UNIVERSIDAD TECNICA FEDERICO SANTA MARIA | DEPARTAMENTO DE INFORMÁTICA

MAGÍSTER EN TECNOLOGÍAS DE LA INFORMACIÓN

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE

“Evaluación de un enfoque de desarrollo de software asistido por IA basado en Spec-Driven Development (SDD) e integrado con Test-Driven Development (TDD)”

ESTEBAN ALEJANDRO ARREDONDO VELÁSQUEZ

TESINA PARA OPTAR AL GRADO
DE MAGÍSTER EN TECNOLOGÍAS DE
LA INFORMACIÓN

Profesor Guía: Marcello Visconti Z.

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☐ Memoria o trabajo de título ☒ Tesis de Postgrado

Título del trabajo: Evaluación de un enfoque de desarrollo de software asistido por IA basado en Spec-Driven Development (SDD) e integrado con Test-Driven Development (TDD)

Nombre del candidato(a): Esteban Alejandro Arredondo Velásquez

Carrera / Grado: Magíster en Tecnologías de la Información.

Campus: San Joaquín **Departamento:** Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, **Marcello Visconti**, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por **(marcar una opción)**:

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 29-10-2026 **Firma:** _____

Estudiante o Candidato(a):

Fecha: 29-10-2026 **Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Evaluación de un enfoque de desarrollo de software asistido por IA basado en Spec-Driven Development (SDD) e integrado con Test-Driven Development (TDD)

Esteban Alejandro Arredondo Velásquez

Compañía del sector asegurador
Santiago, Región Metropolitana

Esteban.arredondo.v@gmail.com

Resumen: El uso de inteligencia artificial en el desarrollo de software ha facilitado la generación automática de código. Sin embargo, cuando se utilizan prompts no estructurados, pueden surgir ambigüedades en los requerimientos que afectan la calidad, trazabilidad y validación del software. Esta investigación evaluó la incorporación de Spec-Driven Development (SDD), implementado mediante OpenSpec e integrado con prácticas de Test-Driven Development (TDD) y con revisión humana de los artefactos generados, en un proceso de desarrollo que anteriormente utilizaba IA sin una estructura formal. Para realizar la evaluación, se compararon dos períodos de desarrollo de un mismo equipo en una organización del sector asegurador: enero a junio de 2025 y enero a junio de 2026. En el primero, el equipo utilizó IA sin especificaciones formales, mientras que en el segundo se incorporó SDD junto con prácticas de TDD. En este último período, las especificaciones se utilizaron como referencia para mantener alineados los requerimientos, las pruebas y el código. El objetivo fue evaluar los cambios observados en la calidad del software, el nivel de retrabajo y la trazabilidad percibida por los desarrolladores entre requerimientos, pruebas e implementación, considerando también su percepción respecto de esta forma de trabajo. Para ello, se realizó un estudio de caso antes/después en un entorno real de desarrollo, utilizando métricas de errores e iteraciones, además de una encuesta al equipo. Los errores en producción disminuyeron de 101 a 82 (−18,8%) con un volumen de tickets estable y, en una muestra de cuatro proyectos de un mismo sistema, la tasa de retrabajo bajó de 15,66% a 9,26% y el tiempo promedio de retrabajo, de 2,26 a 1,71 días hábiles, sin que los proyectos de 2026 fueran de menor complejidad. Los desarrolladores valoraron el enfoque con un promedio de 3,89 sobre 5 y destacaron la claridad de los requerimientos, aunque la relación entre las pruebas automatizadas y los requerimientos fue el aspecto peor evaluado (3,00), por lo que la trazabilidad percibida se respalda solo parcialmente, en línea con una adopción desigual de TDD. Como entre ambos períodos también cambiaron la herramienta de IA y la complejidad de los proyectos, y la muestra es acotada, los resultados muestran una asociación favorable, pero no permiten establecer causalidad. Aun así, aportan evidencia desde un contexto organizacional real para orientar la adopción de enfoques estructurados en el desarrollo asistido por IA.

Palabras clave: Desarrollo asistido por IA; Spec-Driven Development (SDD); Test-Driven Development (TDD); Modelos de lenguaje de gran escala (LLM); Calidad de software; OpenSpec; Trazabilidad de software

1 Introducción

Las herramientas basadas en inteligencia artificial se han incorporado progresivamente a los procesos de desarrollo de software. Sin embargo, cuando las instrucciones entregadas a estos modelos no están claramente definidas, los resultados no siempre se alinean con las necesidades reales del negocio, lo que introduce ambigüedad en los requerimientos y afecta la calidad del software producido.

Esta situación se observó en la organización en que se desarrolló este estudio, donde la inteligencia artificial ya formaba parte habitual del proceso de desarrollo, pero su uso no estaba guiado por una única fuente de verdad. Aunque el equipo disponía de información funcional y técnica suficiente, esta no siempre se encontraba

organizada de una forma que facilitara su uso con herramientas de IA ni la trazabilidad entre requerimientos, pruebas y código, lo que derivó en problemas concretos de calidad, trazabilidad y retrabajo.

Para abordar este problema, este trabajo combinó dos prácticas: el Desarrollo Basado en Especificaciones (SDD), que utiliza especificaciones estructuradas como referencia para diseñar, implementar y validar el software, y el Desarrollo Guiado por Pruebas (TDD), que define tempranamente las pruebas que verifican el comportamiento esperado del sistema. El enfoque incorporó además la revisión humana de los artefactos generados con apoyo de inteligencia artificial y utilizó OpenSpec como framework para implementar los principios de SDD. La investigación no tuvo por objeto evaluar la herramienta en sí misma, sino el impacto del enfoque basado en especificaciones sobre el proceso de desarrollo.

Distintos estudios han analizado las implicaciones del uso de inteligencia artificial en la ingeniería de software, incluyendo sus beneficios y desafíos. Sin embargo, gran parte de la evidencia disponible proviene de investigaciones controladas, de laboratorio o de corta duración, mientras que existen menos estudios realizados en entornos organizacionales reales.

Esa brecha motivó este trabajo. En lugar de demostrar las capacidades técnicas de una herramienta específica, el interés se centró en comprender cómo una metodología basada en especificaciones puede influir en el desempeño de equipos que utilizan inteligencia artificial como parte habitual de sus actividades de desarrollo.

A partir de lo anterior, se plantea la siguiente pregunta de investigación:

¿Qué cambios se observan en la calidad del software y el retrabajo, y cuál es la percepción de los desarrolladores respecto de la trazabilidad entre requerimientos, pruebas e implementación, al incorporar un enfoque de desarrollo asistido por inteligencia artificial basado en Spec-Driven Development (SDD) e integrado con Test-Driven Development (TDD), respecto de un proceso previo que utilizaba inteligencia artificial de forma no estructurada, en un equipo real de desarrollo?

Con este propósito, el objetivo general fue evaluar los cambios asociados a la incorporación de un enfoque de desarrollo asistido por inteligencia artificial basado en Spec-Driven Development (SDD) e integrado con Test-Driven Development (TDD), comparando un período previo de uso de IA sin especificaciones estructuradas con un período posterior en que se aplicó el enfoque propuesto, en un equipo real de desarrollo. La evaluación consideró indicadores relacionados con la calidad del software, la trazabilidad percibida por los desarrolladores y el nivel de retrabajo, complementados con su percepción respecto del nuevo enfoque de trabajo.

Para lograr este objetivo, se planteó la hipótesis de que, en un contexto donde el desarrollo ya es asistido por inteligencia artificial, la incorporación de un enfoque basado en especificaciones estructuradas e integrado con prácticas de TDD se asocia con una mejora en la calidad del software y una reducción del retrabajo respecto del período previo de uso de IA sin dicho enfoque, junto con una percepción favorable de los desarrolladores respecto de la trazabilidad entre requerimientos, pruebas e implementación.

La propuesta se evaluó mediante un estudio de caso en un entorno empresarial real. Para ello se compararon métricas obtenidas antes y después de la implementación, utilizando información proveniente de herramientas de gestión de incidentes, sistemas de seguimiento de defectos y una encuesta realizada a los participantes. Esto permitió analizar la intervención manteniendo las condiciones propias del entorno organizacional donde se realizó el estudio.

El informe se organiza en siete capítulos. Tras esta introducción, el Capítulo 2 revisa los fundamentos y el estado del arte, y el Capítulo 3 describe el enfoque propuesto. El Capítulo 4 presenta el diseño de la evaluación y el Capítulo 5, la aplicación del piloto en la organización. Finalmente, el Capítulo 6 analiza los resultados y el Capítulo 7 expone las conclusiones y proyecciones del trabajo.

2 De los prompts a las especificaciones: fundamentos y estado del arte

2.1 Desarrollo asistido por inteligencia artificial

La aparición de modelos de lenguaje de gran escala (Large Language Models, LLM) ha permitido automatizar actividades que tradicionalmente dependían del trabajo manual de los desarrolladores, incluyendo generación de código fuente, documentación técnica, pruebas automatizadas y análisis de requerimientos [1].

Este avance ha impulsado la adopción de herramientas como GitHub Copilot, Cursor y Claude Code, integradas progresivamente en procesos de desarrollo para aumentar la productividad y acelerar la entrega de software. Sin embargo, la literatura indica que la calidad de los resultados depende en gran medida de la información proporcionada al modelo. Cuando los requerimientos son ambiguos o incompletos, los modelos tienden a inferir aspectos no especificados, generando soluciones técnicamente válidas, pero no necesariamente alineadas con las necesidades del negocio [1], [2].

2.2 El problema de los prompts

La interacción con los LLM se realiza principalmente mediante prompts: instrucciones en lenguaje natural que el modelo debe interpretar.

Los LLM son capaces de completar patrones con alta efectividad, pero no necesariamente de interpretar correctamente la intención del usuario cuando la información entregada resulta ambigua. Solicitudes aparentemente simples pueden obligar al modelo a asumir decisiones funcionales y técnicas no declaradas explícitamente [1]. Como consecuencia, una misma descripción puede producir implementaciones diferentes, aumentando la probabilidad de inconsistencias, retrabajo y correcciones posteriores.

La literatura reciente relaciona este fenómeno con el concepto de vibe coding, utilizado para describir enfoques de desarrollo basados en instrucciones informales y poco estructuradas [3]. Este tipo de interacción facilita la creación rápida de prototipos, pero incrementa los riesgos de pérdida de trazabilidad, resultados variables y funcionalidades alejadas de los objetivos originales. Además, persiste el problema de las alucinaciones, donde los modelos producen información incorrecta o inexistente con apariencia de validez [2].

2.3 Requerimientos y especificaciones en ingeniería de software

La ambigüedad de los requerimientos no es un problema exclusivo de la inteligencia artificial. Históricamente, la ingeniería de software ha identificado la definición incorrecta de requerimientos como una de las principales causas de errores, retrasos y sobrecostos en proyectos tecnológicos.

La ingeniería de requerimientos busca identificar, documentar, analizar y validar las necesidades de usuarios y organizaciones. Dentro de este proceso, las especificaciones surgieron como mecanismos para representar de forma estructurada el comportamiento esperado de un sistema, reduciendo interpretaciones divergentes entre usuarios, analistas, desarrolladores y responsables del negocio.

Otro concepto relevante es la trazabilidad, entendida como la capacidad de establecer relaciones explícitas entre requerimientos, diseño, pruebas e implementación. Esta trazabilidad permite verificar que cada funcionalidad desarrollada tenga un origen claramente definido y que las necesidades del negocio hayan sido correctamente implementadas.

En el desarrollo asistido por IA, el modelo tiende a completar la información faltante en lugar de solicitarla (sección 2.2). Por ello, una especificación más precisa reduce su margen de interpretación y aumenta la probabilidad de obtener resultados alineados con lo requerido [1], [2].

2.4 Spec-Driven Development

Spec-Driven Development (SDD) es un enfoque que propone considerar la especificación como el artefacto principal del desarrollo de software. Bajo este paradigma, las especificaciones actúan como fuente de verdad desde la cual se derivan otros artefactos del proyecto, tales como diseño, implementación y validación [1].

Según Piskala [1], SDD invierte la relación tradicional entre documentación y código. Mientras en enfoques convencionales el código tiende a transformarse en la representación definitiva del comportamiento del sistema, SDD establece que la especificación debe declarar la intención del sistema y que el código debe implementarla o verificarse contra ella.

La literatura distingue distintos niveles de adopción, desde enfoques Spec-First, donde la especificación guía el desarrollo inicial, hasta modelos Spec-Anchored o Spec-as-Source, donde la especificación mantiene mayor autoridad durante el ciclo de vida del software [1].

2.5 OpenSpec

OpenSpec es una iniciativa reciente asociada a los principios de Spec-Driven Development. Su propósito es organizar el trabajo de desarrollo mediante artefactos estructurados, favoreciendo la relación entre especificaciones, decisiones de diseño y tareas de implementación [4].

En la práctica, esto significa que el asistente de IA recibe como contexto especificaciones mantenidas en el repositorio, en lugar de prompts aislados que se pierden entre una sesión y otra.

OpenSpec fue seleccionado como el framework para implementar los principios de SDD en esta investigación, debido a su disponibilidad, simplicidad de adopción y alineación con la filosofía de especificación como fuente de verdad. Si bien su correcta implementación constituye una condición necesaria para la validez del piloto, el objeto de evaluación no es OpenSpec como herramienta, sino el impacto del enfoque basado en especificaciones. Su uso específico dentro de la investigación se describe en el Capítulo 3.

2.6 Test-Driven Development

En Test-Driven Development (TDD), las pruebas automatizadas se escriben antes que la funcionalidad que verifican, de modo que la prueba define primero el comportamiento esperado del sistema.

El proceso sigue el ciclo Red-Green-Refactor: primero se escribe una prueba que falla (Red); luego, el mínimo código necesario para que pase (Green); finalmente, se mejora la estructura interna sin alterar el comportamiento validado (Refactor).

En el contexto del desarrollo asistido por inteligencia artificial, estudios recientes han explorado la integración de TDD con LLM para guiar y validar la generación de código. Mathews y Nagappan [5] evaluaron un enfoque de generación de código basado en pruebas, mientras que Piya y Sullivan [6] estudiaron el uso de TDD junto con asistentes basados en LLM.

2.7 Human-in-the-loop en el desarrollo asistido por IA

El concepto de human-in-the-loop (HITL) consiste en integrar la participación humana dentro de los procesos ejecutados por sistemas de inteligencia artificial, con el fin de revisar, corregir y validar los resultados antes de su aceptación final.

En el desarrollo de software asistido por IA, este enfoque resulta relevante porque los LLM, aunque pueden generar código funcional, no garantizan por sí solos el cumplimiento de los requerimientos del negocio. Piskala [1] señala que el workflow SDD incorpora revisiones humanas en las etapas de especificación, diseño,

implementación y validación para detectar desviaciones de manera temprana. De forma similar, Gupta [2] recomienda la supervisión humana en aquellas salidas de sistemas agentivos que presentan baja confianza.

La intervención humana también ha sido estudiada en procesos de generación de código guiados por pruebas. Fakhoury et al. [7] proponen un enfoque interactivo en el que las pruebas permiten formalizar progresivamente la intención del usuario y proporcionar retroalimentación al proceso de generación mediante LLM.

2.8 Estado del arte

La investigación reciente muestra un creciente interés por comprender cómo las especificaciones estructuradas pueden mejorar el desempeño de herramientas de inteligencia artificial aplicadas al desarrollo de software. Piskala [1] sostiene que los asistentes de programación basados en IA obtienen mejores resultados cuando trabajan sobre contratos explícitos en lugar de instrucciones ambiguas. En la misma línea, Gupta [2] señala que los enfoques basados en especificaciones favorecen la precisión, consistencia y cumplimiento de restricciones en sistemas asistidos por LLM.

Ambos trabajos coinciden en que las especificaciones reducen la ambigüedad presente en prompts tradicionales y facilitan la generación de artefactos más verificables. Ambos señalan, además, que las especificaciones apoyan la creación de pruebas, la validación automatizada y la trazabilidad de requerimientos durante el ciclo de desarrollo [1], [2].

Gran parte de esta evidencia, sin embargo, proviene de estudios experimentales, análisis conceptuales o evaluaciones centradas en herramientas específicas. Queda pendiente estudiar cómo se comportan estos enfoques en equipos reales, con restricciones organizacionales, tiempos de entrega, prácticas ágiles existentes y necesidades concretas del negocio.

2.9 Brecha de investigación

Aunque estudios recientes han explorado la integración de LLM con TDD [5]–[7] y comienzan a aparecer propuestas que combinan especificaciones, pruebas y generación asistida por IA [8], la evidencia se concentra principalmente en experimentos controlados y tareas de programación. En consecuencia, continúa siendo limitada la evidencia sobre la aplicación conjunta de estos enfoques en equipos de desarrollo y proyectos reales dentro de contextos organizacionales.

Esta tesina aborda esa brecha mediante un piloto en la organización, donde se compara el desarrollo asistido por IA antes y después de incorporar SDD con OpenSpec y TDD, en términos de calidad del software, retrabajo y trazabilidad percibida.

3 La especificación como fuente de verdad: integración de Spec-Driven Development y Test-Driven Development

3.1 Enfoque propuesto

Se definió un enfoque de desarrollo asistido por inteligencia artificial que utilizó Spec-Driven Development (SDD) como estructura, Test-Driven Development (TDD) como mecanismo de validación y OpenSpec para organizar las especificaciones, decisiones de diseño y tareas asociadas a cada requerimiento.

El enfoque incluyó la formalización de reglas de negocio y criterios de aceptación, la generación temprana de pruebas y la revisión humana de los artefactos producidos con apoyo de inteligencia artificial. Las especificaciones funcionaron como referencia común para los desarrolladores, las pruebas automatizadas y las herramientas de IA, con el fin de reducir la dependencia de instrucciones informales o prompts aislados [1], [2].

Cuatro principios sustentaron esta propuesta:

- La especificación como fuente única de verdad.
- Trazabilidad entre requerimientos, diseño, pruebas e implementación.
- Validación temprana mediante pruebas automatizadas.
- Control y revisión humana sobre los resultados generados por inteligencia artificial.

3.2 Implementación de Spec-Driven Development mediante OpenSpec

En la práctica, la propuesta se materializó con OpenSpec [4], herramienta que en el piloto sirvió para estructurar los artefactos Spec, Design y Tasks, permitiendo mantener la relación entre requerimientos, decisiones técnicas, pruebas y actividades de implementación.

Especificación (Spec)

La especificación constituyó el punto de partida del proceso. En este artefacto se documentaron los objetivos funcionales, las reglas de negocio, las restricciones operacionales y los criterios de aceptación asociados a cada requerimiento, sin incorporar detalles de implementación. Cada especificación sirvió luego como entrada para la generación del diseño técnico, la definición de pruebas y la generación asistida por IA de componentes de software.

Como el asistente de IA elaboró los artefactos de OpenSpec con el contexto del código fuente de la solución, fue posible detectar información faltante en los requerimientos y refinarla durante la elaboración de este artefacto.

- Diseño (Design)

El artefacto Design documentó la arquitectura propuesta, los componentes involucrados, los modelos de datos, las integraciones y las decisiones técnicas necesarias para implementar cada especificación definida previamente.

- Tareas (Tasks)

Las tareas correspondieron a unidades concretas de trabajo de implementación, pruebas y validación, derivadas de cada especificación y su diseño. Durante el piloto, cada tarea permitió vincular el cambio realizado con la especificación y el diseño que le dieron origen.

3.3 Integración con Test-Driven Development

Mientras Spec-Driven Development definió qué debía construirse, Test-Driven Development proporcionó mecanismos para validar que la implementación cumpliera efectivamente con las especificaciones establecidas.

Para articular ambos enfoques, los requerimientos definidos en las especificaciones se tomaron como base para construir pruebas automatizadas antes de la implementación. Estas pruebas debían orientar el desarrollo siguiendo el ciclo Red-Green-Refactor, permitiendo verificar el cumplimiento de los requerimientos formalizados.

Con esta integración se buscó mantener una relación explícita entre el requerimiento, las pruebas utilizadas para validarlo y su implementación.

3.4 Rol de la inteligencia artificial dentro del proceso

La inteligencia artificial intervino en todas las etapas del proceso. Los modelos no se utilizaron solo para automatizar tareas aisladas, sino de manera continua en actividades de análisis, diseño, implementación y validación de software.

En este esquema, las especificaciones actuaron como mecanismo de comunicación entre los desarrolladores y los modelos de inteligencia artificial. A partir de dichas especificaciones, la IA colaboró en la generación de código, construcción de pruebas automatizadas, elaboración de documentación técnica y propuestas de refactorización.

Se aprovecharon así las capacidades generativas de la inteligencia artificial sin delegar completamente las decisiones técnicas. Los desarrolladores mantuvieron la responsabilidad sobre la revisión de los artefactos producidos, la validación de los resultados y la aceptación final de las implementaciones.

3.5 Flujo de trabajo propuesto

El flujo de trabajo definido como referencia para la intervención se estructuró en cinco etapas principales:

- Definición de la especificación funcional.
- Elaboración del diseño técnico.
- Construcción de pruebas automatizadas basadas en la especificación.
- Implementación asistida por inteligencia artificial.
- Validación mediante ejecución de pruebas y revisión del equipo.



Figura 1. Representación conceptual del enfoque utilizado

Su aplicación práctica podía variar según la naturaleza de la funcionalidad desarrollada y el nivel de adopción alcanzado por las prácticas consideradas.

Este flujo constituyó la referencia utilizada posteriormente durante el piloto descrito en el Capítulo 5.

4 Diseño de la evaluación: midiendo el impacto en un entorno real

4.1 Hipótesis de investigación

La evaluación se estructuró a partir de la siguiente hipótesis principal:

H1: La incorporación de un enfoque estructurado de desarrollo asistido por inteligencia artificial, basado en Spec-Driven Development (SDD) e integrado con prácticas de Test-Driven Development (TDD), se asocia con una mejora en la calidad del software y una reducción del retrabajo respecto del período previo en que la inteligencia artificial se utilizó sin dicho enfoque, junto con una percepción favorable de los desarrolladores respecto de la trazabilidad entre requerimientos, pruebas e implementación.

Para analizar los distintos aspectos involucrados en esta hipótesis se definieron las siguientes subhipótesis:

- H1a Calidad del software: La incorporación del enfoque se asocia con una menor cantidad de errores detectados en producción a nivel de sistema respecto del período previo.
- H1b Retrabajo: La incorporación del enfoque se asocia con una menor tasa de iteraciones de corrección durante aseguramiento de calidad a nivel de proyecto respecto del período previo.
- H1c Trazabilidad percibida: Los desarrolladores presentan una percepción favorable respecto de la capacidad de relacionar requerimientos, pruebas e implementación durante la aplicación del enfoque.

4.2 Diseño metodológico

Para evaluar las hipótesis planteadas se utilizó un diseño antes y después, comparando indicadores obtenidos durante un período previo, comprendido entre enero y junio de 2025, en que el equipo utilizó inteligencia artificial sin especificaciones estructuradas, con aquellos registrados durante un período posterior, comprendido entre enero y junio de 2026, en que se incorporó el enfoque estructurado descrito en el Capítulo 3, basado en especificaciones mediante SDD e integrado con prácticas de TDD. La aplicación de las prácticas de TDD presentó distintos niveles de adopción entre las funcionalidades desarrolladas, aspecto que fue considerado al interpretar los resultados. En ambos períodos el equipo contó con apoyo de herramientas de inteligencia artificial, por lo que la comparación no evaluó la presencia de IA, sino la incorporación de especificaciones estructuradas y prácticas de TDD como mecanismo de guía y validación del trabajo asistido por IA.

Se seleccionaron períodos equivalentes del año para mantener una ventana temporal comparable. Sin embargo, la separación entre ambos períodos impide controlar completamente factores externos, por lo que las diferencias observadas no se interpretaron como evidencia causal atribuible exclusivamente al enfoque evaluado. Estas variables de confusión se detallan en la sección 4.6.

La investigación se desarrolló como un estudio de caso en un entorno organizacional real. El objetivo no fue aislar variables en condiciones controladas, sino observar el comportamiento del enfoque dentro de un contexto de desarrollo real, donde los desarrolladores trabajaron con requerimientos del negocio, herramientas de inteligencia artificial y procesos de desarrollo y validación ya existentes, sujetos a las restricciones y dinámicas propias de una empresa. La evidencia obtenida corresponde, por lo tanto, a la aplicación práctica del enfoque y no a un escenario experimental.

La evaluación combinó técnicas cuantitativas y cualitativas. La componente cuantitativa se centró en la medición de indicadores asociados a calidad del software y retrabajo. La componente cualitativa buscó comprender la percepción de los desarrolladores respecto de la trazabilidad, claridad de los requerimientos, control del proceso y confianza en el software generado con apoyo de inteligencia artificial.

4.3 Variables e indicadores de evaluación

Para operacionalizar las hipótesis de investigación se definieron tres variables de estudio: calidad del software, retrabajo y trazabilidad.

Para efectos de esta investigación, la calidad del software se operacionalizó mediante la ocurrencia de errores detectados una vez que los cambios se encontraban en producción. Este indicador se utilizó como una aproximación observable a la calidad operacional de las funcionalidades desarrolladas y no pretende representar de manera integral todas las dimensiones asociadas a la calidad del software. La medición consideró el número de errores de sistema registrados durante cada período, utilizando como fuente el sistema de gestión de incidentes de la organización.

El retrabajo se definió como el conjunto de correcciones, ajustes o modificaciones adicionales requeridas durante la validación en ambiente de calidad de los entregables asociados a un proyecto. El indicador utilizado fue la tasa de retrabajo, calculada como el número de iteraciones de corrección registradas durante las actividades de validación y aseguramiento de la calidad de cada proyecto, dividido por el total de ingresos a pruebas. Cada transición registrada en el sistema de seguimiento e identificada como retrabajo se contabilizó como una iteración de corrección. En los proyectos de ambos períodos se consideraron todas las iteraciones de retrabajo asociadas al proyecto. Como la complejidad de los proyectos difería entre sistemas, la evaluación de H1b se centró en una muestra de proyectos de un mismo sistema, cuya complejidad se evaluó de forma previa, y se complementó con el tiempo de retrabajo expresado en días hábiles (sección 6.2).

La trazabilidad percibida se definió como la percepción de los desarrolladores respecto de su capacidad para relacionar los requerimientos, las pruebas y la implementación correspondiente. Esta variable fue evaluada mediante una encuesta aplicada a los participantes, cuyo instrumento se describe en la sección siguiente.

La Tabla 1 resume la relación entre las hipótesis planteadas, las variables evaluadas, las métricas utilizadas y las fuentes de información consideradas durante la investigación.

Tabla 1. Variables, métricas y fuentes de información utilizadas en la evaluación.

Hipótesis	Variable	Métrica	Fuente de información
H1a	Calidad del software	Número de errores detectados en producción a nivel de sistema	Incident Management System
H1b	Retrabajo	Tasa de retrabajo (iteraciones de corrección / ingresos a pruebas) a nivel de proyecto	Issue Tracking System
H1c	Trazabilidad	Nivel de trazabilidad percibida entre requerimientos, pruebas e implementación	Encuesta a desarrolladores

4.4 Fuentes de información e instrumentos

Los indicadores de calidad del software y retrabajo se obtuvieron desde los registros del sistema de gestión de incidentes y del sistema de seguimiento de defectos de la organización, respectivamente, según se resume en la Tabla 1. El detalle por proyecto utilizado en los cálculos se presenta en el Anexo A y la rúbrica de evaluación previa de complejidad, en el Anexo B.

La información correspondiente a la trazabilidad percibida y a la valoración del enfoque por parte de los desarrolladores fue recopilada mediante una encuesta aplicada a los participantes del estudio una vez finalizado

el período de observación. El instrumento fue diseñado para evaluar aspectos relacionados con la capacidad de relacionar requerimientos, pruebas e implementación, así como la percepción respecto de la claridad de los requerimientos, el control del proceso de desarrollo y la confianza en el software generado con apoyo de inteligencia artificial.

La encuesta fue aplicada a los integrantes del equipo que participó en la implementación del enfoque evaluado e incorporó preguntas cerradas con una escala tipo Likert de cinco niveles, donde el valor 1 representó el menor nivel de acuerdo y el valor 5 el mayor. Se incluyeron además preguntas abiertas destinadas a recoger observaciones, comentarios y experiencias de los participantes sobre el enfoque utilizado.

Por razones de confidencialidad, en este documento se omite el nombre de la organización en la que se realizó el estudio y se modificó la denominación de los sistemas que permitían identificarla. La información utilizada proviene de los registros internos de la organización y de la encuesta aplicada al equipo, y se emplea exclusivamente con fines académicos.

4.5 Procedimiento de evaluación

El procedimiento consistió en recopilar los indicadores del período previo, aplicar el enfoque descrito en el Capítulo 3, recopilar los indicadores del período posterior y comparar ambos períodos en calidad y retrabajo, este último sobre una muestra con complejidad evaluada de forma previa; la trazabilidad percibida se analizó descriptivamente a partir de la encuesta.

Los indicadores de calidad y retrabajo fueron analizados mediante estadística descriptiva y comparación entre períodos. Las respuestas cerradas de la encuesta fueron analizadas mediante estadística descriptiva, mientras que las respuestas abiertas fueron utilizadas para identificar tendencias y observaciones recurrentes respecto del enfoque evaluado.

4.6 Amenazas a la validez

Como toda investigación aplicada en un entorno organizacional real, este estudio presentó limitaciones que debieron ser consideradas durante la interpretación de los resultados.

Respecto de la validez interna, se identificaron tres amenazas principales. La primera corresponde a posibles diferencias en el alcance, tamaño y complejidad de los proyectos desarrollados durante cada período observado.

La segunda se relaciona con la evolución de las herramientas de inteligencia artificial entre ambos períodos. Durante el primer período se utilizó GitHub Copilot como herramienta de apoyo al desarrollo, mientras que en el segundo período se utilizó Claude Code. Este reemplazo fue motivado por un cambio en el esquema de cobro del proveedor y por las capacidades requeridas por el enfoque, y no por el diseño del estudio, según se detalla en el Capítulo 5. Aun así, el cambio introduce una variable de confusión, debido a que las diferencias observadas entre ambos períodos podrían estar asociadas tanto a la incorporación del enfoque basado en SDD y TDD como a las diferencias en las capacidades de las herramientas de inteligencia artificial utilizadas. En consecuencia, el diseño del estudio no permite aislar completamente el efecto del enfoque evaluado de aquel asociado al cambio de herramienta, por lo que esta condición fue considerada al interpretar los resultados.

La tercera se refiere al nivel de adopción del enfoque. Si bien SDD se utilizó como estructura de referencia para el desarrollo durante el segundo período, las prácticas de TDD no fueron aplicadas de manera homogénea en todas las funcionalidades. Esta variabilidad debe considerarse al interpretar los resultados, ya que dificulta atribuir las diferencias observadas a la aplicación uniforme de todos los componentes del enfoque.

Respecto de la validez externa, los resultados obtenidos corresponden a un único caso organizacional y, por lo tanto, sus conclusiones deben interpretarse dentro del contexto específico donde se realizó la investigación.

No obstante, la descripción detallada del contexto y del enfoque evaluado busca facilitar la comparación con experiencias similares en otras organizaciones.

Finalmente, respecto de la validez de constructo, las variables consideradas fueron operacionalizadas mediante indicadores observables y fuentes de información disponibles dentro del proceso de desarrollo, procurando mantener coherencia entre los objetivos de investigación, las hipótesis planteadas y los mecanismos de medición utilizados.

5 De la IA sin especificaciones al enfoque SDD+TDD: el piloto en la organización

5.1 Contexto organizacional

La evaluación se realizó en la organización, una compañía del sector asegurador, específicamente en la Gerencia de Operaciones. La iniciativa surgió desde la Subgerencia de Desarrollo con el objetivo de evaluar la incorporación de Spec-Driven Development (SDD) y Test-Driven Development (TDD) en la forma de trabajo del equipo.

Antes de esta evaluación, el uso de inteligencia artificial ya era habitual entre los desarrolladores, pero no estaba acompañado de especificaciones estructuradas ni de una práctica sistemática de pruebas tempranas. De ahí la pregunta que motivó el estudio: qué ocurriría al incorporar ambas prácticas dentro del proceso de desarrollo existente.

El estudio se llevó a cabo con el equipo encargado de mantener y evolucionar los canales virtuales de la compañía. Entre los sistemas bajo su responsabilidad se encuentran los portales públicos y privados, el CRM y la plataforma utilizada para gestionar requerimientos de clientes. Durante todo el período analizado, el equipo estuvo conformado por un jefe de proyecto y cinco ingenieros de software.

La evaluación se realizó manteniendo el funcionamiento habitual del equipo: requerimientos reales provenientes de distintas necesidades del negocio, sobre sistemas efectivamente en uso.

5.2 Período 1: desarrollo asistido por IA sin especificaciones estructuradas

Entre enero y junio de 2025, los desarrolladores utilizaban Visual Studio Code y GitHub Copilot como parte de sus herramientas habituales de trabajo. La interacción con los modelos de inteligencia artificial se realizaba principalmente mediante instrucciones escritas en lenguaje natural. No existía en ese momento un mecanismo común para documentar de manera estructurada los objetivos funcionales, las reglas de negocio y los criterios de aceptación antes de solicitar una implementación a la IA. Tampoco se aplicaba TDD de manera formal.

Posteriormente, en diciembre de 2025, se realizó una primera experiencia con el uso de especificaciones. Esta consistió en la modernización de una aplicación existente, cuyo objetivo era migrar una solución construida sobre tecnologías legadas hacia una arquitectura más actual. Para apoyar este trabajo se utilizó OpenSpec.

Esta experiencia permitió identificar una dificultad que posteriormente sería relevante para el estudio. Si bien se habían generado especificaciones para apoyar el desarrollo, estas no contenían todo el contexto funcional necesario para representar el comportamiento esperado de la aplicación. En particular, algunas reglas y situaciones conocidas por los usuarios no habían quedado suficientemente reflejadas en los artefactos utilizados por los desarrolladores.

Al implementar la solución con apoyo de inteligencia artificial, estas omisiones se hicieron visibles. Algunas funcionalidades no respondían completamente a lo esperado por los usuarios y tuvieron que ser ajustadas posteriormente, aumentando el retrabajo del equipo.

A partir de esta experiencia se identificó la necesidad de mejorar la forma en que se construían las especificaciones antes de utilizarlas como entrada para los modelos. Más que limitarse a describir una funcionalidad, era necesario incorporar contexto de negocio, reglas, criterios de aceptación y escenarios que permitieran representar con mayor precisión el comportamiento esperado. También se consideró necesario establecer instancias explícitas de revisión por parte de los desarrolladores.

5.3 Transición hacia el enfoque SDD+TDD

Antes de comenzar el segundo período se realizaron algunos cambios en la forma de trabajo, tomando como referencia lo observado en la experiencia anterior.

Como primera medida, se capacitó al equipo en los conceptos que se utilizarían durante la evaluación. Se realizaron cinco sesiones en las que se abordaron el desarrollo basado en especificaciones (SDD), el desarrollo guiado por pruebas (TDD) y el uso de OpenSpec para elaborar los artefactos que luego servirían de entrada para el desarrollo. La capacitación se complementó con el acompañamiento del equipo durante los primeros proyectos, con el fin de corregir y ajustar la práctica sobre la marcha.

También se modificó la forma de levantar los requerimientos. Además de registrar la necesidad planteada por el usuario, se puso mayor énfasis en identificar las reglas de negocio, los criterios de aceptación y los distintos escenarios que debía resolver cada funcionalidad, incluyendo tanto los casos esperados como las situaciones de error.

Otro cambio fue establecer explícitamente la revisión humana dentro del proceso. Las salidas generadas con apoyo de inteligencia artificial no se consideraban automáticamente válidas. Se definieron reuniones diarias para que cada desarrollador revisara en conjunto con el jefe de proyecto los diseños, tareas, pruebas e implementaciones antes de continuar con la etapa siguiente.

La IA se mantuvo como una herramienta de apoyo al desarrollo, mientras que las decisiones respecto de la validez funcional y técnica de los artefactos permanecieron bajo responsabilidad del equipo.

5.4 Período 2: implementación evaluada con SDD+TDD

El segundo período se desarrolló entre enero y junio de 2026, manteniendo el mismo equipo del período anterior, compuesto por un jefe de proyecto y cinco ingenieros de software. Durante este intervalo se ejecutaron quince proyectos asociados a requerimientos reales del negocio. Para la evaluación general se consideró la totalidad de los proyectos desarrollados (Anexo A), sin realizar una selección previa ni excluir casos en función de sus resultados. De esta forma, se buscó mantener la dinámica habitual del equipo y observar la aplicación de SDD y TDD bajo las condiciones normales de trabajo. Adicionalmente, el análisis de H1b se profundizó sobre una muestra de proyectos de un mismo sistema, según se describe en la sección 6.2.

El proceso comenzaba con reuniones entre los desarrolladores y los usuarios responsables de cada necesidad de negocio. En estas instancias se discutía el requerimiento y se levantaban las reglas de negocio, los criterios de aceptación y los distintos escenarios que debía considerar la solución.

Una de las principales diferencias respecto del primer período fue el contexto proporcionado a la inteligencia artificial. Además de la descripción del requerimiento, se incorporaba información proveniente del código fuente existente. Esto permitía considerar aspectos propios de cada sistema, como su estructura, dependencias, convenciones de desarrollo y componentes previamente implementados.

La forma de abordar cada requerimiento dependía de su naturaleza. En aquellos principalmente visuales, como la creación o modificación de pantallas, se utilizaba inteligencia artificial para generar propuestas funcionales que podían ser revisadas y ajustadas junto con los usuarios. Esto permitía validar tempranamente aspectos de la solución antes de avanzar con su implementación.

Cuando los requerimientos involucraban reglas de negocio, validaciones o procesos de mayor complejidad, se aplicaba el flujo de referencia descrito en el Capítulo 3, utilizando los artefactos Spec, Design y Tasks como base para la implementación y validación. Estos se construían a partir de la información levantada con los usuarios y del contexto técnico disponible en el código fuente.

Los artefactos generados eran revisados por los desarrolladores antes de continuar con la siguiente etapa. Si durante esta revisión se detectaban inconsistencias, información faltante o aspectos que no representaban adecuadamente el comportamiento esperado, se realizaban los ajustes necesarios. Este mismo criterio de revisión se aplicaba posteriormente a las pruebas y al código generado con apoyo de inteligencia artificial.

El enfoque contemplaba la construcción y ejecución de pruebas automatizadas antes de la implementación, siguiendo el ciclo Red-Green-Refactor de TDD. Sin embargo, la aplicación de TDD no fue homogénea en todas las funcionalidades, observándose distintos niveles de adopción durante el piloto. Por esta razón, TDD se consideró una práctica integrada al enfoque evaluado, pero no una intervención aplicada de manera uniforme a todas las unidades de desarrollo.

Para efectos de la evaluación, un proyecto se consideraba completado cuando el usuario validaba satisfactoriamente los resultados asociados al requerimiento en el ambiente de pruebas y confirmaba su conformidad con el resultado obtenido.

5.5 Herramientas utilizadas

Durante ambos períodos, Visual Studio Code fue utilizado como entorno principal de desarrollo. Las herramientas de inteligencia artificial disponibles variaron entre los períodos analizados.

Durante el primer período, GitHub Copilot fue la principal herramienta utilizada por los desarrolladores como apoyo en actividades de programación.

El cambio de herramienta entre ambos períodos no respondió al diseño de esta investigación, sino a dos condiciones externas al estudio. La primera fue de costos: el partner a través del cual la organización tenía contratado Copilot migró desde un plan mensual por usuario hacia un esquema de cobro por token, lo que proyectaba un aumento del gasto mensual desde aproximadamente $1.000.000 a $5.000.000 CLP. La segunda fue de capacidad: los modelos disponibles en Claude Code mostraron un mejor desempeño en las tareas que el enfoque requería, en particular la redacción y revisión de especificaciones y la generación de pruebas a partir de criterios de aceptación.

Durante el segundo período se utilizó Claude Code como apoyo en distintas etapas del proceso, incluyendo la construcción y revisión de especificaciones, la generación de pruebas y la implementación. Para estructurar las especificaciones se utilizó OpenSpec mediante los artefactos Spec, Design y Tasks.

Las observaciones realizadas durante las pruebas fueron registradas en Redmine mediante incidencias asociadas al proyecto correspondiente. Cuando una observación requería trabajo adicional de desarrollo, esta situación quedaba registrada en la plataforma, permitiendo identificar los ciclos de corrección utilizados posteriormente para calcular el indicador de retrabajo.

Los errores identificados una vez que las funcionalidades se encontraban en producción fueron gestionados mediante el sistema interno de incidencias de la organización. Este mecanismo constituyó la fuente utilizada para identificar los errores de producción en ambos períodos.

5.6 Resumen comparativo de los períodos

Los dos períodos representan formas distintas de incorporar inteligencia artificial al proceso de desarrollo del mismo equipo.

En el primer período, la IA ya formaba parte del trabajo cotidiano, pero los requerimientos se entregaban principalmente mediante instrucciones en lenguaje natural y sin una estructura común de especificaciones ni una aplicación formal de TDD.

En el segundo período se mantuvo el uso de inteligencia artificial, pero se incorporaron SDD y TDD al proceso. Al mismo tiempo, se reforzó el levantamiento de contexto funcional y se establecieron revisiones humanas antes de avanzar entre las distintas etapas del desarrollo.

Tabla 2. Características de ambos períodos, destacando los elementos que se mantuvieron y los cambios ocurridos durante la intervención.

Característica	**Período 1: Ene-Jun 2025**	**Período 2: Ene-Jun 2026**
Equipo	1 jefe de proyecto + 5 ingenieros	1 jefe de proyecto + 5 ingenieros
Desarrollo asistido por IA	Sí	Sí
Herramientas / Modelos	GitHub Copilot	Claude Code
Especificaciones estructuradas	No	Sí
OpenSpec	No	Sí
Aplicación de prácticas de TDD	No	Sí, con distintos niveles de adopción según el proyecto
Levantamiento estructurado de contexto funcional	No formalizado	Sí
Revisión humana entre etapas	No formalizado	Sí

6 El antes y el después: resultados y análisis

Este capítulo presenta los resultados de la evaluación comparando el Período 1 (enero-junio de 2025) con el Período 2 (enero-junio de 2026), siguiendo el diseño metodológico del Capítulo 4. Se ordena en función de las tres subhipótesis: calidad del software (H1a), retrabajo (H1b) y trazabilidad percibida (H1c).

6.1 Calidad del software (H1a)

La calidad se midió a través del número de errores detectados en producción a nivel de sistema, tomando como fuente los tickets con categoría “Error - Sistema” registrados en el sistema de gestión de incidentes de la organización.

Tabla 3. Resumen comparativo de errores en producción (Ene-Jun 2025 vs. Ene-Jun 2026).

Indicador	**2025**	**2026**	**Δ**	**Δ%**
Errores en producción (N.° tickets)	101	82	−19	−18,8%

Indicador	2025	2026	Δ	Δ%
Tickets totales	256	261	+5	+2,0%
Tasa de error (errores / tickets totales)	39,5%	31,4%	−8,0 pp	−20,4%
Errores promedio por mes	16,8	13,7	−3,2	−18,8%

Los errores en producción disminuyeron de 101 a 82 casos entre ambos períodos, una reducción de 18,8%. El total de tickets, en cambio, se mantuvo estable (256 a 261), de modo que la caída de errores no acompaña una baja en el volumen de trabajo registrado. El indicador, con todo, no permite controlar diferencias en el alcance, complejidad o esfuerzo de los requerimientos desarrollados en cada período.

Para comprender mejor esta variación, conviene revisar cómo se distribuyeron los tickets por categoría durante ambos períodos:

Tabla 4. Tickets por categoría (Ene-Jun 2025 vs. Ene-Jun 2026).

Categoría	2025	2026
Solicitud	111	119
Error - Sistema	101	82
Mejoras Sistema	44	60

Los errores de sistema disminuyeron, mientras que las categorías “Solicitud” y “Mejoras Sistema” aumentaron (de 111 a 119 y de 44 a 60, respectivamente). Este comportamiento es coherente con lo observado en la Tabla 3, ya que el volumen total de trabajo no se redujo, sino que cambió su composición: los errores pasaron a representar una menor proporción de los tickets, mientras que otras categorías adquirieron mayor presencia.

6.2 Retrabajo (H1b)

El retrabajo se calculó a partir de los registros de Redmine, considerando cuántas veces un requerimiento volvió desde el estado “Liberado a pruebas” a una etapa que requería nuevas correcciones. Esta cantidad se comparó con el total de ingresos a “Liberado a pruebas” para obtener la tasa de retrabajo.

Tabla 5. Retrabajo por sistema (Ene-Jun 2025 vs. Ene-Jun 2026).

Sistema	Retornos 2025	Retornos 2026	Ingresos 2025	Ingresos 2026	Tasa 2025	Tasa 2026	Δ (pp)
Gestor de Campañas	0	18	0	176	—	10,23%	—
Plataforma de Atención de Clientes	4	8	61	98	6,56%	8,16%	+1,61
Portal Privado Administrador de Mutuos	55	13	412	137	13,35%	9,49%	−3,86
Portal Privado Administrador	0	0	0	0	—	—	—

Sistema	Retornos 2025	Retornos 2026	Ingresos 2025	Ingresos 2026	Tasa 2025	Tasa 2026	Δ (pp)
de Mutuos (Pagos)							
Portal Privado	13	5	83	54	15,66%	9,26%	−6,40
Portal Público Administrador de Mutuos	1	5	19	32	5,26%	15,63%	+10,36
Portal Público Arriendos	2	0	27	0	7,41%	—	—
Portal Público Venta de Seguros	0	0	1	0	0,00%	—	—
Portal Público	0	14	26	120	0,00%	11,67%	+11,67

Para interpretar estos resultados es necesario incorporar el concepto de complejidad del proyecto. En este trabajo, la complejidad se entiende como el conjunto de características que determinan el esfuerzo necesario para construir y validar una solución: el alcance funcional, la cantidad de requerimientos involucrados, las reglas de negocio e integraciones que contempla y el grado de madurez del sistema sobre el cual se desarrolla. A mayor complejidad, es esperable que aumente el número de ciclos de prueba y, con ello, la probabilidad de que un requerimiento deba volver a desarrollo. En consecuencia, la tasa de retrabajo no depende solo del enfoque de desarrollo utilizado, sino también de la complejidad de los proyectos que se comparan, factor ya identificado entre las amenazas a la validez interna. Como los registros de Redmine no contienen una medida directa de complejidad, para la muestra analizada se construyó un índice de complejidad previa basado en cinco atributos, que se describe más adelante.

Desde esta perspectiva, Portal Privado Administrador de Mutuos muestra con claridad el efecto de la complejidad sobre la comparación. Durante 2025, toda su actividad correspondió a un único proyecto de renovación completa de la plataforma, que registró 412 ingresos a pruebas y una tasa de retrabajo de 13,35%. Un proyecto de esta magnitud requiere naturalmente más ciclos e iteraciones de prueba que los requerimientos desarrollados sobre un sistema estable y en operación. En 2026, en cambio, la actividad se distribuyó en cuatro proyectos de menor envergadura, que en conjunto sumaron 137 ingresos y una tasa de 9,49%. La disminución de 3,86 puntos porcentuales se produce, por lo tanto, entre proyectos de complejidad muy distinta y no puede interpretarse como una comparación directa del desempeño del sistema entre ambos años.

Para analizar el retrabajo con mayor profundidad, se tomó como muestra Portal Privado. Se eligió este sistema porque se mantuvo en operación durante ambos períodos, fue desarrollado por el mismo equipo y registró proyectos tanto en 2025 como en 2026, lo que permite comparar ambos enfoques sobre una misma base. La muestra quedó compuesta por cuatro proyectos, dos por período. Para controlar el efecto de la complejidad, se decidió definir la complejidad de cada proyecto de forma previa (ex ante), es decir, con la información disponible antes de iniciar el desarrollo, a partir de cinco atributos: tipo de desarrollo, canales o sistemas impactados, integraciones, criticidad normativa y claridad de requerimientos. Cada atributo se puntuó de 1 (muy baja) a 5 (muy alta) según una rúbrica y se ponderó según su incidencia en el esfuerzo de desarrollo y validación: 25% para el tipo de desarrollo y la criticidad normativa, 20% para los canales o sistemas impactados y 15% para las integraciones y la claridad de requerimientos. En este último atributo, un puntaje mayor indica mayor incertidumbre en las definiciones. El puntaje ponderado se normalizó en un índice de 0 a 1, y los proyectos se clasificaron como de complejidad baja bajo 0,33, media desde 0,33 y alta desde 0,66. La Tabla 6 presenta el resultado de esta evaluación.

Tabla 6. Evaluación previa de complejidad de los proyectos de la muestra (Portal Privado).

Proyecto	Año	Tipo de desarrollo (25%)	Canales o sistemas (20%)	Integraciones (15%)	Criticidad normativa (25%)	Claridad de req. (15%)	Índice (0–1)	Clasificación
cc-2025007	2025	2	3	3	1	3	0,31	Baja
cc-2025016	2025	2	4	3	1	4	0,40	Media
cc-2026060	2026	2	3	3	1	4	0,35	Media
cc-2026075	2026	5	5	3	5	3	0,85	Alta

Los proyectos de 2025 obtuvieron una complejidad baja y media (índices de 0,31 y 0,40). Correspondieron a ajustes sobre funcionalidades existentes y sin exigencias normativas, aunque con definiciones pendientes en sus requerimientos. En 2026, cc-2026060, un dashboard para monitorear las desconexiones de las ejecutivas de atención, obtuvo una complejidad media (0,35), comparable a la de los proyectos de 2025, mientras que cc-2026075, que incorporaba mejoras de seguridad al portal y a la gestión de claves, alcanzó una complejidad alta (0,85), al impactar cuatro o más canales y obtener la máxima criticidad normativa. Por lo tanto, la muestra de 2026 no presentó una complejidad menor que la de 2025, de modo que una eventual reducción del retrabajo no puede atribuirse a que los proyectos del segundo período fueran más simples.

La Tabla 7 muestra la tasa de retrabajo de los proyectos de la muestra. En 2025, los dos proyectos registraron 13 retornos sobre 83 ingresos a pruebas, con una tasa de 15,66%. En 2026, cc-2026060 y cc-2026075 registraron 5 retornos sobre 54 ingresos, con una tasa de 9,26%, lo que equivale a una disminución de 6,40 puntos porcentuales. Cabe señalar que esta diferencia se explica principalmente por cc-2026075 (5,88%), ya que cc-2026060, de complejidad comparable a la de los proyectos de 2025, registró una tasa de 15,00%, similar a la de cc-2025007 (14,71%).

Tabla 7. Tasa de retrabajo por proyecto de la muestra (Portal Privado, Ene-Jun 2025 vs. Ene-Jun 2026).

Año	Proyecto	Retornos	Ingresos a pruebas	Tasa de retrabajo
2025	cc-2025007	10	68	14,71%
2025	cc-2025016	3	15	20,00%
Subtotal 2025		**13**	**83**	**15,66%**
2026	cc-2026060	3	20	15,00%
2026	cc-2026075	2	34	5,88%
Subtotal 2026		**5**	**54**	**9,26%**

Luego, para complementar la tasa de retrabajo, se decidió medir el tiempo de retrabajo de los proyectos de la muestra. Este tiempo se calculó para cada issue desde su primer retorno de “Liberado a pruebas” a desarrollo (“Asignado” o “No Solucionado”) hasta su cierre en “Terminado”, y se expresó en días hábiles (lunes a viernes de 9:00 a 18:00, hora de Chile, excluyendo feriados). La Tabla 8 resume los resultados.

Tabla 8. Tiempos de retrabajo por proyecto de la muestra (Portal Privado, días hábiles).

Proyecto	Año	Issues terminados	Issues con retrabajo	Retrabajo prom.	Retrabajo mediana	Tiempo de solución prom.	% del tiempo en retrabajo
cc-2025007	2025	58	8	2,3	1,5	3,7	8,5%
cc-2025016	2025	12	2	2,2	2,2	3,9	9,3%
Total 2025		**70**	**10**	**2,26**	**1,46**	**3,74**	**8,6%**
cc-2026060	2026	17	3	2,1	0,8	2,5	14,7%
cc-2026075	2026	35	2	1,1	1,1	2,3	2,8%
Total 2026		**52**	**5**	**1,71**	**0,98**	**2,38**	**6,9%**
Variación		**−18**	**−5**	**−24,1%**	**−32,4%**	**−36,5%**	**−1,7 pp**

Nota: las variaciones porcentuales se calcularon con los valores sin redondear, por lo que pueden diferir levemente de las que se obtienen con los valores mostrados en la tabla. En cc-2026075, 3 de los 35 issues terminados se registraron directamente en “Revisión Usuario” y se cerraron sin pasar por “Liberado a pruebas”; por ello, sus 34 ingresos a pruebas corresponden a 32 ingresos iniciales y 2 reingresos posteriores a retornos (Tabla 7).

La reducción del tiempo de retrabajo respalda, dentro de la muestra analizada, la mejora observada en la tasa de retrabajo. En 2026 no solo disminuyó la proporción de issues que volvieron a desarrollo (de 14,3% a 9,6%), sino que los retornos que sí ocurrieron se resolvieron en menos tiempo: el retrabajo promedio bajó de 2,26 a 1,71 días hábiles (−24,1%) y la mediana, de 1,46 a 0,98 días hábiles (−32,4%). Además, el retrabajo pasó de representar el 8,6% al 6,9% del tiempo total de solución. Este resultado es especialmente relevante porque se obtuvo con proyectos de complejidad media y alta, frente a proyectos de complejidad baja y media en 2025, y es consistente con lo esperado del enfoque SDD+TDD: que especificaciones más completas y pruebas definidas antes de la implementación reduzcan tanto la ocurrencia como el costo de las correcciones. Con todo, la comparación descansa en dos proyectos por período y en un número reducido de casos de retrabajo (10 issues en 2025 y 5 en 2026), por lo que el resultado debe interpretarse como evidencia favorable dentro de la muestra y no como una generalización al conjunto de sistemas.

En los demás sistemas, las diferencias entre períodos responden en gran medida a cambios en el volumen y la composición de la actividad. Portal Público Administrador de Mutuos pasó de una tasa de 5,26% en 2025 a 15,63% en 2026, pero sobre bases muy pequeñas: 1 retorno en 19 ingresos frente a 5 retornos en 32 ingresos, de modo que unos pocos casos adicionales mueven la tasa varios puntos. Algo similar ocurre con Portal Público, cuyo 0,00% de 2025 corresponde a 26 ingresos, frente a los 120 registrados en 2026. En otros casos la comparación no es posible: Portal Público Arriendos y Portal Público Venta de Seguros solo registraron actividad en 2025, mientras que Gestor de Campañas solo registró actividad en 2026, y Portal Privado Administrador de Mutuos (Pagos), con un proyecto en 2026, no registró ingresos a pruebas en ninguno de los períodos. Debido a esta distribución, varios sistemas no cuentan con observaciones suficientes en ambos períodos para establecer una tendencia individual consistente.

Por esta razón, la tasa de retrabajo total de ambos períodos no se considera una comparación válida, ya que agrupa proyectos y sistemas de complejidad muy distinta, con una distribución de actividad diferente en cada año. La evaluación de H1b se centra, en cambio, en la muestra de Portal Privado, en la que la complejidad de cada proyecto se evaluó de forma explícita y el retrabajo se midió tanto en frecuencia como en tiempo. El resto del análisis por sistema se entiende como información de contexto.

En la muestra de Portal Privado, la tasa de retrabajo disminuyó de 15,66% a 9,26% y el tiempo promedio de retrabajo, de 2,26 a 1,71 días hábiles, pese a que los proyectos de 2026 no presentaron una complejidad menor.

Ambos resultados apuntan en una dirección consistente con H1b. Sin embargo, la muestra es acotada y el resultado descansa en dos proyectos por período, por lo que la evidencia respalda una reducción del retrabajo, pero no permite atribuirla exclusivamente al enfoque evaluado.

6.3 Trazabilidad percibida (H1c)

Para conocer la percepción de los desarrolladores sobre la trazabilidad, se analizaron las respuestas obtenidas en la encuesta aplicada al finalizar el período de observación. En total, se contó con la participación de cuatro de los cinco ingenieros del equipo, quienes evaluaron nueve afirmaciones utilizando una escala Likert de 1 a 5. Estas afirmaciones abordaron aspectos como la trazabilidad, la claridad de los requerimientos, el control del proceso y la confianza en los artefactos generados con apoyo de inteligencia artificial. Los resultados reflejan, por lo tanto, la percepción de esos cuatro participantes.

Tabla 9. Resultados de la encuesta de percepción aplicada a los desarrolladores.

Ítem evaluado	**Promedio**
Identificación del requerimiento que dio origen a la funcionalidad	4,25
Relación entre pruebas automatizadas y requerimientos	3,00
Relación entre código y artefactos Spec/Design/Tasks	3,75
Nivel de detalle del requerimiento antes de implementar	4,50
Reglas de negocio y criterios de aceptación explícitos	4,50
Control sobre las decisiones técnicas	3,75
Revisión de artefactos generados por IA	4,25
Confianza en el código generado con apoyo de IA	3,75
Confianza en la cobertura de las pruebas generadas	3,25

Las nueve afirmaciones obtuvieron un promedio general de 3,89 sobre 5, una valoración mayoritariamente favorable del enfoque. Los resultados más altos se observaron en el nivel de detalle de los requerimientos antes de implementar y en la explicitación de reglas de negocio y criterios de aceptación, ambos con un promedio de 4,50. También fueron bien evaluadas la capacidad de identificar el requerimiento que dio origen a cada funcionalidad (4,25) y la posibilidad de revisar los artefactos generados por IA antes de avanzar a la siguiente etapa (4,25).

Los resultados fueron menos consistentes en los aspectos relacionados con las pruebas automatizadas. La vinculación entre estas y los requerimientos que validan obtuvo un promedio de 3,00, con respuestas que variaron entre 1 y 5, mientras que la confianza en su cobertura de los escenarios relevantes alcanzó 3,25. Esta dispersión coincide con los distintos niveles de adopción de TDD observados durante el piloto: algunos participantes percibieron una relación clara entre requerimientos y pruebas, mientras que otros continuaron apoyándose principalmente en revisiones manuales y pruebas en el ambiente de calidad.

Las respuestas abiertas complementan estos resultados. Entre los aspectos positivos, los participantes destacaron la posibilidad de iniciar el desarrollo con tareas mejor definidas, contar con una documentación más completa del análisis, comprender y resolver problemas con mayor rapidez y disponer de más contexto sobre el sistema. Como contrapartida, señalaron algunas dificultades, entre ellas el volumen de documentación

generado por OpenSpec, la complejidad de mantener la metodología cuando surgen cambios durante una iniciativa o intervienen varios proyectos, y el riesgo de perder conocimiento detallado del código cuando una parte importante de la implementación es generada por IA.

A partir de estas observaciones, los participantes propusieron flexibilizar el uso de OpenSpec mediante un enfoque más híbrido, utilizando funciones de Claude Code, y seguir destinando tiempo a revisiones y documentación del código.

En conjunto, la evidencia disponible respalda parcialmente H1c. Los tres ítems directamente asociados a la trazabilidad (identificación del requerimiento de origen, relación entre pruebas y requerimientos, y relación entre código y artefactos Spec/Design/Tasks) obtuvieron un promedio de 3,67, inferior al promedio general de 3,89. Los resultados muestran una percepción favorable de la trazabilidad entre requerimientos y funcionalidades, así como de la claridad con que se define el trabajo antes de comenzar el desarrollo. Sin embargo, esta percepción es menos sólida cuando la trazabilidad se extiende a las pruebas automatizadas. En este ámbito, el principal desafío consiste en transformar los criterios definidos en las especificaciones en pruebas ejecutables y explícitamente vinculadas con los requerimientos.

6.4 Síntesis general de resultados

Tabla 10. Síntesis de resultados por subhipótesis.

Subhipótesis	Indicador	2025	2026	Variación	Resultado
H1a - Calidad	Errores en producción	101	82	−18,8%	Consistente
H1b - Retrabajo	Tasa de retrabajo (muestra Portal Privado)	15,66%	9,26%	−6,40 pp	Consistente
H1c - Trazabilidad	Percepción general del enfoque (promedio de 9 ítems); ítems de trazabilidad: 3,67	—	3,89	No aplica	Parcialmente

Los resultados de H1a y H1b siguen la tendencia esperada según lo planteado en la hipótesis general. En 2026 se registraron menos errores en producción que en 2025 y, en la muestra de Portal Privado, una menor tasa y un menor tiempo de retrabajo, lo que sugiere una mejora en ambos indicadores. Sin embargo, estas diferencias fueron moderadas y no se presentaron de la misma manera en todos los sistemas analizados.

Este comportamiento debe interpretarse considerando las limitaciones señaladas en la sección 4.6. Los resultados permiten observar una asociación entre el enfoque utilizado y las mejoras obtenidas, pero no que dicho enfoque haya sido, por sí solo, su causa.

7 Conclusiones y proyecciones

Esta investigación nació de una tensión que se vivía a diario. La gerencia general presionaba por mostrar avances concretos con inteligencia artificial y, en el equipo de desarrollo, se sabía que la IA estaba ayudando, pero existía desconfianza respecto de lo que podía aportar cuando se trataba de reglas de negocio. El problema, según la percepción del equipo, estaba en cómo se le entregaban el contexto y los requerimientos a la IA: muchas veces no alcanzaba para representar lo que realmente necesitaba el negocio. Por eso se decidió evaluar la incorporación de un enfoque basado en Spec-Driven Development, integrado con prácticas de Test-Driven Development, buscando contar con una forma más estructurada de trabajar con inteligencia artificial durante el desarrollo.

Para evaluarlo se compararon dos períodos del mismo equipo: uno en que ya se usaba IA, pero sin especificaciones ni TDD formal, y otro en que se trabajó con OpenSpec, revisión humana y TDD, siempre con requerimientos y sistemas reales.

En relación con la calidad del software, durante el segundo período se registraron 82 errores de sistema en producción, frente a 101 durante el período anterior, lo que corresponde a una disminución de 18,8%. Al mismo tiempo, el número total de tickets se mantuvo relativamente estable, pasando de 256 a 261. Estos resultados son consistentes con lo planteado en H1a y muestran una mejora en el indicador utilizado en esta investigación para representar la calidad operacional.

En el caso del retrabajo, la tasa total de ambos períodos no resultó comparable, ya que los sistemas y proyectos desarrollados presentaron diferencias importantes de complejidad, tipo y volumen de trabajo. Por esta razón, se tomó como muestra Portal Privado y se evaluó previamente la complejidad de cada proyecto a partir de cinco atributos. En esta muestra, la tasa de retrabajo pasó de 15,66% durante 2025 a 9,26% durante 2026, y el tiempo promedio de retrabajo bajó de 2,26 a 1,71 días hábiles, pese a que los proyectos de 2026 fueron de complejidad media y alta, y los de 2025, de complejidad baja y media. La baja del tiempo de retrabajo respalda, dentro de la muestra, la mejora observada en la tasa y muestra un resultado consistente con H1b, aunque acotado a un sistema y a un número reducido de proyectos.

La encuesta realizada a los desarrolladores permitió observar otro aspecto del enfoque que no era posible identificar solamente a partir de los indicadores anteriores. Los participantes valoraron favorablemente la forma de trabajo: 3,89 sobre 5 en promedio. Los aspectos mejor evaluados fueron el nivel de detalle de los requerimientos antes de comenzar la implementación y la definición explícita de reglas de negocio y criterios de aceptación, ambos con un promedio de 4,50. Esto coincide con uno de los objetivos principales del enfoque: entregar mayor contexto antes de comenzar a generar una solución con apoyo de inteligencia artificial.

Los resultados relacionados con las pruebas automatizadas, en cambio, fueron menos claros. La relación entre pruebas y requerimientos obtuvo un promedio de 3,00, mientras que la confianza en la cobertura de las pruebas alcanzó 3,25. Tampoco se aplicó TDD con la misma intensidad en todas las funcionalidades. Se estima que faltó rigurosidad en el equipo para sostener la práctica, y probablemente fue un error incorporar SDD y TDD al mismo tiempo: eran demasiados cambios de una sola vez. Con estos resultados, H1c queda respaldada solo en parte. La trazabilidad entre los requerimientos y la implementación fue percibida de manera favorable, pero todavía existen oportunidades para fortalecer la relación entre las especificaciones y las pruebas automatizadas.

Uno de los aprendizajes que surgió durante la aplicación del enfoque fue que disponer de una especificación no significa necesariamente que el problema esté correctamente definido. La experiencia realizada antes del segundo período mostró que, cuando una especificación no incorpora suficiente contexto funcional, reglas de negocio o escenarios relevantes, la inteligencia artificial puede generar una solución técnicamente correcta pero incompleta desde la perspectiva del usuario. Esto ocurrió con una calculadora de seguro de desgravamen: el requerimiento no incluía todas las reglas de negocio del producto y, aunque el código funcionaba, el cálculo final no entregaba los valores correctos. Esta situación llevó a reforzar el levantamiento de información y las instancias de revisión durante el piloto.

A partir de esta experiencia, se observó que el aporte de las especificaciones no estuvo solamente en documentar un requerimiento, sino en disponer de una referencia común durante el desarrollo. Esta referencia permitió entregar mayor contexto a las herramientas de inteligencia artificial y, al mismo tiempo, mantener visibles las reglas y criterios que debía cumplir la implementación. La revisión humana continuó siendo necesaria durante este proceso, especialmente para validar que los artefactos generados representaran correctamente las necesidades del negocio antes de avanzar hacia las siguientes etapas.

El enfoque también tuvo costos, y los desarrolladores los identificaron con claridad. En las respuestas abiertas de la encuesta aparecieron tres dificultades recurrentes: el volumen de documentación que genera OpenSpec, el esfuerzo de mantener la metodología cuando una iniciativa cambia de alcance o intervienen varios

proyectos en paralelo, y el riesgo de perder conocimiento detallado del código cuando una parte importante de la implementación se genera con apoyo de inteligencia artificial. Los propios participantes propusieron un uso más híbrido de OpenSpec y destinar tiempo específico a revisar y documentar el código. Estas observaciones acotan el alcance del enfoque: entregar más contexto a la herramienta tiene un costo de documentación y de atención que el equipo debe administrar, y no todas las iniciativas lo justifican en la misma medida.

Estos resultados deben ser considerados dentro de las características y limitaciones del estudio. Entre ambos períodos no solamente cambió la forma de trabajo, sino también la herramienta de inteligencia artificial utilizada, pasando de GitHub Copilot a Claude Code. Asimismo, los proyectos desarrollados presentaron diferencias de alcance y complejidad, y TDD no fue aplicado de manera homogénea. Debido a estas condiciones, no es posible atribuir los cambios observados exclusivamente a SDD y TDD. Los resultados permiten describir lo ocurrido durante la aplicación del enfoque dentro del caso estudiado, pero no establecer una relación causal ni asumir que el mismo comportamiento se producirá necesariamente en otros equipos.

Actualmente, SDD se sigue utilizando como parte de la metodología del equipo y, después del piloto, la práctica se extendió a los otros equipos de la empresa. Eso abre la posibilidad de evaluar el enfoque durante un período más largo y con más equipos. Una evaluación futura también podría mantener la misma herramienta de inteligencia artificial entre los períodos y extender a todos los sistemas la evaluación previa de complejidad que en este estudio se aplicó solo a la muestra. Esto permitiría realizar comparaciones bajo condiciones más similares y disminuir algunas de las variables que estuvieron presentes en este estudio.

Con TDD el avance ha sido más lento. Fue el punto menos consolidado del piloto y todavía presenta dificultades para madurar su implementación, por lo que se continúa trabajando para darle importancia e integrarlo efectivamente al flujo. El siguiente paso identificado es avanzar hacia una vinculación más explícita entre los criterios de aceptación definidos en las especificaciones y las pruebas que posteriormente los validan.

Finalmente, una posible evolución del enfoque consiste en incorporar mecanismos que permitan mantener la trazabilidad entre los artefactos Spec, Design y Tasks, las pruebas automatizadas y los cambios realizados en el código. Esto permitiría complementar la percepción de los desarrolladores utilizada en esta investigación con indicadores objetivos de trazabilidad y estudiar con mayor precisión cómo se mantiene la intención original de un requerimiento a medida que avanza por las distintas etapas del desarrollo.

Referencias

[1] D. B. Piskala, “Spec-Driven Development: From Code to Contract in the Age of AI Coding Assistants,” arXiv preprint arXiv:2602.00180, 2026.

[2] N. Gupta, “Small Language Models and Spec-Driven Development for High-Accuracy Agentic Frameworks,” SSRN Working Paper, 2025.

[3] Google Cloud, “What is Vibe Coding?” Disponible en: https://cloud.google.com/discover/what-is-vibe-coding

[4] OpenSpec Project, “OpenSpec Documentation.” Disponible en: https://openspec.dev

[5] N. S. Mathews and M. Nagappan, “Test-Driven Development for Code Generation,” arXiv preprint arXiv:2402.13521, 2024.

[6] S. Piya and A. Sullivan, “LLM4TDD: Best Practices for Test Driven Development Using Large Language Models,” in Proceedings of the 1st International Workshop on Large Language Models for Code (LLM4Code), 2024.

[7] S. Fakhoury, A. Naik, G. Sakkas, S. Chakraborty, and S. K. Lahiri, "LLM-Based Test-Driven Interactive Code Generation: User Study and Empirical Evaluation," IEEE Transactions on Software Engineering, vol. 50, pp. 2254–2268, 2024.

[8] G. Rosa, D. Moreno-Lumbreras, G. Robles, and J. M. González-Barahona, "Understanding Specification-Driven Code Generation with LLMs: An Empirical Study Design," 2026.

Anexos

Anexo A. Proyectos considerados y base de cálculo del retrabajo

La Tabla A.1 presenta los proyectos registrados en Redmine para cada período (Ene-Jun 2025 y Ene-Jun 2026) y la base de cálculo de la tasa de retrabajo. Los retornos corresponden a las transiciones de "Liberado a pruebas" a "No Solucionado" o "Asignado", y los ingresos corresponden a toda transición cuyo estado nuevo es "Liberado a pruebas". La tasa se calcula como retornos / ingresos × 100. Cada proyecto se asignó al período en que registró actividad, independientemente del año indicado en su código. No se registraron reaperturas posteriores al cierre ("Terminado" a "Asignado") en ninguno de los períodos.

Tabla A.1. Retornos, ingresos a pruebas y tasa de retrabajo por proyecto (Ene-Jun 2025 vs. Ene-Jun 2026).

Proyecto	Año	Sistema	Retornos	Ingresos a pruebas	Tasa de retrabajo (%)
qase-demo	2025	Portal Público Venta de Seguros	0	1	0,0
cc-2024162	2025	Portal Privado Administrador de Mutuos	55	412	13,3
cc-2024022	2025	Plataforma de Atención de Clientes	4	61	6,6
cc-2024141	2025	Portal Público	0	26	0,0
cc-2024153	2025	Portal Público Administrador de Mutuos	1	19	5,3
cc-2024154	2025	Portal Público Arriendos	0	21	0,0
cc-2025007	2025	Portal Privado	10	68	14,7
cc-2025011	2025	Portal Público Arriendos	2	6	33,3
cc-2025016	2025	Portal Privado	3	15	20,0
Total 2025		**9 proyectos**	**75**	**629**	**11,9**
cc-2025142	2026	Portal Público	5	30	16,7
cc-2026031	2026	Portal Público	5	60	8,3

Proyecto	Año	Sistema	Retornos	Ingresos a pruebas	Tasa de retrabajo (%)
cc-2026005	2026	Portal Público	4	30	13,3
cc-2026007	2026	Gestor de Campañas	17	146	11,6
cc-2026008	2026	Plataforma de Atención de Clientes	8	95	8,4
hipotecon	2026	Portal Privado Administrador de Mutuos	0	2	0,0
cc-2026043	2026	Portal Público Administrador de Mutuos	5	32	15,6
cc-2026036	2026	Portal Privado Administrador de Mutuos	7	25	28,0
cc-2026039	2026	Portal Privado Administrador de Mutuos (Pagos)	0	0	0,0
cc-2026054	2026	Portal Privado Administrador de Mutuos	2	43	4,7
cc-2026057	2026	Portal Privado Administrador de Mutuos	4	67	6,0
cc-2026060	2026	Portal Privado	3	20	15,0
cc-2026075	2026	Portal Privado	2	34	5,9
cc-2026068	2026	Gestor de Campañas	1	30	3,3
cc-2026092	2026	Plataforma de Atención de Clientes	0	3	0,0
Total 2026		**15 proyectos**	**63**	**617**	**10,2**

Fuente: elaboración propia a partir de los registros de Redmine.

Anexo B. Rúbrica de evaluación previa de complejidad

La Tabla B.1 presenta la rúbrica utilizada para construir la Tabla 6. Cada proyecto de la muestra se puntuó de 1 a 5 en cada variable, considerando solo la información disponible antes de iniciar el desarrollo. El puntaje

ponderado se obtiene como la suma de cada puntaje multiplicado por su peso, y el índice de complejidad previa se normaliza entre 0 y 1 como (puntaje ponderado − 1) / 4.

Tabla B.1. Rúbrica de evaluación previa de complejidad.

Variable	Peso	1 – Muy baja	2 – Baja	3 – Media	4 – Alta	5 – Muy alta
Tipo de desarrollo	25%	Ajuste cosmético o de textos	Ajuste menor de funcionalidad	Modificación relevante de funcionalidad existente	Nueva funcionalidad sobre sistema existente	Desarrollo o sistema nuevo desde cero
Canales o sistemas impactados	20%	1 canal, pantalla acotada	1 canal, varios módulos	2 canales	3 canales	4 o más canales / transversal
Integraciones	15%	Ninguna (todo interno)	1 servicio interno existente	2–3 servicios internos o 1 nuevo	Varias internas o 1 externa/tercero	Múltiples externas/terceros o core
Criticidad normativa	25%	Sin exigencia	Buena práctica o recomendación	Estándar o política interna	Regulatorio sin plazo crítico / auditoría	Obligación legal con plazo fijo (NCG, ley)
Claridad de requerimientos	15%	Claros, cerrados y validados	Claros, con detalles menores pendientes	Algunas definiciones pendientes	Definiciones importantes pendientes	Alta incertidumbre o cambios frecuentes

Clasificación: Baja si el índice es menor que 0,33; Media si es mayor o igual que 0,33 y menor que 0,66; Alta si es mayor o igual que 0,66.

Fuente: elaboración propia.