Análisis de Impacto de la Ingeniería de Software Aumentada por IA y Sus Usos

ANDRES GARCIA SOLORZANO [1], (Student Master in Informatics Engineering, UPM), ANIBAL ANTONIO RIVERO RÍOS [1], (Student Master in Informatics Engineering, UPM), ESTEBAN MATTIACCI [2], (Student Master in Electronics and ICT Engineering Technology, KU Leuven University)

[1] Facultad de Informática,Technical University of Madrid, Madrid

[2] KU Leuven University, Leuven, Belgium

ABSTRACT <Insertar Abstracto>

INDEX TERMS Index 1,Index 2, Index 3

I. INTRODUCCIÓN

La Ingeniería de Software Aumentada por Inteligencia Artificial representa uno de los avances más notables y disruptivos en el ámbito de la ingeniería de software en la era moderna. Con la incorporación de la Inteligencia Artificial, el proceso de desarrollo de software ha experimentado una aceleración sin precedentes, abriendo las puertas a la automatización de tareas rutinarias, detección temprana de errores, y asistencia en la toma de decisiones a lo largo del ciclo de desarrollo.

Sin embargo, con la llegada de herramientas, como Copilot, Chat GPT y otros servicios que permiten generar código con inteligencia artificial, surgen tanto oportunidades como desafíos. Históricamente, la evolución de la tecnología ha supuesto para los profesionales de la ingeniería de software la necesidad de formarse constantemente para estar preparados para los cambios del sector. Algunos cambios como la adopción de lenguajes de alto nivel, herramientas de depuración, plataformas en línea en las que los desarrolladores de código se ayudan mutuamente y más recientemente la inteligencia artificial, han supuesto importantes cambios en el paradigma de la programación, haciéndola más accesible y simplificando procesos antes complicados o tediosos.

Podríamos caer en la trampa de pensar que el valor del conocimiento y la experiencia del ingeniero de software se está desvaneciendo, ya que la experiencia puede ser sustituida en cierta medida por colecciones de millones de preguntas de programación y sus respectivas respuestas, las herramientas de diagnóstico automático minimizan la necesidad de una depuración manual exhaustiva, o más recientemente las inteligencias artificiales son capaces de generar código, basándose en descripciones.

Sin embargo, esto está lejos de la realidad. A medida que las tareas básicas se simplifican o directamente se automatizan, se abre un nuevo horizonte de posibilidades que permiten a los ingenieros experimentados explorar desarrollos más complejos y ambiciosos. Si bien es cierto que ciertos conocimientos técnicos pueden ser parcialmente sustituidos por la IA, los desarrolladores necesitan un conocimiento sólido de desarrollo de software no solo para interactuar con estas herramientas, sino también para pensar críticamente y cuestionar las soluciones propuestas por estas.

Así, en lugar de ver a la IA como un reemplazo, debe considerarse como una herramienta que potencia y amplía las capacidades del ingeniero de software. Estamos ante un paradigma donde la simplificación de lo básico conduce a la posibilidad de alcanzar metas más altas, donde tanto novatos como expertos se benefician de la revolución impulsada por la Inteligencia Artificial. Esta transición, aunque beneficiosa, requiere un cambio en la capacitación y en el conjunto de habilidades demandadas. En el futuro es probable que veamos un aumento en la demanda de soft skills y de experiencia con herramientas de inteligencia artificial, llegando a existir el puesto de prompt engineer o ingeniero de instrucciones, ingenieros con formación específica para la interacción con herramientas de generación de código con inteligencia artificial.

Este documento tiene como objetivo explorar el estado actual de la ingeniería de software aumentada por inteligencia artificial, sus implicaciones, desafíos y el futuro. Mediante un análisis detallado, discutiremos cómo la IA está ampliando las posibilidades en el desarrollo de software, y cómo podría ser un camino hacia un desarrollo de software más robusto, eficiente y centrado en el usuario.

II. Funcionalidades

A. *Automatización de tareas básicas*

Al poder implementar las partes más básicas del código, facilita que los programadores se enfoquen en tareas de alta complejidad, particularmente en la elaboración de lógica avanzada, sin verse frenados por la necesidad de generar código y lógica de bajo nivel que, generalmente, no representa un desafío significativo para ellos. Esto implica que a largo plazo los desarrolladores ganarán experiencia en los campos más relevantes para ellos con mayor velocidad y a corto plazo llevarán a cabo los desarrollos más rápidamente. Además, se ha evidenciado que esta metodología eleva el nivel de satisfacción en el trabajo de los programadores, tal como se demuestra en el estudio X.

B. *Autocompletado del código*

La integración de la inteligencia artificial dentro de los IDEs potencia significativamente su capacidad para autocompletar código, en especial cuando la IA tiene acceso completo al código del proyecto y puede analizar patrones recurrentes en el mismo.

Las herramientas de autocompletado de código convencionales se basan en las reglas sintácticas y semánticas del lenguaje de programación en uso y sugieren al desarrollador todas las opciones que podrían escribirse inmediatamente a continuación sin causar errores sintácticos o semánticos. Sin embargo, con la incorporación de la IA, dependiendo de la escala y de cuán repetitivo sea el código, es posible autocompletar líneas de código e incluso funciones completas basándose en el código ya escrito.

Dado que la IA es contextualmente consciente, puede adaptar las sugerencias de autocompletado a la estructura específica del código en desarrollo. Por ejemplo en una clase que contenga decenas de manejadores de eventos con estructuras muy similares llega un punto en el que el desarrollador escribiendo solo el nombre del nuevo evento ya obtendría como sugerencia el código del manejador entero.

C. *Generación de Comentarios y Documentación*

La tarea de documentar el código puede ser percibida como tediosa por muchos programadores, lo que a menudo resulta en documentación insuficiente. La incorporación de la Inteligencia Artificial en el proceso de documentación puede aliviar significativamente esta carga. Una IA, especialmente cuando está integrada en el IDE, puede generar documentación del código a medida que se desarrolla e incluso adaptar el estilo de documentación para mantener la consistencia con las convenciones adoptadas por el equipo de desarrollo, resultando en una documentación más homogénea y comprensible.

Adicionalmente, la IA puede añadir documentación a código de terceros, facilitando así su comprensión y mantenimiento. Las inteligencias artificiales que cuentan con el contexto del proyecto pueden seguir las referencias a otras clases dentro del mismo si es necesario para descifrar la funcionalidad del código, proporcionando comentarios y documentación útiles que clarifican la operación del código para los desarrolladores que lo aborden en el futuro.

Esta automatización no solo eleva la calidad de la documentación, sino que ahorra tiempo y esfuerzo, permitiendo que los desarrolladores se enfoquen en tareas más complejas y gratificantes.

D. *Corrección de Errores*

Los errores en el código son una fuente común de frustración y desmotivación para los programadores y pueden resultar en períodos de estancamiento donde los desarrolladores se encuentran atrapados en un problema particular sin obtener resultados. La IA puede analizar el código en busca de errores y ofrecer posibles soluciones para los mismos.

Además, la IA puede proporcionar explicaciones detalladas sobre la naturaleza de los errores, ayudando a los programadores a entender no sólo cómo corregir el error actual, sino también cómo evitar errores similares en el futuro, una ayuda especialmente útil para los programadores menos experimentados.

E. *Desarrollo de tests*

El desarrollo de pruebas es esencial en el desarrollo de software, aunque a menudo se percibe como una tarea tediosa. Esto puede resultar en pruebas insuficientes o superficiales que no cubren todos los casos críticos, poniendo en riesgo la calidad y la fiabilidad del software.

La IA, con su capacidad para analizar el código y entender las convenciones de codificación, puede generar automáticamente casos de prueba que aborden los escenarios críticos y potencialmente problemáticos. Además, puede seguir las mejores prácticas para evitar "test smells", contribuyendo así a la calidad y la eficacia de las pruebas.

Además de generar nuevos casos de prueba, la IA puede analizar y mejorar los casos de prueba existentes y sugerir nuevas pruebas para cubrir áreas del código que no estén cubiertas.

F. Traducción de Código entre Lenguajes

La migración de código entre diferentes lenguajes representa una tarea tediosa y repetitiva, pero es crucial, ya que una migración incorrecta puede comprometer toda la integridad del código o afectar a su rendimiento. La inteligencia artificial puede ahorrar una cantidad significativa de tiempo al convertir automáticamente el código existente al nuevo lenguaje deseado.

Adicionalmente, para los desarrolladores que buscan aprender un nuevo lenguaje, la posibilidad de usar como ejemplo el código que ellos mismos han desarrollado les permite comparar y entender las diferencias y similitudes entre ambos lenguajes.

En el contexto de proyectos que involucran varios lenguajes de programación un desarrollador podría necesitar entender una sección de código escrita en un lenguaje diferente, especialmente si esta sección interactúa o intercambia datos con el código que está desarrollando. En tales casos, poder traducir el código a un lenguaje con el que esté familiarizado puede facilitar su comprensión y mejorar la colaboración y la eficiencia en el desarrollo del proyecto.

G. Refactorización Asistida

La refactorización del código, que implica reorganizar y optimizar el código existente sin alterar su funcionalidad, juega un papel vital en la preservación de una base de código manejable, comprensible y actualizable.

Con acceso al contexto del proyecto, la IA puede analizar el código existente, identificar áreas de mejora y sugerir o implementar cambios en la legibilidad y la estructura del código evitando malas prácticas, redundancias y otros problemas comunes que puedan surgir en proyectos colaborativos de gran escala.

Además, al observar cómo la IA transforma su código original en una versión refactorizada, los desarrolladores pueden aprender de sus errores y mejorar para sus futuros desarrollos. Esto, a su vez, contribuye a la creación de una base de código más robusta y fácil de mantener, facilitando el trabajo del equipo y reduciendo los costos asociados con la modificación y extensión del código en el futuro.

H. Generación de Interfaces Gráficas a partir de Bocetos y Descripciones

Gracias a la inteligencia artificial es posible que un desarrollador realice un boceto básico en papel o en una herramienta de dibujo digital y usando modelo GPT entrenado, pueda transformar ese boceto en código para usarlo como la base sobre la que construir el sistema, permitiendo al desarrollador enfocarse en refinar y personalizar la interfaz en lugar de construirla desde cero.

Posteriormente, se pueden proporcionar descripciones textuales adicionales a la IA para ajustar la interfaz generada y acercarla más al objetivo final. Este proceso iterativo de refinamiento puede continuar hasta que se considere necesario y supone una alternativa intermedia entre la programación manual y las herramientas no code de diseño de interfaces.

Además, esta forma de trabajo supone una nueva oportunidad de aprendizaje para los programadores novatos en el campo del desarrollo de interfaces gráficas. Al interactuar con la IA y observar cómo interpreta y convierte los bocetos y descripciones en código, los programadores pueden aprender cómo se construyen las interfaces gráficas y cómo se traducen sus requisitos visuales y funcionales en código.

I. Integración con Sistemas de Control de Versiones

La integración de la Inteligencia Artificial en los Sistemas de Control de Versiones aborda una de las tareas más tediosas en el desarrollo de software, especialmente en proyectos de gran envergadura. En estos proyectos, los desarrolladores no solo necesitan navegar a través de miles de líneas de código distribuidas en múltiples clases, sino también rastrear las diferentes versiones del proyecto a lo largo del tiempo, lo que puede resultar en una tarea sumamente compleja.

La situación se alivia si los commits están bien documentados, indicando los cambios realizados y asociándose a tareas específicas que contengan las instrucciones seguidas por los desarrolladores. Sin embargo, esta práctica a menudo se descuida debido a su naturaleza monótona y al tiempo que consume.

La inteligencia artificial permite automatizar varias de estas tareas. Por ejemplo, la IA puede sugerir mensajes para los commits y descripciones para las pull requests, facilitando así que los desarrolladores solo necesiten revisar y, en caso necesario, hacer pequeñas correcciones a estas sugerencias. Esta automatización no

solo ahorra tiempo, sino que también promueve una documentación más consistente y completa.

Además, si la IA está bien integrada en el proyecto y tiene acceso al contexto del mismo, puede convertirse en una fuente valiosa de información. Los desarrolladores podrían consultar a la IA usando lenguaje natural para obtener detalles sobre cuándo se realizaron ciertos cambios o entender las razones detrás de ellos, eliminando la necesidad de buscar manualmente esta información en el historial del sistema de gestión de versiones.

III. Herramientas

A. Integradas en el IDE

Existen varias herramientas que se encuentran integradas dentro del IDE en el que se está programando por lo que cuentan con un mayor contexto del proyecto en el que se está trabajando y actúan de manera proactiva, anticipándose a las necesidades del programador y proporcionando sugerencias automáticas sin requerir una interacción explícita del usuario. Actualmente existen varias herramientas con este comportamiento.

En esta sección, abordaremos seis casos destacados. Los dos primeros casos se enfocarán en las primeras implementaciones de la IA asistiendo en la codificación, mostrando cómo estas iniciativas sirvieron como bases para desarrollos futuros. Los siguientes dos casos presentan algunas de las soluciones más populares y consolidadas en la actualidad, las cuales se han hecho muy populares debido a su aplicabilidad general y facilidad de integración en diversos entornos de desarrollo. Por último, los dos casos restantes exploran soluciones orientadas a escenarios más específicos, ya sea para proyectos con requisitos muy particulares o para proporcionar asistencia exhaustiva en todas las áreas del desarrollo del proyecto. Estas herramientas, aunque menos generalizadas, demuestran la versatilidad y el potencial de la IA en estos campos.

1) Kite

Kite emergió en el escenario de la asistencia de codificación con IA en 2014, marcando un período de innovación que se extendió hasta 2021 [Kite Source 1]. Como startup, Kite se dedicó a desarrollar herramientas que utilizaban la inteligencia artificial para asistir a los desarrolladores en la escritura de código. Su producto, un asistente de codificación impulsado por IA, fue lanzado mucho antes que soluciones como GitHub Copilot, y ofreció completado de código para Python y más tarde llegó a tener soporte para 16 lenguajes distintos [Kite Source 2].

El enfoque de Kite era proporcionar sugerencias inteligentes de código y autocompletado basado en IA, permitiendo a los desarrolladores escribir código de manera más eficiente. Sin embargo, a pesar de sus ambiciones, Kite enfrentó limitaciones significativas. Una de las principales limitaciones fue la incapacidad de sus modelos de aprendizaje automático para entender adecuadamente la estructura del código, como el contexto no local, lo que representó un desafío insuperable para el equipo[Kite Source 3].

A pesar de haber asegurado decenas de millones de dólares en financiamiento de capital de riesgo, Kite encontró dificultades para monetizar su producto, lo que eventualmente condujo a su descontinuación en 2021. Adam Smith, fundador de Kite, reconoció que la tecnología no estaba aún lista para el mercado, y que tomaron demasiado tiempo en darse cuenta de ello. Según Smith, estaban "10+ años adelantados al mercado" [Kite Source 4].
El cierre de Kite fue un evento significativo en la comunidad de desarrollo, y su experiencia sirvió como valiosas lecciones sobre los desafíos inherentes a la comercialización de tecnologías de IA para la codificación. Kite también anunció que estaría liberando su solución como código abierto, permitiendo así que la comunidad de desarrollo continuará explorando y expandiendo las ideas y tecnologías que Kite había introducido[Kite Source 5].

a) Desarrollo

Kite entrenó sus modelos de machine learning utilizando grandes conjuntos de datos de código fuente abierto. En particular, el modelo de Kite para Python fue entrenado utilizando 25 millones de archivos de código de fuente abierta, mientras que su modelo para JavaScript se entrenó con 30 millones de archivos[Kite Source 10]. Además, lanzaron un nuevo modelo de deep learning para JavaScript, entrenado en 22 millones de archivos de código fuente abierto, que permitía completar múltiples líneas de código a la vez [Kite Source 11]. Kite empleaba modelos de deep learning entrenados en código fuente abierto para ofrecer sugerencias de autocompletado con alta confianza, ayudando a automatizar partes repetitivas de la programación3.

b) Costes

Kite proporcionó una gama de opciones de precios para acomodar a diferentes usuarios y necesidades. Existía una versión gratuita conocida como “Kite Free”, que proporcionaba características

esenciales para asistir en la codificación. En particular, Kite Free se destacaba por ofrecer completado de código consciente del contexto y proporcionaba un 50% más de completados en comparación a sus competidores [Kite source 6]. Para aquellos que requerían más funcionalidades, Kite introdujo un plan Pro, que estaba disponible por $19.90 al mes. Este plan Pro incluyó una nueva motorización para Python y soporte para JavaScript, ampliando así las capacidades del asistente de codificación IA de Kite [Kite source 7].

Adicionalmente, Kite eventualmente lanzó para el mercado empresarial Kite Team Server, una versión empresarial de su herramienta de completado de código. Esta versión fue diseñada para ser alojada por la empresa y utilizaba una tecnología conocida como "four token autocomplete" [Kite source 8]. Está versión tenía soporte para 16 lenguajes en 16 IDEs [Kite source 9] y costaba alrededor de $40 al mes [Kite source 8].

2) IntelliSense y IntelliCode

En el contexto de la asistencia de codificación potenciada por inteligencia artificial (IA), Microsoft realizó un avance significativo con la introducción de IntelliCode en el 2018, evolucionando su anterior herramienta, IntelliSense. Aunque IntelliCode trae consigo un conjunto enriquecido de características impulsadas por IA, todavía se encuentra en una etapa menos avanzada en comparación con otras soluciones emergentes como GitHub Copilot.

IntelliSense ha sido la herramienta de autocompletado y asistencia de codificación estándar en Visual Studio y Visual Studio Code por muchos años. Sin embargo, su capacidad se limita a proporcionar sugerencias basadas en el código existente y las bibliotecas importadas. Con el avance de la IA, Microsoft introdujo IntelliCode para llevar la asistencia de codificación a un nivel superior.

Las características principales de IntelliCode son las siguientes [Source]:

- Autocompletado de Líneas Completas: IntelliCode puede completar una línea entera de código a la vez, ajustando sus sugerencias según el contexto del código, incluyendo nombres de variables y funciones.
- Recomendaciones Priorizadas: A diferencia de IntelliSense, IntelliCode prioriza las sugerencias basándose en el análisis de miles de contribuciones de código abierto en GitHub, colocando las opciones más relevantes en la parte superior de la lista de autocompletado.
- Refactorización con Facilidad: La herramienta puede detectar repeticiones en el código y sugerir refactorizaciones, facilitando la aplicación de ediciones consistentes a lo largo del código.

Pero, a pesar de las mejoras significativas que trae IntelliCode, aún se encuentra detrás en términos de capacidades avanzadas cuando se compara con herramientas de asistencia más actuales. Por ejemplo a la hora de generar funciones, herramientas como Tabnine o Copilot tienen la capacidad de generar funciones o tests completos, mientras que IntelliCode se limita a generar hasta una línea completa de código a la vez [Source]. La diferencia también se puede ver en la tecnología subyacente; Copilot por ejemplo utiliza un modelo de IA más avanzado (CODEX) en comparación con el modelo GPT-C de IntelliCode, lo que le permite ofrecer sugerencias de código más completas y contextuales.

Aun así el enfoque en el autocompletado inteligente y la refactorización eficaz ha establecido una sólida base conceptual y tecnológica. IntelliCode, en particular, con su incorporación de inteligencia artificial para analizar patrones en grandes conjuntos de datos de código, ha servido de base para un amplio número de posibilidades en cómo la IA puede ser utilizada para facilitar y optimizar la creación y mantenimiento de código.

3) Github Copilot

[1][2]
Desarrollado por GitHub en asociación con OpenAI, Copilot se presenta como una herramienta revolucionaria para asistir a los programadores en la escritura de código de manera ágil y eficiente.

El núcleo de esta innovación reside en una arquitectura respaldada por GPT-3 (Generative Pre-trained Transformer 3) de OpenAI. A pesar de ser la tercera versión del modelo Generative Pre-trained Transformer de OpenAI, desde su lanzamiento en junio de 2020, ha acaparado titulares por sus asombrosos 175 mil millones de parámetros, posicionándose como uno de los modelos de lenguaje más grandes y avanzados jamás creados. Su precisión al generar texto con calidad humana, responder preguntas, traducir idiomas e incluso escribir código, es producto de un entrenamiento con una extensa colección de datos textuales, permitiendo respuestas coherentes y contextualizadas según la información ingresada.

Posteriormente, GitHub, bajo la sombrilla de Microsoft, emprendió la tarea de perfeccionar la tecnología de Copilot. Mediante el ajuste de los motores Codex con millones de líneas de código público en GitHub, se logró que Copilot operará bajo esta vasta información. Al momento de programar, Copilot, como un experto asistente, sugiere automáticamente fragmentos de código con la mayor precisión, respaldado por todo el conocimiento acumulado durante su entrenamiento.

Al comienzo de la escritura de código por parte del desarrollador, Copilot examina el contexto y propone fragmentos de código pertinentes para la tarea en cuestión. El programador tiene la libertad de aceptar, modificar o descartar la sugerencia, siendo este proceso potenciado por la capacidad del modelo GPT-3 de generar textos coherentes y contextualizados, que en este caso se traducen en código.

a) Uso de la herramienta

[2]

GitHub Copilot se presenta como una extensión disponible en distintos entornos de trabajo. Una vez instalada, el proceso para empezar a utilizarla es directo: activación, inicio de sesión en GitHub y comienzo de la codificación. A través de la extensión, el usuario recibe propuestas de completado de código directamente en su entorno de trabajo. Una característica principal es que las distintas sugerencias se presentan junto al código en curso, permitiendo una visualización completa y formateada del mismo. Si la propuesta inicial no es adecuada, el usuario tiene la flexibilidad de explorar otras opciones mediante combinaciones de teclas. Para aquellos que deseen desactivar temporalmente la función, Copilot ha incorporado un icono fácilmente accesible.

Una limitación importante es que GitHub Copilot tiene una compatibilidad limitada con otros entornos de trabajo. Estos incluyen Azure Data Studio, JetBrains IDEs, Vim/Neovim y Visual Studio. Esta diversidad de entornos muestra un esfuerzo por parte de los desarrolladores de Copilot por adaptarse a diferentes plataformas, aunque aún con margen de expansión. Esto significa que aquellos desarrolladores que no utilicen estos entornos de desarrollo no pueden beneficiarse de la herramienta

b) Privacidad

[2]

En cuanto al tratamiento de datos de GitHub Copilot hay que distinguir entre los usuarios individuales y los empresariales:

Para los usuarios individuales GitHub Copilot recopila información de los archivos con los que trabaja el usuario, además de otros datos adicionales necesarios para proporcionar el servicio.Se recopila información sobre eventos generados durante la interacción con el IDE o editor, como las acciones de edición del usuario y datos de uso general para identificar métricas como latencia y engagement con las características1.Estos datos recopilados se utilizan para mejorar GitHub Copilot y servicios relacionados, y para llevar a cabo investigaciones de productos y académicas. Los datos también se utilizan para detectar abusos y violaciones de las políticas de uso aceptable asociadas con GitHub Copilot 1.Los usuarios tienen la opción de decidir si se retienen los datos de "Prompts" y "Suggestions" en GitHub, y estas preferencias se pueden ajustar en la configuración de GitHub Copilot [GC 3].

Similar a la versión para usuarios individuales, en la versión empresarial de Github Copilot se recogen datos de “engagement” del usuario, “Prompts” (colección de código y información contextual que se envía a GitHub para generar sugerencias) y “Suggestions” (líneas propuestas de código retornadas a la extensión de Copilot después de recibir y procesar un "Prompt")2. Los datos de “engagement” del usuario se retienen por 24 meses, mientras que los "Prompts" y "Suggestions" se utilizan sólo para proporcionar el servicio y no se retienen. Está es la diferencia principal frente a los usuarios individuales que solamente pueden escoger si los datos se mantienen o no; no tienen la habilidad de escoger una fecha de expiración. Los datos de engagement del usuario se utilizan para evaluar GitHub Copilot, ajustar algoritmos, detectar posibles abusos y conducir experimentos y investigaciones relacionadas con los desarrolladores y su uso de las herramientas y servicios de desarrollo2. Los usuarios empresariales tienen control sobre cómo se utiliza su data a través de la configuración de GitHub Copilot, similar a los usuarios individuales2.

c) Costes

En cuanto a costes Github Copilot cuesta $10 al mes mientras que la versión empresarial tiene un coste de $19 por cada usuario. También existe una oferta gratuita para estudiantes verificados y mantenedores de proyectos de código abierto populares. [Source]

d) Copilot X

GitHub Copilot X representa una evolución notable de GitHub Copilot, incorporando avances significativos que apuntan a optimizar la experiencia de desarrollo de software. Este avance intenta integrar la inteligencia artificial en el ciclo de vida del desarrollo de software, facilitando así la autoría, revisión y mantenimiento del código.

Copilot X incorpora interfaces de chat y terminal, soporte para pull requests y una adopción temprana de OpenAI GPT-4. Estas innovaciones se integran en cada parte del flujo de trabajo de desarrollo, desde la autoría hasta la revisión del código1. Además, se ha mejorado la responsividad mediante un modelo ligero en el lado del cliente para Visual Studio Code2. El soporte para pull requests incluye la generación automática de descripciones y la identificación y generación de pruebas unitarias faltantes1

La experiencia de usuario se mejora con una interfaz con un estilo similar a la de Chat GPT pero incorporado dentro de los editores de código, permitiendo una interacción más eficiente con la herramienta. La capacidad de responder preguntas directamente desde la documentación y proporcionar asistencia en la interfaz de línea de comandos resalta la intención de integrar la IA de manera más profunda en el ciclo de vida del desarrollo de software3 4 1.

4) Tabnine

1 https://www.tabnine.com/slides/Case_Study_29_5_23.pdf
2 Tabnine AI: The Ultimate Code Completion Tool for Developers (linkedin.com)
3 Get Tabnine
4 On which code repositories is Tabnine training its AI models? – Tabnine

Tabnine AI es una herramienta de autocompletado de código que funciona utilizando inteligencia artificial. Se integra con IDEs populares, incluyendo IntelliJ, Eclipse y Visual Studio Code. La base de Tabnine está basada en sus propios modelos de GPT que siguen la arquitectura de GPT 3.5 pero están organizados en varias colecciones **[4]**. Utilizando algoritmos de aprendizaje automático, Tabnine analiza el contexto de tu código y sugiere completados de código en tiempo real, ayudándote a escribir código más rápidamente y con menos errores **[2]**. Soporta en total 25 lenguajes de programación y se instala principalmente en IDEs de Jetbrains pero también tiene soporte para VS code, Eclipse, Sublime, Neovim, Android Studio y Visual Studio **[3]**.

En el proceso operativo de Tabnine, mientras el usuario está trabajando, el cliente (plugin) solicita asistencia al clúster de Tabnine. Este proceso se lleva a cabo de forma imperceptible en segundo plano, permitiendo así una experiencia de usuario fluida y sin interrupciones. Para las sugerencias de código, este proceso se ejecuta de manera transparente mientras el usuario está codificando. En el caso del chat, la solicitud de asistencia se activa una vez que el usuario formula una pregunta.

a) Privacidad

Privacy Tabnine
Una de las características principales de Tabnine es la importancia que le da a la privacidad en el ámbito de la codificación. Uno de los pilares fundamentales en la operatividad de Tabnine es que nunca almacena ni comparte el código del usuario con terceros. Esta característica es extremadamente importante, ya que en el mundo digital actual, la privacidad y la seguridad de los datos se han convertido en una de las mayores preocupaciones para individuos y organizaciones.

La interacción entre el usuario y Tabnine se realiza mediante lo que se denomina una "ventana de contexto", la cual incluye fragmentos del código del proyecto local para permitir que Tabnine proporcione respuestas más relevantes y precisas. Esta ventana de contexto puede incorporar varios elementos del entorno local del usuario, tales como el historial de chat en caso de una sesión de chat, líneas de código, declaraciones de tipo, funciones, objetos, importaciones relacionadas del archivo actual, archivos relacionados, y reportes de errores sintácticos y semánticos. Todo este contexto se elimina de manera inmediata después de que el servidor devuelve la respuesta al cliente, garantizando así la no retención de ningún código del usuario más allá del marco de tiempo inmediato necesario para la inferencia del modelo, un proceso que Tabnine denomina como procesamiento efímero.

Tabnine también se distingue por no entrenar sus modelos de IA en el código de los usuarios. Los modelos universales de IA de Tabnine se entrenan exclusivamente en código de fuente abierta con licencias permisivas. En el caso de los modelos privados, estos son pre entrenados en código privado por Tabnine y solo son accesibles por los miembros del equipo del usuario y se almacenan en la configuración privada del mismo.

b) Costes

Tabnine Enterprise vs. GitHub Copilot Business

Para aprovechar las capacidades de inteligencia artificial de Tabnine, se necesitaría contratar el plan Pro o el plan Enterprise. El plan Pro es más económico, con un costo de 12 dólares al mes, e incluye completado de código con AI y soporte estándar. Sin embargo, donde Tabnine realmente destaca es en su categoría Enterprise. Además de contar con las características del plan Pro, también ofrece restricciones de privacidad más estrictas, modelos de AI personalizados, chat integrado en el IDE y generación de AI privada. La versión Enterprise cuesta alrededor de 20 dólares al mes por cada usuario pero dependiendo del uso dado y del estado de la compañía el precio puede variar.

5) CodeWP

En el ámbito del desarrollo de software, la eficiencia y la precisión son cruciales para la entrega exitosa de proyectos. Herramientas como Copilot y Tabnine han ganado reconocimiento por su capacidad para asistir en el desarrollo de proyectos generales, proporcionando sugerencias de código y mejorando la eficiencia del proceso de codificación. Sin embargo, cuando se trata de proyectos muy específicos, especialmente en entornos como WordPress, puede ser beneficioso explorar herramientas diseñadas para abordar desafíos particulares asociados con esa plataforma.

CodeWP se presenta como una plataforma diseñada para simplificar el proceso de desarrollo en WordPress a través de su generador de código impulsado por inteligencia artificial. Fundado en noviembre de 2022 por James LePage de WPAI, Inc., CodeWP genera fragmentos de código basados en la entrada del usuario para diversas tareas, como la creación de WP Queries, funciones y filtros de WooCommerce, abordando así el desarrollo de WordPress que puede ser tedioso y propenso a errores[3].

Algunas de las características clave incluyen:

- Modos IA: CodeWP cuenta con más de 12 modos de IA, cada uno entrenado en un tipo específico de código de WordPress, lo que permite la generación de código de alta calidad para una amplia variedad de tareas.
- Multilingüe: La plataforma puede generar código en más de 11 idiomas, incluyendo inglés, francés, alemán, español, portugués, italiano, ruso, japonés, coreano, chino y árabe.
- Generación de Sub Fragmentos: Permite la generación de sub fragmentos, que son CSS y JS personalizados para fragmentos de PHP, permitiendo así crear código más complejo y dinámico[3].

a) Costes

CodeWP opera bajo un modelo de precios freemium. Ofrece un plan gratuito que permite a los usuarios generar hasta 100 fragmentos de código por mes pero tiene un acceso muy limitado a los distintos modelos IA. El plan Pro, con un costo de $18 por mes, proporciona más características como 10,000 acciones por mes, incluye 4 proyectos, y ofrece más de 28 modos específicos de IA para código y chat. Para una colaboración más avanzada, el plan Agency, con un costo de $48 por mes, proporciona acciones ilimitadas por mes, incluye 3 miembros del equipo, y proyectos ilimitados entre otras características[4]. En el caso de necesitar

6) Katalon

En el campo de desarrollo de software, las herramientas para mantener y probar el código son tan esenciales como las herramientas generales de codificación, como Copilot. Estas herramientas de mantenimiento y prueba permiten a los desarrolladores asegurarse de que el software funcione correctamente y cumpla con los estándares de calidad, lo que, a su vez, garantiza una entrega de software exitosa.

Katalon es una plataforma para la gestión de la calidad del software, que ayuda a los equipos a mantener proyectos digitales de alta calidad. Se compone de varias plataformas, incluyendo Katalon Studio, Katalon TestOps, Katalon Runtime Engine y Katalon TestCloud, cada una con funciones y capacidades únicas para abordar diferentes aspectos del ciclo de vida de las pruebas de software[1].

La IA en Katalon se manifiesta principalmente en la facilitación de la autoría de pruebas y la ejecución de pruebas visuales. Katalon Studio proporciona una experiencia de bajo código que permite a los usuarios generar pruebas automatizadas con facilidad. Además, el entorno incluye capacidades de testing visual impulsadas por IA, que ahorran hasta el 99% del esfuerzo para detectar regresiones visuales mediante métodos de comparación automatizados[2] [3]. Por otro lado, Katalon TestCloud ofrece un entorno de ejecución de pruebas en la nube, permitiendo la ejecución de pruebas en múltiples configuraciones y entornos sin la necesidad de gestionar infraestructuras complejas. Esto es especialmente beneficioso para asegurar la compatibilidad y la funcionalidad en diferentes navegadores y sistemas operativos[1].

Katalon emplea varias características impulsadas por IA para mejorar el proceso de prueba. Las características clave incluyen:

- Creación de Casos de Prueba: Las capacidades impulsadas por IA de Katalon facilitan la creación automática de casos de prueba, reduciendo significativamente el esfuerzo manual y mejorando la eficiencia2.

- Pruebas Visuales: Utilizando IA, las pruebas visuales en Katalon reducen el esfuerzo requerido para identificar regresiones visuales en un 99%, presentando una ventaja sustancial en ahorro de tiempo3.

- Studio Assist: Esta característica aprovecha la tecnología GPT para ayudar a escribir pruebas más rápido generando código y proporcionando explicaciones claras y precisas del código generado1.

- Auto-Reparación: El mecanismo de auto-reparación de Katalon detecta localizadores rotos y propone alternativas funcionales, asegurando que las pruebas continúen ejecutándose sin problemas1.

a) Costes

Katalon ofrece un modelo de precios escalonado que incluye un plan gratuito, para individuos o equipos pequeños que recién comienzan. El plan Premium, con precios desde $167, proporciona creación de pruebas de grado profesional, flujos de trabajo de pruebas dinámicas, ejecución avanzada de pruebas y acceso a las diversas plataformas de Katalon 4 5.

b) Privacidad

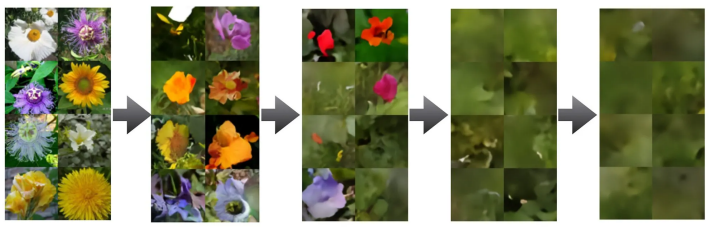
Katalon tiene mecanismos establecidos para proteger los datos del usuario. El Agente de IA de Katalon permite a los usuarios solicitar la eliminación completa de todos los datos rastreados y optar por no usar la función si tienen preocupaciones respecto al cumplimiento de la privacidad de datos. Además, el marco de seguridad de Katalon se alinea con la norma ISO/IEC 27001 para la gestión de la seguridad de la información, asegurando que se implementen robustas medidas de protección de datos6 7.
Katalon Studio continúa evolucionando, con versiones como 7.x, 8.x y la reciente 9.0.0.beta, cada una aportando mejoras como versiones actualizadas de Eclipse IDE, Java y el compilador Groovy, entre otros8 9 10 11 12. Esta evolución refleja el compromiso de Katalon de mantenerse actualizado con los últimos avances tecnológicos para proporcionar una plataforma de pruebas robusta e impulsada por IA.

B. Externas al IDE

IV. Futuro

Union con otros
Microsoft:
Microsoft es actualmente propietaria de diversos servicios, entre ellos, Microsoft Office y GitHub. Recientemente, Microsoft lanzó el servicio "365 Copilot", que consiste en un asistente dotado de inteligencia artificial que se integra en las distintas herramientas de Office, como Excel, Word o Teams. Este software está diseñado con el objetivo de ser utilizado en el mundo laboral. Algunas de las funciones que incluye son …… Otra función muy interesante es la capacidad del Copilot para, tras una reunión de Teams, generar un resumen con bullet points. Esto permite a los asistentes hacer preguntas sobre la reunión si se han perdido algún punto. Además, Microsoft es propietaria de GitHub, y por tanto, del GitHub Copilot previamente mencionado. No es difícil imaginarse cómo podría beneficiar que estas herramientas trabajasen juntas. La primera obtendría de la reunión la información necesaria para las nuevas tareas a realizar por el empleado. A continuación, cuando el empleado comience el desarrollo, el GitHub Copilot sabría de antemano las tareas a realizar, permitiendo un autocompletado más eficiente e incluso sugerencias sobre el código ya realizado, sin que se le consulte, razonando por qué podría dar problemas según la información de la reunión. Estos sistemas aún están en desarrollo y no son más que ideas, pero asumiendo un ritmo de progresión relativamente constante, es solo cuestión de tiempo que se conviertan en realidad.

Problemas: retrofeed



V. Problemas en el futuro

https://www.theregister.com/2023/06/16/crowd_workers_bots_ai_training/

Error acumulado:
Las inteligencias artificiales de todos los tipos tienen un cierto grado de errores o alucinaciones en sus outputs. Herramientas como chat gpt no han tardado en expandirse por todo el mundo. La cantidad de contenido en la red generado por inteligencias artificiales es cada vez mayor, y no siempre tan curada como cabría esperar. Por ejemplo este análisis bla bla bla [primer link de arriba]. Si sumamos a esto los desconocidos orígenes de los datos usados para entrenar varios modelos de inteligencia artificial que nos lleva a suponer que obtienen los datos de internet sin curar, es fácil entender cómo las inteligencias artificiales no tardaran en entrenarse a base de los outputs de otras ia acumulando error y dando resultados cada vez peores.

TheEffectsofAIAssistedProgramming inSoftwareEngineering AnObservationofGitHubCopilotintheIndustry

Ontheotherhand,itwasalsoseenthatcontinuousover-relianceonGAIwhen solvingproblemscouldpotentiallylimittheknowledgesoftwareengineersgainasa resultofrelyingtoomuchonthesetools,ratherthanreadingdocumentationand comparingdifferentsolutionsthemselves.

VI. ANÁLISIS DE IMPACTO

La implementación de la Inteligencia Artificial en la ingeniería de software tiene el potencial de aumentar significativamente la productividad de los desarrolladores. Se estima que la inteligencia artificial tendrá un impacto positivo en la productividad de los ingenieros de software de entre un 20 y un 45% respecto a la actualidad, principalmente reduciendo el tiempo invertido en actividades como la generación de borradores iniciales de código, corrección de código y refactorización1. Además, los ingenieros que utilizan herramientas de IA tienden a ser más productivos, satisfechos y permanecen más tiempo en sus trabajos2. Un estudio empírico interno de McKinsey encontró que los equipos de ingeniería de software que fueron capacitados para usar herramientas de IA generativa redujeron rápidamente el tiempo necesario para generar y refactorizar código, además de reportar una mejor experiencia laboral1. Con herramientas basadas en IA generativa, los desarrolladores pueden completar tareas hasta dos veces más rápido4.

En cuanto a los costos, la implementación completa de soluciones de IA puede variar desde US$ 20,000 hasta US$ 1,000,000, incluyendo costos de hardware y software, consultoría, capacitación y, posiblemente, personal adicional requerido para operar la nueva tecnología5. Para 2023, se estima que las empresas podrían pagar desde $0 hasta más de $300,000 por software de IA, con las soluciones personalizadas variando entre $6,000 y más de $300,0006. Los costos de implementación también pueden variar dependiendo de factores como el uso de servicios en la nube o implementaciones locales para modelos de lenguaje de gran tamaño, así como la infraestructura de nube, la experiencia técnica y la mitigación de riesgos necesarios7 8.

Aunque no se encontraron casos de estudio específicos de empresas que hayan publicado resultados exactos sobre el aumento de la productividad después de implementar la IA, las evidencias sugieren que la IA tiene un impacto positivo significativo en la productividad de los desarrolladores. Las compañías que estén considerando la implementación de IA deben considerar los costos asociados y equilibrarlos con los beneficios potenciales en términos de productividad y calidad del software. Para explorar más sobre los precios de las herramientas de IA específicas, se pueden utilizar guías de comparación de precios en línea como la proporcionada por Capterra9. También se pueden explorar diversas herramientas de IA para ingenieros como se sugiere en Spinach.io10, y considerar la inversión en soluciones AI-augmented dedicadas para apoyar roles, tareas y flujos de trabajo en áreas como diseño, codificación, pruebas e integración11.

---
https://www.linkedin.com/pulse/cu%C3%A1nto-cuesta-la-rotaci%C3%B3n-de-personal-red-maple-language-and-talent/?originalSubdomain=es
https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-economic-potential-of-generative-ai-the-next-productivity-frontier#business-value
https://www.gartner.com/en/articles/set-up-now-for-ai-to-augment-software-development#:~:text=Leverage%20AI%20across%20the

%20software,jobs%20than%20those%20who%20don%E2%80%99t
https://wwwhatsnew.com/2022/06/14/desarrolladores-cambiar-trabajo/
https://www.gartner.com/en/articles/employees-seek-personal-value-and-purpose-at-work-be-prepared-to-deliver

La implementación de la Inteligencia Artificial en la ingeniería de software tiene el potencial de aumentar significativamente la productividad de los desarrolladores.

Se estima que la inteligencia artificial tendrá un impacto positivo en la productividad de los ingenieros de software de entre un 20 y un 45% respecto a la actualidad, principalmente reduciendo el tiempo invertido en actividades como la generación de borradores iniciales de código, corrección de código y refactorización.

Además, los ingenieros que utilizan herramientas de IA tienden a ser más productivos, satisfechos y permanecen más tiempo en sus trabajos. Se estima que el coste promedio de reemplazar a un empleado se encuentra entre el 100 y el 300% de su salario anual, en este gasto se incluyen los gastos de publicidad para la vacante, el proceso de selección de nuevos empleados, el periodo de formación una vez el nuevo empleado comienza a trabajar y el tiempo que tarda en llegar a la productividad completa. Además Gartner estima que en los últimos 3 años los empleados han empezado a dar más importancia a sentirse satisfechos y completos con su trabajo.

Un estudio empírico interno de McKinsey encontró que los equipos de ingeniería de software que fueron capacitados para usar herramientas de IA generativa redujeron rápidamente el tiempo necesario para generar y refactorizar código, además de reportar una mejor experiencia laboral1. Con herramientas basadas en IA generativa, los desarrolladores pueden completar tareas hasta dos veces más rápido4.

VII. CONCLUSIÓN

<Insertar Conclusión>

VIII. VALORACIÓN PERSONAL

<Insertar Valoracion personal>

REFERENCIAS (dejo una de ejemplo)

[1] Afnan, K. Muhammad, N. Khan, M.-Y. Lee, A. Imran, and M. Sajjad, “School of the Future: A Comprehensive Study on the Effectiveness of Augmented Reality as a Tool for Primary School Children’s Education,” Applied Sciences, vol. 11, no. 11, p. 5277, Jun. 2021, doi: https://doi.org/10.3390/app11115277.