema de involucran a los usuarios finales, grupos de involucrados, patrocinadores, solo por mencionar los principales.

El UX es una disciplina que engloba aspectos como: diseño de la interfaz de usuario, diseños de la interacción, diseño industrial y diseño de la arquitectura de la información. Con el objetivo de mejorar la experiencia del usuario dentro y fuera de los productos o servicios. Actualmente existen recomendaciones en el diseño de interfaces de usuario, en el diseño de la interacción, en el diseño de experiencia de usuario, que permiten proponer estrategias para implementar un producto o servicio que cumpla con la usabilidad, experiencias positivas de usuario y permiten una adecuada manera de desplegar la información a los diferentes usuarios involucrados.

Durante el proceso de administración de una aplicación de software, se requiere llevar acabo una buena integración de las recomendaciones del diseño de experiencia de usuario, con las guías y recomendaciones en el diseño de productos a través de un método dirigido al diseño y desarrollo del sistema. Este método puede derivar en la implementación de productos o servicios que al interactuar con los usuarios involucrados en el seguimiento y mejoramiento de los pacientes.

El proceso debe considerar aspectos de diseño de experiencia de usuario para implementar software usable, funcional y con impacto positivo en los usuarios. Tomar encuenta las investigaciones, recomendaciones y guías de los expertos médicos para que el software realice correctamente las necesidades solicitadas por los médicos. Durante el proceso se deben proporcionar estrategias que faciliten a los ingenieros de software el análisis y comprensión de las necesidades de la parte médica para implementar software que cumpla con los requerimientos del software, validando los avances especificados en cada ciclo de desarrollo. Con el objetivo de generar una percepción positiva del sistema del cuidado médico y que a su vez genere una percepción positiva de las personas involucradas o elementos asociados en los servicios de cuidado de los pacientes.

1.2 OBJETIVOS

Objetivo General:

Establecer un proceso de apoyo conceptual durante el levantamiento de requerimientos durante el desarrollo de software, considerando aspectos del diseño centrado en el usuario, empleando como base Scrum como método de desarrollo de software con enfoque ágil.

Objetivos particulares:

- Identificar aspectos y características del levantamiento de requerimientos asociados al desarrollo de Software.
- Buscar componenetes y recomendaciones del diseño de experiencia de usuario que se apliquen al proceso de levantamiento de requerimientos.
- Identificar los procedimientos que se siguen en las etapas de levantamiento de requerimientos de Software siguiendo un enfoque ágil del desarrollo de software.

- Investigar instrumentos que apoyen en la tarea de verificar el avance y mejora del sistema durante el desarrollo de Software.
- Reducir las ambigüedades o huecos en el entendimiento de los requerimientos de software al transmitirlos en el equipo de trabajo.
- Validación y revisión del proceso propuesto, para estimar los beneficios que puede aportar en el desarrollo de software.

1.3 SUPUESTO

La implementación de software correcto, escalable y de calidad, que cumpla con los protocolos y se acople a la forma de trabajar de los servicios de la organización, se lleva acabo mediante la aplicación de procedimientos que consideran como estrategia la adecuada selección de técnicas de levantamiento de requerimientos relacionadas al contexto.

La implementación de un software que cumpla con la usabilidad y eficiencia por los usuarios finales, se consigue si durante el proceso de desarrollo se siguieron las recomendaciones y guías en el diseño centrado en el usuario, así como el empleo de instrumentos que permitan una mejor comunicación entre los miembros del equipo en la parte operativa de los ciclos de desarrollo apegado a los requerimientos del software.

1.4 APORTACIÓN AL DISEÑO

Con esta investigación se pretende contribuir en al campo de la visualización de la información y el desarrollo de software con enfoque ágil al proponer procesos que apoyen en la implementación de productos de software. En el campo del diseño de interfaces de usuario y de diseño de experiencia de usuario se aporta al considerar aspectos del diseño centrado en el usuario durante el levantamiento de requerimientos y el desarrollo del software.

Con lo cual se pretende establecer ciertos factores que deben ser considerados para la realización de un buen diseño de UX en los productos y servicios de software. Y ser un referente en el proceso de diseño de este campo de la industría de software, dotando de factores involucrados en la interacción con el usuario propiciando experiencias positivas.

1.5 PROCEDIMIENTO METODOLÓGICO

Con base en los objetivos del trabajo según (Arturo and Cruz E. Ma. Cristina, 2006) se trata de una investigación tecnológica, dado que sus características se vinculan en forma natural con la innovación tecnológica y la evaluación de la investigación tecnológica pueden ser utilizadas como un instrumento para fomentar la innovación. Dado que se pretende desarrollar un conocimiento enfocado a mejorar procesos en el levantamiento de requerimientos y fases iniciales del proceso de software. Con lo que se desea potenciar a los diseñadores y desarrolladores en sus diseños y lograr propósitos innovadores.

En cuanto a la obtención de información y conocimiento que se requiere para la investigación (Grajale G, 2000), se emplea un estudio de tipo exploratorio y descriptivo, dado que se pretende abordar temas sobre técnicas de levantamiento de requerimientos, el análisis y especificación de los requerimientos para implementar software adecuado y enfocado en las necesidades planteadas por el problema. Y apoyar al diseño de productos de software al emplear la información recabada en las etapas de análisis.

Para llevar a cabo la investigación y de acuerdo a (Arturo and Cruz E. Ma. Cristina, 2006) se requiere una metodología de investigación que implique un trabajo de cuestionamiento e indagación sistemática y metódica que haciendo uso de conocimiento objetivo previo-ordenado en un cuerpo teórico determinado, tenga como finalidad generar un nuevo conocimiento objetivo que contribuya en el avance de la explicación y transformación. Involucrando teórias y metodologías de diseño en los factores involucrados en la investigación. Se propone que el proceso sea evaluado por expertos en el tema, empleando el método Delphi. Con las respuestas del instrumento de evaluación empleado y la retroalimentación proporcionado por los expertos, se corrige y actualiza el proceso propuesto.

1.6 METAS

Considerando la metodología propuesta, para el desarrollo de esta investigación se contemplan una serie de etapas, que involucran las siguientes acciones o actividades:

1. Documentar los aspectos relacionados al levantamiento de requerimientos de software para tener una mayor claridad del contexto de la ingeniería de requerimientos.

2. Documentar y establecer características del levantamiento de requerimientos asociados al contexto de software ágil.
3. Analizar los componentes del diseño de experiencia de usuario. Y determinar que componentes se pueden abordar en el proceso del levantamiento de requerimientos.
4. Describir los pasos recomendables por Scrum para levantar requerimientos de software y determinar como se pueden aplicar.
5. Establecer una matriz de trazabilidad entre los documentos de requerimientos y los resultados que se obtienen del software, para verificar si el resultado se acerca a los esperado.
6. Aplicar el proceso propuesto a un proyecto de software y documentar los resultados obtenidos. Así como generar una retroalimentación de lecciones aprendidas del proceso.
7. Llevar acabo una revisión y pruebas mediante el método Delphi del proceso propuesto a un grupo de expertos en métodos de desarrollo de software, para verificar la propuesta de proceso. Realizar las correcciones al proceso que pueden surgir y documentar los resultados en la tesis.

1.7 ORGANIZACIÓN DE LA TESIS

La presente tesis está estructurada en siete capítulos, como se puede apreciar en la Figura 1. En el actual capítulo se introduce a los temas que aborda el presente trabajo, los objetivos, supuesto, aportación al diseño dada la LGAC del posgrado de la institución educativa, proceso metodológico a seguir en el trabajo y las métas de la investigación. El capítulo 2 comprende los fundamentos y estado del arte de la investigación, donde se describen los conceptos relacionados con paradigmas principales de la Ingeniería de Software, metodologías de desarrollo de software, ingeniería de requerimientos y el levantamiento de requerimientos, la especificación de requerimientos y el documento que formaliza los requerimientos de un proyecto y las dificultades y retos de desarrollar software. En el capítulo 3 se describe la propuesta del proceso del trabajo de investigación, el contexto de la situación del levantamiento de requerimientos de sotfware, los pasos del proceso de selección de requerimientos propuesta por la investigación, la integración del proceso con Scrum y los artefactos de dirección de proyectos recomendados por el actual trabajo. En el capítulo4 se aborda la técnica Delphi

como opción para validar el proceso propuesto, se consideran las límitantes y beneficios que aporta la técnica, así como los resultados de las dos rondas de preguntas y respuestas realizadas con los expertos. En el capítulo 5 se presentan las conclusiones describiendo como se fueron cumpliendo los objetivos de la tesis y la comprobanción de la hipótesis.

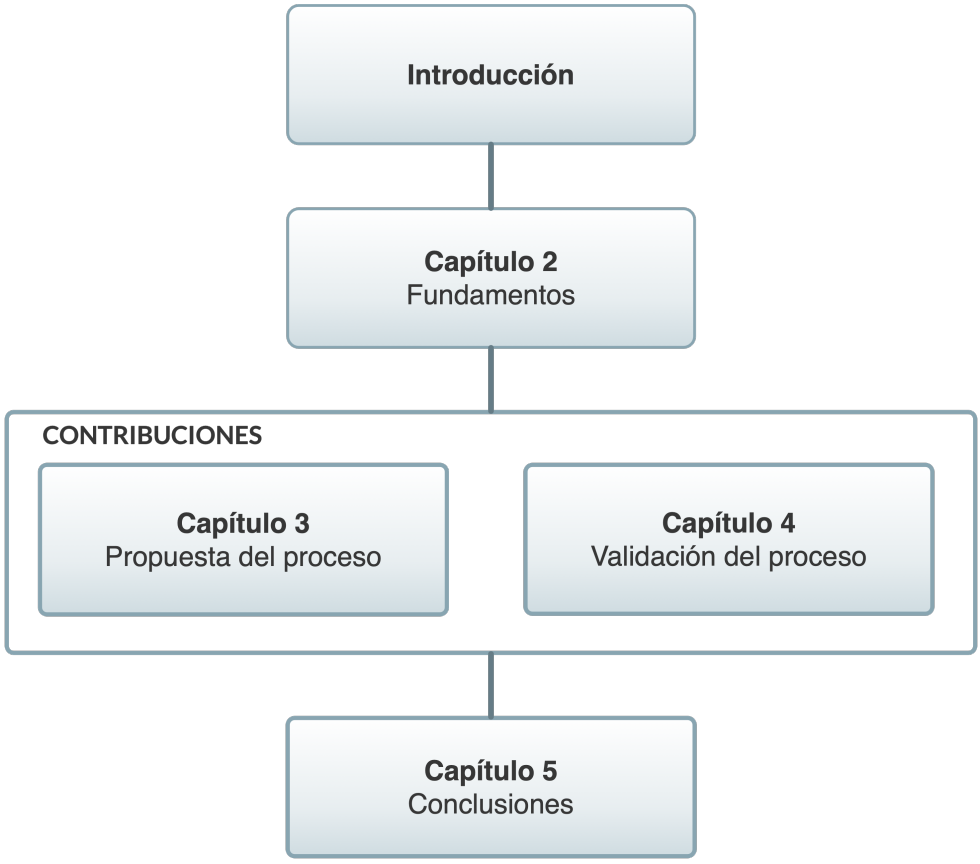


Figura 1: *Organización de la tesis.*

2

FUNDAMENTOS

En este capítulo se describen algunos paradigmas de la Ingeniería de Software, ventajas y desventajas de su utilización, algunas características de los elementos involucrados y ejemplos de proyectos que los implementan. En la sección 2.2 se describen las principales metodologías de desarrollo de software abordando las metodologías tradicionales y los métodos ágiles en el desarrollo de software. El capítulo cierra con un resumen de las ventajas y desventajas de la metodologías respecto al proceso de levantamiento de requerimientos de software.

2.1 PARADIGMAS PRINCIPALES DE LA INGENIERÍA DE SOFTWARE

Las definiciones de la Ingeniería de Software consideran diversos aspectos de lo que contempla esta disciplina (Oktaba, 2010):

ENFOQUE EN EL CICLO DE VIDA DE SOFTWARE: Es la aplicación sistemática, disciplinada y cuantificable del desarrollo, operación y mantenimiento de software (Z. Zakaria and Sani, 1990).

ENFOQUE EN LAS PERSONAS INVOLUCRADAS: Es la construcción de productos de software por grupos de personas, para que sean usados por otras. El cliente es quien solicita el desarrollo del producto y plantea el problema a resolver. El equipo de desarrollo construye y entrega el producto solicitado (Oktaba, 2010).

La Ingeniería de Software utiliza las teorías y características de las computadoras, objetos de estudio de las Ciencias de la Computación, para resolver problemas planteados por un cliente. Es un campo de estudio que genera prácticas y herramientas para resolver estos problemas. Es un área que está evolucionando (que inició en 1968) para ser una disciplina completa con principios y teorías propias (Oktaba, 2010).

Un paradigma es un modelo para comprender la realidad, que nos permite relacionarnos con el mundo que nos rodea y tener un sentido de identidad dentro de lo que percibimos como "mundo real".

Para la ingeniería de Software un paradigma es una agrupación de métodos, herramientas y procedimientos con el fin de describir un modelo.

La ingeniería del software establece y se vale de una serie de modelos que muestran las distintas etapas y estados por los que pasa un producto software, desde su concepción inicial, pasando por su desarrollo, puesta en marcha y posterior mantenimiento, hasta la retirada del producto. A estos modelos se les denomina “Modelos de ciclo de vida del software” (Pressman, 2002).

Para resolver los problemas reales de una industria, un ingeniero de software o un equipo de ingenieros deben incorporar una estrategia de desarrollo que acompañe al proceso, métodos, capas de herramientas, según la naturaleza del proyecto y de la aplicación.

2.1.1 *Modelo Lineal Secuencial o de Cascada*

El primer modelo concebido fue el de Royce (1970), comúnmente conocido como Cascada o “Lineal Secuencial”. Este modelo establece que las diversas actividades que se van realizando al desarrollar un producto software, se suceden de forma lineal (Pressman, 2002).

El modelo sugiere un enfoque sistemático secuencial, para el desarrollo del software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento. Además es requerida la documentación detallada en cada etapa, junto a los comentarios, los documentos archivados, las aprobaciones finales en cada fase del proceso y la administración de la configuración de todo el proyecto (Dooley, 2011).

La Figura 2 muestra el modelo lineal secuencial para la Ingeniería de Software.

El modelo lineal secuencial comprende las siguientes actividades:

- Ingeniería y modelado de Sistemas.

  Como el software forma parte de un sistema más grande, el trabajo comienza estableciendo requerimientos de todos los elementos del sistema y asignando al software algún subgrupo de estos requerimientos. Esta visión del sistema es

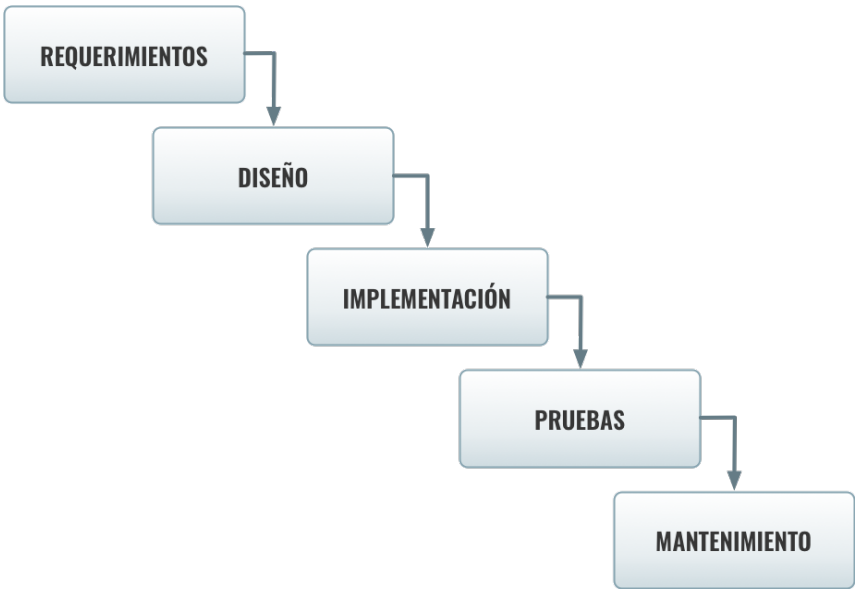


Figura 2: *Modelo secuencial o de Cascada [Imagen elaborada por el autor].*

esencial cuando el software se debe interconectar con otros elementos como hardware, personas y bases de datos. La ingeniería y el análisis de sistemas comprende los requerimientos que se recogen en el nivel del sistema con una pequeña parte de análisis y de diseño.

- Análisis de los requerimientos del software.

  El proceso de reunión de requerimientos se intensifica y se centra especialmente en el software. Para comprender la naturaleza de los programas a desarrollarse, el ingeniero del software debe comprender el dominio de información del software, así como la función requerida, comportamiento, rendimiento e interconexión.

- Diseño.

  El diseño del software es un proceso de muchos pasos que se centra en cuatro atributos distintos de programa: estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental. El proceso del diseño traduce requerimientos en una representación del software donde se pueda evaluar su calidad antes de que comience la codificación.

- Generación de código.

  El diseño se debe traducir en una forma legible por la máquina. El paso de generación de código lleva a cabo esta tarea. Si se lleva a cabo el diseño de una forma detallada, la generación de código se realiza mecánicamente.

- Pruebas.

  Una vez que se ha generado el código, comienzan las pruebas del programa. El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales; es decir, realizar las pruebas para la detección de errores y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos.

- Mantenimiento.

  El software indudablemente sufrirá cambios después de ser entregado al cliente (una excepción es el software empotrado). Se producirán cambios porque se han encontrado errores, porque el software debe adaptarse para acoplarse a los cambios de su entorno externo o porque el cliente requiere mejoras funcionales o de rendimiento. **El soporte y mantenimiento del software vuelve a aplicar cada una de las fases precedentes a un programa ya existente y no a uno nuevo** (Pressman, 2002).

El modelo lineal secuencial es el paradigma más antiguo y más extensamente utilizado en la Ingeniería de Software. A continuación se presentan algunas ventajas y desventajas de este paradigma (Dooley, 2011)(Pressman, 2002)(Weitzenfeld, 2002).

Ventajas del modelo de Cascada

1. El modelo en cascada puede ser apropiado, para proyectos estables (especialmente los proyectos con requerimientos no cambiantes) y donde es posible y probable que los diseñadores predigan totalmente áreas del problema del sistema y produzcan un diseño correcto antes de iniciar la implementación. Funciona bien para proyectos pequeños donde los requerimientos están bien entendidos.
2. Es un modelo en el que todo está bien organizado y no se mezclan las fases. Es simple y fácil de usar.
3. Debido a la rigidez del modelo es fácil de gestionar ya que cada fase tiene entregables específicos y un proceso de revisión. Las fases son procesadas y completadas de una vez.
4. También es un buen modelo para los equipos sin experiencia de trabajo en un nuevo proyecto bien definido, ya que les lleva a través del ciclo de vida.

Desventajas del modelo de Cascada

1. En la vida real, un proyecto rara vez sigue una secuencia lineal, esto produce una mala implementación del modelo, lo que puede llevar al fracaso.
2. Difícilmente un cliente va a establecer al principio todos los requerimientos necesarios, por lo que provoca un gran atraso trabajando en este modelo, ya que este es muy restrictivo y no permite saltar entre fases.
3. Principalmente causa la creencia de que es imposible, para un proyecto no trivial, conseguir tener una fase del ciclo de vida del producto software perfecta antes de moverse a las siguientes fases. Por ejemplo, los clientes pueden no ser conscientes exactamente de los requerimientos que quieren antes de ver un prototipo del trabajo; pueden cambiar los requerimientos constantemente, y los diseñadores e implementadores pueden tener poco control sobre esto. Si los clientes cambian sus requerimientos después de que el diseño está terminado, este diseño deberá ser modificado para acomodarse a los nuevos, invalidando una buena parte del esfuerzo.
4. Muchas veces se considera un modelo pobre para proyectos complejos, largos, orientados a objetos y por supuesto en aquellos en los que los requerimientos tengan un riesgo de moderado a alto de cambiar. Genera altas cantidades de riesgos e incertidumbres.

2.1.1.1 *Cascada con Subproyectos*

Se entiende como una variación sobre el ciclo de vida en Cascada del software, denominada Cascada con Subproyectos (ver Figura 3), porque permite la ejecución de algunas de las tareas de la cascada en paralelo.

Un ejemplo de aplicación de esta metodología es en el desarrollo de un sistema de información para una empresa, en donde deben estar involucradas todas las áreas dondei están compartiendo información. Para ello se pueden ir adelantando de forma paralela las etapas del ciclo de vida como análisis, diseño, desarrollo y pruebas de forma independiente para cada área de la organización y hacia el final se realiza la integración de los resultados de cada subproyecto. Esta metodología tiene el problema que la planificación se tiene que realizar con mayor atención y cuidado, aunque se gana velocidad en el desarrollo.

La implementación del software empleando la metodología de cascada con subproyectos permite iniciar la implementación de áreas menos difíciles y reducir los retrasos

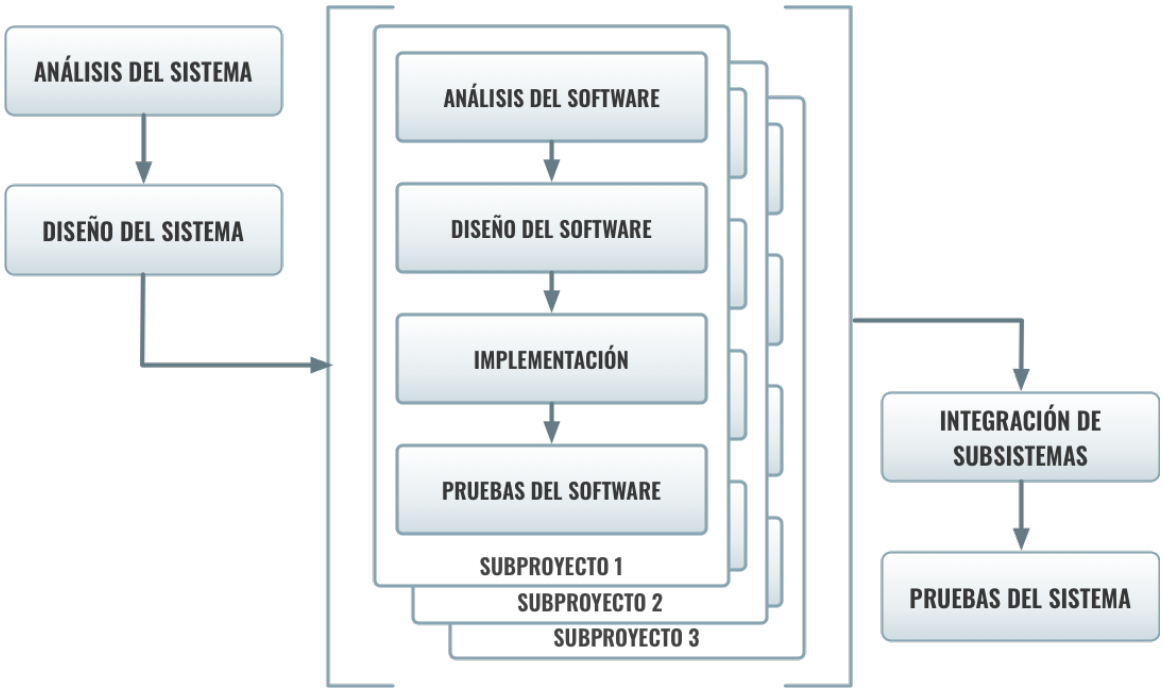


Figura 3: *Cascada con subproyectos [Imagen elaborada por el autor].*

de desarrollo. Al dividir la arquitectura del problema en subsistemas independientes, se puede separar en subproyectos y cada uno puede proceder a su implementación.

2.1.2 *Modelo en Espiral*

El modelo de espiral es una modificación al modelo de cascada desarrollado por Barry Boehm en 1985. Se basa en una estrategia para reducir riesgos, al contrario del modelo de cascada que es dirigido por documentos. Como parte del manejo de riesgo el modelo incorpora una estrategia de uso de prototipos. Enfatiza ciclos de trabajo, cada uno de los cuales estudia el riesgo antes de proceder al siguiente ciclo. Cada ciclo comienza con la identificación de los objetivos para una parte del producto, formas alternativas de lograr los objetivos, restricciones asociadas con cada alternativa, y finalmente procediendo a una evaluación de las alternativas. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado (Pressman, 2002)(Weitzenfeld, 2002).

La Figura 4 muestra un diagrama conceptual del modelo de cascada describiendo los distintos ciclos del espiral.

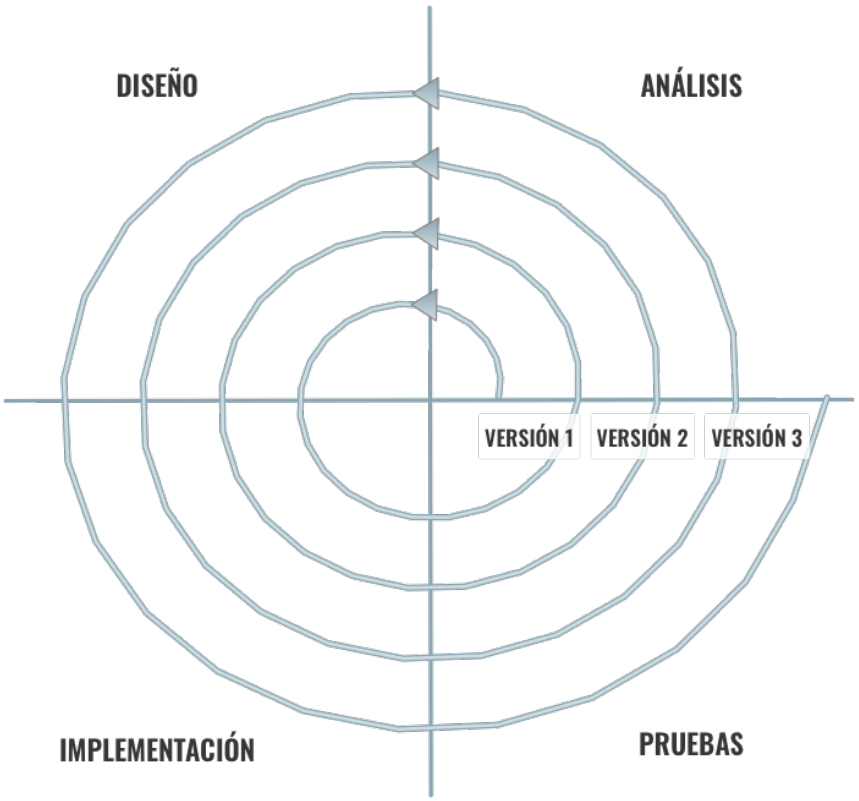


Figura 4: *Modelo en Espiral [Imagen elaborada por el autor].*

Cuando se identifica incertidumbre, se utilizan diversas técnicas para reducir el riesgo en escoger entre las diferentes alternativas. Cada ciclo del modelo termina con una revisión que discute los logros actuales y los planes para el siguiente ciclo, con el propósito de lograr la incorporación de todos los miembros del grupo para su continuación. La revisión puede determinar si desarrollos posteriores no van a satisfacer las metas definidas y los objetivos del proyecto. En tal caso, se termina el espiral.

El modelo se divide en un número de actividades de trabajo, también llamadas regiones de tareas. Generalmente, existen entre tres y seis regiones de tareas. La Figura 5 representa un modelo en espiral que contiene seis regiones de tareas (Pressman, 2002):

- **Comunicación con el cliente**.- las tareas requeridas para establecer comunicación entre el desarrollador y el cliente.
- **Planificación**.- las tareas requeridas para definir recursos, el tiempo y otra información relacionadas con el proyecto.

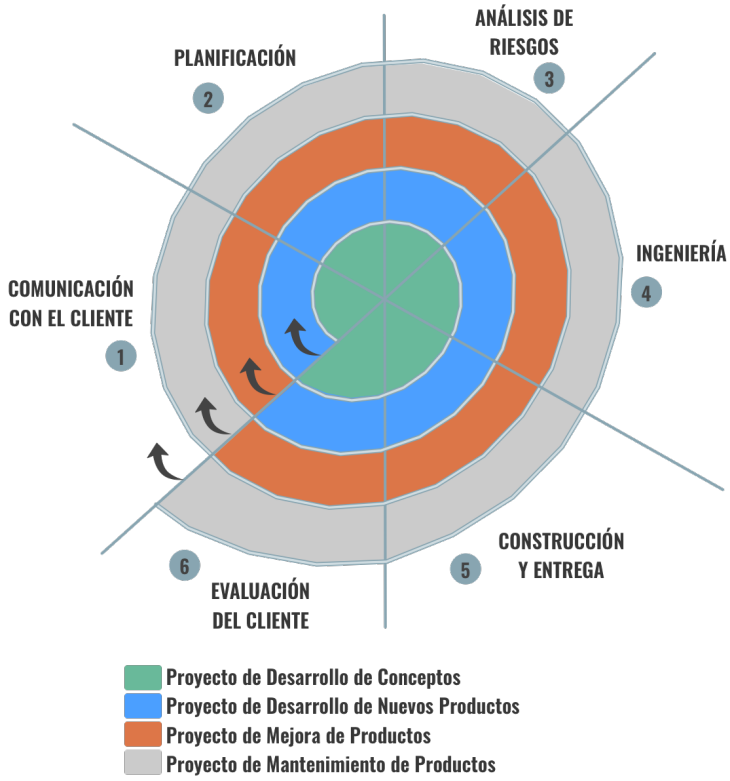


Figura 5: *Modelo en espiral de 6 tareas [Imagen elaborada por el autor].*

- **Análisis de riesgos**.- las tareas requeridas para evaluar riesgos técnicos y de gestión.
- **Ingeniería**.- las tareas requeridas para construir una o más representaciones de la aplicación.
- **Construcción y acción**.- las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario (por ejemplo: documentación y práctica).
- **Evaluación del cliente**.- las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación.

Cada una de las regiones están compuestas por un conjunto de tareas del trabajo, que se adaptan a las características del proyecto a desarrollar. Para proyectos pequeños, el número de tareas de trabajo y su formalidad es bajo. Para proyectos mayores y más críticos cada región de tareas contiene tareas de trabajo que se definen para lograr un nivel más alto de formalidad (Pressman, 2002).

El modelo en espiral es un enfoque realista del desarrollo de sistemas y de software a gran escala. Como el software evoluciona, a medida que progresa el proceso el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos. El modelo permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto. Mantiene el enfoque sistemático de los pasos sugeridos por el ciclo de vida clásico, pero lo incorpora el trabajo iterativo que refleja de forma más realista el proceso real de desarrollo de software.

A continuación se presentan algunas ventajas y desventajas de este paradigma (Dooley, 2011)(Weitzenfeld, 2002).

Ventajas del modelo en espiral

1. El análisis de riesgos se hace de forma explícita y clara.
    a) Reduce riesgos del proyecto.
    b) Incorpora objetivos de calidad.
    c) Integra el desarrollo con el mantenimiento
2. Mediante este modelo se produce software en etapas tempranas del ciclo de vida y suele ser adecuado para proyectos largos de misión crítica
3. Es posible tener en cuenta mejoras y nuevos requerimientos sin romper con el modelo, ya que el ciclo de vida no es rígido ni estático.

Desventajas del modelo en espiral

1. Puede resultar difícil convencer a algunos clientes de que el enfoque evolutivo es controlable.
2. Es un modelo que genera mucho trabajo adicional. Al ser el análisis de riesgos una de las tareas principales exige un alto nivel de experiencia y cierta habilidad en los analistas de riesgos.
3. Puede que sea un modelo costoso. Además, no es un modelo que funcione bien para proyectos pequeños.

Característica importente del modelo en espiral

La diferencia principal entre el modelo espiral y otros modelos es la evaluación del riesgo. Abordamos el riesgo como todo lo que pueda salir mal en un proyecto de desarrollo de software. Por ejemplo, si queremos utilizar un lenguaje de programación para desarrollar una aplicación, un riesgo es que no todos los desarrolladores del equipo

trabajen con la misma versión de bibliotecas y que al integrar el código no sean compatibles. Los riesgos originan problemas en el proyecto, como el exceso de los costos. Es así que, la disminución de los riesgos es una actividad muy importante, para alcanzar un software de calidad.

2.1.3 *Modelo de Prototipos*

Un cliente, puede definir un conjunto de objetivos generales para el software que solicita, pero no identifica los requerimientos detallados de entrada, del proceso o salida. En otros casos el desarrollador del software puede no estar seguro de la eficiencia de un algoritmo, de la calidad de adaptación de un sistema operativo, o de la forma en que debería tomarse la interacción hombre-máquina. Por estas y otras situaciones el paradigma de construcción de prototipos puede ofrecer el mejor enfoque (Pressman, 2002).

El paradigma comienza con la recolección de requerimientos (ver Figura 6). El desarrollador y el cliente definen los objetivos globales para el software, identifican los requerimientos conocidos y las áreas del esquema en donde es obligatorio más definición, por lo que se desarrolla un diseño rápido. El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario-cliente.

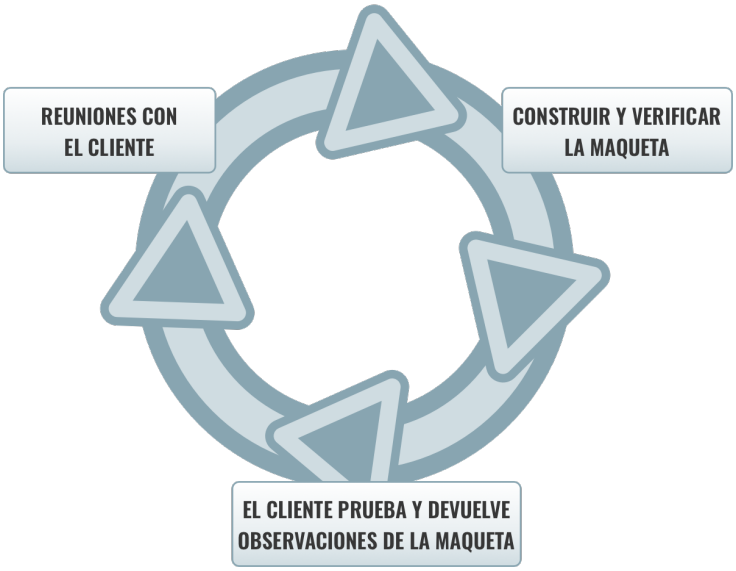


Figura 6: *Modelo de prototipo [Imagen elaborada por el autor].*

El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el cliente-usuario y se utiliza para refinar los requerimientos del software a desarrollar. La

iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

La idea es que el prototipo sirva como un mecanismo para identificar los requerimientos del software. Si se construye un prototipo de trabajo, el desarrollador intenta hacer uso de los fragmentos del programa ya existentes o aplica herramientas que permiten generar rápidamente programas de trabajo (Pressman, 2002). Existe la idea errónea de que gracias al paradigma orientada a objetos los prototipos pueden evolucionar iterativamente hasta llegar al sistema final. Esto realmente varía y depende del tipo de prototipo que se esté desarrollando (Weitzenfeld, 2002):

- **Prototipos de requerimientos**.- el prototipo de requerimientos permite que los usuarios interactúen con una interfaz de usuario del sistema que muestre toda o casi toda la funcionalidad requerida del producto final sin que realmente se implemente. Su objetivo es clarificar los requerimientos y solicitar nuevas ideas, el prototipo puede incluirse como parte de la documentación.
- **Prototipos de análisis**.- el prototipo permite presentar una arquitectura general del sistema de manera rápida que considere las características principales del sistema, como los componentes a ser utilizados.
- **Prototipos de diseño**.- se desarrolla para explorar y comprender la arquitectura particular del sistema. Servir como base para la evaluación de rendimiento y espacio, y para pruebas de redundancia o inconsistencias en el diseño. La evaluación del rendimiento de un prototipo de diseño puede identificar cuellos de botella en el sistema.
- **Prototipos verticales**.- se utiliza para comprender una sección o parte de un problema y su solución completa. Esto se hace cuando los conceptos básicos no están bien comprendidos.
- **Prototipos de factibilidad**.- se utiliza para demostrar si un aspecto del proyecto es posible. Por ejemplo, si una arquitectura particular es aplicable; si la manera de conectarse a una base de datos ofrece un rendimiento adecuado; si es posible que un grupo de desarrollo aprenda a programar en cierto lenguaje, entre otros.

A continuación se presentan algunas ventajas y desventajas de este paradigma (Weitzenfeld, 2002).

Ventajas del modelo de prototipos

1. Ofrece visibilidad del producto desde el inicio del ciclo de vida con el primer prototipo. Esto puede ayudar al cliente a definir mejor los requerimientos y a ver las necesidades reales del producto. Permite introducir cambios en las iteraciones siguientes del ciclo.
2. El prototipo es un documento vivo de buen funcionamiento del producto final. El cliente reacciona mucho mejor ante el prototipo, sobre el que puede experimentar, que no sobre una especificación escrita.
3. Este modelo reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios.

Desventajas del modelo prototipos

1. Entre los inconvenientes que se han observado con este modelo está el hecho de que puede ser un desarrollo lento.
2. Se hacen fuertes inversiones en un producto desechable ya que los prototipos se descartan. Esto puede hacer que aumente el coste de desarrollo del producto.
3. Con este modelo pueden surgir problemas con el cliente que ve funcionando versiones del prototipo pero puede desilusionarse porque el producto final aún no ha sido construido.
4. El desarrollador puede caer en la tentación de ampliar el prototipo para construir el sistema final sin tener en cuenta los compromisos de calidad y de mantenimiento que tiene con el cliente.

2.1.4 *Modelo DRA*

El Desarrollo Rápido de Aplicaciones (**Rapid Application Development RAD**) es una adaptación a alta velocidad del modelo lineal secuencial en el que se logra el desarrollo rápido utilizando una construcción basada en componentes. Si se comprenden bien los requerimientos y se limita el ámbito del proyecto, el proceso DRA permite a los desarrolladores crear un sistema completamente funcional dentro de períodos cortos de tiempo (por ejemplo de rango: de 60 a 90 días)(Aparicio, 2012)(Pressman, 2002).

El modelo DRA (ver Figura 7) comprende las siguientes fase:

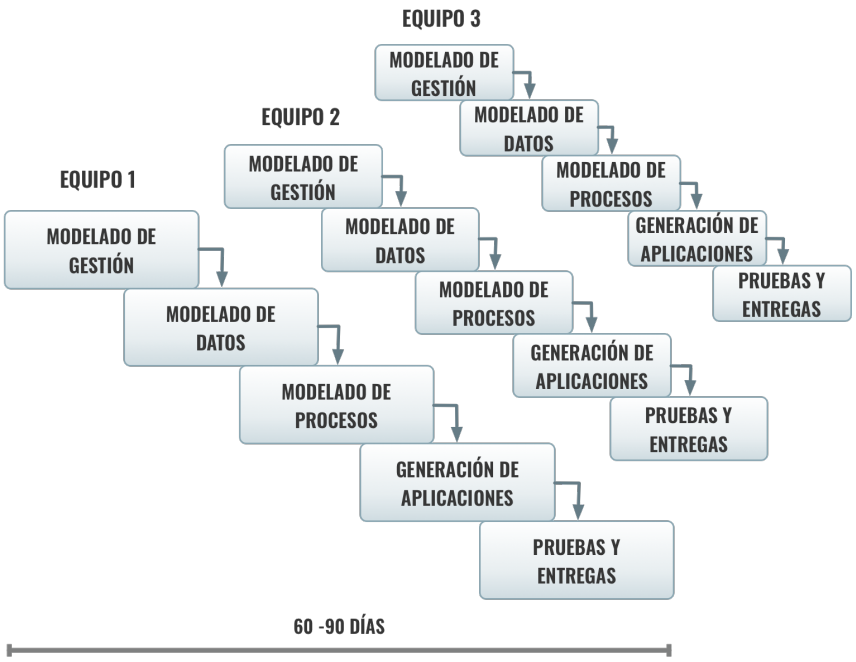


Figura 7: *Fases del modelo DRA [Imagen elaborada por el autor].*

1. **Modelado de Gestión**.- el flujo de información entre las funciones de gestión se modela de forma que responda a las siguientes preguntas: ¿Qué información conduce el proceso de gestión? ¿Qué información se genera? ¿Quién la genera? ¿A dónde va la información? ¿Quién la procesa?.
2. **Modelado de datos**.- el flujo de información definido como parte de la fase de modelado de gestión se refina como un conjunto de objetos de datos necesarios para apoyar la empresa. Se definen las características (llamadas atributos) de cada uno de los objetos y las relaciones entre estos objetos.
3. **Modelado del proceso**.- los objetos de datos definidos en la fase de modelado de datos quedan transformados para lograr el flujo de información necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir, o recuperar un objeto de datos.
4. **Generación de aplicaciones**.- el DRA asume la utilización de técnicas de cuarta generación. En lugar de crear software con lenguajes de programación de tercera generación, el proceso DRA trabaja para volver a utilizar componentes de programas ya existentes (siempre que sea posible) o a crear componentes reutilizables (cada vez que sea necesario). En todos los casos se utilizan herramientas para facilitar la construcción del software.

5. **Pruebas y entrega**.- como el proceso DRA enfatiza la reutilización, un conjunto de componenetes de la aplicación se han probado, esto reduce el tiempo de pruebas (no siempre ocurre esto). A pesar de esto se deben probar todos los componentes nuevos y todas las interfaces a fondo.

El modelo DRA requiere el uso interactivo de técnicas estructuradas y prototipos para definir los requerimientos de usuario y diseñar el sistema final. Usando técnicas estructuradas, el desarrollador primero construye modelos de datos y modelos de procesos de negocio preliminares de los requerimientos.

Los prototipos ayudan al analista y usuarios a verificar tales requerimientos y a refinar formalmente los modelos de datos y procesos. El ciclo de modelos resulta a la larga en una combinación de requerimientos de negocio y una declaración de diseño técnico para ser usado en la construcción de nuevos sistemas.

Los enfoques DRA pueden implicar compromisos en funcionalidad y rendimiento a cambio de permitir el desarrollo más rápido y facilitando el mantenimiento de la aplicación.

A continuación se presentan algunas ventajas y desventajas de este paradigma (Aparicio, 2012).

Ventajas del modelo DRA

1. Velocidad de desarrollo.
2. Aumenta la calidad con la implicación del usuario en las etapas del análisis y del diseño.
3. Visibilidad temprana debido al uso de técnicas de prototipado.
4. Mayor flexibilidad que otros modelos.
5. Ciclos de desarrollo más cortos.

Desventajas del modelo DRA

1. Para proyectos grandes aunque por escalas, el requiere recursos humanos suficientes para crear el número correcto de equipos DRA.
2. Requiere de clientes y desarrolladores comprometidos en las actividades necesarias para completar un sistema en un tiempo abreviado. Si no hay compromiso por ninguna de las partes constituyentes, los proyectos DRA fracasan.

3. No todos los tipos de aplicaciones son apropiados para DRA. Si un sistema no se puede modularizar adecuadamente, la construcción de los componentes necesarios para DRA será problemático.
4. DRA no es adecuado cuando los riesgos técnicos son altos. Esto ocurre cuando una nueva aplicación hace uso de tecnologías nuevas, o cuando el software nuevo requiere un alto grado de interoperatividad con programas ya existentes.

2.1.5 *Modelo de Métodos Formales*

El modelo de métodos formales comprende un conjunto de actividades que conducen a la especificación matemática del software de computadora. Los métodos formales permiten que un ingeniero de software especifique, desarrolle y verifique un sistema basado en computadora aplicando una notación rigurosa y matemática. Algunas organizaciones de desarrollo del software actualmente aplican una variación de este enfoque, llamado ingeniería del software de sala limpia (cleanroom) (Aparicio, 2012)(Pressman, 2002).

Cuando se utilizan métodos formales durante el desarrollo de software, se proporciona un mecanismo para eliminar muchos de los problemas que son difíciles de superar con paradigmas de la ingeniería del software, por ejemplo: la ambigüedad, lo incompleto y la inconsistencia se descubren y se corrigen con más fácilidad, mediante la aplicación del análisis matemático. Cuando se utilizan métodos formales durante el diseño, sirven como base para la verificación de programas y permiten que los ingenieros del software descubran y corrijan errores que no se pudieron detectar de otra manera.

Desvenjatas de lo métodos formales

Los métodos formales ofrecen la promesa de un software libre de defectos, sin embargo se ha hablado de una gran preocupación sobre su aplicabilidad en un entorno de gestión:

1. El desarrollo de modelos formales actualmente es bastante caro y lleva mucho tiempo.
2. Pocos responsables del desarrollo de software tienen los antecedentes necesarios para aplicar métodos formales.
3. Es difícil utilizar los modelos como un mecanismo de comunicación con clientes que no tienen muchos conocimientos técnicos.

No obstante es posible que esté enfoque tenga más partidarios entre los desarrolladores del software que deben construir software de mucha seguridad (por ejemplo: los desarrolladores de aviónica y dispositivos médicos).

2.1.6 *Técnicas de Cuarta Generación*

Las técnicas de cuarta generación (T4G) abarcan un amplio espectro de herramientas de software que tienen algo en común: todas facilitan al ingeniero del software la especificación de algunas características del software a alto nivel.

El paradigma T4G para la ingeniería del software se orienta hacia la posibilidad de especificar el software usando formas de lenguaje especializado o notaciones gráficas que describa el problema que hay que resolver en términos que los entienda el cliente. Actualmente, un entorno para el desarrollo de software que soporte el paradigma T4G puede incluir todas o algunas de las siguientes herramientas: lenguajes no procedimentales de consulta a bases de datos, generación de informes, manejo de datos, interacción y definición de pantallas, generación de códigos, capacidades gráficas de alto nivel y capacidades de hoja de cálculo, y generación automatizada de HTML y lenguajes similares utilizados para la creación de sitios web usando herramientas de software avanzado. Inicialmente, muchas de estas herramientas estaban disponibles, pero sólo para ámbitos de aplicación muy específicos, pero actualmente los entornos T4G se han extendido a todas las categorías de aplicaciones del software, ver Figura 8 (Aparicio, 2012)(Pressman, 2002).

Para transformar una implementación T4G en un producto, los desarrolladores deben dirigir una prueba completa, desarrollar con sentido una documentación y ejecutar el resto de las actividades de integración que son también requeridas por otros paradigmas de ingeniería del software. Además, el software desarrollado con T4G debe ser construido de forma que facilite su mantenimiento de forma expeditiva.

Al igual que otros paradigmas del software, el modelo T4G tiene ventajas e inconvenientes. A continuación se presentan algunas ventajas y desventajas de este paradigma (Aparicio, 2012)(Pressman, 2002).

Ventajas del modelo T4G

1. Las compañías que usan T4G parecen indicar que el tiempo requerido para producir software se reduce mucho para aplicaciones pequeñas y medianas, y que la cantidad de análisis y diseño se reduce.

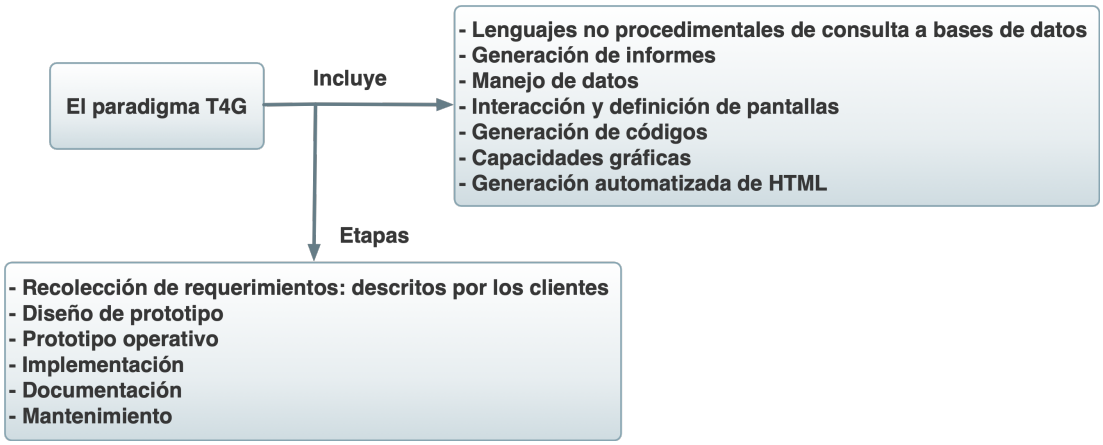


Figura 8: *Paradigma T4G [Imagen elaborada por el autor].*

2. Los defensores aducen reducciones drásticas en el tiempo de desarrollo del software y una mejora significativa en la productividad de la gente que construye el software.

Desvenjatas del modelo T4G

1. El uso de T4G para grandes trabajos de desarrollo de software exige el mismo o más tiempo de análisis, diseño y prueba (actividades de ingeniería del software), perdiéndose así un tiempo sustancial que se ahorra mediante la eliminación de la codificación.
2. Las herramientas actuales de T4G no son más fáciles de utilizar que los lenguajes de programación.
3. El código fuente producido por las herramientas es ineficiente y su mantenimiento es complicado en grandes sistemas de software.

2.1.6.1 *Resumen de las ventajas y desventajas de los paradigmas respecto al proceso de levantamiento de requerimientos*

En la Tabla 1 se muestra un resumen de las ventajas y desventajas que se presentan en el proceso de levantamiento de requerimientos de los paradigmas abordados:

Tabla 1: *Ventajas y desventajas de los paradigmas de dasarrollo de software en el levantamiento de requerimientos*

-	Ventajas	Desventajas
Modelo Lineal Secuencial o de Cascada	- Apropiado para proyectos estables con requerimientos no cambiantes y donde los diseñadores produzcan un diseño correcto antes de iniciar la implementación. - Funciona bien para proyectos pequeños donde los requerimientos están bien entendidos. - La fase de levantamiento de requerimientos tiene como entregable el documento de especificación de requerimientos. - Apto para equipos con poca experiencia de trabajo en un nuevo proyecto con las etapas bien definidas.	- En la vida real un proyecto no sigue una secuencia lineal y al no permitir regresar a fases anteriores se omiten requerimientos. - Difícilmente un cliente establece desde el principio todos los requerimientos necesarios. - Si los clientes cambian sus requerimientos después de que el diseño está terminado, se deben acomodar los nuevos requerimientos invalidando el esfuerzo realizado. - Es un modelo poco apto para proyectos con requerimientos complejos y con alta probabilidad de cambio.
Modelo en Espiral	- El proceso de requerimientos se realiza en cada nueva iteración. - Se puede verificar la calidad del software dado que en cada ciclo se da seguimiento a los requerimientos establecidos al inicio del ciclo de desarrollo actual. - Se realiza un seguimiento de los requerimientos y de los posibles riesgos del proyecto.	- Resulta difícil convencer a algunos clientes de que el enfoque evolutivo es controlable y que los requerimientos se están cumpliendo en cada ciclo de desarrollo de software. - Es un modelo que genera mucho trabajo adicional, desde el análisis de requerimientos y la planeación exige un alto nivel de experiencia y habilidades de los analistas.

Tabla 1: *Ventajas y desventajas de los paradigmas de dasarrollo de software en el levantamiento de requerimientos*

-	Ventajas	Desventajas
Modelo en Espiral	- Se producen versiones tempranas de software que cumplen con los requerimeintos de software dado que se ajusta la carga de trabajo por ciclo de desarrollo. - Es posible tener en cuenta mejoras y nuevos requerimientos sin romper con el modelo, ya que el ciclo de vida no es rígido ni estático.	- No es adecuado para proyectos pequeños por los tiempos, entregables y procesos que plantea el modelo, resulta engorroso para proyectos que no requieren de tantos procesos y entregables.
Modelo de Prototipos	- Ayuda al cliente a definir mejor los requerimientos y ver las necesidades reales del producto. - Permite introducir cambios en las iteraciones siguientes del ciclo y verificar el avance del cumplimiento de los requerimientos. - El cliente reacciona mucho mejor ante el prototipo, sobre el que puede experimentar. - Este modelo reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios.	- Se hacen fuertes inversiones en un producto desechable ya que los prototipos se descartan. - Su empleo puede hacer que aumente el coste de desarrollo del producto. - Pueden surgir problemas con el cliente que ve funcionando versiones del prototipo pero puede desilusionarse del producto final, al no cubrir completamente con los requerimientos. - El desarrollador puede caer en la tentación de ampliar el prototipo para construir el sistema final sin tener en cuenta la calidad y cumplimiento de los requerimientos.

Tabla 1: *Ventajas y desventajas de los paradigmas de dasarrollo de software en el levantamiento de requerimientos*

-	Ventajas	Desventajas
Modelo DRA	- Aumenta la calidad con la implicación del usuario en las etapas del levantamiento de requerimientos, análisis y del diseño. - Visibilidad temprana del avance del cumplimiento de los requerimientos debido al uso de técnicas de prototipado. - Amplio grado de flexibilidad ante el contexto y cambios de requerimientos. - Ciclos de desarrollo cortos, por lo que en cada ciclo se abordan grupos de requerimientos alcanzables.	- Requiere de muchos recursos humanos, para crear el número correcto de equipos DRA. - Requiere de clientes y desarrolladores comprometidos en las actividades necesarias para completar un sistema en un corto tiempo. - La falta de compromiso de las partes interesadas producen fracasos en los proyectos. - No es recomendable cuando la implementación del desarrollo se debe realizar con tecnologías nuevas. Por lo que en la etapa de requerimientos es importante identificar esos aspectos. - No es recomendable cuando el software nuevo requiere un alto grado de interoperatividad con programas existentes.

Tabla 1: *Ventajas y desventajas de los paradigmas de dasarrollo de software en el levantamiento de requerimientos*

-	Ventajas	Desventajas
Métodos Formales	- Permiten que un ingeniero de software especifique, desarrolle y verifique un sistema aplicando una notación rigurosa y matemática. Desde las primesas fases incluyendo recabar los requerimientos. - Reduce la ambigüedad que se puede presentar en los requerimientos y análisis del problema, así como lo incompleto y la inconsistencia de los sistemas. - Durante la fase de diseño verifica que los programadores del software descubran y corrijan errores.	- El desarrollo de modelos formales actualmente es costoso y requiere mucho tiempo. - Pocos responsables del desarrollo de software tienen los antecedentes necesarios para aplicar métodos formales. - Es difícil utilizar los modelos como un mecanismo de comunicación con clientes que no tienen muchos conocimientos técnicos.
Técnicas de Cuarta Generación	- Su empleeo reduce el tiempo requerido para producir software, apoyando desde las etapas tempranas de levantamiento de requerimientos y análisis. - Se emplean para aplicaciones pequeñas y medianas. - La cantidad de análisis, levantamiento de requerimientos y diseño se reduce. - Mejora significativa la productividad de la gente que construye el software.	-El uso de T4G exige el mismo o más tiempo de análisis, ingeniería de requerimientos, diseño y prueba (actividades de ingeniería del software). - Las herramientas actuales de T4G no son fáciles de utilizar. - El código fuente producido por las herramientas es ineficiente y su mantenimiento es complicado en grandes sistemas de software.

2.2 METODOLOGÍAS DE DESARROLLO DE SOFTWARE

El desarrollo de software es una tarea compleja, por lo que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte están las propuestas tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los elementos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero han presentado problemas en muchos otros. Una posible mejora es incluir en los procesos de desarrollo más actividades, más restricciones, basándose en los puntos débiles detectados. El resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto de software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requerimientos cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo manteniendo una alta calidad. Las metodologías ágiles están revolucionando la forma de producir software, y generando un amplio debate entre sus seguidores y quienes no las ven como alternativa para las metodologías tradicionales.

Un objetivo de la Ingeniería de Software de los últimos años ha sido encontrar procesos y metodologías, que cumplan con ser: sistemáticos, predecibles y repetibles, para mejorar la productividad en el desarrollo y calidad del producto de software. La evolución de la disciplina ha traído consigo propuestas diferentes para mejorar los resultados del proceso de construcción. Las metodologías tradicionales continúan en el énfasis de la **planificación** y las metodologías ágiles haciendo énfasis en la **adaptabilidad del proceso**.

2.2.1 *¿Qué es una metodología de desarrollo de software?*

Es un conjunto integrado de técnicas y métodos que permiten abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo.

Las metodologías son combinaciones de los modelos de proceso (cascada e incremental solo por mencionar los más utilizados). Definen artefactos, roles y actividades,

junto con prácticas y técnicas recomendadas. Son un modo sistemático de realizar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Comprende los procesos a seguir para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que se cumple el objetivo para el cual fue creado.

Se destaca de una metodología que:

1. Optimiza el proceso y el producto de software.
2. Los métodos guían en la planificación y desarrollo del software.
3. Define el: qué, cómo y cuándo durante todo el desarrollo y mantenimiento de un proyecto.

Elementos que forman parte de una metodología

1. Fases: las tareas a realizar en cada fase.
2. Productos: Entras y salidas de cada fase, documentos, entre otros.
3. Criterios de evaluación: del proceso y del producto. Saber si se han logrado los objetivos.

Una metodología de desarrollo de software se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. No tiene que ser necesariamente adecuada para ser empleada en todos los proyectos. Cada metodología disponible es más adecuada para ciertos tipos de proyectos, para lo cual se deben considerar: cuestiones técnicas, el levantamiento de requerimientos, la organización del proyecto, el equipo, solo por mencionar algunas.

2.2.2 *Metodologías Tradicionales*

Estas metodologías son denominadas como las metodologías pesadas. Se centran en la documentación, planificación y procesos (levantamiento de requerimientos, técnicas de administración, revisiones, control de la calidad, entre otras más).

En estas metodologías el principal problema es que raras veces se logra planificar bien el esfuerzo requerido para seguir la metodología. Pero si se logran definir métricas que apoyen la estimación de las actividades de desarrollo, muchas prácticas de metodologías tradicionales podrían ser apropiadas. El no poder predecir siempre los resultados de cada proceso no significa que estemos frente a una disciplina de azar.

Lo que significa es que se deben adaptar los procesos de desarrollo que son llevados por parte de los equipos que desarrollan software.

Dentro de las metodologías tradicionales conocidas se encuentran: RUP (Rational Unified Process), MSF(Microsoft Solution Framework) y Win-Win Spiral model.

A continuación se describen algunas de las métodologías tradicionales así como la forma en la que abordan los requerimientos de software o el proceso de levantamiento de requerimientos:

2.2.2.1 *Proceso Unificado Rational*

El Proceso Unificado Racional (por sus siglas en Inglés, Rational Unified Process RUP) es un proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado (UML) constituyen la metodología estándar más utilizada para el análisis, levantamiento de requerimientos, implementación y documentación de sistemas orientados a objetos.

El RUP tiene un enfoque centrado en la arquitectura y basado en casos de uso (Per Kroll, 2003), que trata de un conjunto de metodologías adaptables al contexto y necesidades de cada organización, donde el software es organizado como una colección de unidades atómicas llamados objetos, constituidos por datos y funciones, que interactúan entre sí (Pérez, 2011).

RUP es explícito en la definición de software y su trazabilidad, contempla en relación causal de los programas creados desde los requerimientos de software hasta la implementación y pruebas. Identifica claramente a los profesionales (actores) involucrados en el desarrollo del software y sus responsabilidades en cada una de las actividades (Pérez, 2011)(Per Kroll, 2003).

En el RUP el ciclo de vida del proyecto consiste en cuatro fases. Estas permiten que el proceso sea presentado a alto nivel, la clave del proceso recae en las iteraciones de desarrollo dentro de todas las fases. Cada fase se tiene un conjunto de requerimientos a implementar para cumplir con un objetivo clave y un hito al final que denota que el objetivo se ha logrado.

Las cuatro fases en las que divide el ciclo de vida del proyecto son:

1. **Iniciación**.- se define el alcance del proyecto a través de los requerimientos de software.

2. **Elaboración**.- se analizan las necesidades y requerimientos a mayor detalle y se definen los principios arquitectónicos.
3. **Construcción**.- se crea el diseño de la aplicación y se codífica.
4. **Tansición**.- se realiza la entrega del sistema al usuario.

La metodología admite la personalización y creación de procesos, se puede establecer configuraciones para equipos pequeños o grandes de integrantes, con enfoques de desarrollo disciplinados o menos formales (Per Kroll, 2003).

Contiene varias configuraciones de proceso sobre cómo desarrollar software para guiar a los analistas de requerimientos de software, analistas de datos, desarrolladores, evaluadores, gerentes de proyecto, gerentes de configuración, solo por mencionar algunos roles principales.

Las guias promueven la generación de documentos, diseños y planes, pero según Per Kroll (2003) son una mala indicación del progreso del proyecto, porque la evaluación empleando simplemente esos artefactos es subjetiva y su importancia pasa a ser secundaria en comparación con el código (software), es decir verificar que el código compile y pase con éxito las pruebas es el mejor indicador de progreso. Este aspecto de la metodología muestra los altos conocimientso técnicos que se requieren para establecer las pruebas y verificar que los objetivos se están cumpliendo. Se puede desaprovechar los artefactos visuales y documentales dado que estos mejoran la comunicación dentro del equipo y con los interesados en el proyecto, a la par son considerados como elementos que al utilizarse en el proceso de implementación mejoran los resultados de usabilidad de la aplicación.

2.2.2.2 *Microsoft Solution Framework MSF*

La metodología Microsoft Solution Framework (MSF) es una guía de desarrollo de software flexible que permite aplicar de manera individual e independiente cada unos de sus componentes, es escalable pues está diseñada para adaptarse a la magnitud del proyecto. La metodología MSF está basada en un conjunto de principios, modelos, disciplinas, conceptos, directrices y practicas aprobadas por Microsoft, que asegura resultados con menor riesgo y de mayor calidad, centrándose en el proceso y en las personas (Pérez, 2011)(Pavlov et al., 2008).

MSF se introdujo en 1994 como un conjunto de las mejores prácticas de los esfuerzos en el desarrollo de Software de Microsoft. Esta metodología evoluciona con la experiencia de los grupos de trabajo. El MSF está basado en los modelos espiral y en

cascada, lo cual indica que toma elementos de los métodos tradicionales que aún son referentes importantes para procesos de software, pero toma elementos ágiles como ser adaptable, flexible y escalable (Pérez, 2011)(Pavlov et al., 2008).

El MSF esta formado por 3 componentes: principios fundamentales, modelos y disciplinas.

Principios fundamentales

Los principios fundamentales del MSF son 8 valores y normas que contribuyen a mejorar el trabajo en equipo y a centrarse en mantener los objetivos del proyecto en marcha.

1. Fomentar la comunicación abierta
2. Trabajar bajo una visión compartida
3. Empoderar a los miembros del equipo
4. Establecer la rendición de cuentas claras
5. Centrarse en ofrecer valor empresarial
6. Mantenerse ágil, en espera de un cambio
7. Invertir en la calidad
8. Aprender de todas las experiencias

Modelos

Los modelos describen esquemas a seguir para la organización de los equipos y los procesos del proyecto. Se específica un modelo para el equipo de trabajo y otro para los procesos.

Equipo de trabajo

Se encarga de organizar a las personas que realizan el trabajo y se asegura que todas las metas del proyecto se cumplan. Define principios, roles y actividades involucradas al equipo en todas las decisiones fundamentales.

Proceso

Se encarga de organizar los procesos necesarios para lograr llevar a término una solución. Divide las tareas del proyecto en 5 fases, las cuales proporciona herramientas para mejorar el control sobre el proyecto, minimizar el riesgo y aumentar la calidad del producto. Presenta prácticas inherentes al desarrollo de software como la especificación, el desarrollo, la validación y la evolución del software (Pérez, 2011).

La metodología MSF establece 5 fases con sus respectivas tareas(Pérez, 2011):

1. Fase de visión.- se debe tener el objetivo y limitaciones del proyecto.
2. Fase de planificación.- se debe tener la ingeniería de requerimientos, planificación y administración de riesgos.
3. Fase de desarrollo.- se codífica y se realizan las respectivas pruebas.
4. Fase de estabilización.- se realizan pruebas Beta, se crea un plan de administración de incidencias, se revisa la documentación final, entre otras.
5. Fase de implementación.- se libera el producto de software, se crea un registro de mejoras y sugerencias, se revisa las guías y manuales de usuario.

Las normas son consideras aspectos de la metodología que apoyan en los procesos complejos de levantamiento de requerimientos dado que permiten una participación constante entre los integrantes del equipo, así como con el cliente. Propicia que durante el desarrollo del producto se invierta en la calidad, es decir cumplir con requerimientos marcados al inicio de la planificación y verificar el avance de los requerimientos solicitados. El principio de establecer la rendición de cuentas claras contibuye a que se planifiquen honestamente el conjunto de requerimientos que se van a implementar para entregar un producto de acuerdo a lo solicitado por el cliente.

2.2.2.3 *Ganar ganar (win-win)*

Esta metodología aborda la comunicación con el cliente, con el objetivo de mostrar los requerimientos del cliente. En un contexto ideal, el desarrollador solicita al cliente lo que se necesita y el cliente proporciona detalles suficientes para continuar, en la realidad el cliente y el desarrollador entran en un proceso de negociación, donde se le pregunta al cliente para sopesar la funcionalidad, rendimiento, y otros productos o características del sistema frente al costo y tiempo de liberación del producto(Pressman, 2002).

Las mejores negociaciones se esfuerzan en obtener win-win. Esto es que el cliente gana obteniendo el producto o sistema que satisface la mayor parte de sus necesidades y requerimientos y el desarrollador gana trabajando para conseguir presupuestos y lograr una fecha de entrega realista cubriendo los objetivos propuestos.

El modelo en espiral Win-Win de Boehm en Boehm et al. (2001), define un conjunto de actividades de negociación al principio de cada paso alrededor de la espiral. En el modelo se definen las siguientes actividades:

1. Identificación del sistema o subsistemas clave de los directivos.
2. Determinación de las condiciones de victoria (Win) de los directivos.

3. Negociación de las condiciones de victoria (Win) de los directivos para reunirlas en un conjunto de condiciones victoria-victoria (Win-Win) para todos los afectados, incluyendo el equipo del proyecto de software.

Además del énfasis realizado en la negociación inicial, el modelo en espiral win-win introduce tres hitos en el proceso, llamados puntos de fijación, que ayudan a establecer la completitud de un ciclo alrededor de la espiral y proporcionan hitos de decisión antes de continuar el proyecto de software.

En esencia, los puntos de fijación representan tres visiones diferentes del progreso mientras que el proyecto recorre la espiral. El primer punto de fijación, llamado **objetivos del ciclo de vida**, define un conjunto de objetivos para cada actividad principal de ingeniería de software. El segundo punto de fijación, llamado **arquitectura del ciclo de vida**, establece los objetivos que se deben conocer, mientras que se define la arquitectura del software y el sistema. La **capacidad operativa inicial** es el tercer punto de fijación y representa un conjunto de objetivos asociados a la preparación del software para la instalación y distribución (Daniel, 2010)(Pressman, 2002).

Es ideal para manejar proyectos que requieran la incorporación de nuevas tecnologías, o para desarrollar productos completamente nuevos o con un nivel alto de inestabilidad de los requerimientos. Esta nueva filosofía se encuentra representada por ciclos de desarrollo evolutivo e iterativo en forma de espiral, cuyo avance angular representa el progreso del desarrollo, en tanto que el desplazamiento radial desde el centro hacia fuera indica el incremento de los costos de desarrollo en forma acumulativa ver Figura 9.

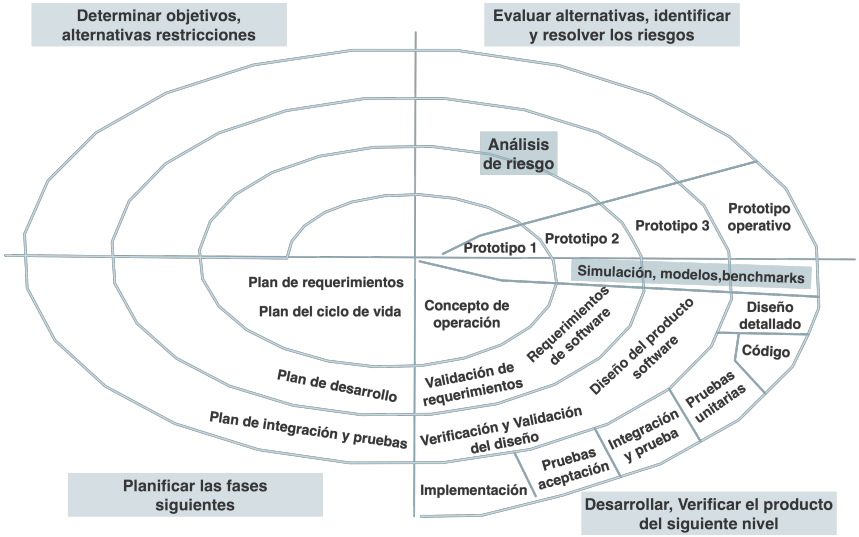


Figura 9: *Metodología Win-Win espiral [Imagen elaborada por el autor].*

2.2.3 *Métodos Ágiles*

A principios de la década de los años 1990 surge un enfoque revolucionario para ese momento dado que iba en contra de la creencia de que para lograr obtener software en tiempo, costo y con la calidad requerida, se consigue únicamente mediante procesos altamente definidos y rígidos. El enfoque fue planteado por Martin Fowler en 1991 y se dió a conocer en la comunidad de Ingeniería de Software como RAD o Rapid Application Development. Estos modelos son basados en iteraciones, se trata de un método de construcción de productos cuyo ciclo de vida está compuesto por un conjunto de entregas de versiones de Software, donde entre versiones se van cubriendo más funcionalidades .

En Utah Estados Unidos en Febrero de 2001 nace el término ágil aplicado al desarrollo de Software. Con la participan de un grupo de 17 expertos de la industria del software, con el objetivo de esbozar los valores y principios que deberían permitir a los equipos desarrollar Software de manera rápida y responder de manera favorable a los cambios que pueden surgir a lo largo del proyecto. Con esto se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas (Daniel, 2010).

El término "Métodos Ágiles" se utiliza para definir aquellos métodos que surgen como alternativa a las metodologías formales, las cuáles se consideran excesivamente "pesadas"(ver sección 2.2.2) y rígidas por su carácter normativo y con una gran dependencia de planificaciones detalladas (etapas o fases), previas al desarrollo como las etapas de levantamiento de requerimientos, planeación, análisis del dominio, entre otras. Las metodologías ágiles reconocen las características inherentes de complejidad del software, y el carácter empírico que debe tener un proceso de desarrollo del mismo. Entre las metodologías ágiles más destacadas hasta el momento podemos nombrar:

1. XP Extreme Programming
2. Scrum
3. Crystal Clear
4. DSDM Dynamic Systems Development Method
5. FDD Feature Driven Development
6. ASD Adaptative Software Development

A continuación se describen algunas de estas metodologías.

2.2.3.1 *Programación extrema*

La programación extrema XP (del inglés Extreme Programming XP), es una disciplina de desarrollo de software basada en los métodos ágiles, que evidencia principios tales como el desarrollo incremental, la participación activa del cliente, el interés en las personas y no en los procesos como elemento principal, y aceptar el cambio y la simplicidad. El trabajo fundamental se publicó por Kent Beck en 1999, y tomó el nombre de Programación Extrema por las prácticas reconocidas en el desarrollo de software y por la participación del cliente en niveles extremos (Pérez, 2011).

El método XP se fundamenta en cuatro valores: comunicación, simplicidad, retroalimentación (feedback) y coraje. Pero tan conocidos como sus valores son sus prácticas que involucran al equipo de trabajo, los procesos y el cliente, los cuales son (Pérez, 2011)(Reynoso, 2004):

1. **Planificación incremental**: se toman los requerimientos de software en Historias de Usuario (HU), las cuales son negociadas progresivamente con el cliente, es decir, existe una participación constante entre los analistas y lo interesados en el proyecto. Buscan determinar rápidamente el alcance de la versión siguiente, combinando prioridades de negocio definidas por el cliente con las estimaciones técnicas de los programadores. El grupo de desarrollo estima el esfuerzo necesario para implementar las historias del cliente y éste decide sobre el alcance y la agenda de las entregas. Con las agendas el equipo de trabajo realiza calendarios individuales para desarrollar los requerimientos (HU) que cumplen los objetivos específicos por entrega. Las historias se escriben en pequeñas fichas, que algunas veces se tiran (para no viciar en el desarrollo del proyecto). Cuando esto sucede, lo único restante que se parece a un requerimiento es un conjunto de pruebas automatizadas y las pruebas de aceptación (los términos o criterios en los que el cliente acepta el resultado) (Reynoso, 2004).
2. **Entregas pequeñas**: se desarrolla primero la más mínima parte útil que le proporcione funcionalidad al sistema, y poco a poco se efectúan incrementos que añaden funcionalidad a la primera entrega, cada ciclo termina con una entrega del sistema, en la Figura 10 se muestra como es el proceso de entrega en XP. El ciclo de vida termina cuando ya no hay más ciclos de entrega y el sistema ha cumplido con los requerimientos y los objetivos para el cual fue diseñado, de no

Figura 10: *Ciclo de entrega de XP [Imagen elaborada por el autor].*

ser así, se deberá continuar con el ciclo especificado en la Figura 10 hasta que la funcionalidad del sistema sea la deseada.

3. **Diseño sencillo**: solo se efectúa el diseño necesario para cumplir con los requerimientos actuales, es decir, no se abordan requerimientos futuros.
4. **Desarrollo previamente aprobado**: una de las características relevantes y propias de XP es que primero se escriben las pruebas y luego se da la codificación, esto con la finalidad de asegurar la satisfacción del requerimiento.
5. **Limpieza del código o refactorización**: consiste en simplificar y optimizar el programa sin perder funcionalidad, es decir, alterar su estructura interna sin afectar su comportamiento externo.
6. **Programación en parejas**: es otra de las características de ésta metodología, que propone que los desarrolladores trabajen en parejas en una terminal, verificando cada uno el trabajo del otro y ayudándose para buscar las mejores soluciones. Se entiende que de esta forma el trabajo será más eficiente y de mayor calidad.
7. **Propiedad colectiva**: el conocimiento y la información deben ser de todos, por lo que todos los programadores poseen todo el código y cualquiera puede sugerir y realizar mejoras.
8. **Integración continua**: al terminar una tarea, ésta se integra al sistema entero y se realizan pruebas de unidad a todo el sistema, ésta práctica permite que la aplicación sea más funcional en cada iteración y garantiza su funcionamiento con los demás módulos del sistema.
9. **Ritmo sostenible**: no es aceptable trabajar durante grandes cantidades de horas ya que se considera que puede reducir la calidad del código y la productividad del equipo a mediano plazo, se sugieren 40 horas semanales.

10. **Cliente presente**: se debe tener un representante (Cliente o usuario final) tiempo completo, ya que en XP éste hace parte del equipo de desarrollo y es responsable de formular los requerimientos para el desarrollo del sistema.

El ciclo de desarrollo de XP consiste en los siguientes pasos (Pérez, 2011):

- **Fase de planeación**: esta fase inicia con las historias de usuario que describen las características y funcionalidades del software. El cliente asigna un valor o prioridad a la historia, los desarrolladores evalúan cada historia y le asignan un costo el cual se mide en semanas de desarrollo.
- **Fase de diseño**: el proceso de diseño debe procurar diseños simples y sencillos para facilitar el desarrollo. Se recomienda elaborar un glosario de términos y la correcta especificación de métodos y clases para facilitar posteriores modificaciones, ampliaciones o reutilización del código.
- **Fase de codificación**: en ésta fase los desarrolladores diseñn las pruebas de unidad que ejercitarán cada historia de usuario. Después de tener las pruebas, los desarrolladores trabajan en parejas concentrados en lo que se debe implementar para pasar la prueba de unidad.
- **Fase de pruebas**: las pruebas de unidad se implementan con un marco de trabajo que permita automatizarlas, con la finalidad de realizarlas sin afectar el desarrollo del proyecto.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo del estimado, ya que se corre el riesgo de perder calidad en el software o no que no se cumplan los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse de que el sistema tenga el mayor valor de negocio posible.

La metodología XP con su enfoque basado en los métodos ágiles, su énfasis en la gestión del recurso humano el cuál es uno de los puntos más críticos en todo proyecto, y sus principios de previsibilidad y adaptabilidad hacen de esta metodología una opción adecuada.

2.2.3.2 *Scrum*

Scrum es un proceso ágil que se puede usar para administrar y controlar desarrollos complejos de software y productos usando prácticas iterativas e incrementales.

Se le atribuye a Hirotaka Takeuchi e Ikujiro Nonaka quienes describieron un enfoque integral que incrementaba la velocidad y flexibilidad del desarrollo de nuevos productos comerciales (Ken and Sutherland, 2017). Los creadores de este enfoque propone que las fases se solapan fuertemente y el proceso entero es llevado a cabo por un equipo multifuncional a través de las diferentes fases. En 1991 se hace referencia a este enfoque como Scrum, un término de rugby en el que los jugadores se reúnen para disputar el balón y el equipo trata de ganar distancia como una unidad pasando el balón una y otra vez.

Scrum da prioridad a los individuos y las interacciones sobre los procesos y las tareas (Ken and Sutherland, 2017), lo cual significa que gran parte del éxito del proyecto radica en la forma cómo el equipo se organice para trabajar. Se debe tener una cohesión fuerte de equipo ya que el triunfo de un hito [1] no es de un sólo miembro sino de todo el equipo, todos se colaboran entre sí, y empujan a los integrantes que no están a la par con el equipo (Pérez, 2011).

El enfoque prioriza obtener software funcional sobre la excesiva documentación, a diferencia de RUP el cual es estricto en documentación. Se presenta al cliente las soluciones operables y no solo reportes de progresos, de esta forma el cliente puede decidir avanzar o parar, en otros enfoques solo se ven resultados al final.

Se promueve la colaboración con el cliente en lugar de una rígida negociación de contratos. Por lo cual, es importante tener capacidad de respuesta para los cambios en lugar de seguir estrictamente una planificación, partiendo del principio que el proyecto software es cambiante. El propósito es que el cliente vaya observando los resultados, pueda decidir cambios en la marcha o incluso darle un giro completo al proyecto.

2.2.3.3 *Roles*

Scrum divide su equipo de trabajo en cinco grupos de personas (Reynoso, 2004)(Pérez, 2011):

- **Propietario del producto**: es la persona que determina las prioridades del proyecto, debe conocer muy bien los requerimientos y necesidades del cliente respecto al producto. Debe conocer de primera mano lo que el cliente o grupos de interesados quiere del producto, para de esta forma guiar al equipo Scrum hacia la consecución de los objetivos.

1 Los hitos en un proyecto son una serie de etapas, que tienen un objetivo y fecha determinada

- **Scrum master**: es el encargado de gestionar y facilitar la ejecución del proyecto, debe asegurar el seguimiento de la metodología y el cumplimiento de las metas trazadas, así como de atender y solucionar los asuntos externos al proyecto.
- **Equipo Scrum**: es el corazón de la metodología pues ellos construyen el producto, está conformado por los desarrolladores.
- **Interesados**: también llamados `StakeHolders` son la parte interesada que hace referencia a una persona, organización o empresa que tiene interés en el proyecto, en el cual puede afectar o beneficiar su desarrollo. En algunos casos patrocinan o promueven el proyecto en ese caso son considerados patrocinadores.
- **Usuarios**: quizá uno de los menos tenidos en cuenta pero finalmente son ellos los que realizaran las pruebas lógicas de la aplicación y verifican si se cumplen sus expectativas. Los clientes pueden aportar ideas o necesidades no consideradas por el propietario del producto.

2.2.3.4 *Reuniones*

Las reuniones cortas del equipo son elementos fundamentales de Scrum y se realizan periódicamente. En el método se definen la forma de las reuniones del equipo de trabajo y los resultados que a partir de ellas se deben generar. Por ejemplo las reuniones no deben durar más de 15 minutos y se deben llevar de pie, para evitar distracciones, cuando se presentan controversias sobre un tema en particular el Scrum master debe solicitar una reunión aparte con la equipo o miembro en particular para evitar retrasar la reunión actual y dar agilidad al proceso.

- **Planificación del sprint** [2]: es una jornada de trabajo muy importante ya que su mala planificación puede arruinar todo el Sprint. En ésta reunión el propietario del producto explica las prioridades y dudas del equipo, estos estiman el esfuerzo de los requerimientos prioritarios incluyendo una lista de miembros y nivel de dedicación, y a partir de ésta se elabora la pila de Sprint. El Scrum master define en una frase el objetivo del Sprint.
- **Reunión diaria**: comprende una reunión de mínimo 15 minutos y máximo 30 minutos de duración, en el mismo lugar de reunión y a la misma hora. La reunión está dirigida por el Scrum master y sólo puede intervenir el Equipo Scrum. Éste hace las siguientes preguntas a cada miembro del equipo:

2 Sprint en Scrum se refiere a la planificación de un esfuerzo para conseguir objetivos y que se encuentran calendarizados por días laborales o por horas de trabajo (Ken and Sutherland, 2017).

1. ¿Qué hiciste ayer?
2. ¿Cuál es el trabajo para hoy?
3. ¿Qué necesitas para realizar tus pendientes?

Una vez conocida la situación actual del equipo se actualiza la pila del sprint y el Scrum master debe tomar decisiones de inmediato, también tiene la responsabilidad de señalar los obstáculos que deben ser resueltos externamente para no alargar la reunión.

- **Revisión del sprint**: es una reunión informativa, aproximadamente de 4 horas, en la que el moderador es el Scrum master. En ésta reunión se hace la presentación del incremento, el planteamiento de sugerencias y anuncio del próximo Sprint.
- **Retrospectiva del sprint**: después de cada sprint, se reúnen los miembros del equipo (Aproximadamente 4 horas) y expresan sus opiniones del sprint recién superado, con la finalidad de mejorar los procesos. Es básicamente una reunión de evaluación y mejoramiento.

2.2.3.5 *Proceso Scrum*

En Scrum a diferencia de XP que también está basado en los métodos ágiles, se divide el proyecto en periodos de 4 semanas aproximadamente, cada periodo se denomina sprint y cada equipo recibe una lista de pedidos a ejecutar en un sprint determinado. En la Figura 11 se ve cómo los valores, artefactos y reuniones se conjugan en el proceso de desarrollo Scrum, que se compone de 5 fases las cuales contienen las actividades a desarrollar durante un periodo, comprendidas de la siguiente manera(Reynoso, 2004)(Daniel, 2010)(Pérez, 2011):

- **Revisión de planes de Release**: es la "planificación del sprint". Ésta fase se ejecuta una vez establecida la pila de producto y es llevada a cabo por el equipo a fin de evaluar las diferentes factibilidades de los requerimientos y estimaciones, basándose en la funcionalidad y las prioridades de la pila de producto.
- **Distribución, revisión y ajustes de estándares de producto**: corresponde a la "Pila de SPRINT". En ésta fase los desarrolladores realizan los ajustes de los estándares y requerimientos mínimos, dejando todo listo para comenzar con la fase de sprint.

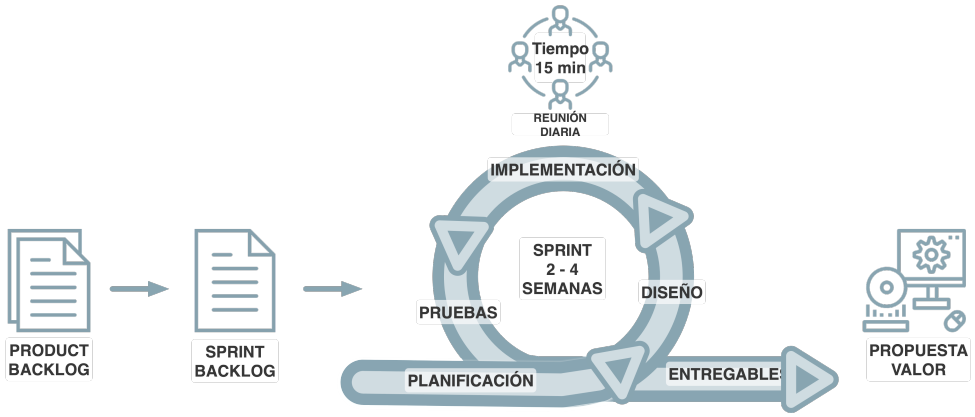


Figura 11: *Flujo de proceso de SCRUM [Imagen elaborada por el autor].*

- **Sprint**: ésta fase de aproximadamente 30 días es donde se efectúa el desarrollo del software y se llevan a cabo las reuniones, consta de las siguientes subfases: elaborar, integrar, revisar y ajustar.
- **Revisión del sprint**: corresponde al ïncremento". En ésta fase se revisa el sprint y si es necesario se añaden nuevos pendientes a la pila de producto. Éste proceso se repite hasta que el producto esté listo para la fase de cierre.
- **Cierre**: en ésta fase se da lugar a la depuración y correcciones de errores (debugging), este proce dimiento se repite hasta alcanzar la calidad en el producto. Posterior a las correcciones y pruebas se realiza el estudio del mercado y promoción del producto y al terminar ésta fase el proyecto queda cerrado.

Puntualizando scrum ofrece herramientas que permiten administrar el equipo de trabajo hasta el punto de proponer tiempos para el proceso de desarrollo de software y para las reuniones del equipo, con la finalidad de asegurar el cumplimento de los objetivos del proyecto.

2.2.3.6 *Crystal Clear*

Crystal Clear es un miembro de la familia de metodologías Crystal como describe Alistair Cockburn y se considera un ejemplo de metodología ágil. Está pensado para aplicarse a equipos pequeños de 6 a 8 desarrolladores ubicados en el mismo sitio trabajando en sistemas que no son críticos (van Vliet, 2007)(Abreu et al., 2009)(Moran, 2014).

El nombre Crystal se deriva de la caracterización de los proyectos en 2 dimensiones (Moran, 2014): tamaño y complejidad:

- Clear: para equipos de hasta 6 personas o menos.
- Amarillo: para equipos entre 7 a 20 personas.
- Naranja: para equipos entre 21 a 40 personas.
- Roja: para equipos entre 41 a 80 personas.
- Marron: para equipos entre 81 a 200 personas.

En el método Crystal se mantiene que a más personas en el proyecto, se requiere más coordinación. A más criticidad en el software, se requiere más rigurosidad en el proceso. Siendo la comunicación entre los participantes el factor más determinante en el proyecto (Cockburn, 2002)(Moran, 2014).

La familia de metodologías Crystal se centra en la eficiencia y habitabilidad como componentes de la seguridad del proyecto. Crystal Clear se centra en las personas, no en los procesos o artefactos.

Crystal cumplen con propiedades esenciales:

1. Entrega frecuente de código usable a los usuarios. Con base en un ciclo de vida iterativo e incremental, puede haber desde entregas semanales hasta trimestrales.
2. Mejora reflexiva se trata de la mejora continua. Las iteraciones ayudan en ajustar el proyecto.
3. El equipo se debe ubicar en una misma ubicación física, para lograr la comunicación cara a cara.
4. Seguridad personal, en la que todos los miembros del equipo puedan expresar su opinión sin miedos y se consideren en cuenta.
5. Enfocarse en periodos de no interrupción al equipo (aproximadamente de 2 horas), donde entan los requerimientos, objetivos y prioridades claros, que permitan definir tareas concretas.
6. Fácil acceso a los usuarios expertos, en crystal no se exige que los usuarios estén continuamente junto al equipo desarrollo (no todas las organizaciones pueden hacerlo), pero como mínimo una reunión semanal.
7. Pruebas automatizadas, administración de la configuración e integración frecuente de los avances.

El método Crystal tiene como premisa que el equipo es el factor clave en el desarrollo de software por lo que la mayor parte de los esfuerzos esta orientado a fortalecer

las destrezas y habilidades del equipo, así como tener bien definidas y aceptadas las política de trabajo en equipo factor que depende del tamaño del grupo (Moran, 2014).

2.2.4 *Resumen de ventajas y desventajas de las metodologías en el proceso de levantamiento de requerimientos*

En la Tabla 2 se muestra un resumen de las ventajas y desventajas que se presentan en el proceso de levantamiento de requerimientos en la metodologías de desarrollo de software abordadas:

Tabla 2: *Resumen de las Ventajas y desventajas de las metodologías respecto al proceso de levantamiento de requerimientos*

-	Ventajas	Desventajas
Proceso Unificado Racional	- En cada fase del desarrollo se revisan los objetivos, por lo que los requerimientos obtenidos en las etapas tempranas de cada ciclo son seleccionados para cumplir los objetivos. - Funciona bien en proyectos de innovación en los que se pueden definir los objetos y acciones que estos realizan desde la etapas tempranas de análisis. - Seguimiento detallado de los entregables y cumplimiento de los objetivos de cada iteración.	- La recolección de requerimientos y la evaluación de riesgos es compleja debido al entendimientos de actividades y descripción de los objetos involucrados. - Una excesiva flexibilidad en el desarrollo puede impedir que no se alcancen los objetivos propuestos. - Los cliente a menudo no son capaces de describir claramente las características de los objetos y los escenarios donde interactuan, por lo que es difícil establecer los requerimientos y el alcance del proyecto.
MSF	- Propueve el trabajo en equipo y colaborativo lo que apoya en los procesos de levantamiento de requerimientos y análisis.	- Este modelo genera mucha documentación en sus fases que puede ser agobiante en el equipo. - El análisis de los requerimientos y de los riesgo demora mucho tiempo frenando el avance del proyecto.

Tabla 2: *Resumen de las Ventajas y desventajas de las metodologías respecto al proceso de levantamiento de requerimientos*

-	Ventajas	Desventajas
MSF	- Con el soporte de Microsoft se cuenta con plantillas que apoya el proceso de documentación en todas las etapas del desarrollo incluyendo el levanamiento de requerimientos y el documento de especificación de requerimientos.	- Al tratarse de un modelo de Microsoft implica que se tiene que utilizar herramientas solo de Microsoft. Lo que puede frenar la adquisión de requerimientos, análisis, planeación y desarrollo, cuando el equipo no se acopla a esas tecnologías o cuando las herramientas propuestas por la Microsoft no cubren las necesidades del proyecto.
Ganar-Ganar	- El cliente puede dar un seguimiento de los cambios y avances en el desarrollo y con ello reaccionar mejor ante los riesgos. - Dado que el cliente da un seguimiento del avance del proyecto se puede obtener una mejor perspectiva de la calidad que tiene el producto. - Durante el proceso se integra el desarrollo con el mantenimiento, a través de establecer los requerimientos negociados en las etapas iniciales del ciclo, con el objetivo de planificar el posible mantenimiento del sistema.	- Es un modelo que genera mucho trabajo adicional, por los entregales y documentos que se elaboran desde el análisis de requerimientos y la planeación. - Se requiere que los analistas tengan mucha experiencia identificando riesgos. - No es adecuado para proyectos pequeños por los tiempos y procesos que plantea el modelo.

Tabla 2: *Resumen de las Ventajas y desventajas de las metodologías respecto al proceso de levantamiento de requerimientos*

-	Ventajas	Desventajas
Programación extrema XP	- Ayuda al cliente a definir mejor los requerimientos y ver las necesidades reales del producto. Dado que se emplean las historias de usuario, donde se resume brevemente el como realizan actualmente las actividades a automatizar un usuario tipo. - Se prueba constantemente el código durante el proyecto y con ello se verifica si se cumplen los requerimientos de software. - Las pruebas de unidad permiten comprobar la calidad del sistema conforme avanza el desarollo. - Permite introducir cambios en las iteraciones siguientes del ciclo y verificar el avance.	- No es adecuado para proyectos a largo plazo por los procesos que plantea el modelo. - No sé pueden definir los costos ni tiempo del producto. - Los requerimientos van creciendo conforme a las entregas y por consiguiente el proyecto puede crecer a un estado no visualizado inicialmente. - Se requiere que los interesados esten presentes o involucrados constantemente con el proyecto para los requerimientos y las entregas.
Scrum	- Administra las necesidades y requerimientos del cliente, en el cual los clientes participan en las iteracines donde expresan sus necesidades y requerimientos y propoene algunas soluciones.	- Funciona con equipos reducidos de trabajo. Las empresas grandes, deben estar sectorizadas en grupos con objetivos y requerimientos de software concretos, si desean abordar el mismo producto de software. De lo contrario el efecto de la técnica se ve reducida o incluso se pierde.

Tabla 2: *Resumen de las Ventajas y desventajas de las metodologías respecto al proceso de levantamiento de requerimientos*

-	Ventajas	Desventajas
Scrum	- Se obtiene una evaluación conjunta de la resolución de los requerimientos con la participación y revisión constante de los interesados. - Cada iteración arroja una serie de resultados, por lo que no es necesario que el cliente espere hasta el final para verificar se cumplan los requerimientos del producto solicitado. - Se adapta a cualquier contexto, área o sector de la gestión. No es una técnica exclusiva de ninguna disciplina, este ocurre por la constante comunicación con los interesados. - Se puede dar una pronta respuesta a los riesgos que pueden afectar a un proyecto, en el mismo momento de su aparición. La intervención de los equipos de trabajo es inmediata.	- Requiere un exhaustivo análisis de los requerimientos e historias de usuario, de ellas parten la definición de las tareas y sus plazos. Cuando estos aspectos no se definen adecuadamente, Scrum presenta conflictos, debido a la división del trabajo en iteraciones (y de éstas en tareas específicas) que son la esencia de este método. - Exige un alto grado de formación. No es una modalidad propia de grupos de principiantes o que apenas estén en proceso de formación. Gran parte del éxito de Scrum radica en la experiencia que aportan los profesionales de los equipos, quienes por lo general con la acumulación años de experiencia, pueden anticipar requerimientos, riesgos y dar soluciones eficaces.
Crystal	- Presenta un levantamiento de requerimientos y planificación transparente para los clientes, dado que se solicitan reuniones constantes.	- Puede no ser factible para proyectos muy grandes con requerimientos complejos y que no se obtienen de forma directa.

Tabla 2: *Resumen de las Ventajas y desventajas de las metodologías respecto al proceso de levantamiento de requerimientos*

-	Ventajas	Desventajas
Crystal	- Se definen cuales serán los requerimientos y objetivos al iniciar cada iteración. - Permite una útil realimentación de los usuarios finales que emplearan el producto, con lo que se puede medir el cumplimiento de los requerimientos. - El equipo puede reducir artefactos intermedios a medida que se produzca código funcional cumpliendo con los requerimientos.	- Las guías se encuentran en constante actualización, dado que se revisan los casos de éxito. - Delimita el alcance del proyecto con el cliente, por lo que se pueden quedar funcionalidad incompletas o se pueden solicitar más requerimientos de los planeados en un inicio. - Se puede dejar abierta la posibilidad de agregar y suprimir fases, requerimientos, productos, estrategias, técnicas que pueden afectar en el desarrollo del sistema.

2.3 LEVANTAMIENTO DE REQUERIMIENTOS

El software es un sistema computacional que permite cumplir con objetivos, prestar servicios y solucionar problemas de los usuarios en sus actividades cotidianas. Algunos ejemplos de estas soluciones de software son: procesadores de texto, hojas de cálculo, navegadores Web, aplicaciones de uso dedicadas, agendas electrónicas, solo por mencionar algunas.

El desarrollo de software consiste de múltiples fases que requieren de amplios conocimientos técnicos en software, en procesos y en trabajo en equipo. La ingeniería de requerimientos es una de esas fases importantes y crítica en el desarrollo de software.

2.3.1 *Ingeniería de software*

La ingeniería de requerimientos es una fase importante y crítica en el desarrollo de software (Aristizábal-Mejía and Torres-Moreno, 2009)(Hickey and Davis, 2002)(Zowghi and Coulin, 2005). Aplicar y dirigir correctamente las actividades de la ingeniería de requerimientos proporciona las bases para que el desarrollo de software se realice exitosamente (Aristizábal-Mejía and Torres-Moreno, 2009)(Zowghi and Coulin, 2005). Cuando alguna de las actividades de la ingeniería de requerimientos se realiza incorrectamente afecta de manera crítica el desarrollo de software (Aristizábal-Mejía and Torres-Moreno, 2009). Esto se debe principalmente a que las actividades están relacionadas de forma cercana con los procesos de diseño, implementación, pruebas, administración, mantenimiento, gerencia del proyecto y la calidad de software (Aristizábal-Mejía and Torres-Moreno, 2009).

En (Martin, 2017) se describen ventajas durante el desarrollo del software, cuando se implementa de manera correcta. Algunas situaciones favorables al desarrollar software correcto son: que no se requiere de grupos numerosos de programadores que dediquen jornadas largas de tiempo (24 horas los 7 días de la semana) en la implementación y mantenimiento; no se requieren grandes sistemas dedicados al seguimiento de errores en los entornos de producción; no se requieren jordandas largas en la ingeniería de requerimientos ni en documentos extensos de requerimientos (es recomendable en cada ciclo de desarrollo tener una etapa de análisis de requerimientos). Desarrollar software correcto, reduce considerablemente la cantidad de esfuerzos económicos, tiempo y recursos humanos en su implementación y mantenimiento.

2.3.2 *Requerimientos de software*

Los requerimientos especifican qué funcionalidades debe proporcionar el software, las características y propiedades esenciales para el correcto funcionamiento, así como las deseables para que el sistema tenga valor y utilidad para la parte interesada. Los requerimientos indican el qué del sistema, pero no el cómo se va a implementar, es en la fase de diseño donde se plantea el cómo se conseguirá qué el software desempeñe las tareas solicitadas (Gómez, 2011)(IIBA, 2015).

Los requerimientos del sistema se dividen en: requerimientos funcionales que son aquellos que proporcionan la especificación de cómo debe comportarse el sistema ante las entradas determinadas; y los requerimientos no funcionales los cuales se refieren a las restricciones de la funcionalidad brindada por el sistema, estas pueden darse por cuestiones relacionadas con el tiempo, proceso de desarrollo, estándares definidos, políticas y normas institucionales.

En la Tabla 3 se muestran los diferentes tipos de requerimientos de software (Gómez, 2011)(IIBA, 2015):

Tabla 3: *Técnicas de levantamiento de requerimientos de software*

Técnica	Descripción
Objetivo	Un estado u objetivo específico cuantificable, que la empresa pretende alcanzar con el sistema
Objetivo comercial	Estado o meta de **la empresa** que pretende conseguir o mantener con el sistema
Objetivo del sistema	Estado u objetivo que **el usuario** pretende conseguir al utilizar el sistema
Restricción de capacidad	Es una restricción de cómo el sistema logra su objetivo
Restricción de calidad	Es una restricción de calidad en el comportamiento del sistema

Tabla 3 – *Continua en la siguiente página*

Técnica	Descripción
Política comercial del negocio	Es una directiva de la empresa que define que es lo que se puede hacer y que no se debe hacer
Regla de negocio	Es una directiva de la empresa que proporciona una recomendación específica para implementar una política comercial.

2.3.3 *Levantamiento de requerimientos*

El proceso de levantamiento de requerimientos es considerada la actividad que requiere de mayores conocimientos en el proceso de ingeniería de requerimientos (Aristizábal-Mejía and Torres-Moreno, 2009)(Zowghi and Coulin, 2005). Se define como el proceso en el que se buscan, definen, descubren y comprenden los requerimientos y necesidades de los interesados en el software(Hickey and Davis, 2002)(Yousuf and Asger, 2015)(Zowghi and Coulin, 2005). Comúnmente estás actividades son abordadas desde una única y temprana etapa en el proceso de desarrollo de software, esta práctica se percibe principalmente en las metodologías tradicionales como por ejemplo en el modelo en cascada. Para el éxito del software el levantamiento y entendimiento de los requerimientos se deben continuar durante todo su desarrollo (Hickey and Davis, 2002)(Zowghi and Coulin, 2005).

Los requerimientos del software se pueden obtener de muchas fuentes de información como pueden ser: documentos, manuales, formatos de la organización, miembros de las partes interesadas, usuarios que tienen el problema, sistemas existentes, por mencionar algunas fuentes (Zowghi and Coulin, 2005).

Los requerimientos de alta calidad sobre el software se obtienen seleccionando a las personas más adecuadas y participativas del grupo de interesados. En el proceso se llevaba acabo la interacción entre ambas partes para obtener las necesidades reales. Los analistas apoyan a los usuarios a expresar sus necesidades y expectativas respecto al sistema para llegar a acuerdos en lo que se va a diseñar e implementar (Yousuf and Asger, 2015). Este proceso es complejo, debido a que los analistas buscan, interpretan y descubren lo que las partes interesadas solicitan del sistema.

Durante el proceso de encontrar el origen de los requerimientos y recolectarlos, el analista se puede encontrar con algunas de las siguientes dificultades (Aristizábal-Mejía and Torres-Moreno, 2009):

- Problema de alcance: Se refiere a que el problema analizado es tan complejo que no se tiene claridad de lo que el sistema debe hacer.
- Problema de entendimiento: se refiere a que los interesados no son capaces de expresar los requerimientos de la forma en la que los analistas lo desean.
- Problema de volatilidad: Se refiere a que si el proyecto de software se toma mucho tiempo los requerimientos pueden ir cambiando o dejan de ser los mismos.

Las tareas de levantamiento de requerimientos son ejecutadas de manera diferente por cada equipo de desarrollo y dependen de los conocimientos y experiencia de los analistas involucrados, así como del tipo de metodología o método de desarrollo de software empleado. Con forme esta disciplina ha avazando los expertos en estás áreas se han apoyado de técnicas y herramientas para facilitar la toma de requerimientos. Las técnicas de levantamiento de requerimientos permiten realizar el proceso de manera organizada y metodológica (Aristizábal-Mejía and Torres-Moreno, 2009)(Hickey and Davis, 2002), lo que permiten dar una base teórica a los analistas de requerimientos.

Debido a que las actividades de levantamiento de requerimientos tiene como naturaleza la comunicación entre las partes interesadas y los analistas, muchas de las técnicas efectivas no provienen de la ingeniería de software, son tomadas de las ciencias sociales, la teorías organizacionales, dinámicas de grupo, psicología y de la experiencia de los analistas (Zowghi and Coulin, 2005).

El tipo de software y su propósito afectan significativamente la forma en que se lleva a cabo el levantamiento de los requerimientos(Aristizábal-Mejía and Torres-Moreno, 2009). Las técnicas de adquisición utilizadas para una situación particular dependen a menudo de una variedad de factores adicionales, incluidos el tiempo y costo, la disponibilidad de los recursos, el nivel de seguridad requerido y las limitaciones legales (Zowghi and Coulin, 2005).

En la práctica los analistas no obtienen los requerimientos de los usuarios simplemente preguntando. En la literatura (Yousuf and Asger, 2015)(Zowghi and Coulin, 2005) se aborda que no existe una técnica ideal que funcione en todas las situaciones, algunas son adecuadas para situaciones específicas mientras que para otro contexto se emplean otras. Como estrategia algunas técnicas se complementan con otras para compensar las debilidades que puedan tener. El problema, el contexto, el tiempo, costos, disponibilidad de recursos, el tipo de sistema y su propósito afectan significati-

vamente la forma en que se lleva a cabo la obtención y las técnicas a emplear (Zowghi and Coulin, 2005).

El uso de diversas técnicas aunque positivo en el aspecto que se puede llevar al descubrimiento de más requerimientos, y con ello un resultado más efectivo en la obtención; tiene como aspecto negativo que los analistas deben dominar las técnicas para realizarlas adecuadamente, así como poseer habilidades recomendadas.

El software se puede medir por la calidad de los requerimientos debidamente analizados y documentados. Omitir requerimientos importantes en el proceso, conduce a fallas en el proceso de desarrollo y a un software defectuoso o que no cumple con lo solicitado (Yousuf and Asger, 2015). El proceso esta sujeto a un grado alto de error, derivado principalmente a problemas de comunicación, dado que es una actividad que depende en gran medida de las habilidades de comunicación de los analistas y del compromiso de las partes interesadas del sistema (Zowghi and Coulin, 2005) . Un ejemplo común de los problemas de comunicación entre las partes, ocurre cuando los interesados describen una característica (“algo”) mientras que los analistas entienden otra cosa, es decir, los conceptos que están claros de una lado, para el otro se entiende otra cosa.

2.4 ESPECIFICACIÓN DE REQUERIMIENTOS

En la fase de análisis se estudia la viabilidad del sistema, se comprueba que las necesidades del usuario se pueden alcanzar empleando tecnologías actuales y el desarrollo este dentro de un presupuesto acordado. La información brindada por el usuario sobre la funcionalidad que debe de cumplir se traduce en un documento llamado especificación de requerimientos. Una vez obtenido ese documento es necesario validar la consistencia de los requerimientos y que comprenda todas las funcionalidades requeridas por el usuario.

La especificación de requerimientos es un paso posterior al levantamiento y análisis de requerimientos, en el cual se le proporciona al grupo de desarrollo y al cliente, los medios para valorar el cumplimiento de resultados, procedimientos y datos, una vez que se desarrolla el software.

En el análisis de requerimientos se busca que el cliente (grupo de interesados) y grupo de desarrolladores lleguen a un acuerdo sobre lo que el sistema debe hacer. El objetivo es que el analista actúe como un negociador y consultor, y proponga solucio-

nes a las necesidades del cliente. La especificación de requerimientos, es el producto de este análisis y proporciona las pautas a seguir en la fase de diseño (Gómez, 2011). La carencia de una buena recolección de requerimientos afecta al resto de las etapas, pasando por el análisis, especificación y el documento técnico, la fase de diseño y con ello la fase de implementación. Toda esta cadena que conduce a un "fracaso"de proyecto de software comenzo con un mal levantamiento de requerimientos y de su mala comprensión.

2.4.1 *Contenido de la especificación de requerimientos*

Los documentos de especificación de requerimientos deben contemplar los siguientes elementos y características de lo que el sistema a desarrollar debe proporcionar (Gómez, 2011):

- Funcionalidad del software
  - Las tareas y funciones que realiza
  - En que momento las realiza
  - Los tipos de operación que tiene
  - En que punto puede escalarse o modificarse
  - Las restricciones de ejecución: velocidad, tiempo y rendimiento
- Ubicación
  - El lugar del equipo o infraestructura donde se va a ejecutar el software
  - Si son varias ubicaciones que tipo de comunicación emplea
  - Las restricciones ambientales como: humedad, temperatura, interferencia, por mencionar algunas.
- Interfaces
  - Se conecta o comunica con más sistemas
  - Los productos y salidas se envían a otros sistemas
  - Se requieren de algún proceso especial con los datos tanto de entrada, como los de salida.
- Usuarios
  - Los usuarios tipo

  - Las habilidades que tienen los usuarios
  - Las guías y entrenamiento que requieren para emplear el software
  - El grado de facilidad que le pueda resultar el software al usuario
  - Los elementos de diseño que pueden ser memorables al usuario
- Documentación del software
  - La documentación que se requiere: la propio del desarrollo d software, manuales técnicos, de instalación, entre otros.
  - Los formatos en los que debe estar: físicos, originales, copia o en consulta en línea.
  - A que tipo de usuario están orientados
- Información y datos
  - Formato de los datos de entrada y de salida
  - Frecuencia en la que deben ser enviados o recibidos
  - Su grado de exactitud
  - El proceso o cálculo que se requiere para obtenerlos
  - La cantidad de datos que circulan o se generan en el software
  - El tipo de almacenamiento: local, externo, ambos, etc.
  - Tiempo de duración que tiene: momentáneo, cache, meses, años, etc.
- Recursos
  - Recursos materiales que requiere el software
  - Recursos humanos que requiere el software: gestión, mantenimiento operación, entre otros
  - Las habilidades y conocimiento del equipo de desarrollo
  - Espacio físico que ocupa el software y sistemas
  - El consumo eléctrico, calefacción o aire acondicionado
  - Recurso de tiempo: cronograma de entregas
  - Recursos monetario: presupuesto, gastos no documentados que pueden salir
- Seguridad

- Tipo de acceso del software
- Manejos de credenciales
- Espacio personal por usuario: ejemplo función chroot.
- Copias de respaldo y su frecuencia
- Lugar en el que se almacenan las copias de seguridad
- Precauciones contra desastres físicos: fuego, agua, terremotos,
- Precauciones contra desastres provocados por personas: perdida de información, robo, eliminar elementos, contaminación de virus, etc.

- Calidad
  - Descripción de los requerimientos de aceptación de confiabilidad, disponibilidad, seguridad y otros atributos de calidad que debe presentar el software
  - El sistema debe detectar los errores y fallas
  - Tiempo de recuperación antes fallas
  - Tiempo entre fallas
  - El mantenimiento constante evitará los errores
  - La portabilidad del software: es factible cambiar la ubicación del sistema de un equipo a otro

2.4.2 *Características de los requerimientos documentados*

Los requerimientos analizados y documentados explican lo que el grupo de desarrolladores ha entendido de lo que el grupo de interesados en el proyecto pretenden del software, a su vez, el grupo de interesados confirman que los requerimientos documentados son lo que buscan del software. También, indican al equipo de desarrollo qué funcionalidades y características del software resultante. Y al equipo de pruebas y mantenimientos se indican las pruebas que se deben hacer para que el cliente y grupo de interesados acepte la entrega del software.

En (Gómez, 2011)(Engineers and Electronics, 2008) se abordan características que deben tener los requerimientos documentados en la especificación:

- Correctos: Los interesados y los desarrolladores deben revisar que los requerimientos no tengan errores

- Consistentes: El complimiento de un requerimiento no afecta o impide el cumplimiento de otro. Es decir, no debe presentarse contradicciones o conflictos entre ellos.
- Completos: Es completo si incluye todos los requerimientos significativos sobre las funcionalidades, desempeño, restricciones, atributos del sistema. Además es completo si contemplan los estados posibles, entradas, salidas y restricciones, en alguno de los requerimientos. Se deben incluir que entradas son aceptables y cuales no. Especificar las unidades de medida.
- Realista: Los requerimientos deben ser revisados para asegurarse que se pueden llevar acabo
- No ambiguo: Cada requerimientos establecido tiene una única interpretación posible tanto para los interesados como para lo desarrolladores.
- Ordenado: Los requerimientos de identifican y ordenan para indicar su importancia o prioridad.
- Grado de estabilidad: Es el número de cambios que puede esperar un requerimiento, producto de eventos futuros
- Verificable: Es verificable si de alguna forma se puede analizar o probar que el software cumple con lo especificado.
- Modificable: Bajo que condición y proceso se permiten los cambios de forma fácil y consistente.
- Rastreable: Cada requerimiento se puede identificar de que tipo es y su origen, en cualquier parte del proceso de desarrollo

En este punto abordamos complicaciones que se presentan en el levantamiento de requerimientos y procesos que actualmente se realizan para alcanzar documentos de especificación de requerimientos adecuados que conduzcan a productos de software exitosos. A continuación se aborda otro tema central del trabajo que consiste en el software para la salud y aspectos que se encuentran en el proceso de desarrollo de este tipo de software, para identificar el punto en el que se cruzan ambas situaciones y dan sentido al trabajo propuesto.

3

PROPUESTA DEL PROCESO

3.1 CONTEXTO DE LA SITUACIÓN DEL LEVANTAMIENTO DE REQUERIMIENTOS DE SOTFWARE

En la fase de ingeniería de requerimientos se puede identificar que la adquisición de requerimientos incorrectos (inconsistentes, inexactos, poco claros o incompletos) son responsables de malentendidos del sistema, de diseños e implementaciones defectuosas, que pueden derivar en un producto de software de mala calidad que puede incluso ser potencialmente dañino a las personas o al medio ambiente si es que se trata de un sistema crítico orientado a la salud o seguridad de la sociedad.

Con lo anterior identificamos la importancia de que durante el proceso del levantamiento de requerimientos se seleccionen las técnicas adecuadas para el proceso, dado que se propicia la participación activa de los interesados, se tiene una comprensión correcta de los requerimientos de los usuarios, se pone atención en los que hacen los involucrados y se proporcionan las bases para que el desarrollo del software sea exitoso.

Considerando estas tareas críticas de la ingeniería de requerimientos y el contexto complicado que se plantea en el desarrollo de software en el contexto mexicano, la selección de técnicas de levantamiento de requerimientos se convierte en una actividad importante para el éxito del proyecto. En esta investigación se realizó un estudio comparativo de 18 técnicas de levantamiento de requerimientos que se encuentra en el Anexo A, dichas técnicas se recabaron de la literatura, en ella se presenta una breve descripción sobre la mécanica y función de la técnica, las principales ventajas y desventajas de su empleo, así como las refenrencias a las fuentes de la información. El objetivo de esta tabla para el proceso propuesto es que los analistas encargados de las tareas del levantamiento de requerimientos seleccionen las técnicas más adecuadas al proceso.

3.2 PROCESO DE SELECCIÓN PROPUESTO

El proceso de selección de técnicas de levantamiento de requerimientos propuesto en este trabajo, esta formado por diferentes etapas, algunas se encuentran agrupadas por un rectángulo que abarca las tareas que se pueden realizar al mismo tiempo o tareas asíncronas. En la Figura 12 se muestra el proceso para llevar a cabo la selección de técnicas de levamiento de requerimientos de software. La figura es el resultado de datos recopilados de la aplicación del proceso en varios desarrollos de software en años recientes (entre los años del 2013 al 2020), tanto en la iniciativa privada, como en dependencias educativas como en la UNAM y el IPN. En dichos proyectos por su complejidad y tiempo de desarrollo, la etapa de análisis y levantamiento de requerimientos presentaban un alto nivel de importancia en el éxito o fracaso de los proyectos.

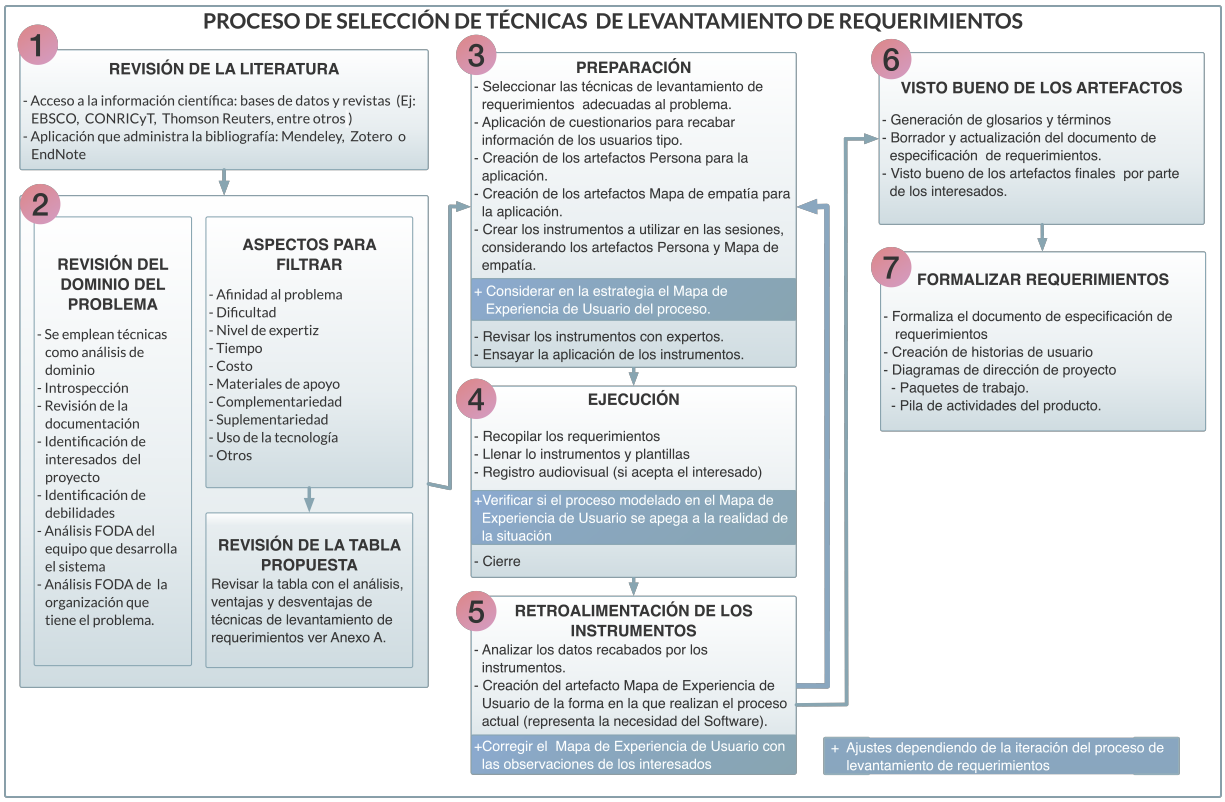


Figura 12: *Proceso de selección de levantamiento de requerimientos [Imagen elaborada por el autor].*

A continuación se describen los pasos que conforman el proceso propuesto de selección de técnicas de levantamiento de requerimientos y la fase de análisis que sigue en el ciclo de vida del desarrollo de software.

3.2.1 *Paso 1: Fase de revisión de literatura*

Metodologías de desarrollo de software consideradas por los analistas como las principales (recurrentes en emplearlas en los proyectos), como el modelo en cascada, modelo en espiral, proceso unificado, entre otras (ver sección 2.1) concuerdan en realizar una etapa de recolección de información sobre el dominio del problema a solucionar, dado que el software que resuelte del proceso debe estar descrito en elementos del dominio del problema. Durante esta etapa las personas encargadas en el levantamiento de requerimientos (normalmente el rol de analista) omiten o realizan de manera somera la revisión bibliográfica del problema a desarrollar, por tal motivo se recomienda como primera etapa recabar información científica y de corte de divulgación para tener un acercamiento conceptual de la problemática a tratar. En (IIBA, 2015) y (Engineers and Electronics, 2008) proponen generar una base de información sobre los diferentes proyectos que desarrolla el equipo con la meta de generar valor a la organización. En (Nonaka, 1994) se describe la importancia de generar conocimiento en la organización, en la cual se debe poner atención a los procesos de creación y valoración del conocimiento en función de la calidad y valor que aportan a la organización y a la sociedad.

En la tarea de recabar documentación bibliográfica se recomienda emplear motores de búsqueda de artículos y revistas de temas relacionados con el dominio del problema como EBSCO, CONRICyT, Thomson Reuters, Google Scholar, entre otros. Para consultar información y buenas prácticas sobre la dirección de proyectos de software se recomienda consultar casos de éxito y de fracaso de proyectos similares o con temas afines, documentados en la página oficial del PMI [1] [2] (Project Management Institute) entre otros recursos relacionados con la gestión de proyectos de software.

3.2.2 *Paso 2: Tareas simultáneas*

La siguiente etapa del proceso de preparación consiste de tareas o sub-etapas que se pueden realizar simultáneamente como: la revisión del dominio del problema; aspectos a filtrar; aspectos cualitativos y cuantitativos; y la revisión de la tabla propuesta de técnicas de levantamiento de requerimientos. A continuación se describen esas tareas.

1 PMI.- Es una organización sin fines de lucro que integra y relaciona a profesionales de la asminsitración de Proyectos. https://www.pmi.org.in/

2 Artículos proporcionados por el PMI.- En la siguiente dirección electrónica se pueden consultar artículos relacionados con la dirección de proyectos. https://americalatina.pmi.org/latam/KnowledgeCenter/Articles.aspx

3.2.2.1 *Revisión del Dominio del Problema*

Se propone una revisión del dominio del problema, empleando las referencias de la literatura y recursos de gestión de proyectos, recabadas por el analista (o grupo de analistas). Con la información se propone generar un modelo mental del problema y sus complicaciones teóricas, para verificar la factibilidad de una solución software y sus posibles alcances. De esta revisión se solicita a los interesados del proyecto información específica del problema, a través de manuales, documentos técnicos, normas nacionales, normas de la organización, video tutoriales, solo por mencionar las principales fuentes de recolección de información del tipo documental. Con esta información el grupo puede formarse una idea del problema a desarrollar y nutrir el banco de conocimientos de la organización. Es importante indicar que en este punto no se ha realizado ningún levantado requerimientos y se debe evitar pensar en algún tipo de implementación, es labor del líder del proyecto dirigir esta etapa para evitar vicios en el equipo de adelantar la implementación sin antes conocer el problema (no confundir esta situación con la técnica de instrospección). En esta etapa el equipo de trabajo se debe familiarizar con el contexto del problema realizando una matriz de los interesados en el proyecto identificados por su rol y un peso de impacto tentativo en la organización. Un estudio de FODA de la organización para establecer el punto de partida en la que se encuentra. Un estudio FODA del equipo de trabajo que implementa el proyecto (para identificar las debilidades, fortalezas y oportunidades del equipo y determinar de los miembros del equipo para afrontar el reto).

3.2.2.2 *Técnicas de levantamiento de requerimientos*

En este paso se propone identificar, analizar y elegir las técnicas de levantamiento de requerimientos más adecuadas para la solución del problema.

ASPECTOS PARA FILTAR: Generar una lista de palabras clave relacionadas con el problema a trabajar. La búsqueda de artículos de investigación, reportes técnicos y casos prácticos de desarrollo de software se realizan mediante las palabras clave recabadas por los analistas. La información documental debe estar dirigida a cuestiones del levantamiento de requerimientos de software en casos anólogos al problema a tratar.

ASPECTOS CUALITATIVOS Y CUANTITATIVOS: Del análisis de los articulos se debe filtrar la información considerando las preguntas: ¿Cuáles son las técnicas de levantamiento de requerimientos empleadas en los documentos de la bibliografía?, ¿Qué técnicas son las más adecuadas al problema?,¿Cuáles de las técnicas

analizadas se pueden sustituir o complementar?,¿Qué nivel de dificultad tienen las técnicas empleadas en los documentos?, ¿Qué instrumentos o artefactos emplean en dichos casos?, ¿Qué dificultad presentan las técnicas empleadas?.

REVISIÓN DE LA TABLA PROPUESTA: Se realiza la búsqueda de las técnicas de levantamiento de requerimientos adecuadas al contexto, a los perfiles y ubicación de los interesados en el proyecto. Como aporte al proceso además de emplear la información de los puntos anteriores, se presenta una tabla resumen de 18 técnicas de levantamiento de requerimientos con su descripción, ventajas y desventajas de emplearlas, así como sus referencias bibliográficas.

Este análisis permite apoyar al analista en identificar las técnicas de levantamiento de requerimientos que puede emplear en la primeras etapas del desarrollo de software, desde la fase de análisis y levantamiento de requerimientos, que como se describe en la sección 2.3.1 es un proceso crítico en el éxito o fracaso del sofware.

3.2.3 *Paso 3: Fase de preparación*

Se levantan encuestas físicas o por medios electrónicos a los usuarios tipo, para crear los artefactos Persona (ver sección 3.4.2) y el Mapa de Empatía (ver sección 3.4.3) con el objetivo de conocer de mejor forma a los usuarios de la aplicación de software.

Se contruyen los instrumentos que se aplican en las sesiones de levantamiento de requerimientos, ejemplo de ellos son los cuestionarios, entrevistas semi-estructuradas, tarjetas para un taller de requerimientos, maquetaciones y prototipos, solo por mencionar algunos. Se planifican y agendan las sesiones con las partes interesadas, se recomenda ensayar la ejecución de las técnicas con el grupo de trabajo, para anticipar anomalías o flujos alternos en el momento de aplicar los instrumentos con los interesados. Estos escenarios de prueba se pueden llevar a cabo considerando las ventajas y desventajas de las técnicas seleccionadas (en el anexo A se muestran algunas). Las entrevistas o cuestionarios a los usuarios tipo se pueden realizar utilizando herramientas Web como "Google Forms"[3] y "SurveyMonkey"[4]. En este proceso se recomienda emplear la herramienta de Google dado que permite exportar a hojas de cálculo o for-

3 Google Forms: Herramienta Web que permite recopilar información de los usuarios a través de encuestas personalizadas. Los datos se recopilan y se vacian automáticamente a una hoja de cálculo para su procesamiento. https://docs.google.com/forms/u/0/

4 SurveyMonkey: Herramienta Web que permite crear encuestas de simples a sofisticadas en línea, con el objetivo de recopilar información, medir y tomar decisiones. https://es.surveymonkey.com/

mato CSV que se pueden cargar en herramientas como CRAN R[5] o en ejecutables de Python para realizar análisis de la información recabada.

De tratarse de una iteración en la cual se cuenta con un artefacto Mapa de experiencia del usuario, se debe considerar la retroalimentación de los aspectos del proceso modelado en el mapa, con el fin de actualizar los instrumentos que a emplear en la próxima sesión de levantamiento de requerimientos. Como se describe en la sección 3.4, consultar y actualizar los artefactos empleados en la propuesta al final de cada iteración, permite llegar a una mejor comprensión de los usuarios que interactuan con el sistema y un mejor entendimiento del proceso que se desea automatizar con el sistema a desarrollar.

Los instrumentos son revisados con expertos en el tema, de no contar con esta opción, se pueden utilizar plantillas de los instrumentos y tomar como guía artículos y casos prácticos de proyectos análogos al que se está desarrollando. Se recomienda medir los resultados de los instrumentos para propiciar un estado de mejora continua.

Antes de aplicar los instrumentos es recomendable realizar una prueba de la aplicación de los instrumentos en el levantamiento de requerimientos con el objetivo de anticiar posibles flujos alternativos y errores en el proceso. Una de las situaciones principales que se presenta en el proceso es el tiempo que demora y que durante el ensayo se pueden ajustar y proponer tiempos estimados dependiendo de los escenarios.

3.2.4 *Paso 4: Fase de ejecución*

En esta fase se ejecuta el levantamiento de requerimientos con los instrumentos e interesados en el proyecto correspondiente al escenario planificado previamente. Se anotan los requerimientos de los interesados llenando los instrumentos conforme a lo ensayado. En este proceso se utilizan documentos en la nube para sincronizar con diferentes dispositivos y colaborar con miembros del equipo. Dentro de las nubes recomendadas en el proceso se utilizan “Google Drive”[6] y “One Drive de Microsoft”[7]. Como

5 CRAN R: R es un entorno de software libre para estadística y gráficos por computadora. https://cran.r-project.org/

6 Google Drive: es un servicio de almacenamiento y sincronización de archivos en la nube desarrollado por Google. https://www.google.com/drive/

7 OneDrive: es un servicio de alojamiento de archivos y servicio de sincronización en la nube operado por Microsoft como parte de su versión Web de Office https://onedrive.live.com/about/en-US/

extra se recomienda emplear la herramienta “Evernote”[8] durante la sesión para tomar notas de texto, audio, capturas de pantalla de la reunión, así como enlaces a páginas de Internet y observaciones de los puntos tratados.

Registrar las actividades en video y/o audio apoya a los analistas en tener un medio para revisar ambiguedades y dudas que surjan después del levantamiento de requerimientos. En algunas sesiones los interesados en el sistema no autorizan la grabación del proceso por tal motivo se debe solicitar el permiso con antelación .

De tratarse de una iteración que cuenta con un mapa de experiencia del cliente que modela el proceso a automatizar, se debe verificar que lo entendido por los analistas a partir de sesiones anteriores y plasmado en el mapa, coincida con lo que los usuarios realizan cotidianamente y con lo solicitado a implementar. De coincidir lo modelado en el diagrama por los analistas con los descrito en sesiones anteriores por los interesados se acepta el proceso y se solicita una confirmación o visto bueno del proceso modelado en el mapa. En el mapa se deben anotar las observaciones que surjan durante la revisión. Cuando lo plasmado en el mapa no coincide con lo que describen los clientes e interesados, se deben anotar los cambios y versiones para llegar a un acuerdo del proceso en una próxima reunión.

La reunión de levantamiento de requerimientos debe tener un cierre, para lo cual se anota en una minuta el tiempo transcurrido, las versiones, los asistentes y acuerdos alcanzados. La minuta se puede compartir de manera física en un documento o través de herramientas como una hoja de texto en la nube o una entrada de evernote. Este tipo de tareas propicia la mejora continua en los procesos del equipo.

3.2.5 *Paso 5: Retroalimentación de los instrumentos*

En este paso se revisa y corrige la redacción de la información obtenida en la reunión con los interesados. La información recabada es analizada para planificar una estrategia ante futuras revisiones con el cliente, y de aprovechar el tiempo con los interesados de requerirse otras sesiones.

De tratarse de una etapa donde no se cuenta con un mapa de experiencia del cliente del proceso, pero se tiene información de anteriores sesiones o entrevistas con los clientes e interesados, se recomienda crear el mapa que modele el proceso entendido hasta el momento. En este punto se debe tomar en cuenta que se trata de primeras

8 Everntote: aplicaciones para crear notas de las reuniones, páginas web, proyectos y listas de cosas por hacer. https://evernote.com/intl/es-latam/

aproximaciones del mapa que describe el proceso en las reuniones. Para obtener una versión cada vez más cercana a la realidad se debe contrastar y revisar directamente con los encargador que realizan la operación, evitando en lo posible versiones del proceso de intermediarios.

En una etapa en la cual se cuenta con un mapa de experiencia del cliente del proceso se deben revisar las observaciones y anotaciones que la persona encargada o interesada proporcionó en la sesión de levantamiento de requerimientos. Se recomienda llevar un control de versiones de las modificaciones solicitadas, para medir aspectos que apunten a una mejora continua del proceso. En algunas organizaciones el versionado y almacenaje se estipula desde un inicio para que todos los involucrados en el desarrollo mantengan los estándares y buenas prácticas.

3.2.6 *Paso 6: Visto bueno de los artefactos*

Durante esta etapa se recopilan los términos y abreviaturas del dominio del problema verificados con los interesados durante las sesiones de levantamiento de requerimientos y revisión de avances. Se recomienda en el documento director del proyecto anexar los términos y abreviaturas utilizados en el entorno del problema a resolver. En el Kanban del proyecto se puede anexar en la columna de “Información General” un enlace con el glosario o documentación completa del proyecto. Utilizando GitBook[9] el equipo puede crear una documentación técnica completa de la implementación del proyecto que se esta desarrollando.

Se mantiene actualizado el documento director del proyecto y el documento de especificación de requerimientos según las observaciones obtenidas en las sesiones con los interesados, en cada iteración del proceso.

La etapa concluye solicitando una firma y las últimas observaciones de los artefactos propuestos por el proceso. Se recomienda dejar un espacio en los artefactos dedicado al visto bueno mediante firma o leyenda por parte de los interesados en el proyecto.

Los artefactos validados con visto bueno se colocan en la columna de “Información General” para que los involucrados en el desarrollo tengan acceso y revisen siempre que lo requieran, en cualquier etapa del proceso.

9 GitBook: es una plataforma de documentación donde los equipos pueden documentar todo acerca de los productos de software. https://docs.gitbook.com/

3.2.7 *Paso 7: Formalizar los requerimientos*

Se formalizan los requerimientos recabados en las direfentes sesiones de levantamiento, mediante el visto bueno del documento de especificación de requerimientos. Como recomendación en los procesos de desarrollo de software los requerimientos aprobados y verificados por los interesados y dueños del proyecto se deben registar en un documento (ver sección 2.4) para formalizar el proceso de ingeniería de requerimientos.

El visto bueno consite en que el proceso descrito en el documento y en los artefactos coincide con lo que los interesados esperan del sistema solicidado. El documento formalizado se anexa al documento director del proyecto y a la columna Kanban de "información general", para futuras consultas del equipo de trabajo.

De los requerimientos documentados se crean las historias de usuario, una historia de usuario puede abarcar varios requerimientos. Las historias de usuario se colocan en la columna de "Pila del producto" (o Product Log) del Kanban del proyecto, dada la prioridad y alcance del proyecto. Se recomienda establecer la pila de producto del Sprint inicial del proceso iterativo. De emplear Scrum en cada iiteración se debe establecer la pila de producto del Sprint.

Los diagramas de dirección de proyectos son creados y formalizados en esta etapa, por ejemplo es el paquete de trabajo, matriz de trazabilidad, casos de pruebas de integración, solo por mencionar algunos.

3.3 INTEGRACIÓN DEL PROCESO CON SCRUM

El proceso de selección de técnicas de levantamiento de requerimientos y fases iniciales del desarrollo de software propuesto en este trabajo y descrito en la sección 3.2, se recomienda complementar con el proceso de Scrum.

En la Figura 13 se muestra la intregración del proceso propuesto a una variante de Scrum empleada en este trabajo. En la sección 2.2.3.2 se aborda el proceso de Scrum y en la sección 2.2.4 se describen ventajas y desventajas de su empleo, esos factores son considerados para utilizar Scrum en la propuesta del trabajo. En el proceso se crean la pila del producto y la pila del Sprint actual, ambas se despliegn en un tablero Kanban para su seguimiento. En cada iteración se pasa por una etapa de planificación,

diseño, implementación, pruebas y entregable del producto. En el cierre de la iteración se cuenta con una propueta de valor ante los clientes o interesados en el proyecto.

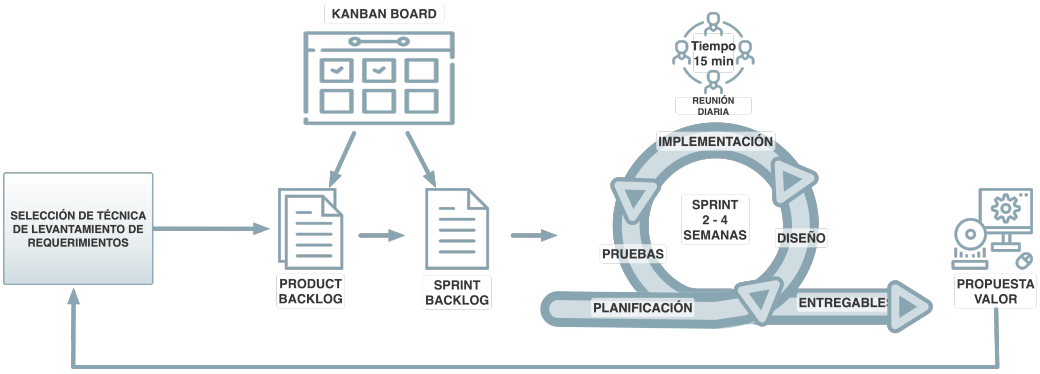


Figura 13: *Ejecución del proceso de desarrollo de software [Imagen elaborada por el autor].*

3.4 ARTEFACTOS EMPLEADOS EN LA PROPUESTA

En el desarrollo de aplicaciones de software, la investigación actual emplea artefactos o instrumentos de análisis y diseño. En la Figura 14 se muestran los artefactos de la propuesta de invistigación, los cuales han presentado cambios y ajustes conforme a las lecciones aprendidas en la aplicación en proyectos reales. Se recomienda que dichos instrumentos se ajusten o se apliquen de manera flexible a los requerimientos y características de cada proyecto.

El ”buen” diseño se consigue a partir de una comprensión profunda de las personas para las que se diseña (usuarios tipo). En esta investigación se emplean los artefactos Persona, mapa de empatía y el ”costumer journey” como técnicas para que los desarrolladores y diseñadores generen empatía con los usuarios tipo.

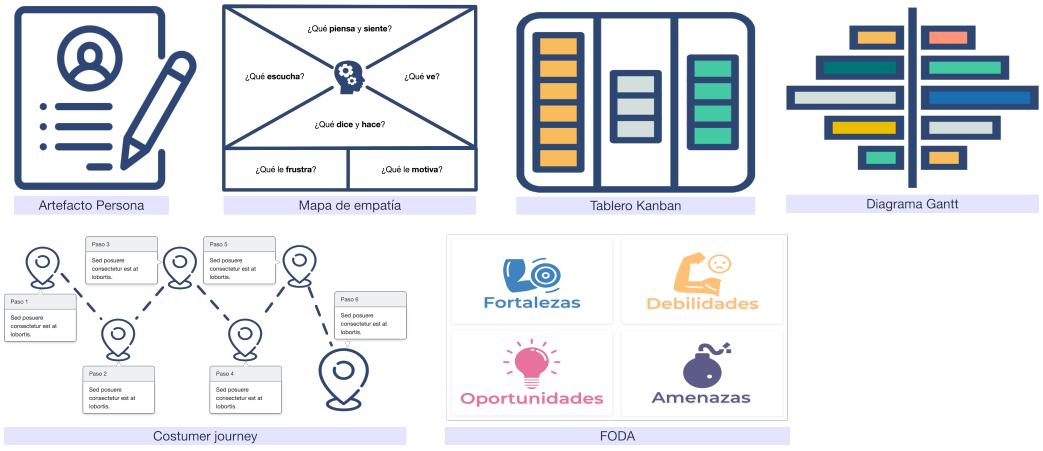


Figura 14: *Artefactos empleados en el proceso propuesto [Imagen elaborada por el autor].*

3.4.1 *Artefacto FODA*

El artefacto FODA proviene del acrónimo en Inglés SWOT (Strengths, Weaknesses, Opportunities and Threats), y que en español se emplean las siglas FODA (Fortalezas, Oportunidades, Debilidades y Amenazas). El análisis FODA consiste en realizar una evaluación de los factores fuertes y débiles que en su conjunto aportan un diagnóstico interno de una organización, y una evaluación externa consideradas por las oportunidades y amenazas. Es una herramienta que se puede considerar sencilla de emplear y permite obtener una perspectiva general de la situación estratégica de una organización determinada. La matriz FODA permite generar una estrategia que logre un equilibrio entre la capacidad interna de la organización y su situación externo para competir (Ponce, 2006).

En la Figura 15 se muestra un ejemplo de la matriz FODA empleada en este trabajo. Cuenta con 5 secciones, que se describen a continuación.

NÚMERO 1: Cabecera con el nombre del proyecto y si es necesario algún control de versiones o identificar si es del personal de la organización o del equipo que va a implementar el software.

NÚMERO 2: Se describen las **Fortalezas** que tiene la organización o equipo de trabajo. Una fortaleza de la organización o del equipo es una función que puede realizar de manera correcta, como ciertas habilidades y capacidades del personal con atributos psicológicos y que cumplen con evidencia en sus competencias. Otra fortaleza son los recursos considerados valiosos con los que cuenta y su capacidad competitiva de la organización, por ejemplo servicios que brinda la organización en aspectos sociales.

NÚMERO 3: Se anotan las **Debilidades** que tiene la organizaicón o equipo de trabajo. En la organización se puede definir como un factor considerado vulnerable en cuanto una actividad que la empresa realiza en forma deficiente o con resultados de calidad deficiente.

NÚMERO 4: Se listan las Oportunidades que tiene la organización o equipo de trabajo. Las oportunidades constituyen los factores de fuerzas ambientales de carácter externo, que no controla la organización, pero representan elementos potenciales de crecimiento o mejora. La oportunidad es un factor de gran importancia para generar las estrategias de las organizaciones.

NÚMERO 5: Se describen las **Amenazas** que tiene la organización o equipo de trabajo. Las amenazas representan la suma de las fuerzas ambientales no controlables

por la organización, pero representan aspectos negativos y problemas potenciales a resolver.

Figura 15: *Artefacto FODA empleado en el proceso propuesto [Imagen elaborada por el autor].*

La matriz FODA es un artefacto para realizar análisis organizacional en relación con los factores que determinan el éxito al cumplir las metas, por este motivo es considerado en este trabajo como parte del proceso de desarrollo de software. Lo importante de este análisis es evaluar las fortalezas y debilidades, las oportunidades y las amenazas que pueden influir en las actividad de la organización; ya que establecen la necesidad de emprender acciones de carácter estratégico y llegar a conclusiones.

3.4.2 *Artefacto Persona*

El artefacto Persona consigue proporcionar información sobre quién es la audiencia y cuáles son sus expectativas y comportamientos. Es un instrumento ampliamente em-

pleado por los diseñadores de experiencia de usuario para desarrollar productos usables y exitosos. Este instrumento permite realizar una verificación en diferentes etapas del proyecto, desde los requerimientos de software, requerimientos comerciales, diseño gráfico, diseño de interfaces, control de calidad, solo por mencionar los principales (Chandler and Carolyn, 2012)(Getto and Cao, 2016).

Las personas son perfiles de usuarios arquetipo, es decir, usuarios reales que son representativos de un grupo demográfico clave de la investigación de los usuarios de la aplicación (Getto and Cao, 2016). Estos artefactos modelan las motivaciones e interacciones de los usuarios con respecto al sistema. Son útiles para presentar información del usuario al equipo de trabajo, interesados en el proyecto y a los clientes. Al realizar una investigación y descripción de usuarios apropiada, las "personas"que se obtienen esbozan una imagen clara en la mente de los involucrados en el desarrollo, sobre quiénes y cómo utilizarán la aplicación (Chandler and Carolyn, 2012).

Los artefactos personas cuando están bien diseñados son punto de partida y visualización del diseño de la aplicación. Cuándo el diseñador de experiencia de usuario se encuentra ante una inquetud, puede emplear la "persona", colocarla en un cierto escenario o situación y realizar las preguntas: ¿Cómo el usuario **A** realizaría la tarea **T1**?, ¿Qué haría el usuario **B** en la situación **S**?. En Chandler and Carolyn (2012) se indica que este tipo de estrategia no es tan precisa para probar la funcionalidad y el diseño como ocurre con los usuarios reales, pero permite avanzar en la implementación del proyecto hasta obtener un prototipo funcional viable para realizar pruebas más extensas.

En la Figura 16 es un ejemplo de la plantilla del artefacto Persona que se utiliza en la propuesta de este trabajo. Es una plantilla que se ha modificado conforme a las lecciones aprendidas al aplicar el instrumento en varios proyectos de la vida real.

A continuación se describen los apartados que conforman la plantilla del artefacto ”Persona” de la Figura 16, puntualizando que es una propuesta que se ha venido conformando con la experiencia de participar en varios proyectos de ingeniería de software y en la impartición de cursos de diseño de interfaces.

NÚMERO 1: Al crear una Persona se coloca un nombre completo (ficticio), así como el número o identificador de la persona en cuestión. Colocarle nombre a la Persona permite a los desarrolladores y diseñadores empatizar con los posibles usuarios de la aplicación. En la práctica algunos diseñadores optan por utilizar nombres tradicionales del país y/o región de los usuarios tipo de la aplicación, en algunas

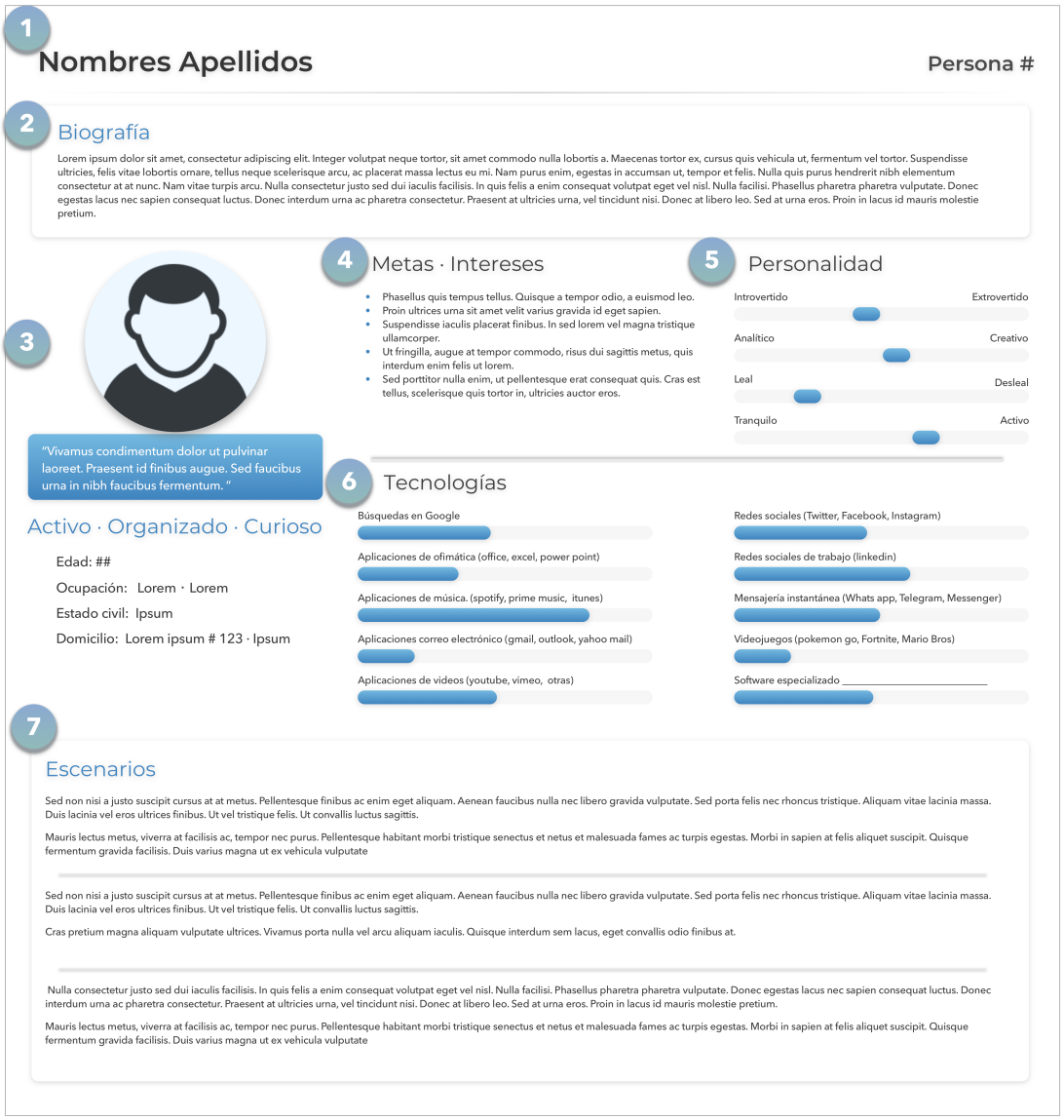


Figura 16: *Plantilla del artefacto Persona propuesto en la investigación [Imagen elaborada por el autor].*

ocasiones se emplean alias como ”John Doe” [10] o ”Juan Peréz” [11], no es necesario seguir con esas tendencia, se pueden emplear cualesquiera nombres que permitan representar a la población tipo.

10 John Doe: Alias usado para referirse a alguien desconocido del sexo masculino en desarrollos de software en Norte América.

11 Juan Pérez: Alias usado para referirse a alguien desconocido de sexo masculino en desarrollos de software en México.

NÚMERO 2: Se describe una breve biografía de la Persona, para que el grupo de diseño y/o desarrollo pueda conocer mejor al tipo de usuario para el cual se están implementando las funcionalidades y empatice de manera más efectiva con la población tipo.

NÚMERO 3: En este apartado se coloca una fotografía que asocia la Persona con nombre (Número 1) y con la biografía (Número 2) consolidando un perfil de usuario [12]. Se coloca una breve frase representativa del usuario y tres cualidades que mejor representen a la persona en cuestión. Está sección también considera la presentar la edad, ocupación, estado civil, domicilio y en algunos proyecto incluso es recomendable indicar si pertenecen a alguna religión o culto.

NÚMERO 4: En el artefacto Persona se deben considerar los intereses y metas que tienen los usuarios antes y después del software que soluciona o facilita las actividades en las que se presenta la problemática, para tener un mejor panronama se emplean los escenarios (Número 7).

NÚMERO 5: En este apartado se indican aspectos de la personalidad de la Persona, estos datos se consideran durante el diseño de la interfaz gráfica del usuario y en el diseño de la interacción de la aplicación (la forma en la que el usuario emplea la aplicación).

NÚMERO 6: Al tratarse de un desarrollo que emplea la tecnología se requeria obtener información de las aplicaciones de software que emplea el usuario, con esta información los diseñadores de forman una imagen mental de los widgets[13] y mecanismos de interacción que emplean habitualmente los usuarios por ejemplo gestos, girar el dispositivo móvil, rotar, solo por mencionar algunos.

NÚMERO 7: Los escenarios abordan la situación o historias de usuario, que permiten empatizar mejor el entendimiento del usuario y lo que puede sentir cuando realiza una actividad de la vida cotidiana a solucionar con el sistema (Getto and Cao, 2016).

12 Observación: En el artefacto se recomienda colocar una fotografía real, en este documento se substituyen por una avatar simple que hace alusión a un ser humano.

13 Widget: Es un elemento gráfico que conforma la Interfaz Gráfica de Usuario, por ejemplo los botones, menú, galería, solo por mencionar algunas.

3.4.3 *Artefacto Mapa de Empatía*

El artefacto Mapa de Empatía en el diseño de experiencia de usuario es una herramienta utilizada para articular y establecer lo que el grupo de diseño y desarrollo conoce sobre un tipo particular de usuario. El mapa comparte el conocimiento para que todos los miembros involucrados del proyecto tengan una comprensión de los usuarios y como apoyo en la toma de decisiones (Gibbons, 2017). Un correcto mapa de empatía sintetiza las observaciones sobre los usuarios y extrae las ideas o acciones inesperadas. Por ejemplo el mapa de empatia sirve de apoyo para determinar el tipo de redacciones que se van a emplear en los contenidos de una aplicación (Gibbons, 2017)(Gibbons, 2018a).

El mapa muestra la perspectiva del usuario tipo respecto a las tareas y actividades relacionadas con el producto de software a implementar. La información plasmada no es cronológica y ni tiene una secuencia específica. En el diseño del software se debe crear un mapa de empatía por cada usuario tipo (Gibbons, 2017).

El empleo del instrumento forza la alineación del equipo y compromiso sobre un tipo de usario. El proceso conmienza con cualquier proceso de diseño y la información se va recabando mediante investigación y entreistas a los usuarios tipo para formar un perfil.

La propuesta de esta investigación es complementar la información del mapa de empatía, con el artefacto Persona (ver sección 3.4.2) para tener un panorama más completo de los tipos de usuarios de la vida real que emplearían la aplicación a desarrollar.

En este artefacto se recomienda considerar las necesidades como verbos (actividades y deseos del usuario) y no sustantivos (las soluciones). Las necesidades se obtienen directamente de los rasgos del usuario, o de las contradicciones que se presentan entre lo que dice y lo que hace. De este tipo de contradicciones se pueden obtener comportamientos reales de los usuarios ante las actividades a resolver con el sistema.

En el diseño las ideas a menudo surgen de contradicciones entre las frases que se encuentran en los cuadrantes del mapa o al preguntarse el **¿Por qué?** cuando se observa un comportamiento extraño. Se recomienda escribir ideas potenciales al lado del mapa de empatía usando tarjetas bibliográficas o notas adhesivas. Las ideas se pueden crear a partir de palabras claves que se identifican en las tensiones y contradicciones mientras se realiza el artefacto.

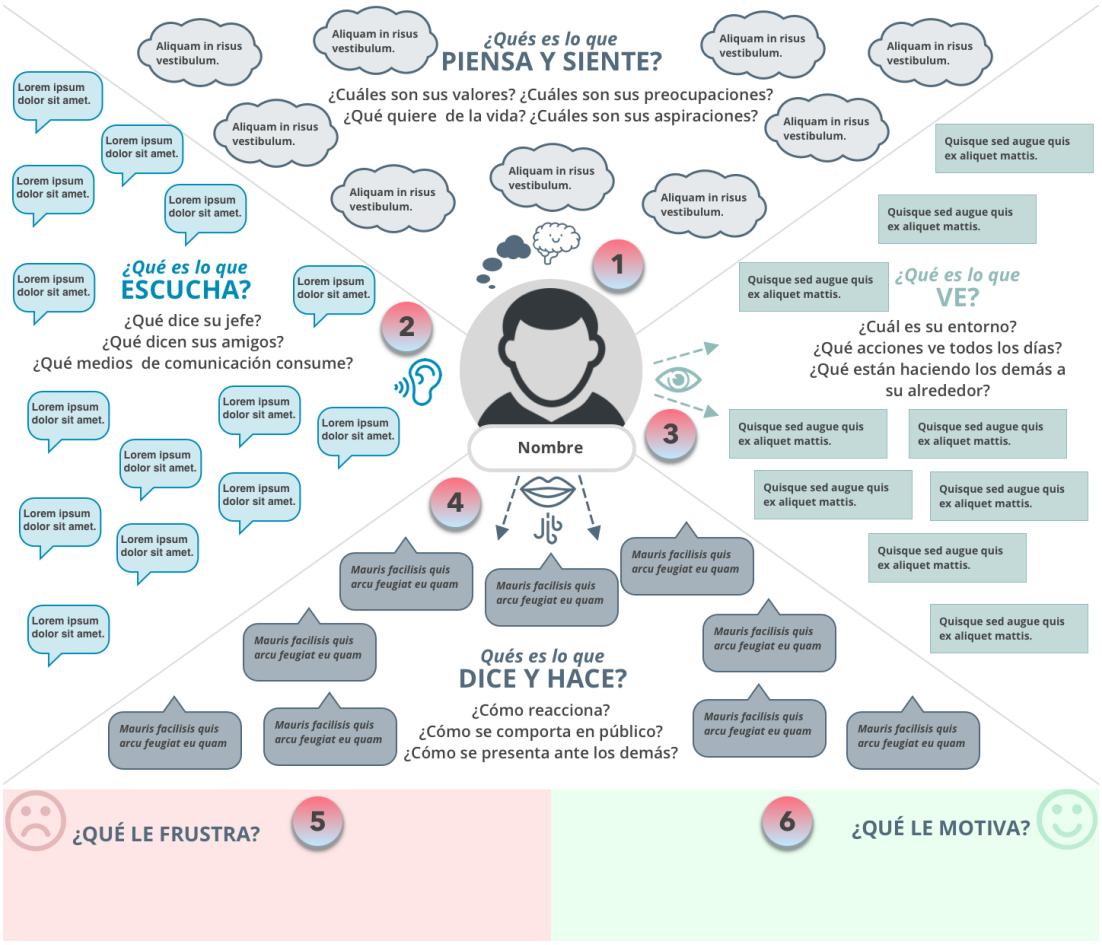


Figura 17: *Plantilla del artefacto Mapa de empatía propuesto en la investigación [Imagen elaborada por el autor].*

A continuación se describen brevemente los apartados que conforman el mapa de empatía de la Figura 17 de la propuesta del trabajo.

NÚMERO 1: Lo que **Piensa** y **siente** el usuario, en esta seción se colocan las frases de lo que el usuario siente respecto a las situaciones que enfrenta diariamente, así como los pensamientos más recurrentes ante esos escenarios. Esta información se obtiene preguntando a un conjunto de personas tipo las siguientes preguntas:

- ¿Cuáles son sus preocupaciones?
- ¿Cuáles son sus metas a mediano y largo plazo?
- ¿Cuáles son sus aspiraciones?
- ¿Cuáles es su plan o estrategia ante la situación?

NÚMERO 2: Lo que **Escucha** el usuario, en este apartado se colocan las frases que normalmente escuchan de amigos, familiares, compañeros de trabajo, noticias o redes sociales sobre las situaciones en las que lidia diariamente. Esta información es recabada de los usuarios tipo realizando las siguientes preguntas:

- ¿Qué dicen sus familiares?
- ¿Qué dicen sus amigos?
- ¿Qué dicen sus compañeros de trabajo?
- ¿Qué dice su jefe?
- ¿Qué dicen al respecto en los medios de comunicación que comsume?

NÚMERO 3: Lo que **Ve** el usuario, en esta sección se colocan las frases que describen lo que normalmente ve el usuario tipo, los comportamientos de otras personas que están en la misma situación, las acciones y medidas que toman responsables que pueden llegar a efectar directa o indirectamente al usuario, entre otros factores que se considere importantes. Esta información es recabada de los usuarios tipo realizando las siguientes preguntas:

- ¿Cuál es su entorno?
- ¿Qué acciones ve todos los días?
- ¿Qué están haciendo los demás a su alrededor?

NÚMERO 4: Lo que **Dice** y **Hace** el usuario, en este apartado se colocan las frases que describen las acciones y actividades que realiza el usuario incluyendo si sigue o no un plan de acción ante las situaciones a las que se enfrenta. Se colocan las acciones principales de los planes de contingencia y de no tenerlos las consecuencias que se pueden llegar a presentar. Esta información es recabada de los usuarios tipo realizando las siguientes preguntas:

- ¿Cómo reacciona?
- ¿Cómo se comporta en público?
- ¿Cómo se presenta ante los demás?
- ¿Lo qué piensa y hace es consecuente?
- ¿Qué hace cuando las cosas no salen como esperaba?

NÚMERO 5: Lo que **Frustra** a los usuarios, en esta sección se describen los temores e inquietudes que puede sentir el usuarios cuando se presenta una situación que el sistema intenta solucionar.

NÚMERO 6: Lo que **Motiva** a los usuarios, en esta sección se describen lo que impulsa y estimula a los usuarios ante los escenarios que se enfrentan, por ejemplo cumplir metas, completar ciclos, adquirir conocimientos, adquirir bienes, solo por mencionar algunos.

3.4.4 *Artefacto Mapa de Experiencia del Cliente*

Las tareas cotidianas o actividades que desempeña un usuario se pueden analizar por etapas, en las que cada una de ellas tiene las siguientes características: entradas de datos o de documentos que emplean para su realización; una serie de procedimientos que hacen uso de la información y documentos; y una serie de salidas que pueden ser empleadas en otras etapas o que devuelven un resultado. Estas actividades se pueden mapear y dar un detalle en cada sección mediante un mapa de experiencia del cliente.

El mapa permite comprender y abordar las necesidades del cliente y los puntos débiles en los procesos. En la Figura 18 se muestra una versión básica de la implementación del mapa, se comienza al colocar una serie de objetivos y acciones del usuario en una línea del tiempo (Gibbons, 2018a)(Gibbons, 2018b). A continuación, en el esquema se registran los pensamientos y emociones del usuario conforme avanza en los pasos del proceso, con ello se pueden identificar puntos críticos del proceso o de mayor grado de estrés. La narrativa se condensa visualmente mediante el listado de las acciones principales de cada paso, junto a sus puntos clave. Los puntos clave de cada paso permiten identificar cuestiones que se deben considerar como roles de los usuarios, tipos de entrada, manejo de los datos, entre otras cuestiones que se deben considerar en el proceso de diseño.

En Gibbons (2017) se drecibe una serie de características que deben presentar los mapas de experiencia de cliente, dentro de los que se encuentran: El mapa se vincula a un producto o servicio específico que emplea un usuario en su actividad cotidiana; Se identifican 4 divisiones de narración: etapas del proceso, acciones, pensamientos y emociones; El mapa debe reflejar la perspectiva desde el usuario considerando lo que piensa y siente aunque se deje fuera de la descripción un mayor detalle del proceso; Se construye de manera cronológica conforme avanza el proceso; El mapa es individual para cada usuario tipo.

El objetivo principal de este mapa es identificar los puntos de contacto del recorrido del cliente en los que presenta frustración o motivación en el desarrollo de una actividad. A partir de ahi la narración permite obtener una comprensión compartida de toda la organización del recorrido del cliente y asignar los puntos clave en el trayecto.

Este tipo de artefacto se puede empelar en cualquier punto del proceso de diseño, como una referencia para todo el equipo a lo largo de un ciclo de diseño e implementación del producto de software (Gibbons, 2017).

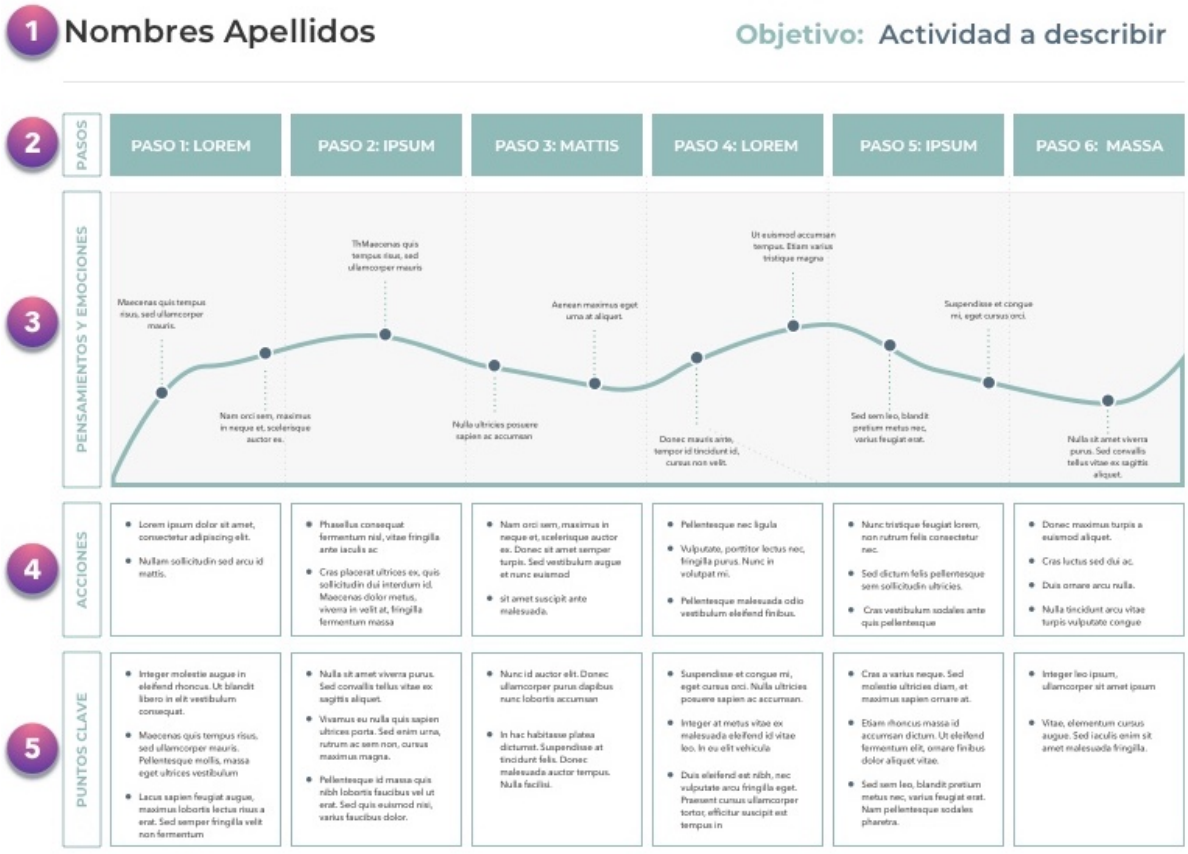


Figura 18: *Ejemplo básico del mapa de experiencia del cliente [Imagen elaborada por el autor].*

A continuación se describen brevemente los apartados que conforman el mapa básico de experiencia del cliente de la Figura 18.

NÚMERO 1: Se indica el nombre de la persona involucrada en realizar la actividad, así como el objetivo principal que se obtiene al realizar el proceso en cuestión.

NÚMERO 2: En una cabecera se escriben los principales pasos que conforman o completan el proceso. En esta tarea es importante realizar un análisis correcto del proceso para identificar sus principales pasos.

NÚMERO 3: Se colocan en una línea del tiempo los pensamiento y emociones principales que presenta el usuario conforme avanza la narración de la descripción de los pasos.

NÚMERO 4: En este apartado se listan las acciones principales de cada paso de manera cronológica.

NÚMERO 5: Se enlistan los puntos clave de cada paso del proceso. Donde el usuario presenta frustración o motivación en el desarrollo de una actividad

3.4.4.1 *Variante propuesta del mapa de experiencia del cliente*

En la dirección de proyectos de software los analistas con más experiencia concuerdan con que en la práctica, los procesos se solapan y pueden interactuar entre ellos de varias formas. Estos procesos se vinculan entre sí mediante las salidas que producen. Este comportamiento se debe a que los procesos son actividades superpuestas que se repiten a lo largo del proyecto (PMI, 2008).

Al trasladar estas ideas al análisis de las actividades cotidianas que realiza el usuario y a la comprensión de los requerimientos que se deben implementar para el software solución, en este trabajo se propone una variante al mapa de experiencia del cliente mediante la identificación de las entradas, las acciones o actividades que se realizan y las salidas que tienen los procesos. Con esta variante se pueden describir de mejor forma los procesos solapados, que son los que ocurren en la mayoría de las actividades cotidianas y laborales de los usuarios y que con frecuencia son los procesos de mayor complejidad para los analistas y desarrolladores de software.

En la Figura 19 se muestra la variación propuesta del mapa de experiencia del cliente, en el cual se caracteriza principalmente por utilizar iconos relacionados a los puntos clave en el proceso cotidiano, estos se deben comprender para realizar correctamente el leventamiento de requerimientos y que criticas en las primeras fases del desarrollo. Los iconos o imágenes alusivas al proceso se pueden acordar entre las partes interesadas y el grupo de analistas con el fin de evitar o reducir las ambiguedades en lo que se requisita. El mapa puede incluir una hoja de diseño con el significado de los iconos o imágenes empleadas en la narración, apoyando la idea de no generar documentación excesiva se recomienda empelar en el mapa imágenes que permitan reforzar las el texto descriptivo, para conseguirlo se identifican los puntos clave de cada paso del proceso.

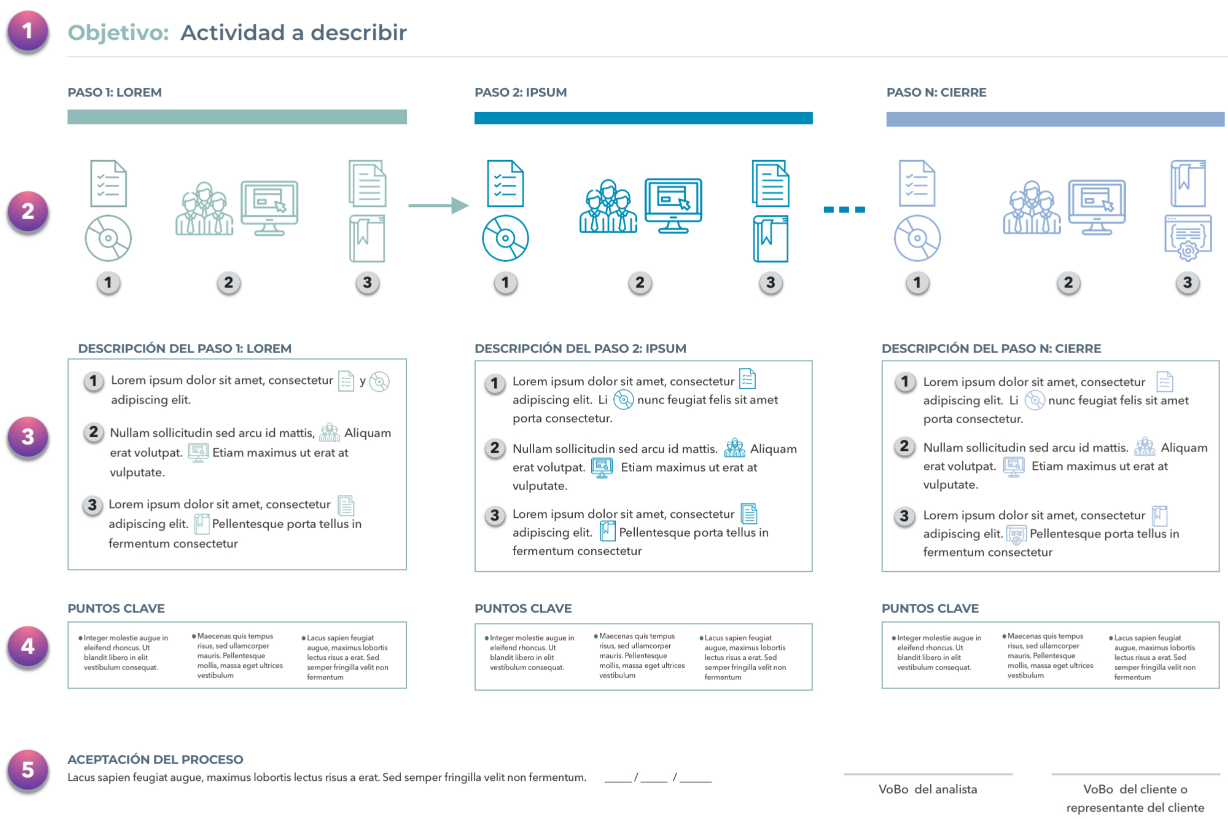


Figura 19: *Propuesta de mapa de experiencia del cliente [Imagen elaborada por el autor].*

A continuación se describen los apartados que conforman el mapa de experiencia del cliente de la Figura 19 propuesto en el actual trabajo.

NÚMERO 1: Se coloca el objetivo principal del proceso a describir en el mapa. Se pueden listar los objetivos secundarios cuando aportan imformación significativa del proceso. Indicar los objetivos permite mantener la narración sobre el mismo camino, evitando la ambigüedad y las distracciones narrativas.

NÚMERO 2: Indicamos el paso a describir de manera concisa y clara, debajo del título se dibuja una línea horizontal que abarca toda la narración en cuestión. Las líneas pemiten dar seguimiento visual a través de las actividades, así como el tipo de relación existente entre los procesos. La narración de las actividades se realiza de izquierda a derecha indicadas por flechas del mismo color representativo del apartado actual, el color juega un papel importante como guía en la descripción.

En el desarrollo de software se han documentado que los procesos guardan una relación entre sí (PMI, 2008), acontinuación se listan los principales:

- El proceso siguiente inicia cuando termina el actual, esto ocurre normalmente cuando un proceso requiere las salidas del paso actual para poder comenzar.
- Los procesos inician al mismo tiempo pero no finalizan igual, estos procesos ocurren de manera simultánea y se diferencian por que no comparten ni salidas ni entradas entre ellos pero un tercer (o más) paso emplea sus resultados.
- Solapamiento en el tiempo, esto ocurre cuando un proceso inicia sin que el anterior (o anteriores) hayan concluido, pueden iniciar incluso a mitad del proceso actual.

Los pasos solapados son más comunes de lo que los analistas piensan y en la propuesta al dibujar líneas de un color abarcando su descripción se permite identificar en que momento ocurre el solapamiento para que el analista tenga los cuidados pertinentes en la descripción de requerimientos. Al presentarse el solapamiento se recomienda indicarlo como un punto clave del proceso (ver número 4).

En la narración visual del proceso se utilizan iconos alusivos con el objetivo de reafirmar el modelo mental de la descripción de actividades. Los iconos permiten identificar los tipos de entrada de datos, tareas internas, interacciones entre los involucrados, salidas o resultados del proceso, solo por mencionar los principales. Se recomienda que los iconos de cada paso tengan el mismo color de la línea que abarca su narración.

En la propuesta del mapa de experiencia del cliente los iconos deben indicar y apoyar visulamente 3 partes que conforman la narración:

1. Entrada(s): los iconos o imágenes que representan las posibles entrada de datos, informes, documentos membretados, folletos, contratos, elementos digitales, manuales, ficha técnicas, o cualquier elemento indispensables para iniciar el proceso y que se emplea durante las operaciones o tareas del paso a describir.
2. Actividad(es) o tarea(s): estos iconos de la narración deben reresentar las operaciones que se realizan con las entradas ingresadas en el paso actual, por ejemplo: operaciones en un sistema, mediciones físicas, alteraciones en

documentos, actualizaciones de datos, intereacción entre interesados en el sistema, solo por mencionar algunos.

3. Salida(s): las salidas deben representar los resultados de las operaciones o tareas realizadas en el paso actual, vamos a considerar como salidas a todos aquellos elementos o productos resultantes de un proceso. Los resultados pueden ser documentos, un valor númerico, actualización de una base de datos, actualización del documento base, actualización de la documentación, corrección de un instrumento, una notificación, un producto mínimo viables (MVP [14]) solo por mencionar algunos.

En la selección de iconos los miembros del equipo de trabajo y responsables en el diseño de interfaces y de experiencia de usuario se encargan de proponer e implementar (o seleccionar de páginas con licencia libre [15] [16] [17]), el conjunto de iconos conforme al dominio y temática del problema del sistema a desarrollar.

En la comunicación con los miembros de otras áreas del desarrollo se pueden emplear fichas técnicas de diseño, donde se presentan los iconos, una breve descripción y su forma de uso en la interfaz gráfica de usarios del software. Los equipos de desarrollo de software están acostumbrados a emplear fichas técnicas de recursos de diseño gráfico, conocidos como el ArtWork o Assets[18] donde se coloca todos los recursos, tamaños, indicaciones de versiones, orientaciones y adaptaciones que se emplean en el software. Al tratarse de una actividad común de los equipos de trabajo, usar una ficha técnica de los iconos es una manera eficiente de usar un conocimiento de trabajo utilizado por los equipos, por lo cual no requiere de una curva de aprendizaje costosa en tiempo.

En una variante que optimiza el trabajo entre las partes de desarrollo de software (back-end y front-end) y la parte del diseño (diseño gráfico, interfaces de usuario y experiencia de usuario) se recomienda emplear una columna en el tablero

14 MVP, producto mínimo viable: En desarrollo de producto de software, es el producto viable mínimo con suficientes características para satisfacer a los clientes y/o interesados en el software y que proporciona retroalimentación para sus futuras entregas.
15 En este trabajo se proponen utilizar los iconos de Boostrap y de Flaticon.
16 Los icons de bootstrao se pueden descargar de https://getbootstrap.com/docs/3.3/components
17 Los icons de Flaticon se pueden descargar de https://www.flaticon.com/home
18 ArtWork o assets: Término empleado en los grupos de desarrollo de software para hacer referencia a todas las artes, diseños y elementos gráficos que se emplean en el desarrollo de las interfaces de usuario del software.

Kanban[19] como se muestra en la Figura 20 donde se colocan las fichas técnicas de diseño con algunos ejemplos de uso, mockups aceptados por el cliente, mapas de navegación, mapas mentales, enlaces de descarga, manuales, entre otros recursos que apoyan en las actividades del equipo y mejoran la comunicación durente el desarrollo. De esta forma se permiten aprovechar las tecnologías y los artefactos propuestos en este trabajo durante el ciclo de desarrollo de software.

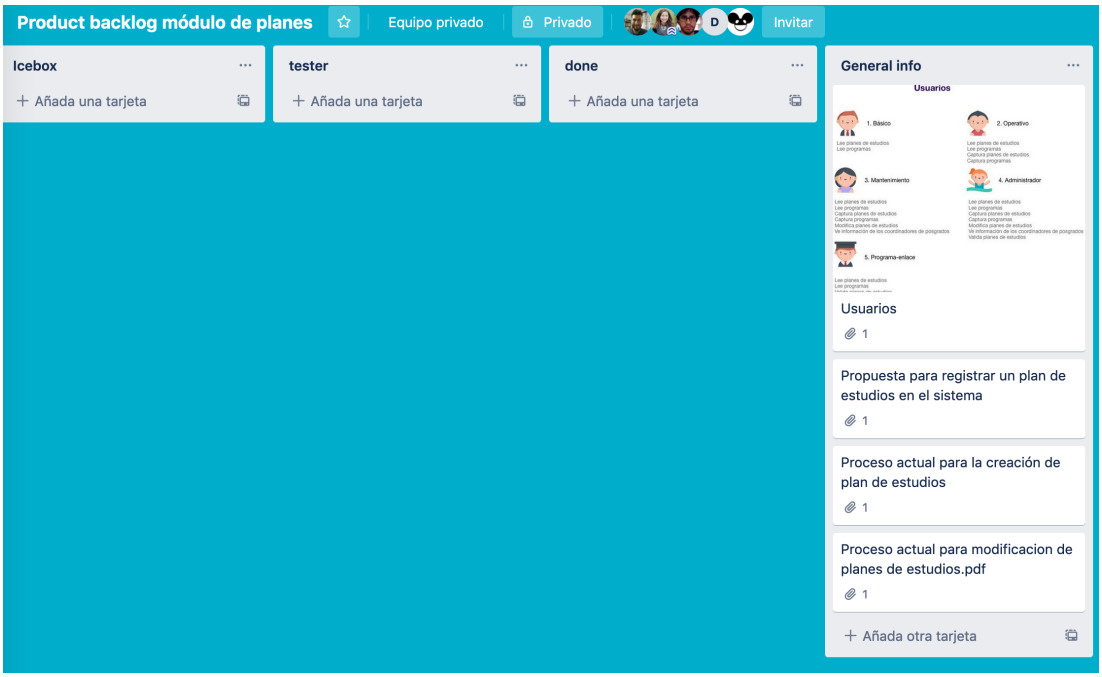


Figura 20: *Ejemplo de una columna Kanban empleada en un proyecto de desarrollo de software en la Unidad de posgrado de la UNAM. Donde se puede apreciar que la columna "General Info" contiene accesos rápidos a los recursos gráficos, directorio de stakeholders, mockups aceptados por el cliente, diagramas UML, entre otros recursos que los desarrolladores pueden emplear en sus tareas [Imagen elaborada por el autor].*

El líder de proyecto revisa el Kanban en un promedio de 20 veces al días tomando mayor atención en las actividades críticas del desarrollo. Los diseñadores y responsables de los recursos gráficos tienen un promedio similar de 15 a 20 revisiones de los Kanban y de los storyboards, debido a la narración y seguimiento visual de las interfaces gráficas de usuario de la aplicación. Los desarrolladores tienen un promedio relativamente menor de revisiones (5 o menos) del Kanban

19 Kanban: artefacto en forma de tablero con cartas empleado en los métodos ágiles para organizar visualmente las actividades a realizar en el proceso de desarrollo de un producto y dependiendo de la etapa en la que se encuentra se coloca la tarjeta en el tablero.

debido a que no se les ha inculcado está cultura de manera temprana durante la carrera, cuando los profesionistas egresan y entran en el campo laborar durante los primeros años trabajando en equipos de desarrollo el uso de estos artefactos de seguimiento de actividades les provoca molestias y presentan una curva de aprendizaje larga. Empleando la columna con los recursos antes mencionados se propicia el uso de los tableros y se permite dar un mejor seguimiento de las actividades en tiempos de entrega del producto.

NÚMERO 3: Se coloca la narración de cada tarea incluyendo los iconos alusivos para apoyar a los diseñadores y desarrolladores en la intereacción de los elementos en el mapa. Esta forma de presentar la información tiene como ventaja principal dar seguimiento en como las entradas del proceso mediante operaciones intermedias generan las salidas. Al describir correctamente como se producen las salidas, se aporta una mayor comprensión de la complejidad del proceso que será sistematizado y de la misma forma indicar si el proceso se puede sistematizar completamente o solo se puede apoyar parcialmente al operador o involucrado en completar dichas tareas.

NÚMERO 4: En este apartado se anotan los puntos clave del proceso, estos elementos permiten conocer y entender de mejor manera la forma en la que los usuarios realizan las actividades. La información permite identificar en que puntos se pueden complicar o facilitar la sistematización. Esto se aprovecha en la etapa del diseño de interfaces, en como puede impactar la intereacción del usuario con la aplicación, por ejemplo la forma en la que colaboran los involucrados para completar las tareas del proceso.

NÚMERO 5: Se incluye una leyenda de aceptación de la narración del proceso descrito por el cliente y presentando en el mapa por los analistas. Si la narración de los pasos coincide con la realidad descrita por el cliente y/o interesado en el proyecto, se solicita una firma de aceptación de que dicho proceso es el que se va a abordar el desarrollo del proyecto. De no coincidir con la realidad solicitada por el cliente se deben realizar observaciones para ajustar el mapa y proceder con las etapas del ciclo de desarrollo. Este elemento en el instrumento permite involucrar más al cliente y/o interesado en el proyecto desde las etapas tempranas. Desde el punto de vista del analista tener retroalimentación sobre los procesos a sistematizar permite obtener resultados más cercanos a la realidad.

Tabla 4: *Comparativa del Mapa de experiencia del cliente propuesta contra la versión tradicional*

Mapa propuesto	Mapa tradicional
Manejo del solapamiento de pasos o actividades	
El solapamiento se consigue colocando el paso solapado en un bloque renglón abajo del paso actual y asociando un color de línea, esta representación visual indica que 2 pasos se están en ejecutanto de manarea simultánea, la asociación por color en la línea del bloque y en los iconos ayuda en la narración de las actividades. En la Figura 21 se muestra un ejemplo de como se representa el solapamiento de los pasos de un proceso, empleando la propuesta de este trabajo.	Es complicado dibujar los pasos solapados dentro de un mismo mapa, por tal motivo se emplea otro esquema con el solapamiento y se describen textualmente en la narración las actividades sumultáneas. Utilizar esta forma se convierte en una labor pesada asimilar la información por parte de los equipos, debido al incremento significativo de texto y a una representación visual que complica el acceso a la información.
Manejo de los sentimientos y emociones	
Se indican en los puntos clave debido a que llegan a ser determinantes en el momento de diseñar y desarrollar la interfaz gráfica del usuario y los mecanismos de interacción del usuario con el sistema. Colocarlos en ese apartado es menos invasivo y permite centrar la antención en la narración del proceso.	Recaba las emociones y sentimientos directamente sobre la línea de tiempo de la narración de las acciones, lo que puede llegar a dificultar su lectura y comprensión. Se considera que el apartado de puntos clave replica parte de esa información o no es aprovechado correctamente. Cuando se recaba una gran cantidad de

Tabla 4 – *Continua en la siguiente página*

Mapa propuesto	Mapa tradicional
Este instrumento se complementa con el artefacto mapa de empatía (ver Figura 14) en el cual se recaban las emociones y pensamientos que tienen los usuarios en ciertos escenarios de la vida cotidiana y que se verían beneficiados con el software.	emociones el diagrama se satura, se dificulta la colocación de más elementos y su contenido se vuelve complicado de entender. De manera contraria colocar una cantidad reducida de emociones se puede interpretar como un desaprovechamiento del espacio o un reflejo de un análisis incorrecto de usuarios y de sus actividades.
Manejo de las entradas, actividades y salidas de los pasos o actividades	
En el mapa se describen las entradas que requieren los pasos para iniciar, sus tareas o actividades y las salidas que arrojan. Mediante iconos representativos permite a los diseñadores y desarrolladores llevar un control mental al estilo lista de control (check-list) de los elementos que debe contemplar para el desarrollo del sistema en esa etapa del levantamiento de requerimientos. Por ejemplo: colocar un icono de documento para indicar que el paso requiere de un oficio físico al iniciar, un icono de personas reunidas indicando que las actividades se realizan en colaboración, y una nube o base de datos para indicar que el resultado de la salida es una actualización de datos en los sistemas.	Las entradas, actividades y salidas de los pasos son descritos en el apartado de acciones y puntos clave del diagrama tradicional. Esto puede generar como se describe en los puntos anteriores que se sature el diagrama, dificultando el contenido para los diseñadores y desarrolladores.
Aceptación del proceso	

Tabla 4 – *Continua en la siguiente página*

Mapa propuesto	Mapa tradicional
En la parte inferior del diagrama se incluye una leyenda indicando que el cliente está de acuerdo con la descripción del proceso, con ello avalan que la comprensión del proceso descrito por los analistas y plasmada en el diagrama se apega de manera satisfactoria con el proceso real a desarrollar proporcionada por los clientes y/o intereados en el proyecto. De presentarse observaciones se anotan en el diagrama indicando los cambios necesarios para que se apegue lo mejor posible a la realidad. Al incorporar este tipo de elementos en los diagrama se propicia en el cliente una mayor atención y compromiso en la dinámica.	El diagrama tradicional no cuenta con ese apartado pero se recomienda incorporarlo para obtener productos más apegados a la realidad y propiciar la participación de los clientes y/o interesados en el proyecto.

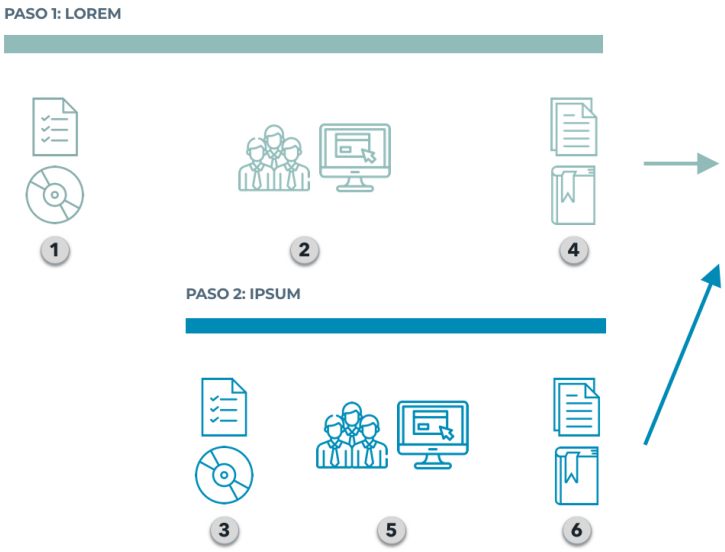


Figura 21: *Ejemplo el manejo de pasos solapados empleando el mapa de experiencia del cliente propuesto en este trabajo [Imagen elaborada por el autor].*

3.4.5 *Artefacto Kanban y pila del producto*

Con el documento de especificación de requerimientos (ver sección 2.4.1), se procede a generar una estrategia de como se van a desarrollar dichos requerimientos, es decir, el orden de priorización en el que se van implementando. En este trabajo para dicha tarea se propone emplear el artefacto Kanban, el cual es un tablero o matriz donde se emplean columnas para indicar el estado del desarrollo del requerimiento del proyecto, en las columnas se colocan las tarjetas con la narración del requerimiento o historia de usuario a desarrollar.

El instrumento viene del Japonés Kanban y se traduce como ”letrero”, dado el éxito de los métodos ágiles se ha convertido en un instrumento ampliamente utilizado. La raíces de Kanban se remonta a los orígenes del sistema de producción de Toyota, a principios de la década de 1950 Taiichi Onho lo desarolló como un artefacto para controlar la producción entre procesos e implementar la fabricación justo a tiempo (del Inglés Just in Time o JIT) en las plantas de fabricación de Toyota (Gross and McInnis, 2003).

El artefacto fue aceptado por los analistas hasta 1970 debido a que en la recesión global, resultó de gran utilidad al minimizar el trabajo entre procesos y redujo el costo asociado con la retención del inventario. Inicialmente se usó Kanban para reducir costos y administrar la maquinaria, sin embargo, actualmente muchas compañías continúan utilizando el sistema para administrar los costos y el flujo, identificar los impedimentos en el flujo y las oportunidades de mejora continua (Gross and McInnis, 2003).

Con el tablero de Kanban los diseñadores y desarrolladores usan elementos visuales para indicar cuando una tarea se lleva acabo, cuando se detiene o cambia de estado. Las reglas Kanban también indican qué hacer cuando se presenta algún problema, dado que se pueden mover a una columna específica para ser atendida cuando surgen esos problemas, un ejemplo común es dedicar una columna a pruebas de calidad, en la cual se envían los productos con el responsable de asegurar la calidad de los entregables del producto.

Una buena planificación del Kanban tiene indicadores visuales que permiten al administrador del proyecto tener un panorama completo del estado del cronograma de las actividades, identificar tareas atrasadas y conocer el avance de las actividades de la ruta crítica. Este artefacto se complementa con el diagrama de gantt (ver sección 3.4.6).

Un tablero Kanban estándar esta conformado por tres columnas: Pendiente, En proceso y Finalizado. Cuando el tablero se construye y administra adecuadamente desempeña la función de almacén de datos en tiempo real, indicando desafíos dentro del proceso y puntos críticos que pueden interferir con el funcionamiento del proyecto.

Kanban usa un mecanismo de control visual para hacer seguimiento del trabajo conforme al flujo de valor. En un principio se utilizaba una pizarra con notas adhesivas, actualmente se emplean software con tarjetas. Se recomienda emplear ambas técnicas para realizar de manera efectiva el seguimiento de los procesos. Esto se debe por experiencia en proyectos con fondos privado que los equipos de desarrollo, aquellos dedicados al backend y front-end se adaptan al uso de herramientas tecnológicas, pero algunos miembros del equipo de diseño de interfaces y de experiencia de usuario están más acostumbrados a dar seguimiento de las tareas a través de un tablero físico con notas adhesivas.

La transparencia que este artefacto genera contribuye al cambio cultural de los equipos de trabajo. Las metodologías Ágiles han obtenido buenos resultados proporcionando transparencia respecto al trabajo en curso y completado, así como en el trabajo a través de métricas como la velocidad (cantidad de trabajo realizado por iteración). Kanban sin embargo va un paso más allá y proporciona transparencia al proceso y a su flujo. El manejo adecuado del artefacto permite exponer cuellos de botella, colas, variabilidad y desperdicios en los recursos. Todas las cosas que impactan al rendimiento de la organización en términos de la cantidad de trabajo entregado y el ciclo de tiempo requerido para entregarlo. Kanban proporciona a los miembros del equipo y a las partes interesadas visibilidad sobre los efectos de sus acciones (o de la falta de acción).

En Gross and McInnis (2003) se mencionan al detalle los beneficios de emplear kanban en la dirección de proyectos, de los cuales a continuación se listan las principales características que fueron consideradas en la utilización de este artefacto en proyectos de tecnología y en el actual trabajo:

1. En empresas con giro de producción reduce el inventario.
2. Mejora el flujo del trabajo.
3. Aumenta la capacidad de gestionar la cadena de suministro
4. Previene la sobreproducción, con lo que se evitan esfuerzos innecesarios.
5. El control se ubica al nivel de las operaciones.
6. Minimiza el riesgo de obsolescencia de inventario

7. Presenta una mejora con respecto al proceso estándar, en la capacidad de respuesta a cambios.
8. Crea una representación visual de la gestión del proceso.

En la Figura 22 se muestra la variación propuesta del Kanban, en el cual se caracteriza por utilizar estados del proceso probados en varios proyectos a lo largo de 5 años en empresas de tecnología y en el que se actualiza a las necesidades y retos que se presentan.

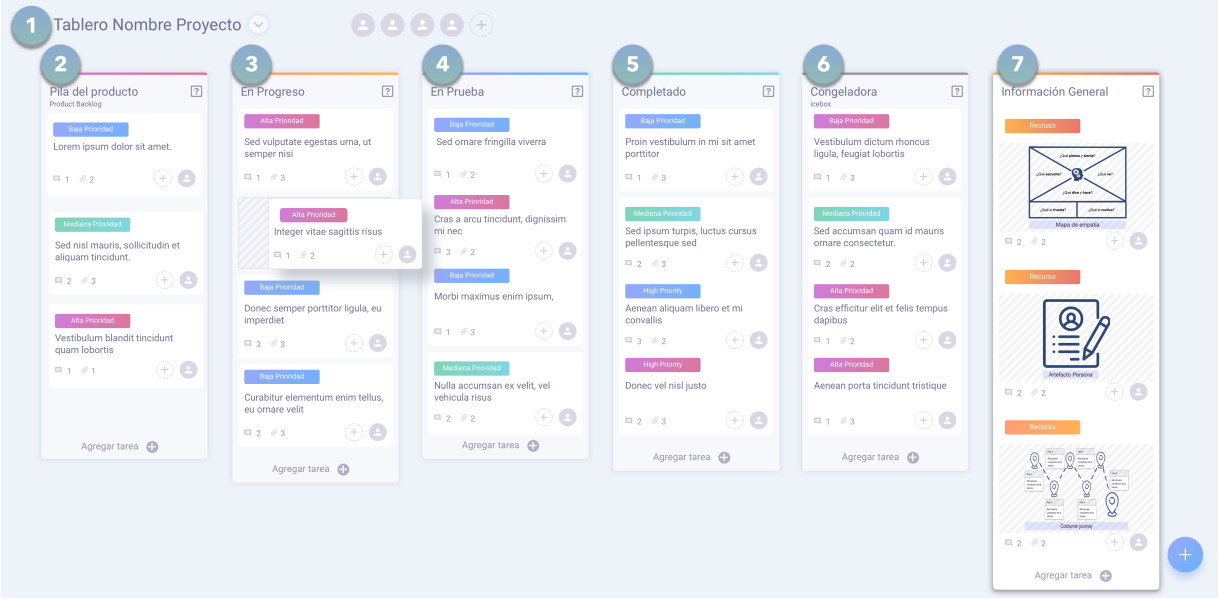


Figura 22: *Ejemplo del artefacto Kanban empleado en el trabajo [Imagen elaborada por el autor].*

A continuación se describen los apartados que conforman el Kanban de la Figura 22 propuesto en el actual trabajo, puntualizando que es una propuesta que se ha venido conformando con la experiencia de participar en varios proyectos de ingeniería de software y en la impartición de cursos de diseño de interfaces.

NÚMERO 1: Se indica el nombre del proyecto a dar seguimiento y la lista de los participantes en el proceso. En varias organización el movimiento de las actividades a otras columnas se monitorea por el líder del proyecto, se recomienda que quién realice un movimiento entre estados notifique al líder del proyecto, esta acción permite llevar una buena comunicación en el equipo y se manifiesta el compromiso que tienen los involucrados con el proyecto.

NÚMERO 2: La pila del producto son los requerimientos o historias de usuario que se van a desarrollar en el actual sprint de desarrollo. Los requerimientos presentan un responsable encargado de realizar las tareas para completarlo, una breve descripción de lo que se debe implementar, una prioridad calculada con anticipación y si es necesario enlaces a recursos de lo que se va a implementar.

NÚMERO 3: Columna del estado ”En Progreso”, se colocan los requerimientos o historias de usuario que se estan implementando en este preciso momento. La tarjeta permanece en ese estado mientras se complete su desarrollo.

NÚMERO 4: Columna del estado ”En Prueba”, se traslada el requerimiento o historia de usuario para realizar pruebas de calidad, de implementación, unitarias o las que corresponda o requiera el proceso. En esta etapa se puede reasignar la tarea a una nueva persona o rol de la organización. Como recomendación se asocian a las personas involucradas en el requerimiento para contactar con los responsables en caso de algún flujo alternativo o de error en los resultados. Si la implementación requiere correcciones se devuelve a la columna de ”En Progreso” para que el responsable realice las correspondientes modificaciones. De presentarse alguna complicación se puede optar por enviarlo a la ”Congeladora” (ver número 6).

NÚMERO 5: Columna del estado ”Completado”, se colocan los requerimientos o historias de usuario que se han completado con éxito y que han pasado las pruebas requeridas en la columna anterior. Como recomendación se registra el tiempo real que llevo completar el requerimiento, estos datos permitiran mejorar la medición en los tiempos de desarrollo del equipo de trabajo.

NÚMERO 6: Columna del estado ”Congeladora” en está sección se colocan los requerimientos que por alguna situación de tiempo, recurso humano, recurso monetario o cualquier otro, no se pueden completar. En experiencia con empresas de tecnología esto se puede presentar cuando se espera el visto bueno de la alta gerencia que proviene de oficinas principales en otra parte del mundo con otro huso horario.

NÚMERO 7: Columna de ”Información General” en este apartado se colocan los recursos de diseño de interfaces, de diseño de experiencia de usuario, casos de uso, mockups, entre otros elementos y artefactos que apoyan en el desarrollo del producto del software.

En este trabajo se recomienda emplear las tecnologías Web de gestión de proyectos Redbooth[20] y Trello[21]. Dichas tecnologías cuentan con un tablero Kanban para la gestión de los procesos y requerimientos del proyecto. En la Figura 23 se muestra un ejemplo de tablero Kanban de un proyecto empleando Redbooth.

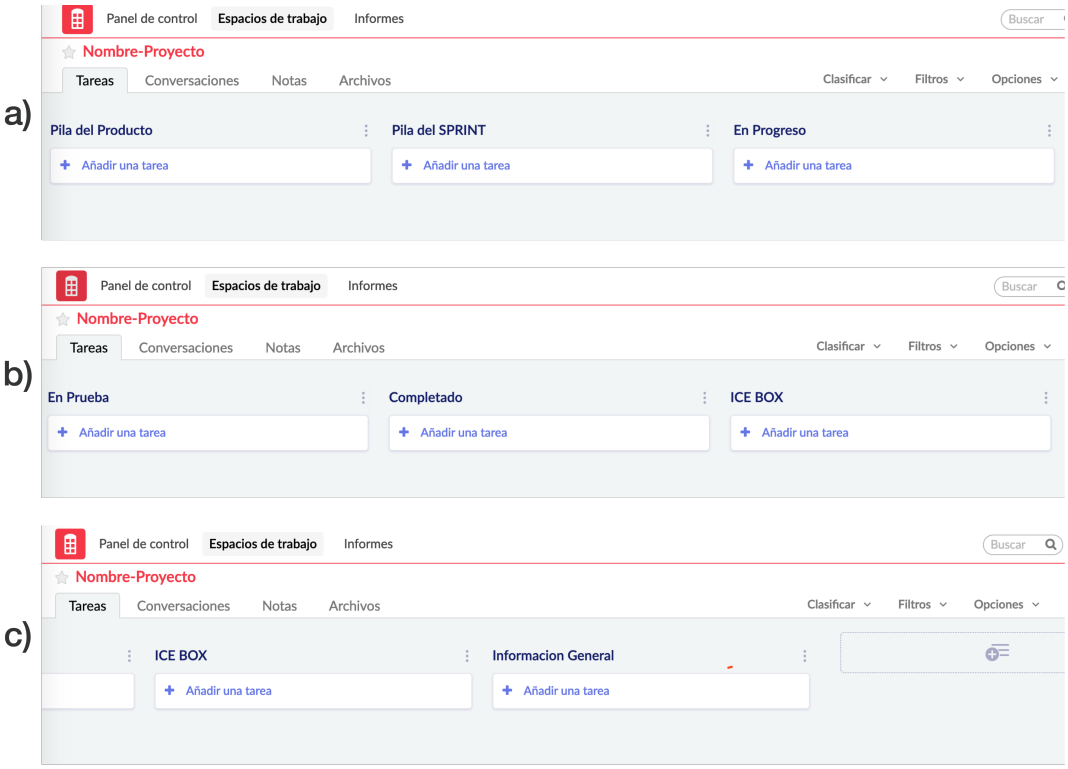


Figura 23: *Las imágenes a), b) y c) son ejemplo de un tablero de Kanban empleando la herramienta Redbooth [Imagen elaborada por el autor].*

20 Redbooth: Es una herramienta de administración de proyectos de software. Recurso disponible en https://redbooth.com/a/

21 Trello: herramienta informática para crear y administrar tableros Kanban. Recurso disponible en https://trello.com/

3.4.6 *Artefacto diagrama de Gantt*

El artefacto de diagrama de Gantt sugió como necesidad de la dirección de proyectos para decidir sobre políticas y acciones a tomar durante el desarrollo de un proyecto, de acuerdo con aquellas políticas que conducen a un resultado deseado. Las decisiones que afectan el futuro de un proyecto se deben basar en el conocimiento de los resultados de tareas anteriores, y para ello se requieren registrar eventos que han tenido lugar o que se ha realizado una cierta cantidad de trabajo para tomar tales decisiones (Clark, 1923).

El diagrama de Gantt permite identificar actividades que se van a desarrollar, los recursos que pueden emplear y una estimación del tiempo que puede tomar (se puede ajustar), de tal modo que se evitan periodos ociosos innecesarios y el administrador tiene una visión completa de la utilización de los recursos que se encuentran bajo su supervisión.

En la Figura 24 se muestra un ejemplo de un diagrama Gantt tradicional el cual esta conformado por: 1) una cabecera con el nombre del proyecto y la lista de los participantes; 2) una fila con las fechas calendarizadas en función de la jornada laboral acordada por el equipo de trabajo y monitoreada por el líder del proyecto; 3) columna con las actividades a desarrollar en el Sprint de desarrollo y 4) barras horizontales indicanto el tiempo calculado que toma desarrollar las actividades o tareas de la columna (3).

Durante el desarrollo de software las herramientas informáticas ayudan a llevar un mejor control y administración de las actividades que se realizan en el ciclo de desarrollo. En la Figura 25 se muestra un ejemplo de un diagrama Gantt empleando la herramienta informática de Redbooth, en ella se muestran actividades ficticias de un proyecto, con sus respectivas barras de tiempo calculadas para un Sprint de desarrollo.

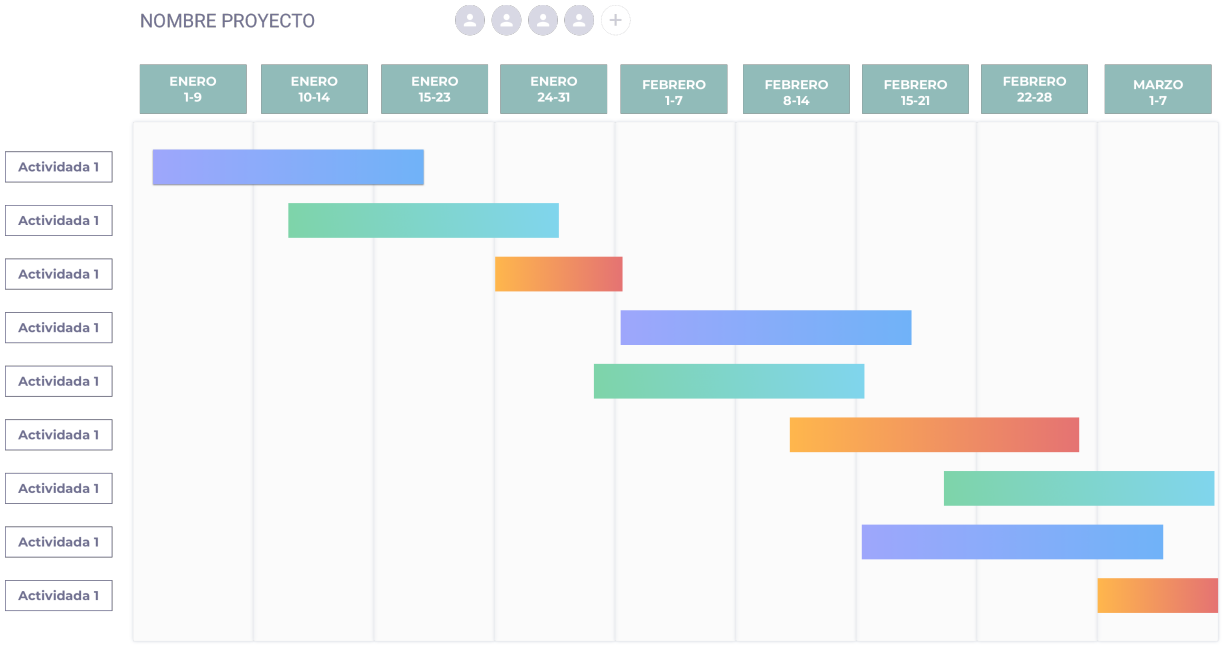


Figura 24: *Ejemplo de un diagrama de Gantt tradicional [Imagen elaborada por el autor].*

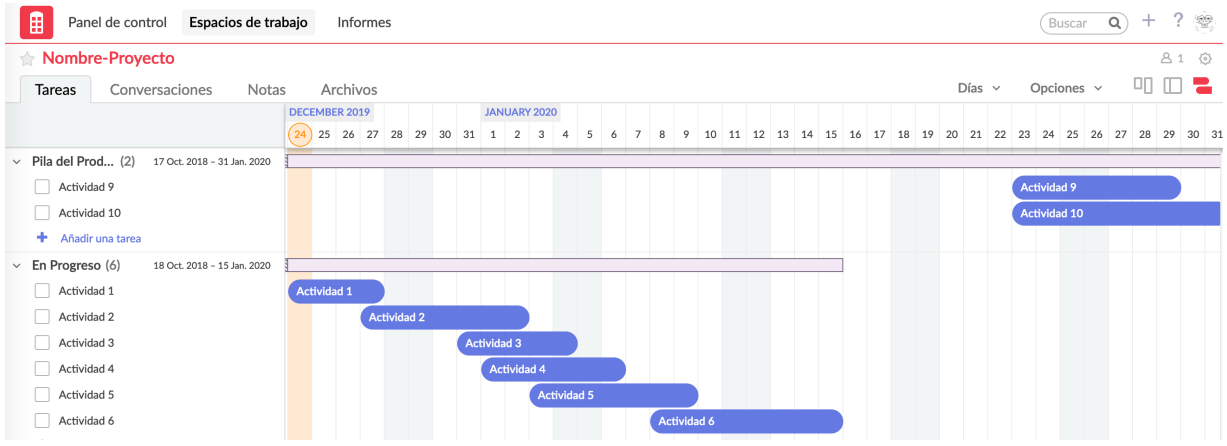


Figura 25: *Ejemplo de un diagrama Gantt utilizando la herramienta Redbooth [Imagen elaborada por el autor].*

4

VALIDACIÓN DEL PROCESO

En este capítulo se presenta la validación de la propuesta de levantamiento de requerimientos y fase inicial de análisis en el proceso de desarrollo de software. Al tratarse de una variante en el proceso de levantamiento de requerimientos, la opinión de expertos es una estrategias muy empleada para su validación. En este proceso se emplea el método Delphi de consulta a expertos, considerado uno de los métodos subjetivos de pronosticación más confiables, al combinar criterios de análisis de base subjetiva con análisis estadístico de los resultados que se pueden emplear en la ejecución del proceso en la práctica del desarrollo de software.

En Harold A. Linstone (2011) se aborda la técnica Delphi como un método para estructurar un proceso de comunicación grupal, de modo que el proceso sea efectivo para que un grupo de individuos, desempeñe un problema complejo. La fuerza del método Delphi es su capacidad de avanzar a nuevas formas de aplicación, en donde se propician y refinan los juicios grupales (Grime and Wright, 2016).

La ténica Delphi es un procedimiento de pronóstico y debido a su uso, se ha empleado en una variedad de problemas en diferentes áreas. En Grime and Wright (2016) se apoya la idea de que más opiniones son mejores que una sola y cuando se usan como herramienta de pronóstico, los esfuerzos grupales bien estructurados conducen a pronósticos más precisos que los no estructurados.

4.1 ASPECTOS PARA EMPLEAR DELPHI

En Harold A. Linstone (2011); Grime and Wright (2016) se describen una serie de aspectos que se deben considerar para decidir si Delphi es la opción correcta al problema a resolver. Por lo general, una o más de las siguientes propiedades lleva a la necesidad de emplear Delphi:

- La interacción colectiva de juicios personales permiten localizar el problema desde diferentes enfoques en beneficio del análisis y solución del problema.

- Las personas consideradas para contribuir al análisis de un problema amplio o complejo no tienen antecedentes de comunicación adecuada y pueden representar antecedentes diversos con respecto a la experiencia.
- Se necesitan más individuos que puedan interactuar efectivamente.
- El tiempo y el costo hacen que las reuniones grupales frecuentes no sean factibles.
- La eficiencia de las reuniones cara a cara se puede aumentar mediante un proceso de comunicación grupal complementario.
- Los desacuerdos entre individuos son tan severos o políticamente desagradables que el proceso de comunicación debe ser arbitrado y asegurado mediante el anonimato
- Se debe preservar la heterogeneidad de los participantes para asegurar la validez de los resultados, es decir, evitar la dominación por cantidad o por la fuerza de la personalidad.

4.2 VARIANTES DEL PROCESO DELPHI

En la actualidad exiten dos versiones del proceso de Delphi ampliamente utilizadas, la forma Delphi convencional y la Conferencia de Delphi (Harold A. Linstone (2011)):

La convencional es una versión en papel y lápiz que comúnmente se conoce como “Ejercicio Delphi”. En esta situación, un equipo reducido de monitores diseña un cuestionario que se envía a un grupo más grande de encuestados. Después de devolver el cuestionario, el equipo de monitores resume los resultados y desarrolla un nuevo cuestionario para el grupo de encuestados. El grupo de encuestados generalmente tiene al menos una oportunidad para reevaluar sus respuestas originales basándose en el examen de la respuesta del grupo. Hasta cierto punto, esta forma de Delphi es una combinación de un procedimiento de votación y un procedimiento de conferencia que intenta desplazar una porción significativa del esfuerzo necesario para que las personas se comuniquen desde el grupo de encuestados más grande al de monitor más pequeño.

Una forma más reciente que emplea el uso de las TIC, es llamada “Conferencia de Delphi”, reemplaza en gran medida el rol del monitor por un sistema en computadora que lleva a cabo la compilación de los resultados del grupo. Este último enfoque tiene

la ventaja de eliminar el retraso causado al resumir cada ronda de Delphi, convirtiendo así el proceso en un sistema de comunicaciones en tiempo real. Sin embargo, requiere que las características de la comunicación estén bien definidas antes de emprender la técnica Delphi.

El proceso Delphi, en cualquiera de sus versiones, se somete a cuatro fases distintas. La primera fase es la exploración del tema en discusión, en el que cada individuo aporta información adicional que considera pertinente para el tema. La segunda fase implica el proceso de llegar a una comprensión de cómo el grupo ve el tema (dónde los miembros están de acuerdo o en desacuerdo). Si hay un desacuerdo significativo, entonces ese desacuerdo se explora en la tercera fase para resaltar las razones subyacentes de las diferencias y posiblemente evaluarlas. La última fase, una evaluación final, ocurre cuando toda la información recopilada previamente ha sido analizada y las evaluaciones han sido retroalimentadas para su consideración.

4.3 LIMITANTES EN LA EJECUCIÓN DE LA TÉCNICA DELPHI

Delphi parece un proceso simple que se puede emplear fácilmente. Debido a esto, muchas personas han dado un salto al intentar el procedimiento sin considerar cuidadosamente los problemas involucrados en la realización del ejercicio. En Harold A. Linstone (2011) se presentan algunas de las razones comunes por las que el proceso Delphi fracasa:

- La **imposición de la visión del monitor** y las ideas preconcebidas de un problema sobre el grupo de encuestados al especificar en exceso la estructura del Delphi y que no propicie la contribución de otras perspectivas relacionadas con el problema.
- Asumir que Delphi puede ser un sustituto de todas las demás comunicaciones humanas en una situación dada.
- Técnicas deficientes en el análisis, interpretación y presentación de la evaluación utilizadas en el ejercicio.
- **Ignorar y no explorar desacuerdos**, para que los disidentes desanimados abandonen y se genere un consenso artificial.

Unos de los tipicos problemas que plantea el proceso Delphi es cómo elegir un “buen” grupo de encuestados. De hecho es un problema para la formación de cualquier grupo de estudio: comités, paneles, grupos de estudio, solo por mencionar algunos. Cualquier estudio tiene este problema sin importar qué modo de comunicación se utilice; por lo tanto, si bien es un problema realmente significativo, no es un problema exclusivo de Delphi. Otro problema virtual con frecuencia surge cuando se emplea un diseño particular de Delphi para ser utilizado en una aplicación particular se tome como plantilla representativa para emplear a todos los Delphis. El problema aquí es hacer una definición demasiado explícita y restrictiva para Delphi. Un tercer problema virtual es la honestidad del equipo de monitoreo, y es la misma preocupación que la honestidad de cualquier grupo de estudio o análisis. Finalmente, los malentendidos pueden surgir de las diferencias en el lenguaje y la lógica si los participantes provienen de diversos orígenes culturales.

4.4 SELECCIÓN DEL GRUPO DE ENCUESTADOS

El proceso de selección de técnicas de levantamiento de requerimientos y fase temprana de análisis, cumple con los siguientes aspectos descritos en la sección 4.1 (Aspectos para emplear Delphi), que apoyan el empleo de la técnica Delhi para su evaluación: El proceso se beneficia de juicios subjetivos de un grupo colectivo de expertos; Proporciona un medio en el que un conjunto de expertos puedan interactuar efectivamente; debido al tiempo limitado que los expertos pueden proporcionar hace que las reuniones presenciales no sean factibles y el uso de herramientas digitales propicia la colaboración; y las opiniones de los expertos se pueden asegurar mediante su anonimato, evitando que las opiones de algún experto se impongan sobre las de los demás.

En la estrategia de evaluación del proceso se selecciona la variante Delphi que emplea las tecnologías de la información (ver sección 4.2. Variantes del proceso Delphi), para la cual se utilizan encuestas digitales (ver anexo B) usando las herramientas de Facebook [1] que permite generar, administrar y distribuir las encuestas en el grupo de expertos revisores del proceso.

El grupo de expertos en el desarrollo de software y sus fases de análisis esta conformado en la primera etapa por 10 participantes en diferentes ubicaciones. Como criterios para la selección se consideraron expertos que un su práctica profesional rea-

1 Herramienta de encuesta de Facebook, no requiere que los participantes encuestados cuenten con una cuenta propia de la red social.

lizan desarrollo y dirección de proyectos de software. Con amplia experiencia en la ingeniería de requerimientos, técnicas de levantamientos de requerimientos y análisis de la información en las etapas iniciales en la dirección de un proyecto de TI, aspectos que son centrales en el presente trabajo.

El primer cuestionario (ver Figuras 31 - 34 del anexo B) consiste de 8 preguntas que tocan los siguientes temas: la dificultad del proceso de levantamiento de requerimientos en el desarrollo de software y su importanción en el producto final; la dificultad al aplicar el proceso; el impacto de los artefactos descritos en el proceso; la dificultad de integrar el proceso a metodologías de desarrollo de software y la factibilidad de emplear el proceso en proyectos reales. El cuestionario cierra solicitando una retroalimentación y opinión general para mejorar el proceso propuesto.

En el segundo cuestionario (ver Figuras 35 y 36 del anexo B) a partir de las observaciones de los expertos se elaboraron 5 preguntas sobre los siguientes aspectos: determinacion de las activiades propuestas a utilizar; apoyo de una matriz de decisión en el proceso; la fexibilidad de adaptarse a proyectos de menor manufactura; la asequibilidad de utilizarlo en proyectos empezados, la facilidad de emplear por analistas nuevos y la capacidad de adaptación del proceso a las necesidades del proyecto.

4.5 Resultado del proceso Delphi

El proceso Delphi empleado en este trabajo consta de dos rondas de encuestas aplicadas a un panel de expertos (ver la sección 4.4. Selección del grupo de encuestados), para obtener una percepción y evaluación del proceso de levantamiento de requerimientos propuesto. En el análisis de los resultados se cuenta con el apoyo técnico de un experto en estadística, el Dr. Mora Gutiérrez Roman miembro del posgrado de Diseño y Visualización de la Información de la UAM, unidad Azcapotzalco. Para el análisis de los resultados se emplea estadística descriptiva de las respuestas proporcionadas por el panel de expertos.

4.5.1 *Análisis de la primera ronda*

De las observaciones registradas en la primera ronda de preguntas (ver anexo C) se obtienen los siguientes resultados.

En la mayoría de los recursos bibliográficos citados en el trabajo, sobre el proceso de levantamiento de requerimientos (ver sección 2.3 Levantamiento de requerimientos), los autores coinciden en que es una etapa crítica para el desarrollo del proyecto de software y por tal motivo son consideradas actividades en las que se debe poner atención y cuidado para realizarlas adecuadamente y no comprometer el éxito del software. El 80 % de los panelistas encuestados coinciden con que son actividades de dificultad moderada y que se pueden realizar adecuadamente empleando técnicas de apoyo. El anterior resultado se explica de la experiencia y del conjunto de habilidades y estrategias que han desarrollado y perfeccionado a lo largo de diferentes proyectos de software. Siguiendo sobre el impacto que tiene un mal levantamiento de requerimientos sobre el proceso de ingeniería de software. El 70 % considera que es muy negativo, mientras un 30 % indica que es algo negativo en el software resultante y sus funcionalidades.

De las complicaciones que se pueden presentar en el levantamiento de requerimientos, se preguntó al panel de expertos sobre que tanto aporta el proceso propuesto en reducirlas. El 40 % indican que aporta mucho en la reducción de errores, mientras que el 60 % perciben que el trabajo da la pauta para reducirlos. El 70 % de los expertos indican que emplear los artefactos propuestos aportan positivamente en el desarrollo del software, mientras el otro 30 % comentan que están dispuestos a emplear los artefactos para ver los beneficios que se puedan presentar.

Al cuestionamiento relacionado a la capacidad que tiene el proceso para incorporarse a metodologías de desarrollo de software. El 100 % del panel de expertos coincide en que se puede incluir con metodologías de desarrollo de software actuales considerando el ciclo de vida del proceso. Los miembros mencionan que es una “buena práctica” emplear los artefactos descritos en el proceso de levantamiento de requerimientos, realizando previamente un análisis y estrategia en la dirección del proyecto.

La primera ronda cierra preguntando a los expertos, si están dispuestos a emplear el proceso propuesto en la dirección de proyectos de software. El 100 % de los miembros consideran emplear la propuesta en la gestión de sus proyectos de software. Algunos puntualizan que han empleado artefactos indicados en los pasos del trabajo, pero les resulta interesante el enfoque presentado y los resultados que se pueden obtener.

4.5.2 *Comentarios de los expertos*

El panel de expertos arrojaron varios comentarios positivos así como observaciones relacionadas a la aplicación del proceso y de su posible utilización en sus proyecto de software. A continuación se presenta un resumen de las observaciones más importantes.

En el aspecto del tiempo que requiere el proceso se registra en las observaciones que la mayoría de los expertos consideran que puede demorar debido a las etapas e instrumentos propuestos en el levantamiento de requerimientos. Esta observación parte del punto de que los clientes y/o dueños del proyecto no consideran el tiempo adecuado que pueden llevar las tareas de la ingeniería de requerimientos. Ante lo complejo de la activida se convierte en un deber de los analistas convencer a los dueños del proyecto en aceptar que es una etapa crítica en el proyecto y que consume tiempo. Las actividades realizadas en esta etapa del desarrollo de software, se ven beneficiadas o perjudicadas por la experiencia que tienen los analistas encargados de dichas tareas. En la práctica el analista experto hace uso de la introspección para realizar una estrategia mental para llevar un manejo adecuado del nivel de abstracción en la definición de los requerimientos.

En lo que refiere al aporte de información que presenta el proceso, los expertos indican que el contenido y artefactos propuestos son adecuados para el proceso de levantamiento de requerimientos aplicando a proyectos. Por lo regular en esta parte del proceso de desarrollo se suelen abordar entrevistas con los interesados seguida de la planificación, sin considerar otras técnicas de levantamiento de requerimientos, ni los beneficios que pueden aportar en la claridad y precisión de los requerimientos del proyecto.

En lo que respecta a la complejidad del proceso indican que es un proceso extenso y dirigido a proyectos grandes, que tienen suficiente tiempo para completar los artefactos y en los que realizar un análsis adecuado de los requerimientos es crucial para realizar una correcta implementación del software. Apoya la flexibilidad de emplear parte de los artefactos en proyectos que urgen o tienen problemas de entregas programadas en tiempo.

4.5.3 *Análisis de la segunda ronda*

De las respuestas y observaciones proporcionadas por los expertos en la segunda ronda del proceso Delphi (anexo B) se obtienen los siguientes resultados.

En los aspectos de asequibilidad y recomendación de emplear el proceso propuesto únicamente en proyectos “Grandes”, el 100 % de los expertos consideran que puede funcionar para cualquier proyecto y hacen énfasis en que lo importante es cumplir todos los requerimientos del proyecto para medir la efectividad y éxito del proyecto de software.

En el cuestionamiento sobre si el proceso propuesto puede ser empleado en proyectos ya comenzados. El 100 % de los expertos comentan que es posible, pero se deben considerar varios factores como: la aceptación de los desarrolladores que tienen que analizar si existe o no una mejora en la utilidad y en el ahorro del trabajo; considerar si la curva de aprendizaje puede demorar y complicar los tiempos de entrega; conseguir que los miembros del equipo se sumen a la estrategia; el alcance del proyecto; que tan comenzado se encuentra el proyecto y su estado de “salud” (si las entregas y avances se han realizado en tiempo y forma) por mencionar los principales factores registrados.

Sobre la utilidad que puede aportar una matriz de decisión en la identificación de pasos e instrumentos del proceso propuesto que se pueden emplear y cuáles omitir en el desarrollo del software. La mayoría de los expertos coinciden en que el uso de una matriz es fundamental para la estrategia del desarrollo de software. Puntualizan que la matriz depende del tipo de proyecto que se esta trabajando, dado que en algunos proyectos se omite ese instrumento durante su desarrollo.

En el aspecto del grado del nivel de experiencia que requiere un analista para emplear los artefactos propuestos. El 100 % de los expertos indican que son adecuados para analistas principiantes, dado que les permite conocer y aplicar artefactos y técnicas de levantamiento de requerimientos diferentes a las tradicionales. Comentan que estos artefactos les pueden facilitar las tareas que se presentan en la ingeniería de requerimientos y que promueven el involucramientos de los interesados en el proceso de desarrollo del software.

Respecto a la utilidad que los instrumentos propuestos pueden proporcionar a los analistas nuevos (que van iniciando en la dirección de proyectos de software). Los expertos coinciden en que los instrumentos pueden ser empleados por los analistas, pero es requerido una capacitación y ejemplos de cada instrumento para conocer la

escencia del proceso y tener un mejor criterio en selección de los artefactos que mejor se acoplen a sus estrategias de desarrollo.

La pregunta relacionada con la capacidad de flexibilidad que tiene el proceso propuesto para omitir o adaptar artefactos dependiendo del proyecto y de la metodología de desarrollo de software seleccionado. El 100 % de los expertos indican que es lo suficientemente flexible para emplear con diferentes metodologías de desarrollo de software, omitiendo artefactos dependiendo de los necesidades del desarrollo. Dos participantes comentan que la mejor forma de medir la flexibilidad es ponerlo en práctica en varios proyectos para analizar la tasa de error y registrar los casos de éxito.

Los expertos opinan que es un proceso perfectible, dado que algunos comentarios están dirigido en incluir una matriz de decisión durante del proceso de ejecución para hacerlo más comprensible. Así como una guí para los nuevos analistas lo puedan realizar el proceso como está plenteado.

4.5.4 *Cierre del proceso Delphi*

El proceso de encuestas empleadas en la técnica Delphi demoró dos meses, principalmente por el tiempo de respuesta de los diferentes expertos y del análisis de las respuestas proporcionadas.

De las observaciones de los expertos se puede concluir que el proceso propuesto es valido para aplicarse en las tareas de levantamiento de requerimientos de los proyectos de software. Esto se debe a que con las respuestas proporcionadas por las 2 rondas de preguntas, se obtiene que los expertos perciben que el proceso cumple con flexibilidad, robustez, adaptación y facilidad de aplicación.

Los expertos comentan que el proceso se puede ajustar a metodologías de desarrollo de software sin realizar cambios significativos, dado la flexibilidad que proporcionan los pasos y los artefactos documentados. Los expertos agregan comentarios favorables para aplicar el proceso en la dirección de proyectos de software, indicando que los artefactos pueden aportar mejoras en la recolección y análisis de los requerimientos de software.

De los comentarios y opiniones registrados ningún experto solicitó algo parecido a realizar cambios en la descripción del proceso propuesto. Respecto a la inclusión de una matriz de decisión para seleccionar que artefactos y fases del proceso propuesto se pueden utilizar durante el desarrollo de software, esta sujeto a los requerimientos del software y a la estrategia de dirección del proyecto de cada administrador de proyecto

o encargado del desarrollo. Por esta razón no se incluye como un artefacto dentro del proceso, dado que es considera como un elemento de análisis y estrategia que depende de cada administrador de proyectos emplear y documentalmente no es empleado por todos los expertos en la dirección del proyecto y cuando es utilizado, siempre depende del contexto del proyecto.

Como cierre al proceso de validación por la técnica Delphi, el experto en estadística consultado para las mediciones (Dr. Mora Gutiérrez Roman), considera que los resultados arrojados de las dos rondas de preguntas a expertos son suficientes para cubrir los aspectos de validación del proceso propuesto. En adición indica que realizar más rondas no presentaría un aporte estadístico significante. Por motivo la validación del proceso se obtuvo mediante esas rondas junto con sus respectivas revisiones.

En la tabla 5 se encuentra un resumen de las opiniones de los expertos sobre los aspectos: sencillo de comprender, sencillo de ejecutar, robusto, flexible, y eficiente, que consideran sobre el proceso propuesto.

Tabla 5: *Aspectos que cumple el proceso propuesto desde la opinión de los expertos*

E	Sencilla de comprender	Fácil de implementar	Robusta	Eficiente	Flexible
E1	✓	✓	✓	✓	✓
E2	✓	✓	✓	✓	✓
E3	✓	✓	✓	✓	✓
E4	✓	✓	✓	✓	✓
E5	✓	✓	✓	✓	✓
E6	✓	*	*	✓	✓

Tabla 5: *Aspectos que cumple el proceso propuesto desde la opinión de los expertos*

E	Sencilla de comprender	Fácil de implementar	Robusta	Eficiente	Flexible
E7	✓	✓	*	✓	*
E8	✓	✓	-	✓	✓
E9	✓	✓	-	✓	✓
E10	✓	✓	*	✓	✓

✓Considera que cumple con ese aspecto.

* El experto considera que cumple con ese aspecto, pero recomienda ponerla en práctica para verificar.

- No hace una mención directa sobre ese aspecto.

5

CONCLUSIONES

En este trabajo se propone una solución empleando artefactos esquemáticos a problemáticas en el levantamiento de requerimientos de software, considerando aspectos relacionados al diseño centrado en el usuario y la experiencia de usuario. A lo largo del trabajo se presentaron las diferentes dificultades que ocurren en el proceso de ingeniería de requerimientos, convirtiendola en una etapa crítica en el éxito o fracaso de un producto de software. En esta etapa llevar acabo un mal levantamiento de requerimientos se asocia a: un mal empleo de las técnicas de levantamiento de requerimientos; un proceso sin planificación ni estrategia; un proceso inconcluso o con huecos en los detalles de los requerimientos; un proceso con comunicación deficiente con los interesados y clientes; por mencionar algunos casos representativos.

El software resultante de una deficiente etapa de levantamiento de requerimientos es reflejo de una serie de errores y fallas en la implementación, que concluye en la mayoría de los casos en un software no funcional, que no coincide con lo esperado por el cliente y los usuarios finales. El proceso propuesto aborda estas dificultad, empleando artefactos para facilitar el entendimiento y socialización de los requerimientos del software con los miembros del equipo de desarrollo y de diseño.

Dentro de las principales dificultades en la ingeniería de requerimientos, los administradores de proyectos se enfrentan a la tarea de comprender y analizar los requerimientos de software, siendo una tarea clave en el éxito del proceso. No obstante la tarea no termina ahí, por lo que el proceso propone emplear esquemas que permitan a los administradores del proyecto, aterrizar y socializar los requerimientos de software adecuadamente con los miembros del equipo (Scrum Team). . En esta tarea el analista (o encargado del equipo de dicha tarea) debe explicar lo que el cliente requiere y el contexto en el que se empleará el sistema. Ejemplificar de ser necesario con escenarios y tareas el funcionamiento del producto esperado, haciendo énfasis en las funcionalidades que el cliente solicita. En este caso una variante del costumer journey tradicional, que agrega las anotaciones de las entradas, actividades y salidas que un proceso al cual se busca desarrollar su versión automatizada

En el actual trabajo se describe un proceso de apoyo conceptual durante el levantamiento de requerimientos durante el desarrollo de software. En el proceso se consideran aspectos del diseño centrado en el usuario y artefactos que permiten a los miembros del equipo de trabajo recabar, analizar y transmitir adecuademente los requerimientos de software. El proceso emplea como base de dirección de proyectos de software un enfoque ágil con el uso del framework Scrum. En el cumplimiento del objetivo de investigar e identificar aspectos del levantamiento de requerimientos asociados al software se consiguió a través de la revisión de artículos relacionados con la ingeniería de requerimientos y de libros de compendio de conocimientos (como PMBOK y BABOK) de la dirección de proyectos de software. El entedimientos de estos aspecto se resumen en una tabla que aborda los rasgos principales de las técnicas de levantamiento de requerimientos consultadas para el trabajo de investigación. Sobre los componentes y recomendaciones del diseño de experiencia de usuario que se aplican al proceso de levantamiento de requerimientos. Este objetivo se desarrollo investigando artefactos que se emplean en el diseño de experiencia de usuario, como son los artefactos Persona, mapa de empatía, mapa de experiencia del cliente y el FODA. Esta investigación propone una variante de aplicación de estos artefactos en el proceso de levantamiento de requerimientos, facilitando el entendimientos de los requerimientos, evitando la ambiguedad y socializando los requerimientos adecuadamente a los miembros del equipo de trabajo (Scrum Team). En el objetivo de identificar y entender el proceso que se sigue en la etapas del levantamiento de requerimientos de software en un enfoque ágil de desarrollo. Se abordó documentando y resumiendo procesos de desarrollo de software apegados al manifiesto ágil. En particular se integra al proceso propuesto el framework Scrum. Desde el surgimiento de la idea de este proceso hace aproximadamente 5 años, ha demostrado ser muy eficiente y flexible en el desarrollo de software, dado que se acopla al contexto de trabajo con equipos reducidos, entregas constantes de avances del proyecto, la filosofía de aceptación a cambios y verificación de la calidad de software, solo por mencionar los aspectos importantes de su incoporación. Este proceso con Scrum se enfoca en equipos de trabajo reducidos que es un fenómeno común en la industria y academia en México y en el éxito que se ha obtenido para acoplarse al desarrollo y se incorporará en el actual trabajo.

Respecto a los mecanismos de verificación del avance de los requerimientos y desarrollo del producto de software, el proceso se apega a lo propuesto por Scrum de dar seguimiento de los requerimientos mediante una tabla que lleva registro del avance por cada ciclo de desarrollo (sprint). Como un beneficio extra en dicho seguimiento el proceso propuesto incorpora el artefacto Kanban que emplea columnas etiquetadas

indicando en que estado de implementación se encuentra el requerimiento y el responsable de la implementación.

Uno de los principales objetivos de este trabajo es abordar el problema del entendimiento de los requerimientos de software solicitados por el cliente y transmitirlos (socializarlos) adecuadamente entre los miembros del equipo sin ambigüedades ni huecos en el proceso. Para cumplir con este objetivo el proceso incorpora el artefacto persona con la descripción de escenarios del uso particular que se desea automatizar con el software. Además del mapa de experiencia del cliente que permite entender el problema a solucionar, a través de la descripción con iconos o imágenes representativas del requerimiento solicitado por el cliente empleando un contexto específico en un escenario de uso de la situación a solucionar con el software. El empleo de estos artefactos en varios proyectos de software a permitido socializar adecuadamente los requerimientos de software entre los miembros del equipo y con ello reducir la ambigüedad de los requerimientos entre los miembros del equipo. En la evaluación del proceso propuesto los expertos coincidieron que emplear los artefactos incorporados en el trabajo ayudan en el desarrollo de software y permiten una mejor comunicación entre los involucrados en la implementación del producto de software.

El proceso propuesto se ha empleado en varios desarrollos de software. Dentro de la UNAM en el desarrollo de la gestión de planes y programas de estudios de posgrado en la Unidad de Posgrados sede CU. En el proceso de inscripción y generación de plantillas de asignaturas para alumnos de la Facultad de Ciencias Políticas y Sociales modalidad SUAyED sede CU, en la misma dependencia se empleó en el sistema de soporte de tickets de apoyo para los estudiantes de la Facultad. En desarrollos fuera de la UNAM se ha empleado en el desarrollo del prototipo funcional de un sistema de gestión para la aplicación de quimioterapia en pacientes con cáncer de mama inscritos en el programa del seguro popular del Instituto Nacional de Ciencias Médicas y Nutrición Salvador Zubirán y en el prototipo funcional del sistema de gestión de prueblos mágicos, una vinculación entre el Instituto Politécnico Nacional, CONACYT e ICTUR. En el trabajo a futuro el proceso propuesto será empleado en el desarrollo de software que permita llevar los tratamientos de quimioterapias y gestión de los medicamentos de la farmacia en el Instituto Nacional de Ciencias Médicas y Nutrición Salvador Zubirán. Siguiendo en la línea de software para la salud, el proceso será aplicado en el desarrollo de sistemas de pacientes de psiquiatría y administración de la revista de divulgación científica del Hospital Psiquiátrico Fray Bernardino Álvarez.

Para fines de esta investigación se decidió no evaluar el proceso mediante la implementación y documentación de un desarrollo de software utilizando las etapas del

proceso propuesto, debidó a los tiempos que pueden llegar a tomar los desarrollos de software con los tiempos propuestos por el programa de posgrado del doctorado. En su lugar para evaluar el proceso se optó por la técnica Delphi con expertos, dado que la investigación coincide con varios aspectos que la hace compatible para dicha variente como: la interacción efectivamente a distancia de los diferentes expertos, las reuniones rémotas para la evaluación del proceso, preservar la heterogeneidad del grupo para asegurar la validez de los resultados y la posibilidad de incorporar diferentes juicios y observaciones de los expertos que permiten mejorar el proceso propuesto.

De las observaciones y modificaciones expuestas por los expertos de la técnica Delphi, así de la mejora constante que ha tenido el proceso propuesto, se concluye que la hopótesis principal del trabajo se ha cumplido, dado que el proceso cumple que durante el proceso de desarrollo de software, en el cual la ingeniería de requerimientos cumple una paso fundamental para el éxito del proyecto, es una importancia fundamental establecer una estrategia adecuada y seleccionar las técnicas de levantamiento de requerimientos correctas al contexto del problema a solucionar con el software. Más aún el desarrollo y producto de software resultante se beneficia de procesos y estrategias que abordan aspectos de la usabilidad y experiencias de usuario, quienes realmente van a utilizar el software. Estos aspectos e instrumentos permitan una mejor comunicación entre los miembros del equipo en la parte operativa de los ciclos de desarrollo apegados a los requerimientos del software.

Como trabajo a futuro se plantea incorporar variantes de estos artefactos en otros procesos de igual importancia en el desarrollo de software. Particularmente en el proceso de control de calidad y mantemiento del desarrollo de software, esto es dada la importancia que ha cobrado la adquisición de software por pago de servicio, modalidad dirigida a dar una constante actualización del software para mantener la atención de los usuarios. El software por renta de servicio requiere de constante actulización, como apoyo en el éxito del software se propone diseñar y probar artefactos que permitan mejorar el entendimiento de las modificaciones entre los miembros del equipo. Apuntando a la mejora continua, es decir, intentar optimizar y aumentar la calidad de un producto, proceso o servicio a través de la evolución de los procesos.

A

ANEXO A. TÉCNICAS DE LEVANTAMIENTO DE REQUERIMIENTOS

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	Técnica
Tradicionales o clásicas	**Entrevista** Se emplea comúnmente en el levantamiento de requerimientos. El analista plantea preguntas a los interesados sobre la forma en la que realizan las tareas, sobre el sistema y procesos que utilizan actualmente, así como en los aspectos y funcionalidades del sistema a desarrollar, los requerimientos provienen de las respuestas a estas preguntas. Los resultados dependen de la experiencia del analista para dirigir la sesión en los temas principales. Existen 3 tipos de entrevistas: Estructuradas, no estructuradas y semi-estructuradas. El instrumento con las preguntas se pueden elaborar en herramientas digitales como Google Forms, Surveymonkey y Microsoft Forms, solo por mencionar algunos. **Ventajas** Permite recopilar grandes cantidades de datos. La entrevista se puede repetir para verificar los datos. Permite identificar conflictos con los interesados en el proyecto. Al ser presencial las ambigüedades se pueden aclarar. **Desventajas** Requiere que el analista tenga buenas habilidades comunicacionales. En entrevistas con enfoques muy formales los interesados puede sentirse incómodo y no revelar mucha información. No siempre es posible cubrir todos los aspectos del software.

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

<table>
<tr><th>-</th><th colspan="2">Técnica</th></tr>
<tr><td rowspan="3">Tradicionales o clásicas</td><td>No Estructuradas:
Se pueden emplear para explorar y comprender mejor el dominio del problema. Son el precursor de entrevistas más detalladas o estructuradas al identificar temas relevantes.</td><td>No estructuradas:
Se puede desviar la atención de temas más relevantes. El analista puede ser parcial al preguntar.
Estructuradas:
Pueden limitar la generación de nuevas ideas.</td></tr>
<tr><td colspan="2">Referencias: Zowghi and Coulin (2005); Gunda (2008); Aristizábal-Mejía and Torres-Moreno (2009); Sood and Arora (2012); Serna (2012); Sharma and Pandey (2013); Swarnalatha et al. (2014); Yousuf and Asger (2015);</td></tr>
<tr><td colspan="2">Cuestionarios
Consisten en un conjunto de preguntas abiertas y cerradas que se aplican a los interesados. Se emplean principalmente durante las primeras etapas para obtener los requerimientos y adquirir conocimientos del dominio del problema.Los cuestionarios deben ser claros, bien definidos y precisos, tener una buena planificación, cuidar los detalles y mencionar el plazo de tiempo en el que se deben devolver al analista.Se emplean cuando un mismo conjunto de preguntas las debe contestar un amplio número de participantes. Los analistas pueden emplear cuestionarios digitales empleando herramientas como Google Forms, Surveymonkey y Microsoft Forms, solo por mencionar algunos.

Ventajas
Es una manera eficiente de recopilar información de múltiples interesados al mismo tiempo. Permite recopilar grandes cantidades de datos.

Desventajas
Poca interacción con los interesados. Para que sean efectivas el analista debe comprender términos y conceptos del dominio del problema.</td></tr>
</table>

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

<table>
<tr><th>-</th><th colspan="2">Técnica</th></tr>
<tr><td rowspan="3">Tradicionales o clásicas</td><td>Se puede emplear como una lista de comprobación para verificar que se han abordando los temas principales. Creación y análisis de cuestionarios con herramientas digitales.</td><td>Las preguntas deben estar bien definidas y enfocadas para evitar obtener información redundante.No porporciona un mecanismo para que los participantes puedan aclarar un punto o corregir un malentendido.</td></tr>
<tr><td colspan="2">Referencias: Zowghi and Coulin (2005); Gunda (2008); Sharma and Pandey (2013); Yousuf and Asger (2015);</td></tr>
<tr><td colspan="2">Introspección
La introspección consiste en que el analista observa e imagina cómo debería ser el diseño del sistema a desarrollar. Se emplea en las etapas tempranas y de su análisis se obtienen requerimientos iniciales. Los analistas deben tener conocimientos en los temas y expertiz en el dominio del problema. Los resultados son mejores, si los analistas cuentan con conocimientos en los procesos comerciales realizados por los clientes.
Ventajas
Se obtienen los primeros requeirmientos. Costos bajos, sino se emplean expertos externos. Fácil de llevar acabo, si se cuenta con conocimientos del dominio del problema.
Desventajas
Si el analista no es experto, la información puede llegar a ser poco útil.Pueden quedar vacíos en los requerimientos. Las ideas de los analistas expertos pueden no reflejar lo que los interesados y clientes esperan del proyecto.
Referencias: Zowghi and Coulin (2005); Gunda (2008); Yousuf and Asger (2015);</td></tr>
</table>

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	Técnica
Tradicionales o clásicas	**Análisis de documentación** Consiste en que el analista revisa y recopila información de documentos existentes tales como manuales, reportes, diseño, plantillas y otras fuentes de información relacionadas a las organización y sus procesos. De la información recolectada se obtienen los primeros requerimientos del proyecto los cuales varian dependiendo de los fuentes disponibles. Como técnica complementaria permite obtener una adecuada comprensión de la organización y de los términos que se emplean. Se puede utilizar para crear instrumentos de otras técnicas que se empleraran en el proceso, como entrevistas o cuestionarios **Ventajas** Es una técnica económica.Permite a los analistas estudiar el dominio del problema. Se puede emplear cuando las partes interesadas y los analistas no están disponibles. **Desventajas** La documentación difícilmente refleja la forma en que realmente se desarrollan las actividades en la organización. Consume mucho tiempo en encontrar información útil. Referencias: Yousuf and Asger (2015); Zowghi and Coulin (2005); Mrayat et al. (2013); Sharma and Pandey (2013); Serna (2012)
	Taller de requerimientos Los interesados se reúnen (promedio 6 a 10 personas) en varias sesiones, para discutir experiencias y opiniones sobre el sistema a desarrollar, así como revisar las características de alto nivel del sistema deseado, de ahí se obtienen los requerimientos. Pueden ser de tipo colaborativas multidisciplinarias y de tipo creativo: la primera consiste en involucrar a interesados de diferentes áreas del negocio y la segunda al fomentar expresiones dirigidas a la innovación.Los temas son dirigidos por un moderador por lo que los resultados dependen de su experiencia para dirigir la sesión.

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	Técnica
Conversacionales: Colaborativas y Grupales	**Ventajas** Es una manera rápida para conocer la percepción de los interesados acerca de un tema o concepto particular. Ayuda en descubrir y comprender requerimientos conflictivos.Promoven la creatividad. Útil cuando los interesados tiene diferente ubicación geográfica **Desventajas** Sesiones poco productivas si el moderador es inexperto Información sesgada si la participación no es equilibrada entre los involucrados. Pueden durar varias horas. Referencias:Aristizábal-Mejía and Torres-Moreno (2009); Yousuf and Asger (2015); Zowghi and Coulin (2005); Swarnalatha et al. (2014); Mrayat et al. (2013); Serna (2012)
	Lluvia de ideas Consiste en reuniones informales entre cuatro y diez personas de la parte interesada y los analistas. Como dinámica se sugieren toda clase de ideas sin centrarse en nadie en particular, ni juzgar su validez, después de recolectarlas se realiza un análisis detallado de cada propuesta respecto al desarrollo del software. Es importante evitar explorar o criticar ideas con gran detalle. Los participantes deben ser de diferentes áreas de conocimiento y cada miembro cuenta con un límite de tiempo para compartir sus ideas, la participación es actividad y no requieren que los participantes sean altamente calificados. Se puede emplear en proyectos de actualización de sistemas.

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	Técnica	
Conversacionales: Colaborativas y Grupales	**Ventajas** Útil para identificar requerimientos que no son claros. Permiten identificar cuando se estan agregando funcionalidades. Se propicia la participación de los invoucrados. Promueve la creatividad. Es de costos bajos. Útil para resolución de conflictos entre interesados. Fácil de implementar	**Desventajas** Sesiones poco productivas si el moderador es inexperto. No es recomendable para problemas complejos. Si la organización y dirección no es organizada puede tomar demasiado tiempo. Se pueden repetir ideas. Se puede pensar que un gran cantidad de ideas, significa una mejor calidad. Algunos participantes se pueden presentar dificultades de expresar sus ideas.
	Referencias: Aristizábal-Mejía and Torres-Moreno (2009); Yousuf and Asger (2015); Zowghi and Coulin (2005); Swarnalatha et al. (2014); Mrayat et al. (2013); Sood and Arora (2012)	
	Grupo de trabajo Consiste en reuniones entre los interesados y los analistas con el objetivo de obtener los requerimientos del proyecto. La sesión es dirigida por un moderador, donde cada requerimiento debe ser discutido a profundidad aportando sugerencias.El moderador mantiene los grupos enfocados y alienta a los participantes a interactuar abiertamente. Los puntos de vista y las opiniones discutidas se registran simultáneamente, lo que hace que la participan activa de los miembros permite obtener requerimientos válidos.	
	Ventajas Útil para identificar requerimientos que no son claros.Permiten identificar cuando se estan agregando funcionalidades Se propicia la participación de los invoucrados.Útil para resolución de conflictos entre interesados Promueve la creatividad	**Desventajas** Sesiones poco productivas si el moderador es inexperto Es engorroso y es difícil reunir a todas las partes interesadas.Información sesgada o incompleta si la participación no es equilibrada entre los involucrados.Demanda mucho esfuerzo y tiempo.

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	Técnica
Observacionales o contextuales	Referencias: (Yousuf and Asger, 2015; Zowghi and Coulin, 2005; Swarnalatha et al., 2014; Mrayat et al., 2013); (Gunda, 2008); (Serna, 2012)
Observacionales o contextuales	**Observación** En la literatura se puede encontrar como análisis social. Consiste en que el analista observa el entorno del usuario sin interferir en su trabajo. Se utiliza cuando los interesados no son capaces de explicar las necesidades y funcionalidades que requieren, así como para formarse una idea de cómo los usuarios interactuarán con el sistema.Para mejores resultados se recomienda que los analistas sean expertos y sepan qué buscar y cómo evaluar la relevancia de lo que observan.Existen dos variantes de observación la pasiva y la activa. La primera es cuando el analista no interactúa con el usuario, mientras que la activa es cuando se realizan preguntas al usuario durante el proceso. **Ventajas** Permite comprobar los pasos que toma una tarea o proceso. Ayuda a identificar eventos críticos no observados por otras técnicas.Permite medir los tiempo de las tareas que se realizan.Permite obtener requerientos útiles.Útil para comprobar requerimientos obtenidos de otras técnicas. Ayuda a descubrir características en un período de tiempo más corto. **Desventajas** Si el analista no es experto, puede que no sepa que buscar o como evaluar lo que observa. La presencia de los analistas puede incomodar la labor de los usuarios y sesgar el resultado.Los usuarios pueden omitir pasos que son relevantes en el análisis Pueden llevar varias sesiones Puede llegar a se una técnica costosa Es un proceso lento Referencias: Aristizábal-Mejía and Torres-Moreno (2009); Yousuf and Asger (2015); Zowghi and Coulin (2005); Mrayat et al. (2013); Gunda (2008)Serna (2012); Zhang (2007)

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	Técnica
Observacionales o contextuales	**Etnografía** En esta técnica los analistas observan las actividades de las personas que realizan las tereas, durante un período de tiempo determinado. El analista se involucrar en el entorno organizacional para entender y recolectar los requerimientos. Se centra principalmente en los usuarios finales. En la práctica es efectiva cuando se emplea en proyectos que surgen de problemas existentes con procesos y procedimientos de la organización y que dependen de la identificación de patrones sociales y relaciones complejas entre los involucrados. Se utiliza en combinación con otras técnicas como las entrevistas y cuestionarios para obtener respuestas más detalladas. **Ventajas** Permite comprobar los pasos que toma una tarea o proceso. Permite medir los tiempo de las tareas que se realizan. Permite obtener requerientos útiles.Útil para abordar la usabilidad del sistema. Ayuda a descubrir características en un período de tiempo más corto. Útil para comprobar requerimientos obtenidos de otras técnicas. **Desventajas** No existe una guía detallada de cómo realizar la técnica con eficacia.Puede llegar a se una técnica costosa. Suele requerir una gran cantidad de tiempo para obtener los requerimientos Las características nuevas y únicas que se desean incorporar al sistema podrían no ser descubiertas. Referencias: Aristizábal-Mejía and Torres-Moreno (2009); Yousuf and Asger (2015); Zowghi and Coulin (2005); Mrayat et al. (2013); Sharma and Pandey (2013); Gunda (2008) Serna (2012); Zhang (2007)

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	Técnica
Técnicas Analíticas	**Análisis de protocolos** En el análisis de protocolos el analista observa como los participantes realizan una actividad o tarea mientras las describen en voz alta, así como acciones que se llevan a cabo o pensamientos detrás de ellas. Esta técnica puede proporcionar al analista información específica sobre los procesos que el sistema debe realizar.Se centra principalmente en los usuarios finales. Es una técnica efectiva cuando se emplea en proyectos que surgen para solucionar problemas relacionados con procesos y procedimientos de la organización. **Ventajas** Permite comprobar los pasos que toma una tarea o proceso. Permite medir los tiempo de las tareas que se realizan.Permite obtener requerientos útiles. Ayuda a descubrir características en un período de tiempo más corto. Útil para comprobar requerimientos obtenidos de otras técnicas. Permite marcar el flujo de trabajo. **Desventajas** Hablar o describir una operación mientras se realiza no es la forma normal de realizar la tarea y puede distraer al operador. Puede no representar necesariamente el proceso correcto.Los operadores suelen dar por sentados los pasos repetitivos o menos que se escapan de la vista del analista. Referencias: Yousuf and Asger (2015); Zowghi and Coulin (2005); Mrayat et al. (2013); Serna (2012); Zhang (2007)
	Reutilización de requerimientos Consiste en que los analistas usen el conocimiento existente para desarrollar un nuevo producto. Cada proyecto tiene sus propios interesados por lo que se deben incluir en el análisis. Un cantidad considerable de requerimientos para un nuevo proyecto se pueden adquirir de los proyectos análogos existentes. Es necesario verificar los requerimientos antes de que se utilicen en el producto propuesto, de tratarse de proyectos análogos el tiempo se reduce. Se puede emplear en proyecto de actualización de sistemas.Se reutilizan glosarios, planes de control de cambios, manejo de riesgos, lecciones aprendidas, entre otros.

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	Técnica
Técnicas Analíticas	**Ventajas** En general tiene un bajo costo. Si el analista es experto puede llevar menor tiempo Se pueden obtener requerimientos útiles validados Propicia una cultura organizacional y documental de la empresa. Se puede seguir la línea gráfica de la organización. Propicia las buenas prácticas de programación. Se reutilizan planes directores de proyectos análagos. **Desventajas** Se pueden descuidar las ideas y necesidades de los interesados del proyecto. Es una práctica complicada en empresas con una mala o nula cultura organizacional.Se pueden seguir cronogramas con tiempos ficticios o poco analizados. Referencias: Mrayat et al. (2013); Gunda (2008); Serna (2012); Zhang (2007)
	Escalamiento Es una técnica basada en entrevista para obtener los objetivos y atributos de los interesadas. A partir de los principales atributos del producto se identifican los usuarios clave. Con la ayuda de atributos principales, el entrevistador profundiza para extraer más información de los usuarios clave sobre los criterios y necesidades.Las preguntas y respuestas se organizan según las prioridades.Los requisitos demasiado grandes dificultan la realización de modificaciones, adiciones o eliminaciones.Buena opción para proyectos de actualización y mantenimiento.

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	Técnica	
Técnicas Analíticas	**Ventajas** Se facilitan empleando herramientas tecnológicas Permite obtener requerimientos priorizados La reutilización de requerimientos ahorra tiempo y costos Permite produnfizar en el domino del problema Permite conocer los principales procesos involucrados en el proyecto.Promueve una cultura organizacional.	**Desventajas** Son laboriosas y complicadas. Puede llegar a ser una técnica costosa, al utilizar expertos externos. Pueden requerir una gran cantidad de tiempo para obtener los requerimientos Es complicado realizar modificaciones en la escalera.
	Referencias: Yousuf and Asger (2015); Zowghi and Coulin (2005); Mrayat et al. (2013); Sharma and Pandey (2013); Serna (2012); Zhang (2007)	
	Clasificación de tarjetas Es una técnica que consiste en obtener la información, objetivos y necesidades de los interesados al solicitarles que clasifiquen tarjetas con los nombres de las entidades de dominio del problema.Permiten agrupar y asociar las tarjetas según la perspectiva del interesado.La clasificación de tarjetas permite comprender el modelo mental del usuario, y explica la forma en que los usuarios agrupan, clasifican y priorizan las tareas antes de llevarlas acabo.	
	Ventajas Permite obtener una comprensión del modelo mental del usuario Es un proceso generalmente rápido Suele ser de costo económico. Útil para proporcionar una buena base.Propicia que la información esté estructurada.Permite conocer los tipos de entradas de datos de los usuarios.	**Desventajas** No es adecuado para proyectos complejos y grandes.Puede proporcionar resultados variables. Incluye principalmente características superficiales Las explicaciones detalladas reducen el valor de la técnica, por lo que las interacciones son limitadas.No proporciona mucha información sobre el contenido involucrado.

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	Técnica
Sintéticas	Referencias: Yousuf and Asger (2015); Zowghi and Coulin (2005); Mrayat et al. (2013); Sharma and Pandey (2013); Serna (2012); Zhang (2007)
	Repertorio cuadricular En esta técnica el analista representa los elementos de dominio en una matriz, donde pide a los interesados que asignen valores y atributos.Se pueden emplear entrevistas semiestructuradas que se aplican a las partes interesadas para identificar nombres de elementos del dominio, atributos y valores. Se obtiene una representación inicial de los requerimientos. Principalmente consiste en la categorización de elementos; donde se evalúa, compara y califica. Se considera poco eficaz para describir las características de los requerimientos complejos. **Ventajas** Se usa en combinación con otros métodos como entrevistas. Es más precisa en la captura de detalles que la clasificación de tarjetas. Permite identificar las diferencias y similitudes entre los elementos del dominio.Propicia la comprensión del dominio desde la perspectiva de los usuarios. **Desventajas** No es adecuado para proyectos complejos y grandes Es menos precisa que escalamiento Requerie mucho esfuerzo. Es un proceso lento. Puede llegar a ser una técnica costosa, al utilizar expertos externos. Referencias: Yousuf and Asger (2015); Zowghi and Coulin (2005); Serna (2012); Zhang (2007)
	Prototipado Es una técnica iterativa en el que se lanzan versiones del producto aportando valor y funcionalidad. En cada iteración se refinan los entregables según los comentarios de los interesados.Se emplean cuando los interesados no tiene una idea clara de lo que necesitan.Es útil en el desarrollo de nuevos sistemas. Proporcionar el diseño y línea gráfica básica de la interfaz de usuario y los modos de intereación. Los prototipos están diseñados principalmente en sistemas que tienen más interacciones con el usuario.

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	Técnica
Sintéticas	**Ventajas** Permite clarificar requerimientos. Se puede combinar con otras técnicas para mejorar los resultados. Propicia la participación del usuario durante el desarrollo. Permite la retroalimentación temprana de los interesados. Por lo general ahorra tiempo y costo en el desarrollo. **Desventajas** Se pueden descuidar los requerimientos relacionados con las funcionalidades internas. Los interesados se pueden acostumbrar a un mismo tipo de particular de sistema. Los interesados se pueden resistir a los cambios. La estimación del esfuerzo y el costo puede aumentar dependiendo del proyecto. Referencias: Aristizábal-Mejía and Torres-Moreno (2009); Yousuf and Asger (2015); Zowghi and Coulin (2005); Swarnalatha et al. (2014); Sharma and Pandey (2013)Sood and Arora (2012)
	JAD Las sesiones de JAD son talleres de colaboración que pueden durar varios días, del cual se obtiene un conjunto adecuado de requerimientos del usuario.Su objetivo es recopilar la mayor información en un período de tiempo corto, mediante una preplaneación que incluye la presencia de participantes claves.Los participantes comparten sus opiniones sobre lo que se hace y lo que se debe cambiar. Los roles de los participantes y los objetivos del sistema están predefinidos. Los participantes son elegidos con mucho cuidado.Los interesados y los usuarios siguen intercambiando las ideas hasta la obtención final de los requerimientos.

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	Técnica
	Ventajas Acelera el diseño del sistema. Propicia la generación de ideas y conduce a resultados creativos.Los elementos visuales (como maquetados o bosquejos) hacen más interectiva y productiva la sesión Puede aportar una mayor satisfacción del usuario respecto al proyecto.Propicia una buena comunicación entre las partes interesadas, analistas y expertos. **Desventajas** Requiere de mucha planificacióny esfuerzo Si no se planifica adecuadamente ocasiona un mal aprovechamiento del tiempo y recursos.Requiere de analistas capacitados Referencias: Aristizábal-Mejía and Torres-Moreno (2009); Yousuf and Asger (2015); Zowghi and Coulin (2005); Gunda (2008); Sood and Arora (2012); Serna (2012)
Sintéticas	**Escenarios** Esta técnica consiste en una representación de la interacción del usuario con el sistema.Se propoenen ejemplos del mundo real de cómo se utilizaría el sistema. Los escenarios se pueden usar cuando se obtienen los requerimientos iniciales.Se incluye una descripción completa de los procesos, desde el estado inicial, el flujo de eventos, actividades concurrentes, y el estado final. Es una técnica útil cuando se analiza el sistema desde el punto de vista del usuario. Para crear los escenarios el analista necesita una comprensión de las tareas realizadas por el sistema y la participación que tienen los usuario.Por lo general están escritos en un lenguaje natural.

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

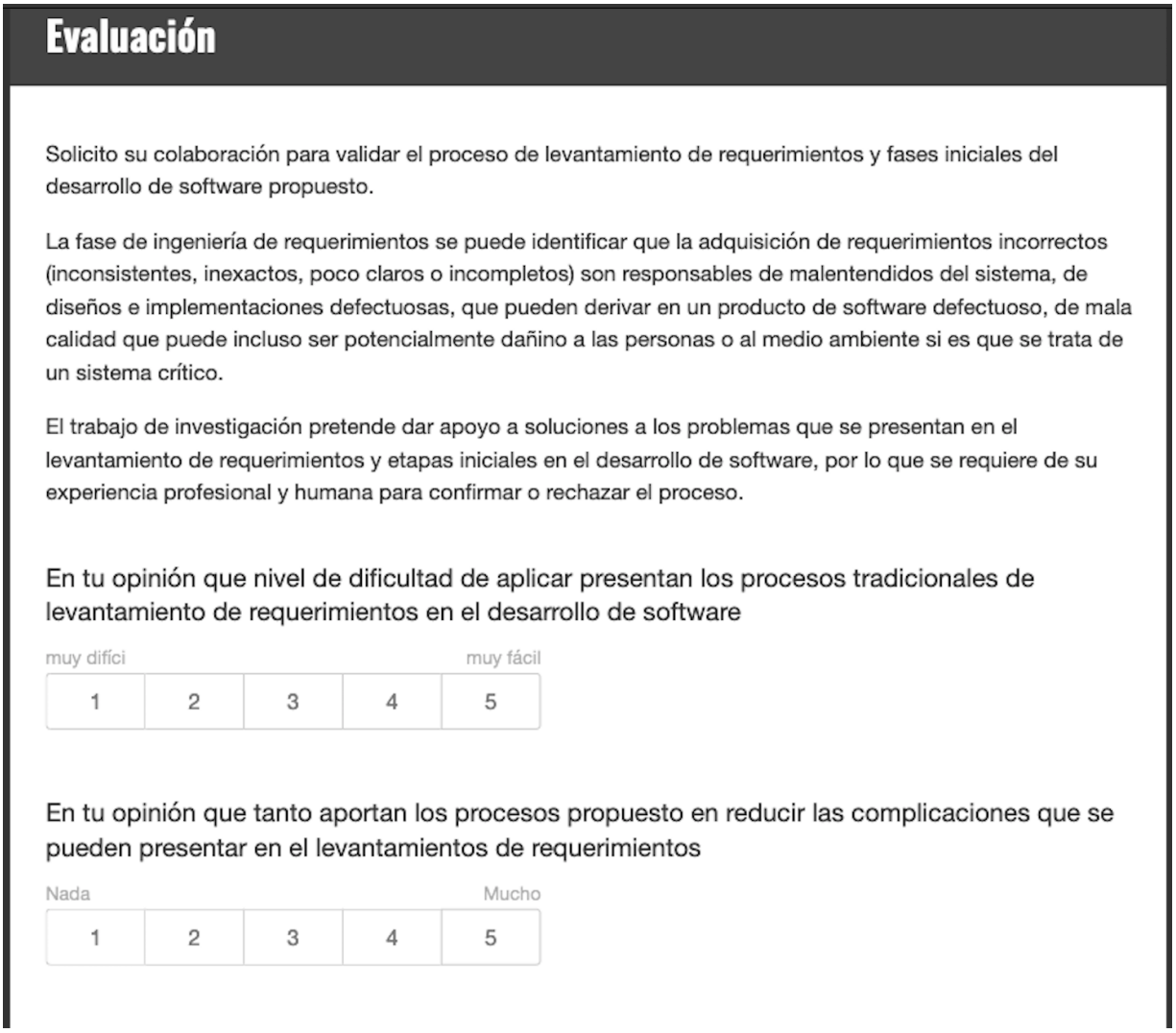
-	Técnica
Sintéticas	**Ventajas** Los escenarios se pueden emplear para validar los requerimientos. Se pueden emplear para crear casos de prueba. Permite identificar a los usuarios principales.Los resultados de la técnica son reutilizables en todo el proyecto.Las personas sin conocimiento técnico pueden también entenderlo.Fácil de entender ya que no se usa un lenguaje especial Al considerar la perspectiva del usuario se asegura una mayor usabilidad. **Desventajas** Es difícil crear escenarios útiles. No es adecuado para todo tipo de proyectos.No cubren todos los procesos.No proporciona una vista completa del sistema a desarrollar. Requiere de analistas capacitados La estimación del esfuerzo y el costo puede aumentar dependiendo del proyecto Referencias: Yousuf and Asger (2015); Zowghi and Coulin (2005); Swarnalatha et al. (2014); Mrayat et al. (2013); Gunda (2008); Sood and Arora (2012); Serna (2012)
	Caso de Uso La técnicas de casos de uso consta de actores, actividades (casos de uso) y relaciones entre ellos, según la definición de UML. Se usa para representar el entorno en el que se desenvuelven los actores (usuarios) y el alcance del sistema, a través de los casos de uso (requerimientos funcionales).La técnica permite representar al actor como un elemento externo al sistema (por ejemplo, un usuario u otro sistema) que interactúa con el sistema como una caja negra. Un caso de uso describe la secuencia de interacciones entre el sistema y sus actores cuando se ejecuta una acción. Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores.Para mejorar el entendimiento de los casos se uso se pueden agregar descripciones textuales o emplear una representación diagramática más detallada.

Tabla 6: *Técnicas de levantamiento de requerimientos de software*

-	**Técnica**
	Ventajas Permite identificar las actividades clave que son los requerimientos Permite identificar a los usuarios principales del sistema.Se pueden emplear para crear casos de prueba Considerar la perspectiva del usuario lo que asegura una mayor usabilidad.
	Desventajas Es difícil crear escenarios útiles. Es ambiguo cuando se definen requerimientos complejos.Requiere de analistas capacitados La estimación del esfuerzo y el costo puede aumentar dependiendo del proyecto
	Referencias: Sood and Arora (2012); Koch and Escalona (2004); Odeh et al. (2004)

B

ANEXO B. INSTRUMENTOS DELPHI EMPLEADOS



Evaluación

Solicito su colaboración para validar el proceso de levantamiento de requerimientos y fases iniciales del desarrollo de software propuesto.

La fase de ingeniería de requerimientos se puede identificar que la adquisición de requerimientos incorrectos (inconsistentes, inexactos, poco claros o incompletos) son responsables de malentendidos del sistema, de diseños e implementaciones defectuosas, que pueden derivar en un producto de software defectuoso, de mala calidad que puede incluso ser potencialmente dañino a las personas o al medio ambiente si es que se trata de un sistema crítico.

El trabajo de investigación pretende dar apoyo a soluciones a los problemas que se presentan en el levantamiento de requerimientos y etapas iniciales en el desarrollo de software, por lo que se requiere de su experiencia profesional y humana para confirmar o rechazar el proceso.

En tu opinión que nivel de dificultad de aplicar presentan los procesos tradicionales de levantamiento de requerimientos en el desarrollo de software

muy difíci — muy fácil

1	2	3	4	5

En tu opinión que tanto aportan los procesos propuesto en reducir las complicaciones que se pueden presentar en el levantamientos de requerimientos

Nada — Mucho

1	2	3	4	5

Figura 26: *Instrumento empleando en la técnica Delphi para la evaluación de la propuesta, Parte 1 [Imagen elaborada por el autor].*

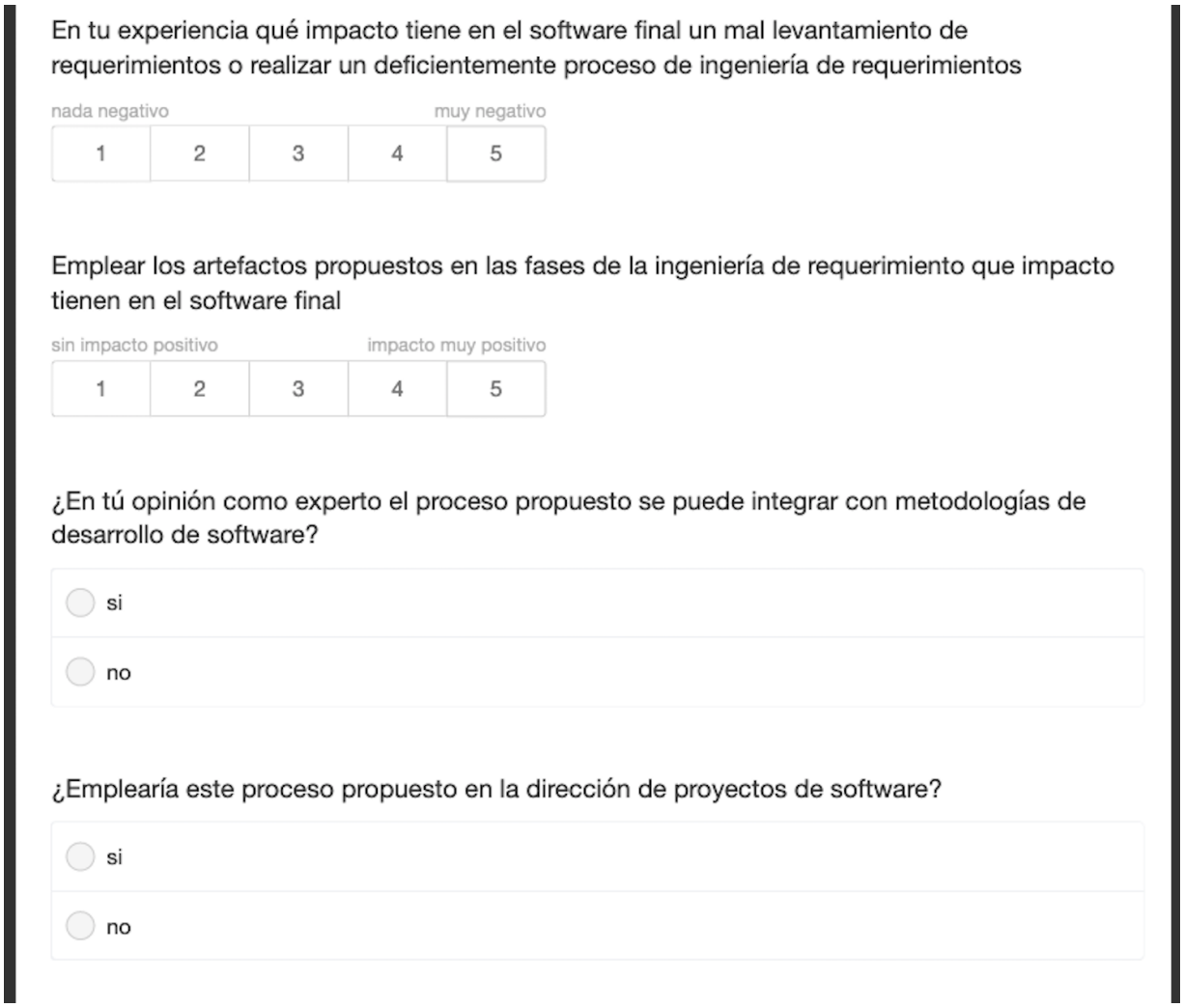


Figura 27: *Instrumento empleando en la técnica Delphi para la evaluación de la propuesta, Parte 2 [Imagen elaborada por el autor].*

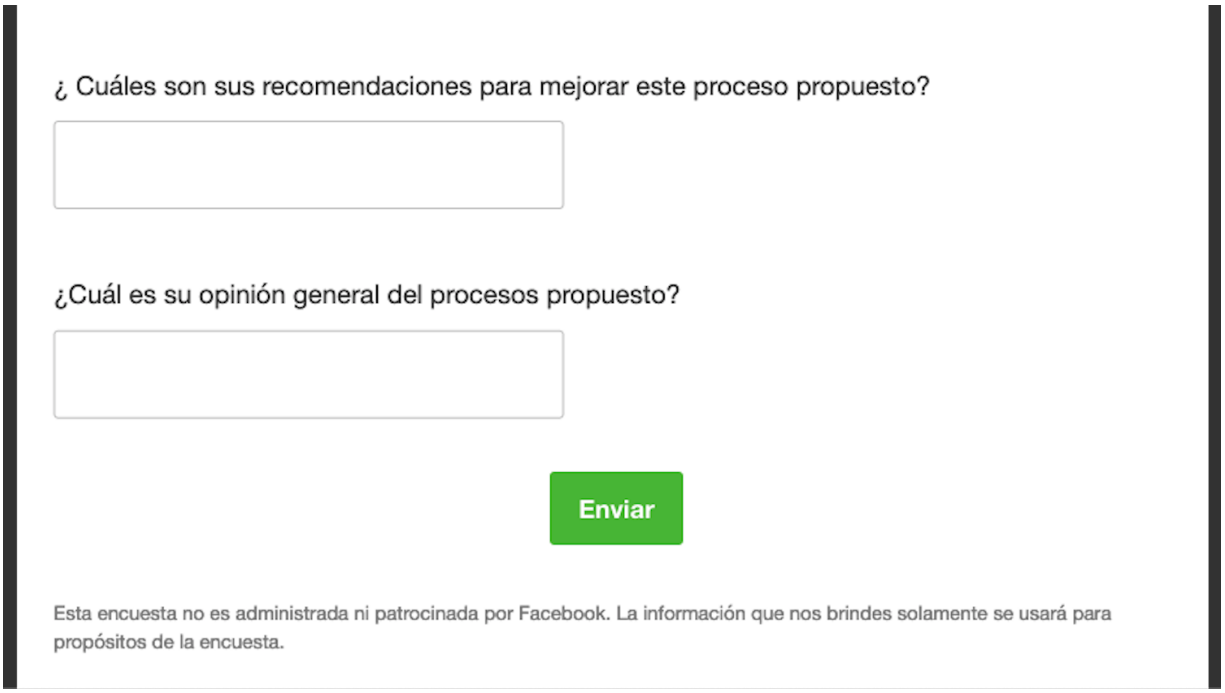


Figura 28: *Instrumento empleando en la técnica Delphi para la evaluación de la propuesta, Parte 3 [Imagen elaborada por el autor].*

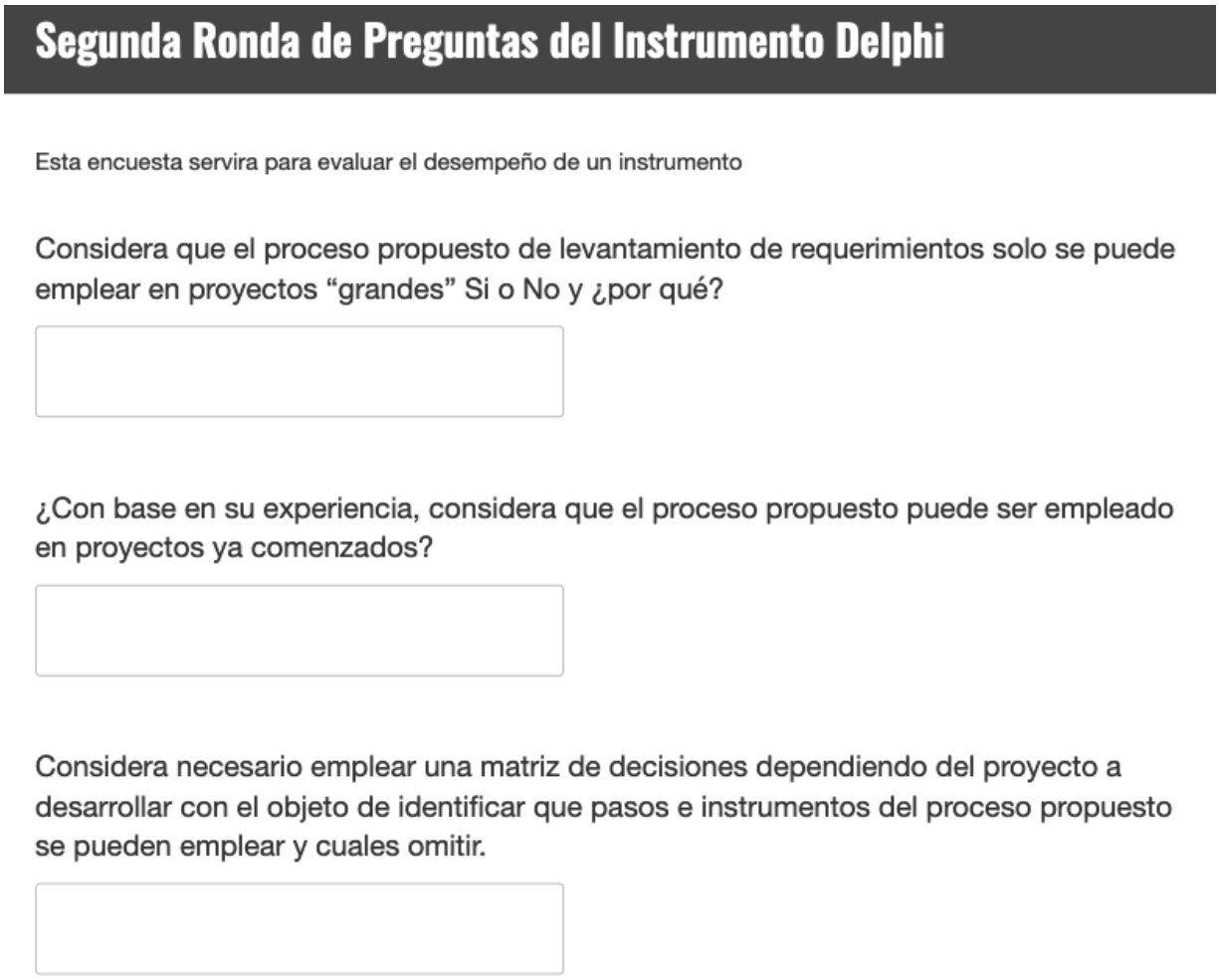


Figura 29: *Instrumento empleando en la técnica Delphi para la segunda evaluación de la propuesta, Parte 1 [Imagen elaborada por el autor].*

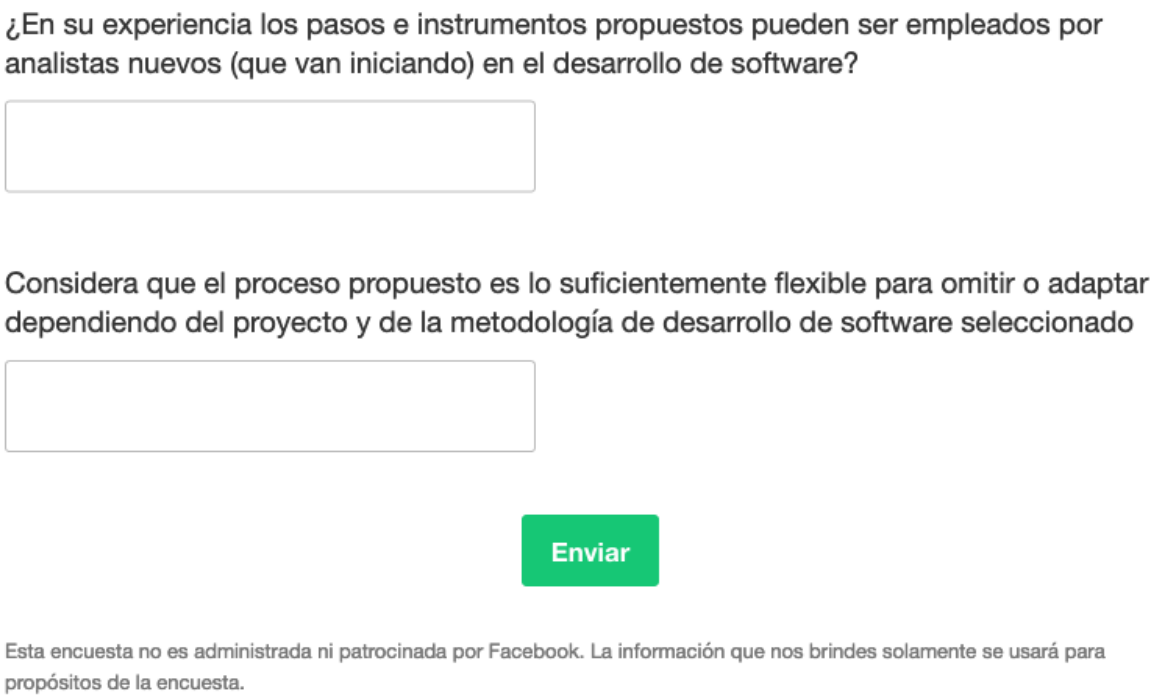


Figura 30: *Instrumento empleando en la técnica Delphi para la segunda evaluación de la propuesta, Parte 2 [Imagen elaborada por el autor].*

C

ANEXO C. RESPUESTAS DEL PANEL DE EXPERTOS

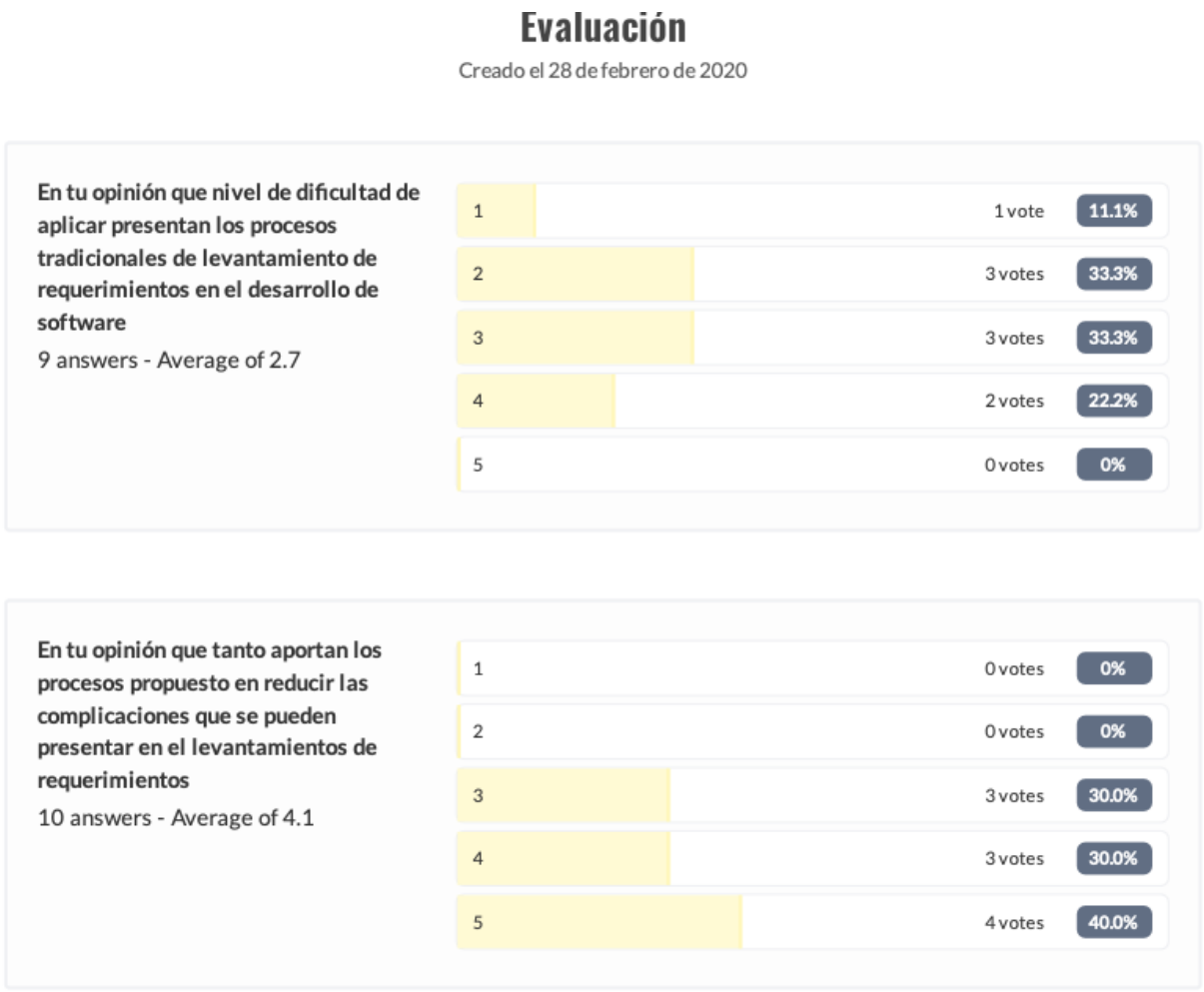


Figura 31: *Respuestas del panel de expertos del instrumento de la técnica Delphi para la evaluación de la propuesta, Parte 1. [Imagen elaborada por el autor]*

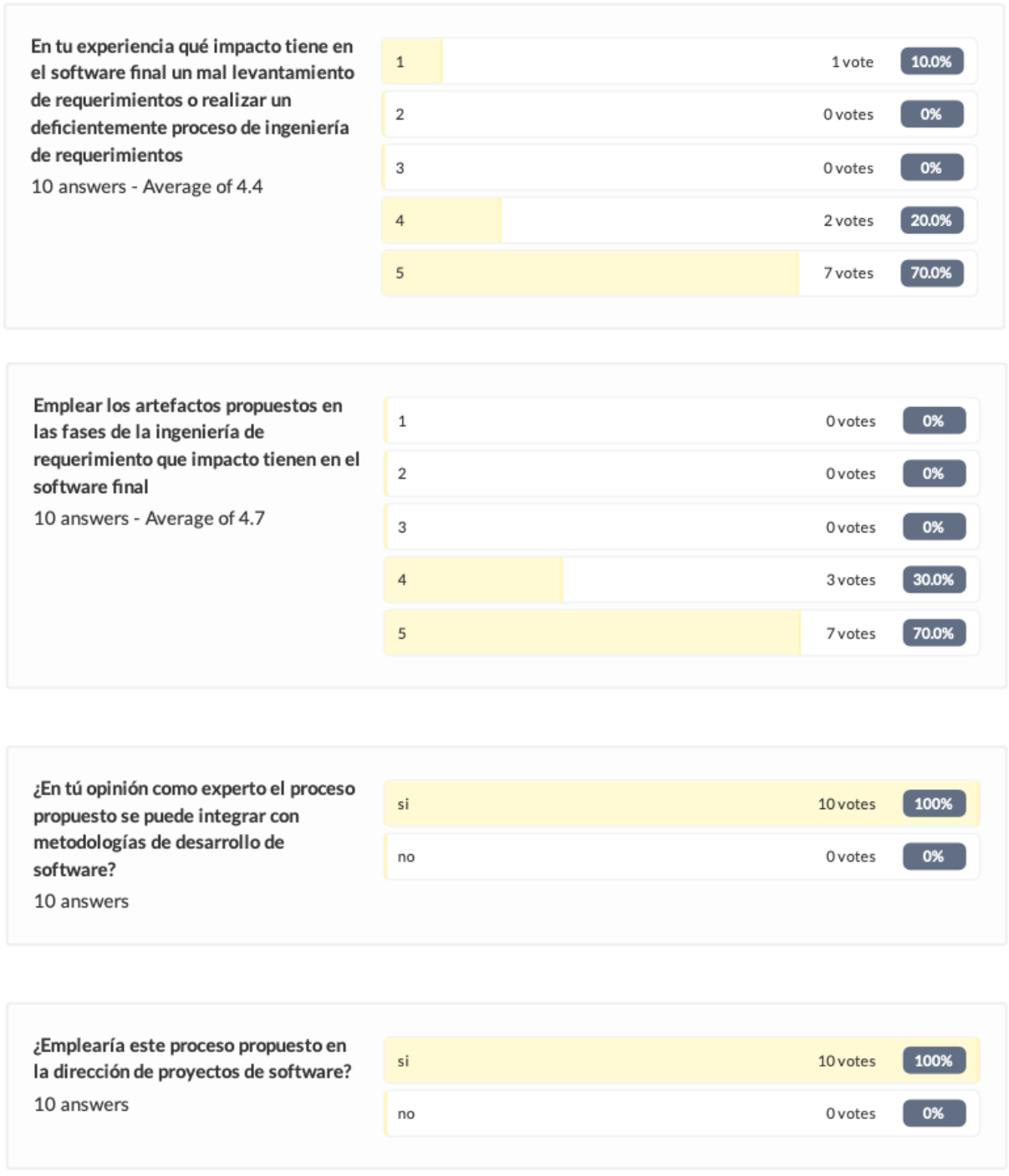


Figura 32: *Respuestas del panel de expertos del instrumento de la técnica Delphi para la evaluación de la propuesta, Parte 2. [Imagen elaborada por el autor]*

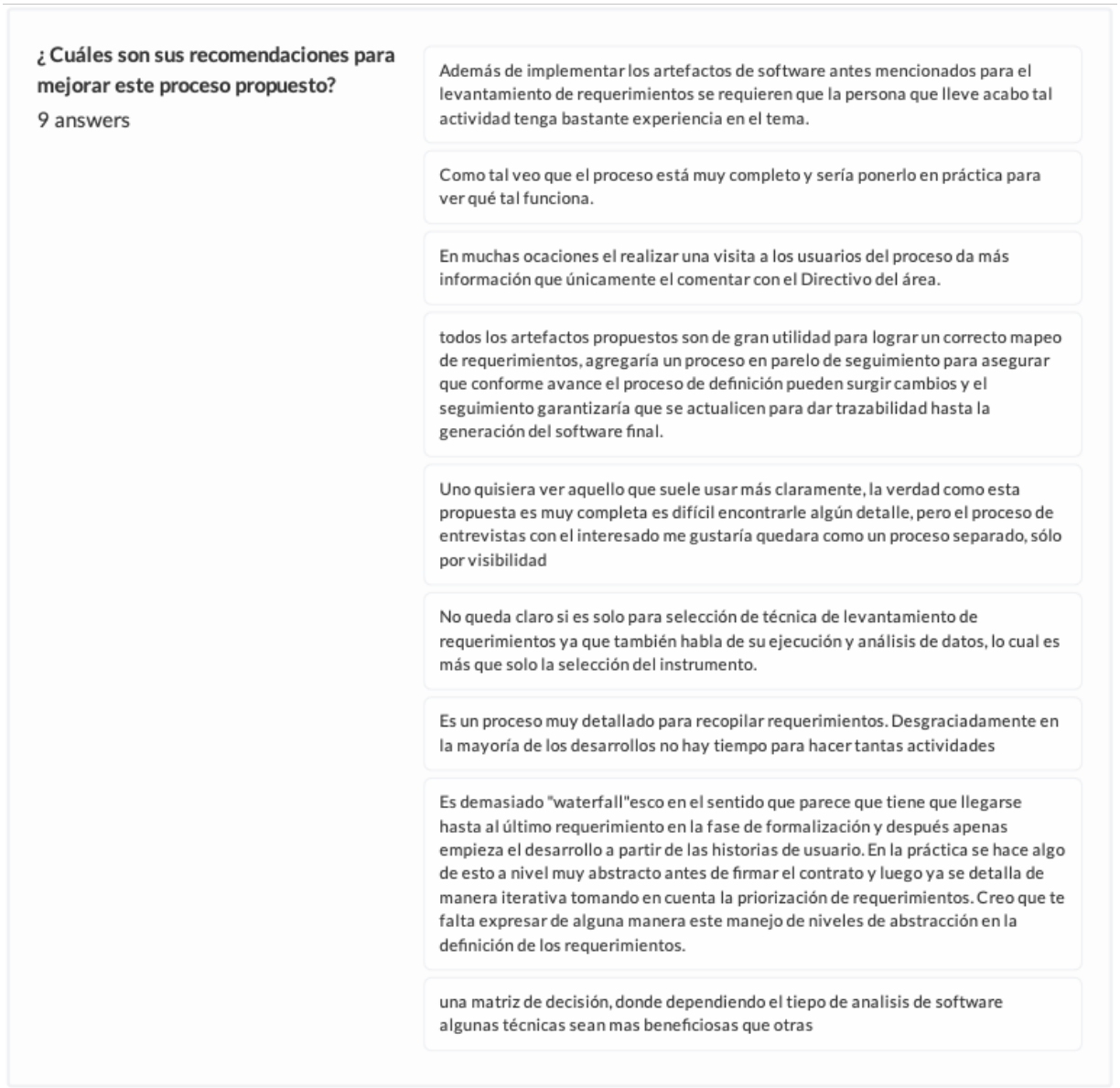


Figura 33: *Respuestas del panel de expertos del instrumento de la técnica Delphi para la evaluación de la propuesta, Parte 3. [Imagen elaborada por el autor]*

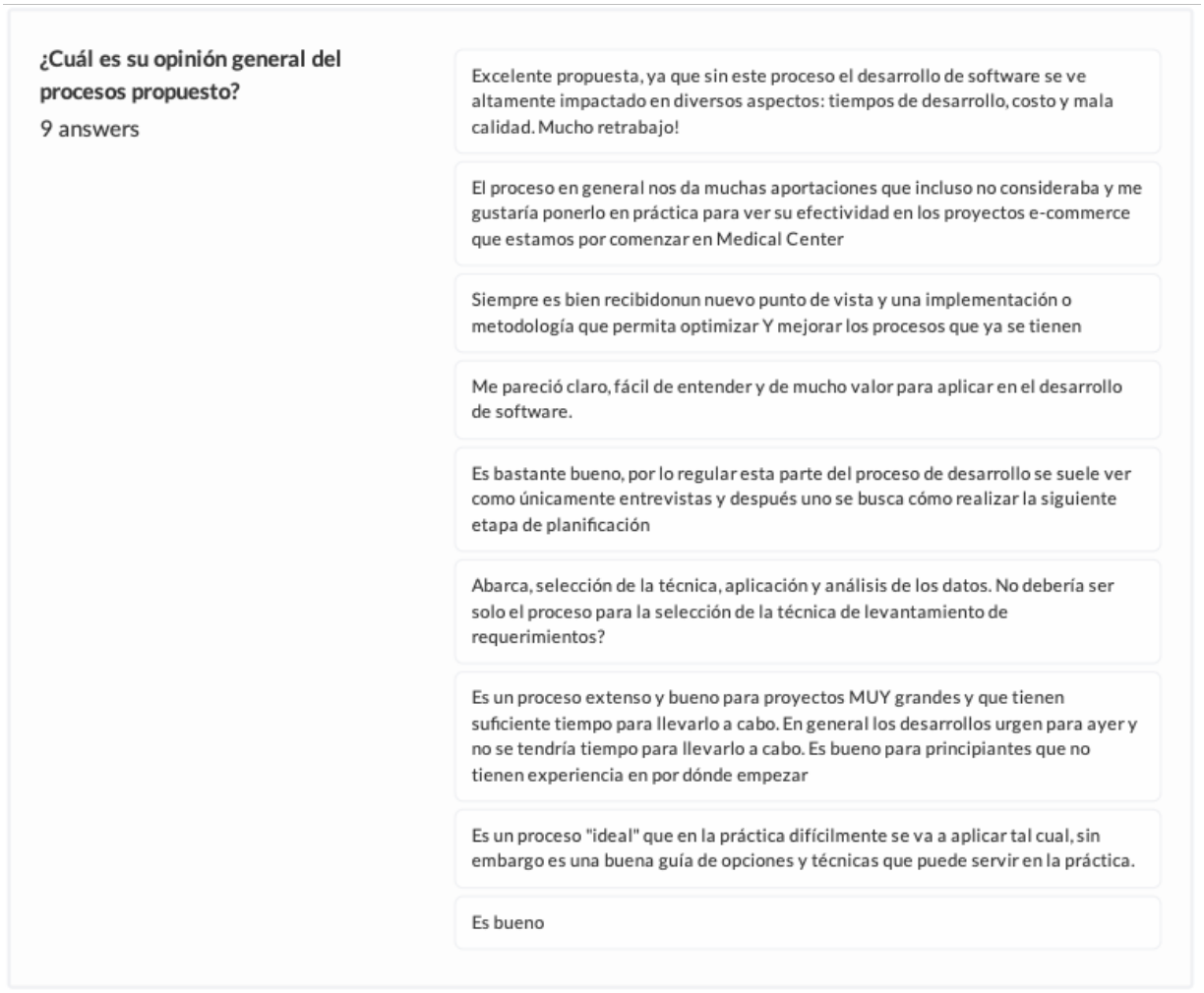


Figura 34: *Respuestas del panel de expertos del instrumento de la técnica Delphi para la evaluación de la propuesta, Parte 4. [Imagen elaborada por el autor]*

Figura 35: *Respuestas del panel de expertos del instrumento de la técnica Delphi para la segunda evaluación de la propuesta, Parte 1. [Imagen elaborada por el autor]*

Considera necesario emplear una matriz de decisiones dependiendo del proyecto a desarrollar con el objeto de identificar que pasos e instrumentos del proceso propuesto se pueden emplear y cuales omitir.

5 answers

- Esta pregunta está mal redactada, le faltan comas y aclarar mas
- No necesariamente
- si, todo marco de referencia hay pasos a considerar, los que no se consideran es conveniente documentarlos para auditoria en vez de generar datos no verídicos
- Claro que si, mas bien es indispensable.
- La matriz de decisiones es importante pero de l misma forma depende del tipo de proyecto que estemos trabajando, habrá algunos proyectos que lo necesiten como tal una matriz de decisiones.

¿En su experiencia los pasos e instrumentos propuestos pueden ser empleados por analistas nuevos (que van iniciando) en el desarrollo de software?

5 answers

- No es tan fácil si no tienen un conocimiento mínimo de los instrumentos. Se deben explicar y poco a poco, los novatos van ganando experiencia
- Sí
- si, con su debida capacitación
- Aqui está el problema: primero tienen que aprender todo y si no lo aplicaron no van a tener criterios adecuados para seleccionar lo que necesitan.
- Si, siempre y cuando se les capacite y se les explique la esencia de la este proceso.

Considera que el proceso propuesto es lo suficientemente flexible para omitir o adaptar dependiendo del proyecto y de la metodología de desarrollo de software seleccionado

5 answers

- Si es flexible
- Sí
- si
- Si, pero tiene el problema de varias propuestas anteriores. Por ejmplo RUP o conjunto de procesos de ISO/IEC 12207 son colecciones muy amplias para abarcar todos los casos prácticos y se deja a los profesionales que ellos elijan lo que necesitan. Pero ellos tienen problema de que primero tienen que aprender todo para poder seleccionar. Y este es el punto débil de colecciones grandes. Criterios de selección que se dejan a los usuarios. Tu propuesta tiene secciones de ventajas y desventajas, pero a mi gusto son muy generales. Lo ideal sería poner en que contextos vale la pena aplicar cada cosa y porqué (beneficios).
- Considero que se puede adaptar dependiendo del proyecto, ahora esto podremos ver cuando se ponga en práctica en varios proyectos y ver los resultados en casa uno.

Figura 36: *Respuestas del panel de expertos del instrumento de la técnica Delphi para la segunda evaluación de la propuesta, Parte 2. [Imagen elaborada por el autor]*

BIBLIOGRAFÍA

Abreu, R. B., Guntín, Y. O., Alfonso, Y. Á., Mena, J. C., and A, G. K. T. S. (2009). Metodología ágil Crystal Clear. Un caso de estudio. *Serie Científica*, 2(3).

Al-rimawi, R., Dwairej, D., Masadeh, A., Al-Ananbeh, E., and Muayyad, A. (2016). E-health Concept Development and Maturity in Literature. *Journal of Health, Medicine and Nursing*, 29:156–166.

Anwar, F. and Razali, R. (2012). A practical guide to requirements elicitation techniques selection-an empirical study. *Middle-East Journal of Scientific Research*, 11(8):1059–1067.

Aparicio, I. A. (2012). *Ingeniería de Software*. Universidad Nacional Abierta y a Distancia.

Aristizábal-Mejía, N. and Torres-Moreno, M. E. (2009). Técnicas de Levantamiento de Requerimientos con Innovación. In *Cuarto Congreso Colombiano de Computación 4CCC*, number January 2009.

Arturo, P. E. and Cruz E. Ma. Cristina (2006). *Metodología Crítica de la Investigación. Lógica, procedimiento y Técnicas*.

Bell, D. (2005). *Software Engineering for Students A Programming Approach*. ADDISON-WESLEY, fourth edition.

Bender, D. and Sartipi, K. (2013). Hl7 fhir: An agile and restful approach to healthcare information exchange. In *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, pages 326–331.

Boehm, B., Qj, H. I. R. U., Grünbacher, P., Vhd, S. J., Olq, X. Q. L., Dw, D. F., Briggs, R. O., Dqg, L., Ri, L., Vwdnhkroghuv, N. H., and Lvvxhv, U. (2001). Easywinwin: A groupware-supported methodology for requerimientes negotiation. pages 720–721.

Bourque, P. and Fairley, R. E. (2014). *Guide to the Software Engineering - Body of Knowledge.*

Cataldi, I. Z. (2000). Metodología de diseño, desarrollo y evaluación de software educativo. Master's thesis, Facultad de Informática. UNLP Facultad de Informática. UNLP Facultad de Informática. UNLP.

Cervantes Ojeda, J.; Gómez Fuentes, M. d. C. (2012). Taxonomía de los modelos y metodologías de desarrollo de software más utilizados.

Chandler, R. U. and Carolyn (2012). *A Project Guide to UX Design*. New Riders, second edition.

Clark, W. (1923). *The Gantt chart a working tool of management*. Segunda edition.

Cockburn, A. (2002). *Agile Software Development*.

Daniel, M. A. (2010). Capa-3: Una innovadora metodología para el desarrollo de software en ambientes de trabajo virtuales. Master's thesis, Universidad Tecnológica Nacional Facultad Regional Buenos Aires.

Dooley, J. (2011). *Software Development and Professional Practice*. Appres.

Engineers, I. o. E. and Electronics (2008). Especificacion de Requisitos segun el estandar de IEEE 830. page 27.

Eysenbach, G. (2001). What is e-health? *J Med Internet Res*, 3(2):e20.

Fernandez-ledesma, J. D. and Duitama, J. F. (2012). Ubiquitous Computing and Ambient Intelligence. 7656(December).

Getto, G. and Cao, J. (2016). UX Design The Definitive Beginner's Guide. *UXPin*, pages 1–96.

Gibbons, S. (2017). Ux mapping methods compared: A cheat sheet.

Gibbons, S. (2018a). Empathy mapping: The first step in design thinking.

Gibbons, S. (2018b). Journey mapping 101.

Gómez, M. (2011). *Material Didáctico Notas Del Curso*. primera edition.

Grajale G, T. (2000). Tipos de Investigacion. *IUPuebla*, page 4.

Grime, M. M. and Wright, G. (2016). Delphi Method. *Wiley StatsRef: Statistics Reference Online*, (May):1–6.

Gross, J. and McInnis, K. (2003). *Kanban made simple: demystifying and applying Toyota's legendary manufacturing process*.

Gunda, S. G. (2008). *Requirements Engineering : Elicitation Techniques*. PhD thesis.

Harold A. Linstone, M. T. (2011). The Delphi method: An efficient procedure to generate knowledge. *Skeletal Radiology*, 40(8):959–961.

He, Z., Marquard, J., and Henneman, E. (2016). Model Guided Design and Development Process for an Electronic Health Record Training Program. *AMIA Annual Symposium proceedings*, (D):1814–1821.

Hickey, A. M. and Davis, A. M. (2002). Requirements Elicitation and Elicitation Technique Selection : A Model for Two Knowledge-Intensive Software Development Processes Unsolved Problem Software Development Software Solutions. (C):2005–2010.

Hohmann, L. (2007). *Innovation Games Creating Breakthrough Products Through Collaborative Play*. Addison-Wesley.

Huis in 't Veld, R. M.; Widya, I. A. B. R. G. S. L. H. H. J. V.-H. M. M. (2010). A scenario guideline for designing new teletreatments: a multidisciplinary approach. *Journal of Telemedicine and Telecare*, 16.

Hussain, A., Mkpojiogu, E. O., and Nawi, M. N. M. (2016). Requirements model for an e-Health awareness portal. *AIP Conference Proceedings*, 1761.

IIBA (2015). A Guide to the Business Analysis Body of Knowledge (BABOK® Guide) – Version 3.0. page 514.

Jain, D., Krishna, P. V., and Saritha, V. (2012). A study on internet of things based applications. *arXiv preprint arXiv:1206.3891*.

Kalbach, J. *Mapping Experiences: A Complete Guide to Creating Value through Journeys, Blueprints, and Diagrams*.

Ken, S. and Sutherland, J. (2017). La Guía de Scrum.

Koch, N. and Escalona, M. J. (2004). Requirements engineering for web applications – a comparative study. *Journal of Web Engineering*, 2(3):193–212.

Larburu, Nekane; Widya, I. B. R. G. A. H. H. J. N. C. (2013). [ieee 2013 ieee 21st international requirements engineering conference (re) - rio de janeiro-rj, brazil (2013.07.15-2013.07.19)] 2013 21st ieee international requirements engineering conference (re) -

early phase telemedicine requirements elicitation in collaboration with medical practitioners.

Leonor Teixeira, Carlos Ferreira, B. S. S. (2012). User-centered requirements engineering in health information systems: A study in the hemophilia field. *Computer Methods and Programs in Biomedicine*, 106.

Levy, J. (2015). *UX Strategy HOW TO DEVISE INNOVATIVE DIGITAL PRODUCTS*.

Maheu, M. M., Whitten, P., and Allen, A. (2001). *E-Health, Telehealth, and Telemedicine*.

Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Robert C. Martin Series. Prentice Hall, Boston, MA.

McDonald, C. J., Marc Overhage, J., Tierney, W. M., Dexter, P. R., Martin, D. K., Suico, J. G., Zafar, A., Schadow, G., Blevins, L., Glazener, T., Meeks-Johnson, J., Lemmon, L., Warvel, J., Porterfield, B., Warvel, J., Cassidy, P., Lindbergh, D., Belsito, A., Tucker, M., Williams, B., and Wodniak, C. (1999). The Regenstrief Medical Record System: A quarter century experience. *International Journal of Medical Informatics*, 54(3):225–253.

McEwen, A. and Cassimally, H. (2013). *Designing the Internet of Things*.

Michael, M., E., C. S., Avery, H., I., M. I., and José, D. M. (2014). ehealth and health literacy: A research methodology review. *Journal of Computer-Mediated Communication*, 19(3):516–528.

Moran, A. (2014). *Agile software development*. Number 9783319050072.

Moule, J. (2012). *Killer UX Design*. SitePoint.

Mrayat, O. A., Norwawi, N., and Basir, N. (2013). Requirements Elicitation Techniques: Comparative Study. *Ijrdet.Com*, 1(3):1–10.

Nonaka, I. (1994). A dynamic theory of organizational knowledge creation. *Organization Science*, 5(1):14–37.

Odeh, M., Hauer, T., McClatchey, R., and Solomonides, T. (2004). A use-case driven approach in requirements engineering : The mammogrid project. *CoRR*, cs.DB/0402008.

of Technology Assessment, U. S. C. O. (1977). *Policy Implications of Medical Information Systems*. OTA (Series). Congress of the United States, Office of Technology Assessment.

Oktaba, G. I. G. H. (2010). *Ingeniería de Software Pragmática*. Facultad de Ciencias UNAM.

Pavlov, V. L., Doroshenko, A., Taganskaya, T., Zhereb, K., and Boyko, N. (2008). An experience of integrating INTSPEI P-Modeling Framework with Microsoft Solutions Framework for Agile Software Development. *Proceedings of the IASTED International Conference on Software Engineering, SE 2008*, pages 318–323.

Per Kroll, Philippe Kruchten, G. B. (2003). *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. Addison-Wesley Professional.

Pérez, O. A. (2011). Cuatro enfoques metodológicos para el desarrollo de software rup,msf, xp, scrum. *Inventum Facultad de Ingeniería UNIMINUTO*, (10):64–78.

PMI, P. M. I. (2008). *Guía de los Fundamentos para la Dirección de Proyectos.*

Ponce, H. (2006). La matriz FODA : una alternativa para realizar diagnósticos y determinar estrategias de intervención en las organizaciones productivas y sociales. *Contribuciones a la Economia*, page 16.

Pressman, R. S. (2002). *Ingeniería del Software un Enfoque Práctico*. Mc Graw Hill, fifth edition.

Ramírez, I. (1993). *El protocolo de investigación: lineamientos para su elaboración y análisis*. Trillas.

Reynoso, C. (2004). Métodos heterodoxos en desarrollo de software.

Rowland, C. and Charlier, M. (2015). *User Experience Design for the Internet of Things*. O'Reilly Media.

Rowland, C., Goodman, E., Charlier, M., Light, A., and Lui, A. (2015). *Designing Connected Products: UX for the Consumer Internet of Things*. O'Reilly Media.

Scandurra, I., Holgersson, J., Lind, T., and Myreteg, G. (2013). Development of Novel eHealth Services for Citizen Use – Current System Engineering vs. Best Practice in HCI. In Kotzé, P., Marsden, G., Lindgaard, G., Wesson, J., and Winckler, M., editors,

14th International Conference on Human-Computer Interaction (INTERACT), volume LNCS-8118 of *Human-Computer Interaction – INTERACT 2013*, pages 372–379, Cape Town, South Africa. Springer. Part 8: Health/Medical Devices.

Scott, R. E. and Mars, M. (2013). Principles and framework for eHealth strategy development. *Journal of Medical Internet Research*, 15(7).

Serna, M. E. (2012). Analysis and selection to requirements elicitation techniques. *2012 7th Colombian Computing Congress, CCC 2012 - Conference Proceedings*.

Sharma, S. and Pandey, S. K. (2013). Revisiting Requirements Elicitation Techniques. *International Journal of Computer Applications*, 75(12):975–8887.

Sood, V. R. and Arora, M. (2012). INTERNATIONAL JOURNAL OF ADVANCES IN COMPUTING AND INFORMATION Comparison of Requirements Elicitation Techniques. (May):378–387.

Stroetmann, K. A. (2014). New Perspectives in Information Systems and Technologies, Volume 2. 276:395–406.

Swarnalatha, K. S., Srinivasan, G. N., Sharma, K., and Dravid, M. (2014). A Contextual Approach for Requirement Elicitation in Requirement Engineering Process.

Tan, J. (2005). *E-health care information systems: an introduction for students and professionals*. Number Book, Whole.

Treder, M. (2013a). *The user experince guide book for product managers*. UXPin.

Treder, M. (2013b). *Ux Design for Startups*. UXPin.

Unger, R. and Chandler, C. (2012). *A Project Guide to UX Design: For user experience designers in the field or in the making*. Voices That Matter. Pearson Education.

van Kranenburg, R. and Dodson, S. (2008). *The Internet of Things: A Critique of Ambient Technology and the All-seeing Network of RFID*. Institute of Network Cultures.

van Rooij, T. and Marsh, S. (2016). Ehealth: Past and future perspectives. *Personalized Medicine*, 13.

van Vliet, H. (2007). *Software Engineering: Principles and Practice*, volume 41. Wiley.

Webster, H. E. and G., J., editors (2016). *Telehealth and mobile health*. CRC Press.

Weitzenfeld, D. A. (2002). *Ingeniería de Software Orientada a Objetos*. División Académica de Ingeniería. ITAM.

Yousuf, M. and Asger, M. (2015). Comparison of Various Requirements Elicitation Techniques. *International Journal of Computer Applications*, 116(4):15.

Z. Zakaria, R. Atan, A. A. A. G. and Sani, N. F. M. (1990). Ieee standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, pages 1–84.

Zhang, Z. (2007). Effective Requirements Development - A Comparison of Requirements Elicitation techniques. In E. Berki, J. Nummenmaa, I. Sunley, M. R. and Staples, G., editors, *Software Quality Management XV: Software Quality in the Knowledge Society*, pages 225–240. British Computer Society.

Zowghi, D. and Coulin, C. (2005). Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. *Engineering and Managing Software Requirements SE - 2*, pages 19–46.