Uso de la inteligencia artificial para el análisis y reporte automático de defectos de software

Jonathan Castro Ramírez
Facultad de TI
ULACIT
San José, Costa Rica
jcastror417@ulacit.ed.cr

Glen Fernández Gamboa
Facultad de TI
ULACIT
San José, Costa Rica
gfernandezg782@ulacit.ed.cr

Luis M. Corrales González
Facultad de TI
ULACIT
San José, Costa Rica
Lcorralesg880@ulacit.ed.cr

Netskar Aguilera Manzanarez
Facultad de TI
ULACIT
San José, Costa Rica
naguileram909@ulacit.ed.cr

Julio Córdoba Retana
Facultad de TI
ULACIT
San José, Costa Rica
jcordoba022@ulacit.ed.cr

***Resumen*— La inteligencia artificial, sumada a la integración de otras tecnologías emergentes de la industria 4.0, han permitido a los ingenieros e informáticos prepararse para enfrentar los retos que el mundo tecnológico actual presenta, creando estándares de aseguramiento de calidad y nuevas experiencias a partir del uso de nuevas herramientas de desarrollo.**

El presente artículo muestra los resultados iniciales obtenidos de una investigación acerca del uso de la inteligencia artificial para el análisis y reporte automático de defectos de software encontrados a través de las pruebas de regresión, haciendo uso de tecnologías como aprendizaje automatizado, procesos robóticos automatizados, inteligencia artificial e interfaces de programación de aplicaciones, con el fin de desarrollar una aplicación capaz de encontrar la causa raíz de los defectos del software, con el análisis de los archivos de resultados obtenidos de las pruebas automatizadas.

***Palabras clave*— Inteligencia artificial, aprendizaje automatizado, automatización de procesos robóticos, pruebas de regresión, análisis de defectos, reporte automático de defectos.**

***Abstract*— Artificial intelligence coupled with the integration of other emerging technologies of industry 4.0 have enabled engineers and computer scientists to prepare to face the challenges that today's technological world presents, creating quality assurance standards and new experiences based on the use of new development tools.**

The present article shows the initial results gotten from the investigation about the use of artificial intelligence to auto analyze and report software defects found in regression tests, using technologies such as machine learning, robotic process automation, artificial intelligence and application programming interfaces in order to develop an application capable of finding the root cause of the software defects analyzing the log files gotten from the auto tests.

***Keywords*— Artificial intelligence, machine learning, robotic process automation, regression testing, analysis of defects, automatic defect reporting.**

I. Introducción

La inteligencia artificial no es una tecnología alejada de todo lo que nos rodea en el vivir de cada día: robots, computadoras que entienden nuestro lenguaje, carros autónomos, entre otros, son parte del futuro que veíamos en la televisión y que ahora lo tenemos en nuestro presente [1].

El uso de la inteligencia artificial en el campo del aseguramiento de la calidad del software ha traído grandes beneficios a las empresas, ayudando en tareas como predicción y ubicación de defectos, aprendizaje de secuencias, clonación de códigos y muchas funciones más. El correcto uso de estas tecnologías ha provocado que las empresas de software disminuyan el esfuerzo, tiempo y costos requeridos tanto para el descubrimiento de defectos como para solucionarlos [2].

En los últimos años, se ha detectado un gran deseo de los ingenieros y empresas por automatizar todo lo que sea posible y por utilizar las últimas tendencias en tecnología, lo cual deja de lado los efectos secundarios que esto pueda tener [3] y permite que la inteligencia artificial sea aplicada en el campo del descubrimiento de defectos, analizándolos de manera autónoma.

En los comienzos del desarrollo de software, las personas escribían el código de los programas y encontraban defectos según lo iban utilizando mientras se desarrollaban. Después con el fin de mejorar la calidad del software, hubo especialización de personas en las pruebas del software, con el fin de encontrar la mayor cantidad de defectos mientras se desarrollaba y no cuando ya el software estaba expuesto en los ambientes de producción. Conforme los programas crecían y la cantidad de pruebas crecían, se convirtió en algo inmanejable

correr pruebas manualmente, ya que algunas eran demasiado complejas, grandes y difíciles de mantener para su ejecución constante, lo cual también aumentaba el riesgo de error debido al factor humano que podía equivocarse al ejecutar una prueba. Según lo anterior, se comenzó con la automatización de las pruebas, para minimizar el error humano y ejecutar grandes cantidades de pruebas de software con muy poca interacción humana.

Al poco tiempo, nacieron los ambientes de regresión, los cuales automáticamente ejecutan cada una de las pruebas y esto lleva a las empresas a mantener una base de datos de pruebas actualizada, lo cual permite a las organizaciones brindar productos fiables y robustos, garantizando que su desempeño sea óptimo por medio de la eficiencia y eficacia de los casos de pruebas ejecutados sobre el software creado.

Cuando un cambio en el software hace que las pruebas automatizadas fallen, los ingenieros a cargo de las pruebas automatizadas y desarrolladores involucrados en el cambio realizado deben analizar el fallo y encontrar la causa raíz de este. Si el error se presenta en los cambios del código de la aplicación informática, son los desarrolladores los encargados de analizar el defecto, detectar donde se encuentra y solucionarlo. Si el error se encuentra en la prueba automatizada, el ingeniero a cargo de la automatización debe analizar por qué fue que la prueba falló, si fue que la aplicación se ejecutó más lento de lo esperado por la prueba automatizada, si hay archivos faltantes, si el tiempo de ejecución esperado fue excedido o si hubo algún suceso inesperado que provocara que la prueba fallara y una vez detectada la causa raíz, proceder a la solución.

El proceso de análisis de fallos de las aplicaciones informáticas o pruebas automatizadas es muy demandante en términos de tiempo, ya que las pruebas deben ser ejecutadas varias veces hasta que los ingenieros encuentren la causa raíz del defecto o error. Por lo tanto, la presente investigación busca cómo implementar el uso de diferentes tecnologías para realizar la tarea de análisis de defectos en las aplicaciones informáticas de manera autónoma, es decir, sin la ayuda de los seres humanos.

Como parte de esta investigación se debe explorar si existen empresas que, en la actualidad, hagan uso de procesos automatizados de análisis de defectos en aplicaciones informáticas, tanto a nivel de desarrollo como de pruebas de calidad, utilizando tecnologías como Artificial Intelligence (AI: Inteligencia Artificial), Machine Learning (ML: Aprendizaje de máquina) y Robotic Process Automation (RPA: Automatización Robótica de Procesos), ya que al momento de escribir este documento, no se encontró tal información. También, se debe investigar si el uso de estas tecnologías provee la forma más rápida, transparente, inteligente y eficiente de realizar estos análisis para que, una vez identificada la causa raíz del problema, haciendo uso de un RPA, se reporte automáticamente el defecto en herramientas de administración de tareas tales como Jira.

Uno de los problemas principales que se podrían encontrar a la hora de realizar la investigación, es que existen múltiples frameworks (marcos de trabajo) para automatizar pruebas, lo cual permite que surja la duda si existe algún estándar para el reporte de los defectos en los distintos frameworks de automatización, ya que si no existe, las organizaciones que quisieran implementar AI, ML y RPA para el análisis automático de pruebas fallidas, deben establecer su propio estándar y con esto desarrollar y entrenar sus propios procesos para realizar el debido análisis de los defectos.

Si el autoanálisis y reporte de defectos se logra implementar, las compañías que hagan uso de estos procesos se podrían beneficiar, disminuyendo las tareas que requerían talento humano, ejecutadas ahora de manera autónoma, liberando así a los ingenieros de estas tareas para permitirles trabajar en nuevos procesos dentro de las organizaciones.

Algunos aspectos que se deben tomar en cuenta, es que si los modelos de aprendizaje de máquina no son entrenados correctamente o con la cantidad suficiente de muestras, se podrían presentar múltiples escenarios no deseados, tales como el reporte de defectos no existentes; es decir, falsos positivos, como el escape de defectos en los ambientes de desarrollo y pruebas, los cuales pueden ser encontrados en los ambientes de producción y causarle pérdidas a los usuarios por los defectos no reportados en los debidos ambientes. Las pérdidas también puede ser para las empresas desarrolladoras de las aplicaciones informáticas al exponer las deficiencias en sus procesos de pruebas, causar gastos y afectar la reputación al no entregar productos de alta calidad a los usuarios finales, provocando que las personas que utilizan estas aplicaciones pierdan la confianza en el producto entregado.

II. METODOLOGÍA

El siguiente apartado se centrará en describir el tipo de investigación a utilizar para así obtener un resultado de manera clara y precisa, basándose en el estudio de investigación de Roberto Hernández Sampieri. La investigación se define como "*un conjunto de procesos sistemáticos y empíricos que se aplican al estudio de un fenómeno o problema*" [4].

Dentro de los métodos y técnicas de investigación es importante considerar que se obtienen nuevos conocimientos en una realidad social y tecnológica para diagnosticar una necesidad, o bien, plantear, de una mejor manera, cómo resolver un problema en específico.

Es por eso que se ha planteado como modelo de investigación un enfoque cualitativo-explicativo, dicho proceso ayuda a establecer el propósito central del trabajo el cual es cómo la AI, junto a otras tecnologías de la industria 4.0, ayudan en el proceso de análisis y reportes automáticos de defectos en el área de desarrollo de software.

Para una mejor búsqueda de datos e información se definen palabras clave como "Artificial Intelligence", "Machine Learning", "RPA", "Frameworks", "Software Development", "Industry 4.0", además, se limita la búsqueda a artículos completos y a un rango de años entre el 2019 y el 2021. Se propone estudiar información tanto en español como en inglés, para enriquecer la investigación, todo esto con el objetivo de obtener información completa, reciente y fiable.

La investigación explicativa ayudará a determinar el o los motivos por los cuales más organizaciones integran en sus procesos o unidades de negocio este tipo de tecnologías, así como conocer las diversas variables relevantes que influyen en el problema investigativo. Asimismo, permitirá establecer explicaciones y conclusiones que ayudará a proporcionar información que pueda servir como guía.

Para obtener buenos resultados, tanto la capacidad del investigador, como la capacidad de síntesis, deben de ser adecuados. Es por eso que, la recopilación, análisis y validación de datos e información ayudará a plantear un estudio de casos que permita el análisis de estos para determinar qué otras compañías o sujetos han presentado un mismo problema y lo han tratado de forma más eficiente.

Durante la recopilación de información se plantea usar información encontrada en las distintas herramientas especializadas para la búsqueda de datos y artículos, como lo son EBSCO, Google Académico, entre otras.

La recopilación de datos y el análisis de la información puede suceder secuencialmente o en paralelo, el propósito es explorar los datos, organizarlos, describir conceptos, temas, patrones y prácticas, con la finalidad de interpretarlos, desarrollarlos y exponerlos para llegar a una explicación sustancial.

III. Marco Teórico

La implementación de la inteligencia artificial se basa en una característica medible y el impacto que esta tiene con respecto a las necesidades del negocio. Un claro ejemplo de esto son los proyectos que hacen uso de la inteligencia artificial para automatizar una gran parte de los procesos manuales, brindando una solución a los usuarios encargados de estas tareas, sin embargo, también existen proyectos de inteligencia artificial que necesitan de constantes justificaciones para mantenerlos en ejecución cuando podrían ser remplazados por otros proyectos que realizan más tareas de una mejor manera [5].

Antes de continuar, se define como inteligencia artificial, aquella simulación de la inteligencia humana por medio de máquinas, en otras palabras, es la disciplina que intenta crear sistemas capaces de aprender y razonar cómo los seres humanos lo hacen, además, es un término que abarca otros conceptos como Aprendizaje Automático, Procesamiento de Lenguaje Natural, Big Data, entre otros.

La automatización robótica de procesos o RPA por sus siglas en inglés, busca reducir la intervención humana principalmente interactuando con aplicaciones de alto nivel, que son las capas de interfaz gráfica, en pocas palabras, es un tipo de software que emula la interacción real que tendría un ser humano con las aplicaciones informáticas convencionales.

Por ultimo, el aprendizaje automático, es una aplicación de inteligencia artificial que proporciona a los sistemas la capacidad de aprender y mejorar automáticamente por medio de patrones de datos; una vez detectados estos patrones el ML ajusta las acciones del programa por medio de algoritmos. [17]

Para entender como funciona la automatización de procesos haciendo uso de la inteligencia artificial, se deben conocer los cuatro principios de la automatización inteligente: pensar y aprender, visión, lenguaje, y ejecución.

El primer principio se basa en el pensar y aprender donde se trabaja con tecnologías como el aprendizaje de máquina, visualización de la información y el Big Data Management (manejo de grandes cantidades de datos).

En el segundo principio, se encuentra la visión, la cual permite a las computadoras analizar y procesar actividades tales como el Optical Character Recognition (OCR: Reconocimiento Óptico de Caracteres), Intelligent Character Recognition (ICR: Reconocimiento Inteligente de Caracteres), el análisis de video e imágenes y, por último, el análisis y procesamiento de datos biométricos.

En el tercer principio, lenguaje, se cuentan con tecnologías tales como los chatbots (Conversaciones con robots inteligentes) y el Unstructured Information Management (UIM: Manejo de información no estructurada). También con actividades como el análisis de sentimientos y habla [6].

Por último, el principio de la ejecución, se basa en tecnologías Low-Code (No-Código), lo cual permite a las personas, sin conocimiento de desarrollo de software, crear sus propias aplicaciones o automatizar procesos repetitivos haciendo uso de tecnologías como los RPA [7].

En la actualidad, existen aspectos importantes en el ciclo de vida del desarrollo de software, una de ellas es el aseguramiento de la calidad de este. La AI ha impactado cada vez más en la gestión de los procesos de una empresa ofreciendo oportunidades para el desarrollo de software y aplicaciones comerciales.

Hoy en día, la AI se ha abierto camino en el proceso de creación de software agregando inteligencia al entorno de desarrollo con el que trabajan los equipos, con el fin de desarrollar nuevas funcionalidades inteligentes dentro de las aplicaciones empresariales. Se puede catalogar la AI como la automatización de la capacidad intelectual humana, ya que puede realizar funciones que normalmente son asociadas al cerebro humano, por lo que si se suma la AI al desarrollo de software se puede lograr innegables beneficios e innovación, tales como los robots autónomos, procesos automatizados y análisis de software [8].

La AI aplicada a las pruebas de software está evolucionando de tal manera que está alterando el panorama de pruebas llevando este tipo de procesos a otro nivel donde ya no se dependerá tanto del razonamiento humano, esta se está enfocando en facilitar el ciclo de vida del desarrollo de software, mediante la aplicación del razonamiento, resolución de problemas, y aprendizaje automático para la resolución de tareas tediosas que tienen que ver con el desarrollo de pruebas de software y sus limitaciones. Por medio de AI se busca revisar los cambios de código recientes, la cobertura del código y otras métricas de pruebas de software, para así decidir qué pruebas ejecutar y cuando ejecutarlas, ayudando a que los desarrolladores se enfoquen en otras tareas de más valor en beneficio de la empresa u organización [9].

Se proyecta que el uso de la AI, en optimización de escenarios de prueba, seguirá creciendo en cada faceta de la tecnología creativa debido al creciente número de aplicaciones que usamos a diario [10].

Entre los ejemplos actuales donde AI está tomando un rol relevante en la automatización de pruebas de software se pueden mencionar aplicaciones web y móviles (pruebas funcionales), interfaces de usuario, por medio de pruebas visuales (donde se aprovecha el aprendizaje basado en imágenes y las comparaciones de pantalla para probar la apariencia de una aplicación), ubicación de elementos de interfaz de usuario (UI: User Interface) y selectores de elementos de autocorrección cuando cambia la interfaz de usuario, adicionalmente, se están utilizando agentes autónomos e inteligentes, denominados "bots de prueba" para automatizar actividades como el descubrimiento de aplicaciones, el modelado, la generación de pruebas y la detección de fallas.

Ahora bien, enfocándose en el aprendizaje automático, este está ayudando a la AI mediante la aplicación de algoritmos que permiten que las herramientas de automatización mejoren por medio de la recopilación y análisis de datos producidos por las pruebas. Por esta razón, el aprendizaje automático resulta valioso, debido a que la recopilación de datos es relevante en la toma de decisiones, afectando positivamente el alcance de los objetivos en los proyectos.

El fin de la AI aplicada a pruebas de software está claro: ayudar a los equipos de la organización a desarrollar y probar su código de manera eficiente y efectiva para crear software de mayor calidad a mayor velocidad. Gracias al uso de tecnologías como RPA, AI, ML, el RPA Automation Anywhere (AA: Automatización en todo lugar) que provee herramientas para la automatización de tareas repetitivas. Cabe destacar que AA es capaz de conectarse con JIRA, la herramienta administrativa para el registro, organización, reporte y revisión de tareas informáticas.

Sabiendo que AA es capaz de automatizar procesos dentro de JIRA y que ha sido utilizado según estudios realizados en Jaypee University of Information Technology para la extracción de reportes de manera autónoma, se puede considerar AA como un RPA, capaz de realizar el reporte automático de defectos en el software, esto es posible debido a que AA utiliza los casos de uso, es decir, cada uno de los pasos que los usuarios realizan para reportar los defectos, y con esto lograr que la herramienta, por sí sola, sea capaz de ejecutar el reporte automático [11].

Investigando cómo podría ser posible el autoanálisis de defectos en el desarrollo de aplicaciones informáticas o de pruebas automatizadas, se descubrió que también se pueden encontrar vulnerabilidades cibernéticas en los ambientes de desarrollo y pruebas haciendo uso de algoritmos supervisados y semi-supervisados de ML. Nunes et al.[12] toman información de la internet oscura e internet profunda para asegurarse de que sus códigos no son vulnerables a lo mencionado en los distintos sitios de internet, al igual que otras compañías y estudios que demuestran que pueden predecir vulnerabilidades en sus códigos, haciendo uso de lo publicado en las redes sociales, Twitter y Reddit, a través de tecnologías como el ML [12]. Esto da una guía de cómo se puede también, a través del ML y AI, automatizar el proceso de predicción de vulnerabilidades, las cuales pueden verse como defectos en el código y una vez predicha la vulnerabilidad, hacer uso del RPA para realizar los reportes necesarios en las herramientas de administración de tareas como JIRA.

Una hipótesis que se puede presentar en este momento es que a través de AI y ML se puede realizar análisis de los logs (reportes de errores y defectos) del software o de las pruebas automatizadas con algoritmos supervisados, sean entrenados con antiguos logs para encontrar la causa raíz del defecto sin necesidad de la supervisión humana o sean lo suficientemente entrenados para realizar la actividad [13].

Dentro de los distintos frameworks de automatización se pueden destacar dos muy reconocidos a nivel mundial: Cypress y Selenium, los cuales son utilizados para automatizar pruebas de aplicaciones web. Selenium es el framework más utilizado, ya que fue creado en el 2005, sin embargo, carece de facilidades para manejar las páginas web dinámicas lo que hace que el rendimiento de las pruebas sea reducido y que las automatizaciones sean intermitentes en sus resultados. Como parte de estas limitaciones, Cypress llega a simplificar las pruebas asíncronas, ejecutándose una vez que todo lo necesario es cargado en la aplicación web. Según [14] Cypress trabajando en conjunto de Test Driven Development (TDD: Pruebas basadas en el desarrollo) y el patrón de diseño Page Object Model (POM: Modelo de objeto de página), provee una alta rentabilidad, aumentando la cantidad de funciones cubiertas por las pruebas, mejorando su ejecución en un 21% y reduciendo la complejidad de automatización en un 31%.

Pensando en cómo los distintos frameworks de automatización pueden ser utilizados para generar reportes que sean útiles para el ML y AI, hay estudios que demuestran que un RPA puede ser también utilizado para la automatización de pruebas de software, con el uso de una interfaz gráfica de usuario [15] y, con esto, se obtendrían los logs de ejecución de las pruebas automatizadas estandarizados según el usado por el RPA y los problemas de como maneja cada framework de automatización su propio log sería solucionado, ya que se pasaría de tener múltiples formatos a uno solo, sin embargo, esto requeriría una inversión tanto económica como de tiempo ya que todas las pruebas automatizadas deberían de ser convertidas para que el RPA sea el ejecutor.

La etapa de pruebas de software puede llegar a requerir una cantidad de tiempo y esfuerzo incluso mayor al desarrollo de software, por ello es de vital importancia para el alcance de objetivos de la empresa, ya que garantiza la robustez de este.

Para el desarrollo de software no solo se necesita saber cómo desarrollar la aplicación, ya que se encuentra todo el ciclo de vida del desarrollo de software, por ejemplo, la obtención de requerimientos, el diseño de la aplicación, la creación del software funcionando, pruebas y muchos pasos más. Sin embargo, la automatización de los análisis de las pruebas fallidas permitiría a las empresas reducir la intervención humana realizando estas tareas con tecnologías tales como AI, ML y RPA [16].

La AI y RPA trabajando de manera conjunta permiten a las empresas automatizar procesos repetitivos, lo cual, sería de

gran utilidad al realizar de manera autónoma el reporte de los defectos en herramientas de administración de tareas tales como JIRA, evitando que personas encargadas de esta tarea puedan especializarse en distintas áreas, ya que la AI junto con ML potencia los RPA acelerando la toma de decisiones y reduciendo el riesgo de error [18]

Cabe destacar que hay diferencias relevantes entre las tecnologías iniciando porque RPA se centra más en el "hacer", mientras que AI y ML se preocupan más por el "pensar y aprender", otra diferencia es que RPA se enfoca en gran medida en procesos, mientras que AI y ML se enfocan en la calidad de los datos, y cómo estos ayudan al buen desarrollo de aplicaciones. RPA usa entradas estructuradas y lógicas, mientras que AI usa entradas no estructuradas y desarrolla su propia lógica. [19]

A través de los procesos realizados manualmente por los usuarios para el reporte de defectos, se pueden crear casos de uso que permitirá al RPA automatizar esta tarea. Mientras que la AI junto con ML analizan la causa raíz de los defectos, el RPA se puede encargar del reporte automático de los defectos, analizando primeramente si hay algún defecto similar reportado en la herramienta de administración de tareas para evitar la creación de tiquetes duplicados [17][20].

Debido a que los usos o funciones del software son diferentes dentro de cada compañía, y lo que es importante para una empresa no es importante para otra, cada organización deberá automatizar sus procesos basados en sus propios manuales de uso o políticas establecidas dentro de la organización.

Con la creciente competencia entre empresas no necesariamente del sector informático, sino cualquier empresa que le dé importancia a su departamento de tecnologías de información, la demanda por desarrolladores de software se mantiene en constante crecimiento, lo cual se traduce en crecientes líneas de código que necesitan ser analizadas y probadas antes de considerarse óptimas para su entrega. Según una encuesta a desarrolladores en 2019 [21] se logró determinar que los mismos gastan el 35% de su tiempo realizando pruebas a su código.

Con la inminente y forzada actualización tecnológica que trajo la pandemia, sumado a la falta y difícil contratación de personal calificado, las empresas se ven cada vez más forzadas a hacer más con el personal que poseen, esto conlleva el tema de esta investigación, buscar cómo automatizar aquellas tareas monótonas, como el análisis de defectos y pruebas en el software, por medio de tecnologías como la AI y ML y, de esta forma, lograr liberar tiempo de calidad para tareas que requieran más aporte humano por parte del personal.

Aunque podría parecer de sentido común, la implementación de la inteligencia artificial en el área de las pruebas de software lucha contra un gran cuello de botella y es la carencia de un oráculo, es decir, un mecanismo que define lo que está bien y lo que está mal. La inteligencia artificial es muy poco usada en la detección de errores debido a lo problemático de automatizar el oráculo, la única excepción a esta regla son las pruebas de regresión, en las cuales se pueden analizar y determinar los resultados esperados, basados en las versiones anteriores del sistema o software [22].

A pesar de los recientes y crecientes avances en la última década, sobre automatización de pruebas, y en los esfuerzos para realzar la relevancia de las pruebas de software en las conferencias de ingeniería de software, queda mucho trabajo por hacer, ya que los desarrollos se tornan cada vez más complejos agregando las mezclas de códigos de terceros, desarrollos en diferentes lenguajes de programación, la ejecución de servicios en diferentes plataformas y la inclusión de las plataformas móviles y en la nube. [23] Todo esto sumado, nos lleva a un panorama complejo incluso con la implementación de IA y ML en el proceso de automatización, para lograr esa unificación, compatibilidad y profundo análisis que conlleva un acercamiento en el proceso de las pruebas de software.

En el ámbito de las pruebas de software se encuentran tres distintas formas de realizar dichas pruebas: White-Box Testing (pruebas de caja blanca), Black-Box Testing (pruebas de caja negra) y Gray-Box Testing (pruebas de caja gris).

Las pruebas de caja blanca consisten en el acceso directo al código por parte del ingeniero de aseguramiento de la calidad, lo cual le permite realizar sus pruebas basado en el código que encuentra, sin embargo, si aparece algún defecto en el futuro, deberá analizar el error y esto implica un mayor esfuerzo de conocimiento por parte del analista. Dentro de las pruebas de caja blanca se encuentran las pruebas unitarias.

Para las pruebas de caja negra, no se requiere el acceso al código y muchas veces se ve limitado a las pruebas de interfaz de usuario o interfaz entre aplicaciones.

Por último, para complementar las pruebas de caja negra junto con las de caja blanca, existen las pruebas de caja gris, la cual combina los métodos anteriores y permite mejores resultados al dar la capacidad de planear las pruebas basado en el uso de la aplicación informática con respecto al código de ésta [24].

Como se ha mencionado anteriormente, el proceso de pruebas en el software es, en muchas formas, más laborioso que la construcción del software en sí misma, por cada trozo de código, se deben escribir pruebas para medir el rendimiento, la funcionalidad e incluso la seguridad [21].

Con una industria de 12 billones de dólares, y con una mayoría de trabajo manual y de contratación externa; el desarrollo y mantenimiento de software se ha convertido en un pilar para empresas como los bancos, que han empezado a adoptar la inteligencia artificial para automatizar las pruebas unitarias en sus procesos. Según la información recopilada por los autores de "Accelerate" [21] las organizaciones que aplican cambios en su código más frecuentemente poseen menos porcentaje de fallos y son 170 veces más rápidas en recuperarse de errores en su software que otras compañías.

Finalmente, todo este ahorro en tiempo significa más tiempo de calidad que los desarrolladores y el resto del equipo pueden destinar a labores más desafiantes intelectualmente, y cualquier otra tarea destinada a satisfacer las necesidades de los clientes o las organizaciones Sin embargo, hay que destacar

que, a pesar de los muchos beneficios que pueda traer consigo la automatización de las pruebas de software, si dicha automatización se basa en procesos mal diseñados, esto solo potenciará los errores y traerá caos y confusión en el equipo.

Por último y más importante, toda esta nueva ola de pensamiento e implementación permitirá al equipo estar libre de tareas repetitivas y tediosas, y dará un respiro y libertad creativa a otras ideas más innovadoras para beneficio de las organizaciones que se arriesguen a invertir en estas tecnologías.

IV. Análisis de resultados

Basado en la información recopilada es posible identificar varios escenarios que demuestran que el autoanálisis y reporte automático de defectos de software es posible de realizarse, sin embargo, durante la ejecución de pruebas se descubrieron puntos negativos en el proceso que no se pudieron realizar debido a la complejidad de este para lograrlo, por lo que se brindará el concepto de cómo sería posible llevar a la realidad el autoanálisis y reporte automático de los defectos encontrados, a través de las pruebas automatizadas.

Utilizando Cypress como framework de automatización para pruebas de interfaz gráfica en entornos web y trabajando con los ejemplos provistos por Cypress Test Runner (ejecutor de pruebas basadas en cypress), se logró identificar que el framework por sí solo no crea ningún archivo de resultados con los errores obtenidos, si no que ejecuta todas las pruebas, imprime el error en consola y al final muestra un resumen de resultados con las pruebas que pasaron y fallaron. Otra forma de obtener la información del error sería revisando el panel de Cypress, donde se muestra el error en una página web.

Con los resultados obtenidos hasta aquí, de parte de las pruebas automatizadas, obtener un reporte de los fallos en las pruebas es complicado, ya que habría que remover aquellas líneas innecesarias de la consola y escribir un archivo con los errores mostrados o que cada prueba automatizada cree desde cero su propio log de errores, lo cual podría causar diferencias de formato en la escritura del archivo, ya que si no se establece un estándar, cada persona que trabaja automatizando las pruebas, podría tener su propio estándar, permitiendo que no exista manera fácil de que un algoritmo logré identificar algún patrón de cómo está acomodada la información del archivo por leer.

Como parte de este inconveniente, buscando cómo se puede obtener un log de errores de todas las pruebas automatizadas y respondiendo a la pregunta que buscaba si existía algún estándar para la creación de logs en los distintos frameworks de automatización, es posible encontrar en la plataforma GitHub una biblioteca dedicada a Cypress que satisface estas necesidades, llamada Cypress-Failed-Log [25].

Cypress-Failed-Log logra identificar cuando hubo un error en la ejecución de la prueba y crea un archivo en formato JSON (JavaScript Object Notation) para cada uno de los errores con información tal como el nombre del archivo donde se encuentra el caso de prueba, el nombre de la prueba, la descripción del error, el URL (localizador uniforme de recursos) de la sección en prueba, el identificador o clase del objeto que se buscaba y el resultado esperado.

Una vez que se obtiene el archivo en formato JSON con la información del error y al saber que este viene ya estandarizado, este archivo servirá como entrada para ser manejado por el procesamiento de lenguaje natural (NLP) que es un campo de la AI y ML donde se logra la interacción entre humanos y computadoras [26]. Se utiliza este algoritmo ya que el error que se puede obtener en el archivo JSON viene especificado en un lenguaje entendible por humanos, por lo que se debe normalizar el texto removiendo todas las palabras que no son necesarias para permitirle a la computadora tener la información lo más sintetizada posible. Es importante aclarar que para realizar este procesamiento de datos se utilizó la biblioteca OpenNLP.

Lo primero que se debe realizar al estar trabajando con ML es entrenar el modelo, en este caso, el modelo de oraciones usando los logs generados con la biblioteca Cypress-Failed-Log detectando el texto presente en el archivo JSON y convirtiendo el texto en oraciones para luego ser tokenizadas, es decir, convertir la información sensible en información no sensible conocida como tokens [27].

Una vez la información fue tokenizada, se deben crear las entidades que serán utilizadas, en este caso, las entidades serían aquellas llaves que se encuentran en el JSON que contiene la información del error y asignar cada valor a su respectiva entidad, esto con el fin de poder dividir cada palabra según su categoría, ya sea que la palabra sea una verbo, conjunción, preposición o adjetivo. Cuando se tiene la división de palabras por categoría, estas serán analizadas para obtener aquellas que son realmente importantes para servir como dato de entrada hacia la aplicación de AI que se encargará de realizar el análisis del error y encontrar donde pudo haber sido originado.

Como parte de la investigación y realización de pruebas de concepto, no se logró encontrar una implementación de AI para realizar el auto análisis de los defectos encontrados, ya que el AI debería de ser lo suficientemente capaz para buscar tanto en los archivos del front-end (cliente), back-end (servidor) como en los archivos de la base de datos y encontrar dónde se origina el error o, por lo menos, dar sugerencias al usuario de donde puede estar el defecto.

Una alternativa para este problema sin solución por el momento sería basarse en las entregas de código realizadas, donde se tomaría como base el primer día en el que la prueba automatizada falló, y por medio de AI buscar en el repositorio todos los cambios de código que hubieron el día anterior, es decir, antes de crear el instalador para empezar con la ejecución de las regresiones y, una vez encontradas, analizar en todos los archivos donde hubo cambios, o qué puede estar relacionado con el error obtenido en la ejecución de la prueba automatizada.

Explicando a alto nivel cómo sería el proceso hasta este paso, se empezaría con la ejecución de la prueba automatizada; si esta falla, entonces se crea el archivo JSON que funcionará como entrada para el algoritmo de ML. El algoritmo de ML analizaría el error y generaría un archivo con la información más importante del error, el cual serviría como entrada para la

AI, la cual se encargará de buscar ya sea en el código de la aplicación o en el repositorio de esta donde pudo haberse introducido el error para realizar el autoanálisis y detectar la causa raíz del problema. Una vez obtenido el autoanálisis, un proceso aparte se encargará de la creación de un archivo con la información que será compartida en el caso de JIRA, obteniendo la información que es de relevancia para cada empresa. Un ejemplo de esta información podría ser el título del defecto, descripción del error, los pasos para reproducir el fallo, el ambiente en el que fue ejecutada la prueba y un video o capturas de pantalla mostrando el error (el vídeo y las capturas de pantalla son provistos por Cypress automáticamente al ejecutar las pruebas con el argumento "--record)".

Para reportar el error de manera automáticamente en JIRA existen tres alternativas: usar un RPA con integración a JIRA, utilizar un REST-API (Transferencia de estado representacional - Interfaz de programación de aplicaciones) o utilizar la integración de Cypress con JIRA para reportar defectos de software.

Para la primer alternativa que se basa en el uso de un RPA, funcionaría tomando el archivo creado con los detalles del defecto a reportar en JIRA, procesarlo y crear el defecto de manera autónoma, esto es posible gracias a investigaciones realizadas en la universidad de Jaypee University of Information Technology, que demuestran que se hacía uso de un RPA para extraer reportes [11], especificando una consulta de bases de datos del reporte que se quería obtener, para luego el RPA ejecutar la consulta definida, esperar los resultados, exportar la información obtenida y descargarla al servidor local, por lo tanto si el RPA era capaz de lograr extraer reportes basándose en la escritura de consultas de bases de datos, también el RPA sería capaz de automatizar el reporte de defectos.

La segunda alternativa se basa en el uso de la integración del panel de Cypress con JIRA, el cual deberá ser automatizado para seleccionar la prueba automatizada que falló, agregar los datos que provienen del archivo que contiene toda la información a agregar en el caso de JIRA y guardar el registro, de esta manera, dentro del panel de Cypress podría ser realizado el autoreporte de defectos y ahorrarse gran cantidad de dinero que pudo haberse utilizado para la implementación de un RPA que se encargará de reportar el defecto.

Y, por último, como tercera alternativa, el uso de un REST-API, el cual sería capaz de realizar la tarea del autoreporte de defectos a través de una tarea automatizada que estaría ejecutando los comandos necesarios para crear el defecto en JIRA. Para la creación de este defecto, se necesita ejecutar un comando de CURL (Comando URL) en el cual se le especifica el usuario y contraseña, el tipo de petición HTTP (Protocolo de transferencia de híper texto), un archivo en formato JSON, el tipo de contenido a enviar y el URL del servidor de JIRA de la siguiente forma "curl -D- -u <usuario>:<contraseña> -X POST –data <ruta del archivo JSON> -H "Content-Type: Application/json" <URL del servidor de Jira>/issue".

Esta tercera alternativa es la más fácil de implementar ya que lo único que necesita es que se le especifique donde se encuentra el archivo de formato JSON, ya que el usuario, contraseña y URL del servidor JIRA pueden estar definidos, por lo que solamente al actualizar la ruta del archivo, por medio de un archivo de formato batch se ejecutaria y crearía el defecto en la herramienta JIRA.

Como se puede observar, esta investigación provee el concepto de cómo podría implementarse el autoanálisis y reporte de defectos en JIRA, sin embargo, con la información y conocimiento adquirido hasta el momento, debido a las complejidades para la implementación de la inteligencia artificial para encontrar la causa raíz del defecto de software, no es posible detallar como la AI debería ser implementada para realizar este análisis. No obstante, para la obtención de los logs de errores de las pruebas automatizadas, el procesamiento de estos a través de ML y el reporte automático de los defectos sí pudo ser detallado y puede ser utilizado como guía para que ingenieros puedan basarse para crear sus propias aplicaciones que permitan realizar un auto análisis de defectos de software y automatizar el reporte de defectos dentro de la herramienta JIRA.

V. Conclusiones y Recomendaciones

Considerando la creciente demanda de desarrolladores de software, debido a la forzada actualización tecnológica que trajo consigo la pandemia del COVID-19 para las empresas, y entendiendo que esta gran cantidad de código a generarse necesita su debido proceso de pruebas que es igual o más exhaustivo que el proceso de desarrollo, se logra entender la importancia que trae consigo la aplicación de AI en los procesos de reporte y análisis de pruebas automatizadas.

Se puede decir, con certeza, que aquellas empresas que apliquen o logren aplicar a futuro cualquier aproximación de automatización en sus procesos de pruebas al software, lograrán una menor taza de fallos y una proporción de 170 veces más agilidad para recuperarse de cualquier fallo en sus sistemas que cualquier otra organización. Además, de obtener tiempo de calidad para sus colaboradores en tareas que sean más desafiantes intelectual o creativamente, y que den paso a mejorar las funciones del software para beneficio de la organización o los clientes finales.

Un gran obstáculo en la implementación de la IA en las pruebas de software es, definitivamente, la creación del oráculo y lo difícil que sería adaptarlo a un flujo automatizado guiado de la mano de la IA y el ML. Sin dejar de lado, lo complicado que es para una organización, la adaptación y aprendizaje a tecnologías tan complejas como lo son la IA, el ML y los frameworks automatizados para pruebas.

Cabe destacar que la calidad de las pruebas, el análisis de estas y los resultados finales irán de la mano con el entrenamiento a los modelos de inteligencia artificial, lo que conlleva una gran cantidad de datos de prueba y a la aplicación de diferentes escenarios.

Como recomendación a aquellos lectores interesados en la automatización de las pruebas de software, para mejorar sus tiempos de desarrollo o recuperación de caídas, ya sea desde un punto de vista del desarrollo o del administrativo, no está de más destacar que, a pesar de las limitantes o complejo que

pueda parecer el tema, este debe ser un factor de impulso para la implementación de dichas tecnologías en sus organizaciones, y no en un desaliente en el proceso.

Pensar en los beneficios al final del camino y no en lo difícil de sus pasos, siempre ayudará en el proceso de adopción de cualquier tecnología novedosa, asimismo, más allá de esta investigación se pueden encontrar u observar otros desarrollos o soporte en esta área.

Tomando en cuenta los datos y resultados obtenidos de las pruebas de concepto realizadas, se puede concluir que sí se puede llevar a cabo la implementación de AI para el análisis de defectos encontrados a través de las pruebas de regresión, sin embargo, debido a limitaciones en el conocimiento de la tecnología, no fue posible implementar tal funcionamiento. Sin embargo, se puede observar que con la ayuda de ML y los algoritmos de procesamiento de lenguaje natural, se puede clasificar la información en distintas categorías con el fin de servir como guía para que la AI tenga un punto de partida para hallar la causa raíz del defecto.

Al no poderse implementar el uso de AI para el análisis de defectos de software, se recomienda a aquellas personas que deseen implementar este tipo de tecnología para encontrar la causa raíz de los defectos, que investiguen y adapten la tecnología a sus necesidades con el fin de garantizar que los resultados obtenidos sean verdaderamente los deseados, ya que se podría caer en la obtención de datos falsos que podrían ser catalogados como verdaderos y solucionar defectos inexistentes, mientras que los defectos reales siguen ahí presentes en el código de la aplicación.

También, como parte de la investigación, se logró conocer que no existe un formato estándar para reportar los errores o defectos que se encuentran en las pruebas automatizadas, es decir, al generar los logs de resultados de las pruebas, estos son generados con un formato ya establecido por cada empresa, de hecho, se conoció que hay empresas utilizando Cypress que ni siquiera usan los logs, ya que corren todas las pruebas basados en los resultados, utilizando el panel de ejecución de Cypress como log de resultados.

Por lo tanto, aún cuando no existe ningún formato estándar para el reporte de estos defectos, se encontró que hay bibliotecas disponibles a través de la red que pueden ser utilizadas para generar los logs con un formato estándar; una vez generados los reportes con un mismo formato, estos pueden ser utilizados para que ML los analicé y categorice cada defecto en distintas capas de la aplicación donde se puede encontrar el error.

Se recomienda considerablemente utilizar un estándar para la generación de los logs, ya sea definido dentro de cada empresa o utilizando bibliotecas disponibles en internet para poder alimentar y usar como base de entrenamiento a la aplicación de ML que categorizará cada defecto. Es importante recalcar que las aplicaciones de ML deben ser modificadas según el estándar que se utilice para obtener los logs de Cypress y es necesario tener gran cantidad de logs para entrenar correctamente la aplicación de ML; idealmente, un 70% de los datos debería ser utilizado para entrenar la aplicación y el otro 30% para validar los resultados obtenidos.

La inteligencia artificial y ML pueden ayudar a disminuir considerablemente el tiempo de análisis de defectos encontrados por medio de su automatización, con esto se logra que los ingenieros asignados a estas tareas (que previamente se hacían manualmente) puedan trabajar en tareas paralelas y que exijan mayor especialización mientras que la computadora hace el trabajo de análisis y reporte de errores por medio de modelos entrenados para estos fines, un ejemplo de esto es que los ingenieros podrían concentrarse en el diseño de casos de prueba que podrían ser más detallados y, por consiguiente, más robustos, adicionalmente, la planificación de pruebas adicionales cuando sean necesarias para mejorar la precisión del diagnóstico, en muchas ocasiones por los tiempos de entrega de proyectos ese tipo de planificación no se hace de forma detallada y esto hace que en muchas ocasiones las pruebas fallen y que los productos no se entreguen en el tiempo acordado, esto provoca pérdidas económicas significativas a la organización.

La inteligencia artificial y ML, a través de la automatización de pruebas de software, facilita la identificación de posibles defectos, su corrección a tiempo y, sobre todo, su predicción para así evitarlos y anticipar posibles problemas, los cuales se pueden gestionar por adelantado, de esta forma es posible alcanzar niveles de máxima calidad en los productos de software, por ello se recomienda que los modelos de ML utilizados sean modelos probados y entrenados de forma correcta para así asegurarnos de que la automatización sea ejecutada satisfactoriamente, minimizando el margen de error de la prueba.

Con base en el análisis de aspectos teóricos y prácticos de esta investigación podemos concluir que la AI, ML y RPA proveen herramientas para la verificación automática de errores que fueron detectados en escenarios de pruebas diseñados previamente, esto ayudaría de gran manera a la disminución de tiempos no solo de análisis, sino de resolución de problemas de software, ya que no habrían duplicación de tiquetes en herramientas como JIRA, los desarrolladores no trabajarían en un mismo tiquete al mismo tiempo, lo cual provocaría que la comunicación se vuelva más efectiva. Ante este escenario, se recomienda que el manejo de tiquetes tenga un estándar definido para que la inteligencia artificial o RPA, por medio de algoritmos, sea más eficaz en la comparación de estos.

En la actualidad es necesario estar preparado en cuanto a tecnologías de automatización de procesos de análisis y reporte de defectos de software, la automatización cognitiva y la inteligencia artificial están a pocos años de ser parte de la vida cotidiana de los negocios, por ello, como sugerencia, las organizaciones deberían de ir modificando los roles actuales e implementar nuevos roles (recopilación de datos es clave para el proceso de entrenamiento de modelos para toma de decisiones) que se adapten a las necesidades de este tipo de tecnologías.

Hay muchas rutas de investigación, pero el objetivo final de la AI (en conjunto con otras tecnologías) en el campo de la automatización de análisis y reporte de defectos de software es claro, ayudar a los equipos a desarrollar y probar su código de manera más eficiente y efectiva, reduciendo tiempos de análisis

y reporte de errores con el fin de para crear software de mayor calidad a mayor velocidad.

Con el resurgimiento de las nuevas tecnologías en la industria 4.0 con el 'Big Data' como materia prima, la AI ha obtenido el potencial de proporcionar información, mejorar prácticas y la toma de decisiones, y de ser una pieza importante en el desarrollo de software.

La AI es una tecnología en desarrollo y no solo crea oportunidades de negocio, sino que también representa desafíos, esto implica identificar y probar diferentes tipos de modelos de aprendizaje automático. Es importante tener en cuenta que las personas implicadas en el desarrollo trabajen en conjunto con otras partes interesadas de la organización para aplicar las técnicas correctas y/o establecer estándares.

Se recomienda siempre estar a la vanguardia de las nuevas herramientas tecnológicas, y es que la versatilidad que presentan algunas de ellas pueden ayudar a abarcar más el poder puro de la AI. Dichas herramientas complementan el desarrollo de software y funcionan como una capa de AI en general, conectándose a la perfección con los propios programas.

El uso de la AI ahorra tiempo a los desarrolladores, esta tecnología ofrece la oportunidad de trabajar de manera más eficiente y productiva. Sin embargo, es importante el manejo cuidadoso de los datos utilizados para el análisis de resultados y el desarrollo del algoritmo para el aprendizaje automático, la AI es tan buena como los datos de entrenamiento que utiliza.

Se puede decir que la AI puede llegar a necesitar de un mantenimiento, y es que su funcionamiento es tan bueno o malo como la calidad de los datos con los que está diseñado el algoritmo de aprendizaje. Si los datos para el análisis y resultados no se actualizan el algoritmo no continuará aprendiendo, por lo que desarrollar un algoritmo que además de analizar la información también tome decisiones basándose en la recopilación de nuevos datos para su uso, creará una nueva instancia de aprendizaje logrando así una mejora en la AI.

Es por eso que se recomienda crear ambientes de aprendizaje supervisado, ayudando a la AI a aprender, sabiendo de antemano cuál es el resultado correcto esperado y cuáles son las relaciones entre los datos en el análisis de resultados.

VI. Trabajos futuros

En el presente documento se dio a conocer como podría implementarse a alto nivel todo el ecosistema para poder autoanalizar y reportar defectos encontrados a través de las pruebas de regresión, sin embargo, no fue posible implementar la forma en la que la inteligencia artificial funcionaria. Por lo tanto, para trabajos futuros se recomienda implementar una inteligencia artificial que se encargue de buscar todos aquellos archivos que fueron cargados al repositorio el día antes de que corrieran las pruebas de regresión y, a partir de ahí, empezar a analizar cada uno de los archivos para encontrar basado en la clasificación del defecto según el ML, donde puede estar la causa raíz de este.

Referencias

[1] Janet Finlay and Alan Dix, An Introduction to Artificial Intelligence, 2020. https://www.taylorfrancis.com/books/mono/10.1201/9781003072485/introduction-artificial-intelligence-janet-finlay-alan-dix

[2] Sumit Mahapatra and Subhankar Mishra, Usage of Machine Learning in Software Testing, 2020. https://link.springer.com/chapter/10.1007/978-3-030-38006-9_3

[3] Daron Acemoglu and Pascual Restropo, Atificial Intelligence, automation and work, 2019. https://www.degruyter.com/document/doi/10.7208/9780226613475-010/html

[4] R. Hernández Sampieri, C. Fernández Collado y M. P. Baptista Lucio, Metodología de la investigación, 6ta ed., McGrow-Hill, Mexico D.F., 2014.

[5] Andrew Moore, When AI Becomes an Everyday Technology, 2019. https://www.investkl.gov.my/assets/multimediaMS/file/H05003-PDF-ENG_FINAL.PD

[6] Pascal Bornet, Ian Barkin and Jochen Wirtz, Intelligent Automation. Welcome to the world of hyperautomation, pp.10, 2020.

[7] Apurvanand Sahay, Arsene Indamutsa, Davide Di Ruscio and Alfonso Pierantonio, Supporting the understanding and comparison of low-code development platforms, 2020. https://ieeexplore.ieee.org/abstract/document/9226356

[8] Cliff Saran, The Art of developing happy customers: Artificial Intelligence is playing a growing role in modern software development, 2020. https://www.computerweekly.com/feature/The-art-of-developing-happy-customers

[9] Testing Experts, An introduction to Artificial Intelligence, 2021, https://www.testingxperts.com/blog/ai-bot#AI%20bots%20and%20Software%20Testing

[10] Rahul Jain, 12 Important Software Testing Trends for 2021 You Ned to Know, 2021, https://www.lambdatest.com/blog/12-software-testing-trends-2021/

[11] Siddhika Seth, Vidyavathi Bagalkoti and M.J. NIgam, JIRA Report Extraction, 2019, http://ir.juit.ac.in/123456789/22908

[12] Sameera Horawalavithana, Abhishek Bhattacharjee, Renhao Liu, Nazim Choudhury, Lawrence O.Hall and Adriana Lamnitchi, Mentions of Security Vulnerabilities on Reddit, Twitter and Github, 2019, https://dl.acm.org/doi/epdf/10.1145/3350546.3352519

[13] Amanpreet Singh, Narina Thakur, Aakanksha Sharma, A review of supervised machine learning algorithms, 2016, https://ieeexplore.ieee.org/abstract/document/7724478

[14] Mobaraya, Fatini, and Shahid Ali. "Technical Analysis of Selenium and Cypress as Functional Automation Framework for Modern Web Application Testing." 9th International Conference on Computer Science, 2019, https://www.academia.edu/43153928/TECHNICAL_ANALYSIS_OF_SELENIUM_AND_CYPRESS_AS_FUNCTIONAL_AUTOMATION_FRAMEWORK_FOR_MODERN_WEB_APPLICATION_TESTING?from=cover_page

[15] Valentyna Panasyuk, Yuliia Trufanova, Oksana Kushnir, Nataliia Yatskiv, Iryna Voytyuk and Solomiya Yatskiv, Improved Method of Software Automation testing Based on the Robotic Process Automation Technology, 2019, https://ieeexplore.ieee.org/abstract/document/8780038

[16] J Mauro, Intelligent Automation. AuraQuantic, 2021, https://www.auraquantic.com/what-is-intelligent-automation/

[17] AI Expert, What is the definition of Machine Learning?, 2020, https://www.expert.ai/blog/machine-learning-definition/

[18] UI Path, AI and RPA Whitepaper – AI Center. UIPath, 2021, https://www.uipath.com/resources/automation-whitepapers/bringing-power-ai-rpa-together-with-ai-center

[19] NICE, AI and RPA: What's the Difference and which is Best for Your Organization? NICE Systems, 2021, https://www.nice.com/rpa/rpa-guide/rpa-ai-and-rpa-whats-the-difference-and-which-is-best-for-your-organization/

[20] CFB Bots, The difference between Robotic Process Automation and Artificial Intelligence, 2018, https://cfb-bots.medium.com/the-difference-between-robotic-process-automation-and-artificial-intelligence-4a71b4834788

[21] Matthew Lodge, Software Testing is Tedious. AI Can Help, 2021, https://hbr.org/2021/02/software-testing-is-tedious-ai-can-help

[22] Rui Lima, António Miguel Rosado da Cruz and Jorge Ribeiro, Artificial Intelligence Applied to Software Testing: A Literature Review, pp.1-6, https://web.a.ebscohost.com/ehost/detail/detail?vid=0&sid=867ee5d2-05a6-4783-9baa-c7d4e962e21c%40sdc-v-sessmgr03&bdata=Jmxhbmc9ZXMmc2l0ZT1laG9zdC1saXZl#

[23] Jenny Li, Andreas Ulrich, Xiaoying Bai and Antonia Bertolino, Advances in test automation for software with special focus on artificial intelligence and machine learning, 2020, https://web.a.ebscohost.com/ehost/detail/detail?vid=0&sid=9664f52f-dcdb-4d50-9867-ddd5c2280b26%40sdc-v-sessmgr01&bdata=Jmxhbmc9ZXMmc2l0ZT1laG9zdC1saXZl

[24] Mohd Ehmer Khan and Farmeena Khan, A Comparative Study of White Box, Black Box and Grey Box Testing Techniques, 2012, https://thesai.org/Publications/ViewPaper?Volume=3&Issue=6&Code=IJACSA&SerialNo=3

[25] Gleb Bahmutov, Rhys Arkins, Lyudmil Latinov and Kristie Howard, Cypress failed log, 2021, https://github.com/bahmutov/cypress-failed-log

[26] Maria Razno, Machine Learning text classification model with NLP approach, 2019, http://ena.lp.edu.ua:8080/handle/ntb/45487

[27] Sina Ahmadi, A tokenization System for the Kurdish Language, 2020, https://aclanthology.org/2020.vardia