UACM

Universidad Autónoma de la Ciudad de México

NADA HUMANO ME ES AJENO

COLEGIO DE CIENCIA Y TECNOLOGÍA

LICENCIATURA EN MODELACIÓN MATEMÁTICA

La Importancia de los Métodos Estadísticos en las Métricas de Velocidad y Defectos para Proyectos de Desarrollo de Software

T E S I S

QUE PARA OPTAR POR EL TÍTULO DE

LICENCIADOS EN MODELACIÓN MATEMÁTICA

P R E S E N T A N

YOSSI LARRY PÉREZ BARTOLÓN

JULIO CÉSAR RODRÍGUEZ VILLALOBOS

D I R E C T O R

M. EN C. RAFAEL MARTÍNEZ VEGA

Ciudad de México, julio de 2026.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

La presente obra y cada uno de sus elementos está protegido por la Ley Federal del Derecho de Autor; por la Ley de la Universidad Autónoma de la Ciudad de México, así como lo dispuesto por el Estatuto General Orgánico de la Universidad Autónoma de la Ciudad de México; del mismo modo por lo establecido en el Acuerdo por el cual se aprueba la Norma mediante la que se Modifican, Adicionan y Derogan Diversas Disposiciones del Estatuto Orgánico de la Universidad de la Ciudad de México, aprobado por el Consejo de Gobierno el 29 de enero de 2002, con el objeto de definir las atribuciones de las diferentes unidades que forman la estructura de la Universidad Autónoma de la Ciudad de México como organismo público autónomo y lo establecido en el Reglamento de Titulación de la Universidad Autónoma de la Ciudad de México.

Por lo que el uso de su contenido, así como cada una de las partes que lo integran y que están bajo la tutela de la Ley Federal de Derecho de Autor, obliga a quien haga uso de la presente obra a considerar que solo lo realizará si es para fines educativos, académicos, de investigación o informativos y se compromete a citar esta fuente, así como a su autor ó autores. Por lo tanto, queda prohibida su reproducción total o parcial y cualquier uso diferente a los ya mencionados, los cuales serán reclamados por el titular de los derechos y sancionados conforme a la legislación aplicable.

A mi madre, a mis hermanas, a mis compañeros y profesores, por su apoyo, enseñanzas e inspiración, que contribuyeron a mi crecimiento personal y profesional. Asimismo, a la Universidad Autónoma de la Ciudad de México, por brindarme la oportunidad de formarme y desarrollarme académicamente, haciendo posible la culminación de este importante logro.

YOSSI LARRY PÉREZ BARTOLÓN

A la Universidad Autónoma de la Ciudad de México, a mis profesores y quienes compartieron conmigo conocimientos, experiencias, tiempo y apoyo durante esta etapa.

JULIO CÉSAR RODRÍGUEZ VILLALOBOS

Índice general

Resumen General de la Tesis

En el desarrollo de software, una buena gestión de proyectos depende de medir y analizar aspectos clave como la velocidad con la que se ejecuta y la cantidad de errores o defectos* en el producto final. Estas mediciones ayudan a evaluar cómo está trabajando el equipo, encontrar posibles mejoras y asegurar que el software entregado sea de buena calidad. Para que estas métricas sean realmente útiles, es esencial usar Métodos Estadísticos*, que nos permiten analizar los datos de manera más precisa y objetiva.

Los Métodos Estadísticos* nos dan herramientas para entender mejor la variabilidad en los datos, encontrar tendencias y hacer predicciones basadas en lo que ha pasado antes en el proyecto. Esto no solo ayuda a tomar mejores decisiones, sino que también reduce riesgos*, como retrasos o problemas con el producto.

La velocidad de ejecución, que normalmente se mide con el número de tareas completadas en cada ciclo de trabajo (o Sprint*), es un indicador importante, especialmente en metodologías ágiles como SCRUM*. Por otro lado, los defectos* o errores en el software son un desafío para su calidad, ya que pueden causar retrasos y aumentar costos si no se gestionan bien. Usar el análisis estadístico para estas métricas nos permite detectar patrones, mejorar el rendimiento del equipo y anticipar problemas antes de que se presenten.

Este trabajo busca mostrar cómo el uso de Métodos Estadísticos* puede mejorar la manera en que se gestionan estas métricas en proyectos de software, logrando que los procesos sean más eficientes y los resultados más predecibles.

Capítulo 1

Introducción

"Lo que no se define no se puede medir.
Lo que no se mide, no se puede mejorar.
Lo que no se mejora, siempre se degrada"
— *William Thomson Kelvin*

1.1. Contexto de la investigación

En el entorno altamente competitivo de la industria del desarrollo de software, la entrega en tiempo y forma de productos de alta calidad es esencial. Para lograrlo, las organizaciones emplean metodologías de desarrollo de software que permiten gestionar proyectos de manera estructurada y adaptable. Entre estas se encuentran las metodologías ágiles, las cuales se caracterizan por utilizar enfoques iterativos e incrementales que facilitan la adaptación a problemas complejos y la generación continua de valor. Por ejemplo, SCRUM* es una de las metodologías ágiles más utilizadas en el desarrollo de software y se define como un marco de trabajo ligero que ayuda a equipos y organizaciones a generar valor mediante soluciones adaptativas, empleando un enfoque iterativo e incremental para mejorar la predictibilidad y controlar el riesgo (Schwaber and Sutherland, 2020). En este contexto, las métricas juegan un papel fundamental, ya que permiten cuantificar aspectos específicos de los procesos, productos o actividades, convirtiendo la información en datos numéricos que pueden ser analizados para apoyar la toma de decisiones.

Dicho lo anterior, el presente trabajo se desarrolla en el contexto de una consultora dedicada al desarrollo de software, en la cual se llevan a cabo actividades relacionadas con la definición, implementación y análisis de métricas de desempeño. En este entorno, se consideran dos perfiles profesionales relevantes: un Analista de Métricas*, responsable del diseño y aplicación de métricas orientadas al seguimiento de proyectos, y un Ingeniero de Datos*, encargado del desarrollo de soluciones de software que permiten la captura, procesamiento y disponibilidad de la información necesaria para su evaluación.

A partir de este contexto, el análisis se centra particularmente en las métricas de velocidad y de defectos*[1], las cuales representan aspectos fundamentales para la medición del progreso y la calidad en proyectos de desarrollo de software. Estas métricas permiten evaluar de manera objetiva el desempeño y la eficiencia de los equipos de trabajo, así como analizar la evolución de los proyectos a lo largo del tiempo. Asimismo, su estudio ofrece un marco adecuado para la aplicación de los conocimientos adquiridos en la Licenciatura en Modelación Matemática, mediante el uso de Métodos Estadísticos* y herramientas computacionales que facilitan la obtención de información cuantitativa y objetiva para la toma de decisiones en proyectos de software.

1.2. Planteamiento del problema

La motivación de esta investigación surge a partir de la observación de que, en el desarrollo de software, aún no existe una comprensión plenamente consolidada sobre cómo los Métodos Estadísticos* pueden apoyar de manera efectiva la gestión y mejora de los proyectos. La literatura especializada señala que, aunque muchas organizaciones recopilan métricas relacionadas con la velocidad de entrega y la calidad del software, estas mediciones suelen realizarse sin un enfoque sistemático ni un análisis riguroso de los datos, lo que limita su utilidad para anticipar problemas y apoyar la toma de decisiones.

[1] En el desarrollo de software, la velocidad es una métrica que evalúa el desempeño del equipo a partir de la relación entre el Esfuerzo planeado* y el Esfuerzo devengado* durante un periodo determinado, mientras que los defectos* representan fallos o desviaciones que impiden que el software cumpla con los requisitos establecidos; ambas métricas permiten evaluar el progreso y la calidad del desarrollo.

En particular, Forsgren, Humble y Kim documentan, a partir del análisis empírico de múltiples organizaciones, que la recolección de métricas sin una adecuada interpretación estadística conduce con frecuencia a prácticas poco efectivas, trabajo adicional y dificultades para comprender el desempeño real de los procesos de desarrollo de software (Forsgren et al., 2018, Cap. 1–2). Por ello, distintos autores enfatizan la importancia de analizar las métricas desde una perspectiva estadística, no únicamente como mecanismos de medición aislados, sino como herramientas para comprender el comportamiento de los procesos y promover mejoras continuas.

1.3. Objetivos de la investigación

Los objetivos que tenemos para esta investigación son los siguientes:

- Analizar la importancia de los Métodos Estadísticos* en las métricas de velocidad y defectos* en proyectos de desarrollo de software.
- Evaluar cómo la aplicación de Métodos Estadísticos* puede contribuir a una mejor gestión de proyectos desde una perspectiva técnica.
- Identificar las mejores prácticas y desafíos en la aplicación de métricas de velocidad y defectos* en proyectos de desarrollo de software. Así como el cuidado de la rentabilidad de estos de manera predictiva.
- Proporcionar recomendaciones prácticas para la selección y aplicación de métricas y Métodos Estadísticos* en proyectos de desarrollo de software.

1.4. Justificación de la investigación

La importancia de esta investigación es por el hecho de tocar puntos claves que muchas veces se pasan por alto en la gestión de proyectos de software. Por un lado, pone sobre la mesa la importancia de equilibrar lo técnico y la calidad en cada decisión que se toma. No solo se trata de entregar rápido, sino hacerlo bien, y eso es algo que en el día a día del desarrollo a veces queda en segundo plano.

Además, este trabajo podría servir de guía a las organizaciones sobre cómo integrar métricas de velocidad y defectos* junto con Métodos Estadísticos* para lograr una gestión más sólida y centrada en resultados reales. Muchas empresas miden el tiempo y los errores por separado, pero pocas veces se analiza cómo estos factores pueden complementarse para mejorar el proyecto completo.

Para los profesionales y equipos de desarrollo, esta investigación también les ofrece un respaldo para pensar en lo técnico de forma más estratégica en su rutina. No se trata solo de programar bien, sino de tomar decisiones basadas en datos que optimicen todo el proceso.

Por último, esta investigación podría contribuir a establecer una diferencia clara en la forma en que se toman decisiones y se gestionan los recursos en proyectos de desarrollo de software, particularmente al comparar organizaciones que emplean métodos sistemáticos de análisis de datos frente a aquellas que basan sus decisiones principalmente en la experiencia o la intuición. Mejorar la toma de decisiones en este contexto no solo implica una asignación más eficiente de los recursos disponibles, sino también una mayor capacidad para alinear los resultados del proyecto con las expectativas y necesidades del cliente.

En un entorno caracterizado por su dinamismo y alta competitividad, la capacidad de tomar decisiones informadas a partir del análisis de métricas puede representar un factor diferenciador entre organizaciones con distintos niveles de madurez en sus procesos de desarrollo de software, influyendo directamente en su desempeño y en la sostenibilidad de sus proyectos.

1.5. Metodología de la investigación

La investigación se desarrolló bajo un enfoque mixto, el cual combina métodos cualitativos y cuantitativos con el propósito de obtener una comprensión integral del fenómeno estudiado. Este enfoque permite integrar la revisión de la literatura con el análisis de datos empíricos, fortaleciendo la consistencia del estudio al contrastar fundamentos teóricos con evidencia obtenida a partir de proyectos reales.

En particular, el estudio contempla la revisión de literatura especializada en la gestión de proyectos de desarrollo de software y en el uso

de métricas de desempeño, así como el análisis de casos de estudio correspondientes a proyectos reales, lo cual facilita el análisis de procesos complejos dentro de contextos organizacionales específicos. Asimismo, se aplican Métodos Estadísticos para el análisis de métricas relacionadas con la velocidad de desarrollo y la ocurrencia de defectos*, las cuales permiten evaluar el progreso y la calidad del software a lo largo de su ciclo de vida.

Este enfoque metodológico posibilita un análisis estructurado del comportamiento de dichas métricas y de su utilidad para apoyar la toma de decisiones en proyectos de desarrollo de software, proporcionando una base objetiva para la evaluación del desempeño de los equipos y de los procesos involucrados.

Los datos utilizados en esta investigación provienen de una base de datos institucional perteneciente a una organización del sector de desarrollo de software. Dicha información es de uso confidencial, por lo que fue proporcionada exclusivamente con fines académicos y de investigación. En consecuencia, los datos fueron tratados de manera agregada y anónima, evitando la divulgación de información sensible que pudiera permitir la identificación de la organización, de los proyectos o de las personas involucradas.

Asimismo, se aplicaron criterios de confidencialidad y resguardo de la información durante todo el proceso de análisis, garantizando que los resultados presentados se enfoquen únicamente en el comportamiento de las métricas y en los hallazgos metodológicos, sin revelar detalles operativos, estratégicos o comerciales de la organización. Este enfoque permite asegurar la validez del estudio, al mismo tiempo que se respetan los principios éticos y de confidencialidad asociados al uso de información organizacional.

1.6. Estructura de la tesis

La tesis se organiza en varios capítulos que permiten abordar de manera progresiva los distintos elementos de la investigación. En un primer momento, se presenta una revisión de la literatura relacionada con la gestión de proyectos de desarrollo de software, el uso de métricas y la aplicación de métodos estadísticos. Posteriormente, se desarrolla un aná-

lisis detallado de los métodos estadísticos y de las métricas de velocidad y defectos* empleadas en el estudio, así como el análisis de casos de proyectos reales.

Adicionalmente, se incluyen recomendaciones prácticas orientadas a la aplicación de dichas métricas en proyectos de desarrollo de software. Para efectos de análisis y comparación, la investigación distingue entre proyectos ágiles y proyectos clásicos, entendiendo por proyectos ágiles aquellos que se caracterizan por enfoques iterativos e incrementales, con entregas frecuentes y adaptación continua a los cambios, y por proyectos clásicos aquellos que siguen un enfoque secuencial, con fases claramente definidas y una planificación más rígida desde el inicio.

Esta distinción permite analizar el comportamiento de las métricas en contextos de gestión diferentes y facilita la comprensión de cómo los enfoques de desarrollo influyen en la planificación, ejecución y control de los proyectos de software.

Capítulo 2

Fundamentos Teóricos

"You can't manage what you don't measure."
— Peter Drucker

2.1. Desarrollo de software ágil y clásico

El desarrollo de software ágil y clásico son dos enfoques diferentes para la gestión de proyectos de desarrollo de software. A continuación, se explica en qué consiste cada uno:

2.1.1. Desarrollo de software ágil

El desarrollo de software ágil es como una filosofía, que pone en primer plano la colaboración, la flexibilidad y la entrega rápida de software que realmente funcione desde el inicio. Los principios ágiles se centran en una comunicación constante con el cliente, adaptándose rápido a cualquier cambio que surja y, sobre todo, manteniendo la flexibilidad en cada paso del proceso. La idea es no quedarse encajado en un plan rígido, sino moverse según las necesidades del proyecto y las expectativas del cliente.

La manera de trabajar en un Desarrollo Ágil* es bastante dinámica: se usan ciclos de desarrollo pequeños, iterativos e incrementales, lo que permite entregar versiones funcionales del software cada cierto tiempo. Esto no solo da la ventaja de recibir feedback temprano, sino que permite ajustar el rumbo sin esperar hasta el final, lo cual, en teoría, debería

hacer el proceso mucho más adaptado a lo que el cliente realmente necesita.

Entre las metodologías ágiles más populares están SCRUM*, Kanban, eXtreme Programming (XP) y Lean[1]. Cada una tiene su propio estilo y prácticas, pero todas siguen esa misma filosofía de cambio constante y entrega gradual. Y aquí es donde se ve un detalle fundamental: el trabajo en equipo. En el desarrollo de software ágil, la colaboración y la comunicación entre los miembros del equipo y con el cliente, son esenciales. Aquí, no se da cabida para que cada quién quiera hacer su parte; es un trabajo donde cada etapa se nutre del contacto y la retroalimentación, buscando que el producto final realmente responda a lo que se espera (Pressman and Maxim, 2020, Cap. 2).

2.1.2. Desarrollo de software clásico:

Por otro lado, el desarrollo clásico que también es conocido como "modelo en cascada" es un enfoque que sigue una secuencia de fases: primero el análisis, luego el diseño, después la implementación, las pruebas y, por último, el mantenimiento. Aquí, no se pasa a una nueva fase hasta que la anterior esté completamente terminada. Es como si cada fase fuera un bloque de construcción que debe colocarse antes de poner el siguiente.

Para que este enfoque funcione, se necesitan especificaciones detalladas desde el principio. Se planifica hasta el más mínimo detalle antes de escribir una línea de código, con el objetivo de tener una hoja de ruta clara. El problema viene cuando, una vez avanzado el proyecto, el cliente decide que quiere algo diferente. Cambiar las especificaciones a mitad del proceso suele ser complicado y, en muchos casos, caro. No es raro que los equipos se topen con problemas si los requisitos cambian a lo largo del proyecto, ya que este modelo no está diseñado para manejar cambios repentinos.

Otra característica de la metodología en cascada es que el cliente solo ve el producto terminado al final del proyecto. Esto puede ser un riesgo si, después de todo el trabajo, resulta que el producto no es lo que el cliente tenía en mente. Aquí es donde el enfoque ágil tiene una ventaja evidente,

[1]Scrum, Kanban, eXtreme Programming (XP) y Lean son enfoques ágiles de gestión y desarrollo de software que promueven la entrega continua de valor, la colaboración y la mejora constante de procesos.

ya que permite ajustar continuamente el proyecto a las necesidades y expectativas del cliente.

Hasta aquí, podemos ir diciendo que cada enfoque tiene su razón de ser: el Desarrollo Ágil* es ideal para proyectos donde se espera que los requisitos cambien y donde es vital adaptarse a como se avanza. En cambio, el enfoque clásico puede ser más adecuado para proyectos donde los requisitos son estables y bien definidos desde el inicio. La realidad es que la agilidad suena mucho más atractiva en un entorno de negocios dinámico, pero también tiene sus retos y requiere de un equipo bien coordinado y dispuesto a colaborar constantemente (Sommerville, 2016, Cap. 2).

2.2. Métricas en proyectos de desarrollo de software

Definición formal de una métrica de rendimiento:

Desde una perspectiva formal, una métrica de rendimiento puede definirse como una función que asigna un valor numérico a un proceso, equipo o proyecto en un periodo determinado, con el propósito de cuantificar su desempeño respecto a un objetivo o referencia:

$$m : \mathcal{X} \times \mathcal{T} \to \mathbb{R}$$

donde X representa el conjunto de procesos o proyectos analizados y $\mathcal{T}$ el intervalo de tiempo considerado. En el ámbito de la gestión, estas métricas suelen expresarse como relaciones entre resultados obtenidos y recursos utilizados, permitiendo evaluar la eficiencia o el grado de cumplimiento de los objetivos establecidos.

Definición en el ámbito empresarial:

En el contexto empresarial, una métrica de rendimiento es un indicador cuantitativo utilizado para evaluar de manera objetiva el desempeño de un proceso, equipo u organización en relación con metas previamente

definidas. Estas métricas permiten monitorear la eficiencia, productividad y calidad de las operaciones, y constituyen una base fundamental para el control, la mejora continua y la toma de decisiones gerenciales.

Diversos autores señalan que una métrica de rendimiento debe estar alineada con los objetivos estratégicos de la organización y proporcionar información relevante para evaluar qué tan eficazmente se están utilizando los recursos disponibles (Kaplan and Norton, 1996).

Medir y analizar métricas en el desarrollo de software es clave para que el proyecto llegue a buen puerto y cumpla con las expectativas. Aunque puede sonar muy técnico, el objetivo de usar métricas es bastante claro: se trata de tener una visión más aterrizada de cómo va el proyecto y poder tomar decisiones con fundamento, en lugar de ir a ciegas o basarse en suposiciones.

Propósito de usar Métricas:
El uso de métricas en proyectos de desarrollo de software responde a la necesidad de contar con mecanismos objetivos que permitan evaluar el desempeño del proyecto y apoyar la gestión a lo largo de su ciclo de vida. Diversos autores coinciden en que las métricas constituyen una herramienta fundamental para el control, la evaluación y la mejora de los procesos de desarrollo (McConnell, 2006, Cap. 1–3, pp. 3–70).

1. **Medir el progreso:** Las métricas permiten cuantificar el avance de un proyecto en relación con lo planificado, proporcionando una referencia objetiva para determinar si el desarrollo se mantiene dentro de los plazos establecidos o si es necesario realizar ajustes. La ausencia de métricas dificulta la detección temprana de desviaciones, lo que puede generar una falsa percepción de avance hasta etapas tardías del proyecto (Project Management Institute, 2021).

2. **Evaluar la calidad:** Las métricas también funcionan como indicadores del nivel de calidad del software, ya que permiten analizar aspectos como la cantidad de defectos, la estabilidad del producto y la efectividad de las actividades de verificación y validación. La literatura destaca que la medición sistemática de la calidad es esencial para comprender el comportamiento del software y para interpretar los resultados dentro del contexto específico del proyecto (ISO/IEC, 2011).

3. **Apoyar la toma de decisiones:** El uso de métricas proporciona información cuantitativa que respalda la toma de decisiones relacionadas con la asignación de recursos, el ajuste del alcance y la priorización de actividades. Contar con datos confiables reduce la dependencia de decisiones intuitivas y permite adoptar un enfoque más analítico y preventivo en la gestión del proyecto (Kaplan and Norton, 1996).

4. **Detectar problemas a tiempo:** Las métricas pueden actuar como una alerta temprana, ayudando a identificar cualquier problema antes de que crezca. Esto puede evitar que pequeñas complicaciones se conviertan en dolores de cabeza y, al final, en gastos adicionales que se pudieron haber evitado (McConnell, 2006, Cap. 1–3, pp. 3–70).

5. **Gestionar riesgos:** La identificación y análisis de métricas relevantes contribuyen a la gestión de riesgos* del proyecto, ya que permiten reconocer puntos críticos y áreas vulnerables del proceso de desarrollo. Al contar con información cuantitativa, los equipos pueden implementar acciones preventivas y correctivas de forma proactiva, reduciendo el impacto de eventos adversos (Project Management Institute, 2021).

Relevancia de usar Métricas:

El uso de métricas en proyectos de desarrollo de software es relevante debido a su capacidad para estructurar el seguimiento del proyecto, evaluar resultados y promover una gestión basada en datos. Su aplicación sistemática favorece la mejora continua, la alineación con los objetivos organizacionales y la transparencia en la comunicación del desempeño del proyecto:

1. **Mejora continua:** La medición constante del desempeño permite identificar áreas de oportunidad y establecer acciones orientadas a la mejora progresiva de los procesos de desarrollo, fortaleciendo la madurez organizacional (McConnell, 2006, Cap. 1–3, pp. 3–70).

2. **Alineación con objetivos del negocio:** Las métricas facilitan la vinculación entre los resultados del proyecto y los objetivos estratégicos de la organización, asegurando que el software desarro-

llado responda a las necesidades del negocio y de los clientes finales (Kaplan and Norton, 1996).

3. **Transparencia y comunicación:** El uso de métricas objetivas promueve una comunicación clara entre los equipos de trabajo y las partes interesadas, generando confianza y una comprensión compartida del estado real del proyecto (Project Management Institute, 2021).

4. **Optimización de recursos:** El análisis de métricas permite evaluar el uso de los recursos disponibles y realizar ajustes para mejorar su aprovechamiento, contribuyendo a una gestión más eficiente del esfuerzo y del tiempo del equipo (McConnell, 2006, Cap. 1–3, pp. 3–70).

5. **Evaluación de resultados:** Al cierre del proyecto, las métricas posibilitan una evaluación integral de los resultados obtenidos, facilitando el aprendizaje organizacional y la incorporación de buenas prácticas en proyectos futuros (Project Management Institute, 2021).

2.3. Métodos estadísticos para medir y evaluar la velocidad y los defectos

El uso de métodos estadísticos en proyectos de desarrollo de software implica la aplicación de conceptos clave que permiten obtener una comprensión más profunda y precisa de los datos recopilados. A continuación, se explican estos conceptos clave:

1. **Métricas de velocidad:** La velocidad en el Desarrollo Ágil* de software se refiere a la cantidad de trabajo (generalmente expresado en Story Points* de usuario) que un equipo de desarrollo puede completar en un período de tiempo determinado, como una iteración* o Sprint*. Los métodos estadísticos permiten analizar la distribución de la velocidad a lo largo del tiempo, identificar tendencias y pronosticar el rendimiento futuro del equipo.

2. **Métricas de defectos:** La métrica de defectos* se utiliza pa-

ra medir la calidad del software y la eficacia de las actividades de pruebas. Estas métricas incluyen la cantidad de defectos encontrados, su gravedad, el tiempo necesario para resolverlos y su tasa de reaparición. Los métodos estadísticos ayudan a analizar la variabilidad en la Tasa de defectos* y a identificar patrones de calidad.

3. **Distribuciones de probabilidad:** son especialmente útiles para modelar fenómenos aleatorios asociados al desarrollo de software, como el tiempo entre la aparición de defectos o el tiempo de resolución de incidentes. Estas distribuciones permiten describir el comportamiento probabilístico de dichas variables y estimar intervalos de tiempo esperados para la entrega de resultados, considerando la incertidumbre inherente al proceso. (Ross, 2014, Cap 4, pp. 139–210)

4. **Métodos de Monte Carlo:** Los métodos de simulación de Monte Carlo se emplean para analizar escenarios futuros posibles mediante la generación repetida de valores aleatorios a partir de distribuciones probabilísticas previamente definidas. En el contexto del desarrollo de software, estos métodos permiten simular distintos escenarios de velocidad de desarrollo y carga de defectos, utilizando datos históricos para incorporar la variabilidad y la incertidumbre del proceso. (Law, 2015, Cap. 1–2, pp. 1–70)

5. **Análisis de tendencias:** El análisis de tendencias consiste en el uso de técnicas estadísticas y de series de tiempo para identificar patrones de comportamiento a lo largo del tiempo. Este tipo de análisis permite determinar si la velocidad del equipo muestra una tendencia creciente, decreciente o estable, así como identificar cambios en la frecuencia o severidad de los defectos, lo cual resulta clave para la evaluación del desempeño del proyecto. (Chatfield, 2004, Cap. 1, pp. 1–22)

6. **Desviación estándar:** La desviación estándar es una medida estadística que cuantifica la variabilidad o dispersión de un conjunto de datos respecto a su Media*. En el análisis de métricas de velocidad y defectos, esta medida permite evaluar la consistencia del rendimiento del equipo. Una desviación estándar elevada indica una mayor variabilidad en los resultados, lo cual puede señalar

inestabilidad o falta de control en el proceso de desarrollo (Montgomery and Runger, 2018, Cap. 2, pp. 55–120).

7. **Control Estadístico de Procesos (CEP):** El Control Estadístico de Procesos* implica el uso de técnicas estadísticas para monitorear y controlar la estabilidad de un proceso a lo largo del tiempo. En el desarrollo de software, el CEP se aplica para identificar desviaciones significativas en métricas como la velocidad o la Tasa de defectos* respecto a valores esperados. Herramientas como los gráficos de control permiten distinguir entre variaciones normales del proceso y variaciones atribuibles a causas especiales. (Montgomery, 2019, Cap. 1, pp. 1–20)
8. **Evaluación de riesgos:** Los métodos estadísticos también se utilizan para la evaluación de riesgos* asociados al proceso de desarrollo de software. El análisis de la variabilidad en la velocidad y en la Tasa de defectos* permite identificar escenarios que podrían afectar el cumplimiento de plazos y la calidad del producto, facilitando la implementación de acciones preventivas y correctivas basadas en evidencia cuantitativa (Vose, 2008, Cap. 5, pp. 87–128).

En conjunto, estos métodos estadísticos permiten analizar tendencias, medir la variabilidad, modelar relaciones, controlar el proceso y evaluar la confiabilidad de las métricas de velocidad y defectos. Su aplicación sistemática constituye un elemento fundamental para una gestión eficaz y una toma de decisiones informada en proyectos de desarrollo de software (Montgomery, 2019, Cap. 1, pp. 1–20).

2.4. Estado del arte en la aplicación de métodos estadísticos

En la actualidad, la medición y el análisis de métricas en el desarrollo de software se consideran una práctica fundamental para el control y la mejora de los proyectos. La literatura especializada señala que, a medida que los enfoques iterativos y ágiles se han consolidado en la industria, ha aumentado la necesidad de aplicar métodos cuantitativos que permitan evaluar el progreso, la calidad y la estabilidad del proceso de desarrollo

de manera objetiva (McConnell, 2006, Cap. 1–3, pp. 3–70).

La aplicación de métodos estadísticos a métricas como la velocidad de desarrollo y la ocurrencia de defectos permite identificar patrones, analizar la variabilidad del proceso y anticipar posibles desviaciones respecto a lo planificado. Este enfoque proporciona una visión integral del comportamiento del proyecto, facilitando la detección temprana de problemas y el ajuste oportuno de las estrategias de desarrollo. No obstante, distintos autores advierten que el valor de las métricas depende en gran medida de su correcta interpretación y de su uso dentro del contexto específico del proyecto, ya que métricas mal comprendidas pueden conducir a decisiones contraproducentes (McConnell, 2006, Cap 3).

En este sentido, el uso de la estadística en el desarrollo de software no persigue eliminar la incertidumbre inherente a los proyectos, sino gestionarla de forma informada, permitiendo a los responsables del proyecto anticiparse y adaptarse a los cambios propios de entornos dinámicos y complejos.

2.4.1. ¿Qué significa la velocidad en el desarrollo de software?

En el contexto de las metodologías ágiles, la velocidad se define como una medida de la cantidad de trabajo que un equipo es capaz de completar durante un periodo determinado. Esta métrica se utiliza comúnmente como un indicador de la capacidad del equipo y de su desempeño a lo largo de iteraciones sucesivas. Sin embargo, la propia naturaleza iterativa del desarrollo de software implica que la velocidad está sujeta a variaciones asociadas a la complejidad del trabajo, cambios en el equipo y factores técnicos imprevistos (Schwaber and Sutherland, 2020).

Debido a esta variabilidad, la velocidad no debe interpretarse como un valor fijo ni como una predicción exacta del desempeño futuro. Su análisis requiere el uso de herramientas estadísticas básicas, como la Media* y la Desviación Estándar*, que permitan comprender las fluctuaciones observadas y distinguir entre variaciones normales del proceso y cambios significativos en el rendimiento del equipo (Montgomery, 2019, Cap. 1–3, pp. 1 -120). En este sentido, la velocidad funciona más como un insumo para el análisis y la planificación que como una garantía de resultados.

2.4.2. Los defectos en el desarrollo de software

Los defectos en el desarrollo de software corresponden a fallos o desviaciones que afectan el cumplimiento de los requisitos funcionales o no funcionales del producto. El análisis estadístico de los defectos permite no solo cuantificarlos, sino también comprender su distribución, severidad y causas subyacentes. Técnicas como el análisis de Pareto resultan especialmente útiles para identificar un conjunto reducido de causas responsables de la mayoría de los defectos observados, facilitando la priorización de acciones correctivas (Montgomery, 2019, Cap. 6).

Asimismo, el uso de herramientas de Control Estadístico de Procesos*, como los gráficos de control, permite monitorear la estabilidad del proceso de desarrollo y detectar desviaciones significativas en la Tasa de defectos*. Estas técnicas contribuyen a mantener el proceso bajo control estadístico y a promover una mejora continua basada en el aprendizaje derivado de los errores, más que en la simple corrección puntual de fallos (Montgomery, 2019, Cap. 1–3).

Capítulo 3

Preparación de los datos para aplicarles métodos estadísticos

"Statistics is the grammar of Science"
— Karl Pearson

Hasta ahora hemos explorado qué son las métricas de velocidad y defectos y por qué importan. Pero una cosa es hablar de teoría, y otra muy distinta es saber cuál es el paso a paso para llevarlo a la práctica. Implementar métodos estadísticos en un proyecto de desarrollo de software tiene que seguir una metodología para que funcione, porque si falla una cosa, falla todo. En este capítulo damos una introducción a la parte práctica y detallada de cómo aplicar estos métodos y métricas para que sean realmente útiles, no solo para tener datos, sino para entender el rendimiento del equipo de desarrollo.

Cuando se trata de usar estadísticas en proyectos de software, la implementación no es tan sencilla como aplicar una fórmula. Aquí, hay decisiones importantes que tomar en cada paso. Desde elegir qué enfoque estadístico tiene más sentido hasta justificar cada cálculo con la lógica matemática que hay detrás. Y no, no es que vayamos a resolver todos los problemas solo con estadística; pero con una buena implementación, se pueden obtener pequeñas piezas de información profundas y accionables.

El enfoque de este capítulo está en que el lector entienda a través de los pasos clave, y su debida justificación, el como se ponen en práctica las métricas para que, al final, se puedan tomar decisiones informadas y, en la medida de lo posible, objetivas. También, dejamos espacio para un ejemplo práctico, donde se verá cómo estos métodos se traducen en algo tangible.

3.1. Selección del enfoque estadístico apropiado

Elegir el enfoque estadístico adecuado para evaluar la velocidad y los defectos en un proyecto de desarrollo de software no es solo una cuestión de usar fórmulas al azar. Es como elegir la herramienta adecuada para arreglar algo: usar la incorrecta puede no solo hacer el trabajo mal, sino complicarlo aún más. Así que en esta sección, vamos a desglosar los principales enfoques estadísticos que pueden adaptarse a esta evaluación y analizar por qué vale la pena usarlos.

3.1.1. Estadística descriptiva

La estadística descriptiva constituye una de las principales herramientas para el análisis inicial de los datos, ya que permite describir y resumir un conjunto de observaciones mediante medidas numéricas representativas. Su objetivo es proporcionar una visión general del comportamiento de los datos, facilitando la comprensión de su tendencia central y su variabilidad antes de aplicar técnicas estadísticas más avanzadas (Montgomery and Runger, 2018, Cap. 2, pp. 55–120).

En el contexto de esta investigación, la Estadística Descriptiva* se utiliza para analizar métricas asociadas a la velocidad de desarrollo y a la ocurrencia de defectos, permitiendo observar patrones generales del desempeño del proyecto sin perder información relevante. Entre las medidas más utilizadas se encuentran la Media*, la Mediana*, la Varianza* y la Desviación Estándar*.

La *Media* aritmética* es una medida de tendencia central que representa el valor promedio de un conjunto de datos y se define matemáticamente como:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

donde x_i representa cada observación individual y n el número total de observaciones. En proyectos de desarrollo de software, la Media* puede emplearse para estimar el tiempo promedio de atención, la velocidad promedio del equipo o el número medio de defectos por iteración* (Montgomery and Runger, 2018, Cap. 2, pp. 55–120).

La *Mediana** corresponde al valor central de un conjunto de datos ordenados de menor a mayor y resulta particularmente útil cuando existen valores extremos que pueden distorsionar la Media*. Esta medida permite representar de forma más robusta el comportamiento típico del proceso cuando la distribución de los datos es asimétrica (Devore, 2016, Cap. 1 pp. 1–48).

La *varianza* es una medida de dispersión que cuantifica el grado de variabilidad de los datos respecto a la Media* y se define como:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Esta medida permite evaluar qué tan alejados se encuentran los valores individuales respecto al comportamiento promedio del proceso (Devore, 2016, Cap. 1 pp. 1–48).

Por su parte, la *Desviación Estándar** se obtiene como la raíz cuadrada de la Varianza* y se expresa como:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

La Desviación Estándar* proporciona una interpretación más intuitiva de la variabilidad, ya que se expresa en las mismas unidades que los datos originales. En el análisis de la velocidad y de los defectos, una Desviación Estándar* elevada indica una alta variabilidad en el desempeño del equipo, lo que puede señalar inconsistencias o falta de estabilidad en el proceso de desarrollo (Montgomery, 2019, Cap. 1, pp. 1–20).

El uso conjunto de estas medidas permite no solo identificar el comportamiento promedio de las métricas analizadas, sino también evaluar la consistencia del proceso, proporcionando una base cuantitativa sólida para la interpretación de resultados y la toma de decisiones en proyectos de desarrollo de software.

3.1.2. Estadística inferencial

Mientras que la Estadística Descriptiva se orienta a resumir un conjunto de observaciones mediante medidas de tendencia central y dispersión, la Estadística Inferencial se ocupa de modelar el comportamiento de una población a partir de una muestra finita de datos. Formalmente, este enfoque permite estimar parámetros poblacionales desconocidos y evaluar hipótesis sobre dichos parámetros, incorporando explícitamente la incertidumbre asociada al proceso de muestreo (Devore, 2016).

En el análisis de proyectos de desarrollo de software, las observaciones disponibles —como la velocidad por iteración* o el número de defectos por periodo— se consideran realizaciones de variables aleatorias definidas sobre un espacio probabilístico. A partir de estas realizaciones, es posible construir estimadores de parámetros de interés, tales como la Media* poblacional de la velocidad o la tasa promedio de defectos, así como intervalos de confianza que cuantifican el rango en el cual dichos parámetros podrían encontrarse con un nivel de confianza preestablecido (Montgomery and Runger, 2018).

Asimismo, la Estadística Inferencial permite formular y contrastar hipótesis estadísticas para evaluar cambios significativos en el comportamiento del proceso. Por ejemplo, puede analizarse si la Media* de la velocidad antes y después de una intervención presenta una diferencia estadísticamente significativa, o si la Tasa de defectos* se mantiene dentro de límites aceptables. Estas pruebas se fundamentan en distribuciones de probabilidad teóricas o aproximadas y en estadísticos de prueba derivados de la muestra (Devore, 2016).

Sin embargo, los datos provenientes de proyectos de software no siempre satisfacen los supuestos clásicos de normalidad, independencia o Varianza* constante, lo que puede afectar la validez de algunos modelos inferenciales. Por esta razón, los resultados obtenidos deben interpretarse como estimaciones sujetas a error y no como predicciones determinis-

tas del comportamiento futuro. En consecuencia, el análisis inferencial debe complementarse con la verificación de supuestos estadísticos y el análisis de la estabilidad del proceso (Montgomery, 2019).

3.2. Recolección y preparación de datos

La recolección y preparación de datos constituye la base del análisis cuantitativo del presente trabajo. Las métricas asociadas a la *velocidad* se obtienen a partir de los sistemas institucionales de gestión del trabajo (Jira*), desde donde se extraen variables relacionadas con el Esfuerzo planeado* y el Esfuerzo devengado* por elemento de trabajo. Por su parte, la información relativa a los *defectos* es registrada de manera manual por los equipos de trabajo en hojas de cálculo institucionales, las cuales concentran reportes agregados por fase del proceso y por iteración*.

Debido a restricciones de confidencialidad organizacional y políticas de seguridad de la información, los datos detallados de defectos no pueden ser presentados de forma directa en este documento. En consecuencia, para efectos del análisis académico, únicamente se utilizan parámetros estadísticos derivados de dichos registros (por ejemplo, rangos, medias y desviaciones estándar), los cuales se emplean como insumo para la simulación de escenarios mediante Método de Monte Carlo*. Toda la información utilizada se procesa de forma anonimizada y agregada, evitando cualquier elemento que permita identificar proyectos, usuarios, clientes o activos internos.

3.2.1. Origen y características generales de los datos

Los datos se obtienen mediante exportaciones controladas desde Jira* hacia hojas de cálculo en formato Excel. La base se organiza a nivel de *incidencia/elemento de trabajo* (por ejemplo, requerimientos o tareas), y contiene campos numéricos de esfuerzo y campos derivados para el cálculo de la velocidad. En el caso de defectos, por restricciones de seguridad, no se cuenta con una base completa de registros históricos; en su lugar, únicamente se dispone de parámetros estadísticos (p. ej., medias y desviaciones estándar) que describen distribuciones utilizadas para simulación con fines académicos.

3.2.2. Identificación de variables clave

Para la métrica de velocidad se consideran como variables principales el Esfuerzo planeado* (*estimación original*) y el Esfuerzo devengado* (*tiempo devengado*). En metodologías ágiles, la velocidad se define tradicionalmente como una medida de la cantidad de trabajo que un equipo es capaz de completar durante un intervalo fijo de tiempo, y se utiliza como un indicador de la capacidad del equipo para cumplir compromisos de entrega en iteraciones sucesivas (Schwaber and Sutherland, 2020).

Desde una perspectiva de ingeniería del software y gestión de proyectos, la comparación entre el esfuerzo estimado y el esfuerzo real constituye un indicador de eficiencia en la ejecución de las tareas, ya que permite evaluar el grado de desviación respecto a la planificación original (McConnell, 2006). En este trabajo, la velocidad se operacionaliza como una variable derivada que representa la relación entre el esfuerzo planificado y el esfuerzo consumido por incidencia:

Sea $\mathcal{I}$ el conjunto de incidencias observadas y sea $i \in \mathcal{I}$ un índice que representa una incidencia individual.

Se definen las siguientes variables:

- $E_{plan,i} \in \mathbb{R}^+$: esfuerzo planeado asociado a la incidencia i
- $E_{dev,i} \in \mathbb{R}^+$: esfuerzo devengado asociado a la incidencia i

Se define la variable de velocidad V_i como:

$$V_i = \frac{E_{plan,i}}{E_{dev,i}}$$

La variable V_i representa una medida del desempeño relativo del proceso, comparando la estimación inicial con el esfuerzo real observado.

donde E_i^{plan} corresponde a la estimación original y E_i^{dev} al tiempo devengado para la incidencia i. Esta formulación permite interpretar $V_i > 1$ como un desempeño por encima de lo planeado (menor consumo relativo de esfuerzo) y $V_i < 1$ como un consumo mayor al esfuerzo estimado, lo cual indica desviaciones negativas respecto al plan.

Esta definición es consistente con el uso de la velocidad como indicador de capacidad y desempeño operativo, y permite analizar el comporta-

miento del proceso desde una perspectiva cuantitativa compatible con los enfoques ágiles y con los principios clásicos de control del esfuerzo en proyectos de software.

3.2.3. Estructura de la base de datos: velocidad

El Cuadro 3.1 resume las variables empleadas para el análisis de velocidad, incluyendo su tipo y descripción. El Cuadro 3.2 muestra un extracto ilustrativo de registros.

Cuadro 3.1: Diccionario de datos para el análisis de velocidad

Variable	**Tipo**	**Descripción**
ID_Incidencia	Categórica (texto)	Identificador anonimizado de la incidencia o elemento de trabajo (por ejemplo, requerimiento o tarea).
Estimación_Original (E^{plan})	Numérica (entero/-real)	Esfuerzo planeado* originalmente para la incidencia, medido en horas.
Tiempo_Devengado (E^{dev})	Numérica (entero/-real)	Esfuerzo realmente consumido para completar la incidencia, medido en horas.
Velocidad (V)	Numérica (real)	Variable derivada definida como $V_i = E_i^{plan}/E_i^{dev}$.

Cuadro 3.2: Extracto ilustrativo de la base (velocidad)

ID_Incidencia	**Estimación original**	**Tiempo devengado**	**Velocidad**
Req-1	224	62	3.61
Req-2	24	8	3.00
Req-3	14	5	2.80
Req-4	68	26	2.62
Req-5	10	5	2.00
Req-8	44	23	1.91
Req-9	784	436	1.80
Req-11	660	386	1.71
Req-14	40	25	1.60

3.2.4. Limpieza y normalización de datos

Previo al análisis, se realiza limpieza para asegurar consistencia y validez de los datos:

- **Consistencia de tipos:** conversión explícita de variables numéricas y verificación de valores faltantes.

- **Reglas de integridad:** exclusión o revisión de incidencias con $E^{dev} = 0$ (indefinición de V), valores negativos o registros incompletos.

- **Tratamiento de valores atípicos:** identificación de registros con esfuerzos extraordinarios (p. ej., órdenes de magnitud superiores al comportamiento típico) para revisión y/o análisis separado.

Cuando se requiere comparar variables en escalas diferentes, se aplican transformaciones o normalizaciones (p. ej., estandarización) únicamente con fines analíticos, manteniendo siempre una versión original para trazabilidad.

3.2.5. Datos de defectos y esfuerzo: restricciones y simulación

Para el componente de defectos, por políticas de seguridad y confidencialidad, no se dispone del detalle completo de incidencias de defectos (p. ej., listados por fecha, severidad o componente). En su lugar, la organización proporcionó *parámetros estadísticos* (rangos y/o valores resumen) que permiten modelar distribuciones asociadas a densidades de defectos y esfuerzo por fase del ciclo de desarrollo. Con fines académicos, estos parámetros se utilizan para generar simulaciones del Método de Monte Carlo* y construir escenarios de defectos y esfuerzo esperados.

El Cuadro 3.3 resume los parámetros utilizados para las simulaciones por fase (Revisión de Producto, Análisis y Revisión, y Pruebas), de acuerdo con los rangos empleados en el modelo.

Cuadro 3.3: Parámetros utilizados para simulación de densidades de esfuerzo y defectos por fase

Fase	**Variable simulada**	**Rango base**	**Uso en el modelo**
RP (Revisión de Producto)	Densidad de esfuerzo	[0.020, 0.040]	Generación de n muestras (Normal) y cálculo de percentiles 2.5 % y 97.5 %.
AR (Análisis y Revisión)	Densidad de defectos	[0.00675, 0.0105]	Simulación de defectos esperados en AR mediante percentiles para un Intervalo de confianza* del 95 %.
PR (Pruebas)	Densidad de esfuerzo	[0.05, 0.3]	Simulación del esfuerzo asociado a la ejecución de pruebas utilizando percentiles al 95 %.
PR (Pruebas)	Densidad de defectos	[0.002, 0.005]	Simulación de defectos esperados en pruebas; se utiliza el valor absoluto para evitar resultados negativos.

Los rangos base presentados en el Cuadro 3.3 representan *densidades* de esfuerzo o de defectos, es decir, relaciones entre la cantidad de defectos o esfuerzo observado y una unidad de tamaño o carga de trabajo (por ejemplo, horas, puntos o unidades funcionales, según la configuración de cada proyecto). Este enfoque es consistente con los modelos de medición de calidad de software, en los cuales los defectos y el esfuerzo se analizan como tasas por unidad de tamaño para permitir comparaciones entre proyectos de distinta magnitud (Pressman and Maxim, 2020, Cap. 23, pp. 665–690) (Sommerville, 2016, Cap. 24, pp. 676–705). En este sentido, los valores no se interpretan como conteos absolutos, sino como tasas que permiten escalar las estimaciones en función del tamaño de la iteración* o del volumen de trabajo considerado.

Formalmente, la *Densidad de defectos** puede definirse como:

$$d_{\text{def}} = \frac{N_{\text{def}}}{X},$$

donde N_{def} representa el número de defectos observados en una fase del proceso y X corresponde a la medida de tamaño o esfuerzo asociada (por ejemplo, horas de trabajo, puntos de historia o puntos de función). Esta razón expresa el número esperado de defectos por unidad de carga de

trabajo, métrica ampliamente utilizada para evaluar la calidad relativa del software (Pressman and Maxim, 2020, Cap. 23, pp. 665-690).

De manera análoga, la *Densidad de esfuerzo*[*] se define como:

$$d_{\text{esf}} = \frac{E_{\text{fase}}}{E_{\text{total}}},$$

donde E_{fase} es el Esfuerzo invertido[*] en una fase específica del proceso (por ejemplo, revisión de producto o ejecución de pruebas) y E_{total} es el esfuerzo total del proyecto o de la iteración[*]. Esta densidad representa la fracción del esfuerzo total que típicamente se destina a una fase determinada, concepto utilizado en modelos de distribución del esfuerzo por etapas del ciclo de vida del software (Sommerville, 2016, Cap. 24, pp. 676–705).

Estas relaciones permiten que el modelo proyecte valores mínimos y máximos esperados multiplicando dichas densidades por la magnitud de la carga de trabajo de cada iteración[*], de acuerdo con:

$$\hat{N}\text{def}, i = d\text{def} \cdot X_i, \qquad \hat{E}\text{fase}, i = d\text{esf} \cdot E_{\text{total},i},$$

donde X_i y $E_{\text{total},i}$ representan el tamaño y el esfuerzo total de la iteración[*] i, respectivamente.

Este enfoque facilita la adaptación del modelo a distintos tamaños de proyecto, ya que las estimaciones se ajustan proporcionalmente al volumen de trabajo, manteniendo la coherencia con los patrones históricos observados. De esta forma, la simulación no genera valores arbitrarios, sino resultados consistentes con las relaciones empíricas entre esfuerzo, tamaño y defectos registradas por la organización.

3.2.6. Flujo de obtención, procesamiento y uso de los datos

Los datos utilizados en esta investigación provienen de sistemas institucionales de gestión del trabajo empleados en proyectos reales de desarrollo de software, particularmente herramientas de seguimiento de

incidencias como Jira. En este contexto, las incidencias representan unidades de trabajo asociadas a actividades ejecutadas por distintos roles del equipo, tales como desarrolladores, testers, analistas y líderes de proyecto, dentro del ciclo de vida del desarrollo de software.

Cada registro en estos sistemas corresponde a una actividad específica del proceso, como la implementación de funcionalidades, la atención de defectos o la ejecución de pruebas, lo que permite capturar de manera estructurada el esfuerzo invertido y el avance real del proyecto.

El proceso de obtención y utilización de los datos se estructura en las siguientes etapas:

En primer lugar, se realiza la extracción de los datos mediante exportaciones controladas desde los sistemas institucionales hacia archivos en formato estructurado (Excel). Estos registros contienen información a nivel de incidencia, incluyendo variables relevantes como el esfuerzo planeado y el esfuerzo devengado, las cuales reflejan la estimación inicial y el esfuerzo real invertido por los distintos roles involucrados en la ejecución de las actividades.

Posteriormente, los datos son sometidos a un proceso de depuración y validación utilizando herramientas computacionales en el lenguaje R que se explicarán más adelante. Este proceso incluye la verificación de consistencia de tipos, el tratamiento de valores faltantes, la identificación de valores atípicos y la normalización de variables, con el objetivo de garantizar la calidad y confiabilidad de la información utilizada en el análisis.

Una vez procesados, los datos se emplean para la construcción de variables derivadas, como la métrica de velocidad, definida como la razón entre el esfuerzo planeado y el esfuerzo devengado. Estas variables constituyen el insumo para la aplicación de métodos estadísticos, incluyendo análisis descriptivo, control estadístico de procesos y simulación de Monte Carlo, lo que permite modelar el comportamiento del proceso de desarrollo bajo condiciones de incertidumbre.

Finalmente, los resultados obtenidos son analizados con el propósito de evaluar el comportamiento del proceso de desarrollo, identificar patrones de variabilidad y proponer mejoras en la gestión del proyecto, considerando la interacción entre los distintos roles y las actividades ejecutadas

en cada fase del ciclo de vida del software.

Debido a restricciones de confidencialidad organizacional, no es posible presentar la estructura completa de las bases de datos ni los registros individuales, sin embargo, se muestra más adelante una estructura de datos con otro identificador pero en esencia cuenta con los mismo datos numéricos.

Capítulo 4

La Velocidad en el Desarrollo de Software

"Without data, you're just another person with an opinion."
— *W. Edwards Deming*

Importancia de la métrica de velocidad

La métrica de velocidad constituye un indicador fundamental en el seguimiento del desempeño de los equipos de desarrollo, particularmente en entornos ágiles como SCRUM*. Tradicionalmente, la velocidad se interpreta como la cantidad de trabajo que un equipo es capaz de completar durante una iteración*, y se utiliza como una medida empírica de la capacidad operativa del equipo (Schwaber and Sutherland, 2020). Desde el punto de vista de la ingeniería de software, la comparación entre esfuerzo estimado y esfuerzo real permite además evaluar la eficiencia del proceso y las desviaciones respecto a la planeación original (McConnell, 2006).

En el contexto de esta investigación, la velocidad se modela como una relación entre Esfuerzo planeado* y Esfuerzo devengado* por incidencia, lo que permite analizar el desempeño de ejecución de cada iteración* con base en datos reales de operación. A partir de esta definición, la métrica de velocidad aporta los siguientes beneficios:

- **Estimación y planificación:** La velocidad histórica permite es-

timar de forma más realista la cantidad de trabajo que puede abordarse en cada iteración*. En metodologías ágiles, esta información se emplea para seleccionar el conjunto de elementos del backlog* que el equipo puede comprometerse a completar, reduciendo el riesgo de sobreasignación de trabajo (Schwaber and Sutherland, 2020).

- **Predicción de entregas:** Al analizar la velocidad promedio y su variabilidad, es posible proyectar el número de iteraciones requeridas para completar un conjunto de funcionalidades. Este enfoque permite realizar proyecciones basadas en desempeño empírico en lugar de estimaciones teóricas, lo cual mejora la confiabilidad de los planes de entrega (McConnell, 2006).

- **Mejora continua del proceso:** El seguimiento sistemático de la velocidad facilita la identificación de desviaciones, cuellos de botella y cambios en el desempeño del equipo. Estas variaciones pueden analizarse mediante técnicas estadísticas para evaluar la estabilidad del proceso y apoyar acciones de mejora continua (Pressman and Maxim, 2020).

- **Priorización informada del trabajo:** Conocer la capacidad real del equipo permite priorizar los elementos del backlog* de acuerdo con el valor del negocio y la carga de trabajo disponible. Esto contribuye a maximizar el uso eficiente del tiempo de desarrollo dentro de cada iteración* (Schwaber and Sutherland, 2020).

- **Evaluación de la capacidad del equipo:** La velocidad refleja el ritmo de trabajo alcanzable bajo condiciones reales de operación, lo que permite ajustar la planeación futura sin recurrir a supuestos optimistas o teóricos. Esta información resulta clave para la toma de decisiones gerenciales basadas en evidencia cuantitativa (McConnell, 2006).

En síntesis, la métrica de velocidad no solo permite monitorear el avance del proyecto, sino que también funciona como un mecanismo de control del desempeño del proceso de desarrollo. Su análisis sistemático proporciona insumos para la planeación, la predicción, la mejora continua y la toma de decisiones, elementos esenciales para la gestión efectiva de proyectos de software en contextos ágiles y tradicionales.

4.1. Velocidad de Desarrollo de Software mediante Control Estadístico de Procesos

4.1.1. Problemática y Contexto Operativo

En la industria del desarrollo de software, la planificación de tiempos constituye un elemento crítico para asegurar la entrega puntual y la estabilidad operativa de los proyectos. No obstante, múltiples estudios en ingeniería de software señalan que las estimaciones iniciales suelen presentar desviaciones significativas respecto al esfuerzo real, debido a factores como la complejidad técnica, la incertidumbre en los requerimientos, la variabilidad en la experiencia del equipo y la presencia de retrabajo (McConnell, 2006).

En el contexto de la organización analizada en este estudio, se ha identificado un retraso promedio aproximado del 35 % respecto a los tiempos inicialmente planificados, lo cual ha generado afectaciones en la programación de entregas, reasignación frecuente de recursos y dificultades para mantener compromisos con las áreas solicitantes. Esta situación evidenció la necesidad de contar con métricas objetivas que permitan evaluar de forma sistemática la relación entre el Esfuerzo planeado* y el Esfuerzo devengado*, así como el comportamiento de los defectos a lo largo del proceso.

Esta problemática se traduce en desviaciones recurrentes entre los valores estimados y los valores reales consumidos, lo que incrementa la incertidumbre en la gestión del proyecto y limita la capacidad de anticipar riesgos* operativos. Ante este escenario, el análisis estadístico de las métricas de velocidad y defectos, apoyado en técnicas de simulación y control cuantitativo del proceso, permite diagnosticar el desempeño real del sistema de trabajo, identificar patrones de variabilidad y evaluar el impacto de acciones de mejora como ajustes en los procesos de estimación, capacitación del personal y estandarización en el registro de la información.

4.1.2. Análisis Exploratorio y Validación Estadística

Sea V una variable aleatoria que representa la velocidad del proceso de desarrollo, definida a partir de las observaciones $\{V_i\}_{i\in\mathcal{I}}$.

Se asume que V sigue una distribución de probabilidad con parámetros θ, estimados a partir de los datos observados.

Fundamento matemático y operacional

Dado que la métrica de velocidad V se define como la razón entre el Esfuerzo planeado* y el Esfuerzo devengado* por incidencia, su análisis estadístico permite evaluar la estabilidad del proceso de estimación y ejecución del trabajo. Para poder aplicar técnicas inferenciales como intervalos de confianza, pruebas de control estadístico y simulación probabilística, es necesario establecer un modelo de distribución adecuado para la variable V.

En particular, se modela la variable V como una variable aleatoria con distribución normal, es decir:

$$V \sim \mathcal{N}(\mu, \sigma^2)$$

Esta aproximación se justifica bajo el supuesto de que la variable V resulta de la agregación de múltiples factores aleatorios de pequeña magnitud, tales como variaciones en la complejidad de las tareas, diferencias en la experiencia de los roles involucrados y condiciones operativas del entorno.

Teorema Central del Límite

Teorema (Central del Límite de Lindeberg–Lévy). Sea $\{X_n\}_{n\geq 1}$ una sucesión de variables aleatorias independientes e idénticamente distribuidas tales que

$$\mathbb{E}[X_1] = \mu, \qquad \mathrm{Var}(X_1) = \sigma^2 < \infty.$$

Defínase la suma parcial

$$S_n = X_1 + X_2 + \cdots + X_n.$$

Entonces,

$$\frac{S_n - n\mu}{\sigma\sqrt{n}} \xrightarrow{d} \mathcal{N}(0,1), \qquad n \to \infty.$$

Demostración. Sea

$$Y_k = \frac{X_k - \mu}{\sigma}.$$

Entonces $\{Y_k\}$ es una sucesión de variables aleatorias independientes e idénticamente distribuidas con

$$\mathbb{E}[Y_k] = 0, \qquad \mathrm{Var}(Y_k) = 1.$$

Definimos

$$T_n = \frac{1}{\sqrt{n}} \sum_{k=1}^{n} Y_k.$$

Obsérvese que

$$T_n = \frac{S_n - n\mu}{\sigma\sqrt{n}},$$

por lo que basta demostrar que $T_n \xrightarrow{d} \mathcal{N}(0,1)$.

Sea $\varphi(t) = \mathbb{E}[e^{itY_1}]$ la función característica de Y_1. Dado que $\mathbb{E}[Y_1] = 0$ y $\mathrm{Var}(Y_1) = 1$, se tiene la expansión:

$$\varphi(t) = 1 - \frac{t^2}{2} + o(t^2), \qquad t \to 0.$$

La función característica de T_n es

$$\varphi_n(t) = \mathbb{E}[e^{itT_n}] = \left(\varphi\left(\frac{t}{\sqrt{n}}\right)\right)^n.$$

Utilizando la expansión anterior,

$$\varphi\left(\frac{t}{\sqrt{n}}\right) = 1 - \frac{t^2}{2n} + o\left(\frac{1}{n}\right).$$

Por lo tanto,

$$\varphi_n(t) = \left(1 - \frac{t^2}{2n} + o\left(\frac{1}{n}\right)\right)^n.$$

Tomando límite cuando $n \to \infty$, se obtiene

$$\varphi_n(t) \to e^{-t^2/2},$$

que es la función característica de la distribución normal estándar $\mathcal{N}(0,1)$.

Por el teorema de continuidad de Lévy, se concluye que

$$T_n \xrightarrow{d} \mathcal{N}(0,1).$$

En consecuencia,

$$\frac{S_n - n\mu}{\sigma\sqrt{n}} \xrightarrow{d} \mathcal{N}(0,1).$$

□

Observación

El resultado anterior permite justificar el uso de modelos normales en el análisis de variables que resultan de la agregación de múltiples factores aleatorios, lo cual es consistente con el enfoque adoptado en este trabajo.

Bajo estas condiciones, el Teorema Central del Límite establece que la combinación de variables aleatorias independientes tiende a una distribución aproximadamente normal.

En este estudio se parte del supuesto de que la variable V puede aproximarse mediante una Distribución normal*, lo cual es consistente con el hecho de que su valor resulta de la combinación de múltiples factores aleatorios de pequeña magnitud, tales como variaciones técnicas, diferencias en la complejidad de las tareas, experiencia del personal y condiciones operativas cambiantes. Bajo estas condiciones, el Teorema Central del Límite* establece que la combinación de efectos aleatorios tiende a producir distribuciones aproximadamente normales (Montgomery, 2019, Cap. 4, pp. 152–210).

Con el fin de evaluar estadísticamente este supuesto, se plantea la siguiente prueba de hipótesis:

$$H_0 : V \sim \mathcal{N}(\mu, \sigma^2)$$

$$H_1 : V \not\sim \mathcal{N}(\mu, \sigma^2)$$

donde μ y σ^2 representan la Media* y la Varianza* poblacional del proceso de velocidad bajo condiciones estables de operación. La hipótesis nula H_0 establece que las observaciones de velocidad pueden modelarse mediante una Distribución normal*, mientras que la hipótesis alternativa H_1 indica que la distribución presenta desviaciones significativas respecto al modelo normal.

Aceptar H_0 no implica que la distribución sea perfectamente normal, sino que no existe evidencia estadística suficiente para rechazar el modelo normal con el nivel de significancia seleccionado. Esto permite aplicar métodos paramétricos para el análisis de variabilidad, construcción de intervalos de confianza y evaluación del desempeño del proceso. En contraste, el rechazo de H_0 sugiere la presencia de patrones no aleatorios o causas especiales de variación, lo cual indicaría inestabilidad en el proceso de estimación o ejecución y la necesidad de realizar análisis causales adicionales.

Desde una perspectiva operativa, esta prueba permite determinar si las variaciones en la velocidad observada pueden atribuirse al comportamiento normal del proceso o si existen desviaciones estructurales que afectan sistemáticamente el cumplimiento de los tiempos planificados, información fundamental para la toma de decisiones de mejora en el proceso de desarrollo.

Cálculo de parámetros y pruebas

La base de datos utilizada para el análisis está estructurada a nivel de incidencia individual e incluye las siguientes variables: identificador de la incidencia, Esfuerzo planeado* (E_i^{plan}), Esfuerzo devengado* (E_i^{dev}) y velocidad calculada como $V_i = E_i^{plan}/E_i^{dev}$. Esta estructura permite evaluar el desempeño del proceso de estimación y ejecución de manera granular, considerando cada elemento de trabajo como una observación independiente del proceso operativo.

Sea $\{V_1, V_2, \ldots, V_n\}$ el conjunto de velocidades observadas. Los parámetros de la distribución se estiman mediante la Media* y la Varianza* muestral:

$$\bar{V} = \frac{1}{n}\sum_{i=1}^{n} V_i, \qquad S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(V_i - \bar{V})^2.$$

La Media* $\bar{V}$ representa el nivel promedio de cumplimiento entre el Esfuerzo planeado* y el esfuerzo real, funcionando como un indicador global del desempeño del proceso. Por su parte, la Varianza* S^2 cuantifica la dispersión de las observaciones respecto a dicho promedio, permitiendo evaluar la estabilidad del proceso: valores elevados de Varianza* indican alta variabilidad en el cumplimiento de las estimaciones, lo cual sugiere un proceso menos predecible.

Para evaluar la plausibilidad del supuesto de normalidad de la variable V, se aplican pruebas de bondad de ajuste, entre ellas la Prueba de Shapiro–Wilk* y la prueba de Kolmogorov–Smirnov. Estas pruebas contrastan la hipótesis nula de que los datos provienen de una Distribución normal* contra la alternativa de no normalidad. Un valor-p elevado indica que no existe evidencia estadística suficiente para rechazar la hipótesis de normalidad, mientras que un valor-p bajo sugiere que los datos presentan desviaciones significativas respecto al modelo normal.

Adicionalmente, se emplean criterios de información como el AIC (Akaike Information Criterion) y el BIC (Bayesian Information Criterion) para comparar el ajuste de distintas distribuciones teóricas a los datos observados. Estos criterios penalizan la complejidad del modelo, por lo que valores menores indican un mejor compromiso entre calidad de ajuste y parsimonia del modelo. En este estudio, la Distribución normal* presenta valores competitivos frente a otras distribuciones evaluadas, lo cual respalda su selección como modelo probabilístico para la variable de velocidad.

A continuación, se presentan los datos utilizados para el análisis de la métrica de velocidad, así como el procedimiento de ajuste de distribuciones probabilísticas mediante la función `fit.cont()` del paquete `rriskDistributions` que se puede encontrar documentado en `https://cran.rstudio.com/web/packages/rriskDistributions/index.html`. Esta función permite estimar los parámetros de distintas distribuciones teóricas y evaluar su grado de ajuste a los datos empíricos, utilizando criterios estadísticos y gráficos diagnósticos para seleccionar el modelo probabilístico más adecuado para la variable de estudio.

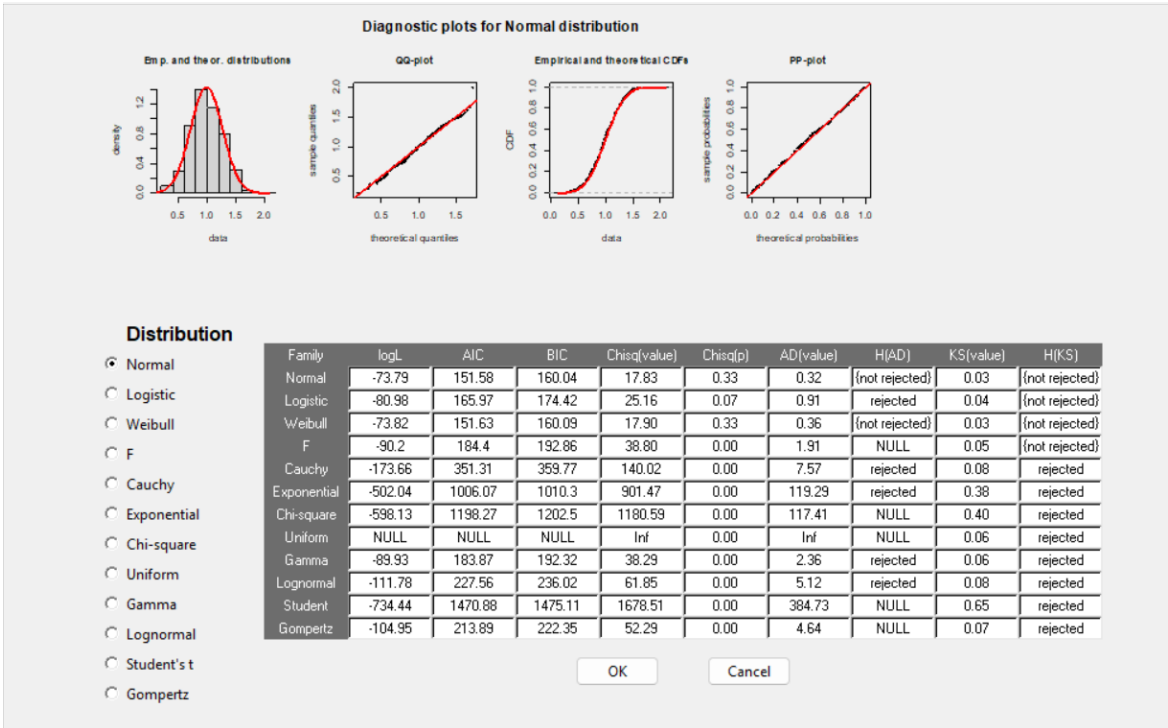
```
# Librerias a utilizar
library(data.table)
library(ggplot2)
library(readxl)
library(rriskDistributions)
library(dplyr) # Extraemos la información del excel y hacemos pruebas

data_v2 <- read_excel("data_velocidad.xlsx", sheet = "Datos Limpios")
data_v2 <- as.data.table(data_v2)

# Muestra datos

head(data_v2, 5)
```

ID Incidencia <chr>	Estimación original <dbl>	Tiempo Devengado <dbl>	Velocidad <dbl>
Req-1	224	180.65	1.24
Req-2	24	33.01	0.73
Req-3	14	9.18	1.53
Req-4	68	75.35	0.90
Req-5	10	6.94	1.44
5 rows			

Cuadro 4.1: head(data_v2, 5)

```
fit.cont(data_v2$Velocidad)
```

```
##
## Chosen continuous distribution is: Normal (norm)
## Fitted parameters are:
##      mean        sd
## 0.9902564 0.2798805
```



Family	logL	AIC	BIC	Chisq(value)	Chisq(p)	AD(value)	H(AD)	KS(value)	H(KS)
Normal	-73.79	151.58	160.04	17.83	0.33	0.32	{not rejected}	0.03	{not rejected}
Logistic	-80.98	165.97	174.42	25.16	0.07	0.91	rejected	0.04	{not rejected}
Weibull	-73.82	151.63	160.09	17.90	0.33	0.36	{not rejected}	0.03	{not rejected}
F	-90.2	184.4	192.86	38.80	0.00	1.91	NULL	0.05	{not rejected}
Cauchy	-173.66	351.31	359.77	140.02	0.00	7.57	rejected	0.08	rejected
Exponential	-502.04	1006.07	1010.3	901.47	0.00	119.29	rejected	0.38	rejected
Chi-square	-598.13	1198.27	1202.5	1180.59	0.00	117.41	NULL	0.40	rejected
Uniform	NULL	NULL	NULL	Inf	0.00	Inf	NULL	0.06	rejected
Gamma	-89.93	183.87	192.32	38.29	0.00	2.36	rejected	0.06	rejected
Lognormal	-111.78	227.56	236.02	61.85	0.00	5.12	rejected	0.08	rejected
Student	-734.44	1470.88	1475.11	1678.51	0.00	384.73	NULL	0.65	rejected
Gompertz	-104.95	213.89	222.35	52.29	0.00	4.64	NULL	0.07	rejected

Gráfico 4.1: Pantalla del paquete

El análisis numérico se complementa con evaluación gráfica, mostrada en el gráfico 4.1, la cual permite validar visualmente los supuestos estadísticos:

- **Histograma con curva normal ajustada:** permite comparar la forma empírica de los datos con la campana teórica. Una coincidencia razonable sugiere simetría y concentración alrededor de la Media*.
- **Gráfico Q–Q (Quantile–Quantile):** compara los cuantiles empíricos con los cuantiles de una Distribución normal* teórica. La alineación aproximada de los puntos sobre una recta indica compatibilidad con el modelo normal, mientras que curvaturas o desviaciones en los extremos sugieren asimetría o colas pesadas.
- **Función de distribución acumulada (CDF):** compara la proporción acumulada de observaciones reales con la función de distribución teórica, permitiendo evaluar el ajuste global a lo largo de todo el rango de valores.
- **Gráfico P–P (Probability–Probability):** compara probabili-

dades acumuladas empíricas y teóricas. La cercanía de los puntos a la diagonal indica buen ajuste del modelo probabilístico.

La concordancia entre las pruebas estadísticas y el análisis visual proporciona una validación robusta del supuesto de normalidad para la variable de velocidad. Este resultado permite aplicar métodos paramétricos para la construcción de intervalos de confianza, análisis de estabilidad del proceso y simulación del Método de Monte Carlo*, los cuales se emplean en las etapas posteriores del estudio.

Este procedimiento evalúa múltiples familias de distribuciones y calcula, para cada una, métricas de ajuste tales como el log-verosímil ($logL$), los criterios de información AIC y BIC, así como estadísticas de pruebas de bondad de ajuste como Kolmogorov–Smirnov (KS) y Anderson–Darling (AD).

En el gráfico 4.1 correspondiente al ajuste de distribuciones se observa que la Distribución normal* presenta uno de los valores más altos de log-verosímil, lo que indica una mayor probabilidad de que los datos observados provengan de dicho modelo en comparación con otras distribuciones evaluadas. Asimismo, los valores de AIC y BIC para la Distribución normal* se encuentran entre los más bajos del conjunto, lo cual sugiere un buen equilibrio entre calidad de ajuste y complejidad del modelo. En contraste, distribuciones como Exponencial, Chi-cuadrada y Lognormal presentan valores considerablemente mayores de AIC y BIC, indicando un ajuste deficiente a los datos de velocidad.

Desde el punto de vista de las pruebas de bondad de ajuste, la Distribución normal* no es rechazada por las pruebas de Kolmogorov–Smirnov y Anderson–Darling, como se indica por los valores-p asociados y por los indicadores de no rechazo reportados en la tabla. En cambio, varias distribuciones alternativas son rechazadas por al menos una de estas pruebas, lo cual refuerza la hipótesis de que la Normal constituye un modelo adecuado para describir la variabilidad observada en la velocidad.

Adicionalmente, el análisis gráfico presentado en el gráfico 4.1 respalda esta conclusión, ya que los puntos del gráfico Q–Q y del gráfico P–P se alinean de forma aproximada con la recta teórica, y la función de distribución acumulada empírica presenta alta concordancia con la Distribución normal* teórica. Estas evidencias visuales confirman que

no se observan asimetrías severas ni colas pesadas que justifiquen el uso de distribuciones alternativas.

Una vez seleccionada la Distribución normal* como modelo probabilístico para la velocidad, se estiman sus parámetros mediante máxima verosimilitud, obteniéndose una Media* estimada de $\mu = 0{,}9903$ y una Desviación Estándar* de $\sigma = 0{,}2799$. La Media* cercana a la unidad indica que, en promedio, el Esfuerzo planeado* y el Esfuerzo devengado* tienden a coincidir, lo que sugiere que el proceso de estimación no presenta un sesgo sistemático fuerte. Sin embargo, el valor de la Desviación Estándar* revela una variabilidad considerable alrededor de este promedio, lo cual explica la presencia de desviaciones relevantes en el cumplimiento de tiempos observadas en la operación real.

En términos operativos, esta dispersión implica que, aunque el proceso es centrado en promedio, existe una probabilidad significativa de que determinadas incidencias requieran más esfuerzo del estimado, generando retrasos acumulados a nivel de iteración* o de proyecto. Por esta razón, el modelo probabilístico de la velocidad resulta fundamental para la construcción de escenarios de simulación y para la estimación de intervalos de cumplimiento, permitiendo evaluar el riesgo de desviaciones futuras en los tiempos de entrega.

4.1.3. Control Estadístico de Procesos: Detección de Variaciones

Conexión entre problema real y modelo matemático

El proceso de estimación y ejecución del esfuerzo en proyectos de software está sujeto a dos tipos de variabilidad: *variabilidad por causas comunes* y *variabilidad por causas especiales*. Las causas comunes representan la variación natural e inherente al proceso, asociada a factores rutinarios como pequeñas diferencias en complejidad de tareas, desempeño normal del equipo o condiciones operativas estándar. Las causas especiales, en cambio, corresponden a eventos no habituales como errores graves de estimación, interrupciones externas, cambios abruptos de alcance o fallas técnicas, las cuales generan comportamientos atípicos en los datos (Montgomery, 2019).

Bajo condiciones de estabilidad estadística, se asume que la métrica de

velocidad V se comporta como una variable aleatoria aproximadamente normal:

$$H_0 : V \sim \mathcal{N}(\mu, \sigma^2),$$

donde μ representa el desempeño promedio del proceso de estimación–ejecución y σ^2 su variabilidad natural bajo operación normal. La aceptación de esta hipótesis implica que el proceso se encuentra bajo control estadístico y que la mayor parte de la variación observada se debe únicamente a causas comunes.

Para monitorear esta estabilidad se emplean gráficos de control, en los cuales se definen límites de control superior (LCS) e inferior (LCI) como:

$$LCS = \bar{V} + ks, \qquad LCI = \bar{V} - ks,$$

donde $\bar{V}$ es la Media* muestral, s es la desviación estándar muestral y k es un coeficiente que determina el nivel de sensibilidad del gráfico. El valor tradicional $k = 3$ se fundamenta en la propiedad de la Distribución normal* de que aproximadamente el 99,73 % de las observaciones se encuentran dentro del intervalo $\mu \pm 3\sigma$, lo que implica una probabilidad teórica de falsa alarma cercana al 0,27 % (Shewhart, 1931, Cap. 1–3) (Montgomery, 2019, Cap. 6).

En contextos de mejora de procesos, una vez que el sistema ha alcanzado estabilidad y se desea incrementar la sensibilidad para detectar variaciones más pequeñas, es posible utilizar límites más estrictos como $\pm 2\sigma$ o incluso $\pm 1{,}5\sigma$. Reducir el valor de k permite detectar cambios sutiles en el desempeño, pero incrementa el riesgo de falsas alarmas, por lo que esta decisión debe tomarse cuando el proceso ya se encuentra suficientemente maduro y controlado (Montgomery, 2019).

Cuando se detectan puntos fuera de los límites de control, estos no deben eliminarse de forma automática. Primero, deben investigarse para identificar si corresponden a causas especiales reales del proceso. Solo después de documentar y, en su caso, corregir dichas causas, los puntos asociados a eventos no representativos pueden excluirse para recalcular los límites de control y así obtener una estimación más precisa de la

variabilidad inherente del proceso. Este procedimiento evita que anomalías temporales distorsionen la caracterización estadística del desempeño normal del sistema.

En el contexto de esta investigación, este enfoque permite distinguir entre desviaciones ocasionales en la métrica de velocidad y problemas estructurales en el proceso de estimación, proporcionando un marco matemático para evaluar si las acciones de mejora implementadas producen una reducción real de la variabilidad del desempeño del equipo.

Interpretación práctica de las cartas de control de velocidad

Para evaluar la estabilidad del proceso de atención de incidencias, se construyó una secuencia iterativa de cartas de control para valores individuales de la métrica de velocidad. Cada carta corresponde a una depuración progresiva del conjunto de datos, siguiendo el principio de que los límites de control deben estimarse únicamente con datos representativos del comportamiento normal del proceso.

En la primera carta, construida con límites de $\pm 3\sigma$, se identificaron múltiples observaciones fuera de control. Estas observaciones fueron analizadas desde el punto de vista operativo y se asociaron con eventos específicos, tales como:

- Caídas temporales del sistema de gestión de tickets,
- Reprocesos derivados de errores de clasificación,
- Incidencias extraordinarias con dependencia de terceros,
- Cambios temporales en la asignación de personal.

Estos eventos no representan el desempeño típico del proceso y, por lo tanto, fueron clasificados como causas especiales. Conforme a la metodología de Control Estadístico de Procesos* (CEP), estos puntos no deben utilizarse para estimar los parámetros μ y σ, ya que inflan artificialmente la variabilidad y distorsionan los límites de control.

Es importante destacar que las observaciones correspondientes a causas especiales no se eliminan del análisis histórico ni del seguimiento operativo; únicamente se excluyen de manera temporal para el cálculo de los límites de control, con el objetivo de caracterizar el comportamiento

estable del proceso.

Después de remover los puntos asociados a causas especiales, se recalcularon la Media* y la desviación estándar, obteniendo límites de control progresivamente más estrechos. Este proceso se repitió hasta que las cartas mostraron un patrón estable, sin señales de fuera de control, lo que indica que la variabilidad remanente corresponde principalmente a causas comunes.

Posteriormente, con el proceso ya depurado de causas especiales, se redujo el valor de k de 3 a 2 sigmas:

$$LCS = \bar{V} + 2\sigma, \qquad LCI = \bar{V} - 2\sigma$$

Este ajuste incrementa la sensibilidad de la carta, ya que el intervalo cubre aproximadamente el 95,45 % de la Distribución normal*, aumentando la probabilidad de detección temprana de desviaciones del proceso. Matemáticamente, la probabilidad de falsa alarma se incrementa de 0,27 % a aproximadamente 4,55 %, lo cual es aceptable en contextos donde el costo de no detectar una degradación del proceso es mayor que el costo de investigar una falsa señal (Montgomery, 2019, Cap. 6, pp. 180–184, 209–214, 256–260) (Woodall, 2000, pp. 341–350).

En este estudio, la reducción de sigmas se justifica debido a que la métrica de velocidad impacta directamente en los compromisos de nivel de servicio (SLA), por lo que se prioriza la detección temprana de cambios sutiles en el desempeño operativo.

La secuencia completa de cartas (Gráficos 4.2 al 4.11) muestra cómo, a medida que se eliminan las causas especiales y se recalculan los límites, el proceso converge hacia un comportamiento estable alrededor de la Media*, evidenciando que el modelo estadístico utilizado es consistente con la operación real del sistema una vez controladas las perturbaciones externas.

Este procedimiento permite no solo verificar la estabilidad estadística, sino también identificar oportunidades de mejora estructural, ya que cualquier reducción posterior de la variabilidad deberá provenir de cambios en el proceso mismo y no de filtrado estadístico.

En esta sección se analizan las cartas de control correspondientes a la

métrica de velocidad del proceso de atención de incidencias. Se presentan de forma secuencial con el objetivo de mostrar cómo la eliminación progresiva de causas especiales y el ajuste de la sensibilidad estadística permiten identificar el comportamiento estable del proceso, condición necesaria para aplicar técnicas de mejora continua y modelado predictivo.

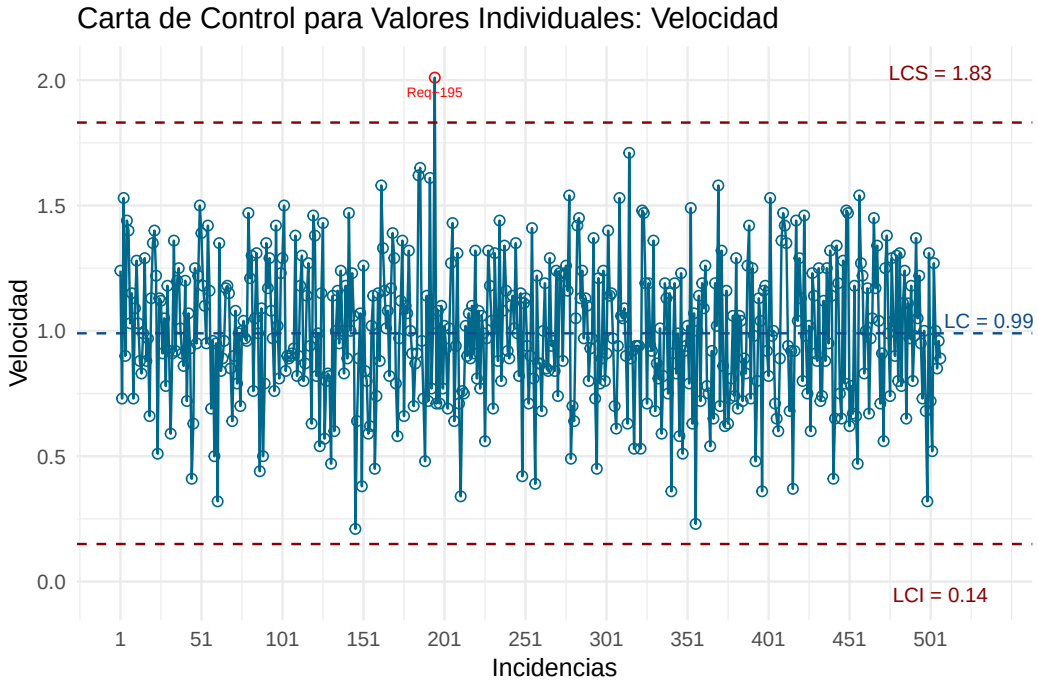


Gráfico 4.2: Carta de Control para Valores Individuales: Velocidad 1

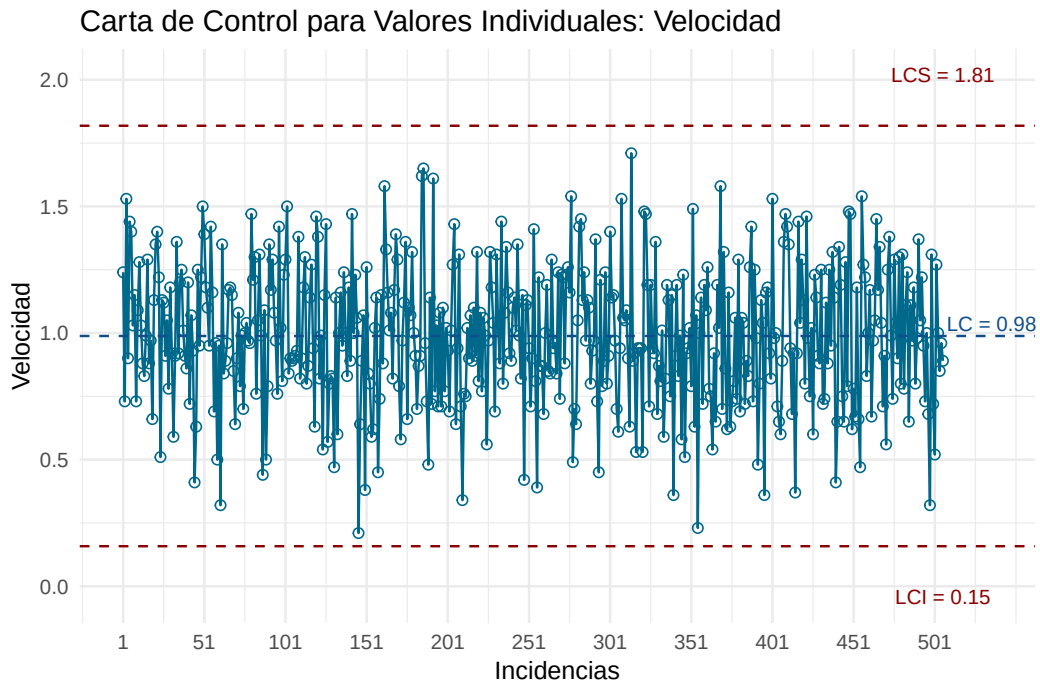


Gráfico 4.3: Carta de Control para Valores Individuales: Velocidad 2

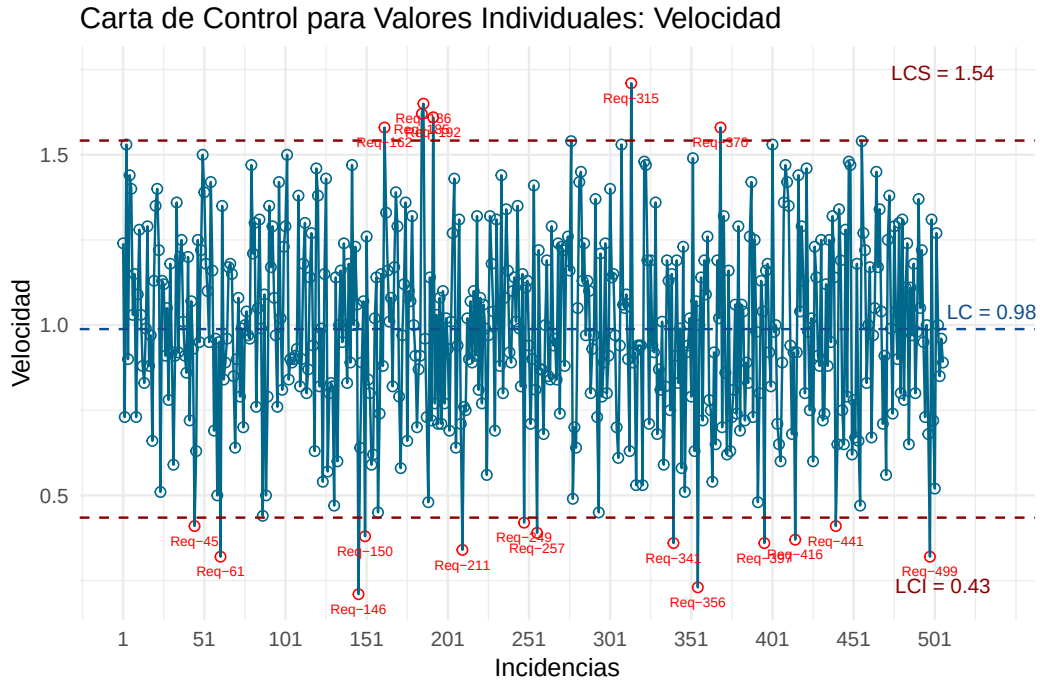


Gráfico 4.4: Carta de Control para Valores Individuales: Velocidad 3

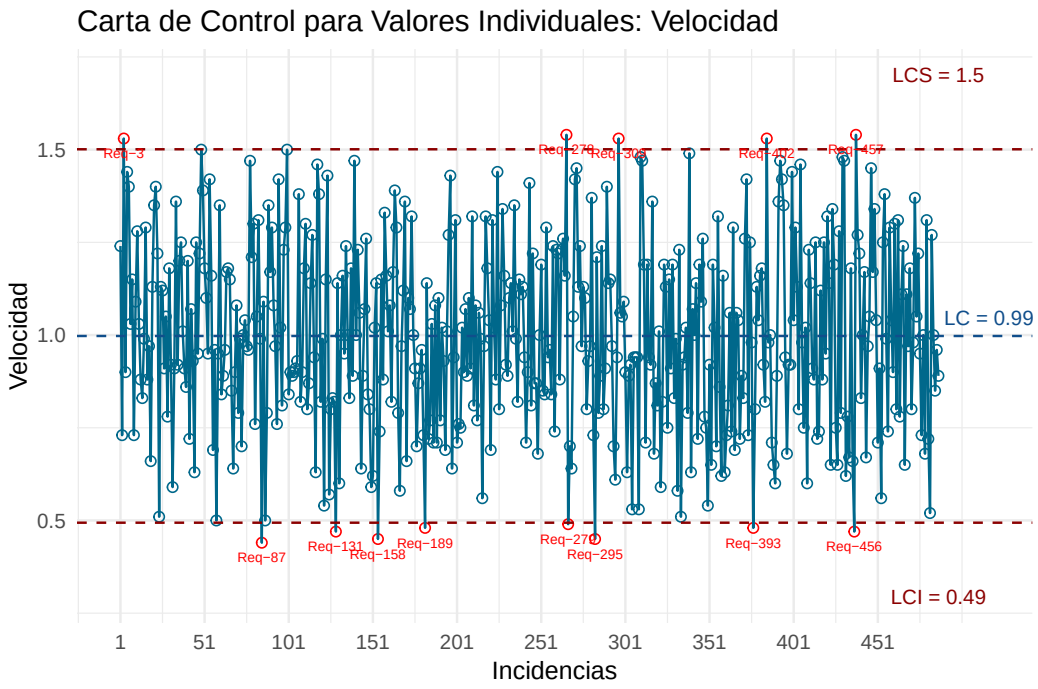


Gráfico 4.5: Carta de Control para Valores Individuales: Velocidad 4

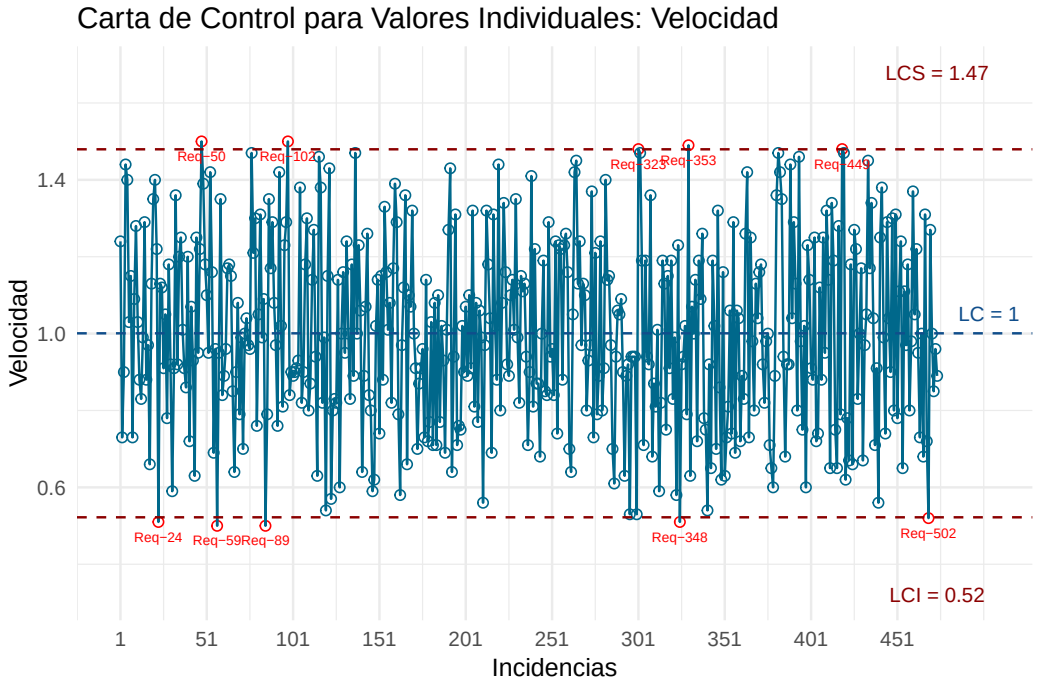


Gráfico 4.6: Carta de Control para Valores Individuales: Velocidad 5

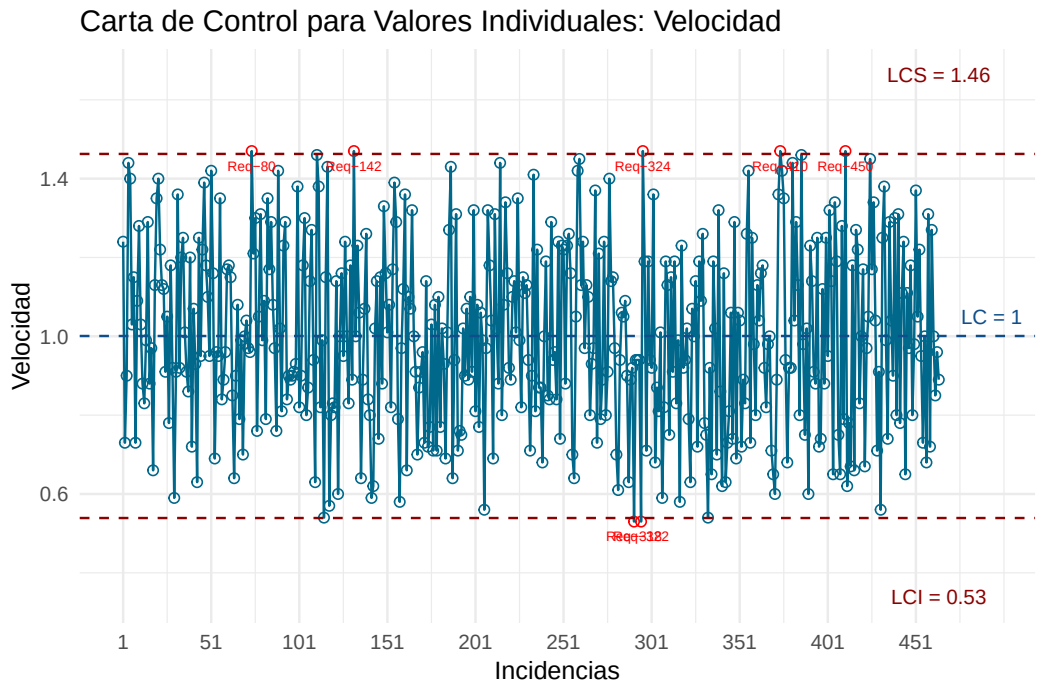


Gráfico 4.7: Carta de Control para Valores Individuales: Velocidad 6

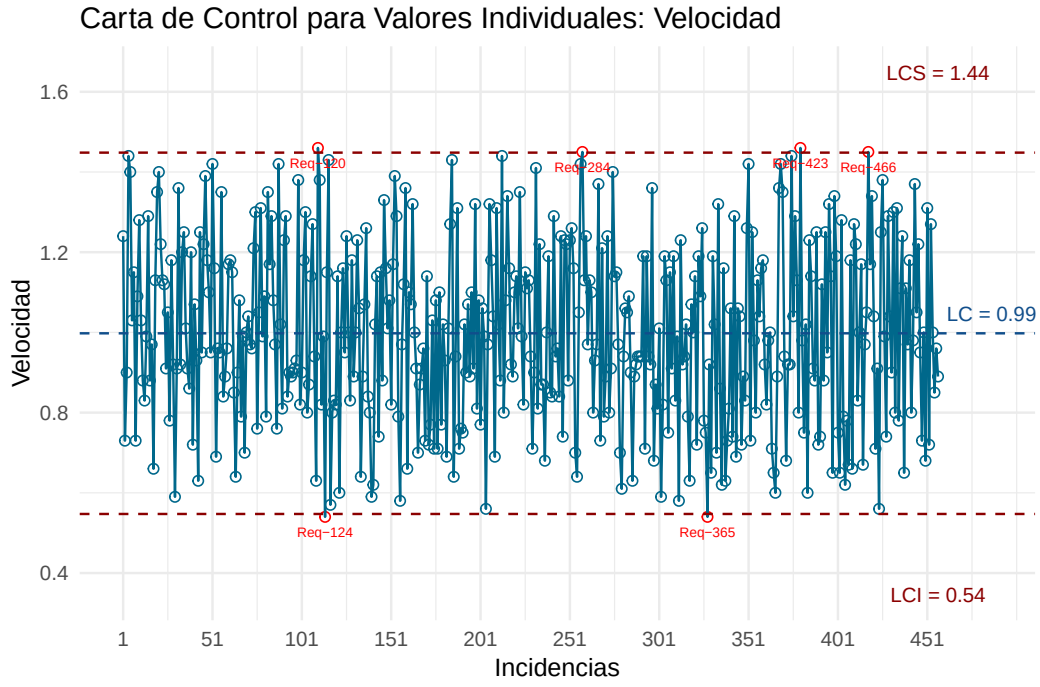


Gráfico 4.8: Carta de Control para Valores Individuales: Velocidad 7

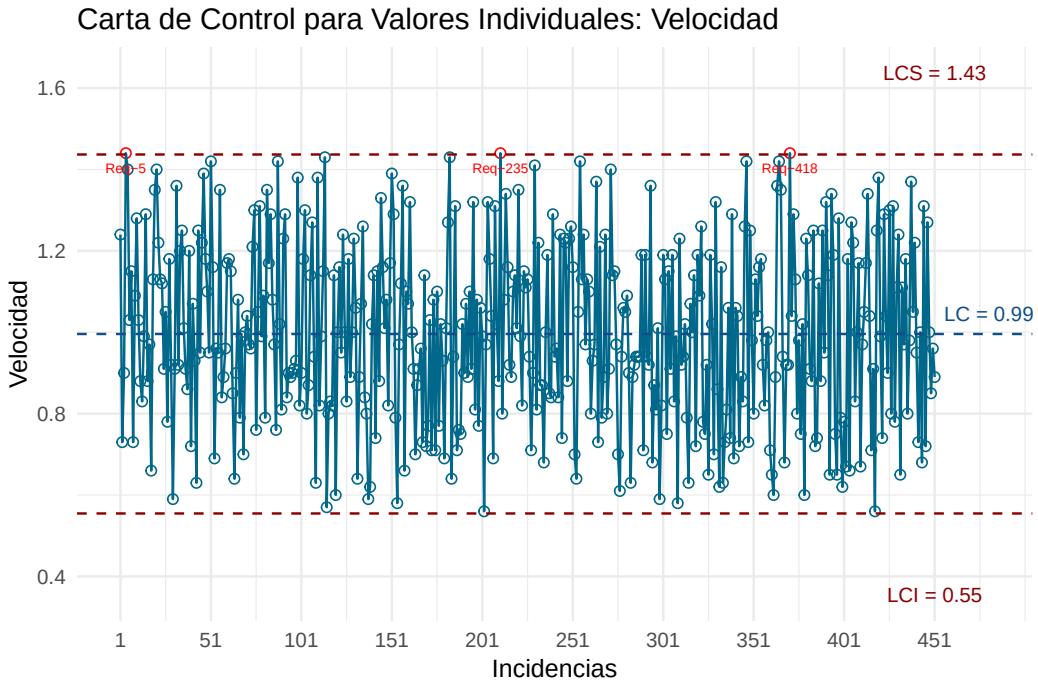


Gráfico 4.9: Carta de Control para Valores Individuales: Velocidad 8

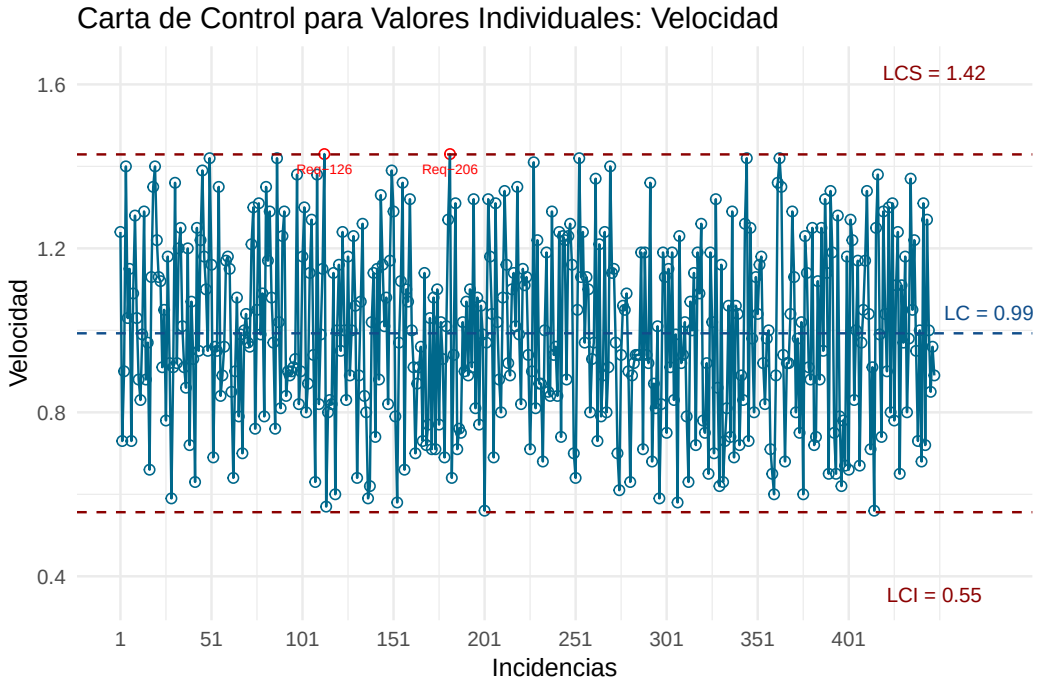


Gráfico 4.10: Carta de Control para Valores Individuales: Velocidad 9

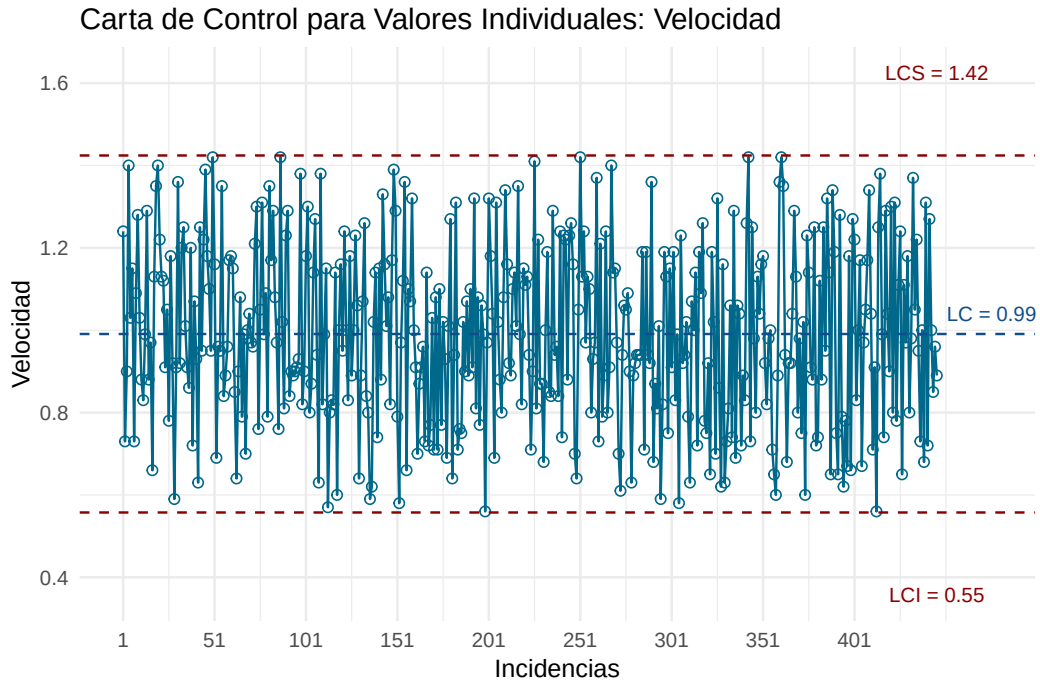


Gráfico 4.11: Carta de Control para Valores Individuales: Velocidad 10

En conjunto, la secuencia de cartas de control permitió evidenciar el proceso iterativo de depuración estadística de la métrica de velocidad, mediante la identificación progresiva de causas especiales de variación y el posterior ajuste de los límites de control. Este procedimiento no tuvo como objetivo mejorar artificialmente los indicadores, sino aislar la variabilidad inherente al proceso bajo condiciones normales de operación, con el fin de obtener una representación más fiel de su comportamiento estable.

La reducción gradual de la dispersión observada y la concentración de las observaciones dentro de los límites de control indican que, una vez excluidos los eventos atípicos asociados a incidencias específicas, el proceso de estimación y ejecución presenta un patrón de variación consistente y predecible. Este resultado es fundamental desde el punto de vista de la gestión cuantitativa, ya que únicamente un proceso estable puede ser utilizado como base para la planeación, la simulación de escenarios y la evaluación objetiva de mejoras.

Asimismo, el uso de límites de control más estrictos en etapas posteriores permitió incrementar la sensibilidad del análisis para detectar desviaciones pequeñas pero sistemáticas, lo cual es especialmente relevante

en contextos donde los retrasos acumulados pueden impactar significativamente la capacidad de cumplimiento. De esta forma, las cartas de control no solo funcionaron como una herramienta de diagnóstico, sino también como un mecanismo de aprendizaje organizacional que permitió comprender mejor las fuentes de variabilidad del proceso y orientar acciones correctivas con sustento estadístico.

4.1.4. Evaluación estadística de las mejoras implementadas

Diseño cuasi-experimental

Con el objetivo de evaluar el efecto de las acciones de mejora implementadas en el proceso de estimación y ejecución (capacitaciones, ajustes en el registro y estandarización operativa), se utilizó un diseño cuasi-experimental de tipo antes–después, basado en el análisis comparativo de dos periodos temporales:

- **Periodo Antes:** conjunto de observaciones históricas de la métrica de velocidad registradas previo a las intervenciones.
- **Periodo Después:** conjunto de observaciones registradas tras la implementación de las mejoras.

Este tipo de diseño es apropiado cuando no es posible realizar asignación aleatoria de tratamientos, pero se dispone de mediciones comparables en el tiempo bajo condiciones operativas similares, permitiendo evaluar cambios estadísticamente significativos en el desempeño del proceso (Montgomery, 2019, pp. 540–545).

Pruebas de hipótesis para comparación de medias

Dado que el interés principal es determinar si las mejoras produjeron un cambio significativo en el nivel promedio de la métrica de velocidad, se plantea el contraste de hipótesis:

$$H_0 : \mu_{\text{Antes}} = \mu_{\text{Después}}, \qquad H_a : \mu_{\text{Antes}} \neq \mu_{\text{Después}}.$$

Bajo el supuesto de normalidad previamente validado y considerando dos grupos de observaciones independientes, el análisis se realiza me-

diante un modelo de análisis de Varianza* (ANOVA*) de un factor, el cual permite evaluar si las diferencias observadas entre los promedios de los periodos pueden atribuirse a la intervención o a variabilidad aleatoria (Montgomery, 2019, Cap. 12, pp. 413–420).

El estadístico de prueba se define como:

$$F = \frac{MS_{\text{Trat}}}{MS_{\text{Error}}} = \frac{SST/(k-1)}{SSE/(n-k)} \sim F_{k-1,n-k},$$

donde SST representa la suma de cuadrados entre tratamientos (periodos), SSE la suma de cuadrados del error, k el número de grupos (en este caso, dos periodos) y n el número total de observaciones.

Un Valor-p* menor que el nivel de significancia establecido indica evidencia estadística suficiente para rechazar H_0 y concluir que la intervención produjo un cambio significativo en la velocidad promedio del proceso.

Pruebas de homogeneidad de varianzas

Además del cambio en el promedio, resulta fundamental evaluar si las mejoras impactaron en la estabilidad del proceso, lo cual se refleja en la reducción de la variabilidad. Para ello se contrasta:

$$H_0 : \sigma^2_{\text{Antes}} = \sigma^2_{\text{Después}}, \qquad H_a : \sigma^2_{\text{Antes}} \neq \sigma^2_{\text{Después}}.$$

La Prueba de Bartlett* se emplea para evaluar la homogeneidad de varianzas entre los grupos, siendo adecuada cuando los datos provienen de distribuciones aproximadamente normales, como se verificó en la etapa de diagnóstico de normalidad (Montgomery, 2019, Cap. 7, pp. 252–255).

Una disminución estadísticamente significativa de la Varianza* posterior a la intervención se interpreta como evidencia de mayor control y consistencia en el proceso de estimación y ejecución, lo cual es coherente con los objetivos de gestión cuantitativa y mejora continua.

Relación con el control estadístico del proceso

El uso combinado de cartas de control y Pruebas de hipótesis* permite evaluar no solo la estabilidad del proceso, sino también la efectividad

de las acciones correctivas implementadas. Mientras que las cartas de control permiten identificar la eliminación de causas especiales y la estabilización del proceso, las pruebas inferenciales permiten validar si dicha estabilización se traduce en mejoras estadísticamente significativas en el desempeño promedio y en la variabilidad (Montgomery, 2019, Cap. 1, pp. 13–15).

De esta manera, el análisis integra control estadístico y evaluación inferencial como componentes complementarios dentro de un esquema de gestión cuantitativa del proceso.

```
# Realizamos las pruebas

# Traemos datos para hipotesis
Data_hip <- read_excel("Data_hip.xlsx")
Data_hip <- as.data.table(Data_hip)
```

```
# Separamos data frames
antes <- Data_hip[Grupo == "Antes"]
despues <- Data_hip[Grupo == "Despues"]

# Prueba de media
summary(aov(antes$Velocidad ~ después $Velocidad))
```

```
##                    Df Sum Sq Mean Sq F value Pr(>F)
## despues$Velocidad   1 39.71 39.71 4901851 <2e-16 ***
## Residuals         505 0.00 0.00
## - - -
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Prueba de Varianzas
bartlett.test(list(antes$Velocidad, después$Velocidad))
```

```
##
## Bartlett test of homogeneity of variances
##
## data: list(antes$Velocidad, después$Velocidad)
## Bartlett's K-squared = 1017.5, df = 1, p-value <2.2e-16
```

Resultados de las pruebas e interpretación

Los resultados del análisis de Varianza* (ANOVA*) muestran un valor del estadístico F extremadamente elevado ($F = 4{,}901{,}851$) con un Valor-p* menor que 2×10^{-16}, lo que indica evidencia estadística contundente para rechazar la hipótesis nula de igualdad de medias entre los periodos antes y después de la intervención. En términos prácticos, este resultado confirma que la velocidad promedio del proceso presenta un cambio estadísticamente significativo posterior a la implementación de las acciones de mejora.

Este hallazgo sugiere que las capacitaciones y ajustes operativos no solo estabilizaron el proceso, sino que también modificaron de manera estructural el nivel de desempeño, reflejando una mejora sostenida en la relación entre esfuerzo planificado y Esfuerzo devengado*. Dado que la métrica de velocidad se construye como un indicador de eficiencia relativa, este cambio implica una mayor capacidad del equipo para cumplir con las estimaciones establecidas.

Adicionalmente, la Prueba de Bartlett* para homogeneidad de varianzas arrojó un estadístico $K^2 = 1017{,}5$ con Valor-p* menor que $2{,}2 \times 10^{-16}$, lo cual indica una diferencia estadísticamente significativa entre las varianzas de ambos periodos. Este resultado confirma que la intervención no solo impactó el promedio del proceso, sino también su nivel de dispersión, lo cual es un indicador directo de mejora en la estabilidad y consistencia operativa.

Desde la perspectiva de Control Estadístico de Procesos*, una reducción en la variabilidad es especialmente relevante, ya que la predictibilidad del sistema depende más de la estabilidad que del valor promedio en sí mismo. Un proceso con menor dispersión permite realizar planeación más confiable, simulaciones más precisas y compromisos de entrega con menor nivel de riesgo.

La combinación de los resultados de las cartas de control, que evidencian la eliminación progresiva de causas especiales, con las pruebas inferenciales, que confirman cambios significativos en Media* y Varianza*, proporciona una validación estadística integral de la efectividad de las acciones de mejora implementadas. Este enfoque integrado es consistente con los principios de gestión cuantitativa del proceso, en los cuales las decisiones se sustentan tanto en monitoreo continuo como en evaluación

estadística formal de los resultados.

En consecuencia, los resultados obtenidos no solo respaldan la existencia de una mejora operativa, sino que demuestran que dicha mejora es estadísticamente significativa, estructural y sostenible, lo cual justifica el uso de modelos predictivos posteriores, como la simulación del Método de Monte Carlo*, basados en parámetros que representan el comportamiento estable del proceso.

4.1.5. Visualización de resultados

Con el objetivo de complementar el análisis inferencial y de control estadístico, se realizó una comparación visual de las distribuciones de la métrica de velocidad antes y después de la implementación de las acciones de mejora, mediante histogramas superpuestos y curvas de densidad estimadas.

El Gráfico 4.12 permite observar dos comportamientos claramente diferenciados. En primer lugar, el periodo posterior a la intervención presenta una concentración significativamente mayor de observaciones alrededor del valor objetivo de velocidad igual a 1, lo que indica una mejora en el cumplimiento de las estimaciones y una reducción del sesgo sistemático entre esfuerzo planificado y Esfuerzo devengado*.

En segundo lugar, se aprecia una disminución notable en la dispersión de la distribución correspondiente al periodo posterior, reflejada en una curva de densidad más alta y estrecha en comparación con el periodo previo. Este comportamiento es consistente con los resultados obtenidos en las pruebas de homogeneidad de varianzas y en las cartas de control, donde se identificó una reducción progresiva de la variabilidad del proceso tras la eliminación de causas especiales.

Desde el punto de vista operativo, esta reducción en la variabilidad implica una mayor predictibilidad del proceso, lo cual es un requisito fundamental para la planeación confiable de entregas y la aplicación de modelos predictivos posteriores, como la simulación del Método de Monte Carlo*. Un proceso con media cercana al valor objetivo y baja dispersión permite estimar con mayor precisión el riesgo asociado a los compromisos de tiempo y capacidad.

En conjunto, la evidencia visual respalda los resultados obtenidos me-

diante las pruebas estadísticas y confirma que las mejoras implementadas no solo generaron un cambio significativo en el nivel promedio del desempeño, sino también una estabilización estructural del proceso, condición necesaria para avanzar hacia una gestión cuantitativa madura del desempeño del servicio.

```
# Histograma ambos grupos

# Crear el gráfico de ggplot con dos histogramas superpuestos como curvas
ggplot(datos, aes(x = Velocidad, fill = Grupo, color = Grupo)) +
geom_histogram(aes(y = ..density..), alpha = 0.4, bins = 30, position = "identity") +
```

```
Líneas verticales de la media geom_vline(data = medias, aes(xintercept = media, color = Grupo), linetype = "dashed", linewidth = 1.1) +
labs(title = "Comparación de Procesos", y = "Densidad", x = "Velocidad", fill = "Periodo", color = "Periodo") +
theme_minimal()
```

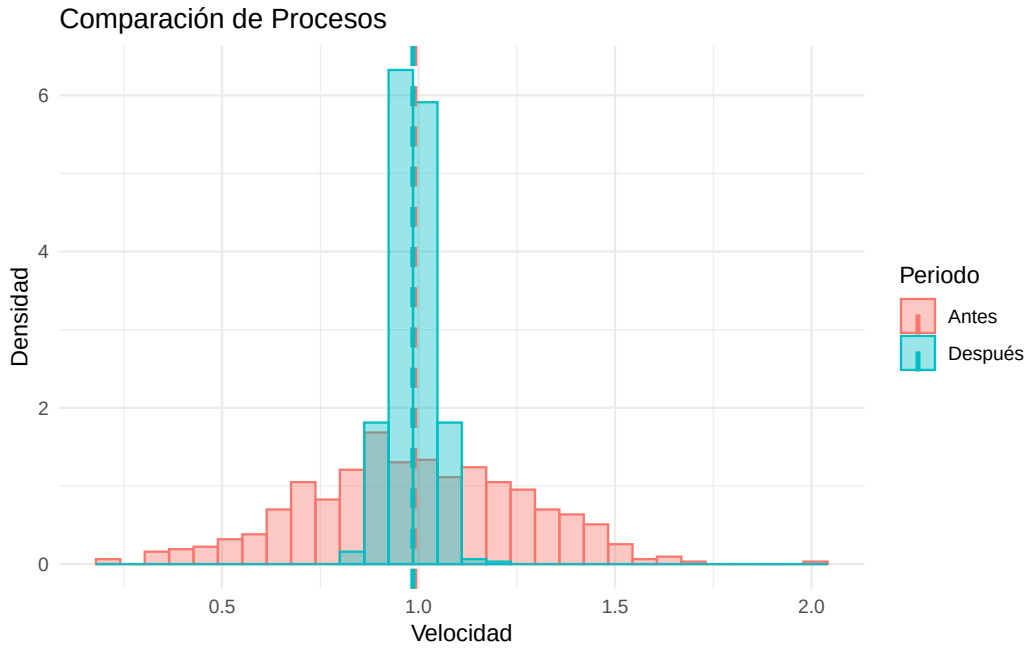


Gráfico 4.12: Comparación de Procesos

4.1.6. Conclusiones del ejercicio, beneficios y retos

El presente ejercicio se originó a partir de una problemática operativa relevante: la organización presentaba desviaciones promedio en los tiempos de entrega cercanas al 35 %, lo que generaba incumplimientos en compromisos de servicio, baja predictibilidad en la planeación y dificultades para la gestión efectiva del portafolio de proyectos. Este nivel de variabilidad evidenciaba un proceso inestable, dominado por causas especiales y sin capacidad de predicción confiable.

A partir de esta situación, se implementó un enfoque cuantitativo integral basado en la métrica de velocidad, definida como la relación entre el esfuerzo planificado y el esfuerzo realmente devengado por incidencia. Esta variable permitió traducir el desempeño operativo del proceso en una medida continua susceptible de análisis estadístico riguroso.

La validación empírica de la aproximación a la normalidad de la variable velocidad sustentó el uso de técnicas paramétricas para el análisis posterior, confirmando que la mayor parte de la variabilidad observada corresponde a la acumulación de múltiples factores pequeños y no a eventos sistemáticos permanentes. Esto habilitó la construcción de cartas de control para valores individuales, que funcionaron como mecanismo principal para diferenciar entre variación común del proceso y causas especiales atribuibles a eventos operativos concretos.

El análisis iterativo de las cartas de control permitió identificar incidencias asociadas a bloqueos técnicos, errores de registro de tiempos, cambios urgentes de prioridad y re-trabajos no previstos. Estas observaciones no fueron descartadas del historial, sino tratadas como eventos especiales para el rediseño del proceso, y posteriormente excluidas del cálculo de límites de control con el fin de estimar la variabilidad inherente del sistema bajo condiciones normales de operación. Esta práctica es consistente con los principios del Control Estadístico de Procesos*, donde los límites deben reflejar únicamente la variación común y no distorsionarse por eventos excepcionales.

Adicionalmente, la reducción deliberada de los límites de control de $\pm 3\sigma$ a $\pm 2\sigma$ incrementó la sensibilidad del sistema de monitoreo, permitiendo la detección temprana de desviaciones menores pero sistemáticas. Esta decisión es especialmente pertinente en entornos de servicios de TI, donde los ciclos de retroalimentación deben ser cortos y el costo de no

detectar una degradación del proceso puede ser significativo.

Las pruebas inferenciales aplicadas (ANOVA* para comparación de medias y Prueba de Bartlett* para homogeneidad de varianzas) confirmaron estadísticamente que las mejoras implementadas —capacitaciones técnicas, estandarización en el registro de tiempos, clarificación de criterios de estimación y priorización operativa— produjeron cambios significativos tanto en el nivel promedio de la velocidad como en la estabilidad del proceso. Estos resultados no solo indican un mejor desempeño, sino una transformación estructural del comportamiento del sistema.

Desde el punto de vista operativo, el efecto más relevante fue la reducción del desvío promedio en los tiempos de entrega, que pasó de valores cercanos al 35 % a niveles actuales alrededor del 15 %, lo cual representa un cambio sustancial en la capacidad de cumplimiento y en la confiabilidad de los compromisos adquiridos con los clientes. Este resultado tiene implicaciones directas en la planeación de capacidad, la gestión de riesgos* y la percepción de calidad del servicio.

No obstante, la implementación inicial de estas prácticas implicó retos relevantes. Entre los principales se encontraron la resistencia cultural al uso de métricas, la interpretación limitada de las cartas de control por parte de algunos equipos y la necesidad de mejorar la calidad del registro de datos operativos. Superar estos retos requirió no solo herramientas estadísticas, sino también actividades de capacitación, acompañamiento y alineación con los objetivos del negocio.

Finalmente, el principal beneficio del enfoque adoptado no radica únicamente en la mejora puntual de indicadores, sino en la creación de una base histórica confiable que permite el uso de modelos predictivos, simulación de escenarios y evaluación continua del desempeño. De esta manera, el ejercicio no se limita a un análisis retrospectivo, sino que sienta las bases para una gestión proactiva del servicio, cerrando efectivamente el ciclo de mejora continua basado en evidencia cuantitativa.

Capítulo 5

Los Defectos en el Desarrollo de Software

"All models are wrong, but some are useful."
— *George Edward Pelham Box*

Importancia de la métrica de defectos

La métrica de defectos constituye uno de los pilares fundamentales para la evaluación objetiva de la calidad en el desarrollo de software. En el contexto de esta investigación, el interés no se centra únicamente en el conteo absoluto de defectos, sino en su caracterización mediante métricas normalizadas, particularmente la *Densidad de defectos*[*], que relaciona el número de defectos detectados con el Esfuerzo invertido[*] o el tamaño del producto desarrollado (Fenton and Pfleeger, 1997, Cap. 7, pp. 173–210)(Kan, 2002, Cap. 5, pp. 137–182).

Desde una perspectiva operativa, un defecto se define como cualquier desviación entre el comportamiento esperado del sistema y su comportamiento real, independientemente de la fase del ciclo de vida en la que se origine. Estos defectos pueden introducirse desde etapas tempranas como el análisis de requisitos y el diseño, hasta fases posteriores como la implementación, pruebas o incluso operación en producción (Pressman and Maxim, 2020, Cap. 4, pp. 90–118) (Sommerville, 2016, Cap. 8, pp. 246–273). La acumulación no controlada de defectos impacta directa-

mente en la confiabilidad del software, en los costos de corrección y en los tiempos de entrega comprometidos con el cliente.

La utilización de métricas de defectos permite transformar la percepción subjetiva de la calidad en información cuantitativa verificable. En particular, la Densidad de defectos* facilita la comparación entre proyectos, iteraciones o fases del proceso, al eliminar el sesgo asociado al tamaño o complejidad del trabajo realizado. De acuerdo con Fenton y Pfleeger, este tipo de métricas es esencial para evaluar la estabilidad del proceso y para identificar patrones sistemáticos de generación de defectos que no son evidentes mediante métricas absolutas (Fenton and Pfleeger, 1997, Cap. 9, pp. 255–292).

Desde el punto de vista económico y operativo, la detección temprana de defectos es un factor crítico. Diversos estudios en ingeniería de software demuestran que el costo de corregir un defecto aumenta exponencialmente conforme avanza el ciclo de vida del producto. Un defecto detectado en fases de revisión o pruebas tempranas requiere significativamente menos esfuerzo que uno identificado en producción, donde puede afectar a usuarios finales y generar retrabajos, penalizaciones contractuales o pérdida de reputación organizacional (Pressman and Maxim, 2020, cap. 23, pp. 561–598). En este sentido, la métrica de defectos no solo mide calidad, sino que actúa como un mecanismo preventivo de control de costos.

Asimismo, las métricas de defectos desempeñan un papel clave en la planificación y estimación de proyectos. El análisis histórico de densidades permite construir rangos esperados de defectos por unidad de esfuerzo, los cuales pueden ser utilizados como parámetros de entrada en modelos predictivos, como los desarrollados en este trabajo mediante simulación del Método de Monte Carlo*. Este enfoque probabilístico proporciona a los gestores una visión realista de los escenarios posibles y permite dimensionar adecuadamente las actividades de revisión y pruebas (Kan, 2002, Cap. 6, pp. 183–224).

Finalmente, el seguimiento sistemático de la métrica de defectos habilita un ciclo de mejora continua basado en datos. Al analizar la evolución de la Densidad de defectos* a lo largo del tiempo, es posible identificar tendencias, evaluar el impacto de acciones correctivas —como capacitaciones o ajustes en los procesos— y distinguir entre variabilidad común

y causas especiales que requieren intervención específica. De esta forma, la métrica de defectos se consolida como un elemento central para la toma de decisiones informadas, la gestión del riesgo y el fortalecimiento de la madurez del proceso de desarrollo de software (Sommerville, 2016, Cap. 24, pp. 694–726) (Fenton and Pfleeger, 1997, Cap. 9, pp. 255–292).

5.1. Modelación estadística de defectos bajo restricciones de confidencialidad

En el contexto organizacional donde se desarrolla este estudio, el acceso a datos detallados de defectos a nivel incidencia o proyecto se encuentra restringido por políticas de confidencialidad. En consecuencia, no se dispone de observaciones individuales históricas, sino únicamente de *parámetros estadísticos agregados* (rangos mínimos y máximos de densidad) por fase del proceso de desarrollo. Esta restricción condiciona el enfoque metodológico adoptado, orientándolo hacia una modelación probabilística basada en tasas y no en conteos absolutos.

Sea una iteración* $i = 1, \ldots, n$ y sea e_i el Esfuerzo invertido* (en horas) en dicha iteración*. Para cada fase f del proceso de desarrollo se define la *Densidad de defectos** como:

$$\rho_{f,i} = \frac{d_{f,i}}{e_i},$$

donde $d_{f,i}$ representa el número de defectos detectados en la fase f durante la iteración* i. Esta magnitud expresa una **Tasa de defectos* por unidad de esfuerzo**, lo que permite comparar proyectos o iteraciones de distinto tamaño bajo una misma escala.

Sea d una variable aleatoria que representa la densidad de defectos del proceso de desarrollo, definida como la razón entre el número de defectos observados y el tamaño del software.

Se asume que d sigue una distribución de probabilidad cuyos parámetros son estimados a partir de los datos históricos analizados.

El método de simulación de Monte Carlo consiste en generar una muestra de realizaciones independientes $\{d_1, d_2, \ldots, d_n\}$ de la variable alea-

toria d, con el fin de aproximar su comportamiento probabilístico.

A partir de estas realizaciones, se estiman cantidades de interés tales como el valor esperado, la varianza y percentiles, los cuales permiten caracterizar la incertidumbre asociada a la densidad de defectos en el proceso de desarrollo de software.

Desde un punto de vista operativo, $\rho_{f,i}$ puede interpretarse como el *número promedio de defectos que se espera encontrar por cada hora de trabajo invertida*. Por ejemplo, un valor $\rho = 0{,}01$ indica que, en promedio, se detecta 1 defecto por cada 100 horas de esfuerzo.

Dado que no se cuenta con observaciones individuales $\rho_{f,i}$, la organización dispone de estimaciones agregadas de dichas tasas, expresadas como intervalos empíricos por fase. En el estado actual del proceso, las tasas promedio observadas se caracterizan de la siguiente forma:

- **Revisión de Producto (RP):**

$$\rho_{\text{RP}} \in [0{,}020,\ 0{,}040]$$

  Este rango indica que, durante la revisión de producto, se detectan entre 2 y 4 defectos por cada 100 horas de esfuerzo. Operativamente, una tasa elevada en esta fase es deseable, ya que refleja una detección temprana de problemas antes de que avancen a etapas más costosas.

- **Análisis y Revisión (AR):**

$$\rho_{\text{AR}} \in [0{,}00675,\ 0{,}0105]$$

  Este intervalo representa entre 0.7 y 1 defecto por cada 100 horas de esfuerzo. Desde una perspectiva práctica, estos valores reflejan la capacidad del proceso de análisis técnico para identificar defectos de diseño o especificación, contribuyendo a reducir fallas posteriores.

- **Pruebas – esfuerzo relativo (PR):**

$$\rho_{\text{PR}}^{(e)} \in [0{,}0614,\ 0{,}2812]$$

Esta tasa no representa defectos, sino la proporción del esfuerzo total del proyecto dedicada a pruebas. En términos prácticos, indica que entre el 6 % y el 28 % del esfuerzo total se invierte en actividades de prueba, reflejando una fase con alta variabilidad operativa.

- **Pruebas – densidad de defectos (PR):**

$$\rho_{\text{PR}} \in [0{,}002,\ 0{,}005]$$

Este rango implica entre 0.2 y 0.5 defectos por cada 100 horas de esfuerzo. Para el cliente final, esta tasa es crítica, ya que defectos detectados en pruebas suelen estar más próximos a producción y, por tanto, su impacto en tiempos de entrega y costos de corrección es mayor.

Estas tasas describen el comportamiento actual del proceso y evidencian que, si bien la Densidad de defectos* disminuye conforme avanza el ciclo de vida, la variabilidad asociada al esfuerzo y a la detección tardía de defectos sigue siendo relevante. Desde la perspectiva del cliente, una densidad elevada en fases avanzadas se traduce en mayor riesgo de fallas en producción y en desviaciones de los tiempos comprometidos.

Bajo el supuesto de estabilidad operativa del proceso, cada densidad ρ_f se modela como una Variable aleatoria* continua que sigue aproximadamente una Distribución normal*:

$$\rho_f \sim \mathcal{N}(\mu_f, \sigma_f^2),$$

donde μ_f representa la tasa promedio de defectos por unidad de esfuerzo y σ_f^2 cuantifica la variabilidad natural del proceso. Para un lector no especializado, esta hipótesis implica asumir que la mayoría de los proyectos se comportan de forma similar, con pequeñas variaciones alrededor de un valor promedio, y que los comportamientos extremos son poco frecuentes.

A partir de esta modelación, se emplea simulación del Método de Monte Carlo* para generar un conjunto de realizaciones $\{\rho_f^{(j)}\}_{j=1}^{M}$ que aproximan la distribución subyacente de cada fase. Para un esfuerzo dado e_i, el número esperado de defectos se obtiene mediante:

$$d_{f,i}^{(j)} = e_i \cdot \rho_f^{(j)}, \quad j = 1, \ldots, M.$$

Desde el punto de vista operativo, esta expresión permite responder preguntas clave como: *"si se invierten e_i horas en esta fase, ¿cuántos defectos es razonable esperar?"*.

A partir de estas simulaciones se construyen intervalos de confianza del 95 % para el número de defectos esperados:

$$d_{f,i}^{\text{mín}} = e_i \cdot \rho_f^{(2{,}5\,\%)}, \qquad d_{f,i}^{\text{máx}} = e_i \cdot \rho_f^{(97{,}5\,\%)},$$

los cuales definen rangos probabilísticos realistas para la planificación de actividades de revisión y prueba. Para perfiles no matemáticos, estos intervalos representan un margen de seguridad: un rango dentro del cual es altamente probable que se encuentren los defectos reales.

Este enfoque permite trabajar rigurosamente con la información disponible, preservando la confidencialidad de los datos y, al mismo tiempo, proporcionando un marco matemático sólido para analizar la tasa actual de defectos, evaluar escenarios de reducción y estudiar el impacto de mejoras orientadas a disminuir ρ_f en cada fase del proceso. En términos prácticos, el objetivo final del modelo es reducir de manera sostenida las tasas ρ_f, disminuyendo el riesgo residual para el cliente y mejorando la previsibilidad y calidad del proceso de desarrollo.

5.2. Densidad de Defectos y Simulación de Monte Carlo en el Desarrollo de Software

En esta sección se describe el funcionamiento operativo del de defectos basado en densidad y simulación del Método de Monte Carlo*, así como su implementación computacional en `R`. El objetivo es explicar cómo el modelo es utilizado en la práctica por un analista para estimar rangos de defectos esperados, apoyar la toma de decisiones y controlar la calidad del proceso.

Debido a restricciones de confidencialidad, no se dispone de datos históricos a nivel de defecto individual. En su lugar, el modelo se construye a partir de parámetros estadísticos consolidados (medias y rangos de Densidad de defectos*) que caracterizan el comportamiento histórico del proceso en distintas fases. Este enfoque es consistente con prácticas industriales de Control Estadístico de Procesos*, donde frecuentemente se trabaja con información agregada y parámetros resumidos (Montgomery, 2019, Cap. 3, pp. 121–188).

El modelo se apoya en la definición de Densidad de defectos* por iteración*,

$$\rho_i = \frac{d_i}{e_i},$$

y utiliza simulación del Método de Monte Carlo* para representar la variabilidad natural del proceso bajo condiciones normales de operación.

5.2.1. Funcionamiento operativo del modelo

Desde el punto de vista del usuario, el modelo sigue una secuencia lógica claramente definida:

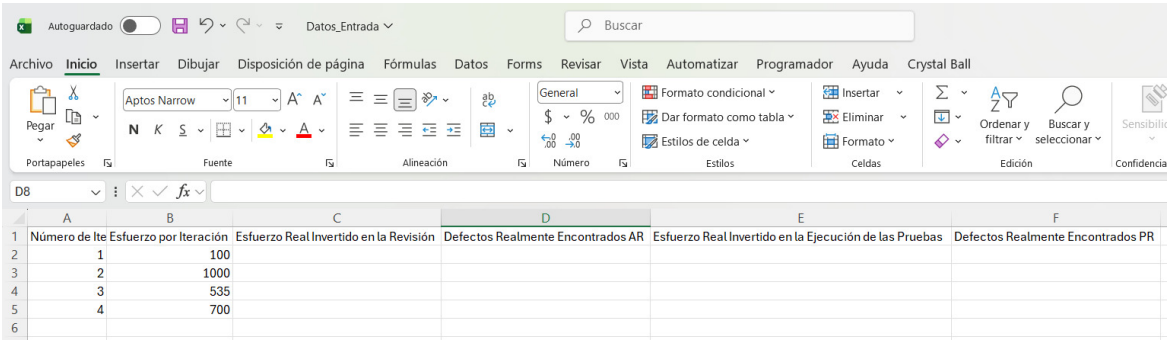
1. Se fijan parámetros estadísticos de Densidad de defectos* para cada fase del proceso (Revisión de Producto, Análisis y Revisión, Pruebas), obtenidos a partir de conocimiento histórico.

2. A partir de dichos parámetros, se generan simulaciones del Método de Monte Carlo* mediante distribuciones normales utilizando la función `rnorm()` en `R`.

3. Se construyen distribuciones empíricas y se calculan intervalos de confianza del 95 % para la Densidad de defectos*.

4. Los intervalos de densidad se transforman en rangos de defectos esperados por iteración* multiplicándolos por el Esfuerzo planeado*.

Este procedimiento permite responder a una pregunta clave en la operación: *¿qué cantidad de defectos es estadísticamente esperable dado el esfuerzo asignado a una iteración*, y cuándo un resultado debe considerarse atípico?*



	A	B	C	D	E	F
1	Número de Ite	Esfuerzo por Iteración	Esfuerzo Real Invertido en la Revisión	Defectos Realmente Encontrados AR	Esfuerzo Real Invertido en la Ejecución de las Pruebas	Defectos Realmente Encontrados PR
2	1	100				
3	2	1000				
4	3	535				
5	4	700				
6						

Gráfico 5.1: Datos Entrada

5.2.2. Implementación computacional en R

El modelo se implementa directamente mediante el código en `R` del Apendice A.2. presentado en este trabajo, el cual cumple una función estrictamente operativa: ejecutar el modelo matemático formulado y producir resultados reproducibles.

En particular, el script realiza las siguientes tareas:

- Simula distribuciones de Densidad de defectos* para cada fase mediante llamadas a `rnorm()`, utilizando medias y desviaciones estándar definidas a partir de rangos históricos.
- Calcula los percentiles 2.5 % y 97.5 % para obtener intervalos de confianza del 95 %.
- Aplica dichos intervalos a los valores de esfuerzo por iteración* contenidos en el archivo `Datos_Entrada.xlsx`.
- Ajusta las estimaciones de defectos en fases posteriores cuando existen defectos reales detectados en fases tempranas, evitando la doble contabilización.

De este modo, el código en `R` no introduce nuevos supuestos estadísticos, sino que materializa computacionalmente el modelo y permite su uso sistemático en contextos reales.

5.2.3. Ejemplo aplicado del uso del modelo

Considérese una iteración* con un Esfuerzo planeado* de $e = 1000$ horas. A partir de la simulación Monte Carlo para la fase de Análisis y Revisión,

el modelo obtiene un intervalo del 95 % para la Densidad de defectos* dado por:

$$\rho_{\text{mín}} = 0{,}0033, \qquad \rho_{\text{máx}} = 0{,}0136.$$

El rango de defectos esperados para dicha iteración* se calcula como:

$$\text{Defectos}_{\text{mín}} = 1000 \times 0{,}0033 = 3{,}3 \approx 3,$$

$$\text{Defectos}_{\text{máx}} = 1000 \times 0{,}0136 = 13{,}6 \approx 14.$$

Esto implica que, bajo condiciones normales del proceso, se espera que el número de defectos detectados en la fase de Análisis y Revisión se sitúe entre 3 y 14. Si el número observado se encuentra dentro de este rango, el comportamiento del proceso se considera estable; si lo excede, se identifica un indicio de causa especial que justifica un análisis más profundo.

Cuando posteriormente se detectan defectos reales en esta fase, el modelo ajusta las estimaciones para la fase de Pruebas, restando los defectos ya corregidos. Este mecanismo permite reflejar el efecto real de la detección temprana y evita sobreestimar la carga de defectos residuales.

5.2.4. Valor del modelo en la gestión del proceso

El principal valor del modelo no reside en predecir un número exacto de defectos, sino en proporcionar rangos probabilísticos consistentes con el comportamiento histórico del proceso. Esta aproximación es coherente con la filosofía del Control Estadístico de Procesos* propuesta originalmente por Shewhart y desarrollada posteriormente en la literatura moderna (Shewhart, 1931, Cap. 1–3, pp. 1–48) (Montgomery, 2019, Cap. 6, 337–394).

Desde una perspectiva operativa, el modelo permite:

- Planificar esfuerzo de revisión y pruebas con base en escenarios realistas.
- Identificar desviaciones significativas que requieren análisis causal.
- Evaluar el impacto de mejoras de proceso en la reducción de la Densidad de defectos*.

- Apoyar decisiones de calidad con evidencia cuantitativa y reproducible.

En conjunto, el modelo constituye una herramienta práctica para integrar métodos estadísticos avanzados en la gestión cotidiana de la calidad del software.

5.3. Impacto del Modelo Predictivo en la Disminución de Defectos en Producción

La implementación del de Densidad de defectos* basado en simulación de Monte Carlo no se limitó a un ejercicio analítico, sino que derivó en acciones operativas concretas que permitieron mejorar de manera medible la calidad del software entregado a producción. En esta sección se presenta el impacto observado, tanto en términos cuantitativos como cualitativos, como resultado de la adopción del modelo.

5.3.1. Contexto inicial y problemática

Previo a la implementación del modelo, la organización presentaba una densidad promedio de defectos en producción elevada, lo cual se traducía en retrabajos frecuentes, afectaciones a los tiempos de entrega y una percepción negativa por parte de los clientes. En términos cuantitativos, la tasa observada era:

$$\rho_{\text{antes}} = 0{,}045 \quad \text{defectos/hora.}$$

Este valor indicaba que, por cada hora efectiva de Esfuerzo invertido* en el desarrollo y mantenimiento del software, se generaba una cantidad significativa de defectos que alcanzaban etapas productivas, reflejando deficiencias en la detección temprana y en la planificación de actividades de revisión y pruebas.

5.3.2. Acciones derivadas de la aplicación del modelo

La aplicación del modelo permitió transformar la información estadística en decisiones operativas específicas. Entre las acciones más relevantes se encuentran:

- Ajuste del esfuerzo asignado a actividades de Revisión de Producto y Análisis y Revisión, utilizando los intervalos de confianza obtenidos mediante simulación Monte Carlo.
- Priorización de revisiones tempranas en iteraciones con rangos elevados de defectos esperados, de acuerdo con la densidad simulada.
- Uso sistemático de rangos probabilísticos de defectos como criterio de alerta para identificar desviaciones anómalas del proceso.
- Retroalimentación estructurada hacia los equipos de desarrollo sobre la relación entre Esfuerzo invertido* y defectos detectados, fortaleciendo la disciplina del proceso.

Estas acciones permitieron intervenir el proceso antes de que los defectos alcanzaran producción, alineando la planeación operativa con el comportamiento estadístico real del proceso.

5.3.3. Resultados cuantitativos

Posterior a la implementación del modelo y a los ajustes de proceso derivados, la densidad promedio de defectos observada en producción se redujo de manera significativa a:

$$\rho_{\text{después}} = 0{,}025 \quad \text{defectos/hora.}$$

La mejora relativa se cuantifica como:

$$\Delta\,\% = \frac{\rho_{\text{antes}} - \rho_{\text{después}}}{\rho_{\text{antes}}} \times 100 = 44{,}4\,\%.$$

Esta reducción evidencia que el modelo no solo permitió estimar defectos de forma anticipada, sino que contribuyó directamente a disminuir la cantidad de errores que alcanzan entornos productivos.

5.3.4. Impacto cualitativo y organizacional

Además de la mejora cuantitativa, la adopción del modelo generó beneficios cualitativos relevantes:

- Se fortaleció la cultura de prevención, desplazando el enfoque reactivo de corrección tardía por uno proactivo de detección temprana.
- Se incrementó la confianza del cliente al reducir incidencias en producción y estabilizar la calidad del servicio.
- Los equipos adquirieron mayor claridad sobre el impacto de su esfuerzo en la calidad final del producto, mejorando la toma de decisiones técnicas.
- El proceso de calidad se volvió más transparente y defendible, al estar respaldado por evidencia estadística y no únicamente por juicio experto.

En conjunto, estos resultados demuestran que el de Densidad de defectos* constituye una herramienta efectiva para conectar métodos estadísticos formales con mejoras reales en la operación, logrando impactos medibles tanto en la calidad del software como en la madurez del proceso organizacional.

Capítulo 6

Conclusiones

"There are three kinds of lies:
lies, damned lies, and statistics."
— Mark Twain

6.1. Conclusiones generales

A lo largo de esta tesis se demostró que la aplicación rigurosa de métodos estadísticos a las métricas de velocidad y defectos permite transformar problemas operativos recurrentes del desarrollo de software en procesos cuantificables, controlables y mejorables. En particular, se evidenció que el uso sistemático de Estadística Descriptiva*, inferencial y Control Estadístico de Procesos* aporta un marco sólido para comprender la variabilidad inherente al desarrollo y distinguirla de desviaciones anómalas que impactan directamente en el desempeño organizacional.

La métrica de velocidad, definida como la relación entre el esfuerzo planificado y el esfuerzo realmente devengado, se consolidó como un indicador clave para evaluar la precisión del proceso de estimación. El análisis estadístico permitió validar su comportamiento aproximadamente normal bajo condiciones estables, lo cual justificó el uso de herramientas paramétricas y cartas de control. A través de la identificación iterativa de causas especiales de variación y el ajuste progresivo de los límites de control, fue posible estabilizar el proceso y mejorar su capacidad predictiva.

Desde el punto de vista operativo, los resultados obtenidos muestran un impacto tangible: la desviación promedio en los tiempos de entrega de los proyectos se redujo de un valor inicial cercano al 35 % a aproximadamente 15 %. Esta mejora no solo es estadísticamente significativa, sino también operacionalmente relevante, al reflejar una mayor confiabilidad en los compromisos de entrega y una reducción sustancial de la incertidumbre en la gestión de proyectos.

En paralelo, el análisis de la métrica de defectos, abordada mediante el concepto de Densidad de defectos* y apoyada en simulación Monte Carlo, permitió modelar escenarios realistas aun en ausencia de datos históricos detallados, utilizando únicamente parámetros estadísticos consolidados. Este enfoque probabilístico facilitó la estimación de rangos esperados de defectos y esfuerzo en distintas fases del proceso, reforzando la toma de decisiones preventivas y la asignación eficiente de recursos de calidad.

La combinación de ambas métricas —velocidad y defectos— puso de manifiesto la necesidad de un enfoque equilibrado entre eficiencia y calidad. Los resultados confirman que mejorar la precisión en la estimación y controlar la variabilidad del proceso contribuye directamente a la reducción de defectos en producción, fortaleciendo la relación entre desempeño del equipo y calidad del producto resultante.

Finalmente, este trabajo demuestra que la estadística aplicada no debe entenderse como un ejercicio académico aislado, sino como una herramienta estratégica que, integrada al conocimiento del negocio, habilita ciclos de mejora continua basados en evidencia cuantitativa confiable.

6.2. Recomendaciones

6.2.1. Consolidación de una cultura de control estadístico

Se recomienda que las organizaciones adopten de manera formal el Control Estadístico de Procesos* como parte de su modelo de gestión. Esto implica no solo la recolección sistemática de métricas, sino su análisis continuo mediante cartas de control, Pruebas de hipótesis* y modelos probabilísticos que permitan distinguir entre variación común y causas

especiales.

6.2.2. Monitoreo continuo y recalibración del modelo

Dado que los procesos de desarrollo evolucionan, se recomienda recalibrar periódicamente los parámetros estadísticos utilizados en los modelos, incorporando nueva información y validando supuestos como normalidad y estabilidad del proceso. Esto asegura que los modelos mantengan su validez a lo largo del tiempo.

6.3. Limitaciones del estudio

Este estudio reconoce limitaciones inherentes al contexto organizacional analizado. En particular, el uso de datos confidenciales y agregados restringió el acceso a información histórica detallada de defectos, lo que motivó el uso de parámetros estadísticos en lugar de observaciones completas. Asimismo, los resultados obtenidos son representativos del contexto específico de la organización estudiada, por lo que su generalización debe realizarse con cautela.

6.4. Líneas de investigación futura

A partir de los resultados obtenidos, se identifican diversas líneas de investigación que pueden derivarse de este trabajo:

- Extender el modelo incorporando técnicas de control multivariado que analicen simultáneamente velocidad, defectos y otras métricas de proceso.
- Explorar modelos de simulación alternativos (por ejemplo, distribuciones no normales o modelos bayesianos) para contextos donde la normalidad no sea un supuesto válido.
- Integrar variables organizacionales adicionales, como experiencia del equipo o complejidad técnica, para enriquecer los modelos predictivos.

- Analizar el impacto del Control Estadístico de Procesos* en entornos ágiles a gran escala y servicios administrados.

6.5. Conclusión final

En conclusión, esta tesis demuestra que la aplicación disciplinada de métodos estadísticos a las métricas de velocidad y defectos constituye un pilar fundamental para la mejora continua en proyectos de desarrollo de software. La evidencia empírica muestra que es posible reducir de manera significativa la variabilidad, mejorar la precisión de las estimaciones y disminuir la Densidad de defectos* en producción cuando se combinan técnicas estadísticas con conocimiento del dominio.

La adopción de este enfoque no solo mejora indicadores cuantitativos, sino que fortalece la madurez organizacional, promueve decisiones basadas en datos y sienta las bases para un desarrollo de software más predecible, eficiente y orientado a la calidad.

Glosario

* Los términos marcados con asterisco dentro de la tesis se encuentran definidos en este glosario.

Analista de Métricas es el profesional encargado de recolectar, procesar y evaluar datos cuantitativos para medir el desempeño y apoyar la toma de decisiones estratégicas. 2

ANOVA técnica estadística utilizada para comparar las medias de tres o más grupos y determinar si existen diferencias significativas entre ellos. 51, 53, 57

backlog es la lista priorizada y dinámica de trabajo pendiente que describe todo lo que el producto necesita para generar valor, como funcionalidades, mejoras, correcciones y tareas técnicas, y que sirve como fuente única de trabajo para el equipo. 30

Control Estadístico de Procesos metodología basada en técnicas estadísticas para monitorear, controlar y mejorar la estabilidad y desempeño de un proceso a lo largo del tiempo. 14, 16, 42, 53, 56, 65, 67, 71, 72, 74

defectos son fallas, errores o desviaciones del comportamiento esperado del sistema, originadas en el diseño, el código, los requisitos o la implementación y las pruebas, que provocan resultados incorrectos, inesperados o no conformes con las especificaciones. v, 2–6, 12

Densidad de defectos métrica que relaciona el número de defectos detectados con el tamaño o esfuerzo del producto desarrollado. 25, 59–61, 63, 65–68, 70, 72, 74

Densidad de esfuerzo relación entre el esfuerzo invertido y una unidad de trabajo o funcionalidad desarrollada. 26

Desarrollo Ágil enfoque de desarrollo de software basado en iteraciones cortas, colaboración continua y adaptación flexible a cambios en los requisitos. 7, 9, 12

Desviación Estándar medida estadística que cuantifica la dispersión o variabilidad de un conjunto de datos respecto a su media. 15, 18, 19, 40

Distribución normal distribución probabilística continua caracterizada por su forma de campana simétrica definida por una media y una desviación estándar. 34–36, 38–41, 43, 63

Esfuerzo devengado cantidad de esfuerzo que ha sido efectivamente consumido o ejecutado durante el desarrollo de una actividad. 2, 21, 22, 29, 31, 32, 35, 40, 53, 54

Esfuerzo invertido cantidad total de recursos, generalmente medidos en horas de trabajo, dedicados a una actividad o proyecto. 26, 59, 61, 68, 69

Esfuerzo planeado cantidad estimada de horas o recursos asignados para completar una actividad o iteración. 2, 21–23, 29, 31, 32, 35, 36, 40, 65, 66

Estadística Descriptiva rama de la estadística orientada a resumir y describir las características principales de un conjunto de datos. 18, 71

Ingeniero de Datos es el profesional que diseña, construye y mantiene sistemas y procesos para recolectar, almacenar y procesar datos de forma eficiente y segura. 2

Intervalo de confianza rango de valores calculado estadísticamente dentro del cual se espera que se encuentre un parámetro poblacional con un nivel de confianza determinado. 25

iteración ciclo de trabajo de duración fija en metodologías ágiles durante el cual se desarrolla y entrega un incremento funcional del software. 12, 19–21, 25, 26, 29, 30, 40, 61, 65–67

Jira es un software de gestión de proyectos y seguimiento de incidencias, usado principalmente para planear, organizar y dar seguimiento al trabajo en equipos de desarrollo y TI. 21

Media medida de tendencia central que representa el promedio aritmético de un conjunto de datos. 13, 15, 18–20, 35, 36, 38, 40, 41, 43, 53

Mediana valor central de un conjunto de datos ordenados que divide la distribución en dos partes iguales. 18, 19

Método de Monte Carlo conjunto de técnicas de simulación que utilizan números aleatorios para modelar fenómenos complejos y estimar resultados probabilísticos. 21, 24, 39, 54, 60, 63–65

Métodos Estadísticos conjunto de técnicas matemáticas utilizadas para recolectar, organizar, analizar e interpretar datos con el fin de apoyar la toma de decisiones. v, 2–4

Prueba de Bartlett prueba estadística empleada para verificar si varias muestras poseen varianzas iguales. 51, 53, 57

Prueba de Shapiro–Wilk prueba estadística utilizada para evaluar si un conjunto de datos sigue aproximadamente una distribución normal. 36

Pruebas de hipótesis procedimientos estadísticos utilizados para evaluar afirmaciones sobre parámetros poblacionales a partir de datos muestrales. 51, 72

riesgos son eventos o condiciones inciertas que, de ocurrir, pueden afectar negativamente el alcance, tiempo, costo o calidad de un proyecto de tecnología de información. v, 11, 14, 31, 57

SCRUM es un marco de trabajo ágil para gestionar y desarrollar productos complejos mediante iteraciones cortas, colaboración continua y mejora incremental. v, 1, 8, 29

Sprint es un ciclo de trabajo de duración fija en la metodología ágil, destinado a desarrollar y entregar un incremento funcional del producto. Usualmente dura dos semanas. v, 12

Story Points son una unidad relativa de estimación utilizada en metodologías ágiles (especialmente Scrum) para medir el esfuerzo total requerido para completar una historia de usuario, considerando de forma conjunta su complejidad, volumen de trabajo, riesgos e incertidumbre, en lugar de tiempo absoluto. 12

Tasa de defectos indicador que expresa la cantidad de defectos por unidad de esfuerzo, tiempo o tamaño del software. 13, 14, 16, 20, 61

Teorema Central del Límite principio estadístico que establece que la distribución de las medias muestrales tiende a ser normal cuando el tamaño de la muestra es suficientemente grande. 34

Valor-p probabilidad de obtener un resultado al menos tan extremo como el observado, suponiendo que la hipótesis nula es verdadera. 51, 53

Variable aleatoria variable cuyo valor depende del resultado de un fenómeno o experimento aleatorio. 63

Varianza medida estadística que representa el promedio de las desviaciones al cuadrado respecto a la media. 18–20, 35, 36, 51, 53

Apéndice A

Códigos en lenguaje R

A.1. Velocidad

```
# Librerías a utilizar.
library(data.table)
library(ggplot2)
library(readxl)
library(rriskDistributions)
library(dplyr)

# Datos.
data_v2 <- read_excel("data_velocidad.xlsx", sheet = "Datos
   Limpios")
data_v2 <- as.data.table(data_v2)

# Análisis exploratorio de datos.
# Disribución de datos y medidas de tendencia central, así
   como de dispersión.
# Análisis de Distribución, se valida que se acepta la
   hipótesis de que se distribuyen los datos normal y
   calculamos los parámetros.
fit.cont(data_v2$Velocidad)

# Contruimos gráfica de control en repetidas ocaciones para
   decidir a partir de negocio cuales son las más optimas y
   hasta que desaparezcan las variaciones especiales en la
   carta.
# Cálculo de límites de control.
media <- mean(data_v2$Velocidad)
limite_superior <- media + 3 * sd(data_v2$Velocidad)
limite_inferior <- media - 3 * sd(data_v2$Velocidad)
```

```
22
23 # Data para gráfico.
24 FC <- subset(data_v2, Velocidad > limite_superior |
      Velocidad < limite_inferior)
25 Incidencias <- 1:length(data_v2$Velocidad)
26 Name <- data_v2$`ID Incidencia`
27 idx <- Name %in% FC$`ID Incidencia`
28 Name[!idx] <- NA
29
30 carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
      Velocidad, label = Name)) + geom_point(aes(colour =
      Velocidad > limite_inferior & Velocidad < limite_superior
      ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
      col="deepskyblue4") + labs(title = "Carta de Control para
      Valores Individuales: Velocidad ") +
      geom_hline(yintercept =  c(limite_superior,
      limite_inferior, media),
      col=c("darkred","darkred","dodgerblue4"), linetype =
      "dashed") + annotate("text", x = length(Incidencias), y =
      limite_superior + 0.2, label = "LCS = 1.83", col =
      "darkred", size = 3) + annotate("text", x =
      length(Incidencias), y = limite_inferior - 0.2, label =
      "LCI = 0.14", col= "darkred", size = 3) +
      annotate("text", x = length(Incidencias) + 30, y = media
      + 0.05, label = "LC = 0.99", col="dodgerblue4", size = 3)
      +  scale_x_continuous(breaks = seq(1,
      length(Incidencias), 50)) + scale_color_manual(values =
      c("red", "deepskyblue4")) +geom_text(vjust = 2, col
      ="red", size = 2) + theme_minimal()
31
32 # Mostrar la carta de control.
33 print(carta_control)
34
35 # Eliminamos el req-195.
36 data_v2 <- data_v2[-195]
37
38 # Volvemos a graficar.
39 # Cálculo de límites de control.
40 media <- mean(data_v2$Velocidad)
41 limite_superior <- media + 3 * sd(data_v2$Velocidad)
42 limite_inferior <- media - 3 * sd(data_v2$Velocidad)
43
44 # Data para gráfico.
45 FC <- subset(data_v2, Velocidad > limite_superior |
      Velocidad < limite_inferior)
46 Incidencias <- 1:length(data_v2$Velocidad)
```

```
Name <- data_v2$`ID Incidencia`
idx <- Name %in% FC$`ID Incidencia`
Name[!idx] <- NA

carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
   Velocidad, label = Name)) + geom_point(aes(colour =
   Velocidad > limite_inferior & Velocidad < limite_superior
   ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
   col="deepskyblue4") + labs(title = "Carta de Control para
   Valores Individuales: Velocidad ") +
   geom_hline(yintercept =  c(limite_superior,
   limite_inferior, media),
   col=c("darkred","darkred","dodgerblue4"), linetype =
   "dashed") + annotate("text", x = length(Incidencias), y =
   limite_superior + 0.2, label = "LCS = 1.81", col =
   "darkred", size = 3) + annotate("text", x =
   length(Incidencias), y = limite_inferior - 0.2, label =
   "LCI = 0.15", col= "darkred", size = 3) +
   annotate("text", x = length(Incidencias) + 30, y = media
   + 0.05, label = "LC = 0.98", col="dodgerblue4", size = 3)
   +  scale_x_continuous(breaks = seq(1,
   length(Incidencias), 50)) + scale_color_manual(values =
   c("deepskyblue4", "red")) +geom_text(vjust = 2, col
   ="red", size = 2) + theme_minimal()

# Mostrar la carta de control.
print(carta_control)

# Reducimos el número de sigmas a 2.
# Volvemos a gráficar.
# Cálculo de límites de control.
media <- mean(data_v2$Velocidad)
limite_superior <- media + 2 * sd(data_v2$Velocidad)
limite_inferior <- media - 2 * sd(data_v2$Velocidad)

# Data para gráfico
FC <- subset(data_v2, Velocidad > limite_superior |
   Velocidad < limite_inferior)
Incidencias <- 1:length(data_v2$Velocidad)
Name <- data_v2$`ID Incidencia`
idx <- Name %in% FC$`ID Incidencia`
Name[!idx] <- NA

carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
   Velocidad, label = Name)) + geom_point(aes(colour =
   Velocidad > limite_inferior & Velocidad < limite_superior
```

```
   ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
   col="deepskyblue4") + labs(title = "Carta de Control para
   Valores Individuales: Velocidad ") +
   geom_hline(yintercept =  c(limite_superior,
   limite_inferior, media),
   col=c("darkred","darkred","dodgerblue4"), linetype =
   "dashed") + annotate("text", x = length(Incidencias), y =
   limite_superior + 0.2, label = "LCS = 1.54", col =
   "darkred", size = 3) + annotate("text", x =
   length(Incidencias), y = limite_inferior - 0.2, label =
   "LCI = 0.43", col= "darkred", size = 3) +
   annotate("text", x = length(Incidencias) + 30, y = media
   + 0.05, label = "LC = 0.98", col="dodgerblue4", size = 3)
   +  scale_x_continuous(breaks = seq(1,
   length(Incidencias), 50)) + scale_color_manual(values =
   c("red", "deepskyblue4")) +geom_text(vjust = 2, col
   ="red", size = 2) + theme_minimal()

# Mostrar la carta de control.
print(carta_control)

# Filtramos las causas especiales de variación.
# Encontrar los ID Incidencia comunes entre data_v2 y FC.
id_comunes <- intersect(data_v2$`ID Incidencia`, FC$`ID
   Incidencia`)

# Filtrar data_v2 para excluir los ID Incidencia comunes.
data_v2 <- data_v2[!data_v2$`ID Incidencia` %in%
   id_comunes, ]

# Volvemos a gráficar.
# Cálculo de límites de control.
media <- mean(data_v2$Velocidad)
limite_superior <- media + 2 * sd(data_v2$Velocidad)
limite_inferior <- media - 2 * sd(data_v2$Velocidad)

# Data para gráfico.
FC <- subset(data_v2, Velocidad > limite_superior |
   Velocidad < limite_inferior)
Incidencias <- 1:length(data_v2$Velocidad)
Name <- data_v2$`ID Incidencia`
idx <- Name %in% FC$`ID Incidencia`
Name[!idx] <- NA

carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
   Velocidad, label = Name)) + geom_point(aes(colour =
```

```
   Velocidad > limite_inferior & Velocidad < limite_superior
   ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
   col="deepskyblue4") + labs(title = "Carta de Control para
   Valores Individuales: Velocidad ") +
   geom_hline(yintercept =  c(limite_superior,
   limite_inferior, media),
   col=c("darkred","darkred","dodgerblue4"), linetype =
   "dashed") + annotate("text", x = length(Incidencias), y =
   limite_superior + 0.2, label = "LCS = 1.5", col =
   "darkred", size = 3) + annotate("text", x =
   length(Incidencias), y = limite_inferior - 0.2, label =
   "LCI = 0.49", col= "darkred", size = 3) +
   annotate("text", x = length(Incidencias) + 30, y = media
   + 0.05, label = "LC = 0.99", col="dodgerblue4", size = 3)
   +  scale_x_continuous(breaks = seq(1,
   length(Incidencias), 50)) + scale_color_manual(values =
   c("red", "deepskyblue4")) +geom_text(vjust = 2, col
   ="red", size = 2) + theme_minimal()
96
97 # Mostrar la carta de control.
98 print(carta_control)
99
100 # Filtramos las causas especiales de variación.
101 # Encontrar los ID Incidencia comunes entre data_v2 y FC.
102 id_comunes <- intersect(data_v2$`ID Incidencia`, FC$`ID
   Incidencia`)
103
104 # Filtrar data_v2 para excluir los ID Incidencia comunes.
105 data_v2 <- data_v2[!data_v2$`ID Incidencia` %in%
   id_comunes, ]
106
107 # Volvemos a gráficar.
108 # Cálculo de límites de control.
109 media <- mean(data_v2$Velocidad)
110 limite_superior <- media + 2 * sd(data_v2$Velocidad)
111 limite_inferior <- media - 2 * sd(data_v2$Velocidad)
112
113 # Data para gráfico.
114 FC <- subset(data_v2, Velocidad > limite_superior |
   Velocidad < limite_inferior)
115 Incidencias <- 1:length(data_v2$Velocidad)
116 Name <- data_v2$`ID Incidencia`
117 idx <- Name %in% FC$`ID Incidencia`
118 Name[!idx] <- NA
119
```

```
120 carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
      Velocidad, label = Name)) + geom_point(aes(colour =
      Velocidad > limite_inferior & Velocidad < limite_superior
      ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
      col="deepskyblue4") + labs(title = "Carta de Control para
      Valores Individuales: Velocidad ") +
      geom_hline(yintercept =  c(limite_superior,
      limite_inferior, media),
      col=c("darkred","darkred","dodgerblue4"), linetype =
      "dashed") + annotate("text", x = length(Incidencias), y =
      limite_superior + 0.2, label = "LCS = 1.47", col =
      "darkred", size = 3) + annotate("text", x =
      length(Incidencias), y = limite_inferior - 0.2, label =
      "LCI = 0.52", col= "darkred", size = 3) +
      annotate("text", x = length(Incidencias) + 30, y = media
      + 0.05, label = "LC = 1", col="dodgerblue4", size = 3) +
      scale_x_continuous(breaks = seq(1, length(Incidencias),
      50)) + scale_color_manual(values = c("red",
      "deepskyblue4")) +geom_text(vjust = 2, col ="red", size =
      2) + theme_minimal()
121
122 # Mostrar la carta de control.
123 print(carta_control)
124
125 # Filtramos las causas especiales de variación.
126 # Encontrar los ID Incidencia comunes entre data_v2 y FC.
127 id_comunes <- intersect(data_v2$`ID Incidencia`, FC$`ID
      Incidencia`)
128
129 # Filtrar data_v2 para excluir los ID Incidencia comunes.
130 data_v2 <- data_v2[!data_v2$`ID Incidencia` %in%
      id_comunes, ]
131
132 # Volvemos a gráficar.
133 # Cálculo de límites de control.
134 media <- mean(data_v2$Velocidad)
135 limite_superior <- media + 2 * sd(data_v2$Velocidad)
136 limite_inferior <- media - 2 * sd(data_v2$Velocidad)
137
138 # Data para gráfico.
139 FC <- subset(data_v2, Velocidad > limite_superior |
      Velocidad < limite_inferior)
140 Incidencias <- 1:length(data_v2$Velocidad)
141 Name <- data_v2$`ID Incidencia`
142 idx <- Name %in% FC$`ID Incidencia`
143 Name[!idx] <- NA
```

```
144
145 carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
      Velocidad, label = Name)) + geom_point(aes(colour =
      Velocidad > limite_inferior & Velocidad < limite_superior
      ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
      col="deepskyblue4") + labs(title = "Carta de Control para
      Valores Individuales: Velocidad ") +
      geom_hline(yintercept =  c(limite_superior,
      limite_inferior, media),
      col=c("darkred","darkred","dodgerblue4"), linetype =
      "dashed") + annotate("text", x = length(Incidencias), y =
      limite_superior + 0.2, label = "LCS = 1.46", col =
      "darkred", size = 3) + annotate("text", x =
      length(Incidencias), y = limite_inferior - 0.2, label =
      "LCI = 0.53", col= "darkred", size = 3) +
      annotate("text", x = length(Incidencias) + 30, y = media
      + 0.05, label = "LC = 1", col="dodgerblue4", size = 3) +
      scale_x_continuous(breaks = seq(1, length(Incidencias),
      50)) + scale_color_manual(values = c("red",
      "deepskyblue4")) +geom_text(vjust = 2, col ="red", size =
      2) + theme_minimal()
146
147 # Mostrar la carta de control.
148 print(carta_control)
149
150 # Filtramos las causas especiales de variación.
151 # Encontrar los ID Incidencia comunes entre data_v2 y FC.
152 id_comunes <- intersect(data_v2$`ID Incidencia`, FC$`ID
      Incidencia`)
153
154 # Filtrar data_v2 para excluir los ID Incidencia comunes.
155 data_v2 <- data_v2[!data_v2$`ID Incidencia` %in%
      id_comunes, ]
156
157 # Volvemos a gráficar.
158 # Cálculo de límites de control.
159 media <- mean(data_v2$Velocidad)
160 limite_superior <- media + 2 * sd(data_v2$Velocidad)
161 limite_inferior <- media - 2 * sd(data_v2$Velocidad)
162
163 # Data para gráfico.
164 FC <- subset(data_v2, Velocidad > limite_superior |
      Velocidad < limite_inferior)
165 Incidencias <- 1:length(data_v2$Velocidad)
166 Name <- data_v2$`ID Incidencia`
167 idx <- Name %in% FC$`ID Incidencia`
```

```
Name[!idx] <- NA

carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
   Velocidad, label = Name)) + geom_point(aes(colour =
   Velocidad > limite_inferior & Velocidad < limite_superior
   ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
   col="deepskyblue4") + labs(title = "Carta de Control para
   Valores Individuales: Velocidad ") +
   geom_hline(yintercept =  c(limite_superior,
   limite_inferior, media),
   col=c("darkred","darkred","dodgerblue4"), linetype =
   "dashed") + annotate("text", x = length(Incidencias), y =
   limite_superior + 0.2, label = "LCS = 1.44", col =
   "darkred", size = 3) + annotate("text", x =
   length(Incidencias), y = limite_inferior - 0.2, label =
   "LCI = 0.54", col= "darkred", size = 3) +
   annotate("text", x = length(Incidencias) + 30, y = media
   + 0.05, label = "LC = 0.99", col="dodgerblue4", size = 3)
   +  scale_x_continuous(breaks = seq(1,
   length(Incidencias), 50)) + scale_color_manual(values =
   c("red", "deepskyblue4")) +geom_text(vjust = 2, col
   ="red", size = 2) + theme_minimal()

# Mostrar la carta de control.
print(carta_control)

# Filtramos las causas especiales de variación.
# Encontrar los ID Incidencia comunes entre data_v2 y FC.
id_comunes <- intersect(data_v2$`ID Incidencia`, FC$`ID
   Incidencia`)

# Filtrar data_v2 para excluir los ID Incidencia comunes.
data_v2 <- data_v2[!data_v2$`ID Incidencia` %in%
   id_comunes, ]

# Volvemos a gráficar.
# Cálculo de límites de control.
media <- mean(data_v2$Velocidad)
limite_superior <- media + 2 * sd(data_v2$Velocidad)
limite_inferior <- media - 2 * sd(data_v2$Velocidad)

# Data para gráfico.
FC <- subset(data_v2, Velocidad > limite_superior |
   Velocidad < limite_inferior)
Incidencias <- 1:length(data_v2$Velocidad)
Name <- data_v2$`ID Incidencia`
```

```
idx <- Name %in% FC$`ID Incidencia`
Name[!idx] <- NA

carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
   Velocidad, label = Name)) + geom_point(aes(colour =
   Velocidad > limite_inferior & Velocidad < limite_superior
   ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
   col="deepskyblue4") + labs(title = "Carta de Control para
   Valores Individuales: Velocidad ") +
   geom_hline(yintercept =  c(limite_superior,
   limite_inferior, media),
   col=c("darkred","darkred","dodgerblue4"), linetype =
   "dashed") + annotate("text", x = length(Incidencias), y =
   limite_superior + 0.2, label = "LCS = 1.43", col =
   "darkred", size = 3) + annotate("text", x =
   length(Incidencias), y = limite_inferior - 0.2, label =
   "LCI = 0.55", col= "darkred", size = 3) +
   annotate("text", x = length(Incidencias) + 30, y = media
   + 0.05, label = "LC = 0.99", col="dodgerblue4", size = 3)
   +  scale_x_continuous(breaks = seq(1,
   length(Incidencias), 50)) + scale_color_manual(values =
   c("red", "deepskyblue4")) +geom_text(vjust = 2, col
   ="red", size = 2) + theme_minimal()

# Mostrar la carta de control.
print(carta_control)

# Filtramos las causas especiales de variación.
# Encontrar los ID Incidencia comunes entre data_v2 y FC.
id_comunes <- intersect(data_v2$`ID Incidencia`, FC$`ID
   Incidencia`)

# Filtrar data_v2 para excluir los ID Incidencia comunes.
data_v2 <- data_v2[!data_v2$`ID Incidencia` %in%
   id_comunes, ]

# Volvemos a gráficar.
# Cálculo de límites de control.
media <- mean(data_v2$Velocidad)
limite_superior <- media + 2 * sd(data_v2$Velocidad)
limite_inferior <- media - 2 * sd(data_v2$Velocidad)

# Data para gráfico.
FC <- subset(data_v2, Velocidad > limite_superior |
   Velocidad < limite_inferior)
Incidencias <- 1:length(data_v2$Velocidad)
```

```
Name <- data_v2$`ID Incidencia`
idx <- Name %in% FC$`ID Incidencia`
Name[!idx] <- NA

carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
   Velocidad, label = Name)) + geom_point(aes(colour =
   Velocidad > limite_inferior & Velocidad < limite_superior
   ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
   col="deepskyblue4") + labs(title = "Carta de Control para
   Valores Individuales: Velocidad ") +
   geom_hline(yintercept =  c(limite_superior,
   limite_inferior, media),
   col=c("darkred","darkred","dodgerblue4"), linetype =
   "dashed") + annotate("text", x = length(Incidencias), y =
   limite_superior + 0.2, label = "LCS = 1.42", col =
   "darkred", size = 3) + annotate("text", x =
   length(Incidencias), y = limite_inferior - 0.2, label =
   "LCI = 0.55", col= "darkred", size = 3) +
   annotate("text", x = length(Incidencias) + 30, y = media
   + 0.05, label = "LC = 0.99", col="dodgerblue4", size = 3)
   +  scale_x_continuous(breaks = seq(1,
   length(Incidencias), 50)) + scale_color_manual(values =
   c("red", "deepskyblue4")) +geom_text(vjust = 2, col
   ="red", size = 2) + theme_minimal()

# Mostrar la carta de control.
print(carta_control)

# Filtramos las causas especiales de variación.
# Encontrar los ID Incidencia comunes entre data_v2 y FC.
id_comunes <- intersect(data_v2$`ID Incidencia`, FC$`ID
   Incidencia`)

# Filtrar data_v2 para excluir los ID Incidencia comunes.
data_v2 <- data_v2[!data_v2$`ID Incidencia` %in%
   id_comunes, ]

# Volvemos a gráficar.
# Cálculo de límites de control.
media <- mean(data_v2$Velocidad)
limite_superior <- media + 2 * sd(data_v2$Velocidad)
limite_inferior <- media - 2 * sd(data_v2$Velocidad)

# Data para gráfico.
FC <- subset(data_v2, Velocidad > limite_superior |
   Velocidad < limite_inferior)
```

```
Incidencias <- 1:length(data_v2$Velocidad)
Name <- data_v2$`ID Incidencia`
idx <- Name %in% FC$`ID Incidencia`
Name[!idx] <- NA

carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
   Velocidad, label = Name)) + geom_point(aes(colour =
   Velocidad > limite_inferior & Velocidad < limite_superior
   ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
   col="deepskyblue4") + labs(title = "Carta de Control para
   Valores Individuales: Velocidad ") +
   geom_hline(yintercept =  c(limite_superior,
   limite_inferior, media),
   col=c("darkred","darkred","dodgerblue4"), linetype =
   "dashed") + annotate("text", x = length(Incidencias), y =
   limite_superior + 0.2, label = "LCS = 1.42", col =
   "darkred", size = 3) + annotate("text", x =
   length(Incidencias), y = limite_inferior - 0.2, label =
   "LCI = 0.55", col= "darkred", size = 3) +
   annotate("text", x = length(Incidencias) + 30, y = media
   + 0.05, label = "LC = 0.99", col="dodgerblue4", size = 3)
   +  scale_x_continuous(breaks = seq(1,
   length(Incidencias), 50)) + scale_color_manual(values =
   c("deepskyblue4", "red")) +geom_text(vjust = 2, col
   ="red", size = 2) + theme_minimal()

# Mostrar la carta de control.
print(carta_control)

# Disminuimos a una 1 sigma.

# Volvemos a gráficar.
# Cálculo de límites de control.
media <- mean(data_v2$Velocidad)
limite_superior <- media +  sd(data_v2$Velocidad)
limite_inferior <- media -  sd(data_v2$Velocidad)

# Data para gráfico.
FC <- subset(data_v2, Velocidad > limite_superior |
   Velocidad < limite_inferior)
Incidencias <- 1:length(data_v2$Velocidad)
Name <- data_v2$`ID Incidencia`
idx <- Name %in% FC$`ID Incidencia`
Name[!idx] <- NA

```

```
265 carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
      Velocidad, label = Name)) + geom_point(aes(colour =
      Velocidad > limite_inferior & Velocidad < limite_superior
      ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
      col="deepskyblue4") + labs(title = "Carta de Control para
      Valores Individuales: Velocidad ") +
      geom_hline(yintercept =  c(limite_superior,
      limite_inferior, media),
      col=c("darkred","darkred","dodgerblue4"), linetype =
      "dashed") + annotate("text", x = length(Incidencias), y =
      limite_superior + 0.2, label = "LCS = 1.20", col =
      "darkred", size = 3) + annotate("text", x =
      length(Incidencias), y = limite_inferior - 0.2, label =
      "LCI = 0.77", col= "darkred", size = 3) +
      annotate("text", x = length(Incidencias) + 30, y = media
      + 0.05, label = "LC = 0.99", col="dodgerblue4", size = 3)
      +  scale_x_continuous(breaks = seq(1,
      length(Incidencias), 50)) + scale_color_manual(values =
      c("red", "deepskyblue4")) +geom_text(vjust = 2, col
      ="red", size = 2) + theme_minimal()
266
267 # Mostrar la carta de control.
268 print(carta_control)
269
270 # Filtramos las causas especiales de variación.
271 # Encontrar los ID Incidencia comunes entre data_v2 y FC.
272 id_comunes <- intersect(data_v2$`ID Incidencia`, FC$`ID
      Incidencia`)
273
274 # Filtrar data_v2 para excluir los ID Incidencia comunes.
275 data_v2 <- data_v2[!data_v2$`ID Incidencia` %in%
      id_comunes, ]
276
277 # Volvemos a gráficar.
278 # Cálculo de límites de control.
279 media <- mean(data_v2$Velocidad)
280 limite_superior <- media +  sd(data_v2$Velocidad)
281 limite_inferior <- media -  sd(data_v2$Velocidad)
282
283 # Data para gráfico.
284 FC <- subset(data_v2, Velocidad > limite_superior |
      Velocidad < limite_inferior)
285 Incidencias <- 1:length(data_v2$Velocidad)
286 Name <- data_v2$`ID Incidencia`
287 idx <- Name %in% FC$`ID Incidencia`
288 Name[!idx] <- NA
```

```
289
290 carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
      Velocidad, label = Name)) + geom_point(aes(colour =
      Velocidad > limite_inferior & Velocidad < limite_superior
      ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
      col="deepskyblue4") + labs(title = "Carta de Control para
      Valores Individuales: Velocidad ") +
      geom_hline(yintercept =  c(limite_superior,
      limite_inferior, media),
      col=c("darkred","darkred","dodgerblue4"), linetype =
      "dashed") + annotate("text", x = length(Incidencias), y =
      limite_superior + 0.2, label = "LCS = 1.10", col =
      "darkred", size = 3) + annotate("text", x =
      length(Incidencias), y = limite_inferior - 0.2, label =
      "LCI = 0.86", col= "darkred", size = 3) +
      annotate("text", x = length(Incidencias) + 30, y = media
      + 0.05, label = "LC = 0.98", col="dodgerblue4", size = 3)
      +  scale_x_continuous(breaks = seq(1,
      length(Incidencias), 50)) + scale_color_manual(values =
      c("red", "deepskyblue4")) +geom_text(vjust = 2, col
      ="red", size = 2) + theme_minimal()
291
292 # Mostrar la carta de control.
293 print(carta_control)
294
295 # Filtramos las causas especiales de variación.
296 # Encontrar los ID Incidencia comunes entre data_v2 y FC.
297 id_comunes <- intersect(data_v2$`ID Incidencia`, FC$`ID
      Incidencia`)
298
299 # Filtrar data_v2 para excluir los ID Incidencia comunes.
300 data_v2 <- data_v2[!data_v2$`ID Incidencia` %in%
      id_comunes, ]
301
302 # Volvemos a gráficar.
303 # Cálculo de límites de control regresamos a dos sigmas.
304 media <- mean(data_v2$Velocidad)
305 limite_superior <- media +  2 * sd(data_v2$Velocidad)
306 limite_inferior <- media -  2 * sd(data_v2$Velocidad)
307
308 # Data para gráfico.
309 FC <- subset(data_v2, Velocidad > limite_superior |
      Velocidad < limite_inferior)
310 Incidencias <- 1:length(data_v2$Velocidad)
311 Name <- data_v2$`ID Incidencia`
312 idx <- Name %in% FC$`ID Incidencia`
```

```
313 Name[!idx] <- NA
314
315 carta_control <- ggplot(data_v2, aes(x = Incidencias , y =
    Velocidad, label = Name)) + geom_point(aes(colour =
    Velocidad > limite_inferior & Velocidad < limite_superior
    ), show.legend = F, shape = 1, size = 1.8) + geom_line(,
    col="deepskyblue4") + labs(title = "Carta de Control para
    Valores Individuales: Velocidad ") +
    geom_hline(yintercept =  c(limite_superior,
    limite_inferior, media),
    col=c("darkred","darkred","dodgerblue4"), linetype =
    "dashed") + annotate("text", x = length(Incidencias), y =
    limite_superior + 0.1, label = "LCS = 1.10", col =
    "darkred", size = 3) + annotate("text", x =
    length(Incidencias), y = limite_inferior - 0.1, label =
    "LCI = 0.84", col= "darkred", size = 3) +
    annotate("text", x = length(Incidencias) + 30, y = media
    + 0.05, label = "LC = 0.97", col="dodgerblue4", size = 3)
    +  scale_x_continuous(breaks = seq(1,
    length(Incidencias), 50)) + scale_color_manual(values =
    c("deepskyblue4", "red")) +geom_text(vjust = 2, col
    ="red", size = 2) + theme_minimal()
316
317 # Mostrar la carta de control.
318 print(carta_control)
319
320
321 #------------ Termina modelo --------
322
323 # Ahora tomamos una muestra del mismo tamaño y vemos si los
    proyectos mejorarón con en media y variación.
324
325 x <- rnorm(507, 0.988, 0.050)
326 write.csv(x, "Simulación_despues.csv")
327
328 # Traemos datos para hipótesis.
329 Data_hip <- read_excel("Data_hip.xlsx")
330 Data_hip <- as.data.table(Data_hip)
331
332 # Separamos data frames.
333 antes <- Data_hip[Grupo == "Antes"]
334 despues <- Data_hip[Grupo == "Despues"]
335
336 # Prueba de media.
337 summary(aov(antes$Velocidad ~ despues$Velocidad))
338
```

```
339 # Prueba de Varianzas.
340 bartlett.test(list(antes$Velocidad, despues$Velocidad))
341
342 # Histograma ambos grupos.
343 # Crear el gráfico de ggplot con dos histogramas
      superpuestos como curvas.
344 ggplot() +
345     geom_histogram(data = antes, aes(x = Velocidad, y =
        ..density..),
346                    fill = "red", alpha = 0.4, bins = 30) +
347   geom_histogram(data = despues, aes(x = Velocidad, y =
      ..density..),
348                  fill = "blue", alpha = 0.4, bins = 30) +
349   labs(title = "Comparación de Procesos",
350        y = "Frecuencia",
351        x = "Velocidad") +
352   theme_minimal()
```

A.2. Defectos

```r
# SIMULACIÓN MONTE CARLO DE DENSIDAD DE DEFECTOS Y ESFUERZO
# Librerías a utilizar.
library(data.table)
library(ggplot2)
library(readxl)

# Se genera una distribución normal simulada de la densidad de esfuerzo
# correspondiente a la fase de Revisión de Producto (RP).
# Los valores se generan con base en la media y desviación estándar de los rangos definidos.
set.seed(42)
Densidad_RP <- rnorm(1000, mean = mean(c(0.020, 0.040)), sd = sd(c(0.020, 0.040)))

# Se calculan los percentiles 2.5% y 97.5%, que conforman el intervalo del 95% de confianza.
lower_bound_RP <- quantile(Densidad_RP, 0.025)
upper_bound_RP <- quantile(Densidad_RP, 0.975)

# Se crea el histograma de la distribución simulada.
hist(Densidad_RP,
     breaks = 20,
     col = "lightblue",
     border = "black",
     main = "Histograma de Densidad de RP en RP\ncon Intervalo del 95% de los Datos",
     xlab = "Densidad de Defectos",
     probability = TRUE)

# Se añaden líneas verticales que marcan los límites del 95% de los datos simulados.
abline(v = lower_bound_RP, col = "red", lwd = 2, lty = 2)
abline(v = upper_bound_RP, col = "green", lwd = 2, lty = 2)

# Se agrega una leyenda que muestra los valores numéricos de los percentiles calculados.
legend("topright", legend = c(paste("Percentil 2.5%:", round(lower_bound_RP, 4)), paste("Percentil 97.5%:", round(upper_bound_RP, 4))), col = c("red", "green"), lwd = 2, lty = 2)

# - - - - - - - - - - - - - - - - - - - - - - - -
```

```r
# Simulación de Densidad de Defectos para la fase de Análisis y Revisión (AR)
s.et.seed(42)
Densidad_AR <- rnorm(1000, mean = mean(c(0.00675, 0.0105)), sd = sd(c(0.00675, 0.0105)))

# Percentiles que representan el intervalo del 95% de confianza.
lower_bound <- quantile(Densidad_AR, 0.025)
upper_bound <- quantile(Densidad_AR, 0.975)

# Visualización del histograma correspondiente a la fase AR.
hist(Densidad_AR,
     breaks = 20,
     col = "lightblue",
     border = "black",
     main = "Histograma de Densidad de Defectos en AR\ncon Intervalo del 95% de los Datos",
     xlab = "Densidad de Defectos",
     probability = TRUE)

# Líneas verticales que marcan el intervalo de confianza.
abline(v = lower_bound, col = "red", lwd = 2, lty = 2)
abline(v = upper_bound, col = "green", lwd = 2, lty = 2)

# Leyenda con valores de percentiles.
legend("topright", legend = c(paste("Percentil 2.5%:", round(lower_bound, 4)), paste("Percentil 97.5%:", round(upper_bound, 4))), col = c("red", "green"), lwd = 2, lty = 2)

# - - - - - - - - - - - - - - - - - - - - - - - - -
# Simulación del esfuerzo en la fase de Pruebas (PR)
set.seed(42)
Densidad_Pruebas <- rnorm(1000, mean = mean(c(0.05, 0.3)), sd = sd(c(0.05, 0.13)))

# Cálculo del intervalo de confianza del 95%.
lower_bound_Pruebas <- quantile(Densidad_Pruebas, 0.025)
upper_bound_Pruebas <- quantile(Densidad_Pruebas, 0.975)

# Histograma del esfuerzo de pruebas.
hist(Densidad_Pruebas,
     breaks = 20,
     col = "lightblue",
     border = "black",
```

```
72      main = "Histograma de Densidad de Defectos en
           Pruebas\ncon Intervalo del 95% de los Datos",
73      xlab = "Densidad de Defectos",
74      probability = TRUE)
75
76  # Líneas de percentiles (intervalo del 95%).
77  abline(v = lower_bound_Pruebas, col = "red", lwd = 2, lty =
       2)
78  abline(v = upper_bound_Pruebas, col = "green", lwd = 2, lty
       = 2)
79
80  # Leyenda con límites de intervalo.
81  legend("topright", legend = c(paste("Percentil 2.5%:",
       round(lower_bound_Pruebas, 4)), paste("Percentil 97.5%:",
       round(upper_bound_Pruebas, 4))), col = c("red", "green"),
       lwd = 2, lty = 2)
82
83  # - - - - - - - - - - - - - - - - - - - - - - - - - -
84  # Simulación de la densidad de defectos en pruebas (fase PR)
85  set.seed(44)
86  Densidad_Pruebas_defectos <- rnorm(1000, mean =
       mean(c(0.002, 0.005)), sd = sd(c(0.002, 0.005)))
87
88  # Los valores negativos se convierten en absolutos para
       evitar inconsistencias.
89  Densidad_Pruebas_defectos <- abs(Densidad_Pruebas_defectos)
90
91  # Cálculo del intervalo del 95%.
92  lower_bound_Pruebas_defectos <-
       quantile(Densidad_Pruebas_defectos, 0.025)
93  upper_bound_Pruebas_defectos <-
       quantile(Densidad_Pruebas_defectos, 0.975)
94
95  # Representación visual de la distribución simulada.
96  hist(Densidad_Pruebas_defectos,
97       breaks = 20,
98       col = "lightblue",
99       border = "black",
100      main = "Histograma de Densidad de Defectos en
            Pruebas\ncon Intervalo del 95% de los Datos",
101      xlab = "Densidad de Defectos",
102      probability = TRUE)
103
104 # Inclusión de líneas de intervalo de confianza.
105 abline(v = lower_bound_Pruebas_defectos, col = "red", lwd =
       2, lty = 2)
```

```
106 abline(v = upper_bound_Pruebas_defectos, col = "green", lwd
      = 2, lty = 2)
107
108 # Leyenda informativa.
109 legend("topright", legend = c(paste("Percentil 2.5%:",
      round(lower_bound_Pruebas_defectos, 4)), paste("Percentil
      97.5%:", round(upper_bound_Pruebas_defectos, 4))), col =
      c("red", "green"), lwd = 2, lty = 2)
110
111 # - - - - - - - - - - - - - - - - - - - - - - - - - - - -
112 # CARGA DE DATOS DE ENTRADA Y CREACIÓN DE VARIABLES DEL
      MODELO
113 Datos_Entrada <- read_excel("Datos_Entrada.xlsx")  # Se lee
      el archivo de entrada.
114 Datos_Entrada <- as.data.table(Datos_Entrada)  # Se
      convierte en data.table para manipulación eficiente.
115
116 # Se inicializan las nuevas columnas que almacenarán los
      valores mínimos y máximos
117 # de esfuerzo y defectos esperados en cada fase del proceso.
118 Esfuerzo_Minimo_de_Revision <- as.numeric(rep(NA,
      length(Datos_Entrada$`Número de Iteración`)))
119 Esfuerzo_Maximo_de_Revision <- as.numeric(rep(NA,
      length(Datos_Entrada$`Número de Iteración`)))
120 Defectos_Minimos_Esperados_AR <- as.numeric(rep(NA,
      length(Datos_Entrada$`Número de Iteración`)))
121 Defectos_Maximos_Esperados_AR <- as.numeric(rep(NA,
      length(Datos_Entrada$`Número de Iteración`)))
122 Esfuerzo_Minimo_de_Ejecucion_de_Pruebas <-
      as.numeric(rep(NA, length(Datos_Entrada$`Número de
      Iteración`)))
123 Esfuerzo_Maximo_de_Ejecucion_de_Pruebas <-
      as.numeric(rep(NA, length(Datos_Entrada$`Número de
      Iteración`)))
124 Defectos_Minimos_Esperados_PR <- as.numeric(rep(NA,
      length(Datos_Entrada$`Número de Iteración`)))
125 Defectos_Maximos_Esperados_PR <- as.numeric(rep(NA,
      length(Datos_Entrada$`Número de Iteración`)))
126
127 # Las columnas se integran al objeto principal
      "Datos_Entrada".
128 Datos_Entrada[, "Esfuerzo Mínimo de Revisión" :=
      Esfuerzo_Minimo_de_Revision]
129 Datos_Entrada[, "Esfuerzo Máximo de Revisión" :=
      Esfuerzo_Maximo_de_Revision]
```

```
130 Datos_Entrada[, "Defectos Mínimos Esperados AR" :=
       Defectos_Minimos_Esperados_AR]
131 Datos_Entrada[, "Defectos Máximos Esperados AR" :=
       Defectos_Maximos_Esperados_AR]
132 Datos_Entrada[, "Esfuerzo Mínimo de Ejecución de Pruebas"
       := Esfuerzo_Minimo_de_Ejecucion_de_Pruebas]
133 Datos_Entrada[, "Esfuerzo Máximo de Ejecución de Pruebas"
       := Esfuerzo_Maximo_de_Ejecucion_de_Pruebas]
134 Datos_Entrada[, "Defectos Mínimos Esperados PR" :=
       Defectos_Minimos_Esperados_PR]
135 Datos_Entrada[, "Defectos Máximos Esperados PR" :=
       Defectos_Maximos_Esperados_PR]
136
137 # - - - - - - - - - - - - - - - - - - - - - - - - - -
138 # MODELO PREDICTIVO DE ESFUERZO Y DEFECTOS
139 # Este primer ciclo calcula el esfuerzo y defectos
       esperados por iteración
140 # con base en las tasas derivadas de los percentiles
       obtenidos en las simulaciones.
141 for (j in 1:length(Datos_Entrada$`Número de Iteración`) ) {
142
143   if (is.na(Datos_Entrada[j, 2]) != T) {
144
145     # Cálculo del esfuerzo en horas para revisión de
         producto (RP).
146     Datos_Entrada[j, 7] <- round(Datos_Entrada[j, 2] *
         0.020, 0)
147     Datos_Entrada[j, 8] <- round(Datos_Entrada[j, 2] *
         0.040, 0)
148
149     # Cálculo de defectos esperados en la fase AR.
150     Datos_Entrada[j, 9] <- round(Datos_Entrada[j, 2] *
         0.00675, 0)
151     Datos_Entrada[j, 10] <- round(Datos_Entrada[j, 2] *
         0.0105, 0)
152
153     # Cálculo del esfuerzo en horas para ejecución de
         pruebas.
154     Datos_Entrada[j, 11] <- round(Datos_Entrada[j, 2] *
         0.005, 0)
155     Datos_Entrada[j, 12] <- round(Datos_Entrada[j, 2] *
         0.3, 0)
156
157     # Cálculo de defectos esperados en la fase de pruebas
         (PR).
```

```
158     Datos_Entrada[j, 13] <- round(Datos_Entrada[j, 2] *
           0.002, 0)
159     Datos_Entrada[j, 14] <- round(Datos_Entrada[j, 2] *
           0.005, 0)
160   }
161 }
162
163 # - - - - - - - - - - - - - - - - - - - - - - - - - - -
164 # AJUSTE DEL MODELO CUANDO EXISTEN DEFECTOS REALES EN LA
       FASE AR
165 for (i in 1:length(Datos_Entrada$`Número de Iteración`)) {
166
167   # Si no hay defectos registrados, se calculan los valores
        estándar.
168   if(is.na(Datos_Entrada[i, 4]) == T){
169
170     Datos_Entrada[i, 7] <- round(Datos_Entrada[i, 2] *
           0.020, 0)
171     Datos_Entrada[i, 8] <- round(Datos_Entrada[i, 2] *
           0.040, 0)
172     Datos_Entrada[i, 9] <- round(Datos_Entrada[i, 2] *
           0.00675, 0)
173     Datos_Entrada[i, 10] <- round(Datos_Entrada[i, 2] *
           0.0105, 0)
174     Datos_Entrada[i, 11] <- round(Datos_Entrada[i, 2] *
           0.005, 0)
175     Datos_Entrada[i, 12] <- round(Datos_Entrada[i, 2] *
           0.3, 0)
176     Datos_Entrada[i, 13] <- round(Datos_Entrada[i, 2] *
           0.002, 0)
177     Datos_Entrada[i, 14] <- round(Datos_Entrada[i, 2] *
           0.005, 0)
178   } else {
179
180     # Si existen defectos detectados en AR, se ajustan los
          valores de PR
181     # restando los defectos ya encontrados para evitar
          duplicidad en la estimación.
182     if((Datos_Entrada[i, 13] + Datos_Entrada[i, 9] -
183         Datos_Entrada[i, 4]) < 0){
184
185       Datos_Entrada[i, 13] <- 0
186       Datos_Entrada[i, 14] <- 0
187
188     } else {
189
```

```
190         # Cálculo ajustado considerando los defectos ya
               corregidos en AR.
191         Datos_Entrada[i, 13] <- round(Datos_Entrada[i, 2] *
               0.002, 0) +
192           round(Datos_Entrada[i, 2] * 0.00675, 0) -
                 Datos_Entrada[i, 4]
193
194         Datos_Entrada[i, 14] <- round(Datos_Entrada[i, 2] *
               0.002, 0) +
195           round(Datos_Entrada[i, 2] * 0.0105, 0) -
                 Datos_Entrada[i, 4]
196       }
197     }
198 }
199
200 # - - - - - - - - - - - - - - - - - - - - - - - - - -
201 # EXPORTACIÓN DE RESULTADOS DEL MODELO
202 library(openxlsx)  # Librería para exportar resultados a
       Excel
203
204 # Se genera el archivo de salida con los resultados finales
       del modelo.
205 write.xlsx(Datos_Entrada, file =
       "Datos_Salida_Modelo.xlsx", rowNames = FALSE)
206
207 # Mensaje de confirmación en consola.
208 cat("Archivo 'Datos_Salida_Modelo.xlsx' generado
       exitosamente en el directorio de trabajo:\n",
209     getwd(), "\n")
```

Bibliografía

Chatfield, C. (2004). *The Analysis of Time Series: An Introduction*. Chapman & Hall/CRC, 6 edition.

Devore, J. L. (2016). *Probability and Statistics for Engineering and the Sciences*. Cengage Learning, 9 edition.

Fenton, N. E. and Pfleeger, S. L. (1997). *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Company, Boston, 2 edition.

Forsgren, N., Humble, J., and Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps*. IT Revolution Press.

ISO/IEC (2011). Iso/iec 25010: Systems and software engineering — systems and software quality requirements and evaluation.

Kan, S. H. (2002). *Metrics and Models in Software Quality Engineering*. Addison-Wesley, Boston, 2 edition.

Kaplan, R. S. and Norton, D. P. (1996). *The Balanced Scorecard: Translating Strategy into Action*. Harvard Business School Press.

Law, A. M. (2015). *Simulation Modeling and Analysis*. McGraw-Hill Education, 5 edition.

McConnell, S. (2006). *Software Estimation: Demystifying the Black Art*. Microsoft Press.

Montgomery, D. C. (2019). *Introduction to Statistical Quality Control*. Wiley, 8 edition.

Montgomery, D. C. and Runger, G. C. (2018). *Applied Statistics and Probability for Engineers*. Wiley, 7 edition.

Pressman, R. S. and Maxim, B. R. (2020). *Software Engineering: A*

Practitioner's Approach. McGraw-Hill Education, New York, 9 edition.

Project Management Institute (2021). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. Project Management Institute, 7 edition.

Ross, S. M. (2014). *Introduction to Probability Models*. Academic Press, 11 edition.

Schwaber, K. and Sutherland, J. (2020). The scrum guide. `https://scrumguides.org`. Scrum.org.

Shewhart, W. A. (1931). *Economic Control of Quality of Manufactured Product*. D. Van Nostrand Company, New York.

Sommerville, I. (2016). *Software Engineering*. Pearson, Boston, 10 edition.

Vose, D. (2008). *Risk Analysis: A Quantitative Guide*. Wiley, 3 edition.

Woodall, W. H. (2000). Controversies and contradictions in statistical process control. *Journal of Quality Technology*, 32(4).