Instituto Tecnológico
y de Estudios Superiores de Occidente
Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática
Maestría en Informática Aplicada

Diseño de marco ágil de gestión de proyectos de desarrollo de software, con enfoque en el ser humano

---

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRO EN INFORMÁTICA APLICADA

Presenta: **FERNANDO ULISES MÉRITO DEL CAMPO**

Asesora **MTRA. DELIA DEL CARMEN RAMÍREZ VÁZQUEZ**

Tlaquepaque, Jalisco. Noviembre de 2021

Dedicatorias

A mi señora Madre que me dio la vida, me inculcó sus valores, me apoyó y me sigue apoyando tanto, y que me enseñó el amor al trabajo duro así como su visión de nunca rendirse.

A mi Esposa que me ama, complementa, escucha, entiende, apoya, motiva e inspira a siempre dar un paso más, nunca conformarme y siempre buscar lo mejor para nuestra familia. A ella que dio a luz a nuestros hijos que tanto amamos y que se han convertido en las mayores motivaciones que tenemos para siempre dar nuestro mejor esfuerzo a diario, así como uno de los mayores motivos para sonreír y nunca detenerme.

A Dios que me fortalece, disciplina, guía y redarguye. Desde que le conocí y recibí en mi vida me convirtió en un ser humano diferente que ahora le alaba y agradece por cada respiro. Gracias le doy por lo que soy, por lo que no soy, por lo que me enseña y por lo que ha de venir.

A Delia Ramírez quién me rescató del limbo en que me encontraba con relación a mi TOG, quién me ha aconsejado y guiado para poder culminar y materializar este trabajo que más que un requisito, se ha convertido en un interesante reto que me ha llevado más allá de lo que pensé ser capaz.

A Continental Automotive México, empresa para la que laboro y que hizo posible el que yo haya podido cursar este posgrado que ahora veo como un sueño realizado. Sin el constante, siempre abundante y completamente abierto apoyo que me da, esto simple y sencillamente no hubiese sido posible. Tengo que agradecer por los retos que me han hecho un mejor profesional pero aún mejor ser humano, así como por el hecho de crean en mí de esta manera tan increíble.

A mis profesores y compañeros que hicieron de este posgrado una enorme aventura, llena de retos, alegrías, incontables expresiones de ¡wow!, pero sobre todo, que me permitió conocer grandes seres humanos que tocaron mi vida de una manera sumamente inspiracional y personal. Gracias por el aprendizaje que no que se queda en el tintero, sino que llevo en el corazón y con la motivación de devolver un poco de lo tanto que se me ha dado.

Finalmente, al ITESO por poner siempre los valores humanos por delante, por siempre buscar la palabra de Dios y los valores cristianos en los procesos de enseñanza y aprendizaje que proveen. Por inspirarme a crecer, a ser, a dar y a nunca detenerme. Gracias por dotar de una infraestructura bellísima e inspiradora. Gracias por darme tanto, por aceptarme y hacerme sentir como de casa.

Tabla de contenido

Lista de figuras

Lista de anexos

Resumen

Las organizaciones deben ser cada vez más ágiles para reaccionar a tiempo y con antelación a las cambiantes necesidades de sus clientes, así como a las tendencias de los mercados. Dicha agilidad permite que las empresas sigan siendo competitivas y vigentes, desarrollando y proveyendo más y mejores soluciones, así como de valor para sus clientes.

Para tal fin, se desarrollan diversas estrategias empresariales que buscan ayudar a lograr los objetivos esperados, los cuales pueden ser alcanzados mediante la implementación de proyectos. Estos requieren el uso de metodologías, marcos de referencia y mejores prácticas para incrementar las probabilidades de éxito, sin embargo, es común que estas suelan especializarse para ciertos tipos de proyectos. En específico, los proyectos de desarrollo de *software* requieren una gestión muy específica, dado que el entregable final no es un tangible.

Históricamente, los proyectos de desarrollo de *software* fueron gestionados utilizando un ciclo de vida de cascada pero, conforme la complejidad se fue incrementando en las organizaciones, entre otros motivos, nació el auge de la gestión de proyectos mediante la utilización de marcos que utilizan el ciclo de vida ágil. El máximo exponente es un marco denominado *Scrum*, el cual es extensivamente utilizado en las organizaciones.

Sin embargo, el solo uso de *Scrum* como marco ágil de gestión de proyectos puede acarrear una serie de problemáticas, las cuales se desprenden de la dinámica diaria de la organización donde labora el autor de la presente propuesta. Por otro lado, se encuentra que dichas problemáticas no solo ocurren en dicha empresa, sino que suelen ocurrir en otros entes donde también se usa *Scrum* como único marco de gestión de proyectos de *software*.

Dado lo anterior, la presente propuesta tecnológica busca crear un método ágil que considere a *Scrum* como la base de operación pero, que considere además otros dos métodos que le complementen y que son el *Lean Startup* y *Design Thinking*, en aras de poner al ser humano en el centro de la toma de decisiones, para poder desarrollar un producto y/o servicio de valor, escalable y sin desperdicios.

1 Introducción y contexto

1.1 Contexto organizacional

Las organizaciones se encuentran en una guerra permanente donde deben ser bastante competitivas, por lo que su calidad debe mejorar continuamente, mientras que los costos tienen que ser contenidos a toda costa; además, es necesario que sean ágiles para encontrar y explotar las oportunidades que los mercados les ofrecen. Para ello, las empresas deben reinventarse y reimaginarse (Project Management Institute [PMI], 2020a), de manera que puedan ser más prontas para reaccionar a las necesidades de sus clientes, anticiparse a las tendencias tecnológicas y de mercado, así como acelerar la generación de valor.

De acuerdo con un estudio del *Project Management Institute* (PMI, 2020a), solo el 25 % de las organizaciones invierte en nuevas oportunidades basadas en tendencias de mercado, lo cual es un problema grave, puesto que imposibilita la creación de productos y servicios orientados a las percepciones cambiantes de los clientes y mercados. Sin embargo, lo anterior no puede ser posible sin el uso extensivo de tecnología que habilite la toma de decisiones fundamentadas en datos y no en percepciones, la rápida entrega de información referente a preferencias de los clientes y de las tendencias en los mercados (PMI, 2020a; Prades et al., 2013).

Dado lo anterior, en aras de ser competitivas y permanecer vigentes, las organizaciones deben delinear estrategias, entre otras, de negocio, estructurales y de visión, de manera que todas ellas colaboren en el logro de sus objetivos, los cuales pueden ser alcanzados mediante la implementación de proyectos (Jovanovic y Beric, 2018). Un proyecto es definido como un esfuerzo temporal enfocado en la obtención de un producto o servicio único (*PMI*, 2017a).

1.2 Rol e importancia de los proyectos para las organizaciones

A pesar de la gestión de dichos proyectos, se requiere del uso de metodologías, marcos de referencia o buenas prácticas que ayuden a incrementar las probabilidades de éxito. En este sentido, la teoría y práctica de la administración de proyectos son otorgadas mediante una diversidad de metodologías, marcos de referencia y mejores prácticas existentes que proveen de una serie de métodos, técnicas, procedimientos, entre otros; los cuales son empleados en la

gestión de los proyectos. También, son consideradas como un grupo de principios y guías que dictan cómo un proyecto será ejecutado (Špundak, 2014).

Un factor común de tales metodologías, marcos de referencia o buenas prácticas es que suelen orientarse a cierto tipo de proyectos en particular, que entre otras, depende de condiciones como la particularidad de la organización donde se hará el proyecto o la clase de gestión y cultura empresarial; por lo tanto, la gestión de proyectos con cada metodología suele ser distinta (Besner y Hobbs, 2012; Jovanovic y Beric, 2018), lo cual sustenta por qué existen diversas metodologías de gestión de proyectos (Ameijide, 2016).

Dado lo anterior, es vital la adecuada elección del marco o la metodología, además de su correcto uso, debido al alto impacto que supone la decisión tomada, ya sea positivo o negativo. De acuerdo con Rojas et al. (2020), la probabilidad de que un proyecto sea más exitoso utilizando la metodología adecuada puede llegar al 90 %. Entre los tipos de proyectos existentes, se encuentra el de desarrollo de software, el cual cuenta con una naturaleza de producción diferente, pues se lleva a cabo a través de un proceso, no se fabrica con materia prima ni se construye a partir de partes más pequeñas, sino gracias un proceso de ingeniería (Hernández y Bravo, 2020).

1.3 Ciclos de vida de desarrollo de software

Es importante indicar que existen distintos ciclos de vida de desarrollo de software, pero entre los más conocidos y utilizados están los tradicionales y los ágiles (Leau et al., 2012). El primero se basa en procesos lineales y secuenciales, donde una etapa inicia al finalizar la anterior. Su utilidad destaca en proyectos de desarrollo de software complejos, en los que se requiere planear con anticipación y en ocasiones, y donde los tiempos de entrega, recursos a utilizar y sus grados de control del alcance estén estipulados de forma contractual. Estos cuentan con una estructura rígida y lineal. Además, su suposición fundamental indica que los desarrollos son totalmente especificables, predecibles y se construyen a través de una planificación meticulosa. Las pruebas del software se realizan una vez se está en etapas maduras del proyecto y a partir de controles estrictos, típicamente cuando la solución está terminada o cerca de culminarse (González et al., 2019).

Por su parte, el ciclo de vida del desarrollo de software ágil es un estilo que se centra en la entrega temprana de valor, mejora continua, flexibilidad, alta colaboración y creación de

soluciones que reflejan las necesidades del cliente. Este tipo de gestión es altamente utilizada para el desarrollo de proyectos que requieren rapidez y flexibilidad en su proceso, pero sobre todo, aquellos que se ejecutan en entornos de alta incertidumbre, donde no es sencillo planear o anticipar (Hernández y Bravo, 2020). Con este tipo de gestión, el énfasis está en las personas, los clientes y en mejorar la capacidad de respuesta, en vez de en los procesos, contratos y métodos de planificación rígidos, que se destacan en el estilo tradicional (Leau et al., 2012). En síntesis, es un estilo enfocado al cliente y sus necesidades, lo que mejora la capacidad de análisis y disminuye el riesgo de fracaso o de errores en el desarrollo de las aplicaciones (González et al., 2019).

1.4 Ciclo de vida a utilizar

Con base en lo expuesto, el estilo de desarrollo ágil de software es el ideal para gestionar proyectos de desarrollo de aplicaciones que requieran alta adaptabilidad a entornos con bastante incertidumbre (Basco et al., 2018), gran flexibilidad y entrega rápida de valor (Hernández y Bravo, 2020), así como la facilidad en otorgar soluciones personalizadas y centradas en los clientes (González et al., 2019). Por otro lado, el marco o la metodología ágil a utilizar depende de diferentes factores, por ejemplo, la experiencia del equipo de trabajo de desarrollo de software, la madurez de los procedimientos organizacionales, el soporte gerencial, entre otros (KPMG, 2019).

2 Planteamiento del problema

El autor de la presente propuesta labora para una empresa del sector automotriz, donde se investigan, desarrollan y manufacturan componentes electrónicos. Dentro de tal organización, se llevan a cabo soluciones de software de diversa índole; empero, cuenta con un área de desarrollo de software que sufre de diversas problemáticas que afectan el logro de sus objetivos. Con base en un análisis literario, se encuentra que dichas problemáticas no son exclusivas de esta empresa, sino que puede considerarse un tema generalizado en mayor o menor medida, lo cual se sustenta una aproximación a cada problemática a continuación.

Por otro lado y aún en relación con la empresa en cuestión, el área de desarrollo mencionada trabaja con *Scrum* como su marco de gestión de desarrollo de software; esto brinda múltiples ventajas si se compara con el uso de metodologías de cascada, por ejemplo, su flexibilidad, adaptabilidad, reducción en tiempos de desarrollo e incremento en calidad, dentro del contexto del desarrollo de software (Dobrigkeit y de Paula, 2017; Signoretti et al., 2020). Sin embargo, a pesar de ser un marco altamente empleado en compañías líder en desarrollo de software (Ionel, 2008), el uso de *Scrum* por sí solo no garantiza que se obtengan siempre los mejores resultados por medio del software entregado, en especial, cuando el software es solo un componente más del producto a desarrollar, cuando se desarrollan soluciones a gran escala o cuando se desea cambiar la manera en que las organizaciones trabajan (Vilkki, 2010).

De este modo, gestionar y desarrollar proyectos de software solo con *Scrum* puede acarrear severos problemas para las organizaciones, tales como lanzar productos equivocados al mercado, una aceptación pobre de estos, retrabajos e inversiones innecesarias (Dobrigkeit y de Paula, 2017). De acuerdo con lo manifestado, se exponen tres problemáticas en el uso exclusivo de *Scrum* como marco de gestión de desarrollo de software, identificadas en la literatura y la práctica recopilada de la empresa mencionada:

1. Se lleva a cabo un proceso de desarrollo ágil, pero al final de este no se entrega todo el valor que el cliente espera, es decir, las expectativas no son cubiertas (Vilkki, 2010).
2. La solución entregada tiene ciclos de vida cortos y adolece de escalabilidad.
3. Falta de enfoque en el ser humano.

2.1 Problemática 1: no se entrega todo el valor esperado

En su literatura, Dobrigkeit y de Paula (2017) consideraron que *Scrum* adolece de características de entendimiento y solución de problemas, así como de falta de atención al diseño, lo que deriva en que el desarrollo de soluciones no entrega todo el valor requerido, en rediseños y retrabajos innecesarios, además de inversiones improvisadas. Ahora, es importante mencionar que al hacer uso de *Scrum,* la visión del cliente influencia de manera significativa el desarrollo, por lo cual, si el cliente no logra transmitir lo que necesita e identificar correctamente su necesidad en colaboración con el equipo de desarrollo, se creará un entregable que generará un valor diferente a lo esperado (Ionel, 2008).

Al respecto, algunas literaturas puntualizan esta tendencia en los equipos que trabajan con *Scrum*, dado que este marco de trabajo suele asumir que en un proyecto recién iniciado, el equipo de trabajo ya cuenta con una visión integral del producto a desarrollar, aunque sin tener claro de dónde y cómo viene dicha visión, si es la correcta y si considera todo lo que la persona necesita en realidad (de Paula y Araújo, 2016; Lindberg et al., 2010).

Sumado a lo anterior, a pesar de ser un marco flexible a los cambios, *Scrum* suele centrarse en una línea de acción inicial dada por el cliente al no permitir tener más opciones de solución a las problemáticas dadas; así, *Scrum* fundamenta su operación en desarrollos incrementales continuos basados en la línea de acción delineada en el inicio, en vez de buscar opciones de solución a las problemáticas del cliente (Lindberg et al., 2010). En resumen, aunque los requerimientos del cliente se ejecutan en tiempo y usualmente dentro del presupuesto, se entrega un producto final que no cumple con todas las expectativas al cierre del proyecto; por lo que pareciese que existiera una brecha entre lo que el cliente expone como su necesidad y la realidad de la situación.

2.2 Problemática 2: la solución entregada tiene ciclos de vida cortos y adolece de escalabilidad

Diversos autores han sustentado este efecto dadas las características de *Scrum*: a manera de ejemplo, *Scrum* propone evitar cualquier pensamiento divergente para buscar la adecuada solución que solvente una necesidad, en aras de enfocar la vista en el siguiente paso de desarrollo (Lindberg et al., 2010). También, Dobrigkeit y de Paula (2017) han mencionado que el solo uso de *Scrum* no facilita la creación de soluciones escalables fácilmente, que se adapten

a los cambios del mercado y las necesidades de los clientes, lo cual sustenta la problemática expuesta, que resulta en retrabajos, inversiones y horas extras de desarrollo.

Por otro lado, a pesar de que *Scrum* hace énfasis en la colaboración entre el equipo de trabajo, adolece en facilitar la colaboración creativa con personal interdisciplinario más allá de los expertos que forman parte del equipo *Scrum* (Lindberg et al., 2010), lo cual limita la visión en las propuestas de solución de problemas. Lo anterior es vital dado que, si la interacción entre los interesados del proyecto no es integral –considerando a todos los que pueden resultar impactados para bien o mal con el proyecto–, el resultado final del puede llevar a un producto o servicio que el cliente no necesita (Grossman-Kahn y Rosensweig, 2012).

En definitiva, por sí solo, *Scrum* no valida que el producto a desarrollarse vaya a ser aceptado en el mercado meta, que siga tenga vigencia o que existan consumidores potenciales para él, pues el enfoque de *Scrum* es el desarrollo ágil del producto (Grossman-Kahn y Rosensweig, 2012). Asimismo, tampoco cuenta con mecanismos para el seguimiento y escalado de las soluciones ya desarrolladas (Dobrigkeit y de Paula, 2017).

2.3 Problemática 3: falta de enfoque en el ser humano

Cuando se usa *Scrum*, se da por hecho que el cliente tiene una visión clara de lo que necesita y se comienza a trabajar con ese supuesto; sin embargo, este marco no permite ni impulsa al equipo de trabajo a ponerse en los pies del cliente para entender de manera empática la necesidad real a satisfacer (Ximenes et al., 2015). Por ende, cuando el cliente llega con una necesidad altamente ligada a una solución tecnológica que no está centrada en el ser humano, impulsará la creación de una solución que tendrá una alta resistencia por parte de los potenciales clientes, siendo altamente escépticos en aceptar esa solución (Grossman-Kahn y Rosensweig, 2012). Lo anterior se refiere a que los requerimientos de la solución solicitada se suelen enfocar en aspectos del software o del proceso, más no de la realidad y experiencia de la persona, esto es, el cliente en su día a día (Lindberg et al., 2010). Este contexto es común, dado que el proceso de desarrollo en *Scrum* es susceptible a entramparse en aspectos técnicos e, incluso, el equipo de trabajo puede crearse una visión propia de la necesidad del cliente (Ximenes et al., 2015).

3 Objetivo

Desarrollar un marco metodológico ágil para la gestión de proyectos de desarrollo de software, tomando como base el marco de *Scrum*, que ayude a aminorar las problemáticas antes expuestas por medio del estudio de literatura relacionada diversa, entendiendo y exponiendo sus debilidades como marco de gestión de proyectos de desarrollo de software, así como otras metodologías y marcos de trabajo como *Lean Startup* y *Design Thinking* que le puedan complementar.

4 Justificación

Como se ha examinado, el uso de *Scrum* como marco ágil de gestión de proyectos de desarrollo de software trae grandes beneficios si se compara con el uso de metodologías de cascada, pero esto no lo convierte en un método infalible ni perfecto. En ese sentido, surgen diversas problemáticas como las ya mencionadas, las cuales se dan en diferentes sectores productivos con su uso, al existir nuevos y diferentes modelos que se proponen para desarrollar otros tipos de proyectos, donde se considera que *Scrum* podría beneficiarse de estos; de esta manera, el presente trabajo se enfoca en desarrollar una propuesta que considere lo expuesto.

Ahora, en cuanto a la gestión de proyectos, existen varios retos y problemáticas que ponen en riesgo su éxito y, por ende, del cumplimiento de los objetivos de las organizaciones. De acuerdo con el PMI (2020b), las empresas desperdiciaron casi el 12 % de su presupuesto de inversión en proyectos debido a un desempeño pobre, cifra prácticamente idéntica a los seis años anteriores. Por su parte, una encuesta realizada por el *Harvard Business Review* indicó que, en promedio, el 27 % de los proyectos de tecnología de la información (IT por su siglas en inglés: *Information Technology*) superan su presupuesto; incluso, uno de cada seis proyectos en este rubro llegan a invertir el 200 % de su presupuesto original, con un retraso del 70 % (Flyvbjerg y Budzier, 2011). Cabe añadir que un aspecto crucial tiene que ver con la preparación de los administradores de proyectos y sus capacidades, punto considerado como uno de los más grandes retos en la gestión de proyectos.

Por otro lado, de acuerdo con Mieritz (2019), los problemas de funcionalidad (22 %) y retrasos en calendario (28 %) son las dos mayores causas de fallas en los proyectos. En un estudio realizado por PwC sobre 10 640 proyectos, se encontró que solo el 2.5 % de estos fueron completados de forma exitosa (PMI, 2017b) y el resto falló en algún objetivo inicial, el presupuesto o la fecha planeada. Todo esto supone grandes pérdidas; por ejemplo, solo en 2019, Estados Unidos (EEUU) gastó entre 50 y 150 billones de dólares en pérdidas de ganancias y productividad (Hardy-Vallee, 2012).

En cuanto a temas de preparación y profesionalización, los administradores de proyectos no siempre se encuentran formados en la materia, lo que resulta en un 40 % de ellos que no usan una metodología de gestión de proyectos, mientras que el 60 % restante no la utiliza de forma continua (Wellingtone, 2018). De hecho, se considera que el uso de una metodología de trabajo adecuada en materia de gestión de proyectos puede apoyar a

incrementar hasta un 20 % de la posibilidad de éxito, sin dejar atrás otros retos que se consideran como la base de la pirámide del éxito en la gestión de proyectos, como la transformación cultural en las organizaciones: personas, valores y sistema (Rączka, 2015). Dado lo anterior, es posible establecer diversos motivos por los que un proyecto puede fallar, los cuales se discuten más adelante. No obstante, una de las razones más comunes por la que falla un proyecto es por vicios en sus procesos de gestión, desde malas prácticas, hasta falta de capacitación, entre otras (Kaur y Sengupta, 2011).

5 Marco teórico

5.1 Gestión de proyectos: conceptos, importancia y estadísticas

De acuerdo con el PMI (2017a), la gestión de proyectos es "la aplicación de conocimientos, habilidades, herramientas y técnicas de gestión a las actividades que los componen, a objeto de cumplir con los requerimientos de este" (p. 3). Así, cada proyecto es único, tiene una meta particular y objetivos definidos, así como una vida útil finita. Dada la complejidad creciente en los proyectos que se desarrollan a nivel mundial, la gestión de proyectos vista como una disciplina que presenta un crecimiento notable (Wallace, 2014).

Dicho incremento en complejidad se presenta debido a que, conforme ha pasado el tiempo, se han agregado controles y restricciones a los proyectos, tales como fechas de inicio y término que en ocasiones son retadoras e inamovibles; límites de costos estrictos; restricciones de salud, seguridad y medio ambiente; alineación a factores políticos y geográficos; incluso, presiones externas como cambios en la economía. Lo expuesto actúa para imponer estándares de desempeño de los proyectos cada vez más rigurosos y con menor margen de maniobra y error; por ende, requieren de una gestión de proyectos más rigurosa y minuciosa (Wallace, 2014), de manera que cobra aún más relevancia la adecuada gestión de proyectos.

Una parte importante del éxito de los proyectos recae en la apropiada preparación de los administradores de proyectos; muchos de ellos cuentan con perfiles técnicos y aprenden con base en la experiencia, adquiriendo en algunos casos, malas prácticas que son complicadas de eliminar (Gray, 2002), de manera que la formación pertinente es esencial para asegurar mejores resultados. En las empresas –al menos a nivel de Latinoamérica–, el uso de herramientas de gestión de proyectos es muy bajo. Este bajo nivel de uso y profesionalización se debe al desconocimiento, la falta de capacitación formal y, en algunos momentos, por influencia en las culturas empresariales locales, fundamentadas en la improvisación para la gestión de proyectos (Lledó y Rivarola, 2006). Incluso, cuando un proyecto es liderado por un administrador experimentado, siempre hay posibilidades de riesgo y que sufra retrasos, sobrecostos, problemas de calidad, entre otros. De este modo, existen muchas causas por las cuales un proyecto puede fallar (Shaker, 2010):

- Pobre alineación: cuando un proyecto no se alinea con los objetivos estratégicos de la organización, así como cuando no provee el valor deseado a los clientes.
- Mala planeación: es otro aspecto clave de la experiencia de un administrador de proyectos. Básicamente, si se falla en la planeación, se fracasará en la ejecución y los resultados a obtener.
- Falta de respaldo: si un administrador de proyectos y su proyecto no cuentan con respaldo ejecutivo, estarán destinados al fracaso.
- Requerimientos incompletos: es vital manejar la incertidumbre de las necesidades del proyecto conforme se avanza en su planeación y ejecución. Si no se sigue una adecuada gestión de los requerimientos, se está en peligro de caer en proyectos eternos.
- Expectativas no claras: diferentes involucrados pueden a un proyecto de formas distintas. Se debe gestionar de manera adecuada cómo cada uno de ellos espera algo del proyecto y cómo gestionar esos diferenciales sin ponerlo en riesgo.
- Fallas en el control del alcance: si no se tiene control sobre los cambios, el presupuesto y tiempo del proyecto se pueden ver comprometidos.
- Falta de recursos: es habitual que las organizaciones no tengan todos los recursos para ejecutar un proyecto y que los pocos que tengan estén participando en diversas actividades, lo cual pone en riesgo el resultado de los proyectos.
- Elección de tecnología: sin un adecuado análisis y planeación, se puede hacer uso de tecnologías costosas, complejas y difíciles de gestionar que, además de costosas, pueden alcanzar el éxito de los objetivos empresariales.
- Inexperiencia: por último, esta es la causa más común de fallas en los proyectos. Algunos administradores de proyectos no usan una metodología de trabajo adecuada, otros no están bien capacitados ni cuentan con la experiencia necesaria (Schibi, 2013).

Ahora, otra de las mayores causas relacionadas a proyectos fallidos tiene que ver con las técnicas de gestión de proyectos empleadas (Hardy-Vallee, 2012), principalmente porque estas no son siempre efectivas en todos los aspectos que un proyecto requiere. Los proyectos de desarrollo de software no son siempre similares; incluso, muchos de ellos pueden requerir factores emocionales (la experiencia que el cliente requiere) que algunas metodologías no involucran (Lindberg et al., 2010).

5.2 Metodologías, marcos de referencia y buenas prácticas de gestión de proyectos

Las organizaciones viven grandes retos cada día, muchos de ellos pueden ser solventados mediante la implementación de proyectos que permitan llevar a cabo acciones que parten de sus estrategias, ya sean de negocio, estructural, de visión, entre otras (Jovanovic y Beric, 2018). Dichos proyectos requieren del uso de metodologías modernas, marcos de referencia o buenas prácticas que colaboren a que el éxito de tales iniciativas tengan un mejor resultado.

Cada metodología cuenta con sus propios procesos, subprocesos, fases, terminología y demás; empero, un problema común es que cada una suele estar orientada a un tipo de proyecto y, por lo tanto, la gestión de proyectos con cada metodología suele ser distinta (Besner y Hobbs, 2012). Dependiendo de ciertos factores –como la particularidad de la organización donde se llevará a cabo el proyecto, el tipo de proyecto, gestión y cultura empresarial– se debe definir la metodología a elegir, lo que sustenta por qué existen diversas metodologías de gestión de proyectos (Jovanovic y Beric, 2018). En el caso particular de este trabajo, el área de gestión de proyectos es referente al desarrollo de software, el cual incrementa su relevancia dado su enfoque en la habilitación de automatizaciones de procesos y la provisión de nuevas funcionalidades a los consumidores finales (Gu et al., 2018).

5.3 Scrum

De acuerdo con la guía de *Scrum* (Schwaber y Sutherland, 2017), este es un marco de trabajo que posibilita entregar productos con el mayor valor posible, mientras se resuelven problemas complejos de una forma adaptativa, productiva y creativa. Se ha demostrado que *Scrum* es particularmente efectivo en los momentos de transferir conocimiento de manera iterativa e incremental, por lo que es usado de forma amplia para productos, servicios y la gestión de las organizaciones.

A lo anterior, cabe añadir que *Scrum* es simple, no cuenta con una gran cantidad de componentes entrelazados. Por otro lado, no es una metodología, sino que se basa en el método científico del empirismo, por lo que "*Scrum* reemplaza un enfoque algorítmico programado por uno heurístico, con respeto por las personas y la autoorganización para abordar la imprevisibilidad y resolver problemas complejos" (Scrum, s.f., párr. 7). El marco de trabajo de *Scrum* cuenta con los siguientes valores (Scrum, s.f.):

- Coraje: los miembros del equipo de trabajo deben tener el coraje para hacer lo correcto y trabajar en problemas complejos.
- Enfoque: todos se centran en el trabajo a desarrollarse en el *Sprint* y los objetivos del equipo de trabajo de *Scrum*.
- Compromiso: todos se implican personalmente en el cumplimiento de los objetivos del equipo.
- Respeto: todos los miembros del equipo se respetan mutuamente para ser personas capaces e independientes.
- Apertura: el equipo de trabajo y los involucrados se comprometen a tener apertura con respecto al trabajo y los retos venideros durante el desarrollo del proyecto.

De manera similar, *Scrum* está alineado con el manifiesto ágil, el cual hace énfasis en cuatro valores principales que deben soportar el desarrollo del software (Herrera y Valencia, 2007):

1. Los individuos y las interacciones por encima de los procesos y las herramientas: esto es, el ser humano es considerado como el factor principal de éxito versus los procesos, las técnicas y herramientas.
2. Software funciona por encima de la documentación: se hace hincapié en realizar documentación excelente, pero solo de lo estrictamente necesario, porque el enfoque principal debe ser en el desarrollo de software, lo que evita cualquier desperdicio de tiempo.
3. La colaboración con el cliente por encima de la negociación del contrato: el cliente forma parte del equipo de trabajo, de modo que tanto este como el equipo de trabajo busquen ir hacia una misma dirección, en vez de discutir términos contractuales y perder el rumbo; dado que el cliente es quien sabe lo que desea, es una piedra angular para realizar los ajustes necesarios durante el proyecto.
4. La respuesta al cambio por encima del seguimiento a un plan: se debe ser sumamente flexible para responder a los cambios en los mercados, las leyes y los requerimientos del cliente, de forma que se entregue el mayor valor posible y los objetivos sean alcanzados; de nada sirve terminar un plan que no genere los resultados que el cliente espera.

Asimismo, el manifiesto ágil cuenta con doce principios (Herrera y Valencia, 2007):

1. “Nuestra mayor prioridad es satisfacer al cliente mediante entregas tempranas y continuas de software con valor” (p. 383).
2. “Bienvenidos los cambios a los requerimientos, incluso los tardíos. Los procesos ágiles aprovechan los cambios para la ventaja competitiva del cliente” (p. 384).
3. “Liberar frecuentemente software funcionando desde un par de semanas a un par de meses, con preferencia por los periodos más cortos” (p. 384).
4. “Las personas del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto” (p. 384).
5. “Construir proyectos en torno a individuos motivados. Darles el entorno y apoyo que necesiten, así como confiar en ellos para que hagan mejor su trabajo” (p. 384).
6. “El método más efectivo y eficiente de compartir información a y dentro de un equipo de desarrollo, es la conversación cara a cara” (p. 384).
7. “El software funcionando es la medida de progreso” (p. 384).
8. “Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deberían ser capaces de mantener relaciones cordiales” (p. 384).
9. “La atención continua a la excelencia técnica y al buen diseño incrementan la agilidad” (p. 384).
10. “La simplicidad –el arte de maximizar la cantidad de trabajo no hech– es esencial” (p. 384).
11. “Las mejores arquitecturas, requerimientos y diseños emergen de los equipos auto organizados” (p. 385).
12. “En intervalos regulares, el equipo reflexiona sobre como volverse más efectivo, entonces afina y ajusta su comportamiento como corresponde” (p. 384).

Un requerimiento en el mundo del desarrollo de software se define como una condición, capacidad o la representación que un sistema o solución debe tener para satisfacer un contrato, estándar u objetivo. Al igual que otros métodos y metodologías ágiles, *Scrum* usa la herramienta denominada “historias de usuario”, la cual posibilita expresar requerimientos de manera sencilla y clara, mediante el establecimiento de conversaciones acerca de las necesidades de los clientes. Estas conversaciones se basan en descripciones cortas y simples acerca de una necesidad que debe ser considerada en el desarrollo del software (Izaurralde, 2013). Por otro lado, los roles dentro del presente marco son (Schwaber y Sutherland, 2017):

- *Product owner*: es el responsable de maximizar el valor del producto resultante del trabajo del equipo de desarrollo. La forma en que se hace esto puede variar entre organizaciones, equipos de *Scrum* e individuos. Esta figura es una persona, no un comité, aunque puede representar a un comité que hace los requerimientos del lado del negocio. La empresa debe dar autoridad al *product owner* para tomar decisiones en lo referente al producto o servicio que se desarrolla.
- *Scrum master*: responsable de promover y cuidar que se sigan los lineamientos establecidos en la guía de *Scrum*, vigilando que se ejecute su teoría, prácticas, valores y reglas. Este rol es el de un líder que sirve al equipo de trabajo, pues ayuda a ser un vínculo entre el equipo y los entes externos para facilitar su interacción de la forma adecuada, con ello se maximiza el valor entregado con el trabajo en desarrollo. Además, es el responsable de que el equipo de trabajo entienda los requerimientos y las necesidades expresados por el *product owner*, entre otros.
- Equipo de trabajo: conjunto de profesionales que hacen el trabajo necesario para entregar valor y resultados al final de cada *Sprint*. Son equipos autodirigidos, con especialidad en algún dominio de la empresa, pero que entienden que el éxito de los proyectos no radica en lo individual, sino en los resultados globales como equipo. Son grupos pequeños, pero con capacidades suficientes para desarrollar el trabajo requerido.

Los eventos prescritos se usan en *Scrum* para crear regularidad y minimizar la necesidad de reuniones no definidas. Estos eventos son reuniones preestablecidas, de manera que no se abra la posibilidad a reuniones innecesarias que no generen valor y reduzca el tiempo efectivo de trabajo del equipo; todos los eventos tienen un calendario (Scrum, s.f.). Una vez que comienza un *Sprint*, su duración es fija y no se puede acortar o alargar. Los eventos restantes pueden finalizar siempre que se logre el propósito del evento, asegurando que se pase una cantidad de tiempo adecuada sin permitir desperdicio en el proceso. Los eventos prescritos de *Scrum* son (Schwaber y Sutherland, 2017):

- *Sprint*: es el corazón de *Scrum*; es un evento con una duración de un mes (o menos) donde se desarrolla un incremento sólido y usable del producto deseado. Una vez que un *Sprint* finaliza, comienza inmediatamente el otro. Cada *Sprint* consiste en:
    - Planificación del *Sprint*.
    - *Standup meetings*.

  - Trabajo de desarrollo.
  - Revisión del *Sprint*.
  - Retrospectiva del *Sprint*.

  Por otro lado, durante el *Sprint*:

  - No se aceptan ni realizan cambios que afecten el cumplimiento de los objetivos del *Sprint*.
  - No se realizan cambios a la duración del *Sprint*.
  - No se reducen los objetivos de calidad.
  - El alcance puede ser renegociado entre el equipo de trabajo y el *product owner*.

  Se busca que un *Sprint* no tenga una duración mayor a un mes, dado que con un plazo mayor hay más riesgo de perder el enfoque en la definición del proyecto, se incrementa la complejidad y el riesgo. El espacio de un mes como máximo posibilita entregar valor rápidamente y realizar los cambios que se consideren pertinentes en caso de que algo no vaya bien, sin tener que hacer un retrabajo significativo.

- Planificación de *Sprint*: se ejecuta al inicio de cada *Sprint*; en este punto, se planea el trabajo a ser desarrollado. Este plan es creado de forma colaborativa entre todo el equipo de trabajo. Por cada mes de trabajo a llevarse a cabo, esta planificación no debe exceder las ocho horas de duración. Es tarea del *Scrum master* que los involucrados entiendan el alcance de lo planeado y, de forma inicial, que la reunión se realice. Se deben resolver las siguientes dos preguntas:
  - ¿Qué debe ser entregado al final del *Sprint*?
  - ¿Cómo se podrá hacer el trabajo necesario para cumplir los objetivos del *Sprint*?
- *Standup meeting*: reunión diaria entre los miembros del equipo de desarrollo y el *Scrum master* que no debe exceder los quince minutos de. Básicamente, cada miembro del equipo debe responder las siguientes preguntas:
  - ¿En qué trabajase ayer?
  - ¿Qué harás hoy?
  - ¿Tienes algún bloqueo que ponga en riesgo tu avance?

  De hecho, el *Scrum master* es el responsable de despejar esos bloqueos para que el equipo de trabajo cumpla sus compromisos para el *Sprint*. Los miembros del equipo de desarrollo son los responsables de llevar la reunión, aunque el *Scrum master* debe

asegurarse de que esta ocurra, así como de evitar que la interrumpan invitados externos. Este tipo de reuniones facilitan la comunicación, evitan otras reuniones innecesarias e incrementan el nivel de conocimiento del equipo de trabajo.

- Revisión de *Sprint*: reunión informal al terminar cada *Sprint*, donde el *product owner* y el equipo de trabajo analizan las tareas desarrolladas y lo que no se haya podido realizar. Se actualiza el *product backlog*, que es una lista de los requerimientos ejecutados para el producto o servicio a desarrollar; igualmente, se discute sobre lo que salió mal y cómo se solventó, en aras de incrementar la base de conocimiento del proyecto y el equipo.
- Retrospectiva de *Sprint*: reunión desarrollada después de la revisión de *Sprint* y antes de la planeación del siguiente *Sprint*. Aquí, de una forma positiva y productiva, en una duración no mayor a tres horas para un *Sprint* de un mes, se analiza lo que salió bien, lo que salió mal, lo que se debe repetir y lo que no. Esta reunión facilita la mejora continua en el equipo, puesto que se planea aquello que se puede mejorar en las prácticas de trabajo actuales y anteriores.

Por su parte, *Scrum* también cuenta con artefactos, los cuales:

> Representan el trabajo o el valor para proporcionar transparencia y oportunidades de inspección y adaptación. Los artefactos definidos por *Scrum* están diseñados específicamente para maximizar la transparencia de la información clave para que todos tengan la misma comprensión del artefacto. (Scrum, s.f., párr. 17), los cuales son:

- *Product backlog*: es una lista ordenada de todo lo que el producto o servicio requiere. Es la única fuente de requerimientos (y sus cambios) a ser realizados para entregar el producto o servicio. El *product owner* es su responsable, tanto del contenido como de la disponibilidad y el orden. Es un artefacto dinámico, pues es plenamente aceptado que cambie durante la evolución del producto o servicio. El *product backlog* enumera todas las características, las funciones, los requisitos, las mejoras y las correcciones que constituyen los cambios que se realizarán en el producto en futuras versiones. Los elementos de la cartera de productos tienen los atributos de una descripción, orden, estimación y valor; asimismo, a menudo incluyen descripciones de prueba que demostrarán su integridad cuando esté hecho (Schwaber y Sutherland, 2017).

- *Sprint backlog*: es el set de ítems del *product backlog* seleccionados para desarrollarlos en el *Sprint* actual, así como un plan para hacerlo posible. Es un pronóstico del equipo de trabajo acerca de las funcionalidades a ser entregadas al finalizar el *Sprint*, además del trabajo necesario para desarrollarlo. El equipo de trabajo puede hacer los cambios que considere necesarios para enderezar el rumbo con el propósito de cumplir con los objetivos del *Sprint*.
- Incremento: es la suma de todos los ítems del *product backlog* completados durante un *Sprint* y de los anteriores.

A partir de lo mencionado, se aprecia que *Scrum* es un marco extremadamente flexible y sencillo que permite una interacción abierta entre los actores del equipo de trabajo. De este modo, la entrega temprana de valor hace factible el contar con una idea pronta de por dónde va el proyecto, así como probar las características solicitadas. No obstante, hay diversas características de *Scrum* que se precisan mejorar, lo cual puede lograrse al complementarlo con otros marcos, como se ve más adelante. Entre las características a mejorar se aluden las siguientes:

- *Scrum* da por hecho que el requerimiento inicial que forma parte del *product backlog* es la totalidad de lo requerido, que contiene la visión completa de la necesidad y considera a los usuarios. Es frecuente que el cliente no siempre sepa definir y explicar claramente su necesidad, por lo cual, es posible comenzar a desarrollar una solución corta o con la visión de solo algunos usuarios potenciales.
- El lema de "bienvenidos los cambios, aunque sean tardíos" que forma parte del manifiesto ágil no es realista. Sin un control adecuado de los requerimientos, el desarrollo se puede volver eterno. Por ello, es vital fortalecer el proceso de levantamiento de requerimientos para evitar cualquier suposición que se vuelva una espiral de solicitud de cambios durante el proceso de desarrollo. Es importante recalcar que cualquier equipo de desarrollo de software tiene una capacidad límite que debe cuidar.
- Incluso cuando el enfoque de *Scrum* sea hacia el ser humano, esto no implica que la toma de decisiones relativas al proyecto lo considere en su totalidad. El vínculo entre el equipo de desarrollo y los clientes es el *product owner*, actor que es una persona, no un comité (Schwaber y Sutherland, 2017). Si este actor no es capaz de traducir todas las necesidades, expectativas y realidades de cada usuario de la

solución requerida en forma de requerimientos al equipo de desarrollo, la solución desarrollada no será aceptada ni utilizada por ellos.

Las pruebas realizadas por el equipo de *Scrum* después de cada *Sprint* consideran obtener retroalimentación temprana por parte del *product owner* acerca del incremento obtenido. Sin embargo, *Scrum* no cuenta con un mecanismo estructurado para integrar los posibles cambios y comentarios al *product backlog*. La guía de *Scrum* solo menciona que el *product owner* es el responsable de agregar tales nuevos requerimientos (Schwaber y Sutherland, 2017), pero si la visión inicial del proyecto plasmada en estos no muestra la realidad de lo que los clientes necesitan, se comenzará con procesos de desarrollo e implementación de parches; ahí es en donde la introducción de otros marcos al flujo de trabajo ayudarán a entender si el desarrollo va por el camino adecuado, si es necesario replantear el proyecto o si solo necesita algunos cambios.

5.4 Design Thinking

Hoy más que nunca la innovación está presente en las organizaciones, las cuales buscan crear más valor de forma dinámica. Debido a la creciente globalización, los cambios en los mercados y enormes pasos en la digitalización de las empresas, una buena idea puede poner rápidamente en el espectro a una organización. A pesar de ello, las ideas no vienen siempre de las empresas de nicho, por ejemplo, Uber no fue creada por una empresa de taxis (Langenfeld, 2019). David Kelley, creador de la agencia de diseño IDEO (2015), introdujo el pensamiento de diseño (*Design Thinking,* en inglés) al ámbito económico, debido a que antes solo era considerado en el científico. Luego, el *Design Thinking* cobró fuerzas en la Universidad de Stanford debido a la fundación de un instituto llamado D.School patrocinado por la empresa SAP (Langenfeld, 2019).

Generalmente, el *Design Thinking* es definido como un proceso analítico y creativo (Razzouk y Shute, 2012) “que impregna todo el espectro de actividades de innovación con una filosofía de diseño centrado en las personas” (Brown, 2008, p. 1); se considera una disciplina que toma en consideración la sensibilidad de un diseñador para consolidar las necesidades de las personas con lo que es tecnológicamente viable y acorde con una estrategia de negocios que genere valor al cliente y una oportunidad de mercado (Brown, 2008). El proceso comienza cuando se ponen manos a la obra y se entienden las preguntas correctas, de modo que se trata

de adoptar cambios de mentalidad simples y abordar los problemas desde una nueva dirección. Esta metodología es esencialmente útil para las organizaciones en los siguientes aspectos (Brown, 2008):

- Mejor entendimiento de las necesidades no cubiertas de la gente para la que se está creando.
- Reduce el riesgo asociado a la creación de un nuevo producto, nuevas ideas y servicios.
- Genera soluciones revolucionarias, no tanto incrementales.
- Se aprende e itera rápido. (IDEO U, s.f., párr. 3)

Por otro lado, cabe añadir que el *Design Thinking* consta de seis fases (Langenfeld, 2019):

1. Inmersión o empatía: se trata de entender la situación en la que el equipo de *Design Thinking* y el cliente o usuario se encuentran. Entre mayor sea la inmersión, más información se obtendrá para uso en las siguientes fases. Implica ponerse en los zapatos del cliente para comprender su necesidad y caminar en su mundo. Observar al usuario es más eficaz que entrevistarlo.
2. Definición: se analiza y organiza la información que se ha recopilado en la fase anterior; esta se basa en encontrar similitudes y relaciones. Se define el cliente, sus necesidades y las particularidades que el producto o servicio debe llevar para cada perfil de este.
3. Ideación: es la fase donde se busca –y se espera encontrar– las ideas. Este proceso puede ser largo, pues hay que escoger las buenas ideas de la cantidad de ellas obtenidas. Implica pensar cómo resolver el problema, planteando distintas soluciones para ello.
4. Prototipos: se construyen prototipos, ya sean objetos o conceptos; un prototipo es la visualización de una idea.
5. Prueba: los prototipos son probados al final, para hallar los que se consideran potenciales para solventar la necesidad. No se trata de observar si un prototipo es funcional, sino cómo el cliente percibe la idea. Fracasar también forma parte de las pruebas, es la única manera de mejorar el prototipo.

6. Implementación: más que una sexta fase, se considera la culminación del proceso, donde se ponen en práctica las ideas elegidas para confirmar que funcionan acorde con lo que el cliente espera.

En síntesis, *Design Thinking* no es un modelo lineal ni hay hitos como en la gestión de proyectos tradicional; este es un proceso iterativo. Por otro lado, *Design Thinking* no es la clave para resolverlo todo. Finalmente, es preciso agregar que algunas fases de *Design Thinking* ocurren en paralelo, antes o después de algún proceso, de una de las metodologías, marcos de referencia o buenas prácticas (Langenfeld, 2019).

5.5 Lean Startup

El emprendimiento siempre ha estado relacionado con la exposición ante el riesgo, lo cual históricamente se ha visto con malos ojos o como algo a lo que temer; con plena justificación, dado el enorme porcentaje de *startups* que fracasan, que llega a cerca del 75 % (Llamas y Fernández, 2018). Uno de los tantos emprendedores que sufrió las consecuencias de los riesgos negativos en sus iniciativas de emprendimiento decidió crear una metodología que permitiera alcanzar mejores resultados, creando valor para los clientes, mientras tomaba los riesgos y fallos como insumos del éxito y no como piedras de tropiezo. Esta metodología fue llamada *Lean Startup* y su creador es Eric Ries.

En este orden de ideas, Ries (2012) expuso que el *Lean Startup* ayuda a crear empresas de éxito a partir de la innovación continua, con lo que afirmó que el éxito de las *startups* no radica en estar en el lugar correcto y en el momento adecuado, sino a partir del diseño de un proceso correcto que implica la creación del producto que el cliente realmente requiere y está dispuesto a pagar, con la cantidad mínima y necesaria de recursos. En esta concepción, el fracaso es parte de los ingredientes vitales de *Lean Startup*, puesto que permite tener retroalimentación temprana de los clientes. Esto es contrario al modelo empresarial típico, donde se tiene la retroalimentación del clientes después de desarrollar un plan de negocio, obtener el financiamiento, crear un producto y lanzarlo; momento donde muchos emprendedores se dan cuenta que crearon algo que el cliente no necesitaba (Llamas y Fernández, 2018). Ahora, el *Lean Startup* cuenta con cinco principios (Ries, 2012):

1. Los emprendedores están en todas partes: un emprendedor es considerado todo aquel que trabaje dentro de una *startup*, de acuerdo con el autor del *Lean Startup*,

esta se percibe como una institución humana que se ha creado para generar productos y servicios nuevos en condiciones de incertidumbre extrema; esto es aplicable para cualquier sector productivo, sin importar su tamaño.

2. El espíritu emprendedor es gestión: una *startup* es una institución y no un producto o servicio, por ello, requiere un tipo de gestión orientado a condiciones de incertidumbre extremas.
3. Aprendizaje validado: una *startup* no solo existe para obtener ganancias ni atender a sus clientes, sino para aprender de forma permanente cómo crear negocios sostenibles.
4. Crear, medir y aprender: este es un proceso de retroalimentación continua, el cual es la actividad fundamental de una *startup*, que se basa en convertir ideas en productos, analizar la respuesta de los consumidores y aprender cuándo pivotear o perseverar. Pivotear es dar un giro de rumbo drástico basado en la retroalimentación que dan los consumidores e integrarla en un producto que deberá ser mostrado de nueva cuenta para buscar retroalimentación; mientras que perseverar es cuando se busca continuar con la idea y el producto con que se cuenta, para buscar más retroalimentación e, incluso, para su liberación y presentación al mercado.
5. Contabilidad de la innovación: se refiere a medir el progreso, establecer hitos y priorizar tareas, esto acorde con el ambiente de innovación de las *startups.*

Los orígenes de *Lean Startup* se remontan a la revolución del *Lean Manufacturing* que Toyota llevó a cabo en la posguerra. El pensamiento *Lean* cambió radicalmente la forma de organizar las cadenas de oferta y los sistemas enteros de producción, que entre otros, muestra la diferencia entre actividades que generan valor y las que no, acerca del desperdicio y expone cómo incorporar calidad en lo que se hace, esto es, procesos y productos (Ries, 2012). *Lean Startup* adapta las ideas del *Lean Manufacturing* al ambiente emprendedor, proponiendo una nueva manera de medición de progreso. En los sectores industriales, el progreso se mide a través de la producción de bienes y servicios, pero el *Lean Startup* introduce una nueva medida de progreso denominada conocimiento validado (Ries, 2012).

El conocimiento validado es un procedimiento de demostración empírica donde el equipo de trabajo descubre información valiosa sobre las verdaderas oportunidades presentes y futuras del negocio. Este proceso suele ser más concreto, riguroso y rápido que los métodos

de previsión y planificación clásicos (Ries, 2012). Es pertinente añadir que la filosofía Lean indicada proviene del mundo de la manufactura y cuenta con los siguientes cinco principios (Ghezzi y Cavallo, 2020):

1. Crear valor para el cliente: implica que el valor se crea cuando se reducen los desperdicios en los procesos internos y los costos disminuyen. El valor se incrementa cuando se entregan productos y servicios nuevos y mejores al cliente.
2. Identificar el flujo de valor: comprende dotar de transparencia relacionada a los costos de operación.
3. Crear flujo: se refiere a generar agilidad, evitando detenciones que afecten altos en la producción.
4. Producir solo lo que es requerido por el cliente: involucra dotar de alta capacidad de respuesta la producción para generar solo lo que el cliente requiere, siendo así más eficientes y efectivos.
5. Perseguir la perfección: de manera continua se detectan y eliminan desperdicios.

El método de conducción de una *startup* se basa en ajustes constantes en vez del típico modelo empresarial fundamentado en la creación de planes complejos que se cimientan en supuestos; lo que se realiza a partir del llamado circuito de retroalimentación de *Lean Startup* (Llamas y Fernández, 2018):

- Crear. cuando una *startup* nace, no se cuenta con los suficientes datos para crear un producto o servicio que se adapte completamente a las necesidades de los clientes. Para ello, es importante construir una primera versión del producto o servicio que cuente con las características mínimas esenciales que permitan obtener la suficiente retroalimentación y que no implique dedicarle demasiado tiempo como para que sea doloroso dar un paso atrás o un giro drástico, la cual es denominada Producto Mínimo Viable (PMV).
- Medir: implica medir cómo responden los consumidores y, a partir de ello, tomar las decisiones siguientes.
- Aprender: aprender de la retroalimentación de los clientes para determinar si es necesario pivotear o perseverar en la idea desarrollada.

De acuerdo con este circuito, se conoce cuándo es necesario dar un giro de rumbo drástico (pivote) o si se debe perseverar en el camino que se sigue. De este modo, el producto

o servicio es el resultado de la estrategia de las *startups*, la cual se conforma del diseño de una misión, visión y estrategia. La misión define quién es la *startup* y la visión determina el objetivo a alcanzar, el cual debe estar alineado con la creación de un negocio próspero que ayude a cambiar al mundo. Por ejemplo, se puede llegar a la visión a partir de la misión usando una estrategia; en este caso, puede ser un modelo de negocio, un mapa de productos, entre otros (Ries, 2012). Cabe mencionar que la visión suele no cambiar dado que indica el rumbo que las acciones deben tomar. Si la estrategia debe modificarse, se pivotea para dar un giro drástico que permita tener mejores resultados. Si el producto tiene que cambiar, se le llama realizar una optimización de producto (Ries, 2012).

En síntesis, el *Lean Startup* propone crear negocios mediante el uso del circuito ágil mencionado, donde una vez establecidas ciertas hipótesis y suposiciones iniciales, se desarrolla un PMV que funge como un instrumento de experimentación con la menor inversión posible. La retroalimentación de los clientes hacia el PMV sirven como un indicativo de la posible aceptación que el producto o servicio tendrá en el mercado. Si la retroalimentación es positiva, se deberá construir sobre el PMV para dotarlo de más funcionalidades, las cuales se deberán poner a escrutinio recurrente de los clientes potenciales. Por el contrario, si la retroalimentación es negativa, se deberá dar otro enfoque al negocio, lo que se denomina como pivotear (Llamas y Fernández, 2018); lo anterior se basa en el rápido aprendizaje mediante la inversión de los menores recursos posibles.

5.6 Lean Manufacturing

El desarrollo de esta serie de principios y herramientas de gestión de la producción se puede ubicar en el pionero y emblemático caso de *Toyota Motor Corporation*, el cual se ha convertido en una alternativa que ha mostrado gran versatilidad al ser adoptada en diversos escenarios del sector industrial. En la década de los ochenta, *Toyota Motor Corporation* estaba desarrollando un modelo de producción que permitía mejorar su eficiencia, productividad y competitividad (Sarria et al., 2017), el cual se consolidó en 1991. De acuerdo con la filosofía *Lean*, existen siete tipos de desperdicios dentro de los sistemas de manufactura (Ohno, 1988):

- Sobreproducción: producir de más, sin un requerimiento previo.
- Transporte: cuando se transportan partes terminadas o material prima de forma innecesaria cuando, quizá, todo se puede ensamblar en un solo sitio.

- Movimiento: cuando se realizan movimientos de materia prima o material terminado de forma innecesaria, que involucra personal o equipo innecesario.
- Esperas: aumentar el tiempo de ciclo de producción, que impide producir más con lo que se tiene.
- Inventario: el inventario cuesta y está relacionado con la sobreproducción, así como con el exceso de materia prima.
- Proceso innecesario: trabajo innecesario o que no genera valor.
- Partes o productos defectuosos: genera basura o retrabajos, así como problemas de calidad con los clientes.

Posteriormente, se propuso que aquel producto o servicio que no satisficiera las expectativas del cliente se considerara como algún tipo de desperdicio (Jones y Womack, 2003). Desde el punto de vista tecnológico, *Lean Manufacturing* puede ser considerado como un complemento de la automatización (Satoglu et al., 2017). De tal manera, la integración de la industria 4.0 con producción tipo Lean es expresada como *Lean Automation*, que busca una mayor variabilidad y flujos de información más cortos para satisfacer las demandas futuras del mercado.

En este sentido, una de las herramientas del *Lean Manufacturing* es el TPM (*Total Productive Maintenance*), que busca mejorar la eficiencia de los equipos de producción, incluyendo la reducción de sus tiempos de operación, incrementar su velocidad y reducir las pérdidas de calidad. Por ejemplo, se pueden agregar sensores que, gracias a analítica avanzada, notifiquen comportamientos anormales y tomen acciones tempranas antes de la generación de desperdicios. Otra herramienta es llamada Kanban, con la que se busca que siempre haya el material mínimo necesario para producir, el cual se puede surtir mediante sensores que detecten su consumo automático y se eviten capturas manuales de información con escáneres y etiquetas con radiofrecuencia para el adecuado rastreo del material.

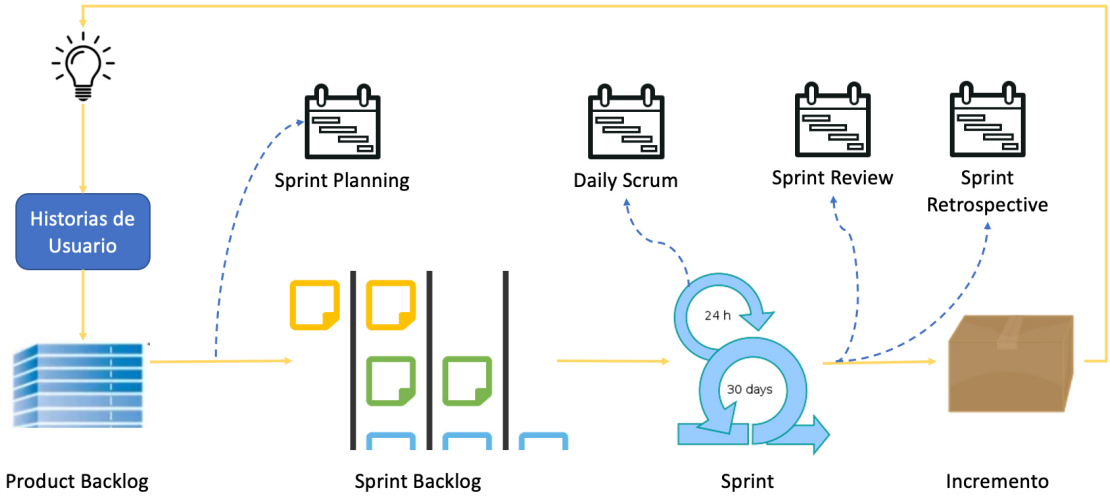
Por su parte, el WIP (*Work in Progress*) permite conocer el estado actual de la producción y su trazabilidad, que puede ser mejorada drásticamente con la implementación de analítica avanzada, comunicación Máquina a Máquina (M2M), internet de las cosas (IoT) y sensores. Finalmente, todos los ejemplos explicados requieren el desarrollo de componentes de software que habiliten la intercomunicación de sus componentes y que la información generada sea explotada de forma eficaz y segura.

6 Metodología

Este capítulo presenta la hoja de ruta del presente trabajo mediante el desarrollo de una propuesta tecnológica; para su preparación, se utilizó *Scrum* como el método que guio su creación de forma estructurada. Así, a partir de *Scrum* como el método utilizado y siguiendo los lineamientos de la guía de Schwaber y Sutherland (2017), se buscó de manera sencilla, ágil, iterativa e incremental crear los elementos que, de forma conjunta, conformaron la propuesta tecnológica final que busca solventar las problemáticas expuestas como sustento de este trabajo. En este sentido, la Figura 1 muestra de manera visual el proceso de trabajo de *Scrum*:

Figura 1

Proceso visual de Scrum. Fuente: diseño propio.



Tal proceso, sus pasos y artefactos inspiraron la creación de otro procedimiento más sencillo que sirvió para la elaboración de la propuesta tecnológica y que se expone en la Figura 2:

Figura 2

Proceso a utilizar. Fuente: Diseño propio.

La Figura 2 muestra los pasos esenciales que se ejecutaron para la preparación de la propuesta tecnológica. En el primer punto del proceso mostrado, el cual se denomina "Definición de historias de usuario", se buscó definir la serie de expresiones que dictan el rumbo del trabajo de los distintos ciclos posteriores denominados *Sprints*. En el segundo punto (las iteraciones), internamente y de manera cíclica, se tienen tres actividades cuyo diseño fue pensado para repetirse las veces necesarias y donde se llevó a cabo el trabajo que permitió la generación de la propuesta tecnológica resultante. A continuación, se muestra con mayor nivel de detalle los pasos indicados en el proceso de la Figura 2, método utilizado para la construcción de la propuesta tecnológica.

6.1 Definición de las historias de usuario

Las historias de usuario son consideradas como un enfoque ágil que permite describir los requerimientos de los clientes que un software debe contener y satisfacer, de manera sencilla y clara, a partir del establecimiento de una conversación desde la perspectiva de la persona que requiere tales características, usualmente con el siguiente formato (Izaurralde, 2013): "Como <tipo de usuario>, necesito <algún objetivo> para poder <alguna razón>".

De esta forma, se pueden especificar con diferentes niveles de detalle los requerimientos de los clientes desde sus propias perspectivas, cubriendo incluso múltiples

funcionalidades. En este sentido, la creación de las historias de usuario se basó en lo expresado en el planteamiento del problema, donde se definió lo siguiente:

1. Se lleva a cabo un proceso de desarrollo ágil, pero al final de este, no se entrega todo el valor que el cliente espera, es decir, las expectativas no son cubiertas.
2. La solución entregada tiene ciclos de vida cortos y adolece de escalabilidad.
3. Falta de enfoque en el ser humano.

Dado lo anterior, se definieron las siguientes historias de usuario, cimentadas en las definiciones de dichas problemáticas:

- Como usuario, quiero que mis necesidades y expectativas sean plenamente entendidas para que la solución que me desarrollarán me sea de valor y útil.
- Como usuario, quiero que la solución que me han de desarrollar sea creada de manera ágil y flexible, para efectos de ver prontamente los resultados que se han obtenido y corregir los errores a tiempo.
- Como usuario, quiero que antes de comenzar con el desarrollo de la solución, el equipo de trabajo comprenda mi realidad, mis problemáticas y mi día a día para que el resultado final se alinee conmigo y no con la preconcepción del equipo de desarrollo.
- Como usuario, quiero que el enfoque en la toma de decisiones y el desarrollo de la solución sea yo y no las funcionalidades ni los objetivos, de modo que el resultado sea una herramienta de utilidad en el logro de mis objetivos y no solo un instrumento más que usar o que complique más mi vida diaria.

Las historias de usuario obtenidas integran el *product backlog*, el cual contiene los requisitos que el equipo de desarrollo deberá solventar mediante los incrementos creados en cada iteración.

6.2 Iteraciones

El proceso de iteraciones consiste en llevar el número de ciclos que sean necesarios para efecto de crear un producto que considere todas las historias de usuario obtenidas previamente; dentro de cada iteración, se llevaron a cabo los siguientes eventos:

1. Planeación de *Sprint*: donde se definió el trabajo a realizar dentro de la iteración en turno al crear el denominado *Sprint backlog*.
2. *Sprint*: donde se hizo la ejecución del trabajo seleccionado dentro del *Sprint backlog*.
3. *Sprint review*: donde se analizó qué trabajo se desarrolló, qué se creó y, si es necesario, indicar si quedó algo pendiente a desarrollar de lo planteado en el *Sprint backlog.*

A partir de esto, se encontró la necesidad de llevar a cabo cinco iteraciones; en cada una de ellas se realizó lo siguiente:

- Iteración 1: creación del marco teórico.
- Iteración 2: elaboración de pregunta de estudio y constructos teóricos.
- Iteración 3: obtención de información adicional.
- Iteración 4: análisis cualitativo.
- Iteración 5: creación de la propuesta final.

De acuerdo con lo anterior, por cada *Sprint* definido, se indican el trabajo seleccionado a realizar, el trabajo que se llevó a cabo y el resultado de cada iteración. El resultado final que integra todos los incrementos generados se muestra en el séptimo capítulo, “Propuesta” donde se plantea la propuesta desarrollada.

6.2.1 Iteración 1: creación del marco teórico

6.2.1.1 **Planeación del Sprint.**

En esta iteración, se buscó la creación del marco teórico del presente trabajo para efectos de contextualizar las problemáticas planteadas, así como obtener los elementos que sustenten los componentes de la solución que se desarrolló y que se verá más adelante. Entre otros, la iteración consideró la obtención, clasificación, selección de información y creación del marco teórico.

6.2.1.2 **Sprint.**

Se obtuvo la documentación necesaria y se creó el marco teórico, el cual fue la estructura vital de estudio para esbozar la presente propuesta tecnológica.

6.2.1.3 Sprint Review.

Con las problemáticas definidas, así como con un marco teórico construido, se procedió a la siguiente iteración, en donde se buscó, mediante un enfoque cualitativo, desarrollar una pregunta de estudio, sus preguntas secundarias y sus constructos teóricos. Cabe resaltar que para efectos de dar respuesta a lo definido en el planteamiento de la problemática, se utilizó parte de la técnica de estudio de caso a partir de un análisis cualitativo, del cual se verán detalles más adelante.

6.2.2 Iteración 2: elaboración de pregunta de estudio y constructos teóricos

6.2.2.1 Planeación del Sprint.

Con la presenta iteración, con la utilización de un enfoque cualitativo, se pretendió desarrollar una pregunta de estudio y sus preguntas de segundo nivel que sirvieran de base para responder a las necesidades y problemáticas exploradas. Desde el enfoque cualitativo, se utiliza como cimiento de la investigación información no numérica que se extrae de procesos de recolección no estandarizados. Usualmente, con este enfoque, se pretende dar solución a preguntas de investigación previamente establecidas, sin probar la hipótesis (Buelvas y Rodríguez, 2021).

6.2.2.2 Sprint.

En el presente *Sprint*, se definieron tanto la pregunta de estudio como las preguntas de segundo nivel y sus respectivos constructos teóricos. Para entrar en materia relativa al objetivo del presente S*print*, a partir del marco teórico y del objetivo, se identificó que la exploración se debía enfocar en el entendimiento de cómo, en su uso como única herramienta de gestión de proyectos ágiles de desarrollo de software, *Scrum* es insuficiente y genera situaciones que afectan a las organizaciones en la búsqueda de sus objetivos.

Por lo tanto, se desprendió la siguiente pregunta de estudio: ¿cómo complementar las prácticas de *Scrum*, de manera que permita considerar al ser humano como el centro de la toma de decisiones, desarrollando las soluciones exactas que este requiere? De igual forma, se desglosaron los siguientes constructos teóricos que fueron ubicados durante el estudio de literatura de soporte del caso en el marco teórico:

1. *Scrum* como complemento y no como la herramienta definitiva de gestión ágil de proyectos de desarrollo de software.
2. El ser humano, sus necesidades y experiencias como factores clave del entendimiento de las problemáticas.
3. Uso del método científico para entender y validar si se está construyendo el producto o servicio que el cliente realmente necesita.

De acuerdo con los constructos teóricos que componen la pregunta de estudio expuesta, se conformó lo que Yin (2019) consideró como las proposiciones de estudio, las cuales sirvieron de punto de partida para la recolección de datos y análisis de estos (Martínez, 2006). Con base el primer constructo teórico, se definió la siguiente proposición de estudio:

P1. *Scrum* como herramienta complementaria que permita gestionar proyectos de desarrollo de software de manera ágil y eficiente.

Asimismo, a partir del segundo constructo teórico, se planteó la siguiente proposición de estudio:

P2. Lograr el entendimiento de las necesidades y experiencias del ser humano ayudan a entender mejor sus problemáticas.

Finalmente, con el tercer constructo teórico, se estableció la siguiente proposición de estudio:

P3. El uso del método científico ayuda a validar y entender si se está trabajando en la solución ideal para las problemáticas encontradas.

Identificadas las proposiciones de estudio, las cuales fueron el origen del diseño de la presente exploración, se encontraron las siguientes preguntas de segundo nivel que ayudan a determinar de manera más clara aquello que se busca en el caso de estudio y que se encuentra relacionado con las proposiciones establecidas. Estas preguntas de estudio de segundo nivel son la referencia utilizada para establecer las fuentes de información y para diseñar sus instrumentos de levantamiento (Maxwell, 1996; Yin, 2009). A continuación se listan las preguntas de segundo nivel identificadas para el caso de estudio:

De la proposición de estudio número 1: *Scrum* como herramienta complementaria que permita gestionar proyectos de desarrollo de software de manera ágil y eficiente.

Preguntas de Segundo Nivel:

- ¿Cuáles son las características más notables de *Scrum* que impulsan su adopción y uso?
- ¿Cuáles son las áreas de oportunidad con que cuenta *Scrum*?

De la segunda proposición de estudio, lograr el entendimiento de las necesidades y experiencias del ser humano ayudan a entender mejor sus problemáticas, se plantearon las siguientes preguntas de segundo nivel:

- ¿Cómo conocer mejor las necesidades y experiencias de una persona ayudan entender mejor sus problemáticas y requerimientos?
- ¿Qué nuevas herramientas habría que utilizar para cumplir con tal fin?
- ¿Qué adiciones o cambios requiere *Scrum* en sus procesos para entender las necesidades y experiencias del ser humano como proceso de levantamiento de requerimientos?

De la tercera proposición de estudio, el uso del método científico ayuda a validar y entender si se está trabajando en la solución ideal para las problemáticas encontradas, se definieron las siguientes preguntas de segundo nivel:

- ¿Cómo el uso del método científico permite determinar si la solución que se está desarrollando o por desarrollar es la adecuada para el cliente, sigue vigente y será adoptada con éxito?
- ¿Qué nuevas herramientas habría que utilizar para cumplir con tal fin?
- ¿Qué adiciones o cambios requiere *Scrum* en sus procesos para integrar el uso del método científico en la validación de la valía y vigencia de la solución en desarrollo?

6.2.2.3 Sprint Review.

En el presente *Sprint*, se definieron tanto la pregunta de estudio, las preguntas de segundo nivel y sus constructos teóricos. Las preguntas de estudio de segundo nivel fueron el cimiento para determinar las fuentes de evidencia. Con base en ellas, se determinó la literatura y los medios adicionales a consultar. De este modo, después de determinar la pregunta de estudio, los constructos que se derivan de las problemáticas del caso y las preguntas de primer y segundo

nivel, se llevaron a cabo las actividades relacionadas con la obtención de información adicional en las siguientes iteraciones.

6.2.3 Iteración 3: obtención de información adicional

6.2.3.1 Planeación del Sprint.

El *Sprint Backlog* de la presente iteración se compuso de la búsqueda de los elementos teóricos adicionales que ayudaron a sustentar los constructos y las preguntas de estudio definidos; para ello, se utilizó la plataforma Google Scholar. De igual manera, se empleó el repositorio del Instituto Tecnológico de Estudios Superiores de Occidente para la consulta y el estudio tanto de trabajos de obtención de grado como de los recursos bibliográficos que pone a disposición dicha institución. Por otro lado, se usó bibliografía especializada y de diversos repositorios institucionales universitarios que sirvieron para consultar, estudiar y contrastar los hallazgos encontrados.

6.2.3.2 Sprint.

Este paso supuso la obtención y el almacenamiento de todo sustento bibliográfico y documental adicional al existente en el marco teórico que sirva de soporte para responder las preguntas definidas en los medios planteados.

6.2.3.3 Sprint Review.

Se alcanzaron los sustentos necesarios en las fuentes determinadas.

6.2.4 Iteración 4: análisis cualitativo

6.2.4.1 Planeación del Sprint.

Para la presente iteración, se propuso realizar un análisis cualitativo de todo el sustento bibliográfico y documental obtenido, a la luz de las preguntas de estudio, las de segundo nivel y los constructos teóricos definidos.

6.2.4.2 Sprint.

Se realiza el análisis cualitativo mencionado.

6.2.4.3 Sprint Review.

Como resultado, se obtuvo un análisis cualitativo que ayuda a contestar las preguntas realizadas, así como aceptar o refutar los constructos teóricos definidos.

6.2.5 Iteración 5: creación de la propuesta final

6.2.5.1 Planeación del Sprint.

Realizado el análisis cualitativo y construido de manera incremental acorde con el proceso definido inicialmente, en el presente *Sprint*, se pretendió desarrollar la propuesta final de trabajo.

6.2.5.2 Sprint.

Se ejecuta la propuesta tecnológica final.

6.2.5.3 Sprint Review.

El resultado es la propuesta tecnológica final, la cual se muestra en detalle en el séptimo capítulo, “Propuesta”.

7 Propuesta

Acorde con lo expresado en el diseño de la metodología, este capítulo muestra la ejecución de las investigaciones y el análisis de diversas fuentes, así como sus resultados, las cuales permiten sustentar y hallar respuesta a las preguntas de estudio trazadas. Por otra parte, se presenta la propuesta resultante que soluciona las problemáticas expuestas y su contribución al campo de acción. Esta sección se conforma por los siguientes componentes:

- Análisis del caso de estudio.
- Propuesta resultante.

7.1 Análisis del caso de estudio

Durante el diseño de la metodología del presente trabajo, se definió como pregunta de estudio: ¿cómo es posible complementar *Scrum*, de manera que permita considerar al ser humano como el centro de la toma de decisiones, así como para desarrollar las soluciones exactas que este requiere? De esta, se desglosaron tres constructos teóricos:

1. *Scrum* como complemento y no como la herramienta definitiva de gestión ágil de proyectos de desarrollo de software.
2. El ser humano, sus necesidades y experiencias como factores clave del entendimiento de sus problemáticas.
3. Uso del método científico para entender y validar si se está construyendo el producto o servicio que el cliente realmente necesita.

A partir de estos, fue posible definir las siguientes tres proposiciones de estudio. Con base en el primer constructo teórico, se planteó la siguiente proposición de estudio:

P1. *Scrum* como herramienta complementaria que permita gestionar proyectos de desarrollo de software de manera ágil y eficiente.

Con el segundo constructo teórico, se determinó la siguiente proposición de estudio:

P2. Lograr el entendimiento de las necesidades y experiencias del ser humano ayudan a entender mejor sus problemáticas.

De acuerdo con el tercer constructo teórico, se formuló la siguiente proposición de estudio:

P3. El uso del método científico ayuda a validar y entender si se está trabajando en la solución ideal para las problemáticas encontradas.

Según estas proposición de estudio, se definieron las preguntas de estudio de segundo nivel. Ahora, se procede a realizar el análisis de cada uno de los constructos teóricos, sus proposiciones y preguntas de segundo nivel, de manera que se obtengan los insumos necesarios para la preparación de la propuesta de trabajo.

7.1.1 Constructo teórico no. 1: Scrum como complemento y no como la herramienta definitiva de gestión ágil de proyectos de desarrollo de software

El presente constructo teórico deriva de la siguiente proposición de estudio: *Scrum* como herramienta complementaria que permita gestionar proyectos de desarrollo de software de manera ágil y eficiente. De este, se desprenden las siguientes preguntas de segundo nivel:

1. ¿Cuáles son las características más notables de *Scrum* que impulsan su adopción y uso?
2. ¿Cuáles son las áreas de oportunidad con las que cuenta *Scrum*?

Con respecto a la primera pregunta, se busca entender qué motiva el uso de *Scrum* así como su adopción, de manera que permita comprender cuáles aspectos le convierten en un marco de gestión de proyectos altamente utilizado y adoptado por sus bondades y fortalezas. Al respecto, los mismos creadores de *Scrum* lo definieron como un marco liviano para la creación de valor a través de soluciones para problemas complejos, mediante la alta adaptabilidad. También, mencionaron con ahínco la simplicidad de este marco, lo cual beneficia el cumplimiento de objetivos y la creación de valor sin la necesidad de dictar reglas estrictas ni detalladas (Schwaber y Sutherland, 2020).

Como método ágil, *Scrum* cuenta con la ventaja de ser altamente aceptado en los equipos de desarrollo de software modernos (Moniruzzaman y Hossain, 2013), los cuales suelen estar distribuidos alrededor del mundo y son equipos a gran escala. De esta manera, usarlo permite contar con equipos autodirigidos, de diversas áreas y perfiles (Gregory et al., 2020). En términos de desarrollo de software, la flexibilidad y adaptabilidad distinguen a *Scrum*; inicialmente, su filosofía de “bienvenido el cambio” se presenta como un antagónico del típico modelo de desarrollo de proyectos de software de cascada (Moniruzzaman y Hossain, 2013; Signoretti et al., 2020).

En este caso, *Scrum* tiene como fundamento una gran ventaja en la rápida entrega de valor, a partir del desarrollo iterativo e incremental del cual, a su fin, brinda tanto valor como sea posible, incluso incluye cambios sobre la marcha en el alcance y las características del producto a desarrollar (Moniruzzaman y Hossain, 2013). De igual manera, *Scrum* destaca en el desarrollo de soluciones dado su alto índice de trabajo en equipo, de forma colaborativa, con una buena relación costo-beneficio, pues produce soluciones de alta calidad y siempre considera las necesidades mutables de los involucrados. La consideración de los posibles cambios así como de reaccionar a tiempo en materia del desarrollo de la solución es clave para el éxito del proyecto, entendiendo que los cambios en los requerimientos de soluciones de software son dinámicos (Moniruzzaman y Hossain, 2013).

Otra cualidad importante de *Scrum* es su procedimiento iterativo de desarrollo, el cual incorpora en cada iteración procesos de aprendizaje, gestión de riesgos y pruebas con los involucrados, con quienes se tiene extensiva comunicación (de Paula y Araújo, 2016). El desarrollo gracias al uso de *Scrum* demanda menos tiempo y estrés en los procesos de análisis y diseño de las soluciones, esto puede ser una ventaja o no, como se ha comentado; además, su uso posibilita la implementación temprana de la solución que se desarrolle (Kaur y Sengupta, 2011). Así, *Scrum* es considerado como el método ágil *de facto* para el desarrollo de aplicaciones debido a su simplicidad y facilidad de adopción (Vilkki, 2010).

Por último, una de las ventajas más reconocidas de *Scrum* es que permite un rápido desarrollo de soluciones mediante una drástica reducción en los tiempos a partir de una estructura de trabajo simple y flexible, donde se entrega valor pronto, principalmente en situaciones donde hay incertidumbre y es difícil planear a futuro (Dobrigkeit y de Paula, 2017).

Con respecto a la segunda pregunta, se busca entender cuáles son las áreas de oportunidad de *Scrum* que impulsan a buscar su complemento en aras de considerar al ser humano como el centro de la toma de decisiones, así como para asegurar que la solución a entregarse realmente solventará sus necesidades. Los creadores de este marco mencionan que considerarlo como incompleto es premeditado, puesto que se busca de manera exclusiva dotarlo de lo necesario para implementar su teoría sin la necesidad de dictar y mandar instrucciones detalladas. Precisamente, se busca que las personas sean las que complementen su uso, lo que ellos definieron como inteligencia colectiva (Schwaber y Sutherland, 2020).

Aunque se considera a *Scrum* como un método liviano y sencillo, su implementación conlleva ciertos retos que no son sencillos de atacar que en las organizaciones. Cada miembro que busque pertenecer a un equipo de *Scrum* debe primero abatir ciertas barreras. Por sí solas, el establecimiento de las prácticas de esta metodología en una organización es una tarea compleja, pues no solo es hablar del conocimiento del negocio que siempre es necesario para entender lo referente a la visión y misión de la empresa, sino que se debe comprender claramente lo referente al proyecto a desarrollar, el producto y su alcance; además, es necesario conocer el ambiente técnico que rodea al proyecto y al equipo.

En este sentido, un equipo de desarrollo *Scrum* involucra que se integren diversos actores de la organización, lo que no siempre es algo fácil de asimilar, comprender ni implementar en las empresas. Igualmente, para un recién llegado a un equipo *Scrum*, se suma la complejidad de tener que entender la manera en la que el equipo trabaja y la adaptación del pensamiento ágil derivado de su entendimiento de *Scrum* (Gregory et al., 2020).

Otra área de oportunidad encontrada en el uso de *Scrum* es que este método se enfoca primordialmente en el alcance de lo que se ha de desarrollar, sea cambiante o no, más que en las problemáticas a solucionar para el cliente y los usuarios o al entendimiento de sus necesidades (Signoretti et al., 2020). Antes de dar por hecho que se cuenta con los requerimientos plenamente identificados, es clave entender plenamente el problema y todo lo que lo rodea (Kaur y Sengupta, 2011).

Por otro lado, a pesar de que la transparencia sea una de las virtudes que *Scrum* presume (Schwaber y Sutherland, 2020), los equipos de desarrollo que usan dicho método suelen sentirse aislados, dado que solo tienen una serie de requerimientos que cumplen y, al terminarlos, perciben que su trabajo está terminado, sin saber realmente más del problema y si las necesidades del cliente fueron subsanadas (Signoretti et al., 2020).

En adición, incluso cuando con *Scrum* se refuerce la participación del equipo de desarrollo y los involucrados del proyecto, puede haber una cierta percepción de que tal involucramiento no es el suficiente para asegurar que el producto a desarrollarse cumplirá con las necesidades del cliente (Signoretti et al., 2020). Por su parte, se considera que el involucramiento de personas con conocimientos especializados y áreas interdisciplinarias no es el fuerte de *Scrum*, además de que el pensamiento divergente es evitado en aras de enfocarse en desarrollar paquetes de trabajo, sin dejar de lado el riesgo de que este método considera que

los requerimientos iniciales del proyecto ya consideran una visión integral de lo que realmente el cliente necesita (de Paula y Araújo, 2016).

Aunque *Scrum* establece desde sus principios el trabajo colaborativo (Schwaber y Sutherland, 2020), su poca participación en los procesos de levantamiento de requerimientos y validación continua de la solución a desarrollar puede generar severos problemas; entre ellos, que el cliente pueda sentirse poco involucrado en la toma de decisiones y terminar con algo que no necesitaba, incluso, los clientes pueden tornarse hostiles cuando se les trata como simples usuarios y no como alguien vital dentro de las actividades del proyectos (Kaur y Sengupta, 2011).

Al respecto, Vilkki (2010) señaló que no conocer bien el negocio al que se dedica la organización ni el entender a la perfección lo que el cliente necesita realmente lleva a la mayoría de problemas en relación con el desarrollo de aplicaciones en las organizaciones; esto es relevante dado que, por lo general, dichas problemáticas son vistas como anormalidades de investigación y desarrollo o simples problemas de calidad en las aplicaciones. Lo planteado es relevante dado que se aprecia este tipo de problemas en proyectos gestionados con *Scrum* (Dobrigkeit y de Paula, 2017). Como se ha mencionado, *Scrum* se enfoca en la construcción ágil de software, aunque no facilita ni cuenta con mecanismos que posibiliten la atención al diseño de las soluciones (Dobrigkeit y de Paula, 2017) ni a la escalabilidad de estas (Vilkki, 2010).

7.1.2 Constructo teórico no. 2: el ser humano, sus necesidades y experiencias como factores clave del entendimiento de sus problemáticas

El presente constructo teórico deriva de la siguiente proposición de estudio: lograr el entendimiento de las necesidades y experiencias del ser humano ayudan a entender mejor sus problemáticas. De esta, se desprenden las siguientes preguntas de segundo nivel:

1. ¿Cómo conocer mejor las necesidades y experiencias de una persona ayudan a entender mejor sus problemáticas y requerimientos?
2. ¿Qué nuevas herramientas habría que utilizar para cumplir con tal fin?
3. ¿Qué adiciones o cambios requiere *Scrum* en sus procesos para entender las necesidades y experiencias del ser humano como proceso de levantamiento de requerimientos?

Con respecto a la primera pregunta, se busca entender cómo al conocer mejor las necesidades de las personas y sus experiencias se puede tener una mejor comprensión de sus verdaderas problemáticas y requerimientos. Como se expresó en la problemática del presente trabajo, el simple uso de *Scrum* no permite conocer cuáles son realmente las verdaderas necesidades y los requerimientos de los usuarios, debido a que este asume que se cuenta con un análisis previo adecuado, el cual se desglosa en paquetes de trabajo en el llamado *backlog* (de Paula y Araújo, 2016; Lindberg et al., 2010).

El primer paso para construir un software es conocer cuáles son los requerimientos que lo conforman. A veces, los objetivos de un proyecto pueden o no cumplirse debido a que, precisamente, los requisitos no son obtenidos de la manera adecuada o con el detalle necesario para que el desarrollador pueda tener plena noción de que lo que está desarrollando es lo que el negocio o el cliente necesita (Kaur y Sengupta, 2011).

Para efecto de permitir a un equipo *Scrum* comenzar con el proceso de desarrollo, es vital que los requerimientos y las necesidades del cliente estén definidos claramente y se apeguen a lo que se necesita en realidad. Sin embargo, en la experiencia, al final del desarrollo de las soluciones, aún se cuenta con necesidades sin cubrir, en otras palabras, se culmina el proceso con algo que nadie quiere. Esto sucede cuando el enfoque hacia la resolución de un problema es meramente técnico y dentro de la visión del propio equipo de desarrollo (Ximenes et al., 2015).

En este contexto, los desarrolladores y miembros de los equipos de desarrollo suelen fungir como los expertos, quienes acaban tomando decisiones sobre los principios de diseño, los requerimientos y las necesidades, lo cual es un error grave, pues se dejan de lado las necesidades reales de los clientes. Esto resulta en el desarrollo de interfaces técnicamente buenas, pero poco intuitivas o funcionales para los usuarios, lo que repercute en rechazo hacia al producto e, incluso, afectación a la productividad y eficiencia de las personas (Lindberg et al., 2010).

Por su lado, las técnicas convencionales de levantamiento de requerimientos –por ejemplo, las entrevistas– suelen arrojar datos inconclusos e incorrectos acerca de las verdaderas necesidades que los clientes tienen. En 2011, Netflix buscaba lanzar una mejora a su interfaz de usuario, la cual se presentó siendo mucho más densa que las versiones anteriores, lo que generó una cantidad considerable de quejas por parte de los usuarios; a pesar de ello, estos

usaban dicha nueva versión más que nunca. Un desarrollador de productos de EEUU resumió esta experiencia a partir de la siguiente frase: “lo que la gente dice y lo que hace es rara vez lo mismo”.

Es mediante este tipo de experiencias, donde la observación del usuario suele ser más efectiva (Langenfeld, 2019); de hecho, este proceso se basa en buscar un mejor entendimiento de los temas mediante la inmersión total en la vida de los involucrados, poniéndose en los zapatos del cliente y caminar en su mundo. Esta falta de habilidad de obtener la misma perspectiva que el usuario para comprender verdaderamente la manera en la que piensan y lo que necesita es uno de los errores más recurrentes en los procesos de desarrollo de software actuales (Ximenes et al., 2015). Esto último es también conocido como empatía, que a su vez se define como la capacidad de entender los sentimientos y las emociones de los demás, mediante el reconocimiento de la otra persona como similar; concepto que proviene de la traducción del alemán de *einfühlung,* que significa sentirse dentro de algo o alguien (López et al., 2014).

Este proceso empático implica conocer a las personas, en este caso, a los clientes y sus necesidades, puesto que el enfoque no debe estar en el producto. Igualmente, este proceso no solo se limita a conocer al cliente sino además también cómo se siente. En este punto, aparece el término experiencia, que de acuerdo con la Real Academia Española (RAE, 2014a) se define como “el conocimiento de la vida adquirido por las circunstancias o situaciones vividas, y como la circunstancia o acontecimiento vivido por una persona” (párr. 3-4).

Estos sentimientos o experiencias del cliente posibilitan conocer mejor qué es lo que desea y cómo lo desea. Por ejemplo, si se está diseñando un hotel, es preciso entender que la primera impresión del cliente, y por lo tanto, el comienzo de la experiencia se vive desde su reservación, ingreso, recepción, etc. En ocasiones, las encuestas se enfocan en la limpieza de las habitaciones o la amabilidad del personal, pero se pierde de vista que el inicio de la jornada empieza con las maneras del personal de reservaciones, la amabilidad o su falta de por parte del personal de seguridad en las entradas, lo fluido de los procesos de ingreso, entre otros aspectos (Langenfeld, 2019).

Por lo tanto, el diseño de un producto o servicio y la comprensión de las necesidades reales por parte de los clientes se logra teniendo la misma visión de los clientes y apropiándose de sus expectativas. En este sentido, cobra mayor relevancia la expresión “ponerse en los

zapatos del cliente", dado que al vivir lo que el cliente experimenta y siente es la manera en la que se puede tener empatía y un mejor entendimiento de lo que realmente se necesita. Cabe añadir que es relevante ser realista y honesto al tratar de comprender mejor el mundo del cliente, dejando de lado preconcepciones e ideas propias; no se trata de confirmar lo que ya se sabe o se espera, sino tratar de buscar nuevo conocimiento que ayude a asimilar mejor lo que el cliente busca satisfacer (Langenfeld, 2019).

De esta forma, el equipo de trabajo de desarrollo de software no debe tomar decisiones sin validación previa por parte de los clientes y, menos, hacerlo con su apreciación del proyecto y producto (Ximenes et al., 2015). Evaluar el diseño de la solución a desarrollar mientras se involucra al cliente y los actores principales de la ecuación posibilita brindar al equipo de trabajo de retroalimentaciones, características y experiencias útiles que permitan entender la eficiencia y alineación del diseño de la solución con las necesidades del cliente, así como encontrar requerimientos ocultos o que hayan sido complicados de explicar por parte de este (International Organization for Standardization [ISO], 2010).

De acuerdo con el ISO (2010), las soluciones tecnológicas deben ser desarrolladas al considerar siempre factores de ergonomía y centrarse en el ser humano. Esto último se define como poner un enfoque en desarrollo de soluciones tecnológicas más usables aplicando factores humanos y ergonómicos, así como técnicas y conocimiento de usabilidad. El diseño de soluciones centradas en el ser humano debe seguir de las siguientes premisas:

- Diseño basado en un entendimiento explícito de los usuarios, sus tareas y entorno.
- Los usuarios son involucrados durante el proceso de diseño y desarrollo, lo cual forma parte integral de algunas de las características esenciales de *Scrum*.
- El diseño de la solución a desarrollar debe ser guiado y probado mediante evaluaciones centradas en el usuario.
- El proceso debe ser iterativo.
- El diseño considera la experiencia del usuario.
- El equipo de diseño incluye personas con habilidades multidisciplinarias.

Por su parte, el término usabilidad se refiere a que cualquier sistema, producto o servicio utilizado por usuarios específicos puede apoyarlos a alcanzar sus metas con eficiencia, efectividad y satisfacción en un contexto determinado de uso. Las soluciones diseñadas con

métodos centrados en el ser humano cuentan, entra otras. con las siguientes ventajas (ISO, 2010):

- Incremento de productividad de los usuarios y de eficiencia operacional en las empresas.
- Mayor facilidad de uso, reduciendo costos de capacitación y soporte.
- Mejor accesibilidad.
- Mejor experiencia de usuarios.
- Reducción de estrés y mejor confort.

Un punto clave es la contribución significativa del diseño centrado en el ser humano para mejorar la identificación y definición de requerimientos funcionales e incrementar las posibilidades de éxito del proyecto al alcanzar de mejor manera las expectativas de los clientes. Por ende, construir sistemas basados en un entendimiento pobre de las necesidades de los usuarios es una de las principales razones de fallas en los sistemas (ISO, 2010).

Po último, al tener más empatía y acercamiento con el cliente y los involucrados del proyecto, el equipo de desarrollo de software los hace sentir de cierta manera indispensables en los procesos, lo cual ayuda a impulsar su trabajo y colaboración. De esa manera, los clientes e involucrados comienzan a ver al equipo de desarrollo como solucionadores de problemas, además de ganar su confianza cuando ven el interés en proveerles un producto o servicio que supla sus necesidades (Signoretti et al., 2020).

Ahora, con respecto a la segunda pregunta, se busca entender cuáles herramientas posibilitan conocer las necesidades y experiencias de las personas para entender sus problemáticas e instrumentos. Para tal fin, se utiliza el *Design Thinking*, que se define como un estilo de pensamiento o un set de herramientas y técnicas para aplicar la manera de pensar y actuar de los diseñadores, que permite explorar nuevos territorios y definir potenciales soluciones con base en ciertas necesidades de los clientes (Schneider, 2017).

Tal cual se comentó, en ocasiones, los clientes no son capaces de expresar de forma adecuada sus necesidades y requerimientos con respecto a las soluciones que buscan, de modo que sus requerimientos suelen ser distintos a lo que necesitan, lo cual puede influenciar de manera negativa en los resultados y el éxito del proyecto. Ahora, en los proyectos gestionados con *Scrum*, la visión del cliente influencia en gran manera su ejecución; si el cliente no tiene

una idea clara de la dirección a la cual debe apuntar la solución a desarrollar, los desarrolladores tomarán el mismo camino y el resultado final puede ser un producto que no cumpla con las necesidades reales del usuario (Ionel, 2008).

Según el *Human centered design toolkit*[1] de IDEO (2015), *Design Thinking* incorpora a su *mindset* herramientas que permiten desarrollar una profunda empatía con el cliente y que posibilita poner por delante la búsqueda y el entendimiento de sus necesidades antes que las de la organización a la que pertenece (Grossman-Kahn y Rosensweig, 2012). De manera semejante, posibilita conocer mejor a las personas a las que se les presta un servicio, de modo que se transforme la información en ideas factibles, se identifiquen nuevas oportunidades y se ayude a incrementar la rapidez y efectividad de la creación de soluciones (IDEO, 2015).

Mediante su uso, el *Design Thinking* permite trazar la ruta inicial para la creación de productos y servicios centrados en el ser humano, por lo que se lleva al equipo de desarrollo de software a mirar a través de los ojos del cliente y ponerse en sus zapatos para ayudarlos a descubrir sus necesidades latentes y apoyar la crear soluciones deseadas, factibles y viables; esto se logra dado que se busca conocer las necesidades reales del cliente, no solo un requerimiento funcional que desarrollar (Grossman-Kahn y Rosensweig, 2012; Ximenes et al., 2015).

De hecho, dentro de su proceso de trabajo, el *Design Thinking* cuenta con una fase llamada “Empatizar”, la cual implica mirar al cliente como la persona que tiene la última palabra y que, gracias a diferentes técnicas y herramientas, es factible conocerlo de mejor manera para entenderlo y saber qué problemáticas le aquejan, qué tipo de decisiones toma y cuáles cambios en su día a día relativos a la solución requerida pueden satisfacer sus necesidades (Ximenes et al., 2015). En resumen, a partir de sus técnicas y herramientas –como la empatía con el cliente–, el *Design Thinking* mejora los procesos de entendimiento de sus problemáticas y trazar el diseño preliminar de la solución que realmente se necesita con un enfoque centrado en el ser humano (Dobrigkeit y de Paula, 2017).

Con respecto a la tercera pregunta de segundo nivel número, se busca entender qué adiciones o cambios requiere *Scrum* en sus procesos para entender las necesidades y

[1] El *Human centered design toolkit* (IDEO, 2015) es un libro que explica el impacto del diseño centrado en el ser humano para la sociedad.

experiencias del ser humano como proceso de levantamiento de requerimientos. Aun cuando *Scrum* y *Design Thinking* cuentan con algunas similitudes notables, como incluir procesos de aprendizaje y desarrollo iterativos y una comunicación fuerte entre los miembros del equipo, existen diferencias notables; por ejemplo, *Scrum* cuenta con un menor énfasis en la colaboración creativa multidisciplinaria, así como con un marcado énfasis en evadir el pensamiento divergente en aras de enfocarse en lo siguiente a desarrollar, además de que mediante el uso de *Scrum,* se da por hecho que la visión y los requerimientos del proyecto están más que delineados y bien definidos (de Paula y Araújo, 2016).

Respecto al tema, en un principio, diversos autores presentan dos aproximaciones distintas. Algunos de ellos han sugerida usar al *Design Thinking* durante las etapas tempranas del proyecto para efecto de levantamiento y refinamiento de los requisitos (de Paula y Araújo, 2016; Grossman-Kahn y Rosensweig, 2012; Lindberg et al., 2010; Ximenes et al., 2015). Por su parte, otros han propuesto converger ambos pensamientos de manera que se unan en un método único (Dobrigkeit y de Paula, 2017; Häger et al., 2015).

Con base en el análisis de ambas posturas, no se considera necesario modificar los procesos de *Scrum*, sino solo crear un bloque inicial para el levantamiento de requerimientos mediante *Design Thinking*; de manera que al final, los requerimientos sean lo suficientemente claros, definidos y acordados con el cliente, así como validado de forma inicial como la postura definitiva que indique la totalidad de lo que este solicita. Lo anterior en busca de que la implementación y adopción de la presente propuesta sea lo más sencilla posible para los futuros equipos de trabajo. Los requerimientos obtenidos se utilizarán como los insumos iniciales del *product backlog* de *Scrum,* lo anterior con fundamento en la alineación con diversos autores, tal cual se muestra a continuación.

En este sentido, Grossman-Kahn y Rosensweig (2012) argumentaron que el desarrollo de cualquier producto nuevo debe iniciar con el *Design Thinking,* debido a que el equipo de desarrollo de *Scrum* necesitará lo antes posible conocer todas las necesidades del cliente, identificar sus problemas y conocer las posibles propuestas de solución. De esta forma, es vital incluir las prácticas de *Design Thinking* a los procesos de desarrollo, principalmente a las fases de levantamiento de requerimientos; dado que los equipos de desarrollo de software siguen un pensamiento deductivo-racional, típico de las áreas informáticas que buscan soluciones a los problemas que se les asignan mediante reglas lógicas (Ximenes et al., 2015); mientras que el

Design Thinking trata los problemas como situaciones complejas que deben ser tratados de forma abierta en busca de evitar cualquier ambigüedad (Lindberg et al., 2010).

Por lo anterior, Lindberg et al. (2010) han sugerido en su trabajo complementar el uso de *Design Thinking* con *Scrum,* precisamente para perfeccionar los métodos de solución de problemas del equipo de desarrollo de software y su pensamiento lógico-matemático con el pensamiento de diseño relacionado al *Design Thinking*; en aras de conseguir soluciones más innovadoras y acorde con las necesidades de los clientes. También, los autores hicieron énfasis en que esta sugerencia hace especial sentido pues, por lo general, los desarrolladores de software no serán los usuarios finales de la solución a desarrollar; por ende, la fase de empatía del *Design Thinking* cobra un sentido enorme (de Paula y Araújo, 2016).

De igual manera, en su literatura, Ximenes et al. (2015) sugirieron el uso del *Design Thinking* al empezar del proceso de desarrollo de software, para determinar el problema mientras se explora y entiende la naturaleza de la problemática y la realidad de sus usuarios, los cuales suelen ser distintos a los del equipo de desarrollo.

Semejantemente, Lindberg et al. (2010) sugirieron usar el *Design Thinking* con *Scrum* en aras de obtener mejores resultados, a la par que con ello se evita que el equipo de desarrollo entienda los requerimientos de los clientes con base en sus concepciones, antecedentes y prototipo de pensamiento; de modo que haya un acercamiento de las técnicas y herramientas que permitan empatizar con los clientes y entender sus necesidades. De forma general, se considera como una opción acertada el uso del *Design Thinking* en los procesos de levantamiento y análisis de requerimientos antes de iniciar los procedimientos de desarrollo de software ágil (Dobrigkeit y de Paula, 2017).

7.1.3 Constructo teórico no. 3: uso del método científico para entender y validar si se está construyendo el producto o servicio que el cliente realmente necesita

El presente constructo teórico deriva de la siguiente proposición de estudio: el uso del método científico ayuda a validar y entender si se está trabajando en la solución ideal para las problemáticas encontradas. De este, se desprenden las siguientes preguntas de segundo nivel:

- ¿Cómo el uso del método científico permite determinar si la solución que se está desarrollando o por desarrollar es la adecuada para el cliente, sigue vigente y será adoptada con éxito?

- ¿Qué nuevas herramientas habría que utilizar para cumplir con tal fin?
- ¿Qué adiciones o cambios requiere *Scrum* en sus procesos para integrar el uso del método científico en la validación del valora y la vigencia de la solución en desarrollo?

Con respecto a la primera pregunta, se pretende entender de qué manera el uso del método científico puede apoyar a validar que la solución que se está desarrollando aún es la que el cliente necesita para solventar sus necesidades y si la adoptará con éxito.

Para ahondar en el tema, es conveniente considerar el origen y significado de método científico. Según la RAE (2014b), la palabra ciencia se define como el "conjunto de conocimientos obtenidos mediante la observación y el razonamiento, sistemáticamente estructurados y de los que se deducen principios y leyes generales con capacidad predictiva y comprobables experimentalmente" (párr. 1). Mediante la ciencia, se crea una vía para llevar a cabo las acciones a partir de un orden, un proceso para encontrar la verdad y enseñar esto gracias a un método analítico o sintético. Precisamente, esta vía es el método, lo que permite que los alcances de la ciencia, el conocimiento generado y sus alcances sean transmisibles mediante algo conocido, como la metodización; esto es, asegurar la transmisión del conocimiento (Labajo, 2015).

Dado lo anterior, es menester recordar que el método científico se concibe como una metodología de investigación que se emplea primordialmente para la generación de conocimientos en el ámbito de las ciencias, la cual para denominarse así, debe consistir en "la observación sistemática, experimentación, mediciones, análisis, formulaciones y modificación de las hipótesis" (Labajo, 2015, p. 4). Ahora, el método científico cuenta con las siguientes etapas (Asuad y Vázquez, 2014):

1. Planteamiento del problema: se delimita de manera clara y precisa el objeto de investigación.
2. Composición del marco teórico: se seleccionan las teorías, los conocimientos científicos, métodos y procedimientos que serán útiles para explicar de forma objetiva el propósito de investigación.
3. Formulación de hipótesis o explicación a ponerse a prueba: se ejecuta el planteamiento preliminar de las posibles causas o razones de un evento. Esta afirmación debe contrastarse con hechos y fenómenos reales.

4. Validación de la hipótesis: mediante la observación, la experimentación, la documentación, las encuestas y los análisis sistemáticos se permite demostrar si la hipótesis dada es válida o no.
5. Conclusiones y resultados: se contrastan los datos versus la hipótesis seleccionada, estableciendo juicios sobre la falsedad o veracidad de ella.

De acuerdo con la definición previa que habla del método científico como un elemento clave para la generación del conocimiento, es preciso explotarlo para asegurar que la solución a desarrollarse o en desarrollo sea y siga siendo la que solvente las necesidades que el cliente expresa y que será bien recibida por este. Dicho proceso de validación debe ser ágil e iterativo para evitar despilfarros desarrollando algo que nadie quiere o utilizará. Esto último es vital y es parte del fundamento de la metodología *Lean* del *Toyota Production System*, esto es, evitar despilfarros a toda costa (Ries, 2012). En ese tenor, un proceso de levantamiento de requerimientos pobre puede ocasionar que se construyan soluciones que los usuarios no necesitan para solventar sus necesidades (Kaur y Sengupta, 2011).

Como se señaló, *Scrum* plantea la colocación de paquetes de trabajo categorizados con apoyo del cliente en un contenedor denominado *product backlog*, el cual se compone de todo el trabajo a realizarse en el desarrollo o proyecto en cuestión, y de donde se obtiene un listado de tareas prioritarias a llevarse a cabo en el siguiente *Sprint*. Esto equivaldría a la primera etapa del método científico: planteamiento del problema (Ángel, 2018).

Por su parte, la segunda y tercera etapa del método científico, la composición del marco teórico y la formulación de las hipótesis, se componen por los requerimientos del cliente, los cuales ayudarán a contrastarlos contra los resultados del trabajo de desarrollo en un momento dado. Gracias al método científico, se busca que el desarrollo de la solución o los componentes con que se cuentan al momento puedan solventar las problemáticas expresadas en la hipótesis.

La cuarta etapa del método científico la validación de la hipótesis, que se da con la interacción del cliente con el desarrollo de software en el momento, lo que posibilita obtener retroalimentación temprana que ayude a entender si el avance y lo desarrollado concuerda con lo que el cliente desea y genera valor o si es momento de reconsiderar el camino que se sigue para retomar aquello que lleve a obtener lo que el cliente necesita. Lo anterior conduce a la última etapa del método científico: las conclusiones y los resultados (Ángel, 2018; Ries, 2012).

Lo expuesto se basa en las mejores prácticas y técnicas del *Toyota Production System,* entre ellas *Just in time*, que indica que las organizaciones ligeras o *Lean* deben ser ágiles en todos sus procesos, con una orientación hacia el aprendizaje y la cultura de la innovación conforme se desarrollan, a partir de la experimentación de los productos, evitando grandes inversiones primordialmente en planificación y diseño (Ries, 2012).

Lo planteado ejemplifica el uso del método científico para experimentar de la mano del cliente cada avance en el desarrollo del producto, en aras de validar si lo que se crea y en lo que se está trabajando es lo que el cliente necesita para solventar sus necesidades y evitar todo despilfarro en recursos, en el desarrollo de algo que nadie usará y que no tendrá mercado. En este sentido, gastar demasiado genera despilfarros de recursos que podrían usarse para otras actividades de valor; así, gastar poco podría llevar a una entrega de valor tardía (Ries, 2012).

Con respecto a la segunda pregunta de segundo nivel que cuestiona qué herramienta se puede utilizar para integrar el uso del método científico, para la validación de las soluciones en desarrollo y que siga dando valor al cliente, se puede mencionar lo siguiente con sustento literario. Para empezar, la herramienta elegida es *Lean Startup*, la cual abraza el uso del método científico para explorar, además de experimentar de la mano del cliente las soluciones que se tienen al momento para comprender conjuntamente si el camino que el equipo de desarrollo lleva es el adecuado.

En este orden de ideas, el pensamiento *Lean* o ligero es una filosofía de gestión que adopta el pensamiento científico para explorar qué tan ciertas son las percepciones y suposiciones del desarrollo en cuestión. Gracias al *Lean Startup* se puede practicar de forma deliberada probar las hipótesis con que se cuenta mediante el uso práctico, observando qué es lo que sucede y con unos ajustes basado en lo concluido (Schneider, 2017). La idea es que las organizaciones puedan aprender haciendo y decidir qué hacer en el siguiente paso; en el caso de hablar de *Scrum*, en el siguiente *Sprint*.

De esta forma, el *Lean Startup* permite experimentar continuamente para aprender a desarrollar el camino que posibilite descubrir las respuestas correctas. En otras palabras, ayuda a identificar cuáles son los aspectos que realmente se deben construir, además de realizar las mejoras necesarias para el sistema de trabajo, de modo que genere valor sin desperdicios. Ahora, el término valor se relaciona con medio donde este se espera que sea producido, en este caso, software (Schneider, 2017).

Con la creación de aprendizaje validado, *Lean Startup* permite la experimentación que crea saberes de la mano con el cliente, la construcción y el desarrollo de planes escalables de pasos que generan valor y su uso impulsa la mejora en productividad, así como en el tiempo de desarrollo y entrega al mercado de la solución, en la calidad y aceptación por parte del cliente (Dobrigkeit y de Paula, 2017). El indicador del progreso que usa *Lean Startup* es el conocimiento validado. Este también está inspirado en el desarrollo de modelos de negocios basados en las necesidades reales de los clientes y la aplicación con ciertas modificaciones del *Lean Manufacturing* (Ries, 2012).

Con respecto a la tercera pregunta de segundo, se desea entender qué se debe modificar de las prácticas comunes de *Scrum* para efectos de integrar las prácticas de *Lean Startup*. Si bien es cierto que *esta* herramienta comparte con *Scrum* la alineación a la filosofía *Lean* derivada del *Toyota Production System*, así como su enfoque en el empirismo que dicta que el conocimiento proviene de la experiencia y que sus decisiones se basan en la observación (Schwaber y Sutherland, 2020), hay algunos aspectos donde *Lean Startup* puede complementar las áreas de oportunidad de *Scrum* y que se aluden a continuación.

En su trabajo, Stray et al. (2020) sugirieron el uso continuo de *Lean Startup* dentro de los procesos de desarrollo, mediante el ciclo de construir, medir y aprender, los cuales buscan de manera continua desarrollar la solución adecuada. Así, cambian la manera de trabajar, en donde solo se desarrollan paquetes de trabajo, sin saber realmente si satisfarán las necesidades del cliente de una manera más activa, donde conocen a fondo la problemática y colaboran a llegar a la solución.

El uso de *Lean Startup* es considerado sumamente útil durante el proceso de desarrollo de software, lo que permite llevar a cabo validaciones o rechazos constantes de los avances, hipótesis, en busca no solo de lograr el objetivo planeado, sino ayudar a encontrar posibles soluciones. Según el proyecto y la organización, se puede repetir de forma cíclica y periódica el ciclo construir, medir y aprender, permitiendo así aprender más del producto y, a la vez, madurar las ideas y los conceptos que se tienen de él (Ximenes et al., 2015).

En ese sentido, la inclusión de *Lean Startup* en el proceso de desarrollo con *Scrum* permite que las hipótesis que se tienen acerca de los requerimientos puedan ser validadas, de forma que durante el avance en el desarrollo de la solución, de modo recurrente, se sigan realizando las validaciones para que se hagan ajustes en caso de ser necesarios en tiempo y

forma, y se evite desperdiciar tiempo con el desarrollo de algo que no se usará (Ximenes et al., 2015).

Una de las prácticas más exitosas que el uso de *Lean Startup* puede adicionar a los procesos de desarrollo en *Scrum* está basado en los principios *Lean* es amplificar el conocimiento. Básicamente, este dota al equipo de desarrollo de retroalimentación constante para la determinación de si el proyecto está en el lugar correcto o no. Además, su uso e integración se considera sencillo, dado que no implica cambiar procesos de *Scrum*, sino adicionar ideas y mejores prácticas; lo cual permite entregar mejores tiempos de desarrollo así como productividad y satisfacción del cliente, y reducir costos relacionados (Sanhueza, 2017).

Al respecto, Sanhueza (2017) ha expuesto que *Lean Startup* y *Scrum* se complementan de manera exitosa, pues posibilite que *Scrum* proporcione las herramientas y técnicas ágiles de desarrollo de software y *Lean* guíe que mediante su filosofía los cambios futuros que no son posibles de entender ni focalizar con *Scrum*. El uso de *Lean Startup* en adición a las técnicas y herramientas de *Scrum* puede colaborar en desarrollar mejores estrategias y planes de crecimiento y escalabilidad de las soluciones, así como reducir las probabilidades de crear soluciones que nadie usará, retrabajos y porcentajes pobres de uso y adopción (Dobrigkeit y de Paula, 2017).

De este modo, la escalabilidad representa uno de los aspectos arquitectónicos relacionados con los sistemas más importantes para cierto tipo de organizaciones y se debe trabajar en ello de manera intensiva y pronta. *Lean Startup* posibilita encontrar cierto balance entre flexibilidad y repetitividad en la gestión del conocimiento y los procesos de las organizaciones, en aras de planear mejor la escalabilidad de sus soluciones (Dobrigkeit y de Paula, 2017).

Por último, la adición de los procesos de experimentación que dota el uso de *Lean Startup* provee al equipo de desarrollo de la posibilidad de agregar valor a su trabajo, gracias a la exploración de los problemas del cliente y su posterior impacto en la solución a desarrollar y entregar al cliente (Signoretti et al., 2020).

7.2 Propuesta resultante

Una vez que las preguntas de segundo nivel fueron contestadas y dieron las respuestas necesarias, se procede a diseñar la propuesta de solución a las problemáticas inicialmente

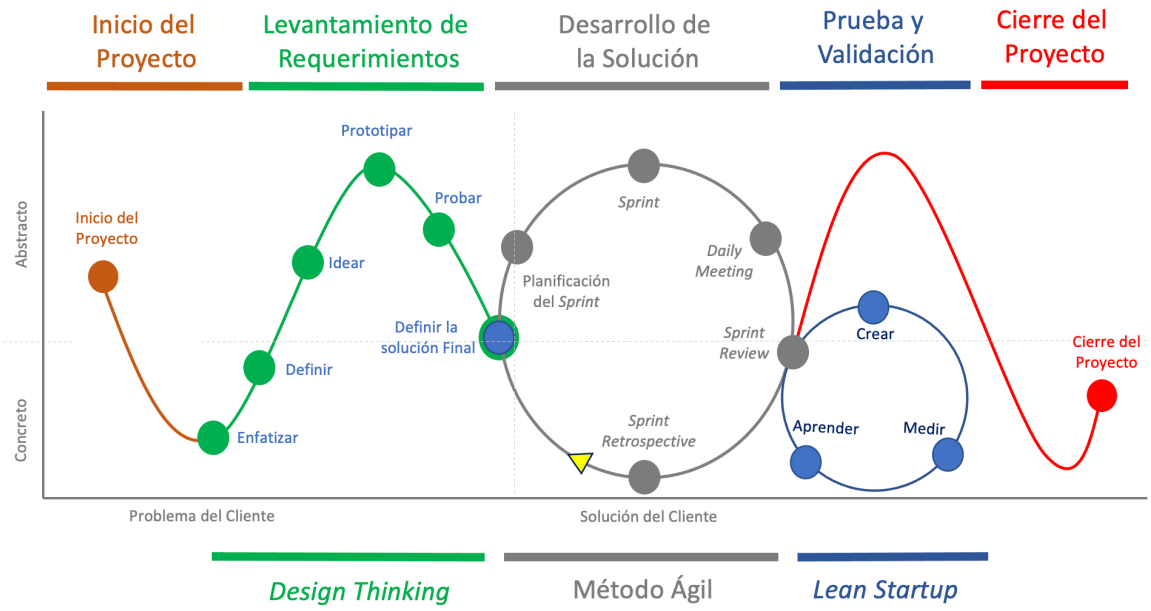
expuestas, mediante los hallazgos localizados y sustentados en el marco teórico y la literatura adicional obtenida y analizada. Así, la base de la propuesta se basa en el uso de *Scrum* como método de gestión ágil de proyectos de desarrollo de software, pero con el uso de un par de marcos que complementa las áreas de oportunidad identificadas y estudiadas, las cuales se mostraron y comprobaron en las preguntas de segundo nivel y sus respuestas. Estos marcos de trabajo son el *Design Thinking* y *Lean Startup* y, serán utilizados en conjunto, de manera que las problemáticas expuestas puedan ser resueltas. El marco resultante propuesto considera las siguientes etapas para la gestión ágil de proyectos de desarrollo de software:

1. Inicio del proyecto.
2. Levantamiento de requerimientos.
3. Desarrollo de la solución.
4. Prueba y validación de la solución.
5. Cierre del proyecto

Tal marco se ejemplifica con el modelo resultante en la Figura 3:

Figura 3

Modelo de la propuesta resultante. Fuente: diseño propio.

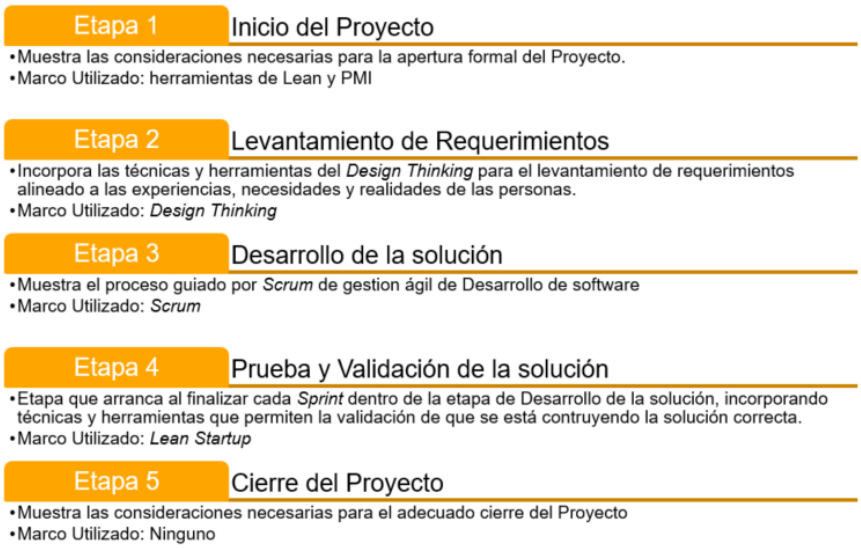


Estas etapas fueron definidas buscando respetar lo mayormente posible la estructura de trabajo de *Scrum*, pero con un énfasis en las adiciones de los dos nuevos marcos con el fin de complementarlo. En la Figura 4, se muestra una breve descripción en cada etapa y la mención

acerca de qué marco es utilizado, con el objetivo de que sea más sencillo entender de forma general la estructura de la propuesta tecnológica.

Figura 4

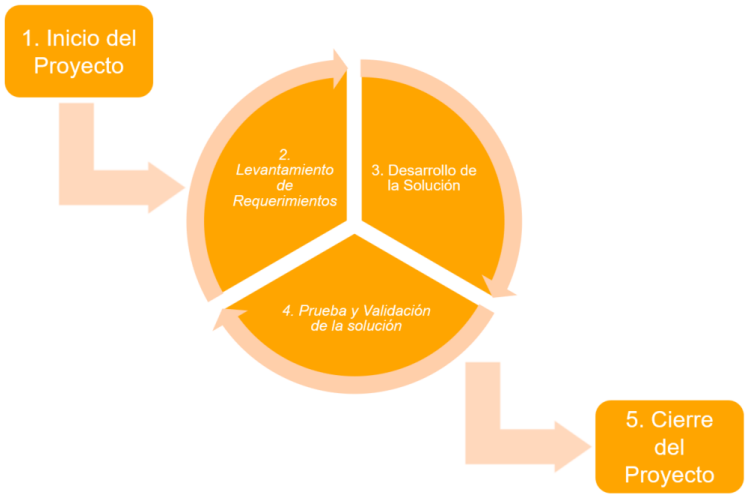
Etapas del proyecto, descripciones y marcos utilizados. Fuente: Diseño propio.



A continuación, se indica cómo la propuesta considera todas las etapas, así como la interacción de cada marco en ella, incluidas sus técnicas y herramientas. Para una ejemplificación más sencilla del flujo de trabajo que se describe enseguida, la Figura 5 plantea de forma simple cómo se lleva a cabo la interacción entre etapas:

Figura 5

Etapas del proyecto y sus interacciones. Fuente: Diseño propio.



Por otro lado, durante cada fase explicada a continuación, se muestra cómo esta se integra a nivel del proceso.

7.2.1 Inicio del proyecto

El inicio del proyecto es la formalización de una necesidad expresa, la cual se busca solventar mediante la solución que el equipo de desarrollo elaborará. El objetivo primordial de esta etapa es tener una definición inicial de la necesidad del usuario, así como del equipo de trabajo. En concordancia con la filosofía de *Scrum* (Schwaber y Sutherland, 2017), no se obliga el uso de algún formato en particular para emplear en esta etapa; lo importante es que se sigan los lineamientos organizacionales existentes para que no se tenga que oficializar el uso interno de nueva documentación, formatos ni procesos para que el requerimiento se haga oficial.

Para este caso y a manera de ejemplo, es común que se utilice la *Acta constitutiva del proyecto* del PMI (2017a) o en el contexto de la industria de la manufactura, el documento denominado A3, el cual es extensivamente utilizado para la definición de problemáticas, arranque de proyectos para solventarlas Y su seguimiento. Acorde con la metodología *Toyota Production System*, existen tres tipos distintos de formato A3 (Rubrich, s.f.):

1. Propuestas.
2. Estatus.
3. Solución de problemas.

Entre otros, el contenido de dicho documento abarca:

- Identificación del problema o necesidad.
- Entendimiento de la situación actual.
- Desarrollar la declaración de alcance u objetivo a perseguir.
- Ejecutar el análisis de causa raíz.
- Lluvia de idea o determinar contramedidas.
- Crear un plan de implementación de contramedidas.
- Validar resultados o confirmación del efecto.
- Actualizar el trabajo estándar.

Con dicho documento, se busca seguir el ciclo de Deming "*Plan-do-check-act*" (Rubrich, s.f.):

- Pasos 1 al 5: corresponde a la etapa de *plan*.
- Paso 6: corresponde a la etapa de *do*.
- Paso 7: corresponde a c*heck*.
- Paso 8: corresponde a *act*.

Inicialmente, el documento es creado por la persona que expresa la necesidad o que es responsable del área que la expresa, aunque dado que se considera como un documento vivo, puede ser modificado sobre la marcha por el autor con apoyo de diversos actores. Por otro lado y de manera ilustrativa, siguiendo la línea del PMI (2017a), se puede utilizar el *Acta constitutiva del proyecto*, la cual busca describir las necesidades, los objetivos y los entregables de este. Además, contiene la lista de suposiciones, restricciones e hitos del proyecto.

Al final, el objetivo es tener un documento formal que contenga la información esencial para arrancar un proyecto; de esta manera, se puede utilizar cualquier tipo de formato de la organización donde se desee implementar el presente método. El siguiente paso es la colección de requerimientos, donde ya se utilizan las técnicas y herramientas del *Design Thinking*, fue el método resultante de la investigación, que arrojó ser la solución para entender las necesidades y experiencias de las personas, de forma que se definieran claramente los requerimientos y las problemáticas que buscan abatir, de la mano del cliente.

7.2.2 Levantamiento de requerimientos

El siguiente paso es entender de forma clara, precisa y completa cuáles son las necesidades de los clientes, sus motivaciones, experiencias y su situación actual. De esta forma, a partir de los lineamientos del *Design Thinking* –tal cual se observó en las respuestas a las preguntas de segundo nivel– y con base en el entendimiento integral de sus necesidades, se podrá definir la serie de requerimientos que el equipo de desarrollo tomará como su *backlog*; los cuales se abarcan a lo largo de la vida del proyecto para, en conjunto, ayudar a solventar las problemáticas encontradas.

Un punto vital a recalcar es que el equipo de desarrollo formará parte del proceso integral de levantamiento de requerimientos. Esto es clave para evitar los vicios y errores comentados, como asumir que la necesidad del cliente es la necesidad propia o que el equipo de desarrollo tiene una visión más completa de lo que el mismo cliente ha expresado (Lindberg

et al., 2010). En ese sentido, se seguirán los siguientes pasos, que se explican a detalle enseguida:

1. Empatizar.
2. Definir el problema.
3. Idear.
4. Prototipar.
5. Probar.
6. Definición de solución final.

7.2.2.1 Empatizar.

El objetivo de este paso es evitar a toda costa el entendimiento de las necesidades del cliente con base en las propias percepciones del equipo de desarrollo, poniéndose en sus zapatos y transitando su mundo (Langenfeld, 2019); esto se logra gracias al involucramiento total con el cliente, lo que permite dotarle de la última palabra en el desarrollo de la visión del producto a ejecutar (Ximenes et al., 2015).

La cantidad de herramientas que se pueden utilizar para tal propósito es inmensa, como se aprecia en el Anexo 1, donde se listan algunas de las técnicas y herramientas de generación de empatía más conocidas y empleadas por el *Design Thinking*. Igualmente, es relevante conocer al cliente, sus problemas y cómo se siente al respecto. Así, gracias a la empatía podemos se pueden dejar de lado las preconcepciones propias acerca de un tema o situación para verlo y vivirlo desde otras perspectivas (Langenfeld, 2019).

Para entender la problemática que detona la necesidad del cliente, hay que realizar una examinación cuidadosa a partir de técnicas y herramientas, más allá de aquellas que hayan causado dicha problemática. Por ende, de forma inicial, se debe entender cuál es el origen del problema y cómo se vincula con el cliente. Entonces, es necesario recopilar todo lo relevante acerca del problema mediante entrevistas, trabajo de campo, talleres, mapas mentales, etc. (Anexo 1). Al final, es vital apreciar las problemáticas de los clientes a través de los ojos del equipo de desarrollo, para que, en ese punto, puedan apropiárselas.

Una técnica que ayuda a entender la realidad de los clientes es la observación, la cual puede realizarse con excursiones. Esto implica vivir en carne propia los procesos que experimentan los clientes y documentar las vivencias obtenidas; ya sea desde un gerente

viviendo la experiencia de un vendedor o un director trabajando en el piso de producción, hasta un desarrollador de software usando las aplicaciones que los clientes piden que sean mejoradas. Este paso es crucial, puesto que no solo permite dejar de lado las concepciones e ideas propias, sino que permite entender con mayor precisión lo que realmente necesita el cliente.

Además de analizar los procesos, hábitos de consumo y uso, es preciso observar y entender el mundo del cliente. De esta forma, hay que comprender sus preferencias, motivaciones y demás. Al respecto, Paul Boag definió las siguientes preguntas como la base para comprender al cliente, las cuales se sugieren como ejercicio, sean documentadas solo con imágenes y dibujos para obtener representaciones visuales de mayor impacto y relevancia (Langenfeld, 2019):

- ¿Qué es lo que el cliente desea lograr en este punto?
- ¿Qué es lo que el cliente quiere saber en este punto?
- ¿Cuáles son los puntos de contacto en este punto?
- ¿Cuál es el sentir del cliente en este punto?
- En este punto, ¿en qué momento se pierde al cliente?
- En este punto, ¿quién o qué influye en el cliente?

Luego de mencionar lo anterior, es necesario llevar registro de lo que se haya obtenido y aún se pueda obtener; suele ser funcional registrar la información con medios escritos, grabaciones de video, notas a mano, fotografías y videos. De esta manera, es vital conseguir tanta información como sea posible en esta etapa, todo de la manera más realista y honesta posible. Aunque se pueda realizar de forma individual, un análisis lo más completo y honesto, siempre será una visión individualista. Por ello, el registrar y documentar todos los hallazgos es esencial.

7.2.2.2 Definir el problema.

Quizá si se ve una fotografía o video entre tres personas, cada uno tendrá una visión distintas con elementos que no todos vieron en común, lo cual enriquece el análisis. En este sentido, la totalidad del material obtenido deberá ser analizado por todo el equipo de trabajo. Finalmente, las observaciones deben ser categorizadas de manera que todo lo documentado ayude al equipo de trabajo a preguntarse por qué. Con ello, se pretende comprender las motivaciones de los

clientes y sus necesidades. En su literatura, Langenfeld (2019) indicó que las palabras más comunes para buscar en la documentación generada son:

- Conducta.
- Deseos.
- Sueños.
- Realidad.
- Necesidades.
- Encargados.
- Persona a contactar.

Esto para no buscar una única necesidad, sino encontrar más de alguna, la cual explícitamente no haya sido expresada por el cliente. Por su parte, el objetivo del proceso de empatía es definir el punto de vista del cliente, el cual se obtiene a partir de los siguientes pasos (Langenfeld, 2019):

- Los resultados de la investigación deberán ser compartidos con todo el equipo.
- Las ideas se agrupan por temas, en busca de identificar las conexiones.
- Las ideas se deberán organizar de forma visual.
- Se resumen y ligan a un actor en particular que tenga relación con el cliente.

Finalmente, lo anterior permite que se cuente con una descripción del problema específico que fue identificado y que contiene las ideas localizadas sobre las necesidades del cliente. Este descubrimiento de las problemáticas del cliente posibilita que se delimiten de manera inicial ciertas posibles soluciones relacionadas. Así, Langenfeld (2019) sugirió que después de recopilar y analizar la información, esta sea impactada en tarjetas que contengan lo siguiente:

- ¿Qué satisface lo observado?
- ¿Por qué satisface?
- ¿Qué es lo que no le gusta al observador?
- ¿Por qué no le gusta?

Una vez detectadas las necesidades del cliente, se lleva a cabo un proceso de validación para asegurar que el análisis haya sido el adecuado y no hayan quedado necesidades ocultas. Entre otros métodos, uno de los más usados es el llamado método SPICE (*Spice, Physical,*

Identity, Communication y *Emotional*); las necesidades de los clientes suelen estar categorizadas en tales términos, por ejemplo (Langenfeld, 2019):

- Necesidades sociales: me importa lo que los demás piensen de mí.
- Emocionales: me siento bien cuando consumo café.
- Físicas: el ejercicio me ayuda a lidiar mejor con el estrés.

Al final de cuentas, tanto las necesidades como las expectativas de los clientes pueden encapsularse en la formación de personajes ficticios gracias al *storytelling*. Por ejemplo:

> A Juan le gusta tomar café, porque opina que es saludable, le mantiene despierto y le gusta su sabor. Quiere mantenerse despierto y activo. También, le resulta adecuado tomar no solo café, sino ocasionalmente tomar tés de hojas orgánicas que va a comprar directamente al huerto. Frecuentemente, sus vacaciones son en las playas de Cancún, donde también practica el esnórquel. Juan labora en el departamento de control de producción.

7.2.2.3 Idear.

Este punto no se refiere a la creación de ese algo que subsanará la necesidad del cliente, sino de descubrir cuáles serán tales soluciones. En este orden de ideas, se busca que, a partir de ideas, entienda cuál es la solución que solventará las necesidades de los clientes. Se sugiere que se sigan utilizando los personajes recién creados para efectos de idear en la búsqueda de soluciones para cada uno de ellos; un ejemplo sería documentando algo similar a lo siguiente:

- ¿Cómo se puede satisfacer el deseo de Juan de café y tés orgánicos?
- ¿Cómo se puede hacer que Juan pruebe tés comerciales y chocolate?

Se sugiere definir preguntas concretas que posibiliten obtener respuestas concretas. El proceso de obtención de ideas permite el uso de diversas técnicas y herramientas, como la lluvia de ideas y sus técnicas derivadas, así como la inspiración analógica (Langenfeld, 2019). Tales concepciones obtenidas también deben ser clasificadas para que hagan más sencilla su visibilidad y evaluación; en este punto, no hay ninguna idea mala, de igual manera, ninguna de ellas será desechada sino puesta a un lado, dado que más adelante puede ser útil de nuevo. Por último, se votarán las ideas que compondrán el prototipo a desarrollar en el siguiente paso, el cual se busca que sea aquel que solvente las necesidades de los clientes.

7.2.2.4 Prototipar.

Se busca que la idea no quede solo en papel, en algún pizarrón o un documento electrónico; sino que dicha idea se pueda visibilizar en la medida de lo posible. La intención no es crear el entregable final, sino un cierto grado de realización de este e, incluso, en ciertos grados de abstracción, ya sea mediante la creación de conceptos o modelos. De esta forma, un concepto debe explicar con claridad y sin ambigüedades cómo se deben resolver las necesidades del cliente y cuáles ventajas se obtendrán. Por otro lado, el prototipo o modelo, busca que el producto a desarrollar sea visible tanto como sea posible, sin la necesidad de terminarlo desde el principio; básicamente, se debe seguir el concepto de realizarlo en el menor tiempo para obtener retroalimentación de los clientes lo más rápido posible (Langenfeld, 2019).

7.2.2.5 Probar.

Una vez realizado el primer prototipo, es tiempo de probarlo de inmediato. La intención no es probar si este funciona o no, sino cuál es la percepción del cliente para obtener su retroalimentación lo antes posible. Es vital sumar a las personas que están ligadas de alguna manera a la problemática señalada para que compongan este grupo inicial de pruebas, conformado con personas con perfiles diversos en la medida de lo posible; debido a que se buscan opiniones diferentes que ayuden a formar una visión más completa del prototipo.

La prueba debe ser lo más interactiva posible y, durante las pruebas, se debe prestar especial atención a todos los comentarios del grupo de pruebas, así como tomar notas de ello. El *Design Thinking* es cíclico, por lo cual, después de las pruebas se pueden repetir los procesos mencionados, pero ahora en vez de buscar de nuevo soluciones, el enfoque puede darse en mejorar el prototipo con base en la retroalimentación obtenida.

La empatía sigue presente durante las pruebas, pues se hace énfasis en que las pruebas no son para comprobar si el modelo o prototipo funciona, sino para entender cómo se siente el cliente, cómo piensa, entiende y usa el producto. Ahora, es preciso entender qué les gusta o no del sí mismos, por qué si y por qué no. Lo expuesto aplica para prototipos y modelos intangibles, tales como aplicaciones de software. Finalmente, en cuanto a las consideraciones de pruebas, puede que estas sean distintas, pero el objetivo no se ha modificado. Posteriormente, si entre los comentarios y la retroalimentación se reportan errores o

características no deseadas, el equipo de trabajo deberá resolverlos para volver a crear una versión que probar, siempre considerando los comentarios y sucesos documentados.

7.2.2.6 Definición de solución final.

Una vez se llevaron a cabo las iteraciones donde se descubrieron, entendieron y modelaron soluciones que ayudan a solventar las necesidades de los clientes, es momento de tomar esa documentación y hacerla llegar al equipo de desarrollo de software. Como el equipo formó parte de todo el procedimiento de levantamiento de requerimientos, el conocimiento de las definiciones es plenamente conocido. Igualmente, como también se desarrollaron personajes, sus *storytellings* y fueron votados como parte de aquellos a emplearse para la definición y construcción del prototipo o modelo final, pueden ser empleados en vez de las historias de usuario que *Scrum* suele usar. Asimismo, claramente, es vital plantear cuál es la definición de terminado del proyecto, cuáles entregables deben ser desarrollados y bajo qué condiciones y con cuáles características para que el proyecto pueda ser considerado como terminado.

7.2.3 Desarrollo de la solución

Como se comentó en el marco teórico, el objetivo de *Scrum* es construir excelentes soluciones de software que se adapten correctamente a las necesidades cambiantes. *Scrum* comienza su magia cuando ya existe una necesidad definida a satisfacer, más no ayuda a obtener ni comprender los requerimientos (Schneider, 2017). Para esta etapa del marco propuesto y tal como se comentó, se utiliza *Scrum* dadas sus ventajas competitivas, en donde brillan sus cualidades en este proceso de trabajo. Si bien la presente investigación no considera explicar paso a paso el uso detallado de *Scrum*, pues escapa del alcance de la presente propuesta, sí se considera explicar a nivel conceptual los pasos, la interacción con actores y, con mayor detalle, la incorporación del segundo marco complementario para solventar las problemáticas definidas, que es el *Lean Startup*; el cual entra en acción una vez que, durante el ciclo de trabajo de *Scrum*, se finalizan los *Sprints*, lo cual se aprecia más a detalle a continuación.

El levantamiento de los requerimientos con el *Design Thinking* permite entender de forma clara y concisa las necesidades del cliente, así como transformarlas en requerimientos que se utilizan para el desarrollo de la solución solicitada. *Scrum* los tomará y serán ubicados en el *product backlog*, desde donde con priorización por parte del cliente (*Product Owner)*, el equipo de desarrollo sabrá cuáles paquetes de trabajo integrar cada *Sprint*, pero ahora con la

adición de *lean startup*. Dado lo anterior, tanto *Scrum* como *Lean Startup* se ubican durante la ejecución del proyecto; la idea de integrar aquí a *Lean Startup* es contar con un marco pertinente que ayude a probar los supuestos y refinar la estrategia mediante aprendizaje validado. Este proceso de aprendizaje con la ejecución requiere de un ambiente altamente adaptativo (Schneider, 2017), el cual es provisto por *Scrum* y permite que ambos marcos se complementen exitosamente. La Figura 1 muestra de manera visual el proceso completo de *Scrum*, el cual cuenta con los siguientes actores:

- *Product owner.*
- *Scrum master.*
- *Equipo de desarrollo y.*
- Todos los involucrados (incluido el cliente).

Asimismo, cuenta con los siguientes pasos:

1. Historias de usuario.
2. *Backlog* del producto.
3. Planeación del *Sprint*.
4. *Sprint Backlog*.
5. *Sprint*.
6. *Scrum daily meeting*.
7. Incremento.
8. *Sprint review*.
9. Retrospectiva del *Sprint*.

A continuación, se plantean cada uno de los pasos, elementos clave a considerar por parte del marco propuesto y consideraciones en su uso.

7.2.3.1 Historias de usuario.

Como se mencionó en el apartado de levantamiento de requerimientos, gracias a las técnicas y herramientas del *Design Thinking* se crearon hipótesis con el *storytelling* de los usuarios; los cuales se tradujeron en historias de usuario, que de manera transparente son tomadas por el equipo de desarrollo, el cual formó parte del equipo que estuvo en tal proceso. Dado que *Scrum* es un método con un proceso cíclico, se pueden seguir generando nuevas historias de diversos

conforme sea necesario, por lo cual aquellas creadas durante el proceso de levantamiento de requerimientos no serán las únicas a ser consideradas.

7.2.3.2 *Backlog* del producto.

Es un artefacto de *Scrum* en donde se listan todas las historias de usuario, las cuales componen en su total y en determinado momento el total de necesidades a solventar con el desarrollo a realizarse o que ya se está desarrollando (Schwaber y Sutherland, 2017). Esta lista ordenarse por prioridades, las cuales son brindadas por el cliente, quién tomará el rol del *product owner*, de tal modo que el equipo de desarrollo sepa cuáles son las tareas a las que habrá que dedicarles fuerza de desarrollo primero (Radigan, s.f.). Además, se deberán ordenar las actividades por *epics*, esto es, conjunto de historias de usuario relacionadas e interdependientes. Un ejemplo para ello podría ser tener *epics* para:

1. Gestión de usuarios.
2. Módulo de ofertas y promociones.
3. Módulo de reservaciones.
4. Otros.

En ese sentido, se podrá tener una mejor estructura y claridad del trabajo a realizarse y el progreso en este. Cabe destacar que el *product owner* podrá cambiar las prioridades en el *backlog* de producto cada que sea pertinente, ya sea por alguna necesidad nueva detectada, la retroalimentación del proceso de *Lean Startup* o solo poderse enfocar en actividades que generen mayor valor en el momento. Estos ajustes de prioridad, así como los refinamientos necesarios deberán ser frecuentes, de lo contrario, el equipo podría verse en un proceso de desarrollo de actividades que quizá no sean las de mayor relevancia para satisfacer las necesidades del cliente; en tal caso, se verá durante el final del *Sprint*, cuando se valide el paquete de trabajo desarrollado, el cual afortunadamente se podrá detectar a tiempo mediante el uso de las técnicas y herramientas de *Lean Startup*.

7.2.3.3 Planeación del *Sprint*.

Es un evento donde se proyecta qué actividades contenidas en el *product backlog* deberán ser ejecutadas en el siguiente ciclo de desarrollo (*Sprint*) por parte del equipo de desarrollo. El *product owner* es el responsable de asegurar que dicha selección sea la adecuada y considere

las prioridades que en su momento son esenciales para la solución a desarrollar (Schwaber y Sutherland, 2017). En este evento, es clave definir la carga de trabajo que podrá ser llevada a cabo de manera realista, esto para evitar incumplimientos y presión innecesaria al equipo de trabajo de desarrollo, quienes serán responsables de indicar cuál será su carga por *Sprint*, decisión que no podrá ser manipulada por el *product owner*.

También, el equipo de desarrollo deberá estimar cuánto trabajo le llevará cada actividad seleccionada para confirmar que todo lo elegido podrá terminarse dentro del tiempo determinado para el *Sprint*. En conjunto, todos los involucrados deberán crear un plan de acción para completar la totalidad del trabajo seleccionado, el cual conformará un siguiente incremento al desarrollo. Finalmente, se deberá dejar claro cuál es la definición de "terminado" para cada actividad y para el trabajo que compone al *Sprint*, de manera que el equipo de desarrollo pueda entender claramente cuando puede cerrar dicha actividad y comenzar otra.

7.2.3.4 Sprint Backlog.

Es la lista de elementos del *product Backlog* seleccionados para el siguiente *Sprint*, así como el plan de acción desarrollado en la planeación del *Sprint*. Esta lista deberá ser lo suficientemente clara y visual para que todo el equipo de desarrollo tenga claro lo que debe hacerse (Schwaber y Sutherland, 2017). De igual manera, deberá ser actualizado durante el *Sprint* para que se pueda apreciar el avance y sea sencillo exponerlo, así como evidenciar cualquier relevancia en el *Scrum Daily Meeting*.

7.2.3.5 Sprint.

Es el ciclo durante el cual se deberá llevar a cabo la transformación de lo expuesto en las historias de usuario, en un código que paulatinamente muestra el valor que el usuario necesita para solventar sus necesidades expuestas. Se sugiere que la duración del *Sprint* sea fija, de un mes; a menos que el equipo de trabajo desee hacer excepciones, las cuales serán eventuales para regresar de inmediato a la duración ideal de un mes. Cabe destacar que la duración del *Sprint* considera todo lo necesario para que este sea llevado a cabo, desde la planeación del *Sprint Daily Scrum Meetings, Sprint Review* y la retrospectiva del *Sprint*. Ahora, ciertas reglas deberán seguirse durante el *Sprint*:

- El objetivo del *Sprint* es inamovible e inalterable.

- No se deberá poner en riesgo nunca la calidad.
- Si se desea, se podrán realizar refinamientos al *product backlog*, con el objeto de entender mejor lo que hay que desarrollar.
- Si fuese necesario, el alcance puede ser aclarado y negociado con el *product owner*; en caso de tener más visibilidad de aspectos que no fueron observados durante la planeación y que implican un mayor esfuerzo y complejidad de lo estimado.

Solo el *product owner* podrá cancelar el *Sprint* actual, lo que podrá darse solo si este detecta que el valor buscado en el objetivo detectado se ha vuelto obsoleto o algún evento se ha dado que haya cambiado las reglas. Por citar un ejemplo, basta imaginar que se está desarrollando una solución de automatización para ciertos procesos de un ERP (*Enterprise Resource Planning)*,[2] pero este se encuentre de manera sorpresiva fuera de servicio. Dado lo anterior, con la autorización del *Product Owner*, el equipo podrá seguir trabajando en otras actividades que también son necesarias y que generan valor, pero que no requieran del ERP en ese momento. Para ello, habrá que realizar todo el proceso del *Sprint* desde el inicio.

7.2.3.6 Scrum Daily Meeting.

El objetivo de esta actividad es revisar de manera diaria, como lo indica su nombre, el progreso de las actividades planeadas y que llevarán al cumplimiento del objetivo pactado, así como adaptar lo necesario en el *Sprint Backlog*. Este evento no durará más de quince minutos, se hará siempre a la misma hora y de preferencia en el mismo lugar. Básicamente, cada integrante del equipo indicará lo que hizo el día anterior, en que estará trabajando en el día actual y, en caso de tener algún problema o impedimento, tendrá que mencionarlo para que el *Scrum master* le ayude a solventarlo de manera que no tenga bloqueos para realizar su trabajo.

7.2.3.7 Incremento.

Al finalizar un *Sprint*, el acumulado de las actividades hechas formarán uno o más incrementos, los cuales se suman a los incrementos realizados previamente. Cada incremento puede verse como un escalón hacia el objetivo del proyecto. Cabe recalcar que un incremento se considera como tal una vez que se cumple con la definición de "Terminado", la cual se concibe en el

[2] Es un sistema de gestión con diversos módulos que facilitan la gestión de las organizaciones, integrando el manejo de datos con sus procesos organizacionales (Esteves y Pastor, 2001).

Sprint Planning. Así, los incrementos realizados durante el *Sprint* serán revisados en el *Sprint Review*.

7.2.3.8 *Sprint Review*.

De acuerdo con la teoría de *Scrum*, en este punto es donde se inspecciona el resultado del *Sprint* y se determinan futuras adaptaciones, además de discutir sobre qué hacer a continuación. Sin embargo, aquí es donde la propuesta de trabajo sugiere el uso de *Lean Startup* para hacer una validación integral del trabajo realizado, experimentando y aprendiendo de la experiencia del usuario para efectos de determinar si se persevera o pivotea.

En este sentido, el procedimiento completo del uso del *Lean Startup* se aborda más adelante. La entrada a tal proceso es el trabajo resultante, esto es, los incrementos y la salida de este será el aprendizaje en conjunto que dictamine si se va en el camino correcto o si hay que corregir algo, lo cual todo en su conjunto ayudará para que el *product owner* haga los ajustes necesarios al *product backlog*.

7.2.3.9 Retrospectiva del *Sprint*.

Este evento es utilizado para reflexionar y discutir sobre lo acontecido en el *Sprint*, desde que salió bien, qué problemas surgieron, si se pudieron solventar y cómo, sobre las interacciones con las personas, el uso y la experiencia con las herramientas, entre otros. La idea fundamental es que el equipo de trabajo pueda generar conocimiento y experiencia, que aprendan de lo que salió bien para que se convierta en mejores prácticas, así como entender lo que salió mal para que sean lecciones aprendidas y buscar no caer de nuevo en ello. Al realizar este evento en conjunto, todo el equipo puede aprender y ganar experiencia, lo cual ayudará a los siguientes pasos de trabajo.

Dado el mecanismo de trabajo iterativo de *Scrum*, se seguirán llevando a cabo *Sprints* hasta que todo el listado de actividades contenidas en el *product backlog* y el *product owner* apruebe que lo creado hasta el momento cumple con la definición de “terminado” del proyecto previamente definida en el punto seis del levantamiento de requerimientos mediante el uso de *Design Thinking*.

Para terminar, una vez concluidas todas las actividades de desarrollo y con la aprobación del *product owner*, se puede llevar a cabo una sesión final de prueba de validación

de la solución mediante el uso de *Lean Startup*, del cual se mostrarán detalles a continuación. Del resultado de esta prueba, pueden salir comentarios, requerimientos nuevos, problemas no identificados y errores que podrán generar nuevas historias de usuario a trabajar por el equipo de desarrollo. En este caso, habría que planear y priorizar de nuevo. Si después del proceso de validación no hay problemas ni nuevos requerimientos y se cuenta con la aprobación del *product owner* o el cliente como se conoce fuera de *Scrum*, se podrá pasar a la etapa del cierre del proyecto, de la cual se darán detalles más adelante.

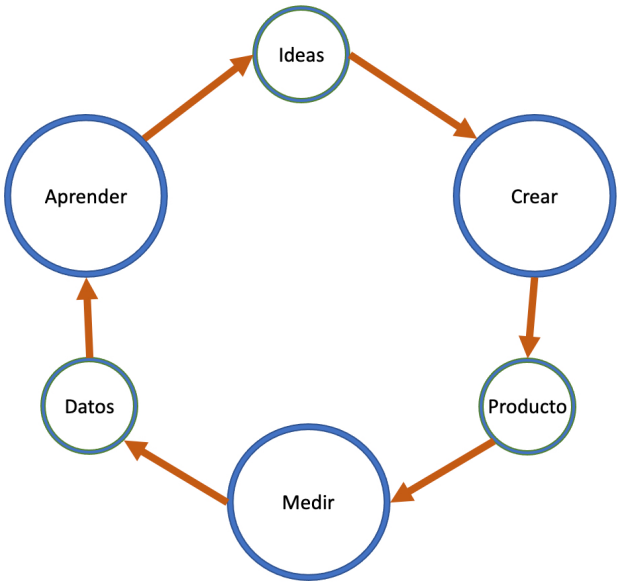
7.2.4 Prueba y validación de la solución

En este punto, se pueden explotar las ventajas competitivas de *Lean Startup*. Para Ries (2012), una *startup* es "una institución humana diseñada para crear nuevos productos y servicios en condiciones de incertidumbre extrema" (p. 39), tal cual se vive en muchos de los desarrollos de software actuales.

Cabe recordar que la actividad principal de cualquier *startup* es transformar ideas en soluciones, medir la respuesta de sus consumidores y aprender cuando es necesario dar un paso atrás; esto es, pivotear o continuar con el camino actual, lo que se considera como perseverar (Ángel, 2018). En este sentido, el *Lean Startup* basa como modelo estructural el uso del ciclo de retroalimentación de crear, medir y aprender. Este ciclo busca en cada una de estas etapas poder llevar a cabo actividades que brinden valor. En la Figura 6, se puede apreciar este ciclo, así sus entradas y salidas principales:

Figura 6

Ciclo: crear, medir y aprender. Fuente: Diseño propio.



A continuación, se lista cómo el marco propuesto sugiere el uso de cada ciclo de *Lean Startup*, sus entradas y salidas, además de su alineación a los otros dos marcos previamente discutidos, esto es, *Scrum* y *Design Thinking*.

7.2.4.1 Crear.

Para el *Lean Startup*, el comienzo de todo su proceso se lleva mediante la identificación de las hipótesis que hay que probar; de este modo, las hipótesis de valor y crecimiento son las que respectivamente se enfocan en asumir que el producto o servicio en cuestión realmente genera valor y que puede crecer de manera sostenible y realista (Ayala, 2019).

Para efectos del presente marco, estas asunciones se crean desde la etapa del levantamiento de requerimientos con *Design Thinking*. Dado lo anterior, el siguiente paso es crear un PMV, esto es, un Producto Mínimo Viable. que posibilite comenzar de inmediato con el ciclo de retroalimentación para poner a prueba tales hipótesis con el aprendizaje. Ahora, es vital crear un mecanismo que posibilite medir su impacto delante de los clientes y clientes potenciales, que en organizaciones establecidas puedan ser potenciales usuarios futuros de la solución a desarrollarse.

Afortunadamente, no hay que comenzar de cero. Dado que esta fase de validación se dispara al término de cada *Sprint* durante el proceso de desarrollo de la solución y de manera

opcional, al final de este, el PMV como tal es el cúmulo de incrementos con que se cuenta al momento. Lo anterior es vital, dado que se respeta uno de los fundamentos *Lean* que es eliminar el desperdicio o la basura, el cual consiste en quitar de lado todo lo que no genera valor, evitando tener que dedicar recursos para la creación del PMV; otro principio que se alcanza es del de fabricar con calidad (Schneider, 2017), dado la regla de *Scrum* de no sacrificarla ni reducirla (Schwaber y Sutherland, 2017).

Lo anterior es clave si se produce un PMV con calidad dudosa, el cliente podrá percibir cierta variabilidad en la calidad del producto o servicio, lo que en el mejor de los casos puede ocasionar retrabajos, pero en el peor puede llevar a la pérdida del cliente; la mala calidad sin embargo, puede ayudarnos a obtener retroalimentación, aunque lo que no se puede tolerar son los defectos pues hacen más difícil la evolución del producto en desarrollo, así como en la capacidad de aprendizaje (Schneider, 2017). En este paso, es necesario documentar las reacciones de los usuarios en la prueba.

7.2.4.2 Medir.

El reto más importante en esta etapa es determinar si los esfuerzos llevados a cabo al momento están bien encaminados o si hay un progreso real; mediante las pruebas realizadas, se puede determinar si el producto o servicio que creado será algo que el cliente todavía quiere y necesita. En este punto, no es relevante si se va a tiempo y dentro de presupuesto. En cambio, es vital ser riguroso con las mediciones que se realicen, además de estar dispuesto a afrontar lo peor con respecto a la apreciación y experiencia que el cliente pueda llegar a tener. Si la retroalimentación es buena, se sabrá que los esfuerzos realizados son los idóneos y que se podrá continuar con el planteamiento actual. Si la retroalimentación no es positiva, se tendrá aprendizaje para encausar los esfuerzos hacia donde el cliente ve el valor y mostrar anotaciones y con sus comentarios se sabrá hacia donde mover el timón en aras de hacer los cambios necesarios, construir una nueva versión del PMV y volver a intentar.

Solo en el caso de que el PMV tenga un resultado desastroso, que el avance actual haya sido rechazado por completo, y que los comentarios y la experiencia del cliente muestre que el trabajo no va por donde debería, el equipo de trabajo deberá considerar volver a la fase inicial de levantamiento de requerimientos para ajustar desde raíz lo que sea necesario corregir. Es mucho más benéfico regresar atrás a tiempo que terminar un producto que nadie usará ni que

generará el valor esperado por más a tiempo y en costes que se culmine, evitando así el despilfarro de recursos que nadie quiere.

La medición suena muy sencilla, vista como una unitaria hacia el incremento en cuestión. Esta es una técnica interesante que puede ser útil cuando los comentarios son muy diversos y en gran cantidad, pues es posible hacer un análisis de cohortes (Ries, 2012). Esta técnica permite que, en vez de analizar acumulados en las cifras, se estudie el comportamiento de los grupos que formaron parte de las pruebas; cada grupo es llamado cohorte. Así se podrá apreciar mejor la tendencia en la aceptación y desaprobación en la solución. Imaginando, por ejemplo, que se está desarrollando una solución que involucra diversos departamentos como clientes, este tipo de análisis permitirá entender si la tasa de aprobación es mejor o peor con un cierto departamento, o si el mismo fenómeno se da a partir de una fecha determinada, ayudando a entender si algún cambio reciente afectó para bien o mal, o si el enfoque está siendo cuidado con algunas áreas y descuidado con otras.

En su literatura, Lindberg et al. (2010) mencionaron que los desarrolladores de software tienden a establecer sus propios juicios y criterios por sobre los de los usuarios, pensando en que ellos tienen siempre una mejor solución a lo que el usuario requiere. A partir de lo anterior, es primordial validar y medir con precisión el resultado de las pruebas y comparar los resultados contra lo planeado.

Otra herramienta útil es el análisis del embudo, la cual es empleada en el ámbito empresarial para analizar el comportamiento de los consumidores, denominados como flujos, los cuales muestran la interacción de los clientes con los productos ofrecidos (Ries, 2012), con lo que se han hallado medidas cuantitativas acerca de cuál nuevo componente o característica tiene mayor o menor aceptación por parte de los clientes. Lo anterior es sumamente importante, pues ayuda a medir la efectividad de los pivotes realizados, lo que muestra qué tan efectivos son contra los experimentos precedentes.

7.2.4.3 Aprender.

El aprendizaje se logra gracias a la pronta retroalimentación de los clientes durante y después el proceso de pruebas del PMV. Es extremadamente importante tener una postura constructiva y humilde al recibir la retroalimentación. Al final de cuentas, los usuarios serán ellos y no el

equipo de desarrollo, por lo cual nadie sabe más de si el camino está siendo el correcto que ellos.

Como desarrolladores de software, se cuenta con la habilidad de mejorar algo, de hacerlo más robusto y lanzar nuevas versiones; empero, es riesgoso hacerlo cuando se trata de mejorar algo que no tiene rumbo o que no le genera valor al cliente. No es lo mismo corregir un pequeño error que tratar de realizar un parche a un barco hundiéndose con cinta y buena voluntad. Además, como se ha mencionado, dedicar recursos a una actividad que no genera valor al cliente es un desperdicio y eso va en contra de los fundamentos del pensamiento *Lean*.

Por otro lado, a pesar de que los resultados no sean los esperados en un inicio, el equipo de trabajo debe mantenerse lo más disciplinado posible. Un equipo disciplinado puede ir a la deriva y sin rumbo, pero podrá dar un golpe de timón y hacer los ajustes a tiempo. También, es importante aprender que cuando las cosas no salen como el equipo de desarrollo espera, no se trata de que no se esté trabajando duro o de manera eficiente. Una gran ventaja de incorporar el uso de *Lean Startup*, incluso desde la creación del primer incremento, es poder validar y aprender muy pronto en el proyecto, buscando con ello no dedicar recursos innecesarios a esfuerzo que no lo merece.

Entonces, ¿es necesario Pivotear o Perseverar? Si se describe que alguna de estas hipótesis no es correcta, será necesario pivotear, de manera inicial para que el cliente determine si el cambio puede ser integrado en el *product backlog* de la etapa de desarrollo de la solución, o incluso si el fallo es tan profundo que se tenga que repetir la sesión de levantamiento de requerimientos para refinar tanto como sea posible el descubrimiento y entendimiento de la nueva necesidad. Aunque suena duro, pivotear no significa algo malo en sí, pues abre nuevas posibilidades a mejorar el producto o servicio que se desarrolla, por lo que se enfrasca en una vorágine, basados en creencias y suposiciones personales. Muchas veces, después del pivote, se podrá comprobar que era necesario, incluso con lamentos de no haberlo realizado previamente.

No es recomendable usar, entonces, mediciones basadas en indicadores que nublan la vista; esto es, de nada sirve tener un porcentaje de avance promisorio cuando lo que se ha hecho al momento no cumple con lo que los clientes esperan. Si la hipótesis inicial es errónea y se niega a aceptarlo, se generarán despilfarros motivando al equipo a continuar con el desarrollo de algo que nadie usará ni aceptará. Por último, es preciso mencionar que realizar pivotes no

es tarea sencilla, pues requiere de mucha valentía y decisión. Los equipos de desarrollo suelen estar acostumbrados a conceptualizar la idea del éxito desde la visión del no equivocarse, del siempre tomar las mejores decisiones. Así, es vital aceptar el error para tomarlo como impulso hacia una nueva idea que ayude a llegar hacia donde los clientes esperan (Ries, 2012).

Como final de esta etapa, a la documentación generada durante la construcción del PMV, del resultado de las pruebas y comentarios de los clientes, se debe documentar cuál es el aprendizaje validado de este ciclo, de manera que el equipo de desarrollo junto con el cliente (*product owner* en lenguaje de *Scrum*) analicen si es prudente perseverar o pivotar y en qué magnitud hacerlo; en definitiva, el *product owner* deberá decidir qué camino seguir.

7.2.5 Cierre del proyecto

Parte vital del marco propuesto es cerrar apropiadamente las actividades y los compromisos obtenidos con el proyecto. El cierre formal de este debe ser estructurado y realizado de manera óptima. Cada organización cuenta con sus propios controles y protocolos, por lo que la presente propuesta no busca obligar el uso de alguna herramienta, formato ni documento en particular. Lo esencial es tener por escrito la aceptación del cliente de la solución entregada, la cual se documentó plenamente al inicio del proyecto, donde se definió cuál sería el estado de "terminado" del proyecto y sus actividades. Este documento inicial es sumamente importante, pues provee al equipo de trabajo de herramientas para entender y visualizar el alcance del trabajo, con el objeto de medir sus progresos y resultados, además de evitar especulación entre los distintos involucrados al tener entendimientos diferentes de lo que supone terminar las actividades del proyecto.

En esta etapa posterior a todos los *Sprints* realizados, validados y aprobados por el cliente, el equipo de trabajo en conjunto con los demás involucrados deberán asegurarse de, al menos, lo siguiente:

- Cierre de contratos establecidos para el proyecto en cuestión, cuando ello aplique sin afectar los cumplimientos legales correspondientes.
- En caso de aplicar, cerrar los compromisos laborales abiertos para el alcance del proyecto con actores que hayan formado parte del *staff* del equipo de trabajo.
- Documentar la aceptación por escrito o medios digitales legalmente aceptados por parte del cliente.

- Documentar las lecciones aprendidas del proyecto, minutas de trabajo, decisiones de pivoteo o perseverancia posterior a los *Sprints*.
- Respaldar los códigos fuente del código desarrollado, así como entregar formalmente al cliente la documentación oficial del proyecto.
- En caso de aplicar, devolver cualquier bien que haya sido prestado para efectos de realizar los adecuados entendimientos de las necesidades de los clientes, elementos físicos, herramientas, materia prima, entre otros.
- Finalmente, hacer entrega formal del proyecto, donde se expliciten las garantías a cubrir, donde se deslinden derechos y obligaciones, así como se establezcan los términos de la entrega de la solución al cliente.

8 Discusión

8.1 Acerca de la propuesta

La propuesta tecnológica resultante busca ayudar a los equipos de desarrollo de software en la construcción de soluciones que generen valor a sus clientes, que sean de utilidad, escalables, que eviten desperdicios en sus procesos de desarrollo, pero sobre todo, que el esfuerzo realizado tenga siempre a la persona en el centro de la toma de decisiones. Los equipos de desarrollo que utilizan *Scrum* buscan crear soluciones ágiles y de valor para sus clientes, con valor temprano mediante su proceso flexible e iterativo (Schwaber y Sutherland, 2017); sin embargo, como se discutió, el solo uso de este marco puede ocasionar problemáticas que sean capaces de poner en riesgo los objetivos del equipo de desarrollo y sobre todo, los del cliente.

De acuerdo con lo expuesto, no se busca poner en tela de juicio los grandes beneficios que acarrea el uso de *Scrum* como marco de trabajo para la gestión ágil de proyectos de desarrollo de software. La intención de la presente propuesta es complementar las áreas de oportunidad encontradas y analizadas con el objeto de que los equipos de desarrollo de software sigan utilizando *Scrum*, pero con la adición de algunas técnicas y herramientas que se desprenden de otros métodos buscando incrementar las probabilidades de éxito de dichos proyectos.

Inicialmente, la propuesta presentada considera como su columna vertebral el uso íntegro de *Scrum*, sus artefactos y pasos. Empero, este no entra en acción de inmediato. Antes de comenzar a desarrollar, se sugiere el inicio formal del proyecto. El proceso de inicio del proyecto busca la formalización de este, con una declaración inicial del alcance, donde se describe cuál es la motivación que detona la necesidad, qué es lo que se espera que sea resuelto con la solución a desarrollar, entre otros. Tal cual se indica, la presente propuesta no exige el uso de algún método o documento; pero se sugieren a manera de ejemplos un par de documentos altamente utilizados pero, a final de cuentas, se es libre en cuanto a que documento o método utilizar.

A estas alturas, tampoco es posible comenzar con el proceso de desarrollo utilizando *Scrum*, pues no entra en acción hasta que ya se cuenta con el proceso de obtención de requerimientos culminado, esto es, hasta el tercer paso de la presente propuesta. Dicho proceso se lleva a cabo con el uso de uno de los dos métodos complementarios propuestos que es el

Design Thinking. Hasta este punto, nada en el uso de *Scrum* ha cambiado, solo se sugiere que antes de comenzar a desarrollar se lleve a cabo un proceso donde se puedan comprender de forma íntegra las necesidades, realidades y experiencias del cliente para efectos de que tal desarrollo considere todo lo que realmente necesita. Mediante este paso, se busca lo siguiente:

- Obtener los requerimientos de viva voz del cliente.
- Analizar en conjunto dichos requerimientos, en busca de entenderlos y refinarlos al grado de validar que realmente vayan a solventar la necesidad que detona el requerimiento.
- Conocer y entender el día a día del cliente para ayudarle a proponer en caso de ser necesario, ajustes a sus requerimientos para considerar lo que realmente necesita,
- Entender su entorno y problemáticas, en aras de considerarlos en los requerimientos,
- Definir una lista final de requerimientos así como una o varias propuestas de solución preliminares,
- Entre otros.

Se busca que al final del proceso, se obtengan las historias del usuario que serán parte del proceso de desarrollo, las cuales integrarán al *Product Backlog*, del cual el *product owner* podrá seleccionar cuales ir integrando en cada *Sprint.*

Una vez obtenidos los requerimientos, el proceso de desarrollo puede comenzar, y es donde se llevan a cabo todos los eventos y artefactos de *Scrum,* en cuyo proceso no cambia nada en lo absoluto. Solo se integra al finalizar cada *Sprint* al uso del otro método sugerido que es el *Lean Startup*, el cual se detona durante el *Sprint Review* como otro paso denominado "Prueba y validación de la solución", cuya intención es asegurar que lo desarrollado al momento siga siendo lo que el cliente necesita; si le sigue generando valor y constatar que no haya habido cambios significativos en la realidad y necesidades del cliente. Si los hay, se deberán hacer los ajustes necesarios para asegurar que la solución final es exactamente lo que el cliente necesita, siempre.

Por último, al finalizar el proceso de desarrollo y después de la cantidad de *Sprints* necesarios, se cierra el proyecto. La intención de este paso denominado "Cierre del proyecto" es cerrar toda actividad, acuerdo o proceso abierto relacionado con este para efectos de culminar de forma ordenada y formal.

8.2 Aporte

La presente propuesta no es la única que sugiere la integración del *Design Thinking* y *Lean Startup* con *Scrum*, pero sí presenta ciertas diferencias de otras propuestas previamente elaboradas. Existe una propuesta denominada el "*Nordstrom Model"*, el cual también sugirió el uso temprano del *Design Thinking* antes siquiera de codificar nada. Sin embargo, dicho modelo sugiere el uso del *Design Thinking* durante todo el proceso de levantamiento de requerimientos, desarrollo y validación de la solución. De Paula y Araújo (2016) mencionaron que esta visión puede ayudar a solventar de mejor manera los problemas que surjan durante el proyecto.

En su trabajo, Ximenes et al. (2015) propusieron el "Converge Model", el cual también consideró el uso del *Design Thinking* y *Lean Startup*, con ciertas diferencias que son notables. Inicialmente, está el hecho de que se usa un híbrido entre *Scrum* y *XP Programming* para la gestión ágil de desarrollo de software. Por otro lado, sugirieron llevar una sesión semanal del ciclo de *Lean Startup* denominada "construir, medir, aprender"; donde se puedan validar las hipótesis con que se cuenta al momento y, utilizando el *Design Thinking* para buscar soluciones bajo demanda de los problemas encontrados en estas. Por otro lado, existe un método llamado "Innodev" (Dobrigkeit y de Paula, 2017) que usa los mismos componentes, comenzando con el proceso del *Design Thinking*, pero continuando con dos fases de desarrollo denominadas:

- Inicial.
- Final.

En esta última fase se llevan a cabo los ciclos de *Lean Startup* para validar las hipótesis iniciales y utilizando el *Design Thinking* para buscar soluciones a las posibles problemáticas encontradas. La propuesta realizada y presentaba en el presente trabajo se diferencia de la siguiente manera con respecto a las mencionadas. Con respecto al *Nordstrom Model*, se cuenta con una estructura de trabajo mucho más sencilla que posibilita a equipos familiarizados con *Scrum*, adaptar algunos pasos de su trabajo de manera sencilla y sin necesidad de alterar los conceptos actuales. Usar el *Design Thinking* al comienzo del proyecto permite obtener una visión realista y empática de las necesidades del cliente, pero al continuar con *Scrum* puro más el ciclo de validación reforzado con *Lean Startup*, facilita su adopción y continuo. Por otro lado, esta propuesta tecnológica tiene un énfasis mayor en el uso del *Lean Startup* que el modelo *Nordstrom*.

Con respecto al *Converge Model*, esta propuesta simplifica su adopción y uso utilizando únicamente *Scrum* para el proceso ágil de gestión de desarrollo de software y sin alteración a sus procesos. Por otro lado, las sesiones de *Lean Startup* se llevaron a cabo al finalizar los *Sprints*, no de forma semanal rigurosa como el modelo con que se compara.

Por último, con respecto al modelo *Innodev*, la presente propuesta cuenta con una única fase de desarrollo que se compone por cantidades ilimitadas de ciclos, las cuales dependen de lo que defina el equipo de trabajo. Por otro lado, también se emplea el *Lean Startup* al finalizar cada *Sprint* y no solo en etapas maduras del desarrollo, buscando con ello desperdicios de recursos y poder realizar cambios (pivotar) a tiempo. La propuesta tecnológica que se plantea busca ser utilizada por equipos ya familiarizados con *Scrum*, de manera que su implementación sea sencilla y natural. Se incorporan dos marcos de manera sencilla y armónica en partes del proceso que no afectan cómo los equipos venían trabajando previamente con *Scrum,* pero ayudándoles a evitar las problemáticas discutidas.

9 Conclusión

La propuesta tecnológica resultante busca ofrecer a equipos de desarrollo de software que utilizan *Scrum* como su método ágil de gestión de proyectos una alternativa que les permita seguir usando sus técnicas y herramientas actuales, pero potenciadas con los beneficios que acarrea el uso de otros métodos como el *Lean Startup* y *Design Thinking*. Un aspecto clave es la forma en que estos dos métodos complementarios se suman, que no cambian en lo absoluto las maneras de trabajo que sugiere *Scrum* pero, permite evitar los riesgos asociados y ya discutidos de su solo uso.

Otro aspecto clave reside en que, al respetar la forma de trabajo estándar que sugiere *Scrum*, la adopción de esta propuesta es más sencilla para equipos acostumbrados al uso de tal método, reduciendo la natural complejidad de todo cambio. Los beneficios esperados con el uso de esta propuesta tecnológica se espera que se enfoquen complemente en el ser humano. Para los buenos resultados de las organizaciones, es primordial que los proyectos que ejecutan sean exitosos y cumplan con los objetivos esperados. Al poner al ser humano en el centro de la toma de decisiones en los proyectos, escuchando al cliente, conociendo sus necesidades, experiencias y su día a día, se podrá tener una mayor probabilidad de éxito al final, puesto que de inicio el cliente lleva la voz cantante en el diseño del proyecto, así como su constante validación y aprobación de los avances, en busca de dar valor inmediato y de manera incremental, y por último, una vez culminadas las actividades, se tendrá un proyecto alineado a lo que el cliente realmente necesita y quiere.

Finalmente, con la presente propuesta tecnológica no se busca proponer un modelo perfecto ni que sea aplicable de manera universal. Cada equipo de desarrollo alrededor del mundo cuenta con sus características y necesidades propias. La intención es proponer una alternativa de valor que pueda ser utilizable y que brinde los elementos necesarios para incrementar las probabilidades de éxito de los equipos que le adopten. Esta es una versión inicial y, por ende, cuenta con potencial de posibles mejoras y adaptaciones futuras.

10 Siguientes pasos

A pesar de que *Scrum* es uno de los métodos ágiles de gestión de proyectos de desarrollo de software más utilizados a nivel mundial, existen otros métodos que son altamente utilizados y que podrían ser considerados en este método propuesto. Una posible área de oportunidad pudiese ser la integración de la metodología ágil de forma neutral, tomando lo mejor de los métodos más utilizados de manera que se potencie aún más los posibles resultados e incluso simplificando y flexibilizando más su implementación y uso.

Otra área de oportunidad reside en profundizar más en la implementación y el uso de los métodos complementarios discutidos: *Design Thinking* y *Lean Startup*, dado que es altamente posible que equipos que no los conozcan ni les hayan usado previamente tengan problemas en su adopción y uso.

A lo anterior, se suma analizar el antes y después del uso de la presente propuesta en ambientes reales, de manera que los resultados permitan comprobar o refutar lo propuesto, por lo que se puedan encontrar más áreas de oportunidad en las cuales trabajar.

11 Referencias

Ameijide, L. (2016). *Gestión de proyectos según el PMI* [Tesis de grado]. Barcelona: Universitat Oberta de Catalunya.

Ángel, C. E. (2018). El método Lean Startup: una revisión teórica. *Gestión Ingenio y Sociedad, 1*, 18-25.

Asuad, N., & Vázquez, C. (2014). *Marco lógico de la investigación científica* [Diapositivas]. Universidad Nacional Autónoma de México: https://bit.ly/3HxktGf

Ayala, Y. J. (2019). El método Lean Startup. Cómo crear empresas de éxito utilizando la innovación continua de Eric Ries. *Emprende y Transforma, 1*(1), 83-88. https://doi.org/10.33829//emprendeytransforma-0101-2019-83-88

Basco, A. I., Beliz, G., Coatz, D., & Garnero, P. (2018). *Industria 4.0. Fabricando el futuro.* Banco Interamericano de Desarrollo. http://dx.doi.org/10.18235/0001229

Besner, C., & Hobbs, B. (2012). An empirical identification of project management toolsets and a comparison among project types. *Project Management Journal, 43*(5), 24-46. https://doi.org/10.1002%2Fpmj.21292

Brown, T. (2008). Design Thinking. *Harvard Business Review*, 1-9.

Buelvas, V., & Rodríguez, U. (2021). *Manual del tesista.* UVR Correctores de Textos.

de Paula, D. F., & Araújo, C. C. (2016). Pet empires: combining design thinking, lean startup and agile to learn from failure and develop a successful game in an undergraduate environment. En C. Stephanidis (Ed.), *HCI International 2016 – Posters' extended abstracts. HCI 2016. Communications in computer and information science* (pp. 30-34). Springer. https://doi.org/10.1007/978-3-319-40548-3_5

Design Thinking en Español. (s.f.). *Inicio.* https://www.designthinking.es/inicio/index.php

Dobrigkeit, F., & de Paula, D. (2017). The best of three worlds - the creation of innodev a software development approach that integrates Design Thinking, Scrum and Lean Startup. En A. Maier, S. Škec, H. Kim, M. Kokkolaras, J. Oehmen, G. Fadel, . . . M. Van der Loos (Eds.), *21st International Conference on Engineering Design* (pp. 319-328). National University of Ireland Galway.

Esteves, J., & Pastor, J. (2001). Enterprise Resource Planning systems research: an annotated bibliography. *Communication of the Association for Information Systems, 7*, 1-52. https://doi.org/10.17705/1CAIS.00708

Flyvbjerg, B., & Budzier, A. (2011). *Why your IT project may be riskier than you think.* Harvard Business Review: https://bit.ly/3cocH3a

Ghezzi, A., & Cavallo, A. (2020). Agile business model innovation in digital entrepreneurship: Lean Startup approaches. *Journal of Business Research, 110*, 519-537. https://doi.org/10.1016/j.jbusres.2018.06.013

González, F., Calero, S. L., & Loaiza, D. F. (2019). *Comparación de las metodologías cascada y ágil para el aumento de la productividad en el desarrollo de software* [Tesis de grado]. Santiago de Cali: Universidad Santiago de Cali.

Gray, N. S. (2002). *Projects don't fail, people do: getting people to excel at what they do.* Project Management Institute Annual Seminars & Symposium: https://bit.ly/3nnw4zS

Gregory, P., Strode, D. E., AlQaisi, R., Sharp, H., & Barroca, L. (2020). Onboarding: how newcomers integrate into an agile project team. En V. Stray, R. Hoda, M. Paasivaara, & P. Kruchten (Eds.), *Processes in software engineering and extreme programming. XP 2020. Lecture notes in business information processing* (pp. 20-36). Springer. https://doi.org/10.1007/978-3-030-49392-9_2

Grossman-Kahn, B., & Rosensweig, R. (2012). Skip the silver bullet: driving innovation through small bets and diverse practices. En E. Bohemia, J. Liedtka, & A. Rieple (Eds.), *Leading innovation through design: proceedings of the DMI. 2012 International Research Conference* (pp. 825-840). DMI.

Gu, Y. Y., He, S. Y., & Lin, X. Y. (2018). Design thinking exploration from three disciplines that influence industry development. *Canadian Social Science, 14*(12), 62-66. http://dx.doi.org/10.3968/10765

Hardy-Vallee, B. (2012). *The cost of bad project management.* Business Journal: https://bit.ly/3oC13Ym

Hernández, L. M., & Bravo, B. (2020). Diseño de un procedimiento para la gestión ágil de proyectos de desarrollo de software alineados a la guía del PMBOK. *Revista Ibérica de Sistemas y Tecnologías de Información,* (E32), 229-241.

Herrera, E., & Valencia, L. E. (2007). Del manifiesto ágil sus valores y principios. *Scientia Et Technica, 13*(34), 381-386.

IDEO. (2015). *The field guide to human-centered design* (2.ª ed.). IDEO.

IDEO U. (s.f.). *What is design thinking?* https://bit.ly/3HwYYp9

International Organization for Standardization [ISO]. (2010). *ISO 9241-210:2010. Ergonomics of human-system interaction. Part 210: human-centred design for interactive systems.* https://www.iso.org/standard/52075.html

Ionel, N. (2008). Critical analysys of the scrum project management methodology. *The Academy of Economic Studies Bucharest, 17*(4), 435-441.

Izaurralde, M. P. (2013). *Caracterización de especificación de requerimientos en entornos ágiles: historias de usuario* [Tesis de especialización]. Córdoba: Universidad Tecnológica Nacional.

Jones, D. T., & Womack, J. P. (2003). *Lean thinking. Banish waste and create wealth in your corporation.* Free Press.

Jovanovic, P., & Beric, I. (2018). Analysis of the available project management methodologies. *Management:Journal of Sustainable Business and Management Solutions in Emerging Economies, 23*(3), 1-13. https://doi.org/10.7595/management.fon.2018.0027

Jovanovic, P., & Beric, I. (2018). Analysis of the available project management methodologies. *Management: Journal of Sustainable Business and Management Solutions in Emerging Economies, 23*(3), 1-13. https://doi.org/10.7595/management.fon.2018.0027

Kaur, R., & Sengupta, J. (2011). Software process models and analysis on failure of software development projects. *International Journal of Scientific & Engineering Research, 2*(2), 1-4.

KPMG. (2019). *Agile transformation.* https://bit.ly/30uu0wU

Labajo, E. (2015). *608104 el método pericial. Máster en Pericia Sanitaria* [Diapositivas]. Universidad Complutense de Madrid: https://bit.ly/32d24yw

Langenfeld, K. (2019). *Design thinking para principiantes: la innovación como factor para el éxito empresarial.* Edición Kindle.

Leau, Y. B., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). Software development life cycle AGILE vs. traditional approaches. *IPCSIT, 37*, 162-167.

Lindberg, T., Meinel, C., & Wagner, R. (2010). Design thinking: a fruitful concept for IT development? En C. Meinel, T. Lindberg, & H. Plattner (Eds.), *Design Thinking. Understanding innovation* (pp. 3-18). Springer. https://doi.org/10.1007/978-3-642-13757-0_1

Llamas, F., & Fernández, J. C. (2018). La metodología Lean Startup: desarrollo y aplicación para el emprendimiento. *Revista Escuela de Administración de Negocios,* (84). https://doi.org/10.21158/01208160.n84.2018.1918

Lledó, P., & Rivarola, G. (2006). *Alcance de la administración de proyectos en Latinoamérica.* PMI® Global Congress 2006: https://bit.ly/3CqmBMg

López, M. B., Arán, V., & Richaud, M. C. (2014). Empatía: desde la percepción automática hasta los procesos controlados. *Avances en Psicología Latinoamericana, 32*(1), 37-51. https://doi.org/10.12804/apl32.1.2014.03

Martínez, P. C. (2006). El método de estudio de caso: estrategia metodológica de la investigación científica. *Pensamiento & Gestión,* (20), 165-193.

Maxwell, J. A. (1996). *Qualitative research design: an interactive approach.* Sage.

Mieritz, L. (2012). *Survey shows why projects fail.* Gartner: https://gtnr.it/3qNmGYw

Moniruzzaman, A. B., & Hossain, S. A. (2013). Comparative study on agile software development methodologies. *Global Journals of Computer and Technology, 13*(7), 4-18.

Ohno, T. (1988). *Toyota production system: beyond large scale production.* Productivity Press.

Prades, L., Romero, F., Estruch, A., García-Domínguez, A., & Serrano, J. (2013). Defining a methodology to design and implement business process models in BPMN according to

the standard ANSI/ISA-95 in a manufacturing enterprise. *Procedia Engineering, 63*, 115-122. https://doi.org/10.1016/j.proeng.2013.08.283

Project Management Institute [PMI]. (2017a). *Guía PMBOK* (7.ª ed.). PMI.

Project Management Institute [PMI]. (2017b). *Pulse of the profession.* PMI.

Project Management Institute [PMI]. (2020a). *The Innovation Imperative.* https://bit.ly/30qUeAp

Project Management Institute [PMI]. (2020b). *El futuro del trabajo.* PMI.

Rączka, M. (2015). *Do your projects fail?: don't change the methodology or people.* PMI® Global Congress 2015: https://bit.ly/3HyTY3i

Radigan, D. (s.f.). *El backlog del producto: la lista de tareas pendientes definitiva.* Atlassian: https://bit.ly/3wZ5UXu

Razzouk, R., & Shute, V. (2012). What is design thinking and why is it important? *Review of Educational Research, 82*(3), 330-348. https://doi.org/10.3102%2F0034654312457429

Real Academia Española [RAE]. (2014a). *Experiencia.* Diccionario de la lengua española: https://dle.rae.es/experiencia

Real Academia Española [RAE]. (2014b). *Ciencia.* Diccionario de la lengua española: https://dle.rae.es/ciencia

Ries, E. (2012). *El método Lean startup: cómo crear empresas de éxito utilizando la innovación continua.* Deusto.

Rubrich, L. (s.f.). *A3 problem solving: what it is... and what it isn't.* Reliable Plant: https://bit.ly/3FrjfKZ

Sanhueza, J. M. (2017). *Metodología Lean-PMI para desarrollo de proyectos de software* [Tesis de grado]. Concepción: Universidad de Concepción.

Sarria, M. P., Fonseca, G. A., & Bocanegra-Herrera, C. C. (2017). Modelo metodológico de implementación de lean manufacturing. *Revista Escuela de Administración de Negocios,* (83), 51-71. https://doi.org/10.21158/01208160.n83.2017.1825

Satoglu, S., Ustundag, A., Çevikcan, E., & Durmusoglu, M. B. (2017). *Lean transformation integrated with industry 4.0: implementation methodology.* https://bit.ly/30BhKKU

Schibi, O. (2013). *Why PMOs do not deliver to their potential.* PMI® Global Congress 2013: https://bit.ly/3x3egx5

Schneider, J. (2017). *Understanding Design Thinking, Lean, and Agile.* O'Reilly Media.

Schwaber, K., & Sutherland, J. (2017). *The Scrum guide. The definitive guide to Scrum: the rules of the game.* Scrum Guides: https://bit.ly/3DyB7TS

Schwaber, K., & Sutherland, J. (2020). *La guía de Scrum. La guía definitiva de Scrum: las reglas del juego.* Scrum Guides: https://bit.ly/3qO0Iog

Scrum. (s.f.). *What is Scrum?* https://bit.ly/3FhooFs

Shaker, K. (2010). Why do projects really fail? *PM Network, 24*(7), 20-21.

Signoretti, I., Salerno, L., Marczak, S., & Bastos, R. (2020). Combining user-centered design and lean startup with agile software development: a case study of two agile teams. En V. Stray, R. Hoda, M. Paasivaara, & P. Kruchten (Eds.), *Agile processes in software engineering and extreme programming* (pp. 39-55). Springer. https://doi.org/10.1007/978-3-030-49392-9_3

Špundak, M. (2014). Mixed agile/traditional project management methodology – reality or illusion? *Procedia - Social and Behavioral Sciences, 119*(19), 939-948. https://doi.org/10.1016/j.sbspro.2014.03.105

Stray, V., Hoda, R., Paasivaara, M., & Kruchten, P. (Eds.). (2020). *Agile processes in software engineering and extreme programming: 21st International Conference on Agile Software Development, XP 2020, Copenhagen, Denmark, June 8-12, 2020, Proceedings.* Springer. https://doi.org/10.1007/978-3-030-49392-9

Vilkki, K. (2010). When agile is not enough. En P. Abrahamsson, & N. Oza (Eds.), *Lean enterprise software and systems* (pp. 44-47). Springer. https://doi.org/10.1007/978-3-642-16416-3_6

Wallace, W. (2014). *Gestión de proyectos.* Edinburgh Business School.

Wellingtone. (2018). *The state of project management. Annual survey 2018.* https://bit.ly/3CldMUd

Ximenes, B. H., Alves, I. N., & Araújo, C. C. (2015). Software project management combining agile, lean startup and design thinking. En A. Marcus (Ed.), *Design, user experience, and usability: design discourse* (pp. 356-367). Springer. https://doi.org/10.1007/978-3-319-20886-2_34

Yin, R. (2009). *Case study research: design and methods* (4.ª ed.). Sage.

12 Anexos

Anexo 1. Técnicas y herramientas más utilizadas y conocidas en el proceso de empatía del Design Thinking

Para efectos de empatizar con el cliente, el listado comprende solo algunas de ellas, dado que cada equipo de trabajo puede utilizar las técnicas y herramientas que mejor le convenga y sirvan. A partir de lo anterior, el presente listado es para efectos ilustrativos y parte de la sugerencia del sitio Design Thinking en Español (s.f.):

- Mapa de actores: representación gráfica que sirve para identificar a los actores que participarán en el uso de un producto o servicio.
- Inmersión cognitiva: es el proceso de ponerse en el lugar del usuario para experimentar su realidad, con base en vivir sus experiencias diarias. La idea es generar sintonía con el usuario al entender su contexto, dejando de lado concepciones propias.
- Interacción constructiva: es una práctica que permite que el usuario o un grupo de usuarios relaten en voz alta el o los procesos con los que actualmente trabajan, de manera que el equipo de desarrollo pueda entender la realidad que viven.
- Mapa mental: ayuda a evaluar la relación entre diferentes variables sobre un tema central, con lo que se obtiene una visión global y simplificada de este.
- Mood board: herramienta visual que se compone de una selección de imágenes, fotografías, así como diversos materiales que puedan indicar conceptos que, al ser difíciles de expresar con palabas, ayuden a obtener conceptos relacionados con la posible solución.
- Observación encubierta: implica conseguir información objetiva sin participar en los procesos con el usuario para evitar toda interferencia, mientras se mantiene la objetividad mediante la observación de la interacción del usuario con algún producto o servicio, sin que el usuario se entere.
- ¿Qué?, ¿cómo?, ¿por qué?: a partir de la colección de material fotográfico obtenido por el equipo de desarrollo, que retrata las escenas donde vive el usuario y su realidad, se busca que cada uno de ellos pueda responder tales preguntas y posibiliten entender mejor lo que vive y cómo lo vive el usuario.

- Entrevistas cualitativas: se pretende empatizar con el usuario, al entender sus motivaciones, emociones y forma de pensar, gracias entrevistas con preguntas concretas.
- Experiencia del cliente: busca generar el mapa de la experiencia del cliente, el cual muestra las diferentes actividades que un usuario lleva a cabo durante el uso de un producto o servicio. Es sumamente útil para diseñar una solución e, incluso, para detectar los conflictos que los clientes tienen cuando usan sus productos y servicios existentes y las mejoras que ellos requieren.