Inteligencia Artificial en Ingeniería de Software Etapas de Elicitación y Análisis de requerimientos

Ailén Panigo, Kristian Petkoff Bankoff, Ariel Pasini, and Patricia Pesado

III-LIDI, Facultad de Informática
Universidad Nacional de La Plata, Argentina
{apanigo,kpb,apasini,ppesado}@lidi.info.unlp.edu.ar

Resumen La inteligencia artificial, en particular los modelos de lenguaje de gran tamaño (LLM), asiste a diversas disciplinas entre las que se encuentra la ingeniería de software. En este artículo se presenta un caso de estudio para analizar la viabilidad del uso de LLMs en el proceso de igneniería de software, aplicando diversas herramientas para la generación de entrevistas en la etapa de elicitación y en la etapa de análisis y especificación de requerimientos mediante historias de usuario.

Keywords: Software Engineering, Large Language Models, Software Requirements Specification

1. Introducción

El desarrollo de software es una disciplina que involucra diversas etapas desde la obtención de requerimientos, la planificación y el diseño hasta la implementación, las pruebas y el mantenimiento del producto. A lo largo de la historia, y aprovechando el avance de la tecnología, se han incorporado nuevas herramientas y metodologías que permiten la optimización de estas fases.

La Inteligencia Artificial (IA) como tecnología ha encontrado diversas aplicaciones en muchas disciplinas. Su capacidad para procesar grandes cantidades de datos, identificar patrones y tomar decisiones automatizadas ha permitido su integración en diversos sectores, entre los que se incluye la ingeniería de software. Con la IA se logra también la automatización de tareas repetitivas, la innovación, precisión y eficiencia en pruebas de software [1].

La asistencia de la IA en la ingeniería de software puede ser aplicada en todas las instancias del proceso, desde la elicitación de requisitos hasta la puesta en producción del software [2]. En este trabajo se hará hincapié en la etapa de elicitación de requisitos analizando la generacion de entrevistas, y en la etapa de análisis y especificación de requerimientos, con la generación de historias de usuario.

En la sección 2 se presentan los conceptos generales del proceso de desarrollo de software e inteligencia artificial, en la tercera sección se presenta la metodología utilizada en el trabajo, seguido por la sección 4 donde se describen los resultados del caso de estudio y finalizando con las conlusiones.

2. Conceptos Generales

En esta sección se mencionarán conceptos fundamentales de las etapas de desarrollo de software, el uso de herramientas de IA y la vinculación de los mismos.

2.1. Proceso de desarrollo de software

El desarrollo de software es un proceso sistemático y estructurado que se compone de varias etapas, formando lo que se conoce como el ciclo de vida del desarrollo de software [3]. Estas etapas generales proporcionan un marco conceptual y metodológico que ayuda a guiar a los desarrolladores en el proceso [4].

Elicitación de requerimientos. La elicitación de requerimientos es la etapa inicial del proceso de desarrollo de software, donde el equipo recopila información detallada de los interesados sobre sus necesidades y expectativas para el software. Esta fase establece la base para todo el proyecto. Los desarrolladores se involucran con los clientes para entender sus requisitos, preferencias y cualquier restricción que pueda afectar el proyecto. Se suele generar uno o más documentos que recopilan toda la información obtenida, incluyendo las características y funciones necesarias que se deben incluir en el software [5].

Las entrevistas en esta etapa funcionan como una herramienta para la obtención de los requerimientos. La comunicación entre el cliente y el equipo de desarrollo permite generar una visión detallada de las necesidades y preferencias para el sistema por parte del entrevistado [6].

Análisis de requerimientos. Implica una evaluación de los requerimientos recopilados. Se clarifican y se priorizan los requisitos para asegurar la comprensión de todas las partes interesadas [5].

Las historias de usuario son una herramienta popular para especificar los requerimientos ya que consisten en una descripción de una funcionalidad de un sistema desde el punto de vista del usuario, donde se puede saber la motivación por la cual es necesaria como requerimiento [7]. Aproximadamente el 70 % de las historias de usuario siguen una plantilla simple que ayuda a capturar los requisitos funcionales de manera efectiva [8].

En la etapa de análisis también se puede incluir la decisión del enfoque que llevará a cabo el proyecto, desde la metodología de desarrollo que se seguirá para guiar el flujo de trabajo y procesos del equipo, hasta la división de tareas y responsabilidades en el proyecto. En este sentido, las historias de usuario permiten que el equipo de desarrollo priorice y se enfoque en las tareas que más valor aportan al cliente [9].

Desarrollo. Se lleva a cabo la implementación del código y se realiza la construcción del producto. Por otra parte, se desarrolla el entorno de pruebas de manera de estar sincronizado con la implementación. También se realiza el monitoreo del progreso y comparación con la planificación, por parte del gerente del proyecto. En esta etapa los desarrolladores documentan el código y lo depuran para mantenerlo actualizado y libre de errores [10].

Pruebas. En la etapa de pruebas se identifican los errores de programación y diseño. Los errores encontrados se reportan a los equipos de desarrollo para que sean solucionados.

Se realizan pruebas unitarias, de integración y de aceptación, para asegurar que el software cumpla con los requisitos especificados y funcione de forma correcta. El propósito es validar el rendimiento, seguridad y usabilidad del software [11].

Entrega y mantenimiento. En esta etapa se pone a disposición el software a los usuarios. Puede incluir capacitaciones y documentación sobre su uso. La entrega de un software se puede hacer por fases o de forma completa y también puede incluir tareas de mantenimiento para solucionar inconvenientes que se presenten durante el uso del sistema o la evolución del producto.

2.2. Herramientas de IA

Las herramientas y modelos de inteligencia artificial se han diversificado, abarcando desde algoritmos de aprendizaje supervisado y no supervisado hasta redes neuronales complejas y sistemas de procesamiento de lenguaje natural. Estas herramientas son fundamentales para diversas aplicaciones, como la predicción de tendencias, la automatización de procesos, la personalización de servicios y la creación de sistemas autónomos, entre las que podemos mencionar:

ChatGPT-3.5. El conocimiento general de ChatGPT sobre eventos ocurridos después de 2021 es bastante escaso y en ocasiones el contenido generado puede ser de opinión o *controversial*. No posee acceso a internet, por lo que hay información en la que no se puede profundizar incluso cuando el modelo ya tiene entrenamiento de base en algún tema.

Limitado solo al ingreso de texto. Permite procesar hasta 175 mil millones de parámetros, lo cual es un número limitante en comparación a otros modelos. Presenta limitaciones a la hora de interpretar enunciados de lógica, razonamiento o matemática avanzada. Soporta varios idiomas pero su mayor efectividad se encuentra en inglés, permite retener hasta 3000 palabras en un contexto dado y genera tokens de respuesta en unos 73 milisegundos.

ChatGPT-4o. Permite el procesamiento de texto, audio e imágenes y la combinación de cualquiera de ellos, todos los inputs y outputs son generados por la misma red neuronal. Es versátil y más poderoso, más capaz de resolver problemas complejos que ChatGPT 3.5. Tiene soporte de más de 50 lenguajes y ofrece traducción en tiempo real. Su capacidad es de 128.000 tokens, y genera respuestas en unos 320 milisegundos.

ChatGPT con integración de Write for Me. Write For Me es una herramienta integrada con ChatGPT que asiste en la generación de correos electrónicos, mensajes y otros tipos de texto, y puede ser utilizado para obtener historias de usuario. Entre otras funcionalidades también puede corregir gramática, cambiar los tonos en un texto, realizar resúmenes o parafraseos.

Gemini 1.0. Gemini es una familia de LLMs desarrollada por Google y publicada en 2023 con las variantes Ultra, Pro y Nano. Específicamente para estas pruebas se usó la versión de Gemini 1.0 Pro a través de Google AI Studio. Al

igual que los otros modelos, soporta texto y archivos y tiene una ventana de contexto de 30.720 tokens. Los estudios hechos por Google indican que este modelo tiene un 71,8 % de precisión para conocimientos generales evaluados mediante MMLU benchmark [12].

Gemini 1.5 pro. Es un *transformer* basado en una mezcla dispersa de expertos, posee una arquitectura de redes neuronales en la que cada red neuronal es un *experto* y el modelo se encarga de asignar una tarea a la red o las redes neuronales especializadas para la misma. La dispersión permite que solo se activen *expertos* específicos para cada entrada, favoreciendo la eficiencia. Su capacidad de contexto es de (2.097.152 *tokens* y es nativamente multimodal, lo que permite que maneje texto, imágenes, audio y video [12].

Mistral. Mistral AI es una empresa francesa que ha desarrollado una gran cantidad de modelos de lenguaje generativos, entre ellos Mistral 7B o Mistral 8x7B. Para las pruebas se usó *Le Chat*, una herramienta de Mistral AI que permite comunicarse con un modelo de lenguaje en varios idiomas. Permite seleccionar entre los modelos Mistral Small y Mistral Large, de manera de obtener respuestas más compactas o más desarrolladas según lo que se necesite.

AI21 Studio. AI21 Studio es una plataforma creada por AI21 Labs especializada en procesamiento de lenguaje natural a través de diferentes servicios. Los modelos de lenguaje desarrollados por esta empresa son Jurassic-2 y Jamba y posee una herramienta de chat para poder experimentar con una preview de Jamba (Jamba Instruct) que cuenta con una ventana limitada a 4096 de los 256000 tokens soportados por el modelo.

2.3. Uso de LLM en la obtención y análisis de requerimientos

La implementación de inteligencia artificial y modelos de lenguaje se podría aplicar a todas las etapas del proceso de desarrollo de software.

En la etapa de elicitación de requerimientos se pueden usar los grandes modelos de lenguaje (LLM) para recopilar la información documentada y obtener los requerimientos de los interesados en el desarrollo del sistema, además de predecir futuros comportamientos del software.

En el análisis puede reconocer patrones en los requisitos y generar una planificación precisa y detallada en base a las necesidades y prioridades que tenga el proyecto, como también analizar datos históricos para predecir desafíos y cambios.

En la fase de desarrollo la inteligencia artificial sirve principalmente como herramienta de generación de código. Esto permite un gran aumento en la productividad al sugerir fragmentos de código, funciones o incluso documentando la implementación. En la etapa de pruebas, las herramientas basadas en inteligencia artificial permiten automatizar procesos como realizar pruebas continuas, identificación de errores o sugerencias [13].

En la entrega de un producto se puede facilitar el proceso de despliegue y es de utilidad para el monitoreo y mantenimiento de las aplicaciones. Es posible, por ejemplo, analizar el comportamiento del usuario y el rendimiento del sistema de cara a la evolución, mantenimiento o actualización del mismo [14].

3. Caso de Estudio

Este estudio se centra en la etapa inicial del proceso de desarrollo de software, específicamente en la obtención y el análisis de los requerimientos. Para realizar esta evaluación, se generan distintos *prompts* y cada uno se envía cada modelo descrito en la sección anterior. El propósito es comparar la efectividad de estos modelos en la captura y análisis de requerimientos, evaluando su desempeño en esta fase del proceso.

Se considera un anuncio ficticio que explica la intención de tener una entrevista para transmitir las necesidades de desarrollo de un producto de software, y también un conjunto de requerimientos que describen las funcionalidades, restricciones, usuarios, y datos necesarios para desarrollar el mismo.

3.1. Metodología

Para la generación de las entrevistas se siguen los siguientes lineamientos:

1. Seleccionar un texto que simule un simple anuncio solicitando un entrevista entre un cliente y un equipo de desarrolladores para elaborar un sistema de interés para el cliente.
2. Seleccionar un enunciado que detalle requerimientos y restricciones de un sistema ficticio.
3. Generar tres prompts
   a) En el primero se incluye tanto el anuncio como un contexto reducido de requerimientos típicos y se solicitan dos entrevistas dirigidas al cliente por parte del equipo de desarrolladores, la primera con el objetivo de obtener información general del negocio y la segunda con requerimientos específicos del producto.
   b) En otro prompt, se incluye solo el anuncio sin más contexto y se solicita el cuerpo de una entrevista para obtener información general.
   c) En un tercer prompt se envía el enunciado junto con los requerimientos obtenidos previamente y se solicita el cuerpo de una entrevista donde se permita profundizar en los requerimientos funcionales.
4. Dividir los prompts en dos contextos distintos y hacer las consultas sobre los modelos de inteligencia artificial. El primer contexto únicamente con el primer enunciado, y el segundo con los dos restantes.
5. Obtener las respuestas de ambos prompt, comparar los resultados obtenidos en cada modelo y al mismo tiempo las diferencias entre consultar las dos entrevistas conjuntamente o de forma separada.

Para la generación de historias de usuario, por otra parte, se sigue el siguiente procedimiento:

1. Seleccionar un enunciado que detalle requerimientos y restricciones de un sistema ficticio.
2. Identificar una estructura simple y una más compleja de historias de usuario. El primer formato solo contiene el título mientras que el segundo incluye reglas de negocio y los criterios de aceptación según la sintaxis Gherkin [15].

3. Generar prompts
   a) Un primer enunciado con los requerimientos y la solicitud de generación de historias de usuario que se adapten a las funcionalidades descritas y que respete el formato simple de historia de usuario.
   b) Un segundo enunciado con los requerimientos y la solicitud de historias de usuario que de igual manera, se adapte a las funcionalidades descritas y que respete el formato completo de historia de usuario.
4. Enviar a los modelos de lenguaje los prompts desarrollados en dos contextos diferentes.
5. Obtener las respuestas de los modelos y realizar una segunda solicitud para obtener más historias de usuario que respeten los requerimientos y el formato correspondiente.
6. Repetir el paso anterior en una tercera iteración.
7. Agrupar todas las historias de usuario generadas en las tres iteraciones. Analizar y comparar los resultados obtenidos.

3.2. Obtención de requerimientos

Para poder evaluar el desempeño en la obtención de requerimientos se generaron dos *prompts* solicitando a los modelos de lenguaje dos entrevistas que un equipo de desarrolladores realizaría a un potencial cliente que demande un producto de software. La diferencia entre ambos casos consiste en solicitar dos entrevistas (una general y otra más detallada) en una sola petición, o realizar dos peticiones independientes, donde la segunda entrevista contará con un contexto más detallado obtenido de la primera.

El primer prompt consta de 2467 palabras incluyendo el anuncio y los requerimientos (2332), y solicitan las preguntas que podrían formar parte de una primera entrevista considerando solo el anuncio, y al mismo tiempo se pide una segunda entrevista donde se tengan en cuenta los requerimientos incorporados como información obtenida en la primera entrevista.

En el caso del segundo prompt se solicitan las preguntas de la entrevista en una primera iteración con 194 palabras, solo incluyendo información del anuncio. Luego de obtenidas las preguntas, se envían los requerimientos en una segunda iteración con aproximadamente 2000 palabras, solicitando las preguntas de una posible segunda entrevista, teniendo en cuenta que los requerimientos enviados en esta iteración fueron obtenidos en la primera.

Este último contexto implica un ajuste en pequeños detalles de los requerimientos enviados en la segunda iteración, para que estos se adecúen a lo generado por el modelo de lenguaje como primera entrevista. El ajuste no puede ser significativo para que no hayan diferencias entre los modelos y posteriormente puedan seguir siendo comparables.

3.3. Historias de Usuario

Para la obtención de requerimientos se solicitó a los modelos de lenguaje la generación de historias de usuario utilizando dos estructuras diferentes. La información enviada junto con los prompts es la misma para ambos casos.

La primera estructura consiste en solo generar un título para las historias de usuario que respete el formato *como [rol], quiero [acción] para [objetivo]*. Luego de esta primera iteración se solicitan más historias de usuario en un máximo de dos iteraciones adicionales, que respeten el formato establecido y sean pertinentes para el enunciado.

La segunda estructura consiste en generar historias de usuario con una estructura más compleja. Además del título mencionado previamente, también se solicita enumerar reglas de negocio que afectan a cada requerimiento, junto con los criterios de aceptación siguiendo la sintaxis Gherkin[15]. De la misma forma que en la primera estructura, la información se envía en una primera iteración y luego se solicita generar más historias de usuario en una segunda y posible tercera iteración.

Para ambos casos se espera un total aproximado de 70 historias de usuario que permitan cubrir los requerimientos funcionales del sistema descrito en el enunciado.

3.4. Criterios de aceptación

Para la evaluación de los resultados se define una escala de criterios atendiendo a las siguientes características: completitud, idioma adecuado, repitencia de conceptos ya resueltos, cumplimiento, alcance o profundidad de desarrollo adecuados.

- **Inaceptable (IN)**: La respuesta es incompleta, en otro idioma distinto al solicitado, repite muchos conceptos ya resueltos, o no cumple con lo especificado.
- **Mínimamente aceptable (MA)**: La respuesta cumple con lo pedido pero no se enfoca en los límites de las especificaciones, para el caso de las historias de usuario, o responde pero podría abarcar más aspectos, para el caso de las entrevistas.
- **Aceptable (AC)**: La respuesta es adecuada y cumple con lo solicitado, pero podría desarrollarse con un poco más de detalle.
- **Satisfactorio (SA)**: Este nivel incluye las respuestas que cumplen con las expectativas. Para el caso de las historias de usuario, son consideradas satisfactorias si alcanza un resultado consistente en entre dos y tres iteraciones. Para el caso de las entrevistas, se consideran satisfactorias si son completas y bien estructuradas, aunque no profundice en el dominio específico.

4. Resultados

En la tabla 1 se muestran los resultados obtenidos en la generación de entrevistas con un solo enunciado, al ser evaluadas con los criterios especificados previamente.

Tabla 1: Resultados de entrevistas con un solo enunciado

Iteración	GPT3.5	GPT4o	GPT+WFM	GEM1.0	GEM1.5	Mistral	AI21
Primera	AC	AC	SA	AC	SA	MA	AC

En la tabla 2 se ilustran los resultados de la generación de entrevistas con dos prompts independientes.

Es de destacar que, en líneas generales, se obtuvieron cuerpos de entrevista con preguntas similares a las esperadas. Se incorporan temas que en un contexto real son de interés, como por ejemplo preguntas sobre el público objetivo o alcance geográfico en donde se planea aplicar el sistema, la posible integración con sistemas existentes o cuestiones de manejo de datos o seguridad. También se destaca la generación de una estructura de entrevista que pueda servir como una guía, teniendo en cuenta una presentación, preguntas introductorias y luego separadas en temáticas, una conclución y cierre.

Tabla 2: Resultados de entrevistas con dos enunciados

Iteración	GPT3.5	GPT4o	GPT+WFM	GEM1.0	GEM1.5	Mistral	AI21
Primera	IN	SA	AC	SA	SA	MA	MA
Segunda	-	AC	AC	IN	SA	MA	MA

A pesar de ser una aclaración realizada en el prompt, la generación de ambas entrevistas en una sola iteración puede provocar la mezcla de los requerimientos que pertenecerían a la segunda entrevista con los de la primera, formando un conjunto de preguntas que no cumplen con las condiciones planteadas. Por ese motivo se consideró que hacer una división entre la primera y segunda entrevista en dos iteraciones distintas podría resultar en un proceso más óptimo y preciso para los objetivos establecidos.

En algunos casos se destaca la creatividad de los modelos al proponer nuevas ideas o funciones para el sistema planteado en los enunciados, como así también en la asistencia a la demostración de interés en el dominio o problema.

En la tabla 3 se pueden visualizar los resultados de la generación historias de usuario donde solo se pide el título de cada una. Se destaca la cantidad de historias de usuario generadas para este caso, que coincide con el número de requerimientos funcionales esperado (alrededor de setenta).

Tabla 3: Resultados de los títulos de historias de usuario

Iteración	GPT3.5	GPT4o	GPT+WFM	GEM1.0	GEM1.5	Mistral	AI21
Primera	MA	AC	AC	MA	AC	AC	MA
Segunda	MA	SA	MA	AC	SA	AC	AC
Tercera	AC	SA	MA	-	MA	IN	MA

En la tabla 4 se pueden ver los resultados de solicitar la generación de historias de usuario con título, reglas de negocio y escenarios de criterios de aceptación. Se observa la generación de una menor cantidad de historias de usuario, en

comparación al enunciado donde solo se solicitó solo el título de cada una. Las historias de usuario no son lo suficientemente atómicas en algunos casos. A modo de ejemplo, el registro de un nuevo cliente incluye un criterio de aceptación para el inicio de sesión, lo que se considera una historia de usuario independiente.

Al ejecutar las terceras iteraciones se observan repeticiones de historias de usuario mencionadas previamente, como así también funcionalidades que excedían al alcance del problema descrito en las entrevistas.

Las celdas completadas con un guión indican que la prueba no se ejecutó en esa iteración debido a un resultado inválido previo.

Tabla 4: Resultados historias de usuario con estructura detallada

Iteración	**GPT3.5**	**GPT4o**	**GPT+WFM**	**GEM1.0**	**GEM1.5**	**Mistral**	**AI21**
Primera	MA	AC	AC	IN	AC	SA	MA
Segunda	MA	MA	SA	IN	AC	MA	MA
Tercera	MA	-	MA	-	AC	-	MA

5. Conclusiones

En este trabajo se definió una secuencia de etapas que abarca el proceso de desarrollo de software y se analizó el uso de modelos de lenguaje de gran tamaño como una herramienta que permite asistir a algunas de las etapas de dicho proceso.

Se acotó el estudio a las etapas de elicitación, análisis y especificación de requerimientos y se definió, y en particular a la generación de entrevistas y de historias de usuario como artefactos para cada etapa respectivamente.

Se desarrollaron cuatro metodologías de evaluación, dos enfocadas en la generación de entrevistas, y las otras dos en la generación de historias de usuario. La evaluación se ejecutó sobre siete modelos de lenguaje de gran tamaño y se compararon los resultados obtenidos en cada caso.

A partir de la comparativa de los resultados obtenidos de las entrevistas se puede concluir que, en líneas generales, los modelos de lenguaje de gran tamaño presentan resultados aceptables, ajustados a los criterios que se definieron, con la excepción de dos mediciones específicas de GPT 3.5 y Gemini 1.0.

Con respecto a los resultados obtenidos en la generación de historias de usuario, se destaca como conclusión la variedad y la creatividad de las historias desarrolladas en todos los modelos de lenguaje. Los mejores resultados se pudieron ver a la hora de generar una estructura más reducida, por lo que se puede pensar a futuro en un desarrollo de historias de usuario que cumplan un formato más acotado y que posteriormente se puedan seguir usando los modelos de lenguaje para ser detallados al mencionar requerimientos más específicos.

El uso de herramientas de inteligencia artificial generativa está en pleno apogeo en diversas industrias y el desarrollo de software no está exento de los cambios en las actividades que se plantean. Sin embargo, a partir de los análisis anteriores es importante reforzar la necesidad y la importancia de que los profesionales

involucrados en la ingeniería de software tengan una mirada crítica sobre los resultados arrojados por los modelos de lenguaje de gran tamaño para poder evaluar y seleccionar los resultados que mejor se adapten al contexto del problema que se busca resolver, y descartar aquellos que no tengan la calidad o la pertinencia adecuada.

Referencias

[1] N. Jha y R. Popli. «Artificial Intelligence For Software Testing-Perspectives And Practices». En: *2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT)* (2021), págs. 377-382. DOI: 10.1109/CCICT53244.2021.00075.

[2] Sahithi Devalla y Manas Kumar Yogix. «Building Trust in AI -A Simplified Guide to Ensure Software Quality». En: *Journal of Soft Computing Paradigm* (2023). DOI: 10.36548/jscp.2023.3.001.

[3] Dr. V. Poornima y Mrs. R. Priya. «A Comprehensive Study of Various Software Process Models in Software Development». En: *International Journal of Advanced Research in Science, Communication and Technology* (2022). DOI: 10.48175/ijarsct-5939.

[4] Shravan Pargaonkar. «A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering». En: *International Journal of Scientific and Research Publications* (2023). DOI: 10.29322/ijsrp.13.08.2023.p14015.

[5] Shreta Sharma y S. K. Pandey. «Integrating AI Techniques In SDLC: Design Phase Perspective». En: *Proceedings of the Third International Symposium on Women in Computing and Informatics*. WCI '15. Kochi, India: Association for Computing Machinery, 2015, págs. 383-387. ISBN: 9781450333610. DOI: 10.1145/2791405.2791418. URL: https://doi.org/10.1145/2791405.2791418.

[6] E. Whitten y L. Bentley. *Systems Analysis and Design Methods*. 7.ª ed. McGraw-Hill, 2007.

[7] Indra Kharisma Raharjana, Daniel Siahaan y Chastine Fatichah. «User Stories and Natural Language Processing: A Systematic Literature Review». En: *IEEE Access* 9 (2021), págs. 53811-53826. DOI: 10.1109/ACCESS.2021.3070606.

[8] F. Dalpiaz y S. Brinkkemper. «Agile Requirements Engineering: From User Stories to Software Architectures». En: *2021 IEEE 29th International Requirements Engineering Conference (RE)* (2021), págs. 504-505. DOI: 10.1109/RE51729.2021.00076.

[9] P. Abrahamsson et al. «Predicting Development Effort from User Stories». En: *2011 International Symposium on Empirical Software Engineering and Measurement* (2011), págs. 400-403. DOI: 10.1109/ESEM.2011.58.

[10] A. Bulajic, S. Sambasivam y R. Stojic. «An Effective Development Environment Setup for System and Application Software». En: *Issues in Informing Science and Information Technology* 2013 (2013), págs. 37-66. DOI: 10.28945/1795.

[11] Japneet Singh, Sanjib Kumar Sahu y Amit Prakash Singh. «Implementing Test Automation Framework Using Model-Based Testing Approach». En: *Intelligent Computing and Information and Communication*. Ed. por Subhash Bhalla et al. Singapore: Springer Singapore, 2018, págs. 695-704. ISBN: 978-981-10-7245-1.

[12] Google Gemini Team. *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. https://storage.googleapis.com/deepmind-media/gemini/gemini_v1_5_report.pdf. [En línea; accedido el 19/07/2024]. 2024.

[13] Kristian Petkoff Bankoff et al. «Quality 4.0 in Software Engineering: Incorporating Scaled Agile Insights to a Framework Proposal». En: *Computer Science – CACIC 2023*. Ed. por Patricia Pesado, Walter Panessi y Juan Manuel Fernández. Cham: Springer Nature Switzerland, 2024, págs. 179-194. ISBN: 978-3-031-62245-8.

[14] Rajesh H. Kulkarni y Palacholla Padmanabham. «Integration of artificial intelligence activities in software development processes and measuring effectiveness of integration». En: *IET Software* 11.1 (2017), págs. 18-26. DOI: https://doi.org/10.1049/iet-sen.2016.0095. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-sen.2016.0095. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-sen.2016.0095.

[15] Ignatios Papadopoulos et al. «Enhancing the student's logical thinking with Gherkin language». En: *2017 IEEE Global Engineering Education Conference (EDUCON)*. IEEE. 2017, págs. 1543-1547. DOI: 10.1109/EDUCON.2017.7943054.