Hacia la Predicción Efectiva de Fallos en Software: Identificación de Factores Críticos

Andrés Ortiz[0009−0004−2139−9484] , Gustavo Sosa-Cabrera[0000−0002−9637−4319]

Facultad Politécnica - Universidad Nacional de Asunción
Asunción, Paraguay
ortizvillalba@fpuna.edu.py, gdsosa@pol.una.py
http://www.pol.una.py

Resumen A día de hoy, la predicción de fallos en aplicaciones de software es fundamental dado que estas aplicaciones, presentes en casi todos los aspectos de la vida moderna, requieren un funcionamiento continuo y confiable para soportar actividades críticas en diversos sectores. Este trabajo explora las características esenciales para la predicción de fallos en el lanzamiento de aplicaciones. Se analizan varias dimensiones críticas como la complejidad, el tiempo, el código, la difusión, los commits y el texto. Además, se presentan modelos de predicción y resultados basados en estas dimensiones, destacando su efectividad y las líneas de acción futuras.

Palabras Claves: Predicción de fallos · Ingeniería de características · Desarrollo de software · Lanzamiento de aplicaciones.

1. Introducción

Hoy en día, en el desarrollo de software, predecir la aparición de fallos en aplicaciones es crucial para mejorar la calidad y reducir el tiempo de comercialización. La ingeniería de características es esencial en la creación de modelos predictivos [4], involucrando la selección y transformación de variables relevantes a partir de datos brutos. En la predicción de fallos en software, se utiliza para identificar factores críticos que influyen en los fallos. Con el objetivo de permitir intervenciones proactivas y mejorar la fiabilidad del sistema, el presente trabajo en desarrollo se enfoca en identificar y analizar las características clave que pueden ser utilizadas para predecir fallos en el lanzamiento de aplicaciones.

2. Características para la predicción de fallos

Predecir fallos en software implica analizar diversas características que pueden influir en la probabilidad de fallos.

2.1. Dimensión de Complejidad

La complejidad del código es una medida fundamental en la predicción de fallos. La longitud del código, el número de ramas, y la complejidad ciclomática son indicadores importantes. Según Li y Henry [1], la complejidad del software está fuertemente correlacionada con la probabilidad de defectos.

- **Líneas de código (LoC).** Mide el número total de líneas de código en el software. Una mayor cantidad de LoC puede indicar un código más complejo y propenso a errores.
- **Complejidad ciclomatica (CC).** Cuantifica el número de caminos linealmente independientes a través del código fuente del programa, lo cual puede indicar áreas potenciales de complejidad y riesgo.

2.2. Dimensión de Tiempo

El tiempo en que se realizan cambios en el código puede influir en la aparición de fallos. El análisis de la frecuencia de commits y las horas de trabajo pico puede proporcionar indicadores sobre posibles áreas problemáticas.

- **Tiempo entre releases (TBR).** Es la duración entre lanzamientos de software. Intervalos más cortos podrían no permitir tiempo suficiente para pruebas exhaustivas, aumentando el riesgo de fallos.
- **Duración del ciclo de desarrollo (DCD).** Es el tiempo total dedicado al ciclo de desarrollo, incluyendo fases de planificación, codificación, pruebas y despliegue.

2.3. Dimensión de Código

La calidad del código es otro factor crucial. Métricas como la densidad de comentarios, la adherencia a los estándares de codificación y la reutilización de código son importantes. Kim et al. [2] destacan la importancia de la calidad del código en la predicción de fallos.

- **Tamaño del archivo (TA).** Es el tamaño de archivos individuales en la base de código. Archivos más grandes podrían ser más propensos a errores debido a su complejidad.
- **Niveles de anidación (NA).** Es la profundidad de las estructuras anidadas dentro del código. Altos niveles de anidación pueden llevar a un código más complicado y propenso a errores.

2.4. Dimensión de Difusión

La difusión de cambios a través de diferentes módulos del software puede indicar áreas de riesgo. Cambios extensos que afectan múltiples módulos son más propensos a introducir fallos.

- **Número de módulos afectados (NMA).** Es el número de módulos diferentes que un cambio impacta. Más módulos afectados pueden aumentar las posibilidades de introducir errores.
- **Difusión de cambios (DC).** Cuán ampliamente se distribuyen los cambios en la base de código. Mayor difusión puede indicar cambios más significativos y potencial inestabilidad.

2.5. Dimensión de Commits

El análisis de los commits individuales puede proporcionar información detallada sobre la introducción de fallos.

- **Frecuencia de commits (FC).** Es la frecuencia con la que se realizan cambios en el código en el repositorio. Commits frecuentes pueden indicar cambios rápidos y continuos que podrían llevar a inestabilidad.
- **Tamaño de commits (TC).** Es el tamaño de los commits individuales. Commits más grandes podrían introducir más errores debido al volumen de cambios.

2.6. Dimensión de Texto

El texto de los mensajes de commit y los comentarios en el código también pueden ofrecer pistas sobre la introducción de fallos. Herramientas de análisis de texto pueden ser utilizadas para identificar patrones de lenguaje asociados con defectos.

- **Descripciones de commits (DC).** La calidad y claridad de los mensajes de commits. Commits bien documentados pueden ayudar a entender los cambios e identificar posibles problemas.
- **Comentarios en el código (CC).** La presencia y calidad de los comentarios en el código. Buenos comentarios pueden ayudar a entender y mantener el código, reduciendo la probabilidad de errores.

3. Materiales y Métodos

Para predecir los fallos en los lanzamientos de aplicaciones, utilizaremos un modelo de regresión logística. La fórmula general del modelo es la siguiente:

$$\text{Probabilidad de Fallo} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n)}}$$

Donde: - β_0 es el término de intercepto. - $\beta_1, \beta_2, ..., \beta_n$ son los coeficientes de las características. - $X_1, X_2, ..., X_n$ son las características de las dimensiones de complejidad, tiempo, código, difusión, commits y texto.

Para validar los modelos, se realizaron pruebas utilizando datos históricos de proyectos de software reales y datos sintéticos.

En el diseño del Experimento se generaron datos sintéticos basados en los patrones observados en proyectos reales. Estos datos incluyen diversas métricas de las dimensiones de complejidad, tiempo, código, difusión, commits y texto (Ver Cuadro 1).

Cuadro 1. Características utilizadas para identificar lanzamientos con fallos

Dimensión	Características
Complejidad	Complejidad ciclomática, número de ramas, longitud del código, acoplamiento entre módulos.
Tiempo	Frecuencia de commits, tiempo entre commits, horas de trabajo pico.
Código	Densidad de comentarios, adherencia a estándares de codificación, reutilización de código, métricas de calidad del código.
Difusión	Número de módulos afectados por cambios, alcance de los cambios en el software.
Commits	Tamaño del commit, tipo de cambios (adición, eliminación, modificación).
Texto	Análisis de mensajes de commit, análisis de comentarios en el código, identificación de patrones de lenguaje asociados con defectos.

Para el experimento se consideraron tres tamaños de proyectos: pequeños (100-500 líneas de código), medianos (501-2000 líneas de código) y grandes (2001-5000 líneas de código). Cada categoría contiene datos de 100 lanzamientos, con una proporción conocida de lanzamientos con fallos.

4. Resultados Preliminares

En el Cuadro 2 y en la Figura 1, se presentan los resultados preliminares obtenidos mediante la aplicación de un modelo de regresión logística sobre los datos sintéticos, donde se puede apreciar la contribución efectiva de los factores identificados en la precisión del modelo predictivo.

Cuadro 2. Evaluación del modelo predictivo para fallos en aplicaciones.

Tamaño del Proyecto	Precisión (%)	Recall (%)	F1-Score (%)
Pequeño	85.0	80.0	82.4
Mediano	88.0	84.0	85.9
Grande	90.0	87.0	88.5

5. Conclusiones

La predicción de fallos en el lanzamiento de aplicaciones es un área crítica en el desarrollo de software. Los resultados preliminares de este trabajo en de-

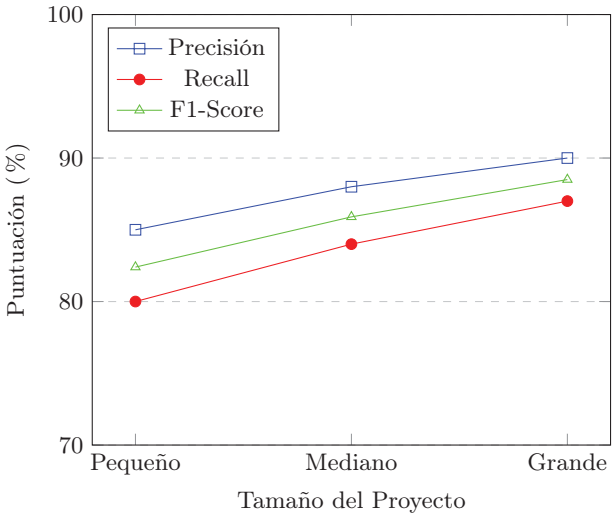

Figura 1. Precisión en la predicción de fallos según el tamaño del proyecto.

sarrollo, demuestra que las características seleccionadas de varias dimensiones pueden ser efectivamente utilizadas para predecir fallos. Así, se identifican los principales factores que contribuyen a los fallos del software, permitiendo intervenciones proactivas y mejorando la fiabilidad del sistema. Las líneas de acciones futuras se enfocarán en refinar estos modelos y explorar nuevas dimensiones para la construcción de modelos predictivos con precisión significativa.

Referencias

1. Li, W., Henry, S.: Maintenance Metrics for the Object Oriented Paradigm. Proceedings of the First International Software Metrics Symposium, pp. 52–60 (1993)
2. Kim, S., Whitehead Jr., E.J., Zhang, Y.: Classifying Software Changes: Clean or Buggy? IEEE Transactions on Software Engineering, vol. 34, no. 2, pp. 181–196 (2007)
3. Lorenz, K., Biere, S.: Feature Engineering for Machine Learning and Data Analytics. Springer (2019).
4. Sosa-Cabrera, G., Gómez-Guerrero, S., García-Torres, M., Schaerer, C.E.: Feature selection: A perspective on inter-attribute cooperation. International Journal of Data Science and Analytics 17(2), 139–151 (2024)
5. Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In AAAI-98 workshop.