UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE CIENCIAS ECONÓMICAS

ESCUELA DE GRADUADOS EN CIENCIAS ECONÓMICAS

MAESTRÍA EN DIRECCIÓN DE NEGOCIOS

TRABAJO FINAL DE APLICACIÓN

Optimización de Proyectos en el Lifecycle de Procesos Analíticos para una Multinacional de Software

Autor:
Ing. Lic. Gonzalo Ulla

Tutor:
MBA Walter Abrigo

Córdoba
2020

Agradecimientos

A mi familia, a mis compañeros de estudio, de trabajo y de vida por apoyarme incondicionalmente en esta etapa y en este contexto atípico.

A mi tutor, a la Escuela de Graduados, a la Facultad de Ciencias Económicas y a la Universidad Nacional de Córdoba por ser una fuente infinita de oportunidades de crecimiento, de mejora y de aprendizaje continuo.

Resumen

El presente trabajo consta de una intervención profesional para optimizar proyectos en el ciclo de vida de procesos analíticos de una compañía multinacional de software, específicamente en el área de Servicios Profesionales de su filial argentina ubicada en la Ciudad Autónoma de Buenos Aires. Para ello, se comenzó caracterizando a la organización bajo estudio así como a las motivaciones existentes en el contexto actual de los sistemas de aprendizaje automático.

Se derivó el problema de estudio junto a sus principales causas: por un lado, aquéllas técnicas y, por otro, aquéllas de gestión. Para solventarlas, se plantearon dos líneas de acción, una basada en prácticas de ingeniería y otra, en herramientas de gestión ágil de proyectos. Así, se dio respuesta a la formulación del problema: ¿cuál será el impacto de optimizar proyectos en el lifecycle de procesos analíticos sobre la rentabilidad de la compañía multinacional?

Se desarrolló el marco conceptual de la Ingeniería de Software Lean, de las metodologías ágiles y de los sistemas de aprendizaje automático. Se aplicaron prácticas Lean como Gestión de la Configuración de Software, Integración y Entrega Continuas, así como el marco de trabajo Scrum para la gestión ágil de proyectos, en cuatro experimentos referidos a sistemas de aprendizaje automático. Se analizó el impacto en horas requeridas para cada uno de los experimentos y se sintetizaron los resultados obtenidos en escenarios pesimista, base y optimista con reducciones de 5, 9 y 12% en costos respectivamente, determinando así su impacto en la rentabilidad de la compañía multinacional.

Índice de contenidos

Índice de tablas

Índice de figuras

1. Introducción

1.1. Contexto del estudio

El presente trabajo final de aplicación representa el proyecto cúlmine de la Maestría en Dirección de Negocios, correspondiente a la Escuela de Graduados de la Facultad de Ciencias Económicas, Universidad Nacional de Córdoba. El mismo consiste en determinar los beneficios que se generarían al aplicar prácticas de ingeniería de software y de gestión de proyectos a lo largo del ciclo de vida[1] de procesos analíticos en una compañía multinacional de software.

La organización bajo análisis se caracteriza por ser una de las empresas de software privadas más grandes del mundo, con más de cuarenta años de presencia en la industria analítica. Actualmente, emplea a más de trece mil colaboradores en todo el mundo, con presencia en 147 países. Este trabajo pretende optimizar los proyectos y procesos de desarrollo u operaciones de esta empresa, generando, de cierta forma, un *efecto aprendizaje* (O'Sullivan & Sheffrin, 2003) al fomentar una retroalimentación positiva entre los distintos proyectos que integran estos procesos.

En su gran mayoría, los proyectos analíticos, es decir, aquellos destinados a generar información a partir del análisis sistemático de datos (Kohavi, et. al., 2002), tienen como objetivo realizar algún tipo de Transformación Digital. Este último fenómeno, cada vez más vigente en la actualidad, da origen a la oportunidad y a las motivaciones subyacentes a este trabajo de aplicación.

1.2. Oportunidad y motivaciones

De acuerdo con Gartner (Forni & Van der Meulen, 2017), hace prácticamente tres años, un 42% de los CEOs (*Chief Executive Officers* o Directores Ejecutivos) ya habían

[1] De ahora en más, *lifecycle*.

comenzado con la Transformación Digital de sus negocios. Esta misma encuesta coloca a los cambios referidos a IT (Tecnologías de Información, por sus siglas en inglés) como la segunda prioridad de dichos ejecutivos, algo inédito hasta aquel momento.

Dos años después, otro estudio de la misma consultora citado por RT Insights (Curry, 2019) develó que el 77% de las organizaciones esperaba aplicar Inteligencia Artificial o AI dentro de los próximos 4 años, a partir de 2019. Sin embargo, no es necesario esperar hasta 2023 para vislumbrar el impacto de estos cambios. Encuestas más recientes de Gartner (Costello, 2020) establecen que un 40% de las organizaciones planean poner en funcionamiento soluciones de AI para fines de 2020.

A pesar de estas claras intenciones de emprender el camino de la Transformación Digital, la realidad es que, si bien lanzar pruebas piloto es excesivamente fácil, poner estas tecnologías innovadoras en producción es sumamente desafiante. Esto significa, para el caso puntual de Inteligencia Artificial por ejemplo, que soluciones aparentemente efectivas durante su fase de construcción, tienen problemas para, o incluso nunca logran, generar resultados organizacionales tangibles y cuantificables.

Tomando como base los resultados del estudio de Costello (2020), aproximadamente sólo la mitad de las compañías tiene métricas o metodologías definidas que les permitan identificar casos de éxito a la hora de transformar digitalmente sus negocios. Claramente, esta situación plantea un problema y pone en riesgo la viabilidad de muchos proyectos tecnológicos, entre ellos, sin dudas, los de naturaleza analítica.

1.3. Identificación del problema

El hecho, tal como se mencionó anteriormente, de que tan sólo un 50% de las organizaciones que desarrollan soluciones basadas en AI tengan herramientas para determinar si las mismas, una vez construidas, son o no exitosas, representa apenas un ejemplo del

Estado del Arte al cual se enfrentan diversas empresas al emprender sus procesos de Transformación Digital (entre ellas, la compañía objeto de estudio de este trabajo). Con foco puesto en la industria, es posible deducir entonces que, de los 24 *billions*[2] de dólares que se destinaron mundialmente en 2019 para financiar start-ups de Inteligencia Artificial (Venture Scanner, 2020), 12 billions fueron dados, prácticamente, a ciegas.

Al intentar diagnosticar las causas que explican esta situación problemática, resulta posible agruparlas en dos grandes categorías: aquéllas de origen técnico y aquéllas de gestión.

En primer lugar, en cuanto a las dificultades técnicas que a menudo se presentan en los proyectos analíticos, cabe resaltar la compleja aplicación de prácticas de ingeniería de software en ellos: su propia naturaleza a menudo propicia la generación de deuda técnica y complica su mantenimiento a lo largo del tiempo (Sculley, et. al., 2015).

En segundo lugar, existen, a su vez, dificultades de gestión en este tipo de proyectos. Su intrincada composición y lo "enredado" de sus componentes, hace que un sistema de AI sea mucho más difícil de separar en módulos en relación a lo que ocurriría en el desarrollo tradicional de software (Amershi, et. al., 2019). Esto complejiza profundamente las tareas de planificación, organización, dirección y control al implementar software de base analítica.

De continuar sin ningún tipo de alteraciones la situación actual, los proyectos de Transformación Digital que ejecuta la multinacional bajo estudio, continuarán perteneciendo a ese 50% que no posee una metodología determinada. Por ende, los procesos operativos de la compañía analizada continuarán teniendo un desempeño subóptimo, cometiendo una erogación de recursos por encima de lo deseable y tomando más tiempo del esperado.

Para revertir este escenario, se proponen dos claras vías de acción: una arraigada en conocimientos técnicos e ingenieriles y otra fundada en prácticas de gestión de proyectos. De

[2] mil millones, en su concepción anglosajona

esta manera, se busca atacar las dos categorías de causas mencionadas y, así, evitar que los proyectos analíticos de la empresa objetivo incurran en costos excesivos a lo largo del ciclo de vida de sus procesos.

Finalmente, es posible formular el problema central de este trabajo de la siguiente forma: ¿cuál será el impacto de optimizar proyectos en el lifecycle de procesos analíticos sobre la rentabilidad de la compañía multinacional?

1.4. Objetivos y alcance del trabajo

El objetivo general del trabajo en cuestión es determinar el impacto sobre la rentabilidad de la compañía multinacional al optimizar proyectos en el lifecycle de sus procesos analíticos.

Para alcanzar este objetivo general, se deberán lograr los siguientes objetivos específicos:

- Describir el contexto, las oportunidades y las motivaciones sobre las cuales se desarrolla este trabajo.
- Desarrollar el marco conceptual de base referido a Ingeniería de Software *Lean* y el estado del arte de sus prácticas.
- Establecer la fundamentación teórica subyacente a las metodologías ágiles de gestión de proyectos y su complemento parcial con otras metodologías tradicionales.
- Analizar las características y las problemáticas de los sistemas de aprendizaje automático y el ciclo de vida de procesos analíticos.
- Desarrollar la aplicación de prácticas de Ingeniería de Software Lean y de gestión de proyectos a aquéllos de la compañía objetivo analizando su efecto en tiempo y costos.

El alcance de este trabajo se delimitó a la Gerencia de Servicios Profesionales de la filial argentina de la compañía multinacional, localizada en la Ciudad Autónoma de Buenos

Aires. A su vez, los respectivos análisis y desarrollos sobre ella fueron realizados comprendiendo un período de observación de seis meses, abarcando desde marzo hasta agosto de 2020. Excede a los límites de este trabajo la implementación de los productos de este último.

Para responder al objetivo general de este trabajo, se hará un análisis ceteris paribus del tiempo necesario para ejecutar los proyectos analiticos de la compañía bajo estudio. Esto significa que toda modificación en la cantidad de horas requeridas para finalizar estos proyectos impactará la rentabilidad ya que todas las demás variables (ingresos y demás factores de costos) se suponen constantes.

1.5. Antecedentes del estudio

El trabajo en cuestión no presenta antecedentes en el pasado corto plazo de la filial argentina de la compañía multinacional. Fuera del territorio nacional, sin embargo, sí es posible encontrar casos de aplicación de metodologías tradicionales de gestión de proyectos como aquellas basadas en los lineamientos del Project Management Institute.

Por otra parte, también a nivel internacional, dentro de esta corporación existe material teórico original referido al proceso de operacionalización de proyectos analíticos en distintas organizaciones. A pesar de esto, dada la reciente producción de este contenido, aún no se encuentran debidamente documentadas experiencias prácticas ni casos de éxito que sirvan de referencia.

Más allá de los límites organizacionales de la empresa bajo estudio, grandes compañías han implementado y publicado casos de estudio y/o las lecciones aprendidas de los mismos. Este es el caso de Microsoft (Amershi, et. al., 2019) y Google (Sculley, et. al., 2015).

En el ámbito académico, resulta posible dar con estudios comparativos o trabajos de campo, tales como el de Lwakatare y otros (2019), que analizan múltiples casos de optimización de procesos en la industria analítica. Estos, junto con el marco conceptual que fundamenta este trabajo, serán presentados en el próximo capítulo.

2. Fundamentación teórica

2.1. Ingeniería de Software Lean

2.1.1. Principios Lean

Es posible definir a la Ingeniería de Software Lean o Desarrollo de Software Lean (LSD de ahora en más, por sus siglas en inglés[3]) como una síntesis de prácticas, principios y filosofía para construir sistemas de software (Charette, 2003). LSD se basa en aplicar un conjunto de principios derivados de *Lean Manufacturing* y del *Toyota Production System* (TPS) con el objetivo de reducir el tiempo de ciclo en el desarrollo de productos de software. (Fowler, 2008).

LSD se entiende como una serie de principios que, fundados en reducción de desperdicios, planificación adaptativa, foco en las personas y diferimiento de las decisiones, ayudan a mejorar la calidad en la ingeniería de software (Poppendieck & Cusumano, 2012). Originalmente, los siete principios de la Ingeniería de Software Lean planteados por Mary y Tom Poppendieck (2003) fueron:

1. Eliminar el desperdicio — *Eliminate waste*
2. Amplificar el aprendizaje — *Amplify learning*
3. Decidir lo más tarde posible — *Decide as late as possible*
4. Entregar lo antes posible — *Deliver as fast as possible*
5. Empoderar al equipo — *Empower the team*

[3] LSD: Lean Software Development

6. Construir con integridad — *Build integrity in*
7. Ver el todo — *See the whole*

El espíritu de Lean no está en el proceso de desarrollo en sí, sino en cómo crear valor para los clientes (Poppendieck & Cusumano, 2012). Por eso mismo, LSD se nutre de la experiencia y de allí que los principios anteriores hayan sido levemente modificados:

1. Optimizar el todo — *Optimize the whole*
2. Eliminar el desperdicio — *Eliminate waste*
3. Construir con calidad — *Build quality in*
4. Aprender constantemente — *Learn constantly*
5. Entregar rápido — *Deliver fast*
6. Involucrar a todos — *Engage everyone*
7. Seguir mejorando — *Keep getting better*

Si se entendiera a LSD sólo como un conjunto de prácticas, hubiera resultado mucho más complicado adaptarlo de entornos operativos a entornos de desarrollo. Al tratarse de principios más abarcativos, éstos permiten encuadrar qué prácticas concretas de desarrollo resultan apropiadas para cada organización en cada situación.

2.1.2. Prácticas Lean

El foco de esta sección es describir ciertas prácticas conocidas de LSD y cómo se vinculan a sus principios. Recordando que el propósito de estos últimos es adaptarse a distintas circunstancias, comprender dichas prácticas servirá a la hora de adaptarlas al contexto de proyectos analíticos y de sistemas de aprendizaje automático.

1. Gestión de la Configuración de Software — *Software Config. Management*

La Gestión de Configuración de Software (SCM de ahora en más, por sus siglas en inglés) se refiere al proceso a través del cual todos los artefactos relevantes de un proyecto, y

las relaciones entre ellos, son almacenados, recuperados, unívocamente identificados y modificados (Humble & Farley, Continuous Delivery, 2011). SCM es la aplicación de Gestión de la Configuración o *Configuration Management* (CM) a proyectos de software, entendiendo por CM a los medios por los cuales el contenido, los cambios o el estado de la información compartida en un proyecto es administrada y controlada (Software Program Managers Network, 1998).

Si bien SCM no es lo mismo que VCS (Sistemas de Control de Version o *Version Control Systems*), una estrategia de SCM debe definir qué VCS usar, cómo gestionar los cambios en los artefactos de un proyecto y cómo colaboran los equipos dentro de uno de éstos. Esta práctica, al basarse en la evolución de artefactos, se vincula directamente a los principios Lean de *Aprender constantemente* y *Seguir mejorando*.

2. Integración Continua — *Continuous Integration*

En el marco de Ingeniería de Software, Integración Continua (CI, por sus siglas en inglés) es la práctica de unir automáticamente los cambios que distintos desarrolladores efectúan sobre el código fuente de un proyecto tan frecuentemente como sea posible. Kent Beck (1999), quien publicó por primera vez este término, se refería a la idea de integrar código sobre un sistema de versión compartido "todo el tiempo".

Esta práctica permite detectar errores de integración con cada nuevo cambio, propiciando que sean arreglados inmediatamente. Esto se vincula con los principios Lean de *Eliminar el desperdicio* y *Construir con calidad*.

3. Entrega Continua — *Continuous Delivery*

Quizás la más conocida de las prácticas de LSD, Entrega Continua (CD de ahora en más, por sus siglas en inglés) reúne una serie de técnicas recopiladas por primera vez por Jez Humble (2011). CD puede definirse como un sistema de técnicas orientadas a poner cambios

en el software, independientemente de su tipo, en producción de forma segura, rápida y sostenible. Para lograrlo, se deben cumplir dos condiciones: 1) el código fuente debe encontrarse siempre en un estado capaz de ser desplegado —*deployable state,* y 2) debe existir una manifestación automatizada del proceso para llevar código hasta los usuarios finales —*deployment pipeline*. En la figura 1, se puede observar un diagrama de secuencia de un ejemplo de un deployment pipeline.

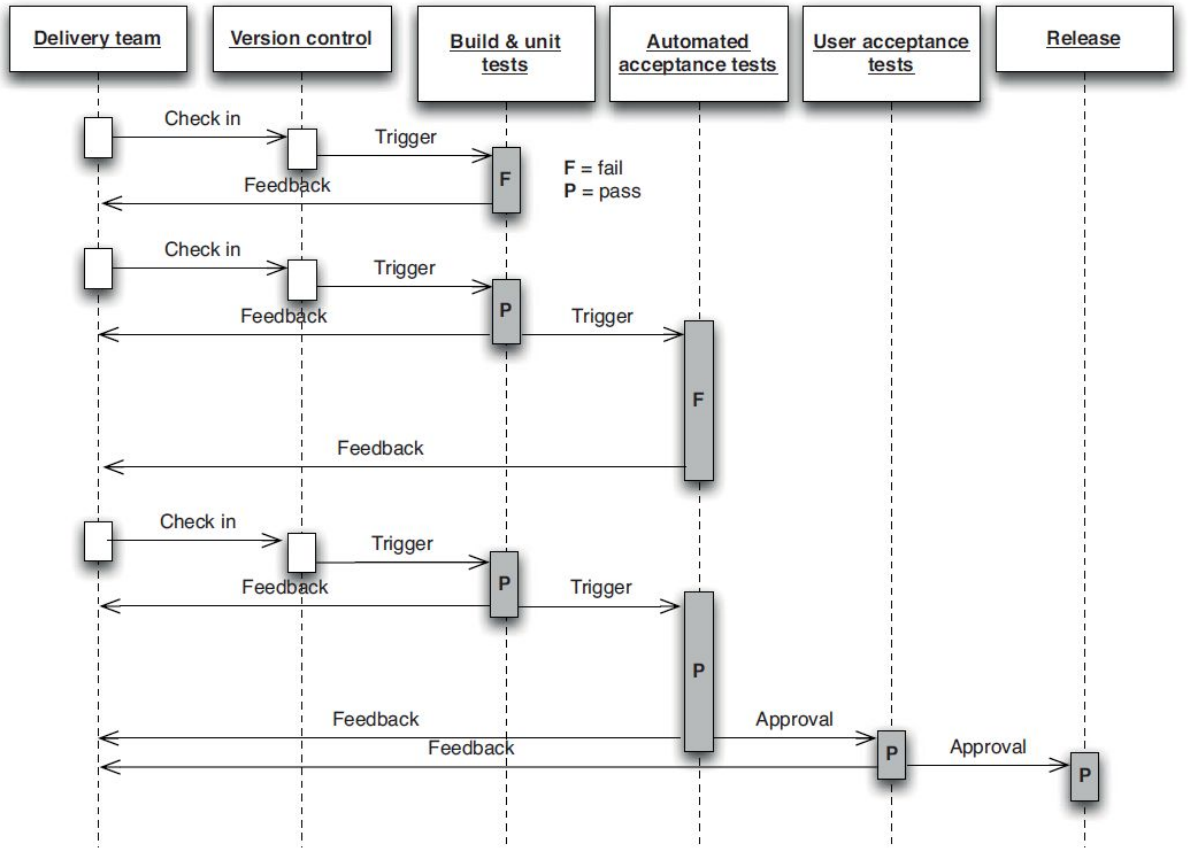

Figura 1. Cambios moviéndose a través del deployment pipeline
Fuente: Humble & Farley, Continuous Delivery, 2011

Esta práctica se encuentra principalmente vinculada con el principio Lean de *Entregar rápido*. Si el último paso del *deployment pipeline*, es decir la puesta a producción de los cambios en el software, también se automatiza, esta práctica se denomina Despliegue Continuo o *Continuous Deployment*. A menudo, diversas organizaciones deciden que este

paso final responda a una decisión de negocio: ya sea por motivos regulatorios o meramente comerciales.

2.1.3. Historia de Lean

El origen de Lean se atribuye generalmente al Sistema de Producción Toyota o *Toyota Production System* (TPS) desarrollado en Japón luego de la década de 1940 (Liker, 2004). Este sistema se desarrolla durante la década de 1950, momento en el que la industria japonesa se estaba recuperando de la Segunda Guerra Mundial. Se considera padre del TPS al ingeniero industrial Taiichi Ohno, y a menudo se lo relaciona con las influencias occidentales de W. Edwards Deming, existiendo incluso referencias de presentaciones de este último a ejecutivos de Toyota en Japón (Bellows, 2019).

A pesar de que el TPS y los principios detrás de éste fueron ideados en Japón, quienes popularizaron el término "Lean" fueron estadounidenses. En el marco de una investigación del Massachusetts Institute of Technology, Womack et. al. acuñaron el término "Lean Manufacturing" (Womack, Jones, & Roos, 1990) para describir las técnicas de producción que les permitieron a las automotrices japonesas imponerse sobre las norteamericanas en numerosas industrias a nivel mundial.

En la industria del software, el libro "Microsoft Secrets" (Cusumano & Selby, 1997) fue la primera obra en relacionar la estrategia de *daily builds* que usaba Microsoft, en la que los ingenieros software tenían que parar y corregir errores diariamente, con la filosofía de producción de Toyota, en la cual los trabajadores paraban las líneas de montaje cuando detectaban problemas. Sin embargo, este libro no empleaba el término "Lean" para describir estas prácticas.

Quienes difundieron la aplicación de "Lean" al software, es decir el *Lean Software Development* (LSD), fueron Mary y Tom Poppendieck (Poppendieck & Poppendieck, 2003).

Además, estos autores lo relacionaron a las metodologías de desarrollo ágil, descritas a continuación.

2.2. Metodologías ágiles

2.2.1. Manifiesto ágil

El desarrollo de software ágil o *Agile Software Development* (ASD) puede entenderse como un término "paraguas" para varios métodos de desarrollo de software que fueron desarrollados en la década de 1990 (Fowler, 2008). Estos métodos de "peso liviano" surgen en respuesta a metodologías mucho más burocráticas, consideradas de "peso pesado" o "procesos definidos", a la hora de desarrollar software.

Haciendo un breve resumen histórico, en un primer momento la industria de software se caracterizaba por la inexistencia de procesos y metodologías (el paradigma *"code and fix"*). Luego, el desarrollo de sistemas viró a estándares fuertemente definidos y metodologías muy estructuradas, como por ejemplo el *Proceso Unificado de Desarrollo* (Jacobson, Booch, & Rumbaugh, 1999). Actualmente, en la industria priman métodos empíricos o de peso liviano como las metodologías ágiles que constituyen el ASD.

La filosofía subyacente al desarrollo de software ágil se guía por los cuatro valores fundamentales establecidos en el manifiesto ágil (Beck, y otros, 2001).

1. Individuos e interacciones sobre procesos y herramientas
2. Software funcionando sobre documentación extensiva
3. Colaboración con el cliente sobre negociación contractual
4. Respuesta ante el cambio sobre seguir un plan

Cabe destacar que el movimiento ágil no es anti-metodologías, no se opone a documentar proyectos o procesos, por ejemplo. Simplemente, busca alcanzar un equilibrio en

la carga metodológica, priorizando las interacciones, las personas, la experiencia y los ciclos cortos de inspección/adaptación durante el desarrollo de software.

2.2.2. Marco de trabajo Scrum

Agile no es Scrum. Scrum es sólo uno de los enfoques ágiles para desarrollar software, es un marco de trabajo para desarrollar, entregar y mantener productos complejos (Schwaber & Sutherland, 2017).

Dicho de otra manera, Scrum es un marco de trabajo para la gestión ágil de proyectos a través del uso de un proceso empírico. Tiene como objetivo guiar equipos en la entrega iterativa e incremental de un producto. Como pilares, Scrum se basa en retroalimentación rápida y decisiones colaborativas que permiten responder rápida, efectiva y eficientemente al cambio (Sliger, 2011).

Scrum define una serie de roles, eventos y artefactos que, en conjunto, componen el marco de trabajo Scrum representado en la figura 2.

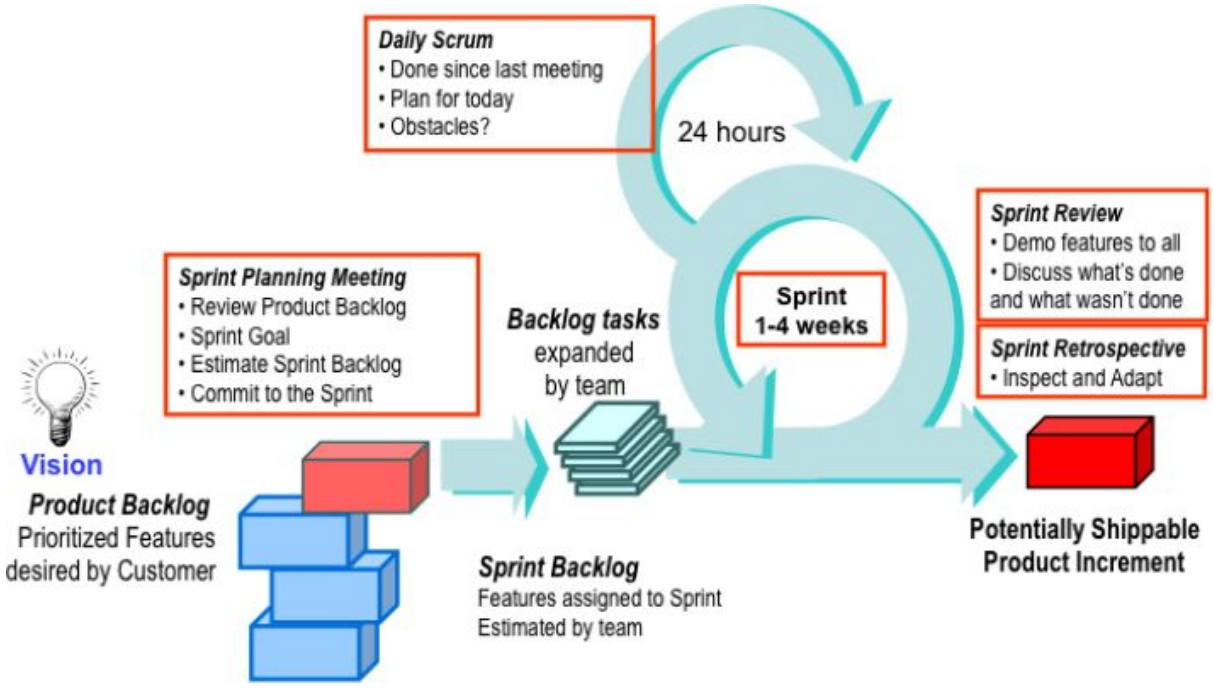

Figura 2. El marco de trabajo Scrum
Fuente: https://www.pmi.org/learning/library/agile-project-management-scrum-6269

1. Roles en Scrum

Cada uno de los equipos Scrum, los cuales se auto-organizan y son polivalentes, consiste en tres roles: Product Owner, Equipo de Desarrollo y Scrum Master. El Product Owner es responsable de maximizar el valor del producto resultante del trabajo del Equipo de Desarrollo. Este último abarca los profesionales que ejecutan las tareas necesarias para entregar un incremento del producto al final de cada Sprint. Finalmente, el Scrum Master actúa como facilitador del equipo Scrum, apoyando las prácticas del marco de trabajo y dando soporte al Product Owner, al Equipo de Desarrollo y a la Organización.

2. Eventos en Scrum

Existen cuatro eventos prescriptos en Scrum, los cuales suceden durante un Sprint. Un Sprint es, en sí mismo, un contenedor de los demás eventos, un período de tiempo definido de entre 1 y 4 semanas en el cual se desarrolla un incremento del producto. Los demás eventos son: *1) Sprint Planning:* consiste en determinar el objetivo del Sprint y seleccionar, del Product Backlog, el trabajo a realizar en el mismo; *2) Daily Scrum:* permite planificar las próximas 24 horas de trabajo en una reunión diaria de 15 minutos; *3) Sprint Review:* realizada al final del Sprint, en esta reunión se revisa el trabajo realizado, el incremento del producto alcanzado y los cambios al Product Backlog; *4) Sprint Retrospective:* el equipo Scrum se inspecciona a sí mismo y plantea mejoras al proceso de trabajo.

3. Artefactos en Scrum

Existen tres artefactos en Scrum: Product Backlog, Sprint Backlog y Burndown Chart. El Product Backlog es una lista ordenada de todos los aspectos necesarios conocidos del producto. Representa la única fuente de requerimientos que, si bien nunca está completa, se encuentra ordenada según el valor de negocio de cada elemento que la compone. El Sprint Backlog es el conjunto de los elementos del Product Backlog seleccionados para el Sprint

correspondiente. Es un pronóstico del Equipo de Desarrollo sobre la funcionalidad a incluir en el próximo incremento del producto. Por último, el Burndown Chart refleja el trabajo acumulado restante en un Sprint con una frecuencia diaria.

2.2.3. Relación entre metodologías ágiles y tradicionales

A la hora de gestionar proyectos, es posible hacerlo empleando tanto metodologías ágiles (por ejemplo, Scrum) como aquéllas tradicionales (por ejemplo, la sugerida por el Project Management Institute). La principal diferencia entre ellas es que, mientras las metodologías ágiles utilizan procesos empíricos para administrar proyectos, los métodos tradicionales se basan en procesos definidos. Esto significa que, para el desarrollo ágil, la experiencia del equipo es lo que distingue a cada proyecto. Por su parte, el desarrollo tradicional pretende estandarizar procesos comunes a múltiples proyectos.

A su vez, las metodologías de administración de proyectos tradicionales fijan los requerimientos en pos de controlar tiempo y costos, mientras que las metodologías ágiles fijan tiempo y costos para controlar los requerimientos (Sliger, 2011).

Si bien ambos tipos de metodologías a menudo se consideran mutuamente excluyentes, en la actualidad existen trabajos que buscan compatibilizar ambos enfoques, ya sea embebiendo los grupos de procesos del PMBOK en metodologías ágiles como XP y Scrum (Usman, Soomro, & Brohi, 2014) o utilizando estos grupos de procesos bajo un enfoque ágil en proyectos administrados con Scrum (Schwalbe, 2012).

2.2.4. Relación entre metodologías ágiles y Lean

Ha existido una conexión entre *Lean Manufacturing* y *Agile Software Development* desde sus orígenes: muchos de los autores de los métodos ágiles fueron influenciados por las ideas de Lean (Fowler, 2008). Como se ha mencionado, los primeros en hacer explícita esta relación fueron Mary y Tom Poppendieck (2003).

Ciertos autores, como el ya citado Fowler (2008), afirman que Lean y Agile poseen una filosofía similar al enfocarse en las personas y en planes adaptativos. Desde esta perspectiva, Lean y Agile no son alternativas, sino que se entremezclan: si se aplica Agile, se está aplicando Lean y viceversa. Bajo la mirada de los Poppendieck, Lean no es una metodología sino que representa un conjunto de herramientas de pensamiento capaces de mezclarse con cualquier enfoque de desarrollo ágil. (Poppendieck & Poppendieck, 2003).

Otros autores, como Robert Charette, señalan que existen diferencias entre Lean y Agile. Según él, la diferencia clave consiste en que Agile es un enfoque *"bottom-up"* o ascendente, mientras que Lean es uno *"top-down"* o descendente (Charette, 2003). En este sentido, también existen posiciones que destacan a Lean como una guía más abarcativa a la hora de elegir prácticas de desarrollo apropiadas para contextos específicos y para situaciones que exceden al mero desarrollo de software (Poppendieck & Cusumano, 2012).

A los fines de este trabajo, no se hará hincapié en las diferencias señaladas en el último párrafo sino que se adoptará la perspectiva detallada precedentemente de que Lean y Agile se entremezclan.

2.3. Proyectos y procesos de software analítico

2.3.1. Sistemas de aprendizaje automático

Como se mencionó en la sección 1.1 de este trabajo, los proyectos analíticos se pueden definir como aquellos destinados a generar información a partir del análisis sistemático de datos (Kohavi, et. al., 2002). Uno de los desarrollos posibles de estos proyectos son los sistemas de aprendizaje automático, sobre los que se enfocará este trabajo.

Los sistemas de *Machine Learning* o Aprendizaje Automático (ML, por sus siglas en inglés) pertenecen a la disciplina de Inteligencia Artificial (Hastie, Tibshirani, & Friedman, 2008). Sin embargo, se diferencian del resto de sistemas analíticos ya que poseen la

capacidad de aprender de los mismos datos (L'heureux, Grolinger, Elyamany, & Capretz, 2017).

La finalidad del aprendizaje automático, como disciplina en sí, es desarrollar técnicas que les permitan a los sistemas aprender y resolver problemas por sí mismos. Aprender, por un lado, implica identificar patrones en millones de datos. Hacerlo de forma automática, por otra parte, hace referencia a la mejora de los sistemas en forma autónoma a lo largo del tiempo. Precisamente, la razón por la cual McKinsey ha declarado que Machine Learning será uno de los principales impulsores de la revolución de *Big Data* (Manyika, y otros, 2011) es su capacidad de aprender de los datos dando soporte a decisiones y predicciones basadas en ellos.

Las dos categorías principales de aprendizaje automático son: aprendizaje supervisado cuando tanto las entradas como las salidas deseadas son conocidas y aprendizaje no supervisado cuando no se conocen las salidas esperadas (L'heureux, Grolinger, Elyamany, & Capretz, 2017). La presunción de que los algoritmos de ML generan predicciones más precisas con más datos (Grolinger, y otros, 2014) y la esencia centrada en datos de su flujo de trabajo (Amershi, y otros, 2019) hacen que estos sistemas presenten características y problemáticas particulares.

2.3.2. Características y problemáticas de los sistemas de aprendizaje automático

Siguiendo los hallazgos de Amershi y otros (2019) en Microsoft, los sistemas de aprendizaje automático tienen tres características fundamentales que los distinguen del desarrollo de software en otros dominios de aplicación:

1. Descubrir, administrar y versionar datos es inherentemente más complejo que hacer lo mismo con código fuente.

2. Para construir modelos flexibles, los equipos deben no sólo tener habilidades de ingeniería de software sino también de ML.
3. Es más difícil mantener una separación entre módulos de un sistema de aprendizaje automático en relación a otros tipos de software.

Estas características distintivas hacen que los sistemas de ML no sólo posean todos los problemas de mantenimiento del código de fuente tradicional sino, además, aquéllos específicos al aprendizaje automático (Sculley, et. al., 2015).

Tomando los desafíos de ML planteados por L'Heureux y otros (2017), es posible resumir las principales problemáticas de los sistemas de aprendizaje automático en:

1. La maldición de la modularidad — *Curse of Modularity*

Muchos algoritmos de ML se basan en el supuesto de que los datos siendo procesados pueden almacenarse completamente en memoria o en un archivo en disco. Cuando esto no es cierto, familias enteras de algoritmos se ven afectadas.

2. La maldición de la dimensionalidad — *Curse of Dimensionality*

La dimensionalidad de los datos consiste en su número de atributos o características. A medida que ésta crece, la capacidad predictiva de un algoritmo, disminuye y así sucede con la precisión de un sistema de aprendizaje automático.

3. Ingeniería de atributos — *Feature Engineering*

Feature Engineering (FE) es el proceso de crear atributos, usualmente usando conocimiento del dominio, para mejorar la performance de los sistemas de ML. A medida que el volumen de los datos aumenta, FE se vuelve una tarea cada vez más complicada.

4. Localización de los datos — *Data Locality*

Los datos no sólo pueden encontrarse distribuidos en distintas ubicaciones físicas por cuestiones de almacenamiento y volumen, sino también por su variedad. Esto genera problemas de latencia e incrementa el tráfico de red requerido.

5. Heterogeneidad de los datos — *Data Heterogeneity*

Cada vez resulta más frecuente integrar datos de distintas fuentes u orígenes. Esto implica heterogeneidad en su tipo, formato, modelo de datos y significado. Un alto nivel de heterogeneidad dificulta la integración de los datos, la cual es necesaria para agregar valor.

6. Disponibilidad de los datos — *Data Availability*

El supuesto de que todos los datos necesarios se encuentran a disposición ya no es siempre cierto en el mundo de ML. Tanto en sistemas de tiempo real, como aquellos que no lo son, por cuestiones de volumen o velocidad de los datos, éstos pueden actualizarse continuamente.

7. *Concept Drift*

Cuando los datos no son estacionarios sino que son recibidos constantemente, los modelos de ML ya entrenados han sido construidos empleando datos que ya no reflejan la distribución real del fenómeno bajo estudio. Cuando esto sucede, la performance y la precisión de los sistemas de aprendizaje automático disminuyen.

8. Procedencia de los datos — *Data Provenance*

Se entiende por Data Provenance al proceso de rastrear y registrar el origen de los datos y sus movimientos. El fin de realizar esto es identificar las causas de posibles errores. Sin embargo, este proceso agrega complejidad y costo computacional a los sistemas de ML.

2.3.3. Ciclo de vida de procesos analíticos

Retomando lo planteado en la sección 1.4, el objetivo general de este trabajo es determinar el impacto sobre la rentabilidad de la compañía multinacional al optimizar

proyectos en el lifecycle de sus procesos analíticos. De aquí la importancia de diferenciar entre procesos y proyectos de software.

Un proceso de software es una serie de actividades relacionadas que conducen a la elaboración de productos de software (Sommerville, 2011). Por otra parte, un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único (Project Management Institute, 2017). En el caso de un proyecto de software, este resultado único es, en efecto, un producto o servicio de software. De las definiciones anteriores, es posible establecer las diferencias entre proceso y proyecto de software en la Tabla 1.

Tabla 1. Diferencias entre proceso y proyecto de software.

Proceso de Software	***Proyecto de Software***
No posee un inicio y un fin determinados	Posee un inicio y fin determinados
Abarca diferentes resultados o proyectos	Crea un único resultado
Genera valor a través de proyectos	Genera valor a través de entregables
Se gestiona con modelos de ciclo de vida	Se gestiona con metodologías
Foco de mejora: incrementar su repetibilidad	Foco de mejora: incrementar su eficiencia

Fuente: elaboración propia

Otra distinción crucial es la existente entre modelo de proceso, ciclo de vida y metodología. Comparando el primero con el último de estos conceptos: mientras que un modelo describe qué hacer, una metodología describe cómo hacerlo. Es decir, un modelo es descriptivo mientras que una metodología es prescriptiva (Ruparelia, 2010). Ahora bien, si se compara un modelo de proceso con un ciclo de vida, el primero representa una estructura y un enfoque para las actividades, acciones y tareas que se requieren para construir software; mientras que un ciclo de vida es un conjunto de elementos del proceso y un flujo para el trabajo del mismo (Pressman, 2010). Dicho de otra forma, un ciclo de vida[4] es una

[4] Se consideran sinónimos a ciclo de vida, modelo de ciclo de vida y modelo de proceso prescriptivo

representación simplificada de las etapas de un proceso. Los dos modelos de ciclo de vida más difundidos son el modelo en cascada y el modelo incremental, representados en la figura 3 y 4 respectivamente.

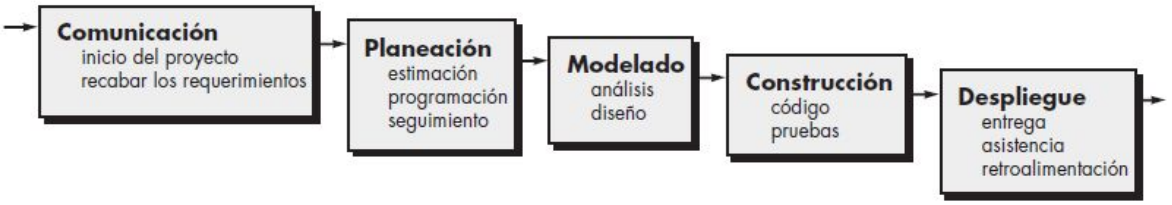

Figura 3. Modelo de ciclo de vida en cascada
Fuente: Pressman, R. S. (2010). Ingeniería del software: Un enfoque práctico

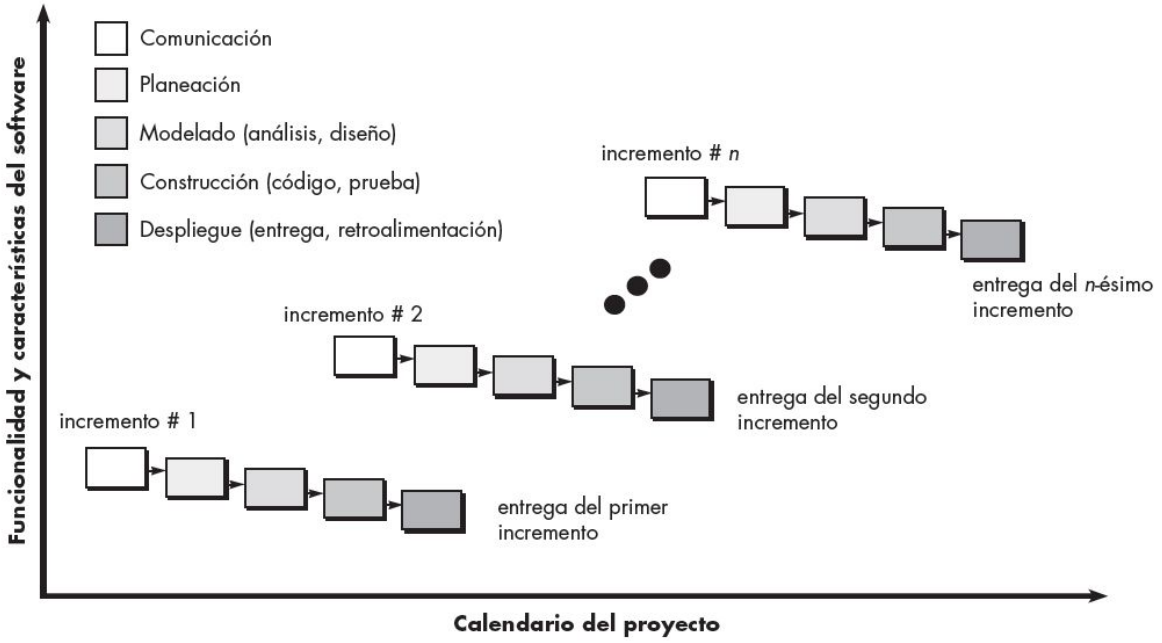

Figura 4. Modelo de ciclo de vida incremental
Fuente: Pressman, R. S. (2010). Ingeniería del software: Un enfoque práctico

¿Cómo aplican los conceptos anteriores a los sistemas de software analítico y, más precisamente, a los sistemas de aprendizaje automático?

En cuanto al modelo de proceso, en sistemas analíticos las tres “actividades estructurales” elementales, en términos de Pressman (2010), estarían representadas por tres conceptos: datos, descubrimiento y despliegue.

Con respecto al ciclo de vida, es posible desagregar cada una de las actividades anteriores en un flujo de trabajo o secuencia de etapas específicas, en el marco de sistemas de aprendizaje automático. En primer lugar, un modelo de ciclo de vida para el flujo de trabajo de datos puede representarse en la figura 5; mientras que uno para las fases de descubrimiento y despliegue (construcción de modelos de aprendizaje automático y su puesta en producción), en la figura 6.

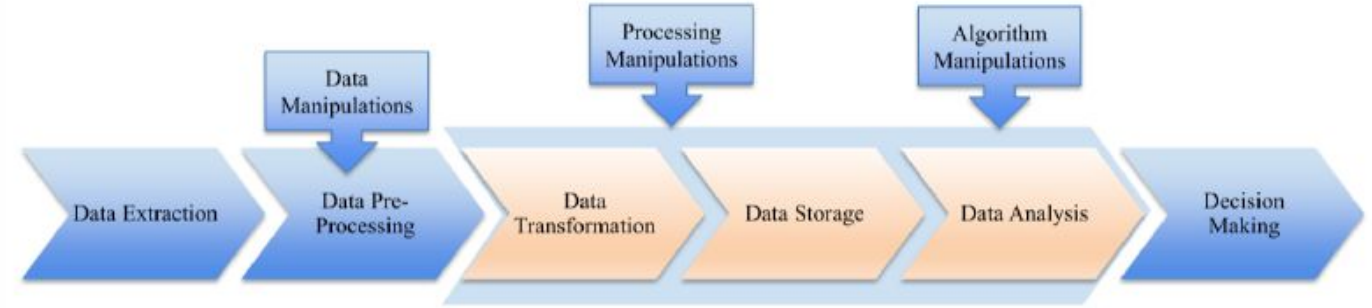

Figura 5. Flujo de trabajo de datos
Fuente: L'heureux, Grolinger, Elyamany, & Capretz (2017)

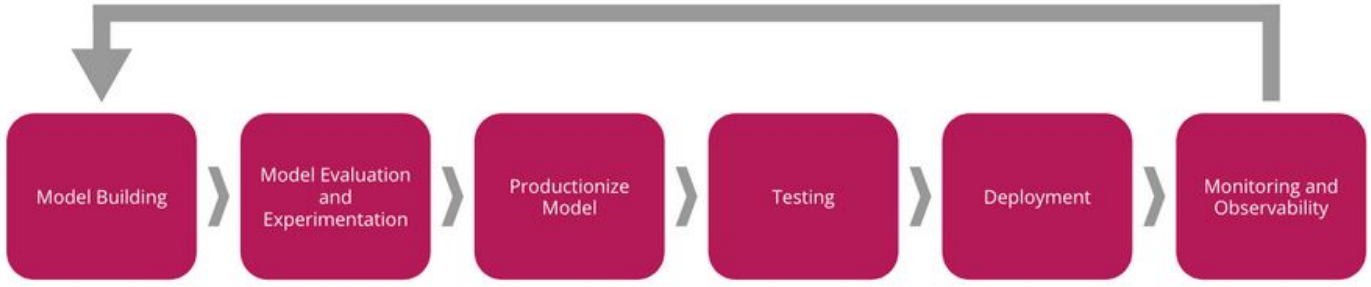

Figura 6. Flujo de trabajo de descubrimiento y despliegue
Fuente: Sato, Wider, & Windheuser (2019)

Por último, en cuanto a metodologías, se desarrollará en la próxima sección cuál metodología de extracción del conocimiento se empleará para definir las entradas, las salidas y los pasos de las actividades de los flujos de trabajo mencionados.

3. Metodología

3.1. Metodología de investigación

Se describe a continuación el marco metodológico del presente trabajo. Para ello, se detallarán los siguientes aspectos de dicho marco: diseño y tipo de investigación, metodología seleccionada, etapas de la investigación y trabajo experimental a realizar.

En cuanto al diseño y tipo de investigación, se adoptará un enfoque cualitativo a la hora de conducir el estudio a realizar. Este enfoque admite cierta subjetividad en el conjunto de procesos de investigación, aplica la lógica inductiva y emplea a la teoría como un marco de referencia (Hernández Sampieri, Fernández Collado, & Baptista Lucio, 2014). En este trabajo de tesis, un diseño cualitativo permitirá aplicar conceptos de Lean Software Development al análisis comparativo de aplicaciones concretas de prácticas de dicha disciplina, estudiando cómo estas últimas afectan los problemas que, en general, presentan los sistemas de aprendizaje automático.

La metodología seleccionada para llevar a cabo esta investigación es aquélla denominada por Carlos Matus (1998) como Intervención Profesional. La misma consiste en atravesar cuatro momentos o etapas sucesivas en la realización de un trabajo profesional cuyo fin es la investigación cualitativa de un determinado problema.

En una primera etapa, el momento explicativo, se deben definir, con la mayor precisión posible, los problemas a resolver a través de la intervención profesional. Aquí se revelará la mayor cantidad de información posible en pos de plantear y formular el problema de investigación, así como las dificultades propias de los sistemas basados en aprendizaje automático y sus efectos en la rentabilidad de la compañía bajo estudio, considerando el objetivo general y el alcance de este trabajo.

En una segunda etapa, el momento normativo, se debe trabajar sobre cómo debería ser la realidad en contraposición a lo que es. Durante esta etapa, se detallarán los objetivos de investigación, las características y metodologías de desarrollo recomendadas para sistemas de aprendizaje automático y, por último, los beneficios esperados de las prácticas de Lean Software Development (LSD).

En una tercera etapa, el momento estratégico, se debe elaborar el planeamiento estratégico completo para alcanzar la situación futura deseada, descrita en la etapa anterior. Aquí, se establecerán los casos de aplicación concretos de prácticas de LSD a realizar, junto con el marco comparativo de referencia a emplearse para su posterior análisis.

La cuarta y última etapa, el momento táctico, consiste en puntualizar específicamente qué actividades se deberán ejecutar para alcanzar dicho estado futuro deseado. En esta etapa, se realizará el análisis comparativo del conjunto de casos de aplicación enumerado en el momento estratégico, concluyendo acerca de sus efectos sobre las problemáticas de los sistemas de aprendizaje automático y la rentabilidad de la empresa analizada.

Finalmente, cabe señalar que el trabajo experimental a realizar en la tesis en cuestión consiste, precisamente, en el análisis comparativo de aplicaciones concretas de prácticas de Lean Software Development en el desarrollo de sistemas de aprendizaje automático, así como los posibles desarrollos que este análisis requiera.

3.2. Metodología de gestión de proyectos y mejora de procesos analíticos

Con el foco puesto en determinar el impacto sobre la rentabilidad de la compañía multinacional en cuestión, al optimizar proyectos en el lifecycle de sus procesos analíticos, y habiendo detallado el marco teórico y metodológico de este trabajo, resulta menester establecer concretamente qué metodologías se aplicarán.

Bajo un enfoque inductivo que permita un análisis desde granularidad fina hacia granularidad media y gruesa, se resumen en la Tabla 2 las metodologías de desarrollo, de gestión de proyectos y de mejora de procesos que se aplicarán en este trabajo.

Tabla 2. Metodologías de desarrollo, proyectos y mejora de procesos seleccionadas.

Nivel Metodológico	***Metodología Elegida***
Metodología de desarrollo	CRISP-DM
Metodología de gestión de proyectos	Scrum
Metodología de mejora de procesos	GQM

Fuente: elaboración propia.

Comenzando por la metodología a aplicar de granularidad más fina, se seleccionó CRISP-DM como metodología de desarrollo de sistemas de aprendizaje automático. CRISP-DM (CRoss Industry Standard Process for Data Mining) fue presentada en el año 1999 por las empresas SPSS, DaimlerChrysler y NCR (Chapman, y otros, 2000). Esta metodología se estructura jerárquicamente por tareas de cuatro niveles diferentes de abstracción, que van desde lo general a lo específico (Wirth & Hipp, 2000). CRISP-DM propone, en el nivel más abstracto, seis fases para el proceso de descubrimiento de conocimiento, las cuales se observan en la Figura 7. Cabe destacar que estas fases no necesariamente se suceden secuencialmente.

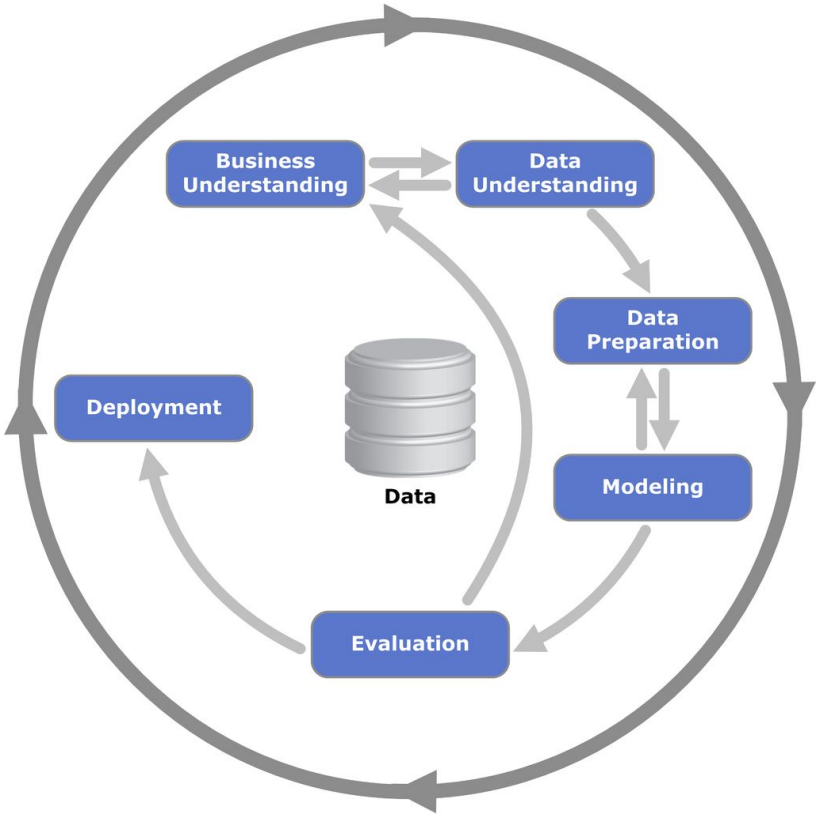

Figura 7. Fases de CRISP-DM
Fuente: Wikipedia (2020)

En un segundo nivel de análisis, el marco de trabajo elegido para la gestión de proyectos es el desarrollado en la sección 2.2.2 de este trabajo: el *framework* Scrum. Sin ahondar en la descripción ya efectuada de este marco de trabajo ágil, cabe destacar que se aplicarán en el mismo algunos conceptos de la Guía del PMBOK (Project Management Institute, 2017). Siguiendo a Schwalbe (2012), es posible ejecutar las actividades de Scrum considerando su relación con los grupos de procesos definidos por el PMBOK del PMI. Por lo cual, en este trabajo se aplicarán, según fuese necesario, prácticas del PMBOK durante la definición de roles, realización de eventos y entrega de artefactos Scrum. Esto incrementará la repetibilidad de los proyectos, conectando la metodología de gestión de proyectos con su análoga de mejora de procesos.

Por último, en cuanto a la metodología de mayor tamaño de grano, se ha optado por un enfoque ágil para la mejora de procesos. Este enfoque se encuentra orientado al desarrollo iterativo y la reducción de las sobrecargas en el proceso de software (Sommerville, 2011). Sin embargo, para cumplir con el objetivo general de este trabajo, se aplicará la metodología *Goal-Question-Metric* (GQM) o Meta-Pregunta-Métrica con un único objetivo: determinar el impacto sobre la rentabilidad al incrementar la eficiencia de proyectos analíticos. Las metas de esta metodología se encuentran en un nivel conceptual, mientras que las preguntas, en uno operativo y las métricas (tiempo, recursos y ocurrencias de un cierto evento), en uno cuantitativo (Basili, Caldiera, & Rombach, 1994). Se incluye un ejemplo de esta metodología en la Figura 8.

Goal	Purpose Issue Object (process) Viewpoint	Improve the timeliness of change request processing from the project manager's viewpoint
Question		What is the current change request processing speed?
Metrics		Average cycle time Standard deviation % cases outside of the upper limit
Question		Is the performance of the process improving?
Metrics		$\frac{\text{Current average cycle time}}{\text{Baseline average cycle time}} * 100$ Subjective rating of manager's satisfaction

Figura 8. Ejemplo de aplicación de GQM
Fuente: Basili, Caldiera, & Rombach (1994)

4. Desarrollo

4.1. Diseño de los experimentos

Siguiendo la metodología de investigación planteada en la sección 3.1, se procede a continuación a desarrollar la tercera de las etapas de la Intervención Profesional planteada en este trabajo. Habiendo relevado tanto las problemáticas como las características deseables de los sistemas de aprendizaje automático —es decir, habiendo culminado con los momentos explicativos y normativos, respectivamente—, el próximo paso consiste en abordar el momento estratégico, tal como lo define Carlos Matus (1998).

En esta tercera etapa, se detalla el plan a seguir durante el desarrollo del presente trabajo de campo. Este plan consiste fundamentalmente en cuatro aspectos:

1. Diseño de los experimentos
2. Ejecución de los experimentos
3. Análisis comparativo de resultados
4. Discusión y síntesis de resultados

En la presente sección, se hace foco en diseñar los experimentos a realizar en pos de cumplir el objetivo general propuesto: determinar el impacto sobre la rentabilidad de la compañía multinacional al optimizar proyectos en el lifecycle de sus procesos analíticos.

En concordancia con lo expuesto en la sección 3.2, se aplicará un marco de múltiples metodologías en estos experimentos. Invirtiendo el orden de la Tabla 2, es decir, yendo desde lo general hacia lo particular, se comienza por definir el modelo GQM que guiará los experimentos. Para este trabajo de campo, el modelo GQM se incluye en la Tabla 3.

Tabla 3. Modelo GQM propuesto.

Nivel	***Identificador***	***Definición***
Objetivo (*Goal*)	Propósito	Incrementar
	Asunto	la rentabilidad
	Objeto	de los proyectos analíticos de la compañía multinacional
	*Modo**	aplicando prácticas de Ingeniería de Software Lean y de gestión de proyectos
Pregunta (*Question*)	$\boldsymbol{P_i}$	¿Cuál es el costo de la etapa ***i*** ?
Métricas (*Metrics*)	$\boldsymbol{M_{i,0}}$	Horas promedio por proyecto estimadas para la etapa ***i***
	$\boldsymbol{M_{i,j}}$	Horas destinadas a la etapa ***i*** en el experimento ***j***

Fuente: elaboración propia.

**Nota:* se ha cambiado el identificador original de "Punto de vista (*Viewpoint*)" descrito en Basili, Caldiera & Rombach (1994) por "Modo" a fines de adaptar la metodología a las necesidades del trabajo de campo.

Como se puede observar en la Tabla 3, este modelo GQM consistirá en un único objetivo o *goal*, el cual se define a partir del objetivo general de este trabajo en conjunción con el último de sus objetivos específicos.

A su vez, cabe destacar que se toma, como principal factor de costos, al número de horas destinadas a cada etapa del modelo de ciclo de vida incremental, acumuladas para los distintos incrementos resultantes de un proyecto analítico. Ver Figura 4 en sección 2.3.3. Siguiendo a Pressman (2010), estas etapas pueden enumerarse de la forma expuesta en la Tabla 4.

Tabla 4. Etapas del modelo de ciclo de vida incremental.

i	***Etapa***	***Actividades***
1	Comunicación	Elicitación de requerimientos
2	Planeación	Estimación, programación y seguimiento
3	Modelado	Análisis y diseño
4	Construcción	Codificación y pruebas
5	Despliegue	Entrega y retroalimentación

Fuente: elaboración propia en base a Pressman (2010).

Cabe destacar que estas fases, si bien se suceden secuencialmente dentro de un incremento, pueden solaparse en el tiempo u ocurrir simultáneamente si se trata de etapas de diferentes incrementos o entregables de un proyecto. Ver Figura 4 en sección 2.3.3.

A continuación, se detallan los experimentos a realizar en el presente trabajo, dispuestos en la Tabla 5.

Tabla 5. Experimentos a realizar.

j	***Identificador***	***Experimento***
0	BASELINE	Situación actual
1	AGILE	Prácticas ágiles en un proyecto de operacionalización analítica
2	SCM-REING	SCM en un proyecto analítico de migración y reingeniería de software
3	SCM-REUSO	SCM orientado al reuso en un proyecto sin experiencia local previa
4	CI/CD	Integración y entrega continua en un proyecto de calificación crediticia

Fuente: elaboración propia.

De esta forma, considerando lo expuesto en las tablas 3, 4 y 5, es posible determinar matricialmente las métricas del modelo GQM propuesto, para Mij con i desde 1 a 5 y j de 0 a 4, en la Tabla 6.

Tabla 6. Determinación de métricas GQM.

	BASELINE	*AGILE*	*SCM-REING*	*SCM-REUSO*	*CI/CD*
Comunicación	$M_{1,0}$	$M_{1,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$
Planeación	$M_{2,0}$	$M_{2,1}$	$M_{2,2}$	$M_{2,3}$	$M_{2,4}$
Modelado	$M_{3,0}$	$M_{3,1}$	$M_{3,2}$	$M_{3,3}$	$M_{3,4}$
Construcción	$M_{4,0}$	$M_{4,1}$	$M_{4,2}$	$M_{4,3}$	$M_{4,4}$
Despliegue	$M_{5,0}$	$M_{5,1}$	$M_{5,2}$	$M_{5,3}$	$M_{5,4}$

Fuente: elaboración propia.

De esta forma, por ejemplo, $M_{3,0}$ representa el número de horas promedio, por proyecto, destinadas a la etapa de modelado considerando una muestra de proyectos analíticos pasados. Análogamente, $M_{3,2}$ representa el número de horas destinadas a modelado durante el experimento *"SCM en un proyecto analítico de migración y reingeniería de software"*. Comparando $M_{3,2}$ con $M_{3,0}$ es posible determinar el impacto sobre la rentabilidad de la compañía al aplicar una práctica específica de Ingeniería de Software Lean o de gestión de proyectos.

Genéricamente:

- Si $M_{i,j} > M_{i,0}$, para j > 0, entonces las horas destinadas durante el experimento ***j*** a la etapa ***i*** superan a la media y el impacto en la rentabilidad es **negativo**.
- Si $M_{i,j} < M_{i,0}$, para j > 0, entonces las horas destinadas durante el experimento ***j*** a la etapa ***i*** son menores a la media y el impacto en la rentabilidad es **positivo**.
- Si $M_{i,j} = M_{i,0}$, para j > 0, entonces las horas destinadas durante el experimento ***j*** a la etapa ***i*** son iguales a la media y el impacto en la rentabilidad es **indiferente**.

Acumulativamente:

- Si la suma de $M_{i,j}$ es mayor que la suma de $M_{i,0}$ para $i = 1, 2, ..., 5$, entonces el experimento ***j*** posee un impacto acumulado a lo largo de todas sus etapas **negativo** en la rentabilidad, ya que requiere un número total de horas superior al promedio.
- Si la suma de $M_{i,j}$ es menor que la suma de $M_{i,0}$ para $i = 1, 2, ..., 5$, entonces el experimento ***j*** posee un impacto acumulado a lo largo de todas sus etapas **positivo** en la rentabilidad, ya que requiere un número total de horas menor al promedio.
- Si la suma de $M_{i,j}$ es igual a la suma de $M_{i,0}$ para $i = 1, 2, ..., 5$, entonces el experimento ***j*** posee un impacto acumulado a lo largo de todas sus etapas **indiferente** en la rentabilidad, ya que requiere un número total de horas igual al promedio.

Cabe señalar que este análisis acumulativo no será aplicado para la totalidad de experimentos a ejecutarse, ya que muchas de las prácticas contenidas en ellos representan una variación a un subconjunto (y no a la totalidad) de etapas del modelo de ciclo de vida.

4.2. Ejecución de los experimentos

4.2.1. Prácticas ágiles en un proyecto de operacionalización analítica

En este primer experimento, se aplicó el marco de trabajo Scrum a un proyecto de operacionalización analítica. El objetivo de este proyecto consistía en relevar, analizar, diseñar, implementar, probar y desplegar un sistema —parte hecho a medida y parte enlatado— que sirviera para gestionar el ciclo de vida de modelos analíticos desarrollados por la empresa cliente.

El cliente de la compañía multinacional, una de las tres empresas de telecomunicaciones más grandes del país, había adquirido una plataforma de analítica avanzada para integrar sus diversas aplicaciones de Inteligencia Artificial en un único sistema. La implementación de esta plataforma requería desarrollar ciertas integraciones a

medida entre las aplicaciones del cliente y dicha plataforma analítica. Las características elementales de este proyecto se resumen en la Tabla 7.

Tabla 7. Características del experimento N° 1: AGILE.

Atributo	***Descripción***
Problemáticas a resolver	Localización, heterogeneidad y procedencia de los datos
Tipo de práctica aplicada	De gestión
Práctica/solución aplicada	Marco de trabajo Scrum
Principios Lean subyacentes	Entregar rápido e involucrar a todos
Fases CRISP-DM bajo estudio	Entendimiento del negocio y evaluación

Fuente: elaboración propia.

Otra de las características fundamentales de este proyecto fue el altísimo nivel de interacción requerido entre la compañía multinacional bajo análisis y la empresa de telecomunicaciones: al existir un fuerte acoplamiento entre el sistema a desarrollar y el ecosistema de aplicaciones preexistentes del cliente, un proceso de retroalimentación constante y bidireccional debió llevarse a cabo.

Para dar respuesta a estos requisitos de interacción, se aplicó el marco de trabajo Scrum. Se definió una duración de 2 semanas o 14 días para cada Sprint, comenzando miércoles y terminando martes. Se programaron sesiones de Product Backlog Refinement con el Product Owner de este experimento (un líder técnico de la empresa cliente). El resto de las ceremonias celebradas consistieron en los eventos contemplados por la Guía de Scrum (Schwaber & Sutherland, 2017).

En pos de gestionar los artefactos y los entregables de este experimento, se recurrió al software Jira® de la firma australiana Atlassian©. Este último se utilizó para administrar el Product Backlog, el Sprint Backlog, las Historias de Usuario, el alcance de cada iteración del producto, etcétera. Si bien por cuestiones de confidencialidad no es posible adjuntar capturas

de los tableros ni de los backlogs generados, se incluyen como Figuras 9 y 10 ejemplos de este software.

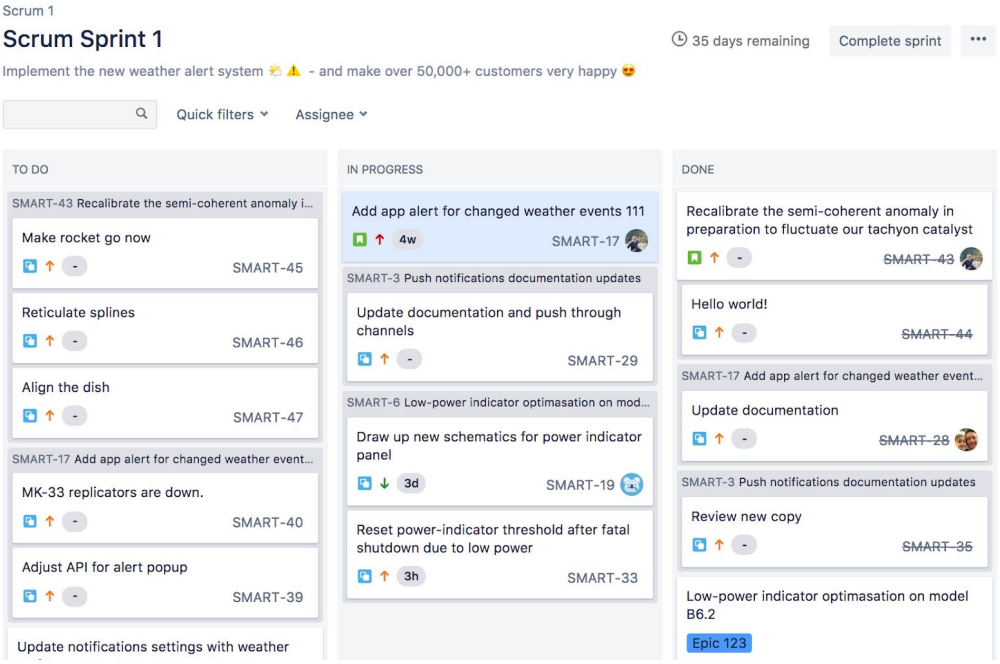

Figura 9. Ejemplo de un tablero Scrum en Jira
Fuente: https://support.atlassian.com/jira-software-cloud/docs/use-active-sprints/

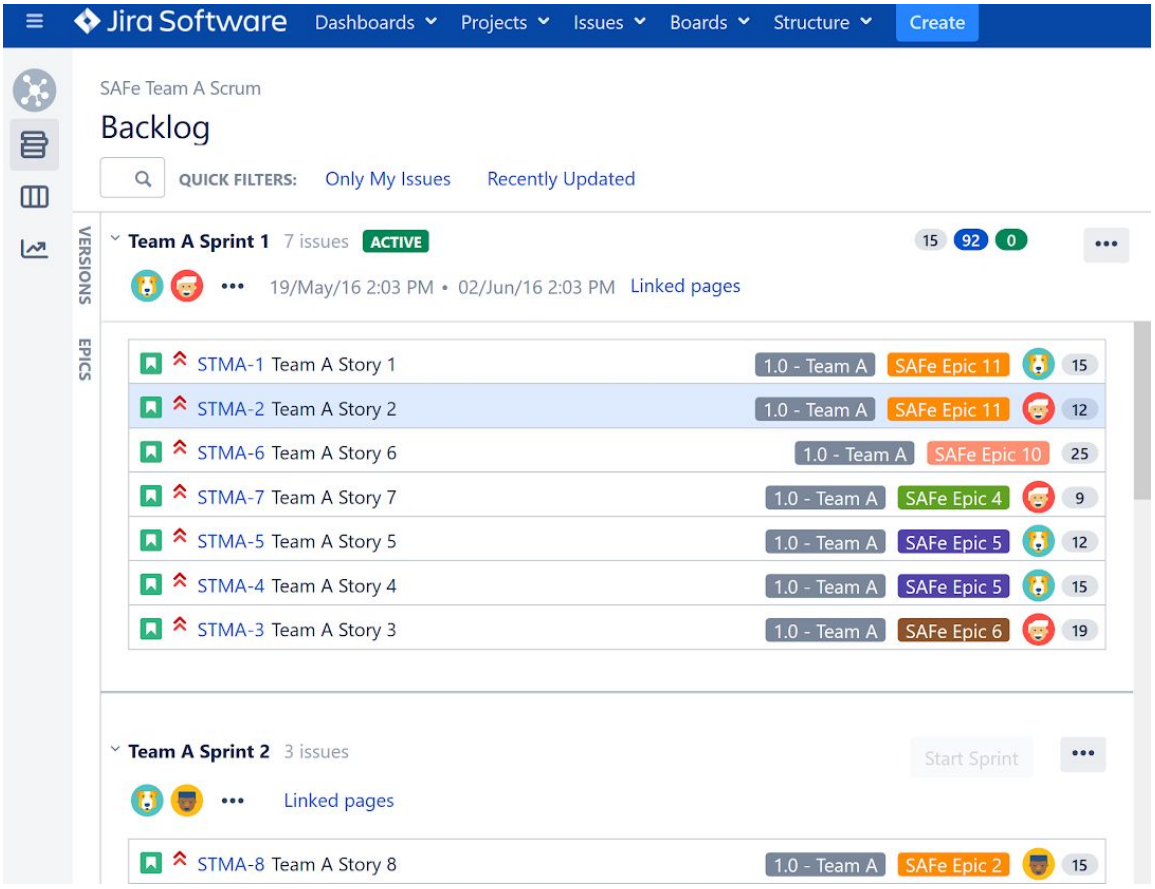

Figura 10. Ejemplo de Sprint Backlogs en Jira
Fuente: https://wiki.almworks.com/display/structure/Structure+on+Agile+Boards

4.2.2. SCM en un proyecto de migración y reingeniería de software

En este segundo experimento, se aplicaron técnicas y herramientas pertenecientes a la disciplina de Gestión de la Configuración de Software (o *Software Configuration Management* —SCM— en inglés) a un proyecto de migración y reingeniería de software analítico. El objetivo de este proyecto consistía en migrar un sistema de aprendizaje automático a una nueva plataforma analítica.

El propósito de este sistema era maximizar los ingresos de los distintos paquetes y productos de telefonía prepaga comercializados por la empresa cliente. Para ello, a través de un análisis predictivo de los consumidores de esta última, se realizaban campañas de marketing direccionadas a los consumidores cuya probabilidad de efectuar un *upsell* o *cross-sell* fuera mayor.

Las características técnicas de la nueva plataforma implicaron que se deba reescribir gran parte del código del sistema de Machine Learning en cuestión, para que el mismo funcione de forma distribuida y en memoria, optimizando así su performance. Las características elementales de este proyecto se resumen en la Tabla 8.

Tabla 8. Características del experimento N° 2: SCM-REING.

Atributo	***Descripción***
Problemáticas a resolver	Maldición de la modularidad y localización de los datos
Tipo de práctica aplicada	Técnica
Práctica/solución aplicada	Gestión de la Configuración de Software
Principios Lean subyacentes	Aprender constantemente y seguir mejorando
Fases CRISP-DM bajo estudio	Modelado, evaluación y despliegue

Fuente: elaboración propia.

Si bien este proyecto no fue la primera migración de un sistema analítico realizada en la filial argentina de la compañía multinacional, por lo que ya existían estimaciones de

referencia, la principal dificultad del mismo consistía en que se trataba de la primera reingeniería de software para que una aplicación de Machine Learning pase de procesamiento secuencial a distribuido y en memoria. Esta dificultad técnica implicó adoptar una estrategia de migración que permitiera a ambas versiones del sistema (la actual y la nueva) coexistir en distintos ambientes productivos y en distintas plataformas.

En pos de mantener y continuar tanto el mantenimiento como el desarrollo de dos versiones completamente diferentes del código fuente de un mismo proyecto, se recurrió a emplear un sistema de control de versiones distribuido (DVCS, por sus siglas en inglés). La herramienta elegida fue GitLab. GitLab es un servicio web de control de versiones y desarrollo de software colaborativo basado en Git y publicado bajo una licencia de código abierto. El flujo de trabajo adoptado para este desarrollo “multi-versión” se basó en el GitHub Flow, ilustrado en la Figura 11.

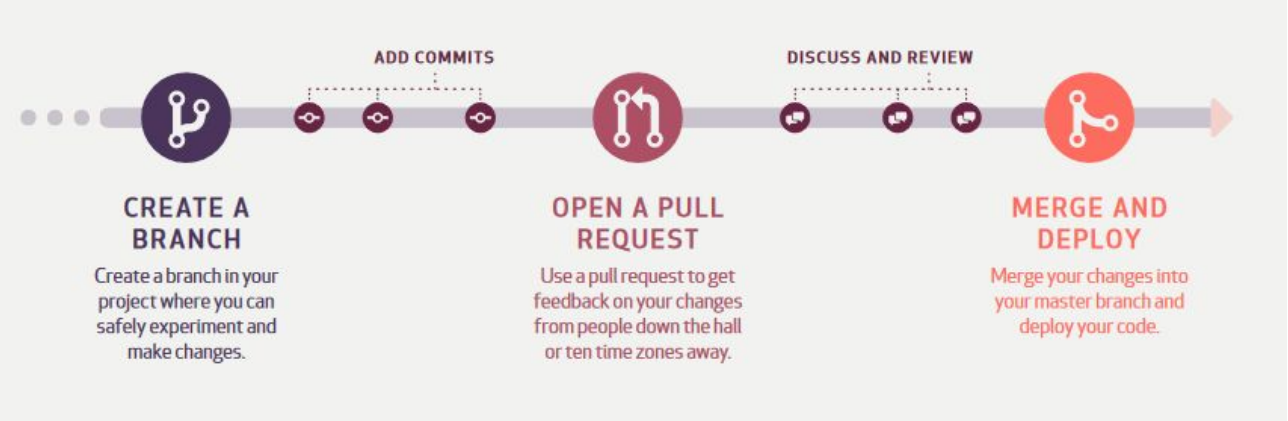

Figura 11. El flujo de trabajo GitHub
Fuente: https://guides.github.com/pdfs/githubflow-online.pdf

En la Figura 12 se observa la estructura de múltiples ramas adoptadas en el proyecto.

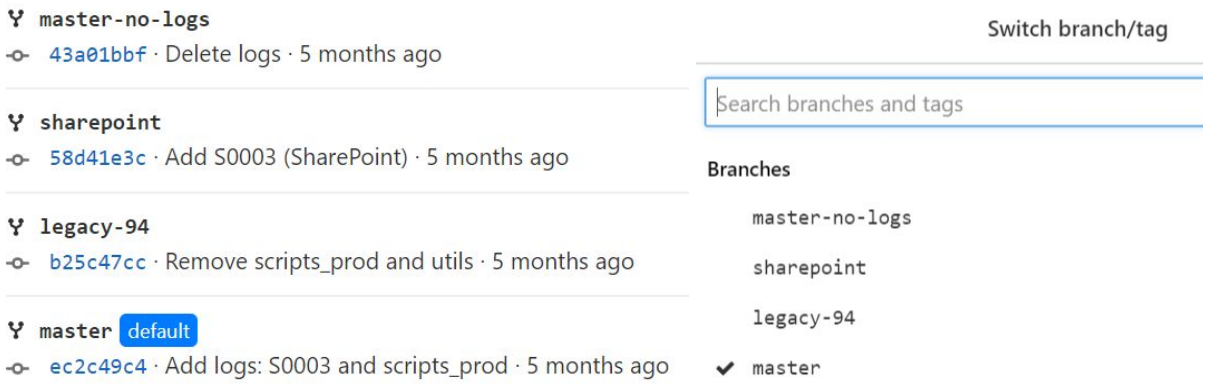

Figura 12. Estructura de ramas de desarrollo adoptada en el experimento N° 2
Fuente: elaboración propia

4.2.3. SCM orientado al reuso en un proyecto sin experiencia local

En este tercer experimento, nuevamente se aplicaron técnicas y herramientas de Gestión de la Configuración de Software (SCM), pero, esta vez, a un proyecto analítico perteneciente al vertical de la compañía multinacional denominado Fraude e Inteligencia de Seguridad (o *Fraud & Security Intelligence* —FSI— en inglés).

Dado el estricto acuerdo de confidencialidad del proyecto, no es posible detallar su propósito pero sí que se trató de un cliente perteneciente al sector público y que su desarrollo implicó realizar implementaciones tanto de modelos analíticos como de código frontend y backend. Otras características de este proyecto se resumen en la Tabla 9.

Tabla 9. Características del experimento N° 3: SCM-REUSO.

Atributo	***Descripción***
Problemáticas a resolver	Localización, heterogeneidad y procedencia de los datos
Tipo de práctica aplicada	Técnica
Práctica/solución aplicada	Gestión de la Configuración de Software
Principios Lean subyacentes	Aprender constantemente y seguir mejorando
Fases CRISP-DM bajo estudio	Preparación de datos, modelado, evaluación y despliegue

Fuente: elaboración propia.

La principal dificultad de este proyecto —lo que diferencia a este experimento del anterior y por lo cual se lo incluye en la comparación— consistía en que no existían previamente en la región otras implementaciones o proyectos que pudieran tomarse de base para la estimación del desarrollo en cuestión. Por ende, debían tomarse como referencia proyectos realizados en otros países e interactuar con equipos globales de la compañía. A su vez, contemplando los estrictos requisitos de auditoría y seguimiento del presente proyecto, era necesario mantener trazabilidad y registro unívoco de cada cambio y desarrollo realizado. Ambas situaciones se reflejan en las Figuras 13 y 14, respectivamente.

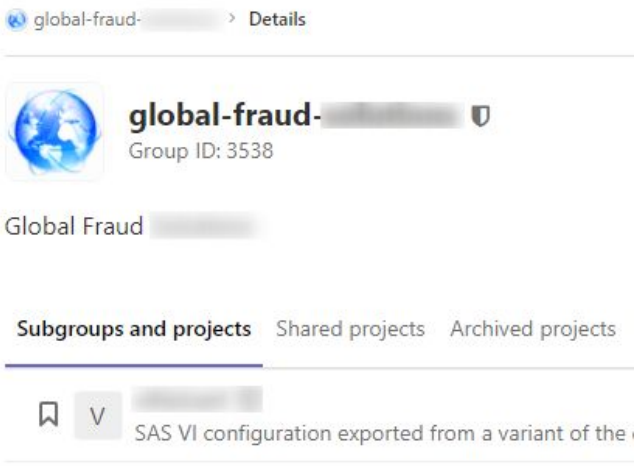

Figura 13. Reuso favorecido por GitLab en el experimento N° 3
Fuente: elaboración propia

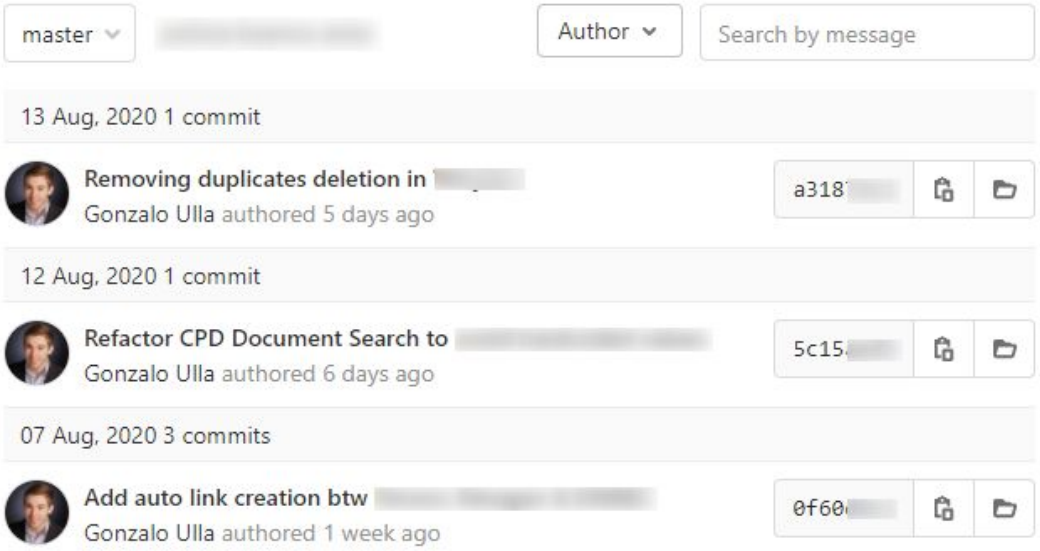

Figura 14. Trazabilidad de cambios en el desarrollo del experimento N° 3
Fuente: elaboración propia

Nuevamente, el DVCS empleado fue GitLab. Esta herramienta favoreció el reuso entre equipos locales y globales de la compañía multinacional y, al mismo tiempo, permitió dar respuesta a los requerimientos de auditoría y trazabilidad.

4.2.4. Integración y entrega continua en un proyecto de calificación crediticia

En este cuarto y último experimento, se aplicaron las prácticas Lean de integración y entrega continua a un proyecto de calificación crediticia desarrollado para un cliente de la industria de servicios financieros. El objetivo de este proyecto consistía en, partiendo de datos provistos por este cliente sobre los consumidores de sus productores financieros, realizar el *credit scoring* o la calificación crediticia de ellos, a fines de determinar su probabilidad de default o incobrabilidad.

El principal propósito de aplicar estas prácticas Lean consistía en reducir el tiempo de ciclo para el desarrollo, entrenamiento y despliegue de nuevos modelos predictivos. Cabe destacar que el proyecto analítico en cuestión debía soportar tanto el entrenamiento como la posterior ejecución de múltiples modelos, potencialmente desarrollados con distintas librerías e incluso diferentes lenguajes de programación. Otro requisito planteado, por lo que se adoptó CI/CD, es que se debía automatizar la ejecución de pruebas, el reentrenamiento de modelos y su puesta de producción ante la disponibilización de nuevos datos como también ante la modificación o incorporación de modelos. Se resumen las características de este proyecto en la Tabla 10.

Tabla 10. Características del experimento N° 4: CI/CD.

Atributo	***Descripción***
Problemáticas a resolver	Localización y disponibilidad de los datos. Concept Drift
Tipo de práctica aplicada	Técnica
Práctica/solución aplicada	Integración Continua (CI) y Entrega Continua (CD)
Principios Lean subyacentes	Eliminar el desperdicio y construir con calidad
Fases CRISP-DM bajo estudio	Preparación de datos, modelado, evaluación y despliegue

Fuente: elaboración propia.

En pos de dar respuesta a los requerimientos planteados, se utilizaron tecnologías tanto de software libre como de software propietario. La arquitectura de alto nivel de la solución desarrollada puede observarse en la Figura 15.

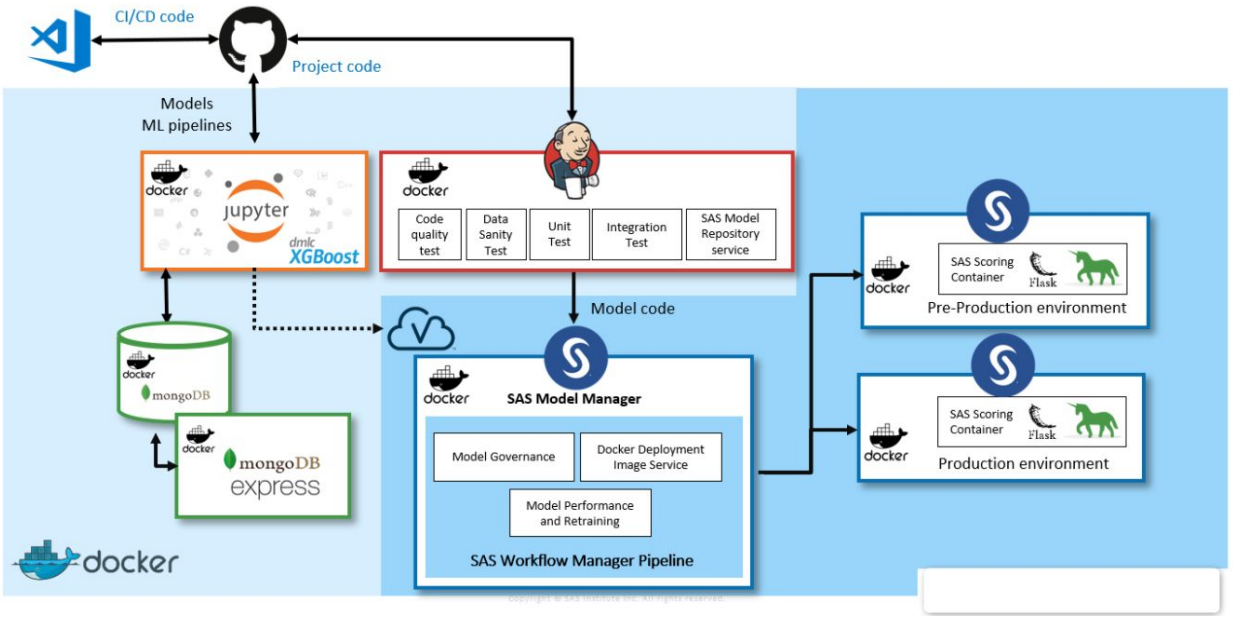

Figura 15. Arquitectura del experimento N° 4
Fuente: elaboración propia

Resulta menester señalar el rol central que tuvo en este proyecto Jenkins como servidor de automatización open source. Esta herramienta se utilizó para automatizar gran parte de las tareas ejecutadas, incluyendo la ejecución de pruebas tales como los Tests Unitarios representados en la Figura 16.

```
stage('Unit Test for Score code') {

    agent {
        docker {
            image 'in92/devops_pyunit:v1.1'
            }
    }

    steps {
        sh 'python project/3_CI_CD/3_unit_test/unit_test_pipeline.py
    }

    post {

        success {
                echo 'Score code file successfully passed Unit Test!'
            }

        failure {
                echo 'Code failed the Unit test, please see logs.'
            }
    }

}
```

Figura 16. Automatización de pruebas con Jenkins en el experimento N° 4
Fuente: elaboración propia

4.3. Análisis de resultados

Se procede, en esta sección, a analizar los resultados obtenidos luego de la ejecución de los experimentos diseñados. Para ello, se incluyen dos tablas por experimento: la primera, detallando el número de horas efectivamente destinadas a cada etapa en comparación al *baseline* del experimento y, la segunda, con el índice de horas requeridas por etapa tomando al *baseline* como base 100 de dicho índice.

El *baseline* o base de comparación para cada experimento resulta de tomar el valor estimado de horas, por etapa, para proyectos similares al del experimento en cuestión. Este valor estimado, a su vez, es obtenido a partir de la experiencia: representa la media aritmética de proyectos de características comparables ejecutados en el pasado y que la compañía multinacional toma como referencia para estimar proyectos del mismo tipo a futuro.

Para el experimento N° 1 AGILE: Prácticas ágiles en un proyecto de operacionalización analítica, los resultados se resumen en las tablas 11 y 12.

Tabla 11. Resultados del experimento N° 1 en horas.

	BASELINE	*AGILE*
Comunicación	600	520
Planeación	360	392
Modelado	800	744
Construcción	1280	1408
Despliegue	320	336
TOTAL	3360	3400

Fuente: elaboración propia.

Tabla 12. Resultados del experimento N° 1 en índice de base 100.

	BASELINE	*AGILE*
Comunicación	100	87
Planeación	100	109
Modelado	100	93
Construcción	100	110
Despliegue	100	105

Fuente: elaboración propia.

Como se puede observar en la tabla 12, la introducción de metodologías ágiles a un proyecto de operacionalización analítica permitió acelerar considerablemente las etapas que hacen eminentemente a la elicitación de requerimientos: comunicación y modelado. El ida y vuelta constante y frecuente que propone agile puede considerarse la causa de esto. Sin embargo, el overhead de agile a la hora de mantener múltiples ceremonias o eventos de seguimiento hizo destinar más tiempo a la etapa de planeación. En proyectos de varios meses de duración, la cadencia de trabajo por la que apuesta Scrum muchas veces acaba ralentizando levemente la etapa de construcción, si esta metodología no es complementada

por otras de origen técnico. No es posible atribuir el incremento de horas para la fase de despliegue a la aplicación de metodologías ágiles.

Para el experimento N° 2 SCM-REING: SCM en un proyecto analítico de migración y reingeniería de software, los resultados se resumen en las tablas 13 y 14.

Tabla 13. Resultados del experimento N° 2 en horas.

	BASELINE	*SCM-REING*
Comunicación	320	296
Planeación	240	256
Modelado	520	504
Construcción	1040	1080
Despliegue	192	128
TOTAL	2312	2264

Fuente: elaboración propia.

Tabla 14. Resultados del experimento N° 2 en índice de base 100.

	BASELINE	*SCM-REING*
Comunicación	100	93
Planeación	100	107
Modelado	100	97
Construcción	100	104
Despliegue	100	67

Fuente: elaboración propia.

De las variaciones anteriores, expresadas en índice de base 100, se observa una clara mejora en los tiempos de despliegue. Esta última puede atribuirse a la implementación de SCM para desarrollar y mantener múltiples versiones de código simultáneamente. Sin embargo, esta mejora fue pagada destinando más horas a la fase de desarrollo. No es posible atribuir el resto de variaciones a la aplicación de SCM en el experimento.

Para el experimento N° 3 SCM-REUSO: SCM orientado al reuso en un proyecto sin experiencia local previa, los resultados se resumen en las tablas 15 y 16.

Tabla 15. Resultados del experimento N° 3 en horas.

	BASELINE	*SCM-REUSO*
Comunicación	700	780
Planeación	400	414
Modelado	1400	1266
Construcción	2200	1920
Despliegue	300	308
TOTAL	5000	4688

Fuente: elaboración propia.

Tabla 16. Resultados del experimento N° 3 en índice de base 100.

	BASELINE	*SCM-REUSO*
Comunicación	100	111
Planeación	100	104
Modelado	100	90
Construcción	100	87
Despliegue	100	107

Fuente: elaboración propia.

Se puede observar que la utilización de SCM para reutilizar desarrollos globales en proyectos similares al del experimento N° 3 permitió reducir las horas destinadas tanto a la etapa de modelado como a la de construcción. Sin embargo, por más de que esta reutilización de componentes de software hizo más fácil el análisis, diseño y codificación del sistema, no logró suplir la falta de experiencia local durante la etapa de despliegue del mismo. Es importante recordar que el *baseline* en este caso representa una referencia internacional. No es posible atribuir las variaciones en las horas de comunicación y planeación a la aplicación de SCM en este experimento.

Para el experimento N° 4 CI/CD: Integración y entrega continua en un proyecto de calificación crediticia, los resultados se resumen en las tablas 17 y 18.

Tabla 17. Resultados del experimento N° 4 en horas.

	BASELINE	*CI/CD*
Comunicación	520	506
Planeación	280	304
Modelado	960	936
Construcción	1480	1352
Despliegue	240	168
TOTAL	3480	3266

Fuente: elaboración propia.

Tabla 18. Resultados del experimento N° 4 en índice de base 100.

	BASELINE	*CI/CD*
Comunicación	100	97
Planeación	100	109
Modelado	100	98
Construcción	100	91
Despliegue	100	70

Fuente: elaboración propia.

Es posible concluir, de la tabla 18, que el impacto de aplicar integración y entrega continua en este cuarto y último experimento afectó principalmente a las etapas de construcción y despliegue, acelerando la codificación, pruebas y posterior puesta en producción del sistema de calificación crediticia en cuestión. Las demás variaciones no resultan atribuibles a la incorporación de CI/CD en este experimento.

4.4. Discusión de resultados

4.4.1. Síntesis de resultados

Partiendo de las tablas 12, 14, 16 y 18, se procede a sintetizar los resultados obtenidos en los cuatro experimentos realizados. Para ello, se detallan estos resultados, expresados como variaciones en índices de base 100, en la Tabla 19.

Tabla 19. Síntesis de resultados obtenidos.

	BASELINE	*AGILE*	*SCM-REING*	*SCM-REUSO*	*CI/CD*
Comunicación	100	87	93	111	97
Planeación	100	109	107	104	109
Modelado	100	93	97	90	98
Construcción	100	110	104	87	91
Despliegue	100	105	67	107	70

Fuente: elaboración propia.

Cabe destacar que, en la tabla anterior, si una celda se encuentra coloreada, significa que la aplicación de una técnica tuvo efectos en determinada etapa del proyecto. Así, por ejemplo en el experimento *AGILE*, hubo una reducción del 7% en las horas destinadas a "Modelado" atribuible a la implementación de metodologías ágiles, pero también hubo un incremento del 5% en las horas de "Despliegue" que no son atribuibles a esta práctica.

4.4.2. Análisis de escenarios

En pos de dar respuesta al objetivo general del presente trabajo, es necesario construir escenarios que permitan determinar el impacto de aplicar simultáneamente distintas prácticas ágiles y Lean, de gestión y técnicas, a múltiples proyectos. Para esto, en primer lugar, es necesario ponderar los resultados obtenidos en las diferentes etapas del modelo de ciclo de vida en función de la participación de cada una de ellas en los experimentos. Tomando como muestra a los cuatro experimentos ejecutados, y el número de horas destinadas a cada etapa

detallado en las tablas 11, 13, 15 y 17, se determina un factor de ponderación por etapa en la Tabla 20.

Tabla 20. Ponderación de etapas del modelo de ciclo de vida.

	AGILE	*SCM-REING*	*SCM-REUSO*	*CI/CD*	***Factor de Ponderación (w)***
Comunicación	0,1529	0,1307	0,1664	0,1549	0,1512
Planeación	0,1153	0,1131	0,0883	0,0931	0,1024
Modelado	0,2188	0,2226	0,2701	0,2866	0,2495
Construcción	0,4141	0,4770	0,4096	0,4140	0,4287
Despliegue	0,0988	0,0565	0,0657	0,0514	0,0681

Fuente: elaboración propia.

Cada valor del vector de factor de ponderación fue calculado determinando la participación de cada etapa del modelo de ciclo de vida en el total de horas consumidas por un cierto experimento. Por ejemplo, para el experimento N° 1 AGILE, la participación de la etapa Comunicación es igual a 520 horas de Comunicación divididas por 3400 horas totales del experimento. A su vez, el valor del factor de ponderación (*w*) para Comunicación surge de promediar la participación de esta etapa en los cuatro experimentos realizados.

Con el vector del factor de ponderación calculado en la tabla 20, y la síntesis de resultados de la Tabla 19, es posible construir tres escenarios: pesimista, base y optimista.

El escenario pesimista consiste en tomar, para cada etapa del ciclo de vida, el promedio de las variaciones en índice base 100 de la Tabla 19 que hayan ocurrido en experimentos que afectaron el resultado de dicha etapa. Dicho de otra manera, este escenario toma, para cada fila, el promedio sólo de las celdas coloreadas en la Tabla 19 (reflejado en la columna *DELTA* de la Tabla 21). Se resume el impacto de este escenario pesimista en la Tabla 21, reflejado en la columna ***w * DELTA***.

Tabla 21. Escenario pesimista.

	Factor de Ponderación (*w*)	*BASELINE*	*DELTA*	*w * BASELINE*	***w * DELTA***
Comunicación	0,1512	100	87	15,12	13,16
Planeación	0,1024	100	109	10,24	11,17
Modelado	0,2495	100	91,5	24,95	22,83
Construcción	0,4287	100	98	42,87	42,01
Despliegue	0,0681	100	81	6,81	5,52
TOTAL	1			100	94,68

Fuente: elaboración propia.

En este escenario pesimista, el impacto de aplicar prácticas tanto de origen técnico como de gestión a los proyectos analíticos representados en los cuatro experimentos, resulta en una reducción del 5,32% en el número de horas requeridas por proyecto (100 - 94,68).

El escenario base consiste en tomar, para cada etapa del ciclo de vida, la peor variación de los índices base 100 de la Tabla 19, pero primando variaciones positivas sobre negativas en aquellos experimentos que afectaron el resultado de dicha etapa. Es decir que, si existe una única celda coloreada para una fila de la Tabla 19, ese será el valor del *DELTA*. Ahora bien, si existen dos o más celdas coloreadas (variaciones atribuibles con respecto al *BASELINE*) para una misma fila de la Tabla 19, se tomará la peor variación positiva (el mayor valor de las celdas verdes) y éste será el valor de la columna *DELTA* en la Tabla 22. Se resume el impacto de este escenario base en la Tabla 22, reflejado en la columna ***w * DELTA***.

Tabla 22. Escenario base.

	Factor de Ponderación (*w*)	*BASELINE*	*DELTA*	*w * BASELINE*	***w * DELTA***
Comunicación	0,1512	100	87	15,12	13,16
Planeación	0,1024	100	109	10,24	11,17
Modelado	0,2495	100	93	24,95	23,21
Construcción	0,4287	100	91	42,87	39,01
Despliegue	0,0681	100	70	6,81	4,77
TOTAL	1			100	91,31

Fuente: elaboración propia.

En este escenario base, el impacto de aplicar prácticas tanto de origen técnico como de gestión a los proyectos analíticos representados en los cuatro experimentos, resulta en una reducción del 8,69% en el número de horas requeridas por proyecto (100 - 91,31).

El escenario optimista consiste en tomar, para cada etapa del ciclo de vida, el menor valor de los índices base 100 de la Tabla 19 que haya ocurrido en uno de los cuatro experimentos. Se resume el impacto de este escenario en la Tabla 23, reflejado en la columna ***w * DELTA***.

Tabla 23. Escenario optimista.

	Factor de Ponderación (*w*)	*BASELINE*	*DELTA*	*w * BASELINE*	***w * DELTA***
Comunicación	0,1512	100	87	15,12	13,16
Planeación	0,1024	100	104	10,24	10,65
Modelado	0,2495	100	90	24,95	22,46
Construcción	0,4287	100	87	42,87	37,29
Despliegue	0,0681	100	67	6,81	4,56
TOTAL	1			100	88,13

Fuente: elaboración propia.

En este escenario optimista, el impacto de aplicar prácticas tanto de origen técnico como de gestión a los proyectos analíticos representados en los cuatro experimentos, resulta en una reducción del 11,87% en el número de horas requeridas por proyecto (100 - 88,13).

Habiendo efectuado el análisis y la discusión de los resultados alcanzados, se completa el momento táctico de la metodología de intervención profesional propuesta y, con este último, también se da por finalizada la aplicación del marco metodológico propuesto por Matus (1998).

5. Conclusiones

5.1. Objetivos alcanzados

En pos de determinar el nivel de cumplimiento del objetivo general de este trabajo, se evalúa a continuación si se han alcanzado cada uno de los objetivos específicos detallados en la sección 1.4.

El primer objetivo específico, describir el contexto, las oportunidades y las motivaciones del trabajo, ha sido cubierto en las secciones 1.1 y 1.2. En cuanto al contexto, se estableció que este trabajo tiene lugar en una compañía multinacional de software y, más precisamente, en sus proyectos analíticos. En cuanto a oportunidades y motivaciones, se reunieron una serie de estudios cuyos resultados resaltan la importancia actual de estos proyectos y la de tecnologías de Transformación Digital como AI.

El segundo objetivo específico, desarrollar el marco conceptual de Ingeniería de Software Lean, fue tratado en la sección 2.1 de este trabajo. En la misma, se describió la historia de Lean así como sus principios y prácticas, incluyendo el estado del arte de tres de ellas: Gestión de la Configuración de Software, Integración Continua y Entrega Continua.

El tercer objetivo específico, establecer la fundamentación de las metodologías ágiles, fue alcanzado en la sección 2.2. En ella, partiendo del manifiesto ágil, se enumeraron los

valores de estas metodologías y se desarrollaron los componentes de uno de los marcos de trabajo ágiles: Scrum. Luego, se relacionó este tipo de metodología con aquéllas tradicionales y con la disciplina de Ingeniería de Software Lean.

El cuarto objetivo específico, analizar las características y las problemáticas de los sistemas de aprendizaje automático y el ciclo de vida de procesos analíticos, fue cubierto en la sección 2.3 de este trabajo. En esta última, se definieron, categorizaron y caracterizaron los sistemas de aprendizaje automático para, posteriormente, describir las principales problemáticas que los distinguen de otros sistemas. Por último, se desarrolló un modelo de ciclo de vida para procesos analíticos.

El quinto y último objetivo específico, aplicar prácticas Lean y de gestión de proyectos en la compañía bajo estudio, fue trabajado en las secciones 4.1 y 4.2. En ellas, primero se diseñaron los experimentos que iban a permitir aplicar estas prácticas en proyectos analíticos y se determinaron las métricas a tomar siguiendo GQM, para luego ejecutar los experimentos y registrar sus resultados.

Finalmente, habiendo analizado el cumplimiento de los objetivos específicos de este trabajo, se procede a establecer si el objetivo general del mismo ha sido alcanzado. Para determinar el impacto sobre la rentabilidad de la compañía multinacional al optimizar proyectos en el lifecycle de sus procesos analíticos, se analizaron los resultados de cada uno de los experimentos en la sección 4.3 y se sintetizaron todos ellos en escenarios desarrollados en la sección 4.4. Partiendo de lo expuesto en esta última, se espera que la aplicación simultánea de las distintas prácticas ágiles y Lean estudiadas genere, en un escenario pesimista, uno base y uno optimista, una reducción del 5,32%, 8,69% y 11,87% respectivamente en el número de horas requeridas por proyecto. Esto se refleja en la Tabla 24.

Tabla 24. Resultados esperados por escenario.

	Escenario pesimista	Escenario base	Escenario optimista
Reducción de horas por proyecto	5,32%	8,69%	11,87%

Fuente: elaboración propia.

Ahora bien, resulta menester retomar lo establecido en el alcance de este trabajo (ver sección 1.4) para determinar el impacto sobre la rentabilidad. Considerando ceteris paribus las demás variables, una reducción en la cantidad de horas requeridas por proyecto conlleva un menor costo que, consecuentemente, incrementa la rentabilidad esperada en la misma cuantía. De esta forma, se da respuesta al objetivo general de este trabajo.

5.2. Limitaciones y estudios futuros

Si bien se ha determinado el impacto que tendría optimizar proyectos analíticos en la rentabilidad de la compañía bajo estudio, existen limitaciones en este trabajo que dan lugar a plantear estudios futuros a ser realizados.

En primer lugar, ninguno de los cuatro experimentos ejecutados ha reunido en su ejecución simultáneamente todas las prácticas de Ingeniería de Software Lean y de gestión ágil de proyectos. Debido a las características propias de cada experimento, sólo algunas de las prácticas mencionadas resultaron aplicables. Esta primera limitación resulta difícil de atender, principalmente ya que, en la práctica, pocos proyectos analíticos reúnen las características necesarias para aplicar todas las técnicas estudiadas al mismo tiempo. Una alternativa sería, en trabajos futuros, diseñar una prueba piloto que permita esta aplicación simultánea. Sin embargo, este tipo de pruebas no siempre se asemeja a proyectos reales y, precisamente, por eso en este trabajo se priorizaron experimentos plausibles de ocurrir en la realidad, a pesar de que ellos no permitieron aplicar todas las prácticas ágiles y Lean a la vez.

De la primera limitación mencionada, surge la segunda: los criterios utilizados para la definición de los distintos escenarios poseen un impacto directo en los resultados del trabajo. Si bien no se ha efectuado un estudio exhaustivo de la objetividad de estos criterios, el hecho de que existan tres escenarios permite evaluar los resultados de este trabajo bajo diversos enfoques. Incluso bajo condiciones pesimistas, los resultados son positivos. Por ende, la importancia relativa de esta limitación disminuye. A pesar de ello, en trabajos futuros, podría estudiarse en mayor profundidad la objetividad de los criterios empleados para la definición de escenarios, así como también podrían explorarse nuevos escenarios posibles.

La tercera limitación hallada a este trabajo es el supuesto ceteris paribus a la hora de determinar el impacto en la rentabilidad. No se ha evaluado si la implementación de prácticas ágiles y Lean a proyectos analíticos tiene efectos sobre los ingresos presentes y futuros generados por estos proyectos. Se propone este estudio como parte de los trabajos futuros. Sin embargo, al tratarse de prácticas centradas en incrementar la eficiencia operativa de este tipo de proyectos, resulta esperable que las mismas no afecten negativamente su calidad ni las demás variables de costos. De hecho, es probable que, por ejemplo, al aplicar Integración y Entrega Continuas, la calidad de los proyectos aumente, representando incluso una mejora en ingresos reales y potenciales.

La cuarta y última limitación radica en que se ha excluido, del alcance de este trabajo, a la implementación de las prácticas estudiadas (ver sección 1.4). Resulta necesario entonces, en trabajos futuros, detallar cuáles serían los pasos a seguir para institucionalizar la aplicación de prácticas ágiles y Lean en todos los proyectos analíticos de la subsidiaria argentina de la compañía multinacional. Es decir, realizar un plan, un cronograma y un presupuesto para que estas prácticas sean un estándar. Con respecto al presupuesto, se deberían evaluar los costos

de capacitación, o más bien la inversión en capital humano, que debería efectuarse para permitir un exitoso proceso de implementación.

5.3. Contribuciones del trabajo

Se dividen las contribuciones del trabajo en técnicas, de gestión y de negocios.

Desde un punto de vista técnico, este trabajo ha permitido probar la viabilidad de prácticas de Ingeniería de Software Lean en proyectos analíticos. Esto se logró al aplicar Gestión de la Configuración de Software, Integración y Entrega Continuas en experimentos referidos a sistemas de aprendizaje automático. La contribución en esta dimensión ha sido demostrar que dichas prácticas no sólo son aplicables, sino también que su implementación da respuesta a determinadas problemáticas específicas de este tipo de sistemas.

Considerando una perspectiva de gestión, este trabajo ha presentado dos aportes principales. Por un lado, la aplicación de un marco de trabajo ágil para gestionar proyectos como lo es Scrum y la puesta en práctica de sus principales componentes: roles, eventos y artefactos. Por otro lado, la evaluación cuantitativa del efecto que tanto prácticas de ingeniería como de gestión han tenido en el esfuerzo asociado a cada uno de los cuatro experimentos ejecutados.

Finalmente, desde un punto de vista de negocios, este trabajo ha permitido determinar el impacto en costos, y su traslado a rentabilidad, que el conjunto de prácticas de ingeniería y de gestión tiene sobre la ejecución de proyectos en del lifecycle de procesos analíticos de la compañía. Esto da comienzo a un proceso de mejora continua basado en un marco metodológico multinivel, en métricas objetivas y en resultados de negocio. Se sugiere desarrollar un plan táctico que permita institucionalizar las prácticas desarrolladas en este trabajo ya que, incluso en un escenario pesimista, la reducción en costos es significativa y sólo requiere que el capital humano posea las competencias necesarias para adoptarlas.

6. Bibliografía

Amershi, S.; et al. (2019). *"Software Engineering for Machine Learning: A Case Study"*. IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Montreal, QC, Canadá, 2019, pp. 291-300, doi: 10.1109/ICSE-SEIP.2019.00042.

Basili, V. R., Caldiera, G., & Rombach, D. H. (1994). The Goal Question Metric Approach. Maryland-Kaiserslautern, United States of America-Germany. Retrieved July 16, 2020, from http://www.cs.umd.edu/~mvz/handouts/gqm.pdf

Beck, K. (1999). Embracing change with extreme programming. *Computer, 32*(10), 70-77. doi:10.1109/2.796139

Beck, K., Grenning, J., Martin, R. C., Beedle, M., Highsmith, J., Mellor, S., . . . Marick, B. (2001). Manifesto for Agile Software Development. Retrieved from Agile Alliance: http://agilemanifesto.org/

Bellows, B. (2019, September 23). *Reflections on the Fabric of the Toyota Production System*. Retrieved from The W. Edwards Deming Institute: https://blog.deming.org/2019/09/reflections-on-the-fabric-of-the-toyota-production-system/

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000, August). *CRISP-DM 1.0.* Retrieved from The Modeling Agency: https://www.the-modeling-agency.com/crisp-dm.pdf

Charette, R. N. (2003). Challenging the Fundamental Notions of Software Development. *IT Metrics and Productivity Inst.*

Costello, K. (2020, February 5). *Gartner Predicts the Future of AI Technologies.* Retrieved from Gartner: www.gartner.com/smarterwithgartner/gartner-predicts-the-future-of-ai-technologies.

Curry, D. (2019, October 3). *77% Organizations Aim To Deploy AI, Staff Skill Holds Adoption Back*. Retrieved from RTInsights: www.rtinsights.com/gartner-ai-adoption/

Cusumano, M., & Selby, R. (1997). *Microsoft Secrets* (New Ed ed.). New York: HarperCollins Business.

Forni, A. & Van der Meulen, R. (2017, April 24). *Gartner Survey Shows 42 Percent of CEOs Have Begun Digital Business Transformation.* Retrieved from Gartner: https://www.gartner.com/en/newsroom/press-releases/2017-04-24-gartner-survey-shows-42-percent-of-ceos-have-begun-digital-business-transformation

Fowler, M. (2008, June 26). *AgileVersusLean*. Retrieved from martinFowler: https://martinfowler.com/bliki/AgileVersusLean.html

Grolinger, K., Hayes, M., Higashino, W. A., L'Heureux, A., Allison, D. S., & Capretz, M. A. (2014). Challenges for MapReduce in Big Data. *2014 IEEE World Congress on Services* (pp. 182-189). Anchorage, AK: IEEE. doi:10.1109/SERVICES.2014.41

Hastie, T., Tibshirani, R., & Friedman, J. (2008). *The Elements of Statistical Learning.* Stanford, California, United States of America: Springer Series in Statistics.

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la Investigación.* Ciudad de México: McGraw-Hill.

Humble, J., & Farley, D. (2011). *Continuous Delivery.* Upper Saddle River, New Jersey, United States of America: Addison Wesley.

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unified Software Development Process.* Addison-Wesley.

Kohavi, R.; Rothleder, N. & Simoudis, E. (2002). *Emerging Trends in Business Analytics*. Communications of the ACM. 45 (8): 45–48. doi:10.1145/545151.545177.

L'heureux, A., Grolinger, K., Elyamany, H. F., & Capretz, M. A. (2017). Machine Learning With Big Data: Challenges and Approaches. *IEEE Access, 5*, 7776-7797.

Liker, J. (2004). *The Toyota Way.* New York: McGraw-Hill.

Lwakatare, L.; et. al. (2019). *A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation.* 10.1007/978-3-030-19034-7_14.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Hung Byers, A. (2011). *Big data: The next frontier for innovation, competition, and productivity.* New York: McKinsey Global Institute.

Matus, C. (1998). *Estrategia y Plan.* Ciudad de México: Siglo XXI Editores.

O'Sullivan, A. & Sheffrin, S. M. (2003). *Economics: Principles in Action*. Upper Saddle River, New Jersey 07458: Pearson Prentice Hall. p. 214. ISBN 0-13-063085-3.

Poppendieck, M., & Cusumano, M. A. (2012, September-October). Lean Software Development: A Tutorial. *IEEE Software, 29*(5), 26-32. doi:10.1109/MS.2012.107

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit.* Upper Saddle River, New Jersey, United States of America: Addison Wesley.

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

Project Management Institute. (2017). *Guía del PMBOK.* Newtown Square, Pennsylvania: Project Management Institute, Inc.

Ruparelia, N. B. (2010, May). Software Development Lifecycle Models. *ACM SIGSOFT Software Engineering Notes, 35*(3), 8-13.

Sato, D., Wider, A., & Windheuser, C. (2019, September 19). *Continuous Delivery for Machine Learning*. Retrieved from martinFowler.com: https://martinfowler.com/articles/cd4ml.html

Schwaber, K., & Sutherland, J. (2017, November). The Scrum Guide™. Retrieved from Scrum Guides: https://www.scrumguides.org/scrum-guide.html

Schwalbe, K. (2012). Managing a Project Using an Agile Approach and the PMBOK® Guide. In K. Schwalbe, *Information Technology Project Management.* Cengage Learning.

Sculley, D.; et al. (2015). *Hidden technical debt in machine learning systems.* In: Cortes, C.; Lawrence, N.D.; Lee, D.D.; Sugiyama, M.; Garnett, R. (eds.). *Advances in Neural Information Processing Systems 28*, pp. 2503–2511. Curran Associates, Inc.

Sliger, M. (2011). Agile project management with Scrum. *PMI® Global Congress.* Dallas, TX: Project Management Institute.

Software Program Managers Network. (1998). *Little Book of Configuration Management.* Arlington: Computer & Concepts Associates.

Sommerville, I. (2011). *Ingeniería de Software.* Naucalpan de Juárez, Estado de México: Pearson Education.

Usman, M., Soomro, T. R., & Brohi, M. N. (2014). Embedding project management into XP, Scrum and RUP. *European Scientific Journal, 10*(15).

Venture Scanner (2020, March 12). *AI 2019 Funding Achieved Banner Year.* Retrieved from Venture Scanner: www.venturescanner.com/2020/03/12/ai-2019-funding-achieved-banner-year/.

Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. *IV International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, (pp. 29-39).

Womack, J. P., Jones, D. T., & Roos, D. (1990). *The Machine that Changed the World.* New York: Rawson Associates.