



GitLab-EffortMiner: Un framework para la extracción de datos de entornos DevOps para la estimación de esfuerzo

GitLab-EffortMiner: A framework for data extraction from DevOps environments for effort estimation

Octavio Martínez Hernández ✉

Universidad Tecnológica de la Mixteca, Huajuapán de León, México
maho000826@gs.utm.mx
ORCID: 0009-0001-5822-9401

Verónica Rodríguez López

Universidad Tecnológica de la Mixteca, Huajuapán de León, México
veromix@gs.utm.mx
ORCID: 0000-0002-5976-9338

Carlos Alberto Fernández y Fernández

Universidad Tecnológica de la Mixteca, Huajuapán de León, México
caff@mixteco.utm.mx
ORCID: 0000-0002-1586-8772



<https://doi.org/10.36825/RITI.14.35.002>

Recibido: Junio 14, 2026

Aceptado: Octubre 02, 2026

Resumen: La estimación de esfuerzo en proyectos de software es una actividad fundamental para la planificación, que puede verse afectada por la falta de datos históricos y por la experiencia del equipo. Este artículo presenta un estudio preliminar sobre el desarrollo de un *framework* para la extracción de información en repositorios de software, orientado a la estimación de esfuerzo en el desarrollo ágil de proyectos. El *framework* está constituido por procesos para minar la información, transformarla y adaptarla para ajustar la complejidad de los puntos de historia de acuerdo con el *método de Generalized Estimation Method*. La utilidad de los datos obtenidos por el *framework* se validó mediante modelos de regresión para la estimación de puntos ajustados, construidos a partir de datos extraídos de 33 proyectos de código abierto provenientes de *User Story NeoDataset*. Los resultados experimentales indican que la información obtenida por el *framework* es de utilidad para estimar el esfuerzo.

Palabras clave: Ágiles, Estimación de Esfuerzo, Historias de Usuario, Ingeniería de Software, Repositorio.

Abstract: Effort estimation in software projects is a fundamental planning activity that can be affected by the lack of historical data and level of experience of the teams. This article presents a preliminary study on the development of a framework for extracting information from software repositories, aimed at effort estimation in agile software development projects. The framework consists of processes for mining, transforming, and adapting information to adjust the complexity of story points in accordance with the Generalized Estimation Method. The usefulness of the data obtained through the framework was validated using regression models for estimating adjusted story

points, built with data extracted from 33 open-source projects from the *User Story NeoDataset*. The experimental results indicate that the information obtained through the framework is useful for estimating effort.

Keywords: *Agile, Effort Estimation, User Stories, Software Engineering, Repository.*

1. Introducción

La estimación de esfuerzo es una actividad de planificación que permite prever los recursos necesarios para desarrollar un proyecto de software, considerando aspectos como tiempo, costo, alcance, personal, calidad y riesgo [1], [2]. En este contexto, dicha actividad se realiza en las etapas tempranas del proyecto, cuando se toman decisiones relacionadas con el presupuesto, el calendario y la asignación de recursos [3].

En entornos ágiles, la estimación suele apoyarse en historias de usuario y puntos de historia como medida relativa del esfuerzo [4], [5]. La calidad de estas estimaciones depende de la experiencia del equipo y de la información disponible sobre el proyecto [6], [7]. Ante estas limitaciones, el aprendizaje automático ha surgido como una alternativa viable gracias a su capacidad para generar modelos predictivos a partir de datos históricos [8], [9], aunque su desempeño depende de la disponibilidad y la calidad de los datos [10], [11]. Específicamente, existen pocos conjuntos de datos públicos para generar modelos de predicción de esfuerzo en proyectos ágiles. Algunos conjuntos disponibles, como *Agile User Story Point Estimation* [12] y *User Story NeoDataset* [13], incluyen información textual sobre las historias de usuario y su complejidad, pero omiten características del equipo de desarrollo y del proyecto, lo que limita el desarrollo de modelos de predicción de esfuerzo. Aunque *TAWOS* [14] es un conjunto más completo, integra información de proyectos administrados mediante Jira, como historias de usuario, tareas y errores. Esta información resulta adecuada para gestionar y dar seguimiento al trabajo del equipo de desarrollo.

Los repositorios de software representan una fuente importante para desarrollar modelos de predicción de esfuerzo, ya que, además de almacenar evidencia sobre la actividad de desarrollo, registran características del proyecto y del equipo [15], [16]. Esta información se encuentra distribuida en diferentes partes del repositorio y presenta distintos niveles de calidad, por lo que se requieren de procesos para seleccionarla, integrarla y transformarla en datos útiles para desarrollar modelos de predicción de esfuerzo.

El presente trabajo tiene como objetivo proponer y evaluar un *framework* para extraer y transformar información de repositorios *GitLab* que sea útil para desarrollar modelos predictivos de estimación de esfuerzo en proyectos ágiles de software. A partir de este objetivo se plantean las siguientes preguntas de investigación:

- **PI1:** ¿Qué información puede extraerse de repositorios de *GitLab* que sea útil en la estimación del esfuerzo de proyectos ágiles de software?
- **PI2:** ¿En qué medida la información obtenida y transformada mediante *GitLab-EffortMiner* permite desarrollar modelos predictivos de puntos de historia?

Esta investigación contribuye con el *framework* denominado *GitLab-EffortMiner*, que adapta la información extraída de los repositorios *GitLab* a los factores considerados por el método *Generalized Agile Estimation Method* (GEM) [17] para la estimación de esfuerzo. Esta herramienta permite integrar un conjunto de datos que apoye el desarrollo y la validación de modelos predictivos de esfuerzo ágil.

El resto del artículo se organiza de la siguiente manera. La Sección 2 presenta el estado del arte sobre la estimación de esfuerzo en proyectos ágiles a partir de información extraída de repositorios de software. La Sección 3 describe el Método de Estimación Generalizado (GEM). La Sección 4 presenta *GitLab-EffortMiner* y detalla los procesos de minería y transformación de la información de *GitLab* en factores GEM. En la Sección 5 se describen los resultados de la evaluación de la información extraída por *GitLab-EffortMiner*. La Sección 6 discute los principales hallazgos, las amenazas a la validez, las limitaciones y el trabajo futuro. Finalmente, la Sección 7 presenta las conclusiones.

2. Estado del arte

En el desarrollo ágil, el esfuerzo suele estimarse mediante el juicio de expertos, *Planning Poker* y puntos de historia. Aunque estas técnicas son ampliamente utilizadas, su precisión depende tanto de la información disponible como de la experiencia y la madurez del equipo [18], [19], [20]. Además, se ha observado que las estimaciones relativas, como los puntos de historia, dependen del desarrollador y no necesariamente son más precisas que las expresadas en unidades de tiempo [21].

Para reducir la dependencia del juicio humano, diversas investigaciones han explorado el uso de datos extraídos de repositorios de software para construir modelos de predicción del esfuerzo en proyectos ágiles. Algunos trabajos se han enfocado en extraer el texto de los *issues* de proyectos de código abierto para entrenar modelos de lenguaje orientados a la predicción de puntos de historia [6], [22], [23]. Aunque estos modelos muestran un buen desempeño predictivo, ignoran características importantes del equipo de desarrollo y del proyecto de software que podrían afectar las estimaciones.

Por otra parte, diversos estudios han mostrado que la actividad de desarrollo registrada en plataformas DevOps puede utilizarse como fuente de información para apoyar la estimación del esfuerzo [24], [25], [26], [27], [28]]. Algunas investigaciones utilizan información extraída directamente de proyectos de GitHub, como características relacionadas con el tamaño funcional y la participación de los desarrolladores, para construir modelos de estimación de esfuerzo [24]. Otros estudios analizan la información alojada en repositorios de GitHub y GitLab para transformarla en métricas aplicables a la estimación. El número de *commits* y los perfiles de actividad se han utilizado para distinguir niveles de dedicación y aproximar el esfuerzo en personas-mes [26]. El historial de cambios, mediante las relaciones entre desarrolladores y archivos modificados, ha permitido obtener métricas de colaboración y productividad [25]. Asimismo, los *commits* y *issues* se han transformado en métricas para analizar la actividad de los proyectos [27]. Para estudiar el volumen de cambios, los tiempos de revisión y la experiencia de los participantes, se ha explorado el uso de métricas generadas a partir de los *merge requests* [28]. En resumen, estos antecedentes evidencian la necesidad de generar herramientas que apoyen tanto la minería de información de los repositorios como su transformación en datos útiles para estimar el esfuerzo.

3. Método de Estimación Generalizado (GEM)

El *Generalized Estimation Method* (GEM) [17] estima el tiempo de desarrollo ágil a partir de la complejidad de las historias de usuario y de las características del proyecto y el equipo. Una historia de usuario representa brevemente un requisito y suele seguir la estructura: Como [tipo de usuario], quiero [acción], para [beneficio] [29]. Su esfuerzo relativo se expresa en puntos de historia (*SP*), que GEM ajusta según la complejidad del dominio y los factores vitales mediante:

$$ASP = SP + CD + U \cdot \sum_{i=1}^k CF_i, \quad (1)$$

donde CD es la complejidad del dominio, U la incertidumbre del proyecto y CF_i la complejidad del factor vital i . La complejidad del dominio (CD) refleja el propósito y entorno del sistema. La incertidumbre, con valores entre 0 y 1, representa el riesgo o variabilidad asociados con requisitos, tecnología y recursos. Los factores vitales representan recursos, herramientas, métodos, equipo y entorno. Su complejidad, CF_i , es el producto del peso w_i y la intensidad I_i . El peso, entre 0 y 1, expresa la importancia relativa del factor según el subdominio y los pesos suman uno. La intensidad indica un impacto bajo, medio o alto, normalmente ponderado por el líder del equipo.

4. GitLab-EffortMiner

El *framework GitLab-EffortMiner*, implementado en el lenguaje de programación Python versión 3.12.9, funciona para proyectos de desarrollo ágil de software alojados en *Gitlab* que incluyen historias de usuario terminadas con información textual acerca de la funcionalidad del software. Para lograr su objetivo integra los procesos de minería de información, transformación a factores GEM, y estimación de esfuerzo.

GitLab-EffortMiner se enfoca principalmente en la extracción de datos de cuatro etapas esenciales de la fase de desarrollo (CI) del ciclo DevOps: Planificación y Seguimiento, a través de la gestión de *issues*; desarrollo (CI) del ciclo DevOps: Planificación y Seguimiento, a través de la gestión de *issues*; Codificación y Control de

Versiones, mediante el análisis de *commits*; Integración y Revisión, evaluando las solicitudes de fusión o *merge requests*; y Construcción y Pruebas Automatizadas, por medio del rendimiento de los *pipelines*. Al centrarse en estos artefactos, la herramienta no obtiene métricas vinculadas al despliegue, la operación y la monitorización continua que son consideradas por otras propuestas [30].

4.1. Minería de información

El proceso de minería de información tiene como objetivo ampliar los datos originales de los proyectos de software. Para acceder a esta información, se requiere autenticación mediante un *token* de acceso a la API de *GitLab*, el cual cumple una doble función: habilitar el acceso a datos que no están disponibles en consultas anónimas, ampliando el límite de solicitudes permitidas y automatizar el proceso sin que las consultas consecutivas generen bloqueos o restricciones por parte de la plataforma. Una vez establecida la autenticación, se consulta la API por cada proyecto. Para optimizar el proceso, se trabaja con una lista de los identificadores de los proyectos de software. Se toman en cuenta aspectos técnicos como la paginación de resultados, el control de la tasa de solicitudes, así como la implementación de mecanismos de reintento ante fallos de conexión, para garantizar la estabilidad del proceso y minimizar la pérdida de información.

La información que extrae *GitLab-EffortMiner* se describe en la Tabla 1 y está relacionada con los siguientes aspectos del proyecto de software:

1. **Información general del proyecto:** Considera los datos que permiten identificar y contextualizar cada proyecto: nombre, espacio al que pertenece (organización o usuario), tipo de visibilidad (público o privado) y fechas relevantes como la de creación y la de última actividad.
2. **Actividad de desarrollo:** Incluye las métricas que reflejan el volumen de trabajo registrado en el repositorio: cambios realizados al código, tareas documentadas, solicitudes de integración y ejecuciones de integración continua. Estos indicadores permiten evaluar qué tan activo es un proyecto y cómo ha evolucionado a lo largo del tiempo.
3. **Participación del equipo:** Comprende las características de las personas que contribuyen al proyecto, incluyendo el número total de participantes. Esta información resulta útil para estimar el tamaño del equipo y su grado de involucramiento en el desarrollo del software.
4. **Características técnicas:** Incluye la información sobre las tecnologías empleadas, en particular el número de lenguajes utilizados y los lenguajes de programación predominantes. Estos datos permiten identificar el perfil técnico del proyecto y las herramientas sobre las que está constituido.
5. **Popularidad e interacción:** Considera los indicadores que reflejan el interés que genera el proyecto en la comunidad, como la cantidad de veces que ha sido marcado como favorito o reutilizado como base para otros desarrollos.

Tabla 1. Descripción de la información extraída de los proyectos de *GitLab*.

Variable	Descripción
Identificador del proyecto	Identificador único asignado por <i>GitLab</i> .
Nombre del proyecto	Nombre con el que se identifica el proyecto dentro de <i>GitLab</i> .
Espacio de trabajo del proyecto	Ubicación del proyecto dentro de <i>GitLab</i> .
Fecha de creación	Fecha y hora en que el proyecto fue creado en <i>GitLab</i> .
Fecha de última actividad	Fecha y hora de la actividad más reciente dentro del proyecto.
Visibilidad del proyecto	Nivel de acceso del repositorio, por ejemplo, si es público o privado.
Número de estrellas	Indicador de popularidad, interés o relevancia del repositorio dentro de la comunidad.
Cantidad de <i>forks</i>	Grado de reutilización.
Cantidad total de issues abiertos	Número de tareas que permanecen abiertas en el proyecto.
Estado de archivado	Indica si el proyecto se encuentra archivado.

Variable	Descripción
Total de <i>commits</i>	Número total de confirmaciones de cambios registradas en el repositorio.
Total de versiones liberadas	Cantidad de versiones o entregas publicadas del proyecto.
Total de incidencias	Número total de incidencias o tareas registradas en el proyecto.
Cantidad de <i>merge requests</i>	Cantidad de solicitudes de integración realizadas en el proyecto
Cantidad total de pipelines	Indica el número de ejecuciones de pipelines asociadas al proyecto.
Lenguaje principal	Lenguaje de programación predominante en el proyecto.
Cantidad de lenguajes utilizados	Lenguajes más representativos del proyecto.
Cantidad total de colaboradores del proyecto	Número total de personas que han colaborado en el proyecto.
Principales colaboradores	Lista los colaboradores con mayor participación dentro del repositorio.
Puntos de historia	Valor asignado a la historia de usuario para representar su complejidad.
Fecha de creación de la historia de usuario	Fecha en que la historia de usuario fue registrada.
Fecha de finalización de la historia de usuario	Fecha en que la historia de usuario fue finalizada.

Fuente: Elaboración propia.

4.2. Transformación a factores GEM

Este proceso se encarga de transformar la información extraída en el proceso anterior a los factores requeridos por GEM. Este proceso, que se resume en el Algoritmo 1, primero relaciona la información extraída de los repositorios con siete factores vitales de GEM, tal y como se describe en la Tabla 2.

Algoritmo 1: Transformación de la información de un repositorio <i>GitLab</i> en factores GEM.
Entrada: Proyecto de GitLab previamente minado P
Entrada: Historias de usuario del proyecto con puntos de historia SP_j
Entrada: Perfil de dominio y pesos de la Tabla 3
Entrada: Tabla de perfiles con CD y pesos $w_1, \dots, w_7$
Salida: Factores GEM $CF_1, \dots, CF_7$ y puntos de historia ajustados ASP_j
1: Transformar las métricas obtenidas de la minería ($G_1, \dots, G_7$) en factores GEM($F_1, \dots, F_7$), según la Tabla 2
2: Identificar el dominio y subdominio del proyecto P
3: Obtener la complejidad del dominio (CD) de la Tabla 3
4: Obtener los pesos asociados a cada uno de los factores GEM ($w_1, \dots, w_7$) de la Tabla 3
5: Calcular la mediana (M) de las siete métricas del proyecto ($G_1, \dots, G_7$)
6: Obtener la intensidad de cada uno de los factores GEM como $I_i, i = 1, 2, 3, \dots, 7$ como:
$I_i = \begin{cases} 1 (\text{Intensidadbaja}) & \text{si } F_i < M \\ 5 (\text{Intensidadmedia}) & \text{si } F_i = M \\ 9 (\text{Intensidadalta}) & \text{si } F_i > M \end{cases}$
7: Calcular la complejidad del factor vital: $CF_i \leftarrow I_i \cdot w_i$
8: Calcular la incertidumbre del proyecto: $U \leftarrow \frac{G_1 + G_2 + G_5 + G_6}{4 \cdot \max\{G_1, G_2, G_5, G_6\}}$
9: Ajustar las historias de usuario utilizando la Ecuación 1
10: return $CD, U, CF_1, \dots, CF_7$ y los valores de ASP_j

Tabla 2. Asociación de características del repositorio en Gitlab con factores vitales de GEM.

Variables <i>GitLab</i>	Factor vital GEM	Justificación
G1: Cantidad de colaboradores del proyecto	F1: Tamaño del equipo	Los colaboradores representan las personas que participan en el desarrollo del proyecto.
G2: Cantidad de lenguajes utilizados	F2: Lenguaje	Indica la diversidad tecnológica del proyecto; a mayor número de lenguajes, mayor complejidad técnica potencial.
G3: Cantidad total de pipelines	F3: Herramientas de automatización	Los <i>pipelines</i> corresponden a procesos automáticos de integración, pruebas o despliegue.
G4: Cantidad total de <i>issues</i>	F4: Técnicas de gestión de proyectos	Refleja las técnicas de gestión del proyecto, al representar cómo se registran, organizan y controlan las tareas.
G5: Cantidad total de <i>merge requests</i>	F5: Integración	Los <i>merge requests</i> corresponden a cambios revisados e incorporados al repositorio principal.
G6: Cantidad de <i>issues</i> abiertos	F6: Seguridad	Representa problemas pendientes o no resueltos dentro del proyecto.
G7: Cantidad de <i>forks</i>	F7: Reusabilidad	Los <i>forks</i> representan su adaptación o reutilización del proyecto.

Fuente: Elaboración propia.

A continuación, *GitLab-EffortMiner* obtiene los pesos e intensidades necesarios para establecer la complejidad de cada factor vital. En la Tabla 3 se describe la propuesta de pesos asociados a cada factor vital, de acuerdo al dominio y subdominio del proyecto, que fue elaborada a partir de los criterios de complejidad identificados por [31], [32]. El factor de intensidad (I_i) asociado a cada factor vital i se determina a partir de la mediana de todos los factores vitales. *GitLab-EffortMiner* asigna un valor de 1 (intensidad baja) cuando el factor vital es menor a la mediana, 5 (intensidad media) cuando es igual y 9 (intensidad alta) si es mayor.

Posteriormente, a partir del tamaño del equipo (G_1), la cantidad de lenguajes (G_2), el total de *merge requests* registrados (G_5) y el número de *issues* sin resolver (G_6), *GitLab-EffortMiner* obtiene el factor de incertidumbre del proyecto (U) aplicando la expresión:

$$U = \frac{G_1 + G_2 + G_5 + G_6}{4 \cdot \max G_1, G_2, G_5, G_6} \quad (2)$$

Este cálculo se fundamenta en el modelo GEM, el cual considera que la incertidumbre de un proyecto está asociada principalmente con tres niveles: los requisitos, la tecnología y los recursos disponibles. El tamaño del equipo (G_1) y el número de *issues* sin resolver (G_6) están asociados con el nivel de recursos del proyecto [18], [33]. Por su parte, la cantidad de lenguajes de programación empleados (G_2) está relacionada con la dimensión tecnológica [34], y el total de *merge requests* registrados (G_5) representa una aproximación del nivel de requisitos [28].

Finalmente, después de haber obtenido la complejidad de los factores vitales y su incertidumbre, *GitLab-EffortMiner* ajusta la complejidad de cada historia de usuario usando la Ecuación **¡Error! No se encuentra el origen de la referencia..**

5. Resultados

En esta sección se muestran los resultados preliminares obtenidos al evaluar la utilidad del conjunto de datos generado por *GitLab-EffortMiner*. Esta utilidad se evaluó a partir del desempeño de modelos de regresión para predecir los puntos de historia ajustados.

Tabla 3. Complejidad del dominio (CD) y pesos de los factores vitales de acuerdo al perfil de dominio y subdominio del proyecto.

Dominio	CD	Tamaño del equipo w1	Lenguaje w2	Herramientas automatizadas w3	Técnicas de gestión w4	Integración w5	Seguridad w6	Reusabilidad w7
Software crítico								
SS: Sistema de software	4	0.10	0.10	0.16	0.16	0.10	0.15	0.08
SM: Software militar	9	0.07	0.10	0.12	0.15	0.06	0.25	0.05
SV: Software que salva vidas	5	0.08	0.10	0.12	0.14	0.08	0.22	0.06
Software de aplicación								
AW: Aplicación web	1	0.10	0.09	0.18	0.12	0.14	0.12	0.12
SI: Sistema de información	4	0.12	0.10	0.14	0.16	0.12	0.12	0.12
SC: Software comercial	4	0.12	0.09	0.16	0.12	0.14	0.12	0.14
SB: Software subcontratado	2	0.10	0.12	0.13	0.18	0.10	0.14	0.10
AUF: Aplicación de usuario final	1	0.14	0.11	0.13	0.09	0.14	0.12	0.16

Fuente: Elaboración propia.

5.1. Configuración experimental

En la evaluación se consideraron los proyectos de código abierto descritos en el conjunto de datos *User Story User Story NeoDataset* [13]. Este conjunto de datos comprende 20,479 historias de usuario finalizadas, extraídas de 33 proyectos de código abierto alojados en GitLab que emplean metodologías ágiles de desarrollo de software. A partir de los identificadores de los 33 proyectos, GitLab-EffortMiner realizó una consulta a la API de GitLab y recuperó 169,401 *issues* cerrados. Después de eliminar duplicados y conservar únicamente los registros con identificadores, puntos de historia y fechas de creación y finalización válidos, se integró un conjunto 25,740 historias de usuario descritas mediante su complejidad en puntos de historia sin ajustar (*SP*) y ajustada (*ASP*), su duración en real en días, así como a través de los descriptores del proyecto al que pertenecen, extraídos del repositorio ($G_1, G_2, G_3, G_4, G_5, G_6, G_7$) y su transformación a complejidad de los factores GEM ($CF_1, CF_2, CF_3, CF_4, CF_5, CF_6, CF_7, CD$). Este conjunto de datos fue depurado eliminando las cuatro historias de usuario asociadas al proyecto Lazarus y las historias de usuario con una duración mayor que 900 días. Por lo que el conjunto de datos usado en los experimentos quedó constituido por 24,062 historias de usuario. La descripción de los estadísticos de este conjunto de datos se presenta en la Tabla 4.

La utilidad de la información extraída por GitLab-EffortMiner se evaluó a través de dos experimentos:

- Experimento 1: Evaluación del impacto de los pesos e intensidades de los factores GEM en la precisión de los modelos de predicción de puntos de historia ajustados (Modelos GEM).
- Experimento 2: Análisis de la influencia de la complejidad de los factores vitales ajustados sobre el rendimiento de los modelos de predicción de puntos de historia ajustados (Modelos GitLab).

En los experimentos se consideró el aprendizaje de cuatro tipos de modelos de predicción: Regresión lineal, Árbol de decisión, Bosque aleatorio y *Gradient Boosting*. Para evaluar objetivamente su desempeño, se utilizó el 80%

del conjunto de datos para entrenar los modelos y el 20% restante para su validación. Además, en el entrenamiento se aplicó validación cruzada con $K = 5$. Como métricas del desempeño de los modelos se utilizaron el error cuadrático medio (MSE), el error absoluto medio (MAE) y R cuadrado (R^2). En cada iteración de la validación cruzada, los registros con puntos de historia ajustados extremos (identificados de acuerdo al criterio de rango intercuartil) se reemplazaron por la mediana. Además, el conjunto de entrenamiento fue normalizado aplicando Z-score.

Tabla 4. Estadísticos descriptivos del conjunto de datos obtenido con *GitLab-EffortMiner*.

	Media	Mediana	Desviación estándar	Mínimo	Máximo
GitLab					
G1: Tamaño del equipo	63.98	52.00	74.22	1.00	500.00
G2: Cantidad de lenguajes	4.90	5.00	0.33	3.00	5.00
G3: Cantidad total de <i>pipelines</i>	594.27	1.00	1782.64	1.00	8234.00
G4: Cantidad total de <i>issues</i>	3815.54	3877.00	2525.51	50.00	9072.00
G5: Cantidad total de <i>merge requests</i>	2593.45	2704.00	2003.50	1.00	8453.00
G6: Cantidad de <i>issues</i> abiertos	572.74	527.00	495.50	1.00	2602.00
G7: Cantidad de <i>forks</i>	189.00	38.00	453.36	1.00	3294.00
G8: Velocidad	3.72	2.84	3.37	1.08	12.82
G9: Puntos de historia	3.71	2.00	7.00	0.00	300.00
G10: Duración de las historias de usuario	107.11	26.28	175.39	0.00	899.99
GEM					
CF1: Complejidad del tamaño del equipo	0.06	0.05	0.05	0.01	0.18
CF2: Complejidad del lenguaje	0.02	0.01	0.02	0.01	0.08
CF3: Complejidad de las herramientas automatizadas	0.06	0.03	0.06	0.01	0.32
CF4: Complejidad de las técnicas de gestión	0.09	0.11	0.05	0.02	0.29
CF5: Complejidad de la integración	0.13	0.13	0.06	0.01	0.25
CF6: Complejidad de la seguridad	0.11	0.11	0.06	0.01	0.27
CF7: Complejidad de la reusabilidad	0.08	0.06	0.08	0.01	0.29
Objetivo					
ASP: Puntos de historia ajustados	5.83	4.55	7.21	1.52	305.16

Fuente: Elaboración propia.

5.2. Modelos para predecir los puntos de historia ajustados

En el aprendizaje de los modelos predictivos del primer experimento, modelos GEM, se utilizaron los puntos de historia (G9), la duración real (G10) y los datos asociados a las variables de complejidad del dominio y factores vitales ($CD, CF_1, CF_2, CF_3, CF_4, CF_5, CF_6, CF_7$). Los modelos de predicción del segundo experimento, modelos *GitLab*, fueron construidos a partir de los datos asociados a las diez variables *GitLab* ($G1, G2, G3, G4, G5, G6, \dots, G10$).

En las Tablas 5 y 6 se presenta el desempeño obtenido por los Modelos GEM y Modelos GitLab, respectivamente. En la Tabla **¡Error! No se encuentra el origen de la referencia.** se puede observar un sobreajuste en todos los modelos GEM, lo cual se explica por la naturaleza de los datos. En estos modelos se utiliza la complejidad de los factores vitales, que fue calculada por *GitLab-EffortMiner* usando parámetros establecidos de acuerdo a la literatura (ver Sección 4.2). Por esta razón, los modelos tienen un buen desempeño con los datos de entrenamiento y este desempeño decae drásticamente con los datos de prueba. Por otra parte, los modelos GitLab muestran un buen desempeño tanto en entrenamiento como en pruebas. El rendimiento en R^2 indica que la mayoría de los modelos logran explicar casi la totalidad de los datos. El rendimiento de las métricas restantes muestra que las estimaciones de los puntos de historia ajustados presentan un margen de error mínimo. Adicionalmente, el desempeño superior de los modelos basados en árboles al obtenido por el modelo de Regresión lineal, sugiere que no es lineal la relación entre las variables de GitLab y los puntos de historia ajustados.

En resumen, el desempeño de los modelos GEM y de los modelos GitLab preliminarmente sugiere que la información obtenida de los repositorios, sin transformar a factores GEM, es útil y más adecuada para la estimación de puntos de historia ajustados. Para confirmar estos hallazgos es necesario realizar más experimentos, integrando información de proyectos de otros dominios y validando con pruebas estadísticas.

Tabla 5. Desempeño obtenido por los modelos GEM en la estimación de puntos de historia ajustados.

Modelo	Entrenamiento			Prueba		
	MSE	MAE	R2	MSE	MAE	R2
Gradient Boosting	0.02	0.06	0.98	12.41	0.55	0.06
Bosque aleatorio	0.01	0.02	0.99	12.56	0.53	0.05
Regresión lineal	0.21	0.27	0.79	12.54	0.71	0.06
Árbol de decisión	0.01	0.02	0.99	12.56	0.53	0.05

Fuente: Elaboración propia.

Tabla 6. Desempeño obtenido por los modelos GitLab en la estimación de puntos de historia ajustados.

Modelo	Entrenamiento			Prueba		
	MSE	MAE	R2	MSE	MAE	R2
<i>Gradient Boosting</i>	0.00014	0.0059	0.99	0.28	0.10	0.78
Bosque aleatorio	0.000002	0.00024	0.99	0.28	0.09	0.79
Regresión lineal	0.15	0.22	0.84	0.30	0.27	0.76
Árbol de decisión	0.000002	0.0002	0.99	0.28	0.09	0.78

Fuente: Elaboración propia.

6. Discusión

En relación con la PI1, el *framework* propuesto (*GitLab-EffortMiner*) muestra viabilidad para la extracción automatizada de indicadores útiles para la estimación de esfuerzo. Específicamente, la herramienta recopila métricas operacionales clasificadas en actividad, participación, tecnología, popularidad e interacción (tales como colaboradores, lenguajes, *pipelines*, *issues*, *merge requests*, *commits*, versiones, estrellas y *forks*). Estos atributos permiten caracterizar la dinámica del equipo de desarrollo, la base tecnológica, la gestión de artefactos y el potencial de reutilización del software. La extracción exitosa de estos metadatos indica que los repositorios de *GitLab* albergan información histórica suficiente y accesible para respaldar los procesos de estimación automatizada en entornos ágiles. En lo concerniente a la PI2, los resultados experimentales indican que la

información extraída y transformada por *GitLab-EffortMiner* permite estructurar modelos con una capacidad predictiva aceptable para estimar los puntos de historia ajustados (ASP). No obstante, la utilidad de estos modelos está limitada a los tipos de proyectos considerados en la experimentación. Además, los indicadores recolectados se asocian de manera indirecta con la complejidad del dominio y con los factores vitales GEM. Por ejemplo, el volumen de *issues* sin resolver no captura de forma directa la seguridad del sistema, así como el número de colaboradores no refleja fielmente la experiencia acumulada ni la madurez del equipo.

Por otra parte, el diseño experimental se ve afectado por la falta de una separación o estratificación explícita por proyecto. Asimismo, se identificaron indicios de sobreajuste en los modelos GEM, lo que sugiere que los atributos obtenidos por el *framework* no permiten generalizar la tendencia subyacente. Adicionalmente, ante la falta de pruebas estadísticas, las diferencias observadas en el rendimiento predictivo de los modelos deben interpretarse como tendencias indicativas y no concluyentes. La validez externa se limita a proyectos de código abierto de GitLab incluidos en *User Story NeoDataset* [13]; por lo que no se puede asegurar que el rendimiento obtenido por los modelos predictivos pueda generalizarse a la estimación de esfuerzo de proyectos industriales, alojados en repositorios privados u otras plataformas, o que sigan otras prácticas de documentación. Por el alcance preliminar de la investigación, no se incorporaron en la experimentación repositorios adicionales ni industriales. Como trabajo a futuro, se propone incluir proyectos independientes, industriales y de otras plataformas. También como trabajo a futuro, se considera integrar la agrupación por proyecto en la evaluación de los modelos predictivos, justificando los hallazgos con las pruebas estadísticas apropiadas.

7. Conclusiones

En este trabajo se presenta un *framework* para la extracción de información de proyectos GitLab que apoyen en la estimación de esfuerzo en proyectos ágiles de software. Un componente fundamental de esta propuesta radica en su proceso de transformación de datos en factores cuantitativos, que representan las características del proyecto con mayor impacto en la estimación del esfuerzo.

Los resultados sobre el desempeño del *framework*, evaluado con la información de 30 proyectos de código abierto, muestran que la información extraída es útil para la estimación de esfuerzo, especialmente cuando se integra en modelos basados en árboles. Como trabajo a futuro, se contempla que el *framework* obtenga métricas que ayuden en la estimación del tiempo nominal en días requerido para la conclusión de los proyectos.

8. Agradecimientos

Los autores agradecen a la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) por su apoyo a esta investigación a través de la beca de maestría otorgada.

9. Referencias

- [1] Rashid, C. H., Shafi, I., Ahmad, J., Thompson, E. B., Vergara, M. M., de la Torre Diez, I., Ashraf, I. (2023). Software cost and effort estimation: Current approaches and future trends. *IEEE Access*, 11, 99268–99288. <https://doi.org/10.1109/ACCESS.2023.3312716>
- [2] Terlapu, P. V., Raju, K. K., Kiran Kumar, G., Jagadeeswara Rao, G., Kavitha, K., Samreen, S. (2024). Improved software effort estimation through machine learning: Challenges, applications, and feature importance analysis. *IEEE Access*, 12, 138663–138701. <https://doi.org/10.1109/ACCESS.2024.3457771>
- [3] Carbonera, C. E., Farias, K., Bischoff, V. (2020). Software development effort estimation: A systematic mapping study. *IET Software*, 14 (4), 328–344. <https://doi.org/10.1049/iet-sen.2018.5334>
- [4] Coelho, E., Basu, A. (2012). Effort estimation in agile software development using story points. *International Journal of Applied Information Systems*, 3 (7), 7–10. <https://doi.org/10.5120/ijais12-450574>
- [5] Alsaadi, B., Saeedi, K. (2022). Data-driven effort estimation techniques of agile user stories: A systematic literature review. *Artificial Intelligence Review*, 55, 5485–5516. <https://doi.org/10.1007/s10462-021-10132-x>
- [6] Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., Menzies, T. (2019). A deep learning model for estimating story points. *IEEE Transactions on Software Engineering*, 45 (7), 637–656. <https://doi.org/10.1109/TSE.2018.2792473>

- [7] Rodríguez Sánchez, E., Vázquez Santacruz, E. F., Cervantes Maceda, H. (2023). Effort and cost estimation using decision tree techniques and story points in agile software development. *Mathematics*, 11 (6), 1-31. <https://doi.org/10.3390/math11061477>
- [8] Zakaria, N. A., Ismail, A. R., Ali, A. Y., Khalid, N. H. M., Abidin, N. Z. (2021). Software project estimation with machine learning. *International Journal of Advanced Computer Science and Applications*, 12 (6), 726–734. <https://doi.org/10.14569/IJACSA.2021.0120685>
- [9] Rahman, M., Sarwar, H., Kader, M. A., Gonçalves, T., Tin, T. T. (2024). Review and empirical analysis of machine learning-based software effort estimation. *IEEE Access*, 12, 85661–85680. <https://doi.org/10.1109/ACCESS.2024.3404879>
- [10] Cunningham, P., Cord, M., Delany, S. J. (2008). Supervised learning. En M. Cord, P. Cunningham (Eds.), *Machine learning techniques for multimedia: Case studies on organization and retrieval* (pp. 21–49). Springer. https://doi.org/10.1007/978-3-540-75171-7_2
- [11] Huang, J., Li, Y.-F., Xie, M. (2015). An empirical analysis of data preprocessing for machine learning-based software cost estimation. *Information and Software Technology*, 67, 108–127. <https://doi.org/10.1016/j.infsof.2015.07.004>
- [12] Truong, L. (2018). *Agile user story point estimation* [Conjunto de datos y código]. GitHub. <https://github.com/mrthlinh/Agile-User-Story-Point-Estimation>
- [13] Neo, G. (2024). *User Story NeoDataset* (Versión 1) [Conjunto de datos]. Mendeley Data. <https://doi.org/10.17632/skk2wn9j86.1>
- [14] Tawosi, V., Al-Subaihin, A., Moussa, R., Sarro, F. (2022). *The TAWOS dataset* [Conjunto de datos]. University College London. <https://solar.cs.ucl.ac.uk/os/tawos.html>
- [15] Vidoni, M. (2022). A systematic process for mining software repositories: Results from a systematic literature review. *Information and Software Technology*, 144. <https://doi.org/10.1016/j.infsof.2021.106791>
- [16] Schreiber, A., de Boer, C., von Kurnatowski, L. (2021). *GitLab2PROV—Provenance of software projects hosted on GitLab*. 13th International Workshop on Theory and Practice of Provenance (TaPP 2021). USENIX Association. <https://www.usenix.org/conference/tapp2021/presentation/schreiber>
- [17] Bahlara, S., Ingle, M. (2011). Generalized agile estimation method. *International Journal on Advanced Science, Engineering and Information Technology*, 1 (3), 262–267. <https://ijaseit.insightsociety.org/index.php/ijaseit/article/view/55>
- [18] Usman, M., Mendes, E., Weidt, F., Britto, R. (2014). *Effort estimation in agile software development: A systematic literature review*. Proceedings of the 10th International Conference on Predictive Models in Software Engineering, Turin, Italy. <https://doi.org/10.1145/2639490.2639503>
- [19] Conoscenti, M., Besner, V., Vetrò, A., Méndez Fernández, D. (2019). Combining data analytics and developers feedback for identifying reasons of inaccurate estimations in agile software development. *Journal of Systems and Software*, 156, 126–135. <https://doi.org/10.1016/j.jss.2019.06.075>
- [20] Pasuksmit, J., Thongtanunam, P., Karunasekera, S. (2024). A systematic literature review on reasons and approaches for accurate effort estimations in agile. *ACM Computing Surveys*, 56 (11), 1-37. <https://doi.org/10.1145/3663365>
- [21] Jørgensen, M., Escott, E. (2022). Relative estimates of software development effort: Are they more accurate or less time-consuming to produce than absolute estimates, and to what extent are they person-independent? *Information and Software Technology*, 143. <https://doi.org/10.1016/j.infsof.2021.106782>
- [22] Fu, M., Tantithamthavorn, C. (2023). GPT2SP: A transformer-based agile story point estimation approach. *IEEE Transactions on Software Engineering*, 49 (2), 611–625. <https://doi.org/10.1109/TSE.2022.3158252>
- [23] Alaswad, F., Poovammal, E., Aljaddouh, B. (2026). From COCOMO to GPT: A comprehensive evaluation of LLM-based software effort estimation. *IEEE Access*, 14, 36051–36075. <https://doi.org/10.1109/ACCESS.2026.3671204>
- [24] Fumin, Q., Xiao-Yuan, J., Xiaoke, Z., Xiaoyuan, X., Baowen, X., Shi, Y. (2017). Software effort estimation based on open source projects: Case study of GitHub. *Information and Software Technology*, 92, 145–157. <https://doi.org/10.1016/j.infsof.2017.07.015>
- [25] Gote, C., Scholtes, I., Schweitzer, F. (2021). Analysing time-stamped co-editing networks in software development teams using git2net. *Empirical Software Engineering*, 26, 1-41. <https://doi.org/10.1007/s10664-020-09928-2>

- [26]Robles, G., Capiluppi, A., González-Barahona, J. M., Lundell, B., Gamalielsson, J. (2022). Development effort estimation in free/open source software from activity in version control systems. *Empirical Software Engineering*, 27, 1-37. <https://doi.org/10.1007/s10664-022-10166-x>
- [27]Fairbanks, J., Tharigonda, A., Eisty, N. U. (2023). *Analyzing the effects of CI/CD on open source repositories in GitHub and GitLab*. IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA), Orlando, FL, USA. <https://doi.org/10.1109/SERA57763.2023.10197778>
- [28]Kansab, S., Sayagh, M., Bordeleau, F., Tizghadam, A. (2025). An empirical study on the amount of changes required for merge request acceptance. *arXiv*. <https://doi.org/10.48550/arXiv.2507.23640>
- [29]Lucassen, G., Dalpiaz, F., van der Werf, J. M. E. M., Brinkkemper, S. (2016). The use and effectiveness of user stories in practice. En M. Daneva, O. Pastor (Eds.), *Requirements engineering: Foundation for software quality* (pp. 205–222). Springer. https://doi.org/10.1007/978-3-319-30282-9_14
- [30]Hany Fawzy, A., Wassif, K., Moussa, H. (2023). Framework for automatic detection of anomalies in DevOps. *Journal of King Saud University - Computer and Information Sciences*, 35 (3), 8–19. <https://doi.org/10.1016/j.jksuci.2023.02.010>
- [31]A G, P. V., K, A. K., S, R. (2025). Enhancing software effort estimation with random forest tuning and adaptive decision strategies. *Scientific Reports*, 15, 1-17. <https://doi.org/10.1038/s41598-025-14372-7>
- [32]Al-Rousan, T. (2024). Success factors for conducting software-process improvement in web-based software projects. *Cybernetics and Information Technologies*, 24 (1), 64–81. <https://doi.org/10.2478/cait-2024-0004>
- [33]Rincy N, T., Gupta, R. (2020). *A survey on machine learning approaches and its techniques*. IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), Bhopal, India. <https://doi.org/10.1109/SCEECS48394.2020.190>
- [34]Li, Z., Qi, X., Yu, Q., Liang, P., Mo, R., Yang, C. (2021). *Multi-programming-language commits in OSS: An empirical study on Apache projects*. IEEE/ACM 29th International Conference on Program Comprehension (ICPC), Madrid, Spain. <https://doi.org/10.1109/ICPC52881.2021.00029>