Uso de Modelos Conversacionales Avanzados para Evaluar la Ambigüedad en Requerimientos de Software

Pedro E. Colla [0009-0004-7463-3035]

FCyT-UADER-Concepción del Uruguay-Entre Ríos-Argentina
(colla.pedro@uader.edu.ar)

Abstract. Este artículo propone evaluar preliminarmente mediante un experimento la aplicabilidad y ventajas de utilizar la tecnología de modelos conversacionales avanzados durante el ciclo de desarrollo de proyectos de software específicamente en la tarea de detectar y corregir ambigüedades en los requerimientos. La utilización de ésta tecnología tiene un enorme potencial para mejorar la calidad resultante del proyecto de software y la drástica reducción del esfuerzo de retrabajo con ventajas significativas en el costo, calidad y satisfacción del cliente.

This article proposes to use an experiment to preliminarily evaluate the applicability and advantages of using advanced conversational model technology during the development cycle of software projects specifically in the task of detecting and correcting ambiguities in requirements. The use of this technology has enormous potential to improve the resulting quality of the software project and drastically reduce rework effort with significant cost & quality advantages and customer satisfaction.

Keywords: IA. ChatGPT. Modelos conversacionales avanzados. Ambigüedad de los requerimientos

1 Introducción

En un contexto de alta competencia las organizaciones desarrolladoras de software desean ser capaces de entregar sus servicios en tiempo y dentro de presupuesto, además de libres de fallas, incluso al punto de ser un factor clave de éxito. A pesar de los avances significativos en la tecnología de desarrollo, y el estado del arte en las disciplinas de ingeniería de software, el potencial para introducir defectos durante el ciclo de vida sigue siendo importante. En tal sentido no es posible producir software completamente libre de defectos, y entonces su solución, denominado usualmente re-trabajo o costo de calidad pobre, suele ser la causa raíz de retrasos en las entregas, excesos presupuestarios e insatisfacción de los clientes.

A menudo las organizaciones despliegan una estrategia basada en mejorar la calidad realizando actividades de validación y verificación, en distintas etapas del proceso de desarrollo, pero primariamente concentrada en la evaluación de código ya construido. Desafortunadamente, esa estrategia tropieza con la dificultad de ser poco eficiente

Proceedings of the 27th Workshop on Requirements Engineering (WER24), August 07-09, 2024, Buenos Aires, Argentina.
DOI: 10.29327/1407529.27-29

puesto que remover un defecto una vez introducido es costoso [1, 2], cuando siquiera posible. Adicionalmente, los defectos son identificados en etapas tardías de los ciclos de desarrollo, provocando deslizamientos significativos en los calendarios y costos en etapas donde los calendarios y presupuestos ya están avanzados. Esto ocurre también cuando se apela al uso de metodologías ágiles para el desarrollo, donde tanto el calendario como el costo puede ser gestionado mediante decisiones u *opciones* [3], pero aun así los problemas de calidad conducen con frecuencia al incremento de la deuda técnica, deterioro de la velocidad, dificultades para entregar componentes críticos según las necesidades del negocio y en forma última insatisfacción del cliente [4].

La evidencia bibliográfica y empírica muestra que cuando se aplica esfuerzo en determinar la trazabilidad de los defectos a la etapa donde fueron introducidos, una proporción muy significativa resultan originados en la etapa de recolección de los requerimientos. Martin [5] indica que aproximadamente el 56% de todos los defectos se originan al formular el requerimiento, y de esa proporción aproximadamente la mitad se corresponde a problemas en el encuadre semántico o en la estructura con el que es expresado el requerimiento, esto es, el mismo es inconsistente, ambiguo, incompleto o confuso. Lazic [2, 2] identifica en forma coincidente que 82% del esfuerzo de re-trabajo está relacionado con errores en los requerimientos, mientras que el 44% de las razones que motivan la cancelación de proyectos se originan en esa etapa; los problemas en el relevamiento de requerimientos son evidente cuando solo el 54% de los requerimientos relevados inicialmente son de hecho implementados alguna vez.

Si las organizaciones típicamente insumen un 60% o más de su esfuerzo de desarrollo en concepto de retrabajo o *costo de calidad pobre* [6] entonces puede recurrirse a un modelo simple de productividad de desarrollo para estimar que del total de costos de desarrollo hay una oportunidad en 20% del costo total asociada con problemas estructurales de los requerimientos en cuanto a cómo están expresados, por lo que cualquier mejora se traduce en la mejora de la productividad del desarrollo, mejoras de calidad del producto, cumplimiento de plazos y satisfacción del cliente.

Desafortunadamente, las organizaciones son a menudo reticentes de aplicar recursos de proyecto a mejorar los requerimientos ante la inherentemente baja productividad de las herramientas disponibles para hacerlo, principalmente diferentes métodos basados en inspecciones, así como la dificultad de conectar claramente el impacto en el negocio que esa acción produce, hasta que es demasiado tarde en todo caso. Las inspecciones son un método formal de encontrar defectos en documentos de diseño y código. Es conocido que la utilización de inspecciones sobre estos documentos presenta numerosos beneficios [7]. Las inspecciones tradicionales, tales como las propuestas por Fagan [8] [9][32] o incluso variantes más modestas en su alcance tales como revisión de entregable (del inglés work product review) [10] o incluso metodologías ágiles como pair programming [11] se han usado eficazmente para intentar realizar la detección temprana de defectos mediante análisis estático de código fuente, pero para ese entonces el aporte de valor realizado sobre los defectos inyectados en las etapas anteriores se transforma en desperdicio.

El esfuerzo involucrado en detectar defectos mediante inspecciones es alto comparado con los defectos que se previenen por ese mecanismo, Kelly y otros [12] [13] reporta que se requiere 1.1 horas de trabajo de inspección por defecto encontrado, sin contar el

tiempo posterior de reparación, y adicionalmente que el número de defectos por unidad de esfuerzo decae con el volumen abordado en cada sesión de inspección, limitando la escala en la que puede aplicarse la metodología. Al mismo tiempo la densidad de defectos capturada mediante inspecciones es mayor durante las instancias de requerimientos y decae a continuación perdiendo su eficacia como herramienta a medida que progresa el ciclo de desarrollo. Múltiples autores resaltan la importancia relativa de detectar las ambigüedades en los requerimientos como forma estructural de mejorar su calidad [14] [15]. El intento de automatizar éste proceso, aún con aplicaciones de dominio estrecho dedicadas a la detección de ambigüedades a partir de coincidencias sintácticas basadas en búsquedas por diccionario de palabras clave o expresiones "*ambiguas*" [16] [17], tropieza con el costo elevado de mantener los diccionarios involucrados y la baja eficacia del método, propenso a tener tanto errores tipo I como tipo II en la detección.

La evolución de las tecnologías relacionadas con inteligencia artificial está transformando el mercado profesional y laboral y su potencial de aplicación en distintas etapas del proceso de desarrollo de software es enorme. Basta considerar los múltiples reportes de éxito en la aplicación directa a la generación de código a partir de una descripción del requerimiento o problema de dominio a resolver [18].

Los avances recientes permiten además substituir mediante tecnología, al menos parcialmente, trabajos que objetivamente pueden ser considerados creativos y que tradicionalmente ha sido realizado por profesionales tales como solución de problemas, desarrollo de software y actividades artísticas [19].

El despliegue, relativamente reciente, de modelos conversacionales avanzados implementados mediante autómatas como ChatGPT (*Generative Pre-trained Transformer*), desarrollado por la empresa OpenAI [20], introduce avances significativos y también desafíos en este campo. Este tipo de plataformas permite a partir de un modelo lingüístico entrenado mantener conversaciones cuyas respuestas pueden crear la ilusión de implicar comprensión; otros autómatas de comportamiento y performance similares han aparecido también recientemente [21].

ChatGPT pertenece a la categoría de las "*inteligencias artificiales generativas*", redes neuronales informáticas capaces de generar textos o imágenes por su propia cuenta.

El mecanismo de interacción usado por la plataforma utiliza el paradigma de una conversación basada en texto, similar quizás a la imaginada por Turing al definir su legendario experimento denominado "*test de Turing*" [22] para dilucidar sobre la supuesta inteligencia de un autómata. La interacción resultante es relativamente sencilla y ágil, basándose en un paradigma "acción-respuesta" que permite establecer la ilusión de una conversación. Como parte de esta interacción la plataforma puede ser utilizada para generar, con sorprendente eficacia, código en distintos lenguajes a partir de un requerimiento expresado en texto llano. Pero aun cuando se aplique esta tecnología a la generación de código, y esto fuera eficaz como solución, esto no resuelve los problemas que se originan en un requerimiento deficiente como punto de partida.

El análisis holístico de los requerimientos de un aplicativo en particular probablemente exceda largamente las posibilidades técnicas de un autómata de este tipo pues depende de la disponibilidad de información de contexto y conocimiento de dominio que puede ser costoso de integrar. Pero es natural preguntarse qué clase de beneficio puede obte-

nerse en su aplicación al análisis de requerimientos desde una perspectiva más semántica del lenguaje natural en el que están expresados, en particular si los mismos expresan una idea en forma completa, sintácticamente correcta y en forma no ambigua.
La contribución de este artículo es realizar una exploración preliminar sobre estas cuestiones mediante las siguientes preguntas de investigación:

- *¿Cuál es la capacidad de utilizar un modelo conversacional avanzado para el análisis de requerimientos con el objetivo de identificar ambigüedades en los mismos?*

- *¿Como se compara su performance en costo respecto del mismo proceso realizado en forma manual?*

2 Diseño de experimento

Se conducirá un experimento consistente en colectar requerimientos reales, desde organizaciones participantes que estén dispuestas a cederlos, para procesarlos utilizando ChatGPT bajo determinados contexto o declaración de misión o "*prompt*", directiva y parámetros empíricamente seleccionados. Se observará el resultado, el cual será validado mediante inspección manual, tal como se haría en una revisión de pares.
El número de requerimientos mínimo a procesar definirá las características de error y validez del soporte estadístico de los resultados obtenidos. Al respecto se considera que los requerimientos son obtenidos desde un repositorio de origen y seleccionados al azar, siendo el número extraído muy bajo respecto a la población de requerimientos origen. El número de muestras a utilizar (n_0) vendrá dado por [23] según la Ec 1

$$n_0 = \frac{Z_\alpha^2 pq}{e^2} \qquad \textbf{Ec 1}$$

Para un alfa de Cronbach α=0,05 resulta Z_α=1,96, tomando p=q=0,5 y e=25% para un intervalo de confidencia del 95% se obtiene que n0 requiere como mínimo n_0=15 muestras. Para las evaluaciones que impliquen explorar relaciones mediante regresiones lineales se utilizará como criterio de aceptación el obtener ρ^2>0.70 con lo que se asegura que al menos el 70% de la variación de la variable dependiente sea explicada por la variable independiente del modelo.

3 Análisis de la muestra

Se recolectan requerimientos reales desde dos organizaciones no relacionadas entre sí, extraídos desde repositorios asociados a desarrollos en curso en las mismas. El número total de requerimientos obtenidos es n_s=22 con lo que se satisface la condición de volumen de muestra establecida. Los requerimientos están formulados en español lo que conduce que todo el experimento utilice este idioma. Los requerimientos obtenidos responden a mecanismos de colección diferentes por lo que son desiguales en cuanto a extensión. No se hace esfuerzo por adecuar el texto de los requerimientos ni suplementar su semántica, solo realizándose una adecuación basada en la corrección de errores

de tipeo y signos de puntuación necesarios para adaptar el formato de ingreso de las consultas a ChatGPT.

4 Procesamiento de la muestra

La muestra es procesada mediante un script escrito en Python utilizando el API definido para ChatGPT [24] [25] con los siguientes parámetros. *temperatura* (2), *max_tokens* (4096), *top_p*=1, *frequency_penalty*=0, *presence_penalty*=0, estos parámetros fueron seleccionados a partir de recomendaciones del proveedor, ejemplos evaluados y experiencia empírica del autor, otros parámetros pueden ser requeridos de acuerdo al dataset utilizado para la evaluación. El modelo utilizado fue *gpt_3.5-turbo-0125*. Al momento del procesamiento otros modelos mas avanzados estaban disponibles, pero se evaluó que el modelo utilizado era suficiente para el alcance de la evaluación y su costo de utilización es significativamente mas reducido.
Se utiliza como contexto el enunciado de declaración de misión "*Tu eres un joven, inteligente, muy meticuloso, ordenado y exhaustivo ingeniero de software al que le han asignado la difícil tarea de identificar ambigüedades en los requerimientos de un aplicativo*" y como tarea generadora "*identifica y enumera las ambigüedades que puedan encontrarse mediante la lectura del siguiente requerimiento*". Como tarea o consulta generativa se utiliza directamente el requerimiento. No se hace esfuerzo alguno para que el motor conversacional relacione requerimientos sucesivos más allá de la ventana conversacional default que tenga, los efectos de esta decisión se discutirán posteriormente.

5 Resultados de la ejecución

Se obtiene un archivo (dataset) consistente en pares requerimiento-respuesta, todos los requerimientos ingresados producen una respuesta que es sujeta al análisis. Una primera cuestión a resolver es establecer las métricas básicas apropiadas de tamaño de entrada y salida por cada requerimiento. ChatGPT dimensiona los pedidos y las respuestas con una métrica interna denominada "*token*", la que tiene directa relación con el esfuerzo de procesamiento y por lo tanto su costo, pero no se asocia en forma explícita con el texto utilizado como pregunta ni el producido como respuesta.
Una primera evaluación correlaciona el tamaño de cada requerimiento (S_{in}, en bytes) con los tokens de entrada (T_{in}, en tokens) reportadas por ChatGPT, ambos resultan relacionados ($r_{Pearson}$=0.999). Por su parte el tamaño de las respuestas (S_{out},en bytes) también correlaciona con los tokens de salida ($T_{out,}$ en tokens) con $r_{Pearson}$ =0.975, por lo que en el resto del trabajo se asumirá poder realizar evaluaciones utilizando tamaño en bytes o tokens indistintamente. No existe correlación entre el tamaño de entrada y de salida de un dado requerimiento ($r_{Pearson}$=0.398). Al mismo tiempo comparando las poblaciones de requerimientos según su origen y analizando las medianas del tamaño de requerimiento mediante un *2-sample test* con medianas iguales como hipótesis nula se puede confirmar que las medianas del tamaño no son iguales (p-value≈0). Bajo el mismo test, pero comparando el tamaño de las respuestas para ambos conjuntos no se

puede establecer que el tamaño de la salida sea distinto al margen seleccionado (p-value=0.06) mostrando que los resultados no difieren significativamente en tamaño para requerimientos en ambos grupos de requerimientos.
Como resultado de la ejecución cada requerimiento recibe a modo de respuesta un número de recomendaciones (δ). La cantidad de recomendaciones obtenida por requerimiento no se correlaciona ni con el tamaño de la entrada ($r_{Pearson}$=0.321) ni con el tamaño de la salida ($r_{Pearson}$=0.545). De hecho, la distribución de la cantidad de recomendaciones es aproximadamente normal (test de Anderson-Darling p-value=0.019), mostrando una mediana de 6 recomendaciones por requerimiento con poca dispersión (σ=0.25).
Se revisan manualmente las recomendaciones dadas como resultado sin observarse errores tipo I o tipo II significativos entre lo recomendado y lo revisado, se toma nota del tiempo insumido en la revisión manual de cada requerimiento insumiendo una mediana de 200 segundos por requerimiento (min=69,5 seg y max=659,5 seg), el tiempo de revisión manual correlaciona con el tamaño del requerimiento medido en tokens ($r_{Pearson}$=0.94).

6 Factores de costo

Independientemente de los resultados analíticos del experimento el mismo tiene sentido si se obtiene ventajas de costo en su utilización. Es posible obtener el tiempo de ejecución de cada requerimiento, pero el mismo no es factor de costo. El modelo de costos de ChatGPT está basado en la cantidad total de tokens de entrada y salida involucrados en una consulta. Distintos modelos tienen costos diferentes, naturalmente los más modernos son más potentes, pero al mismo tiempo más costosos. El modelo utilizado tiene un costo por cada 1000 de token de entrada de USD 0.0005 y cada 1000 tokens de salida de USD 0.0015. Es posible establecer un modelo de regresión lineal entre el costo total de procesamiento y el tamaño en bytes de la entrada ($r_{Pearson}$=0.712) dados por la [Ec 2]

$$C_{GPT} = 0.000000207\, S_{in} + 0.00068. \qquad \text{[Ec 2]}$$

Cada organización tendrá su propia línea de base relativa a sus costos medios por ingeniero, a los efectos de este análisis se utiliza una figura basada en la experiencia de *costo por ingeniero* (CPE) de USD 18/hora. Es también posible establecer un modelo de regresión lineal entre el tamaño de entrada del requerimiento y la relación entre costo de inspección manual y automatizado, como una manera de evaluar las ventajas en costo del mecanismo automatizado (ρ^2=0.732) dado por la [Ec 3

$$\frac{C_{manual}}{C_{GPT}} = 0.75\, S_{in} + 656. \qquad \text{[Ec 3]}$$

La relación anterior sugiere que la relación entre el costo de revisión manual y automática para para un dado requerimiento es de tres órdenes de magnitud superior, mostrando una clara ventaja del método automático asumiendo iguales resultados. Esta con-

clusión permite estimar que esta herramienta tiene el potencial de reducir significativamente el retrabajo en un proyecto informático y que si hoy no se realiza puede ser incorporada a los procesos de desarrollo a un costo y eficacia muy competitivo respecto a cualquier otra alternativa disponible hasta el momento.

7 Conclusiones

Abordando las preguntas de investigación formuladas previamente

- *¿Cuál es la capacidad de utilizar un modelo conversacional avanzado para el análisis de requerimientos con el objetivo de identificar ambigüedades en los mismos?*

  El análisis realizado muestra que el modelo conversacional avanzado puede efectivamente ser utilizado para identificar ambigüedades en los requerimientos sin identificarse errores tipo I y tipo II significativos proveyendo recomendaciones y hallazgos relevantes en los requerimientos de la muestra utilizada.

- *¿Como se compara su performance en costo respecto del mismo proceso realizado en forma manual?*

  En el experimento el costo de análisis automatizado es ventajoso respecto de la inspección manual en aproximadamente tres órdenes de magnitud. Los resultados concretos obtenidos por la organización variarán con el tamaño del proyecto en entregables y esfuerzo, pero con los guarismos previamente se puede apelar al *criterio de Gott* o *principio de la mediocridad* [26] para especular que si se reduce el retrabajo originado en ambigüedades en los requerimientos con el uso de ésta herramienta en un 50% puede resultar en mejoras de costo del 7 al 10% para la totalidad del proyecto, lo que es una magnitud significativa comparado con otros mecanismos de mejora de la productividad, como por ejemplo despliegue de marcos de calidad [27, 28].

8 Amenazas a la validez

Siendo un trabajo preliminar, donde se integran bibliografías desde diferentes industrias y proyectos, es necesario tomar los resultados con cautela y perseverar en medidas ulteriores de validación. La muestra utilizada para la evaluación es pequeña y los márgenes de error aceptados altos, pero eso permite un primer abordaje de la temática y un anclaje bibliográfico para trabajos futuros previamente no encontrados en la revisión bibliográfica. Las distribuciones y parámetros utilizados para la ejecución de escenarios son susceptible de cambiarse para acomodar variaciones en organizaciones o proyectos específicos. Está implícito en los comentarios precedentes que respecto a su utilización en la validación de requerimientos la consistencia en las respuestas provistas por esta plataforma están acompañados por desafíos y amenazas significativas a la validez, las que deben ser conocidas, abordadas, mitigadas o resueltas para su uso efectivo [29].

Al respecto es importante tener en cuenta que ChatGPT no tiene conciencia ni intención, sus respuestas mimetizan con variado éxito una interacción con personas, lo que hace inevitable su "antropomorfización"; pero sólo es un modelo matemático que se basa en patrones de lenguaje, por lo que las respuestas generadas no son originales, sino que son combinaciones de frases y palabras previamente vistas en el texto con el que fue entrenado [30]. Es por lo tanto difícil seguir los pasos clásicos de validación de un esfuerzo de modelado [31].

Esto nos enfrenta con el mayor desafío para su uso. Sabemos muy poco, o no sabemos nada, sobre el corpus y las políticas de información involucradas en su entrenamiento. Por lo tanto, no estamos en condiciones de evaluar, mucho menos gestionar, los riesgos derivados de inexactitudes y sesgos introducidos durante ese proceso, sean los mismos intencionales o no, en tópicos relacionados con el género, la raza y la posición socioeconómica de los involucrados, entre otras.

Al mismo tiempo toda la información que un usuario provee para interactuar con esta plataforma está a disposición de los desarrolladores de la misma, y no hay políticas explícitas ni mecanismo de control o transparencia alguno sobre su uso, lo que claramente deriva en la preocupación sobre la privacidad de las personas que lo usan y la seguridad física o lógica de los proyectos cuya información es expuesta. No es claro el marco legal aplicable sobre la propiedad intelectual de la información intercambiada.

La naturaleza del algoritmo que implementa ChatGPT hace que siempre responda, aunque en ocasiones debería no hacerlo. El ingresar requerimientos en forma sucesiva aceptando la ventana de contexto que el motor utilice puede generar algún problema de interacción entre requerimientos de distintos orígenes que deberá ser evaluada en el futuro, con experimentos específicos sobre la cuestión.

La respuesta ante preguntas similares puede variar significativamente en función de parámetros de ejecución, por ejemplo, el parámetro denominado "temperatura" que controla la "creatividad" permitida al modelo en la formulación de las respuestas; y eso puede conducir, justamente, a que preguntas similares generan respuestas disimiles o incluso francamente erróneas si se incrementa la "libertad" en la elección de posibles respuestas. No hay una medida tangible que permita evaluar la relevancia o exactitud de una respuesta, solo la verificación separada e independiente, la cual se dificulta por la falta de referencias provistas por el motor en la respuesta sobre las fuentes utilizadas. Es necesario al mismo tiempo de establecer un protocolo de utilización que proteja, al menos estadísticamente, sobre ocasiones donde la respuesta del modelo a un requerimiento dado no sea correcta.

El modelo utilizado (*ChatGPT 3.5*) no es la versión más reciente, esto en principio ocurre por restricciones en el costo de la investigación realizada, pero también para poder establecer una base de viabilidad que debería mejorar con la evolución de los modelos.

9 Trabajo Futuro

Se observan comportamientos en las respuestas que resultan poco intuitivos a-priori y que pueden indicar la necesidad de refinar el uso del modelo, en tal sentido es necesario

continuar haciendo evaluaciones para explicar satisfactoriamente porque el número de recomendaciones es similar para tamaños de requerimiento muy disimiles y porque el número de recomendaciones obtenidas es similar para para todos los casos presentados. El número de posibles vías de trabajo es significativamente alto. Las organizaciones u otros esfuerzos de investigación pueden repetir el mismo experimento con sus propios requerimientos, incluso de diferente dominio o estructura semántica, puesto que la plataforma es pública. Todas las interacciones utilizan idioma español, asumiendo que los resultados obtenidos se replicarán en otros idiomas, hipótesis que hay que validar. Claramente es necesario repetir los experimentos realizados con algún énfasis en establecer la sensibilidad de los resultados a factores tales como parámetros de configuración, declaración de misión (prompt engineering), contexto e incluso corridas sucesivas. También es un objetivo futuro repetir los experimentos con modelos más avanzados del motor conversacional (ChatGPT4 en particular) y comparar los resultados con los obtenidos.

10 Referencias

1. Lazić, L. Software Testing Optimization by Advanced Quantitative Defect Management, *Computer Science and Information Systems Computer Science and Information Systems,* pp. 459-487, 2010.
2. Lazić, L. Requirements-Based Testing Process in Practice, *ComSIS,* pp. Vol. 7, No. 3,, June 2010.
3. Colla, P. Marco para evaluar el valor en metodología SCRUM, de *XIII Simposio Argentino de Ingeniería de Software (JAIIO-ASSE)*, CABA, 2012.
4. Ruiz de Mendarozqueta A. y et.al., Agile in practice, a systemic approach, de *XXI Simposio Argentino de Ingeniería de Software (ASSE-2020)*, Virtual, 2020.
5. Martin, J. An Information Systems Manifesto, Prentice HallI ISBN-10 : 0134647696, 1984.
6. Knox, S. Modeling the Cost of Software Quality, *Digital Technical,* pp. pp 9-16, 1993.
7. Boulanger J. y et al, Certifiable Software Applications, Elsevier Ltd., 2017.
8. Fagan, M. Design and code inspections to reduce errors in program development, *IBM Systems Journal. 15,* p. 182–211, 1976.
9. Vreede G. y et al, A Collaborative Software Code Inspection: the Design and Evaluation of a Repeatable Collaboration Process in the Field, *International Journal of Cooperative Information Systems,* pp. 205-228, 2006.
10. Souza de Olivera, P. y et al, Work Product Review Process Applied to Test Cases Review for Software Testing, de *SBQS '23: Proceedings of the XXII Brazilian Symposium on Software Quality*, 2023.
11. Williams, L. Integrating pair programming into a software development process, de *14th Conference on Software Engineering Education and Training*, Charlotte, NC, USA, 2001.
12. Kelly, J. An analysis of defect densities found during software inspections, NASA N92-19435, 1992.

13. Zapata, S. Obtención de métricas a partir del proceso de inspección de software, de *IX Congreso Argentino de Ciencias de la Computación*, 2003.
14. Audacio E. y et al, A Method to Evaluating Requirements, de *Proceedings of the 24th Workshop on Requirements Engineering (WER21)*, 2021.
15. Neto D. y et.al., Ambiguity resolution of legal requirements: an exploratory study in the literature, de *Proceedings of the 26th Workshop on Requirements Engineering (WER23)*, Porto Alegre, RS, 2023.
16. Schmitz S., An Experimental Ambiguity Detection Tool, *Science of Computer Programming Science of Computer Programming,* pp. 71-84, 2010.
17. Peterson R., Requirements Ambiguity Checker Tool - Installation and Usage Instructions, California, Department of Technology, 2014.
18. Bubeck S.y et al, Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.
19. Hutter, S. L. A Formal Measure of Machine Intelligence, *IDSIA, Galleria 2,* pp. 1-8, 2006.
20. OpenAI, OpenAI, *https://platform.openai.com/docs/guides/text-generation,* 2024.
21. Singh, S. Chat GPT & Google Bard AI: A Review, de *2023 International Conference on IoT, Communication and Automation Technology (ICICAT)*, 2023.
22. Turing, A. Computing machinery and intelligence, 1950.
23. Jones, S. y et al, An introduction to power and sample size estimation, *Emergency medicine journal: EMJ,* pp. 453-458, 2003.
24. Younis, M. y et al, Enhancement of ChatGPT using API Wrappers Techniques, *Al-Mustansiriyah Journal of Science V34 N2,* pp. 82-86, 2023.
25. OpenAI, OpenAI ChatGPT API, *https://platform.openai.com/docs/guides/text-generation,* 2024.
26. Gott, J. Implications of the Copernican principle for our future prospects, *Nature, Volume 363, Issue 6427,* pp. pp. 315-319, 1993.
27. Clark, B. The Effects of Software Process Maturity on Software Development Effort PhD dissertation, USC, 1997.
28. Clark, B. Quantifying the effects of Process Improvement on Effort, *IEEE Software Nov/Dec,* pp. 65-70, 2000.
29. Zhai, X. ChatGPT User Experience: Implications for Education., *Available at SSRN 4312418,* 2022.
30. Olson, P. Cuidado con ChatGPT tratando de enseñar a sus hijos matemáticas... o cualquier cosa., 2023.
31. Sargent, R. Verification and validation of simulation models, de *Proceedings of the 2010 winter simulation conference*, 2010.
32. Fagan, M. A history of software inspections. Software pioneers: contributions to software engineering, de *A history of software inspections. Software pioneers: contributions to software engineering*, 2002, p. 562–573.