Universidad Nacional Autónoma de México

Programa de Posgrado en Ciencias de la Administración

Factores que impactan a proyectos de desarrollo de software, en los que se emplea Scrum.

T e s i s

Que para optar por el grado de:

Maestría en Informática Administrativa

Presenta:
Margarita Hernández Galindo

Tutor:
Angélica María Ramírez Bedolla
Posgrado de la Facultad de Contaduría y Administración

Ciudad de México, septiembre de 2021

Dedicatoria

A mis padres
Por su cariño y apoyo en cada momento de mi vida

A Gina
Por impulsarme a seguir con mi formación académica y desarrollo personal

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología por el apoyo otorgado para la realización de este trabajo, al Programa de Posgrado en Ciencias de la Administración por la oportunidad de ampliar mis conocimientos y ser una fuente de conocimiento constante.

Gracias a mi directora de tesis la M.A. Angélica María Ramírez Bedolla por el conocimiento compartido, las valiosas recomendaciones y por el tiempo dedicado a la revisión de este trabajo. A todos los profesores con los que tuve el gusto de trabajar durante mi estancia en el posgrado, cada uno a través de su experiencia aportó grandes aprendizajes que sin duda serán de ayuda en mi desarrollo profesional.

Quiero agradecer especialmente a Brian por todos los consejos y recomendaciones desde que decidí iniciar la maestría y por todo el tiempo de asesoría que hizo posible la conclusión de este escrito.

Tuve la oportunidad de hacer grandes amigos en el posgrado, gracias a ellos y a todos mis compañeros de clase que aportaron su experiencia, dudas y conocimiento, haciendo más enriquecedora esta etapa.

Factores de éxito en proyectos de desarrollo de software en los que se emplea *Scrum*

Resumen

Año tras año el uso de la tecnología cobra mayor relevancia a nivel mundial, sin importar el giro o actividad comercial, las empresas han incorporado su uso para potencializar su negocio, por ello la industria del desarrollo de software también ha crecido, tan solo “en 2017 dicha industria presentó un incremento del 7.84% respecto al empleo generado en 2016” (Pliego F., 2018, p. 8), de acuerdo con cifras del INEGI.

Un eje que debe incorporarse en la creación de software para asegurar la calidad con la que el mismo es producido y entregado a los usuarios finales es la ingeniería de software, la cual contempla diversos enfoques y métodos para su desarrollo. Existen diversas metodologías para el desarrollo de software, están las metodologías tradicionales que fueron propuestas entre los 70’s y 80’s y que dominaron la industria durante bastante tiempo debido a su rentabilidad y a su carácter predictivo y sostenible; es a finales de los 80’s y principios de los 90’s que aparecen las metodologías ágiles, mismas que no han parado de evolucionar y de incorporarse a los proyectos que trabajan con altos niveles de innovación e incertidumbre, pues brindan la flexibilidad necesaria para la incorporación de cambios que en una gestión tradicional se tornaba más complicado.

Dentro del enfoque ágil para el desarrollo de software se encuentran diversos *frameworks* que buscan estructurar la forma de trabajo tanto de los programadores como de los responsables de negocio, además de beneficiar proyecto con la adaptabilidad de sus procesos y técnicas. Uno de los marcos de trabajo más empleados es Scrum, el cual es un conjunto de prácticas sugeridas que intentan potencializar la colaboración del equipo, pero sobre todo la entrega de valor temprana al usuario final.

Así como la industria del desarrollo de software ha crecido en México, también lo ha hecho el uso de Scrum, con el fin de conocer mejor el uso de Scrum y su estado actual en la industria, la pregunta en la que se centra esta investigación es la

siguiente: “¿Por qué Scrum no logra implementarse de manera correcta en los proyectos de desarrollo de software?”

Esta investigación es de tipo cualitativa, las encuestas para el análisis de datos fueron aplicadas a diversos profesionistas que ya tienen tiempo trabajando con Scrum. A través de la información obtenida se identificaron diferencias en la forma en la que se utiliza el marco de trabajo contra lo que propone la teoría y se tomaron como punto de partida para realizar una serie de sugerencias para mejorar la adopción de Scrum.

Abstract

Year after year, the use of technology becomes more relevant worldwide, regardless of the line of business or business activity, companies have incorporated its use to enhance their business, which is why the software development industry has also grown. Just “in 2017 this industry presented an increase of 7.84% compared to the employment generated in 2016” (Pliego F., 2018, p. 8), according to INEGI figures.

A core idea that must be incorporated into the creation of software to ensure the quality with which it is produced and delivered to end users is software engineering, which contemplates various approaches and methods for its development. There are several methodologies for software development, such as the traditional methodologies that were proposed between the 70's and 80's and that dominated the industry for a long time due to their profitability and their predictive and sustainable nature. It is at the end of the 80's and the beginning of the 90's when agile methodologies appear and have not stopped evolving and being incorporated into projects that work with high levels of innovation and uncertainty, since they provide the necessary flexibility for the incorporation of changes that in a traditional management became more complicated.

Within the agile approach to software development there are various frameworks that seek to structure the way of working of both programmers and business managers, in addition to benefiting the project with the adaptability of its processes and techniques. One of the most used frameworks is Scrum, which is a set of suggested practices that try to enhance team collaboration, but above all, the early delivery of value to the end user.

Just as the software development industry has grown in Mexico, so has the use of Scrum; in order to better understand the use of Scrum and its current state in the industry, the question on which this research focuses is the following: "Why is Scrum not being implemented correctly in software development projects?"

This research is qualitative, the data analysis surveys were applied to various professionals who have been working with Scrum for some time. Through the

obtained information, differences were identified in the way in which the framework is used versus what the theory proposes and they were taken as a starting point to make a series of suggestions to improve the adoption of Scrum.

Contenido

Índice de Tablas

Índice de Figuras

Introducción

Planteamiento del problema

Las empresas mexicanas dedicadas al desarrollo de software han adoptado *Scrum* como una herramienta para agilizar la producción de software; sin embargo, se percibe que su implementación no ha brindado todos los beneficios que plantean sus autores Ken Schwaber y Jeff Sutherland. Scrum es un marco de trabajo que segmenta un proyecto partiendo de la identificación de funcionalidades que se desarrollarán en *iteraciones* cortas de 4 semanas o menos. El objetivo por conseguir al final de cada iteración es entregar valor al negocio por medio de la liberación continua de características funcionales de software, objetivo que no siempre es alcanzado.

Las empresas que se dedican al desarrollo de software sentaron las bases de su operación en la conocida *metodología de cascada*, en la que se elabora un plan de trabajo detallado que describe las actividades a realizar desde el primer hasta el último día que durará proyecto y en el que se pretende que en la etapa inicial del proyecto se tengan claros los requerimientos y que estos no cambien durante su ejecución, por lo que si el desarrollo se enfrenta a altos niveles de incertidumbre hay que incorporar los cambios a través de un ejercicio de gestión que puede resultar demasiado burocrático y esto puede impactar significativamente el costo y tiempo del proyecto, ocasionando a su vez pérdidas económicas para la empresa desarrolladora, retrasos en las entregas y por consiguiente la molestia de los clientes.

Justificación

Las empresas mexicanas dedicadas al desarrollo de software han adoptado Scrum como una herramienta para agilizar el desarrollo de sus proyectos; a pesar de su implementación, no se obtienen los resultados esperados; por ello es fundamental identificar cuáles son estos factores para encontrar la forma de solventarlos e implementar este marco de trabajo de forma correcta.

En esta investigación se recabaron datos sobre la ejecución de proyectos de desarrollo de software ya concluidos, con la finalidad de identificar prácticas y herramientas, así como el impacto que tienen sobre los proyectos.

Al identificar qué elementos impiden la correcta adopción de Scrum, puede abordarse una estrategia correcta y de atención oportuna de mejoras, misma que permita obtener todos los beneficios que brinda este marco de trabajo ágil.

Pregunta de investigación

¿Por qué Scrum no logra implementarse de forma correcta en los proyectos de desarrollo de software?

Objetivo general y particulares

Analizar factores que favorecen la efectividad en proyectos de desarrollo de software en los que se usa Scrum como herramienta para su administración, para proponer una estrategia que permita mejorar las condiciones en las que se desarrolla el software.

- Comparar los artefactos, reuniones y otros elementos que propone Scrum como marco de trabajo, contra como suelen ser usados por los profesionales en el desarrollo de proyectos de software entrevistados para el estudio.
- Generar observaciones para mejorar la efectividad de Scrum, con base en las fortalezas y debilidades identificadas a través de la información recabada en las entrevistas.

Hipótesis

Scrum no logra implementarse de manera efectiva debido a factores de diversa índole, como: características propias de cada proyecto, prácticas arraigadas de metodologías tradicionales, estricto control sobre la liberación temprana de funcionalidades a nivel organizacional e incluso el rechazo del marco de trabajo por parte del personal involucrado; es decir, los factores que limitan los resultados de

Scrum se relacionan con la estructura organizacional y la cultura laboral en la que se desarrollan los proyectos.

Diseño y alcance

Esta investigación es de carácter cualitativo: pues se abordó desde la perspectiva profesional que se tiene sobre el tema, tanto del investigador como de los participantes en el estudio. Tiene un horizonte de tiempo transversal: se realizó en un periodo no mayor a dos años, tiempo que se dedicó al desarrollo de este trabajo.

Para conseguir los objetivos planteados se consultaron distintas fuentes de información, como: bibliografía relacionada, material elaborado por empresas certificadoras en la materia, estudios realizados por diversas compañías, etc. También se entrevistaron a profesionistas que cuentan con experiencia usando Scrum y las preguntas aplicadas se enfocaron en proyectos concluidos para poder recabar información útil.

Resumen capitular

Capítulo 1. Modelos y metodologías para el desarrollo de software.
Se introduce al lector en los modelos de desarrollo de software y las metodologías derivadas de los mismos, explicando en qué consiste cada una y sus particularidades.

Capítulo 2. La industria del desarrollo de software.
Se habla sobre el desarrollo que ha tenido la industria del software tanto a escala internacional como en México.

Capítulo 3. Scrum
Se aborda el origen de Scrum, sus características y particularidades, principales beneficios, roles, flujo de trabajo, reuniones, etcétera.

Capítulo 4. La cultura organizacional con Scrum
Se explica la relación de la estrategia organizacional con la adopción de Scrum como una cultura de trabajo colaborativo que impulsa la consecución de objetivos.

Capítulo 5. Investigación y análisis de resultados
En este capítulo se describe a detalle el camino que siguió la investigación, las herramientas utilizadas, la selección de la muestra y los resultados obtenidos a través de la aplicación de cuestionarios.

Capítulo 6. Conclusiones
Se comparan los resultados obtenidos en la investigación con la hipótesis propuesta y se enlistan las áreas de oportunidad detectadas al momento de adoptar Scrum.

Capítulo 1. Modelos y metodologías para el desarrollo de software

Desarrollar software involucra el uso de habilidades diversas, habilidades técnicas y blandas como: la negociación, trabajo en equipo, resolución de problemas, administrativas, entre otras; estas habilidades, además de ser desarrolladas por el personal involucrado en un proyecto, deben ser inherentes a la organización y sus procesos de trabajo.

Cada proyecto tecnológico tiene sus propias características y es importante tomarlas en consideración para gestionarlo eficientemente y concluirlo con éxito. La naturaleza diversa de los proyectos de desarrollo de software ha derivado también en una amplia variedad de modelos y metodologías para su ejecución.

De acuerdo con Garcés y Egas (2015), la década de los 60's se caracterizó por la "búsqueda de alternativas para esquematizar de alguna manera la producción de software" (Gamboa, 2018, p. 4) y así surge la *Ingeniería de software,* que "incluye procesos, métodos y herramientas que permiten elaborar a tiempo y con calidad sistemas complejos basados en computadoras" (Pressman, 2010, p. 21). Modelos y metodologías establecen las bases sobre las que la industria de desarrollo de software se desenvolvió durante los años siguientes y que siguen usándose hoy en día.

El concepto *Modelo de desarrollo de software* es comúnmente confundido con *Metodología de desarrollo de software*, sin embargo, no son lo mismo. "Un modelo de desarrollo de software es una descripción simplificada del proceso para el desarrollo de software, presentada desde una perspectiva específica" (Sommerville, 2005, p. 8). Un modelo de desarrollo de software es una abstracción del proceso de desarrollo del software.

En cambio, de acuerdo con Maida y Pacienzia (2015), "Una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información" (Maida, 2015, p. 13). Sobre la misma línea, Avon y Fitzgerald (1995) aportan una explicación más amplia, apuntan que: "Una metodología es una colección de procedimientos,

técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en subfases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo" (Gómez, 2010, p. 70).

Derivado de lo anterior podemos sentar que el objetivo de una metodología de desarrollo de software es planificar y controlar de manera eficiente los procesos involucrados en la ejecución del proyecto, apoyándose de técnicas y herramientas apropiadas.

Los dos conceptos descritos, forman parte de la Ingeniería de software, que tiene como principal objetivo la entrega de sistemas de calidad, solo que abordan perspectivas distintas: un modelo es una representación ideal del proceso de desarrollo, en cambio una metodología incorpora fases, procesos y técnicas a ejecutar para garantizar el desarrollo de buen software. Los modelos de desarrollo de software, al ser una abstracción de procesos sugeridos, sientan las bases de las metodologías que se han implementado a lo largo del tiempo en la industria tecnológica, por lo que haremos una revisión de los principales modelos de desarrollo, para así comprender su esencia y evolución.

1.1. Modelo de cascada

Es de los modelos más estudiados e implementados en la industria, "El modelo de cascada, a veces llamado *Ciclo de vida clásico* sugiere un enfoque sistemático y secuencial para el desarrollo de software, que comienza con la especificación de requerimientos por parte del cliente y avanza a través de la planeación, modelado, construcción y despliegue, para concluir con el apoyo del software terminado" (Pressman, 2010, p. 34).

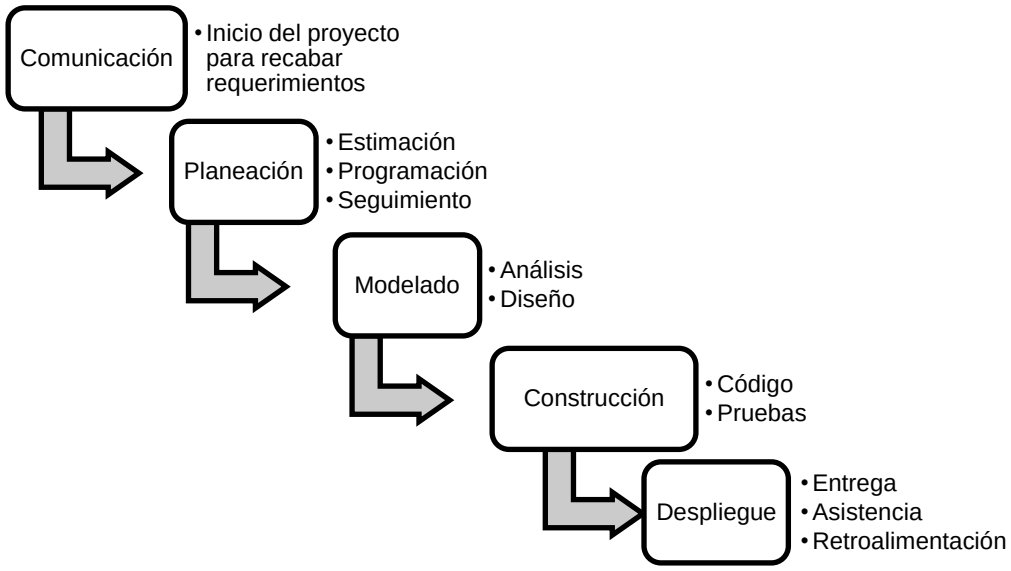

Fig. 1 Pressman R. (2010). *Modelo de cascada*.

Usando este modelo, el proyecto se divide en fases que se ejecutan de manera secuencial y el producto funcional es entregado al final del proyecto; se hace hincapié en la planeación por tiempos de entrega y avances, por lo que se invierte una mayor cantidad de tiempo y recursos en el análisis y diseño; requiere personal altamente capacitado para reducir errores y el cliente se ve poco involucrado durante el desarrollo.

Del modelo de cascada se desprende la metodología del mismo nombre, la cual sigue un flujo de proceso lineal para la ejecución secuencial de sus fases, lo que implica sostenibilidad en los requerimientos durante todo el proyecto, pues la incorporación de cambios en el momento del despliegue podría resultar más costosa por el cambio de alcance.

Fig. 2 Pressman R. (2010). *Flujo de proceso lineal*.

1.2. Modelo Incremental

“El modelo incremental aplica secuencias lineales en forma escalonada a medida que avanza el calendario de actividades. Cada secuencia produce incrementos de software susceptibles de entregarse” (Pressman, 2010, p. 35).

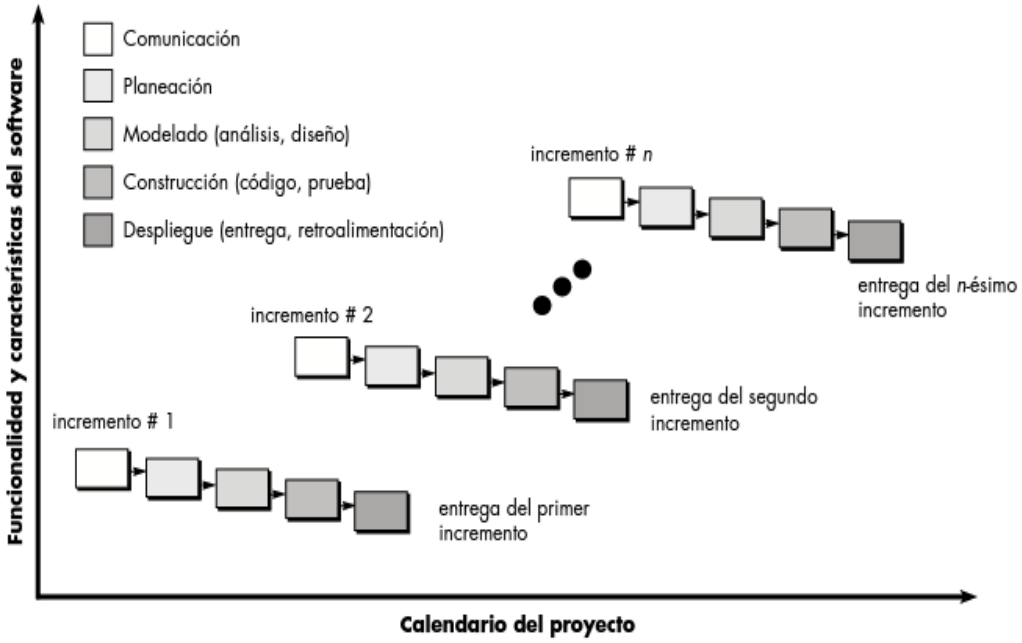

Fig. 3 Pressman R. (2010). *Modelo incremental*.

Este modelo usa un flujo de proceso evolutivo en el que, al trabajar de manera cíclica, es posible tener una versión más susceptible de entrega. El enfoque que utiliza parte de planear una implementación inicial con los requisitos esenciales, para que el usuario pueda utilizarlo y evaluar si es lo que requiere y de ser así continuar con su construcción o hacer los cambios correspondientes. Las actividades de comunicación, planeación, modelado, construcción y despliegue se repiten en cada incremento.

El principal atractivo del modelo incremental es la entrega temprana de valor al cliente, reduciendo el riesgo producido por los cambios que surgen en el desarrollo del proyecto, lo cual lo hace muy flexible. Este modelo de desarrollo es el precursor de las metodologías ágiles como *Kanban* o Scrum, su diferenciador principal es que se necesita que los requerimientos sean descubiertos en los primeros incrementos para reducir la incertidumbre.

1.3. Modelo de proceso evolutivo

Un sistema de información complejo tiende a evolucionar durante su desarrollo, ya sea porque los *stakeholders* no tienen certeza de lo que quieren o porque se trata de un proyecto de innovación y se tiene que trabajar en el descubrimiento de necesidades de los usuarios o del mercado mismo. En estos casos se recomienda un modelo de desarrollo que se adapte a un producto cambiante. “Los modelos evolutivos son iterativos. Se caracterizan por la manera en la que permiten desarrollar versiones cada vez más completas del software” (Pressman, 2010, p. 36). Este modelo tiene dos vertientes:

- Prototipado: Este modelo es recomendado cuando los requerimientos no son del todo claros. Inicia con un primer contacto (comunicación) con el cliente, se identifican los requerimientos que se tienen más claros y se realiza un modelado rápido del que se obtiene el primer prototipo, el cual se somete a retroalimentación y se sustituye por un nuevo sistema. El objetivo del prototipado es detallar requerimientos.

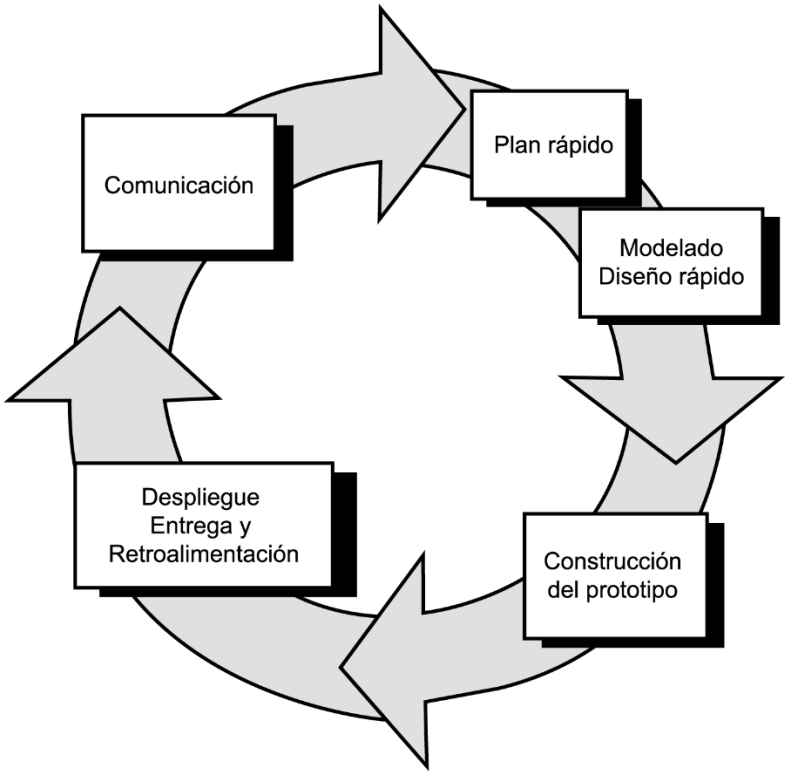

Fig. 4 Pressman R. (2010). *Modelo prototipado*.

- Espiral: El modelo espiral usa prototipos como mecanismo de reducción de riesgos y permite usar esta estrategia en cualquier etapa del desarrollo de un sistema, aplicando un enfoque parecido al de desarrollo lineal.

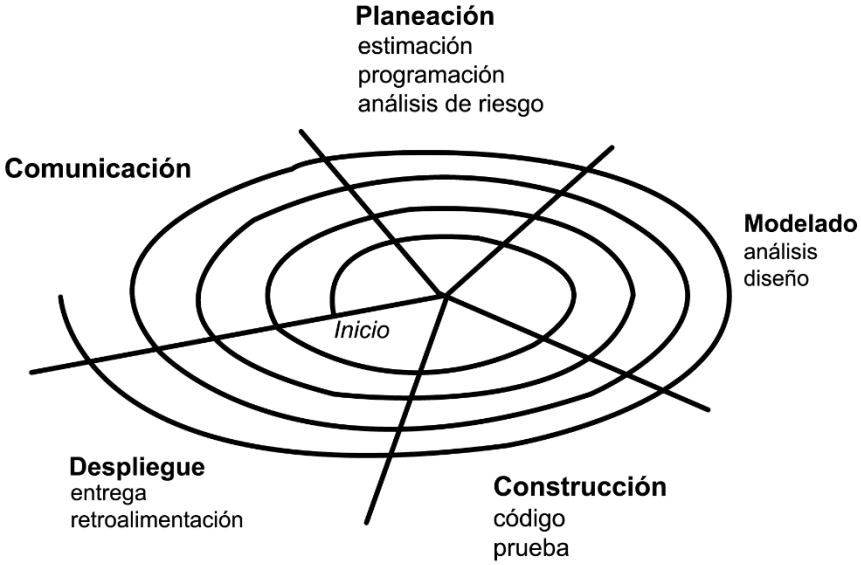

Fig. 5 Pressman R. (2010). *Modelo de espiral común.*

Como puede apreciarse tanto el modelo evolutivo como el incremental se derivan del *Flujo de proceso evolutivo*, pues de manera general inician con la comunicación de requerimientos que habrán de desarrollarse, se planean los recursos a invertir, las actividades a ejecutar y se procede al modelado o diseño de las funcionalidades, se construyen y se despliegan, entrando nuevamente a un ciclo en el que el cliente comunica las nuevas necesidades a atender, hasta que después de varios ciclos se obtiene la versión final del software solicitado.

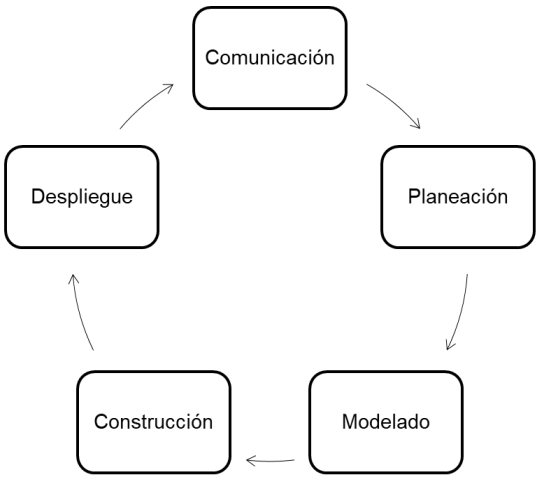

Fig. 6 Pressman R. (2010). *Flujo de proceso evolutivo*.

1.4. Comparativo de modelos

La selección de un modelo de desarrollo adecuado es decisiva para el éxito de un proyecto y esto dependerá de las particularidades de éste. En la siguiente tabla se muestran las ventajas y desventajas de los modelos que se explicaron con anterioridad.

Modelo	Ventajas	Desventajas
Cascada	● La gestión del proyecto es clara desde el inicio, pues se detallan todas las fases y actividades que se realizarán en el proyecto. ● Los requerimientos son definidos desde el inicio.	● El usuario será beneficiado hasta el final del proyecto. ● El software se ve limitado en cuanto a las funcionalidades definidas al inicio.
Incremental	● El cliente recibe características funcionales en un periodo corto de tiempo.	● El equipo de trabajo debe conocer de manera profunda el negocio o proyecto.

	● Se pueden integrar nuevos requerimientos durante el avance. ● Se reduce el tiempo de planeación.	
Evolutivo	● Funciona muy bien para el desarrollo de proyectos de innovación. ● Reduce riesgos.	● Está enfocado solo en la creación de prototipos.

Tabla 1. *Ventajas y desventajas de los modelos de desarrollo de software*.
Elaboración Propia

Los modelos de desarrollo de software abordados en este capítulo son los propuestos por Roger Pressman, reconocido ingeniero de software y de procesos a escala mundial. Estos modelos y las metodologías derivadas, son considerados como 'tradicionales', ya que son las primeras propuestas que se implementaron en la industria para poner orden a los procesos de desarrollo de sistemas, se basan en la planeación y documentación extensiva y han tenido un gran éxito cuando se trata de proyectos largos y predecibles; sin embargo, en palabras de Alistair Cockburn (2002), estos modelos "olvidan las flaquezas de las personas cuando construyen software" (Pressman, 2010, p. 56), los desarrolladores tienen estilos de trabajo distinto y los modelos tradicionales exigen disciplina en su ejecución y esto podría ser una de las grandes debilidades a las que se enfrentan.

1.5. Metodologías ágiles

Hemos señalado que los modelos tradicionales dejaron en segundo plano la importancia que tiene el equipo, los proyectos son realizados por personas, mismas que se ven influenciadas por infinidad de factores externos, lo cual puede tornarse en una debilidad al momento de incorporar procesos que necesitan un alto nivel de disciplina, como requieren estos modelos; por ello la agilidad surge como una nueva

propuesta para solventar esta necesidad, en la que se trata de reincorporar el factor humano y su interacción para el cumplimiento de objetivos, para así adaptarse a entornos y proyectos innovadores, aunque no hay que olvidar que estas propuestas ágiles siguen formando parte de la Ingeniería de Software, como lo explica Pressman: “La ingeniería de software ágil combina una filosofía con un conjunto de lineamientos de desarrollo. La filosofía pone el énfasis en: la satisfacción del cliente y en la entrega rápida de software incremental, los equipos pequeños y muy motivados para efectuar el proyecto, los métodos informales, los productos del trabajo con mínima ingeniería de software y la sencillez general en el desarrollo” (Pressman, 2010, p. 55).

Las metodologías ágiles más comunes son *Extreme programming (XP), Desarrollo adaptativo de software (DAS), Método de desarrollo de sistemas dinámicos (MDSD)*, por supuesto *Scrum,* entre otras*.*

A continuación, se muestra una comparativa entre ambos estilos de metodologías, propuesta por Canós Joseph (2005). Se puede apreciar cómo las metodologías ágiles usan como principal herramienta la comunicación y la adaptación, contrario a las tradicionales que se centra en el control exhaustivo de los procesos.

Metodologías Ágiles	**Metodologías Tradicionales**
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.

El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 2. Canos J. (2005). *Diferencias entre metodologías ágiles y tradicionales*. Tomado de: (Gómez, 2010, p. 72)

La elección de una metodología de desarrollo apropiada permitirá que el equipo trabaje de manera eficiente, pues esta brinda la perspectiva desde la cual se abordará el proyecto, las técnicas y herramientas que se usarán, generando así claridad y certeza para el equipo. Esta elección debe hacerse con base en las necesidades de cada proyecto. Es contraproducente usar una misma metodología para todos los proyectos ya que ningún proyecto es igual a otro.

Capítulo 2. La industria de desarrollo de software

El uso de las Tecnologías de la información (TI) ha generado grandes cambios en la vida de las personas, no sólo en lo social y cultural, sino educativo, económico e incluso industrial. La adopción de la tecnología nos ha llevado a un cambio de paradigmas y a competir por su dominio y producción, “este nuevo ciclo industrial se vincula con la llamada nueva economía o economía del conocimiento reflejada en el gran crecimiento del sector (informático) durante la década de los noventa” (Mochi, 2006, p. 82); esta economía del conocimiento establece sus bases en el uso de la información y es por ello por lo que la industria del software ha cobrado relevancia.

Para dimensionar la complejidad de la formación de un ecosistema empresarial en el sector del desarrollo de software, la doctora Dora González (2007), menciona que hay que tomar en consideración dos clasificaciones:

Clasificación de Mercado	Producto	Descripción
Forma de Entrega	Software empaquetado	Responde a especificaciones de uso extendido en una industria o actividad.
	Software a medida	Nuevo producto o modificación para que responda a especificaciones particulares de un cliente.
Área de utilización o fin	Empresarial	Orientado a resolver funciones de negocio de mayor complejidad.
	Otros	Específicos al giro industrial dónde se aplique

Fig. 7 González D. (2007). *Clasificación del mercado de software*. Elaboración propia.

Como bien explica la Dra. González los productos dirigidos al sector masivo tienen como factor clave el número de licencias vendidas y existe poca interacción directa entre proveedores y clientes, en cambio los productos de soluciones empresariales requieren algún grado de adaptación a las necesidades de los clientes, lo que trae una interacción importante entre proveedores y usuarios. Ambos sectores presentan un gran reto para la industria de software a nivel global.

Las empresas que buscan permanecer en el mercado implementan sistemas de información que les permitan, en primera instancia: cumplir con sus obligaciones tributarias (como en el caso de México), regulatorias, conocer el estado de sus finanzas y automatizar sus procesos administrativos. En un segundo momento y, dependiendo de la madurez de la organización, se trabaja por el análisis de la información para la creación de estrategias que impulsen la atracción de clientes, mejora de productos, incrementar las ventas, generar una nueva línea de negocios, y más. Las posibilidades que brinda el uso de la información son muy amplias.

2.1. El mercado de software a escala internacional

En una economía globalizada y que usa las TI como su principal herramienta, resulta imprescindible sumarse a la producción de éstas. “Los mayores productores y exportadores de software se concentran principalmente en los Estados Unidos, la India, Alemania, Japón, el Reino Unido y Francia. Los mismos dominan ciertos sectores de la oferta de software, sobre todo los segmentos de mayor tamaño y mayor uniformidad de requerimientos funcionales. Además, en los Estados Unidos, Alemania y Japón se encuentran las mayores 20 empresas del mundo” (González, 2007, p. 120).

De acuerdo con el informe de Price Waterhouse Coopers, *Global 100 Software Leaders* (PwC, 2014), el mercado de software reportaba para el año 2014 cerca de US $385 mil millones en ingresos de forma anual. El mercado de TI se encuentra concentrado en Norteamérica debido a su grado de industrialización; sin embargo, la incertidumbre y desaceleración económica a escala mundial, pone en jaque a los líderes de la industria, pues las empresas de capital privado buscan nuevas opciones para la inversión de capital y las nuevas empresas representan un camino

viable. Esto da una oportunidad a las economías latinoamericanas emergentes para reducir la brecha económica y de desarrollo industrial por medio de *startups* de tecnología.

2.2. El mercado del software en México

Nuestro país ha adoptado el uso de las TI de diversas formas, desde su implementación para la automatización de procesos complejos de manufactura, hasta el uso de recursos compartidos en la nube y colaboración en línea de pequeñas empresas. “Estudios llevados a cabo en Argentina, Chile, Colombia y México concluyen que existe un crecimiento inclusivo debido al impacto de la adopción de TIC, que redunda en una mayor productividad y a los efectos positivos del aumento de la producción sobre los trabajadores poco especializados.” (Dutz M.A., 2018, p. 3); podemos afirmar que con el uso adecuado de la tecnología es posible incrementar la productividad y con ello impactar positivamente la economía mexicana.

Hacia el 2005 el mercado de TI representaba poco más del 1% del (Producto Interno Bruto) PIB mexicano. El valor de mercado de la industria de TI ha crecido a una tasa anual promedio del 14% en 10 años (de 2002 a 2012), “mientras que las exportaciones se han incrementado 12.2% y el empleo 11%” (PROSOFT, 2013. p. 18). Pese a estas cifras el mercado de producción de software sigue cautivo para el consumo nacional, esto representa una gran área de oportunidad y sin duda alguna retos que superar.

Capítulo 3. Scrum

La producción de software ha tenido que adaptarse a los constantes cambios tecnológicos como: la mejora constante de hardware, la aparición de diversos lenguajes de programación, nuevas tendencias, y más; así como la búsqueda de nuevos métodos para la administración de proyectos.

En los años 70's cuando la industria de TI empezó a expandirse, surgió también la necesidad de gestionar proyectos de desarrollo de software. El primer esfuerzo para proporcionar estructura a las actividades involucradas fue con la implementación de métodos de administración pertenecientes a otras industrias (por ejemplo, la de la construcción) y fueron estas las que ayudaron a dar orden a los procesos a través del control documental, trabajando linealmente y por fases (metodologías tradicionales); sin embargo, esto no resultaba del todo útil en proyectos que lidiaban con requerimientos de alta incertidumbre e innovación; así que varios ingenieros y desarrolladores de software empezaron a buscar nuevas formas de gestión que permitiera adaptarse al cambio de manera rápida.

En el punto 1.5 de este trabajo se explicó cómo es que la agilidad surge como una nueva propuesta para la gestión tradicional de proyectos de desarrollo de software. Autores tan reconocidos como Pressman las han llamado Metodologías ágiles de desarrollo; sin embargo, al proponer al equipo de trabajo, sus interacciones, experiencia, conocimiento y responsabilidad compartida, como principal componente de esta corriente, pudiera resultar contraproducente (dependiendo del contexto donde se aplique) el llamarlo *metodología*, pues se espera que el equipo decida la mejor forma de alcanzar su objetivo, que elija las técnicas, herramientas, procesos que usará para ello; por ello otros autores como Guerrero (2014) lo denominan un *Marco de Trabajo. "*Los marcos de trabajo contienen patrones y buenas prácticas que apoyan el desarrollo de un producto y un proceso con calidad" (Guerrero, Gutiérrez, & Londoño, 2014).

Aunque son conceptos distintos, tanto *metodología* y *marco de trabajo* se refieren al conjunto de técnicas, herramientas y buenas prácticas que apoyan a un equipo de trabajo en la consecución del objetivo de un proyecto determinado.

3.1. Manifiesto ágil

Derivado de las distintas corrientes que estaban surgiendo para gestionar los proyectos de alta incertidumbre, en 2001 se reunieron: Alistair Cockburn, Jim Highsmith, Ken Schwaber, Jeff Sutherland, entre otros ingenieros de software (que tenían más de una década usando técnicas distintas a las tradicionales) para generar algunos lineamientos que permitieran trabajar de una forma más amigable con este tipo de proyectos y, sobre todo, centrarse en la satisfacción del cliente a través de la entrega de un producto de valor para su negocio; es así como surge el *Manifiesto ágil,* en el cual declaran cuatro valores fundamentales:

> "Individuos e interacciones sobre procesos y herramientas
> Software funcionando sobre documentación extensiva
> Colaboración con el cliente sobre negociación contractual
> Respuesta ante el cambio sobre seguir un plan" (Beck, y otros, 2001).

El Manifiesto Ágil (2001) pone por escrito los lineamientos a seguir para el desarrollo ágil de software tomando como base los cuatro valores mencionados; además, se detallaron los principios que dan soporte a los mismos:

> "Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
>
> Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
>
> Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
>
> Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
>
> Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

> El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
>
> El software funcionando es la medida principal de progreso.
>
> Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
>
> La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
>
> La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
>
> Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
>
> A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.” (Beck, y otros, 2001).

Los principios del manifiesto describen los comportamientos que se esperan de un equipo ágil. Como se puede apreciar, se enfocan en la interacción de persona a persona, en el involucramiento del cliente durante el desarrollo y en la entrega temprana de valor. El manifiesto ágil podría ser considerado como el ADN que debe tener cualquier metodología o marco de trabajo para que se considere ágil.

3.2 El origen de Scrum

Ken Schwaber (1997) explica que las bases de este marco de trabajo surgen en 1986, fueron Hirotaka Takeuchi e Ikujiro Nonaka quienes escribieron el articulo *The New New Product Development Game* sobre los procesos de producción de diversas empresas de Japón y los Estados Unidos, quienes lanzaban constantemente al mercado productos novedosos (como Honda, Canon, Epson, NEC, Hewlett-Packard, Fuji- Xerox, 3M, entre otros) y detectaron ciertos patrones de ejecución como: equipos pequeños multidisciplinarios y colaborativos, enfocados en lograr un solo objetivo a la vez y no en cadena de fases secuenciadas; a esta forma de trabajo la compararon con la formación característica de rugby: *Scrum*.

En 1995 Ken Schwaber y Jeff Sutherland formalizan Scrum al compartir los roles, reglas y esencia como marco de trabajo para el desarrollo de software en la conferencia OOPSLA. (Sutherland, 2020).

Fig. 8 *Evolución de Scrum*. Elaboración propia.

Después de que Scrum fue lanzado formalmente como un marco de trabajo para el desarrollo de software, ha ido cambiando y adaptándose a las necesidades de la industria. Sus autores redactaron *La Guía de Scrum,* en la que describen las reglas básicas a seguir para su adopción, misma que ha sido actualizada en varias ocasiones. Cabe señalar que muchas de las herramientas y técnicas usadas de manera regular en Scrum, han sido detalladas por empresas certificadoras en la materia y por otros autores.

3.3 El uso de *Scrum* en la industria del desarrollo de software

Scrum fue la metodología de software más usada en 2019 según el reporte *State of Software Development*, publicado por la compañía Coding Sans; este dato es también respaldado por el *14th Annual State of Agile Report,* en el que 75% de los 40 mil encuestados practican Scrum o algún híbrido que lo incluye.

Entre los principales retos a los que se enfrenta la industria de desarrollo, detectados en el *State of Software Development* se encuentran:

- Capacidad: Entregar software funcionando, mientras el *backlog* se encuentra lleno y la capacidad de desarrollo es limitada, se encuentra en el primer lugar de desafíos con el 21.29%.

- Compartir conocimiento: Se posiciona en el segundo lugar con el 20%, cuando el *backlog* está lleno y la fecha límite se acerca, el intercambiar conocimiento es afectado directamente.
- Contratación de talento: es necesario integrar personas al equipo para aumentar la capacidad de desarrollo y, para ello, también es necesario compartir el conocimiento; este desafío se encuentra en tercer lugar con el 15.83%.

Por otra parte, en el *14th Annual State of Agile Report* se detectó que los desafíos más altos para adoptar y escalar Agile están relacionados con la cultura organizacional:

- Resistencia organizacional al cambio
- Insuficiente participación de liderazgo
- Procesos y prácticas inconsistentes entre equipos
- Cultura organizacional en desacuerdo con los valores
- Apoyo y patrocinio inadecuado de la administración de las organizaciones

Los beneficios que aporta la adopción de la agilidad en las organizaciones como: la habilidad de gestionar el cambio de prioridades, visibilidad del proyecto, alineación de TI al negocio, entrega temprana al mercado, incremento de la productividad, etcétera, hacen que los retos relacionados no se vean como un obstáculo sino como una necesidad prioritaria a atender por la industria.

Scrum es uno de los marcos de trabajo más usados para desarrollar software. Profesionistas que ya lo utilizan declaran que mejora la calidad de la vida laboral, que la entrega de valor al cliente es el resultado más valorado por los ejecutivos y que seguirán usando Scrum en el futuro (Scrum Alliance, 2019).

3.4 La esencia de Scrum

Sus autores Jeff Sutherland y Ken Schwaber definen Scrum como “Un marco de trabajo por el cual las personas pueden abordar problemas complejos adaptativos, a la vez que entregan productos del máximo valor posible productiva y creativamente”. (Schwaber K., 2017)

En su definición podemos observar que su uso no está limitado exclusivamente para el desarrollo de software, sino que Scrum fue desarrollado para la gestión y desarrollo de productos y ha demostrado su utilidad para proyectos que se ven sometidos a altos niveles de incertidumbre, esto gracias a la transferencia iterativa de conocimiento que promueve a través del control de procesos empírico y sus tres pilares.

3.4.1 Control de procesos empírico

Toma como base la teoría del empirismo, la cual asegura que el conocimiento procede de la experiencia y la observación, lo que permite una toma de decisiones más apegada a la realidad. Los tres pilares que sugieren como parte del control de procesos empírico y que son usados por Scrum son:

Transparencia	Los procesos deben ser visibles para todos los involucrados. Los observadores comparten un entendimiento común de los aspectos del proceso, es decir usan una definición y lenguaje común.
Inspección	Los usuarios de Scrum deben revisar los artefactos y el progreso de manera constante, pero sin interferir en el trabajo diario, principalmente para detectar desviaciones o riesgos que se puedan materializar e impactar negativamente el proyecto.
Adaptación	Si en una inspección se detecta un desvío que pudiera provocar el rechazo del producto, el proceso deberá ajustarse para minimizar el impacto.

Tabla 3. Schwaber y Sutherland. (2017). *Pilares del control de procesos empírico.* Elaboración propia.

Estos tres pilares ayudan a que el equipo alinee su actuar y los procesos que ejecutan al trabajar con Scrum, algunos beneficios adicionales relacionados al proceso de control empírico son los siguientes:

- Centrado en el cliente. Un beneficio clave que aporta Scrum es que es un marco de trabajo centrado en el cliente pues se enfoca en el valor que aporta al negocio y en la colaboración y retroalimentación continua de los interesados.
- Entrega de valor continua. Scrum permite la entrega temprana y continua de valor al final de cada iteración, con un incremento al producto potencialmente liberable.
- Retroalimentación oportuna. Tanto el cliente como el equipo de desarrollo se benefician de la retroalimentación que es proporcionada en el *Sprint Review*.
- Alineación de la estrategia de TI con el negocio. Al liberar más rápido el producto al mercado, se puede obtener información valiosa para el negocio e implementar estos cambios al desarrollo del producto.
- Reducción de riesgos. El desarrollo del producto está siendo inspeccionado de manera constante en cada iteración, lo que permite visualizar riesgos y actuar ante ellos de manera anticipada.

Los eventos que se realizan en cada iteración buscan impulsar principalmente la inspección y adaptación de los procesos que se ejecutan, a través la conversación cara a cara entre los miembros del equipo y usando artefactos que permitan visualizar el progreso y trabajo pendiente.

Schwaber y Sutherland en *La Guía de Scrum* no detallan formalmente algunas de las herramientas comúnmente utilizadas por Scrum, pues cada proyecto debe usar los artefactos que satisfagan mejor sus necesidades, siempre alineándose a los pilares del proceso de control empírico en el que basaron este marco de trabajo.

Otro aspecto que consideran importante es que el equipo debe incorporar a su forma de trabajo los valores: coraje, foco, compromiso, apertura y respeto para maximizar el desarrollo personal de los individuos, la confianza entre ellos y con esto usar Scrum de manera exitosa.

3.5 Roles

Un equipo Scrum está formado por tres roles principales: un *Scrum Master*, un *Product Owner* (Dueño de Producto) y el *Scrum Team* (Equipo Scrum), este equipo debe ser pequeño para permitir la flexibilidad y comunicación entre sus integrantes, pero lo suficientemente grande para completar una cantidad de trabajo significativa. Se espera que sea un equipo multifuncional para que pueda desarrollar el producto en su totalidad sin depender de personas externas al equipo; de esta forma se fomenta la autoorganización pues es el mismo equipo quien elige cómo realizar su trabajo.

Estos roles son mencionados en *La Guía de Scrum* y deben existir obligatoriamente para la ejecución de un proyecto; también son llamados en su conjunto como *Equipo principal Scrum*, sus principales responsabilidades son las siguientes:

Rol	Función	Características
Product Owner (Dueño de Producto)	Es quien representa a los interesados del negocio y el responsable de maximizar el valor del producto desarrollado.	- Único responsable de gestionar el *Product Backlog* - Da a conocer al *Scrum Team* los requerimientos del producto - Es el único que puede cancelar un *sprint*
Scrum Master	Es un líder facilitador que garantiza que el *Scrum Team* tenga un ambiente laboral adecuado para completar con éxito el desarrollo del producto.	Da servicio a - Al *Product Owner* - Al *Scrum Team* - A la organización misma Apoyando a todos a entender la teoría, prácticas y valores de *Scrum*.

Scrum Team *(Equipo Scrum)*	Son los responsables de entender el negocio y trabajar para la entrega de un incremento al producto al final de cada *sprint*.	- Autoorganizado, responsable y multifuncional - Su tamaño es pequeño (más de tres, pero menos de nueve personas)

Tabla 4. Roles Scrum. Elaboración propia.

Existen roles que no son obligatorios para el desarrollo de un proyecto con Scrum, los cuales de manera general llamaremos *stakeholders* pues son las personas interesadas en el éxito del proyecto, que pueden ser impactadas por el mismo, pero que no trabajan directamente en la creación del producto, pudiéndose tratar de clientes, usuarios o patrocinadores del proyecto.

3.6 Artefactos

"Los artefactos de Scrum representan trabajo o valor en diversas formas que son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación" (Schwaber y Sutherland, 2017, p. 15). Es importante que todos los miembros del equipo tengan el mismo entendimiento de estos artefactos para promover la visibilidad y la responsabilidad de cada miembro sobre las actividades que se están realizando.

3.6.1. Product Backlog

El *Product Backlog* o *Lista de Producto*, es la lista priorizada de funcionalidades o requerimientos en un alto nivel que se han identificado que formarán parte del producto a desarrollar durante el proyecto. El *Product Owner* es el único responsable de este artefacto, es quien prioriza e incluye nuevos requerimientos.

El *Product Backlog* es un elemento vivo pues nunca se encuentra completo, sino que evoluciona cuando el producto y el ambiente lo hacen, además de que a través de cada iteración se le va proporcionando más detalle, pues como se mencionó está formado por requerimientos a alto nivel que pueden desglosarse aún más a lo largo

del proyecto. Solo existe uno por proyecto aun cuando más de un equipo trabaje en el desarrollo.

Refinamiento

Los elementos del *Product Backlog* se someten continuamente a refinamiento para añadir orden, detalle y nuevos requerimientos. El refinamiento lo realiza el *Product Owner*, quien prioriza los elementos de acuerdo con las necesidades del negocio mismas que descubre a través de la interacción con los *stakeholders* y las expectativas que tienen sobre el proyecto.

También el trabajo de refinamiento podría realizarlo en conjunto con el equipo de desarrollo, en este caso el proceso no debe consumir más del 10% de la capacidad del equipo por *sprint*.

Definición de “Terminado”

El *Scrum Team* debe tener claro y compartir los criterios para determinar cuando el trabajo se considera completado; a esta ‘definición de terminado’, también se le conoce como como ‘criterios de terminado’, y es utilizada para evaluar el trabajo terminado al final del sprint y se enfoca a los aspectos que proporcionan calidad a todas las funcionalidades desarrolladas.

Los criterios de terminado varían en cada proyecto, en ocasiones esta definición está determinada por la organización y los procesos que siga para el desarrollo y liberación de un incremento; de no estar definida por la organización, el *Equipo principal Scrum* es el encargado de determinar los criterios para dar un elemento por completado y así asegurar la calidad del producto desarrollado.

Scrum es un marco de trabajo que sugiere prácticas para el desarrollo de software enfocándose en el trabajo del equipo para crear funcionalidades, es responsabilidad de cada organización incorporar las normas o procesos necesarios para cuidar aspectos como la seguridad de la información o el aseguramiento de la calidad, el *Manifiesto ágil* (2001) menciona “La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad” refiriéndose a que la incorporación de los aspectos mencionados debe realizarse para desarrollar software de calidad, los criterios de terminado ayudan a incorporarlos como parte de las reglas del juego para el equipo,

es decir un incremento no se considerará terminado si no cumple con ciertos criterios, por ejemplo: pruebas de penetración y de rendimiento exitosas. Se considera un incremento a “la suma de todos los elementos del *Product Backlog* completados durante un Sprint y el valor de los incrementos de todos los *Sprints* anteriores” (Schwaber y Sutherland, 2017, p. 17); este resultado debe ser inspeccionable y cumplir la definición de terminado y potencialmente estar en condiciones de ser utilizado.

3.6.2. Sprint Backlog

El *Sprint Backlog* o “Lista de pendientes del sprint es el conjunto de elementos del *Product Backlog* seleccionados para el *Sprint*, más un plan para entregar el incremento del producto y conseguir el *Objetivo del Sprint*” (Schwaber & Sutherland, 2017, p. 16), es decir, el *Scrum Team* planea las tareas y estima el trabajo necesario para entregar una funcionalidad o incremento al producto.

El *Sprint Backlog* promueve la visibilidad del trabajo que deberá realizar el equipo para conseguir el *Objetivo del sprint*. Este artefacto debe ser lo suficientemente claro y detallado para iniciar la iteración, aunque seguirá evolucionando conforme se detecta trabajo por realizar en el sprint; el *Scrum Team* es el único que puede cambiar el *Sprint Backlog*, siempre teniendo en consideración que el detectar tareas o actividades nuevas e incluirlas en los pendientes no debe impactar el *Objetivo del sprint.*

Al *Sprint Backlog* hay que darle seguimiento y transparentar el estado de cada una de las actividades, por ejemplo, con el uso de un *Scrum Board*. En todo momento del *sprint*, cualquier miembro del equipo debe ser capaz de visualizar la cantidad de trabajo restante.

3.7 Eventos

Uno de los principales diferenciadores que tienen las metodologías ágiles con respecto a las tradicionales, es fomentar la conversación cara a cara, para generar confianza dentro del equipo. Scrum no es la excepción, se enfoca en la interacción de todos los involucrados en el desarrollo de un proyecto usando diversos eventos o ceremonias que se describen a continuación:

Evento	Descripción
Sprint	Es un bloque de tiempo (de un mes o menos) durante el cual se crea un incremento de producto utilizable
Sprint Planning *(Planeación del sprint)*	Reunión en la cual se eligen los elementos del *Product Backlog* a trabajar en el sprint y las tareas relacionadas para alcanzar el *Objetivo del sprint (Sprint goal).*
Daily Scrum *(Scrum diario)*	Reunión diaria (breve) que sirve al *Scrum Team para* evaluar su progreso hacia el *Objetivo del sprint*.
Sprint Review *(Revisión del sprint)*	Se lleva a cabo al final del sprint para inspeccionar el incremento al producto, recibir retroalimentación de los *stakeholders* y adaptar el *Product Backlog* de ser necesario.
Retrospective Sprint *(Retrospectiva del sprint)*	Esta reunión se realiza después del *Sprint Review*, en ella el equipo discute sobre el aprendizaje obtenido durante el sprint anterior y las mejoras a incorporar en su forma de trabajo para la siguiente iteración.

Tabla 5. *Eventos Scrum*. Elaboración propia.

En la tabla anterior sólo se definió el concepto de cada uno de los elementos o reuniones, ya que el detalle de cada una de ellas se revisará a lo largo de este trabajo.

3.7.1 Time-box

Cada evento que se lleva a cabo en Scrum tiene que apegarse a un *Time-box,* es decir “toda reunión o evento debe tener un máximo temporal conocido antes de

empezar y llegado ese tiempo máximo, la reunión se termina" (Garzás, 2017, p. 188), en pocas palabras el *Time-box* es la duración máxima sugerida para cada evento.

Evento	Duración	Objetivo
Sprint	1-4 semanas	Desarrollar una funcionalidad útil del producto.
Sprint Planning	8 horas	Determinar el *Objetivo del Sprint* (incremento) y cómo se logrará.
Daily Scrum	15 minutos	Identificar impedimentos relacionados al trabajo que está por realizarse en el *sprint*.
Sprint Review	4 horas	Validar que el *Objetivo del Sprint* se consiguió inspeccionando el incremento del producto.
Sprint Retrospective	3 horas	Identificar qué salió bien y qué se puede mejorar con respecto a los procesos y herramientas utilizadas en el sprint que acaba de finalizar.

Tabla 6. Schwaber y Sutherland (2017). *Bloques de tiempo de los eventos de*
Elaboración propia.

La duración del *sprint* depende de cuál sea el incremento del producto que se incluye como objetivo de éste y será determinado por el *Scrum Team* a través de la estimación, el tiempo máximo de la duración del sprint es de cuatro semanas, aunque se da preferencia a intervalos de tiempo más cortos para que la incertidumbre se reduzca. En el caso del *Sprint Planning*, *Sprint Review* y *Sprint Retrospective* el *time-box* sugerido corresponde a un *sprint* de cuatro semanas, en caso de que sea un *sprint* más pequeño estos tiempos se reducirán y viceversa.

Vale la pena mencionar que, aunque cada evento tiene un tiempo estipulado, si se consigue antes el objetivo de cada uno de ellos puede darse por concluida dicha reunión.

3.8 Interacciones y responsabilidades

Cada rol tiene responsabilidades asignadas en cada uno de los eventos. Estas ayudan a conseguir el objetivo de cada ceremonia y a guiar el flujo de procesos en Scrum. La esencia de Scrum es el sprint; sin embargo, el éxito de cada iteración está relacionado directamente con el nivel de madurez que tiene el equipo y ésta sólo se logra a través de las interacciones que tiene el equipo, la mejora de estas y el conocimiento que incorporan a través de la experiencia adquirida.

3.8.1 Sprint Planing

La *Sprint Planning* busca establecer el objetivo que perseguirá el equipo y el cómo llegará ahí; se busca bosquejar las actividades a realizar y que el equipo se autoorganice y responsabilice de las mismas.

Esta reunión busca responder las siguientes preguntas "¿Qué puede entregarse en el Incremento resultante del *Sprint* que comienza?, ¿cómo se conseguirá hacer el trabajo necesario para entregar el Incremento?" (Schwaber y Sutherland, 2017, p. 10).

Rol	Responsabilidad
Scrum Master	- Asegura que se lleve a cabo la reunión. - Verifica que la reunión se realice en el time-box adecuado y que se logre el objetivo de esta.
Product Owner	- Discute el objetivo del sprint y los elementos del *Product Backlog* a atender.
Scrum Team	- El equipo completo participa en el entendimiento del trabajo a realizar. - Compromete los elementos del *Product Backlog* que serán atendidos en el *sprint*.

Tabla 7. *Responsabilidades en la Sprint Planning*. Elaboración propia.

Ken Schwaber y Jeff Sutherland explican que existen diversas estrategias para abordar los procesos que plantea Scrum, y algunas de esas técnicas se enfocan en

la proyección y estimación del trabajo por hacer, como la estimación basada en puntos de historia de usuario que será explicada más adelante.

3.8.2 Daily Scrum

En esta reunión diaria se busca transparentar el avance del trabajo y principalmente identificar los impedimentos que pueden materializarse en el sprint. El *Scrum Team* debe de asistir de manera obligatoria; el *Scrum Master* como en las demás ceremonias debe cuidar la forma en la que se realiza; la presencia del *Product Owner* no es obligatoria, si llega a asistir debe hacerlo de forma pasiva.

Rol	Responsabilidad
Scrum Master	- Asegura que se lleve a cabo la reunión. - Verifica que la ceremonia se realice en el time-box adecuado y que se logre el objetivo del evento.
Scrum Team	- El *Scrum Team* dirige la reunión para evaluar el progreso del trabajo. - Responden a tres preguntas: o ¿Qué hice ayer? o ¿Qué haré hoy? o ¿A qué impedimentos me enfrento?

Tabla 8. *Responsabilidades Daily Scrum*. Elaboración propia.

Este evento existe para inspeccionar el trabajo avanzado y evaluar qué tendencia se sigue hacia el final del *sprint.* Esto se logra respondiendo las preguntas propuestas, mismas que están enfocadas sólo al objetivo del *sprint* en curso. Dura 15 minutos para mantenerla simple y evitar el desgaste de los asistentes, a los impedimentos detectados se les da seguimiento de manera posterior.

3.8.3 Sprint Review

En la *Sprint Review* el *Scrum Team* inspecciona el incremento junto con el *Product Owner* y los *stakeholders* que pudiera haber invitado; por ejemplo, un usuario final del producto o incluso un patrocinador.

Rol	Responsabilidad
Scrum Master	- Asegura que se lleve a cabo la reunión. - Verifica que la reunión se realice en el *time-box* adecuado y que se logre el objetivo del evento.
Product Owner	- Revisa el incremento mostrado por el equipo y decide si este se ha completado de manera satisfactoria. - El *Product Owner* explica qué elementos se 'han terminado' y el estado actual del *Product Backlog*.
Scrum Team	- El *Scrum Team* presenta el trabajo 'terminado' y responde dudas al respecto. - Habla de los problemas a los que se enfrentó y cómo se solucionaron (todo esto relacionado al producto).

Tabla 9. *Responsabilidades Sprint Review*. Elaboración propia.

Esta reunión se lleva acabo al final del *sprint* y se revisa el trabajo que ha realizado el equipo y se enfoca en la revisión del producto, se evalúa si se acepta o rechaza tomando en consideración los criterios de terminado y de aceptación relacionados.

3.8.4 Sprint Retrospective

En la *Sprint Retrospective* se indaga sobre las mejoras que pueden implementarse en la siguiente iteración con base en lo aprendido en el último *sprint* de trabajo, ya sean técnicas, herramientas e incluso la forma de interactuar en el equipo, etcétera. Esta reunión se lleva acabo al finalizar el *sprint*, justo después de la *Sprint Review*. La participación del *Product Owner* no es obligatoria.

Rol	Responsabilidad
Scrum Master	- Asegura que se lleve a cabo la reunión, la modera buscando la productividad y un ambiente de respeto. - Verifica que la ceremonia se realice en el *time-box* adecuado y que se logre el objetivo del evento.

Scrum Team	- El equipo identifica elementos que salieron bien y posibles mejoras a incorporar en la forma de trabajo en el siguiente *sprint*.

Tabla 10. *Responsabilidades Sprint Retrospective*. Elaboración propia.

Como se observa en las tablas de responsabilidad anteriores, el *Scrum Master* realiza las mismas actividades en cada evento como: facilitar las reuniones, cuidar el *time-box* y apoyar al equipo en cualquier duda con respecto a Scrum. Otros de sus compromisos se encuentran:

- Ser un líder servicial, que apoye al equipo a eliminar los impedimentos que surjan durante el proyecto.
- Apoyar al *Scrum Team* con el entendimiento y adopción de la metodología.
- Promover los valores y principios de Scrum para lograr un ambiente productivo, disciplinado, autoorganizado y multifuncional.

Se espera que el equipo trabaje de manera colaborativa para conseguir la meta establecida en cada *sprint*, siguiendo altos estándares de calidad y guiándose con el plan flexible que se delimitó en la *Sprint Planning*.

3.8.5 Flujo de trabajo de Scrum

Los roles, eventos y artefactos descritos hasta este momento, son mencionados en *La Guía de Scrum*, elaborada por Ken Schwaber y Jeff Sutherland (creadores de Scrum). El propósito de esta guía consiste en definir los roles, eventos y artefactos de este marco de trabajo y, de manera general, las reglas que los relacionan; sin embargo, los autores también mencionan que “Las estrategias específicas para usar el marco de trabajo Scrum son diversas y están descritas en otros lugares” (Schwaber y Sutherland, 2017, p. 7). Esto da pie a que exista una variedad de técnicas y herramientas propuestas por distintas organizaciones certificadores en este marco de trabajo. Las empresas certificadoras en Scrum han propuesto técnicas y herramientas que se adecuan a las bases que han sentado los autores de Scrum y, a su vez, las empresas que desarrollan software tratan de implementarlo de acuerdo con el conocimiento previo que tenga el personal que lidera el cambio.

SCRUMstudy es un organismo de acreditación global para certificaciones de Scrum y creó su propia guía para administrar proyectos usando Scrum llamada *Guía SBOK*, misma que propone el siguiente flujo de trabajo:

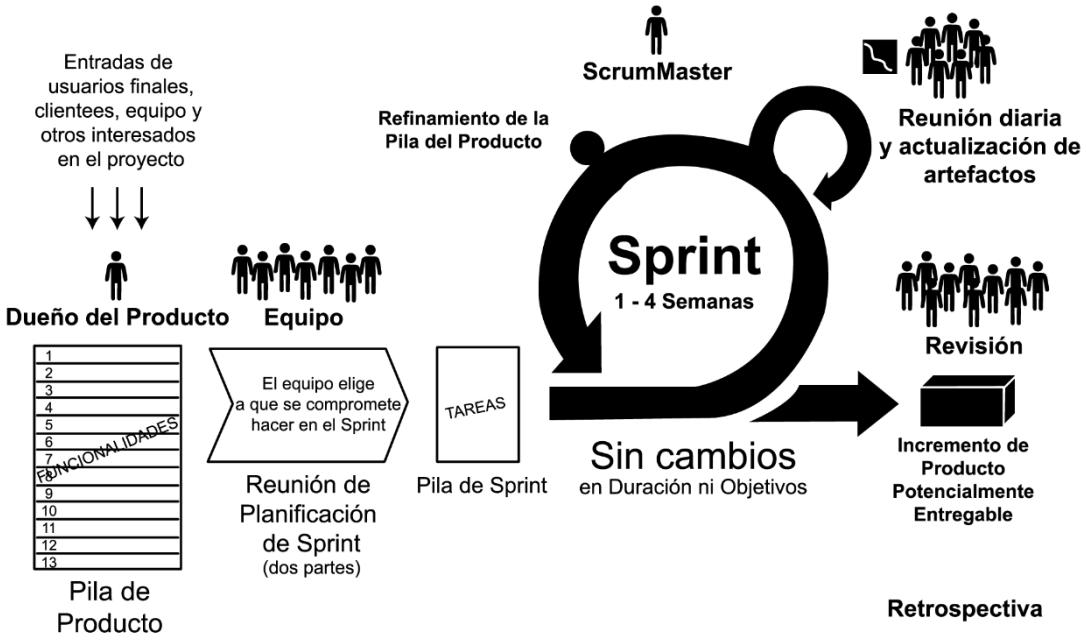

Fig. 9 SCRUMStudy. (2016). *Flujo de trabajo Scrum.*

El flujo de trabajo mostrado incluye las interacciones, roles y artefactos propuestos por Schwaber y Sutherland: se inicia con un *Product Backlog* gestionado por el *Product Owner*; el *Scrum Team* compromete las tareas a realizar en el *sprint*; inicia la iteración y se realizan las actividades de desarrollo junto con el *Daily Scrum*; al final se revisa el incremento en el *Sprint Review*, se acepta o rechaza y se tiene la *Sprint Retrospective*; todo esto se repite cada *sprint* hasta completar un proyecto. Este es el flujo que debe adoptarse al momento de usar Scrum*;* es simple, fácil de entender, pero a la vez difícil de dominar, pues como se ha descrito, se requieren muchos cambios profundos en la forma de relacionarse dentro de un equipo. El factor humano es clave para una adopción exitosa de Scrum.

3.9 Scrum como metodología de desarrollo

La *Guía SBOK* de SCRUMstudy propone el uso de Scrum dividiéndolo en fases, es por esto por lo que afirmamos que Scrum también puede ser abordado desde una perspectiva de metodología de desarrollo de software, pues pueden identificarse: entradas, procesos, herramientas y salidas formales en cada uno de sus procesos. A continuación, se mencionan las principales actividades a realizar en cada fase propuesta.

Fase	Actividades
Inicio	- Crear la visión del proyecto - Identificar Scrum *Master* y *Stakeholders* - Formar equipo *Scrum* - Desarrollar épicas - Crear Lista de Producto - Estimar liberaciones
Planeación y estimación	- Crear historias de usuario - Aprobar y estimar historias de usuario - Crear y estimar tareas - Crear Lista de Pendientes del Sprint
Ejecución	- Crear entregables - Actualizar *Scrum board* - Scrum Diario - Actualizar la Lista de Producto
Revisión y retrospectiva	- *Sprint Review* - *Sprint Retrospective*
Entrega	- Envío de entregables

	- Retrospectiva del proyecto

Tabla 11., SCRUMstudy. (2016). *Fases de un proyecto Scrum*. Elaboración propia.

Muchas de las actividades propuestas en cada proceso tienen la esencia Scrum como marco de trabajo, de hecho cada una de estos procesos se alinean a los principios del proceso de control empírico: transparencia, inspección y adaptación, pero incorporan el uso de herramientas que como mencionamos anteriormente Schwaber y Sutherland no describen en *La Guía de Scrum,* mismas que a través de la presente investigación hemos identificado que son usadas al momento de implementar Scrum para el desarrollo de software.

3.10 Herramientas Scrum

Como ya se mencionó, la *Guía SBOK* describe el uso de otras herramientas que son implementadas en un proyecto con Scrum*;* sin embargo, no son exclusivas de esta organización; otras empresas certificadoras como Scrum Alliance y Certiprof también las mencionan por lo que se vuelve importante describirlas, ya que, al ser organismos internacionales, son quienes marcan la pauta para su integración en la ejecución a escala organizacional.

3.10.1 Épicas e Historias de Usuario y Tareas

El *Product Backlog* es un conjunto de necesidades a satisfacer en el proyecto y éstas son representadas por *Épicas* e *Historias de usuario*, mismas que hay que gestionar de manera efectiva a través de su refinamiento y priorización.

“Una Historia de Usuario es la representación de un requisito de software escrito en una o dos frases, utilizando el lenguaje común del interesado” (Cohn M. , 2004). Una o más historias de usuario pueden ser seleccionadas para trabajarse en un sprint. Mike Cohn (2004) sugiere redactar Historias de Usuario bajo el siguiente formato:

Como (**rol**) Necesito (**funcionalidad**) Para (**beneficio**)	Como **vendedor** Necesito **crear remisiones** Para **dar salida a mercancía del almacén**

Fig. 10 *Estructura de historia de usuario y ejemplo*. Elaboración Propia.

Una historia de usuario puede detallarse tanto como el equipo lo requiera, y debe complementarse estableciendo *criterios de aceptación,* para así proporcionar claridad al equipo sobre lo que se espera obtener al finalizar el sprint.
Hay que aclarar que los *criterios de terminado* delimitan parámetros de calidad que hay que cumplir en todas las HU, en cambio los criterios de aceptación son aquellos que se enfocan en una sola HU y se enfoca en el desempeño técnico de cada una.

Para Lucho Salazar, agilista latinoamericano, una *épica* no es más que una historia de usuario muy grande “es una historia de usuario cuya implementación tarda más de un sprint” (Salazar, 2020). Las épicas ayudan a manejar de manera eficiente el *Product Backlog* y dimensionar las posibles entregas y *sprints* relacionados.

Cohn sugiere que las HU cumplan con las siguientes características, las cuales han sido denominadas con el acrónimo INVEST:

- **Independent** (Independiente): En medida de lo posible, las historias de usuario no dependen entre sí.
- **Negotiable** (Negociable): el detalle de cada historia de usuario será acordado por el cliente y el equipo Scrum, por lo que debe mantenerse flexible hasta el momento de entrar a un *sprint*.
- **Valuable** (Valiosa): Tiene que aportar valor al negocio o al cliente.
- **Estimable** (Estimable): Debe ser fácilmente estimable (en esfuerzo) para ayudar al cliente y al equipo a priorizarla.
- **Small** (Pequeña): Deben ocupar máximo un *sprint*, si es más grande debe dividirse.

- **Testable** (Verificable): para verificar que una historia de usuario esté terminada, debe ser probada por el usuario y el equipo. (Cohn M. , 2004)

Una HU debe ser dividida en tareas para que el equipo pueda construirla, una tarea es la representación técnica de las actividades a realizar para completar una historia de usuario. Una tarea debe ser:

- Específica
- Medible
- Alcanzable
- Relevante
- Ser temporal (tener un inicio y un fin)

Los tres elementos descritos podemos representarlos de la siguiente forma:

Fig. 11 *Representación de Épicas, Historias de usuario y Tareas*. Elaboración propia.

En la figura anterior se puede apreciar el *Product Backlog* de un proyecto, formado por dos Épicas, éstas fueron divididas en varias Historias de Usuario que a su vez han sido divididas en Tareas. La granularidad de cada uno de estos elementos será decidida por el *Scrum Team*.

3.10.1.1 Estrategias de estimación

En cualquier proyecto es importante saber cuánto tiempo se invertirá; la administración de proyectos tradicional usa una fecha de inicio y fin, controlado a través de una gráfica de Gantt para verificar el avance en días (incluso horas). Scrum aborda este reto desde la perspectiva de la estimación. Existen diversas técnicas que puede utilizar el *Scrum Team* en el momento de la planeación.

Planning póker

Esta es una de las técnicas más usadas para estimación ágil. Según resultados del *14th Annual State of Agile Report,* 60% de los participantes mencionaron usarla al momento de implementar un marco de trabajo ágil. Lo que se busca con esta herramienta es generar conversación en torno a las historias de usuario que el equipo ha de desarrollar (Digital.ai, 2020).

Cada miembro del equipo tiene un juego de cartas, cada carta está numerada con el nivel de complejidad (referente al esfuerzo) que implica una historia de usuario. Se propone una historia de usuario y cada miembro elige una carta que representa su estimación; si la mayoría selecciona la misma tarjeta, entonces la estimación indicada por esa tarjeta será asignada a esa HU. Si no hay consenso, entonces los miembros del equipo discuten hasta llegar a un acuerdo.

Usualmente las cartas del *Planning póker* usan la serie de Fibonacci (1, 2, 3, 5, 8, 13, 21, 34), también se puede incluir “?”, para en caso de que se considere que no se tiene suficiente información para hacer una estimación. Cada uno de los números en las tarjetas representa una estimación relacionada al esfuerzo complejidad y riesgo, asociada a la HU en cuestión.

Por afinidad

Los miembros del equipo anotan las historias de usuario en notas adhesivas y las colocan en una pared, ordenándolas de pequeñas a grandes, de acuerdo con la complejidad a la que la asocian. Cada miembro del equipo comienza con un conjunto de HU para colocar por tamaño relativo. Este orden inicial se realiza en silencio. Una vez que todos han colocado sus HU en la pared, el equipo revisa todas las ubicaciones y puede moverlas según corresponda, esta segunda parte implica

una discusión sobre porque la han evaluado así, qué riesgos visualizan y de esta forma llegar a acuerdos dentro del equipo.

Método Delphi

Es una técnica de estimación grupal, en la que los miembros del equipo proporcionan anónimamente estimaciones para cada HU; las estimaciones iniciales se trazan en un gráfico. El equipo discute los factores que pudieron influir en las estimaciones y proceden a una segunda ronda de votación. Este proceso se repite hasta que las estimaciones de los individuos lleguen a un consenso para la estimación final.

3.10.1.2 Puntos de Historia de usuario

A través de cualquiera de los métodos descritos se puede determinar el esfuerzo relacionado al desarrollo de cada historia de usuario, mismas que pueden representarse en *Puntos de Historia de Usuario* (PHU) o *Story Points* y "Los *Story Points* son una medida de complejidad y esfuerzo, algunos equipos optan por tomarlos como medida complejidad; pero estos también se pueden orientar para estimar el esfuerzo requerido para desarrollar un producto, incluyendo los riesgos y la incertidumbre" (Iglesias-Solano, 2011, p. 58).

Fig. 12 *Representación de puntos de Historia de usuario*. Elaboración propia.

Para usar Puntos de historia de usuario (PHU) hay que partir del consenso, es decir, todos los miembros del equipo deben elegir una historia de usuario que comprendan

el trabajo que involucra y asignarle un valor, por ejemplo 1; partiendo de este entendimiento se hará la comparación de las siguientes historias de usuario.

3.10.2 Scrum Board

Es una herramienta que asegura que el progreso del trabajo sea visible para todo el equipo, apegándose de esta forma al principio de la transparencia e incluso a la inspección. Un *Scrum Board* o *Tablero de Scrum* regularmente contiene tres columnas para indicar el progreso de las tareas estimadas para el Sprint.

Fig. 13 *Scrum Board.* Elaboración propia.

Una columna de tareas pendientes “Por Hacer” en la que van las tareas aún no iniciadas, una columna "En curso" para las tareas iniciadas pero que aún no se terminan y una columna "Terminado" para las tareas que se han completado exitosamente. Al comienzo de un sprint, todas las tareas para esa iteración se colocan en la columna "Tareas pendientes" y posteriormente se mueven en el tablero según su progreso.

No hay reglas estrictas para el uso de este tipo de tableros, simplemente todo el equipo debe conocer cómo es que lo usarán, quién será el responsable de actualizarlo, en qué momento se actualizará, etcétera. El tablero permitirá al *Scrum Team* identificar posibles atrasos o cuellos de botella y podrá actuar en consecuencia.

3.11 Métricas

Todo lo que puede ser medido puede ser mejorado, esto aplica para la industria del software y también para la agilidad. Las métricas se utilizan para evaluar la calidad

de un producto y la efectividad de los procesos de producción, Pressman nos dice al respecto que “Un ingeniero de software recolecta medidas y desarrolla métricas de modo que se obtengan indicadores. Un indicador es una métrica o combinación de métricas que proporcionan comprensión acerca del proceso de software, el proyecto de software o el producto en sí. Un indicador proporciona comprensión que permite al gerente de proyecto o a los ingenieros de software ajustar el proceso, el proyecto o el producto para hacer mejor las cosas” (Pressman, 2010, p. 527).

Una medida, una métrica y un indicador son términos relacionados, pero no son lo mismo. La medición es el resultado de la recolección de datos de varias fuentes, por ejemplo, varios componentes se revisan para saber cuántos errores hay en cada uno (medición); una métrica relaciona las medidas individuales; por ejemplo, el promedio de errores por componente y, finalmente, un indicador es el conjunto de métricas que permiten la mejora, retomando el ejemplo, un indicador sería el promedio de errores por componentes encontrados en una revisión por pares.

Principios de medición

El software como producto puede ser sometido a evaluación de sus atributos y de esta forma determinar su calidad. El desarrollo de software como proceso, también puede ser sometido a medición para determinar la eficiencia de la metodología que se está implementando, Roche (1994) sugiere un proceso básico para facilitar la medición.

1. Formulación. La identificación de medidas y métricas de software apropiadas al proyecto que se está desarrollando.
2. Recolección. Recabar los datos requeridos para derivar las métricas formuladas.
3. Análisis. El cálculo de métricas y la aplicación de herramientas matemáticas.
4. Interpretación. Evaluación de las métricas resultantes para comprender la situación actual.
5. Retroalimentación. Recomendaciones derivadas de la interpretación de las métricas del producto, transmitidas al equipo de software. (Pressman, 2010, p. 528)

Scrum tiene algunas métricas que también son conocidas como radiadores de información y que ayudan a dar seguimiento a las tareas del equipo y asegurar la transparencia, describiremos a continuación las más conocidas.

3.11.1 Burndown chart

El gráfico de *Burndown chart* presenta la cantidad de trabajo pendiente por hacer en el *sprint*. Se acompaña de una estimación previa y se da seguimiento diariamente, permitiendo detectar desviaciones en el trabajo planeado. Para su elaboración se debe contar con los puntos de historia de usuario totales de las HU a desarrollar en el *sprint*.

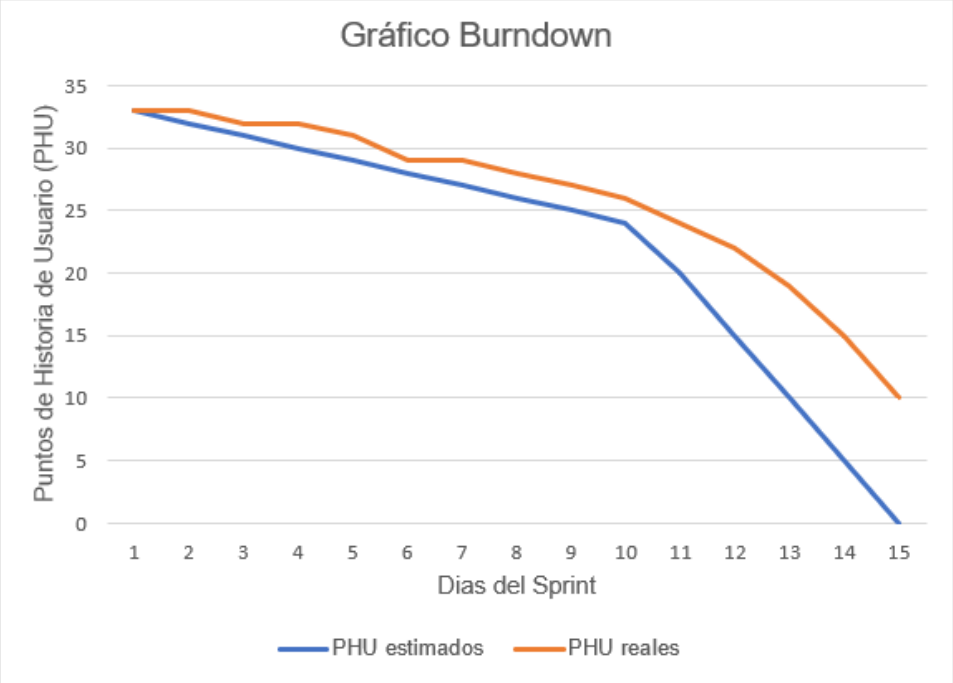

Fig. 14 *Gráfico de burndown del sprint. Elaboración propia.*

En la figura anterior se representa a un sprint de tres semanas (15 días hábiles) y los puntos de historia pendientes en el primer día del sprint son 33. La línea azul muestra el avance esperado, los PHU van disminuyendo conforme avanza el sprint hasta llegar a 0 en el día 15 que es el último. La línea naranja representa el avance real, como podemos apreciar la estimación realizada presenta una desviación con respecto a los PHU realmente completados y el equipo no alcanza el objetivo del *sprint*. En este caso se puede interpretar de diversas formas:

- La estimación de las historias de usuario no se hizo adecuadamente, es decir las HU fueron estimadas como de menos complejidad a la real.
- El equipo sobreestimó la cantidad de puntos de historia que podía realizar en el sprint.
- El equipo se enfrentó a dificultades que le impidieron realizar el trabajo comprometido.

Este gráfico sirve para dar seguimiento a la cantidad de trabajo pendiente, pero para que tenga un alto impacto en un proyecto se debe analizar profundamente con el equipo para encontrar las causas de las desviaciones y mejorar su uso en cada iteración.

3.11.2 Velocidad

Por velocidad se entiende la cantidad de trabajo que pueda completar *el Scrum Team* durante el *sprint*. Esta métrica está relacionada con el desempeño del equipo a lo largo de un proyecto, por lo que si un equipo es de reciente formación es probable que no exista información al respecto. Para su cálculo se usan los puntos historias de usuario estimadas y las completadas.

Sprint	PHU Comprometidos	PHU Realizados
1	90	75
2	80	78
3	75	73
4	80	80
5	85	83

Fig. 15 *Datos para gráfico de velocidad. Elaboración propia*.

La figura anterior muestra simplemente la comparativa entre los PHU comprometidos y realizados, esto puede ser de ayuda para visualizar la tendencia de capacidad de trabajo del equipo.

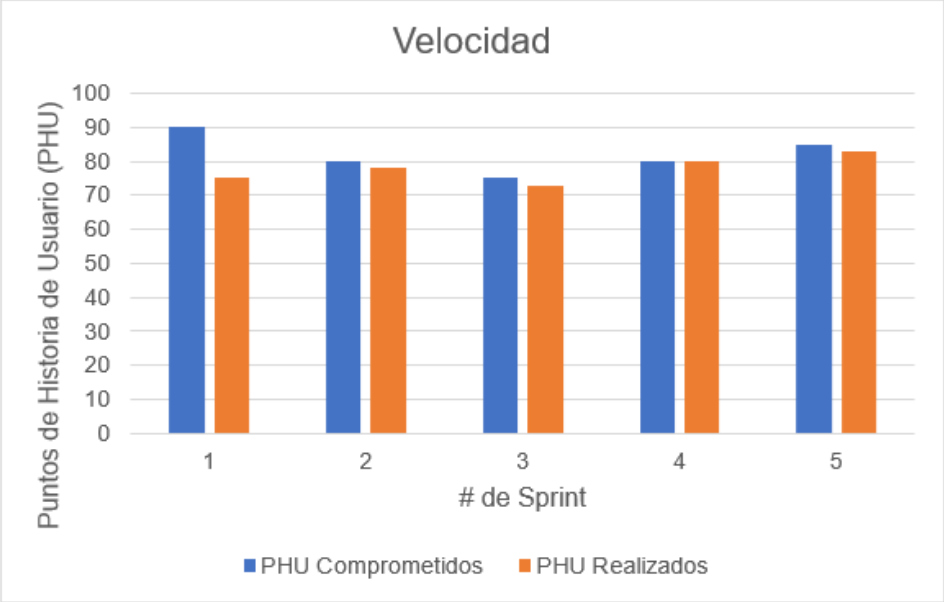

Fig. 16 *Gráfico de velocidad del equipo Scrum. Elaboración propia.*

En el ejemplo de la figura 16 se puede ver como el equipo en el primer *sprint* no completó el trabajo comprometido y esto puede deberse a varias razones; la primera pudiera ser que el equipo realizó una mala estimación en cuanto a los puntos de historia de usuario que podía completar; para el tercer *sprint* su estimación y el trabajo realizado se acercaron mucho, lo cual puede significar varias cosas: que el equipo ya aprendió a estimar, que están colaborando mejor, que el *Scrum Master* ha mejorado en la resolución de impedimentos, entre otras situaciones que pudieran ocurrir al interior del equipo. Para el cuarto y quinto *sprint* vemos que la estimación es muy parecida al trabajo realizado y que incluso han incrementado los puntos de historia completados, lo que implica un incremento en la velocidad de desarrollo del equipo.

Mike Cohn en *Succeeding with Agile* explica que es posible realizar un cálculo más detallado de la velocidad que permita estimar de manera más eficiente, a continuación, un ejemplo:

Tenemos las siguientes historias de usuario:

HISTORIA	PHU	HISTORIA	PHU
H1	13	H4	5

H2	8	H5	21
H3	5	H6	3

El equipo comprometió las primeras cinco HU para el primer sprint de 15 días, pero solo se completaron cuatro. El equipo está formado por tres desarrolladores y olvidaron agendar un día de asueto laboral, lo cual reduce las jornadas a 14 cada uno. Para calcular la velocidad se usa la siguiente fórmula: Velocidad = Puntos completados / Jornadas reales, por lo que el cálculo quedaría de la siguiente forma:

Puntos completados: 13 + 8 + 5 + 5 = 31

Jornadas reales: 14 + 14 + 14 = 42

Velocidad= 31 / 42 = 0.73

La velocidad del equipo es del 73%, lo cual quiere decir que el otro 27% se dedicaron a otras cosas o en todo caso fue “la pérdida” por el día de asueto no contemplado. Este dato puede usarse para comprometer PHU en una próxima iteración.

3.11.3 Diagrama de flujo acumulativo

Este diagrama también conocido como *Cumulative Flow* se usa para proporcionar un estado del proyecto a nivel superior, en vez de por sprint. “Este tipo de diagrama puede ser una gran herramienta para identificar obstáculos y cuellos de botella en los procesos.” (Guía SBOK, 2016).

Sprint	Historias de Usuario		
	Por hacer	En desarrollo	Completadas
1	2	2	2
2	4	2	1
3	5	1	1
4	3	3	3
5	6	0	0

Se toma como base las historias de usuario que están pendientes por hacer, las que están en desarrollo y las completadas. En la figura 17 podemos ver cómo hay más historias de usuario en la columna Por hacer, lo que implica que el área de

desarrollo está teniendo problemas y se ha convertido en un cuello de botella e incluso que el *sprint* 5 aún no ha iniciado. Estos datos también se pueden graficar para una mejor apreciación por parte de los usuarios.

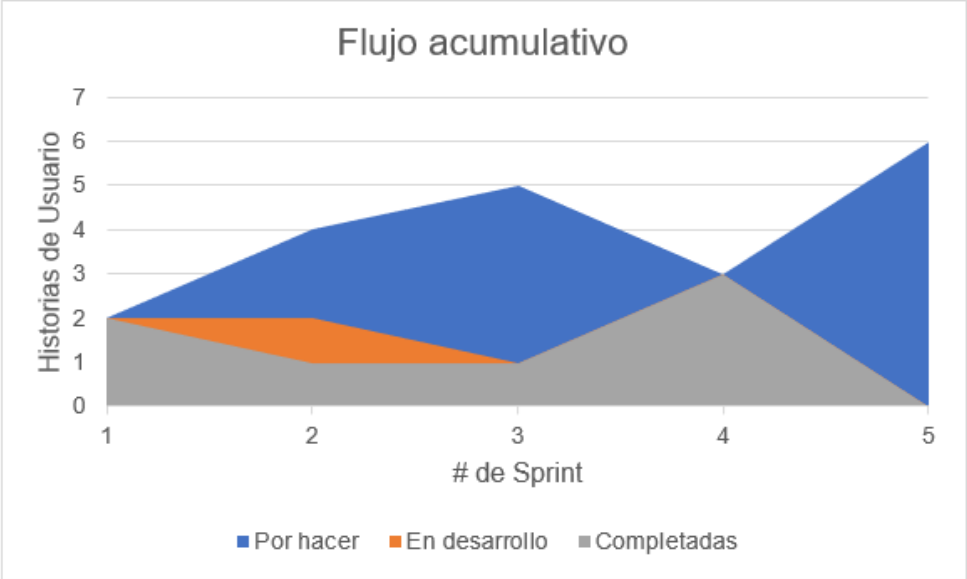

Fig. 17 *Gráfico de flujo acumulativo.* Elaboración propia.

3.12 Scrum en equipos paralelos

Una de las características de Scrum es el trabajo en pequeños equipos, pero no por ello es limitativo a proyectos pequeños, es posible trabajar con equipos en paralelo cuando el proyecto es más complejo.

Los proyectos grandes pueden tener múltiples equipos Scrum trabajando en paralelo, sólo hay que hacer lo necesario para sincronizar y facilitar el flujo de información y mejorar la comunicación entre los equipos involucrados. Es común que estos proyectos complejos formen parte de un programa o cartera. Para trabajar con equipos en paralelo es importante mapear los roles base al nivel estratégico requerido.

NIVEL	ROLES
Portafolio	*Portfolio Product Owner* *Portfolio Scrum Master* *Stakeholders del portafolio*
Programa	*Program Product Owner* *Program Scrum Master* *Stakeholders del programa*
Proyectos	*Chief Product Owner* *Chief Scrum Master*

Tabla 12. *SCRUMstudy (2016). Roles escalados en Scrum*

Una forma de realizar la coordinación de equipos es con la reunión llamada *Scrum de Scrums*, en un proyecto/programa luciría así:

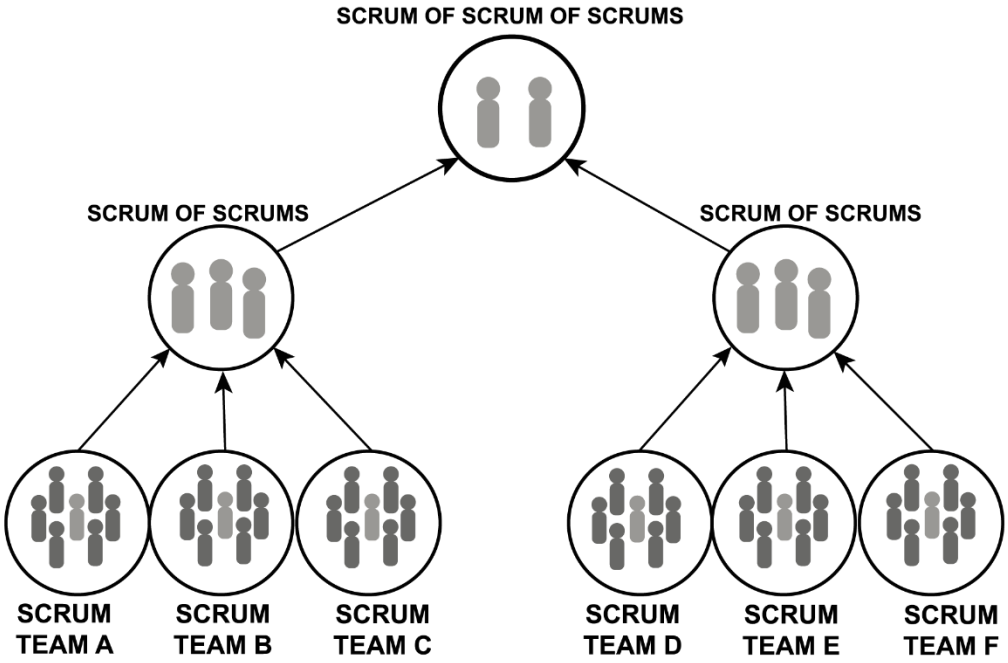

Fig. 18 SCRUMstudy. (2016). *Scrum de Scrums.*

Como puede verse en la figura anterior, hay una reunión de sincronización en cada nivel que se encuentra involucrado. El *Scrum de Scrums* es la reunión que busca

asegurar que los equipos se comuniquen para visualizar los riesgos compartidos, las dependencias entre las tareas y el trabajo futuro que cada equipo realizará. Además de esto deben establecerse lineamientos claros para el control del *Product Backlog.*

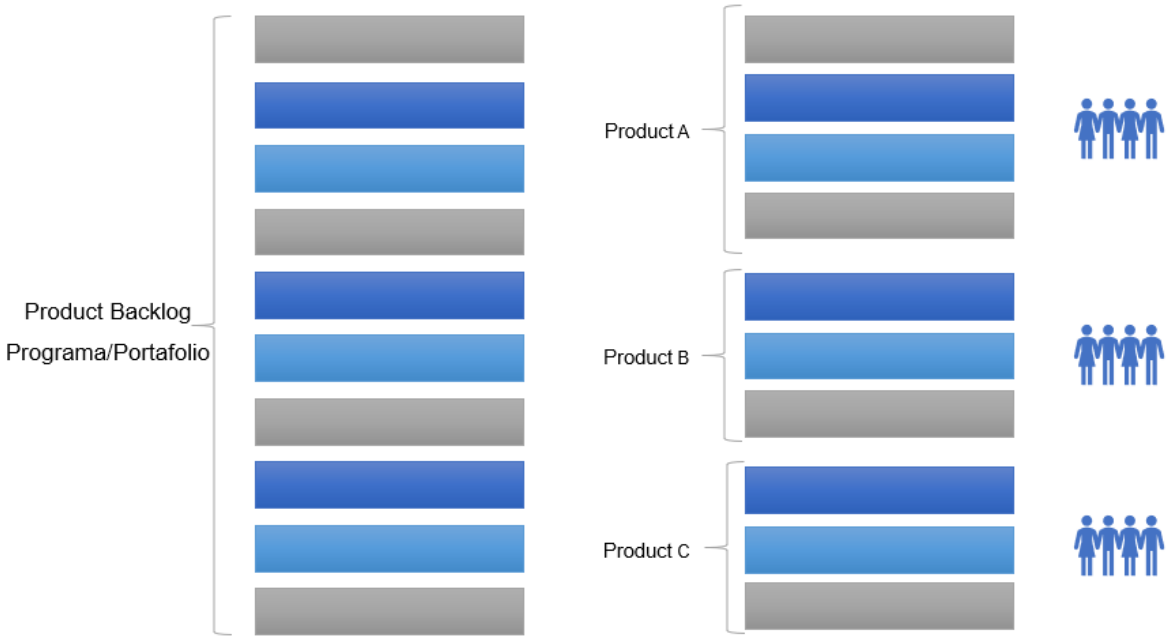

Fig. 19 *Product Backlog de un programa/portafolio. Elaboración propia.*

Un proyecto, programa o portafolio al usar Scrum tiene un solo *Product Backlog*, sólo se divide y se asigna a los equipos necesarios; cada equipo se encarga de la gestión de la parte que le corresponde.

Capítulo 4. Cultura organizacional con Scrum

La adopción de cualquier marco ágil va de incorporar cambios en la forma de trabajo a nivel individual, equipo e incluso a nivel organización. Scrum no sólo se puede abordar como un marco de trabajo para el desarrollo de software, sino que también puede adoptarse como parte de la cultura laboral de una empresa, pues como se vio en el capítulo anterior, la mayoría de sus principios giran en torno a individuos, sus interacciones y la mejora continua de los distintos elementos con los que interactúan, como: herramientas, procesos, flujo de comunicación, entre otros, de tal manera que tanto individuos, como equipos y la empresa evolucionan.

Una organización es definida como: “un sistema de actividades conscientemente coordinadas formado por dos o más personas. La cooperación entre ellas es esencial para la existencia de la organización, y ésta existe cuando hay personas capaces de comunicarse que estén dispuestas a actuar conjuntamente para obtener un objetivo común” (Chiavenato, 1995, p. 7). Partiendo del concepto anterior, las empresas establecen diversos objetivos a cumplir, en el caso de las que desarrollan software una de las principales metas es crear soluciones innovadoras y de alta calidad y esto se logrará solo con la participación de las personas que pertenecen a ella y con la ejecución de planes a corto, mediano y largo plazo a lineados a la planeación estratégica de la organización.

4.1 Planeación estratégica

Sin importar el tipo de proyecto y la metodología a usar para su gestión, el planear de manera correcta ayuda a establecer prioridades, para así poder ejecutarlo con un alcance, tiempo y costo adecuado. La planeación “implica seleccionar misiones y objetivos, así como las acciones necesarias para cumplirlos, y requiere por lo tanto de la toma de decisiones; esto es, de la elección de cursos futuros de acción a partir de diversas alternativas” (Koontz & Weihrich, 1998, p. 35). Si queremos ajustar este concepto a nivel organización tenemos a la planeación estratégica que según Idalberto Chiavenato “se refiere a la manera como una empresa intenta aplicar una determinada estrategia para alcanzar los objetivos propuestos. Es generalmente una planeación global y a largo plazo” (Chiavenato, 1995, p. 59). Por su parte, Octavio Reyes la define como “el esfuerzo sistemático y formal de una empresa

para establecer sus propósitos objetivos, políticas y estrategias básicas, desarrollando planes detallados con el fin de ponerlos en práctica, lograr sus propósitos y proporcionar los resultados que satisfacen las expectativas de sus clientes” (Reyes, 2012, p. 68).

Tomando en cuenta las definiciones anteriores, podemos identificar puntos de coincidencia, por ejemplo: el establecimiento de objetivos, la toma de decisiones y sobre todo los individuos que la conforman, pues son ellos quien ejecutan los planes que la llevara a conseguir objetivos en todos los niveles estratégicos de la misma, y por otro lado son quienes realizan las actividades relacionadas con el desarrollo de proyectos, de productos o con la entrega de servicio, por lo que la operación y procesos de la empresa deben hacer sentido con la estrategia, y para ello construir una cultura organizacional fuerte será de gran utilidad, para que los individuos y el ambiente donde se desempeñan puedan evolucionar.

4.2 Cultura organizacional

La cultura por si sola, pudiéramos definirla como aquellos valores y normas no escritas que comparte un grupo de personas, nuevamente el factor humano se hace presente, por lo que cualquier grupo de individuos que compartan un objetivo (como en el caso de los equipos de trabajo y organizaciones) son propensos a desarrollar su propia cultura.

“La cultura organizacional se puede entender como un sistema de representaciones, capacidades y habilidades, que comparte un grupo de individuos para lograr sus objetivos y metas, actuando como colectivo en el marco de su sociedad específica” (Gómez C., 2001, p. 111). En cambio, Schneider tiene una acotación más simple: "La cultura organizacional es la forma en que hacemos las cosas en orden para tener éxito” (Schneider, 1999, p. 72). El desarrollo de esta cultura no solo está relacionado por el compartir objetivos, sino que está ligado a la interacción de las personas, los procesos a los que se apegan, los valores que comparten y el conocimiento que crean en conjunto; se puede planear la creación una cultura organizacional y cuando queremos que está se centre en el cliente, en la experimentación, al aprendizaje, a la entrega de valor continua y el desarrollo del personal, Scrum se vuelve una gran herramienta para lograrlo.

4.3 Scrum como cultura de trabajo

Varios exponentes de Scrum como Ordóñez (2015) así como blogs relacionados al tema, explican la diferencia entre 'hacer *agile'* y 'ser *agile'*.

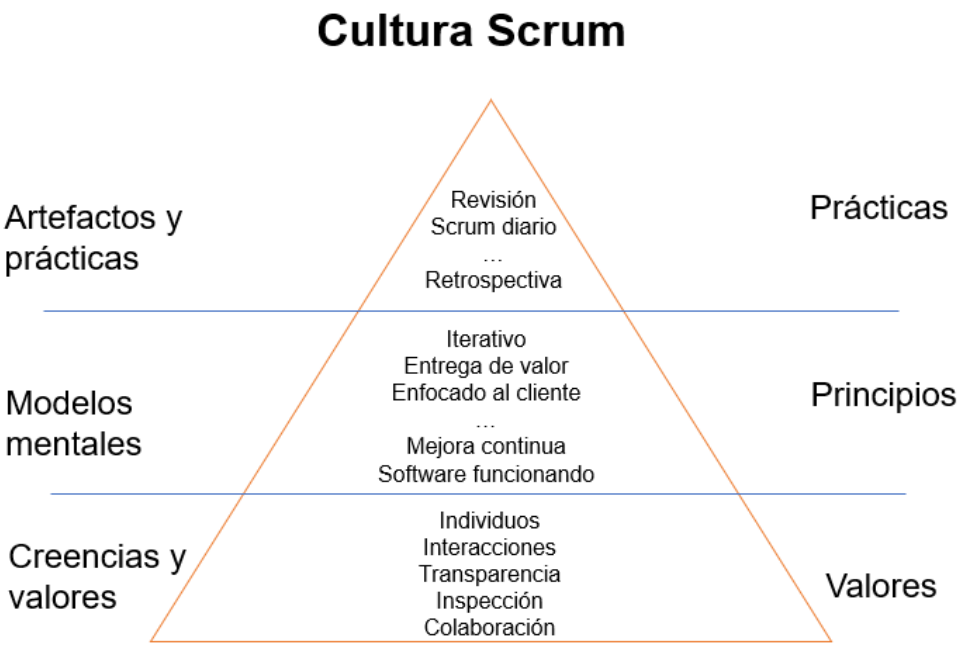

Fig. 20 *Scrum como cultura.* Elaboración propia, basado en Ordóñez, J. (2015*).*

'Hacer agile' se refiere a ejecutar las prácticas que proponen las metodologías ágiles (en este caso Scrum), que pudiera compararse también como la punta de un iceberg y con lo que comúnmente se asocia con empezar el cambio en la cultura de la organización; sin embargo, el 'ser agile' es lo que genera el verdadero cambio y esto se encuentra en la base de la pirámide, que son los principios y valores de Scrum propuestos en el manifiesto ágil.

"El modelo cultural de Schneider define cuatro culturas distintas:

- La cultura de la colaboración es sobre trabajar juntos.
- La cultura del control es sobre conseguir y mantener el control.
- La cultura de la competencia es sobre ser el mejor.
- La cultura de cultivación es sobre el aprendizaje y el crecimiento con un sentido de propósito." (Sahota, 2012, p. 8)

Como puede observarse este modelo busca generar cambios culturales a través de cuatro perspectivas distintas, Michael Sahota adapta el modelo de Schneider a Scrum a través del siguiente diagrama.

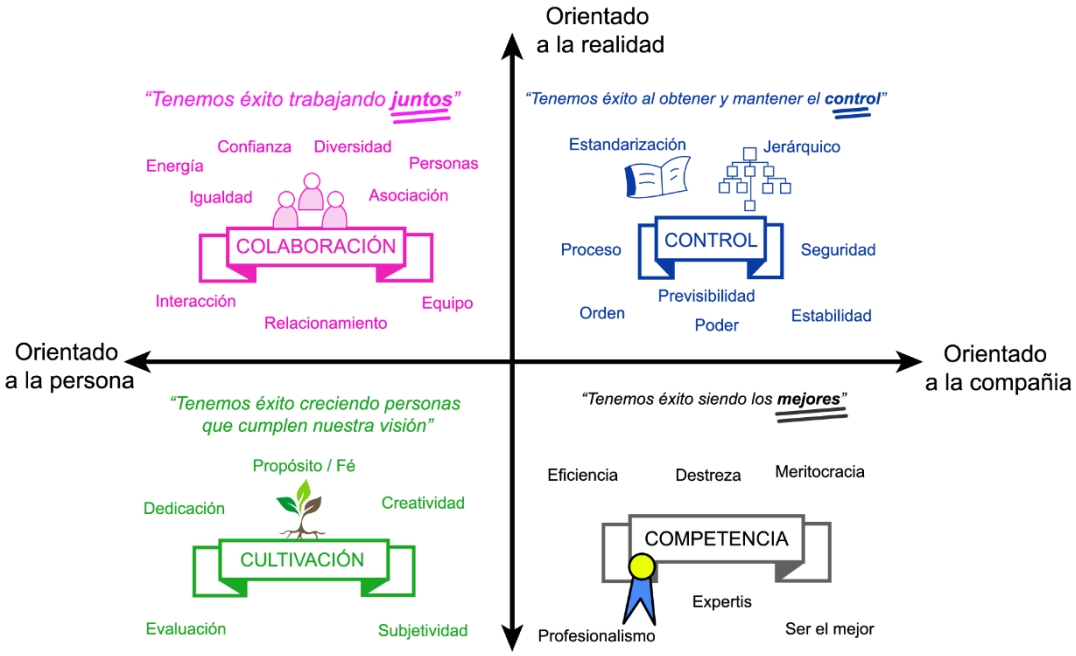

Fig. 21 Sahota, M. (2012*). Modelo de Schneider sobre la cultura de Scrum*.

Ninguna cultura es mejor que la otra, simplemente varían dependiendo de la organización, de su objetivo e incluso en el giro en el que se desempeñe. La cultura de la agilidad (y de *Scrum*) está basada en la cultura de la colaboración y de la cultivación, esto de acuerdo con una encuesta realizada por Michael Spayd (2010).

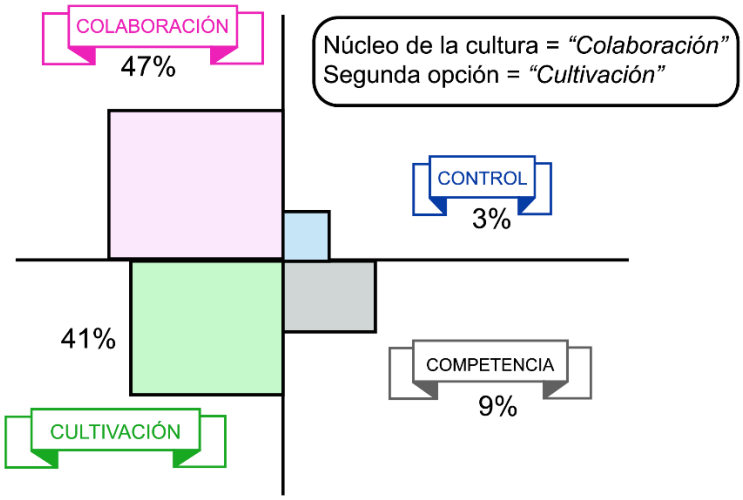

Fig. 22 Sahota, M. (2012). *Cultura de la agilidad*. Encuesta realizada por Spayd.

La cultura de la colaboración se ve sostenida por los valores del manifiesto ágil que pone como prioridad a los individuos y sus interacciones sobre procesos y herramientas. La cultura de la cultivación es apoyada por todos aquellos principios de la agilidad que hablan sobre el equipo motivado y autoorganizado, la inspección continua, la calidad y demás aspectos que promueven la mejora continua.

Sahota menciona en su libro que “Cuando las personas están orientadas al propósito de la iniciativa de cambio Ágil está apoyando (el POR QUÉ), son más capaces de evitar distraerse por los detalles del proceso (el QUÉ y el CÓMO).” (Sahota, 2012, p. 32).

Cada empresa tiene su propia cultura, misma que debe estar alineada a los objetivos estratégicos y ser apoyada por las prácticas que adoptan los equipos de trabajo en sus actividades diarias.

Capítulo 5. Investigación y análisis de resultados

El diseño de la presente investigación fue estructurado para realizar un análisis deductivo, quedando de la siguiente forma:

- Revisión de la literatura: comenzando por la teoría general de los modelos de desarrollo de software y su evolución, para después centrarse en el origen de la agilidad como una nueva tendencia en las metodologías empleadas para la ingeniería de software y haciendo una revisión sobre los fundamentos descritos para el uso Scrum por parte de Ken Schwaber y Jeff Sutherland, entre otros autores destacados.
- Revisión del Estado del Arte: Se revisaron diversos informes y encuestas que dan parte de la situación que vive la industria del desarrollo de software a escala internacional respecto al uso de metodologías ágiles.
- Recolección y análisis de datos: Se elaboró un cuestionario para sondear el uso de Scrum en el área tecnológica dedicada al desarrollo de software en la Ciudad de México, para así detectar prácticas y herramientas comúnmente utilizadas por profesionales de la industria que han completado proyectos, usando este marco de trabajo.

5.1. Selección de la muestra

Al ser ésta una investigación de carácter cualitativo, “el principal factor es que los datos recabados proporcionen una comprensión profunda del ambiente y el problema de investigación, por lo que la muestra seleccionada no debe ser empleada para representar a una población” (Sampieri, 2014).

Al tratarse de una investigación cualitativa, se siguió la recomendación que da Sampieri en cuanto al tamaño sugerido para la muestra de un estudio del estilo de teoría fundamentada (de 20 a 30 informantes), esta magnitud de la muestra es conveniente para que el análisis de información y el tiempo que se tuvo para realizarlo.

Es una muestra homogénea de 30 participantes que cumplen los siguientes aspectos:

- Personas dedicadas al desarrollo de software.
- Desempeñándose laboralmente en la Ciudad de México.
- Han ejecutado alguno de los roles del equipo principal de Scrum.
- Tienen experiencia mínima de 1 año en el uso de Scrum como metodología de desarrollo.
- Han concluido proyectos de desarrollo de software usando Scrum.
- En su último año laboral han empleado Scrum.

De esta forma las personas que participaron en el estudio, al cumplir con los requisitos anteriores, se convierten también en una muestra de expertos en el área.

También es una muestra por conveniencia, ya que los participantes han sido recomendados por colegas del ámbito tecnológico o se interesaron en participar en este estudio por la invitación publicada en diversos grupos especializados en el uso de Scrum.

5.2. Descripción de herramientas de recolección de información

Entrevista Estructurada

El cuestionario usado (véase Anexo 1) está formado por una sección de datos generales del participante y cinco apartados relacionados con el uso de Scrum. El primer apartado trata sobre la relación que tiene el participante con Scrum y de forma general su experiencia al implementarlo. El segundo apartado se enfoca a conocer las prácticas al iniciar un proyecto con Scrum. El tercer apartado está encaminado a identificar los artefactos y ceremonias ejecutados regularmente en un proyecto Scrum. En el apartado cuatro se abordan preguntas relacionadas a Scrum aplicado a proyectos con equipos trabajando en paralelo. El apartado cinco, está dedicado a conocer prácticas comunes para el cierre de proyectos con Scrum.

Aplicando las herramientas previamente descritas se obtuvo información relevante sobre cómo algunos equipos de desarrollo de software usan Scrum como un marco para gestionar un proyecto, las prácticas y herramientas más utilizadas y su percepción con respecto al uso de estas.

5.3. Relación con Scrum

Los participantes en este estudio tienen más de un año ejecutando algún rol descrito en Scrum (véase Gráfico 2, Anexo 2), no sólo en la organización en la que laboran actualmente, sino que su experiencia con este marco de trabajo comenzó en otras organizaciones, por lo que se comprueba que su uso en la industria tecnológica mexicana viene de tiempo atrás, incluso más de una década, pues uno de los entrevistados mencionó llevar ejecutando el rol de Scrum Master desde hace 12 años.

El 80% de ellos se encuentra certificado en el rol que ejecutan actualmente (véase Gráfico 3, Anexo 2), algunos dijeron haber empezado con la práctica y lograron formalizar este conocimiento por medio de cursos; el otro 20% que no se encuentra certificado ha adquirido conocimiento de manera autodidacta, consultando libros, videotutoriales en internet y de la práctica misma en el campo laboral, por lo que podemos confirmar que Scrum es uno de los marcos de trabajo más usados para desarrollar tecnología en la Ciudad de México, pues hay gran cantidad de fuentes de información a las que se puede recurrir para su conocimiento e implementación. SCRUMStudy se posiciona como la empresa a la que más se recurre para la certificación en Scrum.

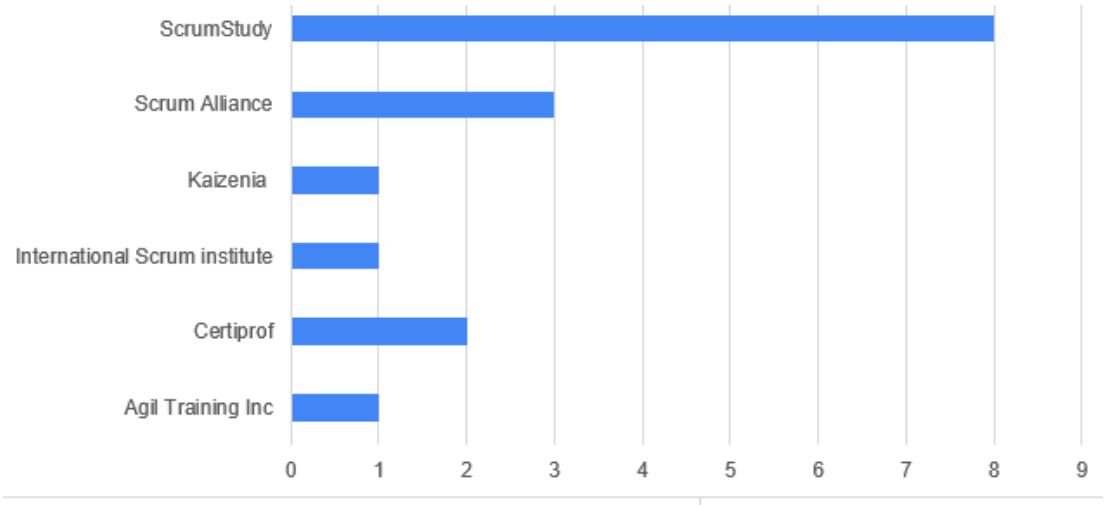

Fig. 23 Empresas certificadoras a las que recurrieron los participantes. Elaboración propia

Cualquier equipo que intentan usar Scrum por primera vez en sus proyectos se enfrentan a diversos retos en su adopción, relacionados con:

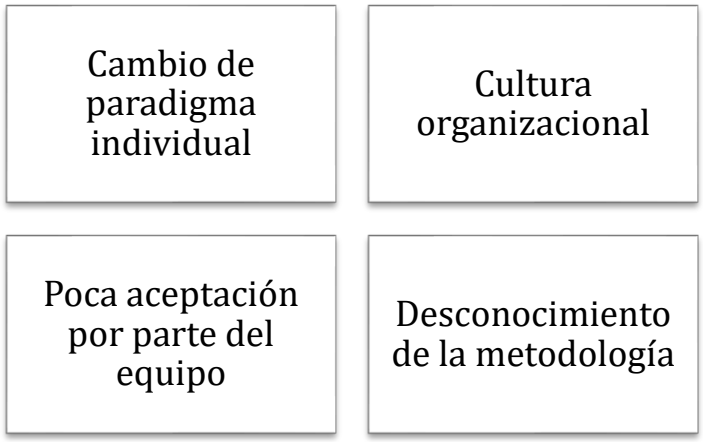

Fig. 24 *Desafíos en la implementación de Scrum*. Basado en respuestas de los participantes del estudio.

Al preguntar a los participantes sobre el mayor desafío al que se han enfrentado al implementar Scrum, varios comentaron que el desconocimiento del marco de trabajo es a lo primero que se enfrentan; la Dirección puede no estar familiarizada con Scrum, por lo que no se invierte en la capacitación formal de los equipos ni en el cambio organizacional; esto repercute también en todo el equipo pues se llega a asumir el primer proyecto sin habilidad y conocimiento necesario.

Scrum promueve la comunicación cara a cara y reduce la documentación, tal como se declara en el *Manifiesto Ágil*, y esto implica cambios en el entorno del individuo sobre la forma en la que se interactúa laboralmente e incluso del desarrollo de habilidades blandas que permitan mejorar el flujo de comunicación e incrementar la confianza y responsabilidad al interior de los equipos. Esto se relaciona directamente con la aceptación del marco, pues Scrum busca responsabilizar al equipo completo del éxito (o fracaso) de un proyecto, democratizando a la vez la organización del trabajo; todo lo opuesto a cuanto estila la administración de proyectos tradicional a la que se siguen apegando muchos de los procesos actuales de las organizaciones y que se basa en el comando y control exhaustivo.

Como se explicó en capítulos anteriores, Scrum se basa en el control de proceso empírico, por lo que es ideal para proyectos que lidian con un alto porcentaje de incertidumbre, al igual que el enfoque tradicional necesita de planeación, pero no en altas cantidades por adelantado, sino que se va ajustando, lo cual deriva en muchas particularidades en cada proyecto desarrollado.

5.4. Sobre el inicio del proyecto

Sin importar el tipo de perspectiva desde la que se aborda un proyecto (enfoque tradicional o ágil), la etapa de inicio es la piedra angular del mismo, pues es cuando se establece el objetivo que ha de perseguir el equipo de trabajo, su alcance, los recursos a invertir, entre otras variables adicionales.

Parte de la planeación inicial incluye la selección del equipo que se encargará de su desarrollo, Scrum propone trabajar en equipos pequeños que permitan comunicarse de manera adecuada; sin embargo, en nuestra investigación pudimos observar que: el equipo más pequeño detectado fue de 3 personas y el más grande de 30, obteniendo una media 11 personas trabajando en un equipo Scrum, dos más de lo recomendado pues “más de nueve miembros en el equipo requiere demasiada coordinación” (Schwaber y Sutherland, 2017, p. 7). En algunos casos el tamaño del equipo se fue ajustando durante el avance del proyecto, en otros casos fue calculado de acuerdo con la complejidad y necesidades del negocio, o asignados por la disponibilidad del recurso o incluso por la cantidad de requerimientos a atender y la urgencia con la que debían atenderse.

Recordemos que el equipo principal de Scrum está formado por: un *Scrum Master*, un *Product Owner* y el *Scrum Team*; estos son los tres roles obligatorios para usar la metodología de manera correcta; con las respuestas obtenidas, sólo el 75% de los proyectos desarrollados contaron con los tres roles.

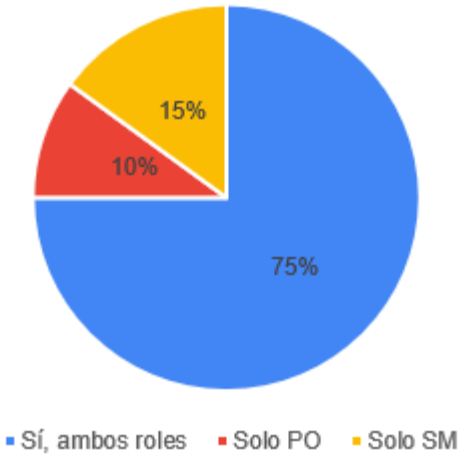

Fig. 25 *Roles involucrados en los proyectos*. Basado en respuestas de los participantes del estudio.

El 10% solo tuvo el rol de *Product Owner*, al no contar con un *Scrum Master* que apoyara al equipo con el uso de Scrum y sus prácticas, el desempeño en el proyecto pudo verse impactado; algo parecido ocurre con el 15% que no tuvo a un *Product Owner*, algunos entrevistados coincidieron en que era complicado satisfacer las necesidades del cliente, pues no había un rol que aclarara las dudas del negocio de manera dedicada.

La asignación de estos roles principales se realizó de la siguiente forma:

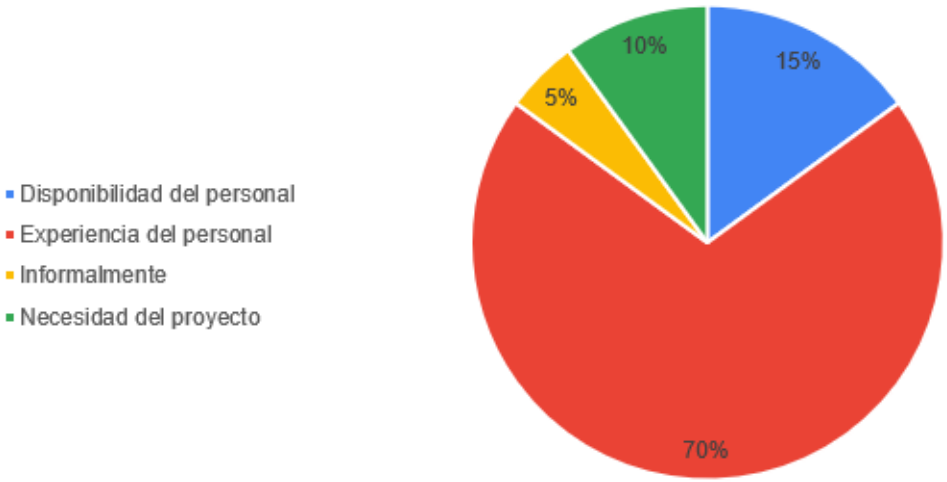

Fig. 26 *Parámetros para la asignación de SM y PO*. Basado en respuestas de los participantes del estudio.

En el 70% de los casos, el SM y el PO fueron asignados con base en su experiencia en el rol, lo cual concuerda con los datos mostrados al principio de este análisis; el 15% fue asignado por la disponibilidad de tiempo del personal; el 10% con base a la experiencia relacionada al tipo de tecnología o giro del proyecto, y al 5% restante se le asignaron algunas responsabilidades del rol, sin hacerlo de manera formal.

Una vez seleccionado el equipo principal, se debe proceder con el levantamiento de información del proyecto, para esto pueden usarse diversas técnicas, entre las que destacó el uso de entrevistas (véase Gráfico 9, Anexo 2), seguida del uso de formatos diseñados por la organización. Ambas estrategias permiten generar la información necesaria para compartir con el equipo y realizar estimaciones generales a todo el proyecto, mismas que pueden ser plasmadas en documentos de inicio de proyecto, en el Product Backlog, tableros y otras herramientas digitales compartidas con los miembros del equipo e incluso con otras áreas de la organización como Ventas, Soporte, Infraestructura, entre otras.

Una vez recopilada la información necesaria del proyecto (con un objetivo y alcance establecido y el equipo de trabajo seleccionado) en el 100% de los casos se usó una *Reunión de inicio de proyecto* o también llamado por los participantes *Kick-Off* para iniciar formalmente con el desarrollo. En ocasiones estas reuniones son usadas para trazar una ruta o plan de liberaciones del proyecto, en el cual se establecen hitos a conseguir durante el desarrollo y el número de iteraciones a ejecutar; lo anterior siempre basado en la estimación que realice del *Scrum Team* y (si se cuenta con ello) las métricas de un proyecto anterior para tomarlas como referencia. Se encontró que el número de *sprints* se determinó de la siguiente forma:

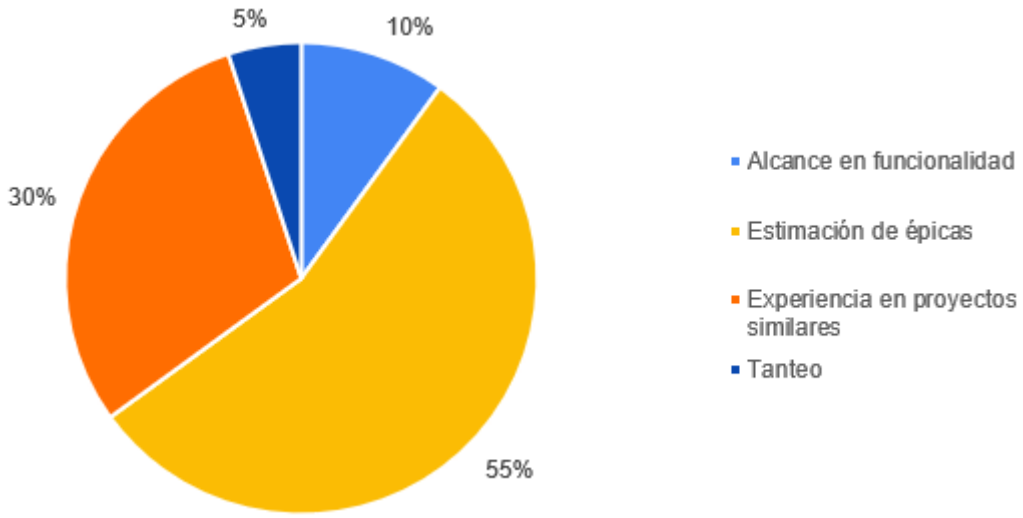

Fig. 27 *Aspectos tomados en cuenta para determinar el número de Sprints de un proyecto*. Basado en respuestas de los participantes del estudio.

La estimación de épicas es la estrategia más usada para determinar el número de iteraciones que tendrán un proyecto, la herramienta más mencionada para realizar esta aproximación fue el *Planning Póker*. La segunda estrategia usada para la estimación fue la experiencia en proyectos similares, aunque su uso fue debido a que el *Product Owner* realizó la estimación sin el consenso del *Scrum Team*.

5.5. Sobre los ceremonias y artefactos

La primera reunión por realizarse durante una iteración es la *Sprint Planning* que, como se vio en la revisión de la teoría, requiere al equipo principal; sin embargo, esto no siempre se lleva a cabo.

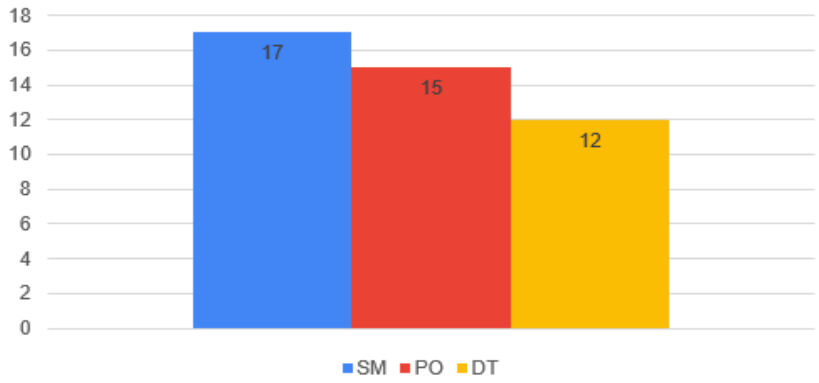

Fig. 28 *Participación de roles en la Sprint Planning*. Basado en respuestas de los participantes del estudio.

Gracias a estos datos podemos inferir que, en los casos en lo que el *Scrum Team* no asiste al *Sprint Planning*, quienes estiman y asignan las tareas a realizar son el SM y el PO, lo cual puede traer consigo errores que pongan en riesgo el incremento esperado del producto, mismo que también puede verse comprometido cuando el PO no asiste a esta reunión para dar claridad sobre el incremento y las historias de usuario relacionadas.

Relacionado con la claridad que debe proporcionar el PO sobre las historias de usuario, se detectó que esto es abordado en sesiones de refinamiento, en las cuales se toma como base las reglas de negocio identificadas al inicio del proyecto.

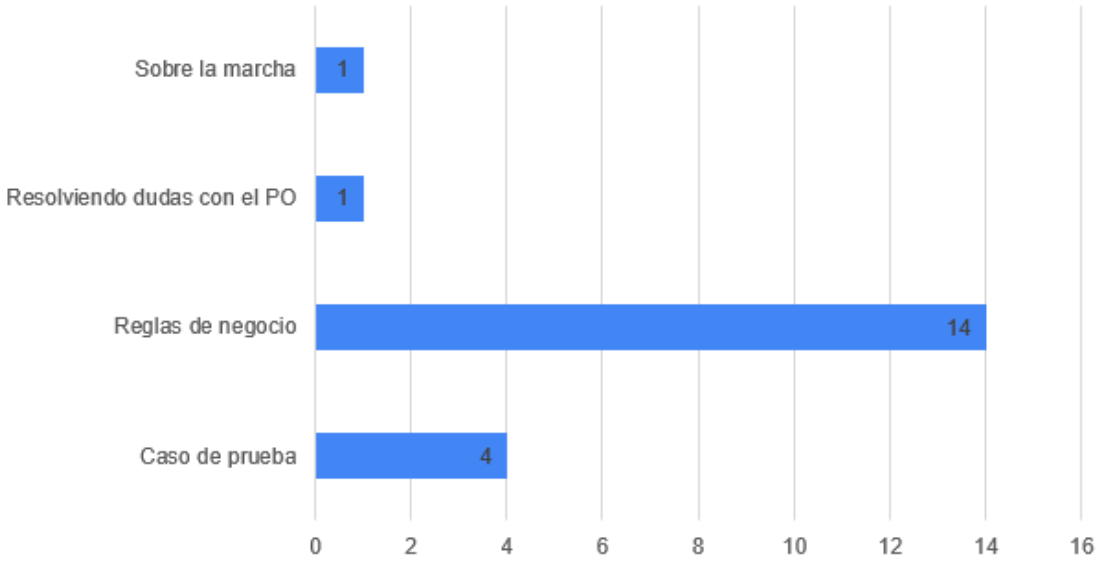

Fig. 29 *Técnicas usadas para el refinamiento de Historias de usuario*. Basado en respuestas de los participantes del estudio.

Las HU seleccionadas para trabajar en el *sprint* fueron aquellas que aportaron mayor valor al negocio, o dependiente para otra funcionalidad (véase Gráfico 13, Anexo 2) y cada una de ellas (en su mayoría) contó con los *Criterios de terminado* y *Criterios de aceptación* relacionados (véase Gráfico 14, Anexo 2), tal como lo señalan las prácticas sugeridas de Scrum.

Las historias de usuario (HU) deben dividirse en tareas para ser ejecutadas por el equipo, Scrum promueve que cada miembro seleccione aquellas tareas de las que

se hará responsable, pero como veremos en el siguiente gráfico, el SM aún se encarga en algunos proyectos, de realizar la asignación de tareas.

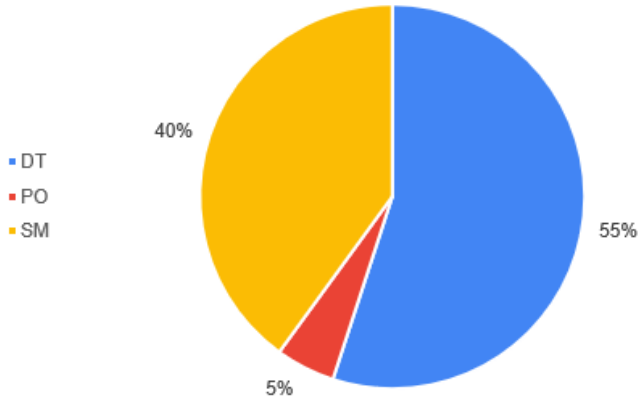

Fig. 30 *Asignación de tareas*. Basado en respuestas de los participantes del estudio.

Al preguntar a los participantes sobre qué mejorarían en la ceremonia de *Sprint Planning*, se expresó repetidamente que el refinamiento de las HU debería hacerse con anticipación, no en la misma reunión, pues no debería realizarse, ni durante, ni después de la *Sprint Planning*, ya que esto afecta la certeza de la estimación del trabajo que puede ser completado durante la iteración. Aunque mayormente las tareas fueron realizadas y completadas a tiempo (véase Gráfico 17, Anexo 2) surgieron diversos problemas como: mala comunicación en el equipo, actividades no programadas, mal manejo de las expectativas del cliente, dependencias externas, mala estimación, entre otras; estas eventualidades fueron manejados con una inversión de tiempo extra, atendido por el mismo equipo o con la incorporación de nuevos miembros al equipo.

El uso de controles del estilo de Gantt, es un detalle particular relacionado con el control de tareas del equipo, pues fue de las herramientas más mencionadas, seguida del uso de tableros digitales como Jira, Trello y Monday.

En cuanto a las *Daily Scrum*, aún hay equipos que no las realizan (véase Gráfico 17, Anexo 2) y cuando se realizan, menos de la mitad dice respetar los 15 minutos del *time-box* propuesto:

Fig. 31 *Duración de la reunión diaria*. Basado en respuestas de los participantes del estudio.

El tiempo llega a rebasar los 15 minutos debido a que las personas buscan resolver los problemas o generar acuerdos en esta reunión, por la falta de práctica de comunicación cara a cara, falta de interés, impuntualidad, etcétera. Esto según las respuestas de los participantes.

Pasando a la *Sprint Review*, de acuerdo con los datos recabados si se ha incluido al cliente en este evento (véase Gráfico 21, Anexo 2), aunque aún puede mejorarse este aspecto. Las estrategias que se han implementado y que hemos analizado a este momento, permitieron que el objetivo de algunos *sprints* haya sido alcanzado sin contratiempo mayores.

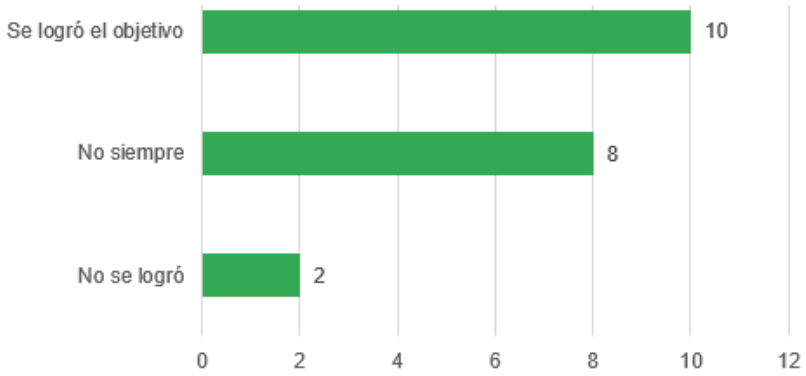

Fig. 32 *Consecución del objetivo del Sprint.* Basado en respuestas de los participantes del estudio.

Las ocasiones que no se logró la meta, fue debido a la atención a actividades que no se encontraban programadas, mala estimación de tareas e Historias de usuario y la falta de disciplina por parte del equipo.

Continuando con la *Sprint Retrospective*, recordemos que Scrum menciona que el *Scrum Team* y el *Scrum Master* son los roles obligatorios que deben asistir a esta ceremonia, pues son ellos quienes revisarán el desempeño del sprint anterior con respecto a herramientas, procesos y demás. Así mismo acordarán mejoras y crearán planes de acción para incorporarlas, sin embargo, se identificó que no en todos los casos se realiza la *Sprint Retrospective.*

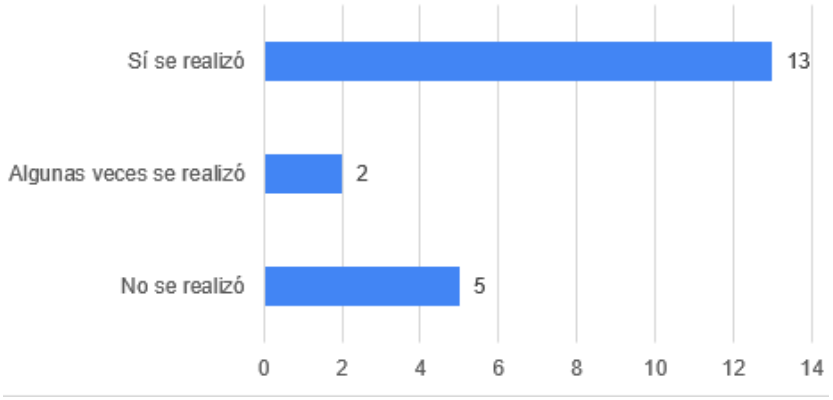

Fig. 33 *Ejecución de Sprint Retrospective*. Basado en respuestas de los participantes del estudio.

Esto impide seguir con el proceso de control empírico en el que se basa Scrum, pues al no realizar las retrospectivas, no se pueden incorporar mejoras a nuestra forma de trabajo. Además, esta sesión es ideal para apoyar al equipo a lidiar con el resultado de la *Sprint Review* para luego iniciar una nueva iteración con mejor enfoque, por medio de dinámicas de integración que fomenten la honestidad, el respeto y la confianza que impulsarán al equipo a alcanzar un nivel de productividad más alto.

5.6 Sobre el trabajo en equipos en paralelo

De acuerdo con el SBOK para trabajar un proyecto/programa con más de un equipo Scrum, es necesario mapear las responsabilidades de los roles SM y PO. Estas actividades, de acuerdo con la investigación realizada, recaen en los siguientes niveles jerárquicos de la organización, como en gerentes, directores y en ocasiones

a figuras como *Agile Coach*, que son quienes se encargan de liderar el cambio a la agilidad a nivel organización.

Con respecto a la sincronización de los equipos, su frecuencia varía de organización en organización, el *Scrum de Scrums* suele realizarse: semanal, quincenal, mensual o trimestralmente, esto depende del tamaño del proyecto y prioridad que tiene para el negocio.

Para reducir el riesgo que implica abordar proyectos complejos y con varios equipos trabajando a la vez, se vuelve necesario crear planes en conjunto de: comunicación, seguimiento de tareas, reglas de negocio, criterios de terminado, gestión del *Product Backlog*, rastreo de dependencias, entre otras estrategias que permitan el avance coordinado de los equipos y su trabajo asignado.

5.7 Sobre el cierre del proyecto

Para finalizar un proyecto se encontraron tres prácticas recurrentes:

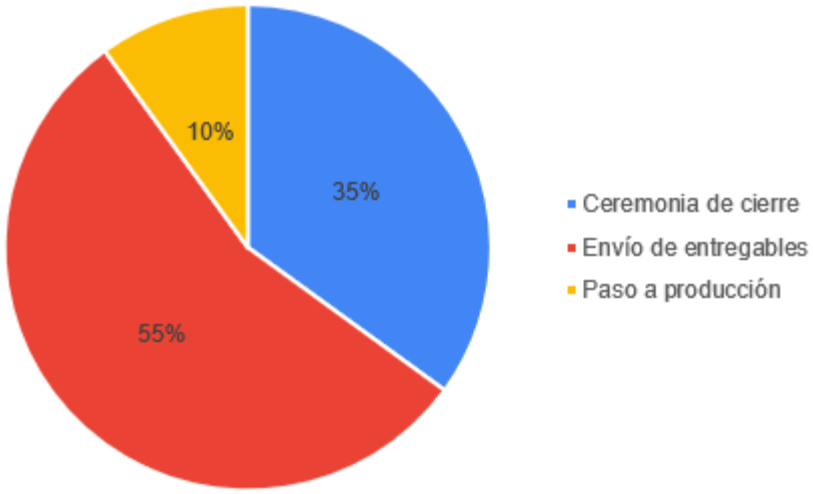

Fig. 34 *Estrategias para la finalización de un proyecto*. Basado en respuestas de los participantes del estudio.

El envío de entregables es la estrategia de cierre más común; los entregables pueden ser: informes técnicos de la instalación, código fuente de la aplicación o sistema, documentación técnica relacionada, manuales de operación, evidencias de funcionamiento, pero sobre todo: software funcionando.

El principal parámetro para saber si el objetivo del proyecto fue alcanzado es el cumplimiento de los criterios de aceptación establecidos por el cliente, las pruebas

que realizan los usuarios finales y la carta de aceptación con el visto bueno del cliente.

Con respecto al aprendizaje obtenido a lo largo del proyecto, Schwaber y Sutherland no sugieran una herramienta específica en *La Guía de Scrum,* por lo que cada organización decide cómo incorporar las lecciones aprendidas. Preguntando al grupo de expertos se detectó que mayormente los aprendizajes de documentan para que puedan ser consultados por la organización para proyectos posteriores. En segundo lugar, queda en la experiencia adquirida por el equipo y, se espera, sean capaces de aprovecharlo de manera intrínseca en los proyectos posteriores. Un porcentaje menor de los participantes afirma que no se hace nada con la experiencia obtenida en el proyecto.

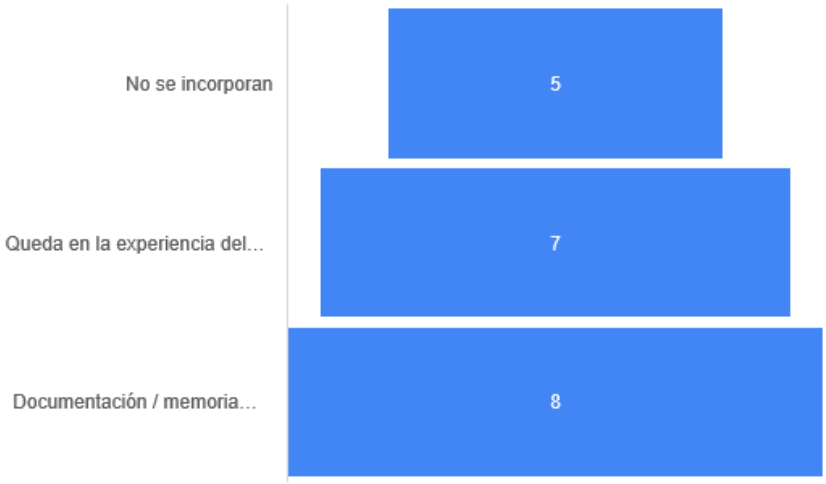

Fig. 35 *Uso del aprendizaje obtenido al finalizar un proyecto*. Basado en respuestas de los participantes del estudio.

Tomando como base el marco teórico revisado en este trabajo y la información compartida por los entrevistados, podemos declinar la hipótesis propuesta al inicio de esta investigación, pues se planteó que "*Scrum no logra implementarse de manera efectiva*…" y este supuesto es incorrecto, ya que se trata de un *Marco de Trabajo*, compuesto de diversas prácticas que deben adecuarse a las necesidades del proyecto y del equipo de trabajo, es decir, debe adaptarse no implementarse. Scrum no es una norma que deba ejecutarse al pie de la letra, sino todo lo contrario, al tratarse de un *framework* iterativo se espera que los procesos de trabajo, herramientas e interacciones del equipo mejoren a cada ciclo, se espera que todos

estos elementos evolucionen de manera continua para alcanzar los objetivos establecidos para el proyecto y para la organización misma. Aunque Scrum no sea una metodología es importante resaltar que Schwaber y Sutherland han dejado claras las reglas básicas para que el marco de trabajo funcione y es ahí donde hay áreas de oportunidad para la industria, mismas de las que hablaremos en el siguiente capítulo.

Capítulo 6. Conclusiones

La necesidad de cambio es innegable, la innovación se vuelve necesaria para permanecer en el mercado. Aquellos profesionistas que lo han entendido han marcado la tendencia de nuevas formas de trabajo y son quienes lideran negocios altamente redituables. Las organizaciones también han evolucionado, la administración tradicional se ha visto rebasada al momento de gestionar procesos relacionados con la solución de problemas y la generación de conocimiento. Claramente el entorno de producción ha cambiado, las empresas más rentables son aquellas que trabajan con conocimiento e innovación por sobre las empresas que trabajan con procesos lineales como líneas de producción.

La industria tecnológica vive en un entorno que cambia constantemente; año tras año surgen innovaciones de todo tipo, principalmente de hardware y software. Gracias a los principios de la Ingeniería de software, se pudo estructurar el trabajo que realizan los equipos de desarrollo. Este primer esfuerzo tomó como base la administración tradicional de proyectos, marcada por el uso de fases secuenciales y alta planeación por adelantado, lo cual es altamente recomendado si los requerimientos del producto están completos desde el inicio. La agilidad surge como una alternativa de Ingeniería de software enfocada a esos proyectos que tienen un alto nivel de incertidumbre, ya sea porque trabajan con requerimientos que no son tan claros al momento de iniciar el proyecto (por el mismo giro del que depende) o por tratarse de algo que jamás se ha hecho.

Scrum es un marco de trabajo para el desarrollo y mantenimiento de proyectos que puede emplearse en cualquier industria. Desde su enfoque ágil busca gestionar proyectos que requieren adaptarse de manera rápida a su entorno, y la palabra clave es *marco de trabajo.* Sus autores, Schwaber y Sutherland, han hecho énfasis en eso: se trata de un conjunto de prácticas y procesos sugeridos que cada equipo debe adaptar a las necesidades de cada proyecto. Por eso su base es el control es de proceso empírico, pues trata de generar conocimiento a través de la observación y la experimentación. Por ello que existen tantas prácticas, métricas y herramientas distintas e incluso conceptos que varían de autor a autor; cada equipo y organización abordan Scrum de distinta forma, es lo que ha enriquecido a este marco, pero siempre debe basarse tanto en los valores y principios que se describen

en el *Manifiesto ágil*, como en las reglas que han establecido Schwaber y Sutherland para Scrum.

Se dice que Scrum es fácil de entender pues el flujo de trabajo que propone es sencillo, los artefactos son pocos y las reuniones tienen un objetivo claro e incluso un tiempo asignado. Sin embargo, es difícil de dominar pues requiere que los equipos cambien su forma de interactuar e incluso que exista un cambio en la cultura de trabajo a escala organizacional.

El primer problema detectado en esta investigación es que no todos los proyectos cuentan con los roles de *Scrum Master* y *Product Owner*, y estos son indispensables para la correcta adopción de Scrum: cada rol tiene responsabilidades específicas que hace que el flujo de procesos funcione de forma correcta; al no existir uno u otro, no se crean acciones para mitigar el impacto relacionado, sino que el único rol de liderazgo que existe trabaja a marchas forzadas para abordar las responsabilidades de ambos roles.

Retomando el tema del control de proceso empírico, recordemos que sus pilares son: la transparencia, la inspección y la adaptación, y prevalecen en cada uno de los elementos que forman Scrum, por lo que, si faltan elementos, el ciclo se ve afectado.

- ✓ Transparencia: esta característica busca hacer visible para todos los involucrados la mayor cantidad de información posible, relacionada al proyecto. Scrum plantea que el Equipo principal debe comprender el objetivo a alcanzar desde el inicio del proyecto; pero se ha detectado que el *Scrum Team* no siempre participa en la planeación del proyecto y en algunos casos no participa en la *Sprint Planning*. Los problemas que esto acarrea son diversos, pero el principal es comprometer el objetivo del sprint con estimaciones hechas por el *Scrum Master* o el *Product Owner*, lo cual confirma que sigue usándose el método de gestión y control, en lugar de la auto organización del equipo. Esto se comprueba en nuestra investigación, pues en el 40% de los casos los entrevistados respondieron que el *Scrum*

master es quien asigna las actividades por realizar y el *Product Owner* sólo en un 5%.

- ✓ Inspección: implica revisar de forma constante el producto y los procesos que se siguen para asegurar calidad en el proyecto. La inspección se hace en distintos momentos, uno de ellos implica llevar a cabo el *Daily Scrum*, pues el equipo habla sobre las actividades que cada uno de los miembros del *Scrum Team* ha realizado, qué planea hacer y qué problemas enfrenta; esto también proporciona transparencia al avance del sprint. En el 80% de los casos revisados esta reunión sí se realizó, pero en el 45% de las veces no se respetó el tiempo asignado de 15 minutos; lo cual sucede porque el equipo trata de resolver ahí mismo los problemas mencionados durante la reunión, siendo que ese no es su objetivo.

  En la *Sprint Review* se inspecciona el incremento del producto; por ello que es ideal invitar a los *stakeholders*. En el 80% de los casos sí se involucró al cliente, lo cual es un buen indicador, pues habla de que se está colaborando con él y que se le está sensibilizando sobre esta forma de trabajo.

  El 35% de los entrevistados aseguró que la *Sprint Retrospective* no se realiza siempre; este aún es un porcentaje considerable por mejorar, pues este evento es en el que se cierra el ciclo de inspección-adaptación propuesto por Scrum.
- ✓ Adaptación: relacionado con el punto anterior, la adaptación surge de los hallazgos encontrados durante la inspección. Si no se realiza de manera adecuada la revisión de los procesos ejecutados, éstos no pueden ser mejorados, lo que impide la generación de conocimiento y rompe con el control de proceso empírico.

Hay diversos puntos de mejora que los profesionales en el desarrollo de software pueden tomar en consideración al momento de utilizar Scrum para gestionar sus proyectos:

El *Scrum Team* debe participar activamente en la planeación del proyecto para realizar estimaciones de las Historias de usuario que formarán parte del *Product Backlog*, ya que ellos son quienes realizarán los entregables y se espera que las estimaciones mejoren a cada iteración junto con la forma de trabajo del equipo, solo de esta manera la productividad podrá incrementar.

Los roles de liderazgo, como el *Scrum Master* y *Product Owner*, deben promover y permitir la auto organización de los equipos, para que pueda generarse el sentido de propiedad compartida. El *Scrum Team* debe asumir responsabilidad sobre la gestión del proyecto y el éxito de este.

Los equipos deben formalizar el conocimiento. Aunque la mayoría de los expertos consultados están certificados, muchos aprendieron Scrum de forma autodidacta y se certificaron cuando ya tenían tiempo ejecutando algún rol. A esto se suma que 35% del conocimiento adquirido se quede en la experiencia individual de cada miembro del equipo y 25% de las ocasiones no se aproveche. Crear una base de lecciones aprendidas a escala organizacional, ayudaría a que se mejore el uso de Scrum y se obtengan mayores beneficios.

Adoptar Scrum implica cambios culturales. Tanto si su implementación viene impulsada de áreas operacionales o directivas, toda la organización debe sensibilizarse en el uso de este marco de trabajo para identificar áreas de oportunidad en la flexibilización de procesos. Scrum trata de empoderar a sus equipos, para que tomen decisiones de manera rápida; sin embargo, los controles predominantes en las empresas no lo impulsan. No hacer cambios estructurales limita e incluso frustra a los equipos que usan Scrum.

Esta investigación se enfocó en estudiar el punto de vista operativo de los equipos que desarrollan software y al momento de finalizarla *La Guía de Scrum* sufrió una actualización por parte de los autores (noviembre 2020), los cambios fueron revisados y no impactan significativamente las conclusiones obtenidas. Se podrían desarrollar a futuro líneas de investigación que profundicen en la perspectiva de gerentes y directivos, para tener un panorama aún más claro sobre las necesidades que presentan las organizaciones para seguir creando software de calidad.

Bibliografía

Beck, K., Beedle, M., Bennekun, A., Cockburn, A., Cunningham, W., Fowler, M., Greenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J. y Thomas, D. (2001). *Manifiesto por el Desarrollo Ágil de Software.* Agilemanifesto. https://agilemanifesto.org/iso/es/manifesto.html.

Chiavenato, I. (1995). *Introducción a la Teoría General de la Administración.* Colombia: Mc Graw-Hill.

Coding Sans. (2019). *State of Software Development in 2019.* Condingsans. https:\\codingsans.com\state-of-software-development-2019.

Cohn, M. (2004). *User stories applied: For agile sofware development.* Estados Unidos: Addison-Wesley Professional.

Cohn, M. (2010). *Succeding with agile: Software development using Scrum.* Estados Unidos: Pearson Education.

Digital.ai. (2020). *14th Annual State of Agile Report.* Stateogagile. https:\\stateofagile.com\

Dutz M.A., A. R. (2018). *El empleos del mañana. Tecnología, productividad y prosperidad en América Latina y el Caribe.* Washinton: Banco Mundial.

Gamboa, J. (2018). Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software. *INNOVA Research Journal, 3(10)*, 4. https://dialnet.unirioja.es/servlet/articulo?codigo=6777227.

Garzás, J. (2017). *Peopleware y equipos ágiles.* Madrid: 233 Grados de TI.

Gómez, C. y. (2001). Teorías de la cultura organizacional. *Revista Contabilidad y Auditoría*, (115).

Gómez, O. T. (2010). Criterios de selección de metodologías de desarrollo de software. *Revista de la Facultad de Ingeniería Industrial*, 13(2),70.

González, D. (2007). *La influencia de la innovación tecnológica, la orientación al mercado y el cápital relacional en los resultados de las empresas de un sector de alta tecnologia. Aplicación a la industría del software de México.* [Tesis de Doctoral. Programa Doctoral: Integración de las Tecnologías de la Información en las Organizaciones]. https://riunet.upv.es/handle/10251/1833.

Guerrero, C. A., Gutiérrez, L. E., & Londoño, J. M. (2014). Estudio comparativo de marcos de trabajo para el desarrollo de software orientados a aspectos. *Información Tecnológica*, 25(2) P. 67-78.

Henández Sampieri, R., Fernández, C., & Baptista, M. (2014). *Metodología de la investigación.* México: McGraw Hill.

Iglesias-Solano, A. (2011). Story Points y su importancia en la estimación de proyectos bajo enfoque ágil. *Revista Investigación y Desarrollo en TIC*, 2 (1).

Maida, E. P. (2015). *Metodologías de desarrollo de software.* [Tesis de Licenciatura en Sistemas y Computación. Facultad de Química e Ingeniería “Fray Rogelio Bacon”. Universidad Católica Argentina].

Mochi, P. (2006). *La industria del software en México en el contexto internacional y latinoamericano.* Morelos: Centro Regional de Investigaciones Multidisciplinarias UNAM.

Ordóñez, J. (2015). *Agile Transformatio and Cultural Change.* [Presentación de diapositivas]. https://www.slideshare.net/JohnnyDark/agile-transformation-and-cultural-change/16-Organizational_Cultureby_Michael_Sahota_Olaf

Pliego F., C. A. (2018). *Boletín economía.* México: Salles, Sainz Grant Thornton, S.C.

Pressman, R. S. (2010). *Ingeniería del software. Un enfoque práctico.* México: McGraw-Hill.

PROSOFT. (2013). *PROSOFT. La política pública que ha transformado una industria.* México: Travesías Editores S.A. de C.V.

PwC. (2014). *The Global 100.* PwC. https:\\www.pwc.com\gx\en\industries\technology\publications\global-100-software-leaders\the-big-picture.html

Reyes, O. (2012). *Planeación estratégica para la alta dirección.* Estados Unidos: Palibrio.

Sahota, M. (2012). *Una guía para la supervivencia a la adopción y transformación ágil: trabajando con cultura organizacional.* Agiltrix. https://www.agilitrix.com/wp-content/uploads/2018/08

Salazar, L. (2020). *Ágil es algo que eres.* LuchoSalazar. https://luchosalazar.com/2020/05/04/de-temas-y-de-epicas-y-de-otras-etiquetas-para-las-historias-de-usuario/

Sampieri, R. (2014). *Metodología de la investigación.* México: McGraw Hill.

Schneider, W. (1999). *The reenginering alternative. A plan for making your current culture work.* Nueva York: McGraw Hill.

Schwaber, K. (1997). *SCRUM Development Process.* Londres: Springer.

Schwaber, K., & Sutherland, J. (2017). *La Guía de Scrum.* Scrumguides. https://www.Scrumguides.org/docs/Scrumguide/v2017

Scrum Alliance. (2019). *State of Scrum 2017-2018.* Scrum Alliance. https://www.Scrumalliance.org/learn-about-Scrum/state-of-Scrum

SCRUMstudy. (2016). *Una guía para el conocimiento de SCRUM (SBOK).* Estados Unidos: SCRUMstudy.

Sommerville, I. (2005). *Ingeniería del software.* Madrid: Pearson Eduación, S.A.

Sutherland, J. (2020). *Scrum: Manual del campo.* Oceano.

Glosario

Agilista: Quien propone “metodologías ágiles” como alternativas a las anteriores metodologías formales consideradas más rígidas y dependientes de las planificaciones detalladas.

Aplicabilidad: Cualidad de aplicable.

Auto organizado: Calco del inglés “Self-organization”. Término que no registran los diccionarios en español, pero que puede definirse como “orden espontánea”. Se refiere a un proceso de orden que no depende de ningún factor externo y que puede ser debido a un proceso químico, biológico, físico, robótico o incluso cognitivo.

Backlog: una lista ordenada de todo el trabajo pendiente. Dependiendo del método ágil empleado, los elementos incluidos en el *backlog* se denominan *ítems*, historias de usuario, unidades de trabajo, etcétera.

Código: instrucciones que debe seguir un programa durante su ejecución.

Heurística: Del griego εὑρίσκειν / heurískein: 'hallar', 'inventar' y –́tico. 1. adj. Perteneciente o relativo a la heurística. 2. f. Técnica de la indagación y del descubrimiento. 3. f. Búsqueda o investigación de documentos o fuentes históricas. 4. f. En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas,
Incremental: Calco del inglés. Que ocurre de forma gradual, en una serie de pequeños agregados.

Ítem: Cada uno de los elementos que forman parte de un conjunto de datos.

Iteración: Del latín *iteratio, -ōnis.* Acción y efecto de iterar. Repetir.

Software: Voz inglesa. Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

Sprint: cada uno de los ciclos o iteraciones que se van a tener dentro de un proyecto Scrum. Permite tener un ritmo de trabajo con un tiempo prefijado.

Startup: organización humana (por lo regular empresa incipiente) con gran capacidad de adaptabilidad, que desarrolla productos o servicios de gran innovación, altamente deseados o requeridos por el mercado, donde su diseño y comercialización están orientados completamente al cliente. Buscan resolver problemas prácticos o sociales con base, sobre todo, en nuevas tecnologías y el uso del *big data.*

Stakeholder: persona interesada o impactada por el resultado de un proyecto.

Tecnología de la información (TI)*:* aplicación de ordenadores y equipos de telecomunicación para almacenar, recuperar, transmitir y manipular datos, con frecuencia usados en el contexto de los negocios u otras empresas.

Anexo 1. Cuestionario para entrevistas

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO DE LA FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
MAESTRÍA EN INFORMÁTICA ADMINISTRATIVA

El presente cuestionario tiene como objetivo identificar los elementos y prácticas de la metodología Scrum que han permitido a diversos equipos de trabajo terminar proyectos de desarrollo de software de manera eficiente, como parte de una investigación que tiene por finalidad, el contribuir a generar propuestas de mejora en la implementación de la metodología.

El cuestionario consta de 42 preguntas, el tiempo estimado de respuesta es de 30 minutos. Sus respuestas serán de carácter anónimo, por lo que le pedimos contestar con la mayor sinceridad posible. Muchas gracias.

Al inicio de cada apartado encontrará instrucciones, léalas cuidadosamente.

Datos Generales

Sexo: ☐ Femenino ☐ Masculino

Edad: _____

Especialidad técnica: _____

Apartado 1. Relación con Scrum

Instrucciones: Le pedimos que sus respuestas se enfoquen a su experiencia en el uso de SCRUM.

1. ¿Qué rol de Scrum ejecutó en este proyecto?
    a. Scrum Master (SM)
    b. Product Owner (PO)

c. Development Team (DT)

2. ¿Cuánto tiempo lleva ejecutando ese rol dentro de la organización (y en otras organizaciones)?

   _____

3. ¿Cómo se le capacitó en Scrum?

   _____

4. ¿Se encuentra certificado en Scrum?
   a. Sí Mencione el organismo certificador _____
   b. No
5. ¿Cuál es el mayor desafío al que se ha enfrentado al implementar Scrum?

   _____

Apartado 2. Inicio del proyecto.

Instrucciones:

- Le pedimos que sus respuestas se enfoquen a su experiencia en un proyecto de desarrollo de software en el que haya implementado Scrum y que se encuentre concluido.
- Lea cuidadosamente cada una de las preguntas, seleccione la opción que representa la alternativa más apropiada según su criterio.

6. Proporcione la siguiente información sobre el proyecto de desarrollo de software**.**
   - Tamaño del equipo Scrum: _____
   - ¿Cómo se determinó el tamaño del equipo?: _____ _____ _____

- Duración del proyecto: _____
- Tipo de desarrollo: Nuevo ☐ Modificación ☐ Actualización
- Total de Sprints en el proyecto: _____

7. A lo largo del proyecto ¿Existieron los roles de Scrum Master (SM) y Product Owner (PO)?
    a. Sí, ambos roles
    b. Solo SM
    c. Solo PO
8. ¿Cómo se asignó al SM y al PO?
    a. Disponibilidad del personal
    b. Experiencia del personal
    c. Otro, explique: _____
9. ¿Qué técnicas o herramientas se emplearon para conocer el proyecto a desarrollar?
    a. Entrevistas
    b. Cuestionario
    c. Formatos diseñados por la organización
10. ¿Qué medios se utilizaron para registrar la información obtenida?

    _____
11. ¿Cómo se compartió con el equipo el objetivo del proyecto?
    a. Reunión de inicio de proyecto
    b. No se compartió
    c. Otra
12. ¿Qué otras áreas de la organización encargada del desarrollo estuvieron involucradas en el desarrollo del proyecto?

    _____

13. ¿Con base en qué se determinó el número de Sprints a ejecutar en el proyecto?
    a. Estimación de épicas
    b. Experiencia en proyectos similares.
    c. Otro, descríbalo: _____

Apartado 3. Artefactos y Ceremonias.

14. ¿Qué roles participaron en la Sprint Planning? Puede seleccionar más de uno.
    ☐ SM ☐ PO ☐ Development Team

15. ¿Cómo se refinaron las historias de usuario del proyecto?
    a. Caso de prueba
    b. Reglas de negocio
    c. Otro, descríbalo: _____
16. ¿Con base en qué se eligieron las historias de usuario a desarrollar en el Sprint?
    a. Máxima prioridad para el negocio
    b. Deuda técnica
    c. Otro, descríbala _____
17. ¿Se contó con las definiciones de terminado y los criterios de aceptación para las historias de usuario a desarrollar en el sprint?
    a. Si
    b. No
18. Enliste ¿con base en qué parámetros se realizó la asignación de tareas? _____
19. ¿Quién asignó las tareas?

a. Development team
b. SM
c. PO

20. A partir de su experiencia usando Scrum ¿cuál es la estructura ideal para realizar la Sprint Planning de manera efectiva?

_____

21. ¿Las tareas se desarrollaron en el tiempo planeado?
    a. Si, se cumplieron en tiempo.
    b. No.
22. En caso de que la respuesta a la pregunta anterior sea “No”, ¿cuál fue el porcentaje de avance?

_____

23. Retomando la respuesta de la pregunta anterior, ¿Cuál o cuáles fueron los problemas que surgieron para la ejecución de tareas en tiempo?

_____

24. ¿Qué herramientas se utilizaron para dar seguimiento a las tareas?

_____

25. ¿Se llevaron a cabo las reuniones diarias del equipo Scrum?
    a. Sí
    b. No
26. ¿Se respetó el tiempo definido de 15 minutos para las reuniones diarias del equipo Scrum?
    a. Sí
    b. No
27. ¿Qué tipo de documentación se desarrolló en la reunión diaria?
    a. Ninguna
    b. Minuta

c. Otra, descríbala:

_____

28. A partir de su experiencia usando Scrum ¿cuál es la estructura ideal para realizar la Reunión Diaria de manera efectiva?

_____

_____

29. En las Revisiones del *Sprint* ¿se involucró al cliente?
    a. Si
    b. No
    c. Algunas veces
30. ¿Se logró el objetivo en cada Sprint?
    a. Sí
    b. No
    c. No en todos
31. En los Sprints que no se logró el objetivo ¿cuáles fueron los problemas que se presentaron?

_____

32. A partir de su experiencia usando Scrum ¿cuál es la estructura ideal para realizar la Sprint Review de manera efectiva?

_____

_____

33. Al final de cada *Sprint* ¿se realizó la Reunión de Sprint Retrospective?
    a. Si
    b. No
    c. No en todos
34. En la Reunión de Sprint Retrospective ¿qué roles participaron?

    ☐ SM ☐ PO ☐ Development Team

35. A partir de su experiencia usando Scrum ¿cuál es la estructura ideal para realizar la Sprint Retrospective de manera efectiva?

_____

_____

Apartado 4. Escalado de SCRUM.

Instrucciones:
Conteste este apartado en caso de que en el proyecto seleccionado para responder este cuestionario haya trabajado más de un equipo, usando SCRUM para su desarrollo.

36. ¿Qué otros roles se involucraron para apoyar el trabajo en paralelo de los equipos?

    _____

    _____

37. ¿Con que frecuencia se llevó a cabo la sincronización de equipos?

    _____

    _____

38. A partir de su experiencia ¿Qué recomendaciones generales daría para poder trabajar con equipos en paralelo?

    _____

    _____

Apartado 5. Cierre de proyecto.

39. ¿Cómo se hizo el cierre del proyecto?
    a. Ceremonia de cierre
    b. Envío de entregables
    c. Otro. Describa

    _____

40. ¿Qué entregables se proporcionaron al cliente?

_____

_____

41. ¿Qué elementos o indicadores se consideraron para determinar el cumplimiento del objetivo del proyecto?

_____

_____

42. ¿Cómo se incorporaron las lecciones aprendidas del proyecto a la base de conocimiento de la empresa en la que se desarrolló el proyecto?

_____

_____

Anexo 2. Resultados de entrevistas

Para facilitar el entendimiento de las respuestas obtenidas en las entrevistas, se realizaron gráficas de aquellas que fue posible organizar y contabilizar.

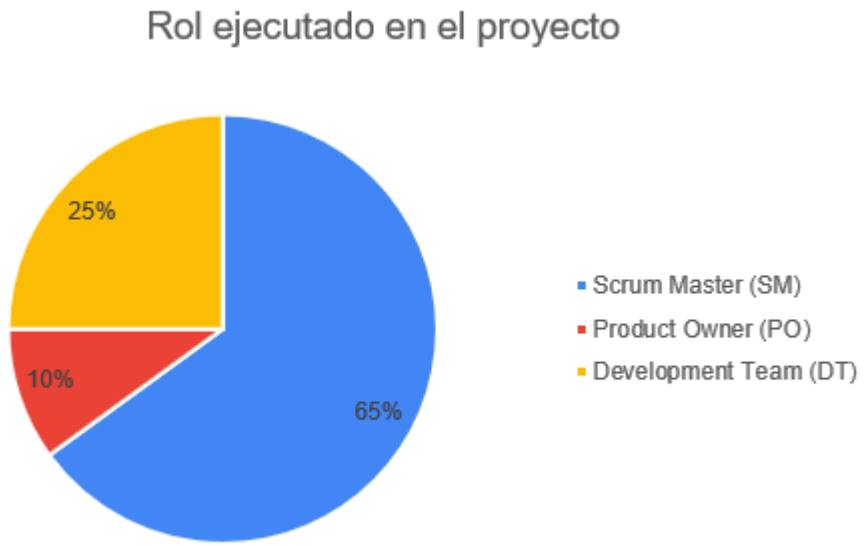

Gráfico 1. Rol ejecutado en el proyecto

Gráfico 2. Tiempo ejecutando el rol

Gráfico 3. ¿Se encuentra certificado en Scrum?

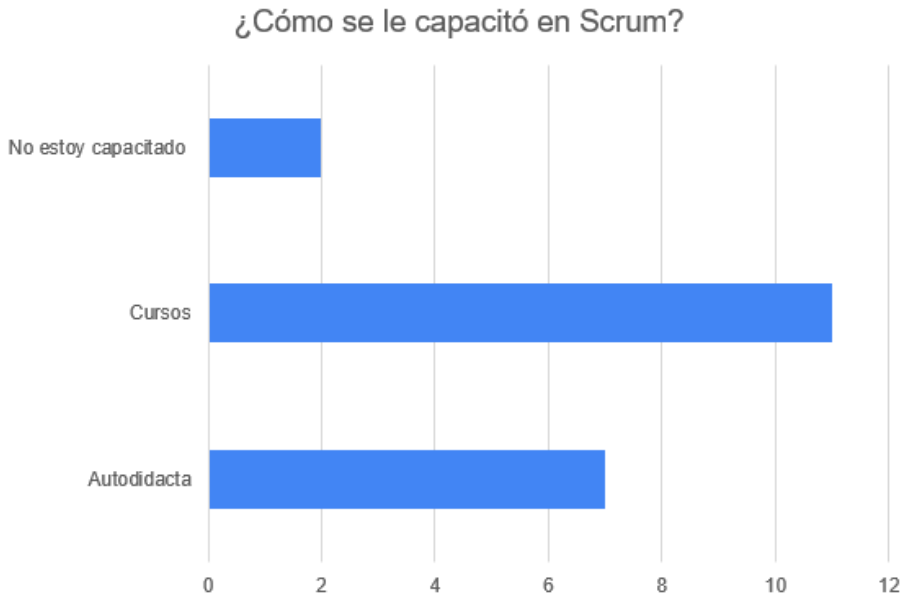

Gráfico 4. ¿Cómo se le capacitó en Scrum?

Gráfico 5. ¿Con qué organización se le certificó?

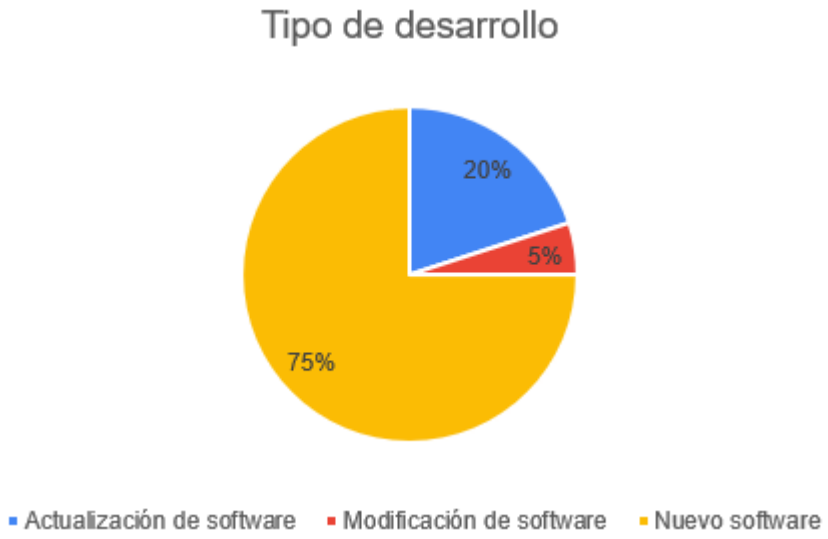

Gráfico 6. Tipo de desarrollo

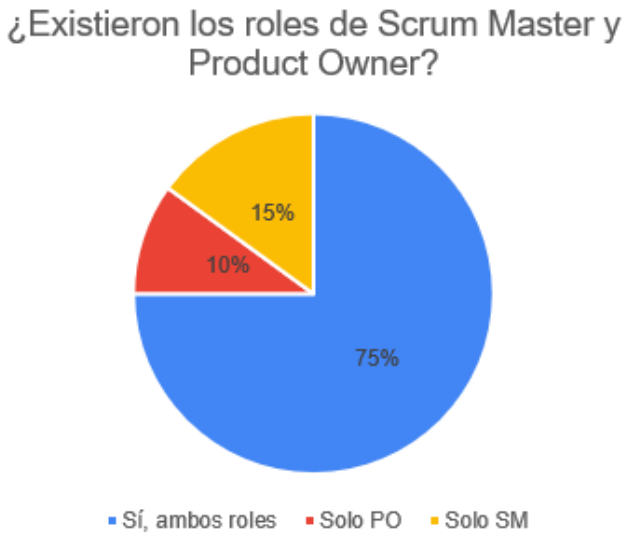

Gráfico 7. Roles involucrados

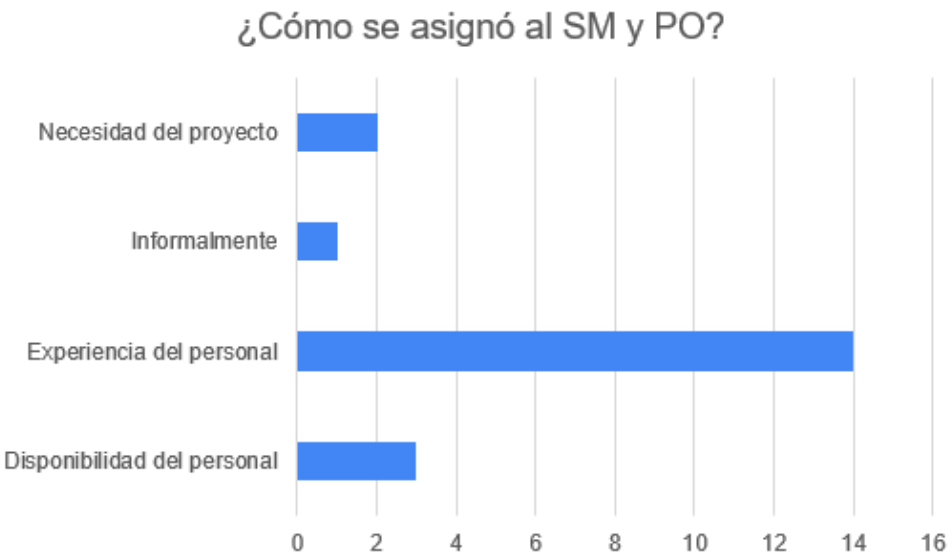

Gráfico 8. ¿Cómo se asignó al SM y al PO?

Gráfico 9. Técnicas o herramientas usadas para conocer el proyecto.

Gráfico 10. ¿Con base en qué se determinó el número de sprints a ejecutar en el proyecto?

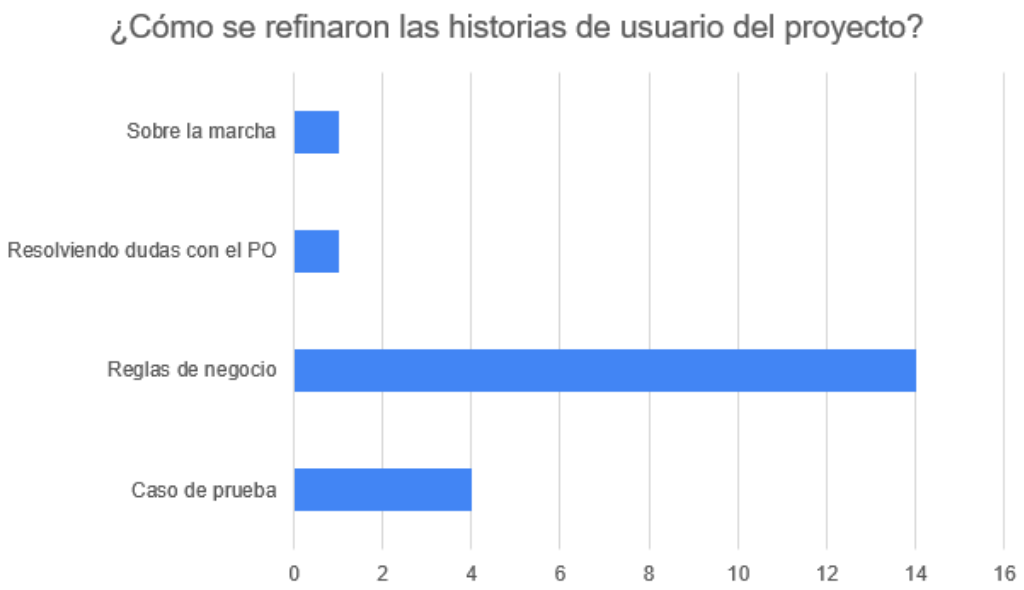

Gráfico 11. ¿Cómo se refinaron las historias de usuario del proyecto?

Gráfico 12. Roles que participaron en la planeación.

Gráfico 13. ¿Con base en qué se eligieron las historias de usuario a desarrollar en el Sprint?

Gráfico 14. ¿Se contó con las definiciones de terminado y los criterios de aceptación para las historias de usuario a desarrollar en el Sprint?

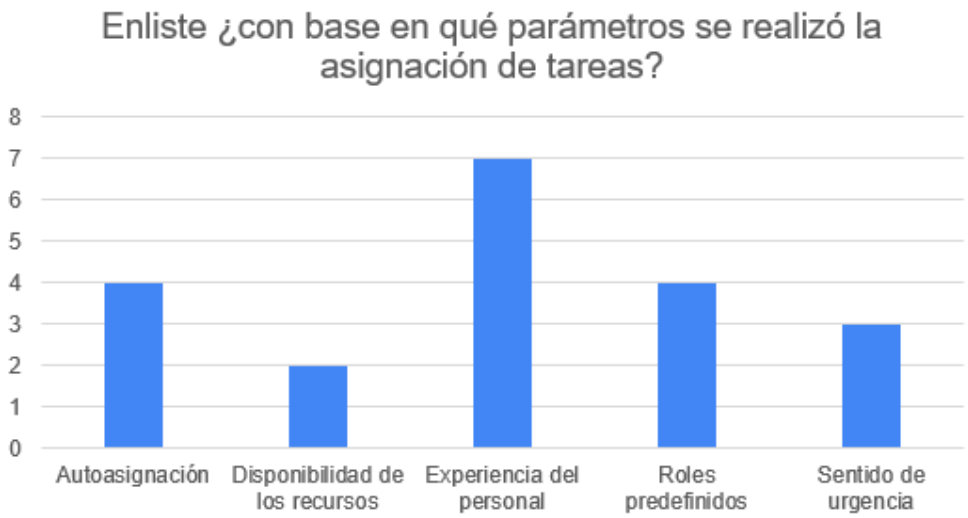

Gráfico 15. Parámetros para asignación de tareas.

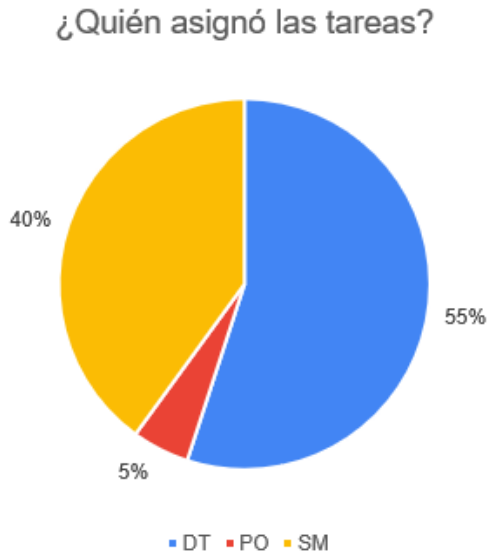

Gráfico 16. ¿Quién asignó las tareas?

Gráfico 17. ¿Las tareas se desarrollaron en el tiempo planeado?

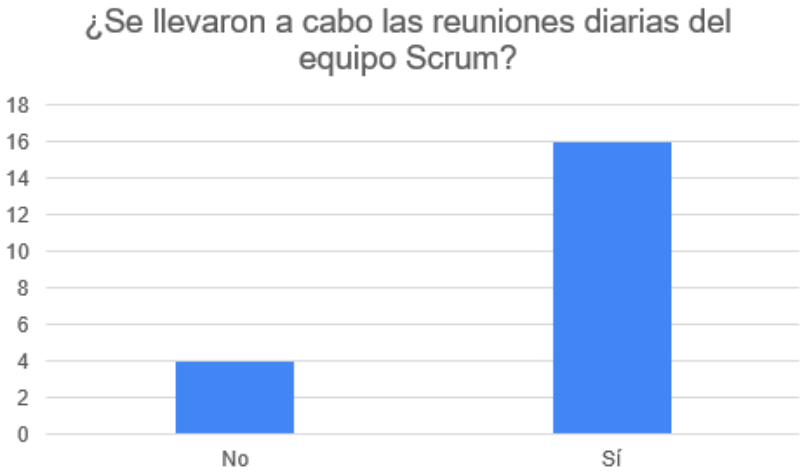

Gráfico 18. ¿Se llevaron a cabo las reuniones diarias del equipo Scrum?

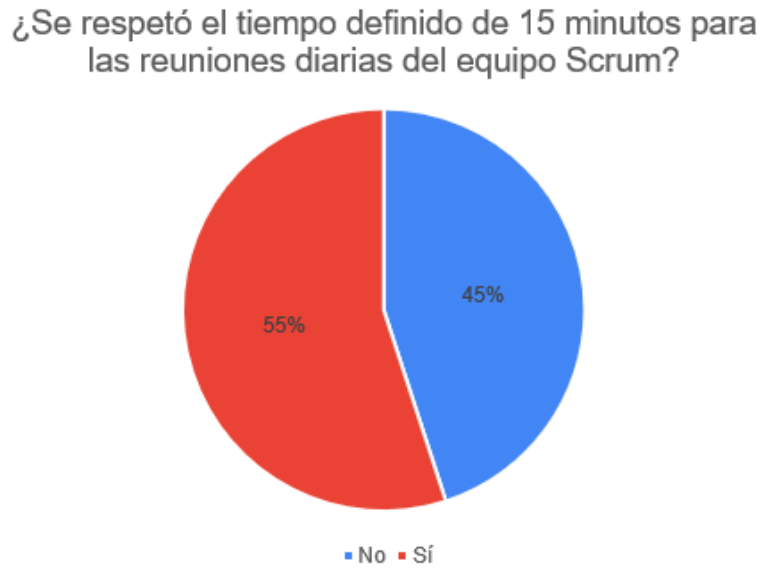

Gráfico 19. ¿Se respetó el tiempo definido de 15 minutos para las reuniones diarias del equipo Scrum?

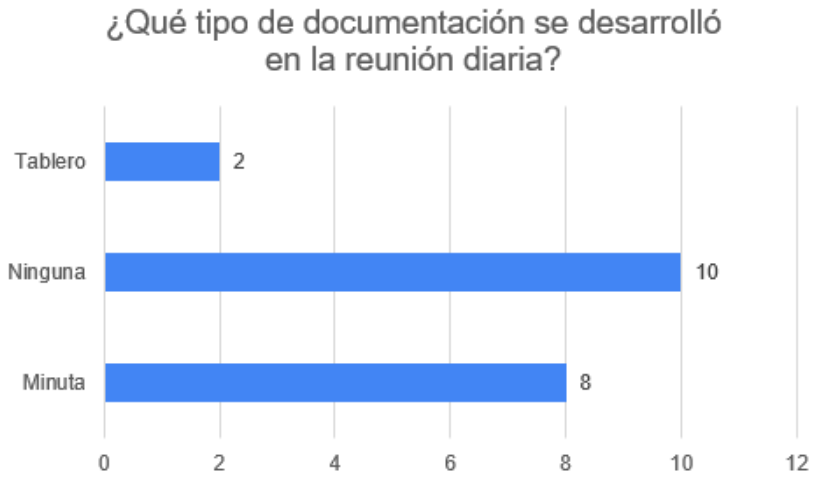

Gráfico 20. Documentación generada en la reunión diaria.

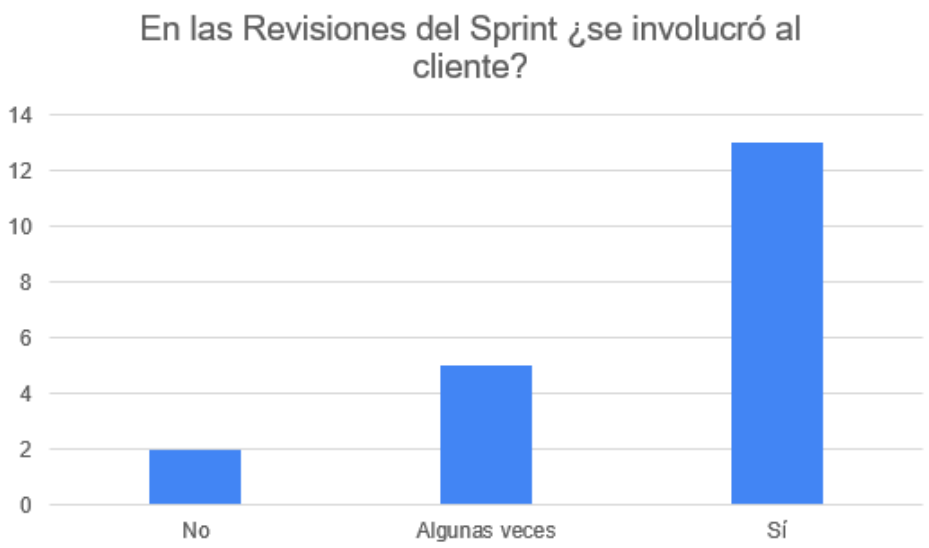

Gráfico 21. En las revisiones del Sprint ¿se involucró al cliente?

Gráfico 22. ¿Se logró el objetivo en cada Sprint?

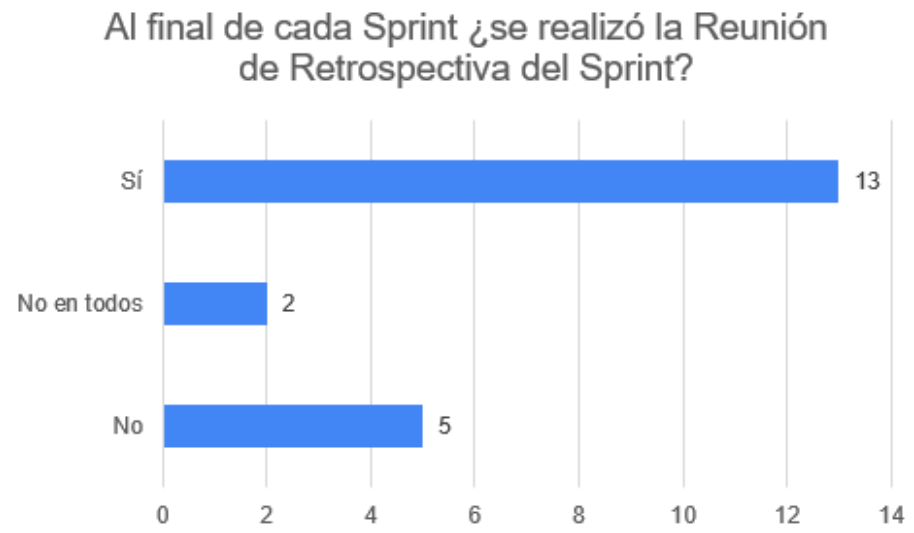

Gráfico 23. Al final de cada Sprint ¿se realizó la reunión de Sprint Retrospective?

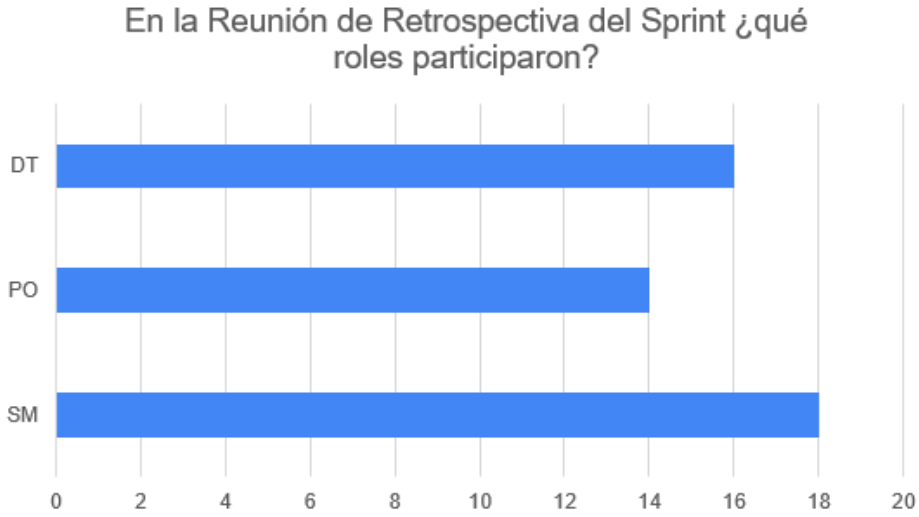

Gráfico 24. En la reunión de Sprint Retrospective ¿qué roles participaron?

Gráfico 25. ¿Cómo se hizo el cierre del proyecto?