UNIVERSIDAD DON BOSCO

DIRECCIÓN DE EDUCACIÓN A DISTANCIA

TRABAJO DE GRADUACIÓN:

IMPLEMENTACIÓN DE UN MODELO DE CICLO DE VIDA ÁGILE PARA PROYECTOS DE TI BASADOS EN DATA SCIENCE

PARA OPTAR AL GRADO DE:

MAESTRO EN ARQUITECTURA DE SOFTWARE

AUTORES:

ING. JOVEL BARRERA, ROBERTO ERNESTO JB 090575

LIC. TOCHE HERNÁNDEZ, RODOLFO FRANCISCO TH 203232

ASESOR**:**

MG. DAVID ARTURO MARTÍNEZ

ANTIGUO CUSCATLÁN, LA LIBERTAD, EL SALVADOR, CENTRO AMÉRICA

MAYO, 2023

Rector Universidad Don Bosco

Dr. Mario Rafael Olmos

Secretaria General

Inga. Yesenia Xiomara Martínez Oviedo

Director de Educación a Distancia

Mg. Eduardo Menjívar Valencia

Coordinador de la Maestría

Mg. Mauricio Orlando Figueroa Chicas

Asesor del proyecto de graduación

Mg. David Arturo Martínez

Lector/a del proyecto de graduación

Mg. Joshua Villavicencio

Agradecimientos

En primer lugar, quiero agradecer a mi esposa Gisella por su apoyo y comprensión durante los años dedicados a esta maestría.

A todos los maestros, por compartir sus experiencias y conocimientos durante la maestría en especial a Mauricio Flores a quien tuve un gran aprecio.

A Francisco mi compañero en este trabajo de graduación un gran profesional y de quien he aprendido mucho.

También agradecer a nuestro asesor, Mg David Martínez, por el seguimiento apoyo y tiempo dedicado a acompañar nuestro trabajo.

Y a todas las personas que de una u otra manera estuvieron involucradas en esta maestría y en la elaboración de nuestro trabajo de graduación y con su acciones e ideas nos permitieron llegar hasta aquí. Muchas gracias.

Roberto Jovel

Tomé la decisión de comenzar esta maestría justo durante la pandemia 2020 para continuar en este camino del aprendizaje y porque durante el encierro comprendí que debemos aprovechar las oportunidades de la vida y no detenernos, estudiar es una de ellas.

Agradezco a mi familia por su apoyo incondicional mi mama, mi pareja y hermanas; a mi pequeña compañera, mi gatita Matilda, que me acompañó muchas veces hasta que terminaba mis tareas de la maestría, a mi país El Salvador que por razones del crecimiento personal tuve que dejar y salir de sus fronteras.

Los proyectos son más fáciles en equipo, agradezco a Roberto compañero en la elaboración de esta investigación y durante cursos en el desarrollo de la maestría, sin duda su apoyo, conocimiento y liderazgo hicieron el camino más fácil.

Desde la primera asignatura hasta el proceso de graduación la universidad siempre puso a disposición personas expertas en las diferentes áreas que nos formaron hasta este punto, en particular quiero agradecer a Mg David Martínez por su tiempo, aporte en conocimiento y disponibilidad para culminar la parte final.

Finalmente, doy gracias a nuestro ser supremo por la fortaleza durante todo este tiempo.

Resumen

En esta investigación se estudia la eficacia o beneficios de implementar un ciclo de vida ágil a proyectos de Data Science, para ayudar a las compañías o departamentos de tecnologías de la información a mejorar los procesos de administración de proyectos de esta especialidad.

La investigación realizo una comparación entre los marcos de agiles más conocidos diseñados para Data Science entre los que destacan Data Driven Scrum, CRISP-DM y TDSP y un análisis de prácticas como el MLOps para generar un ciclo de vida. Para medir los beneficios al aplicarla se tomaron en cuenta tres componentes:

- Satisfacción del cliente,
- Calidad de la solución y
- Bienestar del equipo de trabajo

Estos fueron medidos antes y después de implementar mejoras en el flujo de trabajo basados en el ciclo de vida en dos equipos reales de una compañía salvadoreña que desarrolla proyectos de Data Science.

La estrategia de implementación básicamente fue proveer al equipo los mecanismos de funcionamiento de este ciclo de vida, conociendo previamente la situación actual de cómo se han administrado sus proyectos hasta la fecha.

Finalmente se concluye que la aplicación de prácticas agiles a proyectos de Data Science impacta de manera positiva a los equipos en cuanto a su efectividad y abre los espacios para que adaptar sus formas de trabajo para que estas mejoras sean continuas durante todo el proyecto.

Índice General

Índice de Tablas

Índice de Figuras

Capitulo 1: Introducción

1.1 Antecedentes

Actualmente los datos y como están siendo empleados están revolucionando la forma en que vivimos, trabajamos y pensamos, Modelos de Datos, Inteligencia Artificial, Machine Learning y en general todas estas tecnologías que convergen en el Data Science están provocando cambios culturales, tecnológicos y académicos. Como resultado, se producen cambios en los paradigmas de una gran cantidad de disciplinas.

La adquisición, el almacenamiento y la computación de datos que antes planteaban enorme reto para el avance de estas tecnologías ya no son más un obstáculo, por lo que las diferencias entre el conocimiento tradicional y el obtenido con Data Science es cada vez más notorio en varias disciplinas y presenta ventajas para las organizaciones que toman ventaja de ello.

Las teorías tradicionales no abordan los problemas de los datos, por tal razón expertos en diversos campos empiezan a mostrar un gran interés por la aplicación de estas tecnologías.

La Ciencias de la Computación no son la excepción y dada la relación compatible que pudiera apreciarse entre el desarrollo de software y el desarrollo de las Modelos de Data Science el primer punto de exploración de estas nuevas tecnologías en las organizaciones será a través de sus departamentos de TI o proveedores de Software.

Las organizaciones y profesionales de estas ramas deben ser capaces de responder efectivamente a estas demandas si buscan mantener pretenden seguir siendo competitivos en este nuevo mundo.

1.2 Problemática

El desarrollo de soluciones de software y en particular aquellas innovadoras basadas en tecnologías nuevas para nuestros mercados regionales como el Data Science presentan el reto de ser problemas de dominio complejo cuando utilizamos el framework Cynefin de toma de decisiones para clasificarlos (Kurtz & D, 2003) esto significa que no existe una forma definitiva para resolverlos y que se requiere una aproximación experimental para lograrlo.

Por un lado, a nivel de gestión, típicamente al inicio del proyecto se desconoce las soluciones, y lo más probable es que cambien los requisitos del producto, el trabajo se debe modularizar y es necesaria una estrecha colaboración y una rápida retroalimentación por parte de los usuarios finales.

A nivel técnico el Data Science combina matemáticas y estadísticas, programación especializada, análisis avanzado, inteligencia artificial (IA) y Machine Learning (ML) para descubrir información procesable oculta en los datos de que se pueden utilizar para guiar la toma de decisiones y la planificación estratégica.

Por tal razón el ciclo de vida de esta clase de proyectos requiere de roles, herramientas y procesos que difieren en gran medida a los que podrían utilizarse normalmente en los proyectos de TI tradicionales.

Esto plantea un reto para los Arquitectos de Software y niveles gerenciales quienes para asegurar el mayor éxito para sus proyectos deberán de tener en cuenta en este nuevo paradigma sus capacidades y las necesidades particulares de sus clientes, al ser estas tecnologías aún emergentes.

1.3 Justificación

Es de prioridad estratégica para toda compañía exitosa el mantener altos niveles de satisfacción de clientes, de calidad en sus soluciones y de bienestar de sus colaboradores.

Los proyectos de TI basados en Ciencia de Datos tienen particularidades que los hacen diferentes de los proyectos de desarrollo tradicionales. Mientras que en el desarrollo de software tradicional se tienen varios años perfeccionando las prácticas de desarrollo, herramientas y los modelos de negocio en el caso de los proyectos Data Science es diferente ya que estas tecnologías son relativamente nuevas y los productos derivados de ella aún están muchos en fases experimentales. Suelen ser de naturaleza no determinista, con dependencias de la información contenida en los datos que podrían no conocerse al iniciar el proyecto.

La implementación de un ciclo de vida exitoso que retome las particularidades de un proyecto de TI basados en ciencia de datos; deberá aplicar prácticas ágiles que servirán para analizar los requerimientos y planificar eficientemente el proceso de solución permitiendo a las compañías y equipos tener mejor conocimiento sobre las soluciones, trabajando en conjunto la mejora continua y alcanzando estándares de calidad.

Un ciclo de vida agile para proyectos de Data Science podrá identificar problemas, limitaciones, necesidades y oportunidades dentro de la organización y reaccionar a ellos en etapas tempranas impactando de manera positiva en la satisfacción de sus clientes, la calidad de sus soluciones y el bienestar de sus colaboradores.

1.4 Supuestos

Para lograr los objetivos que la investigación pretende y para que el conocimiento generado pueda ser aplicado por otros equipos se consideran los siguientes supuestos:

- Los equipos de trabajo desarrollan proyectos de Data Science y profesionales dedicados a esta área.
- La organización activamente busca mejores formas de llevar a cabo sus proyectos de Data Science y esta propuesta de ciclo de vida esta alineada con sus objetivos.
- La organización está abierta a experimentar nuevas prácticas y brinda un espacio seguro a los equipos para cometer errores y validar su aprendizaje.
- Los equipos en los que se implementa el ciclo de vida tienen un sólido conocimiento del agilismo y aplican sus principios en su día a día, cuentan con profesionales de la agilidad que entienden estas formas de trabajo, tienen influencia en la organización y son guía.
- En concordancia con lo anterior, la implementación de este ciclo de vida no implica para los equipos de trabajo un cambio drástico en su forma de trabajo ni para los procesos de negocio de la organización.

1.5 Alcance

La investigación se limita a proponer un ciclo de vida agile para proyectos basados en Data Science que sea congruente y adaptable a la realidad de cada uno de los equipos de trabajo en los que se realiza el estudio.

Se realiza un acompañamiento para comprender el ciclo de vida los principios que lo sustentan y apoyar esta adaptación a través de Retrospectivas con el equipo. Sin embargo, cada equipo es responsable de la implementación de este ciclo de vida de manera que haga sentido a sus realidades necesidades.

La medición del impacto del ciclo de vida apoyará de la herramienta *Scrum Team Survey* aplicada en dos momentos tal como se describe en el Capítulo 3 y se enfocará en los componentes de Satisfacción del Cliente, Calidad de las Soluciones y Bienestar de los Equipos de Trabajo. Quedan fuera del alcance de la investigación la tabulación y cálculo de los resultados y se limita a presentarlos tal cual como la herramienta los entrega.

1.6 Objetivos

1.6.1 General

Producir un modelo de ciclo de vida para la gestión de proyectos TI basados en Data Science, que contemple principios e instrumentos ágiles, mejores prácticas y prácticas emergentes conocidas hasta el momento en el área de Data Science y medir el impacto de sus beneficios al implementarlo junto a equipos en proyectos de este tipo.

1.6.2 Específicos

- Proponer un Ciclo de Vida enfocado que tome en cuenta las particularidades de los proyectos de Data Science y los incorpore de manera efectiva el desarrollo agile de productos de software.
- Adaptar, manteniendo los principios de agilidad, el Ciclo de Vida a las necesidades de los equipos en los proyectos que se implementará y obtener retroalimentación de estos.
- Medir el impacto de un Ciclo de Vida agile en proyectos de TI basados en Data Science en cuanto a la Satisfacción del Cliente, la Calidad de las Soluciones y Bienestar de los Equipos de Trabajo.

Capítulo 2: Marco Teórico

2.1 Introducción proyectos de Data Science

En la actualidad compañías salvadoreñas utilizan la información recolectada de sus operaciones para construir recursos estadísticos que les permitan tomar decisiones que potencian su desarrollo en el mercado, mantener la posición alcanzada e incluso posibles aperturas de negocios. Esto ha empujado a las áreas de tecnología de algunas empresas que incluyan en sus proyectos un área de la tecnología conocida como Data Science.

Los proyectos de Data Science, y proyectos agiles en general, incluyen conceptos que deben ser comprendidos dentro de su contexto los cuales se describen a continuación:

2.1.1 Stakeholders

Edward Freeman en su teoría de los Stakeholders define este concepto como aquellos individuos o grupos de individuos que pueden verse afectados por las actividades de la empresa y que, a su vez, pueden ellos afectar a la propia empresa con sus acciones. (Freeman, 2008)

Básicamente el nombre Stakeholders designa todos los que tienen el interés o la porción en el proyecto. Ellos pueden ser los dirigentes directos o, por ejemplo, las personas que financian el proyecto.

Sean proyectos de TI tradicionales o de Data Science los Stakeholders están siempre presentes y la relación en que los equipos se relacionan con estos es fundamental para el éxito del proyecto.

2.1.2 ¿Qué son los datos?

Los datos se crean constantemente por diferentes recursos tecnológicos como teléfonos móviles, redes sociales, tecnologías de imagen para determinar un diagnóstico médico, sistemas de facturación; todo esto y más crean nuevos datos, y deben almacenarse en algún lugar para algún propósito. Los dispositivos y sensores generan automáticamente información de diagnóstico que debe ser almacenado y procesado en tiempo real. El almacenamiento, así como la estructura de datos son importantes para la transformación de lotes de datos para identificar patrones significativos y extraer información útil.

Tres atributos que definen una estructura de datos masivos (Big Data):

- Gran volumen de datos: en lugar de miles o millones de filas, pueden ser miles de millones de filas y millones de columnas.

- Complejidad de los tipos y estructuras de datos: Big Data refleja la variedad de nuevas fuentes, formatos y estructuras de datos, incluidos los rastros digitales que se dejan en la web y otros repositorios digitales para su posterior análisis.
- Velocidad de creación y crecimiento de nuevos datos: Big Data puede describir datos de alta velocidad, con una ingesta de datos rápida y un análisis casi en tiempo real.

2.1.3 Estructura de Datos

Los datos masivos pueden presentarse en múltiples formas, incluidos datos estructurados y no estructurados, como datos financieros, archivos de texto, archivos multimedia y mapeos genéticos. A diferencia de gran parte del análisis de datos tradicional realizado por las organizaciones, la mayor parte de Big Data es de naturaleza no estructurada o semiestructurada, lo que requiere diferentes técnicas y herramientas para procesar y analizar. Los entornos informáticos distribuidos y las arquitecturas de procesamiento paralelo masivo (MPP) que permiten la ingesta y el análisis de datos en paralelo son el enfoque preferido para procesar datos muy complejos.

La siguiente figura muestra cuatro tipos de estructuras de datos, con un 80-90 % del crecimiento futuro de datos proveniente de tipos de datos no estructurados. Aunque diferentes, los cuatro se mezclan comúnmente. Por ejemplo, un sistema de administración de bases de datos relacionales (RDBMS) clásico puede almacenar registros de llamadas para un centro de llamadas de soporte de software. El RDBMS puede almacenar características de las llamadas de soporte como datos estructurados típicos, con atributos como marcas de tiempo, tipo de máquina, tipo de problema y sistema operativo. Además, es probable que el sistema tenga datos no estructurados, cuasi o semiestructurados, como información de registro de llamadas de forma libre tomada de un ticket de correo electrónico del problema, historial de chat del cliente o transcripción de una llamada telefónica que describe el problema técnico y la solución o archivo de audio de la conversación telefónica. Se podrían extraer muchos conocimientos de los datos no estructurados, casi o semiestructurados en los datos del centro de llamadas.

Figura 1. Niveles de Estructuración de Datos

Si bien el análisis de datos estructurados tiende a ser la técnica más familiar, se requiere una técnica diferente para enfrentar los desafíos de analizar datos semiestructurados (que se

muestran como XML), casi estructurados (que se muestran como un flujo de clics) y datos no estructurados.

Los cuatro tipos principales de estructuras de datos:

- **Datos estructurados:** datos que contienen un tipo, formato y estructura de datos definidos (es decir, datos de transacciones, cubos de datos de procesamiento analítico en línea [OLAP], RDBMS tradicional, archivos CSV e incluso hojas de cálculo simples).
- **Datos semiestructurados:** archivos de datos textuales con un patrón discernible que permite el análisis (como los archivos de datos del lenguaje de marcado extensible [XML] que se describen por sí mismos y están definidos por un esquema XML).
- **Datos casi estructurados:** datos textuales con formatos de datos erráticos que se pueden formatear con esfuerzo, herramientas y tiempo (por ejemplo, datos de flujo de clics web que pueden contener inconsistencias en los valores de datos y formatos).
- **Datos no estructurados**: datos que no tienen una estructura inherente, que pueden incluir documentos de texto, PDF, imágenes y videos.

2.1.4 Repositorios de Datos

La introducción de hojas de cálculo permitió a los usuarios comerciales crear una lógica simple sobre datos estructurados en filas y columnas y crear sus propios análisis de problemas comerciales. No se requiere capacitación del administrador de la base de datos para crear hojas de cálculo: se pueden configurar para hacer muchas cosas de manera rápida e independiente de los grupos de tecnología de la información (TI).

Las hojas de cálculo son fáciles de compartir y los usuarios finales tienen control sobre la lógica involucrada. Sin embargo, su proliferación puede dar lugar a "muchas versiones de la verdad". En otras palabras, puede ser un desafío determinar si un usuario en particular tiene la versión más relevante de una hoja de cálculo, con los datos y la lógica más actualizados. Además, si se pierde una computadora portátil o se corrompe un archivo, los datos y la lógica dentro de la hoja de cálculo podrían perderse. Este es un desafío continuo porque los programas de hojas de cálculo como Microsoft Excel todavía se ejecutan en muchas computadoras en todo el mundo. Con la proliferación de islas de datos (o spreadmarts), la necesidad de centralizar los datos es más apremiante que nunca.

A medida que crecían las necesidades de datos, también lo hacían las soluciones de almacenamiento de datos más escalables. Estas tecnologías permitieron que los datos se administraran de forma centralizada, brindando beneficios de seguridad, conmutación por error y un repositorio único donde los usuarios podían confiar en obtener una fuente de datos "oficial" para informes financieros u otras tareas de misión crítica. Esta estructura también permitió la creación de cubos OLAP y herramientas analíticas de BI, que proporcionaron acceso rápido a un conjunto de dimensiones dentro de un RDBMS. Las características más avanzadas permitieron el desempeño de técnicas analíticas en profundidad, como regresiones y redes neuronales. Los almacenes de datos empresariales (EDW, por sus siglas en inglés) son

fundamentales para la generación de informes y tareas y resuelven muchos de los problemas que presenta la proliferación de hojas de cálculo, como cuál de las múltiples versiones de una hoja de cálculo es la correcta. Los EDW, y una buena estrategia 81, proporcionan fuentes de datos directas de fuentes que se administran, respaldan y protegen de forma centralizada.

A pesar de los beneficios de EDW y BI, estos sistemas tienden a restringir la flexibilidad necesaria para realizar análisis de datos sólidos o exploratorios. Con el modelo EDW, los datos son administrados y controlados por grupos de TI y administradores de bases de datos (D8A), y los analistas de datos deben depender de TI para el acceso y los cambios en los esquemas de datos. Esto impone plazos de entrega más largos para que los analistas obtengan datos; la mayor parte del tiempo se dedica a esperar aprobaciones en lugar de comenzar un trabajo significativo. Además, muchas veces las reglas de EDW impiden que los analistas construyan conjuntos de datos. En consecuencia, es común que surjan sistemas adicionales que contengan datos críticos para construir conjuntos de datos analíticos, administrados localmente por usuarios avanzados. A los grupos de TI generalmente no les gusta la existencia de fuentes de datos fuera de su control porque, a diferencia de un EDW, estos conjuntos de datos no están administrados, protegidos ni respaldados. Desde la perspectiva de un analista, EDW y BI resuelven problemas relacionados con la precisión de los datos y disponibilidad. Sin embargo, EDW y BI introducen nuevos problemas relacionados con la flexibilidad y la agilidad, que eran menos pronunciados cuando se trataba de hojas de cálculo.

Una solución a este problema es el sandbox analítico, que intenta resolver el conflicto de los analistas y científicos de datos con EDW y datos corporativos administrados de manera más formal. En este modelo, el grupo de TI aún puede administrar los entornos limitados analíticos, pero se diseñarán a propósito para permitir un análisis sólido, mientras se administran y protegen de manera centralizada. Estos sandboxes, a menudo denominados espacios de trabajo están diseñados para permitir que los equipos exploren muchos conjuntos de datos de forma controlada y no suelen utilizarse para informes financieros de nivel empresarial ni paneles de ventas.

Muchas veces, los sandbox analíticos permiten la computación de alto rendimiento mediante el procesamiento en la base de datos; el análisis se produce dentro de la propia base de datos. La idea es que el rendimiento del análisis sea mejor si se ejecuta en la propia base de datos, en lugar de llevar los datos a una herramienta analítica que reside en otro lugar. Análisis en la base de datos, respecto a tecnología y herramientas donde se crea relaciones con múltiples fuentes de datos dentro de una organización y ahorra el tiempo dedicado a crear estas fuentes de datos de forma individual. El procesamiento en la base de datos para análisis profundos permite un tiempo de respuesta más rápido para desarrollar y ejecutar nuevos modelos analíticos, al mismo tiempo que reduce, aunque no elimina, el costo asociado con los datos almacenados en sistemas de archivos "sombra" locales. Además, en lugar de los típicos datos estructurados en el EDW, los sandbox analíticos pueden albergar una mayor variedad de datos, como datos sin procesar, datos textuales y otros tipos de datos no estructurados, sin interferir con las bases de datos de producción críticas.

Repositorio de Datos	Características
El analista depende de los extractos de datos.	Hojas de cálculo y bases de datos de bajo volumen para el mantenimiento de registros.
	El analista depende de los extractos de datos.
Data Warehouse	Contenedores de datos centralizados en un espacio especialmente diseñado.
	Admite BI e informes, pero restringe los análisis sólidos.
	Analista dependiente de TI y DBA para acceso a datos y cambios de esquema.
	Los analistas deben dedicar mucho tiempo a obtener extractos de datos agregados y desagregados de múltiples fuentes.
Sandbox Analítico (workspaces)	Activos de datos recopilados de múltiples fuentes y tecnologías para su análisis.
	Permite un análisis flexible y de alto rendimiento en un entorno que no sea de producción; puede aprovechar el procesamiento en la base de datos.
	Reduce los costos y los riesgos asociados con la replicación de datos en sistemas de archivos "sombra".
	"Propiedad del analista" en lugar de "Propiedad del DBA"
Nota: Tipos de repositorios de datos, desde una perspectiva analítica	

Tabla 1. Características de los repositorios de datos

Hay varias cosas a considerar con los proyectos de Big Data Analytics para garantizar que el enfoque se ajuste a los objetivos deseados. Debido a las características de Big Data, estos proyectos se prestan al soporte de decisiones para la toma de decisiones estratégicas de alto valor con alta complejidad de procesamiento. Las técnicas analíticas utilizadas en este contexto deben ser iterativas y flexibles, debido al alto volumen de datos y su complejidad.

Realizar un análisis rápido y complejo requiere conexiones de red de alto rendimiento y tener en cuenta la cantidad aceptable de latencia. Estas consideraciones requieren un enfoque diferente para pensar en los desafíos analíticos, que se explorarán más a fondo en la siguiente sección.

2.1.5 Estado de la practica en el análisis

Los problemas comerciales actuales brindan muchas oportunidades para que las organizaciones se vuelvan más analíticas y basadas en datos, como se muestra en la Tabla 1·2

Línea de negocio	Ejemplo
Optimizar las operaciones comerciales	Identificar el riesgo empresarial.
Predecir nuevas oportunidades de negocio	Cumplir con las leyes o los requisitos reglamentarios
Ventas, precios, rentabilidad, eficiencia	Pérdida de clientes, fraude, incumplimiento
Ventas adicionales, ventas cruzadas, mejores prospectos de nuevos clientes	Anti-Lavado de Dinero, Fair Lending

Tabla 2. Líneas de negocio para análisis avanzados

La tabla anterior describe cuatro categorías de problemas comerciales comunes con los que se enfrentan las organizaciones en las que tienen la oportunidad de aprovechar los análisis avanzados para crear una ventaja competitiva. En lugar de solo realizar informes estándar sobre estas áreas, las organizaciones pueden aplicar técnicas analíticas avanzadas para optimizar los procesos y obtener más valor de estas tareas comunes. Los primeros tres ejemplos no representan nuevos problemas.

Las organizaciones han estado tratando de reducir la rotación de clientes, aumentar las ventas y realizar ventas cruzadas de clientes durante muchos años. Lo nuevo es la oportunidad de fusionar técnicas analíticas avanzadas con Big Data para producir análisis más impactantes para estos problemas tradicionales. El último ejemplo describe los requisitos regulatorios emergentes. Muchas leyes reglamentarias y de cumplimiento han existido durante décadas, pero cada año se agregan requisitos adicionales, lo que representa una complejidad adicional y requisitos de datos para las organizaciones. Las leyes relacionadas con el lavado de dinero (AML) y la prevención del fraude requieren técnicas analíticas avanzadas para cumplir y administrar adecuadamente

2.1.6 Arquitectura analítica actual

Como se describió anteriormente, los proyectos de Data Science necesitan espacios de trabajo diseñados específicamente para experimentar con datos, con arquitecturas de datos flexibles y ágiles. La mayoría de las organizaciones todavía tienen almacenes de datos que brindan un soporte excelente para los informes tradicionales y las actividades de análisis de datos simples, pero desafortunadamente tienen más dificultades para respaldar análisis más sólidos. Esta sección examina la arquitectura de datos analíticos atípica que puede existir dentro de una organización.

La siguiente figura muestra una arquitectura de datos típica y varios de los desafíos que presenta para los científicos de datos y otras personas que intentan realizar análisis avanzados. Esta sección examina el flujo de datos al científico de datos y cómo este individuo encaja en el proceso de obtención de datos para analizar en proyectos.

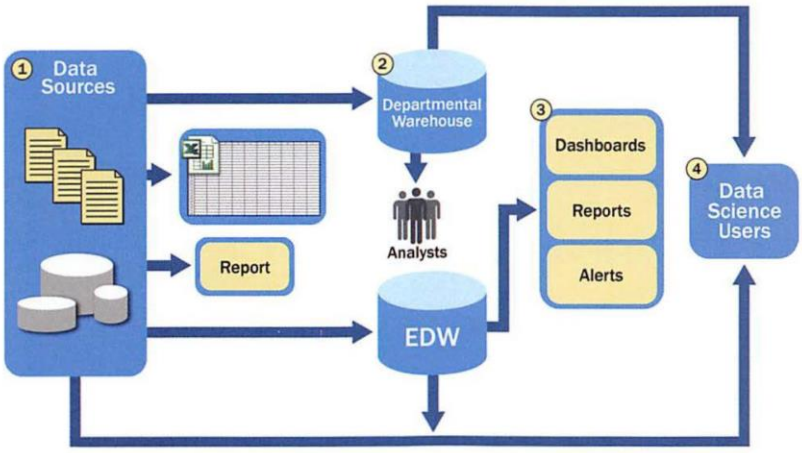

Figura 2. Arquitectura analítica típica

Para que las fuentes de datos se carguen en el almacén de datos, los datos deben comprenderse, estructurarse y normalizarse bien con las definiciones de tipo de datos adecuadas. Si bien este tipo de centralización permite la seguridad, la copia de seguridad y la conmutación por error de datos muy críticos, también significa que los datos generalmente deben pasar por un preprocesamiento y puntos de control significativos antes de que puedan ingresar a este tipo de entorno controlado, que no se presta a la exploración de datos y el análisis iterativo.

Como resultado de este nivel de control sobre el EDW, pueden surgir sistemas locales adicionales en forma de almacenes departamentales y datasmarts locales que los usuarios comerciales crean para adaptarse a su necesidad de análisis flexible. Estos datasmarts locales pueden no tener las mismas restricciones de seguridad y estructura que el EDW principal y permiten a los usuarios realizar un nivel de análisis más profundo. Sin embargo, estos sistemas únicos residen de forma aislada, a menudo no están sincronizados ni integrados con otros almacenes de datos y es posible que no se les realice una copia de seguridad.

Una vez en el almacén de datos, los datos son leídos por aplicaciones adicionales en toda la empresa para Bl y fines de generación de informes. Estos son procesos operativos de alta prioridad que obtienen fuentes de datos críticos de los almacenes y repositorios de datos.

Al final de este flujo de trabajo, los analistas obtienen datos para sus análisis posteriores. Debido a que los usuarios generalmente no pueden ejecutar análisis personalizados o intensivos en bases de datos de producción, los analistas crean extractos de datos del EDW para analizar datos fuera de línea en R u otras herramientas analíticas locales. Muchas veces, estas herramientas se limitan a análisis en memoria en escritorios que analizan muestras de datos, en lugar de la población completa de un conjunto de datos. Debido a que estos análisis se basan en extractos de datos, residen en una ubicación separada, y los resultados del análisis, y cualquier

información sobre la calidad de los datos o las anomalías, rara vez se retroalimentan al depósito de datos principal.

2.1.7 Machine Learning

El Machine Learning (ML) o aprendizaje automático es el estudio científico de algoritmos y modelos estadísticos que utilizan los sistemas informáticos para realizar una tarea específica sin estar programados explícitamente. Los algoritmos de aprendizaje se encuentran en muchas aplicaciones que utilizamos a diario. Cada vez que se utiliza un motor de búsqueda web como Google para buscar en Internet, una de las razones por las que funciona tan bien es porque un algoritmo de aprendizaje ha aprendido a clasificar las páginas web.

Estos algoritmos se utilizan para diversos fines, como la extracción de datos, el procesamiento de imágenes, el análisis predictivo, etc., por nombrar algunos. La principal ventaja de usar el aprendizaje automático es que, una vez que un algoritmo aprende qué hacer con los datos, puede hacer su trabajo automáticamente.

Para conocer el termino de Machine Learning debemos definir los modelos en la ciencia de datos que son la especificación de una relación matemática entre diferentes variables, por ejemplo, marketing en redes sociales pueden tomar un modelo en base a número de seguidores, las reacciones y como incrementan los ingresos en comparativa a publicaciones. De esta manera, Machine Learning se define como la creación y uso de modelos que se observan o analizan con los datos, que permite predecir diversos resultados para nuevos datos. (Grus, 2015)

2.1.8 Kanban

Kanban es un marco popular utilizado para implementar el desarrollo de software ágil y DevOps. Requiere una comunicación en tiempo real de la capacidad y total transparencia del trabajo. Los elementos de trabajo se representan visualmente en un tablero Kanban, lo que permite a los miembros del equipo ver el estado de cada trabajo en cualquier momento.

Kanban es una estrategia para optimizar el “Flow” o *flujo de valor* mediante un proceso que usa un sistema visual y con limitación del trabajo en progreso(work-in-progress). El concepto de Flow es esencial para la definición de Kanban. El Flow es el movimiento del valor a través del sistema de desarrollo de producto. Kanban optimiza el Flow mediante la mejora general de la eficiencia, efectividad y predictibilidad del proceso. (Scrum.org, 2021)

Kanban proporciona cuatro prácticas para alcanzar la optimización del flujo de valor:

- Visualización del Flujo de Trabajo
- Límite del Trabajo en Progreso (WIP)
- Gestión activa de los ítems de trabajo en progreso
- Inspección y adaptación la Definición de Flujo de Trabajo del equipo

Kanban y Scrum comparten algunos de los mismos conceptos, pero tienen enfoques muy diferentes. No deben confundirse entre sí.

	Scrum	Kanban
Cadencia	Sprints regulares de longitud fija (ejemplo 2 semanas)	Flujo continuo
Metodología de entrega	Al final de cada sprint si es aprobado por el Product Owner	Entrega continua o a discreción del equipo
Roles	Product Owner, Scrum Master, Developers	No hay roles existentes. Algunos equipos solicitan la ayuda de un entrenador ágil.
Métricas clave	Velocity	Cycle Time
Enfoque con respecto al cambio	Los equipos deben esforzarse por no hacer cambios en el pronóstico del sprint durante el sprint. Hacerlo compromete los aprendizajes en torno a la estimación.	El cambio puede ocurrir en cualquier momento

Tabla 3. Comparación Scrum vs Kanban

Kanban es una de las metodologías de desarrollo de software más populares adoptadas por los equipos ágiles hoy en día. Kanban ofrece varias ventajas adicionales para la planificación de tareas y el rendimiento para equipos de todos los tamaños.

Un equipo Kanban solo se centra en el trabajo que está activamente en curso. Una vez el equipo completa un elemento de trabajo, arrancan con el siguiente elemento de trabajo de la parte superior del Backlog, un Product Owner es libre de volver a priorizar el trabajo en el backlog sin interrumpir al equipo, porque cualquier cambio fuera de los elementos de trabajo actuales no afecta al equipo. Siempre y cuando este mantenga los elementos de trabajo más importantes en la parte superior del backlog, el equipo de desarrollo está seguro de que está entregando el máximo valor al negocio. Así que no hay necesidad de las iteraciones de longitud fija como en Scrum.

2.2 Introducción al agilismo y metodologías agiles

Agile es una metodología de desarrollo de software para construir un software de forma incremental utilizando iteraciones cortas de 1 a 4 semanas para que el proceso de desarrollo esté alineado con las necesidades cambiantes del negocio. En lugar de un desarrollo de un solo paso de 6 a 18 meses donde todos los requisitos y riesgos se predicen por adelantado, Agile adopta un proceso de retroalimentación frecuente donde se entrega un producto viable después de una iteración de 1 a 4 semanas.

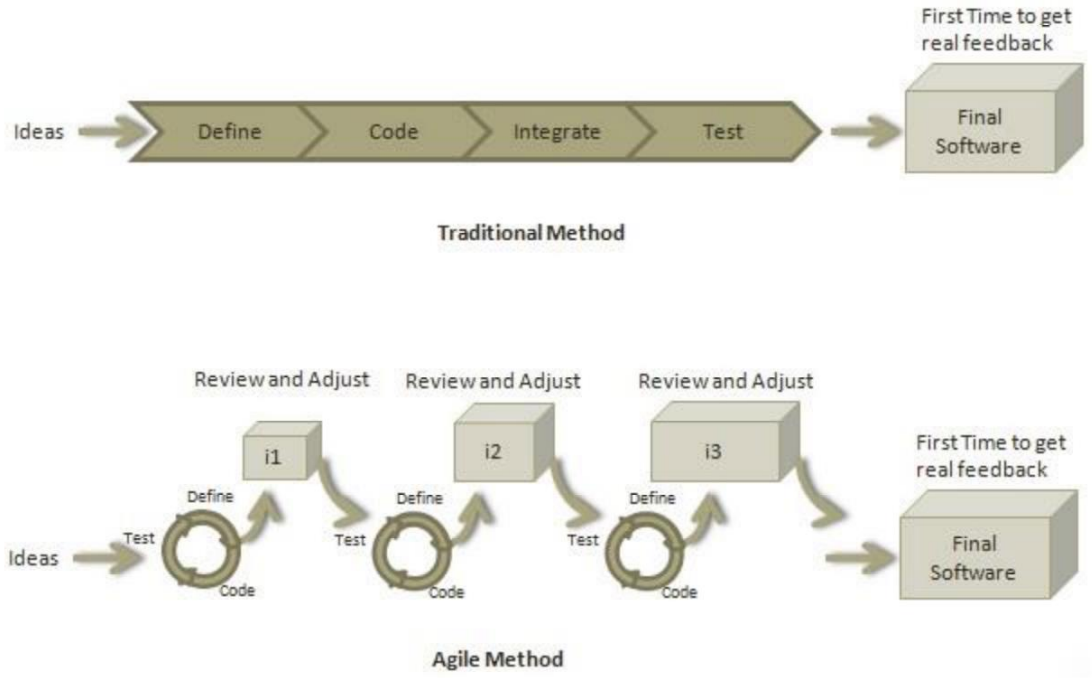

Figura 3. Desarrollo de Software en metodología ágil vs metodología tradicional

La figura anterior muestra la diferencia entre el desarrollo de proyectos bajo el método tradicional conocido de avance progresivo, cascada, y por otro lado controlado por una metodología ágil. Como se mencionó anteriormente el agilismo adopta un método donde el producto se parte en piezas listas para pruebas y que generan un valor agregado a la solución final; a medida que las piezas se van terminando estas son ensambladas y entregadas a los dueños del producto para obtener la versión del software final. Cada pieza lleva ciclo de definición, codificación y pruebas que al final son revisadas y ajustadas a los criterios de aceptación definidos por el negocio.

2.2.1 Como un equipo ágil planea su trabajo

Un equipo Agile trabaja en iteraciones para entregar historias de usuarios donde cada iteración es de 10 a 15 días. Cada historia de usuario se planifica en función de la priorización y el tamaño de su cartera de pedidos. El equipo usa su capacidad (cuántas horas están disponibles con el equipo para trabajar en tareas) para decidir cuánto alcance tienen para planificar.

Figura 4. Planificación de iteraciones

Cada iteración es planificada separando la pieza en historias de usuarios que contienen los diferentes criterios de aceptación de la pieza a revisar. La historia de usuario es asignada a un desarrollador el cual deberá analizar la definición entregada, hacer el diseño y codificarlo. La última etapa de cada historia de usuario es el proceso de pruebas en el cual el encargado de calidad es asignado para corroborar que cumple con lo definido por el Product Owner. La Figura anterior muestra una iteración de dos semanas la cual incluye la planificación, ejecución y revisión.

2.2.2 Manifiesto Ágil

El surgimiento de este manifiesto tuvo origen a principios del 2001 a raíz de los inconvenientes encontrados hasta ese tiempo relacionado con calidad del producto que los equipos de desarrollos de software entregaban, se elaboró un documento que aporto una mejora a los ciclos de desarrollo de software. Hoy en día existen una variedad de frameworks basados en la metodología ágil, pero todos buscan la misma finalidad convertir piezas o elementos tangibles que sean útiles, sin embargo, es necesario considerar los doce principios del manifiesto:

1. Nuestra mayor prioridad es satisfacer al cliente a través de la entrega temprana y continua de software valioso.
2. Los requisitos cambiantes son bienvenidos, incluso tarde desarrollo. Los procesos ágiles aprovechan el cambio para la ventaja competitiva del cliente.
3. Entregue software que funcione con frecuencia, desde un par de semanas a un par de meses, de preferencia una escala de tiempo más corta.

4. Los empresarios y los desarrolladores deben trabajar juntos diariamente a lo largo del proyecto.
5. Construir proyectos en torno a personas motivadas.
6. Bríndeles el entorno y el apoyo que necesitan, y confiar en ellos para hacer el trabajo.
7. El método más eficiente y eficaz de transmitir información hacia y dentro de un equipo de desarrollo es una conversación cara a cara.
8. El software que funciona es la medida principal del progreso.
9. Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben poder mantener un ritmo constante indefinidamente.
10. Atención continua a la excelencia técnica y un buen diseño mejora la agilidad.
11. Simplicidad: el arte de maximizar la cantidad del trabajo no hecho--es esencial.
12. Las mejores arquitecturas, requisitos y diseños. surgen de equipos autoorganizados.
13. A intervalos regulares, el equipo reflexiona sobre cómo para volverse más efectivo, luego afina y ajusta su comportamiento en consecuencia.

2.3 Scrum Framework

El Marco de Trabajo Scrum fue desarrollado por Ken Schwaber y Jeff Sutherland en la década de 1990. Se formalizó en 1995 para abordar la complejidad inherente al desarrollo de productos y de software. Recientemente, el Marco de Trabajo Scrum está siendo aplicado con éxito a problemas complejos en una amplia variedad de áreas. Desde el marketing al cambio organizacional. Y de la investigación científica al desarrollo de software. El Marco de Trabajo Scrum, está construido sobre tres pilares que permiten el control del proceso empírico.

2.3.1 Pilares del Empirismo

El Marco de Trabajo Scrum (Scrum org, 2020) está construido sobre tres pilares que permiten el control del proceso empírico:

- Transparencia: Se recopilan datos, como métricas, feedback, y otras experiencias, para descubrir lo qué ocurre;
- Inspección: Se examina el progreso con todos los involucrados y se descubre el significado que tiene eso para la visión del producto;
- Adaptación: Se realizan cambios que permitan acercarse más a la visión del producto;

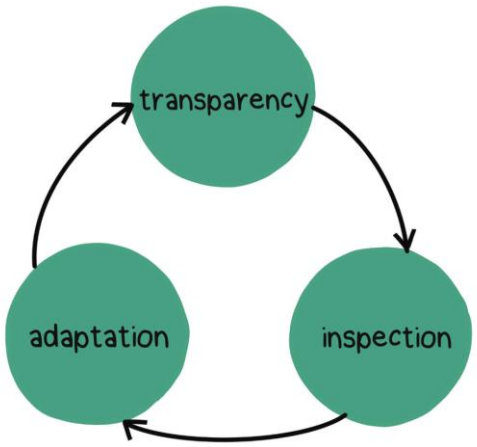

Figura 5. Pilares del Empirismo.

Este ciclo se repite tanto como sea necesario para detectar desviaciones, hallazgos inesperados y oportunidades potenciales que surjan mientras se realiza el trabajo. Esto no ocurre una vez al año o cuando se haya terminado el proyecto, sino de forma continua: diaria, semanal o mensualmente. En lugar de tomar decisiones basadas en suposiciones sobre posibles futuros, las decisiones son tomadas con base a datos recogidos hasta el momento. Esto es el empirismo.

2.3.2 Scrum: Un Marco de Trabajo para el Control del Proceso Empírico

Poner en práctica, de manera tangible y concreta la transparencia, inspección y adaptación es lo que hace de Scrum un marco de trabajo; Scrum provee cinco eventos, que se repiten, para trabajar en tres artefactos, tres roles para dar apoyo, más varios principios y reglas que lo fusionan todo en un conjunto bien cohesionado.

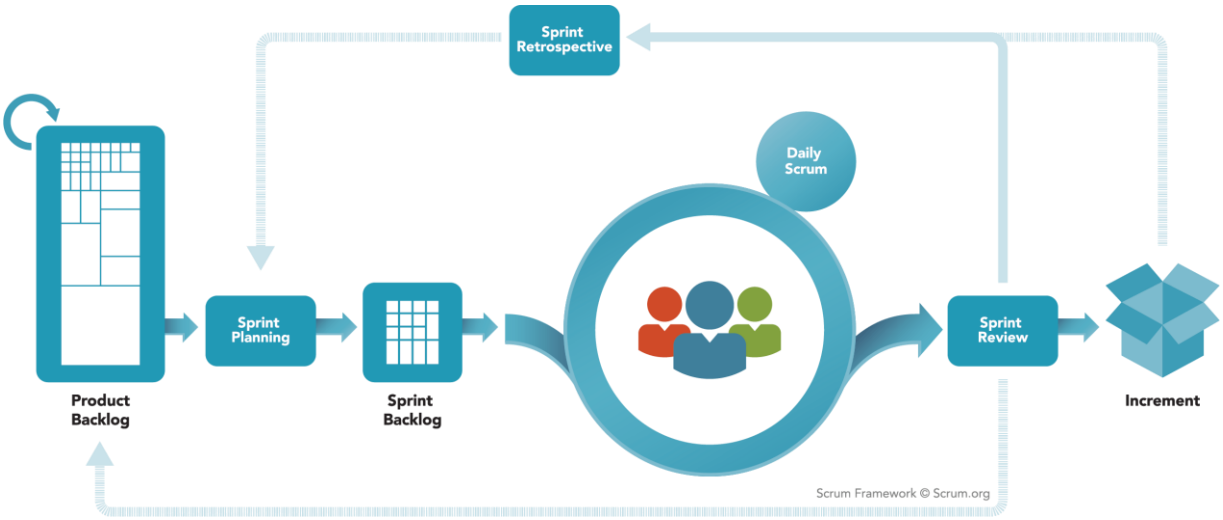

Figura 6. Scrum es un marco de trabajo para el Control del Proceso Empírico.

2.3.3 Artefactos de Scrum

El primer pilar del control del proceso empírico es la transparencia. Para inspeccionar el progreso de trabajo sobre un producto de forma frecuente y tomar decisiones sobre qué se necesita, es necesario tener algo que poder inspeccionar. Al hablar de hacer “transparente” el trabajo sobre un producto a lo que se hace referencia es a posibilitar que esté disponible de tal manera, que todas aquellas personas que tengan algún interés en él puedan verlo, para validar suposiciones y para generar nuevas ideas.

El Marco de Trabajo Scrum requiere de los equipos transparencia en cuanto a, al menos, tres elementos del trabajo sobre el producto. A estos elementos se les llama artefactos en Scrum, y son los medios principales para recoger los datos y experiencias necesarios para la toma de decisiones de cara al futuro.

Product Backlog

El primer artefacto es el Product Backlog. Permite dar visibilidad a todo el trabajo que aún queda pendiente sobre el producto. Cada idea, pregunta, hipótesis, característica, defecto o cualquier otra tarea está representada en el Product Backlog como un elemento individual. Algunos elementos serán muy amplios y poco claros mientras que otros serán pequeños y específicos. Todos los elementos se ordenan de acuerdo con su relevancia en relación con la visión del producto y sus stakeholders. A medida que el trabajo se va realizando, se producen continuos cambios en el Product Backlog para recoger nuevas perspectivas y oportunidades que surgen a lo largo del camino.

Un producto tiene un único Product Backlog, sin importar cuantos equipos estén trabajando en él. Si no, la transparencia se ve afectada, ya que no hay una única "fuente de la verdad" sobre el trabajo que se necesita hacer y el orden en el que se debe hacer.

Sprint Backlog

El segundo artefacto es el Sprint Backlog. Cada Sprint contempla un objetivo alineado a las metas del Producto. Este objetivo es una breve explicación de lo que el equipo planea lograr durante el transcurso de un sprint y se conoce como *Sprint Goal*. El Sprint Backlog consiste en la selección de elementos que los Developers extraen del Product Backlog al inicio del Sprint y que considera que son necesarios para lograr su Sprint Goal. El Sprint Backlog hace transparente todo el trabajo que los Developers están realizando, o va a realizar, durante el Sprint actual. Cada Equipo Scrum tiene su propio Sprint Backlog, incluso cuando varios equipos están trabajando en un único producto. El Sprint Backlog no es estático y cambia conforme el equipo va aprendiendo durante el Sprint. El Equipo de Desarrollo, en estrecha colaboración con el Product Owner, añade o elimina elementos basándose en cuánto tiempo queda en el Sprint y la relevancia o necesidad de esos elementos para el Sprint Goal.

Incremento

El tercer artefacto es el Incremento. Representa la recopilación de todos los elementos del Product Backlog, como nuevas características, correcciones de errores y otros ajustes, que un equipo ha completado durante un Sprint.

El propósito del Incremento es validar las asunciones sobre el trabajo que se ha hecho hasta la fecha. Quienes tengan un interés en el producto pueden echarle un vistazo y determinar si cumple sus necesidades, si entienden cómo funciona y si satisface sus expectativas. El incremento es un impulsor de nuevas ideas, dado que interactuar con algo tangible permite descubrir nuevas posibilidades.

El Incremento es el medio principal por el que puede ejercitarse el control del proceso empírico aplicado a problemas complejos. Cada Incremento comienza con una suposición, una

hipótesis o una idea potencialmente valiosa de cómo un producto puede acercarse a su objetivo. Esto se refleja en el Sprint Goal. El trabajo necesario para lograr ese objetivo se refleja en el Sprint Backlog.

Si múltiples Equipos Scrum trabajan en un único producto, se aseguran de que su trabajo quede integrado en un único Incremento al final de cada Sprint.

2.3.4 Los Roles de Scrum

La colaboración entre profesionales competentes es esencial para la resolución de problemas complejos. Las soluciones creativas e innovadoras surgen con más facilidad cuando se reúnen las perspectivas de profesionales competentes. Para facilitar esta colaboración y reducir la complejidad en la comunicación y toma de decisiones, el Marco de Trabajo Scrum limita, de manera intencionada, los roles a tres. En lugar de roles en sentido jerárquico, donde uno tiene autoridad sobre los demás, cada uno de ellos representa una perspectiva diferente que debe incluirse para trabajar empíricamente. Los tres juntos, son suficientes para trabajar empíricamente en cualquier entorno. Ningún otro rol es necesario.

Product Owner

El Product Owner incluye la perspectiva del "qué" es valioso (y qué no lo es) en la visión del producto. A medida que el Equipo Scrum invierte tiempo y dinero trabajando en el producto, el Product Owner se asegura de que esa inversión devolverá valor a los stakeholders. Es importante una colaboración estrecha con las personas que tienen interés en el producto, así como con el Equipo de Desarrollo, para decidir qué es valioso y qué no.

Un producto tiene un Product Owner y un Product Backlog. Para mantener alta la velocidad en la toma de decisiones y que la adaptación pueda realizarse rápidamente, los Product Owners deben tener la máxima autoridad con respecto al producto. Ellos tienen la última palabra sobre la visión del producto, el contenido del Product Backlog y cómo gastar el presupuesto (o incluso establecerlo).

El Product Owner se asegura de que hay un Product Backlog, ordenado y disponible tanto para el Equipo Scrum como para los stakeholders. Esto no quiere decir que el Product Owner sea la única persona en el Equipo Scrum que lo gestione. A fin de maximizar el valor del trabajo hecho por los Developers cada Sprint, tiene sentido que el Product Owner trabaje de forma activa con este para escribir elementos, refinarlos y ordenarlos.

Developers

En la versión más reciente de la Guía Scrum 2020 (Scrum org, 2020) se elimina la figura de "Equipo de Desarrollo". Ahora hay un solo Scrum Team enfocado en el mismo objetivo, con tres diferentes conjuntos de responsabilidades: Product Owner, Scrum Master y Developers.

Los Developers incluyen la perspectiva del "cómo" hacer el trabajo necesario para materializar la visión de un producto y mantener alta su calidad. Los Developers son el grupo

de personas principalmente responsable de entregar un nuevo incremento de producto en cada Sprint, logrando el Sprint Goal. Debido a que el desarrollo de productos es un trabajo complejo, incluso el futuro de un único Sprint es difícil de predecir. Es muy probable que surjan cuestiones, desafíos y problemas que impidan al equipo entregar ese incremento. También es probable que surjan nuevas ideas durante el Sprint sobre lo que se debe incluir o dejar fuera del Incremento. Esta naturaleza impredecible de los Sprints determina tres requisitos de vital importancia en cuanto a "cómo" los Developers trabajan y se organizan.

El primer requisito es trabajar para minimizar sus dependencias con otras personas, departamentos y habilidades que no son parte de su equipo pero que son necesarias para entregar un incremento considerado como "terminado". Cada dependencia es algo sobre lo que tienen un control limitado y puede frenar o bloquear completamente su capacidad de entregar un Incremento terminado cada Sprint. Esto afecta negativamente a su capacidad para trabajar empíricamente. Las dependencias se pueden minimizar de diferentes formas, con la automatización (por ejemplo, para el despliegue y el testeo), el entrenamiento de nuevas habilidades o la incorporación de gente con las habilidades necesarias al Equipo de Desarrollo.

El segundo requisito es que el los Developers reduzcan los cuellos de botella derivados de cómo se distribuyen las habilidades dentro del mismo. Tener un tester es mejor que no tener ninguno en absoluto. Pero si el tester está atascado con el trabajo, todo el equipo ve mermada su velocidad. Lo mismo ocurre con otras habilidades, como el desarrollo de backend, el diseño y el análisis. El equipo trabaja duro para asegurarse de que uno o más de sus miembros sean capaces de llevar a cabo trabajos que normalmente no realizan. El objetivo aquí no es que todos sean capaces de hacer todos los trabajos con el mismo nivel de maestría, esto es obviamente poco realista e irrespetuoso con las habilidades que las personas llevan años desarrollando, sino asegurarse de que otros pueden ayudar o asumir otros trabajos según sea necesario. El Marco de Trabajo Scrum sintetiza esta idea afirmando que "Los Equipos de Desarrollo son multifuncionales".

El tercer requisito es que el Equipo de Desarrollo, a menudo, debe confiar en su inteligencia y creatividad para superar los muchos acontecimientos inesperados que están destinados a suceder durante un Sprint. Por eso los Equipos de Desarrollo deben autogestionar, en la medida de lo posible, decisiones sobre cómo trabajar y qué mejorar en cómo se hace ese trabajo. En equipos autogestionados, no hay un "jefe" designado que tome las decisiones por el equipo. Todas las personas que hacen el trabajo (es decir, el Equipo de Desarrollo) trabajan conjuntamente para tomar decisiones dentro de los límites del Marco de Trabajo Scrum.

Scrum Master

El Scrum Master aporta la perspectiva del control del proceso empírico y de la calidad, mediante los cuales la transparencia, inspección y adaptación toman forma dentro y fuera del Equipo Scrum. El Scrum Master está ahí para hacer que los elementos del Marco de Trabajo Scrum cobren vida en el equipo y de forma más amplia en la organización. Para conseguir esto, los Scrum Masters adoptan diferentes papeles, dependiendo de la situación en la que se encuentren:

Maestro: enseña y explica el propósito del Marco de Trabajo Scrum como un medio para trabajar empíricamente. Trabaja duro para asegurarse de que todos comprendan cómo los artefactos, eventos, roles y principios promueven el empirismo y la agilidad;

Facilitador: facilita los pilares del Marco de Trabajo Scrum para maximizar las oportunidades de transparencia, inspección y adaptación. Un ejemplo de esto es la facilitación de eventos Scrum, cuando el Equipo Scrum los solicite o requiera. Otro ejemplo es ayudar al Product Owner a encontrar y usar técnicas para gestionar el Product Backlog y a los stakeholders;

Eliminador de Impedimentos: el Scrum Master elimina (o ayuda a eliminar) obstáculos que impiden que los Developers logre sus Sprint Goals. El Scrum Master ayuda a los Developers a aumentar su capacidad de resolver problemas de manera autónoma. Esto es algo que los equipos tienen que aprender, y el Scrum Master les ayuda a hacerlo. Lo que puede considerarse un impedimento durante el primer Sprint, puede haberse convertido en un problema que el equipo puede resolver fácilmente por sí mismo durante un Sprint futuro;

Agente de Cambio: ayuda a eliminar impedimentos que bloquean el empirismo y la agilidad en los entornos de trabajo de los Equipos Scrum. Por ejemplo, algunas organizaciones separan los "pruebas de calidad" y "despliegues" en diferentes equipos o departamentos. O prácticas de recursos humanos que recompensan a los individuos, no a los equipos. La eliminación de estos impedimentos no es algo que un Scrum Master pueda hacer solo, por lo que generalmente trabajará junto a otros Scrum Masters, Product Owners, etc.

Entrenador y Mentor: entrenan al Equipo Scrum realizando "preguntas poderosas" abiertas. Son mentores de otros Scrum Masters y ayudan a los miembros del Equipo Scrum a encontrar mentores que tengan la experiencia y las habilidades para ayudarlos;

Los Scrum Masters son "Líderes Servidores". En lugar de dirigir la atención hacia sí mismos, ayudan a otros a ser tan efectivos como sea posible. No gestionan ni dirigen al equipo diciéndoles qué hacer o cómo hacerlo. Solo hay un escenario en el que los Scrum Masters deben mantener una posición fuerte y es aquel en el que las decisiones que se tomen impacten negativamente en el proceso empírico o en la seguridad del Equipo Scrum.

Scrum Team

Juntos, los tres roles representan a un Scrum Team. Ellos tienen plena autoridad para tomar todas las decisiones relevantes para su producto. El Product Owner decide cómo usar el presupuesto disponible para maximizar el valor que se entrega a los stakeholders. Los Developers deciden cómo construir ese producto y tiene todas las habilidades necesarias para hacerlo. Y el Scrum Master garantiza que esto se haga de manera que se maximice el empirismo. No hacen falta más roles.

Puede ser tentador agregar roles, reglas, prácticas y estructura al Marco de Trabajo Scrum, porque se crea que la organización es particular y diferente. Y aunque algunas situaciones pueden justificar este pensamiento, preservar la firmeza en mantener las cosas tan

simples como sea posible es la mejor manera de avanzar en entornos complejos. Esto también se aplica al escalado. Agregar más equipos añade complejidad. En estos casos puede explorarse formas de hacer más con menos personas.

2.3.5 Eventos de Scrum

Para poder la toma de decisiones es necesario dar sentido, de forma frecuente, al Product Backlog, al Sprint Backlog y al Incremento, contando para ello con todos los interesados. En esto consiste la "Inspección". Y se presenta de muchas formas. Es posible dar sentido al Product Backlog observándolo y obteniendo conclusiones sobre, por ejemplo, su tamaño, la prioridad de sus elementos, o que tan rápido puede entregase a los interesados. Se puede dar sentido al Incremento involucrando a los usuarios y validando las hipótesis del producto con ellos como, por ejemplo, si los usuarios entienden alguna nueva funcionalidad, o si esta funcionalidad satisface su necesidad. Y se da sentido al Sprint Backlog al asegurar su factibilidad y cuando refleja el plan para el Sprint.

Sea cual sea la forma en la que se presente, la inspección ofrece la mejor base para la toma de decisiones cuando es fundamentada en algo tangible y concreto. En el caso del desarrollo de productos, la inspección de una funcionalidad completa y desplegada es la mejor manera de realmente validar las suposiciones sobre esa funcionalidad y cómo se usa. Ver una presentación o una propuesta de diseño de una funcionalidad puede parecer útil; sin embargo, siempre se basará una gran cantidad de suposiciones (probablemente diferentes) sobre cómo funcionará cuando se llegue a implementar.

Con el objetivo de trabajar de forma empírica, el Marco de Trabajo Scrum propone que los equipos tengan al menos cinco momentos que se repiten, en los que la inspección tiene lugar. Cada uno de ellos tiene un límite máximo de tiempo (o time-box) y ofrece una perspectiva concreta del trabajo que se está haciendo. En su conjunto, ofrecen una imagen completa. Estos son los cinco Eventos de Scrum. Aunque son mencionados frecuentemente como "reuniones", no están pensados como reuniones en el sentido tradicional de la palabra. De hecho, cuando los Eventos de Scrum se llevan a cabo teniendo en cuenta su propósito, la necesidad de otras reuniones formales disminuye y se crean las condiciones adecuadas para que la colaboración y la coordinación fluyan de forma natural y surjan en los momentos precisos.

Sprint

El primer evento es la duración acotada (o time-box) del propio Sprint. El propósito de cada Sprint es intentar resolver una parte de un problema complejo. Esta solución parcial está representada por una versión incremental del producto llamada Incremento, que puede ser inspeccionada por todos para decidir qué debería pasar después. Este Incremento debería estar, como mínimo, en un estado que permita ser entregado a los stakeholders, después del Sprint, si el Product Owner decide hacerlo. Sería incluso mejor poder entregar estos incrementos a lo largo del Sprint, acelerando el aprendizaje. Aunque el éxito de los Sprints variará (dado que incluso un único Sprint conlleva trabajo complejo e impredecible) puede aprenderse mucho del producto en cada iteración.

Los Sprints deberían ser siempre de la misma longitud con el objetivo de crear cadencia y predictibilidad para el equipo y sus stakeholders. Cuando un Sprint es demasiado largo, se pierden valiosas oportunidades para validar suposiciones y para asegurar que el trabajo progrese en la dirección adecuada. A medida que la longitud de los Sprints aumenta, también lo hace el riesgo de malgastar tiempo creando algo incorrecto. El Marco de Trabajo Scrum no especifica la longitud de los Sprints, solo que deben durar menos de un mes. Es decisión del equipo determinar cuán rápido quieren y pueden aprender. De forma general, los Sprints deberían ser lo más cortos posible mientras permitan que los Developers entregue una nueva versión significativa del producto que se ajuste a la meta de un Sprint.

Mientras que el Sprint es una oportunidad con límite máximo de tiempo para explorar un área particular del problema complejo que supone el desarrollo de productos, los otros cuatro eventos representan oportunidades específicas dentro del Sprint que promueven la inspección y la adaptación.

Sprint Planning

Cada Sprint en Scrum comienza cuando el Equipo Scrum realiza un plan básico para dicho Sprint. Este es el Sprint Planning. La primera cuestión, y la más importante, es cuál va a ser la meta para el Sprint (Sprint Goal). Sin una meta, no hay un propósito claro que anime a los miembros a unir esfuerzos durante el Sprint. Una meta para un Sprint puede ser entregar un conjunto coherente de funcionalidades que resuelva un problema particular. Puede ser satisfacer una necesidad de un grupo de stakeholders. Puede ser una hipótesis o suposición crítica que el Product Owner quiere verificar con el Incremento resultante del Sprint.

Aunque el Product Owner hace bien al asistir al Sprint Planning con un objetivo en mente, el Equipo Scrum al completo trabaja para refinar este objetivo de forma que se convierta en una meta valiosa y factible dentro de los límites temporales del Sprint. Los Developers trabajan con el Product Owner para seleccionar del Product Backlog el trabajo que debería llevarse a cabo durante el Sprint para lograr esa meta, reordenando el Product Backlog según sea necesario. Esta selección se convierte en el Sprint Backlog. Dado que los Developers son

los encargados de hacer el trabajo, es el que posee la última palabra acerca de qué se incluye en el Sprint Backlog.

Para Sprints de un mes, el Sprint Planning no debe durar más de 8 horas. Cuanto más corto sea el Sprint menos tiempo debe durar. El Sprint Planning se completa cuando hay un esquema aproximado para el Sprint y un plan más detallado para el primer par de días. Este plan suele reflejarse en un desglose del trabajo para los primeros días del Sprint. A medida que el Sprint progresa, los Developers traducen ese plan aproximado, de acuerdo con el Sprint Goal y el Sprint Backlog, en un plan más concreto sobre cómo colaborar para llevarlo a cabo. Al menos una de las oportunidades para llevar a cabo la concreción colaborativa del plan inicial es la Daily Scrum.

Daily Scrum

La Daily Scrum tiene lugar cada 24 horas y permite a los Developers manejar la complejidad que cualquier Sprint implica. Los miembros del equipo evalúan el progreso de su trabajo hacia el Sprint Goal, progreso que se muestra de manera transparente en el Sprint Backlog, y se hacen los ajustes oportunos. Por ejemplo, pueden encontrarse problemas inesperados que requieran una colaboración estrecha o pueden descubrir que necesitan añadir más trabajo al Sprint Backlog para lograr el Sprint Goal. Pueden encontrar impedimentos; problemas que están bloqueando el logro del Sprint Goal y que están fuera de su alcance.

La Daily Scrum no debería durar más de 15 minutos. Es una reunión corta y de mínimos, en la que el equipo coordina el trabajo colaborativo para las siguientes 24 horas. Si más coordinación fuese necesaria, los Developers pueden llevarla a cabo durante el resto del día. El Marco de Trabajo Scrum no prescribe cómo hacer una Daily Scrum de forma efectiva, sino que anima los Developers a encontrar la receta que mejor funcione para ellos.

Sprint Review

La Sprint Review tiene lugar al final del Sprint. Su propósito es inspeccionar el trabajo que se ha hecho hasta la fecha y decidir cuáles son los siguientes pasos, con base en lo aprendido. La Sprint Review es, al menos, un momento en el que la gente que construye el producto y la gente que está interesada en él se reúnen e inspeccionan los resultados del Sprint. Teniendo en cuenta también las condiciones del mercado, los cambios organizacionales, el presupuesto y el cronograma, se deciden de manera conjunta los siguientes pasos.

La Sprint Review no debe durar más de 4 horas para un Sprint de un mes. Cuanto más corto sea el Sprint, más corta tiende a ser la Sprint Review. El resultado de la Sprint Review son una serie de ajustes del Product Backlog basados en lo aprendido. Esto puede suponer nuevas ideas que surjan durante la Sprint Review, fallos detectados, cambios en elementos existentes en el Product Backlog o la repriorización del propio Product Backlog.

En cierto sentido, la Sprint Review gira en torno a la pregunta: “basándonos en lo que hemos aprendido en este Sprint, ¿cuáles son los siguientes pasos a seguir?”. Su respuesta nos proporciona una información muy valiosa para el Sprint Planning y futuros Sprint Goals.

Realizar solo una presentación durante la Sprint Review sobre lo que los Developers han hecho no implica inspección. De hecho, inspeccionar el incremento significa probarlo juntos y dar feedback. Es más acertado referirse a las Sprint Reviews como “fiestas del feedback” en lugar de “demos”. Las Sprint Reviews no son la primera ocasión en la que el Product Owner ve qué ha hecho el Equipo de Desarrollo. La Sprint Review es, en realidad, el momento en el que el Product Owner y los stakeholders, que acuden invitados por este, inspeccionan el producto juntos.

Sprint Retrospective

La Sprint Retrospective ocurre al final del Sprint, normalmente después de la Sprint Review. Participa el Equipo Scrum completo. Su propósito es inspeccionar cómo trabajó el Equipo Scrum de manera conjunta para lograr el Sprint Goal, y qué puede ser mejorado en el siguiente Sprint. Mientras que la Sprint Review está más enfocada hacia el producto y el contenido del trabajo que se ha hecho, la Sprint Retrospective se dirige a inspeccionar el proceso de cómo se ha hecho el trabajo.

Para Sprints de un mes, la Sprint Retrospective no debe durar más de 3 horas. Pero cuanto más corto es el Sprint, más corta tiende a ser su duración. El Equipo Scrum puede hacer ajustes para trabajar de forma más efectiva, basándose en lo que ha aprendido, tras inspeccionar su colaboración. Esto puede implicar una actualización de la Definición de Terminado (Definition of Done), investigar nuevas herramientas o tecnologías, un cambio en los acuerdos de trabajo o en la composición del equipo. Al menos una mejora debe incluirse directamente al Sprint Backlog del siguiente Sprint.

2.3.6 Refinamiento del Product Backlog

Si bien es una parte esencial del Marco de Trabajo Scrum, no es un evento como los otros. La guía de Scrum lo menciona como algo que tiene que ocurrir en algún momento, una actividad continua. Sin embargo, existen diferentes formas de refinamiento:

- Aclaración de los elementos en el Product Backlog que son demasiado confusos como para empezar a trabajar con ellos. Esto se hace preferiblemente con la gente para la que estás construyendo los elementos los “stakeholders”.
- Descomposición de los elementos del Product Backlog que por su tamaño no se pueden completarse en un Sprint (lo que generalmente también significa que son poco claros).

- Re-ordenamiento del trabajo en el Product Backlog según sea necesario para hacer los siguientes Sprints tan tranquilos y valiosos como sea posible.
- Añadir o eliminar elementos del Product Backlog a medida que surgen nuevas ideas.
- Estimación del esfuerzo que conlleva implementar ciertos elementos. Esto no tiene por qué ser tan "formal" como asignar puntos de historia (una práctica opcional en Scrum), tallas de camiseta o cualquier otra técnica de medida que se use. Una estimación a grandes rasgos (sí, sabemos que lo que hay que hacer cabe dentro de un Sprint) también es válida.

En cierto sentido, los elementos en un Product Backlog son recordatorios de "conversaciones que deben tenerse en un futuro". Y el refinamiento es el proceso continuo de llevar a cabo esas conversaciones. Algunas veces esto significa hablar con los stakeholders sobre un elemento que puede acabar en el siguiente Sprint, mientras que otras, puede ser clarificar un elemento en el que un equipo ya está trabajando. Algunas de esas conversaciones requerirán al equipo completo y otras a pequeños grupos. Algunos refinamientos pueden hacerse incluso individualmente.

Por lo tanto, en lugar de ver el refinamiento del Product Backlog como una reunión formal, que ocurre una vez en el Sprint, en la que participa todo el equipo, ésta debería ser una serie de actividades recurrentes para refinar el trabajo del Product Backlog para los siguientes Sprints. Es una actividad continua que ocurre de diferentes formas, en la que participan diferentes configuraciones de personas. Solo el Equipo Scrum puede decidir cómo llevarlo a cabo.

2.3.7 Dos Principios Rectores

Cuando se describe el Marco de Trabajo Scrum, a menudo se lo hace explicando los eventos y roles. Aunque estos elementos son ciertamente importantes, su efectividad está determinada por lo bien que los equipos comprendan los dos principios rectores.

Entregar un Incremento Terminado Cada Sprint

Si se tuviera que sintetizar el propósito del Marco de Trabajo Scrum en una sola idea, esta sería trabajar empíricamente entregando un incremento "Terminado" cada Sprint. Esta también es la razón por la que es tan importante que los Equipos Scrum dediquen tiempo a determinar qué hace falta para considerar un incremento como "Terminado". ¿Qué trabajo requiere esto? ¿Qué controles y pruebas son necesarios para cumplir con las pautas de calidad internas? ¿Quién necesita estar involucrado? Esta comprensión común se llama "Definición de Terminado", y generalmente se traduce en una lista de verificación.

El trabajo en el Product Backlog tiende a estar lleno de suposiciones hasta que se completa y se pone en manos de los usuarios. Por ejemplo, "¿Implementar este elemento va a mejorar la experiencia de los usuarios?", "¿Entenderán cómo funciona?", "¿Funciona lo suficientemente bien?" Habrá otras suposiciones sobre el trabajo que debe hacerse para implementar cierto elemento, tales como "¿Qué tan fácil será probar y desplegar este

elemento?", "¿Qué problemas se encontrarán cuando trabajemos en este elemento?", "¿Se comprende realmente bien qué es lo que se necesita crear?".

Todas estas suposiciones representan riesgos. El riesgo de malinterpretar lo que ha pedido un usuario y tener que rehacer el trabajo. El riesgo de encontrarse con limitaciones técnicas cuando se comienza a escribir el código. El riesgo de que la funcionalidad tenga un rendimiento mucho peor de lo que se esperaba. El riesgo de que la escritura de pruebas automatizadas resulte ser mucho más difícil. O el riesgo de gastar dinero y tiempo en una funcionalidad que acabe no siendo utilizada. Lo que todos estos riesgos comparten es que suponen un aumento inesperado del trabajo en algún momento. Y este tipo de trabajo "arrastrado" tiende a ser invisible hasta que aparecen repentinamente, impactando lo que se esté desarrollando en ese Sprint. Esto se llama "trabajo no-terminado".

La mejor manera de prevenir el riesgo del "trabajo no- terminado" es asegurarse de que cada Sprint culmine en, al menos, un Incremento "Terminado" que pueda ser potencialmente desplegado. Eso significa que el Incremento ha sido completamente testeado y está funcionando. Los textos y las imágenes son los definitivos. La documentación y los paquetes de despliegue están actualizados. Y el rendimiento y la seguridad se ajustan a los estándares de la organización. Llegado este punto, el incremento se puede poner a disposición de los usuarios. A partir de este momento, ya se sabe que habrá poco o ningún trabajo potencial sin terminar que pueda estropear el enfoque y la previsibilidad de futuros Sprints. Y lo que es más importante, al desplegar el producto puede comprobarse de manera empírica si el incremento realmente acerca el producto a la visión, en vez de asumir que lo hará.

Crear un incremento "Terminado" cada Sprint es realmente un desafío. Pero es así a propósito. Porque poner tal cantidad de presión en el sistema que crea el producto (hacer todo lo necesario para conseguirlo) mostrará de forma clara dónde son necesarias las mejoras. Dónde faltan habilidades, herramientas y tecnologías. Dónde está estorbando la burocracia. Dónde se necesita eliminar impedimentos para poder trabajar, de verdad, empíricamente y reducir el riesgo que supone el trabajo complejo. Al mantener el enfoque en la entrega de un Incremento "Terminado" cada Sprint, el sistema comenzará a resentirse en los lugares correctos creando así oportunidades para la inspección y la adaptación.

Usar una Meta Compartida para Crear Cohesión durante el Sprint

En la Guía de Scrum se hace mención 27 veces al Sprint Goal, más que a cualquier otro elemento del marco de trabajo. Esto debería decirnos algo sobre la importancia que tiene.

La razón por la cual los Sprint Goals son tan importantes se remonta a para qué está diseñado el Marco de Trabajo Scrum; para sortear de forma más efectiva los problemas complejos. Los Sprint Goals ayudan con esta complejidad de tres maneras complementarias.

La primera es que ofrece una guía a los Developers cuando estos tienen que decidir en qué emplear su tiempo. Algunas tareas del Sprint Backlog pueden ser más importantes que otras

para conseguir el Sprint Goal. La segunda es que permite al equipo concentrarse en lo que es importante. Cuando las cosas se ponen difíciles, como normalmente sucede, el equipo puede decidir dejar a un lado cierto trabajo y priorizar otro. O, cuando el tiempo lo permite, durante el Sprint pueden agregar trabajo que descubren que es necesario para lograr el objetivo. Y tercero, los Sprint Goals promueven la colaboración al brindar a los Developers un propósito claro y común para autoorganizarse, en lugar de trabajar de forma individual. La colaboración hace posible el tipo de pensamiento divergente y espíritu de equipo que se necesita para resolver problemas complejos.

Los Sprint Goals también dan color y propósito a los diferentes Eventos de Scrum, haciendo de ellos mucho más que simples reuniones de trabajo. El Sprint Planning gira en torno a la definición de la meta para el siguiente Sprint y a la selección del trabajo que se necesite para llevarla a cabo. La Daily Scrum se centra en cómo trabajarán los Developers las siguientes 24 horas para lograr esta meta, y qué impedimentos pueden estar bloqueando su capacidad para lograrlo. La Sprint Review trata de verificar el resultado del Sprint Goal con los stakeholders y trabajar con ellos para identificar las próximas metas en las que centrarse. La Sprint Retrospective se centra en descubrir formas de trabajar juntos de manera más efectiva para lograr los Sprint Goals.

Pero los Sprint Goals también ayudan a enfocar los roles del Marco de Trabajo Scrum. Guían y orientan a los Developers a la hora de autoorganizarse. Permiten a los Product Owners concentrarse en la visión del producto y cómo traducirla en objetivos para varios Sprints, en lugar de involucrarse en los detalles de cómo se van a implementar. Y finalmente, los Scrum Masters pueden aumentar la efectividad del control del proceso empírico. Que un equipo tenga problemas para crear los Sprint Goals es un buen motivo para que los Scrum Masters investiguen para descubrir la causa. Los equipos pueden ser demasiado grandes o pequeños, pueden carecer de ciertas habilidades o personas. El Product Owner puede no tener autoridad o visión. O el refinamiento puede no estar teniendo lugar.

Para resumir, en trabajos complejos, los Sprint Goals son faros que ayudan a llegar al puerto a través de la densa niebla. Sin ellos, es probable que los equipos se pierdan o se queden varados.

2.3.8 Cinco Valores Fundamentales que Permiten el Empirismo

Hasta este punto, se han cubierto las mecánicas y los principios del Marco de Trabajo Scrum. Aunque funcionan bien para permitir el control del proceso empírico, por sí mismos, sin un comportamiento en el que apoyarse, no supondrían nada diferente ¿cómo de honesta puede ser la inspección durante una Sprint Review cuando los equipos Scrum tienen miedo a hablar de forma abierta sobre los desafíos técnicos que ven, y en su lugar recurren al mensaje de que “todo va bien”? ¿cómo de efectivo puede ser un Equipo Scrum cuando el Product Owner tiene miedo de declinar o rechazar elementos en el Sprint Backlog que no se ajustan al Goal? ¿cuánto valor generarán Developers que siguen distrayéndose con la tecnología más nueva y popular? ¿cómo afecta el cambio de composición del equipo a la voluntad de comprometerse a trabajar

como tal, cuando las decisiones acerca de esos cambios se toman sin el conocimiento o aprobación de los afectados?

Es difícil trabajar de forma empírica, especialmente en organizaciones que no están acostumbradas a ello. Con el fin de dar a los equipos y a sus stakeholders una guía en la toma de decisiones, el Marco de Trabajo Scrum enfatiza específicamente cinco valores rectores. Por cada decisión a la que se enfrenten, los equipos pueden preguntarse cómo esta afectará a esos cinco valores:

- **Apertura:** afrontar de forma abierta cómo van las cosas. ¿Qué está yendo bien? ¿qué no? ¿dónde están los desafíos y las oportunidades?
- **Coraje:** ser valiente para hacer lo correcto. Decir "no" a las cosas que impiden el proceso empírico. Mostrar coraje trabajando juntos en retos difíciles. Pedir y estar dispuesto a dar feedback sobre cosas de las que no se está seguro. Hacer preguntas y admitir lo que no se sabe o aquello sobre lo que se tiene dudas;
- **Foco**: mantenerse enfocado en el Sprint Goal y las metas del Equipo Scrum. Crear un espacio donde la gente pueda conservar y mantener el foco de atención;
- **Respeto:** respetar las habilidades, experiencia e inteligencia de los miembros del Equipo Scrum. Confiar en su habilidad para autoorganizarse en torno a problemas complejos. Y respetar la incertidumbre inherente al trabajo complejo;
- **Compromiso:** crear un entorno donde las personas puedan comprometerse personalmente a trabajar juntos como equipo hacia el Sprint Goal;

Aprender a integrar estos valores en el comportamiento diario de los equipos es un trabajo continuo. Cuanto más competentes sean los equipos con ellos, más efectivas serán la transparencia, la inspección y la adaptación frecuentes. La buena noticia es que el Marco de Trabajo Scrum proporciona un excelente conjunto de límites para aprender y crecer en la adopción de estos valores.

2.4 Gestión Agile de Proyectos de Data Science

Aunque la Data Science y la ingeniería de software son campos diferentes, muchas organizaciones las tratan de la misma manera. Por lo tanto, no es sorprendente que muchas organizaciones también estén impulsando la adopción de Agile en la Data Science. Sin embargo, los resultados son mezclados, a menudo debido a intentos fallidos de aplicar prácticas de software Agile a la Data Science.

¿Cuántos equipos de Data Science usan Agile? No se conoce con certeza, pero un estudio de 2017 encontró que entre el 25% y el 50% de los equipos de Data Science actualmente utilizan un enfoque Agile y que es probable que este porcentaje aumente en el futuro. (Kobielus, 2017)

A continuación, se describen algunos de las principales metodologías y frameworks de trabajo con enfoque agile que existen.

2.4.1 CRISP-DM

El proceso estándar de **CR**oss **I**ndustry **S**tandard **P**rocess for **D**ata Mining (*CRISP-DM*) es un modelo de proceso que sirve como base para un proceso de Data Science. Tiene seis fases secuenciales:

1. Comprensión del negocio: ¿Qué necesita el negocio?
2. Comprensión de datos: ¿Qué datos tenemos/necesitamos? ¿Están limpios?
3. Preparación de los datos: ¿Cómo organizamos los datos para el modelado?
4. Modelado – ¿Qué técnicas de modelado debemos aplicar?
5. Evaluación: ¿Qué modelo cumple mejor con los objetivos de negocio?
6. Despliegue: ¿Cómo acceden las partes interesadas a los resultados?

Publicado en 1999 para estandarizar los procesos de minería de datos en todas las industrias, desde entonces se ha convertido en una de las metodologías más comunes para proyectos de minería de datos, análisis y Data Science.

Equipos de Data Science que combinen una implementación flexible de CRISP-DM con enfoques generales de gestión ágil de proyectos obtienen los mejores resultados.

2.4.1.1 Fases de CRISP-DM

I. Comprensión del negocio

Cualquier buen proyecto comienza con una comprensión profunda de las necesidades del cliente. Los proyectos de minería de datos no son una excepción y CRISP-DM lo reconoce.

La fase *de comprensión empresarial* se centra en comprender los objetivos y requisitos del proyecto. Dejando de lado la tercera, el resto de las tareas en esta fase son actividades fundamentales de gestión de proyectos que son universales para la mayoría de los proyectos:

1. **Determinación de los objetivos de negocio:** Es preciso comprender completamente, desde una perspectiva de negocio, lo que el cliente realmente quiere lograr (Corporation, IBM, 2021) y luego definir los criterios de éxito empresarial.
2. **Evaluación de la situación:** Determinar la disponibilidad de recursos, los requisitos del proyecto, evaluar los riesgos y las contingencias y realizar un análisis de costo-beneficio.
3. **Determinación de los objetivos de minería de datos:** Además de definir los objetivos de negocio, también se debe definir cómo es el éxito desde una perspectiva técnica de la minería de datos.
4. **Producción de un plan de proyecto:** Seleccionar tecnologías y herramientas y definir planes detallados para cada fase del proyecto.

Si bien muchos equipos se apresuran a superar esta fase, establecer un fuerte entendimiento del negocio es absolutamente esencial.

II. Comprensión de datos

La siguiente es la fase *de comprensión de datos*. Además de la base de la *comprensión de negocio*, esta fase impulsa el enfoque para identificar, recopilar y analizar los conjuntos de datos que ayudarán a lograr los objetivos del proyecto. Esta fase también tiene cuatro tareas:

1. **Recopilación de datos iniciales:** Adquirir los datos necesarios y, de ser requerido, cargarlos en las herramientas de análisis.
2. **Descripción de los datos:** Examinar los datos y documentar sus propiedades superficiales, como el formato de los datos, el número de registros o las entidades de los campos.
3. **Exploración de datos:** Profundizar en los datos. Consultarlos, visualizarlos e identificar las relaciones entre los datos.
4. **Verificación de calidad de datos:** ¿Qué tan limpios/sucios son los datos? Documentar cualquier problema de calidad.

III. Preparación de los datos

En esta fase se preparan los conjuntos de datos finales para el modelado. Tiene cinco tareas:

1. **Selección de datos:** Determinar qué conjuntos de datos se utilizarán y documentar las razones de inclusión/exclusión.

2. **Limpieza de Datos:** a menudo esta es la tarea más larga. La calidad de los datos de entrada impacta directamente en la calidad de los datos de salida. Una práctica común durante esta tarea es corregir, imputar o eliminar valores erróneos.
3. **Construcción de nuevos datos:** Derivar nuevos atributos que serán útiles a partir de los datos existentes. Por ejemplo, derivar el índice de masa corporal de alguien a partir de los campos de altura y peso.
4. **Integración de datos:** Crear nuevos conjuntos de datos combinando datos de múltiples fuentes.
5. **Formato de datos:** Volver a formatear los datos según sea necesario. Por ejemplo, convertir valores de cadena de texto que almacenan números en valores numéricos para que realizar operaciones matemáticas.

IV. Modelado

Aquí es probable que se construyan y evalúen varios modelos basados en varias técnicas de modelado diferentes. Esta fase tiene cuatro tareas:

1. **Selección de técnicas de modelado:** Determinar qué algoritmos probar (por ejemplo, regresión, red neuronal).
2. **Generación de un diseño de comprobación:** a la espera de su enfoque de modelado, es posible que se deba dividir los datos en conjuntos de entrenamiento, prueba y validación.
3. **Generación de los modelos:** Existen una gran cantidad de herramientas y librerías que facilitan la ejecución de modelos y que pueden ser empleados en esta etapa.
4. **Evaluación del modelo:** En general, varios modelos compiten entre sí, y el científico de datos necesita interpretar los resultados del modelo en función del conocimiento del dominio, los criterios de éxito predefinidos y el diseño de la prueba.

Aunque la Guía CRISP-DM sugiere parar la construcción y evaluación de modelos hasta que se tenga certeza de haber encontrado el mejor modelo, en la práctica, los equipos deben continuar iterando hasta que encuentren un modelo “lo suficientemente bueno”, continuar a través del ciclo de vida de CRISP-DM y luego mejorar aún más el modelo en futuras iteraciones.

V. Evaluación

Mientras que la tarea de evaluar el modelo de la fase de modelado se centra en la evaluación del modelo técnico, la fase de Evaluación analiza más ampliamente qué modelo se adapta mejor al negocio y qué hacer a continuación. Esta fase tiene tres tareas:

1. **Evaluación de los resultados:** ¿Cumplen los modelos con los criterios de éxito empresarial? ¿Cuáles deberían aprobarse para el negocio?

2. **Proceso de revisión:** revisar el trabajo realizado. ¿Se pasó algo por alto? ¿Se ejecutaron todos los pasos correctamente? Se documentan los hallazgos y se realizan las correcciones necesarias.
3. **Determinación los pasos siguientes:** en función de las tres tareas anteriores, se determina si proceder a la implementación, iterar nuevamente o iniciar un nuevo proyecto.

VI. Despliegue

Dependiendo de los requisitos, la fase de implementación puede ser tan simple como generar un informe o tan compleja como implementar un proceso de minería de datos repetible en toda la empresa.

Un modelo no es particularmente útil a menos que el cliente pueda acceder a sus resultados. La complejidad de esta fase varía mucho. Esta fase final tiene cuatro tareas:

1. **Planificación de despliegue:** Desarrollar y documentar un plan para implementar el modelo.
2. **Planificación del control y del mantenimiento:** Desarrollar un plan completo de supervisión y mantenimiento para evitar problemas durante la fase operativa (o fase posterior al proyecto) de un modelo.
3. **Creación de un informe final:** el equipo del proyecto documenta un resumen del proyecto que podría incluir una presentación final de los resultados de la minería de datos.
4. **Revisión final del proyecto:** realizar una retrospectiva del proyecto sobre lo que salió bien, lo que podría haber sido mejor y cómo mejorar en el futuro.

Como marco de proyecto, CRISP-DM no describe qué hacer después del proyecto (también conocido como "operaciones"). Pero si el modelo va a la producción, es necesario su mantenimiento. A menudo se requiere un monitoreo constante y ajustes ocasionales del modelo.

¿Es CRISP-DM agile o cascada?

Existen distintos argumentos sobre si CRISP-DM es rígido o si puede ser flexible y agile.

Cascada: por un lado, CRISP-DM puede visualizarse como un proceso rígido de cascada, en parte debido a que sus requisitos de presentación de informes son excesivos para la mayoría de los proyectos. Además, la guía afirma en la fase de comprensión del negocio que "el plan del proyecto contiene planes detallados para cada fase", un aspecto distintivo de los enfoques tradicionales de cascada que requieren una planificación detallada y por adelantado.

De hecho, si se elige desarrollar un enfoque de CRISP-DM con precisión, definiendo planes detallados para cada fase al inicio del proyecto e incluyendo cada informe, y se elige no iterar con frecuencia, entonces se estará operando un proceso de cascada.

Ágil: Por otro lado, CRISP-DM propone indirectamente los principios y prácticas ágiles afirmando: "La secuencia de las fases no es rígida. Siempre es necesario ir y venir entre diferentes fases. El resultado de cada fase determina qué fase, o tarea particular de una fase, debe realizarse a continuación".

Por lo tanto, un enfoque de CRISP-DM más flexible, en el que se itera frecuentemente acompañado de otros procesos ágiles podrían considerarse más agile.

Para ilustrar cómo se podría implementar CRISP-DM de manera ágil o en cascada, tómese de ejemplo un proyecto de predicción de la tasa de cancelación de servicio de usuarios con tres entregables: un modelo de rotación voluntaria, un modelo de rotación de desconexión no pagada y una propensión a aceptar una oferta centrada en la retención.

Cascada CRISP-DM: Corte horizontal

En una implementación de estilo cascada, el trabajo del equipo abarcaría de manera integral y horizontal a través de cada entregable, como se muestra a continuación. El equipo podría volver con poca frecuencia a una capa horizontal inferior solo si es críticamente necesario. El producto completo se entrega al final del proyecto.

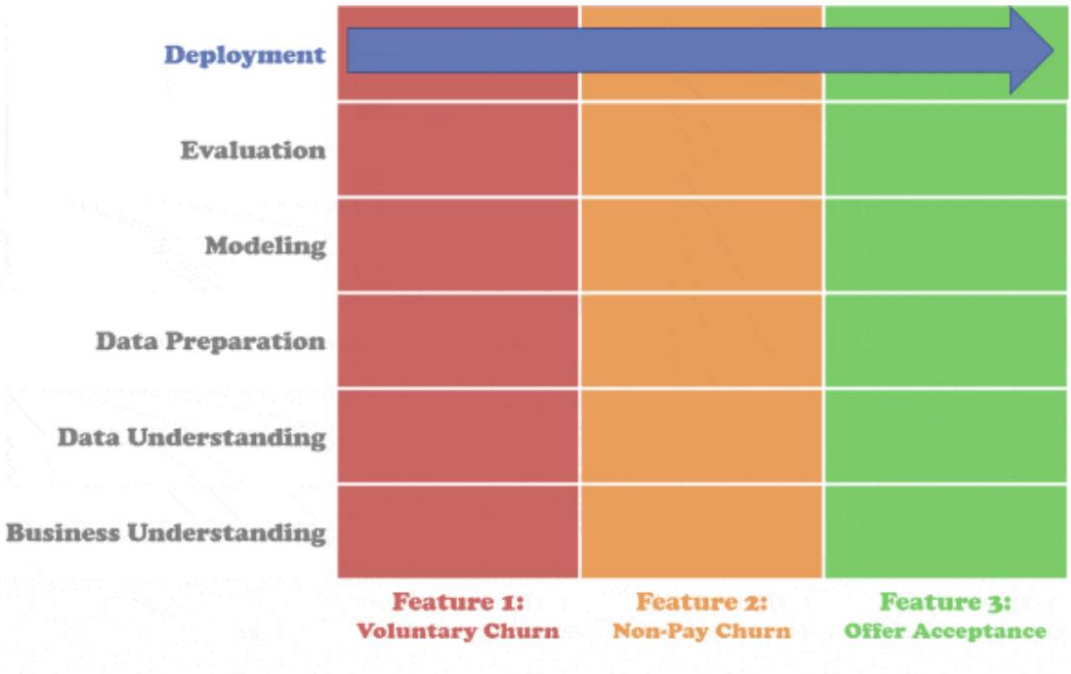

Figura 7. CRISP-DM enfoque cascada

CRISP-DM Agile: Corte vertical

Alternativamente, en una implementación ágil de CRISP-DM, el equipo se centraría estrechamente en entregar rápidamente una porción vertical de la cadena de valor a la vez, como se muestra a continuación. Entregarían múltiples versiones verticales más pequeñas y con frecuencia solicitarían comentarios en el camino.

Figura 8. CRISP-DM enfoque agile

Un enfoque ágil y corte vertical tiene los siguientes beneficios:

- Las partes interesadas obtienen valor antes
- Las partes interesadas pueden proporcionar comentarios significativos
- Los científicos de datos pueden evaluar el rendimiento del modelo antes
- El equipo del proyecto puede ajustar el plan en función de los comentarios de las partes interesadas

Beneficios del CRISP-DM

Desde la perspectiva actual de la Data Science, esto parece sentido común. Este es exactamente el punto. El proceso común es tan lógico que se ha incrustado en la educación, formación y práctica. (Vorhies, 2016)

- **General:** Aunque diseñado para la minería de datos, debido a que todos los proyectos de Data Science comienzan con la comprensión del negocio, tienen datos que deben recopilarse y limpiarse y aplican algoritmos de Data Science, "CRISP-DM proporciona una fuerte orientación incluso para las actividades más avanzadas de Data Science
- **Sentido común:** Cuando se le pidió a un grupo de estudiantes que hicieran un proyecto de Data Science sin dirección de gestión de proyectos, estos se inclinaron por una metodología similar a CRISP e identificaron las fases e hicieron varias iteraciones. Además, los equipos que fueron entrenados y se les dijo explícitamente que implementaran CRISP-DM tuvieron un mejor rendimiento que los equipos que usaban otros enfoques. (Saltz, Shamshurin, & Crowston, 2017)
- **Adopción:** al igual que Kanban, CRISP-DM se puede implementar sin mucha capacitación, cambios de rol organizacional o controversia.
- **Comienzo correcto:** El enfoque inicial en la *comprensión del negocio* es útil para alinear el trabajo técnico con las necesidades empresariales y para alejar a los científicos de datos de saltar a un problema sin comprender adecuadamente los objetivos empresariales.
- **Final fuerte:** su paso final *el despliegue* también aborda consideraciones importantes para cerrar el proyecto y la transición al mantenimiento y las operaciones.
- **Flexible:** una implementación suelta de CRISP-DM puede ser flexible para proporcionar muchos de los beneficios de los principios y prácticas ágiles. Al aceptar que un proyecto comienza con incógnitas significativas, el usuario puede recorrer los pasos, cada vez que obtiene una comprensión más profunda de los datos y el problema. El conocimiento empírico aprendido de los ciclos anteriores puede alimentarse en los siguientes ciclos.

Debilidades y desafíos

En el mismo experimento controlado, los estudiantes que usaron CRISP-DM fueron los últimos en comenzar a programar y no entendieron completamente los desafíos de codificación a los que iban a enfrentar (Saltz, Shamshurin, & Crowston, 2017)

- **Rígido:** Por otro lado, dependiendo de su implementación CRISP-DM puede presentar las mismas debilidades de las metodologías en cascada y obstaculizar la rápida iteración.
- **Documentación pesada:** casi todas las tareas tienen un paso de documentación. Si bien la documentación del trabajo es clave en un proceso maduro, los requisitos de documentación de CRISP-DM podrían retrasar innecesariamente que el equipo entregue incrementos.

- **No moderno:** autores argumentan que, al ser un proceso anterior al desarrollo del Big Data, podría no ser adecuado para proyectos esta clase de proyectos debido a sus tres V variedad, velocidad y volumen. (Nabati & Thoben, 2016)
- **No es un enfoque de gestión de proyectos:** Tal vez lo más significativo, CRISP-DM no es una verdadera metodología de gestión de proyectos porque asume implícitamente que su usuario es una sola persona o un equipo pequeño y unido e ignora la coordinación de trabajo en equipo necesaria para proyectos más grandes.

Recomendaciones

CRISP-DM es un gran punto de partida para equipos que buscan comprender el proceso general de la Data Science. Cinco consejos para superar estas debilidades son:

- **Iterar rápidamente:** evitar el trabajar en cascada esperando finalizar por completo las capas del proyecto y seguir un enfoque vertical de entregas frecuentes e incrementales.
- **Documenta lo suficiente pero no demasiado:** evitar pasar la mayor parte del tiempo documentando en vez de generar valor.
- **Establecer expectativas:** CRISP-DM carece de estrategias de comunicación con las partes interesadas por lo que necesario establecer expectativas y establecer una comunicación fluida con ellos con frecuencia.
- **Combinar con un enfoque de gestión de proyectos:** dado que CRISP-DM no es realmente un enfoque de gestión de proyectos, es recomendable combínalo con otros marcos de trabajo. Los enfoques ágiles populares incluyen: Kanban, Scrum, Data Driven Scrum.

2.4.2 Data Driven Scrum (DDS)

Data Driven Scrum (DDS) es un marco ágil diseñado específicamente para equipos de Data Science. En resumen, DDS tiene como objetivo mejorar la colaboración y la comunicación de un equipo de Data Science.

La Data Science Process Alliance (Alianza de Procesos de Data Science) creó Data Driven Scrum para abordar el hecho desde su perspectiva que otros enfoques ágiles bien conocidos, como a menudo Scrum y Kanban, a menudo no cumplen con las necesidades únicas de los proyectos de Data Science. (Data Science Process Alliance, 2022)

Bajo el esquema del DDS hay tres conceptos clave que permiten a un equipo obtener los beneficios de la agilidad dentro de un proyecto de Data Science. Esta agilidad ayuda a garantizar que un equipo se centre en tareas de máxima prioridad, al tiempo que permite que las tareas futuras se reordenen según sea necesario. Estos tres conceptos ágiles son:

1. Agile pretende ser una secuencia de ciclos iterativos de experimentación y adaptación.
2. El objetivo de cada ciclo debe ser tener una idea o experimento en mente, construirlo, observar el análisis, y luego analizar esas observaciones para crear la siguiente idea o experimento.
3. Pasar de una idea inicial, a la implementación, y al análisis de los resultados debería ser la base para una iteración. La finalización del proceso empírico debe marcar el final de esa iteración y no un número predeterminado de horas transcurridas.

Descripción del flujo de trabajo de alto nivel del DDS

- En primer lugar, los equipos de DDS hacen una lluvia de ideas sobre posibles preguntas para responder o experimentos para realizar.
- Luego, el equipo prioriza esas preguntas y experimentos, eligiendo el ítem de mayor prioridad en el que trabajar como equipo. Esto incluye identificar los datos a utilizar y los modelos que deben crearse.
- A continuación, el equipo interpreta colectivamente los resultados del trabajo.
- En función de los resultados, el equipo despliega los resultados y prioriza el trabajo futuro.

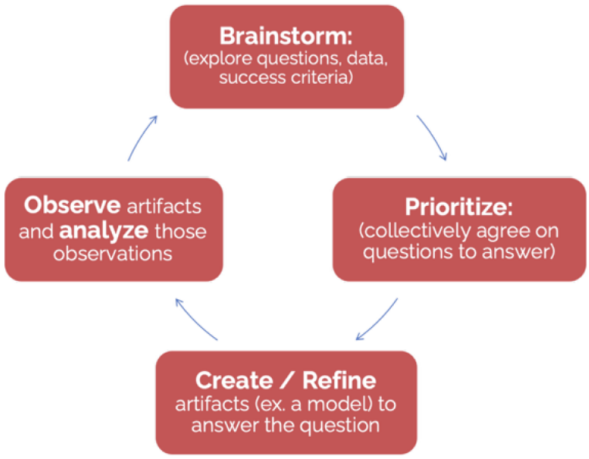

Figura 9. Descripción del flujo de trabajo de alto nivel del DDS

Roles del DDS

Al igual que Scrum, hay tres funciones clave en DDS:

Product Owner: El Product Owner en DDS es el punto central empoderado del liderazgo del producto la "voz del cliente", la persona que decide sobre los incrementos del producto. Prioriza qué características y funcionalidades construir, el orden en el que construirlos y qué aspectos de ellos observar y analizar. En resumen, el Product Owner es el propietario del Backlog y prioriza sus elementos, asegurando que cada uno esté claramente definido y que el trabajo próximo y las prioridades del equipo sean visibles y transparentes.

Experto en procesos: el experto en procesos actúa como entrenador, facilitador y eliminador de impedimentos. El experto en procesos también ayuda al equipo a comprender y adoptar los valores, principios y prácticas de DDS para ayudar a la organización a obtener resultados excepcionales de la aplicación de DDS.

Miembros del equipo DDS: Al igual que Scrum, cada equipo DDS es un grupo de tres a nueve personas. El equipo de DDS está compuesto por un grupo multifuncional de profesionales (científicos de datos, ingenieros de software, ingenieros de pruebas, etc.) que tienen todas las habilidades necesarias para crear artefactos, por ejemplo, modelos, para responder a las preguntas / experimentos, es decir, para diseñar, construir, probar e implementar el producto deseado.

Tanto el Product Owner como el Experto en Procesos forman parte del equipo de DDS y pueden contribuir a crear, observar y analizar a lo largo de una iteración. El equipo se autoorganiza para determinar la mejor manera de lograr el objetivo definido por el Product Owner.

Artefactos DDS

DDS tiene cuatro artefactos que describen el trabajo a realizar:

Ítem: Un ítem puede adoptar una variedad de formas, como "historias de usuarios", "experimentos" o "hipótesis a probar" similar a XP y Lean. En Data Science, los estos ítems suelen ser preguntas que el equipo necesita responder o hipótesis para evaluar.

Backlog: El Backlog es una lista priorizada de ítems.

Tablero de desglose de ítems (IBB): El tablero de desglose de ítems (IBB) es el lugar donde cada ítem del Backlog se divide en tareas. Los ítems del Backlog se desglosan en sus tareas componentes antes de que el equipo los trabaje, y cada ítem tiene su propio IBB. Este trabajo para crear el IBB se realiza durante el refinamiento del Backlog. Para cada ítem, debe haber al menos tarea de:

- Crear
- Observar
- Analizar

Tablero de tareas: El tablero de tareas es una representación visual de los ítems actualmente en curso. Para que el trabajo en un ítem se inicie, es decir, que el equipo sea trabajado, las tareas de ese ítem se mueven del IBB al Tablero de tareas. Estas tareas se muestran en el tablero, normalmente en la columna “To do” o “por hacer”. El Tablero de tareas tiene varias columnas adicionales como mínimo, “To Do”, “In Progress” y “Done” ("por hacer", "en progreso", "hecho") y cada tarea fluye a través de la tabla, mostrando visualmente el trabajo que se está haciendo dentro del equipo. El equipo se esfuerza por completar las tareas en el tablero lo antes posible.

El estado de estos ítems siempre se representa visualmente en el Tablero de tareas y la iteración se completa cuando todas las tareas de ese ítem están en la columna “Done”. Debe tenerse en cuenta que el Product Owner debe estar de acuerdo en que las tareas de la columna “Done” se han completado realmente. Al igual que con Kanban, para facilitar el rendimiento de las tareas, cada equipo define un número máximo de tareas dentro de una sola columna, que se conoce como el límite de trabajo en progreso de esa columna.

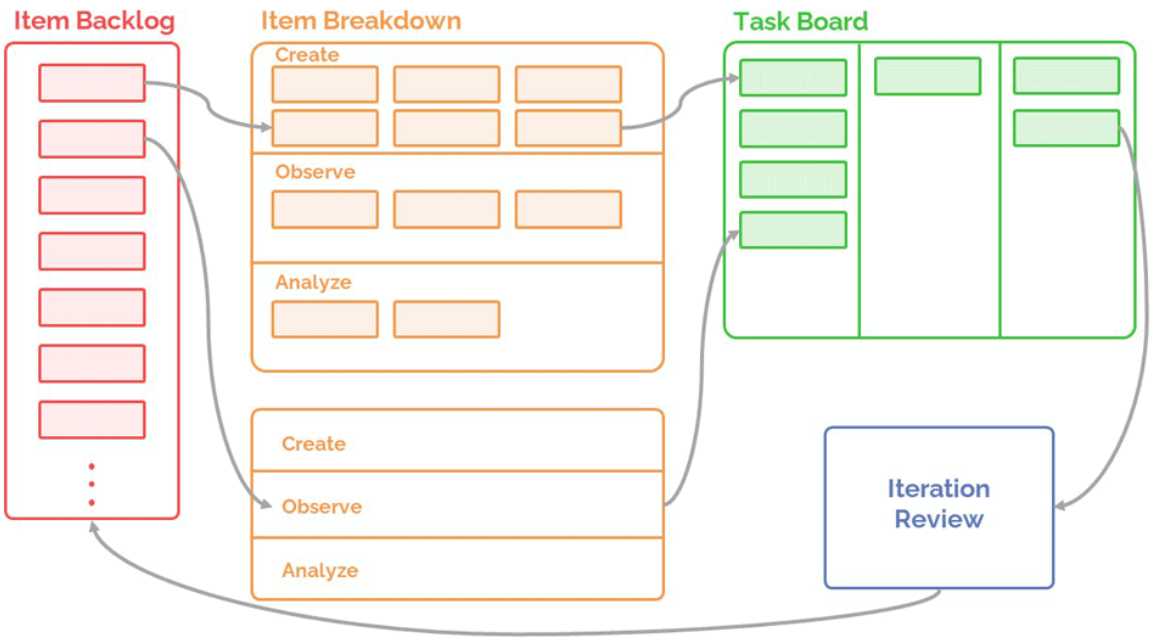

Figura 10. Tablero de desglose de ítems

Actividades de DDS

Las actividades de DDS definen el trabajo que debe realizar el equipo. Estas actividades no están en delimitadas dentro un periodo de tiempo, sino que se centran en definir y luego ejecutar iteraciones. A continuación, se muestra una explicación de estos conceptos:

Refinamiento del Backlog: Además de que el equipo de DDS trabaja en una o más iteraciones, el equipo también dedica tiempo a evaluar los ítems de Backlog para que puedan ser priorizados. Esta evaluación incluye:

- Una estimación relativa del valor del ítem cuando se completa
- Una estimación relativa del esfuerzo necesario para completar el ítem
- Una estimación relativa de la probabilidad de éxito para crear el ítem

Como parte del proceso de perfeccionamiento, el equipo define la unidad relativa de medidas. Esta estimación podría ser basada en la técnica de tamaño de camiseta (grande, mediano, pequeño) o un número que represente una estimación del valor relativo, el esfuerzo y la probabilidad de éxito de los ítems. Debe tenerse en cuenta que la estimación del esfuerzo se utiliza para ayudar a priorizar los ítems del Backlog, pero no para definir lo que es parte de una iteración (por ejemplo, si dos ítems entregan el mismo valor, pero uno se considera un esfuerzo

"pequeño" y uno es un esfuerzo "grande", el equipo podría seleccionar el ítem de menor esfuerzo).

Mientras que el Product Owner es el propietario del proceso de priorización, el resto de los miembros del equipo pueden emplear del 5 % al 10 % de su capacidad total para ayudar al Product Owner con el refinamiento del Backlog (por ejemplo, dividir un ítem en dos más pequeños, pero aún útiles, aclarando o simplificando un ítem, proporcionando estimaciones de esfuerzo, etc.).

Priorización del Backlog: El Product Owner, con la aportación de las partes interesadas y los demás miembros del equipo, es responsable de mantener el Backlog, que evoluciona y cambia a lo largo del proyecto. El Product Owner utiliza esta información para priorizar y seleccionar los ítems a desarrollar durante la iteración.

Iteraciones: Una iteración es una colección de uno o más ítems del Backlog. El objetivo de cada iteración debe ser permitir que una parte lógica del trabajo se libere de una manera coherente.

Cada iteración incluye el trabajo para ***crear*** un artefacto que responda a una pregunta / hipótesis (por ejemplo, un modelo), el trabajo para ***observar*** ese artefacto (por ejemplo, cómo funciona el modelo en los datos de prueba) y el ***análisis*** del equipo de esas observaciones. La información obtenida de la iteración debe tener un valor derivado del artefacto que se crea o del análisis de la tarea completada.

Duración de la iteración: cada iteración se basa en la capacidad del equipo, no un periodo determinado de tiempo como tal. Cada iteración debe tener como objetivo ser un conjunto de trabajo mínimamente viable que pueda ofrecer valor y permita que se pruebe la hipótesis dada, y no debe durar más de un mes, pero puede ser tan corto como el equipo lo requiera, por ejemplo, un día.

Una iteración se completa cuando el trabajo requerido para responder a la pregunta ha terminado. Cada iteración permite a los equipos responder rápidamente a una pregunta, validar o rechazar una hipótesis, por lo que las iteraciones facilitan la agilidad. Aprender de las iteraciones actuales ayuda a priorizar las iteraciones futuras.

Dado que la iteración se basa en la capacidad y es el conjunto mínimo viable de ítems que pueden ofrecer valor. Responder a múltiples preguntas en una sola iteración generalmente solo es deseable en el caso de que la hipótesis asociada o los datos observables se superpongan. Sin embargo, la tarea de observación dentro de una iteración puede llevar algún tiempo, por ejemplo, la recopilación de datos de un modelo o artefacto. En esta situación, a menudo tiene

sentido que se comience la siguiente iteración, es decir, puede haber varias iteraciones que se ejecuten en paralelo.

Incrementos de producto: un objetivo de alto nivel que el equipo puede lograr en un período de tiempo fijo (por ejemplo, 3 meses) utilizando múltiples iteraciones se conoce como incremento de producto. Los incrementos ayudan a los equipos a priorizar las iteraciones dentro del incremento y establecer expectativas con los clientes.

Eventos DDS

Hay cuatro eventos que ocurren regularmente, es decir, los eventos ocurren sobre una base de calendario, no sobre la finalización de una iteración. Estos eventos, que son facilitados por el experto en procesos de DDS, ayudan al equipo a mantenerse coordinado. En resumen, estos eventos ayudan a planificar las iteraciones a través de la selección de ítems del Backlog, a revisar los resultados de la iteración a través de revisiones de iteración y aprender para futuras iteraciones, a reflexionar sobre cómo mejorar el proceso de un equipo a través de retrospectivas y a comprender los posibles obstáculos en la iteración a través de reuniones diarias.

Estos eventos se describen con más detalle a continuación:

Selección de ítems del Backlog: ocurre cuando el equipo tiene capacidad para iniciar una nueva iteración (por ejemplo, cuando se ha completado una iteración anterior, o cuando la iteración en curso no requiere un enfoque a tiempo completo, generalmente durante la fase de "observación"). Los equipos pueden tener múltiples iteraciones en curso simultáneamente, pero deben priorizar terminar una iteración en curso en lugar de comenzar una nueva siempre que sea posible. El equipo revisa los ítems del Backlog priorizados (que se han actualizado a través del refinamiento) y selecciona los artículos principales que ahora serán el foco del equipo.

Reunión diaria: se produce cada día laborable, cuando el equipo se reúne para una actividad de inspección y adaptación de 15 minutos. Un objetivo importante de esta reunión es ayudar a un equipo autoorganizado a gestionar mejor el flujo de su trabajo, por ejemplo, ayudar a un miembro del equipo a superar un problema.

Revisión de la iteración: se produce de forma regular y repetida y es programada por el Product Owner. Las revisiones podrían ser semanales y se basan en el calendario para tener en cuenta el hecho de que podría haber varias iteraciones por semana, y habría rendimientos decrecientes si las revisiones de iteración se produjeran a diario (o con más frecuencia). También sería logísticamente difícil de programar si se necesitaran de forma ad hoc. La revisión tiene como objetivo fomentar la conversación sobre la funcionalidad completa y las observaciones y

análisis que el equipo ha generado con respecto al rendimiento de la o las iteraciones completas. Los participantes incluyen al equipo, a las partes interesadas, a los clientes y a cualquier otra persona interesada en el resultado del proyecto.

Una revisión exitosa da como resultado un flujo de información bidireccional. Las personas que no están en el equipo pueden sincronizarse con el esfuerzo del proyecto, el rendimiento observado del producto y el análisis del equipo de ese rendimiento. Al mismo tiempo, además de recibir comentarios sobre la iteración entregada actualmente, el equipo puede obtener sugerencias de los otros asistentes para posibles características, métricas y experimentos para futuras iteraciones. Además, durante esta reunión, el grupo discute la priorización de los ítems del Backlog, ya que, los conocimientos adquiridos podrían sugerir un cambio en la prioridad o la creación de nuevos ítems. Al final de la revisión, se archivan las tareas relacionadas con los ítems discutidos y ya completados.

Retrospectiva: ocurre a intervalos regulares y es un momento para inspeccionar y adaptar el proceso. En el espíritu de la mejora continua, el equipo se reúne para discutir lo que está y no está trabajando con el proceso actual y las prácticas técnicas asociadas. El objetivo es ayudar a un buen equipo de DDS a ser grande. Al final de una retrospectiva, el equipo deberá haber identificado y comprometido al menos una accione de mejora de procesos que el equipo llevará a cabo en el futuro.

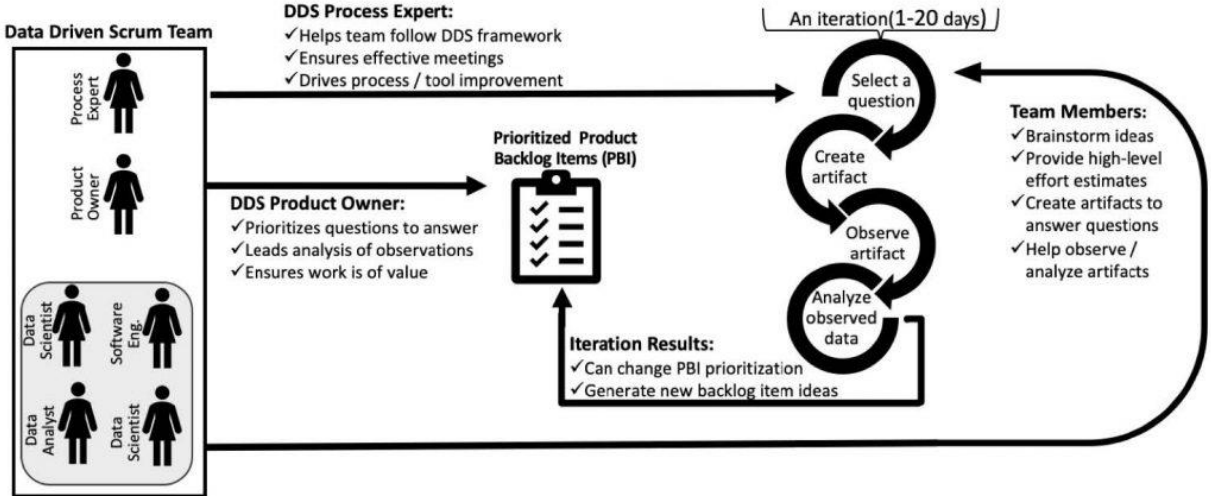

Figura 11. Flujo de trabajo durante un proyecto de DDS

Cómo el DDS es diferente del Scrum tradicional

Iteraciones funcionales (basadas en la capacidad)

La diferencia más importante entre DDS y Scrum es que la Guía Scrum requiere que todas las iteraciones (Sprints) sean de igual duración en tiempo. Sin embargo, las iteraciones en DDS varían en duración, de modo que se permita que se realice una parte lógica de trabajo en una iteración, en lugar de definir la cantidad de trabajo que se puede hacer en una unidad de tiempo específica. En otras palabras, las iteraciones DDS tienen iteraciones de longitud desconocidas y variables, en comparación con los Sprints Scrum tradicionales, que tienen duraciones de tiempo fijo y pueden ser más cortas o largas que una iteración "promedio" por ejemplo, una iteración podría ser más corta de lo normal debido a que es capaz de aprender de un experimento rápido.

Duración de la tarea incierta

Dado que las iteraciones de DDS se basan en la capacidad, los equipos de DDS no se ven obligados a estimar lo que se puede completar en 1 o 2 semanas. Por lo tanto, a diferencia del Scrum tradicional (que necesita estimaciones lo suficientemente precisas como para saber qué puede encajar en un sprint), DDS se adapta naturalmente a tareas que son difíciles de estimar, ya que la estimación de tareas a menudo es difícil dentro de un contexto de Data Science. Por lo tanto, DDS no requiere que el equipo genere estimaciones detalladas y precisas de las tareas. Sin embargo, las estimaciones de nivel de esfuerzo de "Camiseta" de alto nivel suelen ser necesarias para priorizar las tareas potenciales que se deben realizar.

Análisis colectivo

En muchas implementaciones de Scrum, observar, analizar y reaccionar a la retroalimentación es responsabilidad exclusiva del Product Owner. Esta parte del trabajo del Product Owner queda en gran medida fuera del proceso codificado. Recopilar y analizar datos bien elegidos y sacar conclusiones adecuadas es una parte crucial del proceso. Al construir los pasos de observación y análisis directamente en el flujo de trabajo central de DDS, ayuda a los equipos a tomar mejores decisiones basadas en datos. Específicamente, todo el equipo se centra en crear, observar y luego analizar una hipótesis, análisis o característica, a menudo en el scrum tradicional, este análisis lo realiza el Product Owner fuera del proceso codificado.

Desacoplamiento de las reuniones de una iteración

Dado que una iteración podría ser muy corta, por ejemplo, un día para un análisis exploratorio específico, las reuniones, como una retrospectiva para mejorar el proceso del equipo, deben basarse en una frecuencia calendarizada que el equipo considere apropiada, es decir, no vinculada al final de cada iteración como se hace en el Scrum tradicional. En resumen, las retrospectivas y las revisiones de artículos no se realizan al final de cada iteración, sino más bien a una frecuencia que el equipo considere apropiada.

Las iteraciones superpuestas son mucho más comunes en el DDS

Una tarea de observación puede llevar tiempo, por ejemplo, la recopilación de datos. Dado que DDS tiene iteraciones basadas en la capacidad, es fácil iniciar las siguientes iteraciones y pausar la siguiente iteración cuando se haya completado la tarea de observación.

Cómo se basa DDS en el Scrum Tradicional

Funciones

Al igual que el Scrum tradicional, cada equipo de DDS es un grupo de hasta unas diez personas, una de las cuales es el Product Owner y una de las cuales es el Experto en Procesos.

Eventos

Al igual que en el Scrum tradicional, hay una reunión diaria, así como iteraciones, Reviews y retrospectivas.

Proceso para crear y priorizar los ítems

Al igual que el Scrum tradicional, los ítems se crean, priorizan y ven en un tablero de tareas.

Centrado en la agilidad

Tanto DDS como Scrum se centran en permitir que el equipo sea ágil a través de pequeñas iteraciones

Cómo se construye DDS en Kanban

Si bien no hay una guía oficial de Kanban, los equipos que usan Kanban suelen seguir dos *principios* clave de *Kanban.* DDS se adhiere a estos principios clave de Kanban:

Visualizar el flujo de trabajo: ver el trabajo en un tablero con tarjetas para representar el trabajo que se debe hacer y en curso, a través del uso de columnas del tablero.

Límite de trabajo en curso (WIP): Establecer un límite en la cantidad de trabajo en curso a la vez en cada columna. En otras palabras, ¿cuántas tareas pueden estar en cada columna en un momento dado? Esto garantiza que las tarjetas se muevan sin problemas por todo el tablero a medida que el equipo esté listo para ellas.

2.4.3 Team Data Science Process (TDSP)

El Proceso de Data Science en equipo (TDSP) es una metodología de Data Science ágil e iterativa para proporcionar soluciones de análisis predictivo y aplicaciones inteligentes de manera eficiente. TDSP ayuda a mejorar la colaboración y el aprendizaje en equipo al sugerir cómo los roles de equipo funcionan mejor juntos. TDSP incluye procedimientos recomendados y estructuras de Microsoft y otros líderes del sector para ayudar a implementar correctamente iniciativas de Data Science. El objetivo es ayudar a las empresas a que se den cuenta de las ventajas de su programa de análisis.

A continuación, se describe de forma general el proceso que se puede implementar con distintos tipos de herramientas además de ideas sobre cómo implementar el TDSP mediante un conjunto específico de herramientas de Microsoft y la infraestructura que se usa para implementar el TDSP en los equipos.

Principales componentes del TDSP

TDSP tiene los siguientes componentes principales:

- Una definición de ciclo de vida de Data Science
- Una estructura de proyecto estandarizada
- Infraestructura y recursos recomendados para proyectos de Data Science
- Herramientas y utilidades recomendadas para la ejecución de proyectos

Ciclo de vida de Data Science

Team Data Science Process (TDSP) proporciona un ciclo de vida recomendado que se puede utilizar para estructurar proyectos de Data Science.

Cabe señalar que este ciclo de vida está diseñado para contextos de proyectos de Data Science que conducen a la creación de productos de datos y aplicaciones inteligentes que incluyen análisis predictivo utilizando modelos de Machine Learning o inteligencia artificial. Proyectos exploratorios de Data Science y proyectos de análisis ad hoc y otros pueden utilizar este proceso, pero es posible que no se necesiten algunos pasos de este ciclo de vida.

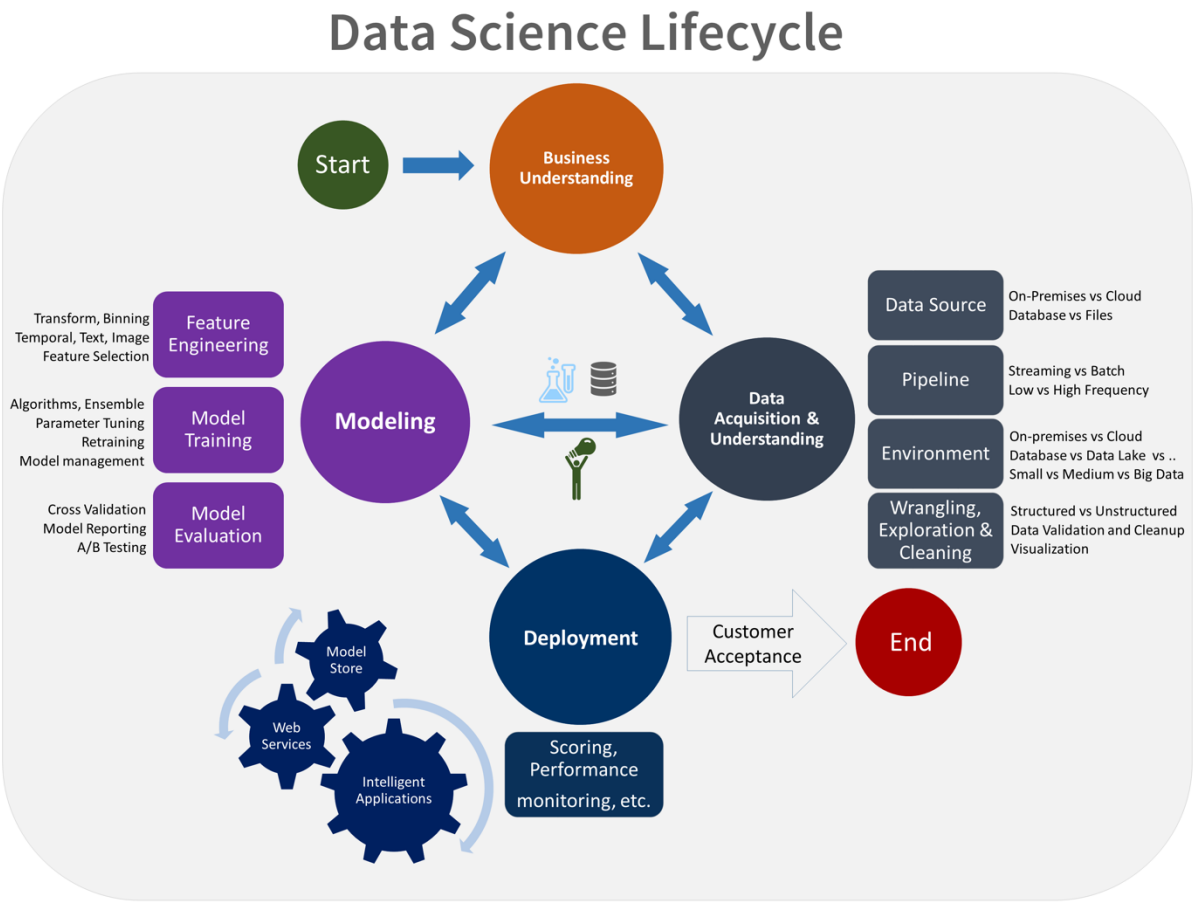

Figura 12. Representación visual del ciclo de vida del Proceso de Data Science en Equipo.

El ciclo de vida de la Data Science de TDSP se compone de cinco etapas principales que se ejecutan de forma iterativa. Esto incluye:

- Comprensión de negocios
- Adquisición y comprensión de datos
- moldeado
- Despliegue
- Aceptación del cliente

Comprensión de negocios

Los objetivos de esta etapa son:

- Especificar de forma Clara y explícita el o los objetivos del modelo como preguntas "señas" que se utilizan para impulsar la participación del cliente.
- Especificar claramente dónde encontrar las fuentes de datos de interés.
- Definir el objetivo del modelo predictivo en este paso y determinar si se necesita extraer datos auxiliares de otras fuentes.

En esta etapa, se trabaja con los clientes y partes interesadas para comprender los problemas comerciales que pueden mejorarse significativamente mediante el análisis predictivo. Un objetivo central de este paso es identificar las variables clave del negocio (como el pronóstico de ventas o la probabilidad de que un pedido sea fraudulento) que el análisis debe predecir para satisfacer estos requisitos. También se debe desarrollar una comprensión de las fuentes de datos necesarias para abordar los objetivos del proyecto desde una perspectiva analítica. Hay dos aspectos principales de esta etapa: definir objetivos e identificar fuentes de datos.

Adquisición y comprensión de datos

Los objetivos de esta etapa son:

- Incluir los datos en el entorno analítico de destino
- Determinar si los datos que tenemos se pueden usar para responder a la pregunta.

En esta etapa, comenzará a desarrollarse el proceso para mover los datos desde la ubicación de origen a las ubicaciones de destino donde se ejecutarán las operaciones de análisis como el entrenamiento y las predicciones.

Antes de entrenar a los modelos, necesita desarrollarse un profundo conocimiento de los datos. Los datos del mundo real a menudo son desordenados con datos incompletos o incorrectos. Mediante el resumen de los datos y la visualización de los datos, puede identificarse rápidamente la calidad de los datos e informar sobre cómo manejar la calidad de estos.

La visualización de datos puede ser particularmente útil para responder a preguntas como: ¿Se ha medido las características lo suficiente de manera consistente como para que sean útiles o hay muchos valores que faltan en los datos? ¿Se han recopilado los datos de forma consistente durante el período de tiempo de interés o hay bloques de observaciones faltantes? Si los datos no pasan esta comprobación de calidad, es posible que se tenga que volver al paso anterior para corregir u obtener más datos.

De lo contrario, puede comenzarse a comprender mejor los patrones inherentes a los datos que ayudarán a desarrollar un modelo predictivo sólido para los objetivos planteados. Específicamente, buscar evidencia de lo bien conectados que están los datos con el objetivo y si

los datos son lo suficientemente grandes como para avanzar en los siguientes pasos. A medida que se determina si los datos están conectados o si se tienen suficientes datos, es posible que se necesite encontrar nuevas fuentes de datos con datos más precisos o relevantes para completar el conjunto de datos identificado inicialmente en la etapa anterior. TDSP también proporciona una utilidad automatizada llamada IDEAR para ayudar a visualizar los datos y preparar informes de resumen de datos.

Además de la ingestión inicial de datos, normalmente se tendrá que configurar un proceso para puntuar nuevos datos o actualizar los datos regularmente como parte de un proceso de aprendizaje continuo. Esto se puede hacer configurando una canalización de datos o un flujo de trabajo. En esta etapa se desarrolla una arquitectura de solución de la canalización de datos.

Modelado

Los objetivos de esta etapa son:

- Desarrollar nuevos atributos o características de datos (también conocidas como ingeniería de características), para construir el modelo de Machine Learning.
- Construir y evaluar un modelo informativo para predecir el objetivo.
- Determinar si tenemos un modelo que sea adecuado para uso en producción

Hay dos aspectos principales en esta etapa: ingeniería de características y formación de modelos.

La ingeniería de características implica la inclusión, agregación y transformación de variables sin procesar para crear las características utilizadas en el análisis. Si se quiere una idea de lo que está impulsando el modelo, entonces se necesita entender cómo las características están relacionadas entre sí y cómo el método de Machine Learning utilizará esas características. Este paso requiere una combinación creativa de la experiencia del dominio y la información obtenida del paso de exploración de datos. Debe existir un equilibrio entre incluir variables informativas sin incluir demasiadas variables no relacionadas. Las variables informativas mejorarán el resultado; las variables no relacionadas introducirán ruido innecesario en el modelo.

En cuanto a la formación de modelos dependiendo del tipo de pregunta que esté tratando de responder, hay múltiples opciones de algoritmos de modelado disponibles.

El proceso de formación de modelos es:

- Los datos de entrada para el modelado generalmente se dividen al azar en un conjunto de datos de entrenamiento y un conjunto de datos de prueba.
- Los modelos se construyen utilizando el conjunto de datos de entrenamiento.
- Evaluar (conjunto de datos de entrenamiento y prueba) una serie de algoritmos de Machine Learning de la competencia junto con los diversos parámetros de ajuste

asociados (también conocidos como barrido de parámetros) que están orientados a responder a la pregunta de interés con los datos que tenemos actualmente a mano.
- Determinar la "mejor" solución para responder a la pregunta comparando la métrica de éxito entre métodos alternativos.

Despliegue

El objetivo de esta etapa es:

Desplegar los modelos y pipelines (flujo automatizado de procesamiento del código fuente y datos) en un entorno de producción o similar para la aceptación del usuario final.

Una vez que se tiene un conjunto de modelos que funcionan bien, se pueden poner en funcionamiento para que otras aplicaciones los consuman. Dependiendo de los requisitos del negocio, las predicciones se hacen en tiempo real o por lotes. Para ser operacionales, los modelos tienen que estar expuestos con una interfaz API abierta que se consume fácilmente desde varias aplicaciones, como sitios web en línea, hojas de cálculo, paneles o líneas de negocio y aplicaciones de backend. También es una buena idea incorporar la telemetría y la supervisión del despliegue del modelo de producción y la canalización de datos para ayudar con los informes de estado del sistema y la solución de problemas.

Aceptación del cliente

El objetivo de esta etapa es:

Finalizar los entregables del proyecto confirmando los pipelines, el modelo y su implementación en un entorno de producción.

El cliente validaría que el sistema satisfaga sus necesidades comerciales y las respuestas a las preguntas con una precisión aceptable para implementar el sistema en producción para su uso por la aplicación de su cliente. Toda la documentación está finalizada y revisada. Se realiza una transferencia del proyecto a la entidad responsable de las operaciones que podría ser un equipo de TI o Data Science en el cliente o un agente del cliente que será responsable de ejecutar el sistema en producción.

En resumen, el ciclo de vida del TDSP, se modela como una secuencia de pasos iterados que proporcionan orientación sobre las tareas necesarias para utilizar modelos predictivos que se pueden implementar en un entorno de producción para la creación aplicaciones inteligentes. El objetivo de este ciclo de vida del proceso es continuar moviendo un proyecto de Data Science hacia un fin claro. Si bien es cierto que la Data Science es un ejercicio de investigación y descubrimiento, poder comunicar esto claramente a los clientes utilizando un conjunto bien definido de artefactos en una plantilla estandarizada puede ayudar a evitar malentendidos y aumentar las probabilidades de éxito.

Roles del TDSP

En el TDSP se describen objetivos, tareas y artefactos de documentación de cada fase del ciclo de vida. Estas tareas y artefactos están asociados con roles de proyecto:

- Arquitecto de soluciones
- Gerente de proyecto
- Ingeniero de datos
- Científico de datos
- Desarrollador de aplicaciones
- Líder de proyecto

En el siguiente diagrama se proporciona una vista de las tareas (en azul) y los artefactos (en verde) asociados con cada fase del ciclo de vida (eje horizontal) de estos roles (eje vertical).

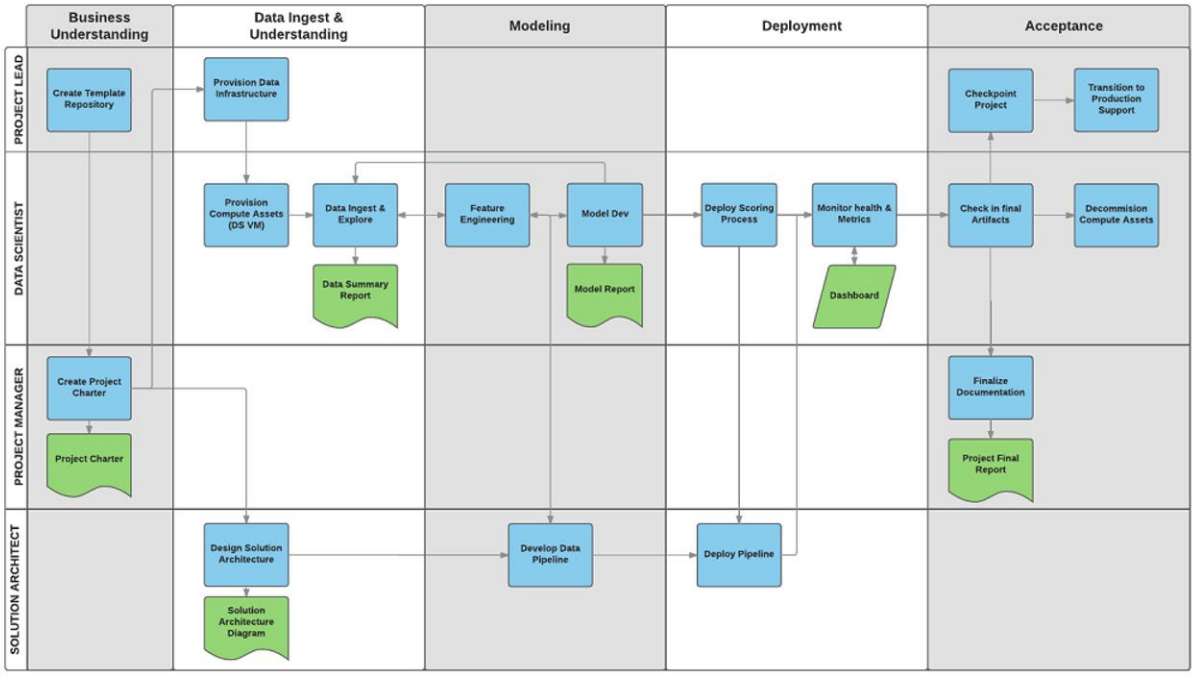

Figura 13. Fases del ciclo de vida TDSP

Estructura de proyecto estandarizada

Cuando todos los proyectos comparten una estructura de directorio y usan plantillas para los documentos del proyecto, resulta fácil para los miembros del equipo encontrar la información. Todo el código y los documentos se almacenan en un sistema de control de versiones (VCS), como Git, TFS o Subversion para permitir la colaboración en equipo. El seguimiento de tareas y características, en un sistema de seguimiento de proyectos ágil, como Jira, Rally y Azure DevOps permite seguir más de cerca el estado del desarrollo del código. Este seguimiento también permite a los equipos obtener mejores estimaciones de los costos. TDSP recomienda crear un repositorio independiente para cada proyecto en el VCS para el mejorar la seguridad de la información y la colaboración. Una estructura estandarizada para todos los proyectos ayuda a crear conocimiento institucional en toda la organización.

Este framework proporcionan plantillas para la estructura de carpetas y documentos necesarios en ubicaciones estándar. Esta estructura de carpetas organiza los archivos que contienen código para la exploración de datos y la extracción de características, y que registran las iteraciones de los modelos. Estas plantillas permiten a los miembros del equipo comprender el trabajo que otros realizan, y agregar nuevos miembros a los equipos de forma fácil. Las plantillas se pueden ver y actualizar fácilmente en formato de marcado. Estas plantillas pueden usarse para proporcionar listas de comprobación con preguntas clave en cada proyecto y de esta forma garantizar que los problemas estén bien definidos y que los resultados entregados satisfagan la calidad esperada. Algunos ejemplos son:

- Una carta de constitución de proyecto para documentar los problemas empresariales y el ámbito del proyecto
- Informes de datos para documentar la estructura y las estadísticas de los datos sin procesar
- Informes de modelo para documentar características derivadas
- Métricas de rendimiento de modelo, como curvas ROC o MSE

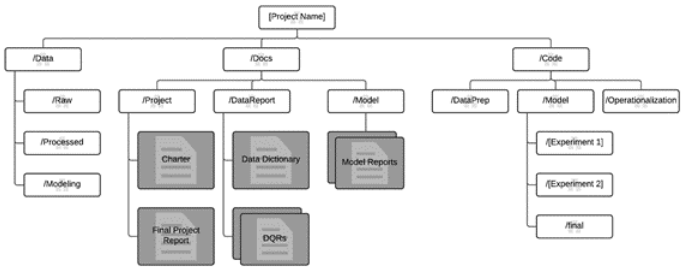

Figura 14. Ejemplo de Estructura de un Proyecto TDSP

Infraestructura y recursos para los proyectos de Data Science

TDSP proporciona recomendaciones para administrar análisis compartido e infraestructura de almacenamiento, por ejemplo:

- Sistemas de archivos en la nube para almacenar conjuntos de datos

- Bases de Datos
- Clústeres de macrodatos (SQL o Spark)
- Servicio de Machine Learning

La infraestructura de análisis y almacenamiento, donde se almacenan los conjuntos de datos sin procesar y los procesados, puede estar en la nube o en un entorno local. Esta infraestructura permite un análisis reproducible. También evita la duplicación, lo que puede llevar a incoherencias y costos de infraestructura innecesarios. Se proporcionan herramientas para aprovisionar los recursos compartidos, realizar un seguimiento de ellos y permitir que cada miembro del equipo se conecte a dichos recursos de forma segura. También es una buena práctica pedir a los miembros del proyecto que creen un entorno de proceso coherente. Luego, diferentes miembros del equipo pueden replicar y validar los experimentos.

Herramientas y utilidades para la ejecución de proyectos

En la mayoría de las organizaciones la introducción de procesos presenta ciertos desafíos. Las herramientas proporcionadas para implementar el proceso y el ciclo de vida de Data Science ayudan a reducir las barreras a su adopción y la normalizan. TDSP proporciona un conjunto inicial de herramientas y scripts para impulsar su adopción dentro de un equipo. También ayuda a automatizar algunas de las tareas comunes del ciclo de vida de Data Science, como la exploración de datos y el modelado de línea de base. Existe una estructura bien definida que se proporciona a los individuos para que contribuyan con herramientas y utilidades compartidas al repositorio de código compartido de su equipo. Estos recursos se pueden aprovechar luego en otros proyectos dentro del equipo o en la organización. Microsoft proporciona herramientas extensas en Azure Machine Learning que admiten marcos de código abierto (Python, R, ONNX y Deep Learning) y también herramientas propias de Microsoft como AutoML.

2.5 Machine Learning Operations (MLOps)

La creación de productos de Machine Learning (ML) o características de productos asistidos por Machine Learning implica dos disciplinas distintas:

- **Desarrollo de modelos:** Los científicos de datos, altamente cualificados en estadística, álgebra lineal y cálculo, entrenan, evalúan y seleccionan el modelo estadístico o de red neuronal de mejor rendimiento.
- **Despliegue de modelos:** Los desarrolladores, altamente cualificados en diseño e ingeniería de software, construyen un sistema de software robusto, lo implementan en la nube y lo escalan para atender un gran número de solicitudes a los modelos de forma simultáneas.

Por supuesto, eso es una simplificación pues se requieren varios otros conocimientos vitales para construir productos útiles y exitosos asistidos por ML:

- **Ingeniería de datos:** Construir pipelines de datos para recopilar datos de fuentes dispares, curarlos y transformarlos, y convertirlos en datos homogéneos y limpios que se pueden utilizar de forma segura para modelos de entrenamiento.

- **Diseño de productos:** Comprender las necesidades comerciales, identificar objetivos y matrices de negocios relevantes; definir características de productos o historias de usuarios para esos objetivos, reconocer los problemas subyacentes que ML es más adecuado para resolver; diseñar la experiencia del usuario no solo para utilizar la predicción del modelo de ML sin problemas con el resto de las características del producto, sino también recopilar acciones de los usuarios para mejorar los modelos.

- **Análisis de seguridad:** Asegurar que el sistema de software, los datos y el modelo estén seguros, y que no se revele ninguna información de identificación personal (PII) combinando los resultados del modelo y otra información o datos disponibles públicamente.

- **Ética de la IA:** garantizar el cumplimiento de todas las leyes aplicables y agregar medidas para proteger contra cualquier tipo de sesgo (por ejemplo, limitar el alcance del modelo, agregar supervisión humana, etc.)

A medida que se implementan más modelos en producción, la importancia de MLOps ha crecido de forma natural. Cada vez hay más enfoque en el diseño y el funcionamiento sin fisuras de los modelos de Machine Learning dentro del producto en general. El desarrollo de modelos no se puede hacer en un silo dadas las consecuencias que puede tener en el producto y el negocio.

Necesitamos un ciclo de vida de Machine Learning que esté en sintonía con las realidades de los productos asistidos por Machine Learning y los MLOps. Debería facilitar la visibilidad de los Stakeholders, sin causar demasiados cambios en los flujos de trabajo existentes de los científicos e ingenieros de datos.

2.5.1 Ciclo de vida del Machine Learning

Los científicos de datos han estado construyendo modelos estadísticos y de redes neuronales durante más de una década. A menudo, estos modelos se utilizaban sin conexión (es decir, se ejecutaban manualmente) para el análisis predictivo.

El desarrollo de modelos consiste en dos conjuntos de actividades: preparación de datos y formación de modelos. El ciclo de vida tradicional del Machine Learning comienza con la formulación de un problema de Machine Learning y termina con evaluaciones de modelos.

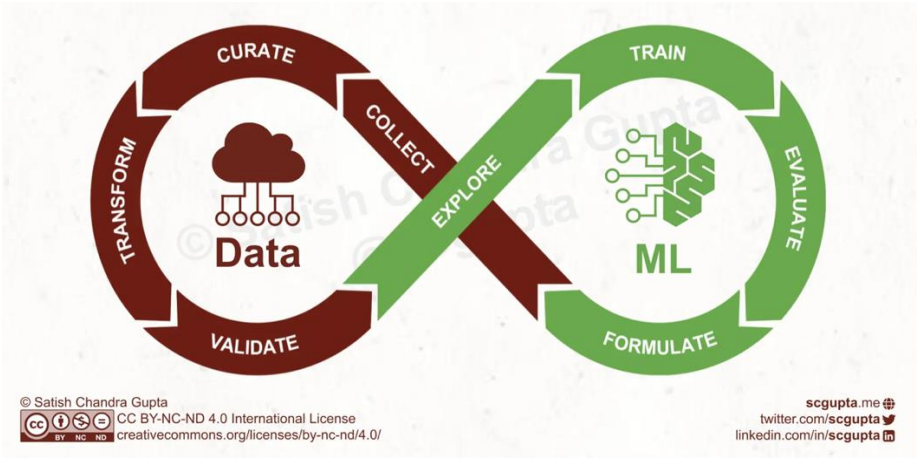

Figura 15. Ciclo de vida del Machine Learning: bucle Data-ML para el desarrollo de modelos

2.5.1.1 Formular

Los científicos de datos traducen un Objetivo de negocio en un problema de Machine Learning. Hay varios factores que a tomar en cuenta:

- **Objetivo de negocio:** Reducir a un pequeño conjunto de problemas de Machine Learning que pueden servir al objetivo de negocio.

- **Costo de los errores:** Ningún modelo de Machine Learning puede ser 100 % preciso. ¿Cuál es el costo de los falsos positivos y falsos negativos? Por ejemplo, si un modelo de clasificación de imágenes predice erróneamente el cáncer de mama en una persona sana, otras pruebas lo rectificarán. Pero si el modelo no diagnostica el cáncer en un paciente, entonces puede llegar a ser fatal debido a la detección tardía.

- **Disponibilidad de datos:** puede suceder que se inicie sin datos y se comience su recopilación. A medida que los datos se enriquecen, más tipos de modelos son viables. Por ejemplo, si tuviera que hacer la detección de anomalías sin datos etiquetados, puede comenzar con varios tipos de algoritmos de agrupación no supervisados y marcar puntos que no están en ningún clúster como anomalías. Pero a medida que se recopilen las reacciones de los usuarios a su modelo, se tendrá un conjunto de datos etiquetados. Entonces es posible que se deba probar si un modelo de clasificación supervisado funcionará mejor.

- **Métricas de evaluación:** Dependiendo de la formulación del problema, también debe especificarse una métrica de rendimiento del modelo para optimizar, que debe alinearse con la métrica de negocio para su objetivo de negocio.

2.5.1.2 Recolección de Datos

Recopilar los datos necesarios de aplicaciones internas, así como de fuentes externas. Puede ser mediante el *scrapping* de páginas web, la captura de flujos de eventos de una aplicación móvil o servicio web, registros de aplicaciones, etc.

2.5.1.3 Curación de Datos

Los datos recopilados casi nunca son infalibles. Necesitan limpiarse, eliminar duplicados, rellenar valores faltantes y almacenarlo en la zona adecuada, ya sea un Data Lake o Data Warehouse. Si es para entrenar un modelo de Machine Learning supervisado, también tendrán que etiquetarse. Además, debe catalogarse para que pueda ser fácilmente analizado y entendido correctamente. Idealmente debe automatizarse todo lo posible, pero habrá procesos que deberán hacerse manualmente (el etiquetado de datos, por ejemplo).

2.5.1.4 Transformar

Una vez que se hayan limpiado los datos, pueden transformarse para que se adapten al análisis y al modelado de Machine Learning. Esto puede requerir cambiar la estructura, unirse con otras tablas, agregar o resumir a lo largo de dimensiones importantes, calcular características adicionales, etc.

2.5.1.5 Validar

Implementar comprobaciones de calidad, mantener registros de distribuciones estadísticas a lo largo del tiempo y crear disparadores para alertar cuando alguna de las comprobaciones falla o la distribución se balancea más allá de los límites esperados. Los ingenieros de datos, en consulta con los científicos de datos, implementan estas validaciones en los pipelines de datos.

2.5.1.6 Explorar

Los científicos de datos realizan un análisis exploratorio de datos (EDA) para comprender las relaciones entre varias características y el valor objetivo que requiere que el modelo pueda predecir. También hacen ingeniería de características, lo que probablemente conduzca a la adición de más comprobaciones de transformación y validación (las dos etapas anteriores).

2.5.1.7 Entrenamiento

Los científicos de datos entrenan múltiples modos, realizan experimentos, comparan el rendimiento del modelo, ajustan los hiperparámetros y seleccionan un par de modelos de mejor rendimiento.

2.5.1.8 Evaluar

Evaluar las características del modelo en comparación con los objetivos y métricas del negocio. Algunos comentarios pueden resultar incluso en ajustar y formular el problema de Machine Learning de manera diferente, y repetir todo el proceso de nuevo.

Este bucle infinito de Data-ML no es lineal. En cada etapa, no siempre se avanza a la siguiente etapa. Al descubrir los problemas, se vuelve a la etapa anterior relevante para ajustarlos.

Es similar al bucle de DevOps que siguen los desarrolladores. No todos los códigos que van a la etapa de prueba progresan a la versión. Si las pruebas fallan, se vuelve a la etapa de Código (a veces incluso a Plan) para que se rectifiquen los problemas.

2.5.2 Ciclo de vida del desarrollo de software

El bucle infinito de DevOps es el estándar de facto para el ciclo de vida del desarrollo de software para construir e implementar rápidamente aplicaciones y servicios de software en la nube.

Consiste en dos conjuntos de actividades: diseño y desarrollo de un sistema de software, y despliegue y supervisión de servicios y aplicaciones de software.

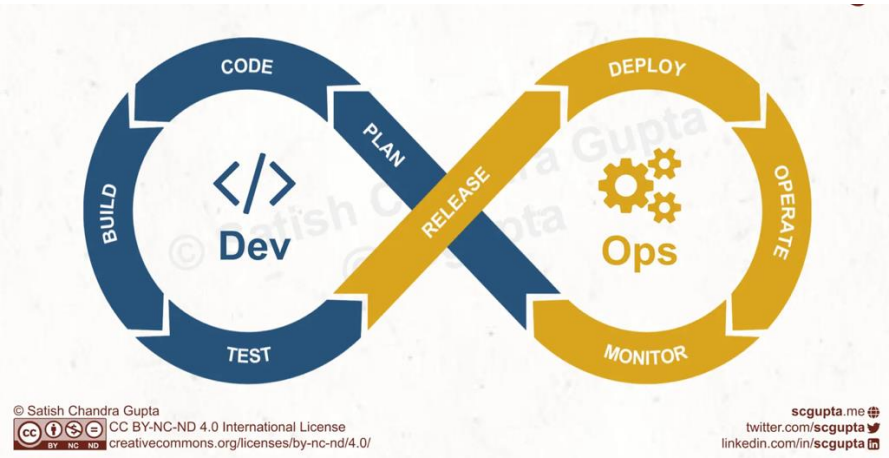

Figura 16. Ciclo de vida de desarrollo de software: DevOps bucle para el desarrollo de software

2.5.2.1 Plan

Esta es la primera etapa para cualquier producto o característica del producto. Se discuten los objetivos y las métricas de negocio clave, y qué características del producto pueden ayudar a lograrlos. Profundizar en los problemas del usuario final y debatir sobre los flujos de los usuarios para abordar esos problemas y recopilar los datos necesarios para evaluar el rendimiento de un modelo de Machine Learning en el mundo real.

2.5.2.2 Codificación

Diseño y desarrollo del software, el producto o la aplicación de extremo a extremo, y no solo los modelos de ML. Se establecen contratos y APIs que el código de la aplicación utiliza para invocar la inferencia del modelo y consumir sus resultados, y también qué reacciones y comentarios de los usuarios se recopilarán.

Es muy importante que los desarrolladores, ingenieros de datos y científicos de datos lleguen a acuerdos. Eso reducirá las sorpresas desagradables más adelante.

2.5.2.3 Construcción

Esta etapa alimenta la integración continua de varias partes a medida que evolucionan y se empaquetan para ser desplegadas. Puede ser una biblioteca o un SDK, una imagen de Docker o un binario de aplicaciones (por ejemplo, apk para aplicaciones de Android).

2.5.2.4 Pruebas

Pruebas unitarias, pruebas de integración, pruebas de cobertura, pruebas de rendimiento, pruebas de carga, pruebas de privacidad, pruebas de seguridad y pruebas de sesgo. Todo tipo de pruebas de software y modelos de Machine Learning son aplicables en esta etapa.

Las pruebas se realizan en un entorno de pre-producción que es similar al entorno de producción objetivo, pero no está diseñado para una escala similar. Puede tener datos ficticios, artificiales o anónimos para probar el sistema de software de extremo a extremo.

2.5.2.5 Liberación

Una vez que todas las pruebas automatizadas pasan y, en algunos casos, los resultados de las pruebas se inspeccionan manualmente, el código o los modelos de software se aprueban para su lanzamiento. Al igual que el código, los modelos también deben ser versionados y los metadatos necesarios capturados automáticamente. Al igual que las imágenes de Docker se versionan en un repositorio de Docker, el modelo también debe persistir en un repositorio de modelo.

Si los modelos se empaquetan junto con el código para el microservicio que sirve al modelo, entonces la imagen del Docker también tiene la imagen del modelo. Aquí es donde termina la integración y la implementación continuas se hace cargo.

2.5.2.6 Despliegue

Recoger los artefactos liberados del repositorio Docker o de modelos y desplegarlos en la infraestructura de producción. Dependiendo de sus necesidades, puede elegir Infraestructura como Servicio (IaaS), Contenedor como Servicio (CaaS) o Plataforma como Servicio (PaaS).

También puede utilizarse Tensor Flow Serve, PyTorch Serve o servicios como SageMaker y Vertex AI para implementar los servicios del modelo.

2.5.2.7 Operación

Una vez que se implementan los servicios, es posible que se decida enviar primero un pequeño porcentaje del tráfico. Canary Deployment es una táctica común para actualizar en fases (p. ej. 2%, 5%, 10%, 25%, 75%, 100%). En caso de un problema, un comportamiento inesperado o una caída en las métricas, puede revertirse la implementación.

Una vez que la puerta esté abierta al 100% del tráfico, la infraestructura de implementación debería sustituir el servicio antiguo. También debería escalar a medida que la carga alcanza su punto máximo y cae. Kubernetes y KubeFlow son herramientas comunes para este propósito.

2.5.2.8 Monitoreo

En esta fase final, se supervisa constantemente el estado de los servicios, los errores, las latencias, las predicciones del modelo, los valores atípicos y la distribución de las características del modelo de entrada, etc. En caso de que surja un problema, dependiendo de la gravedad y el diagnóstico, puede revertirse el sistema a una versión anterior, lanzar una revisión, activar el reciclaje del modelo o hacer cualquier otra cosa que sea necesaria.

2.5.3 Ciclo de vida de MLOps

Es bastante común que los científicos de datos desarrollen un modelo y hagan llegar a los desarrolladores e ingenieros de Machine Learning para que se integren con el resto del sistema y lo desplieguen en producción.

Los silos de ML y Dev y la propiedad fragmentada son una de las razones más comunes por las que muchos proyectos de ML fracasan. La unificación del modelo y el desarrollo de software en un ciclo de vida de Machine Learning proporciona una visibilidad muy necesaria para todas las partes interesadas.

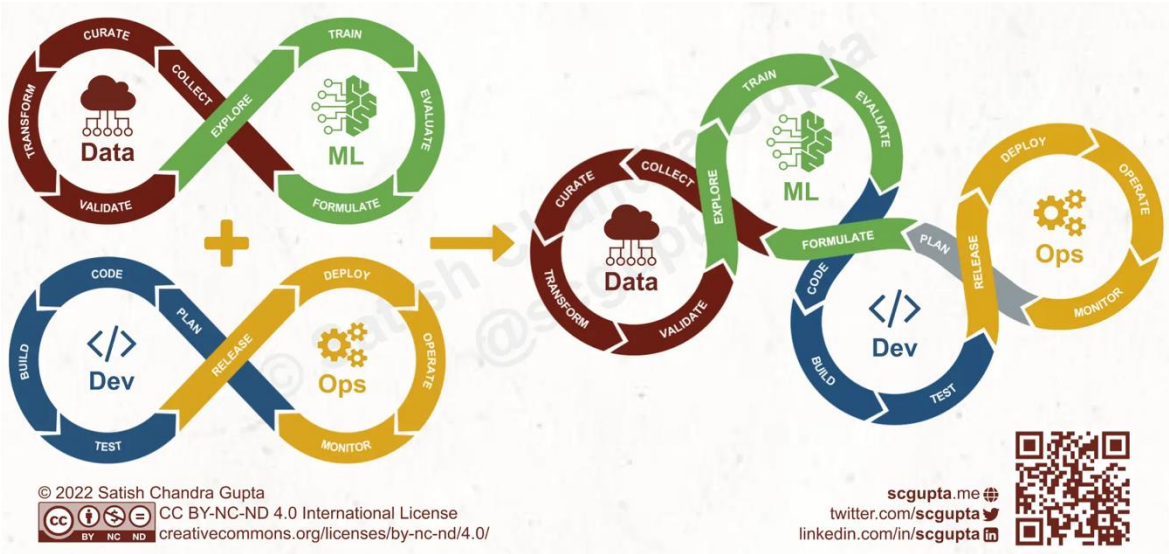

Figura 17. Ciclo de vida de MLOps: El desarrollo de modelos y el desarrollo de software deben unificarse en un ciclo de vida de Machine Learning unificado

2.5.3.1 La fase de planeación es el punto de partida

La planificación de los productos viene antes que todo lo demás. La definición de los objetivos de negocio y el diseño de las experiencias de los usuarios deben incluir no solo la funcionalidad del producto, sino también cómo los resultados del modelo y la captura de las reacciones de los usuarios se combinarán en el diseño de la producción.

A diferencia del software tradicional, cuando se recopilan más datos con el tiempo, la experiencia del usuario de los aspectos de Machine Learning de un producto puede necesitar una actualización para beneficiarse de él, a pesar de que no hay una "nueva funcionalidad".

2.5.3.2 Primero se construye el producto sin ML

A menudo se construye primero una aplicación de extremo a extremo con una heurística o un modelo ficticio basado en reglas, cortando por completo el bucle Data-ML. Eso funciona como un modelo de referencia y es útil para recopilar datos. También da contexto a los científicos de datos al mostrar cómo se utilizará el modelo en el producto.

2.5.3.3 Cadencia diferente para el desarrollo de modelos y software

Desarrollar un modelo de Machine Learning es bastante diferente a desarrollar software. Los sistemas de software se pueden desarrollar de forma incremental (con algunas partes que no funcionan). A diferencia de las piezas de software, los modelos de ML no se pueden dividir en una granularidad fina.

Un solo ciclo de vida no impide que las ruedas de Data, ML, Dev y Ops giren a diferentes velocidades. De hecho, ya sucede en DevOps. En algunos equipos, no todos los Sprints de

desarrollo dan como resultado una nueva versión que se despliega. Por otro lado, algunos equipos implementan nuevas versiones cada hora, es decir, cientos de veces en un solo sprint.

2.5.3.4 Equipos autogestionados, Integrar Temprano e Iterar Frecuentemente

Estas son tres aspectos deseables para mejorar la tasa de éxito en el desarrollo e implementación de productos asistidos por Machine Learning:

- **Equipos autogestionados:** fomentar en los equipos un alto grado de organización multifuncional, excelencia técnica, compromiso, que sea responsable del proyecto de extremo a extremo.

- **Integrar temprano:** Implementar un modelo simple (basado en reglas o ficticio) y desarrollar primero una característica del producto por completo.

- **Iterar frecuentemente:** construir mejores modelos y reemplazar los anteriores, monitorear y repetir.

2.6 Propuesta de Modelo de Ciclo de Vida para Proyectos Ágiles de TI Basados en Data Science

El Ciclo de Vida propuesto toma las características de los Ciclos de Vida Agiles convencionales e incorpora los aspectos particulares de los proyectos de data Science.

Una única aproximación técnica del Data Science como herramienta no es suficiente, para que lograr el éxito de este tipo de proyectos. Un enfoque de producto es de suma importancia dado que alinea las estrategias de trabajo del equipo hacia lograr resultados sobre la base de los objetivos de negocio a través del producto como tal. El Ciclo de Vida propuesto toma en cuenta esta perspectiva y se representa en el siguiente diagrama:

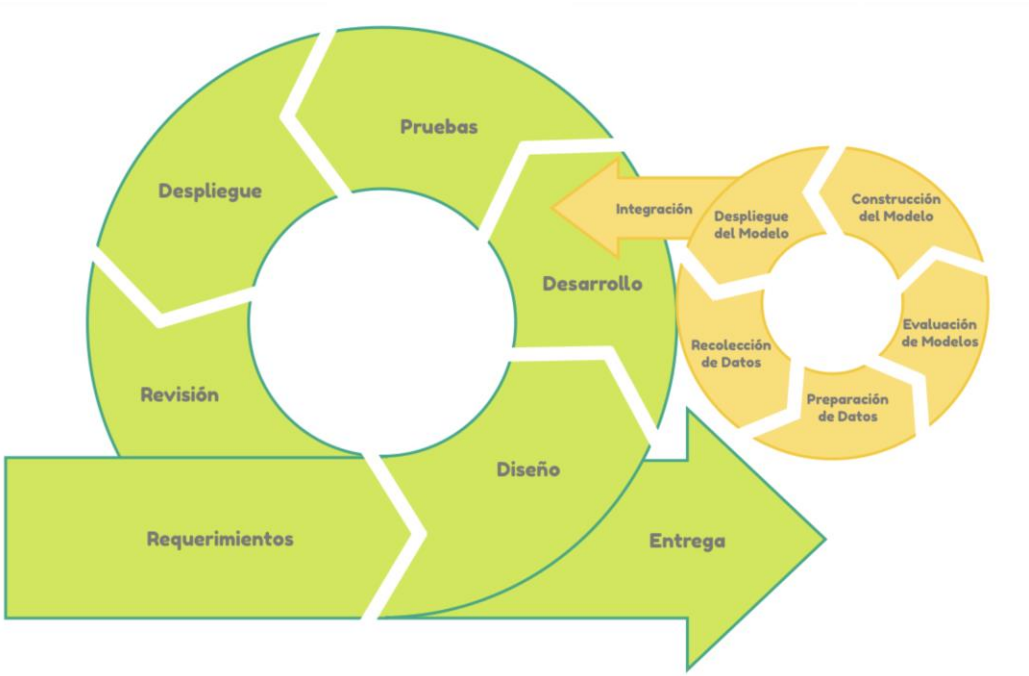

Figura 18. Modelo de Ciclo de Vida para Proyectos Agiles de TI Basados en Data Science

El Ciclo de Vida Ágil pasa por cada una de las etapas de desarrollo de manera iterativa, es decir al llegar a la última etapa se termina una iteración e inmediatamente después inicia una nueva desde la etapa inicial. Esta forma de desarrollo habilita la Inspección y Adaptación del proceso de desarrollo del producto. Cada iteración genera un incremento potencialmente entregable y el proceso completo finaliza al haber alcanzado el objetivo del producto.

Se integra el componente de Data Science que toma en cuenta el proceso particular del manejo de datos en la etapa de Desarrollo del Ciclo de Vida.

Los Modelos de Datos son el resultado del proceso de Data Science, los Modelos son componentes del producto y en escenarios complejos, dependencias a integrar dentro del mismo. La generación de Modelos y su despliegue no necesariamente coinciden con el ciclo de

iteraciones sin embargo es necesario sincronizarlos siempre que sea posible, por tal motivo para el Ciclo de Vida propuesto su desarrollo es paralelo al desarrollo del software.

2.6.1 Fases del Ciclo de Vida

A continuación, se describen cada una de las fases del ciclo de vida propuesto.

2.6.1.1 Requerimientos

El propósito de esta etapa inicial es tener identificados y documentados en el Backlog del Producto los requisitos para la iteración. Se define el objetivo de la iteración, las características útiles y las limitaciones de soporte. Esto permitirá tener un enfoque claro sobre qué lograr y también limita la adición de características innecesarias al diseño del sistema.

2.6.1.2 Diseño

En la etapa de diseño, el equipo de desarrollo analiza los requerimientos de la iteración y planifica el mejor curso de acción, el mejor marco y las mejores herramientas para lograr la mejor calidad. El empleo de diagramas y pruebas de concepto es común es esta etapa.

2.6.1.3 Desarrollo

En la etapa de desarrollo se construye un incremento del producto que satisfaga el objetivo definido en la fase de requerimientos. En esta etapa también de forma paralela se ejecuta el proceso de Data Science con las siguientes etapas:

2.6.1.4 Recolección de datos

En esta etapa se identifican y obtienen las fuentes de datos relevantes para el producto. Dependiendo del caso puede ser necesario explorar conjuntos de datos externos como datos de dominio público o de pago disponibles o, por el contrario, si solo se quiere lograr un Objetivo de negocio específico dentro de la organización, pueden explorarse datos internos de la organización ERP, CRM, DW, etc.

Idealmente, debe recopilarse la mayor cantidad de datos posible para extraer patrones significativos e información procesable utilizando un enfoque más completo.

2.6.1.5 Preparación de datos

Una vez se han adquirido los datos sin procesar, se preparan para que estos provean algún valor. Los datos organizados de forma altamente estructurada suponen una ventaja, los datos no estructurados deben transformase a un formato específico (archivos separados por comas, columnas separadas por tabulaciones, estructura de filas de columnas, etc.) y que resulten útiles para su procesamiento.

Después de reacondicionar los datos, deben eliminarse todos los problemas subyacentes con las propias entradas de datos: inconsistencias, duplicidad, vacíos, anomalías, etc.

Un aspecto que tomar en cuenta es que, durante toda la transformación y limpieza de datos, debe cumplirse con la licencia de uso de datos, así como con todas las legislaciones relativas a la privacidad y la protección de los datos personales y políticas de la organización.

Consideraciones al respecto debe identificarse en la fase de requerimientos para identificar riesgos que podrían poner en riesgo el alcance de los objetivos del producto.

2.6.1.6 Evaluación de modelos

En esta etapa, se tienen los datos formateados y ordenados en un gran número de categorías o variables. En este punto se empieza a explorar y trazar la relación entre estas variables.

La evaluación de modelos permite analizar, comprender y proporciona una visión general en profundidad de los datos, lo que le ayuda a elegir un mejor modelo para representar los datos.

En la mayoría de los casos, tendrá que aplicarse un *Análisis Exploratorio de Datos (EDA por sus siglas en ingles*) para escoger un modelo de formación para los datos. Puede utilizarse la visualización y las fórmulas estadísticas para utilizar varias técnicas de EDA, incluyendo gráficos de barras, gráficos de líneas, gráficos circulares, histogramas, gráficos de cajas, análisis de tendencias, etc.

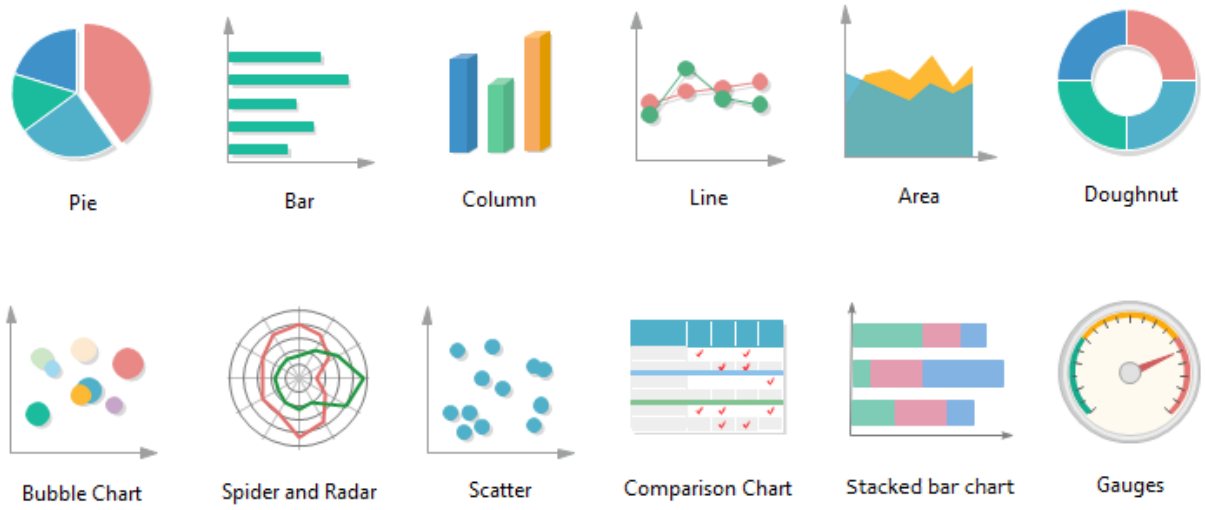

Figura 19. Tipos de gráficos para la visualización de datos (Li, s.f.)

2.6.1.7 Construcción del modelo

Una vez realizado un análisis exhaustivo de los datos y decidido un modelo adecuado, se desarrollan las pruebas el modelo real.

Antes de construir el modelo, se dividen los datos en conjuntos de datos de entrenamiento y pruebas. Normalmente, el conjunto de datos de entrenamiento constituye el 80% de los datos, mientras que el conjunto de datos de prueba consiste en el 20% restante.

En primer lugar, se emplean datos de entrenamiento para construir el modelo, y luego se utilizan datos de prueba para evaluar si el modelo desarrollado funciona correctamente o no.

Hay varias bibliotecas empaquetadas en diferentes lenguajes de programación (R, Python, MATLAB), que pueden utilizarse para construir el modelo con solo introducir los datos de entrenamiento etiquetados y algoritmos definidos para distintos tipos de problemas.

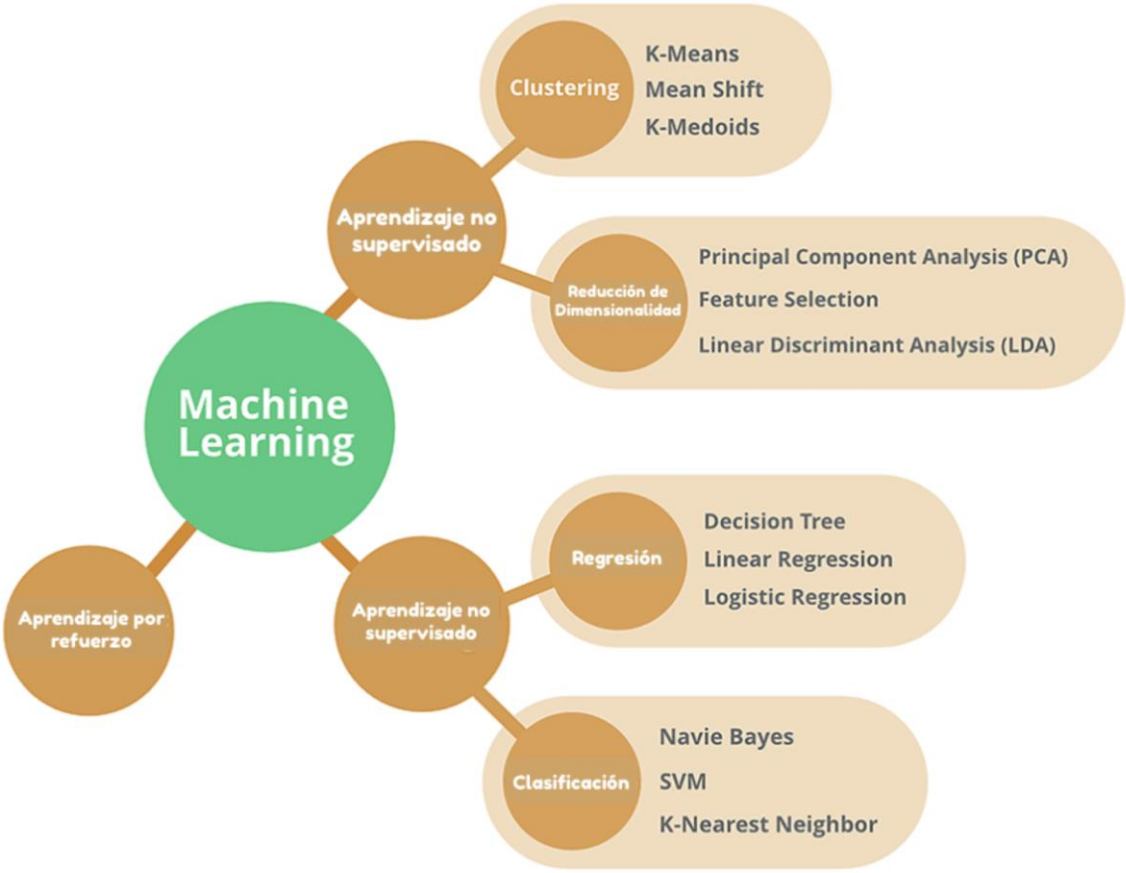

Figura 20. Algoritmos de Machine Learning para la construcción de modelos

Si se entrena un modelo y no encaja bien para los datos de prueba, debe volverse a entrenar el modelo utilizando más datos de entrenamiento o probar una técnica de aprendizaje diferente, es decir, regresión, clasificación, asociación, etc.

2.6.1.8 Despliegue del Modelo

El despliegue del modelo se realiza primero en un entorno de preproducción para proceder a las pruebas correspondientes donde se asegure que este servirá como solución antes de implementarlo para clientes reales y luego en producción para realizar un seguimiento de su rendimiento a lo largo del tiempo.

Además, debe revisarse regularmente para evaluar si ha habido cambios significativos dentro del entorno en el que operaba el modelo. Si ese es el caso, debe volverse a entrenar el modelo para incorporar todos los cambios significativos que ocurrieron durante el último período de innovación y comercialización.

Prácticas como el MLOps son recomendables para llevar a cabo el proceso de despliegue. MLOps es un enfoque útil para la creación y la calidad de soluciones de Machine Learning e Inteligencia Artificial. Al adoptar un enfoque de MLOps, los científicos de datos pueden aumentar el ritmo de desarrollo y producción de modelos, mediante la implementación de prácticas de integración y despliegue continuos (CI/CD) con el monitoreo, la validación y la gobernanza adecuados de los modelos de ML.

2.6.1.9 Integración

Para que las evaluaciones de los Modelos de Datos puedan ser útiles a los usuarios, estos deben poder integrarse con las interfases gráficas del producto. La etapa de integración busca precisamente esto. Una forma de realizar la integración puede ser a través de una REST API que permitan el intercambio de la información entre el Modelo y las instancias clientes.

2.6.1.10 Pruebas

Luego de generar el código, y tener una versión del modelo listo para esta etapa, se prueba con los requisitos para asegurarse de que se están resolviendo las necesidades del producto abordadas y recopiladas durante la etapa de requisitos.

Durante esta etapa, se realizan pruebas unitarias, pruebas de integración, pruebas de sistema y pruebas de aceptación.

2.6.1.11 Despliegue

Después de la codificación y la verificación a través de pruebas, se implementan los nuevos cambios realizados en la iteración para generar un incremento en el producto. Esta nueva versión debe ser potencialmente entregable a los usuarios y debe permitir a los Stakeholders Inspeccionar el progreso al objetivo del Producto y su Adaptación.

2.6.1.12 Revisión

Tras la integración del sistema de las nuevas versiones de software, los usuarios y los Stakeholders proporcionan feedback o comentarios con respecto al estado del producto. Todos los comentarios deben abordarse en la siguiente iteración y, por lo tanto, comenzar nuevamente el ciclo de iteraciones.

2.6.1.13 Entrega

Si desde la perspectiva de los Stakeholders el estado del producto es suficiente para que este pueda ser entregado a los usuarios finales entonces este es puesto en producción.

2.6.2 Implementación del ciclo de vida

La implementación del ciclo de vida se basa en el framework Scrum y retoma algunas de las prácticas planteadas por el Data Driven Scrum. Los eventos Scrum y prácticas de DDS se relacionan con las fases del ciclo de vida como se muestran en el siguiente diagrama.

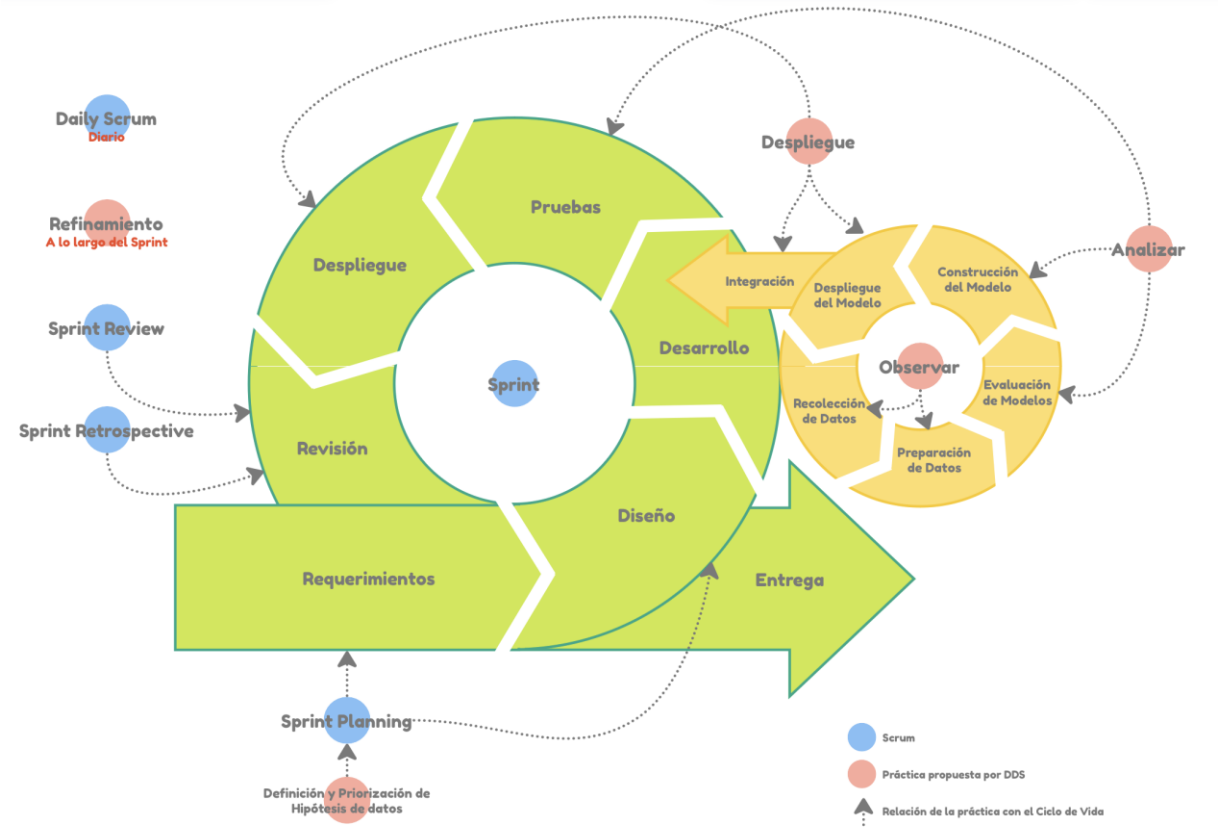

Figura 21. Implementación del Ciclo de Vida con Scrum y DDS

El *Sprint* corresponde al tiempo fijado para la iteración. Todas las fases del ciclo de vida se desarrollan dentro de un *Sprint*. Uno de los retos de los en los proyectos Data Science es la incertidumbre y variabilidad en el esfuerzo estimado para completar las tareas relacionadas con el procesamiento de los datos. La sugerencia es mantener *Sprints* de la misma longitud en la que se definan metas de producto a lograr en la iteración y que tengan en cuenta esta particularidad. Esto supone también sincronizar los avances del producto que no tengan que ver con los datos con los que sí preferiblemente en el menor tiempo posible. Esto se realiza en la fase de *Integración*.

La fase de *Requerimientos* y de *Diseño* están contempladas en el *Scrum Planning*, en este evento también se retoma la práctica del DDS de identificar y priorizar las hipótesis sobre los datos que serán validadas dentro de la iteración. El *Scrum Planning* no es la única oportunidad en donde la etapa de *Requerimientos* y *Diseño* ocurren, diariamente la *Daily Scrum* brinda una oportunidad para adaptar el diseño del software, los datos y en general el producto y sus requerimientos.

La *Daily Scrum* tiene como propósito inspeccionar el progreso hacia el Sprint Goal, es decir a esa meta de producto que se espera lograr en la iteración. Se realiza de forma diaria y es una gran oportunidad para compartir los resultados de las evaluaciones de los modelos con los stakeholders adaptar el rumbo con base a su feedback y planear las sincronizaciones mencionadas anteriormente. Este evento tiene influencia en todas las fases del Ciclo de Vida por las conversaciones que en ella se generan y que nos habilitan para tomar decisiones sobre las otras fases.

El DDS también propone en su flujo de trabajo además de la *Definición y Priorización de las Hipótesis* sobre los datos, la *Observación* y el *Análisis* de las tareas relacionadas a estas hipótesis y su posterior despliegue. Como se mencionaba anteriormente en la fase de *Requerimientos* se realiza la definición de hipótesis, mientras que la *Observación* es llevada a cabo en las fases de *Recolección* y *Preparación de Datos*. Luego de esta, el *Análisis* de los resultados es realizada durante la *Evaluación* y *Construcción de Modelos* asimismo en la fase de *Pruebas* se realizan los procesos de aseguramiento de calidad en la que se evalúan los casos en los que se verifican y validan que estos resultados sean los esperados.

Otra práctica del DDS es el *Despliegue*, su implementación corresponde en el Ciclo de Vida a las fases de *Despliegue del Modelo* e *Integración*, esto pudiera no suceder en todos los Sprints o al contrario suceder en múltiples ocasiones, lo importante es realizar esa sincronización lo más frecuentemente posible para que estos se incorporen en los incrementos del producto. Además, se retoma en la fase de *Despliegue* del Ciclo de Vida en donde se despliegan los incrementos de software junto a los Modelos.

El *Refinamiento* es una práctica que, aunque no es parte de Scrum, es realizada por equipos Scrum continuamente en el transcurso de los *Sprints*. DDS menciona de manera explícita esta práctica. El refinamiento permite tener claridad sobre los requerimientos futuros y adaptarlos con forme se aprende y se conoce más del producto en cada *Sprint*. Permite al equipo tener un panorama a futuro sobre las tareas y sus dependencias, estimar su esfuerzo e identificar prioridades, es una gran herramienta que apoya la planeación del trabajo de los equipos.

Finalmente, en la fase de *Revisión* se llevan a cabo la *Sprint Review*, en la que se inspecciona el incremento, se recoge el feedback de los usuarios y Stakeholders y se adapta el rumbo del producto y las prioridades para la siguiente iteración. También la *Sprint Retrospective*, en la que se identifican oportunidades de mejora en los procesos, herramientas e interacciones, que permitan mejorar el flujo de trabajo del equipo.

Capítulo 3: Metodología de la Investigación

3.1 Enfoque Metodológico

El enfoque metodológico seleccionado es el método comparativo debido a que es particularmente efectivo para este estudio con pequeñas muestras. En un estudio comparativo se establecen los parámetros para el cotejo inicial de las unidades de análisis (Bray, Adamson, & Mason, 2007).

La comparación se realizó con base a los frameworks de Data Science disponibles y su mayor o menor afinidad con el Ciclo de Vida propuesto y las condiciones de los proyectos.

Metodología	Pros	Contras	Herramientas
Data Driven Scrum (DDS)	• Iteraciones basadas en capacidad. • Se ajusta con organizaciones SCRUM. • Flexible para varios ciclos de vida. • Revisiones y retrospectivas en cada iteración.	• No es comprensible, es necesario un entrenamiento. • Cuando un recurso se libera antes de tiempo debe esperar próxima iteración para tomar una asignación si el backlog está vacío.	• Jira. • Rally. • Herramientas que soporten Kanban.
Crisp-DM	• Generalizable. • Perceptible o intuitivo. • Fácil de utilizar. • De inicio confiable. • Resultados seguros. • Flexible.	• Demasiada documentación. • Reuniones generales difíciles de dominar.	• Monday.com. • Wrike. • Project Manager. • Jira
Team Data Science Process (Microsoft TDSP)	• Comprensible • Inclusión opcional de Scrum • Mantenibilidad	• Ciertas inconsistencias. • Empinada ruta de aprendizaje. • Aspectos específicos de Microsoft	• Github • Azure Pipelines • Azure Boards • Azure Monitor • Visual Studio • AKS

Tabla 4. Tabla comparativa de Metodologías agiles para Data Science

Al ser las condiciones de la organización más propicias a adoptar un framework de trabajo más apegado a Scrum, la alternativa de Data Driven Scrum es la más favorable en el contexto de la investigación y la escogida para la implementación del Ciclo de Vida.

3.2 Diseño de la Investigación

Partimos de la hipótesis que un Ciclo de Vida Ágil que incorpora las particularidades del proceso de Data Science impacta positivamente en los componentes de Satisfacción del Cliente, Calidad de la Solución y Bienestar del Equipo en los proyectos de TI basados en estas tecnologías.

Para llevar a cabo la comprobación de esta hipótesis se implementará un Ciclo de Vida con estas características en dos proyectos a través de la adopción del framework de Data Driven Scrum, adecuado a las realidades del equipo, que mantenga la esencia de la guía Scrum, pero tomando en cuenta la naturaleza del proceso de Data Science.

3.3 Selección de los Casos de Estudio

Para la puesta en marcha de la investigación se escogieron dos Casos de Estudio correspondientes a equipos reales de una empresa salvadoreña que desarrollan productos que incluyen componentes de Modelos de Data Science y que se encuentran en fase productiva, es decir cuentan con usuarios activos.

Caso de Estudio A

El *Caso de estudio A* consiste en un proyecto para el desarrollo de una herramienta capaz de analizar datos relevantes para la identificación y predicción de clientes potenciales. El equipo está conformado por alrededor de 10 personas, entre los roles de Developers que incluyen Ingenieros de Software y de Científicos de Datos, Scrum Master, Product Owner y Project Manager. El proyecto lleva más de 6 meses desde su inicio.

Caso de Estudio B

El *Caso de estudio B* consiste en un proyecto de desarrollo de una herramienta para la automatización de procesos de reclutamiento de una empresa. El equipo de alrededor de 8 personas también está compuesto por desarrolladores, incluyendo Ingenieros de Software y Científicos de Datos, Scrum Master, Product Owner y Project Manager. Del mismo modo el proyecto lleva más de 6 meses desde su inicio.

3.4 Métricas e instrumentos

El impacto de la implementación del Ciclo de Vida se cuantifica a través con la medición de sus resultados. Los parámetros para confirmar su efectividad se enfocan en tres componentes principales descritos a continuación:

1. **Satisfacción del Cliente**: este componente se define como un proceso de evaluación de la experiencia de los Stakeholders del producto, donde se comparan sus expectativas con los resultados alcanzados (Giese & Cote, 2000). Desde una perspectiva agile la satisfacción del cliente se logra mediante la entrega temprana y continua de software con valor. (Agile Alliance, 2001)
2. **Calidad de la Solución**: para este componente definimos calidad como el grado en el que un conjunto de características inherentes (al producto) cumple con los requisitos (Pinto, 2020), sin embargo, no se limita a esto, sino también a la transparencia en cuando a lo que calidad significa para los Stakeholders y equipos y a los procesos que permitan una mejora continua en el desarrollo del producto.
3. **Bienestar del equipo de trabajo**: por último, este componente toma en cuenta el impacto en las personas y sus relaciones los proyectos se desarrollan en torno a individuos motivados por lo que debe que proveérseles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo. (Agile Alliance, 2001)

3.4.1 Satisfacción del Cliente

3.4.1.1 Capacidad de Respuesta (Perspectiva de los Stakeholders)

Los equipos son más efectivos cuando son capaces de responder rápidamente a las necesidades emergentes de las partes interesadas (Stakeholders). Esto incluye nuevas ideas, oportunidades comerciales y también problemas o errores.

3.4.1.2 Felicidad de los Stakeholders

Se definen equipos efectivos como aquellos que logran satisfacer a sus stakeholders con resultados valiosos y que tienen miembros del equipo satisfechos.

3.4.1.3 Valor del equipo (Perspectiva de los Stakeholders)

Los equipos son más efectivos cuando las partes interesadas consideran valioso y útil el trabajo que se realiza.

3.4.2 Calidad de la Solución

3.4.2.1 Atención por la Calidad

Los equipos son más efectivos cuando tienen una fuerte preocupación por la calidad. Esto significa que con frecuencia hablan de ello y exploran formas de mejorarlo.

3.4.2.2 Calidad (Perspectiva de los Stakeholders)

Los equipos son más eficaces cuando las partes interesadas aprecian la calidad que ofrecen. Esto significa que su trabajo está libre de problemas graves y que las partes interesadas pueden obtener apoyo cuando sea necesario.

3.4.3 Bienestar del equipo de trabajo

3.4.3.1 Seguridad Psicológica

Los equipos son más efectivos cuando operan en entornos que hacen que sea seguro para los miembros asumir riesgos interpersonales. Por ejemplo, admitir que no saben algo, ofrecer retroalimentación, ofrecer ayuda o pedir ayuda.

3.4.3.2 Moral del Equipo

La moral del equipo es la voluntad de los equipos de seguir adelante incluso si las cosas se ponen difíciles. La moral alta es un amortiguador contra los contratiempos y el agotamiento y un gran indicador de la atmósfera en un equipo.

3.4.4 Instrumento de medición

El instrumento empleado para la medición de los parámetros descritos anteriormente es la **Scrum Team Survey (STS)** (The Liberators, 2023) una encuesta diseñada para tal fin.

La encuesta es una herramienta para apoyar equipos ágiles en su ciclo de mejora continua, crear transparencia sobre el estado actual y ayuda a los equipos a identificar mejoras y hacer adaptaciones.

Las métricas empleadas en esta investigación están basadas en los estudios realizados por Christiaan Verwijs y Daniel Russo a partir de su modelo de efectividad de equipos Scrum (Verwijs & Russo, 2022).[1]Las métricas empleadas para la realización de la evaluación de los componentes se basan en los estudios realizados por Christiaan Verwijs y Daniel Russo donde

[1] A partir de las respuestas de las encuestas la herramienta asigna una serie de pesos a las diferentes respuestas y calcula la efectividad en cada una de sus áreas. Al ser las fórmulas empleadas para los cálculos propiedad intelectual de los autores, quedan fuera del alcance de la investigación la tabulación y cálculo de los resultados y se limita a presentarlos tal cual como la herramienta los entrega.

se identifican los factores que son impactan directamente en la efectividad de equipos agiles. (Verwijs & Russo, 2022) .[2]

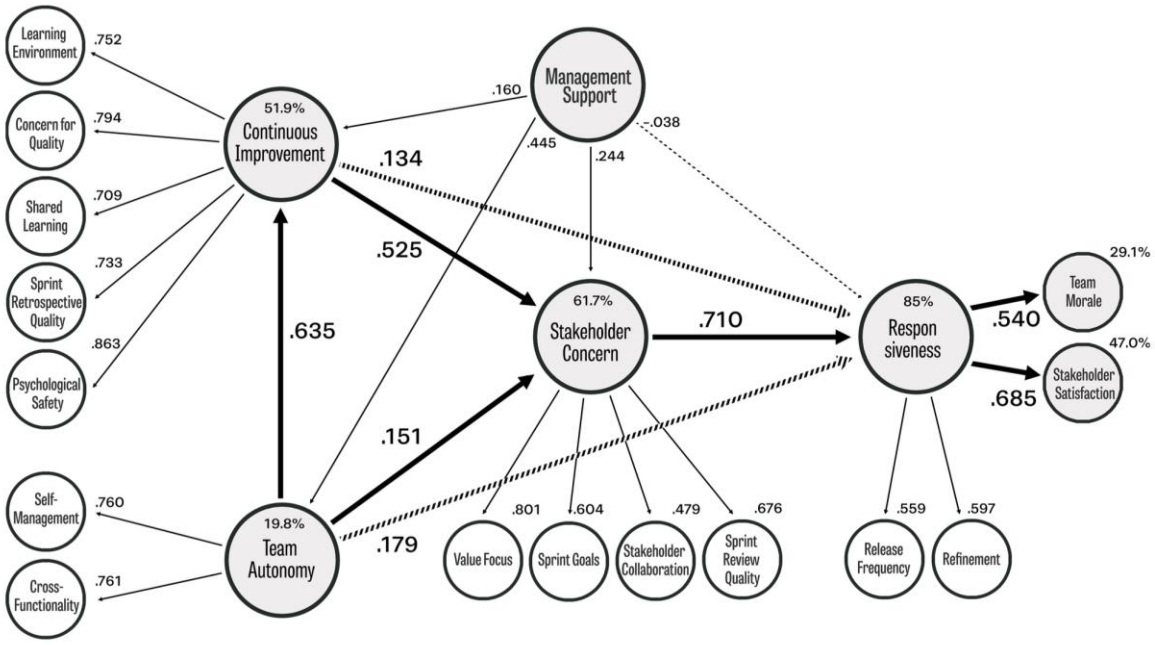

Figura 22. Relaciones entre los factores de efectividad de equipos Scrum en la STS

La encuesta realiza una evaluación de seis áreas:

1. Mejora Continua
2. Preocupaciones de los Stakeholders
3. Efectividad del Equipo
4. Capacidad de Respuesta
5. Autonomía del Equipo
6. Apoyo del área de Gestión

Sin embargo, la medición para el experimento se limitará estas áreas del modelo:

1. Mejora Continua
2. Preocupaciones de los Stakeholders
3. Efectividad del Equipo

[2] A partir de las respuestas de las encuestas la herramienta asigna una serie de pesos a las diferentes respuestas y calcula la efectividad en cada una de sus áreas. Al ser las fórmulas empleadas para los cálculos propiedad intelectual de los autores, quedan fuera del alcance de la investigación la tabulación y cálculo de los resultados y se limita a presentarlos tal cual como la herramienta los entrega.

Forma de medición

Las mediciones se realizaron en dos momentos. La primera iteración de la encuesta se realizó previo a la implementación del Ciclo de Vida y sus resultados han servido como línea base sobre la que se contrastaron los resultados de la segunda iteración cuatro semanas después.

Participantes

La Encuesta esta supuesta a ser completada por todos los miembros del equipo que participen de manera activa en el proyecto además de los Stakeholders.

La encuesta tiene 3 claves diferentes (*véase Anexos*) dependiendo del rol del participante:

- Miembro del Equipo Scrum (Scrum Master, Product Owner, Developers)
- Stakeholders (Cliente)
- Roles de Suporte (Managers de la Organización, Gerente de Proyecto u otros)

Cada uno provee información valiosa desde su perspectiva que en conjunto permite una medición más precisa del estado de la efectividad del equipo.

Capítulo 4: Resultados

Los resultados descritos en este capítulo corresponden a la primera y segunda iteración de las encuestas para cada caso de estudio. Los resultados de cada iteración se inspeccionaron junto con los equipos en una sus reuniones de Retrospectiva brindando así transparencia y dando una oportunidad al equipo para la adaptación del Ciclo de Vida.

Los diagramas presentados en cada resultado de las iteraciones de la encuesta muestran los resultados completos del STS, estos toman en cuenta el conjunto de las áreas de efectividad de equipos Scrum como se describe en la sección *Instrumento de medición* del capítulo anterior. De estos resultados se han extraído los puntajes que corresponden a los componentes estudiados.

4.1 Primera iteración (Línea Base)

La organización del equipo de trabajo previo a la implementación del ciclo de vida propuesto consta de un grupo de ingenieros de datos que pertenecen a la compañía donde se implementa el proyecto piloto trabaja de manera conjunta con el administrador de proyecto asignado a cada iniciativa de negocio, este a su vez vigila que el producto de ciencia de datos pase por el equipo de control de calidad y después de las sesiones de presentación de avances obtienen la aprobación del cliente. Respecto a la metodología y marco de trabajo utilizado estos equipos implementan una metodología SCRUM apegando las piezas de ciencia de datos como una pieza de software tradicional.

Roles del equipo a evaluar:

- 2 dueños del producto (Product Owner).
- 1 administrador de proyecto.
- 2 analistas de datos.
- 2 ingenieros de Machine Learning para creación de modelos de datos.
- 1 ingeniero y científico de datos.
- 2 ingenieros de revisión de la calidad.

La primera iteración de la encuesta establece la línea base sobre la cual se evaluó el impacto del ciclo de vida en cada uno de los equipos.

4.1.1 Caso de Estudio A

Los resultados iniciales corresponden a 10 encuestas y muestran un puntaje relativamente alto en los componentes a evaluar, de estos el componente de *Bienestar del Equipo de Trabajo* muestra el puntaje más bajo.

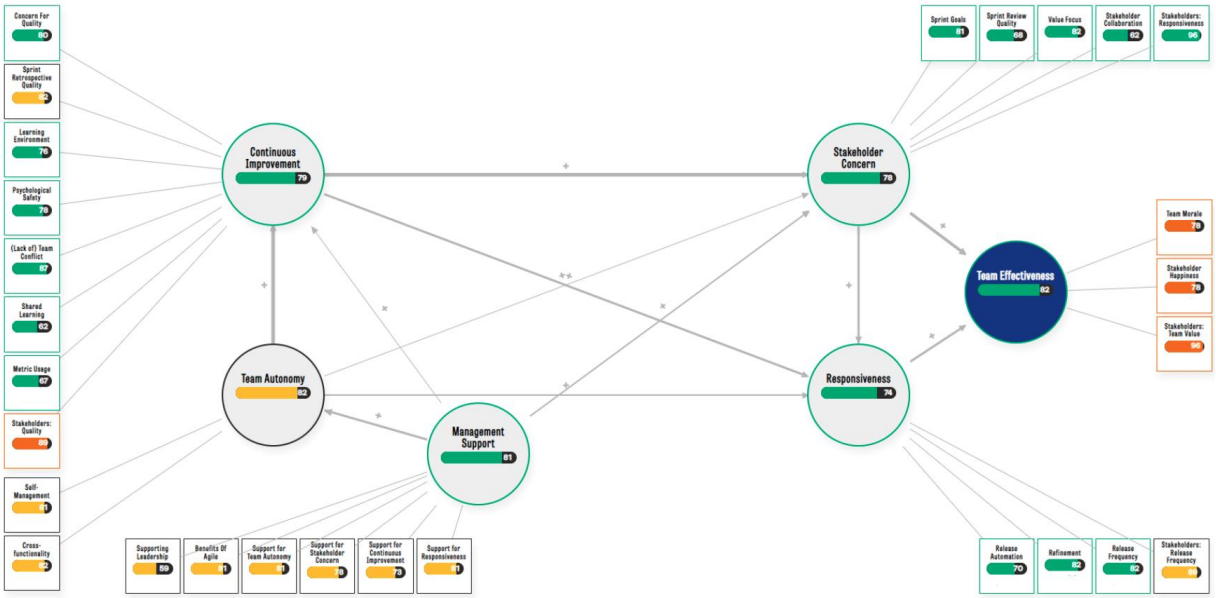

Figura 23. Resultados Primera Iteración Caso de Estudio A

En las siguientes tablas se muestran el detalle de los subcomponentes evaluados:

Satisfacción del Cliente	**Puntaje**
Capacidad de Respuesta (Perspectiva de los Stakeholders)	*96*
Felicidad de los Stakeholders	*78*
Valor del equipo (Perspectiva de los Stakeholders)	*96*
Promedio	***90***

Tabla 5. Resultados del Componente Satisfacción del Cliente, Línea Base Caso de Estudio A

Calidad de la Solución	**Puntaje**
Atención por la Calidad	*80*
Calidad (Perspectiva de los Stakeholders)	*89*
Promedio	***84.5***

Tabla 6. Resultados del Componente Calidad de la Solución, Línea Base Caso de Estudio A

Bienestar del equipo de trabajo	**Puntaje**
Seguridad Psicológica	*78*
Moral del Equipo	78
Promedio	***78***

Tabla 7. Resultados del Componente Bienestar del Equipo de Trabajo, Línea Base Caso de Estudio A

4.1.2 Caso de Estudio B

Del mismo modo los resultados iniciales para este Caso de Estudio corresponden a 8 encuestas y también muestran un puntaje relativamente alto. El componente de *Bienestar del Equipo de Trabajo* destaca como área de oportunidad.

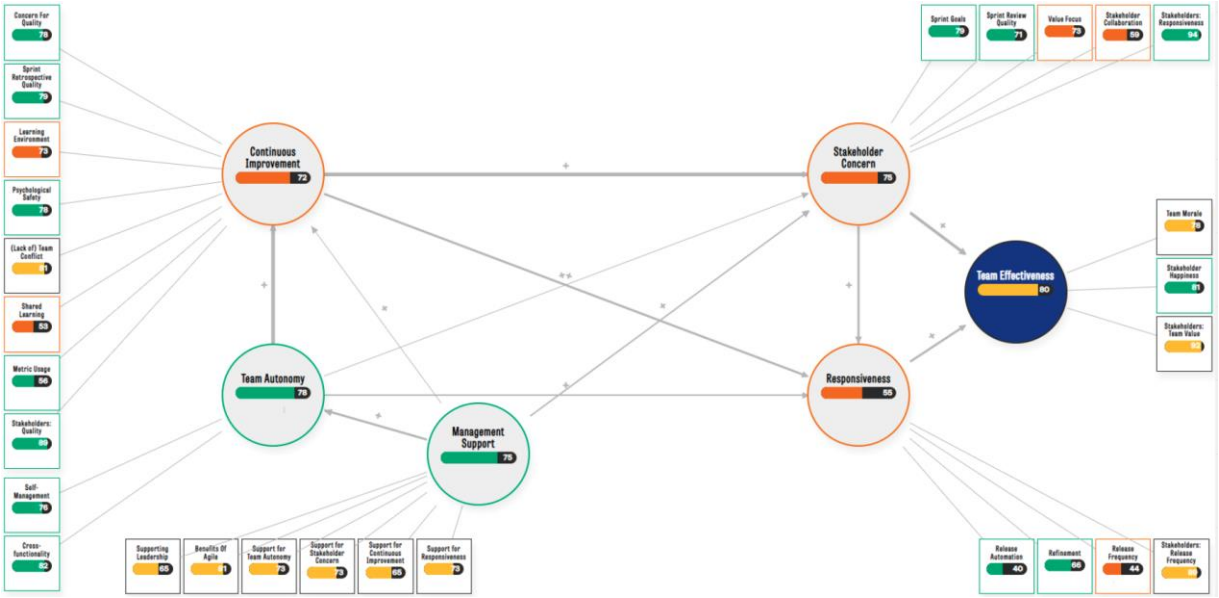

Figura 24. Resultados Primera Iteración Caso de Estudio B

Las siguientes tablas muestran el detalle de los subcomponentes evaluados:

Satisfacción del Cliente	**Puntaje**
Capacidad de Respuesta (Perspectiva de los Stakeholders)	*94*
Felicidad de los Stakeholders	*81*
Valor del equipo (Perspectiva de los Stakeholders)	*92*
Promedio	***89***

Tabla 8. Resultados del Componente Satisfacción del Cliente, Línea Base Caso de Estudio B

Calidad de la Solución	**Puntaje**
Atención por la Calidad	*78*
Calidad (Perspectiva de los Stakeholders)	*89*
Promedio	***83.5***

Tabla 9. Resultados del Componente Calidad de la Solución, Línea Base Caso de Estudio B

Bienestar del equipo de trabajo	**Puntaje**
Seguridad Psicológica	*78*
Moral del Equipo	78
Promedio	***78***

Tabla 10. Resultados del Componente Bienestar del Equipo de Trabajo, Línea Base Caso de Estudio B

4.2 Segunda Iteración

Cuatro semanas después de la primera iteración se realizó nuevamente la encuesta para medir el impacto de la incorporación de las sugerencias basadas en el Ciclo de Vida propuesto en las formas de trabajo de los equipos, los resultados fueron los siguientes:

Roles del equipo a evaluar:

- 2 dueños del producto (Product Owner).
- 1 administrador de proyecto.
- 2 analistas de datos.
- 2 ingenieros de Machine Learning para creación de modelos de datos.
- 1 ingeniero y científico de datos.
- 2 ingenieros de revisión de la calidad.

4.2.1 Caso de Estudio A

Los resultados de la segunda iteración de la encuesta corresponden a las respuestas de los mismos 10 participantes que contestaron la primera vez. Se obtuvo un mejor puntaje en el promedio total del componente de Satisfacción del Cliente al igual que en el de Calidad de la Solución, sin embargo, el componente de Bienestar del Equipo de Trabajo tuvo una disminución.

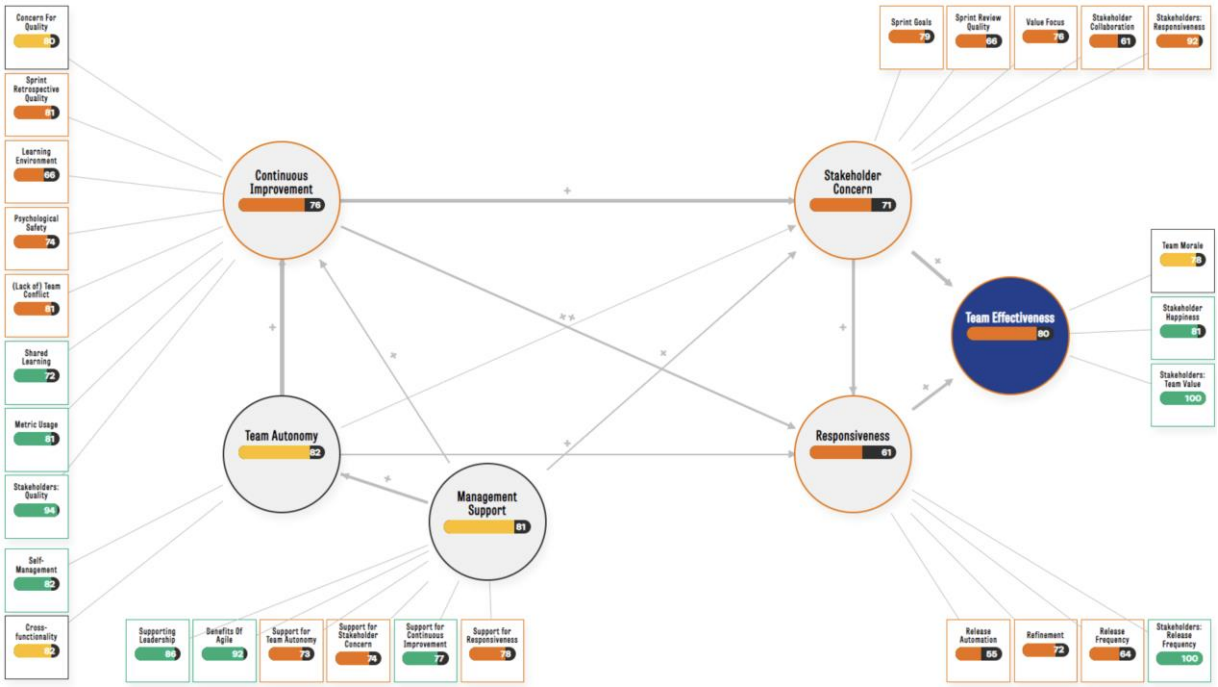

Figura 25. Resultados Segunda Iteración Caso de Estudio A

Las siguientes tablas muestran el detalle de los subcomponentes evaluados:

Satisfacción del Cliente	**Puntaje**	**Variación**
Capacidad de Respuesta (Perspectiva de los Stakeholders)	*92*	*-4*
Felicidad de los Stakeholders	*81*	*+3*
Valor del equipo (Perspectiva de los Stakeholders)	*100*	*+4*
Promedio	***91***	

Tabla 11. Resultados del Componente Satisfacción del Cliente, Segunda Iteración Caso de Estudio A

Calidad de la Solución	**Puntaje**	**Variación**
Atención por la Calidad	*80*	*0*
Calidad (Perspectiva de los Stakeholders)	*94*	*+5*
Promedio	***87***	

Tabla 12. Resultados del Componente Calidad de la Solución, Segunda Iteración Caso de Estudio A

Bienestar del Equipo de Trabajo	**Puntaje**	**Variación**
Seguridad Psicológica	*74*	*-4*
Moral del Equipo	78	0
Promedio	***76***	

Tabla 13. Resultados del Componente Bienestar del Equipo de Trabajo, Segunda Iteración Caso de Estudio A

4.2.2 Caso de Estudio B

Asimismo, los resultados de esta segunda iteración corresponden a las respuestas de las 8 personas que contestaron la primera vez. En este caso también se obtuvo un mejor puntaje en el promedio total del componente de Satisfacción del Cliente y en el de Calidad de la Solución, el componente de Bienestar del Equipo de Trabajo tuvo una disminución.

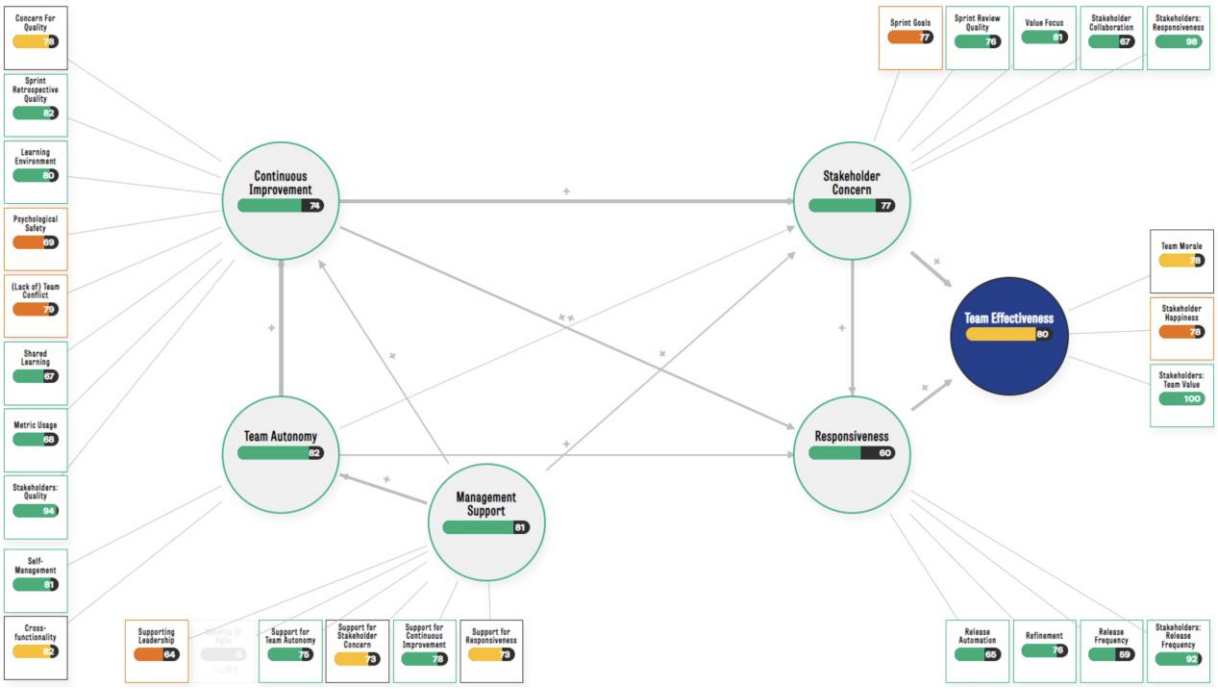

Figura 26. Resultados Segunda Iteración Caso de Estudio B

Las siguientes tablas muestran el detalle de los subcomponentes evaluados:

Satisfacción del Cliente	**Puntaje**	**Variación**
Capacidad de Respuesta (Perspectiva de los Stakeholders)	*98*	*+4*
Felicidad de los Stakeholders	*78*	*-3*
Valor del equipo (Perspectiva de los Stakeholders)	*100*	*+8*
Promedio	***92***	

Tabla 14. Resultados del Componente Satisfacción del Cliente, Segunda Iteración Caso de Estudio B

Calidad de la Solución	**Puntaje**	**Variación**
Atención por la Calidad	*78*	*0*
Calidad (Perspectiva de los Stakeholders)	*94*	*+5*
Promedio	***86***	

Tabla 15. Resultados del Componente Calidad de la Solución, Segunda Iteración Caso de Estudio B

Bienestar del equipo de trabajo	**Puntaje**	**Variación**
Seguridad Psicológica	*69*	*-8*
Moral del Equipo	78	0
Promedio	***73.5***	

Tabla 16. Resultados del Componente Bienestar del Equipo de Trabajo, Segunda Iteración Caso de Estudio B

Capítulo 5: Discusión

5.1 Resultados generales

Al analizar los resultados obtenidos en ambos equipos luego de implementar las adaptaciones con base al ciclo de vida propuesto, estos mejoraron sus puntajes para los componentes de Satisfacción del Cliente y Calidad de la Solución. Buena parte de estos resultados se deben a que el equipo incorporaba ya en su forma de trabajo muchas de las fases del ciclo de vida, esto facilitó en gran manera que la adopción fuese más efectiva en los equipos y explica los altos puntajes en los resultados de la encueta desde la línea base.

En ambos casos de estudio, un sólido dominio de Scrum como framework de trabajo, su enfoque en la frecuente entrega de valor al cliente y la perspectiva de producto en el desarrollo, habilitan de manera positiva la adopción del ciclo de vida.

El componente de *Bienestar del Equipo de Trabajo* se ve influenciado en gran medida por la relación entre el equipo y los Stakeholders y la manera en que ambas partes perciben el valor generado. En cada caso de estudio hubo motivos distintos por los que este componente se vio afectado, sin embargo, el ciclo de vida como tal estima eventos de inspección y adaptación de la efectividad del equipo, puntualmente las *Sprint Retrospectivas*, que permiten dar respuesta a estas preocupaciones.

5.2 Caso de estudio A

El ciclo de vida permitió al equipo realizar una Primera Retrospectiva en la que se identificaron los retos que se presentaban en cuanto a los componentes a mejorar. Parte de las preocupaciones tenían que ver con: la forma en la que se espera que los usuarios utilicen la herramienta y comprender el fujo de trabajo que los usuarios realizan manualmente, para ser capaces de automatizar estos procesos. Estas fueron de las principales oportunidades de mejora.

Adicionalmente el feedback provisto por los Stakeholders, desde la perspectiva del equipo, podía ser aún mejor para ayudar a definir más claramente los objetivos del producto y de esa manera enfocar sus esfuerzos y evitar reprocesos.

Con base en esto se incorporaron ajustes retomando el ciclo de vida propuesto para en las prácticas de *Refinamiento* y el evento de *Sprint Review*. Las acciones tomadas al respecto de estos puntos de mejora permitieron al equipo mejorar en los componentes de *Satisfacción del Cliente* y *Calidad de la Solución* al final de la segunda iteración de la encuesta.

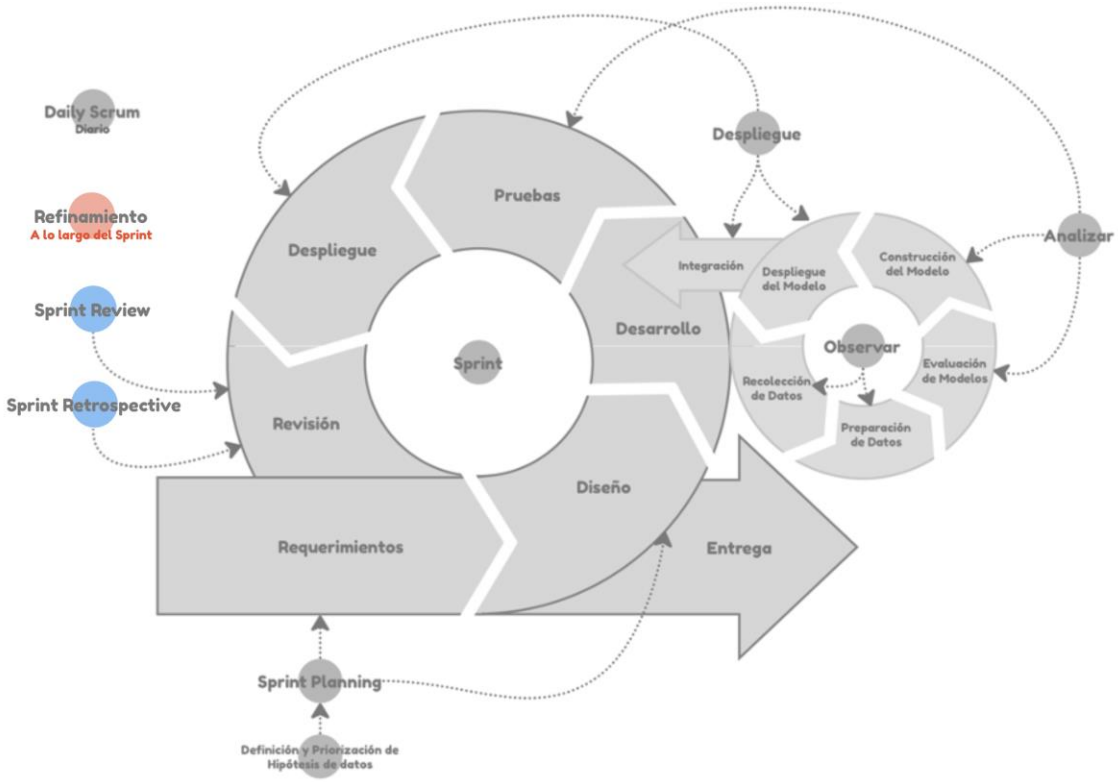

Figura 27. Enfoque de mejora sobre el Ciclo de Vida Caso de Estudio A

En cuanto al componente de *Bienestar del Equipo de Trabajo* una Segunda Retrospectiva reveló que el equipo manifiesta una desalineación de la comunicación con los Stakeholders. Uno de los factores por lo que existe esa percepción es que en los eventos de *Sprint Review*, al momento de inspeccionar los incrementos, las conversaciones giran en torno a los detalles técnicos, lo que podría confundir a los Stakeholders al ser su enfoque más del lado de producto. La forma en que se comunican los avances y la reacción de los Stakeholders ante estos logros, al mejorarse generarían un impacto positivo en la moral del equipo y permitirían aumentar su puntaje en este componente.

5.3 Caso de estudio B

En este caso los retos para el equipo identificados en la Primera Retrospectiva se centraron en la claridad de los requerimientos, puntualmente en la mejora de los criterios de aceptación de las historias de usuario. La necesidad de eventos de refinamiento más efectivos en los que se desglosen de manera más detallada estos requerimientos en una necesidad expresada por el equipo para lograrlo.

Del mismo modo la identificación de dependencias entre las Historias de Usuario permitiría al equipo realizar una mejor estimación de los esfuerzos y planear con mayor precisión los posibles incrementos al final de la iteración.

En este equipo se reforzó la implementación de las prácticas de *Refinamiento*, y *Definición y Priorización de Hipótesis de Datos*, además de aprovechar los eventos de *Sprint Planning*, *Daily Scrum* y *Sprint Review* para incorporar las acciones de mejora compartidas por el mismo equipo. De esta forma queda de manifiesto la relación estrecha que existe entre la transparencia de los requerimientos y como esto impacta de manera positiva en los componentes *Satisfacción del Cliente* y *Calidad de la Solución* al trabajarse en estas preocupaciones el equipo obtuvo un mejor puntaje en la segunda iteración de la encuesta.

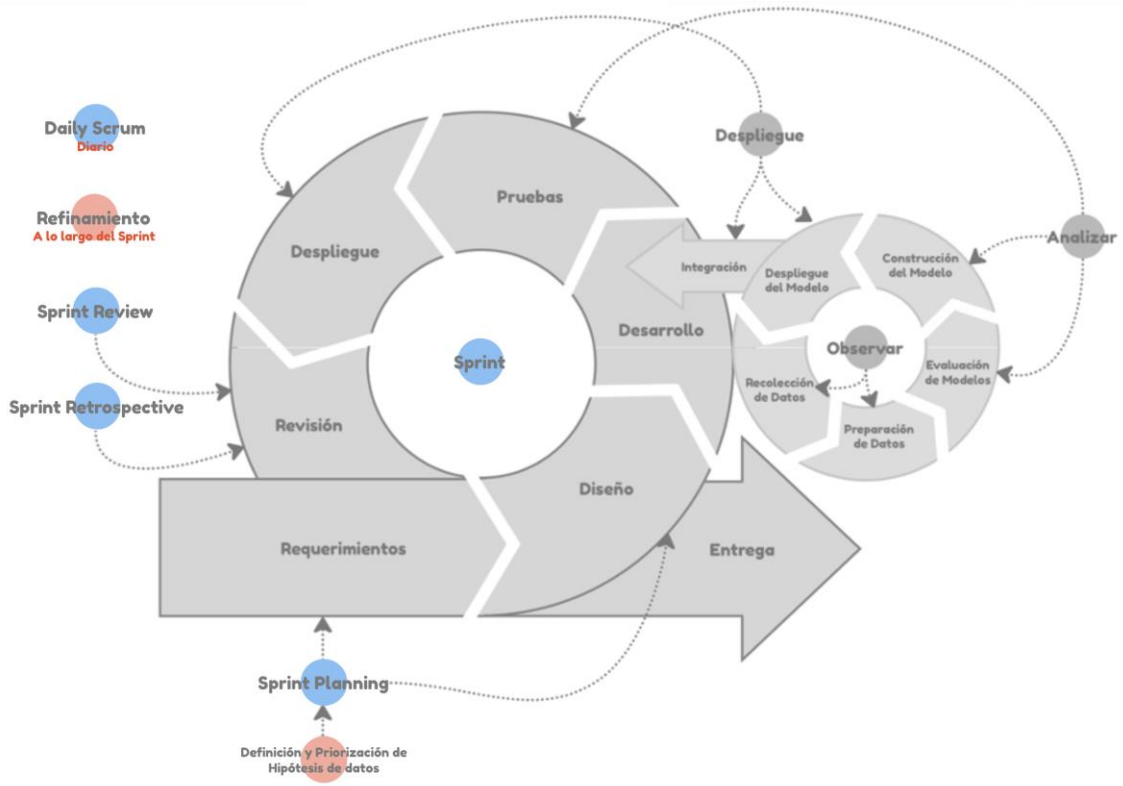

Figura 28. Enfoque de mejora sobre el Ciclo de Vida Caso de Estudio B

Con respecto al componente de *Bienestar del Equipo de Trabajo*, durante la Segunda Retrospectiva, el equipo manifiesta que los retos se enfocan en la capacidad del equipo para poder completar las metas definidas al principio de su iteración.

Superar las dificultades que suponen realizar despliegues de manera más frecuente, y de esta forma entregar avances significativos a los Stakeholders durante el Sprint mejoraría el bienestar del equipo al generar un ambiente de éxito al final de cada iteración y en general de progreso hacia los objetivos.

Capítulo 6: Conclusiones y Recomendaciones

Como ya se ha explicado en esta investigación, el Data Science es una disciplina que emplea métodos estadísticos y de aprendizaje automático para convertir los datos en conocimiento práctico. Estas tecnologías están actualmente en un proceso de transición de su etapa de innovación a convertirse en herramientas a emplear con fines comerciales cada vez más accesibles para los mercados regionales.

A diferencia de las disciplinas tradicionales de desarrollo de software, el Data Science adopta el patrón de pensamiento centrado en datos, reconoce que la propiedad de los datos es más activa que pasiva, convierte estos datos en inteligencia, resuelve tareas intensivas de datos, mejora y enriquece las experiencias de los usuarios y abre las posibilidades a nuevos productos.

Bajo esta perspectiva al realizar este estudio en equipos que desarrollan proyectos de Data Science resulta valioso resaltar los aspectos identificados que influyen en el éxito de estos proyectos.

La adopción del agilismo y sus prácticas dentro de la organización permite que los equipos puedan afrontar los retos que los proyectos de Data Science presentan de forma efectiva, teniendo espacios para la experimentación y la reflexión periódica en la que se identifican oportunidades de mejora y se adapta la forma de trabajo. La flexibilidad que permiten estos marcos de trabajo habilita la creatividad en la resolución de problemas complejos.

Por otro lado, el éxito comercial de este tipo de proyectos va de la mano con mantener un enfoque de producto donde las tecnologías del Data Science son una herramienta y no el fin en sí mismo. Más allá de las capacidades técnicas de los modelos generados, estos deben ser capaces de responder a las necesidades de los usuarios que utilizarán el producto, por lo que la visión de producto debe estar presente desde el inicio y a lo largo de todo el ciclo de vida.

Esto también implica saber comunicar de manera precisa en términos de negocio los progresos en cada incremento del producto. Una explicación técnica profunda de las aproximaciones estadísticas o matemáticas de los modelos empleados podría ser adecuada en un contexto diferente, una conversación enfocada al valor generado a los usuarios y al negocio a partir de los datos enriquece y potencia el interés e involucramiento de los Stakeholders lo que nos lleva al siguiente punto.

El éxito en general de los proyectos (y los proyectos de Data Science no son la excepción) pasa necesariamente por el involucramiento de los Stakeholders en todo el ciclo de vida. La interacción de los equipos de trabajo con los Stakeholders debe ser una relación en donde la retroalimentación fluya en ambos sentidos de manera constante, sin esto, resulta muy difícil que pueda entregarse valor a los usuarios y al negocio al final de cada iteración.

Para concluir, los resultados de los casos de estudio muestran que un ciclo de vida agile impacta de manera positiva en la Satisfacción del Cliente y en la Calidad de las Soluciones y

abre la posibilidad a tener conversaciones sobre cómo mejorar en otros aspectos como el Bienestar del Equipo de Trabajo. El Scrum Team Survey como herramienta de los equipos integrada dentro del ciclo de vida permite cuantificar estos componentes. Para la investigación se realizaron dos iteraciones y los resultados fueron positivos, más iteraciones de este ejercicio significan más oportunidades para mejorar por lo que es recomendable mantener estas revisiones de manera continua para así avanzar en los retos que aún quedan pendientes.

El desarrollo de software es complejo, la incorporación de nuevas tecnologías aumenta aún más esta complejidad y nos empujan a buscar nuevos rumbos, la agilidad no nos muestra un camino como tal, pero es la brújula que nos permite crear nuestro propio camino en la incertidumbre.

Referencias

Agile Alliance. (2001). *The Agile Manifesto.* Recuperado en Enero de 2023, del sitio web oficial https://agilemanifesto.org/principles.html

Bray, M., Adamson, B., & Mason, M. (2007). *Comparative Education Research.* Hong Kong: The University of Hong Kong - Springer.

Corporation, IBM. (Marzo de 2021). *Guía de CRISP-DM de IBM SPSS Modeler*. (IBM) Recuperado en Enero de 2023, de https://www.ibm.com/docs/es/SS3RA7_18.4.0/pdf/ModelerCRISPDM.pdf

Data Science Process Alliance. (2022). *A Jumpstart to Data Science Project Management*. (Data Science Process Alliance) Recuperado el Enero de 2023, de https://www.datascience-pm.com/wp-content/uploads/2022/11/Jumpstart2022.pdf

Freeman, E. (2008). Dialogue: Toward Superior Stakeholder Theory. *Business Ethics Quarterly*.

Giese, J., & Cote, J. (2000). Defining Consumer Satisfaction. *Academy of Marketing Science Review*.

Grus, J. (2015). *Data Science from Scratch.* Gravenstein Highway North, Sebastopol, CA, United States: O'Reilly Media.

Kobielus, J. (Mayo de 2017). *Agile Development in Team Data Science*. (Wikibon) Recuperado en Enero de 2023, de https://wikibon.com/agile-development-in-team-data-science/

Kurtz, C. F., & D, S. J. (2003). The new dynamics of strategy: Sense-making in a complex and complicated world. *IBM SYSTEMS JOURNAL, 42*. Obtenido de https://alumni.media.mit.edu/~brooks/storybiz/kurtz.pdf

Li, L. (s.f.). *Data visualization*. Recuperado en Marzo de 2023, de https://medium.com/@Lynia_Li/as-you-know-there-are-many-types-of-charts-to-be-used-in-data-visualization-54da9b97092e

Nabati, E. G., & Thoben, K.-D. (2016). On Applicability of Big Data Analytics in the Closed-Loop Product Lifecycle: Integration of CRISP-DM Standard. *13th IFIP International Confer- ence on Product Lifecycle Management (PLM)*.

Pinto, N. S. (Julio de 2020). *Framework para la evaluación de calidad de proyectos ágiles de software.* Recuperado en Enero de 2023, del sitio web http://sedici.unlp.edu.ar/bitstream/handle/10915/109769/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y

Saltz, J., Shamshurin, I., & Crowston, K. (2017). Comparing Data Science Project Management Methodologies via a Controlled Experiment. *50th Hawaii International Conference on System Sciences*.

Scrum org. (2020). *Scrum Guides*. (Scrum org) Recuperado en Diciembre de 2022, de https://scrumguides.org/

Scrum.org. (Enero de 2021). *La Guía Kanban para Scrum Teams.* Recuperado en Mayo de 2023, de https://scrumorg-website-prod.s3.amazonaws.com/drupal/2021-03/2021-Kanban-Guide-Spanish-European.pdf?nexus-file=https%3A%2F%2Fscrumorg-website-prod.s3.amazonaws.com%2Fdrupal%2F2021-03%2F2021-Kanban-Guide-Spanish-European.pdf

The Liberators. (Enero de 2023). *Model for Scrum Team Effectiveness*. (The Liberators) Recuperado en Enero de 2023, de https://scrumteamsurvey.org/research

Verwijs, C., & Russo, D. (2022). A Theory of Scrum Team Efectiveness. *ACM Trans. Softw. Eng. Methodol.*

Vorhies, W. (Julio de 2016). *Data Science Central.* Recuperado en Enero de 2023, de https://www.datasciencecentral.com/crisp-dm-a-standard-methodology-to-ensure-a-good-outcome/

Wiley, J. (2015). *Data Science & Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data.* Indianapolis,: John Wiley & Sons, Inc.

Anexos

Encuesta de Equipos Scrum

Gracias por participar en esta encuesta, todas tus repuestas son anónimas y servirán para la identificación de oportunidades de mejora en el equipo. La encuesta se divide en 7 secciones que cuyos resultados proveen información sobre los aspectosmás importantes en la efectividad de los equipos Scrum

* Requerido

Mejora continua

Sobre este equipo, cada vez que las personas expresan su punto de vista, también preguntan qué piensan los demás.

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Sobre este equipo, las personas ven el aprendizaje como parte de su trabajo. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Durante las Retrospectivas del Sprint, este equipo habla abiertamente sobre mejoras. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Sobre este equipo, las personas son recompensadas por aprender. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Sobre este equipo, las personas escuchan las opiniones de los demás antes de hablar. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Las Retrospectivas de Sprint de este equipo generalmente dan como resultado al menos una mejora útil.

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Los equipos de esta organización comparten lo que aprenden con otros equipos. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Las decisiones sobre lo que hace este equipo a menudo están influenciadas por métricas. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Este equipo a menudo inspecciona métricas para identificar mejoras en los procesos. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Este equipo siempre está buscando formas de mejorar la calidad. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Las diferentes personalidades de este equipo a menudo chocan o no están de acuerdo entre sí.

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Los miembros de este equipo a menudo experimentan momentos de fricción entre ellos. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Los miembros de este equipo tienen un entendimiento compartido de lo que significa calidad para ellos.

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

La gente de este equipo habla con frecuencia sobre la calidad y cómo mejorarla. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Sobre este equipo, las personas se dan retroalimentación abierta y honesta entre sí. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

A menudo hay momentos de tensión entre los miembros de este equipo. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Los miembros de este equipo se reúnen con frecuencia con otros equipos para identificar mejoras.

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Sobre este equipo, se alienta a las personas a aprender nuevas habilidades, técnicas o prácticas.

Sobre este equipo, las personas tienen tiempo para apoyar el aprendizaje. *

Este equipo trabaja frecuentemente con otros grupos o equipos para resolver problemas compartidos.

Este equipo usa Retrospectivas al final de los Sprints para explorar soluciones para desafíos persistentes.

Nivel de Respuesta

Acá puedes encontrar la aclaración de algunos conceptos usados en esta sección.

El **Sprint Backlog** es la lista de ítems que el equipo seleccionó para el Sprint actual. Muchos equiposutilizan un "Scrum Board" o "Kanban Board" para visualizar este trabajo.

Una **parte interesada** es cualquier persona fuera de su equipo que tenga una participación sustancial en el resultado de su trabajo como equipo. Ejemplos de partes interesadas son usuarios, clientes, inversores y personas dentro de su organización que dependen de su trabajo.

Un **Incremento** es el resultado de un Sprint. Es una nueva versión del producto que incluye el trabajo de este Sprint. Los equipos deberían ser capaces de entregar al menos un incremento en cada Sprint.

Una **Liberación** es cuando se entrega una versión actualizada, mejorada o nueva del producto a laspartes interesadas (como usuarios y clientes). Es también llamado "release a la producción" o "incremento".

La mayoría de los Sprints de este equipo resultan en un incremento que se puede entregar a partes interesadas.

Para este equipo, la mayoría de los Sprints resultan en un incremento que puede ser entregado a los usuarios.

El Backlog del Sprint de este equipo suele contener muchos elementos pequeños. *

El proceso que utiliza este equipo para implementar software en producción está mayormente automatizado.

La mayoría de los Sprints de este equipo dan como resultado software que se puede implementar en producción.

Durante el Sprint, este equipo dedica tiempo a aclarar el trabajo para los próximos Sprints. *

Durante el Sprint, este equipo dedica tiempo a desglosar el trabajo para los próximos Sprints

Liberaciones a producción generalmente se puede realizar sin pasos manuales. *

Autonomía del equipo

Este equipo es capaz de elegir la forma de realizar su trabajo. *

Este equipo tiene control sobre la planificación del trabajo en equipo. *

Todos en este equipo tienen más que suficiente entrenamiento y experiencia para el tipo de trabajo que tienen que hacer.

Este equipo es libre de elegir los métodos a utilizar para realizar el trabajo. *

La mayoría de las personas en este equipo tienen la capacidad de resolver los problemas que surgen en su trabajo.

Preocupaciones de las partes interesadas

Acá puedes encontrar la aclaración de algunos conceptos usados en esta sección.

Product Owner Esta es la persona de tu equipo que decide lo que es importante para que tu equipotrabaje y en qué orden. Por lo general, interactúan con las partes interesadas con frecuencia.

Sprint Review es el momento recurrente en el que su equipo revisa lo que se ha hecho en un Sprint.Algunos equipos llaman a esto la "Demo" o "Sprint Demo".

Una **Parte Interesada** es cualquier persona fuera de su equipo que tenga una participación sustancial en el resultado de su trabajo como equipo. Ejemplos de partes interesadas son usuarios, clientes, inversores y personas dentro de su organización que dependen de su trabajo.

El **Objetivo de Sprint** es un objetivo único que el equipo tiene la intención de lograr en un Sprint. CadaSprint tiene un objetivo único que debería ser capaz de explicar la mayor parte del trabajo que sucedeen ese Sprint.

Product Backlog es la lista ordenada de ítems que deben realizarse en el producto que está construyendo. Uno o más equipos pueden trabajar desde un solo Product Backlog. Algunos equiposllaman a esto simplemente "Backlog"

Mayoría Sprint Reviews producen cambios útiles en el Backlog de Producto de este equipo *

El Product Owner de este equipo tiene una visión clara del producto. *

Todos en este equipo están familiarizados con la visión del producto. *

Durante la Sprint Planning, este equipo formula un objetivo claro para el Sprint. *

Este equipo en general tiene Objetivos de Sprint claros *

Las personas de este equipo a menudo invitan o visitan a personas que **usan** el producto queel mismo equipo desarrolla.

Los miembros de este equipo se reúnen con frecuencia con los usuarios o clientes del producto que el mismo equipo desarrolla.

Antes de implementar una funcionalidad completa, este equipo a menudo prueba primero versiones más simples con los usuarios.

Con frecuencia durante las Sprint Reviews, las partes interesadas prueban lo qué este equipo ha estado trabajando durante el Sprint.

Este equipo realiza con frecuencia experimentos o talleres para descubrir cómo los usuarios

quieren usar el producto.

Las personas de este equipo colaboran estrechamente con los usuarios, clientes y otras partes interesadas.

El Product Owner de este equipo utiliza el Sprint Review para recopilar comentarios de las partes interesadas.

El Backlog del Producto de este equipo se ordena con una estrategia en mente. *

Apoyo de la gerencia

Las personas en posiciones gerenciales animan a este equipo a mejorar sus procesos, tecnologías y métodos de trabajo.

Las personas en una posición gerencial alientan a este equipo a tomar sus propias decisiones en lugar de que se les diga qué hacer.

Las personas en una posición gerencial ayudan a este equipo a trabajar con Scrum. *

Las personas en una posición gerencial crean un ambiente para que este equipo aprenda, experimente y mejore.

Las personas en una posición gerencial eliminan los obstáculos que hacen que sea más difícil para este equipo realizar liberaciones más frecuentemente.

Las personas en una posición gerencial apoyan activamente a este equipo para trabajar más de cerca con las partes interesadas.

Las personas en un puesto de dirección ayudan a este equipo a entender por qué nuestro trabajo es importante.

Las personas en una posición gerencial hacen lo que pueden para ayudar a este equipo a realizar liberaciones más frecuentemente.

Las personas en una posición gerencial generalmente entienden por qué este equipo trabaja con Scrum.

Las personas en una posición gerencial apoyan activamente a este equipo para administrar cómo hacemos nuestro trabajo.

Eficacia del equipo

Las partes interesadas nos felicitan por el valor que les entregamos. *

Las partes interesadas generalmente están contentas con la rapidez con que este equipo responde a sus necesidades.

Estoy entusiasmado con el trabajo que hago para este equipo. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Encuentro el trabajo que hago para este equipo lleno de significado y propósito. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Estoy orgulloso del trabajo que hago para este equipo. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

En general, las partes interesadas están satisfechas con el software que ofrece este equipo.

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Considero que este equipo tiene mucha experiencia con Scrum. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Sobre ti

No importa con quién esté hablando, siempre escucho atentamente. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

¿Con qué frecuencia trabajas desde casa? *

- ☐ Uno o más días a la semana
- ☐ Uno o más días al mes
- ☐ Rara vez o nunca

¿Qué papel describe mejor lo que haces en este equipo? *

- [ ] Scrum Master
- [ ] Product Owner
- [ ] Desarrollador
- [ ] Diseñador UI/UX
- [ ] Tester / QA
- [ ] Márketing / Ventas
- [ ] Analista
- [ ] Ingeniero de infraestructura
- [ ] Other: _____

¿Has recibido una certificación en Scrum o Agile de una organización como Scrum.org, Scrum Alliance, ProKanban, ¿PMI o Agile Alliance?

- [ ] Sí
- [ ] No o no me acuerdo

Siempre soy cortés incluso con personas "desagradables" para mí. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

¿Has recibido capacitación sobre cómo trabajar con Scrum o Agile? *

- [ ] Sí, un entrenamiento de un día o menos.
- [ ] Sí, un entrenamiento de dos o más días.
- [ ] No o no me acuerdo

Encuesta de Equipos Scrum - Stakeholders

Gracias por participar en esta encuesta, todas sus repuestas son anónimas y servirán para identificaroportunidades de mejora en el equipo.

* Requerido

Su experiencia con este equipo

Estoy satisfecho con el valor que ofrece este equipo. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Cuando tengo una idea o sugerencia, los miembros del equipo están disponibles para escucharme.

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Tengo una buena idea de en qué está trabajando este equipo. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

El equipo me pide con frecuencia mi opinión, ideas o pensamientos. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Estoy satisfecho con la calidad de lo que ofrece este equipo. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Cuando el equipo entrega una nueva versión, generalmente no tiene errores graves. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Estoy satisfecho con la frecuencia con la que se publican nuevas versiones. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Estoy contento con el valor que este equipo ofrece en cada Sprint. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Lo que ofrece este equipo es de alta calidad. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Este equipo entrega con frecuencia nuevas versiones. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

La frecuencia de nuevos entregables es lo suficientemente buena para mis necesidades. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Frecuentemente me reúno o interactúo con miembros de este equipo. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Acerca de Usted

¿Qué describe mejor su interés en lo que este equipo entrega? *

- [ ] Yo quiero usarlo en la vida diaria
- [ ] Yo pago por su desarrollo
- [ ] Yo quiero usarlo en la vida diaria y pago por su desarrollo
- [ ] Me interesa, pero no lo uso ni pago por su desarrollo

Encuesta de Equipos Scrum - Support

Gracias por participar en esta encuesta, todas sus repuestas son anónimas y servirán para identificar oportunidades de mejora en el equipo.

* Requerido

Sobre tu apoyo

Acá puedes encontrar la aclaración de algunos conceptos usados en esta sección.

Una **parte interesada** es cualquier persona fuera de su equipo que tenga una participación sustancial en el resultado de su trabajo como equipo. Ejemplos de partes interesadas son usuarios, clientes, inversores y personas dentro de su organización que dependen de su trabajo.

Una **Liberación** es cuando se entrega una versión actualizada, mejorada o nueva del producto a laspartes interesadas (como usuarios y clientes). Es también llamado "release a la producción" o "incremento".

Para mí es importante crear un sentido de comunidad entre los empleados. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Dedico tiempo a formar relaciones de calidad con este equipo. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Elimino los obstáculos que dificultan que este equipo realice liberaciones más frecuentemente. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Ánimo a este equipo a mejorar sus procesos, tecnologías y métodos de trabajo. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Apoyo activamente a este equipo para administrar cómo hacen su trabajo. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Animo a este equipo a tomar sus propias decisiones en lugar de que les digan qué hacer. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Ayudo a crear un ambiente para este equipo donde pueden aprender, experimentar y mejorar. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Me aseguro de que este equipo sepa por qué su trabajo es importante. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Apoyo activamente a este equipo para trabajar más de cerca con las partes interesadas. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Frecuentemente interactúo con este equipo para saber dónde necesitan mi apoyo. *

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Hago lo que puedo para ayudar a este equipo a realizar liberaraciones más frecuentemente.

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Acerca de Agilismo

En comparación con las metodologías basadas en tradicionales, las metodologías ágiles permiten que los equipos reduzcan el riesgo de construir el producto equivocado.

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Las metodologías ágiles generalmente permiten que los equipos se adapten más rápidamente a los cambios en comparación con las metodologías tradicionales.

	1	2	3	4	5	6	7	
Completamente en desacuerdo	○	○	○	○	○	○	○	Completamente de acuerdo

Las metodologías ágiles generalmente permiten que los equipos entreguen más valor a las partes interesadas en comparación con las metodologías tradicionales.

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

Acerca de ti

No importa con quién esté hablando, siempre escucho atentamente. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

¿A cuántos equipos estás apoyando? *

- [ ] 1 equipo
- [ ] 2-5 equipos
- [ ] 6-15 equipos
- [ ] Más de 16 equipos

Siempre soy cortés incluso con personas “desagradables” para mí. *

1 2 3 4 5 6 7

Completamente en desacuerdo ○ ○ ○ ○ ○ ○ ○ Completamente de acuerdo

¿Qué papel describe mejor su papel de apoyo hacia este equipo? *

- [ ] Gerencia intermedia
- [ ] Alta dirección
- [ ] Entrenador o consultor (externo)
- [ ] Other: _____