UNIVERSIDAD NACIONAL DE LOJA

ÁREA DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES

CARRERA DE INGENIERÍA EN SISTEMAS

TÍTULO:

"Aplicación de Algoritmos Genéticos en la Ingeniería del Software: Revisión Sistemática del Estado del Arte"

TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS

AUTOR:

Milton Darío Quizhpe Villavicencio

DIRECTOR:

Ing. Pablo Fernando Ordoñez Ordoñez, Mg.Sc

LOJA – ECUADOR

2017

CERTIFICACIÓN DEL DIRECTOR.

Ing. Pablo F. Ordoñez Ordoñez, Mg.Sc. en calidad de director del trabajo de titulación designado por disposición de la coordinación de la carrera de Ingeniería en Sistemas, certifico que el Egresado Milton Darío Quizhpe Villavicencio, ha culminado el trabajo de titulación, con el tema "Aplicación de Algoritmos Genéticos en la Ingeniería del Software: Revisión Sistemática del Estado del Arte", quien ha cumplido con todos los requisitos legales exigidos por los que se aprueba la misma. Es todo cuanto puedo decir en honor a la verdad, facultando al interesado hacer uso de la presente, así como también se autoriza la presentación para la evaluación por parte del jurado respectivo.

Loja, 10 de noviembre de 2016.

Atentamente,

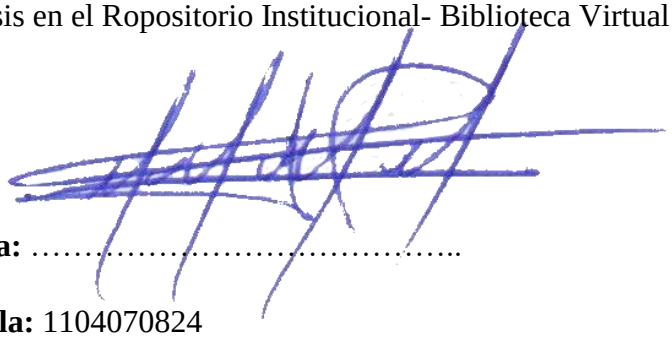
Ing. Pablo Fernando. Ordoñez Ordoñez, Mg.Sc.

DIRECTOR DE TESIS

AUTORÍA

Yo **MILTON DARÍO QUIZHPE VILLAVICENCIO**, declaro ser autor del presente trabajo de tesis y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido de la misma.

Adicionalmente acepto y autorizo a la Universidad Nacional de Loja, la publicación de mi tesis en el Ropositorio Institucional- Biblioteca Virtual



Firma: ……..

Cédula: 1104070824

Loja, 2 de febrero de 2017

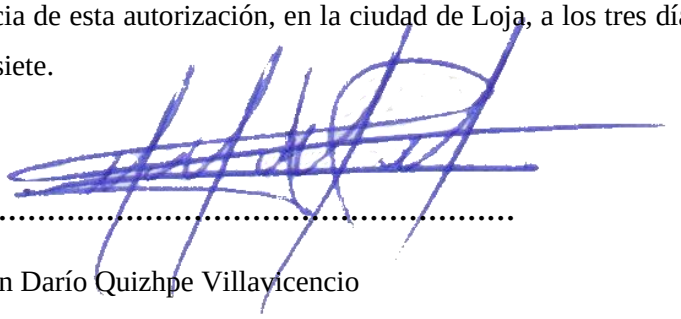
CARTA DE AUTORIZACIÓN DE TESISI POR PARTE DE EL AUTOR, PARA LA CONSULTA, REPRODUCCIÓN PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO

Yo **MILTON DARÍO QUIZHPE VILLAVICENCIO**, declaro ser autor de la tesis titulada: **APLICACIÓN DE ALGORITMOS GENÉTICOS EN LA INGENIERÍA DEL SOFTWARE: REVISIÓN SISTEMÁTICA DEL ESTADO DEL ARTE**, como requisito para obtener el grado de **INGENIERO EN SISTEMAS**; autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional:

Los usuarios pueden consultar el contenido de este trabajo en el RDI; en las redes de información del país y del exterior; con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia de la tesis que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los tres días del mes de febrero del dos mil diecisiete.



Firma:……

Autor: Milton Darío Quizhpe Villavicencio

Cédula: 1104070824

Dirección: Loja (Av. Eugenio Espejo y Adolfo Valarezo Esquina)

Correo Electrónico: milton_qui25@hotmail.com

Teléfono: 072544313 **Celular:** 0989175441

DATOS COMPLEMENTARIOS

Director de Tesis: Ing Pablo Fernando Ordoñez Ordoñez, Mg.Sc

Tribunal DE Grado: Ing. Edwin Rene Guamán Quinche, Mg. Sc.

Ing. Manuel Alberto Córdova Neira, Mg. Sc.

Ing. Franco Hernán Salcedo López, Mg. Adm

DEDICATORIA

Primeramente a Dios por haberme dado la vida y siempre darme la fortaleza espiritual para concluir este trabajo.

Dedico este trabajo a mis padres, y hermanos quienes son el pilar fundamental y los testigos de cada lucha y esfuerzo por ser mejor, a ellos mi vida, mi alegría y la culminación de este trabajo y lo que este representa.

AGRADECIMIENTO

A Dios, por estar conmigo en cada paso de mi vida, fortaleciendo mi corazón y espíritu, y por haber puesto en mi camino a aquellas personas que constantemente me acompañado y apoyado durante todo el periodo de estudio.

A mi familia que se constituyen en el pilar fundamental de mi vida personal y profesional, apoyándome en mis estudios de forma irrestricta y continua.

Un sincero agradecimiento a la Universidad Nacional de Loja , al Área de Energía las Industrias y los Recursos Naturales no Renovables y en especial a la Carrera de Ingeniería en Sistemas y sus catedráticos quienes han impartido sus conocimientos y valores durante todo mi periodo de estudio.

De manera muy especial quiero agradecer al Ing. Pablo Ordoñez Ordoñez director de mi trabajo de titulación quien aportó su conocimiento, criterios, comentarios y dedico su valioso tiempo para la culminación de este trabajo.

A todos mi respeto y agradecimiento.

INDICE DE CONTENIDO

CONTENIDO **PÁG.**

INDICE DE FIGURAS

INDICE DE TABLAS

1. TÍTULO

“APLICACIÓN DE ALGORITMOS GENÉTICOS EN LA INGENIERÍA DEL SOFTWARE: REVISIÓN SISTEMÁTICA DEL ESTADO DEL ARTE”

2. RESUMEN

La Ingeniería del software nació de la necesidad de establecer una metodología adecuada y eficiente para el desarrollo del software, al no emplear métodos apropiados, el software tenía gran cantidad de errores. Hoy en día el software ha evolucionado drásticamente gracias a que es considerada una disciplina, es decir que tiene sus propios principios y exigencias para tener soluciones más estructuradas con una debida planeación, desarrollo y culminación. Los algoritmos genéticos presentan una alternativa para solucionar problemas que se presentan durante varias etapas en el desarrollo de la Ingeniería del software. En la presente investigación se realiza una Revisión sistemática del estado del arte de la Aplicación de los algoritmos genéticos en la Ingeniería del software y así poder responder las interrogantes planteadas al inicio de la misma, ¿qué problema soluciona en la ingeniera del software los Algoritmos Genéticos?, ¿qué Aplicación tienen los Algoritmos Genéticos en la Ingeniería del Software? y ¿qué tecnología utilizan los algoritmos genéticos para su ejecución?

Los artículos en los que se basa este proyecto aportarán en el estudio de las diferentes aplicaciones de los algoritmos genéticos en las áreas de la Ingeniería del software. Los resultados que se muestran están basados en el análisis de 20 documentos los mismos que se obtuvieron de un total de 127, luego de pasar por un protocolo de Revisión y selección ya que son los que se encuentran más acorde con el tema de investigación.

SUMMARY

Software engineering arose from the necessity to establish an efficient and adequate methodology for the software development, by not using the appropriate methods, the software had lots of errors, and nowadays software has drastically developed and is considered a discipline, therefore we can say that it has its own principals and needs in order to have more structured solutions with proper planning, development and culmination, The genetic algorithms have an alternative to solve problems that arise during various stages during the development of software engineering. In the following investigation a systematic review of the stage of the art of the application of genetic algorithms in software engineering was performed and made it possible to answer all the interrogates made at the beginning. What problem is solved in software engineering?, with genetic algorithms, what applications genetic algorithms have in Software engineering? and what technology genetic algorithms use for their performance?.

The articles that bring this project is based on the study of the different applications of genetic algorithms in the areas of software engineering. The results shown are based on the analysis of the same 20 documents that were obtained from a total of 127, after passing through a protocol review and selection as they are those who are more in line with the research topic.

3. INTRODUCCIÓN

Ingeniería de Software es una aplicación del enfoque sistemático y disciplinado para el diseño, desarrollo, operación y mantenimiento de software[1], la detección de vulnerabilidades de software es un paso crítico para asegurar la calidad y la seguridad del software [2]Sin embargo, las pruebas de software es una tarea que consume tiempo y costo, se consume casi el 50% de los recursos que se proporcionan para el desarrollo total del software del sistema [3]–[5]. Las pruebas de software de forma automatizada son mejor que las pruebas manuales. Sin embargo, muy pocas herramientas basadas en algoritmos genéticos que ayuden en la generación de datos de prueba están disponibles comercialmente en la actualidad [1].

Los algoritmos genéticos (GA) se forman a partir de los algoritmos evolutivos concebidos por John Holland en los Estados Unidos que fueron conocidos durante finales de los sesenta [6].

Esta investigación se centrará en el punto de vista teórico de como los algoritmos genéticos puede ser aplicados en los problemas que tiene la ingeniería de software

Esta Revisión Sistemática está basada en el protocolo propuesto por Kitchenham [7], [8], y tiene como finalidad satisfacer las dudas en torno a las siguientes preguntas:

- ¿Qué problema soluciona en la ingeniera del software los Algoritmos Genéticos?
- ¿Qué aplicación tienen los Algoritmos Genéticos en la Ingeniería del Software?
- ¿Qué tecnología utilizan los algoritmos genéticos para su ejecución?

En la Revisión de la Literatura se elaboró cinco capítulos que ayudaron a sustentar los conocimientos de esta área de estudio. En la sección de Materiales y Métodos se describió todas las herramientas utilizadas tanto de hardware, software y recursos de oficina, luego se desarrolló el método (Revisión Sistemática) para la selección de los estudios primarios. A continuación, en la sección de Resultados se presentó todos los datos relevantes obtenidos del análisis de los artículos seleccionados. En la sección Discusión se expresó un análisis de la Revisión Sistemática en base a los resultados, y además, se manifestó como se cumplió con los objetivos, en la sección Conclusiones se expresó las ideas más relevantes que se rescató luego de la revisión. Finalmente la sección Recomendaciones se planteó aspectos a considerar para el desarrollo de futuros Trabajos.

4. REVISIÓN DE LITERATURA

4.1. Evolución de la ingeniería del software con la inteligencia artificial

4.1.1. Qué es la ingeniería del software.

Es una disciplina que comprende todos los aspectos de la producción de software desde los requerimientos del sistema hasta el mantenimiento del mismo. En esta definición, existen dos frases clave:

Disciplina de la ingeniería: Aplican teorías, métodos y herramientas donde sean convenientes, pero se utilizan de forma selectiva y tratando de descubrir soluciones a los problemas aun cuando no existan teorías y métodos aplicables para resolverlos. Los ingenieros saben que deben trabajar con restricciones financieras y organizacionales, por lo que buscan soluciones tomando en cuenta estas restricciones.

Todos los aspectos de producción de software: La ingeniería del software tiene actividades tales como la gestión de proyectos de software, el desarrollo de herramientas, métodos y teorías de apoyo a la producción de software. No sólo comprende los procesos técnicos del desarrollo de software[9].

El desarrollo informal es apropiado para el desarrollo de sistemas basados en Web, los cuales requieren una mezcla de técnicas de software y de diseño gráfico. En general, los ingenieros de software adoptan un enfoque sistemático y organizado en su trabajo, ya que es la forma más efectiva de producir software de alta calidad [10].

4.1.2. Qué es software.

Las personas asocian el término software con los programas de computadora. El software no son sólo programas, sino también todos los documentos asociados y la configuración de datos que se necesitan para hacer que estos programas operen de manera correcta. Por lo general un sistema de software consiste en diversos programas independientes. Archivos de configuración que se utilizan para ejecutar estos programas, un sistema de documentación que describe la estructura del sistema, la documentación para el usuario que explica cómo utilizar el sistema y sitios web que permitan a los usuarios descargar la información de productos recientes. Existen dos tipos de productos de software:

Productos genéricos: Estos sistemas son producidos por una organización de desarrollo y que se venden al mercado abierto a cualquier cliente que le sea posible comprar los

ejemplos de este tipo de producto son tales como bases de datos, procesadores de texto, paquetes de dibujo y herramientas de gestión de proyectos.

Productos personalizados (o hechos a medida): Estos Sistemas son requeridos por un cliente en particular. El ingeniero de software desarrolla este producto específicamente para ese cliente. Ejemplos de este tipo de software son los, sistemas desarrollados para llevar a cabo procesos de negocios específicos y sistemas de control del tráfico aéreo[9].

4.1.3. Producto Software.

Los ingenieros del software son los que que diseñan y construyen el producto de cualquier tamaño y arquitectura.

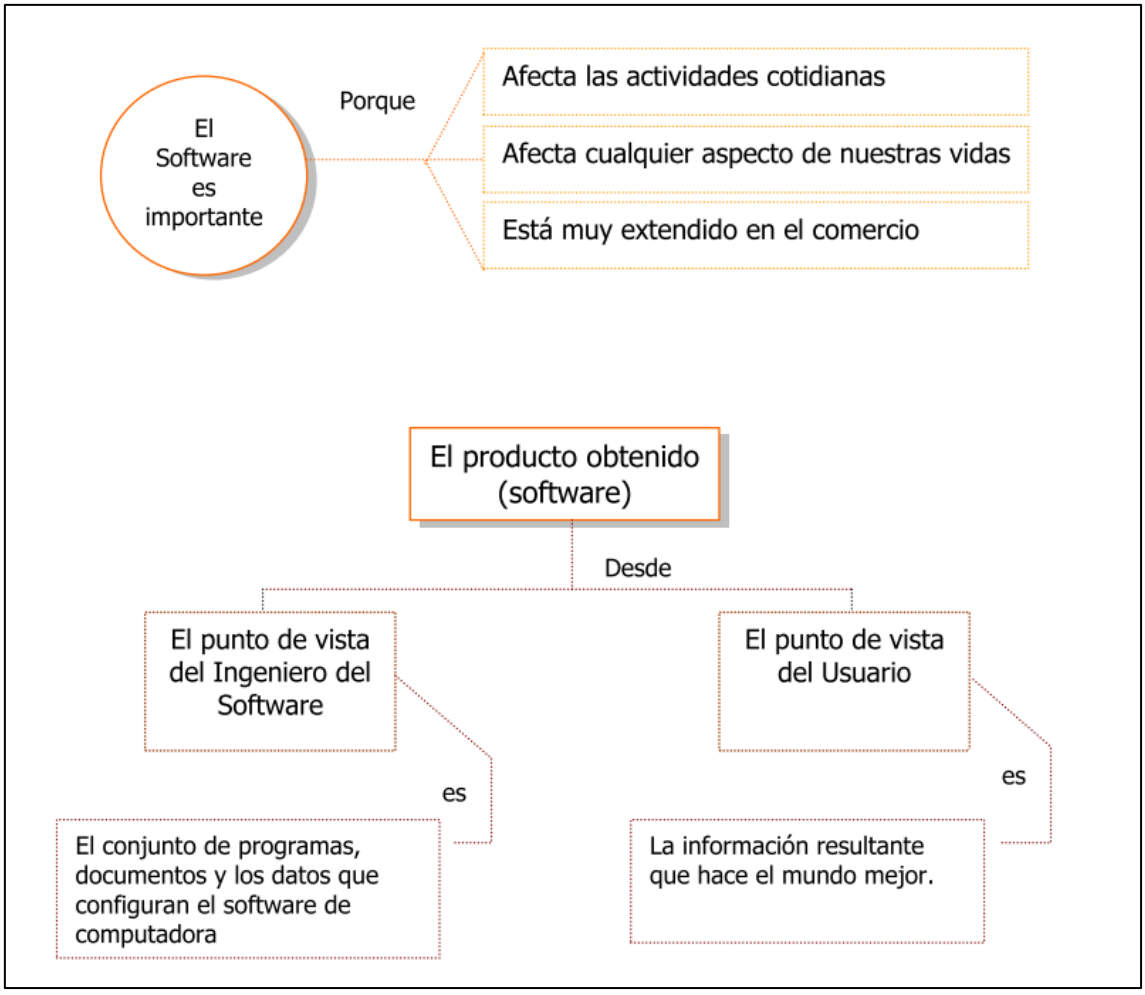


Figura 1: Producto de Software [10]

Pago por Uso: El pago del cliente varía en función del uso que realiza y del servicio contratado en la nube, es decir, que la facturación es basada en el consumo.

Abstracción: Esto se consigue con la virtualización, con lo que el cliente no necesita de personal dedicado al mantenimiento de la infraestructura. Las actualizaciones, pruebas y demás tareas asociadas quedan del lado del proveedor.

Escalabilidad: Son nuevos servicios ofrecidas al cliente, todo esto sin la necesidad de nuevos contratos o penalizaciones.

Multiusuario: Nos permite tener varios usuarios y compartir recursos, con la finalidad de evitar la subutilización de recursos informáticos.

Autoservicio bajo demanda: Los usuarios pueden acceder de manera automática a medida que sea necesario a las capacidades de computación en la nube, es decir el cliente puede añadir o quitar recursos sin interacción del proveedor.

Acceso sin restricción: Permite que los usuarios puedan acceder a los servicios contratados en cualquier momento en cualquier lugar y desde cualquier dispositivo que disponga una conexión a Internet[10].

4.1.4. Evolución del Software

Primera era

En los primeros años el software estaba en su infancia, era un añadido, había métodos limitados para la programación, no existía documentación, el software era diseñado empíricamente.

Segunda era

Aparece la multiprogramación y los sistemas multiusuario establecimiento del software como producto para ser comercializado, se empezó a distribuir software para grandes computadoras y minicomputadores y el mantenimiento de software comenzó a absorber recursos en una gran medida.

Tercera era

Se comenzó a la planificación en el proceso del desarrollo de software con lo cual se realizaron sistemas informáticos más complejos debido a la incorporación de inteligencia y funciones.

Cuarta era.

La industria del software es la cuna de la economía ya que se implementa las aplicaciones con inteligencia artificial y redes neuronales, las tecnologías orientadas a objetos con técnicas de cuarta generación para el desarrollo de software, la programación de realidad virtual y sistemas multimedia[11].

4.1.5. Que es la inteligencia artificial.

Si bien la IA es un campo joven, es heredera de diversas ideas, puntos de vistas y técnicas de otras disciplinas. Durante más de 2000 años la filosofía ha trabajado sobre diversas teorías del razonamiento y aprendizaje. La matemática ha desarrollado en varios siglos teorías formales relacionadas con la lógica, la probabilidad, la toma de decisión y con modelos matemáticos de computación. La psicología ofrece herramientas que permiten la investigación de la mente humana. La lingüística brinda teorías sobre la estructura y significado del lenguaje. Por último, en la ciencia e ingeniería de la computación las que proveen las herramientas y el soporte que permiten que la IA sea realidad[12].

La Inteligencia Artificial es la parte de la Ciencia que se ocupa del diseño de sistemas de computación inteligentes, es decir, sistemas que exhiben las características que asociamos a la inteligencia en el comportamiento humano que se refiere a la comprensión del lenguaje, el aprendizaje, el razonamiento, la resolución de problemas[13].

4.1.6. Programas de IA.

Los programas de IA exhiben cierto comportamiento inteligente fruto de la aplicación hábil de heurísticas en sentido amplio, entendiendo como heurística un tipo de conocimiento difícilmente formalizable, fruto de la experiencia y que se establece implícitamente para tratar de encontrar respuestas más o menos correctas, pero siempre validas, a un problema concreto. La utilización de conocimiento heurístico no garantiza encontrar soluciones óptimas, pero si permite garantizar el hallazgo de soluciones aceptables, si existen, a través de los denominados procesos inferenciales.

Inferencia es el proceso que permite la comprensión de un significado en función de cierta información relacionada. Su idea está ligada a los procesos de razonamiento, que frecuentemente exigen la realización de varias inferencias para lograr establecer conclusiones válidas[14].

4.1.7. Sistemas basados en conocimiento

El siguiente nivel es el de los sistemas basados en conocimiento, en los que los conocimientos del dominio concreto y las estructuras de control que se utilizan para manipularlo se encuentran físicamente separados, lo que va a requerir la definición e implementación de arquitecturas diferentes a las que estamos habituados, y en las que unos y otras puedan ser desarrollados independientemente entre sı, de forma que una misma estructura de control pueda ser utilizada en muchas bases de conocimiento diferentes y viceversa[14].

4.1.8. Resolución de Problemas

No podemos decir que algo o alguien exhibe comportamiento inteligente si no explota de manera eficaz y eficiente un conjunto mínimo de conocimientos. Decimos que un sistema es eficaz cuando es capaz de resolver correctamente un problema, y decimos que es eficiente cuando, además de comportarse eficazmente, optimiza los recursos disponibles.

Una arquitectura, natural o artificial, bien definida y estructurada pero vacía, no puede utilizarse para resolver problemas mientras no incorpora procedimientos de resolución y conocimientos propios del dominio de los problemas planteados. Está claro que no todos los problemas son iguales, por los que los tipos de conocimiento necesarios (las técnicas para abordarlos) van a ser diferentes también. En general, el tipo de problema planteado condiciona la técnica de resolución a emplear. No obstante, la decisión de utilizar una técnica de IA o una técnica convencional de programación puede ser también cuestión de planteamiento. Como norma general, el empleo de técnicas de IA debe permitir la construcción de programas que:

Capten generalizaciones, de forma que cada situación individual que se produzca no tenga que ser representada de forma separada, sino que todas aquellas situaciones que compartan propiedades deben ser agrupadas.

Hagan explıcito su conocimiento (que generen explicaciones en lenguaje natural), al objeto de facilitar su comprensión.

Puedan actualizarse continuamente, de forma que sea factible modificar el conocimiento sin tener que manipular ni alterar todo el programa.

Puedan ser empleados en muchas situaciones, aun cuando las respuestas que generen sean parcialmente correctas o imprecisas[14].

4.1.9. Aplicaciones de la Inteligencia Artificial

Los esfuerzos de la IA se clasifican según varias categorías. Mientras la investigación y el desarrollo como Robótica y Visión Artificial, se relacionan con el hardware y software, la investigación y el desarrollo en otras áreas sólo se relacionan con el software.

- **Sistemas de visión:** Incluyen equipos y software que les permite a las computadoras capturar, almacenar y manipular imágenes visuales y fotografías. Los sistemas de visión se pueden usar junto con robots para darles "visión" a estas máquinas y que pueda tomar decisiones con base a lo que ve y reconocer la información visual de acuerdo con patrones generales.
- **Procesamiento de Lenguaje Natural:** Son programas diseñados para tomar lenguajes humanos como entrada y traducirlo en un conjunto estándar de instrucciones que una computadora ejecuta. El propósito de estos complejos programas es permitir a los seres humanos usar su propio lenguaje natural cuando interactúan con programas como sistemas de administración de bases de datos (DBMS) o sistemas de apoyo para la toma de decisiones.
- **Sistemas de aprendizaje:** Una combinación de software y equipos que le permite a la computadora cambiar su modo de funcionar o reaccionar a situaciones, basado en la retroalimentación que recibe.
- **Agentes Inteligentes:** La creación más reciente en IA son los agentes inteligentes, programas de computadora que automáticamente revisan enormes cantidades de datos y seleccionan y entregan la información más adecuada para el usuario, de acuerdo con requisitos contextuales o específicos. La aplicación más importante de los Agentes Inteligentes se encuentra en la WEB[15].

4.1.10. Ramas de la Inteligencia Artificial

Lógica Difusa: La Lógica Difusa se basa en reglas que no tienen límites discretos, sino que se prolongan en un continuum, permitiendo a un sistema manejar mejor la ambigüedad. Esto es muy útil para reflejar cómo tienden a pensar las personas, en términos relativos, no absolutos. Cuando la lógica difusa se incorpora a un SE, el

resultado es un sistema que limita mejor la manera natural en que un experto humano resolvería un problema[16].

- **Redes Neuronales Artificiales:** Denominadas habitualmente como RNA o en inglés como: ANN (Artificial Neural Networks) son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida[17].

Una de las misiones en una red neuronal consiste en simular las propiedades observadas en los sistemas neuronales biológicos a través de modelos matemáticos recreados mediante mecanismos artificiales (como un circuito integrado, una computadora o un conjunto de válvulas). El objetivo es conseguir que las máquinas den respuestas similares a las que es capaz de dar el cerebro que se caracterizan por su generalización y su robustez[16].

- **Algoritmos Genéticos:** Los Algoritmos Genéticos son funciones matemáticas que usan los principios de Darwin para mejorar una aplicación. Las funciones se diseñan para simular en software, en cuestión de minutos o segundos, lo que sucede en ambientes naturales durante millones de años[15].

Los Algoritmos Genéticos establecen una analogía entre el conjunto de soluciones de un problema, llamado fenotipo, y el conjunto de individuos de una población natural, codificando la información de cada solución en una cadena, generalmente binaria, llamada cromosoma. Los símbolos que forman la cadena son llamados los genes. Cuando la representación de los cromosomas se hace con cadenas de dígitos binarios se le conoce como genotipo. Los cromosomas evolucionan a través de iteraciones, llamadas generaciones. En cada generación, los cromosomas son evaluados usando alguna medida de aptitud. Las siguientes generaciones (nuevos cromosomas), llamada descendencia, se forman utilizando dos operadores, de cruzamiento y de mutación[14].

- **Robótica:** La Robótica es una rama del árbol tecnología, que estudia el diseño y construcción de máquinas capaces de desempeñar tareas repetitivas o peligrosas para el ser humano. Las ciencias y tecnologías de las que deriva podrían ser: el álgebra, los

autómatas programables, las máquinas de estados, la mecánica, la electrónica y la informática[15].

La Robótica Incluye el desarrollo de dispositivos mecánicos o de computación que tengan la capacidad de realizar funciones, tales como pintar automóviles, de hacer soldaduras de precisión y realizar otras tareas que requieran de un alto grado de precisión o que sean tediosas o impliquen peligro para los seres humanos. En la robótica contemporánea se combinan las capacidades de alta precisión de la máquina con un software controlador sofisticado[18].

4.1.11. Que son los Algoritmos Genéticos

Son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas[19].

El poder de los AG proviene del hecho de que se trata de una técnica robusta, y pueden tratar con éxito una gran variedad de problemas provenientes de diferentes áreas, incluyendo aquellos en los que otros métodos encuentran dificultades. Si bien no se garantiza que el AG encuentre la solución óptima del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de optimización combinatoria[20].

4.1.12. Por qué utilizar Algoritmos Genéticos en la Optimización

La razón del creciente interés por los AG es que estos son un método global y robusto de búsqueda de las soluciones de problemas. La principal ventaja de estas características es el equilibrio alcanzado entre la eficiencia y eficacia para resolver diferentes y muy complejos problemas de grandes dimensiones[21].

Lo que aventaja a los AG frente a otros algoritmos tradicionales de búsqueda es que se diferencian de estos en los siguientes aspectos:

Trabajan con una codificación de un conjunto de parámetros, no con los parámetros mismos.

- Trabajan con un conjunto de puntos, no con un único punto y su entorno (su técnica de búsqueda es global.) Utilizan un subconjunto del espacio total, para obtener información sobre el universo de búsqueda, a través de las evaluaciones de la función a

optimizar. Esas evaluaciones se emplean de forma eficiente para clasificar los subconjuntos de acuerdo con su idoneidad.

- No necesitan conocimientos específicos sobre el problema a resolver; es decir, no están sujetos a restricciones. Por ejemplo, se pueden aplicar a funciones no continuas, lo cual les abre un amplio campo de aplicaciones que no podrían ser tratadas por los métodos tradicionales.
- Utilizan operadores probabilísticos, en vez de los típicos operadores determinísticos de las técnicas tradicionales.
- Resulta sumamente fácil ejecutarlos en las modernas arquitecturas masivas en paralelo.
- Cuando se usan para problemas de optimización, resultan menos afectados por los máximos locales que las técnicas tradicionales (i.e., son métodos robustos).

Ahora bien; un esquema del funcionamiento general de un algoritmo genético podría ser el siguiente:

- **Algoritmo Genético**:
- Generar una población inicial.
- Iterar hasta un criterio de parada.
- Evaluar cada individuo de la población.
- Seleccionar los progenitores.
- Aplicar el operador de cruce y mutación a estos progenitores.
- Incluir la nueva descendencia para formar una nueva generación[19].

4.1.13. Cómo Saber si es Posible usar un Algoritmo Genético

La aplicación más común de los AG ha sido la solución de problemas de optimización, en donde han mostrado ser muy eficientes y confiables. Sin embargo, no todos los problemas pudieran ser apropiados para la técnica, y se recomienda en general tomar en cuenta las siguientes características del mismo antes de intentar usarla:

- Su espacio de búsqueda (i.e., sus posibles soluciones) debe de estar delimitado dentro de un cierto rango.
- Debe permitir definir una función de aptitud que nos indique que tan buena o mala es una cierta respuesta.

- Las soluciones deben codificarse de una forma que resulte relativamente fácil de implementar en el computador.

El primer punto es muy importante, y lo más recomendable es intentar resolver problemas que tengan espacios de búsqueda discretos aunque éstos sean muy grandes. Sin embargo, también podrá intentarse usar la técnica con espacios de búsqueda continuos, pero preferiblemente cuando exista un rango de soluciones relativamente pequeño[19].

4.1.14. Aplicaciones de los Algoritmos Genéticos

Como hemos podido observar, el área de aplicación de los AG es muy amplia, y en general sus aplicaciones se pueden implementar a muchos de los problemas de la vida cotidiana, de igual forma, se hayan aplicado a diversos problemas y modelos en ingeniaría, y en la ciencia en general cabe destacar entre ellos:

- **Optimización:** Se trata de un campo especialmente abonado para el uso de los AG, por las características intrínsecas de estos problemas. No en vano fueron la fuente de inspiración para los creadores estos algoritmos. Los AG se han utilizado en numerosas tareas de optimización, incluyendo la optimización numérica, y los problemas de optimización combinatoria.
- **Programación automática:** Programación automática: Los AG se han empleado para desarrollar pro- gramas para tareas específicas, y para diseñar otras estructuras computacionales tales como el autómata celular, y las redes de clasificación
- **Aprendizaje máquina:** Los AG se han utilizado también en muchas de es- tas aplicaciones, tales como la predicción del tiempo o la estructura de una proteína. Han servido asimismo para desarrollar determinados aspectos de sistemas particulares de aprendizaje, como pueda ser el de los pesos en una red neuronal, las reglas para sistemas de clasificación de aprendizaje o sistemas de producción simbólica, y los sensores para robots.
- **Economía:** En este caso, se ha hecho uso de estos Algoritmos para moderlizar procesos de innovación, el desarrollo estrategias de puja, y la aparición de mercados económicos.
- **Sistemas inmunes:** A la hora de modelizar varios aspectos de los sistemas inmunes naturales, incluyendo la mutación somática durante la vida de un individuo y el

descubrimiento de familias de genes múltiples en tiempo evolutivo, ha resultado útil el empleo de esta técnica.

- **Ecología**: En la modelización de fenómenos ecológicos tales como las carreras de armamento biológico, la coevaluación de parásitohuesped, la simbiosis, y el flujo de recursos.
- **Genética de poblaciones**: En el estudio de preguntas del tipo "¿Bajo qué condiciones será viable evolutivamente un gene para la recombinación?".
- **Evolución y aprendizaje:** Los AG se han utilizado en el estudio de las relaciones entre el aprendizaje individual y la evolución de la especie.
- **Sistemas sociales:** En el estudio de aspectos evolutivos de los sistemas sociales, tales como la evolución del comportamiento social en colonias de insectos, y la evolución de la cooperación y la comunicación en sistemas multiagentes[21].

4.2. CASOS DE ESTUDIO

Las aplicaciones reales de los AG son muchas y muy variadas, alcanzando campos tan distintos como la biología y biotecnología, a la gestión logística de grandes corporaciones o la optimización de procesos de elevada complejidad. Pero el empleo de los AG no termina en la técnica, ciencias sociales como la economía o la sociología están implementando en algunas de sus áreas esta herramienta de cálculo como modelo de evolución del pensamiento social o económico.

4.2.1. Desarrollo de un marco para la priorización de casos de prueba Utilizando Algoritmos Genéticos

Resumen.- Las pruebas de software es una parte que consume tiempo y esfuerzo en el ciclo de vida del desarrollo desoftware. Volver a probar una aplicación de software durante la fase de mantenimiento, con todo el conjunto de pruebas y casos de prueba adicionales para las modificaciones en el software, dentro del presupuesto y el tiempo, es un reto para los probadores de software. La priorización de casos de prueba se utiliza para superar este problema, dando prioridad a los casos de prueba con el fin de maximizar ciertos objetivos de las pruebas como la tasa de detección de fallos, la cobertura de sentencias, etc. En este trabajo, se propone un marco para el caso de prueba de priorización que hace hincapié en una nueva métrica, APBC (Porcentaje promedio de

Cobertura de bloque). Esta métrica evalúa la tasa de cobertura de código mediante la incorporación de conocimientos sobre la importancia de los bloques de código en la forma de pesas. Hemos utilizado esta medida como función de evaluación de la aptitud en un algoritmo genético con el fin de evaluar la eficacia de una secuencia de casos de prueba. También hemos desarrollado una herramienta que implementa el algoritmo genético en el lenguaje Java con el fin de calcular y validar los resultados. A partir de entonces, se utiliza la herramienta de caso de prueba de priorización, y comparar y evaluar los resultados con los producidos por la herramienta cuando se utiliza APBC (Porcentaje promedio de cobertura bloque) como función de aptitud.

Introducción

Las pruebas de software es el proceso de ejecución de un programa con la intención de encontrar errores. Se trata de una investigación que se lleva a cabo para facilitar a los interesados la información sobre la calidad del producto / servicio que se está probando.

Los estudios empíricos realizados hasta ahora, la comparación de los diferentes técnicas de priorización de casos de prueba, se han utilizado ya sea APBC (Porcentaje Promedio de cobertura de bloque) o APFD (porcentaje promedio de detecciónde fallos) métricas. Estas métricas revelan la velocidad a la que los defectos son descubiertos o la velocidad a la que se logra la cobertura de código.

Por lo tanto, en una aplicación, ciertos bloques contribuyen más a fallos que otras. Nuestro trabajo tiene como objetivo explotar este hecho y asignar pesos a los bloques.

Marco propuesto para la priorización de casos de prueba

En esta sección, proponemos el marco de priorización de casos de prueba.

- El marco

El marco propuesto incluye tres principales componentes-GA

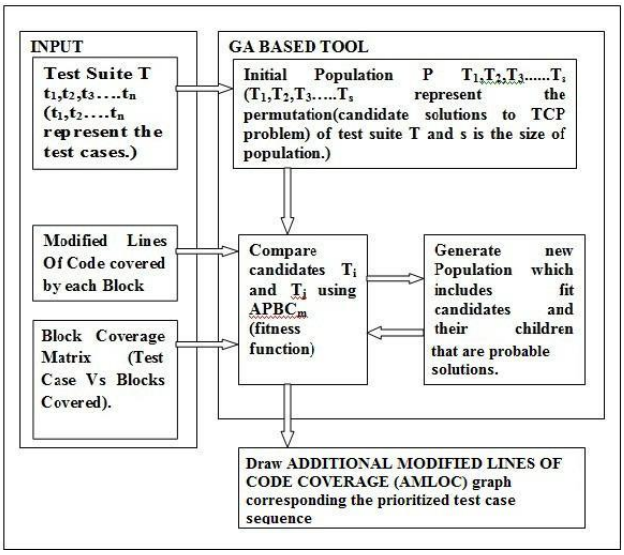


Figura 2: Marco para la priorización de casos de prueba[22]

En caso de herramienta de prueba basada en la priorización, la modificación APBC Métrico (APBCm) y las líneas modificadas adicionales de cobertura de código gráfico (AMLOC). La entrada inicial comprende las permutaciones del conjunto de pruebas que sirven como la población inicial de la herramienta de priorización. La herramienta utiliza la APBCmetrométrica para comparar entre dos permutaciones y decidir cuál de permutación candidato es mejor y debe ser llevada a la siguiente generación de la población. La salida final de la herramienta es una secuencia de caso de prueba priorizada

que maximiza la APBCmetro. La Figura ref{marco} da una visión general del marco propuesto para el caso de prueba priorización.

- **Herramienta basada en Algoritmo Genético**

Desarrollamos un marco que emplea una herramienta de priorización de casos de prueba basado en un algoritmo genético (GA), desarrollado en el lenguaje Java utilizando el IDE Eclipse. La herramienta también utiliza una nueva métrica APBC modificado (APBC), Explicado en el apartado siguiente, que es una forma modificada de métrica APBC originales. La herramienta utiliza los datos de entrada y el modificado APBCm métrica para producir la secuencia de casos de prueba y las Líneas Modificadas adicionales de cobertura de código gráfico (AMLOC) para la secuencia.

APBC Métrico (APBCm) Modificado

En el marco propuesto se utiliza una herramienta basada en GA para dar prioridad a los casos de prueba y el porcentaje de utilización media modificada del bloque de Cobertura (APBCm) Para evaluar el valor de la aptitud de un individuo en la población de GA. Esta métrica se da como sigue:

$$APBC_m = 1 - \frac{(w_1 * TB_1) + (w_2 * TB_2) + \ldots\ldots\ldots + (w_m * TB_m)}{n * (\sum_{i=0}^{m} w_i)} + \frac{1}{2n}$$

Donde ‘n’ es el número de casos de prueba en el caso de prueba conjunto de entrada, ‘m’ es el número de bloques básicos en el programa a ensayar, ‘TB_j’ Denota la ubicación del caso de prueba en la secuencia de casos de prueba que primero encuentra el bloque ‘j’ y ‘w_i’ Denota el peso del Bloque ‘i_{th}’. APBC trata de lograr una cobertura de bloque a un ritmo más rápido. Sin embargo, la APBC métrica tiene los siguientes problemas:

a. La métrica APBC no tiene en cuenta los factores de ponderación prácticas como la importancia de los bloques cubiertos. Varios factores que se pueden utilizar para poner de relieve la importancia de los bloques son los siguientes:

i. Algunos bloques de código, como el código de control de excepciones, no son frecuentemente ejecutado.. Se ha demostrado que la propensión a fallos se correlaciona con las métricas orientadas a objetos, incluyendo las métricas de diseño.

ii. Ciertas construcciones de programación son más propenso a errores que otros, se ha demostrado cómo el análisis de la estructura del programa puede ser usado para encontrar módulos importantes (por ejemplo, métodos) en un código fuente.

iii. Algunos bloques de código contienen un mayor porcentaje de código modificado. La asignación de una prioridad a estos bloques debe dar lugar a una identificación más rápida de fallos puesto que las modificaciones son más susceptibles de contener errores.

Líneas Modificadas adicionales de cobertura de código gráfico (AMLOC)

La herramienta basada GA produce una secuencia priorizada caso de prueba junto con su gráfico AMLOC. Para una secuencia de casos de prueba TS', t_1 , t_2, t_3 , ... t_{p-1}, El valor AMLOC correspondiente a un caso de prueba t_k Con respecto a TS', es la relación entre el número total de líneas únicas modificadas de código fuente que están cubiertos o alcanzado mediante la ejecución de los casos de prueba t_1, t_2 , ... t_k , Donde k_j= n, con el número total de líneas modificadas de código fuente. Utilizamos el gráfico AMLOC para validar los resultados.

Análisis de resultados

En esta sección presentamos y validar los resultados producidos por la estructura propuesta en este documento.

- **Estudio experimental**

El programa de referencia Triángulo ha sido utilizado anteriormente por varios investigadores en los estudios de ingeniería de software. Dadas tres lados de una triángulo, el programa apunta a clasificarlo como un escaleno, isósceles, equilátero o no un triángulo. Para el objetivo de nuestro estudio experimental, usamos una versión simplificada del programa original Triángulo, traducido del 'FORTRAN' en 'C'. El conjunto original de 14 pruebas de casos se redujo a 10 casos de prueba teniendo en cuenta la redundancia de casos de prueba en cuanto a la cobertura de bloque. Utilizamos los bloques básicos algoritmo de identificación, para identificar los elementos básicos del código fuente. La figura 3 muestra el conjunto original de casos de prueba.

En el código, 21 bloques básicos fueron identificados por el algoritmo. Los bloques identificados y obstaculizan cobertura se dan en la figura 4 y la figura 5 respectivamente. A partir de la figura 5 se puede observar que los casos de prueba T1, T2,T3 y T11, T13,

T14 son redundantes en términos de cobertura bloque. El equipo de prueba incluye minimizada T3, T4, T5, T6, T7, T8, T9, T10, T11, T12 y descarta T1, T2, T13, T14.

Input	Expected Output	Input	Expected Output
0,0,0	4 (Not a triangle)	1,2,1	4 (Not a triangle)
1,0,0	4 (Not a triangle)	2,1,1	4 (Not a triangle)
1,1,0	4 (Not a triangle)	3,2,2	2 (Isosceles triangle)
1,1,1	3 (Equilateral triangle)	3,2,1	4 (Not a triangle)
2,2,1	2 (Isosceles triangle)	4,3,2	1 (Scalene triangle)
1,1,2	4 (Not a triangle)	2,3,1	4 (Not a triangle)
2,1,2	2 (Isosceles triangle)	2,1,3	4 (Not a triangle)

Figura 3: Conjunto original de casos de prueba[22]

Block	Lines Comprising the Block	Block	Lines Comprising the Block
B1	1-4	B12	21-24
B2	5-6	B13	25
B3	7-8	B14	26-27
B4	9-10	B15	28
B5	11	B16	29-30
B6	12-13	B17	31
B7	14	B18	32[-33
B8	15-16	B19	34
B9	17	B20	35-36
B10	18	B21	37-38
B11	19-20		

Figura 4: Líneas de código que comprende bloques individuales[22]

Para los efectos de nuestro experimento creamos otra versión modificada del triángulo mediante la introducción de las siguientes características.

i. Si el triángulo es isósceles imprimir la altura del triángulo.

ii. Si el triángulo es escaleno, imprimir el área, y circunferencia del triángulo.

La figura 6 muestra el peso de los bloques después de modificar el programa Triángulo.

La fracción de líneas modificadas contenidas dentro de cada bloque se toma como el peso del bloque. La idea detrás de ello es que, cuanto mayor es el número de modificaciones en un bloque, mayor es la propensión a error del bloque. Al hacerlo, se encontró que algunos bloques de tener peso 0. Con el fin de superar este "problema de peso cero", "1", se agrega a cada uno de los valores. Los pesos se calculan para los bloques usando el programa Triángulo modificado y se muestran en la figura 5.

Test Cases	Blocks Covered	Test Cases	Blocks Covered
T1	B1,B2	T8	B1,B3,B5,B6,B7,B9, B13,B15,B17,B19,21
T2	B1,B2	T9	B1,B3,B5,B7,B8,B9, B13,B15,B17,B19,B21
T3	B1,B2	T10	B1,B3,B5,B7,B8,B9, B13,B15,B17,B19,B20
T4	B1,B3,B4,B5, B6,B7, B8,B9,B13,B14	T11	B1,B3,B5,B7, B9,B10,B11
T5	B1,B3,B4,B5, B7,B9,B13 B15,B16	T12	B1,B3,B5,B7, B9,B10,B12
T6	B1,B3,B4,B5, B7,B9,B13 B15,B17,B19, B21	T13	B1,B3,B5,B7, B9,B10,B11
T7	B1,B3,B5,B6, B7,B9, B13,B15,B17, B18	T14	B1,B3,B5,B7, B9,B10,B11

Figura 5: Bloque Matriz de cobertura[22]

Block	Number of Modified Lines Of Code NMLOC	Weight NMLOC / ∑NMLOC
B1	1+1	0.0645
B2	0+1	0.032
B3	0+1	0.032
B4	0+1	0.032
B5	0+1	0.032
B6	0+1	0.032
B7	0+1	0.032
B8	0+1	0.032
B9	0+1	0.032
B10	0+1	0.032
B11	0+1	0.032
B12	6+1	0.226
B13	0+1	0.032
B14	0+1	0.032
B15	0+1	0.032
B16	1+1	0.0645
B17	0+1	0.032
B18	1+1	0.0645
B19	0+1	0.032
B20	1+1	0.0645
B21	0+1	0.032
Total	31	0.996~1

Figura 6: Peso de los bloques[22]

En nuestro estudio, se realizaron dos experimentos. Experimento 1 produjo el caso de prueba TS1 secuencia de prioridad utilizando los datos de la figura 3 (después de la eliminación de los casos de prueba redundantes) como entrada y APBC como función de aptitud en el algoritmo genético. Experimento 2 produjo la secuencia de caso de prueba priorizado TS2 utilizando los datos de la figura 3 (después de la eliminación de los casos de prueba redundantes) como entrada y APBCm como la función de aptitud en el algoritmo genético. Ambos experimentos se realizaron con la herramienta de priorización de casos de prueba basados en el GA. Los criterios de convergencia utilizados en la herramienta es la aptitud máxima de cualquier individuo contenida en una población que se establece en 0,78. A continuación, comparar los resultados del Experimento 1 y el experimento 2.

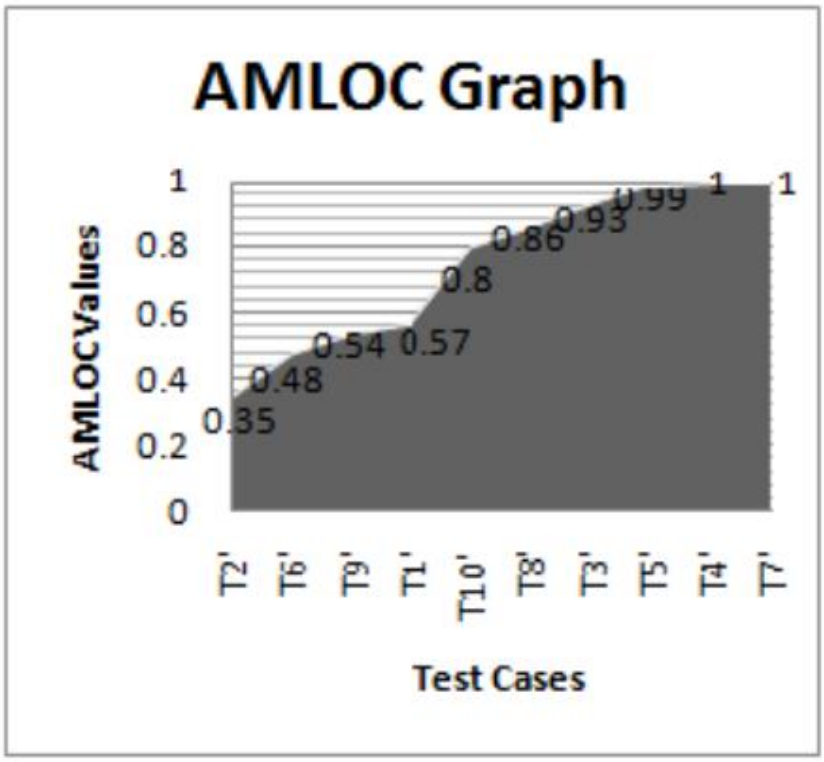


Figura 7:Los valores AMLOC menos por caso de prueba en TS1 producidos en el Experimento 1[22]

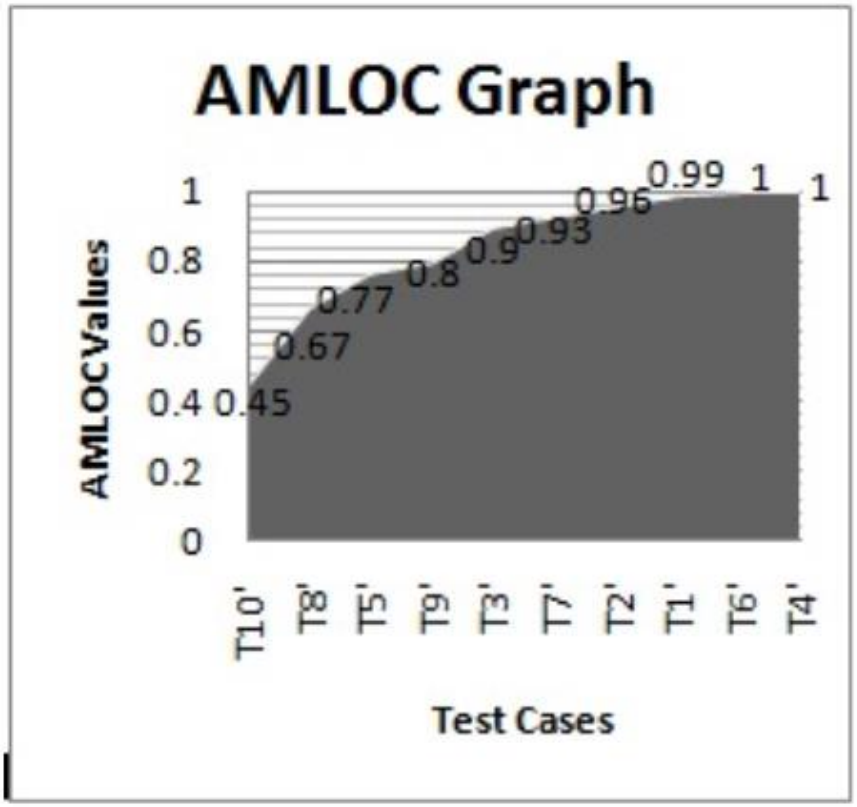


Figura 8: Una mayor convexidad (mejora de la tasa de cobertura) gráfico usando APBC en el Experimento 2[22]

Los resultados de prioridad, tal como se obtienen por nuestra herramienta se muestran en la Figura 7 y la Figura 8 Los valores calculados según APBC modificado y APBC se muestran en la Figura 10 y la Figura 11 A partir de las tablas es evidente que el uso de APBC para comparar el candidato Las soluciones pueden producir secuencias de casos de prueba priorizados ineficaces como de salida. TS2 es mejor, ya que cubre el bloque con un mayor número de líneas modificadas primera (es decir, los bloques con grandes pesos). Por lo tanto, APBC modificado es mejor y se espera que proporcione una ayuda importante en la versión específica de priorización de casos de prueba y ayudar en la revelación de defectos.

Metric / Test Case sequence	APBC
TS1(T2',T6',T9',T1',T10',T8',T3',T5',T4',T7')	0.79285

Figura 9: Resultados del Experimento 1[22]

Metric / Test Case sequence	Modified APBC
TS2(T10',T8',T5',T9',T3',T7',T2',T1',T6',T4')	0.80285

Figura 10: Resultados del Experimento 2[22]

El caso de prueba priorización es una tarea esencial que reduce el esfuerzo de prueba en la fase de mantenimiento en un grado considerable. En este artículo se propone un marco para la priorización de casos de prueba utilizando una herramienta basada en un algoritmo genético, desarrollado en lenguaje Java. El marco también pone de relieve la necesidad y las ventajas de utilizar la nueva métrica APBCm como función de evaluación de la aptitud en el GA. Varios factores que pueden ser utilizados para incrustar el conocimiento acerca de la significación de los bloques en el APBCm métrica también se han discutido. Sin embargo, el cálculo exacto de los pesos teniendo en cuenta todos los factores mencionados, sigue siendo un desafío abierto. Por último, los resultados han sido analizados y comparados con los producidos cuando la métrica tradicional APBC se

utiliza como función de evaluación de la aptitud en la herramienta basada GA. A continuación, se encontró que APBCm métrica es mejor y más eficiente que APBC.

El enfoque, que se presenta aquí, es la priorización de caso de prueba que tiene su aplicación en las áreas de versión específica, sino también se puede generalizar con ligeras modificaciones. Teniendo en cuenta los factores de peso prácticos es un concepto general que puede ayudar a mejorar el costo de las pruebas de regresión y también se puede extender a los problemas de la prueba de reducción al mínimo de baño y selección de pruebas de regresión[22].

4.3. SITUACIÓN ACTUAL DE LOS ALGORITMOS GENÉTICOS/INTELIGENCIA ARTIFICIAL EN LA INGENIERÍA DE SOFTWARE

4.3.1. Panorama actual

Temas como calidad de software, calidad de productos, calidad de procesos también han llegado a oídos hispanos. A falta de contar con una norma específica para países latinoamericanos y al continuo afán por contar con el reconocimiento mundial, las organizaciones latinoamericanas dedicadas al desarrollo y manutención de software han tenido que aplicar modelos o normas ideados por países de primer mundo.

Hablando de modelos y herramientas para guiar el mejoramiento del proceso de software, ¿es posible aplicarlos en forma directa a una empresa u organización Informática altamente inmadura de América Latina? Creemos que esta es una de las preguntas que usualmente pueden surgir cuando se plantea la opción de aplicar determinado estándar o modelo en una empresa dedicada al desarrollo de software.

Antes que nada, debemos pensar que la mayoría de las normas fueron ideadas y construidas bajo economías totalmente diferentes a la que enfrenta Latino América. Aun así, la economía no es el único factor de importancia: la cultura, la modalidad de trabajo y la manera de afrontar los problemas son algunos “atributos”, que entre otros, caracterizan a cada país y que no necesariamente serán homogéneos aún dentro del área que llaman Latino América. Como consecuencia, los modelos deben ser ajustados a nuestra propia realidad[23].

4.3.2. Mejoramiento del proceso de software

El mejoramiento del proceso de software abarca un conjunto de actividades que conducirán a un mejor proceso de software y, en consecuencia, a software de mayor calidad y a su entrega en forma más oportuna. cite{Jos2008}

QUÉ ES MPS

El término mejoramiento del proceso de software (MPS) implica muchas cosas. Primero, que los elementos de un proceso de software efectivo pueden definirse en forma efectiva; segundo, que un enfoque organizacional existente sobre el desarrollo del software puede valorarse en contraste con dichos elementos; y tercero, que es posible definir una

estrategia de mejoramiento significativa. La estrategia MPS transforma el enfoque existente sobre el desarrollo del software en algo que es más enfocado, más repetible y más confiable (en términos de la calidad del producto producido y de la oportunidad de la entrega).

Puesto que el MPS no es gratuito, debe entregar un rendimiento sobre la inversión. El esfuerzo y el tiempo que se requieren para implementar una estrategia MPS deben pagar por sí mismos en alguna forma mensurable. Para hacer esto, los resultados del proceso y la práctica mejorados deben conducir a una reducción en los "problemas" del software que cuestan tiempo y dinero. Debe reducir el número de defectos que se entregan a los usuarios finales, la cantidad de repetición de proceso debida a problemas de calidad, los costos asociados con el mantenimiento y el soporte del software y los costos indirectos que ocurren cuando el software se entrega tarde[24].

4.3.3. Inteligencia Artificial mediante Ingeniería de Software

La inteligencia artificial es una de las ciencias que avanza con bastante fuerza en la actualidad, ya que genera nuevos conceptos y técnicas para la solución de diferentes problemas. Para facilitar el proceso de desarrollo de soluciones mediante la inteligencia artificial, se puede dar uso a la ingeniería de software, el cual permitiría modelar las diferentes soluciones de la inteligencia artificial.

El diseño y desarrollo confiable, robusto, bien elaborado en su arquitectura y fácil de extender en aplicaciones de Software o herramientas en cualquier campo exige la conformidad con principios racionales y las normas de ingeniería de software. Sistemas inteligentes, especialmente el desarrollo de Herramientas IA, no son una excepción. Aunque la Inteligencia Artificial (IA) siempre ha sido una fuente de ideas que la ingeniería de software ha adoptado, la mayor parte de sus joyas siguen en laboratorios, disponible sólo a unos pocos profesionales de la IA. Las herramientas de IA se deberían integrar con las principales herramientas de desarrollo de software, de esa manera, llegaría a ser más conocido y utilizado[25].

4.3.4. Por qué utilizar Algoritmos Genéticos en la Optimización

La razón del creciente interés por los AG es que estos son un método global y robusto de búsqueda de las soluciones de problemas. La principal ventaja de estas características es

el equilibrio alcanzado entre la eficiencia y eficacia para resolver diferentes y muy complejos problemas de grandes dimensiones.

Lo que aventaja a los AG frente a otros algoritmos tradicionales de búsqueda es que se diferencian de estos en los siguientes aspectos:

- Trabajan con una codificación de un conjunto de parámetros, no con los parámetros mismos.
- Trabajan con un conjunto de puntos, no con un único punto y su entorno (su técnica de búsqueda es global.) Utilizan un subconjunto del espacio total, para obtener información sobre el universo de búsqueda, a través de las evaluaciones de la función a optimizar. Esas evaluaciones se emplean de forma eficiente para clasificar los subconjuntos de acuerdo con su idoneidad.
- No necesitan conocimientos específicos sobre el problema a resolver; es decir, no están sujetos a restricciones. Por ejemplo, se pueden aplicar a funciones no continuas, lo cual les abre un amplio campo de aplicaciones que no podrían ser tratadas por los métodos tradicionales.
- Utilizan operadores probabilísticos, en vez de los típicos operadores determinísticos de las técnicas tradicionales.
- Resulta sumamente fácil ejecutarlos en las modernas arquitecturas masivas en paralelo.
- Cuando se usan para problemas de optimización, resultan menos afectados por los máximos locales que las técnicas tradicionales (i.e., son métodos robustos[26].

4.3.5. Algoritmos Genéticos en la Actualidad

En la actualidad los algoritmos genéticos se utilizan para infinidad de cosas, existen trabajos y proyectos tanto comerciales como no comerciales. Entre ellas tenemos empresas de aplicaciones como las siguientes:

JGAP: Aplicación hecha en java para generar soluciones a problemas implementando algoritmos genéticos, tiene diversas funcionalidades como: educativas, de investigación o simple entretenimiento.

Optimal Synthesis Inc.: Organización que ofrece herramientas para búsqueda genética a partir de los algoritmos genéticos, abalada por la marina de Estados Unidos, ofrece su producto para trabajar con MatLab.

Anglo American Chaos: Modelado y análisis de datos no lineales en conjunto con los algoritmos genéticos y otras tecnologías de la IA como programación genética, predicción de patrones, estimación de ruidos, etc.

Perfect tablePlan: Programa funcional que ayuda a planear la localización de las personas en una boda.

RML Technologies Inc.: Programa Discipulus™ 5 que usa la programación genética basada en los algoritmos genéticos para predecir modelos.

JAGA: Software para crear aplicaciones que funcionen con algoritmos genéticos y programación genética.

En la actualidad los usos y tecnologías de los algoritmos genéticos van mejorando y creciendo cada día más. A continuación se mencionan algunas áreas y clases donde se aplican los algoritmos genéticos.

Áreas de aplicación y clases

Optimización: Se trata de un campo especialmente abonado para el uso de los AG, por las características intrínsecas de estos problemas. No en vano fueron la fuente de inspiración para los creadores de estos algoritmos. Los AG se han utilizado en numerosas tareas de optimización, incluyendo la optimización numérica, y los problemas de optimización combinatoria.

Programación automática: Los AG se han empleado para desarrollar programas para tareas específicas, y para diseñar otras estructuras computacionales tales como el autómata celular, y las redes de clasificación.

Aprendizaje máquina: Los AG se han utilizado también en muchas de estas aplicaciones, tales como la predicción del tiempo o la estructura de una proteína. Han servido asimismo para desarrollar determinados aspectos de sistemas particulares de aprendizaje, como pueda ser el de los pesos en una red neuronal, las reglas para sistemas

de clasificación de aprendizaje o sistemas de producción simbólica, y los sensores para robots.

Economía: En este caso, se ha hecho uso de estos Algoritmos para modelar procesos de innovación, el desarrollo de estrategias de puja, y la aparición de mercados económicos.

Sistemas inmunes: Al momento de modelar varios aspectos de los sistemas inmunes naturales, incluyendo la mutación somática durante la vida de un individuo y el descubrimiento de familias de genes múltiples en tiempo evolutivo, ha resultado útil el empleo de esta técnica.

Ecología: En el modelado de fenómenos ecológicos tales como las carreras de armamento biológico, la coevolución de parásito-huesped, la simbiosis, y el flujo de recursos.

Evolución y aprendizaje: Los AG se han utilizado en el estudio de las relaciones entre el aprendizaje individual y la evolución de la especie.

Sistemas sociales: En el estudio de aspectos evolutivos de los sistemas sociales, tales como la evolución del comportamiento social en colonias de insectos, y la evolución de la cooperación y la comunicación en sistemas multiagentes[27].

4.4. EL PROCESO DE LA INGENIERIA DEL SOFTWARE

4.4.1. EL PROCESO

Es una serie de pasos a seguir para construir un producto o un sistema. El proceso del software es importante porque proporciona estabilidad, control y organización a una actividad que puede, si no se controla, volverse caótica.

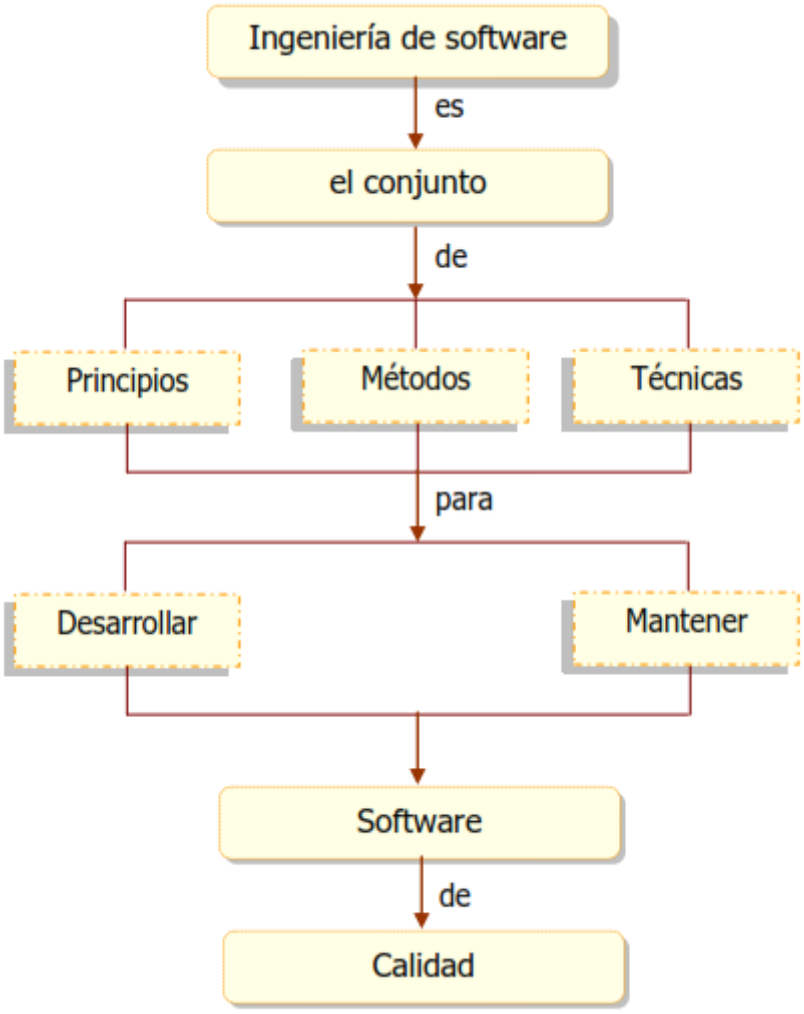


Figura 11: Proceso[10]

Esquema de la Ingeniería del Software

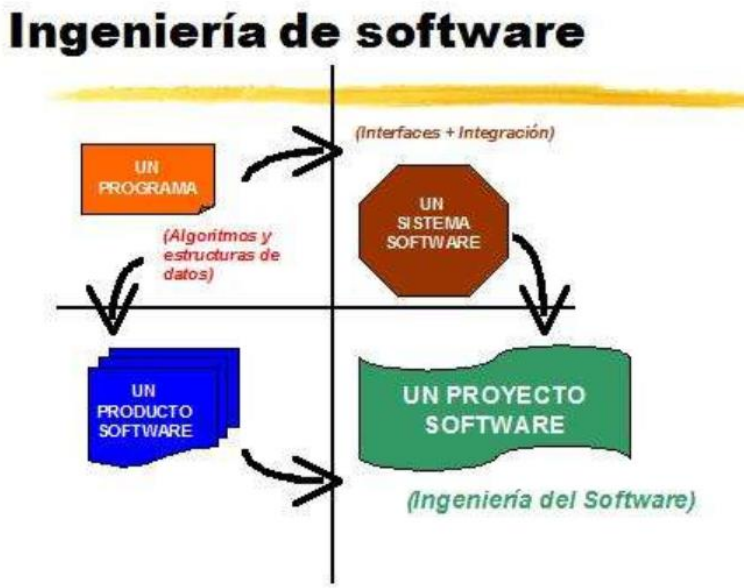


Figura 12: Proceso de la Ingeniería software[10]

Es muy simple el esquema que consiste en desarrollar un programa sencillo que resuelve una tarea bien determinada. Lo normal es que se evolucione al desarrollo de un:

- Sistema software: integra varios programas, o
- Producto software: programa usado en diferentes aplicaciones/entornos

Ambos desarrollos "dan lugar a la Ingeniería del Software": Programas integrados que pueden trabajar en varios entornos[10].

Proceso, métodos y herramientas

La ingeniería del software es una tecnología multicapa, y que se apoya sobre un enfoque de calidad.

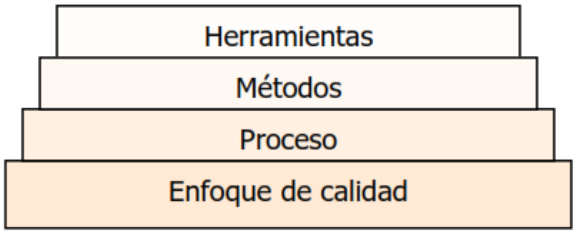


Figura 13:Proceso, métodos y herramientas[11].

- **Enfoque de calidad:** Gestión total de calidad y de mejoras de procesos.
- **Proceso:** Define un número de actividades del marco de trabajo aplicables a todos los proyectos del software.
- **Métodos:** Indican "cómo" construir técnicamente el software. Abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento.
- **Herramientas:** Soporte automático o semi-automático para el proceso y los métodos[11].

Fasces de la Ingeniería de Software

La ingeniería es el análisis, diseño, construcción, verificación y gestión de entidades técnicas.

El trabajo que se asocia a la ingeniería del software se puede dividir en tres fases, con independencia del área de aplicación, tamaño o complejidad del proyecto

Fase de definición

Se centra sobre el textbf{qué}. Identificar qué información ha de ser procesada, que función y rendimiento se desea, qué comportamiento del sistema, qué interfaces van a ser establecidas, qué restricciones de diseño existen, y qué criterios de validación se necesitan para definir un sistema correcto. Identificar los requisitos del sistema y del software. Las tareas específicas de esta fase son:

- Ingeniería de Sistemas o de información
- Planificación del proyecto software
- Análisis de requerimientos

Fase de desarrollo

Se centra en el cómo. Definir cómo han de diseñarse las estructuras de datos, cómo ha de implementarse la función dentro de una arquitectura de software, cómo ha de implementarse los detalles procedimentales, cómo han de caracterizarse interfaces, cómo ha de traducirse el diseño en un lenguaje de programación y cómo ha de realizarse la prueba. Las tareas específicas de esta fase son:

- Diseño del software
- Generación de código

- Prueba del software

Fase de mantenimiento

Se centra en el cambio. Corrección de errores. Adaptaciones requeridas a medida que evoluciona el entorno del software. Cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente. Se encuentran cuatro tipos de cambio:

- Corrección
- Adaptación
- Mejora
- Prevención[18].

Proceso del Software

Un proceso de software se puede caracterizar así:

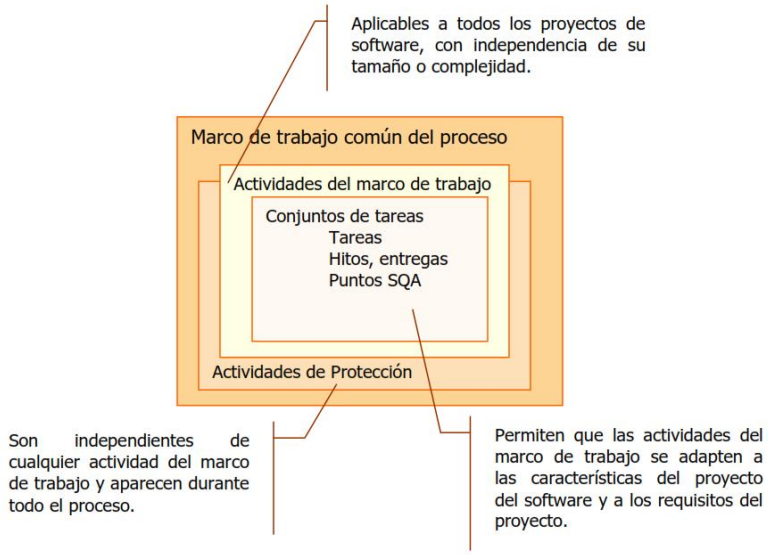


Figura 14: Proceso de software[28].

Modelos de proceso del software

Es importante incorporar estrategias de desarrollo que acompañe al proceso, métodos y a las herramientas.

Una estrategia a menudo de llama modelo de proceso o paradigma de ingeniería del software. Se selecciona un modelo de proceso para la ingeniería del software según la

naturaleza del proyecto y de la aplicación, los métodos y las herramientas a utilizarse, y los controles y entregas que se requieren.

Modelo lineal secuencial: Llamado algunas veces "ciclo de vida básico" o "modelo en cascada", el modelo lineal secuencial sugiere un enfoque sistemático, secuencial, para el desarrollo del software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento.

Es un ciclo de vida en sentido amplio, que incluye no sólo las etapas de ingeniería sino toda la vida del producto: las pruebas, el uso (la vida útil del software) y el mantenimiento[18].

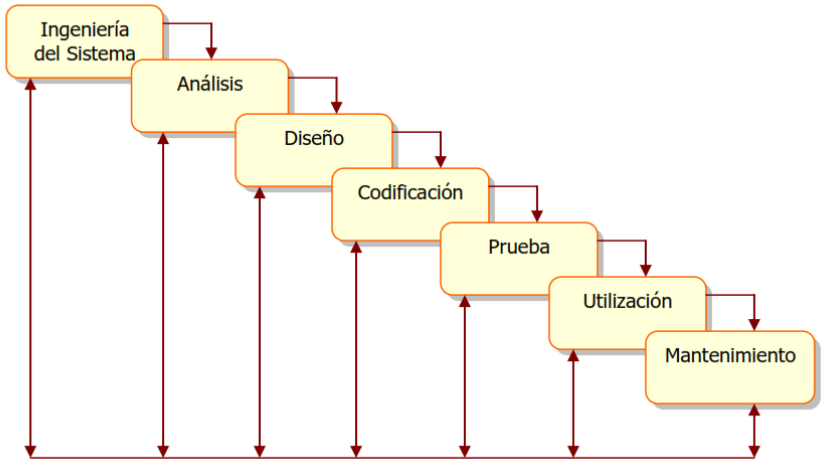


Figura 15: Modelo lineal secuencia[18]*l.*

Ingeniería del Sistema: Análisis de las características y el comportamiento del sistema del cual el software va a formar parte.

Para un sistema nuevo: Se debe analizar cuáles son los requisitos funciones del sistema, y luego asignar un subconjunto de estos requisitos y funciones al software.

Para un sistema ya existente: se debe analizar el funcionamiento de la organización y sus operaciones y se asigna al software aquellas funciones que se van a automatizar.

Está formado por diagramas y por descripciones en lenguaje natural[9].

Análisis: Se debe comprender cuáles son los datos que se van a manejar, cuál va a ser la función que tiene que cumplir el software, cuáles son las interfaces requeridas y cuál es el rendimiento y otros requisitos no funcionales que se esperan lograr.

Los requisitos, tanto del sistema como del software deben documentarse y revisarse con el cliente. Como resultado de la fase de análisis, se obtiene la especificación de requisitos del software[28].

También está formado por diagramas y descripciones en lenguaje natural.

Diseño: El diseño se aplica a cuatro características distintas del software: la estructura de los datos, la arquitectura de las aplicaciones, la estructura interna de los programas y las interfaces[9].

El diseño es el proceso que traduce los requisitos en una representación del software de forma que pueda conocerse la arquitectura, funcionalidad e incluso la calidad del mismo antes de comenzar la codificación[10].

En el diseño, los requisitos del software se traducen a una serie de diagramas que representan la estructura del sistema software, de sus datos, de sus programas y de sus interfaces[11].

Codificación: Consiste en la traducción del diseño a un formato que sea comprensible para la máquina. Si el diseño es lo suficientemente detallado, la codificación es relativamente sencilla, y puede hacerse de forma automática, usando generadores de código[9].

Se traducen los diagramas de diseño a un lenguaje fuente, que luego se traduce - se compila - para obtener un programa ejecutable[10].

Prueba: El objetivo es comprobar que no se hayan producido errores en alguna de las fases anteriores, especialmente en la codificación.

Se deben probar todas las sentencias, y todos los módulos que forman parte del sistema[11].

Utilización: El software se entrega al cliente y comienza la vida útil del mismo.

Mantenimiento: El software sufrirá cambios a lo largo de su vida útil. Estos cambios pueden ser debidos a tres causas:

Que, durante la utilización, el cliente detecte errores en el software: los errores latentes.

Que se produzcan cambios en alguno de los componentes del sistema.

Que el cliente requiera modificaciones funcionales no contempladas en el proyecto[18].

Modelo de construcción de prototipos

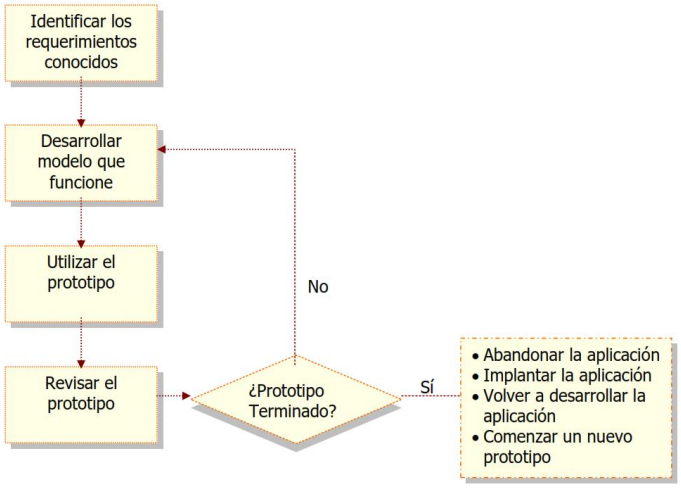


Figura 16: Modelo de construcción de prototipos[18].

Desarrollar modelo que funcione: Los desarrolladores explican a los usuarios:

- El método
- Las actividades a realizar
- La secuencia en que se llevará a cabo
- La responsabilidad de cada participante

El proceso de construcción del prototipo se debe iniciar con el desarrollo de un plan general que permita conocer el proceso de desarrollo[10].

Es importante definir un cronograma para el inicio y fin de la primera iteración.

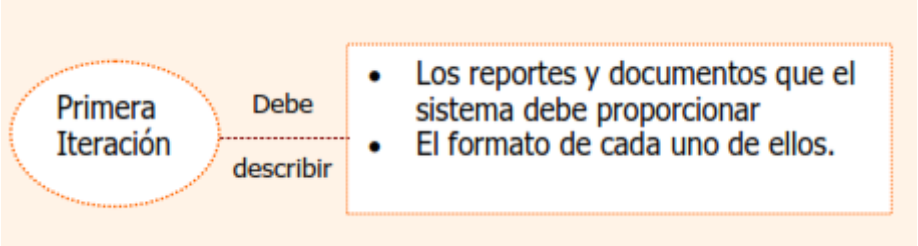


Figura 17: Modelo funcional[10].

El desarrollador estima los costos asociados con el desarrollo del prototipo.

- En el desarrollo del prototipo se preparan los siguientes componentes:
- El lenguaje de diálogo o conversación entre el usuario y el sistema
- Pantallas y formatos para la entrada de datos
- Módulos esenciales de procesamiento
- Salida del sistema

En esta fase no se prepara la documentación ni las especificaciones de salida o de diseño del software[9].

Utilizar el prototipo: La responsabilidad de trabajar con el prototipo y evaluar sus características y operación es del usuario[10].

La experiencia con el sistema bajo condiciones reales permite determinar los cambios o mejoras o eliminar características innecesarias[11].

Revisar el prototipo: Se realiza la evaluación y con la información obtenida se levantan las características que debe llevar la siguiente versión del prototipo.

La evaluación permite profundizar los rasgos de los usuarios y los de la organización que tienen influencia sobre la aplicación y en su implementación.

Los cambios en el prototipo son planificados con los usuarios antes de llevarlos a cabo por el analista[29].

¿Prototipo terminado?: Los pasos anteriores se repiten varias veces (4 o 6 iteraciones) cuando los usuarios y desarrolladores están de acuerdo en que el sistema ha evolucionado lo suficiente e incluye todas las características necesarias[29].

Cuando el prototipo está terminado, el paso que sigue a continuación es tomar la decisión sobre cómo proceder, para lo cual existen cuatro opciones:

- **Abandonar la aplicación:** Se descartan el prototipo y la aplicación. El desarrollo del prototipo proporcionó información a partir de la cual se determinó que la aplicación o el enfoque seleccionado son inapropiados para justificar un desarrollo adicional.
- **Implantar el prototipo:** Las características y funcionamiento del prototipo satisfacen las necesidades de los usuarios ya sea en forma permanente o para un futuro.
- **Volver a desarrollar la aplicación:** El desarrollo del prototipo proporcionó suficiente información para determinar las características necesarias de toda la aplicación. La información se utiliza como punto de partida para el desarrollo de la aplicación en forma tal haga el mejor uso posible de los recursos.
- **Comenzar un nuevo prototipo:** La información ganada con el desarrollo del prototipo inicial sugiere otras opciones o circunstancia. Se construye un prototipo diferente para añadir información relacionada con los requerimientos de aplicación.

Un prototipo puede tener alguna de las tres formas siguientes:

- Un prototipo, en papel o ejecutable en computador, interacción hombre-máquina y los listados del sistema.
- Un prototipo que implemente algún(os) subconjunto(s) de la función requerida, y que sirva para evaluar el rendimiento de un algoritmo o las necesidades de capacidad de almacenamiento y velocidad de cálculo del sistema final.
- Un programa que realice en todo o en parte la función deseada pero que tenga características que deban ser mejoradas durante el desarrollo del proyecto[18].

Modelo DRA (Desarrollo Rápido de Aplicaciones)

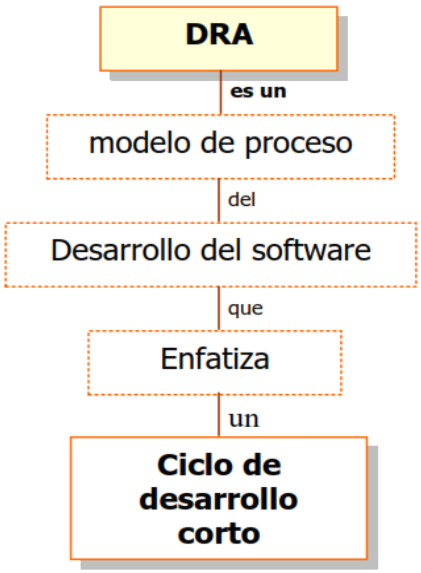


Figura 18: Desarrollo Rápido de Aplicaciones[18].

El proceso DRA permite al equipo de desarrollo crear un "sistema completamente funcional" dentro de periodos cortos de tiempo (de 60 a 90 días). El enfoque DRA comprende las siguientes fases[10]:

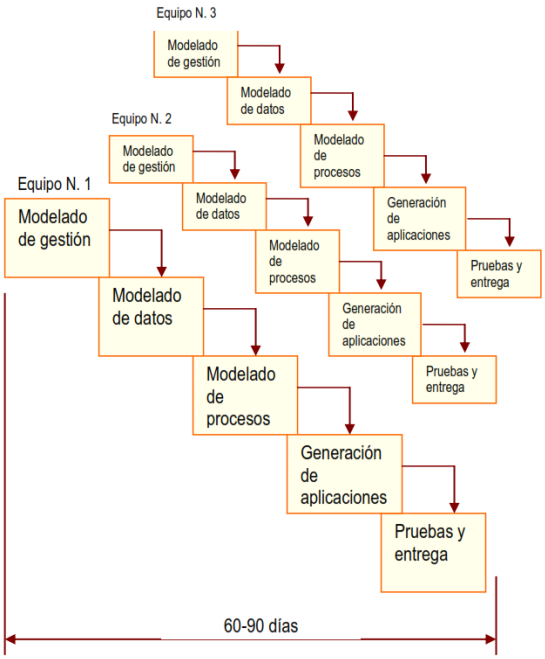


Figura 19: Desarrollo Rápido de Aplicaciones[10].

Modelado de gestión: El flujo de información entre las funciones de gestión se modela de forma que responda a las siguientes preguntas: ¿Qué información conduce al proceso de gestión? ¿Qué información se genera? ¿Quién la genera? ¿A dónde va la información? ¿Quién la procesa?

Modelado de datos: Conjunto de objetos de datos necesarios para apoyar la empresa.

Se definen las características (atributos) de cada uno de los objetos y las relaciones entre estos objetos.

Modelado del proceso: Los objetos de datos definidos en la fase de modelado de datos quedan transformados para lograr el flujo de información necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir o recuperar un objeto de datos.

Generación de aplicaciones: El DRA asume la utilización de técnicas de cuarta generación. En lugar de crear software con lenguajes de programación de tercera generación, el proceso DRA trabaja para volver a utiliza componentes de programas ya existentes o crear componente reutilizables.

Pruebas y Entrega: Como el proceso DRA enfatiza la reutilización, ya se han comprobado muchos de los componentes de los programas. Esto reduce tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y se deben ejercitar todas las interfaces afondo[10].

5. Materiales y Métodos

5.1. Materiales

Para el desarrollo del Trabajo de Titulación se necesitó de algunos materiales, estos se los detalla en la Tabla 1.

Hardware	
Computador	Se utilizó para realizar todos los procesos digitales y facilitar el trabajo.
Impresora	Se realizó las respectivas impresiones del trabajo cuando fue necesario.
Dispositivo de Almacenamiento	Se ocuparon herramientas que tienen almacenamiento en la nube: Sharelatex, Mendeley.
Software	
Sharelatex	Para la edición de texto así como las revisión de avances online
Tex Maker	Para la edición de texto desde el escritorio en los momentos cuando no se disponía de servicio de internet
Mendeley	Para la bibliográfica creando una base de datos .bib para poder referenciar o citar desde el Sharelatex.
Fuentes de Búsqueda	
ACM	Base de datos científica desde la que se obtuvo estudios para el desarrollo de la investigación.
IEEE	Base de datos científica desde la que se obtuvo estudios para el desarrollo de la investigación.
Scopus	Base de datos científica desde la que se obtuvo estudios para el desarrollo de la investigación.
Otros Materiales	
Internet	Una de las principales herramientas que permitió el acceso a las diferentes herramientas utilizadas.
Material de Oficina	Sirvieron como complemento para el desarrollo del Trabajo de Titulación.
Transporte	Permitió dirigirme hasta los diferentes destinos que fueron necesarios acudir para la culminación exitosa del presente trabajo.

Tabla 1: Materiale

5.2. Método

El método utilizado para la realización de este Trabajo de Titulación es una Revisión Sistemática a través de la cual se identifica, evalúa, interpreta y discute toda la documentación existente acerca de la problemática

Revisión Sistemática

La revisión Sistemática es una revisión de la literatura existente utilizando un enfoque que nos ayuda a identificar, evaluar e interpretar los estudios más importantes o destacados (llamados estudios primarios) que van a dar respuesta a una pregunta de investigación en particular.

Es por esto que se destacan varios puntos que nos demuestran la importancia de utilizar este enfoque:

- Resume la evidencia existente concerniente a un tratamiento o tecnología
- Identifica brechas en la investigación actual para sugerir áreas de posterior investigación.
- Proporciona un marco para posicionar apropiadamente nuevas actividades de investigación.

La técnica de revisión sistemática comienza definiendo un protocolo de revisión que especifica la cuestión investigada y los métodos a utilizar, documentando su estrategia de búsqueda para que los lectores puedan conocer su rigor y compleción. Este tipo de estudio se basa en una estrategia de investigación definida que pretende detectar toda la literatura relevante posible, requiriendo de criterios explícitos de inclusión y exclusión para evaluar cada estudio primario potencial y especificando la información de cada uno de estos estudios primarios incluyendo criterios de calidad.

Barbara Kitchenham propone un método para la realización de revisiones sistemáticas en el contexto de la Ingeniería del Software[8], el cuál será utilizado en la presente revisión sistemática. A continuación podemos ver un cuadro resumen con las principales etapas y actividades de la revisión sistemática:

Etapa 1	Planificación de la revisión
	Identificación de la necesidad de la revisión
	Desarrollo de un protocolo de revisión
Etapa 2	Desarrollo de la revisión
	Identificación de la investigación
	Selección de los estudios primarios
	Evaluación de la calidad del estudio
	Extracción y monitoreo de datos
	Síntesis de datos
Etapa 3	**Publicación de los resultados**

Tabla 2: Etapas de una Revisión Sistemática[8]

En relación con el método propuesto por Kitchenham, Biolchini define una plantilla para el protocolo que nos sirve como guía en las etapas de planificación y ejecución de la revisión sistemática[8].

Dicha plantilla ha sido utilizada como guía para el desarrollo de esta revisión sistemática y está organizada en base a una serie de secciones y subsecciones tal y como podemos ver en la Tabla 3.

Planificación de la revisión
Formulación de la pregunta • **Foco de la pregunta** • **Amplitud y calidad de la pregunta**
Selección de fuentes • **Definición del criterio de selección de fuentes** • **Lenguaje de estudio** • **Identificación de fuentes** • **Selección de fuentes después de la evaluación** • **Comprobación de las fuentes**
Selección de los estudios • **Definición del criterio de inclusión y exclusión de estudios** • **Definición de tipos de estudio** • **Procedimiento para la selección de los estudios**
Ejecución de la revisión
Ejecución de la selección en la fuente "x" • **Selección de estudios iniciales**

• **Evaluación de la calidad de los estudios** • **Revisión de la selección** • **Extracción de información**
Análisis de resultados

Tabla 3: Plantilla de Kitchenham, Biolchini para una Revisión Sistemática[8].

Una vez definidas las etapas y los procedimientos que se deben seguir para la realización de una revisión sistemática, en el presente documento mostraremos la ejecución de la misma, tomando en cuenta cada uno de los conceptos antes expuestos y llegaremos al análisis de los resultados obtenidos y por ende a la conclusión que nos da como respuesta la revisión sistemática realizada.

Planificación de la Revisión

En esta etapa se identifica la necesidad de realizar una revisión para lo cual definimos primeramente los objetivos que esta persigue, además de las fuentes que nos proporcionarán los estudios primarios, y los criterios de búsqueda que se van a implementar así mismo los razonamientos utilizados para la inclusión o exclusión de la bibliografía a estudiar y los procesos que nos guiarán para obtener las conclusiones y los resultados deseados.

5.2.1. Formulación de las preguntas

Este proceso de estudio se comienza formulando las preguntas de investigación en las cuales vamos a basar todo el proceso de investigación, estas se sintetizan de forma que se tomen en cuenta el área de interés de investigación y queden definidos no solo el problema a tratar, sino también sus principales características.

Foco de las preguntas

En esta revisión sistemática se busca establecer principalmente los efectos que existen al aplicar algoritmos genéticos en la ingeniería del software y todas las repercusiones que estos ocasionan tanto en la optimización como en los diferentes procesos en los que estos intervienen.

Amplitud y calidad de las preguntas.

Para establecer la amplitud y la calidad de las preguntas que nos planteamos, debemos tomar en cuenta las respuestas que se dan en los diferentes documentos y estudios que se

muestran una vez definidos los parámetros de búsqueda y las palabras claves que se utilizarán para la misma.

Problema

Al revisar los diferentes procesos que se utilizan en el desarrollo de la Ingeniería de Software, notamos que la aplicación de algoritmos genéticos no solo ha sido empleada por la agilización de varios de estos sino por la optimización de varias etapas de desarrollo de la misma, por lo que es necesario el estudio que muestre la optimización que se presenta al utilizar algoritmos genéticos en el desarrollo de las etapas de la Ingeniera de Software.

Preguntas de investigación

Una vez identificado el problema podemos definir de manera correcta las preguntas de investigación, las mismas que se expresan de la siguiente manera:

- ¿Qué problema soluciona en la ingeniera del software los Algoritmos Genéticos?
- ¿Qué aplicación tienen los Algoritmos Genéticos en la Ingeniería del Software?
- ¿Qué tecnología utilizan los algoritmos genéticos para su ejecución?

Palabras clave y sinónimos

Antes de realizar la extracción de los trabajos que se van a analizar es necesario definir un conjunto de palabras claves que nos servirán para la creación de las cadenas de búsqueda y selección lo que nos ayudará a obtener los filtros más adecuados que nos brindarán un resultado óptimo en cuanto a clasificación de trabajos primarios.

Para la selección de estas palabras clave y sus conceptos relacionados, así como para conocer la traducción correcta al inglés de cada término, se revisaron varios artículos tanto de ingeniería de software como de algoritmos genéticos, y basados en estudios preliminares.

Keywords: genetic algorithms, software engineering, evolutionary algorithms, optimization, software requirements, software design, software patrón,

Intervención

En el contexto de la revisión sistemática planificada se van a observar las propuestas existentes sobre algoritmos genéticos en la ingeniería de software, extrayendo las más importantes y procediendo a un posterior análisis de las mismas.

Control

En la presente revisión sistemática, aunque se han observado algunos trabajos para poder obtener las palabras clave, no se considera ningún dato o trabajo inicial que deba estar incluido como trabajo primario en el conjunto de resultados, de forma que todos los trabajos primarios incluidos vengan derivados de la aplicación de los criterios definidos y sepamos que cumplen con el objetivo buscado.

Resultado

Los resultados esperados de esta revisión es conocer las propuestas existentes en cuanto a la aplicación de los algoritmos genéticos en la ingeniería de software, para posteriormente analizarlas y conocer qué comparten y en que difieren, además de identificar necesidades de investigación.

Medida de salida

Para medir los resultados obtenidos utilizaremos en primer lugar una agrupación de las propuestas encontradas, por etapas o las fases de desarrollo software donde se aplican los algoritmos genéticos.

Población

La población a analizar se compone de las publicaciones presentes en los repositorios de las fuentes de datos seleccionadas que estén relacionadas con el objetivo de esta revisión.

Aplicación

Los beneficiarios de la revisión sistemática serán las personas (académicos, investigadores, profesionales, etc.) relacionadas directamente con la aplicación de los algoritmos genéticos en la ingeniería de software, así como las relacionadas con alguno de los dos campos de forma independiente, que estén interesadas en conocer los trabajos relevantes existentes que hayan tratado el problema de la optimización de procesos en la ingeniería de software.

Diseño experimental

El meta-análisis de la revisión está enfocado a analizar los algoritmos genéticos en la ingeniería del software en los estudios primarios para, por un lado conocer las tendencias actuales analizando el área de interés en la que se centra y por otro lado comparar los estudios más significativos identificados en esta revisión mediante un marco de comparación, lo cual nos dará una visión del panorama actual y detectará las deficiencias de las algoritmos genéticos, en caso de que las hubiera.

5.2.2. Selección de fuentes

En este apartado analizamos principalmente las fuentes que se usarán para realizar la ejecución de la revisión. Posteriormente se utilizarán los elementos definidos en la planificación para aplicar el procedimiento de obtención de estudios primarios en cada una de las fuentes seleccionadas.

Definición del criterio de selección de fuentes

El criterio para la selección de las fuentes de búsqueda está basado a un ranquink mundian donde se encuentran las mejores bases de datos científicas. Otros requisitos exigidos a las fuentes para su selección son su accesibilidad vía web y la inclusión motores de búsqueda que permitan consultas avanzadas.

Lenguaje de estudio

El lenguaje de los estudios primarios será el inglés y estos serán extraídos mediante consultas en las que las palabras clave están en inglés. El informe de la revisión sistemática se realiza en español.

Identificación de fuentes

En este punto identificamos las fuentes sobre las que se ejecutará la revisión sistemática planificada, definiendo el método de selección de fuentes y la lista de fuentes consideradas, así como estableciendo las cadenas de búsqueda que usaremos en la ejecución de la revisión.

Método de selección de fuentes

Para obtener estudios de calidad que contribuyan a responder las preguntas de investigación se utilizó fuentes de búsqueda conocidas; considerando algunos aspectos

como: accesibilidad a la web, inclusión de motores de búsqueda que permitan realizar consultas avanzadas y área de investigación.

Lista de fuentes

En la Tabla 4 se muestra las bases de datos utilizadas con su respectiva dirección web.

Base de Datos	Dirección Web
ACM	http//ieeexplore.ieee.org
IEEE	https:ieeexplore.ieee.org/
Scopus	https://www.scopus.com/

Tabla 4: Fuentes de Búsqueda

Cadenas de búsqueda

Una vez definidas las bases de datos e identificadas las palabras claves se realizó las consultas posibles utilizando los operadores lógicos AND y OR, generando con esto las cadenas de búsqueda que se muestran en la Tabla 5.

Base de Datos	Identificación	Cadena de Búsqueda
ACM	CB01	(+genetic +algorithms +software +engineering +software +requirements)
ACM	CB02	(+ genetic +algorithms +software +requirements
ACM	CB03	+genetic +algorithms +software +testing)
ACM	CB04	(+optimization +computer +systems +genetic +algorithms)
IEEE	CB05	(("Abstract":software engineering) AND ("Abstract":genetic algorithms))
IEEE	CB06	(("Abstract":genetic algorithms) AND ("Abstract":software requirements))
IEEE	CB07	(("Abstract":genetic algorithms) AND ("Abstract":software requirements))
SCOPUS	CB08	(TITLE-ABS-KEY (optimization software engineering) AND TITLE-ABS-KEY (genetic algorithms)) AND PUBYEAR > 2012 AND (LIMIT-TO (SUBJAREA , "COMP"))

SCOPUS	CB09	(TITLE-ABS-KEY (genetic algorithms) AND TITLE-ABS-KEY (software requirements) AND TITLE-ABS-KEY (software design)) AND PUBYEAR > 2012 AND PUBYEAR < 2017 AND (LIMIT-TO (SUBJAREA , "COMP")
SCOPUS	CB10	(TITLE-ABS-KEY (genetic algorithms) AND TITLE-ABS-KEY (web software) OR TITLE-ABS-KEY (software patron)) AND PUBYEAR > 2012 AND (LIMIT-TO (SUBJAREA , "COMP")
SCOPUS	CB11	(TITLE-ABS-KEY (genetic algorithms) AND TITLE-ABS-KEY (engineering phase of the software development)) AND PUBYEAR > 2012

Tabla 5: Cadenas de Búsquedas

Selección de fuentes después de la evaluación

Cabe resaltar que se estimó como estudio todos los artículos de conferencia y artículos de revistas, además se consideró: los artículos que contengan las palabras claves en el abstract y los resultados encontrados en el área de Computación.

Comprobación de las fuentes

En primer lugar las fuentes identificadas fueron Dialnet, Scopus e IEEE, las cuales se sometieron a una fase de comprobación dentro del ranquink mundial de las mejores bases de datos científicas, las que tienen mejor credibilidad..

5.2.3. Selección de los estudios

Una vez que se han sido definidas las fuentes, es necesario describir el proceso y el criterio que vamos a seguir en la ejecución de la revisión para la selección y evaluación de los estudios primarios. Para ello definiremos el proceso completo de selección así como los criterios de inclusión y exclusión que vamos utilizar.

Procedimiento para la selección de los estudios

El proceso de selección de los estudios primarios de una fuente concreta se lo desarrolló en forma de algoritmo como se ve en la Figura 20.

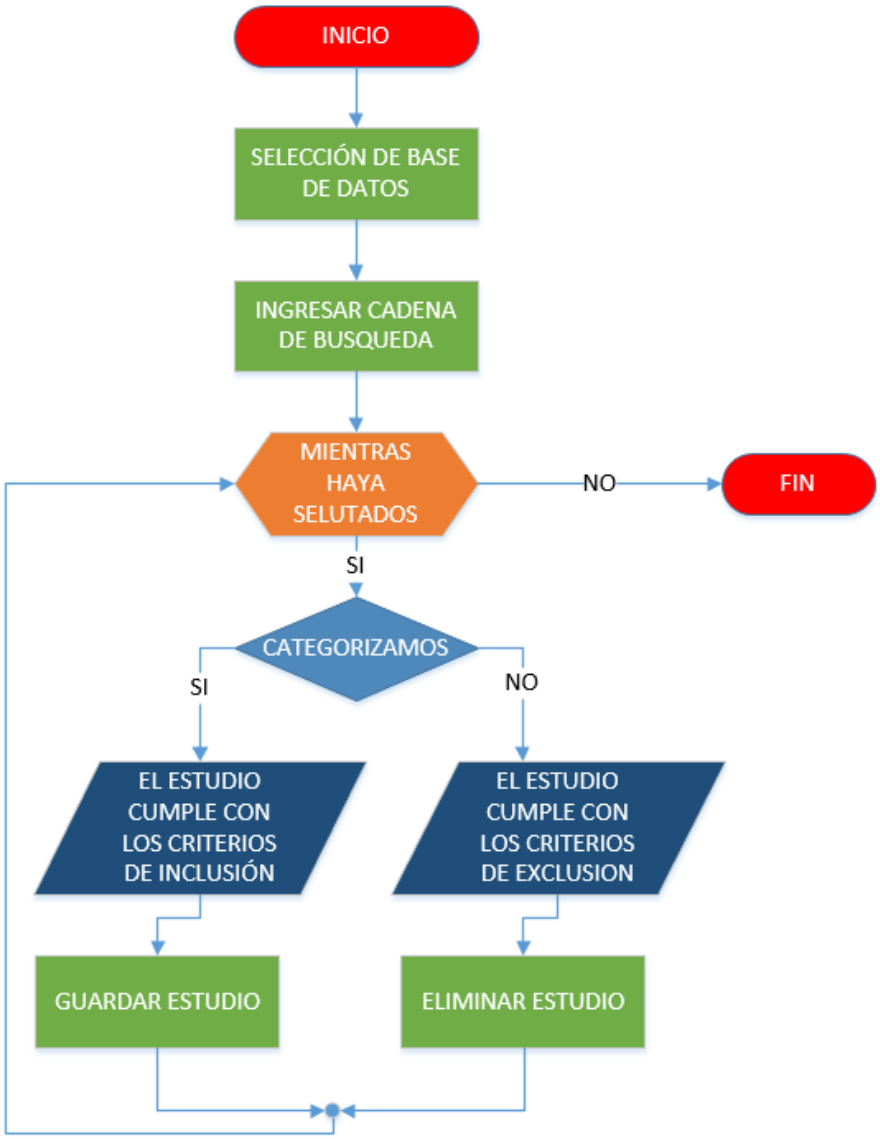


Figura 20: Proceso de selección de estudios incluidos y excluidos

Luego de aplicar las cadenas de búsqueda y obtener los primeros estudios se describe los criterios de inclusión y exclusión que se va a utilizar para la selección de estudios primarios. Estos son:

Criterios de inclusión.

- Artículos científicos publicados a partir del 2013.
- Artículo científico que en el resumen contenga las palabras claves.
- Artículos científicos que sus títulos tengan relación con el tema de investigación

- Artículos científicos que sean del área de las ciencias de la computación
- Artículos que hayan sido citados

Criterios de exclusión.

- Publicaciones informales que no siguen una metodología científica.
- Todas las que no cumplen con los criterios de inclusión.

Definición de tipos de estudio

Los tipos de estudios primarios que van a ser seleccionados durante la ejecución de la revisión sistemática son los artículos presentes en las fuentes seleccionadas, que cumplan con los criterios establecidos y hayan sido obtenidos como producto al aplicar el procedimiento de selección.

Ejecución de la Revisión

Una vez planificada la revisión, en esta sección pasamos a ejecutar la revisión sistemática en cada una de las fuentes seleccionadas aplicando todos los criterios y procedimientos especificados. A continuación tenemos varias secciones en las que se detalla la ejecución en cada una de las fuentes para finalizar con una sección de refinamiento en la que se valida los estudios importantes que hayan podido quedar atrás en estas búsquedas.

5.2.3.1. Ejecución de la selección en la fuente ACM.

Detallamos cómo se realizó la ejecución de la revisión en la fuente ACM en base principalmente a cómo se adaptó la cadena de búsqueda al motor en cuestión, los resultados obtenidos y el análisis de cada estudio primario.

Selección de estudios iniciales

La búsqueda fue realizada mediante la opción de “búsqueda avanzada”, seleccionando las siguientes opciones:

- Búsqueda en: title, abstract y keywords.
- Sources: journals, book series y handbooks.
- Subject: computer science.
- PUBYEAR $>$ 2012

Una vez configurada la búsqueda pasamos a realizar la consulta con las cadenas de búsqueda.

CB01	**(+genetic +algorithms +software +engineering +software +requirements)**
CB02	(+genetic +algorithms +software +requirements)
CB03	(+genetic +algorithms +software +testing)
CB04	(+optimization +computer +systems +genetic +algorithms)

Tabla 6: Cadenas de Búsquedas RRAE

- La ejecución de la primera búsqueda nos da como resultado 153 estudios, después de aplicar el criterio de inclusión nos quedaron 9 documentos relevantes, de los cuales aplicando el criterio de exclusión consideramos los siguientes como estudios primarios

Identificador	Articulo Cientifico
S01	**Minimizing test suites in software product lines using weight-based genetic algorithms** Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference
S02	**Cost-Priority Cognizant Regression testing** Harsh Bhasin Assistant Professor, Department of Computer Science FMIT Jamia Hamdard, Delhi
S03	**UML Modeling of Load Optimization for Distributed Computer Systems Based on Genetic Algorithm** 2013 3rd International Workshop on Replication in Empirical Software Engineering Research, RESER 2013

Tabla 7: Resultados Cadena CB01 ACM

- La ejecución de la segunda búsqueda nos da como resultado 183 estudios, después de aplicar el criterio de exclusión e inclusión nos quedó 1 documentos relevante

Identificador	Articulo Cientifico
S04	**Improved heuristics for solving OCL constraints using search algorithms** 16th Genetic and Evolutionary Computation Conference, GECCO 2014

Tabla 8: Resultados Cadena CB02 ACM

- La ejecución de la primera búsqueda nos da como resultado 289 estudios, después de aplicar el criterio de inclusión nos quedaron 9 documentos relevantes, de los cuales aplicando el criterio de exclusión consideramos los siguientes como estudios primarios

Identificador	Articulo Cientifico
S05	**Critical components testing using hybrid genetic algorithm** ceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation

Tabla 9: Resultados Cadena CB03 ACM

- La ejecución de la primera búsqueda nos da como resultado 1346 estudios, después de aplicar el criterio de inclusión nos quedaron 8 documentos relevantes, de los cuales aplicando el criterio de exclusión consideramos los siguientes como estudios primarios

Identificador	Articulo Cientifico
S06	**Development of a framework for test case prioritization using genetic algorithm** School of Computer Science South China Normal University Guangzhou 510631, China
S07	**Random or Genetic Algorithm Search for Object-Oriented Test Suite Generation** Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation
S08	**Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation**

	17th IEEE International Multi Topic Conference: Collaborative and Sustainable Development of Technologies, IEEE INMIC 2014 - Proceedings

Tabla 10: Resultados Cadena CB04 ACM

Evaluación de la calidad de los estudios

Todos los documentos presentes en la fuente ACM tienen presunción de calidad debido a que para estar publicados en esta fuente han debido pasar una serie de filtros y evaluaciones.

Revisión de la selección

La selección de los estudios primarios realizada ha sido validada de forma que tengamos la seguridad de no haber dejado atrás ningún estudio relevante en esta fuente.

Extracción de información

En esta sección se realiza la extracción de la información relevante de cada uno de los estudios primarios que hemos obtenido.

5.2.3.2. Definición del criterio de inclusión y exclusión de información

Para extraer la información relevante de los estudios primarios según nuestros objetivos, nos centramos principalmente en estudiar las aportaciones de interés que realizan sobre la implementación de los algoritmos genéticos en la ingeniería de software, considerando las siguientes:

- ¿Qué problema soluciona los Algoritmos Genéticos en la ingeniera del software?
- ¿Qué aplicación tienen los Algoritmos Genéticos en la Ingeniería del Software?
- ¿Qué tecnología utilizan los algoritmos genéticos para su ejecución?

Formulario para la extracción de información

El formulario utilizado para documentar la extracción de información realizada sobre cada estudio primario consta de varias partes. Una primera parte de identificación del estudio en la que mostramos el título, publicación, autores y referencia. Otra parte de descripción general en la que analizamos el área de interés sobre las que realiza aportaciones relevantes, así como una breve descripción o resumen del contenido del estudio. Por último, una parte en la que documentamos varios aspectos a destacar, los

cuales se han considerado importantes a medida que se realizaba el análisis del estudio primario.

Identificación	**Título** **Publicación** **Autor** **Referencia**
Descripción	Área Impacto Resumen
Aspectos relevantes	Problema/solución Aplicación de los AG en la IS Tecnología aplicada a los AG

Tabla 11: Formulario

Extracción de resultados objetivos y subjetivos

En este apartado aparece la información extraída de cada estudio primario representada mediante los formularios definidos anteriormente.

CB01 S01

Identificación	
Título	Minimizing Test Suites in Software Product Lines Using Weight-based Genetic Algorithms
Español	La minimización adecuada de prueba en Líneas de Producto Software Usando Algoritmos Genéticos basados en el peso
Publicación	Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference
Autor	Wang Shuai, Ali Shaukat, Gotlieb Arnaud
Referencia	[30]
Año	**2013**
Fecha	**06/07/2013**
Descripción	
Área	Ciencias de la Computación
Impacto	13
Resumen	Las técnicas de minimización tienen por objeto identificar y eliminar los casos de prueba redundantes de las series de pruebas con el fin de reducir el número total de casos para ejecutar, mejorando así la eficiencia de la prueba. Sin embargo, lo que minimiza el banco de pruebas para un producto requiere abordar dos problemas potenciales: 1) El conjunto de pruebas minimizada no puede cubrir todos los requisitos de la prueba en comparación con el paquete de originales. 2) El conjunto de pruebas minimizada puede tener menos capacidad de revelar que el fallo del paquete original. En este artículo aplicamos algoritmos genéticos (gas) basados en el peso para minimizar el conjunto de pruebas que se aplica a un producto, preservando al mismo tiempo la capacidad de detección de fallos y la cobertura de las pruebas
Aspectos Relevantes	
Problema/ Solución	Los casos de prueba redundantes en las líneas de producción de software Aplicación de algoritmos genéticos (gas) basados en el peso para minimizar el conjunto de pruebas aplicado a un producto.
Aplicación de los AG en la IS	Usando Algoritmos Genéticos con función de peso, para minimizar los bancos prueba en la línea de producción de software
Tecnología aplicada a los AG	Se desarrolló una herramienta llamada Import Plugin and Transformation (IPT) (Plugin de importación y transformación) para apoyar esta metodología automatizado de selección de casos de prueba y se ha aplicado en Cisco Systems, Noruega.
Conclusión	En el presente trabajo, hemos propuesto una aplicación de gas con sede en peso para reducir al mínimo el banco de pruebas que se aplica a un producto, al mismo tiempo lograr una cobertura alta de pares de características y capacidad de detección de fallos en el contexto de la línea de productos de software. Los resultados muestran que los algoritmos genético-aleatorios alcanzan los umbrales mencionados, las buscadas aleatorias apenas alcanza los umbrales determinados. Entre todos estos algoritmos, los algoritmos genéticos tiene el mejor rendimiento.
	El objetivo es encontrar un subconjunto mínimo de casos de prueba para aplicar a un producto los cuales se obtienen de todo el conjunto de casos de prueba disponibles para la línea de productos. Al mismo tiempo, lograr una cobertura de características y alta capacidad de detección de fallos.

Tabla 12: Formulario Articulo S01

CB01 S02

Identificación	
Título	Cost-Priority Cognizant Regression testing
Español	Costo-Prioridad Las pruebas de Regresión Consciente
Publicación	Harsh Bhasin Assistant Professor, Department of Computer Science FMIT Jamia Hamdard, Delhi
Autor	Bhasin Harsh
Referencia	[31]
Año	**2014**
Fecha	**20/05/2014**
Descripción	
Área	Ciencias de la Computación
Impacto	2
Resumen	Cualquier cambio en el software requiere de todos los casos de prueba, desarrollados anteriormente, para ser re-ejecutado. Esto se hace para asegurarse de que los cambios no han afectado el correcto funcionamiento del software. Este trabajo propone una función de aptitud, esta función decidiría la prioridad de los casos de prueba. Puesto que es un problema de optimización, los algoritmos genéticos se han utilizado para realizar esta tarea. Con el fin de verificar la técnica, se han utilizado los datos de prueba de cuatro ciclos de prueba de un sistema de planificación de recursos empresariales profesional.
Aspectos Relevantes	
Problema/ Solución	Los casos redundantes en las pruebas de regresión Los algoritmos genéticos se han utilizado para optimizar la priorización de los casos de prueba de regresión
Aplicación de los AG en la IS	En la fase de pruebas se lo implementa en el paquete de pruebas de regresión tiempo consciente y pruebas de regresión costo-consciente.
Tecnología aplicada a los AG	Se ha implementado y aplicado en un sistema de planificación profesional de recursos empresariales (ERP), desarrollado en C #, de alrededor de 8K líneas de código.
Conclusión	La idea de combinar el coste y el tiempo de los paquetes de pruebas, de priorización está determinada por la función de aptitud, es tanto teóricamente sólida y da un mejor valor de porcentaje promedio de detección de fallos (APFD) en comparación con la técnica que consideraba sólo el costo. La técnica propuesta ha sido validada mediante un software profesional, eliminando así las dudas sobre su validez. La técnica se ha verificado mediante la incorporación de pruebas de estrés. El algoritmo de priorización del paquete de casos de prueba puede hacer que las pruebas sean más confiables y reducir las posibilidades de que el software sea propenso a errores, incluso si la prueba se termina abruptamente.
	Los algoritmos genéticos son procesos heurísticos de búsqueda, que nos ayudan a lograr la priorización de las pruebas de una manera más óptima. Se aplican cuando la población se puede pedir en la base de un valor de aptitud y el problema se reduce a un problema de optimización en un gran espacio de búsqueda.

Tabla 13: Formulario Articulo S02

CB01 S03

Identificación	
Título	UML Modeling of Load Optimization for Distributed Computer Systems Based on Genetic Algorithm
Español	UML Modelado de carga Optimización de Sistemas Distribuidos por ordenador basado en Algoritmos Genéticos
Publicación	2013 3rd International Workshop on Replication in Empirical Software Engineering Research, RESER 2013
Autor	Saxena Vipin, Arora Deepak, Mishra Nimesh
Referencia	[32]
Año	**2013**
Fecha	**15/01/2013**
Descripción	
Área	Ciencias de la Computación
Impacto	3
Resumen	La computación distribuida se ha convertido en una de las configuraciones del sistema de red más eficientes para exhibir paralelismo en sistemas de acoplamiento holgado. Estos sistemas se caracterizan por una mayor fiabilidad, disponibilidad, escalabilidad y robustez, destinado a proporcionar computación de alto rendimiento de una manera muy eficiente. La composición de los sistemas distribuidos se compone de varios equipos autónomos que pueden estar dispersos geográficamente e interconectados entre sí para proporcionar una óptima utilización de los recursos. El grado de utilización de los recursos es uno de los criterios clave para evaluar el funcionamiento de tales sistemas. Se propone un enfoque basado en el algoritmo genético de optimización para cargar en un entorno de computación distribuida. Este trabajo de investigación demuestra la implicación de los algoritmos genéticos para optimizar el tiempo de espera total para un conjunto de procesos para ser ejecutado en un conjunto de servidores. Con el fin de comprender la complejidad del diseño, modelamos el enfoque propuesto el uso de clases y diagramas de secuencia UML. Los resultados del modelo propuesto se han encontrado beneficios implementado y probado bajo diferentes escenarios de prueba usando C ++.
Aspectos Relevantes	
Problema/ Solución	La asignación de los recursos en los sistemas distribuidos Implementación de los algoritmos genéticos para optimizar el tiempo de espera en la asignación de recursos en los sistemas distribuidos
Aplicación de los AG en la IS	Optimización de carga en los sistemas distribuidos
Tecnología aplicada a los AG	Se ha diseñado el modelo de clases UML para los algoritmos genéticos y se los ha implementado en el lenguaje C ++
Conclusión	En el presente trabajo, se ha diseñado y desarrollado un enfoque para optimizar el tiempo de espera general para un conjunto de procesos bajo entorno de computación distribuida. El método ha sido probado para diferentes escenarios de población. De los resultados se puede concluir que el enfoque de optimización basado en algoritmos genéticos sería una mejor opción en caso de aplicarse en cualquier escenario de computación distribuida.
	Se ha implementado el enfoque propuesto con el uso de lenguaje de programación C ++ y encontrado resultados óptimos en términos de reducción global de tiempo de espera para un conjunto de genes. Para evaluar el enfoque propuesto, los resultados han sido probados hasta tres niveles anidados de la generación de la población.

Tabla 14: Formulario Articulo S03

CB02 S04

Identificación	
Título	Improved heuristics for solving OCL constraints using search algorithms
Español	Heurística mejorados para la resolución de restricciones OCL utilizando algoritmos de búsqueda
Publicación	16th Genetic and Evolutionary Computation Conference, GECCO 2014
Autor	Ali Shaukat, Iqbal Muhammad Zohaib, Arcuri Andrea,
Referencia	[33]
Año	**2014**
Fecha	**12/07/2014**
Descripción	
Área	Ciencias de la Computacion
Impacto	1
Resumen	El lenguaje de restricciones de objetos (OCL) es un lenguaje estándar para especificar las limitaciones de los modelos Unified Modeling Language (UML). Las restricciones especificadas se pueden utilizar para varios propósitos incluyendo la verificación y el ensayo basado en modelos (por ejemplo, la generación de datos de prueba). En este trabajo, se propone una mejora en la heurística existente para resolver restricciones OCL utilizando algoritmos de búsqueda. Evaluamos nuestra heurística mejorada utilizando dos estudios empíricos con tres algoritmos de búsqueda: Método de variable alterna (AVM), Algoritmo Evolutivo (EA), y un algoritmo genético (GA). El primer estudio empírico se realizó con problemas artificiales cuidadosamente diseñados (restricciones) para evaluar cada heurística individualmente. El segundo estudio empírico se basa en un estudio de caso industriales proporcionado por Cisco sobre pruebas basadas en modelos de sistemas de videoconferencia. Los resultados de ambas evaluaciones empíricas revelan que la eficacia de los algoritmos de búsqueda, se mejora significativamente cuando se utilizan las nuevas heurísticas presentados en este documento. En particular, nuestros experimentos muestran que con la nueva heurística tiene el índice de éxito más alto, ya que requiere el menor número de iteraciones para resolver limitaciones.
Aspectos Relevantes	
Problema/ Solución	Limitaciones en los modelos UML Mejorar la heurística existente para resolver restricciones OCL. utilizando algoritmos de búsqueda
Aplicación de los AG en la IS	En los modelos UML aplicado a las diferentes clases Diagramas
Tecnología aplicada a los AG	El estudio es de un caso industrial parte de un proyecto destinado a apoyar la automatización y comprobación de la solidez basado en el modelo de un subsistema central de videoconferencia (VCS) llamado Saturno desarrollado por Cisco Systems, Inc, Noruega
Conclusion	Los resultados de los estudios empíricos muestran que las nuevas heurísticas conducen a un rendimiento significativamente mejor para cada algoritmo.
	Ejecutamos cada uno de los cuatro algoritmos de 100 veces con ambas mensajerías instantáneas y secundarias para los 28 problemas. Dejamos que los algoritmos se ejecutan hasta 10.000 evaluaciones de aptitud en cada problema y se recogieron datos sobre si los algoritmos encontraron soluciones para mensajería instantánea y secundaria. Se utilizó un PC con procesador Intel Core Duo CPU 2,20 GHz con 4 GB de RAM, que ejecuta el sistema operativo Microsoft Windows 7 para la ejecución del experimento.

Tabla 15: Formulario Articulo S04

CB03 S05

Identificación	
Título	Critical components testing using hybrid genetic algorithm
Español	Prueba críticas de componentes que utilizan Algoritmo Genético Híbrido
Publicación	Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation
Autor	Jeya Mala D., Sabari Nathan K., Balamurugan S.
Referencia	[34]
Descripción	
Área	Ciencias de la Computación
Impacto	1
Resumen	Dado que la calidad del software desempeña un papel vital en los sistemas de tiempo real, es esencial para identificar las partes cruciales del sistema y para probar de manera efectiva. En el enfoque propuesto, los componentes críticos se identifican por medio de análisis de impacto basados en mutación. La siguiente tarea es poner a prueba los componentes críticos utilizando el algoritmo genético híbrido (HGA) generación de casos de prueba basada en la optimización y el enfoque. Los mutantes se generan de forma automática mediante la siembra de fallos en cada método de todos los componentes en el software bajo prueba (SUT). El conjunto inicial de casos de prueba se genera a partir de datos de prueba aleatorios. Los casos de prueba generados se ejecutan sobre el original y el mutante para identificar si el caso de prueba detecta el error o no. Basándose en los resultados, la puntuación de mutaciones (MS) se calcula, que siempre se encuentra entre 0 y 1. Los mejores casos de prueba se eligen basándose en el que tiene puntuaciones de mutación más altas y se ejecutan en los mutantes para analizar cómo cada componente afecta a los otros componentes de la SUT. Con base en el análisis, se identifican los componentes críticos y necesitan pruebas rigurosas utilizando los casos de prueba generados por el HGA. En la unidad de pruebas, los casos de prueba se ejecutan con el original para el mutante. La optimización de caso de prueba se realiza mediante la evaluación de la efectividad de los conjuntos de pruebas utilizando la puntuación de mutaciones y el Poder de Valor de cobertura (BCV). Los resultados trazados se comparan con los resultados esperados que se almacenaron previamente en el repositorio y los estados se actualizan. La eficacia del enfoque propuesto se compara con el algoritmo genético (GA) y llegamos a la conclusión de que el tamaño de la de prueba final y el tiempo total de ejecución se reducen en el enfoque propuesto.
Aspectos Relevantes	
Problema/ Solución	Los componentes críticos de prueba Optimización basada en Algoritmos Genéticos Híbridos
Aplicación de los AG en la IS	Inicialmente se aplican en las pruebas al azar para obtener los primero resultados y así poder mejorar los casos de pruebas de componentes.
Tecnología aplicada a los AG	Sobre la base de esta investigación, se ha desarrollado una herramienta de prueba JImpact Arbiter la misma que se encuentra realizada en la plataforma Java que puede generar automáticamente mutantes para analizar el Impacto de los componentes.
Conclusión	El nivel de impacto se clasifica como Catastrófico, Crítico, Marginal o Menor. Basándose en el análisis de impacto, se identifica una lista completa de componentes críticos y se prueban utilizando casos de prueba generados por HGA. La prueba unitaria y la prueba en par se realizan para todos los componentes del SUT. Para la optimización, evaluamos la puntuación de mutación y el valor de cobertura para los casos de prueba. Finalmente, en base a los resultados, la eficiencia de HGA y GA se compara y se representa en forma de gráficos para hacer una evaluación sobre el enfoque propuesto.
	La creación de la primera serie de casos de prueba al azar es muy fácil, pero para realizar la mejora de la calidad en dichos casos es necesario mucho esfuerzo. Mediante el uso de GA, podemos automatizar este proceso de mejoramiento mediante la creación de mutaciones de los casos. El conjunto básico de casos de prueba lleva la información que puede ser optimizado y las mutaciones de los casos de prueba.

Tabla 16: Formulario Articulo S05

CB04 S06

Identificación	
Título	Development of a framework for test case prioritization using genetic algorithm
Español	Desarrollo de un marco para la priorización de casos de prueba Utilizando Algoritmos Genéticos
Publicación	School of Computer Science South China Normal University Guangzhou 510631, China
Autor	Malhotra Ruchika, Tiwari Divya
Referencia	[22]
Año	**2013**
Fecha	**11/15/2013**
Descripción	
Área	Ciencias de la Computación
Impacto	1
Resumen	Pruebas de software es una parte que consume tiempo y esfuerzo del ciclo de vida del software de desarrollo. Volver a probar una aplicación de software durante la fase de mantenimiento, con todo el conjunto de pruebas y casos de prueba adicionales para las modificaciones en el software, dentro del presupuesto y el tiempo, es un reto para los probadores de software. El caso de priorización de pruebas es una tarea esencial que reduce el esfuerzo en la fase de mantenimiento en un grado considerable. En este artículo se propone un framework para la priorización de casos de prueba utilizando una herramienta basada en un algoritmo genético, desarrollado en lenguaje Java. El marco también pone de relieve la necesidad y las ventajas de utilizar la nueva métrica APBCm como función de evaluación de la aptitud en el GA. Varios factores que pueden ser utilizados para incrustar el conocimiento acerca de la significación de los bloques en el APBCm métrica también se han discutido. Sin embargo, el cálculo exacto de los pesos teniendo en cuenta todos los factores mencionados, sigue siendo un desafío abierto. El enfoque, que se presenta aquí, tiene su aplicación en las áreas de específica de caso de prueba de priorización, sino también en otro tipo de pruebas con ligeras modificaciones. Teniendo en cuenta factores de ponderación prácticos es un concepto general que puede ayudar a mejorar el costo de las pruebas de regresión y también se puede extender a los problemas de pruebas de minimización y pruebas de regresión.
Aspectos Relevantes	
Problema/ Solución	La priorización de los casos de prueba en la fase de mantenimiento Framework para la Priorización de casos de prueba utilizando algoritmos genéticos
Aplicación de los AG en la IS	Su aplicación se la realiza en la fase de mantenimiento realizando la priorización de las pruebas
Tecnología aplicada a los AG	Se ha desarrollado en framework que emplea un instrumento de medida caso priorización basada en un algoritmo genético (GA), desarrollado en el lenguaje Java utilizando Eclipse
Conclusión	El enfoque, que se presenta en este artículo, tiene su aplicación en las áreas de la priorización de casos de prueba específicos, pero también puede generalizarse con ligeras modificaciones. Considerar factores de peso prácticos es un concepto general que puede ayudar a mejorar el costo de las pruebas de regresión y también puede extenderse a los problemas de minimización de conjuntos de pruebas y selección de pruebas de regresión.
	Se trata de una investigación que se lleva a cabo para facilitar a los interesados la información sobre la calidad del producto / servicio que se está probando y la priorización de las pruebas de regresión que se deben aplicar. El problema de priorización de casos de prueba se define como: Dado un conjunto de pruebas 'T', un conjunto 'PT' de todas las permutaciones de T y la función 'f' 'que mapea PT a números reales, la técnica de priorización de casos de prueba tiene como objetivo encontrar un T'∈ PT tal que (∀T') (T"∈ PT) [f (T ') ≥ f (T' ')].

Tabla 17: Formulario Articulo S06

CB04 S07

Identificación	
Título	Random or Genetic Algorithm Search for Object-Oriented Test Suite Generation
Español	Algoritmo Aleatorio o Genético de Búsqueda Para la Generación de conjuntos de pruebas orientada a objetos
Publicación	Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation
Autor	Shamshiri Sina, Rojas José Miguel, Fraser Gordon, Mcminn Phil, Court Regent
Referencia	[35]
Año	**2015**
Fecha	**11/07/2015**
Descripción	
Área	Ciencias de la Computación
Impacto	1
Resumen	Alcanzar una alta cobertura estructural es un objetivo importante en las pruebas de software. Varias técnicas basadas en la búsqueda han resultado exitosas en la generación automática de pruebas que logran una alta cobertura. Sin embargo, a pesar de los argumentos bien establecidos detrás de la utilización de algoritmos de búsqueda evolutiva (por ejemplo, algoritmos genéticos) en lugar de la búsqueda aleatoria, sigue siendo una pregunta abierta si los beneficios pueden ser observados en la práctica al generar conjuntos de pruebas unitarias para clases orientadas a objetos. En este artículo se presenta un estudio empírico sobre los efectos de la utilización de un algoritmo genético (GA) para generar conjuntos de pruebas de forma incremental con búsqueda aleatoria, aplicando el generador de conjuntos de pruebas unitarias EvoSuite a 1,000 clases seleccionadas aleatoriamente de la SF110 corpus de proyectos de código abierto.
Aspectos Relevantes	
Problema/ Solución	Prueba unitarias en las clases orientadas a objetos Algoritmo genético en las pruebas unitarias
Aplicación de los AG en la IS	La generación automática de pruebas unitarias en las clases orientado a objetos
Tecnología aplicada a los AG	La generación de prueba se la implemento en la herramienta EvoSuite, que genera las ramas que cubre los conjuntos de pruebas para las clases de Java. Se realizaron experimentos en una muestra aleatoria de 1.000 clases del corpus SF110 de proyectos de código abierto
Conclusión	Los resultados muestran poca diferencia entre la cobertura alcanzada por los conjuntos de pruebas generados con la búsqueda evolutiva en comparación con los generados mediante la búsqueda aleatoria. Un análisis detallado revela que el algoritmo genético cubre más ramas del tipo donde las funciones estándar de la aptitud proporcionan la dirección.
	En este trabajo, estudiamos la aplicación de la búsqueda aleatoria y GA para la generación automática de conjuntos de pruebas, tal como se implementa en la herramienta EvoSuite, tiene como objetivo generar conjuntos de pruebas unitarias que cubran tantas ramas de una clase Java como sea posible, mientras que también ejecuta todos los métodos que están desprovistos de ramas, denominados métodos aleatorios.

Tabla 18: Formulario Articulo S07

CB04 S08

Identificación	
Título	Minimizing feature model inconsistencies in software product lines
Español	Reducir al mínimo las incoherencias de operaciones del modelo de Líneas de Producto Software
Publicación	17th IEEE International Multi Topic Conference: Collaborative and Sustainable Development of Technologies, IEEE INMIC 2014 - Proceedings
Autor	Afzal Uzma, Mahmood Tariq, Rauf Imran, Shaikh Zubair Ahmed
Referencia	[36]
Año	**2015**
Fecha	**15/05/2015**
Descripción	
Área	Ciencias de la Computación
Impacto	1
Resumen	La Línea de productos de software (SPL) es una metodología de ingeniería de software para crear y administrar una familia de productos de software similares mediante el uso de modelos de características reconfigurables. En un SPL a gran escala, la selección del conjunto relevante de características para configurar un producto dado es un reto clave, cada unidad de software está configurada por un conjunto de características y las características de combinación de cada unidad pueden generar inconsistencias que se resuelven mediante deliberación manual entre los sistemas diseñadores, conduciendo a la pérdida posible de recursos de negocio valiosos. En este trabajo, empleamos Algoritmos Genéticos (AG) para minimizar tres inconsistencias de modelo de entidad primaria, es decir, obligatorias, inclusivas y exclusivas / alternativas, con una función cruzada dispersa y una tasa de mutación del 1%. Utilizando modelos de características del mundo real de un SPL de teléfono inteligente local, optimizamos un modelo de características a pequeña escala (que contiene 100 características) y dos de gran escala (que contienen 500 y 1000 características) y muestran que GA puede ser de 95-97 % Consistente (libre de conflictos) modelos de características en tiempos drásticamente reducidos en comparación con técnicas de resolución de conflictos manuales. También mostramos que una función cruzada dispersa produce mejores resultados que las funciones de punto único o multipunto.
Aspectos Relevantes	
Problema/ Solución	Selección de características para la configuración de un productos Optimización el proceso de selección de las características mediante algoritmos genéticos.
Aplicación de los AG en la IS	En el modelado de las características para la producción de un producto en la línea de producción de software
Tecnología aplicada a los AG	
Conclusión	Nuestros resultados empíricos muestran que para una configuración de pequeña escala GA produce una solución óptima y totalmente consistente. A medida que aumenta la magnitud de los productos, la optimización de las soluciones GA disminuye ligeramente (1-3%). Los resultados también muestran que aumenta un poco la tasa de mutación mejora la optimización.
	Los modelado de características son la técnica de SPL común para la producción de un producto válido, que puede clasificar de acuerdo con su dependencia de uno a otro y características independientes, su tipo también puede ser derivado de las restricciones, es decir, normas y reglamentos que rigen una configuración de producto válida.

Tabla 19: Formulario Articulo S07

5.2.3.3. Ejecución de la selección en la fuente IEEE Digital Library

En este apartado detallamos el proceso realizado en la ejecución de la revisión en la fuente IEEE.

Selección de estudios iniciales

La búsqueda fue realizada mediante la opción de "búsqueda avanzada", seleccionando las siguientes opciones:

- Búsqueda en: title, abstract y keywords (comando tak).
- Sources: journals, book series y handbooks.
- Subject: computer science.
- PUBYEAR $>$ 2012

Una vez configurada la búsqueda pasamos a realizar la consulta con las cadenas de búsqueda.

Identificador	Cadena de Búsqueda
CB05	(("Abstract":software engineering) AND "Abstract":genetic algorithms)
CB06	(("Abstract":genetic algorithms) AND "Abstract":software requirements)
CB07	(("Abstract":genetic algorithms) AND "Abstract":software requirements)

Tabla 20: Cadenas de Búsquedas IEEE

- La ejecución de la primera búsqueda nos da como resultado 172 estudios, después de aplicar el criterio de inclusión nos quedaron 10 documentos relevantes, de los cuales aplicando el criterio de exclusión consideramos los siguientes como estudios primarios

Identificador	Articulo Cientifico
S09	**A Dynamic Approach for Retrieval of Software Components Using Genetic Algorithm** Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on

Tabla 21: Resultados Cadena CB05 ACM.

La ejecución de la segunda búsqueda nos da como resultado 114 estudios, después de aplicar el criterio de inclusión nos quedaron 10 documentos relevantes, de los cuales aplicando el criterio de exclusión consideramos los siguientes como estudios primarios

Identificador	Articulo Cientifico
S10	**Component-Based Software System Test Case Prioritization with Genetic Algorithm Decoding Technique Using Java Platform** 2015 Fifth International Conference on Intelligent Systems Design and Engineering Applications

Tabla 22: Resultados Cadena CB06 ACM.

- La ejecución de la segunda búsqueda nos da como resultado 114 estudios, después de aplicar el criterio de inclusión nos quedaron 10 documentos relevantes, de los cuales aplicando el criterio de exclusión consideramos los siguientes como estudios primarios

Identificador	Articulo Cientifico
S11	**Optimization of soft cost estimation using genetic algorithm for NASA software projects** 2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW)
S12	**Software quality assurance for object-oriented systems using meta-heuristic search techniques** Department of Computer Science and Engineering BMS Institute of Technology Management Bengaluru - 560 064, India
S13	**A novel approach for test case generation from UML activity diagram** 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)

Tabla 23: Resultados Cadena CB07 ACM.

Evaluación de la calidad de los estudios

Todos los documentos presentes en la fuente ACM tienen presunción de calidad debido a que para estar publicados en esta fuente han debido pasar una serie de filtros y evaluaciones.

Revisión de la selección

La selección de los estudios primarios realizada ha sido validada de forma que tengamos la seguridad de no haber dejado atrás ningún estudio relevante en esta fuente.

Extracción de información

En esta sección se realiza la extracción de la información relevante de cada uno de los estudios primarios que hemos obtenido.

5.2.3.4. Definición del criterio de inclusión y exclusión de información

Para extraer la información relevante de los estudios primarios según nuestros objetivos, nos centramos principalmente en estudiar las aportaciones de interés que realizan sobre la implementación de los algoritmos genéticos en la ingeniería de software, se consideran los puntos anteriores en el inciso 5.2.3.2

Extracción de resultados objetivos y subjetivos

En este apartado aparece la información extraída de cada estudio primario representada mediante los formularios definidos anteriormente.

CB05 S09

Identificación	
Título	A Dynamic Approach for Retrieval of Software Components Using Genetic Algorithm
Español	Un enfoque dinámico para la recuperación de los componentes de software por medio de algoritmos genéticos
Publicación	Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on
Autor	Vodithala Swathy,
Referencia	[37]
Año	**2015**
Fecha	**23/09/2015**
Descripción	
Área	Ciencias de la Computación
Impacto	1
Resumen	Reutilización de software es una de las áreas de investigación más importantes en la ingeniería de software basado en componentes (CBSE). Es un área que integra todas las otras áreas técnicas como la minería de datos, la inteligencia artificial, etc. Las principales áreas que van a ser utilizadas en la reutilización del software son la clasificación, agrupación, la búsqueda, indexación y recuperación de componentes de software. Hay muchas técnicas descritas en la literatura, cada uno con sus ventajas y limitaciones. En este trabajo, se propone un enfoque dinámico en el que los componentes se recuperan mediante el uso de algoritmos genéticos. El algoritmo propuesto se centra principalmente en la condición de terminación o criterios de detección del Algoritmo Genético, que cambia dinámicamente en función de la consulta del usuario. El Algoritmo Genético (GA), alcanza la convergencia cuando el valor medio de la aptitud de la población se hace igual al valor de la aptitud de la consulta del usuario.
Aspectos Relevantes	
Problema/ Solución	Reutilización de Software Recuperación mediante el uso de algoritmos genéticos
Aplicación de los AG en la IS	La ingeniería de software basada en componentes (CBSE)
Tecnología aplicada a los AG	Se realiza el almacenamiento de componente de software en un repositorio, que pueden ser un código, procedimiento, módulo, subsistema, requisitos, diseño o incluso la documentación.
Conclusión	Este algoritmo da buenos resultados a pesar de que no está integrado con alguna otra técnica, porque el propio GA converge cuando los componentes necesarios de usuario se buscaron, ya que este cambia dinámicamente en función de las consultas del usuario.
	El algoritmo propuesto tiene una sola fase, es decir, la aplicación de solamente GA. En este método propuesto el valor de la aptitud de la consulta del usuario se calcula cada vez que el usuario entra y se compara con el valor de la aptitud de los componentes almacenados. El objetivo principal de este algoritmo se basa en la condición de terminación de GA es decir, GA llega a la convergencia cuando el valor medio de la aptitud de la población es igual al valor de la aptitud de la consulta del usuario.

Tabla 24: Forulario Articulo S09

CB06 S10

Identificación	
Título	Component-Based Software System Test Case Prioritization with Genetic Algorithm Decoding Technique Using Java Platform
Español	Software basado en componentes de casos de prueba con el sistema de priorización con algoritmo genético de decodificación técnica que utiliza la plataforma Java
Publicación	2015 Fifth International Conference on Intelligent Systems Design and Engineering Applications
Autor	Mahajan Surendra, Joshi Shashank D., Khanaa, V.
Referencia	[38]
Año	**2015**
Fecha	**26/02/2015**
Descripción	
Área	Ciencias de la Computación
Impacto	2
Resumen	La priorización de casos de prueba incluye experimentos de prueba que construye la viabilidad en el cumplimiento de algunos objetivos. La importancia entre los objetivos de prueba más imperativos es la rápida tasa de reconocimiento de fallas. Los casos de prueba deben ejecutarse en una solicitud que amplía la probabilidad de descubrimiento de fallos, además, que detecta los problemas más graves en la primera fase de la prueba de ciclo de vida. En este trabajo, desarrollamos y probamos la necesidad del Framework de priorización de pruebas de Software basado en Componentes que planea descubrir fallos más extremos en una etapa temprana y mejorar la calidad del producto de software utilizando Algoritmo Genético (GA) con la técnica de decodificación java.
Aspectos Relevantes	
Problema/ Solución	Pruebas de regresión Framework de priorización de pruebas de Software basado en Componentes
Aplicación de los AG en la IS	La ingeniería de software basada en componentes (CBSE)
Tecnología aplicada a los AG	Se ha desarrollado un framework con la técnica de decodificación java.
Conclusión	La técnica propuesta dio la mejor tasa para el reconocimiento de errores graves a través de los resultados. Como los que se dan por la ejecución actual de la prueba piloto, podemos concluir que, en el enfoque actual podemos reducir el número de casos de prueba en general. Basado en el análisis de ejecución KAM, la estrategia propuesta es viable da prioridad a los casos de prueba y reduce el costo de tiempo de ejecución de cualquier tipo de proyecto de software de la industria del software.
	Este trabajo se demuestra cómo las pruebas de software pueden ser eficientes con la administración del factor de integridad de los datos importantes para evitar problemas de seguridad. Una de las principales ventajas de nuestro enfoque es que la semántica específica del dominio puede integrarse con la priorización de los casos de prueba de calidad de datos, pudiendo así descubrir problemas de calidad de datos de alimentación de prueba más allá de las medidas de calidad convencionales.

Tabla 25: Forulario Articulo S10

CB07 S11

Identificación	
Título	Optimization of soft cost estimation using genetic algorithm for NASA software projects
Español	Optimización de Estimación de Costos suave usando Algoritmos Genéticos para la NASA Proyectos de Software
Publicación	2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW)
Autor	Algabri Mohammed, Saeed Fahman, Mathkour Hassan, Tagoug Nejmeddine
Referencia	[39]
Año	2015
Fecha	17/02/2015
Descripción	
Área	Ciencias de la Computación
Impacto	1
Resumen	La Estimación de costos de software realiza una estimación aproximada del costo y tiempo requerido para completar el proyecto con éxito. La estimación de costos suele medirse en términos de esfuerzo. El Modelo de Coste Constructivo (COCOMO) es uno de los modelos más importantes para la estimación de costos de software. En este trabajo el algoritmo genético se ha utilizado para ajustar los parámetros del modelo COCOMO para predicar el costo del software con mayor precisión. Además, se utilizó el rendimiento de los métodos propuestos comparados con el conjunto de datos de la NASA.
Aspectos Relevantes	
Problema/ Solución	Estimación de costos con modelo COCOMO Ajuste de los parámetros de los coeficientes de COCOMO usando algoritmos genéticos
Aplicación de los AG en la IS	Gestión de proyectos de software
Tecnología aplicada a los AG	Los experimentos se realizaron con 93 proyectos de la NASA
Conclusión	Los resultados están dando un tiempo de desarrollo más realista en comparación con el tiempo de desarrollo. Estos resultados precisos ayudan al administrador de software a administrar los recursos de las empresas.
	Los Algoritmos Genéticos son la técnica utilizada para la optimización. En esta investigación, GA se utiliza para optimizar la COCOMO II mediante la regulación de los coeficientes de forma automática.

Tabla 26: Forulario Articulo S11

CB07 S12

Identificación	
Título	Software quality assurance for object-oriented systems using meta-heuristic search techniques
Español	Aseguramiento de la calidad del software para los sistemas orientados a objetos utilizando técnicas de búsqueda de meta-heurísticos
Publicación	Department of Computer Science and Engineering BMS Institute of Technology & Management Bengaluru - 560 064, India
Autor	A Suresh Yeresime.
Referencia	[40]
Año	2015
Fecha	25/05/2015
Descripción	
Área	Ciencias de la Computación
Impacto	3
Resumen	Identificar los módulos propensos a fallos en la etapa muy temprana del ciclo de vida de desarrollo de software es muy necesario. Esto ayuda a los desarrolladores de software a concentrarse más en la garantía de calidad, utilizar la energía del hombre en la perspectiva adecuada y principalmente reducir el coste de eliminación de fallos para ser curado en el sistema de software que se está desarrollando. En la literatura se encuentra que numerosos autores han elaborado marcos de evaluación basados en costos para encontrar la efectividad del modelo de predicción de fallas propuesto basado en la aplicación de modelos de redes neuronales, pero se enfatiza menos en la forma en que las redes son entrenadas, en un enfoque elemental de asignar pesos aleatorios a los nodos que siguen. Este artículo evalúa la capacidad de las técnicas de búsqueda metaheurística en la clasificación de fallos de software para el Framework de Integración de Apache. El algoritmo genético (GA) y optimización de enjambre de partículas (PSO) se acopla con la red neuronal para el diseño de modelos de predicción. Se observa que el modelo modificado de Neuronas es más eficaz y eficiente en la clasificación de fallas con precisión en comparación
Aspectos Relevantes	
Problema/ Solución	Calidad de software para sistemas orientada a objetos Utilización de técnicas de búsquedas para la optimización de software de predicción de fallos
Aplicación de los AG en la IS	Predicción de fallos durante el ciclo de vida del desarrollo de software
Tecnología aplicada a los AG	Las herramientas Chidamber y Kemerer Java Metrics (CKJM) extraen métricas orientadas a objetos procesando el código de bytes de las clases Java compiladas. Los programas se ejecutaron especificando los archivos de clase (archivos jar / class) en su línea de comandos o entrada estándar
Conclusión	Los algoritmos de optimización se han utilizado en la asignación de los pesos para los nodos de las capas durante la fase de aprendizaje. Se observa que el modelo modificado de Neuronas es más eficaz y eficiente en la clasificación de fallas con precisión en comparación
	CKJM herramienta se utiliza para extraer los valores de la métrica de 965 clases en el marco de la integración de Apache.

Tabla 27: Forulario Articulo S12

CB07 S13

Identificación	
Título	A novel approach for test case generation from UML activity diagram
Español	Un nuevo enfoque para la generación de casos de prueba de UML Diagrama de actividad
Publicación	2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)
Autor	Jena Ajay Kumar, Swain Santosh Kumar, Mohapatra Durga Prasad
Referencia	[41]
Año	**2014**
Fecha	**07/02/2014**
Descripción	
Área	Ciencias de la Computación
Impacto	1
Resumen	Las pruebas de software se dividen principalmente en tres tipos, es decir, las pruebas basadas en código, las pruebas basadas en la especificación y las pruebas basadas en modelo. En las pruebas basadas en el modelo, la prueba se puede iniciar desde el proceso de diseño en la fase de inicio. Por lo tanto, la detección temprana de fallos se puede lograr mediante el uso de este enfoque. Un enfoque para la generación de casos de prueba a partir de UML (Unified Modelling Language) diagrama de actividad utilizando el algoritmo genético se ha presentado en este documento. Detección precoz de fallos se puede conseguir mediante este método y sin duda reducir el tiempo, costo y el esfuerzo de los desarrolladores en gran medida. Proponemos un modelo para generar una tabla de flujo de Actividad (AFT) a partir de un diagrama de actividad y luego convertirlo a actividad de flujo Gráfico (AFG), los criterios de cobertura son muy importantes en la generación de casos de prueba. Mediante el uso de la actividad criterio de cobertura se recorre la AFG y se generan las trayectorias de prueba final, que generan los casos de prueba a partir de estos caminos. Algoritmo Genético se ha aplicado para generar casos de prueba y también para optimizarlos. El modelo se implementa en un estudio de caso de sistema de retiro en cajero automático.
Aspectos Relevantes	
Problema/ Solución	Pruebas basadas en modelos Aplicación de algoritmos genéticos para la generación de los casos de pruebas
Aplicación de los AG en la IS	Las pruebas basadas en el modelo a partir del diagrama de actividades UML
Tecnología aplicada a los AG	En este trabajo se ha utilizado las características de UML 2.0
Conclusión	El modelo propuesto se refiere a los nodos máximos de actividad en el gráfico. Se aplica en muchos diferentes diagramas de actividad de diferentes dominios. Además, también se reducen los casos de prueba. El enfoque no está automatizado. Así que una herramienta automatizada puede ser desarrollada para el enfoque propuesto. Este enfoque puede ser extendido por otros diagramas de UML con fines de generación de casos de prueba.
	En el enfoque de este artículo se consideró el Diagrama de Actividad para el sistema de retiro del cajero automático. Utilizamos diagrama de actividad debido al comportamiento dinámico de su modelado. Los aspectos dinámicos del objeto se pueden construir, visualizados, especificados y documentados por el diagrama de actividad

Tabla 28: Forulario Articulo S12

5.2.3.5. Ejecución de la selección en la fuente Scopus}

En este apartado detallamos el proceso realizado en la ejecución de la revisión en la fuente Scopus.

Selección de estudios iniciales

La búsqueda fue realizada mediante la opción de "búsqueda avanzada", seleccionando las siguientes opciones:

- Búsqueda en: title, abstract y keywords (comando tak).
- Sources: journals, book series y handbooks.
- Subject: computer science.
- PUBYEAR $>$ 2012

Una vez configurada la búsqueda pasamos a realizar la consulta con las cadenas de búsqueda.

Identificador	Cadena de Búsqueda
CB08	(TITLE-ABS-KEY (optimization software engineering) AND TITLE-ABS-KEY (genetic algorithms)) AND PUBYEAR > 2012 AND (LIMIT-TO (SUBJAREA , "COMP"))
CB09	(TITLE-ABS-KEY (genetic algorithms) AND TITLE-ABS-KEY (software requirements) AND TITLE-ABS-KEY (software design)) AND PUBYEAR > 2012 AND PUBYEAR < 2017 AND (LIMIT-TO (SUBJAREA , "COMP")
CB010	(TITLE-ABS-KEY (genetic algorithms) AND TITLE-ABS-KEY (web software) OR TITLE-ABS-KEY (software patron)) AND PUBYEAR > 2012 AND (LIMIT-TO (SUBJAREA , "COMP"))
CB011	(TITLE-ABS-KEY (genetic algorithms) AND TITLE-ABS-KEY (engineering phase of the software development)) AND PUBYEAR > 2012

Tabla 29: Cadenas de Búsquedas Scopus

- La ejecución de la primera búsqueda nos da como resultado 1052 estudios, después de aplicar el criterio de inclusión nos quedaron 19 documentos relevantes, de los

cuales aplicando el criterio de exclusión consideramos los siguientes como estudios primarios

Identificador	Articulo Cientifico
S14	**A novel strategy for automatic test data generation using soft computing technique** Computer Science and Engineering Department, Thapar University, Patiala, India.
S15	**Prioritization of test scenarios using hybrid genetic algorithm based on UML activity diagram** College of Computer Science and Technology Nanjing University of Aeronautics and Astronautics, Nanjing, China Corresponding
S16	**Efficient parallel evolutionary algorithms for deadline-constrained scheduling in project management** Universidad de la República, Herrera y Reissig 565, Montevideo, 11300, Uruguay

Tabla 30: Resultados Cadena CB08 Scopus.

- La ejecución de la segunda búsqueda nos da como resultado 334 estudios, después de aplicar el criterio de inclusión nos quedaron 13 documentos relevantes, de los cuales aplicando el criterio de exclusión consideramos los siguientes como estudios primarios

Identificador	Articulo Cientifico
S17	**Automatic generation of basis test paths using variable length genetic algorithm** Department of Mathematics and Computer Science, Faculty of Science, Beni-Suef University, Beni-Suef, Egypt
S18	**Retrieving sequence diagrams using genetic algorithm** Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on

Tabla 31: Resultados Cadena CB09 Scopus.

- La ejecución de la tercera búsqueda nos da como resultado 1124 estudios, después de aplicar el criterio de inclusión nos quedaron 10 documentos relevantes, de los cuales aplicando el criterio de exclusión consideramos los siguientes como estudios primarios

Identificador	Articulo Cientifico
S19	**Methods for cost estimation in software project management** International Conference on Applied Sciences 2015, ICAS 2015; Military Economics Academy of WuhanWuhan; China; 3 June 2015 through 5 June 2015

Tabla 32: Resultados Cadena CB10 Scopus.

- La ejecución de la cuarta búsqueda nos da como resultado 148 estudios, después de aplicar el criterio de inclusión nos quedaron 4 documentos relevantes, de los cuales aplicando el criterio de exclusión consideramos los siguientes como estudios primarios

Identificador	Articulo Cientifico
S20	**Predicting project effort intelligently in early stages by applying genetic algorithms with neural networks** Business School of Central South University, Changsha, 410083, China

Tabla 33: Resultados Cadena CB11 Scopus.

Evaluación de la calidad de los estudios

Todos los documentos presentes en la fuente Scopus tienen presunción de calidad ya que para estar publicados en esta fuente han debido pasar una serie de filtros y evaluaciones.

Revisión de la selección

La selección de los estudios primarios realizada ha sido validada de forma que tengamos la seguridad de no haber dejado atrás ningún estudio relevante en esta fuente.

Extracción de información

En esta sección se realiza la extracción de la información relevante de cada uno de los estudios primarios que hemos obtenido.

5.2.3.6. Definición del criterio de inclusión y exclusión de información

Para extraer la información relevante de los estudios primarios según nuestros objetivos, nos centramos principalmente en estudiar las aportaciones de interés que realizan sobre la implementacion de los algoritmos genéticos en la ingeniería de software, se consideran los puntos anteriores en el inciso 5.2.3.2

Extracción de resultados objetivos y subjetivos

En este apartado aparece la información extraída de cada estudio primario representada mediante los formularios definidos anteriormente.

CB08 S14

Identificación	
Título	A novel strategy for automatic test data generation using soft computing technique
Español	Una nueva estrategia para la generación de datos de prueba automático utilizando la técnica de soft computing
Publicación	Computer Science and Engineering Department, Thapar University, Patiala, India
Autor	Chawla P, Chana I, Rana A
Referencia	[42]
Año	2015
Fecha	03/01/2015
Descripción	
Área	Ciencias de la Computación
Impacto	1
Resumen	Las pruebas de software es uno de los aspectos más cruciales para asegurar que el software desarrollado cumple normas de calidad. El proceso de desarrollo de software invierte al menos el 50% del coste total en el proceso de pruebas de software. El diseño de datos de prueba de software es una actividad importante y difícil debido a la estructura no lineal de software. Por otra parte, el tipo de caso de prueba y el alcance determina la calidad de los datos de prueba. Para abordar esta cuestión, las herramientas de prueba de software deben emplear la inteligencia soft computing basado técnicas como la optimización de enjambre de partículas (PSO) y algoritmo genético (GA) para generar la prueba inteligente de forma automática. Este artículo presenta un PSO híbrido y heurística basada en GA para la generación automática de conjuntos de pruebas. En este artículo, describimos el diseño y la aplicación de la estrategia propuesta y evalua nuestro modelo mediante la realización de experimentos con diez clases de contenedores desde el estándar de Java biblioteca. Se analizaron estadísticamente con nuestro algoritmo de adecuación de prueba como criterio de cobertura de sucursales. El criterio se toma como porcentaje de cobertura por unidad de tiempo y el porcentaje de fallos detectados por los datos de prueba generados. Hemos comparado nuestro trabajo con la heurística basada en GA, PSO, estrategias híbridas existentes basadas en GA y PSO y el algoritmo genético. Los resultados mostraron que el caso de prueba generación es eficientes en nuestro trabajo.
Aspectos Relevantes	
Problema/ Solución	Generación de datos de pruebas automáticas Heuristica basada en AG para la generación automática de conjuntos de pruebas
Aplicación de los AG en la IS	El algoritmo propuesto es implementado en Java y se integra las ideas de evolución de ambas técnicas de los soft-computing (es decir, PSO y GA),
Tecnología aplicada a los AG	Hemos probado la estrategia propuesta en diez clases de Java y los resultados han sido muy alentadores en cada una de estos resultados.
	Este documento propone la generación automatizada de datos de prueba de software para programas orientados a procedimientos y objetos. Se ha abordado el objetivo de minimizar el tiempo y maximizar la cobertura de ramas y la detección de fallos.

Tabla 34: Formulario Articulo S14

CB08 S15

Identificación	
Título	Prioritization of test scenarios using hybrid genetic algorithm based on UML activity diagram
Español	Priorización de Escenarios de prueba utilizando algoritmo genético híbrido Basado en el Diagrama de Actividad UML
Publicación	College of Computer Science and Technology Nanjing University of Aeronautics and Astronautics, Nanjing, China Corresponding
Autor	Wang Xinying, Jiang Xiajun, Shi Huibin
Referencia	[43]
Año	2015
Fecha	12/07/2015
Descripción	
Área	Ciencias de la Computación
Impacto	6
Resumen	Software de pruebas es una parte esencial de la SDLC (Desarrollo de Software Ciclo de Vida). Los escenarios de prueba se utilizan para derivar los casos de prueba basados en modelo. Sin embargo, con el software de rápido crecimiento en tamaño y complejidad, el costo del software será demasiado alto si queremos poner a prueba todos los casos de prueba. Por lo que este artículo se presenta un enfoque híbrido utilizando Algoritmo Genético (HGA) para dar prioridad a los escenarios de prueba, lo que mejora la eficiencia y reduce los costos también. El algoritmo combina algoritmo genético (GA) con la optimización de enjambre de partículas (PSO) y el algoritmo utiliza la estrategia de búsqueda local para actualizar la mejor información local y global del PSO. El algoritmo propuesto puede dar prioridad a los escenarios de prueba con el fin de encontrar un escenario crítico. Por último, se aplica el método propuesto a varios diagramas UML actividad típica, y se compara con el algoritmo genético simple (SGA). Los resultados experimentales muestran que el método propuesto no sólo da prioridad a los escenarios de prueba, sino que también mejora la eficiencia y mejoran aún más esfuerzo, tiempo y costo.
Aspectos Relevantes	
Problema/ Solución	Los escenarios de los casos de pruebas Priorización de Escenarios de prueba utilizando algoritmo genético híbrido
Aplicación de los AG en la IS	La priorización de los escenarios de prueba derivados del diagrama de actividades UML
Tecnología aplicada a los AG	La tecnología orientada a objetos de desarrollo de software basado en UML
Conclusión	Los resultados experimentales muestran que el método propuesto no sólo da prioridad a los escenarios de prueba, sino que también mejora la eficiencia y mejoran aún más esfuerzo, tiempo y costo que se utilizan durante esta etapa.
	Este enfoque puede identificar con eficacia el camino de prueba que debe ser probado por primera vez. Las ventajas del enfoque propuesto son las siguientes: en primer lugar, los escenarios de prueba se pueden generar temprano en el proceso de desarrollo. En segundo lugar, nos ayuda a descubrir muchos problemas en la fase de diseño, es decir, antes de la ejecución del programa. En tercer lugar, los escenarios de prueba se priorizan de tal manera que puede aumentar la probabilidad de detección temprana de fallos. Por último, también mejor mejora la eficiencia en comparación con la simple algoritmo genético

Tabla 35: Formulario Articulo S15

CB08 S16

Identificación	
Título	Efficient parallel evolutionary algorithms for deadline-constrained scheduling in project management
Español	Algoritmos evolutivos paralelos eficientes para la programación plazo con limitaciones en la gestión de proyectos
Publicación	Universidad de la República, Herrera y Reissig 565, Montevideo, 11300, Uruguay
Autor	Nesmachnow Sergio
Referencia	[44]
Año	2016
Fecha	03/18/2016
Descripción	
Área	Ciencias de la Computación
Impacto	5
Resumen	La fecha límite, programación limitada en la gestión de proyectos es un problema de optimización con mayor relevancia en ingeniería de software y otras situaciones de la vida real que se ocupan de la planificación de las actividades que debe ser completado antes de las fechas específicas. En este artículo se introduce versiones paralelas eficientes para dos algoritmos evolutivos (algoritmo genético y algoritmo evolutivo híbrido), para resolver el problema plazo de programación con limitaciones en la gestión de proyectos. Los algoritmos propuestos han sido diseñados para calcular soluciones precisas en los tiempos de reducidos de ejecución. Los algoritmos fueron implementados en Galib, una biblioteca bien conocido para Los Algoritmos Evolutivos desarrollados en C / C ++, que se amplió para proporcionar soporte para múltiples hilos de procesamiento y la computación distribuida. El análisis experimental realizado tanto en un conjunto de instancias de problemas estándar y los nuevos casos de problemas grandes demuestran que las soluciones exactas se calculan por las técnicas propuestas, sobre todo para la versión subpoblación distribuida del algoritmo evolutivo híbrido. La evaluación experimental comparativa demuestra que los algoritmos evolutivos paralelos son capaces de superar en los tiempos de ejecución reducidos los resultados calculados utilizando una de las mejores técnicas deterministas bien conocidas para el problema, en particular en la resolución de casos con limitaciones plazo ajustado
Aspectos Relevantes	
Problema/ Solución	Estimación de tiempos en la planificación de proyectos de desarrollo de software Algoritmos evolutivos paralelos para resolver el problema plazo de programación con limitaciones en la gestión de proyectos
Aplicación de los AG en la IS	En la gestión de proyectos
Tecnología aplicada a los AG	Los algoritmos propuestos se llevaron a cabo mediante la ampliación de Galib, una biblioteca para los Algoritmos Evolutivos desarrollados en C / C ++ utilizando el paradigma orientado a objetos
Conclusión	La evaluación experimental comparativa demuestra que los algoritmos evolutivos paralelos son capaces de superar en los tiempos de ejecución reducidos los resultados calculados utilizando una de las mejores técnicas deterministas bien conocidas para el problema, en particular en la resolución de casos con limitaciones plazo ajustado
	Las funciones y métodos dentro de cada EA fueron diseñados para calcular soluciones precisas en tiempo reducido y para proporcionar un buen patrón de exploración, utilizando operadores evolutivos ad-hoc.

Tabla 36: Formulario Articulo S16

CB09 S17

Identificación	
Título	Automatic generation of basis test paths using variable length genetic algorithm
Español	Generación automática de rutas de prueba básicos usando longitud variable algoritmo genético
Publicación	Department of Mathematics and Computer Science, Faculty of Science, Beni-Suef University, Beni-Suef, Egypt
Autor	Ghiduk Ahmed S.
Referencia	[45]
Año	2014
Fecha	26/01/2014
Descripción	
Área	Ciencias de la Computación
Impacto	2
Resumen	Las pruebas de ruta es el criterio de cobertura más fuerte en las pruebas de caja blanca. Encontrar rutas de destino es un desafío clave en la prueba de ruta. Los algoritmos genéticos se han utilizado con éxito en muchas actividades de pruebas de software, tales como la generación de datos de prueba, la selección de los casos de prueba y casos de prueba de priorización. En este trabajo, presentamos un nuevo algoritmo genético para la generación de trayectorias de prueba. En este algoritmo de la longitud del cromosoma varía de iteración a otro de acuerdo con el cambio en la longitud de la trayectoria. Basado en el algoritmo propuesto, se presenta una nueva técnica para la generación automática de un conjunto de rutas de prueba base que pueden ser utilizados como caminos de prueba en cualquier método de ensayo. La técnica propuesta utiliza un método para verificar la independencia de las rutas generadas para ser incluido en el conjunto de base de caminos. Además, esta técnica emplea un método para comprobar la viabilidad de los caminos generados. Introducimos nuevas definiciones de los conceptos clave de algoritmo genético como la representación del cromosoma, cruce, mutación, y la función de la aptitud para ser compatible con la generación de la trayectoria. Además, se presenta un estudio de caso para mostrar la eficiencia de nuestra técnica. Hemos llevado a cabo una serie de experimentos para evaluar la efectividad de la técnica de generación de la trayectoria propuesta. Los resultados mostraron que la técnica propuesta causa una reducción sustancial en el esfuerzo de generación de la trayectoria, y que el algoritmo propuesto GA es eficaz en la generación de la trayectoria de prueba.
Aspectos Relevantes	
Problema/ Solución	Pruebas de ruta de destino Algoritmo genético para la generación de trayectorias de prueba.
Aplicación de los AG en la IS	En las trayectorias de prueba base
Tecnología aplicada a los AG	La arquitectura de la herramienta consta de cuatro módulos: el módulo de análisis, el módulo de generación de la trayectoria, el módulo de verificación, viabilidad e independencia de módulo de comprobación. Su codificación fue desarrollada en C++
Conclusión	Los resultados mostraron que la técnica propuesta causa una reducción sustancial en el esfuerzo de generación de la trayectoria, y que el algoritmo propuesto GA es eficaz en la generación de la trayectoria de prueba.
	Este documento presenta una nueva longitud variable de algoritmo genético. En el algoritmo propuesto, la longitud de cada cromosoma varía de iteración a iteración de acuerdo con el cambio de la longitud de la trayectoria. Basado en el algoritmo planteado, el documento presenta una nueva técnica para la generación automática de un conjunto de rutas de prueba base.

Tabla 37: Formulario Articulo S17

CB09 S18

Identificación	
Título	Retrieving sequence diagrams using genetic algorithm
Español	Recuperación de Diagramas de Secuencia Usando Algoritmos Genéticos
Publicación	Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on
Autor	Salami HO, Ahmed Moataz
Referencia	[46]
Año	2014
Fecha	14/05/2014
Descripción	
Área	Ciencias de la Computación
Impacto	3
Resumen	Los beneficios de la reutilización de software se multiplican si se lleva a cabo en las primeras etapas de desarrollo de software. Los diagramas de secuencia se utilizan comúnmente para modelar la funcionalidad de los sistemas de software en las etapas iniciales (por ejemplo, durante los requisitos de análisis) del ciclo de vida de desarrollo de software. En este trabajo se utilizan algoritmo genético (GA) para la determinación de la similitud de las representaciones gráficas de los diagramas de secuencia, con el fin de ayudar a la recuperación de los diagramas de secuencia similares a partir de un repositorio. Los resultados experimentales muestran que la introducción de GA en el cálculo gráfico de similitud conduce a mejoras muy significativas en la calidad de recuperación en comparación con el método existente que utiliza un algoritmo gráfico.
Aspectos Relevantes	
Problema/ Solución	Reutilización de software Algoritmo genético (GA) para la determinación de la similitud de las representaciones gráficas de los diagramas de secuencia
Aplicación de los AG en la IS	La recuperación de los modelos Unified Modelling Language (UML)
Tecnología aplicada a los AG	En la implementación actual de Matlab, toma 22.06 y 292.99 segundos para buscar un depósito de 60 diagramas, respectivamente
Conclusión	Los resultados experimentales muestran que la introducción de GA en el cálculo gráfico de similitud conduce a mejoras muy significativas en la calidad de recuperación en comparación con el método existente que utiliza un algoritmo gráfico.
	En este trabajo, los diagramas de secuencia se transforman en gráficos dirigidos que se comparan utilizando una técnica de coincidencia de gráfico que hace uso de algoritmo genético (GA).

Tabla 38: Formulario Articulo S18

CB10 S19

Identificación	
Título	Methods for cost estimation in software project management
Español	Los métodos para la estimación de costos en la gestión de proyectos de software
Publicación	International Conference on Applied Sciences 2015, ICAS 2015; Military Economics Academy of WuhanWuhan; China; 3 June 2015 through 5 June 2015
Autor	Briciu C V, Filip I, Indries I I
Referencia	[47]
Año	2016
Fecha	01/02/2016
Descripción	
Área	Ciencias de la Computación
Impacto	1
Resumen	La velocidad en la que los procesos utilizados en el campo de desarrollo de software han cambiado hace muy difícil la tarea de la previsión de los costos globales de un proyecto de software. Para muchos investigadores, esta tarea se ha considerado inalcanzable, pero hay un grupo de científicos para los que esta tarea se puede resolver utilizando los métodos ya conocidos matemáticos (por ejemplo, regresión lineal múltiple) y las nuevas técnicas como la programación genética y redes neuronales. El artículo presenta una solución para la construcción de un modelo para los modelos de estimación de costes en la gestión de proyectos de software utilizando algoritmos genéticos a partir de los conjuntos de datos relacionados COCOMO 81 modelo.
Aspectos Relevantes	
Problema/ Solución	La mala estimación costos en la gestión de proyectos de software Estimación de costes en la gestión de proyectos de software utilizando algoritmos genéticos
Aplicación de los AG en la IS	Modelos de estimación de costes en la gestión de proyectos de software
Tecnología aplicada a los AG	Se utilizan los algoritmos genéticos a partir de los conjuntos de datos relacionados COCOMO 81 modelo
Conclusión	Se concluyó que los modelos de estimación deben adaptarse a las nuevas tecnologías y sistemas emergentes y dependen en gran medida por el método de desarrollo de software elegido.
	Este articulo trata de ver la perspectiva del modelo desarrollado vinculado con la realidad actual del desarrollo de software considerando como base el ciclo de vida del producto de software, los retos actuales y las innovaciones en el área de desarrollo de software.

Tabla 39: Formulario Articulo S19

CB11 S20

Identificación	
Título	Predicting project effort intelligently in early stages by applying genetic algorithms with neural networks
Español	Predecir el esfuerzo del proyecto de manera inteligente en etapas tempranas mediante la aplicación de algoritmos genéticos con Redes Neuronales
Publicación	Business School of Central South University, Changsha, 410083, China
Autor	Li Zhen You
Referencia	[48]
Año	**2014**
Fecha	**06/02/2014**
Descripción	
Área	Ciencias de la Computación
Impacto	3
Resumen	En las primeras etapas de un proyecto de desarrollo de software, la estimación de la cantidad de esfuerzo es una de las preocupaciones más importantes de gestión de proyectos. En este estudio se ha reproducido con éxito los modelos óptimos reducidos que predicen de forma inteligente la estimación de costos de software mediante el empleo de redes neuronales con algoritmos de aprendizaje de propagación hacia atrás, combinados con algoritmos genéticos (GA-NN) para determinar las variables explicativas más significativas entre los parámetros de costos COCOMO 16. El rendimiento del modelo completo de GA-NN es muy superior a la de la COCOMO, mientras que el rendimiento de la predicción de su modelo global óptimo reducida también es comparable a la de la COCOMO. Los modelos óptimos reducidos y sus factores significativos encontrados pueden ofrecer apoyos potentes para los administradores de proyectos para tomar decisiones correctas en las primeras etapas de los proyectos.
Aspectos Relevantes	
Problema/ Solución	Estimación costos en la gestión de proyectos de software Modelos de estimación proyectos de software mediante redes neuronales y algoritmos genéticos
Aplicación de los AG en la IS	En la gestión de proyectos
Tecnología aplicada a los AG	
Conclusión	En los experimentos GA-NN, se encontró que los subconjuntos óptimos locales de las variables con diferentes parámetros de diversas tasas de cruce y las tasas de mutación, y los nueve conductores de costes significativos entre el subconjunto óptimo local, entonces se identifica como conjuntos óptimos globales en varios estímulos, que se puede concluir que los modelos reducidos son fiables y la estimación de esfuerzo es precisa y robusta.
	La estimación del esfuerzo de desarrollo de software siempre ha sido una gran preocupación de los gestores de proyectos de software. La identificación de los factores más importantes que contribuyen al esfuerzo de desarrollo de software es especialmente importante para los administradores en la toma de decisiones en la etapa temprana del desarrollo de software.

Tabla 40: Formulario Articulo S20

6. RESULTADOS

Análisis de Resultados

En esta parte del proceso de la revisión empezamos con la clasificación de los resultados obtenidos (sección 4.1) los cuales se han logrado una vez identificadas las fuentes de información y definidos los estudios primarios, esta clasificación según las áreas en las que se dan estos trabajos (sección 4.2) y luego se realiza una comparación del tipo formal (sección 4.3) de los principales estudios de la aplicación de algoritmos genéticos en la Ingeniería del Software, como finalización de este proceso se definirán las conclusiones obtenidas de esta comparación (sección 4.4).

Estudios analizados

Para poder empezar esta etapa se debe mostrar un resumen de los estudios que se han analizado los cuales se muestran en la Tabla 41 en la cual se definen los que se han considerado como relevantes y los que se han tomado en cuenta como estudios primarios. Para llegar a esta selección de los diferentes estudios utilizamos el procedimiento antes descrito, que sigue un sistemático proceso el mismo se inicia con la ejecución de la consulta en la fuente de datos seleccionada previamente, de este paso se obtiene un conjunto de estudios a los que se les debe aplicar los criterios de inclusión para llegar a definir los estudios Relevantes y luego al aplicar los criterios de exclusión poder definir los estudios Primarios.

Una vez determinados los estudios Primarios se realiza una última fase de refinado la que nos sirve para definir de los estudios importantes aquellos que se pueden añadir como primarios debido a su contenido y la relación que prestan con los estudios ya seleccionados. Se puede observar que en la presente revisión se han tomado en cuenta para el análisis 127 de los cuales se consideraron como primarios 20.

Fuentes	Estudios	Relevantes	Primarios
ACM	616	27	8
IEEE	245	40	5
Scopus	450	60	7
TOTAL	1311	27	20

Tabla 41: Estudioas analizados

Presentación de resultados

La presentación de los resultados que se obtuvieron durante este proceso se presenta según la clasificación de estudios y la aportación de cada uno de estos, los estudios que se tomaron en cuenta principalmente son:

- Estudios realizados a proyectos en los que se aplicaron Algoritmos Genéticos en la Ingeniería del Software
- Análisis de los resultados obtenidos en la Aplicación de Algoritmos Genéticos
- Trabajos centrados en el desarrollo de le Ingeniería del Software
- Desarrollo de los Algoritmos Genéticos
- Evolución de los Algoritmos Genéticos y de la Ingeniería del Software

Descripción del Marco de Comparación Formal

Existen varios métodos para medir, comparar y evaluar los resultados, según la aplicación que se le va a dar, por lo que hemos elegido la comparación sistemática ya destajo, la cual nos permite realizar un sondeo en base a la importancia de cada uno de estos y tomando en cuenta las principales similitudes que muestran los diferentes estudios.

Aunque varios de estos estudios no fueron desarrollados para cumplir el mismo objetivo, ni sobre el mismo dominio específico (en general el de desarrollo), es interesante poder analizarlas de algún modo para poder definir que parte de los procesos de desarrollo pueden reutilizarse en futuras aplicaciones, y exponer cuales de éstas parecen más completas dentro del área de la ingeniería del software y los algoritmos genéticos. La comparación ha sido realizada sobre aquellas propuestas cuyos desarrollos y procesos se muestran bajo un sistema similar o han sido proporcionadas directamente por los autores, el resto estaban aún en proceso de desarrollo y por lo tanto, no disponibles para su comparación.

Nº	Titulo	Español	Public	Autor	Problema	Solución
S01	Minimizing test suites in software product lines using weight-based genetic algorithms	La minimización adecuada de prueba en Líneas de Producto Software Usando Algoritmos genéticos basados en el peso	2013	Wang Shuai, Ali Shaukat, Gotlieb Arnaud,	Eliminar los casos de prueba redundantes en las líneas de producción de software	Aplicación de algoritmos genéticos (gas) basados en el peso para minimizar el conjunto de pruebas para probar un producto.
S02	Cost-Priority Cognizant Regression testing	Costo-Prioridad Las pruebas de regresión Consciente	2014	Bhasin Harsh	Eliminar los casos de prueba redundantes en las líneas de producción de software	Aplicamos algoritmos genéticos (gas) basados en el peso para minimizar el conjunto de pruebas para probar un producto.
S03	UML Modeling of Load Optimization for Distributed Computer Systems based on Genetic Algorithm	UML Modelado de carga Optimización de Sistemas Distribuidos por ordenador basado en Algoritmos Genéticos	2013	Saxena Vipin, Arora Deepak, Mishra Nimesh	Utilización de recursos en la computación distribuida	Implementación de los algoritmos genéticos para optimizar el tiempo de espera de un conjunto de procesos
S04	Improved heuristics for solving OCL constraints using search algorithms	Heurística mejorados para la resolución de restricciones OCL utilizando algoritmos de búsqueda	2014	Ali Shaukat, Iqbal Muhammad Zohaib, Arcuri Andrea,	Limitaciones en la heurística de las restricciones OLC.	Mejorar la heurística existente para resolver restricciones OCL. utilizando algoritmos de búsqueda
S05	Critical components testing using hybrid genetic algorithm	Prueba críticas de componentes que utilizan Algoritmo Genético Híbrido	2013	Monção Ana C L, Camilo-Jr Celso G, Queiroz Leonardo T, Rodrigues Cassio L, Leitão Jr , Plínio de Sá, Vincenzi Auri M R,	Los componentes críticos de prueba	Optimización basada en Algoritmos Genéticos Híbridos
S06	Development of a framework for test case prioritization using genetic algorithm	Desarrollo de un marco para la priorización de casos de prueba Utilizando Algoritmos Genéticos	2013	Malhotra Ruchika, Tiwari Divya	Prueba en la fase de mantenimiento	Priorización de casos de prueba utilizando algoritmos genéticos

S07	Random or Genetic Algorithm Search for Object-Oriented Test Suite Generation	Aleatorio o Algoritmo Genético Buscar Generación conjunto de pruebas orientada a objetos	2015	Shamshiri Sina, Rojas José Miguel, Fraser Gordon, Mcminn Phil, Court Regent	Prueba de cobertura en la programación orientada a objetos	Algoritmo genético en las pruebas de cobertura
S08	Minimizing feature model inconsistencies in software product lines	Reducir al mínimo las incoherencias de operaciones del modelo de Líneas de Producto Software	2014	Afzal Uzma, Mahmood Tariq, Rauf Imran, Shaikh Zubair Ahmed	Inconsistencias en el modelo de las líneas de producto de software	Algoritmos genéticos (GA) para minimizar tres inconsistencias modelo de características primarias
S09	A Dynamic Approach for Retrieval of Software Components Using Genetic Algorithm	Un enfoque dinámico para la recuperación de los componentes de software por medio de algoritmos genéticos	2015	Afzal Uzma, Mahmood Tariq, Rauf Imran, Shaikh Zubair Ahmed	Reutilización de Software	Recuperación mediante el uso de algoritmos genéticos
S10	Component-Based Software System Test Case Prioritization with Genetic Algorithm Decoding Technique Using Java Platform	Software basado en componentes de casos de prueba con el sistema de priorización con algoritmo genética de decodificación técnica que utiliza la plataforma Java	2015	Afzal Uzma, Mahmood Tariq, Rauf Imran, Shaikh Zubair Ahmed	Pruebas de priorización	Sistemas de priorización con algoritmos genéticos
S11	Optimization of soft cost estimation using genetic algorithm for NASA software projects	Optimización de Estimación de Costos suave usando Algoritmos Genéticos para la NASA Proyectos de Software	2015	Algabri Mohammed, Saeed Fahman, Mathkour Hassan, Tagoug Nejmeddine	Estimación de costos con modelo COCOMO	Ajuste de los parámetros de los coeficientes de COCOMO usando algoritmos genéticos
S12	Software quality assurance for object-oriented systems using meta-heuristic search techniques	Aseguramiento de la calidad del software para los sistemas orientados a objetos utilizando técnicas de búsqueda de meta-heurísticos	2015	Algabri Mohammed, Saeed Fahman, Mathkour Hassan, Tagoug Nejmeddine	Calidad de software para sistemas orientada a objetos	Utilización de técnicas de búsquedas para la optimización de software de predicción de fallos
S13	A novel approach for test case generation from UML activity diagram	Un nuevo enfoque para la generación de casos de prueba de UML Diagrama de actividad	2014	Jena Ajay Kumar, Swain Santosh Kumar,	Pruebas basadas en modelos	Aplicación de algoritmos genéticos para la generación de los casos de pruebas

				Mohapatra Durga Prasad		
S14	A novel strategy for automatic test data generation using soft computing technique	Una nueva estrategia para la generación de datos de prueba automático utilizando la técnica de soft computing	2014	Chawla P, Chana I, Rana A	Generación de datos de pruebas automáticas	GA heurísticos para la generación automática de conjuntos de pruebas
S15	Prioritization of test scenarios using hybrid genetic algorithm based on UML activity diagram	Priorización de Escenarios de prueba utilizando algoritmo genético híbrido Basado en el Diagrama de Actividad UML	2015	Wang Xinying, Jiang Xiajun, Shi Huibin	Los escenarios de los casos de pruebas	Priorización de Escenarios de prueba utilizando algoritmo genético híbrido
S16	Efficient parallel evolutionary algorithms for deadline-constrained scheduling in project management	algoritmos evolutivos paralelos eficientes para la programación plazo con limitaciones en la gestión de proyectos	2016	Nesmachnow Sergio	Estimación de tiempos en la planificación de proyectos de desarrollo de software	Algoritmos evolutivos paralelos para resolver el problema plazo de programación con limitaciones en la gestión de proyectos
S17	Automatic generation of basis test paths using variable length genetic algorithm	Generación automática de rutas de prueba básicos usando longitud variable algoritmo genético	2014	Ghiduk Ahmed S.	Pruebas de ruta de destino	Algoritmo genético para la generación de trayectorias de prueba.
S18	Retrieving sequence diagrams using genetic algorithm	Recuperando Diagramas de Secuencia Uso Genético Algoritmo	2014	Rodriguez Leidy Garzon, Diosa Henry Alberto, Rojas-Galeano Sergio	Reutilización de software	Algoritmo genético (GA) para la determinación de las representaciones gráficas de los diagramas de secuencia
S19	Methods for cost estimation in software project management	Los métodos para la estimación de costes en la gestión de proyectos de software	2016	Briciu C V, Filip I, Indries I I	La mala estimacion costos en la gestión de proyectos de software	Estimación de costes en la gestión de proyectos de software utilizando algoritmos genéticos
S20	Predicting project effort intelligently in early stages by applying genetic algorithms with neural networks	La predicción Esfuerzo Proyecto inteligente en estadios tempranos mediante la aplicación de algoritmos genéticos con Redes Neuronales	2014	Li Zhen You	Estimación costos en la gestión de proyectos de software	Modelos de estimación proyectos de software mediante redes neuronales y algoritmos genéticos

Estadísticas por Año

Figura 21: Estadísticas por Año

En el Figura 21 se muestra los años en los que se han desarrollado los estudios analizados, lo que nos permite ver la evolución de desarrollo ya que en el 2013 no se muestra mayor cantidad de estudios realizados sobre la aplicación de algoritmos, 2014 y 2015 es el año en el cual existen más artículos relevantes para nuestra investigación lo que nos indica que los investigadores han dado más prioridad a la investigación para dar solución a los problemas de optimización en el desarrollo de software y así tener productos de calidad y confiables en un menor tiempo. En el presente año 2016 se muestra un porcentaje de estudios muy bajo ya que este aún lo estamos cursando, pero las aplicaciones en las que se realizan los estudios son de mayor profundidad ya que muchos de ellos ya toman como base estudios de años anteriores y por tanto no se repiten o se centran en estudios aún no realizados.

Estadísticas por Impacto

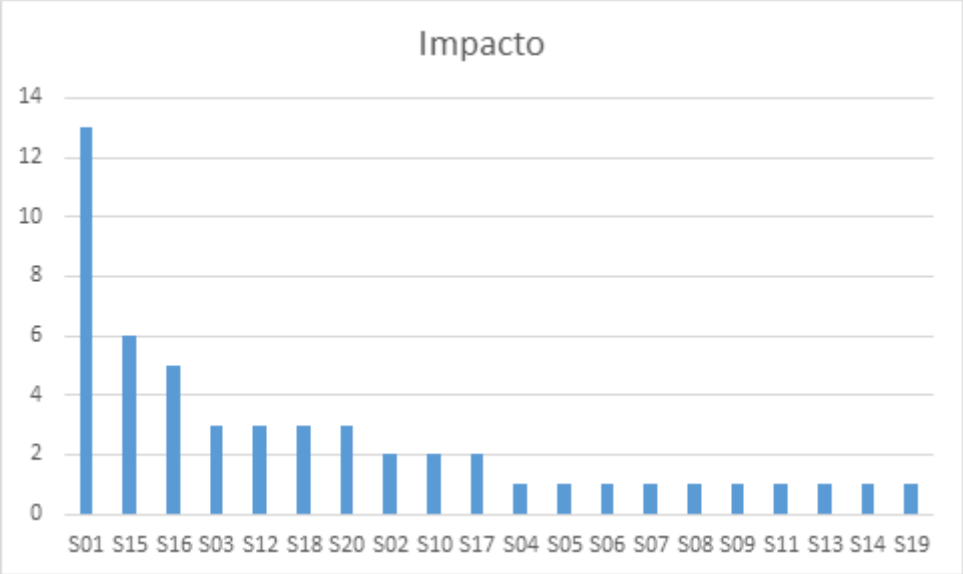


Figura 22: Estadísticas por Impacto

En el Gráfico ref{impacto} del Impacto, se muestra la incidencia que han tenido los estudios que hemos analizado en otros estudios realizados posteriormente, en esta tabla se visualiza que solamente uno de ellos S01 cite{Wang2013} ha mantenido una influencia muy destacada ya que se ha tomado como referencia directa para otros 13 estudios, los S15 y S16 muestran que el impacto que se ha obtenido ha sido mediano ya que se han tomado en cuenta más de 4 veces y el resto de los estudios el impacto que han mostrado ha sido de nivel bajo ya que solo se los ha considerado en menos de 4 estudios y en tres casos S08, S09 y S13 la incidencia ha sido nula ya que no se los ha considerado para otros estudios.

Estadistica de la Aplicaciones de los AG

APLICACIÓN DE LOS AG EN LA IS					
Nº	Pruebas	Gestión de proyectos	SPL	CBSE	Sistemas Distribuidos
S01			X		
S02	X				
S03					X
S04	X				
S05	X				
S06	X				
S07	X				
S08			X		
S09				X	
S10	X				
S11		X			
S12	X				
S13	X				
S14	X				
S15	X				
S16		X			
S17	X				
S18	X				
S19		X			
S20		X			

Tabla 42: Aplicación de los AG en la IS

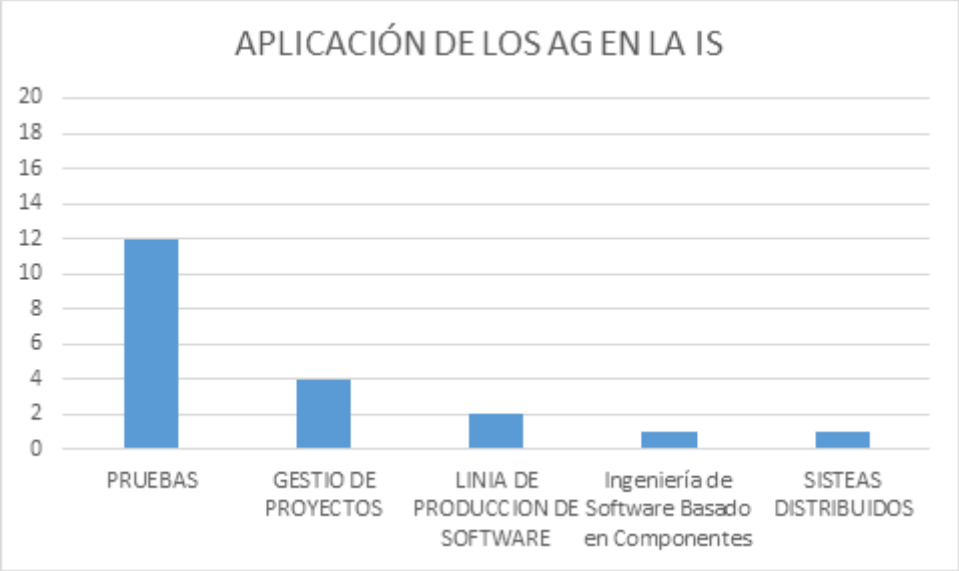


Figura 23: Estadísticas de la Aplicación de los AG

En el Figura 23 de las Aplicaciones de los Algoritmos Genéticos se muestra de manera evidente que la principal aplicación es en la fase de pruebas del proyecto, ya que en la mayoría de los estudios se los ha realizado en las diferentes fases de prueba las mismas que pueden ser regresivas, iniciales o terminales. Además se muestra que se pueden aplicar estos algoritmos en la parte de Gestión de proyectos lo que ayuda la optimización a nivel general del proyecto por lo que el análisis de este se lo ha realizado en 4 de los estudios , en cuanto a la Línea de Producción, Sistemas Distribuidos y el Ingeniería del Software basado en componentes en los estudios analizados solamente se ha probado la aplicación de estos algoritmos en un solo de los proyectos cada uno de estos, por lo que se puede concluir que no se los considera puntos de estudios relevantes para la aplicación de los algoritmos genéticos.

Estadísticas de la tecnología aplicada a los AG

TECNOLOGÍA APLICADA A LOS AG					
Nº	**OLC**	**Java**	**Matlab**	**C++**	**web**
S01		X			
S02				X	
S03				X	
S04	X				
S05		X			
S06		X			
S07		X			
S08					
S09					X
S10		X			
S11					
S12		X			
S13					
S14		X			
S15		X			
S16				X	
S17				X	
S18			X		
S19				X	
S20					

Tabla 43: Tecnología aplicada a los AG

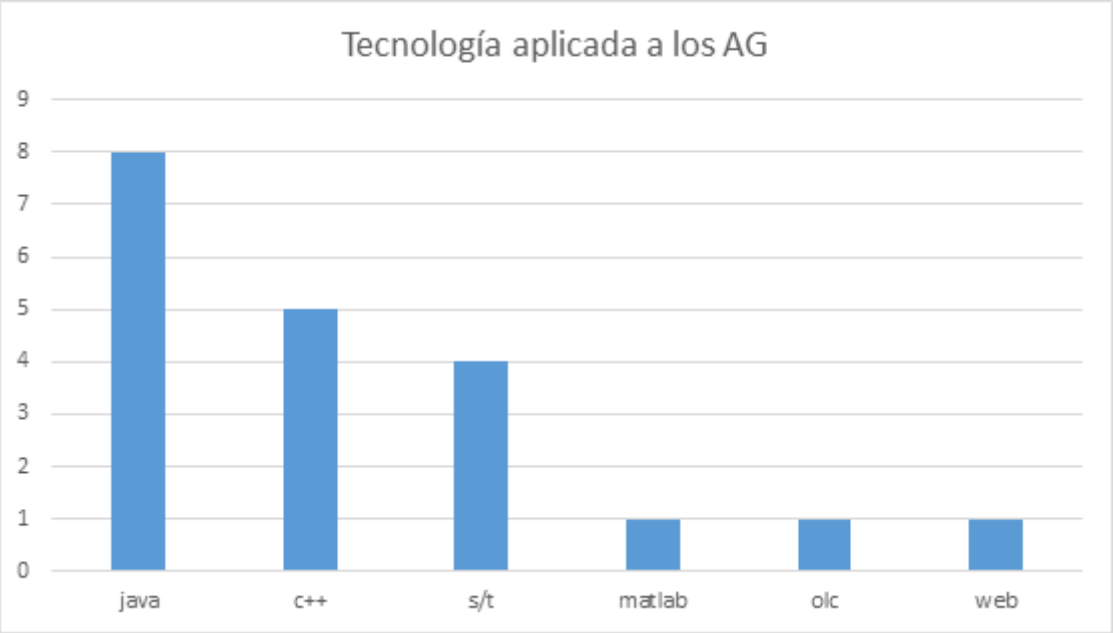


Figura 24: Estadística de Tecnologia

En la Figura 24 se muestra la tecnología aplicada a los algoritmos genéticos en el cual podemos notar que para el desarrollo se aplica mayormente Java, no solo por ser un lenguaje de programación libre sino por ser multiplataforma y que en la actualidad es el de mayor auge en el desarrollo de software en general. Aunque ya no se utiliza con frecuencia C++ los estudios que se analizaron muestran que varios de estos utilizaron este lenguaje para la programación de estas aplicaciones. Un gran porcentaje de estos estudios no han implementado aún alguna tecnología de desarrollo, ya que solo fueron aplicadas en alguna etapa de diseño o inicial. Las tecnologías Matlab, OLC y Web han sido aplicadas únicamente en dos de los estudios analizados, por lo que se puede definir que estos no son muy relevantes o que tienen poco auge dentro del desarrollo de aplicaciones en la actualidad, ya que en su mayoría las tecnologías de desarrollo que se utilizan son orientadas a objetos y de lineamientos libres.

7. Discusión

7.1. Discusión de la Revisión Sistemática

S01 y S08 coinciden en que se puede minimizar adecuadamente las pruebas en Líneas de Producto Software usando Algoritmos genéticos ya que se reduce el número total de casos de prueba, mejorando así la eficiencia de la prueba.

S02, S14 y S17 plantean a los Algoritmos Genéticos como solución para la generación automática de pruebas, ya que el proceso de desarrollo de software invierte al menos el 50 % del coste total en el proceso de pruebas de software.

S03, S04, S13, S15 y S18 coinciden que los beneficios de la reutilización de software se multiplican si se lleva a cabo en las primeras etapas de desarrollo de software. Los diagramas de secuencia se utilizan comúnmente para modelar la funcionalidad de los sistemas de software en las etapas iniciales del ciclo de vida de desarrollo de software.

En S05, S06 y S10 destacan que los casos de pruebas de priorización es una tarea esencial que reduce el esfuerzo de prueba en la fase de mantenimiento en un grado considerable. Estos artículos plantean un marco para la priorización de casos de prueba utilizando una herramienta basada en un algoritmo genético, desarrollado en lenguaje Java.

S07 y S12 coinciden que la identificación de fallas en la fase muy temprana de desarrollo de software de ciclo de vida es muy necesario. Esto ayuda a los desarrolladores de software a concentrarse más en la garantía de calidad, utilizar la mano de obra en la perspectiva adecuada y reducir sobre todo el coste de eliminación de fallos en el desarrollando software.

S11, S16, S19 y S20 coinciden que la fecha límite, programación limitada en la gestión de proyectos es un problema de optimización con mayor relevancia en ingeniería de software y otras situaciones de la vida real que se ocupan de la planificación de las actividades que debe ser completado antes de las fechas especiadas, para resolver el problema plazo de programación con limitaciones en la gestión de proyectos.

7.2. Desarrollo de la propuesta

Al realizar un análisis sobre el avance de la Ingeniería del Software y los cambios que se han dado en el desarrollo del mismo se puede notar que un cambio significativo le ha dado la aplicación de Algoritmos Genéticos por lo que para la realización de este estudio se planteó el tema "Aplicación de Algoritmos Genéticos en la Ingeniería del Software: Revisión Sistemática del Estado del Arte", para esto se han considerando algunos aspectos, en primer lugar la poca información que existe a nivel local sobre las ventajas que ofrece la aplicación de estos algoritmos, y dentro de que fases de desarrollo de software, no se los aplica para un mejor optimización de procesos. Para el exitoso desarrollo de este se cuenta con una gama de métodos aplicables al área de investigación, para lo cual se decidio aplicar o utilizar una revisión sistemática considerando que es una de las pocas que se direcciona a la ingeniería y además cuenta con un protocolo claramente establecido y estandarizado que asegura la claridad y transparencia en el proceso de la revisión. Con el desarrollo de esta revisión debemos cumplir los siguientes objetivos:

7.2.1. Ejecutar un proceso de planificación de revisión con el fin de determinar el objeto de indagación y la guía a seguir.

Para ejecutar este objetivo primeramente se identifican la necesidad de realizar la revisión sistemática que es la aplicación de los Algoritmos Genéticos en la Ingeniería de Software.

A continuación se realizó el paso más importante de una revisión que es la especificación de las preguntas de investigación, las mismas que nos guiaron en la recolección de la información, el análisis de esta y por ende a llegar a establecer las conclusiones en base a los objetivos propuestos al iniciar este estudio.Para culminar se propuso el protocolo de revisión a seguir que en este caso es el de Kitchenham[8].

7.2.2. Desarrollar la revisión en base a la planificación para determinar eventos más relevantes en la investigación.

Es esencial que se cumpla la planificación del desarrollo de la revisión ya que esto permitirá establecer de manera adecuada los pasos más relevantes a seguir por lo que la accesibilidad a la web así como la inclusión de motores de búsqueda que permitan realizar consultas avanzadas, luego con las palabras claves ya definidas se realizó concatenaciones utilizando los operadores lógicos AND y OR para formar las cadenas de

búsqueda que nos sirvieron para recopilar 127 estudios primarios, los mismos que luego de pasar por los criterios de selección se redujeron a 20 estudios relevantes los cuales se analizó y se sintetizo principales tomando en consideración los aspectos legales, aspectos contractuales y conclusiones principales.

7.2.3. Analizar los resultados, y elaborar el estado del arte de las aplicaciones de los algoritmos genéticos en la ingeniería de software.

Luego de haber cumplido rigurosamente todo el proceso antes descrito para el análisis de los estudios seleccionados se procede al análisis de los resultados obtenidos la misma que se realiza a través de una discusión entre los aspectos más relevantes y destacados para lo cual se ha tomado en cuenta 4 aspectos relevantes que son: que problemas solucionan los algoritmos genéticos, su aplicación en la Ingeniería de software y que tecnología utilizan para su ejecución, además se tomaron en cuenta las conclusiones relevantes.

En el transcurso de la revisión se pudo determinar que no existe información con respecto a la aplicación de algoritmos genéticos en la Ingeniería del Software en la base de datos científica del Ecuador (RRAAE) lo que dificultó en parte el proceso de la investigación, pero pese a esto se logró concluir con éxito rescatando valiosos resultados que nos ayudaron a emitir las respectivas conclusiones y a finalizar el presente Trabajo de Titulación.

7.3. Valoración Social, Técnica, Económica y Científica.

La valoración del presente Trabajo de Titulación se expresa describiendo los beneficios presentados en cuatro aspectos fundamentales:

7.3.1. Valoración Social

Conocer los diferentes problemas y la profundidad de los mismos que se solucionan con la aplicación de los algoritmos genéticos en la ingeniería de software. Conocer en qué fase o etapa de la Ingeniería de software se aplica con mayor frecuencia los Algoritmos genéticos y en cuál de ellas es más aconsejable aplicarlo para obtener mejores resultados. Conocer cuáles son las tecnologías que se ocupa para su implementación dentro del desarrollo de un proyecto en general en la ingeniería de software.

7.3.2. Valoración Técnica

- A través del gestor bibliográfico Mendeley se ahorró tiempo ya que este permite organizar las referencias de manera sencilla desde las fuentes y de varios modos.
- Con la utilización del editor de texto Sharelatex se facilitó la revisión del Trabajo de Titulación ya está diseñado específicamente para permitir la fácil y rápida colaboración de varios usuarios a la vez en un mismo proyecto en tiempo real.
- El uso del correo electrónico y las redes sociales permitió la constante comunicación entre el investigador y el director del Trabajo de Titulación.

7.3.3. Valoración Económica

- Uno de los principales beneficios es el aporte de la UNL con el control y seguimiento del Trabajo, ya que cubre los gastos del Tutor o Director de Tesis.
- El uso de herramientas tecnológicas colaboró al ahorro de tiempo y dinero pues se evitó realizar impresiones innecesarias así como asistencias personales a la UNL

7.3.4. Valoración Científica

El beneficio en el aspecto científico radica en el aporte que presta a la realización de trabajos futuros ya que esta revisión contiene una variedad de literatura que es relevante en este tema y una variada bibliografía lo que permitirá agilizar la búsqueda de documentos que aportan conocimientos sobre el mismo.

8. CONCLUSIONES

Los resultados mostrados están basados en el análisis de 20 documentos, los mismos que se obtuvieron de un total de 127, luego de pasar por un protocolo e revisión y selección, ya que son los que se encuentran más acorde con el tema de investigación.

Una vez analizados cada uno de los estudios seleccionados podemos concluir que ninguno de estos contienen información completa sobre la utilización de algoritmos genéticos en cada uno de los procesos que se siguen en la ingeniería del software, así mismo determinamos que los beneficios, las aplicaciones y los problemas que solucionan son varios y en diferentes niveles.

¿Qué problema soluciona en la ingeniera del software los Algoritmos Genéticos?

Al realizar el análisis de los estudios en los que se ha implementado Algoritmos genéticos se ha podido comprobar que el principal problema que soluciona es el desperdicio de tiempo al realizar la categorización de pruebas, ya que al optimizar este proceso se agilitan varias etapas durante el diseño, programación e implementación. Por todo lo antes expuesto se puede resumir que la Aplicación de Algoritmos Genéticos en la Ingeniería del software reduce la pérdida de tiempo en la categorización y aplicación de pruebas en las diferentes etapas del desarrollo de un software.

¿Qué aplicación tienen los Algoritmos Genéticos en la Ingeniería del Software?

En el presente estudio mostramos varias aplicaciones que tienen los Algoritmos Genéticos pero la más relevante es en la optimización de la relación tiempo-coste, ya que nos permite agilizar los procesos que se desarrollan en varias de estas etapas, por lo que no solo se ahorra en el plano de tiempo sino en el de recursos humanos lo que optimiza la utilización de todos los recursos asignados para el desarrollo completo de la aplicación.

¿Qué tecnología utilizan los algoritmos genéticos para su ejecución?

Para el desarrollo de los algoritmos genéticos en la mayoría de los estudios se aplica plataforma Java, no solo por ser un lenguaje de programación libre sino por ser multiplataforma y que en la actualidad es el de mayor auge en el desarrollo de software en general, aunque ya no se utiliza con frecuencia C++ los estudios que se analizaron

muestran que varios de estos utilizaron este lenguaje para la programación de estas aplicaciones.

Al analizar los estudios primarios podemos concluir que la optimización se da principalmente en lo que es el análisis de resultados en la aplicación de pruebas, lo que es posible ya que estos algoritmos nos ayudan fundamentalmente a realizar la realimentación y categorización de las pruebas a aplicarse en las diferentes etapas de desarrollo.

Cabe recalcar que una gran parte de los estudios seleccionados se encuentran aún en análisis o desarrollo por lo que los resultados que de ahí se extrajeron son a nivel de UNL o diseño.

Por lo que se puede concluir una vez finalizado este análisis que existen estudios primarios que son específicos y que ofrecen una respuesta clara sobre los problemas que ha resuelto la aplicación de algoritmos genéticos. Asi mismo se puede definir que su aplicación permite la optimización de varias etapas que se definen en la ingeniería del software, además que las tecnologías en las que se pueden ejecutar estos algoritmos no son limitadas o no muestran restricciones. Además se puede definir que los alcances que tienen los algoritmos en su mayoría se muestran más definidamente en la etapa de diseño y de implementación, ya que permite agilitar las pruebas y su categorización, lo que mejora la relación entre el tiempo y coste, permitiendo así tener una mejor utilización de los recursos empleados en el desarrollo de un proyecto.

9. Recomendaciones

- La carrera de Ingeniería en Sistemas de la universidad Nacional de Loja debería incluir en su pensum de estudio el desarrollo y la aplicación de Algoritmos Genéticos, permitiendo que los estudiantes sepan la manera adecuada de aplicarlos y que podan así obtener los resultados deseados.

- Como línea de investigación dentro de los trabajos de titulación se recomienda tomar en cuenta el análisis del uso de los algoritmos genéticos en las diferentes etapas de la Ingeniería del software.

Se recomienda la aplicación de Algoritmos Genéticos en el desarrollo de aplicaciones y desarrollo de software en general ya que se ha mostrado firmemente que permite la optimización principalmente de la fase de prueba que es la que consume la mayor parte de los recursos.

Trabajos Futuros

Al finalizar este estudio podemos definir que como trabajo futuro se debería considerar la inclusión de nuevos análisis desarrollados o de algunos que fueron excluidos en la selección inicial por encontrarse incompletos o en fase de desarrollo, lo que nos permitirá obtener nueva información que complete o que mejore los resultados obtenidos mediante este estudio, ya que falta muchos niveles de análisis concretamente en cada una de las etapas de desarrollo, lo que permitirá la observación de resultados que se muestren en cada una de las mismas.

10. BIBLIOGRAFÍA

[1] M. Alzabidi and A. Kumar, "Automatic Software Structural Testing by Using Evolutionary Algorithms for Test Data Generations," *J. Comput. Sci.*, vol. 9, no. 4, pp. 390–395, 2009.

[2] B. Shuai, M. Li, H. Li, Q. Zhang, and C. Tang, "Software vulnerability detection using genetic algorithm and dynamic taint analysis," *2013 3rd Int. Conf. Consum. Electron. Commun. Networks, CECNet 2013 - Proc.*, pp. 589–593, 2013.

[3] C. Doungsa-Ard, K. Dahal, and M. Hossain, "An automatic test data generation from UML state diagram using genetic algorithm.," *Test*, vol. 4, no. 91712, pp. 1–5, 2007.

[4] C. Sharma, S. Sabharwal, and R. Sibal, "A Survey on Software Testing Techniques using Genetic Algorithm," *Int. J. Comput. Sci. Issues*, vol. 10, no. 1, pp. 381–393, 2013.

[5] I. Sommerville, *Software Engineering*. 2010.

[6] F. Baccichetti, F. Bordin, and F. Carlassare, "lambda-Prophage induction by furocoumarin photosensitization.," *Experientia*, vol. 35, no. 2, pp. 183–184, 1979.

[7] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele Univ.*, vol. 33, no. TR/SE-0401, p. 28, 2004.

[8] B. Kitchenham, R. Pretorius, D. Budgen, O. P. Brereton, M. Turner, M. Niazi, and S. Linkman, "Systematic literature reviews in software engineering-A tertiary study," *Inf. Softw. Technol.*, vol. 52, no. 8, pp. 792–805, 2010.

[9] R. S. Pressman and J. M. Troya, "Ingeniería del software," no. 001.64 P74s., 1988.

[10] R. S. Pressman, "Ingeniería de Software," 2010.

[11] R. S. Pressman, *Ingeniería del software. Un enfoque práctico*. 2002.

[12] A. Casali, "¿Qué es la Inteligencia Artificial?"

[13] M. Gestal, D. Rivero, J. R. J. Rabuñal, J. Dorado, and A. Pazos, *Introducción a los algoritmos genéticos y la programación genética*. 2010.

[14] S. Norvig, P., & Russell, “Inteligencia artificial,” *Elsevier Bras.*, vol. 1, no. 3, p. 1021, 2014.

[15] A. Guerra-Hernández, “Inteligencia Artificial II: conocimiento, razonamiento y planeación,” p. 26, 2013.

[16] P. Ponce, *Inteligencia artificial con aplicaciones a la ingenieria*. 2010.

[17] S. Russell and P. Norvig, *Inteligencia Artificial. Un enfoque moderno. 2da Edición*. 2004.

[18] M. A. Moreno Martin, “La Ingeniería del Software,” pp. 4–32, 2010.

[19] N. Gil, “Algoritmos genéticos,” *Investig. Cienc.*, 1992.

[20] M. Correia, *Algoritmos genéticos*. 2003.

[21] A. M. Andaluz, “Algoritmos Evolutivos y Algoritmos Genéticos,” *Intel. en Redes Comun.*

[22] R. Malhotra and D. Tiwari, “Development of a framework for test case prioritization using genetic algorithm,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 3, p. 1, 2013.

[23] L. Am and L. Am, “Capítulo 3. panorama actual en américa latina,” pp. 19–24, 1999.

[24] F. Jos, C. Gonz, and S. B. Mart, *Ingeniería del Software*. 2008.

[25] H. Arturo and F. Fernández, “Inteligencia Arti fi cial mediante Ingeniería de Software Arti fi cial Intelligence driven Software Engineering,” pp. 49–60, 2008.

[26] “Los avances de la Inteligencia Artificial | Centro de Innovación BBVA.” [Online]. Available: http://www.centrodeinnovacionbbva.com/noticias/los-avances-de-la-inteligencia-artificial.

[27] C. Meli and Z. K. Oplatkova, “Software Engineering in Intelligent Systems,” vol. 349, pp. 399–411, 2015.

[28] I. Software, “Ingeniería del Software,” 2008.

[29] O. Funlam, “La Ingeniería De Sistemas Y Su Evolución Hacia La Arquitectura De Sistemas,” no. 2, pp. 96–105, 2009.

[30] S. Wang, S. Ali, and A. Gotlieb, “Minimizing test suites in software product lines using weight-based genetic algorithms,” *Proceeding fifteenth Annu. Conf. Genet. Evol. Comput. Conf.*, pp. 1493–1500, 2013.

[31] H. Bhasin, “Cost-Priority Cognizant Regression testing,” vol. 39, no. 3, pp. 1–7, 2014.

[32] V. Saxena, D. Arora, and N. Mishra, “UML Modeling of Load Optimization for Distributed Computer Systems Based on Genetic Algorithm,” *SIGSOFT Softw. Eng. Notes*, vol. 38, no. 1, pp. 1–7, 2013.

[33] S. Ali, M. Z. Iqbal, and A. Arcuri, “Improved heuristics for solving OCL constraints using search algorithms,” *16th Genet. Evol. Comput. Conf. GECCO 2014*, pp. 1231–1238, 2014.

[34] D. Jeya Mala, K. Sabari Nathan, and S. Balamurugan, “Critical components testing using hybrid genetic algorithm,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 5, p. 1, 2013.

[35] S. Shamshiri, J. M. Rojas, G. Fraser, P. Mcminn, and R. Court, “Random or Genetic Algorithm Search for Object-Oriented Test Suite Generation ?,” *Proc. 2015 Annu. Conf. Genet. Evol. Comput.*, pp. 1367–1374, 2015.

[36] U. Afzal, T. Mahmood, I. Rauf, and Z. A. Shaikh, “Minimizing feature model inconsistencies in software product lines,” *17th IEEE Int. Multi Top. Conf. Collab. Sustain. Dev. Technol. IEEE INMIC 2014 - Proc.*, pp. 137–142, 2015.

[37] S. Vodithala and S. Pabboju, “A dynamic approach for retrieval of software components using genetic algorithm,” 2015, pp. 406–410.

[38] S. Mahajan, S. D. Joshi, and V. Khanaa, “Component-Based Software System Test Case Prioritization with Genetic Algorithm Decoding Technique Using Java Platform,” *2015 Int. Conf. Comput. Commun. Control Autom.*, pp. 847–851, 2015.

[39] M. Algabri, F. Saeed, H. Mathkour, and N. Tagoug, “Optimization of soft cost

estimation using genetic algorithm for NASA software projects," *2015 5th Natl. Symp. Inf. Technol. Towar. New Smart World*, pp. 1–4, 2015.

[40] Y. Suresh, "Software quality assurance for object-oriented systems using meta-heuristic search techniques," pp. 441–448, 2015.

[41] A. K. Jena, S. K. Swain, and D. P. Mohapatra, "A novel approach for test case generation from UML activity diagram," *2014 Int. Conf. Issues Challenges Intell. Comput. Tech.*, pp. 621–629, 2014.

[42] P. Chawla, I. Chana, and A. Rana, "A novel strategy for automatic test data generation using soft computing technique," *Front. Comput. Sci.*, vol. 9, no. 3, pp. 346–363, 2015.

[43] X. Wang, X. Jiang, and H. Shi, "Prioritization of test scenarios using hybrid genetic algorithm based on UML activity diagram," 2015, pp. 854–857.

[44] S. Nesmachnow, "Efficient parallel evolutionary algorithms for deadline-constrained scheduling in project management," vol. 7, no. 1, pp. 34–49, 2016.

[45] A. S. Ghiduk, "Automatic generation of basis test paths using variable length genetic algorithm," vol. 114, no. 6, pp. 304–316, Jun. 2014.

[46] H. Salami and M. Ahmed, "Retrieving sequence diagrams using genetic algorithm," *Comput. Sci. Softw. …*, pp. 324–330, 2014.

[47] C. V Briciu, I. Filip, and I. I. Indries, "Methods for cost estimation in software project management," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 106, no. 1, p. 12008, 2016.

[48] Z. Y. Li, "Predicting project effort intelligently in early stages by applying genetic algorithms with neural networks," vol. 513–517, pp. 2035–2040, 2014.